

L.EEC025 - Fundamentals of Signal Processing (FunSP)

2023/2024 – 1st semester

Week12, 04 Dec 2023

Objectives:

-use the FFT in a number of simulation and real-time experiments so as to:

- understand and recognize the leakage phenomenon
- understand and recognize the aliasing phenomenon
- understand and recognize the effect of windowing in moderating the leakage phenomenon
- observe and understand real-time spectrum analysis using both the STM32F7 platform and the oscilloscope
- perform basic spectrum analysis using either a sinusoidal input or a more structured periodic waveform such as the sawtooth wave

DSP Education Kit

LAB 10

Spectrum analysis using the Fast Fourier Transform

Issue 1.0

Contents

1	Introduction.....	1
1.1	Lab overview	1
2	Requirements	1
3	The basic spectrum analysis setup	1
3.1	Windowing.....	3
3.2	The leakage phenomenon	4
3.3	Configuring the oscilloscope as a spectrum analyzer	6
4	Programming the STM32F7 as a real-time spectrum analyzer	7
4.1	The FFT of a signal in real-time	10
4.2	Modifying the program to reduce spectral leakage	10
4.3	Compiling and running the C code.....	11
5	Performing various spectrum analysis experiments	11
5.1	Setting-up for this lab experiment.....	11
5.2	Observing leakage due to the rectangular window.....	12
5.3	Observing leakage due to the Hamming window.....	14
6	Conclusions.....	15
7	Additional References.....	15

1 Introduction

1.1 Lab overview

This Lab motivates experimentation with the DFT that is implemented by means of an FFT in order to understand its practical usefulness in spectrum analysis. In particular, an emphasis is given to the understanding of the spectral leakage phenomenon, and the tradeoff between near-end and far-end leakage when the Rectangular window is used, or when the Hamming or Hanning window is used, in a real-time spectrum analysis setting.

2 Requirements

To carry out this lab, you will need:

- An STM32F746G Discovery board
- A PC running Keil MDK-Arm
- MATLAB
- An oscilloscope *equipped with the FFT Math functionality*
- Suitable connecting cables
- An audio frequency signal generator

3 The basic spectrum analysis setup

Our basic spectrum analysis setup implements the signal processing that is introduced in this week's P2P Exercise 2 (week12) and whose block diagram is illustrated in Figure 1.

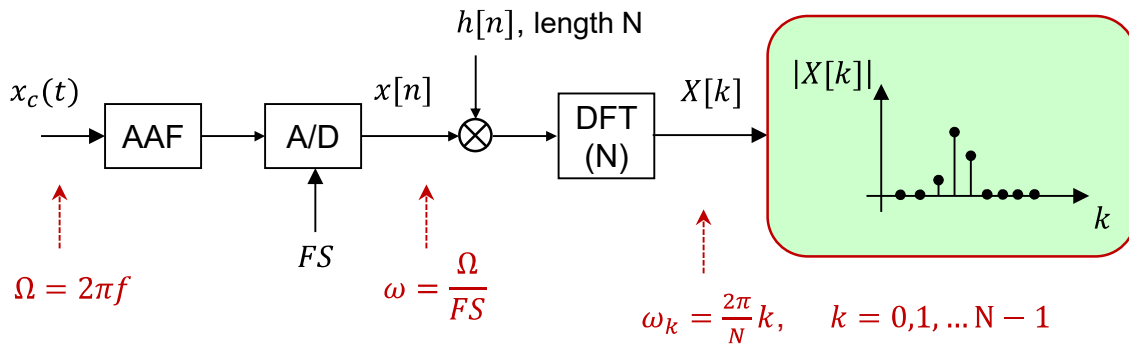
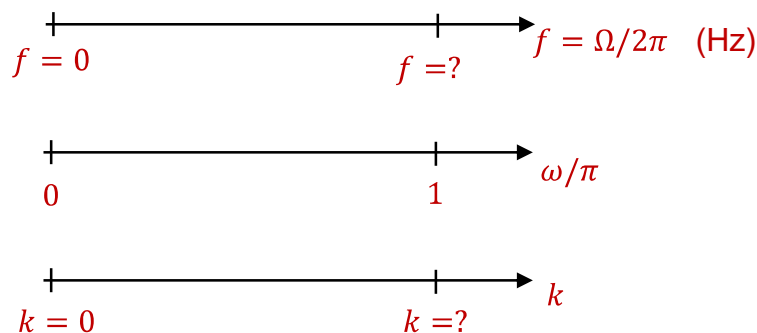


Figure 1: Basic spectrum analysis setup.

This signal processing setup uses the DFT in order to compute and represent the magnitude spectrum $|X[k]|$ (i.e. the magnitude of the absolute value of the sampled discrete-time Fourier Transform -DTFT) of a continuous-time signal, for example, a sinusoid $x_c(t) = \sin(2\pi ft)$. This analog sinusoidal signal is (ideally) sampled (the sampling frequency is FS) and then a segment of length N of the discrete-time signal $x[n]$ is multiplied by a window $h[n]$ having also length N . As illustrated in Figure 1, after multiplication, the signal is then DFT transformed and the resulting discrete-frequency sequence, $X[k]$, is analyzed. A local maximum in the magnitude spectrum $|X[k]|$ denotes (*but does not directly specify, except in very special conditions*) the frequency of the sinusoid. This Lab uses the STM32F7 Discovery kit, as well as an oscilloscope equipped with the FFT MATH functionality, in order to implement the whole signal processing chain that is illustrated in Figure 1, and to allow appreciating its real-time operation and, in particular, to understand what spectral leakage is, and how the spectral leakage phenomenon depends on f , $h[n]$, and N .

Question 1: Recalling P2P Exercise 2, in different points of the signal processing chain illustrated in Figure 1, frequency is looked at in different perspectives, i.e. according to different (but equivalent) axes as the following scheme illustrates. Clarify in the following scheme the correspondence between the three axes (two of them continuous, one of them discrete) and replace the question marks by the correct values (assume that $FS = 8000$ Hz).



Question 2: The frequency resolution of the DFT is defined as the frequency separation between two consecutive DFT lines (also referred to as DFT bins). If $FS = 8000 \text{ Hz}$, and $N = 256$, what is the DFT frequency resolution in Hertz ?

Spectral resolution $\rightarrow \frac{\text{---}}{N} = \text{---} \text{ (Hz)}$

3.1 Windowing

In P2P Exercise 2 (and also in previous FunSP classes) it was clarified already that if $h[n] = w_{Rect}[n] = u[n] - u[n - N]$, i.e. if $h[n]$ consists of the N -point rectangular window, then the magnitude of its frequency response is given by

$$|W_{Rect}(e^{j\omega})| = N \left| \frac{\text{sinc}(\omega N/2)}{\text{sinc}(\omega/2)} \right| \quad (1)$$

where $\text{sinc}(\theta) = \frac{\sin(\theta)}{\theta}$. (Note that Matlab considers a different definition: $\text{sinc}(\theta) = \frac{\sin(\pi\theta)}{\pi\theta}$)

The rectangular window is the default window (i.e. even if one does not realize that one window is in fact being used). In spectrum analysis, other windows are commonly used that offer a more convenient control of the leakage phenomenon (which will be specifically addressed in Section 3.2). A family of interesting windows is called the generalized Hamming window and is defined as

$$w_H[n] = \left[\alpha - (1 - \alpha) \cos \frac{2\pi n}{N-1} \right] w_{Rect}[n], \quad n = 0, 1, \dots, N-1 \quad (2)$$

where α is a parameter. Two particular cases that are widely used correspond to the Hanning window ($\alpha = 0.50$), and to the Hamming window ($\alpha = 0.54$).

Question 3: Show that the frequency response of $w_H[n]$, as in Equation (2), is given by

$$W_{Hamm}(e^{j\omega}) = N e^{-j\omega \frac{N-1}{2}} \left(\alpha \frac{\text{sinc}(\frac{\omega N}{2})}{\text{sinc}(\frac{\omega}{2})} + \frac{1-\alpha}{2} \left(\frac{\text{sinc}(\frac{(\omega - \frac{2\pi}{N-1})N}{2})}{\text{sinc}(\frac{(\omega - \frac{2\pi}{N-1})1}{2})} + \frac{\text{sinc}(\frac{(\omega + \frac{2\pi}{N-1})N}{2})}{\text{sinc}(\frac{(\omega + \frac{2\pi}{N-1})1}{2})} \right) \right) \quad (3)$$

3.2 The leakage phenomenon

The leakage phenomenon can be easily understood by running the Matlab script (available on the Moodle platform) named `FunSP_sinus_spectra_RectHamm.m` which demonstrates the complete signal processing that is illustrated in Figure 1 when the input is a continuous-time sinusoid $x_c(t) = \sin(2\pi ft)$ having frequency f Hertz. The `FunSP_sinus_spectra_RectHamm.m` demonstrator assumes that $FS = 8000$ Hz and, to facilitate visualization, assumes that $N=64$. Both the rectangular and the Hamming windows are admitted and are selected by uncommenting the desired case, as illustrated in the following section of the Matlab code.

```
%
% unselect ONLY one of the following: RECTANGULAR or HAMMING window
%
window='Rect';
% window='Hamm';
```

As provided, the `FunSP_sinus_spectra_RectHamm.m` demonstrator not only represents $|X[k]|$ but it also represents the most relevant part of the true DTFT the DFT is a sampling of. This helps to understand how the magnitude of the different DFT bins vary, and why. The frequency of the analog sinusoid, f , varies between 500 Hz and 1600 Hz, in steps of 2.5 Hz.

When the `FunSP_sinus_spectra_RectHamm.m` demonstrator starts, a graphical illustration similar to Figure 2 is displayed. In this case, the $|X[k]|$ spectrum consists of only one non-zero DFT line (or DFT bin) in the Nyquist range (i.e. between 0 and the Nyquist frequency).

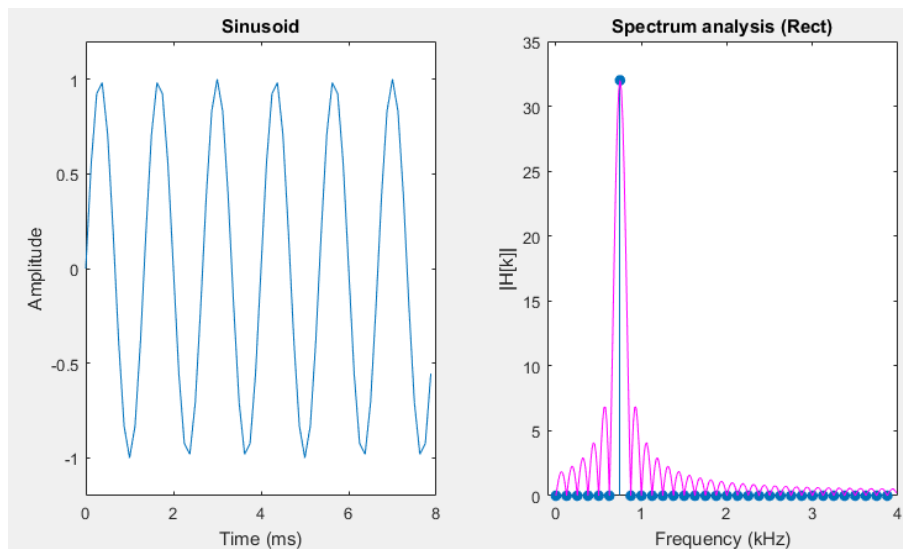


Figure 2: Spectrum analysis of a sinusoid when the analysis window is the rectangular window and when there is no leakage. The $|X[k]|$ spectrum is represented in blue, and the most relevant part of the DTFT is represented in magenta.

This is because one DFT bin falls exactly in the middle of the $|W_{Rect}(e^{j\omega})|$ function when it is shifted to the frequency of the sinusoid. All other DFT bins fall on the zeros of that function, and, thus, are zero-valued. Under these circumstances, as illustrated in Figure 2, the index k of the non-zero

$|X[k]|$ DFT bin also indicates the number of complete periods of the sinusoid that fall within the length of $w_{Rect}[n]$.

Given that only one DFT bin is different from zero (in the Nyquist range), we say that there is no leakage. Leakage refers to the fact that a sinusoid may “leak” to other DFT bins that do not directly reflect the frequency of that sinusoid. Figure 3 illustrates such a case.

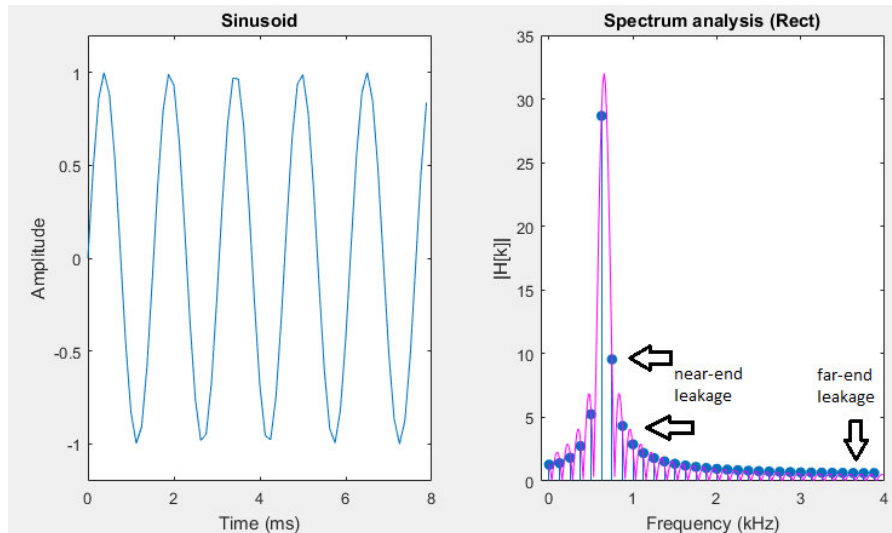


Figure 3: Spectrum analysis of a sinusoid when the analysis window is the rectangular window and leakage exists. The $|X[k]|$ spectrum is represented in blue, and the most relevant part of the DTFT is represented in magenta.

In this case, the frequency of the sinusoid is not exactly sampled by any DFT bin and, therefore, the sinusoid “leaks” to all DFT bins, with the consequence that the DFT bin closer to the exact sinusoid frequency exhibits the highest DFT peak, whereas the remaining ones exhibit lower magnitude values, especially those more distant from the peak. The DFT bins that are closer to the highest peak reflect ‘**near-end leakage**’ and those that are more distant reflect ‘**far-end leakage**’.

Different windows exist (the same ones that were presented in the context of the ‘window method’ of FIR filter design) that offer different tradeoffs between near-end leakage and far-end leakage. Two popular (and frequently used) windows, as we shall see in the context of this Lab, are the Hamming and the Hanning windows (see Section 3.1).

Figure 4 illustrates the graphical output of the `FunSP_sinus_spectra_RectHamm.m` demonstrator when the Hamming window is selected. In this case, the significant part of the near-end leakage is limited to a few DFT bins surrounding a local peak, and the far-end leakage is very small. This is very convenient, for example, to minimize the interference between different co-existing sinusoids.

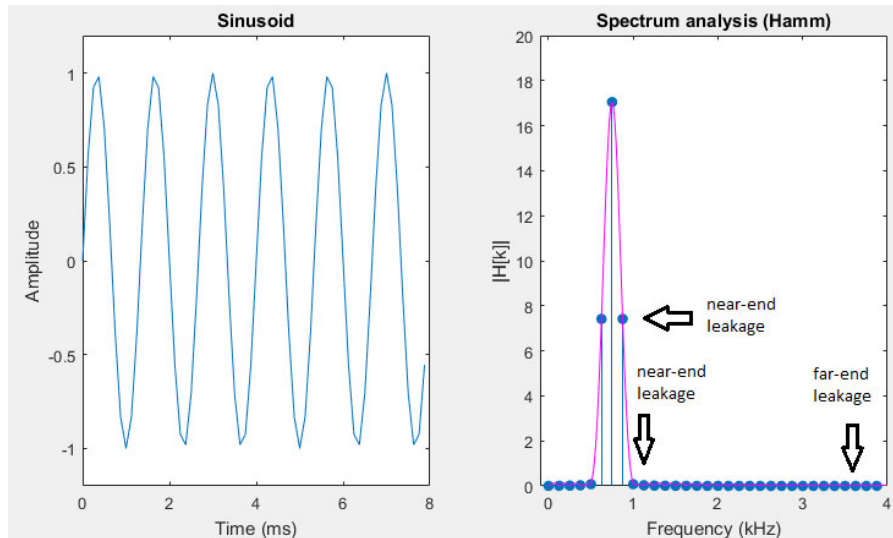


Figure 4: Spectrum analysis of a sinusoid when the analysis window is the Hamming window. The $|X[k]|$ spectrum is represented in blue, and the most relevant part of the DTFT is represented in magenta.

Question 4: When using either window (but especially in the case of the rectangular window), as the frequency of the input sinusoid increases (or decreases) the leakage varies cyclically between a minimum and a maximum, why is that ?

Question 5: Using the rectangular window, consider two situations in the Matlab code `FunSP_sinus_spectra_RectHamm.m`:

-the default code that includes the instruction: `x=sin(OMEGA0*n/FS);`

-a modified version of the code where the instruction `x=sin(OMEGA0*n/FS);` is replaced by `x=0.5*exp(1j*OMEGA0*n/FS);`

In these two cases, as the frequency of the input sinusoid increases (or decreases) the leakage behaves differently. Explain that different behavior and explain what fact justifies that different behavior.

3.3 Configuring the oscilloscope as a spectrum analyzer

In this Lab, the oscilloscope is configured as a real-time spectrum analyzer so as to compare its operation with the real-time operation of the STM32F7 Disco kit, when both implement the same signal processing that is illustrated in Figure 1, albeit with different parameters.

Turn on the oscilloscope, press once on the MATH button and, on the screen menu select the MATH Operation (Operação MATH). Then, adjust the Horizontal Scale to “500 Hz (10kS/s)”, this configures the A/D sampling frequency of the oscilloscope to 10 kHz. The “500 Hz (10kS/s)” indication appears in the lower part of the oscilloscope screen, see Figure 5. Then, select Source CHAN 1 (Fonte CHAN 1) and select Window rectangular (Janela rectangular).

Question 6A: Look for the INFO menu on the oscilloscope and find out what is the size of the FFT that is implemented on the oscilloscope.

Note: don't worry if you cannot find that information immediately, Question 6B proposes an experimental approach for you to find the answer. In the case of the modern oscilloscopes, this information is not readily available and you may ignore this question.

Question 6B: Taking advantage of the predictable behavior that is mentioned in Question 4, and by either increasing or decreasing smoothly the input sinusoidal frequency above and below 650 Hz (in steps of 0.1 Hz), you can find very easily the size of the FFT that is implemented on the oscilloscope (consider that the size is a power-of-two number).

Note: this experimental procedure is also addressed in Question 11. In the case of the modern oscilloscopes, the 0.1 Hz step is too large and, therefore, you may ignore this question

Oscilloscope FFT length → $N =$

4 Programming the STM32F7 as a real-time spectrum analyzer

In this Lab, we use the `main()` project file that is named `stm32f7_fft256_dma.c` and that is available on the usual STM32F7 source code directory (SRC). Its C code is listed next.

```
// stm32f7_fft256_dma.c

#include "stm32f7_wm8994_init.h"
#include "stm32f7_display.h"
#include "hamming256.h"

#define SOURCE_FILE_NAME "stm32f7_fft256_dma.c"
```

```

extern volatile int32_t TX_buffer_empty; // these may not need to be int32_t
extern volatile int32_t RX_buffer_full; // they were extern volatile int16_t in F4 version
extern int16_t rx_buffer_proc, tx_buffer_proc; // will be assigned token values PING or
PONG

#define N (PING_PONG_BUFFER_SIZE)
#define MAGNITUDE_SCALING_FACTOR 32
#define TRIGGER 32000

typedef struct
{
    float real;
    float imag;
} COMPLEX;

#include "fft.h"

COMPLEX twiddle[N];
COMPLEX cbuf[N];
int16_t sinebuf[N];
float32_t outbuffer[N];
float32_t inbuffer[N];

void process_buffer()
{
    int i;
    int16_t *rx_buf, *tx_buf;
    int16_t left_sample, right_sample;

    // determine which buffers to use
    if (rx_buffer_proc == PING) {rx_buf = (int16_t *)PING_IN;}
    else {rx_buf = (int16_t *)PONG_IN;}
    if (tx_buffer_proc == PING) {tx_buf = (int16_t *)PING_OUT;}
    else {tx_buf = (int16_t *)PONG_OUT;}

    for (i = 0; i < N ; i++)
    {
        left_sample = *rx_buf++;
        right_sample = *rx_buf++;
        cbuf[i].real = ((float)left_sample);
        // cbuf[i].real = ((float)left_sample)*hamming[i];
        cbuf[i].imag = 0.0;
        inbuffer[i] = cbuf[i].real;
    }

    fft(cbuf,N,twiddle);
    arm_cmplx_mag_f32((float32_t *) (cbuf), outbuffer,N);
    for (i = 0; i < N ; i++)
    {
        outbuffer[i] = outbuffer[i]/MAGNITUDE_SCALING_FACTOR;
        if (i==0)
            *tx_buf++ = TRIGGER;
        else
            *tx_buf++ = (int16_t)(outbuffer[i]);
        *tx_buf++ = inbuffer[i];
    }

    TX_buffer_empty = 0;
    RX_buffer_full = 0;
}

```

```

int main(void)
{
    int n;
    for (n=0 ; n< N ; n++)
    {
        twiddle[n].real = cos(PI*n/N);
        twiddle[n].imag = -sin(PI*n/N);
    }

    stm32f7_wm8994_init(AUDIO_FREQUENCY_8K,
                       IO_METHOD_DMA,
                       INPUT_DEVICE_INPUT_LINE_1,
                       OUTPUT_DEVICE_HEADPHONE,
                       WM8994_HP_OUT_ANALOG_GAIN_6DB,
                       WM8994_LINE_IN_GAIN_0DB,
                       WM8994_DMIC_GAIN_0DB,
                       SOURCE_FILE_NAME,
                       GRAPH);

    while(1)
    {
        while (!(RX_buffer_full && TX_buffer_empty)){}
        BSP_LED_On(LED1);
        process_buffer();
        plotFFT(outbuffer, PING_PONG_BUFFER_SIZE, NO_AUTO_SCALING);
        BSP_LED_Off(LED1);
    }
}

```

This code reuses C code that was discussed and implemented in the context of Lab 4 (DMA operation and LCD graphical capabilities), namely `stm32f7_loop_dma.c` and `stm32f7_loop_graph_dma.c`, and configures the STM32F7 operation to be oriented to frame-based processing, instead of interrupt-based processing. This means that the signal processing algorithms run during the time that is required for a new buffer (i.e. e vector) of A/D samples to be filled and to become available for processing, instead or running during a single sampling period (i.e. according to the interrupt-based method), as in previous labs addressing IIR or FIR filtering.

Thus, frame-based processing is naturally suited for FFT computation, and FFT-based processing. In summary, rather than processing one sample at a time, the FFT algorithm is applied to blocks, or frames, of samples.

Take a moment to analyze the `stm32f7_fft256_dma.c` code. Notice that the default value of `PING_PONG_BUFFER_SIZE` (defined in header file `stm32f7_wm8994_init.h`) is 256.

Question 7: What is the DFT frequency resolution (in Hertz) of the FFT that is implemented in the `stm32f7_fft256_dma.c` code ?

Note: this question is obviously related to Question 2.

Spectral resolution → _____ = (Hz)

4.1 The FFT of a signal in real-time

Program `stm32f7_fft256_dma.c` combines function `fft()` with a real-time program to implement a very simple spectrum analyzer. The program uses the DMA-based frame-processing mechanism implemented in program `stm32f7_loop_dma.c` and calls function `fft()` from function `process_buffer()`.

Blocks of `PING_PONG_BUFFER_SIZE = 256` real-valued input samples (for both left and right channels) are transferred from the WM8994 ADC, via the SAI peripheral, to memory using DMA, and their 256-point complex DFTs are computed using function `fft()`. Left channel input samples are written to buffer `inbuffer` for later plotting using MATLAB (which we will **not** analyze in this Lab). The scaled magnitudes of the elements of the frequency-domain representations are written to the DAC (again via the SAI peripheral and using DMA) and to buffer `outbuffer` for plotting (which we will **not** analyze in this Lab). The value of `PING_PONG_BUFFER_SIZE` is set in file `stm32f7_wm8994_init.h` and is equal to the number of sampling instants over which data in one DMA transfer block correspond to. Since the WM8994 is a stereo codec, each DMA transfer block comprises `PING_PONG_BUFFER_SIZE` 16-bit left-channel sample values plus `PING_PONG_BUFFER_SIZE` 16-bit right-channel sample values.

Question 8: The STM32F7 LCD screen just represents the first 128 values of $|X[k]|$. Why is that the second half of $|X[k]|$ does not provide more information than that provided by the first half?

Note: this question is obviously related to Question 1.

4.2 Modifying the program to reduce spectral leakage

As discussed in Section 3.2, one method of reducing spectral leakage is to multiply each block of input samples by a tapered window function prior to computing its DFT. This can be done by altering the line of program `stm32f7_fft256_dma.c` that reads

```
cbuf[i].real = ((float)left_sample);
```

to

```
cbuf[i].real = ((float)left_sample)*hamming[i];
```

which relies on the following line of the C code

```
#include "hamming256.h"
```

File `hamming256.h` contains the declaration of an array `hamming`, initialized to contain a 256-point Hamming window.

You can investigate the effects of several other window functions using header files `bartlett256.h`, `hann256.h`, `blackman256.h`, and `kaiser256.h`.


4.3 Compiling and running the C code

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

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

```
C:\uivision\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 (μ Vision). Replace the existing `main()` file in that project by the new `main()` that is `stm32f7_fft256_dma.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 Performing various spectrum analysis experiments

5.1 Setting-up for this lab experiment

Set the function generator to generate a sine wave having 5 Vpp and 1500 Hz. Using a “T-BNC” 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**).

The oscilloscope should be configured as specified in Section 3.3 (Configuring the oscilloscope as a spectrum analyzer).

5.2 Observing leakage due to the rectangular window

When the function generator is set to generate a 1500 Hz sinusoidal wave, you should obtain a graphical representation on the oscilloscope, and on the STM32F7, as illustrated on the **left** hand-side of Figure 5.

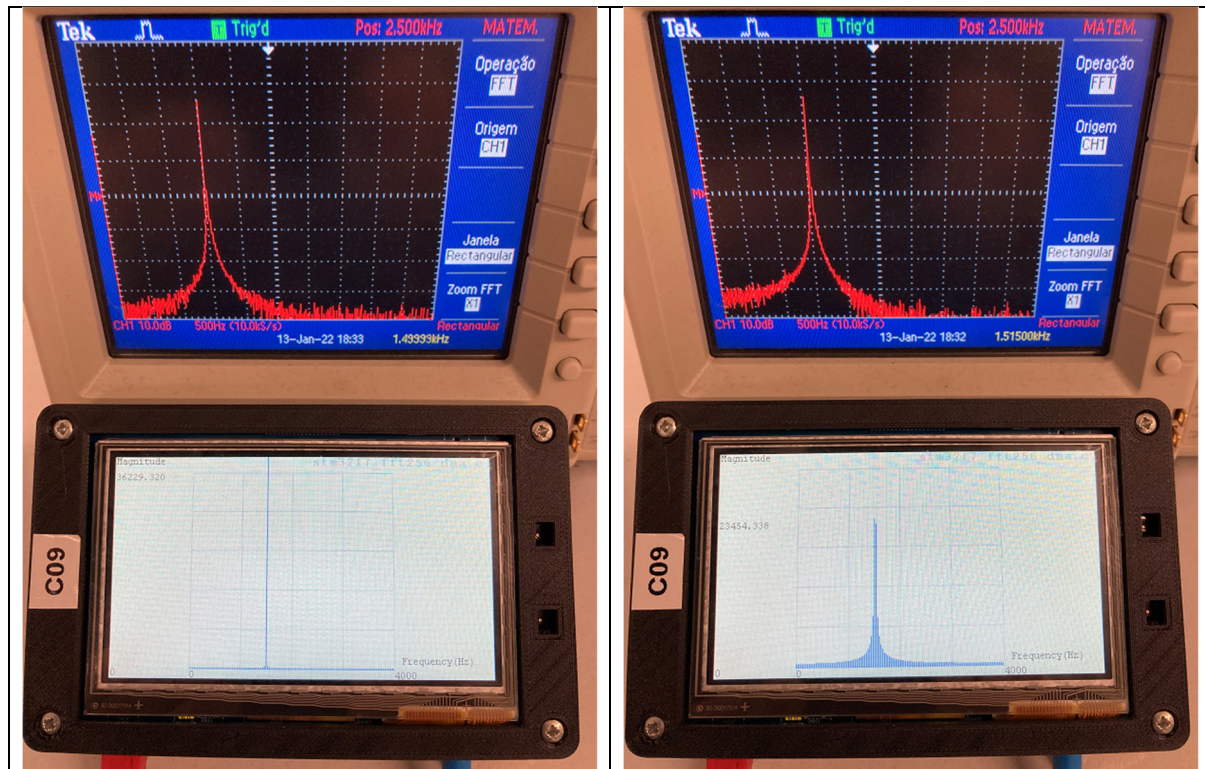


Figure 5: Real-time spectrum analysis by the oscilloscope and the STM32F7 of a sinusoid when the analysis window is the Rectangular window.

Question 9: In this case (1500 Hz sinusoidal frequency), the spectrum analysis on the oscilloscope exhibits leakage while the STM32F7 does not, how do you explain that ?

Now, adjust the function generator to generate a 1515 Hz sinusoidal wave. You should obtain a graphical representation on the oscilloscope, and on the STM32F7, as illustrated on the **right** hand-side of Figure 5.

Question 10: In this case (1515 Hz sinusoidal frequency), the spectrum analysis on both the oscilloscope and STM32F7 exhibits leakage, how do you explain that ?

Question 11: What is the next sinusoidal frequency (above 1515 Hz) that makes that the spectrum analysis on the STM32F7 does not exhibit leakage, why ?

Question 12: When the sinusoidal frequency increases above 4000 Hz in the case of the STM32F7, you should notice that the spectral peak on the STM32F7 screen disappears. However, when the sinusoidal frequency increases above 5000 Hz in the case of the oscilloscope, the spectral peak on the oscilloscope screen moves to the left (which reflects that the frequency “decreases”), how do you explain that ?

Now, adjust the function generator to generate a 625 Hz sawtooth wave (i.e. a triangular wave having a 5% duty cycle). You should obtain a graphical representation on the oscilloscope, and on the STM32F7, as illustrated in Figure 6.

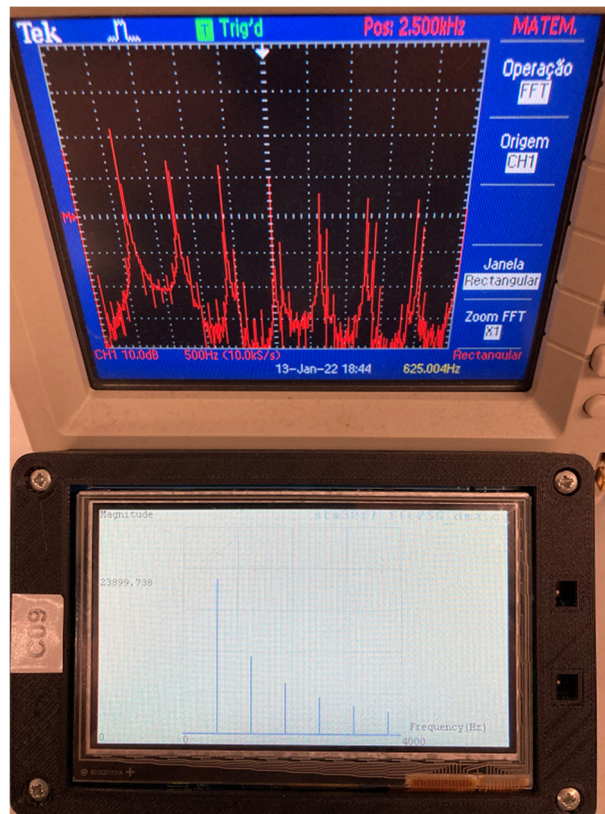


Figure 6: Real-time spectrum analysis by the oscilloscope and the STM32F7 of a sawtooth wave when the analysis window is the Rectangular window.

Question 13: In this case (625 Hz sawtooth frequency), the spectrum analysis on the oscilloscope exhibits leakage while the STM32F7 does not, i.e. all of the sinusoidal components pertaining to the sawtooth wave are displayed on the STM32F7 spectral analysis without significant leakage. How do you explain that ?

5.3 Observing leakage due to the Hamming window

Now, stop the STM32F746G board real-time execution and modify the `stm32f7_fft256_dma.c` C code to set the time analysis window to the Hamming window as suggested in Section 4.2. Then, proceed as usual to compile the code, to download it to the STM32F746G board (by starting the debugger), and then to run the code.

On the oscilloscope menu, set the window to Hanning window (Janela Hanning)

Set again the function generator to generate a 1500 Hz sinusoidal wave, you should obtain a graphical representation on the oscilloscope, and on the STM32F7, as illustrated in Figure 7.

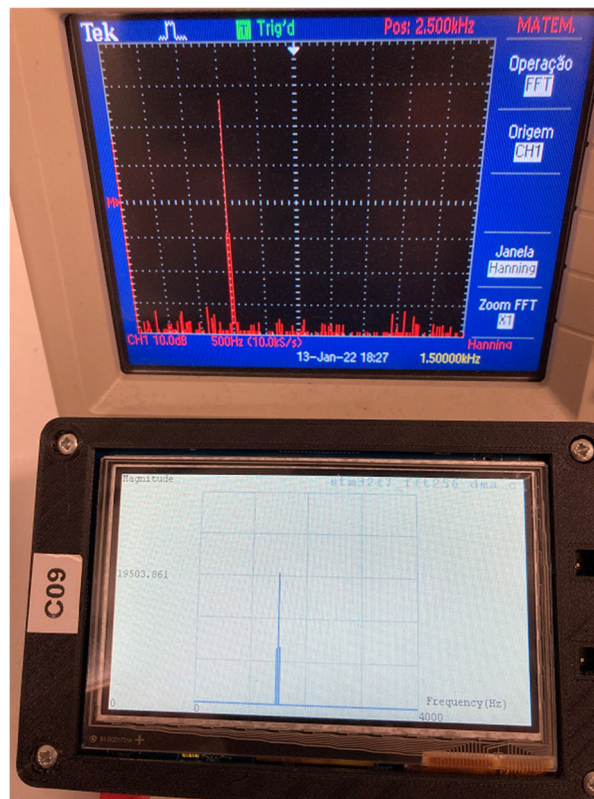


Figure 7: Real-time spectrum analysis by the oscilloscope and the STM32F7 of a sinusoid when the analysis window is the Hanning window (oscilloscope) or the Hamming window (STM32F7).

Question 14: Considering that the Hamming and the Hanning windows are not very much different from each other, is the spectrum analysis on both the oscilloscope, and STM32F7, similar in terms of leakage behavior? As you increase or decrease the sinusoidal frequency, is the leakage behavior on both platforms as expected taking into consideration the simulations in Section 3.2 involving the Hamming window? In what sense is it as expected?

6 Conclusions

This Lab motivated the use of the FFT in a number of simulation and real-time experiments so as to i) understand and recognize the leakage phenomenon as well as the aliasing phenomenon, ii) to understand and recognize the effect of windowing in moderating the leakage phenomenon, iv) to observe and understand real-time spectrum analysis using both the STM32F7 platform and the oscilloscope, and v) to perform basic spectrum analysis using either a sinusoidal input or a more structured periodic waveform such as the sawtooth wave.

7 Additional References

Using DMA controllers in STM Discovery boards:

https://www.st.com/content/ccc/resource/technical/document/application_note/27/46/7c/ea/2d/91/40/a9/DM00046011.pdf/files/DM00046011.pdf/jcr:content/translations/en.DM00046011.pdf

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

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