L.EEC025 - Fundamentals of Signal Processing (FunSP)

2021/2022 – 1st semester

Week05, 15 Nov 2021

Objectives:

-measuring the frequency response of a moving average filter

- **clarifying that a moving average filter may be implemented using different difference equations**
- **clarifying the theoretical frequency response of a moving average filter**
- **measuring the frequency response of a real-time moving average filter**

DSP Education Kit

LAB 4

Measuring the frequency response of a moving average filter

Issue 1.0

Contents

1 Introduction

1.1 Lab overview

The examples in this exercise introduce some of the concepts of Finite Impulse Response (FIR) filtering. A simple experimental method is explored of estimating the magnitude frequency response of a moving-average filter that is implemented in real-time using two different difference equations.

2 Requirements

To carry out this lab, you will need:

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

3 The Moving Average Filter

The Moving Average filter is widely used in DSP and is arguably one of the easiest of all digital filters to understand. It is particularly effective at removing high-frequency random noise from a signal.

The moving average filter operates by taking the arithmetic mean, or average value of a number of past input samples, in order to form each output sample. This may be represented by the following equation:

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i],$$

where $x[n]$ represents the nth sample of an input signal, and $y[n]$ is the nth sample of the filter output, which is equal to the average value of the previous N input samples. A five-point moving average filter is implemented by the provided example program `stm32f7_average_intr_modNOV2021.c`. using two different difference equations.

4 A quick clarification regarding the previous lab

A clarification is in order regarding question 4 in the previous lab experiment (LAB 3, Figure 1). That question was about understanding what the complete STM32F746G board implemention was equivalent to, considering the real-time discrete-time system that was implemented in that lab experiment.

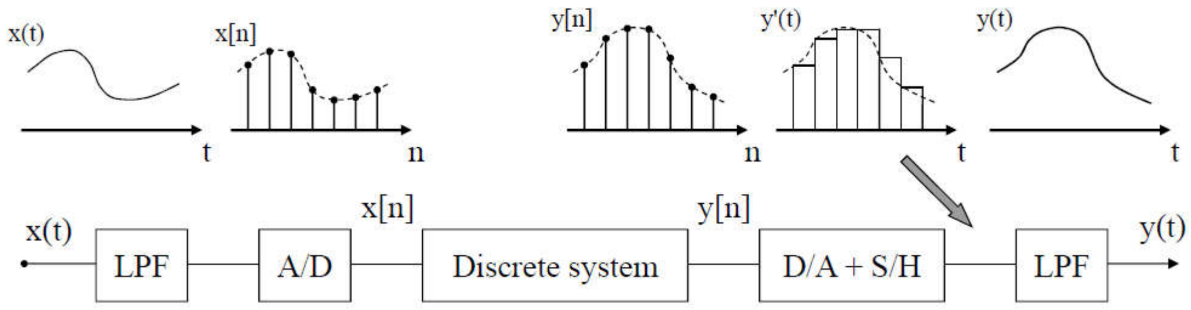


Figure 1: Complete sampling and reconstruction chain implemented by the STM32F746G board.

It should be clear by now to all FPS Students that the whole (meaning: end-to-end) system behaves as an analog system, or filter. Unless stated otherwise, both input analog filter (the “anti-aliasing” filter before A/D conversion), and the output analog filter (the “anti-imaging” filter after D/A conversion and Sample-and-Hold), are low-pass filters (LPF) and their cut-off frequency is the Nyquist frequency (F_N), i.e. half the sampling frequency (F_S). On the other hand, if y[n]=x[n], as it was the case of last week’s lab experiment, then the discrete-time system acts as an all-pass filter. Thus, from a system end-to-end point of view, the cascade of two identical low-pass filters having cut-off frequency $F_N=F_S/2$, and an all-pass discrete-time system, is equivalent to a single low-pass filter cutting-off all frequencies above F_N.

5 Preliminary analysis before the current lab

The current lab implements two causal discrete-time systems whose difference equations are as specified in Equation (1) and Equation (2).

$$y[n] = \frac{x[n]+x[n-1]+x[n-2]+x[n-3]+x[n-4]}{5} \qquad (1)$$

$$y[n] = \frac{x[n]-x[n-5]}{5} + y[n-1] \qquad (2)$$

Using Z-Transform analysis, show that the system implemented according to Equation (2) is the same as the system that is implemented according to Equation (1). In this demonstration, you just need to take the Z-Transform of Equation (2) and show that it can be reduced to the Z-Transform of Equation (1). The demonstration implies a polynomial division.

As a clarification on how the polynomial division should be performed, considerer the following toy example. Let us admit that we want to perform the following polynomial division: $\frac{1+z^{-1}+z^{-2}}{1-z^{-1}}$.

$$\begin{array}{ccc|cc}
z^{-2} & z^{-1} & 1 & -z^{-1} & 1 \\ \cline{4-5}
-z^{-2} & z^{-1} & & -z^{-1} & -2 \\ \cline{1-2}
0 & 2z^{-1} & 1 & & \\
 & -2z^{-1} & 2 & & \\ \cline{2-3}
 & 0 & 3 & &
\end{array}$$

Thus, for this toy example, we can express $\frac{1+z^{-1}+z^{-2}}{1-z^{-1}} = -z^{-1} - 2 + \frac{3}{1-z^{-1}}$.

6 Observation of Frequency Response Using a Sinusoidal Input Signal

In this lab experiment, we will use the modified `main()` project file that is named **`stm32f7_average_intr_modNOV2021.c`** and that is available on the Moodle platform. Its C code is listed next.

```
// stm32f7_average_intr.c

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

#define SOURCE_FILE_NAME "stm32f7_average_intr.c"
#define N 5
#define invN 1.0/N

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 h[N];
float32_t x[N+1] = {0, 0 ,0, 0, 0, 0};
float32_t yn_1=0.0;

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 yn = 0.0;


  x[0] = (float32_t)(rx_sample_L);
  for (i=0 ; i<N ; i++) yn += h[i]*x[i];
  tx_sample_L = (int16_t)(yn);

  yn = (x[0]-x[N]) * (float32_t) (invN) + yn_1;
       //yn = (x[0]-x[N]);

  tx_sample_R = (int16_t) (yn);

       for (i=N ; i>0 ; i--) x[i] = x[i-1];
  yn_1 = yn;

  BSP_LED_Toggle(LED1);

  return;
}

int main(void)
{
  int i;

  for (i=0 ; i<N ; i++) h[i] = (float32_t)(invN);

  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:

- what lines of code implement Equation (1)
- what lines of code implement Equation (2)
- that the output according to Equation (1) is routed to one of the two output channels, and that the output according to Equation (2) is routed to the second output channel.

After unzipping it, take the `stm32f7_average_intr_modNOV2021.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_average_intr_modNOV2021.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.

6.1 Setting-up for this lab experiment

Set the function generator to generate a sine wave having 5 Vpp and 100 Hz. Using a BNC-BNC cable, take the output of the function generator to the oscilloscope just to ascertain first that the sine wave is being generated as intended.

As shown in Figure 2, connect the output of a sinusoidal signal generator (ie. 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 two BNC-BNC cables, take the LEFT and RIGHT channels of the STM32F746G LINE OUT output to CHAN1 and CHAN2 inputs of the oscilloscope.

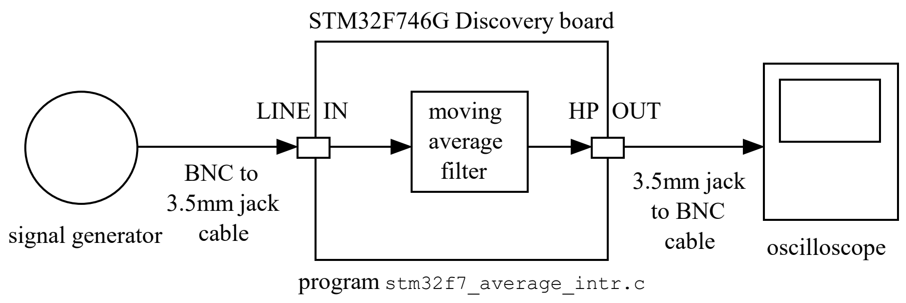


Figure 2: Connection diagram for measuring the magnitude frequency response of the five-point moving average filter implemented by program `stm32_average_intr_modNOV2021.c` using a signal generator and an oscilloscope.

Using the oscilloscope SETTINGS button and menu, make sure that the Vpp and frequency of the signals on both channels are being measured in real-time. You should obtain a graphical representation as that illustrated in Figure 3.

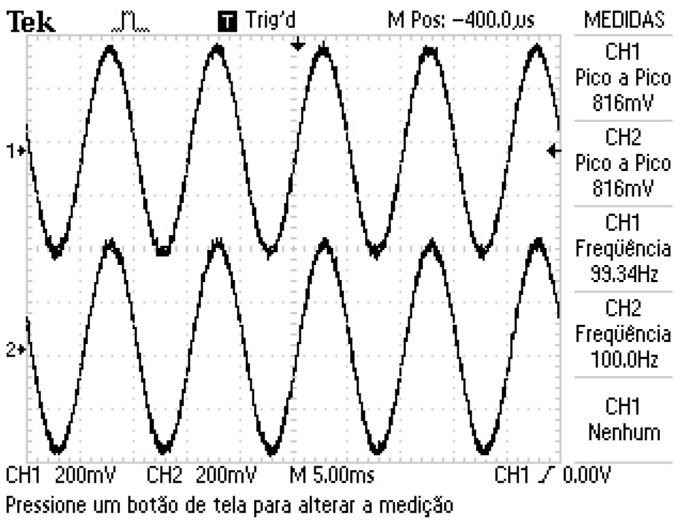


Figure 1: Output signals from program `stm32f7_average_intr_ modNOV2021.c` *viewed on the oscilloscope.*

Question: by increasing the input frequency, do you notice any difference between the output signals that are generated according to Equation (1) and Equation (2) ? What does that tell you ?

Question: by considering the answer to the previous question, is there a strong reason why one of the difference equations may be preferred, in practice, over the other ?

6.2 Sketching the frequency response of the moving average filter

The frequency response of a filter tells us its gain at different frequencies, and hence 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. The gain of the moving average filter is higher at low frequencies than at high frequencies, and there are some frequencies at which the gain is zero. Overall, the moving average filter has a low pass characteristic.

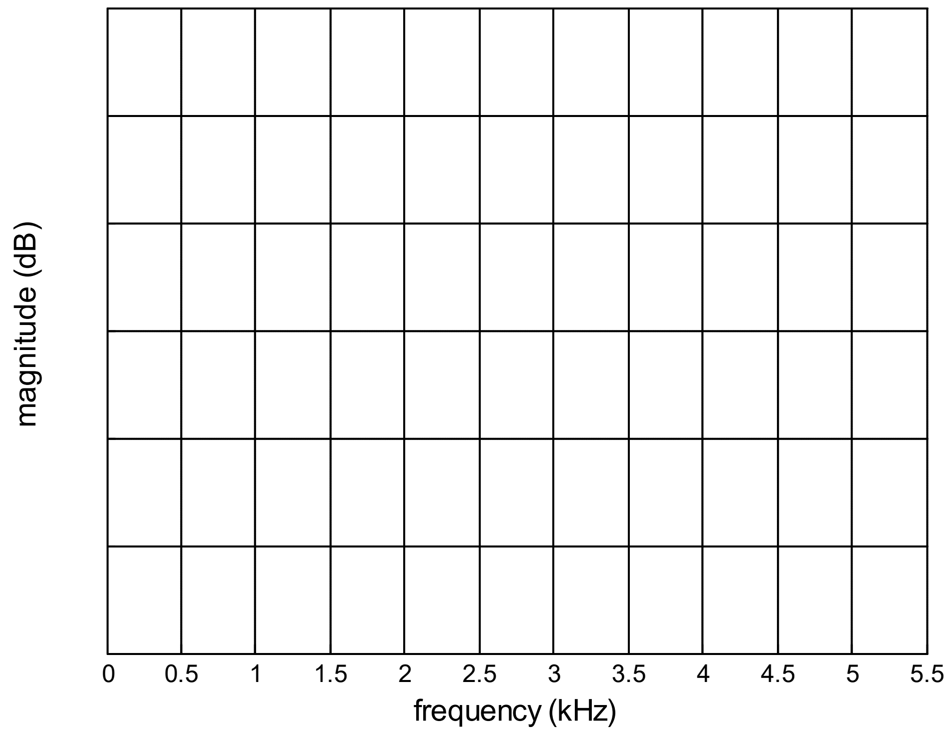
In this experiment, you will take note of the Vpp of the wave (just one of them, as Vpp levels should be quite similar between the two channels) represented on the oscilloscope, starting with 100 Hz, and then taking frequencies that are multiples of 500 Hz, i.e. 0.1 kHz, 0.5 kHz, 1 kHz, 1.5 kHz, …

Question: show first that if the frequency of the input sinusoid is `freq` (in kHz), then the theoretical gain of the moving average filter of length 5 is given by:

$$\left|\frac{1}{5}\frac{\sin(5\pi\, freq\,/8)}{\sin(\pi\, freq\,/8)}\right|$$

Question: Identify first what input frequencies make the output signals to become zero. Are those frequencies as expected ?

Use the following plot in order to take note of the (approximate) Vpp levels for the different frequencies. In this plot, the vertical axis should be labelled “Amplitude (Vpp)” and not “Magnitude dB”.



Note that the voltage level, in the Vpp sense, is not important (and depends, in any case, on the amplitude of the input signal). What is important is its *relative* level across the range of frequencies measured.

Now, take the ratio between the Vpp level for each tested frequency, and the Vpp level that was obtained for `freq=0.1` kHz (which we take as an approximation for `freq = 0` Hz).

Question: are these ratios consistent with the theoretical gains ?

6.3 Changing the filter in one of the channels (extra, only if time permits)

The above code included the implementation of a difference equation by using the following code:

```
  yn = (x[0]-x[N]) * (float32_t) (invN) + yn_1;
       //yn = (x[0]-x[N]);
```

Now, change that part of the code such that is becomes:

```
       // yn = (x[0]-x[N]) * (float32_t) (invN) + yn_1;
  yn = (x[0]-x[N]);
```

Question (not mandatory for this lab class): What is the theoretical magnitude frequency response that emerges from this new difference equation ?

Now, compile the modified code, download it to the STM32F746G board and run it.

Question (not mandatory for this lab class) As you increase the frequency of the sinusoidal input from around 0.1 kHz to around 5 kHz, is the Vpp level of the filtered signal consistent with the magnitude frequency response of the new filter ?

7 Conclusions

This laboratory exercise has introduced FIR filtering using as a baseline a moving average filter of length 5. Students should now be ready to assess in the lab the frequency response of other FIR filters running in real-time on the STM32F746G board.

8 Additional References

Moving average filters:

https://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch15.pdf