Lab 5 - FIR and IIR Filter Design and Analysis

Section	L01	L02	L03	L04
Lab Date	Dec. 2	Nov. 25	Dec. 3	Nov. 26
Due Date for Report	Dec. 5	Dec. 5	Dec. 5	Dec. 5

Assessment: 5% of the total course mark.

Objectives:

- To gain experience in FIR and IIR filter design and analysis in `MATLAB` and explore Moving Target Indicator (MTI) with the TMS 320 DSP processor

Assessment:

- Your grade for this lab will be based on your ability to use filter design tools from the `MATLAB` Signal Processing Toolbox and to analyze and implement the resultant filters within `MATLAB` and to understand and implement MTI in the TMS 320 DSP processor, and on your reporting of the results.
- Clearly label all plots and their axes (points for style will be deducted otherwise).
- Please attend the lab section to which you have been assigned.
- You can complete this lab with one lab partner.
- By the end of the lab session you must demonstrate to your TA the MATLAB code and the filter output on `MATLAB` and the TMS 320 DSP processor.
- All MATLAB source code and c code (updated "RadarProcessor_lab5.c") must be submitted on Avenue to Learn by the end of your designated lab session.
- Each pair of students should complete one lab report together. The report has to be submitted by **11:59 pm on Dec. 5, 2023**.

Experiments:

1. **The truncation method of FIR design**
   (a) Follow the math from the lecture slides to design linear-phase lowpass filters with the cutoff frequency $\omega_c = \frac{\pi}{3}$ using the truncation method for filter orders $M = 20, 50$ and 150 (i.e., $N = 21, 51$ and 151).
   (b) Demonstrate the Gibbs phenomenon for these filters.
   (c) Confirm that these filters have linear phase.
   (d) Show that the `MATLAB` Signal Processing Toolbox function `firls()`, which designs linear-phase FIR filters using least-squares error minimization, produces identical impulse responses to the truncation method for each of the filter orders $M = 20, 50$ and 150.
   (e) Create a signal 1000 samples long of white Gaussian noise. Convolve this signal with each of the filter impulse responses obtained in part (a). Now try using the `MATLAB` function `filter()`. What similarities and differences are there between the outputs of `filter()` and `conv()`?
2. **Chebyshev Type II lowpass IIR filter**

(a) Design linear-phase lowpass **FIR** filters with the cutoff frequency $\omega_c = \frac{\pi}{3}$ using the window method with a Dolph-Chebyshev window for filter orders $M = 20, 50$ and 150. The `MATLAB` Signal Processing Toolbox function `fir1()` can be used to implement the window method and the `MATLAB` Signal Processing Toolbox function `chebwin()` can be used to obtain the desired Dolph-Chebyshev windows. Note that you will need to specify the amount of sidelobe attenuation for your Dolph-Chebyshev windows. Try some different values, and demonstrate what effect this has on:

i. the passband ripple and stopband attenuation.

ii. the slope of the transition region.

for one of the filters ($M = 20, 50$ or 150).

(b) Design Chebyshev Type II lowpass **IIR** filters with the cutoff frequency $\omega_c = \frac{\pi}{3}$ using the `MATLAB` Signal Processing Toolbox function `cheby2()` for filter orders $M = N = 20, 50$ and 150. Note that you will need to specify the stopband attenuation – make it the same as one of the values that you used for the Dolph-Chebyshev window sidelobe attenuation in part (a).

(c) Determine the impulse responses of these filters. Do they appear to be infinite?

(d) Compare and contrast the IIR filter frequency responses (both the magnitude and the phase) with the FIR frequency responses from part (a). Explain the differences in light of filter theory as it relates to causal FIR and IIR filters.

(e) Create a signal 1000 samples long of white Gaussian noise. Using the `filter()` function, calculate the output of the IIR filter for this signal. Now estimate the output by convolving the input signal with a very long impulse response that you obtained from part (c). How good is the estimate?

3. **Detecting Targets in Background Clutter Utilizing a Moving Target Indicator with the TMS 320 DSP Processor**

(a) **Background Knowledge**

i. Scenario With Barriers

Most of the highway roads have barriers installed at the sides for safety consideration, see Figure 1. However, this brings issues to the radar in detecting other vehicles. The barriers appear as obstacles that extend from near the radar to the maximum detection range, and the signal from the barrier dominates the received signal. As a result, the true target is indistinguishable from the barriers.

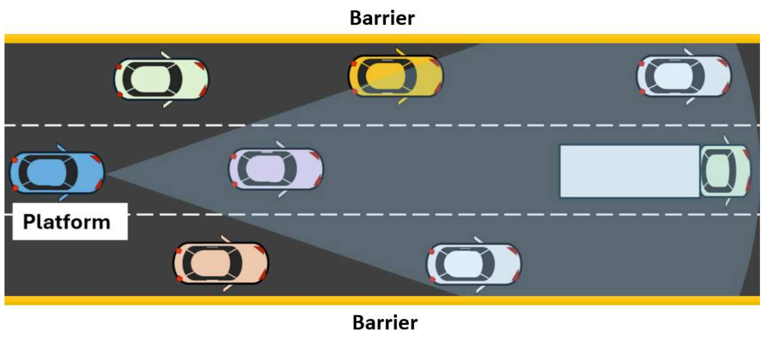

Figure 1: Illustration of the radar detection case with barriers.

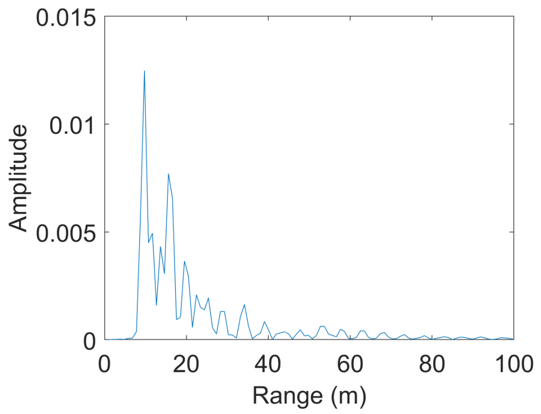

Figure 2: The range plot of the signal with barrier present before filtering.

Figure 2 shows a sample amplitude vs range plot in this case. Since a barrier is present at every range, the target return signal is mixed with barrier returns. Doppler shift can be used to differentiate the signal that comes from the vehicle over the one from a barrier. The Doppler shift, f_{Doppler}, is given by

$$f_{\text{Doppler}} = \frac{v_{\text{object}} - v_{\text{platform}}}{c} f_{\text{emitted}} \tag{1}$$

where v_{object} and v_{platform} are the speed of the object and the platform in the range direction, respectively, f_{emitted} is the transmitted frequency and c is the speed of light. In our simulated data, the platform is stationary, so the Doppler shifts of the barriers, which are stationary, are measured to be zero.

ii. Two-Pulse Canceller Moving Target Indicator

To filter out the signal that comes from the barrier, Moving Target Indicator (MTI) processing is used. As discussed before, the barriers are always static in the range, while the targets are not static. Therefore, a straightforward way to distinguish them is to calculate the difference in the range information at slightly different times (i.e., different sweeps). The static part (the barriers) will be cancelled out, and the non-static (the targets) part will remain. Specifically, in our case, FFTs of dechirped signals (amplitude vs range) corresponding to two adjacent sweeps are subtracted. This is the idea of the simplest two-pulse canceller MTI with the filter equation given as follows:

$$y[n] = x[n] - x[n-1] \tag{2}$$

iii. Two-Pulse Canceller MTI in Frequency Domain

To understand better how the MTI works, let us explore it in the frequency domain. In the previous labs, we obtained the range information from the radar signals. By evaluating the Fourier transform of the received signal from different sweeps at a particular range, the Doppler information can be obtained. Figure 3 shows the Doppler frequency plots in different cases.

In figure 3(a)-(b), there is no barrier in this specific case, and the first target is around 21 m away from the radar. Therefore, at a range of 7.8125 m, we only have noises since no signal is reflected back at this point. At a range of 21.4844 m, where roughly

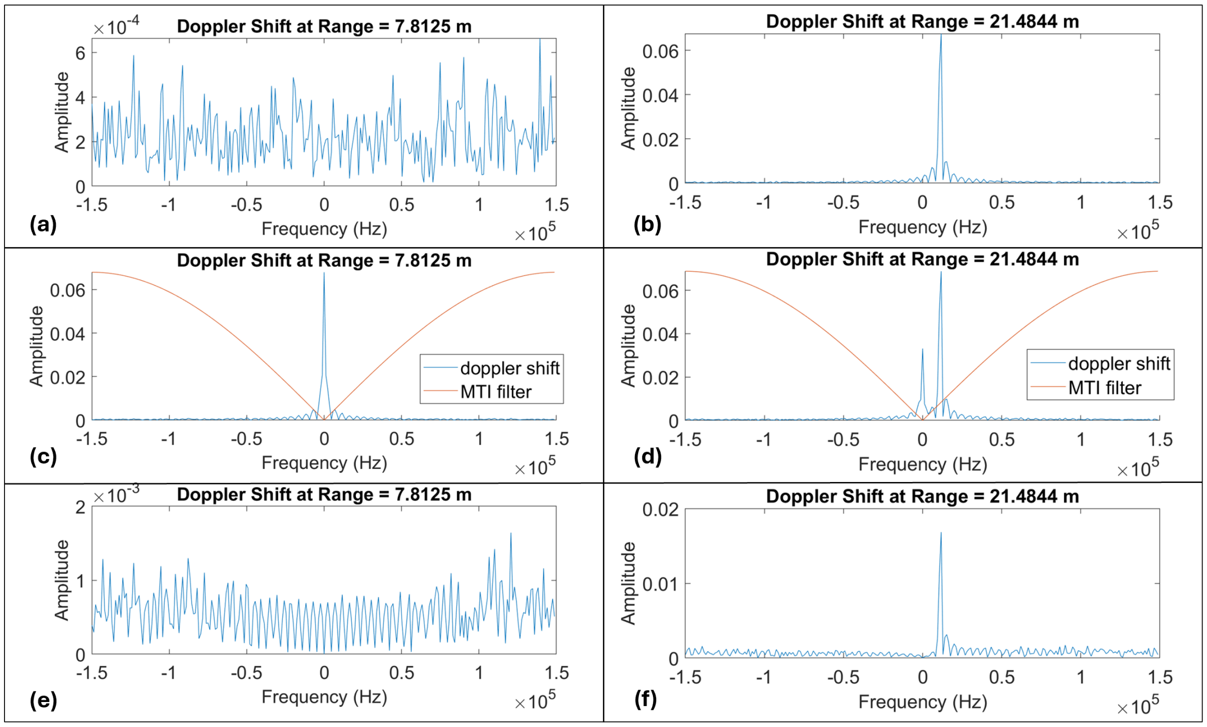

Figure 3: The Doppler frequency plots of (a) signal without barrier at range = 7.8125 m, (b) signal without barrier at range = 21.4884 m, (c) signal with barrier at range = 7.8125 m, (d) signal with barrier at range = 21.4844 m, (e) signal with barrier after MTI filtered at range = 7.8125 m, and (f) signal with barrier after MTI filtered at range = 21.4844 m.

the first target is located, there is a positive Doppler shift, representing the first target is moving toward the radar.

In figure 3(c)-(d), we simulated everything the same as the previous simulation, except that barriers were present. The closest point of the barrier is around 7 m away from the radar. Therefore, at range of 7.8125 m, the barrier is detected with Doppler shift at 0 Hz, meaning it is relatively static to the radar. At range of 21.4844 m, the first target presents with a velocity toward the radar, plus the barrier with Doppler shift at 0 Hz.

It is obvious that the barrier signal always has 0 Hz Doppler shift. Considering the targets we are detecting are always moving relative to the radar, a high-pass filter should be able to do a good job at filtering out the barrier signal while maintaining the moving target signal, and this is exactly what MTI does. If we write the equation of the two-pulse MTI filter (2) in impulse response:

$$h[n] = \delta[n] - \delta[n-1] \tag{3}$$

Plotting it in frequency domain (figure 3(c)-(d)), we found that it is a simple high-pass filter as expected. If we do a convolution of the signal with the impulse response, we get the figure 3(e)-(f). At range of 7.8125 m, the barrier Doppler shift at 0 Hz is filtered out completely and everything we have here is noise, just like figure 3(a). At range of 21.4844 m, the barrier signal is filtered out and the target signal remains, even though it is attenuated. However, since this low-pass filter is not a perfect low-pass filter (not sharp at the edge), the target signal is also attenuated, and the high frequency noise is amplified relatively to the target signal, making the signal to noise ratio (SNR) smaller than the case in figure 3(b).

(b) **Lab Procedure**

i. Test Lab 4 code with barrier data.

A. Right click the project under the explorer, pull down and click “Properties”.

B. Go to “Build”, then go to “include Options”. Add the path “Y:/COE4TL4/RadarProcessor/data/groupXX”

C. Change line 9 of “RadarProcessor_lab4.c”: change **#include “data_dechirp.h”** to **#include “with_barrier/data_dechirp.h”**

D. Compile and run the code with your lab 4 code. Observe the oscilloscope and explain to the TA why it is hard to detect the targets.

ii. Import the lab 5 source file into the project:

A. Create a copy of the project that you worked on in lab 4 and rename as “RadarProcessor_Lab5” and open this project in CSS.

B. Delete “RadarProcessor_Lab4.c” from this project.

C. Right click the project under the explorer, and choose “Add Files..”.

D. Direct to the folder “Y:/COE4TL4/RadarProcessor/src”, and add the source code “RadarProcessor_lab5.c” to the project. MAKE SURE TO CHOOSE THE OPTION OF “COPY FILES”, NOT “LINK FILES”.

iii. Run the code with “USE_MTI = 0” and then with “USE_MTI = 1” (“USE_MTI” is an integer variable defined on top of the source C code). Observe the oscilloscope, describe the difference, and explain why.

iv. The code is running with two-pulse canceller, please complete the function “set_mti_h()”, so that it also supports three-pulse canceller. The three-pulse canceller filter equation is given as:

$$y[n] = x[n] - 2x[n-1] + x[n-2] \tag{4}$$

Compile and run with three-pulse canceller.

v. Draw the DTFT of the three-pulse canceller MTI filter on MATLAB and explain whether it is more effective than the two-pulse canceller MTI in our case. Verify that with what you observed in the oscilloscope.

vi. Modify the function “prepare_fft_data()” to add a hanning window for FFT calculation. Observe the effect of it.

A. Create a function “double hanning_window(int n, int N)” and implement it using the following equation:

$$w_h = 0.5\left(1 - \cos\left(\frac{2\pi n}{N}\right)\right) \tag{5}$$

B. Multiple the signal with w_h.

vii. In the provided data, the platform is nearly stationary. How would you modify the MTI filter if the platform moves? Please explain the reason.

4. **Bonus Question**
Bonus [1 mark]: Explore and implement a better IIR filter for the MTI filtering. Demonstrate the improvement to the TAs. You need to include the explanation in the report as well to get the bonus mark.

Report: The report should contain:

- **ALL** the `MATLAB` plots and the oscilloscope screenshots **WITH BRIEF DESCRIPTIONS**.
- Answer the questions 1(e), 2(a), 2(d), 2(e), 3(b-iii), 3(b-v), 3(b-vii) with some discussions.

You **do not** need to include the `MATLAB` code or the C code in the report. However, you have to submit the `MATLAB` code and C code separately.