CPX 3 Report

ECE 434: Digital Signal Processing

November 22, 2024

C1C Victor Chen

C1C Ben Cometto

C1C Nick Csicsila

C1C Geoff Stentiford

Contents

1 Introduction **3**

2 Block 1: Calibration **4**

3 Block 2: Time Gain Compensation **5**

4 Block 3: Noise Remove Filter and Bandwidth Limiting **6**

5 Block 4: Upsampling (2x) **7**

5.1 Implementation 7

5.2 Analysis 8

6 Block 5: Beamforming **11**

7 Conclusion **12**

List of Figures

1 Signal Processing Stages 3

2 Frequency Response of Upsampling Low Pass Filter 7

3 Upsampled Linear Function 8

4 Zoomed Portion of Upsampled Linear Function 8

5 Upsampled Data in Time 9

6 Upsampled Data in Frequency 9

7 Zoomed Portion of Upsampled Data in Time 10

8 Zoomed Portion of Upsampled Data in Frequency 10

List of Tables

1 Parameters for Upsampling Low Pass Filter 7

2 Parameters for Upsampling Low Pass Filter 10

Abstract

Sonar, sending and receiving sounds as a sensor, is a valuable tool that electrical and computer engineering provides. In this project, CPX 3, we are improving upon a provided system. The system collected data from a phased array of four omni-directional microphones, after a pulse from a single approximately omni-directional speaker. We are working to improve both the speed and quality of the processing of this data, using the digital signal processing and problem solving techniques learned this semester. [REWRITE THIS ONCE REST IS WRITTEN AND WRITE MORE ABOUT METHODS AND RESULTS]

1 Introduction

Sonar, ultrasound, and radar are all based on the same principle: transmit a wave, and record its echos. The only differences are the type of wave or the frequencies involved. In this project, we are using sonar. Working in the audible range enables the use of low-cost, common speakers and microphones. Additionally, it allows us to hear to pulses and echos, which can serve as another problem solving tool.

For this project, we are working with recorded data. The recorded data was produced using a single omni-directional speaker to transmit the pulse and a phased array of four omni-directional microphones to receive the echos. Each microphone is given a channelL X_1, X_2, X_3, and X_4.

The signal processing is done inside of a MATLAB script, following the process diagrammed in Figure 1. **NEEDS TO BE REDRAWN WITH OUR NEW ORDER**

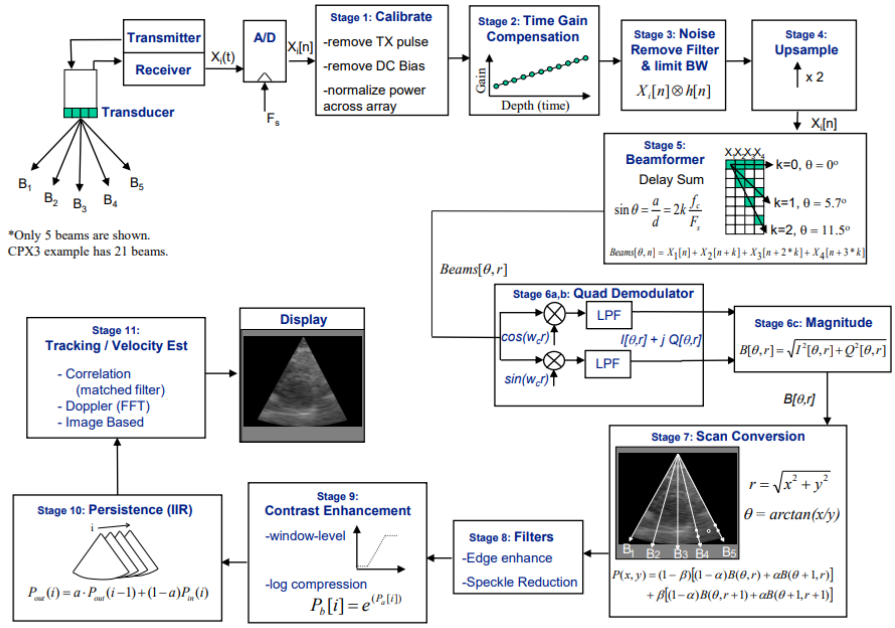

Figure 1: Signal Processing Stages

Each stage is described in more detail, to include its purpose, design goals, implementation, and analysis results, in its respective section.

2 Block 1: Calibration

3 Block 2: Time Gain Compensation

4 Block 3: Noise Remove Filter and Bandwidth Limiting

5 Block 4: Upsampling (2x)

Upsampling involves increasing the number of samples by interpolating the existing data. In this case, we are upsampling our data so that we are able to form more beams, and thus have a higher resolution image.

The goal for the upsampling block is to double the number of samples. In order to do this, there are two steps. First, zeros must be added in between the datapoints. Second, the zero-padded data must be low pass filtered with

$$f_c = \frac{F_{s,old}}{2} = 50\,\text{kHz}$$

to remove the reflected image of the data.

5.1 Implementation

We must implement both of these steps in a very fast way, without sacrificing quality.

First, MATLAB is meant for working with vectorized functions, and has vectorized methods of replacing certain indexes. Thus, the quickest way to add a zero after every sample (double the number of samples) is to replace every other entry of a matrix of zeros with the data point.

Second, in order to interpolate, we will use the minimum order filter possible that preserves the necessary information, because a lower order results in less multiplies and thus a quicker function. Additionally, FIR was chosen in order to preserve the relative phase of the received signals, which is necessary for preventing distortion in the sonar image. Equirriple was used as the design method.

In this case, an order 3 filter with the specifications in Table 1 works. Our information is contained in the band less than 20 kHz , and thus we can set f_{pass} to 20 kHz . Then, because our $F_{s,old} = 100\,\text{kHz}$, there is a significant amount of bandwidth between our data's frequency spectrum and its reflected image. Thus, we can set f_{stop} to be 69 kHz , the sharpest it can be while still being order 3. Through various trials, an A_{stop} of 40 dB sufficiently suppresses the image, will also allowing the filter to be order 3.

Order	Fpass	Fstop	Apass	Astop
3	20 kHz	69 kHz	1 dB	40 dB

Table 1: Parameters for Upsampling Low Pass Filter

The frequency response of the low pass filter can be seen in Figure 2.

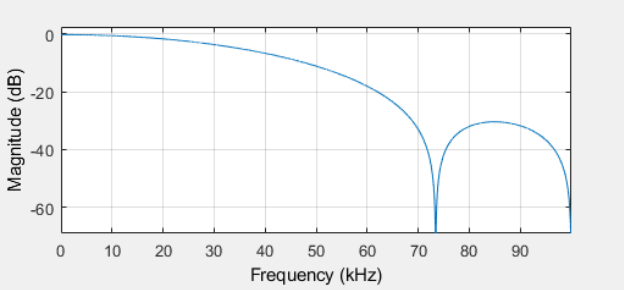

Figure 2: Frequency Response of Upsampling Low Pass Filter

In order to preserve this speed, the function header was modified to accept the numerator taps of the low pass filter, matrix of zeros, and the length of the matrix of zeros as input in addition to the data to be upsampled.

5.2 Analysis

An initial, simple test to confirm that the function upsamples properly is by inspecting a linear function. This can be seen in Figure 3.

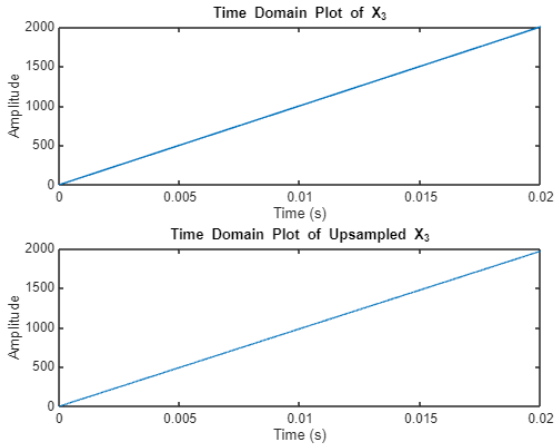

Figure 3: Upsampled Linear Function

It can be seen in Figure 4 that the upsampled version contains the same data, but has twice as many points.

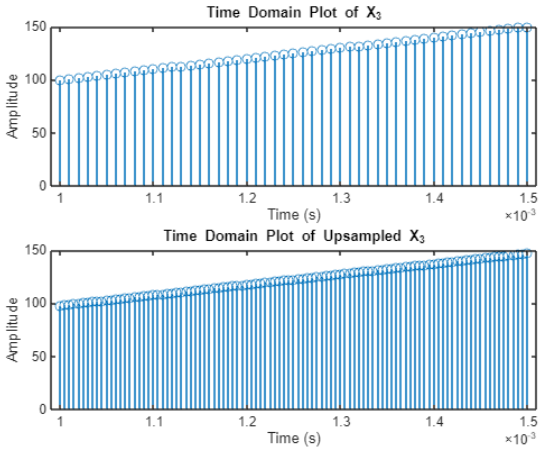

Figure 4: Zoomed Portion of Upsampled Linear Function

Testing on various other types of synthetic data, such as sinusoids and linear combinations of sinusoids, further confirmed that the function was implemented as intended.

As the next test, we can upsample the provided test data (unmodified by the other stages) to ensure it has no unintended effects. As can be seen in Figure 5 and Figure 6, te data was interoplated correctly. Each channel is represented by a color.

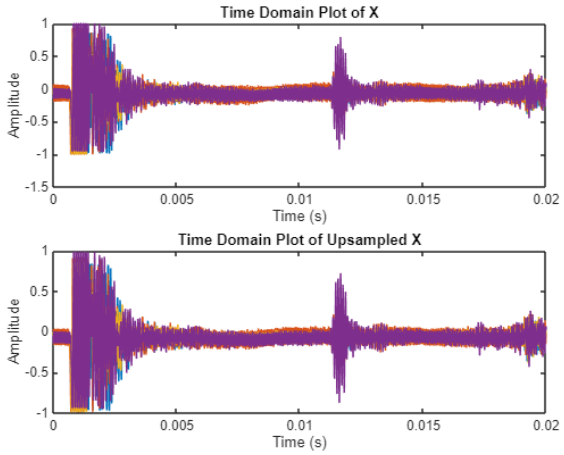

Figure 5: Upsampled Data in Time

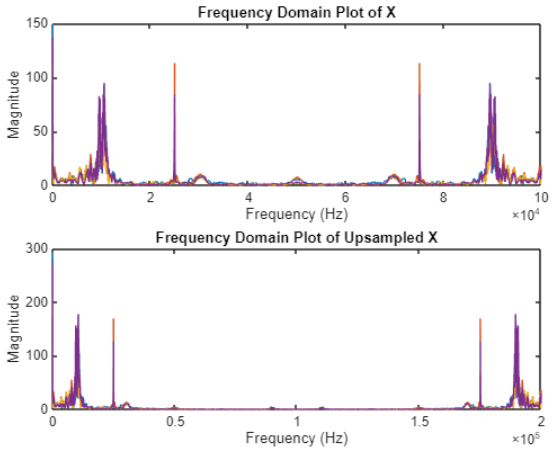

Figure 6: Upsampled Data in Frequency

Zooming in for Figure 7 and Figure 8, we can more clearly see that the interpolation was successful.

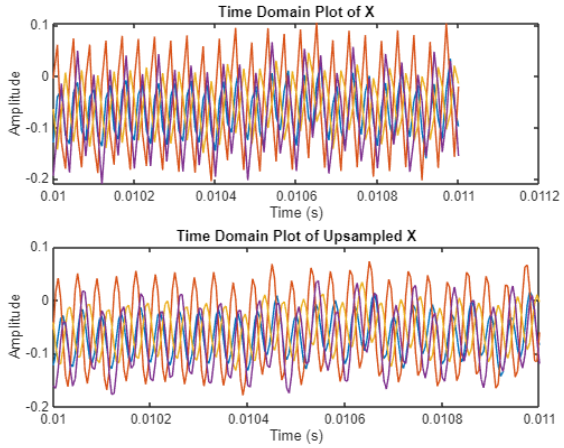

Figure 7: Zoomed Portion of Upsampled Data in Time

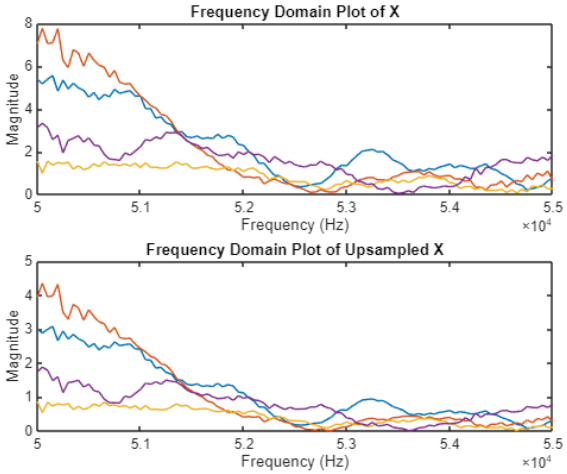

Figure 8: Zoomed Portion of Upsampled Data in Frequency

Therefore, the function successfully interpolates the data at a high quality. The next step in the analysis is in the timing. Benchmarking is difficult, but here we are only comparing the newly written function to the provided `.p` file. In this case, each function was timed (using `tic` and `toc`) over 10,000 iterations, and the average runtimes can be seen in Table 2. The same data (the provided test data) was used, and the computer was in as similar a state as possible.

Function	Time (μs)
Provided `.p`	5455
Rewritten `.m`	64.7

Table 2: Parameters for Upsampling Low Pass Filter

Therefore, the rewritten `upsampling.m` function provides a high quality upsampling by 2 and is about 100 times faster than the original function.

6 Block 5: Beamforming

7 Conclusion

[Needs to be written]

Documentation

We worked as a team on this project. Additionally, Ben Cometto used resources such as the mathworks website for MATLAB syntax help (ex, confirming how to index every other entry of a matrix) and had a discussion with ChatGPT regarding recording phase, available here.