Laboratory 1 – Aliasing

Objective

In this lab you will explore how the sample rate for discrete signals affects the frequency content.

Introduction

If a signal is sampled too slowly to cover the full range of frequencies in the signal, high frequency components will be aliased into lower frequencies, producing a sampled signal that does not accurately represent the original one. When a signal is sampled, the frequency spectrum becomes a series of periodic spectra, with each one having a frequency range up to the Nyquist rate, which is half the sampling rate. For example, if you have a signal with frequencies up to 100 Hz, you must sample at a rate of 200 Hz to accurately capture all the information in the signal. If sampled at only 120 Hz, the highest frequency in the discrete signal would be 60 Hz. Frequencies above 60 Hz in the original signal would be "aliased" into lower frequency components. (This topic was covered in detail in Class 5.)

The diagram illustrating aliasing in the frequency domain is shown in Figure 1.

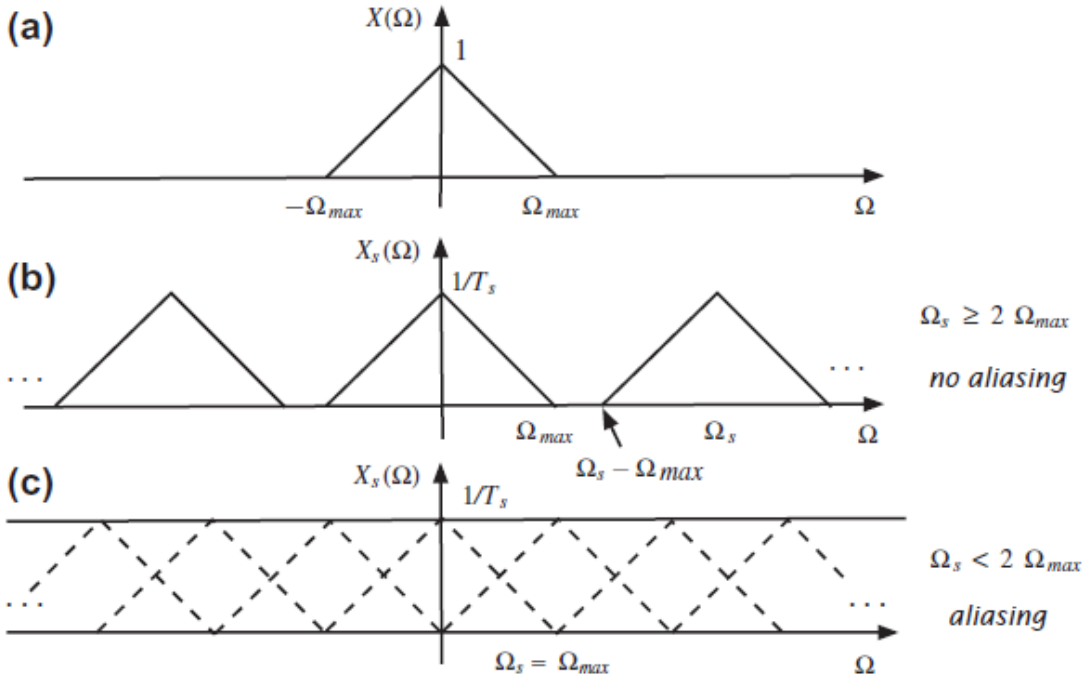

Figure 1: Diagram illustrating aliasing in the frequency domain.

The spectrum of an analog signal, with a maximum frequency of Ω_{max}, is shown in Fig. 1(a). If the signal is sampled at more than twice Ω_{max}, the repeated spectra do not overlap, as shown in Fig. 1(b). If sampled at less than $2\Omega_m$, the repeated copies overlap the original one, as shown in Fig. 1(c), producing frequency content in the sampled signal that was not in the original one.

Digitized music, such as WAV or MP3 files, typically have sampling rates of 44.1 kHz, which means that the maximum frequency that can be represented in the music is 22.05 kHz. This is above the usual range of human hearing which is about 20 kHz. Children and young adults may be able to detect these high frequencies, but as we age, hearing sensitivity decreases. You might still be able to hear frequencies of 20 kHz; I can probably only hear frequencies up to 15 kHz.

For this lab, you will experiment with subsampling digital music that you download, producing signals with lower sampling rates.

Instructions

Part A – Getting some data

Download a piece of music, of your choice. You should be able to find a WAV or MP3 file. If you Google the name of the piece you want, you will probably find it somewhere. If you have any trouble getting a file, let me know and I will help.

Using MATLAB, read the file with the following command:

```
>> [music, Fs] = audioread('your_file_name.mp3');
```

or

```
>> [music, Fs] = audioread(your_file_name.wav');
```

Listen to the piece using the command:

```
>> sound(music,Fs);
```

If it sounds weird, the sampling rate, Fs, in the file is probably wrong. (This has happened to me, where the return argument Fs was 22050, when in fact it is really 44100.) You will need to know the correct sampling rate, so be sure to check this.

Look at the size of the file to see if there is more than one channel. For example, my music file is in stereo, so the size of the file is:

```
>> size(music)
   ans = 1525872  2
```

Work with only one channel. For example,

```
>> music_ch1 = music(:,1);
```

will create a signal with one channel.

Create a time vector so that you can plot your file to visualize the signal and see how long it is:

```
>> time = (0:length(music_ch1)-1)/Fs;
>> plot(time,music_ch1);
```

An example is shown in Figure 2.

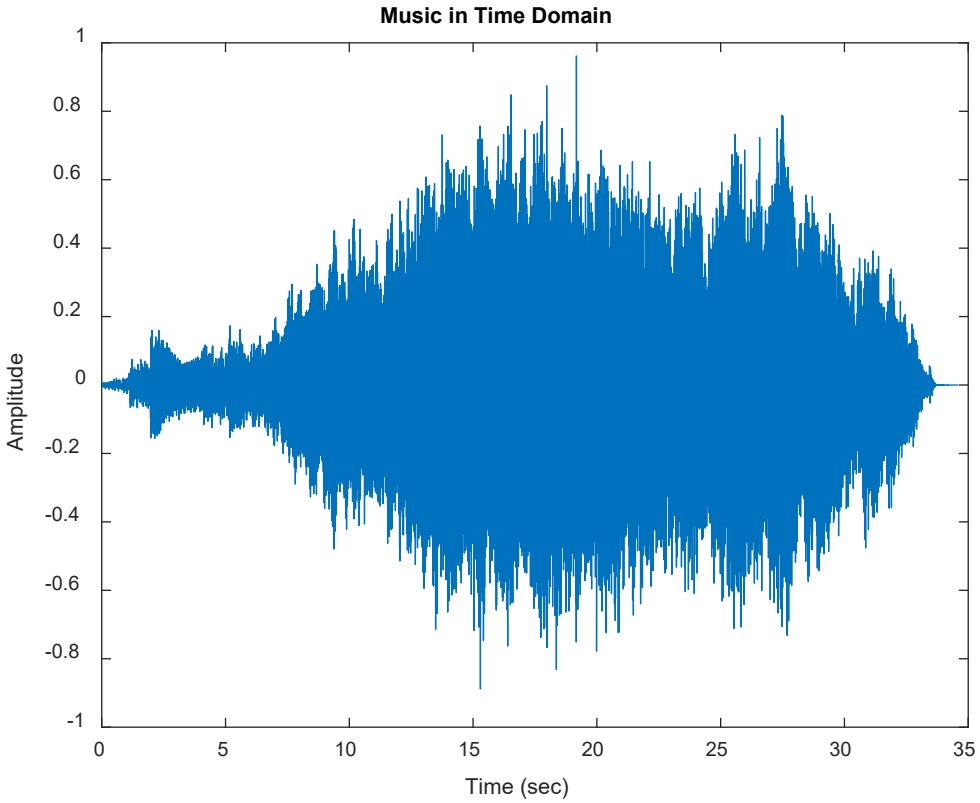

Figure 2: Plot of sound file in the time domain.

You can see from Fig. 2 that this piece is about 35 seconds. You may want to reduce the length to something that you will not get tired of listening to several times. For example,

```
>> music = music_ch1(1:450000);
```

cuts this piece down to 10 seconds.

Now that you have your sound file to work with, the interesting part begins.

Part B – Aliasing Experiments, Time-domain Observations

With your music in hand, play the sound back at several sampling rates and listen to how it changes. The commands below make copies of the music at sampling rates of Fs/5 = 8820 Hz, Fs/10 = 4410 Hz, etc. (You do not need to make this many sub-samplings; use your judgement as to what sub-samplings you want to experiment with.)

```
>> music_by_5  = music(1:5:end);
>> music_by_10 = music(1:10:end);
>> music_by_15 = music(1:15:end);
>> music_by_20 = music(1:15:end);
>> music_by_25 = music(1:25:end);
>> music_by_30 = music(1:30:end);
```

Frequencies above half the sampling rate will be aliased in each of these cases. The sound will get weirder and weirder as you decrease the sampling rate. For example, music_by_20 will

have an effective sampling rate of Fs/20 = 2205 Hz, which means you will only hear frequencies up to 1102.5 Hz.

To play the sound at a reduced sampling rate of 8820 Hz, for example, you could use the commands:

```
>> Fs_by_5 = Fs/5;
>> music_by_5  = sound(music_by_5,Fs_by_5);
```

What to report for Part B:

Describe your subjective observations for how the music is distorted at each subsampling.

Plot your signals in the time domain. You will need to create different time vectors corresponding to each sampling rate. For example,

```
>> time_by_5 = (0:length(music_by_5)-1)/Fs_by_5;
```

will create the time vector for this subsampling. Note that the total time for each version will be the same; there will just be fewer samples.

Include your time-domain plots in your report and describe any observations you make about how similar or different they look in the time domain.

Part B – Aliasing Experiments, Frequency-domain Observations

To make frequency domain observations, calculate the Fourier transform for the signals. We have not discussed the algorithm for doing this yet, but Matlab has an easy-to-use function, `fft`, for finding the Fast Fourier Transform. See the code `Lab1 example code.m` for examples of how to do this. This code shows how to make plots of the magnitude of the frequency spectrum in both linear and dB units.

What to report:
For your report, include your fft plots and describe your observations.

Next step:
You will also use a graphical Matlab tool to visualize the power spectrum: `signalAnalyzer`.
To start `signalAnalyzer`, in your Matlab workspace type:

```
>> signalAnalyzer
```

You should see a window pop up that looks like the one in Fig. 3. Notice that there is a pane on the left-hand side of the window that shows all the signals in your workspace. You can drag these signals into the chart window, as shown in Fig. 4 for the signal, ch1_part_sub15. You will see a time display as shown in the figure. Notice that there are no time units. The x-axis is the sample number. Getting the units right can be a bit tricky. Right click on the signal name in the upper left pane ("Filter Signals"). You should see a pull-down menu, with the top entry "Time Values …". Left click on this and pull-down on the "Time Specification" menu. Select "Sample Rate and Start Time." Leave the start time at 0 and enter the sample rate for this signal. (If you have the sample rate as a variable in your Matlab workspace, you can use the variable name.) After entering the sample rate, the x-axis will have units of seconds. Now, click on "Spectrum"

at the top of the signalAnalyzer window. The window with the graph will now have two sections, shown in Fig. 5. The lower window is the power spectrum. This is like the fft magnitude plots you have made, but it shows the magnitude squared using a method taking multiple time windows and smoothing. This creates a better representation of the distribution of power per frequency (power spectral density).

Look at the power spectra for each of your sub-sampled signals. You can display multiple signals and spectra by dragging the file name into the time window. It will complain that it cannot combine signals with and without time information. You will need to go through the same procedure as before to enter the sampling frequency of a new signal you drag in. Then check the box next to the signal to have it show up in the display. If you drag in a signal with a higher sampling rate, uncheck the other boxes so that it will show the full frequency range for the least subsampled signal. Then recheck the boxes next to the other signals. An example of subsampling rates of 10 and 15 is shown in Fig. 6.

What else to report for Part B:
For your report, discuss your observations of the power spectra for each sampling rate using signalAnalyzer.

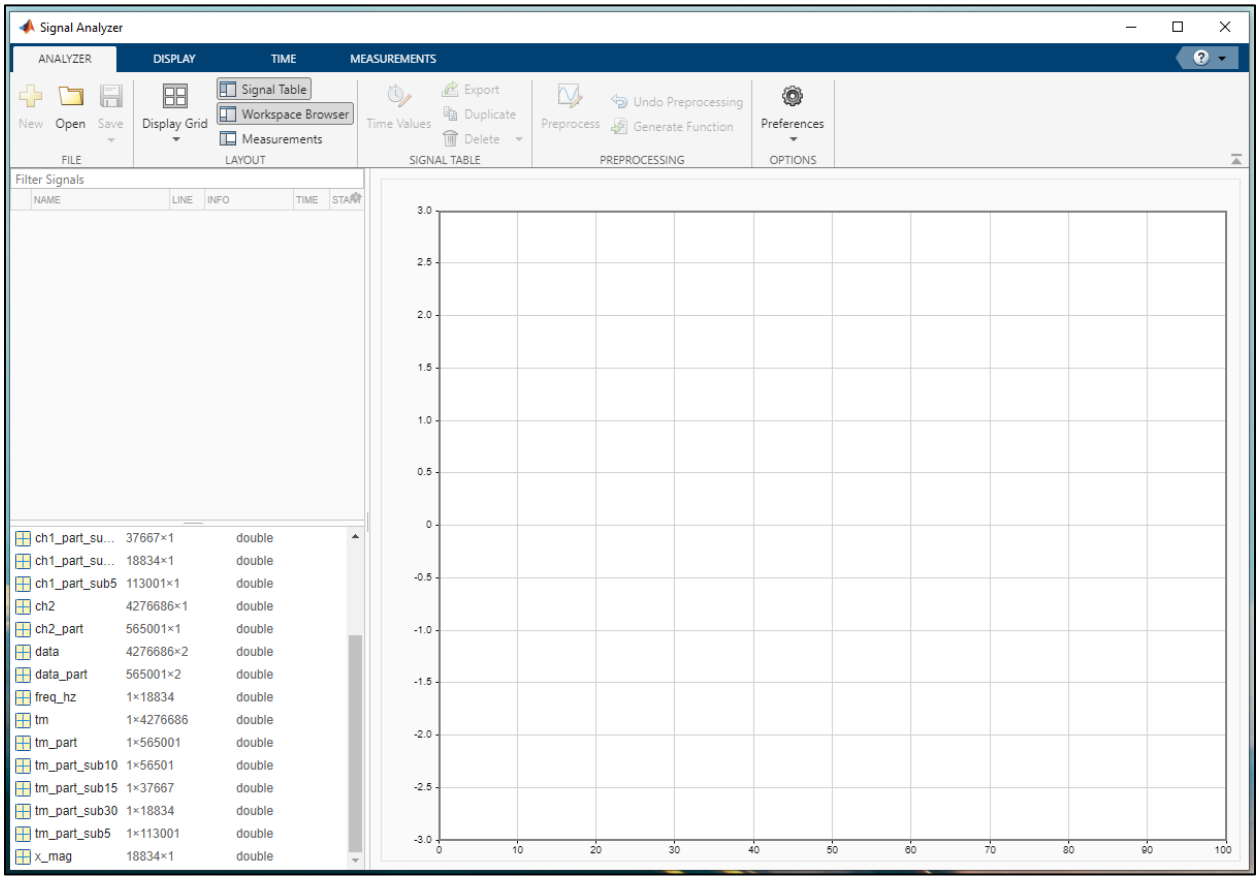

Figure 3: Initial window you will see when you start signalAnalyzer

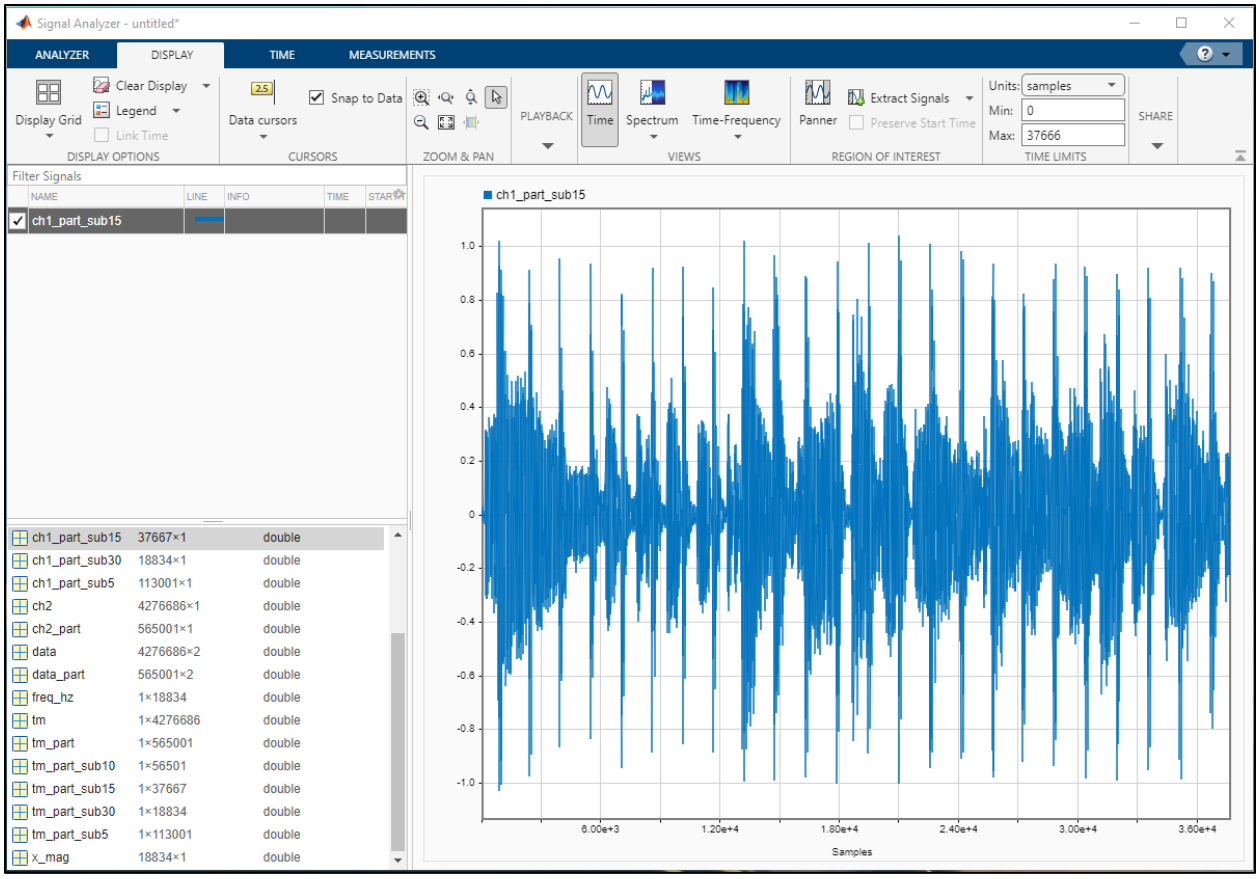

Figure 4: signalAnalyzer window after you drag a signal into the chart section.

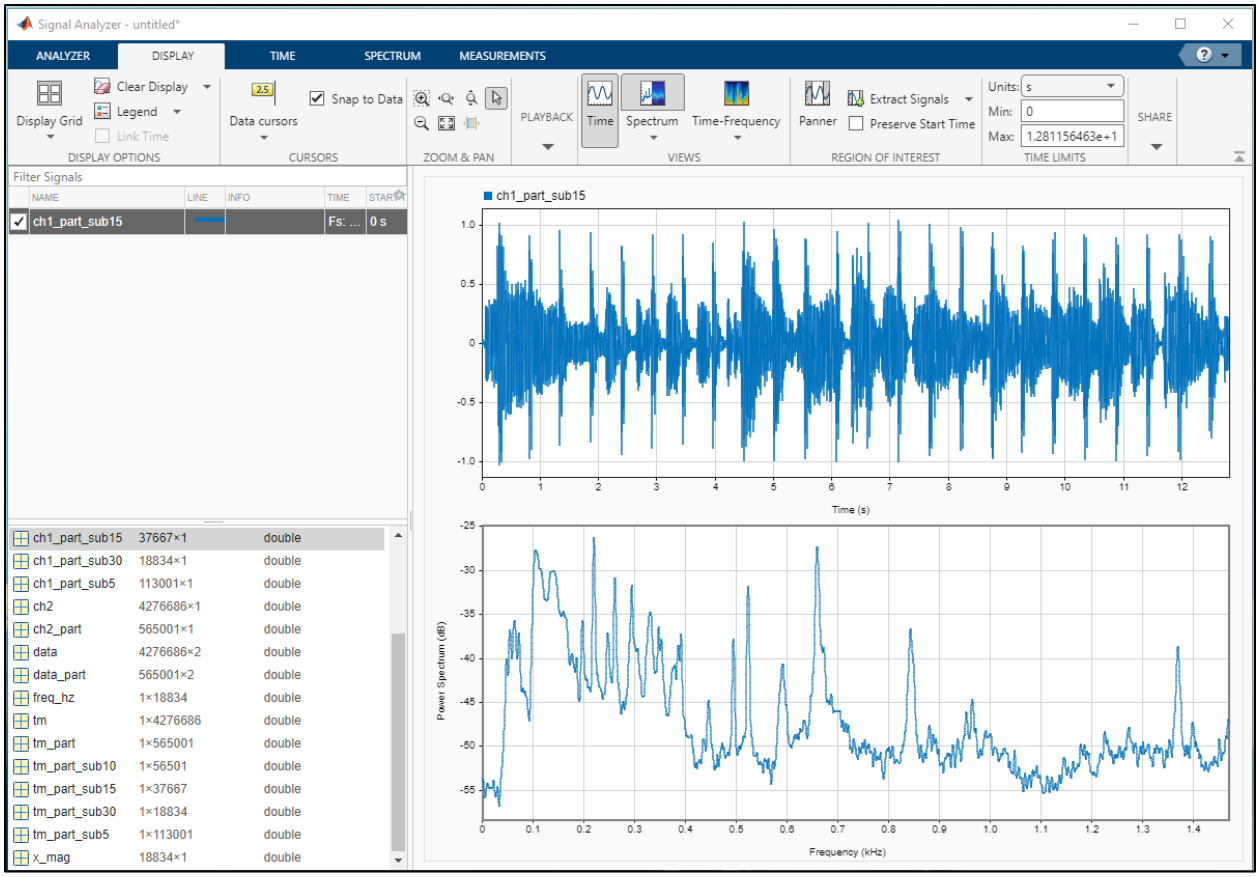

Figure 5: signalAnalyzer window showing both the time-domain signal and power spectrum.

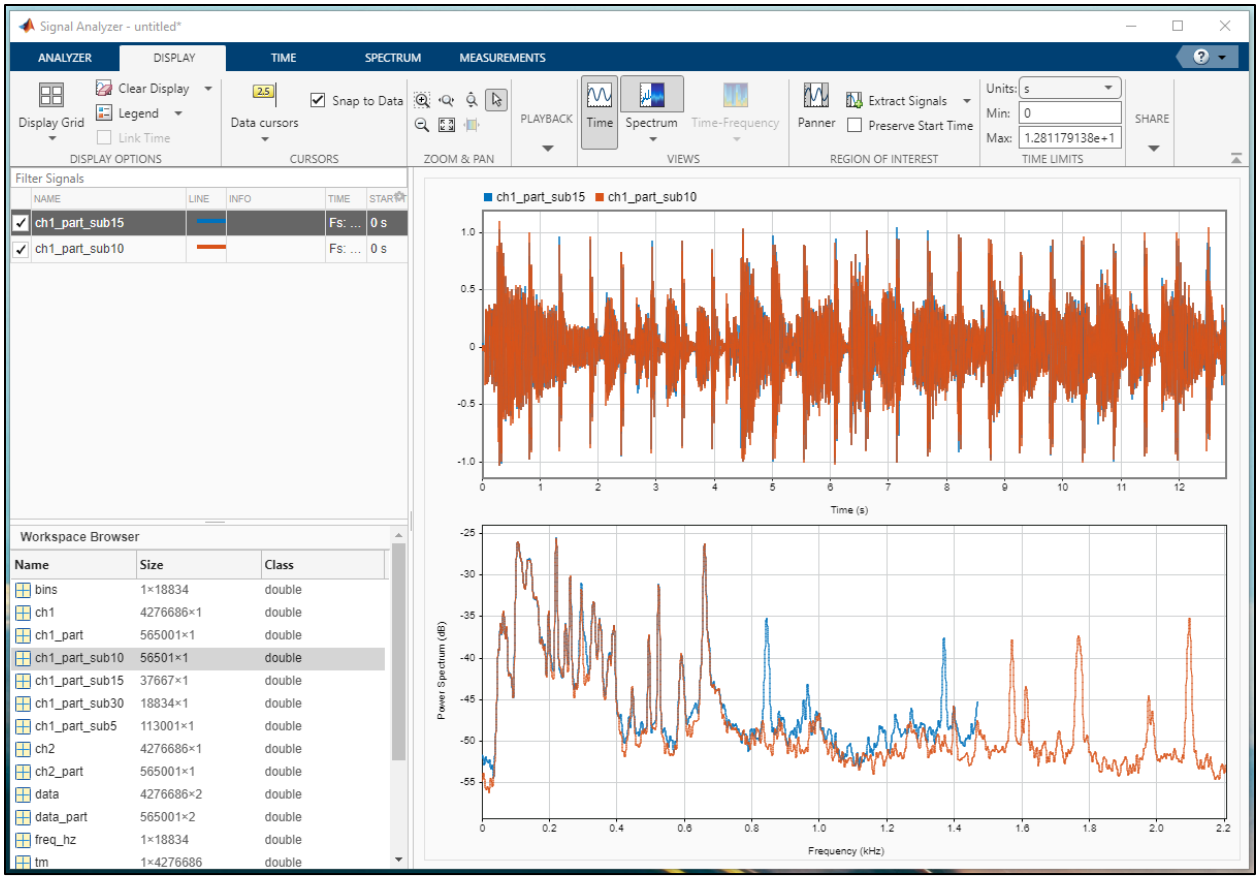

Figure 6: signalAnalyzer window showing two signals.

Part C – Anti Aliasing Filters

For this part, you are going to create filters to remove frequency components from your original signal prior to subsampling. You only need to do this for one of your subsampling experiments. Pick one where you can clearly hear the effect of aliasing in the music.

How to create a filter:

In Matlab, start `filterDesigner`:

```
>> filterDesigner
```

A window will pop up which should look like the one shown below in Fig. 7:

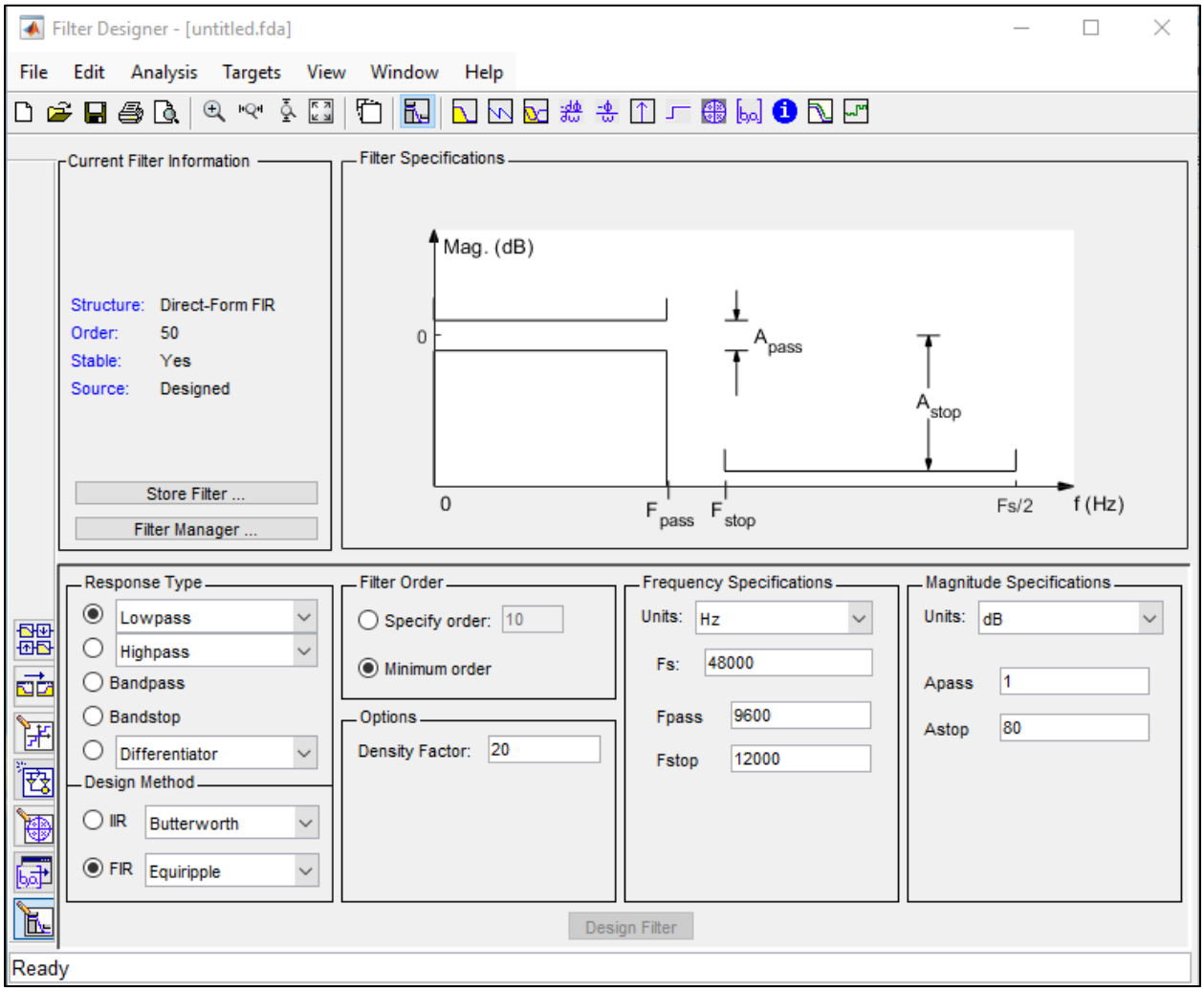

Figure 7: Window you will see when you start filterDesigner

We will discuss filtering and how to construct filters in detail, but `filterDesigner` is an intuitive tool that should make it easy to construct a filter for the purposes of this lab. The idea behind anti-aliasing filtering is to remove frequencies that would be aliased at too low of a sample rate.

Assuming that your original signal was sampled at 44.1 kHz, construct a filter to remove all frequencies that would be aliased in the subsampling you choose for this part. Suppose that the

subsampling is by 15. The sampling rate would then be Fs_by_15 = Fs/15 = 2.940 kHz. This means that any frequencies above 2.940/2 = 1.470 kHz will be aliased. To remove frequency components above 1.470 kHz in the original signal, you will need to apply a low-pass filter. Notice that, by default, in the `filterDesigner` window, "Lowpass" is selected for the Response Type and "FIR Equiripple" is selected for the Design Method. Leave these as they are. The part you need to modify is the Frequency Specifications section. Enter the frequency of your original signal in the Fs box (44100). You will want to pass frequencies up to the Nyquist frequency for your subsampling. Select a value a little less than this to enter in the Fpass box. (For subsampling by 15, 1350 Hz would be a good choice.) You cannot have too sharp of a cut-off or the filter will be difficult to implement and have very high order. For Fpass, in this case, 1550 will work. What this filter provides are the *a* and *b* coefficients in the difference equations we discussed in Class 6. (See Eqs. 1 and 2)

$$1 \cdot y(n) + \sum_{k=1}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k) \qquad (1)$$

For an FIR filter, *a* will be 1. The *b*'s to achieve this filter are what will be calculated.

$$1 \cdot y(n) = \sum_{k=0}^{M} b_k x(n-k) \qquad (2)$$

Below is a picture of filterDesigner with these selections.

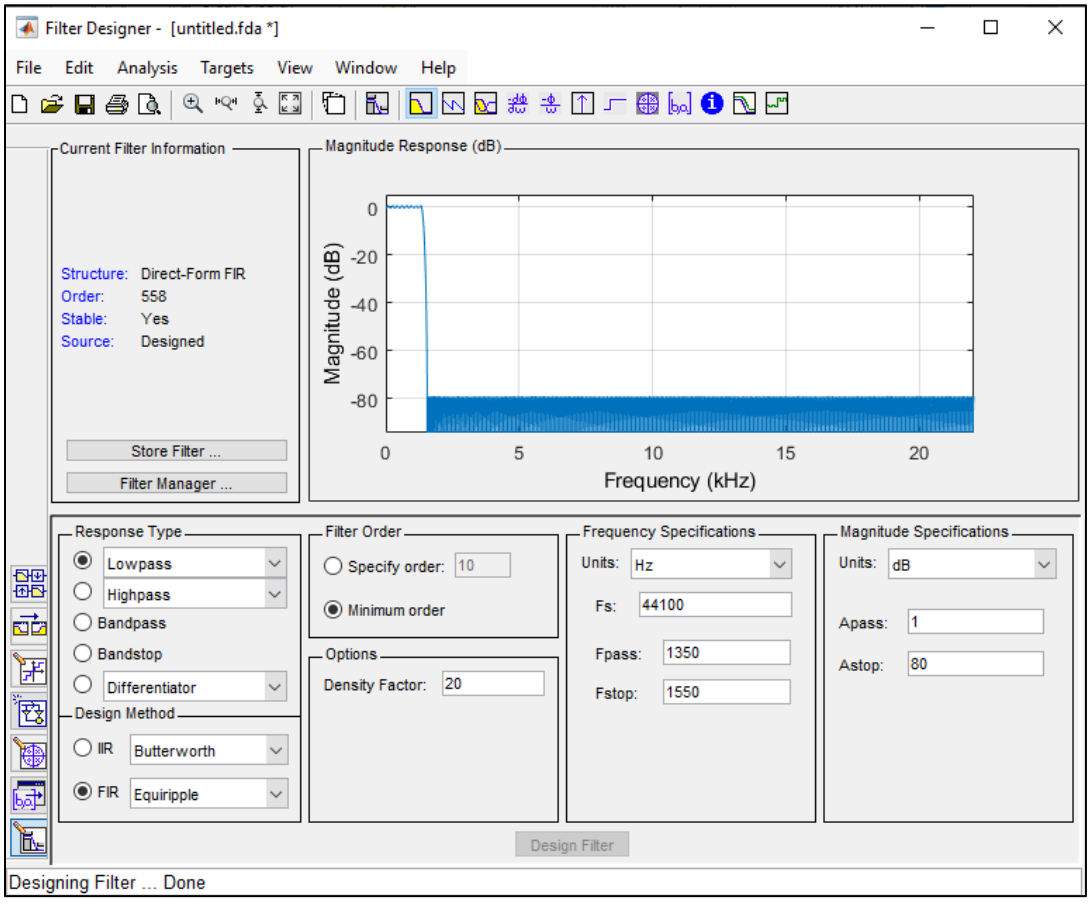

Figure 8: filterDesigner with sampling rate and pass and stop bands set.

You may want to expand the view around 1500 Hz, to see more detail for this filter. (See Fig. 9)

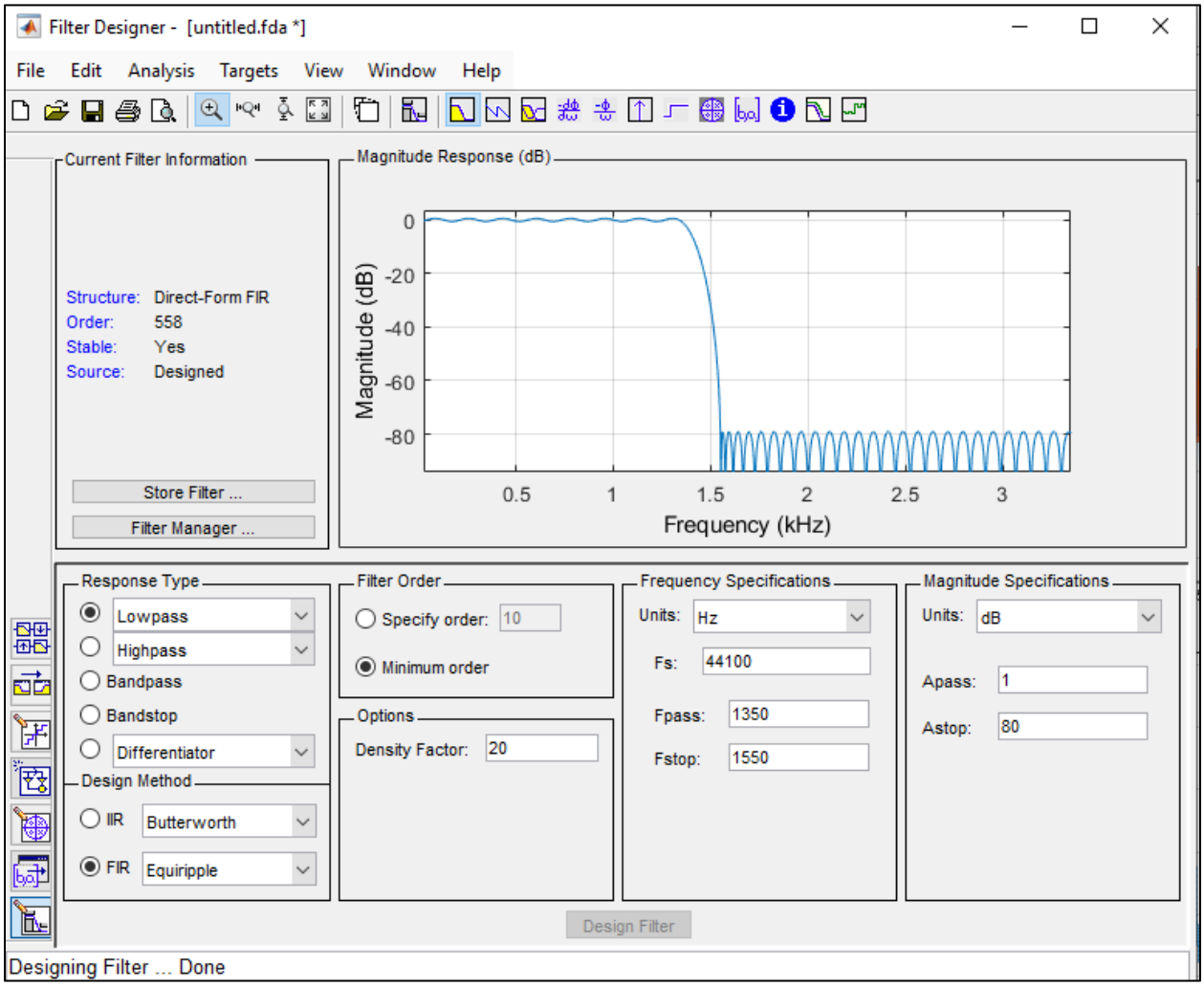

Figure 9: filterDesigner zoomed in around 1.5 kHz.

To use this filter on your original signal, you will have to export the *b* coefficients. Under the "File" tab in filterDesigner, choose the selection: "Export …". A window will pop up telling you what it is going to do, namely export to the Matlab workspace the coefficients (*b*'s) and place them in a variable named "Num". This is all fine. Click "Export". There should now be a variable named "Num" in your workspace. If you want, you can look at the numbers by typing Num at the command prompt. (The vector for the selections described above will have about 550 elements.)

Now, filter your original signal, using the Matlab command `filtfilt`. (This runs the filter forward and backwards to avoid introducing a time delay in the filtered signal.) The arguments for `filtfilt` are *b*, *a*, and the variable name for the signal to be filtered. Here is the command, assuming your original, 44.1 kHz signal is named `music`.

```
>> music_filt = filtfilt(Num,1,music);
```

The Matlab variable, `music_filt`, will be your original music (still sampled at 44.1 kHz), with no frequencies above 1500 Hz. (The Fstop of your filter.) Frequencies above 1350 Hz, will be attenuated as shown in your filterDesign window.

Play your original music:

```
>> sound(music,Fs);
```

Play your filtered music:

```
>> sound(music_filt,Fs);
```

Play your subsampled music:

```
>> sound(music_by_15,Fs_by15);
```

or if you didn't save the subsampled file, you can use the command:

```
>> sound(music(1:15:end),Fs/15);
```

Now, play the subsampled, filtered file:

```
sound(music_filt(1:15:end),Fs/15);
```

What to report for Part C:

Describe the differences you hear in the signals you play back.

Look at the power spectra of these signals in signalAnalyzer. Include these plots and explain your observations.

Part D – A Mystery (This is the fun part.)

Click on the following link: 2001_a_space_oddity.wav. This will hopefully download the file to your computer. If your computer is like mine, you can probably double-click on the file to play this WAV file. What does it sound like?

Read the file into Matlab with audioread:

```
>> [mystery_sound,Fs] = audioread('2001_a_space_oddity.wav');
```

Play this in Matlab with the command:

```
>> sound(mystery_sound,Fs);
```

What does it sound like? The same as before? Not much of a mystery … yet.

Now, play it back, listening only to every 5'th sample.

```
>> sound(mystery_sound(1:5:end),Fs/5);
```

What do you hear now? Remember, this is exactly the same file; you are just listening to every 5'th sample.

What to report for Part D:

What are your observations listening to the sound without and with subsampling by 5. Explain what is going on here. How was this done?

Report Requirements

A formal lab report is not necessary, but include all figures with some explanation of what they mean, and discuss the items shown above under, "**What to report …**"