Experiment 5: Multirate Signal Processing

Bruno Korst - bkf@comm.utoronto.ca

Abstract
In this experiment, you will use decimation and interpolation along with FIR filter design to efficiently manipulate different frequency bands within the spectrum of a given signal. You will achieve band splitting through the use of mirrored low pass and high pass filters that you will design, followed in sequence by decimating your signal. You will apply this approach sequentially, in order to target a specific band within the signal spectrum and will modify it with gain or attenuation. You will then recombine the outputs of all stages in order to produce a meaningful output.

Keywords
sampling rate — decimation — interpolation — quadrature mirror filters

Contents

Introduction

The purpose of this experiment is to introduce *multirate signal processing*. As the name implies, this methodology uses multiple *decimation* stages, combined with delays, filters and interpolation stages, to make available different sampling rates that will be used simultaneously in processing a signal. As you have by now seen in class and in other experiments, filtering a signal (Experiment 3) will single out a particular band within the full spectrum of that signal, whereas decimation (Experiment 1) will cause the spectrum of such signal to be moved, or shifted, to a lower portion of the spectrum in the frequency domain. Another very useful consequence of decimation is that by decreasing the sampling *frequency*, or rate, one increases the amount of time inbetween samples, allowing for more complex computation to take place within the constraints of a real-time system. That is to say, as processing routines have to be finished before other crucial interruptions take place, having more time to process will reduce the likelihood that data will be lost.

If one wishes to modify multiple narrow bands within the full bandwidth of a signal, multiple filters will be needed, mostly bandpass filters. These bandpass filters will likely need a somewhat sharp cuttoff, both in the lower and upper cutoff frequencies, which is to say that the quality factor – or Q – of the filters will be higher. One may also be interested in keeping the phase linear and ensuring stability of the response, which calls for the use of an FIR topology. This all means, after what was seen in Experiment 3, that higher order FIR filters are needed. For a time-domain implementation (that is, through convolution), the higher the order, the longer the time needed to produce an output. If these higher order filters are called sequentially, one might not be able to keep the real-time constraints of the design. Signal data is likely to be lost.

Let us consider that either the lower or the higher end of the spectrum of the signal above needs manipulation. If so, one could split the whole band into two equal parts with wider, lower order filters and repeat this process a number of times

(cascade) to close in on the band of interest. Furthermore, if the band of interest is in the lower end of the spectrum, one could then reduce the sampling rate *only of that lower band*, operate on it, and then use interpolation to recover the signal. The *decimation* will allow for you to effectively "gain time" to perform computation on the band of interest, and will allow for the use of lower order filters, since after decimation the spectrum is already "compressed" proportinally to the decimation factor used.

Through the use of multiple stages of low order filtering and decimation, one is able to single out individual bands within the bandwidth of the signal of interest. However, in order to produce an accurate output, one *must* recombine the signal that was split into these multiple bands and multiple sampling rates. The recombination must take into account the alignment of the samples arriving from the multiple paths. Interpolation is used to restore the sampling rates at every stage, and sums are used to recombine the signal paths and produce a meaningful output.

Not many undergraduate textbooks will cover multirate signal processing, but all will cover decimation, interpolation and filtering. For further, in depth reading on multirate, one can read [1]. This experiment will be divided into the following parts:

- First, you will create a simple signal path with a *decimator* and an *interpolator*;
- Second, you will design a low pass filter and a high pass filter that are mirrors of each other;
- Third, you will combine these two previous systems into a one-stage multirate topology;
- Finally, you will create a multiple stage topology, and use it to manipulate a particular band of interest of a given signal.

1. Experiment

1.1 Decimating and Interpolating

When you *decimate* or *downsample* a signal by a factor, you *reduce* the sampling rate of the signal by that factor. Moreover, you *compress* the spectrum of that signal by this same factor; if yor sampling rate was f_s, the new sampling rate will be f_s/n, where n is the decimation factor.

This is to say that effectively you will discard a number of samples when you decimate a sampled signal. As a numerical example, if you have a 2KHz sinusoidal signal sampled at 48KHz, you will have 24 samples per period to represent that signal digitally. When you decimate this signal by a factor of 2, you will discard every other sample and end up with 12 samples per period. This is equivalent to sampling the 2KHz signal at a 24KHz sampling rate.

You will now design a system that takes a sinusoid sampled at 48KHz, downsample it by a factor of 2 and then upsample it by the same factor. Your objective here is to see what is needed to recover the input signal after these operations are performed (and to convince yourself that this does work).

Open a new model on `Simulink`. It will have a DSP `Sine Wave` generator with a $1V_{pp}$ and 2KHz, using a 48KHz sampling rate, and 1 sample per frame. Add to your model a `Time Scope`, a `Gain`, a `Downsample` block and a `Upsample` block.

Your model should look like Figure 1 below.

Set the decimation factor to be 2 and select "allow multirate", and the input processing to be "sample based". [1] Set the interpolation factor to 2 as well.

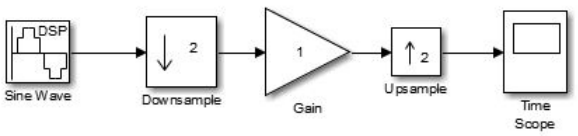


Figure 1. Suggested Downsampling-Gain-Upsampling Simulation

[1] If you select "force single rate", you are effectively selecting one sample and holding its value for the duration of the other (it's a factor of 2). This corresponds to applying a rectangular window [1 1] to a sample (i.e., hold one, discard the other), which will have the effect of a low pass filter in the frequency domain. You *do not want that*.

Run your model. You can set the simulation run time to `0.01` or `inf`. You may want to add an extra port to your `Time Scope`, in order to monitor your (unmodified) input signal. To do this, right-click on the `Time Scope` and select `Signals and Ports -- Number of Input Ports`. Clearly, the exercise here is to bring down the sampling rate by a certain factor, and bring it back up again to recover the signal perfectly. If one stage discards samples, the other stage pumps samples back in and *interpolates* them. It would be nice if the semester would end of this rather elementary exercise, but we have more to do. Answer some questions on the answer sheet.

1.2 Designing Mirrored Filters

At this late point in the term, the Engineer feels like a charged capacitor; compelled to apply knowledge accumulated in previous experiments. Perhaps this is the time to apply such knowledge. You know that an FIR filter has a few characteristics: it is stable (there is no feedback to cause trouble – no $H(z)$ denominator going to zero and blowing things up!), its phase is linear (audio lovers will thank you for that), and it is rather straight forward to implement in code by using a convolution. However, if you are looking for a sharp cutoff while keeping all of the constraints above, we are really looking at a higher order, and a longer computation time taken to produce outputs.

In the preparation for this experiment, you were required to describe how to design a mirrorred high pass filter from a given low pass filter. Your task now is to *design both* filters. These two filters will be used to expand the model you created in the previous section. The two filters that will accomplish the splitting of the full available band in two halves, one being the mirror of the other. You will use the `FDA Tool` to design your filters. The only parameters you need is f_s as 48KHz and the order, which is 20. To give you a reference, the frequency response of the filters should look like the ones presented in Figure 2 below. Note that the plot presented in the figure is *strictly* the output of a 1024 point FFT on the impulse response (coefficients) of the filter. The x axis shows only the point numbers from the FFT and the y axis represents magnitude values of the FFT on a linear scale. Design the filters now, and answer the questions on the answer sheet. Make sure to *export* the coefficients of the filters into two separate variables – say `h1` and `h2` – that will be available for a call from the variable workspace in `Matlab`.

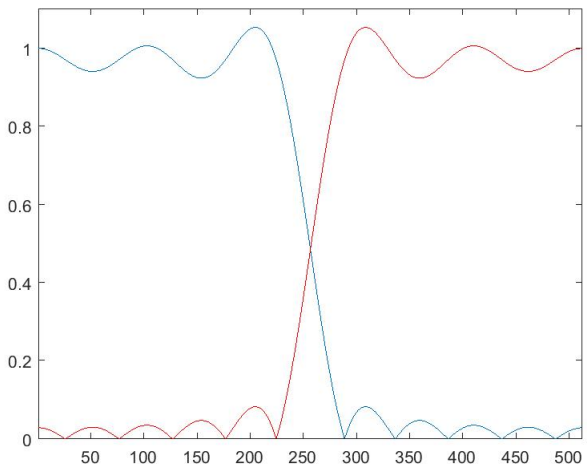


Figure 2. FFT of Low Pass and High Pass Filters

Hopefully you have gone the extra mile and wrote a function to implement the filtering routine, taking one sample as input, convolving with the filter(s) impulse response(s) and producing two outputs at the same time, from *only one* convolution function call. Did you use only one array to store the filters coefficients?

1.3 A One Stage Multirate System

You will now create a one-stage multirate system using the filters you have just designed above and the decimation-interpolation system you created on the first section. This is an exercise for you to add gain to half of the available band of your system. By now you will likely realized that you have only really designed *one* filter, and you are adding variations to where and how in the system you use it.

Use `Simulink` to create a system similar to the one presented in Figure 3 below. Note that the filter blocks shown are the `Discrete FIR Filter` block, found in the `Simulink` library. You will have to specify the coefficients appropriately, corresponding to the coefficient variables you created in the previous section of this experiment. Also, you will need a pure `Delay` block, found under the `DSP System Toolbox/Signal Operations` library.

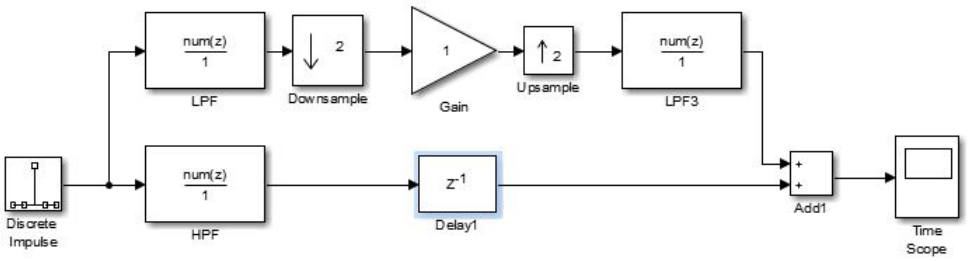


Figure 3. A One-Stage Multirate System

You will notice three features in the block diagram presented in Figure 3: a) it has a pure delay block on the signal path corresponding to the upper half of the spectrum; b) it is using a `Discrete Impulse` as the input to the system; and c) the only modification done on the lower half of the spectrum is a pure gain. Regarding the latter, it should be pointed out that for a real application, if the only modification required is a gain, then there would be no need to downsample prior to applying the gain. Likewise, there would be no need to upsample and filter the signal afterwards. It should be obvious by now that a gain is accomplished by a simple multiplication in code, which takes a single cycle of the processor to be accomplished. However, if the operation on the lower half of the spectrum is more complex, such as a frequency domain filtering (involving FFT/IFFT) or an adaptive filter, then the decimation will allow for more time for these more complex operations to be implemented. Afterwards, the interpolation-low pass filter will then be needed for the proper recombination of the signal.

For this part of the experiment, you will run two exercises: one with the discrete impulse to determine the gain and pure delay needed, and one with a `Band-Limited White Noise` generator as an input to your system. When you use white noise as an input, you should use the `Spectrum Analyzer` as your signal sink. It is found under the `DSP System Toolbox/Sinks`. Move to the answer sheet and answer the questions.

1.4 A Multi-stage Multirate System

Note that for the one-stage system above, you only split the full band in half *once*. That is, your sampling rate was 48KHz, the LPF selected from DC to the cutoff frequency at 12KHz and the HPF selected from the cutoff at 12KHz to the Nyquist limit at 24KHz. Your design challenge now is to implement a ”complex” operation (it will be gain, don’t worry) at the frequency band located between DC and 2KHz. Note that depending on your application, as you use these mirrored filters to split the full band sequentially, you may choose to operate on *any* of the intermediate bands. For this experiment, though, we will focus on the lowest portion, which is from DC to 2KHz.

Your first objective, then, is to find a combination of band splitting and downsampling to *get there*. You will place gains at the lowest portion of the split band. After you operate on the band you want, you must *recombine* your signal. Your multirate model should resemble Figure 4.

Note two things about the multirate system presented in Figure 4: first, each QMF stage is comprised of the two filters (HPF and LPF) and one of these two bands is downsampled by a factor of 2 (which one?). Second, the recombination of each band (otherwise called a ”synthesis” stage) is made of a low pass filter and an upsampling stage, and you may use yet again the same filter that you have designed before to be the low pass filter here.

You will also need to add delays appropriately on each of the signal paths that are *not the target band*, since they will be needed for the recombination (synthesis) and the generation of a meaningful output. Place gains at the signal paths so that you can manipulate the bands separately. You may use either the `Gain` block or the more convenient `Slider Gain`, which you can modify by dragging with the mouse while the system runs.

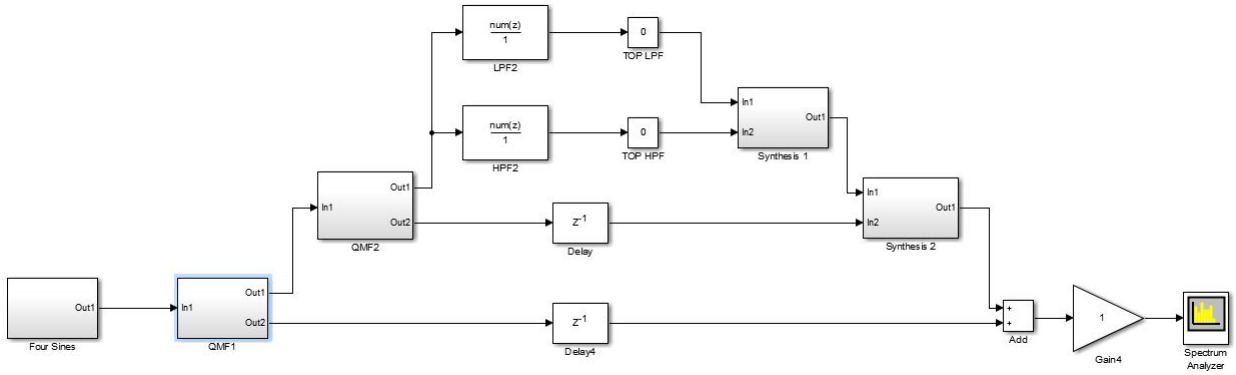
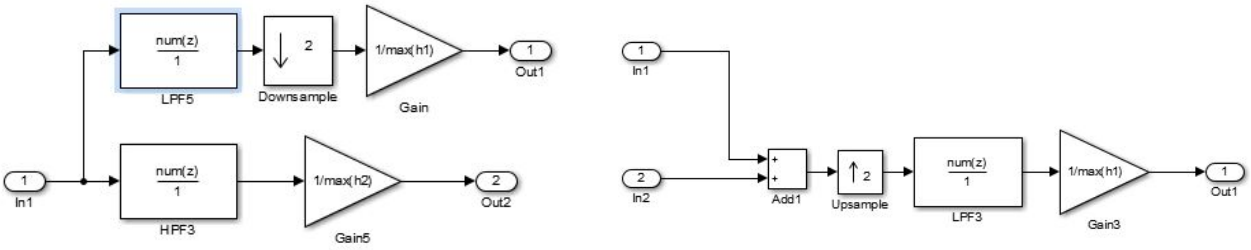


Figure 4. A Two-Stage Multirate System

In order for you to explore fully the operation of this multirate system, you should use the white noise generator used in the previous section as your input here as well. Moreover, The `Spectrum Analyzer` should be configured as a *spectrogram*, which will generate an excellent display as the system runs.

For reference, the figures below show further details of the QMF stage and the synthesis stage.



(a) QMF Stage

(b) Synthesis Stage

Figure 5. Analysis (QMF) and Synthesis Stages

Move to the answer sheet to work on answers related to this part of the experiment.

2. Accomplishments

In this experiment, you acquired a better understanding of multirate signal processing systems. You had the opportunity to see that through the use of different combinations of filtering, downsampling and upsampling you are able to operate on multiple sub bands of a given input signal.

You explored further downsampling and upsampling, you designed and utilized quadrature mirror filters, and you simulated a multiple stage multirate signal processing system, targeting the lowest sub-band of a given signal. You also used a running spectrogram to verify that the stages of your multirate system were working as expected.

The advantages of using multirate signal processing systems rest primarily on the fact that, by slowing down the sampling rate of a particular signal path, one would allow for more time to perform other complex computations that may be required on that path. In addition, one may notice that all filters used in this experiment stem from a single design. This means a very effective use of memory (to store only one set of coefficients) and of computation resources is accomplished, leaving room for more demanding tasks.

Acknowledgments

Thanks for all the students who have provided input on the previous versions of this experiment.

References

[1] P.P. Vaidyanathan. *Multirate systems and filter banks*. 1993.