DAT116 (Mixed-signal system design)
Lab 6: Oversampling and noise shaping

Lars Svensson
`larssv@chalmers.se`

Version 1.4, December 17, 2018

1 Introduction

In this lab session, we will study several aspects of oversampled converters, in particular those which employ noise shaping. We will only use MATLAB and Simulink. The instructions are rather terse, as most of the tasks should be familiar from previous labs in the series.

2 Preparation

Read sections 6.1 – 6.3 of Maloberti. Pay special attention to the SNR calculations of Section 6.3.

3 Oversampling

- Start MATLAB as in previous labs, set the path and current directory suitably, and launch Simulink.
- Set up a simulation of an A/D conversion with 2^7 levels, using the `Lookup Table` block and the `lutix` and `lutdata` convenience functions. Fit 5 cycles of a sine wave in 10 seconds and 1024 samples in total; run the simulation; and save the result to the MATLAB workspace (you will need it later). Use `sigspectrum` to check that the spectrum looks OK.

- Extract the power spectrum values from `sigspectrum` and calculate the SNR for the conversion. Does the value correspond to your expectations?

The sample rate in the simulation above defines the maximum baseband bandwidth that may be represented; the signal we actually converted is nowhere near the Nyquist limit of $f_S/2$. Next, we will *oversample* the signal by using a much higher rate, and compare and contrast the results for the two rates.

- Repeat the previous simulation, but increase the number of samples by a factor of 2^5. (Do not change the duration of the simulation!) Save the result under another variable name, and plot both spectra together. Save the plot for your report.
- Zoom in on the spectrum values at some frequency covered by both simulations. Observe that for every power component at the lower sample rate, there is a component at the same frequency at the higher sample rate.
- Calculate the SNR at the higher sample rate in the same way as before. Discuss the appearance of the combined spectrum plot and the SNR values, and compare with your expectations.

Provided that the sample rate in the first simulation was adequate for conversion of the baseband signal of interest, the added frequency slots in the spectrum of the oversampled signal contain no useful information, only noise. We may remove the noise at those frequencies with a linear low-pass filter. For the sake of simplicity, we will initially assume that a *perfect* filter is used; that is, we simply ignore values at higher frequencies.

- Recalculate the SNR value resulting from the oversampled simulation, but include only those power values which correspond to the frequencies represented in the non-oversampled case. Does the SNR improvement correspond to your expectations?

The simple SNR improvement formula assumes that the quantization noise is white (that is, its power density is independent of frequency), regardless of sample rate and input signal waveform. This is true only *on average*, across *all possible input signals*, but not necessarily for any given signal alone! The specific quantization noise spectrum seen here (in the `sigspectrum` plots) falls off slowly towards higher frequencies, and thus the ideal low-pass filter removes less noise than might be expected.

Save the outputs of your simulation for later use.

4 Single-pole noise shaping

In this section, we will use a single-pole noise-shaping feedback loop around the quantizer used in the previous section.

- Use the same 7-bit, `Lookup Table`-based quantizer as in the previous task, and the same sine wave as a signal source. Include a `Discrete-Time Integrator` block (from the **Discrete** collection) and connect its output to the input of the `Lookup Table`.
- The integrator input should be the *difference* of the source sine wave and the output of the `Lookup Table`. Use a `Sum` block and make sure to change the sign of the feedback input to ensure negative feedback.
- Be careful to set the parameters of the `Discrete-Time Integrator` correctly: the **Integrator Method** must be set to **Accumulation: Forward Euler**. Also set the parameter **Gain** to 1.
- Save the output from the lookup table to the MATLAB workspace with a `To Workspace` block, as above. Include another `To Workspace` block for the input signal.
- Use the higher sample rate from the previous experiment. Reduce the power of the input signal by 1 dB by adjusting the `Amplitude` parameter of the sine wave. Run the simulation and inspect the converter input and output signals with `sigview`. Verify that the output approximates the input.
- Zoom in close to one of the extreme values of the output. The quantized signal seems to mostly oscillate between two adjacent values, such that its *average* value tracks the input signal. Save the zoomed-in plot for your report.

Compared to the oversampled case without feedback (Section 3), we should expect a smaller error at frequencies close to the signal (since the average quantized value tracks the signal better), but more energy at high frequencies (due to the apparent oscillation).

- View the quantized-signal spectrum together with that of the oversampled case without feedback. Verify that the feedback loop has caused lower noise density at low frequencies, but higher density at high frequencies. Save the plot for your lab report.
- Calculate the SNR for the quantized signal, using the same "perfect-filter" assumption as above. Does the result correspond to your expectations?

- Make a new plot of the spectrum of the latest signal only. From this plot, estimate the noise density increase per decade of frequency. Does the slope correspond to your expectations?

5 Filters

In Section 4, the simple approximation of a perfect filter allowed us to study the properties of the noise shaping without regard to the filter properties. In practice, a real low-pass filter must be designed, which suppresses the high-frequency noise to a degree sufficient for the desired SNR gain. We will now vary the parameters given to a Simulink filter design tool to determine what is needed.

- Include in your simulation model an instance of the block `Digital Filter Design`, which is found next to the `Analog Filter Design` block used already in Lab 1. Connect the filter input to the output of the quantizer, and the filter output to yet another `To Workspace` block.
- Double-click the filter instance you just added. In the bottom half of the block control panel, select a **Lowpass** response type; an **IIR** design method with a **Butterworth** characteristic; and use the `Filter Order` 3.

The filter passband should cover all signals that fall inside the original baseband before oversampling. The cutoff frequency should therefore be set to the *inverse* of the oversampling ratio (so use a cutoff frequency of 0.5 for an OSR of 2, etc).

- In the **Frequency Specifications** pane of the `Digital Filter Design` control panel, set `Units` to `Normalized`, and enter the inverse of the oversampling ratio in the cutoff frequency specification field `wc`.
- Click `Design Filter`. Verify that the filter response shown in the top half of the block control panel changes to a low-pass characteristic.
- Close the block control panel. Use **File**$\rightarrow$**Save** to save your simulation model.

Simulation should now produce both the filtered and the unfiltered version of the quantized signal. As in Lab 1, the initial transient at the filter output must be avoided for the spectrum computations to come out right.

- Extend the runtime of the simulation by a factor less than 2, and run the simulation.

- Display the filtered and unfiltered waveforms with `sigview`. Do the waveforms correspond to expectation?
- Extract and display the spectra of the unfiltered and filtered signals. Do the spectra correspond to expectation?
- From the spectrum of the *filtered* signal, estimate the rate of noise power density *reduction* per decade of frequency beyond the cutoff. Does the slope agree with your expectations?

The overall SNR for the combination of quantizer and filter depends on the amount of noise removed: a steeper, higher-order filter will result in a better SNR. However, as steeper, higher-order filters are more expensive, it is useful to know what order is needed for an acceptable result, and when further filter-order increments yield diminishing returns.

- Use `sigspectrum` to generate the power spectrum values for the filtered signal. Calculate the overall SNR based on these values[1]. Compare the value with the result achieved with the "perfect" filter. How much worse is the SNR with a realistic filter than with the ideal filter?
- Repeat these simulations for filter orders from 1 to 7: For each filter order value, enter the value in the `Filter Order` field in the `Digital Filter Design` control panel; click `Design Filter` to update the transfer function; run the simulation; and compute the SNR for the filtered signal.
- Plot the SNR values as a function of the filter orders, and include the plot in your lab report. Discuss how to choose the filter order for a converter such as the one simulated here.

6 One-bit sigma-delta converter

In the final part of this lab session, we will study a *one-bit* A/D converter (a comparator, really), again with a first-order noise-shaping feedback loop around it.

- Change the parameters in the `Lookup Table` block to correspond to a one-bit converter. Re-run the simulation and verify that only two levels appear at the quantizer output.

[1]Note: do not ignore part of the spectrum as you did in the previous experiments!

- Re-run the simulation with a 3rd-order digital filter. Calculate the SNR as before[2]. Does the result correspond to your expectations?

The single-bit, first-order converter is susceptible to spurious tones and limit cycles at low-frequency, low-amplitude inputs. We will illustrate this deficiency with some deft parameter changes.

- Double-click on the `Sine Wave` block to open its parameter panel. Set the sinewave amplitude to 0 and close the block.
- Re-run the simulation and view the *unfiltered* output signal with `sigview`. Does the signal correspond to your expectations?
- View also the *filtered* output signal. Does the signal correspond to your expectations?

The negative feedback and the integrator in the loop will ensure that the average value of the quantized signal is the same as that of the input. A small DC value of the input will therefore have deterimental consequences.

- In the `Sine Wave` parameter panel, set the `Bias` value to 0.01. Re-run the simulation and view the *unfiltered* signal. What has changed with respect to the previous simulation? (You will need to zoom in.)
- View also the *filtered* signal. A "new" component has appeared here. What is the approximate frequency of this component? Why?
- View the spectrum of the filtered signal. Comment on the spectral content of the filtered signal compared to that of the DC input signal.

As you have seen, the highly non-linear nature of the sigma-delta loop may cause periodic signals, or "limit cycles", to appear even when no periodic signals are applied at the input. Limit cycle frequencies may depend on the DC offset.

- Make copies of the previous simulation result variables with different MATLAB variable names.
- Change the input bias level from 0.01 to 0.001 and re-run the simulation. View the filtered signals of both DC-bias simulations together in one `sigview` window. What has changed with respect to the previous simulation? Why?

[2]Be aware that at these low quantizer resolutions, the largest noise power component may before filtering be as large as your desired signal. Thus, `max()` may not return the signal power. Use the correct index explicitly instead.

- Plot the spectra of the filtered signals of both the simulations together in one diagram. What has changed with respect to the previous simulation? Why?

In many applications, it is highly undesirable that a small DC offset causes tones at the output. Fortunately, several remedies have been developed for this problem. First, since the behavior appears at small DC values, a larger DC offset may be applied, such that any limit cycle frequencies are high enough to fall outside of the band of interest. Second, a small amount of noise may be added to the input signal; the resulting irregularity may break up the repetitive cycles. Third, the problem is much less pronounced in converters with in-loop filters of higher order than 1.

7 Wrap-up

In this lab session, we have studied oversampling and noise shaping as a means to improving the SNR of a quantizer beyond the value given by the classic formula $6.02 \cdot N + 1.76$ dB. After completing the session, you are supposed to be able to do the following:

- Set up a Simulink simulation of a first-order noise-shaping converter.
- Select a low-pass filter to suppress the shaped quantization noise.
- Discuss the limit-cycle problems of first-order, low-resolution converters.

Reflection questions:

As described by Maloberti, the achievable SNR of a noise-shaping converter is decided not only by the number of quantization levels but also by the OSR and the feedback loop order.

- For a first-order converter, is it more useful to double the number of quantization levels or to double the OSR?
- For a first-order converter with a given quantizer resolution and a given signal bandwidth, what limits the achievable SNR?