DAT116 (Mixed-signal system design)
Lab 1: Sampling

Lars Svensson
lars.svensson@chalmers.se

DRAFT of Version 1.12, October 11, 2023

1 Introduction

In this lab session, we will study some effects of sampling. In conventional clocked digital systems, a time-continuous signal is represented by a sequence of instantaneous values taken at uniform intervals.

At several points, you will be asked for possible explanations for the behavior you observe. *Note: be sure to carefully consider these "why" issues,* and check your conclusions with your lab assistant. These insights are a significant part of what this lab attempts to achieve!

This PM is split into several sections (as are other PMs in the course). **Note:** The end of a section is a good point to check your results with the lab TA, before continuing to the next section.

2 Preparation

It is assumed that you are aquainted with the software, such as per Lab 0.

You need to be familiar with decibel calculations (they will be used frequently during the course!). If you need a brush-up, we suggest this Wikipedia link: `http://en.wikipedia.org/wiki/Decibel#Power_quantities` .

You need to be familiar with the asymptotic high-frequency behavior of low-pass filters, particularly Butterworth filters. A nice illustration can be found at `http://en.wikipedia.org/wiki/Butterworth_filter#Transfer_function` .

Go through this PM, locate the tasks marked **Preparation**, and follow the instructions.

3 Uniform sampling of single sine wave

We will start the first exercise where we left off in Lab 0: by examining the spectrum of a single sine wave. (Task descriptions are rather brief inasfar as they repeat steps taken in the previous lab session.)

- Launch MATLAB. Set the working directory to your designated lab directory and make sure MATLAB's search path includes the directory where the `.m` files are.
- Launch Simulink. Open a new simulation model. Save it as `lab1a`. Set the total simulation time to one second. Fix the simulation step to 1/512 second.
- Include a sine wave generator in the model. Set it to generate 7 cycles in the total simulation time. Set the amplitude to 0.1 and the bias to 0.
- Add a `To Workspace` block and connect it to save the output of the sine wave generator to the MATLAB workspace. (Don't forget to select the save format `Structure With Time`.) Select a name for the MATLAB variable and save the model. Uncheck the `Single simulation output` option in the **MODELING** tab, run the simulation, and use `sigview` to inspect the output. Verify that the signal corresponds to your expectations.

Preparation task 1.

- In general, the power of a signal is given by the average of the square of the instantaneous signal value[1]. What is the power of the sine signal produced in the simulation described above?
- Express this power in decibels referred to the power 1.0 .

---

[1] It may be helpful to imagine a voltage signal applied across a resistance of 1 Ω.

Next, we will study the spectrum of the generated signal.

- Apply `sigspectrum` to the signal you just generated. In the plot, one data point deviates significantly from the rest. What is the x-axis value of the deviating data point? Why?
- What is the y-axis value of the deviating data point? Why?
- What is the approximate y-axis value of the other data points? Why?

4 Multiple sine waves

We will now construct a signal from several sine waves, and study it in the same manner.

- Make three copies of the sinewave source in your simulation model, for a total of four sources. Change the frequencies in the three copies so that they generate 18, 46, and 65 full cycles in the total simulation time.
- Include a `Sum` block in your model (you will find it in one of the other libraries). Set the number of inputs for the block to 4 by editing the parameter "`List of signs`". Use the `Sum` block to add all signals and save the result to the MATLAB workspace as before.
- Run the simulation and display the output signal in the time and frequency domains. Do the signals correspond to your expectations? If not, why not?

The next few steps are intended to illustrate how the choice of sample frequency can influence the signal in time and frequency domain. In a fixed-step simulation such as those used here, the sample interval is most easily represented by the simulation step.

- Increase the sample interval by a factor of 2 and re-run the simulation. Inspect the saved signal in time and frequency domain. Do the plots correspond to expectations?
- Increase the sample interval by another factor of 2 (for a total factor of 4 over the original) and re-run the simulation. Inspect the saved signal in the frequency domain; pay particular attention to the x-axis coordinates of the deviating data points. (You may need to move the legend label to see all points.) What has happened?

- Again increase the sample interval by another factor of 2 (for a total factor of 8 over the original) and re-run the simulation. Inspect the saved signal in the frequency domain. What has happened?

- Open the parameter dialog of the 46-cycle sine wave generator. Change the `Phase` parameter for that signal to π. Re-run the simulation and inspect the output signal spectrum. What has changed compared with the previous simulation result? Why?

5 Anti-alias filtering

According to the Shannon form of the sampling theorem[2], a sequence of uniformly spaced samples uniquely represents a continuous signal if the latter has no components with frequencies larger than or equal to half the sample frequency. It should be clear from the experiments of the previous section that after sampling, input signals originating outside this limit may be *indistinguishable* from signals originating inside it. A usable representation requires that signal components originating outside the frequency band of interest are small enough to be insignificant. This requirement is typically filled through the use of *linear, frequency-selective pre-sampling filters*.

We will now include a block that simulates a time-continuous pre-sampling filter in the simulation model.

- In the Simulink library browser, open the `DSP System Toolbox`, open the `Filtering` folder, descend further into `Filter Implementations`, and then drag a copy of the block `Analog Filter Design` to your simulation model. Connect its input to the output of the `Sum` block. Include another `To Workspace` block and connect the filter output to the input of this new block. Enter a distinguishing MATLAB variable name, such as "`filtered`", in the new `To Workspace` block.

- Open the parameter dialog of the `Analog Filter Design` block and select a lowpass filter with Butterworth characteristic. Use a filter order of 3 and a passband edge frequency of 23 Hz (note that the block expects the *angular* frequency, ω, in rad/s; convert as appropriate). This filter should pass frequencies lower than 23 Hz and attenuate higher frequencies.

- Reset the simulation interval to its original value of 1/512. Run the simulation and plot the spectra of both the filtered and the unfiltered signals

[2] `http://en.wikipedia.org/wiki/Sampling_theorem`

in the same plot (see `sigspectrum` documentation for how to do this). Do the plots correspond to your expectations? If not, why not?

All linear filters have internal state variables, the values of which will initially be 0 unless special arrangements are made. Therefore, the filter output suffers from an initial transient which must not be included when the spectrum is calculated (the spectrum calculations used here rely on the signal being periodic). By design, `sigspectrum` ignores the initial part of a signal when the number of samples is not a power of two, so the initial transient may be avoided simply by extending the simulation time by a bit less than a factor of 2 (for example 1.9).

- Extend the simulation time, re-run the simulation, and inspect the results. Are the results in better accordance with your expectations?
- From the `sigspectrum` diagram, estimate the filter suppression of the 46-Hz signal in dB. Is this estimate in agreement with your expectations? Why / why not?

6 Signal-to-Noise Ratio

The `sigspectrum` function has the ability to return a vector of the spectral component powers. We will now use that ability to calculate signal-to-noise ratios (SNRs).

- Create a new Simulink model and save it as `lab1b`. Set the sample interval to 1/512, and the total simulation interval to 1.9 s, as above.
- Include a sine wave generator in **lab1b**. Set its frequency to 7 periods per second and its amplitude to 0.1, as above.
- Also include a `Random Number` block, that is, a white-noise generator. You will find the block in the **Sources** folder where you found the sinewave generator. The parameters include the variance of the number series, which is equivalent to the power of the noise source. Select it to have the same power as the sine wave. Also set the parameter `Sample time` to `-1` to indicate that the sample rate should be as for the rest of the simulation.
- Use a `Sum` block to add the sine wave and the noise, and a `To Workspace` block to transfer the resulting signal into MATLAB, as before. Remember to set the save format and to uncheck the `Single simulation output` option.

- Run the simulation. A diagnostic message will appear at the bottom of the window. Click on this message. A `Diagnostic Viewer` window will open to show a warning, together with instructions about how to turn it off if desired. Do that.
- Examine the results using `sigview` and `sigspectrum`. The `sigspectrum` plot should show the power of the sine wave being clearly higher than the other components. Verify that its level is what you expect.

To extract the power value vector from `sigspectrum`, you may use it on the right-hand-side of an assignment.

- Re-run `sigspectrum` and extract the power value vector into a variable, like so:

  ```
  >> arf = sigspectrum( ... argument list ...) ;
  ```

  Inspect the vector of power values returned.
- Calculate the total power of the spectrum (by summing all the vector elements). Does the total power agree with what you expect (given the parameters you set for the sources)?
- Identify the index of the vector element which contains the power at the sinewave frequency. Calculate the ratio of the signal power to the total power at all *other* frequencies. This is the Signal to Noise Ratio (SNR). Express the SNR in decibels.

We will now investigate how a linear filter can help improve the SNR.

- Add a third-order Butterworth lowpass filter to **lab1b** and save both the filtered and the unfiltered signals, as you did previously with **lab1a**; use the same cutoff frequency. Run the simulation and inspect the results in time and frequency domains. Does the suppression of the high-frequency noise conform to expectations for a third-order filter?

Recall that the spectral analysis relies on the signal being periodic. In the noiseless case in section 5, extension of the simulation time took care of the initial transient of the filter, by ensuring that the input signal before the interval analyzed was identical to that at the end of the interval, and that the filter was therefore in a steady state during the interval. Here, however, the noise signal at

the end of the interval is *not* identical to the noise just *before* the interval (the values are random!). This effect affects noise suppression at high frequencies. We may reduce this artifact by further extending the simulation interval, thereby making the artifact less significant compared to the entire simulation.

- Change the duration of the simulation to 65 seconds; `sigspectrum` will then analyze the last 64 seconds, or 32768 samples. Re-run the simulation, and use `sigspectrum` to plot the spectrum. Verify that the third-order noise suppression extends to higher frequencies.
- Has the power of the sinewave signal changed from the previous (shorter) simulation to this case? Why / why not?
- Has the apparent level of the "noise floor" changed from the previous (shorter) simulation to this case? Why / why not?

The filter has removed some of the noise from the signal; we should expect to find an improvement in the signal-to-noise ratio.

- Calculate the SNR values, in dB, for both the filtered and the unfiltered signals using the longer simulation duration. What percentage of the noise power was removed by the filter? (Compare with the ratio of the filter passband edge frequency and the maximum frequency of the simulation.)
- Reduce the simulation duration back to 1.9 seconds. Re-run the simulation and re-calculate the SNR improvement as above. Consider the difference with respect to the previous values and discuss whether the 34-fold increase in simulation effort is worthwhile in this case.

7 Wrap-up

In this lab session, we have studied the spectra of sampled signals, and illustrated the use of anti-alias filters to improve the SNR of the sampled signal. After completing this lab session, you are supposed to be able to do the following:

- Identify sinewave signals in a discrete-time spectrum, given their frequency and the sample rate, with folding taken into account.
- Estimate the noise suppression possible with a simple low-pass filter over the non-filtered white-noise case, and express the suppression in dB.
- Discuss how the FFT length affects the apparent spectrum noise floor.

Reflection questions

- If the sample rate in a data acquisition system is much higher than what Nyquist would require for the signal band of interest, some linear filtering can be applied *after* sampling (in the digital domain) to suppress any spurious signals outside the band of interest. Discuss how to select the bandwidth of the analog pre-sampling filter (as considered in the lab above) in such a case.

- In order to avoid discontinuities in our analyzed signals, we have been careful to avoid the initial transient in our simulations. Another way to avoid discontinuities would be to use *signal windowing* instead, as discussed in the texts of the theme reading directions.

  Compare and contrast these alternatives, *with respect to their pedagogical qualities*. Which alternative would best illustrate the issues addressed in this lab? Which one would be more complicated to describe in lab PM and report? Discuss.