# Laboratory Assignment 1
# Sampling Phenomena

2.171 Analysis and Design of Digital Control Systems

## 1 Main Topics

- Signal Acquisition
- Audio Processing
- Aliasing, Anti-Aliasing Filters
- Digital Filter Design and Implementation

## 2 Equipment

- Personal Computer with DS1102 (dSPACE, Inc.) Controller Card
- Matlab, Simulink, Real-Time Workshop, Real-Time Interface, and Control Desk Software
- Digital Storage Oscilloscope
- Breadboarding System with Analog Electronics
- Earphones
- Function Generator
- Music Source

## 3 Introduction

This set of exercises familiarizes you with the laboratory hardware and shows you how to use the hardware to perform real-world tasks such as acquiring signals and processing them. We also ask you to look at the non-idealities of these operations. You will look at the effects of sampling on signals from the generator and on music. You will also study quantization. Finally, you will implement first-, second-, and higher-order discrete-time filters. These filters will be designed from the viewpoint of approximating continuous-time analog filters and directly as digital filters.

## 4 Aliasing and Smoothing

### 4.1 Background and Equipment

In this lab, you will use sine waves and music as a source for signal processing. For the music source, you can provide your own, or play music out from the computer at each station. When processing a signal you will want to use nearly the full range of the A/D converters, so as to utilize the maximum number of quantization levels. The signal from the most music sources is at the 100 mV level, which is much lower than the full-scale $\pm 10$ V range of the A/D's. Thus you need to use an amplifier to increase this signal to the level of volts. We have designed an appropriate interface circuit for you as shown in Figure 1.

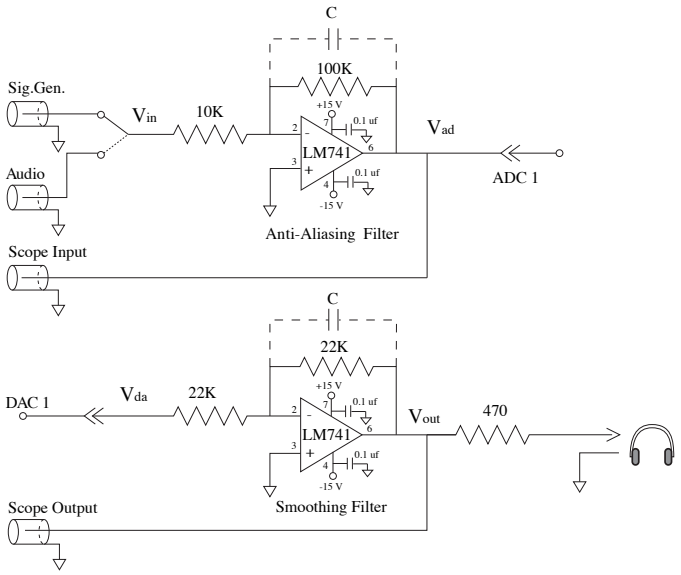


Figure 1: Anti-aliasing and smoothing filter circuit diagram.

We also want to be able to listen to the audio signals. Headphones have been provided at each station for this purpose. Since the DAC cannot supply sufficient current to drive these headphones directly, we have built an interface amplifier which can supply the headphone current. This amplifier is also shown in Figure 1. Both of these circuits have been pre-assembled for you on the electronic protoboard.

The lab explores the use of *anti-aliasing* and *smoothing* filters. The anti-alias filter attempts to remove high frequencies at the input before they reach the A/D. This is implemented by the input amplifier. The smoothing filter attempts to remove the high-frequencies associated with the staircase output from the D/A. This is implemented in the output amplifier which drives the headphones.

The anti-aliasing filter has a transfer function of

$$\frac{V_{ad}(s)}{V_{in}(s)} = \frac{-R_f}{R_1} \cdot \frac{1}{R_f C s + 1} \tag{1}$$

$$= \frac{-10}{10^5 C + 1}. \tag{2}$$

This is a first-order filter with a low-frequency gain of -10, and a -3 dB frequency set by the capacitor value.

The transfer function for the smoothing filter is similar to that of the anti-aliasing filter. It has a low-frequency gain of 1, and the capacitor value sets the cutoff frequency.

## 4.2 Experimental Procedure

**a)** Use Simulink and `rtilib` to create a model like that shown in Figure 2. Set the sampling rate by selecting `Simulation:Parameters` from the `simpleio` menu bar. Set the options to a fixed-time-step solver with a step size of `0.0001`. Close the window, save the file again, and select `Tools:RTW Build`. This step converts the block diagram into C-code, compiles this code, and downloads the executable to the DSP. At the end of this step, the DS1102 should be reading from the A/D and writing to the D/A at 100 $\mu$s intervals.

**b)** We will now examine the effect of *aliasing*. Aliasing occurs whenever a signal is sampled at a rate less than twice the maximum frequency contained in the signal. Aliasing is easily demonstrated

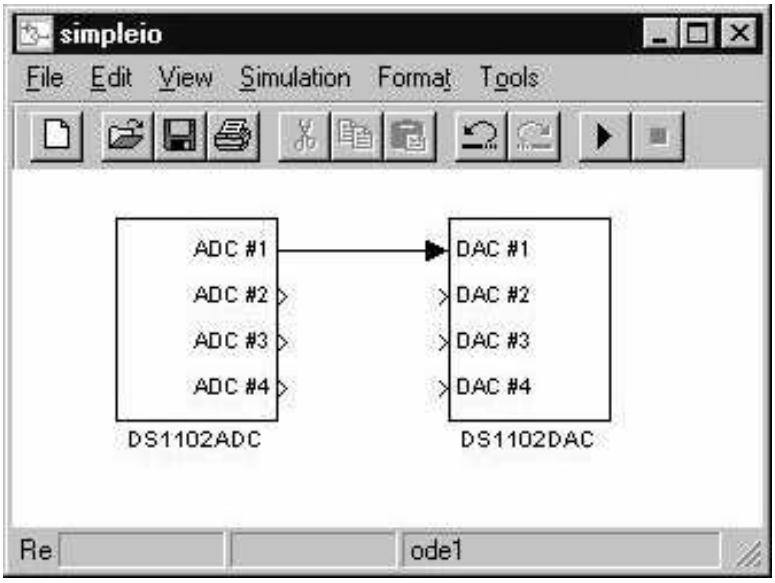


Figure 2: Simulink block diagram `simpleio.mdl` used to demonstrate aliasing effects.

using a sinusoidal signal source and sampling at less than twice its frequency. More complex effects occur when we sample a music signal. Pass a sinusoidal waveform from the signal generator through the anti-aliasing filter, and into A/D Channel 1. Initially run with the capacitors switched out, and thus no filtering. What effects do you observe in the output waveform? Try varying the input frequency and look at the frequency content of the output waveform. You may want to try changing the sample rate and recompiling to examine that effect. You should clearly see aliasing when the input frequency is higher than half the sampling rate.

**c)** Switch in the anti-aliasing and/or smoothing filters. Investigate the effect that these have. We want you to both observe the waveforms on the scope and to listen to the outputs and inputs on the headphones. Try to correlate what you see on the scope with what you hear on the headphones. Notice the beating phenomenon which occurs with the input frequency near half the sampling frequency.

**d)** Sinusoids are a simple waveform. Music is much more complex, but allows us to see/hear many effects simultaneously. Switch the input of your system to the music signal, and repeat the experiments above to understand the effects of sampling on the music signal. The music enters the system through the first BNC connector on your board, and the function generator enters through the second BNC. Move the wire to switch the input between them. Note the serious distortions introduced by aliasing when sampling audio signals at low sample rates. Look at the traces on the oscilloscope and listen to the music on the earphones as you lower the sampling rate; what effects do you see and hear? This gives you an appreciation for the effect of aliasing on real-world signals. What are the effects of the anti-alias and smoothing filters?

# 5 Quantization

## 5.1 Background and Equipment

Every digital system discretizes an analog signal in both time and amplitude. Only fixed levels (quanta) are permitted in a digital system. The DSP board used in this course has 16-bit A/D's on channels 1 and 2, and 12-bit A/D's on channels 3 and 4. All four of the D/A channels have 12-bit resolution.

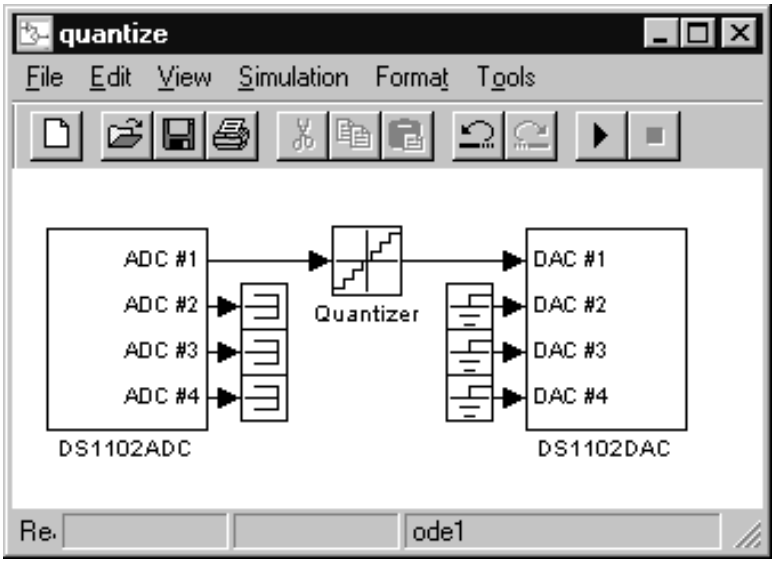


Figure 3: Simulink block diagram `quantize.mdl` used to demonstrate aliasing effects.

For the 12-bit channels, the operating voltage range of $\pm 10$ volts is quantized into 4096 different levels, which corresponds to 4.88 mV per quanta. For the two 16-bit A/D's the signals are quantized into 65536 levels, which corresponds to 0.305 mV per quanta. The software interface to the DSP board also takes the convention that the analog range of $\pm 10$ V is mapped to a numerical range of $\pm 1$ in the block diagram. **Be sure to note this scaling of 10:1!**

### 5.2 Experimental Procedure

**e)** Create a model with quantization as shown in Figure 3. The quantization block is located in the `Simulink:Nonlinear` library, and using it allows us to effectively increase the quanta size (thus decreasing the resolution). To quantize to $2^N$ levels, double-click the quantization block and set the level to $1/2^{N-1}$. Note that this value is chosen in light of the internal full range of $\pm 1$ corresponding to analog signals of $\pm 10$ V. As a starting point, let $N = 4$. Set the sample time (in `Simulation:Parameters`) to 0.0001 s with a fixed-time solver. Save the file as `quantize.mdl`, and download with `Tools:RTW Build`.

**f)** Choose the quantization bits $N$ at several settings ranging from 2 to 12, recompiling after each change. The resulting distortion due to quantization when using fewer bits can be appreciated by listening to and looking at the quantized and analog signals. Remember that you have to keep the magnitude of the analog signal large enough to be able to see all the levels, but not so large as to be entering saturation. Use the function generator as the source initially and perform the experiment with sinusoidal signals. Observe the effect of increasing and decreasing the amplitude of the sinusoid. You should able to count off 4 levels for a 2 bit quantization level and 16 levels for a 4 bit quantization level on the oscilloscope. Try the same experiment with the music source instead of the waveform generator. Note the deterioration of sound quality as fewer bits are used.

## 6 Discrete-time filtering

### 6.1 First-order discrete-time filter

Discrete-time filters are described by difference equations in much the same way that continuous-time filters are represented by differential equations. The discrete-time equivalent of a continuous-time filter

can be obtained by various approximations to the derivatives. A common approximation is by the forward Euler rule. Let $x_a(t)$ be an analog signal and its value at the $k^{\text{th}}$ sampling interval be given by $x(k) = x_a(kT)$ where $T$ is the sampling period. The Euler rule then approximates the derivative as

$$\begin{aligned} \dot{x_a}(kT) &\approx \frac{x_a((k+1)T) - x_a(kT)}{T} \\ \Rightarrow \dot{x}(k) &= \frac{x(k+1) - x(k)}{T} \end{aligned} \tag{3}$$

$x_a(t)$ at $t = kT$.

Let us now apply the above rule to a first-order unity-gain continuous-time filter given by the transfer function

$$\frac{Y(s)}{U(s)} = \frac{b}{s+b} \tag{4}$$

The corresponding differential equation is then given by

$$\begin{aligned} sY(s) + bY(s) &= bU(s) \\ \Rightarrow \dot{y}_a(t) + by_a(t) &= bu_a(t) \end{aligned} \tag{5}$$

As before, define $y(k) = y_a(kT)$ and $u(k) = u_a(kT)$. Then the difference equation which approximates this via the forward difference is given by

$$\begin{aligned} \frac{y(k+1) - y(k)}{T} + by(k) &= bu(k) \\ \Rightarrow y(k+1) - y(k) &= bTu(k) - bTy(k) \\ \Rightarrow y(k+1) &= bTu(k) + (1 - bT)y(k) \end{aligned} \tag{6}$$

Equation 6 gives the difference equation in a form suitable for implementation in a computer program. That is, we can calculate the present value of $y$ with knowledge of its previous value and of the previous value of the input $u$.

We can also specify this discrete-time filter via its transfer function. Introducing the z-transform variable $z$ to represent a unit forward shift in discrete time, we can rewrite the difference equation above in transfer function form as

$$H(z) = \frac{bT}{z - (1 - bT)} \tag{7}$$

where $Y(z) = H(z)\,U(z)$.

The above system can be implemented in Simulink by using a discrete-time transfer function block with `num` $= [0\ bT]$ and `den` $= [1\ -(1-bT)]$. In discrete system representation, it is good practice to enter the numerator and denominator polynomials as equal length vectors, padded with zeros as needed. This avoids confusions which can occur due to the use of $z$ or $z^{-1}$ in transfer function polynomials. For instance, in the signal processing literature, it is common to use $H(z^{-1})$, whereas in the controls area it is more standard to use $H(z)$. We recommend $H(z)$, but with the zero padding mentioned above, both forms are equivalent.

**h)** Build and run a filter which implements this discrete-time transfer function.

**i)** Examine the effect of the digital filter on sinusoidal input signals from the function generator. Initially, switch out the analog filter capacitors so that we do not have any anti-aliasing or smoothing effects. Initially, set $T = 0.0001$ and $b = 1000$, and thus the resulting first-order digital filter should approximate a continuous-time filter with a time contant of 1 ms. Using the scope, look at a sinusoidal waveform before and after the digital filter. Does it have the response you expect? Look at the response for inputs at frequencies higher than the Nyquist (i.e. above 5000 Hz). You should see the amplitude of the output grow again. Can you use aliasing to explain this effect?

**j)** Change the output of the function generator to a square wave, and adjust the scope to display the step response of the digital filter. While keeping the sampling rate fixed, change the value of $b$ and examine the step response of the digital filter. At low values of $b$, the response should approximate a continuous system. Examine the shape of the response as you raise $b$. At what value of $b$ does the response depart significantly from the continuous-time response? At what value of $b$ does the response become unstable? Note the z-plane pole location for each value of $b$ so that you can develop an intuition for the z-plane. Be sure you understand the response characteristics in light of the z-plane pole location.

### 6.2 Second-order discrete-time filter

We will now create a discrete-time approximation to a second-order continuous-time system. A unity-gain second-order continuous-time filter can be described by the transfer function

$$\frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \tag{8}$$

We will next apply the forward Euler rule to derive a corresponding transfer function,

$$H(z) = \frac{\omega_n^2 T^2}{z^2 + (2\zeta\omega_n T - 2)\, z + (T^2\omega_n^2 - 2\zeta\omega_n T + 1)} \tag{9}$$

Please check this derivation to be sure that you understand it before proceeding further. The experimental procedure for this section of the lab is described as follows.

**k)** Create a model which implements this filter with a 10 kHz sampling rate.

**l)** Look at the filter output when using the function generator for a sinusoidal input. Does the frequency response match your expectations? At frequencies past the Nyquist, can you see the increasing amplitude (as in the first-order system)? Change the function generator to produce a square wave output, and look at the step response on the scope. How well does the response correspond to the modeled continuous-time system? Look at different values of $\zeta$ and $\omega_n$. How does the response correspond with the location of the discrete-time poles? Determine a relationship between $\zeta$, $\omega_n$, and $T$ for stability. Compare this with the mapping viewpoint of the approximate filter design. How does the approximation of the second order filter hold up for large $\omega_n$ or small values of $\zeta$ ?

### 6.3 Other discrete-time filters

Experiment with additional discrete-time transfer functions in order to understand the effect of pole and zero locations in the z-plane.

**m)** Take a look at the effect of pole and zero locations in the transfer function

$$H(z) = \frac{b_0 z^2 + b_1 z + b_2}{a_0 z^2 + a_1 z + a_2}. \tag{10}$$

**n)** Design some interesting finite impulse response (FIR) filters, and implement these on the real-time system. Compare the step and frequency responses with that predicted in Matlab, and explain these responses in terms of the pole and zero locations. A separate handout on FIR filters will be available shortly.