W. E. Dunn

ME 360: FUNDAMENTALS OF SIGNAL PROCESSING, INSTRUMENTATION AND CONTROL

Laboratory No. 2
Signal Conditioning and Analog-to-digital Signal Conversion Issues

1. CREDITS

Originated: N. R. Miller, July 1997
Last Updated: S.Y.Olmez and D. Block, August 2020

2. OBJECTIVES

(a) Become familiar with the PC-based signal conversion hardware and software available in the laboratory.
(b) Study the problems of aliasing and quantization error associated with the digital representation of analog signals.

3. KEY CONCEPTS

(a) Information is irreversibly lost when an analog signal is converted to digital form. The loss is minimized by sampling at a high rate and using a high-resolution digital representation.
(b) A digitized sine wave appears to be a sine wave at a lower frequency if the sampling rate is less than twice the frequency. This effect, known as aliasing, distorts the frequency spectrum of any signal by mapping higher frequency components into lower frequency components.

4. SYNOPSIS OF PROCEDURE

(a) Observe the effect of aliasing using the function generator, oscilloscope, audio earphones, PC-based data acquisition hardware, and MATLAB software.
(b) Observe quantization error in a ramp function using the PC-based digital-to-analog conversion simulation tools in MATLAB/ Simulink.

5. MATLAB PACKAGES

This semester, you will have no access to the physical lab. This lab is modified to get you familiar with aliasing and quantization by simulating these phenomena on Simulink. **Before you start this lab, please make sure that you installed DSP System Toolbox add-on.** You can do this by clicking on Home->Add-Ons button on the top bar menu and searching for DSP System Toolbox.

5. PROCEDURE

5.1 Sine-wave Reconstruction Using Sampled Data

Overview of Procedure

The purpose of this part of the experiment is to illustrate the effect of aliasing. Aliasing occurs when the sampling rate of a periodic signal is less than twice the frequency of the signal. For a 1000-Hz sine wave, aliasing occurs if the sampling rate is less than $2 \times (1000\ \text{Hz}) = 2000$ samples per second. Aliasing is seen in the reconstructed waveform, which appears to be sine wave at a lower frequency. For example, consider an 1100-Hz sine wave sampled at 2000 Hz. The reconstructed signal will appear to be a sine wave at 900 Hz. Aliasing alters the observed frequency spectrum of a signal by mapping higher frequency components into lower frequency components.

The effect of aliasing on spectrum analysis can be reduced by passing the signal through an anti-aliasing filter before it is digitized. An anti-aliasing filter is a high-order, analog, low-pass filter with a cutoff frequency that is half the sampling rate. The anti-aliasing filter attenuates the troublesome high-frequency components of the signal. For example, consider again the case of an 1100-Hz signal sampled at 2000 Hz. The signal will be aliased appearing to be a sine wave at 900 Hz. The proper anti-aliasing filter will have a cutoff frequency of 1000 Hz. If possible, the better solution is to sample at a rate sufficiently high to resolve all components of the signal. In this case, we should sample at a rate greater than 2200 samples per second.

To demonstrate the effect of aliasing, we shall produce a sine wave at a specified frequency using the Sine Wave block in Simulink. We shall then use Sample and Hold block to demonstrate the sampled signal. This approach will allow us to make an easy and direct comparison between (a) the original signal from the Sine Wave block and (b) the reconstructed signal at the output of the Sample and Hold block. We shall visually compare the two signals by drawing them both on a single MATLAB plot. We shall also use audio headphones/speakers to hear the difference. The original signal will produce a tone first, followed by the reconstructed signal five seconds later. Aliasing will be evident when the base frequencies are different. **By base frequency, we mean the lowest pitch note in the sound.**

We shall begin with a sine wave at 100 Hz. Because the sampling rate (1000 samples per second) is much greater than two times the frequency of the signal (2 $\times$ 100 Hz = 200 samples per second), the reconstructed signal will be a very good approximation of the original signal. We shall then increase the frequency of the sine wave using the block parameters of the Sine Wave block. Aliasing will begin when the frequency of the sine wave reaches half the sampling rate or 500 Hz. After this point, higher frequency signals will be mapped to lower frequency ones, due to inadequate sampling rate. We shall verify this behavior visually using the plotting function of MATLAB and aurally using the earphones/speakers.

5.1.1 Open MATLAB and Simulink

(a) Open MATLAB. Open Simulink.

5.1.2 Set Up the Simulink Diagram

(a) In your Simulink start menu, select *Blank Model.* Set stop time to 5.
(b) Open *Library Browser.*
(c) On your *Simulink Library Browser,* navigate to Simulink -> Sources. Drag *Sine Wave* to your diagram. Set the amplitude to 10 and frequency to 100 Hz (2*pi*100 rad/s). Set its sample time to 1/44100.
(d) Go to *Sinks* and drag two *To Workspace* blocks to your Simulink diagram. Set the name of one of these blocks to “sampledSound” and the other to “originalSound”.
(e) In your *Library Browser*, go to Simulink -> Sources and drag *Signal Generator* to your Simulink diagram. Set its waveform to **square** and frequency to 1000 Hz. Amplitude must be a non-zero value.
(f) In your *Library Browser,* go to DSP System Toolbox -> Signal Operations and drag *Sample and Hold* to your diagram. Trigger type should be set to *Rising Edge* with initial condition set to 0.
(g) Connect your blocks as in Figure 1.

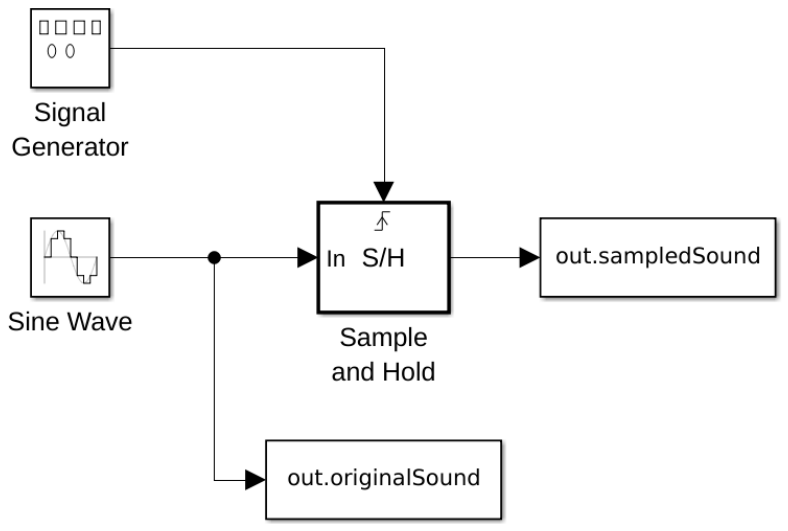

Figure 1:Simulink Diagram for Aliasing

5.1.3 Run the Simulation and Observe Aliasing

(a) Make sure that your computer has a speaker/microphone installed in good working condition. **For your health, it is recommended that you start with low volumes and adjust accordingly.**
(b) In your Simulink diagram, hit Run and wait until the simulation is completed.
(c) Now, go back to your MATLAB. Create a new script and run the following:

```
Fs=44100;
originalTime = out.originalSound.Time;
originalSound = out.originalSound.Data;
sampledTime = out.sampledSound.Time;
sampledSound = out.sampledSound.Data;
disp('Playing the original sound...')
soundsc(originalSound,Fs);
pause(6); % 1 second delay between sounds (pause time – simulation time)
disp('Playing the sampled sound...')
soundsc(sampledSound,Fs);
pause(6);

figure(1)
plot(originalTime(originalTime>4.9),originalSound(originalTime>4.9))
axis([4.9 5 -15 15])
hold on
grid on
plot(sampledTime(sampledTime>4.9),sampledSound(sampledTime>4.9))
hold off
legend('Original','Sampled')
```

(d) First, listen to the original sound. Once the original sound is completed, you will hear the sampled sound. Compare the base frequencies. **By base frequency, we mean the lowest pitch note in the sound.** Note down your observations: Does the base frequency change?

(e) Now, look at the figure that appears as a result of the script. Is it consistent with your judgement from (d)?

5.1.4 Try Different Frequencies

(a) Open your Simulink diagram again. Repeat your observations in the previous part by running the simulation with 500 Hz, 975 Hz, 1000 Hz, 1200 Hz, 1500 Hz, 1700 Hz, 2000 Hz as the frequencies for the *Sine Wave* block. Record your observations for each frequency. (Feel free to automate this step by using *for* or *while* loops!)

5.2 Quantization Error in a Ramp Function

Overview of Procedure

In this part of the experiment, we shall illustrate quantization error, which is the possible error between the analog voltage and the digital representation of that voltage. Quantization error occurs because only discrete or integer values exist in a digital representation. Quantization error is the result of *amplitude discretization* and may be contrasted with aliasing, which is the result of *time discretization*.

We shall use a slightly contrived situation here. We shall use *Ramp* block in Simulink to produce a digital ramp from 0 to 25 mV over a 10-s period. The resolution of the digital-to-analog converter in the physical laboratory is 4.88 mV (See Appendix A). In order to simulate this, we use *Quantizer* block. As a result, the analog output will actually consist of five discrete steps of 4.88 mV rather than a smooth linear ramp.

5.2.1 Open MATLAB and Simulink

1) Open MATLAB. Open Simulink.

5.2.2 Set Up the Simulink Diagram

(a) In your Library Browser, go to Simulink -> Sources. Drag *Ramp* block to your Simulink diagram. Set its slope to 0.0025.
(b) In the Library Browser, go to Simulink -> Signal Routing. Drag *Mux* to your diagram.

(c) In the Library Browser, go to Simulink -> Discontinuities. Place *Quantizer* block to your diagram. Set its quantization interval to 20/(2^12).
(d) In the Library Browser, go to Simulink -> Sinks. Drag two *Scope* to your diagram.
(e) Go to Simulink->Sources and add *Band-Limited White Noise* to your diagram. Set its noise power to 10e-11 and its sample time to 0.001.
(f) Get a *Constant* block from Sources. Set its value to 0.003.
(g) Go to Simulink->Commonly Used Blocks and get two *Sum* blocks.
(e) Connect your blocks as in Figure 2 below. Here, note that the name of the *Constant* block is changed to "DC Offset". The scopes are named "Scope 1" and "Scope 2". These are intentional and will be explained in the next subsection.

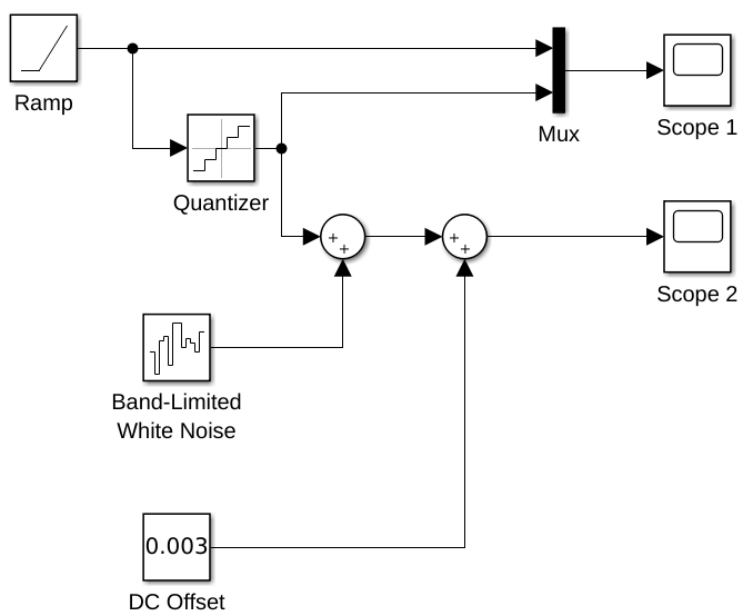

Figure 2: Simulink Diagram for Quantization Error

5.2.3 Run the Simulation and Observe Quantization Error

(a) In Simulink, hit Run and wait until the simulation is completed.
(b) First, examine Scope 1 (refer to Figure 2 for naming). Here, the yellow line is the analog ramp input that we want to produce. However, our DAC is ±10 V and 12-bits. As a result, the resolution of our DAC is approximately 4.88 mV (refer to Appendix B.2 to learn how to calculate this). That's why, the blue stepped signal is the best that we can get out of this DAC.
(c) Now, examine Scope 2. This is a typical illustration of what is observed on an oscilloscope screen. There will be considerable amount of noise. There will also be a DC offset (the signal will not start from zero as it should). Submit both scopes in your report.

6. INSTRUCTIONS FOR THE REPORT

6.1. Format

You are expected to write a **brief** report demonstrating your work and your understanding of the concept. The report should have the following:

- Must be in pdf format.
- Name the file as 'Lastname_Firstname_360_Lab2.pdf'.
- Use either 11pt or 12pt font, Arial/Calibri/Times New Roman, single spaced.
- 1'' margins.
- Number all of your pages.
- Add a title page.

6.2. Content

Your report should include all of the following:

Concept Questions:

1) What is the resolution (expressed as a voltage) of a ±5 V, 16-bit, digital-to-analog converter (DAC)? (Note the one used in the lab is a ±10 V 12-bit DAC)
2) What are three methods of waveform reconstruction?
3) For each reconstruction method, what would the max percent error be for n = 7 samples per period? For n = 20?
4) If the sampling rate is 2500 samples per second, at which frequency does aliasing begin?
5) If a 1555-Hz sine wave is sampled at the rate of 2000 samples per second, what will be the frequency of the reconstructed waveform?
6) If the ADCs in the lab have as an input a 200Hz, 1Vpp sine wave, what *sample period* (1/sample rate) should be chosen so the computer collects 30 samples per period.

Simulation Results/ Discussion

- In the aliasing example, submit your aural observations on the base frequency. Submit your plots for each of them as well.
- What are three methods of waveform reconstruction are discussed in Appendix B1? All else being equal, which one yields the smallest reconstruction error? Which one yields the largest error? Which one did we use in this lab?
- If the sampling rate is 2500 samples per second, at which frequency does aliasing begin?
- The output of the reconstructed signal produces multiple tones. What is the origin of the lowest frequency tone? What is the origin of the higher frequency tone?
- How would you measure the frequency of a reconstructed waveform?
- For the quantization example, submit the outputs of your scopes.
- In Figure 3 below, you will find a ramp input generated by a virtual DAC. Assume that this DAC is also ±10 V. Find the DC offset and the number of bits of this DAC.

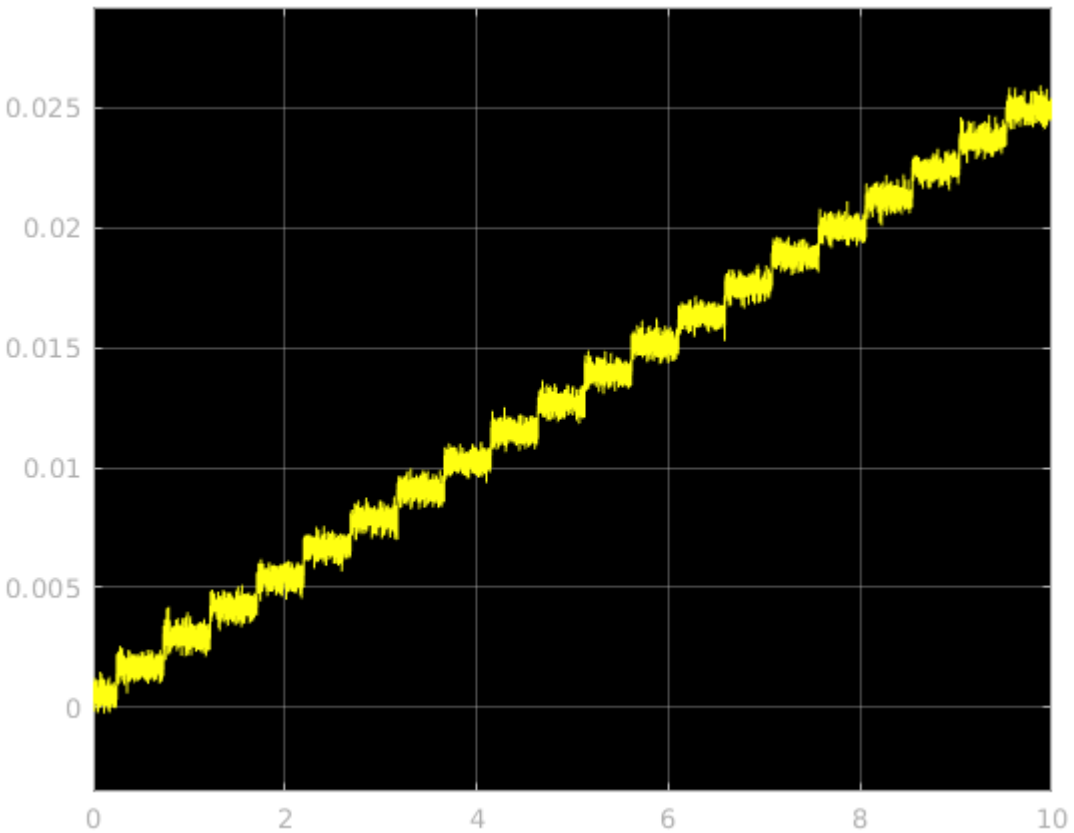

Figure 3: Ramp Signal Generated by a Virtual DAC

Appendix A. PC-Based Hardware and Software

Each laboratory station has a Windows®-based personal computer with internal hardware for performing analog-to-digital (A/D) and digital-to-analog (D/A) conversion. The basic configuration is shown in Fig. A.1.

The analog inputs from the patch panel are passed to two Analog Devices 5B41 isolation amplifiers each with a gain of 0.5. These modules map a voltage range of -10 to +10 VDC at the patch panel inputs onto the range of -5 to +5 VDC accepted by the PC-based analog-to-digital converter. The 0.5 gain of the isolation amplifier is accounted for in the data acquisition block of our SIMULINK model.

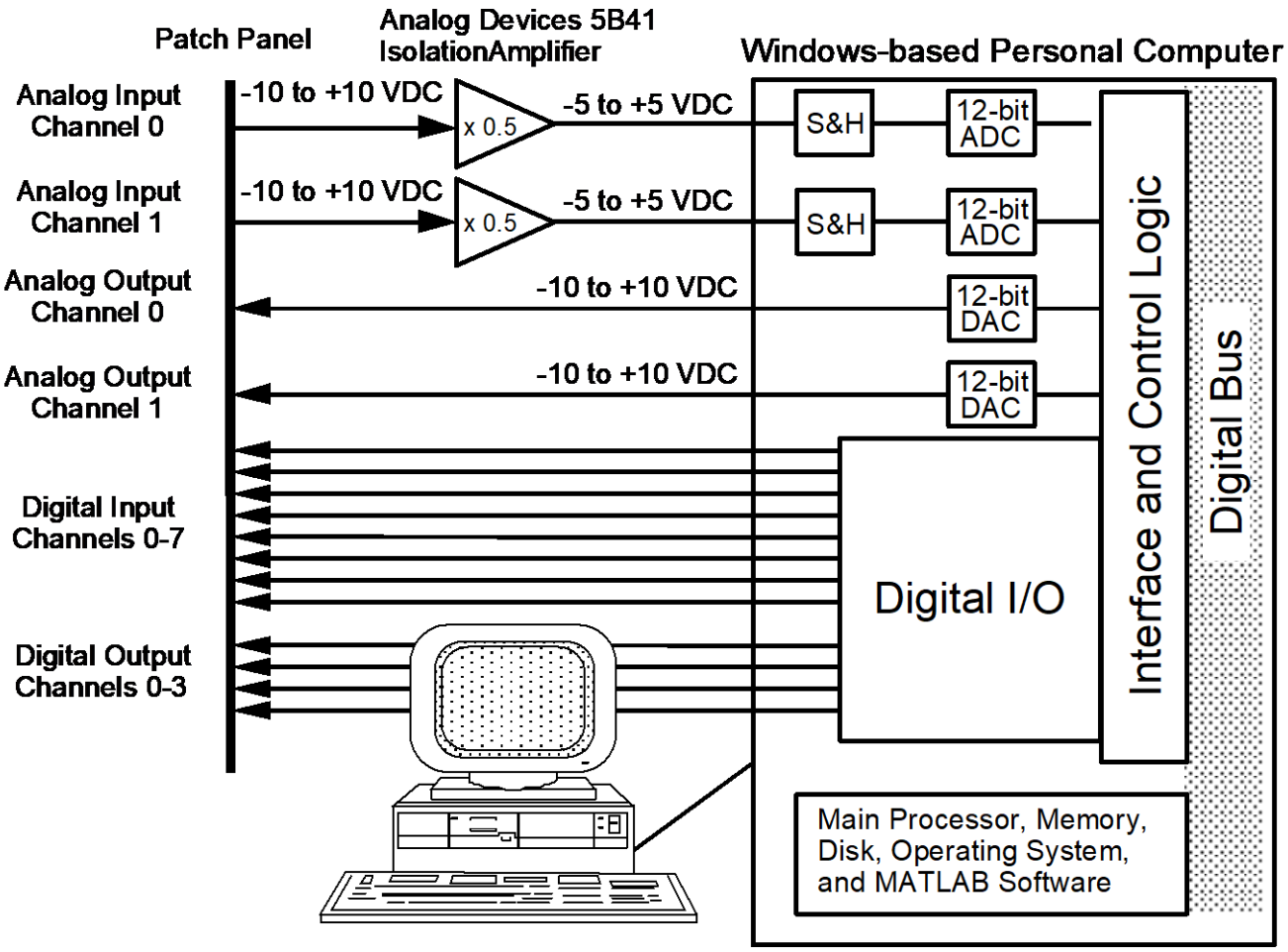

Figure A.1 PC-based Hardware for A/D and D/A Conversion.

The PC has a 12-bit analog-to-digital converter. It uses successive approximation to convert the analog input voltage to a 12-bit binary integer (i. e., an integer between 0 and 4095 or between 000000000000_2 and 111111111111_2 where the subscript "$_2$" denotes a base-2 or binary number). The sample-and-hold circuit of the analog-to-digital converter samples the input voltage at a particular instant of time and holds this value steady while the conversion is made. The digital or integer representation of the input voltage n_{in} is given by

$$n_{in} = \text{truncate}\left(2^{n_b} \frac{V_{in} - V_{min}}{V_{max} - V_{min}} \right) = \text{truncate}\left(\frac{V_{in} - V_{min}}{\Delta V_{bit}} \right) ,$$

where $n_b = 12$ = number of bits in the integer representation, $V_{min} = -10$ VDC = the minimum patch-panel input voltage, $V_{max} = +10$ VDC = maximum patch-panel input voltage, $\Delta V_{bit} = (V_{max} - V_{min}) / 2^{n_b} = 4.88$ mV = the bit resolution of the conversion, and "truncate" is a function that converts a real number to an integer by dropping the decimal fraction.

Similarly, the digital-to-analog converter converts a 12-bit binary integer n_{out} to a voltage between -10 V and +10 V according to

$$V_{out} = n_{out} \frac{V_{max} - V_{min}}{2^{n_b}} + V_{min} = n_{out} \Delta V_{bit} + V_{min}.$$

The conversions are made at a software-selectable discrete time step Δt. The sampling rate f_s given by $1/\Delta t$ is expressed in units of samples per second or s^{-1}. For the analog-to-digital conversions, we assume that the sample-and-hold acquisition time is much smaller than the time step so that the sampling may be considered instantaneous.

Appendix B. Analog-to-digital Conversion Issues

When an analog signal is digitized, both time and amplitude are mapped onto a set of integers. This process is often referred to as "discretization" because the digital representation consists of discrete (integer) values. Time discretization occurs because the analog signal is sampled at particular instants or over particular intervals of time. Amplitude discretization occurs because the amplitude of the analog signal is converted to a binary integer between 0 and $2^{n_b} - 1$ where n_b is the number of binary digits or bits in the digital representation. Some information is inevitably lost when the real variables time and amplitude are mapped to integers. *Waveform reconstruction error* and *aliasing* are associated with the *discrete time* nature of the digital representation whereas *quantization error* is associated with the *discrete amplitude* nature of the digital representation. We consider reconstruction error and aliasing in Section B.1 and treat *quantization* error in Section B.2.

B.1 Waveform Reconstruction Error and Aliasing

Waveform reconstruction is the process by which the original signal is reconstructed from the digital representation. Reconstruction error is the difference between the reconstructed signal and the original signal. The magnitude of this error depends on the sampling rate and the method of reconstruction. Aliasing is a special type of reconstruction error that occurs when the sampling rate is less than twice the frequency of the original signal. Under these conditions, a reconstructed sine wave (or frequency component of a more complex waveform) appears to be at a lower frequency than it actually is.

Here, we examine three methods of waveform reconstruction: (i) histogram reconstruction, (ii) vector reconstruction, and (iii) Fourier reconstruction (See Figure B.1). These three methods are illustrated in Fig. B.1. In histogram reconstruction, the voltage level is changed at the start of each time step and then held constant until the beginning of the next time step. Most digital-to-analog converters use this method of approximation because of its simplicity.

One level up in terms of both complexity and accuracy is the vector or trapezoidal method. With the vector method, the voltage is linearly interpolated between two successive values. This method of reconstruction is used by the digital oscilloscope at your station.

The Fourier reconstructed waveform $V_f(t)$ is given by

$$V_f(t) \quad = \quad \sum_{k=-\infty}^{k=+\infty} \left\{ V_k \, \frac{\sin\left[\pi\left(\frac{t}{\Delta t} - k\right)\right]}{\pi\left(\frac{t}{\Delta t} - k\right)} \right\}$$

where V_k is the sampled voltage given by

$$V_k \quad = \quad V(k\,\Delta t) \quad .$$

Note that Fourier reconstruction requires an infinite number of samples ranging from $k = -\infty$ to $k = +\infty$. If the signal is periodic and sampled at an integer multiple of the waveform frequency, then the required values can be obtained by periodic extension. Otherwise, the data to fully implement Fourier reconstruction do not exist. Fourier reconstruction is immensely important because this method can be used to completely recover the original waveform provided that certain conditions are met. One of the conditions that has to be met is known as the Nyquist sampling theorem. *Fourier reconstruction returns the original waveform without error if the sampling rate is at least twice the highest frequency component of the waveform.* If the sampling rate is not at least twice the highest frequency than aliasing occurs. So aliasing occurs when a signal is sampled too slowly (See Figure B.2).

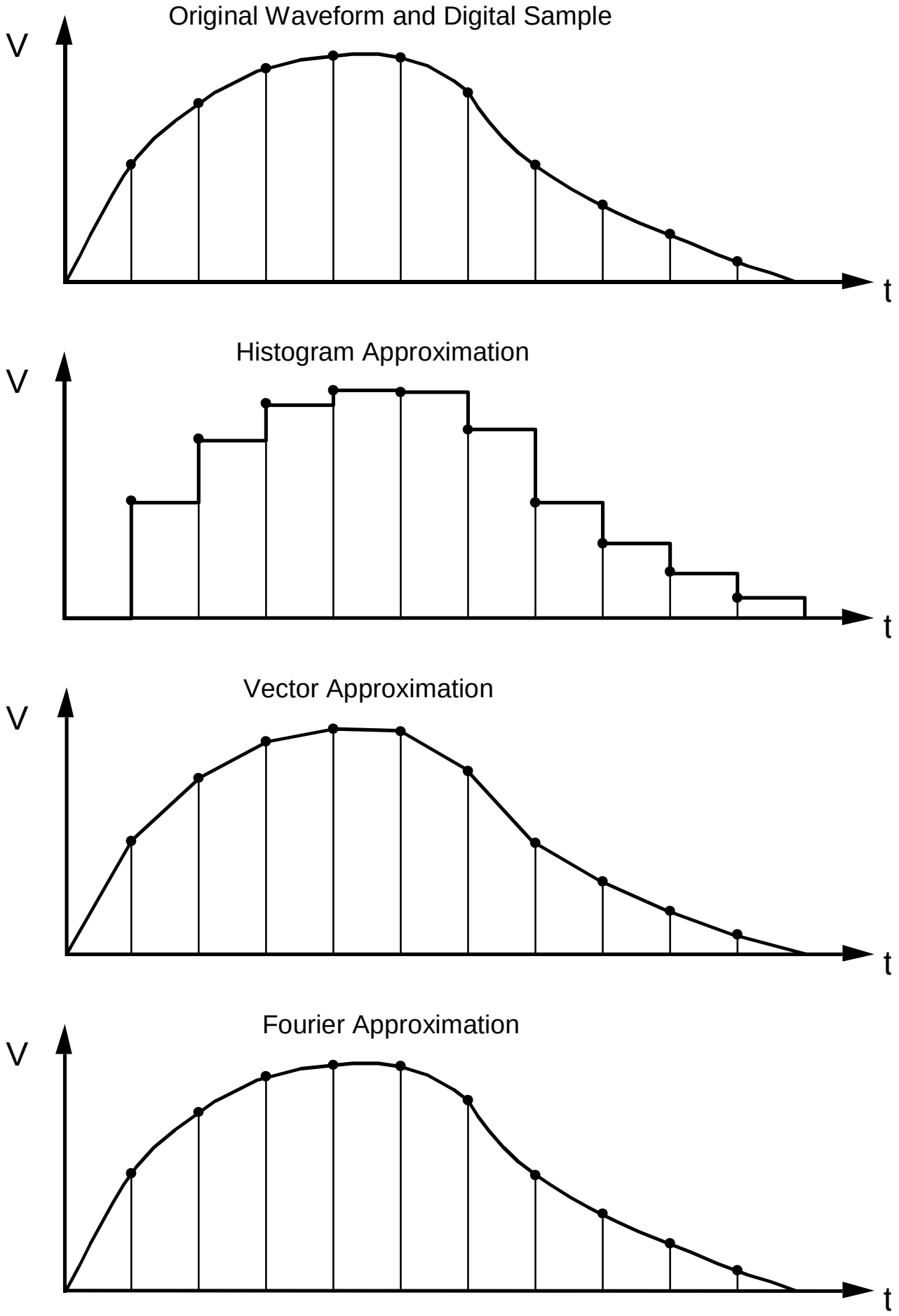

Figure B.1 Three Methods of Reconstructing a Waveform from its Digital Representation.

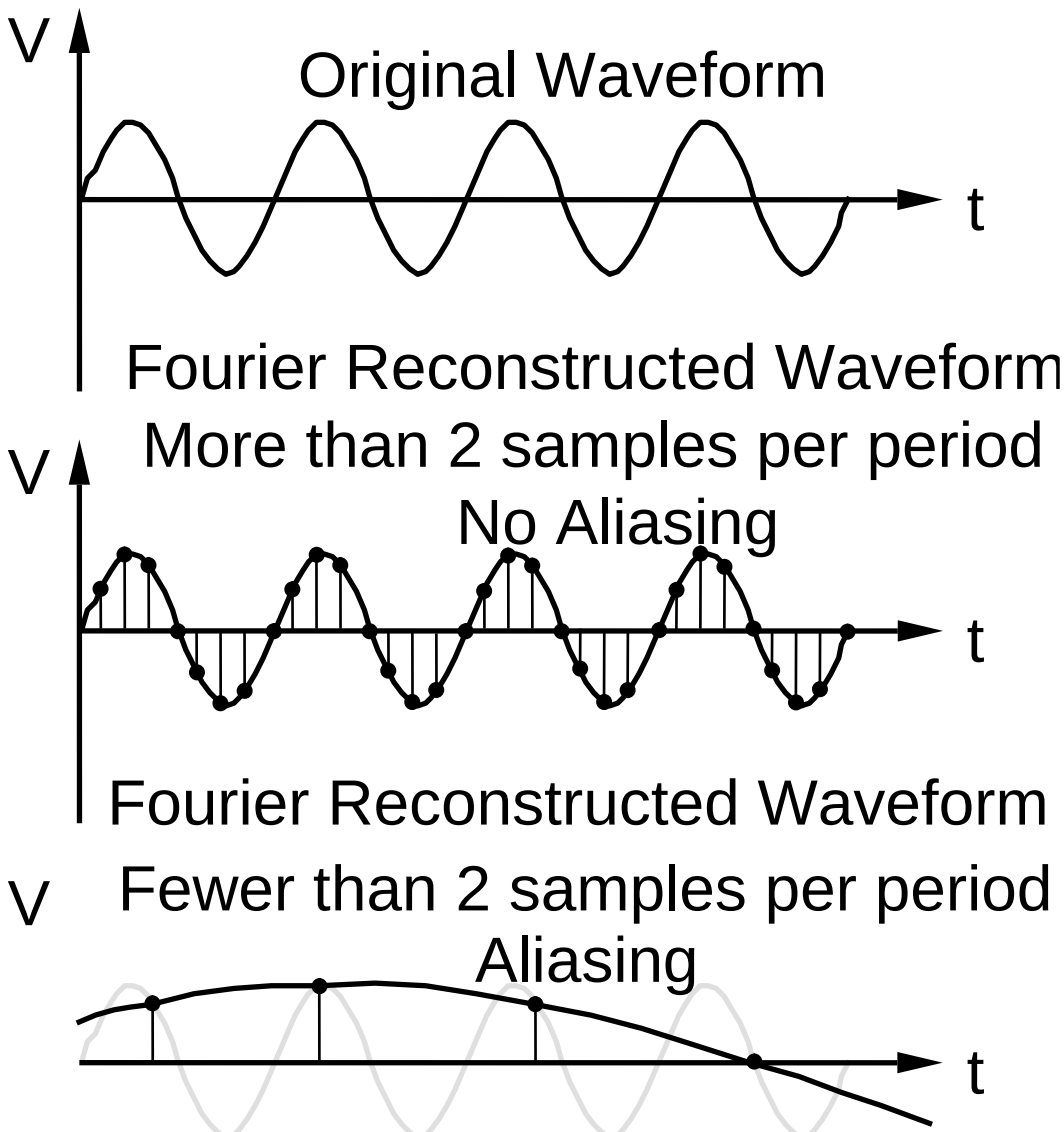

Figure B.2 Effect of Sampling Rate on Waveform Reconstruction.

Aliasing can only be reduced by sampling at a higher rate. When this is not possible, then low-pass filtering can be used to reduce the amplitude of the high frequency components and thus lessen the negative effect of aliasing on spectral analysis. A low-pass filter serving this latter purpose is called an "anti-aliasing filter".

Although a sampling rate of 2 samples per period is sufficient to avoid aliasing, such a low sampling rate is hardly sufficient to properly define a digitized waveform in most practical applications. Figure B.3 shows how reconstruction error varies with the number of samples per period for the three methods considered here. The test case is the approximation of a sinusoidal waveform, and the error is defined as the ratio of the maximum possible absolute error in the reconstructed waveform to the amplitude of the original signal.

The histogram method requires a large number of samples per cycle to produce a reasonable approximation. From Fig. B.3, we see that, even for 30 samples per cycle, the maximum error is still about 10 %. With the vector method, the maximum error drops below 1 % for 12 or more samples per cycle. The oscilloscope at your station normally uses the vector method. The vector method can be turned off by first pressing the "display" key on the front panel of the instrument and then turning "vectors off" with the context-sensitive keys below the display. The reconstructed waveform will then be shown as a series of dots.

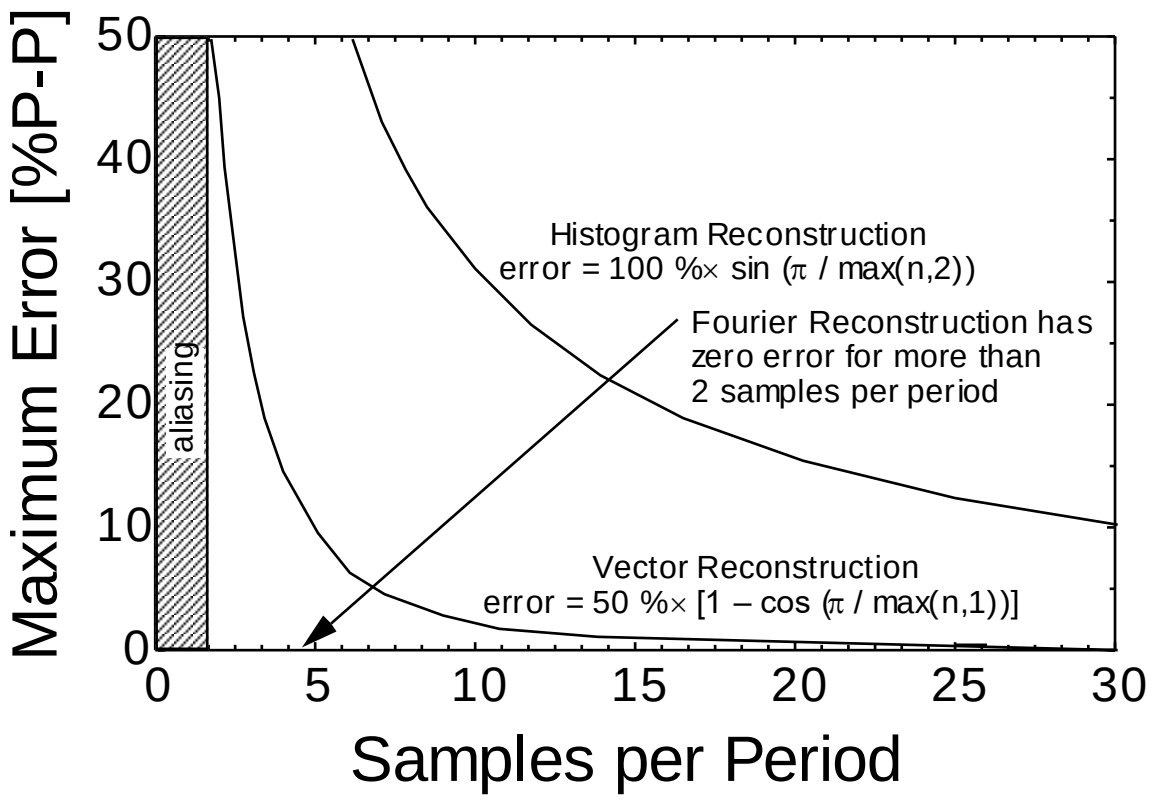

Figure B.3 Effect of Sampling Rate on Maximum Error in Reconstructed Waveform.

B.2 Quantization Error

Quantization error arises because the signal is digitized in discrete amplitude steps. Quantization error is reduced by decreasing the range of the measurement or by increasing the number of bits in the binary integer representation (e. g., converting an 8-bit A/D to a 12-bit, 16-bit or even 18-bit A/D). In general, quantization error is a much less severe problem than sampling speed and aliasing. Even an 8-bit A/D converter provides a resolution of 1 part in 256 or about 0.4 %. This corresponds to roughly 256 samples per period in terms of sampling speed. A 10-bit converter provides a resolution of better than 0.1% which is adequate for many instrumentation applications. As far as experimental error goes, this is fairly small. There are, however, some digital audio and video applications where a dynamic range of three orders of magnitude or more is needed in which case quantization error is a concern. To achieve adequate resolutuion at low levels while maintaining sufficient dynamic range using an integer representation requires a large number of bits.

B.3 Noise

Noise is often a far more serious problem than either aliasing or quantization error. The waveform seen by an instrument can be separated into three components as follows.

$$\text{waveform} = \text{signal} + \text{line noise} + \text{high frequency noise}$$

Here, "signal" represents the valuable part of the waveform. "Line noise" which emanates from AC power lines and AC powered devices is at the power-line frequency (60 Hz in the U. S.) and its harmonics (integer multiples of the power-line frequency). The "high-frequency noise" component comes from a wide variety of sources and may range in frequency from a few kHz to several MHz or even GHz. Three basic methods of dealing with noise are (a) shielding, (b) filtering, and (c) integration. We shall examine the latter two here; the issue of shielding will be treated in the next experiment.

B.3.1 Filtering

Filtering can be effective against both power-line noise and high-frequency noise although it is probably best at reducing the latter. Low-pass filters can be very effective against high-frequency noise provided that the frequency of the noise is much higher than the frequency of the signal. For example, an eighth-order, low-pass filter can be obtained at reasonable cost. If the high-frequency noise is only one decade higher in frequency than the signal, then eight orders of magnitude in reduction of the high-frequency noise is achieved. Even a first-order, passive, RC filter can be effective against high-frequency noise. Notch or band-reject filters can be used to reject power-line frequencies, but these filters typically do not suppress harmonics and they may distort the desired signal.

APPENDIX C.MATLAB Tutorial

The following MATLAB tutorial can be used to familiarize yourself with basic MATLAB syntax in preparation for the use of this software during the rest of the semester.

C.1 Basic commands and syntax

(a) The MATLAB command line can be used just like a calculator to perform calculations. The result is stored in the default variable "ans". Try the following simple commands. Enter the command shown followed by a return.

6*4+3*3-2^3/2	note that standard algebraic operator preference is used
5 * cos(2*pi)	note the use of the built-in variable "pi"
5 > 3 5 < 3	note the use of the relational operator ">"; the relational operators are: "=" equal "~=" not equal "<" less than ">" greater than "<="less than or equal ">=" greater than or equal
3 <= 5 & ... 4 >= 1	note the use of the logical operator "and" and the continuation syntax "..." (three consecutive periods); the logical operators are "&" and "|" or "~" not xor (a,b) exclusive or
% comment line	the percent sign allows comments to be mixed in with MATLAB commands

C.2 MATLAB Variables

(b) MATLAB allows variables to be defined and used in algebraic expressions. Try the following simple examples.

a = 5; b = 3; c = a * b^2	note that each input line can contain multiple expressions separated by a semicolon
d = a/b; d	note that the semicolon also suppresses the result display after the first line; whereas a variable name on a line by itself displays its current value
whos	displays detailed information about the variables defined
clear d; whos clear; whos	"clear" deletes a specific variable or all variables if no name is specified

C.3 MATLAB Arrays

(c) The fact that MATLAB provides a natural and simple syntax for handling arrays is one of its most powerful features. Try the following simple commands. Predict the result before actually entering the command.

a = [1 3*pi sqrt(2)]	note that array elements which are enclosed in brackets "[ ]" and separated by spaces can be any algebraic expression
b = [1 2 3; 4 5 6]	creates the 2-by-3 array $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$; rows can also be separated by a return
b(2,3) = 9	sets the element in row 2 and column 3 of array b to 9
b(1,2)	displays element in row 1 and column 2 of b
b(1,1:3)	displays elements in row 1 and columns 1 through 3 of b
b(2,1:2) = [7 8]	sets elements in row 2 and columns 1 through 2 of b to 7 and 8, respectively
c = 1:0.2:2	creates a row vector starting at 1 and ending at 2 with an increment of 0.2
d = linspace(0,1,6)	creates a row vector starting at 0 and ending at 1 with 6 evenly spaced elements
e = logspace(1,3,5)	creates a row vector starting at 10^1 and ending with 10^3 with 5 elements each succeeding pair of which differs by the same factor
f = zeros(2,3)	creates an array of zeros with 2 rows and 3 columns

g = 5*ones (3,1)	creates an array with 3 rows, 1 column and all elements equal to 5
h = b'	sets h equal to the transpose of b
p = sqrt(b) / 2	computes the square root of each element in b, then divides each element by 2
q = 4 * p.^2	squares each element in p, then multiplies by 2; note the special operator element-by-element operator ".^"
r = b * h	matrix multiplication in the conventional manner
s = 2 * r^2	same as 2 * r * r; matrix multiplication of r by itself is done first, then each element of the result is doubled; r must be a square matrix
t = c + d	adds two arrays in the conventional (element-by-element) manner
u = c .* d	multiplies two arrays element-by-element yielding [$c_1 * d_1$ $c_2 * d_2$ $c_3 * d_3$ $c_4 * d_4$ $c_5 * d_5$ $c_6 * d_6$]
whos	displays information on arrays defined
clear	clears all scalar and array variables

C.4 Complex Arithmetic in MATLAB

(d) MATLAB also handles complex numbers in a natural manner. Try the following examples.

a = 3 + 4 j	"j" is used to signify $\sqrt{-1}$
b = 2 + sqrt(-9) c = exp(j * pi / 6) d = abs(a) e = angle(a) f = real(a); g = imag(a)	conventional complex functions
h = a * b p = a / b	conventional complex arithmetic
q = [1+2j 3+4j]	complex arrays are constructed from complex numbers in the natural way
whos	display information about variables
clear	clear all variables

C.5 MATLAB Plotting Capability

(e) MATLAB provides powerful plotting capability. Try the examples below.

x = 0:0.01:10;	creates a row vector of 1001 elements from 0 to 10 by increments of 0.01 *The semicolon at the end of the command line suppresses the long listing of the array elements.*
y = sin(x); z = cos(x); u = x .^ (-2.5);	creates two row vectors of 1001 elements each such that $y_i = \sin(x_i)$, $z_i = \cos(x_i)$, and $u_i = x_i^{-2.5}$ *Again, the semicolons are a big time saver.*
plot (x, y, x, z)	plots y and z versus x. Note the plot may be automatically minimized. To display a minimized plot, double click on the appropriate button in the window tray at the bottom of the screen
axis ([2 5 0 1])	modifies the axes so that xmin = 2, xmax = 5, ymin = 0 and ymax = 1. Note that the axis function takes an array of four elements as a single argument and not four scalars as individual arguments. Restore the plot to the screen and note the effect.
axis auto;	restores default axes limits
loglog (x, u)	makes a log-log plot of u vs. x
semilogx (x, u)	makes a semi-log plot with the log scale of the abscissa
semilogy (x, u)	makes a semi-log plot with the log scale of the ordinate
clear	clears all variables