Sampling and Aliasing

With this chapter we move the focus from signal modeling and analysis, to converting signals back and forth between the analog (continuous-time) and digital (discrete-time) domains. Back in Chapter 2 the systems blocks C-to-D and D-to-C were introduced for this purpose. The question is, how must we choose the sampling rate in the C-to-D and D-to-C boxes so that the analog signal can be reconstructed from its samples.

The lowpass sampling theorem states that we must sample at a rate, $f_s$, at least twice that of the highest frequency of interest in analog signal $x(t)$. Specifically, for $x(t)$ having spectral content extending up to $B$ Hz, we choose $f_s = 1/T_s > 2B$ in forming the sequence of samples

$$x[n] = x(nT_s), \quad -\infty < n < \infty.$$ (4.1)

Sampling

- We have spent considerable time thus far, with the continuous-time sinusoidal signal

$$x(t) = A \cos(\omega t + \phi),$$ (4.2)

where $t$ is a continuous variable

- To manipulate such signals in MATLAB or any other computer too, we must actually deal with samples of $x(t)$
Recall from the course introduction, that a discrete-time signal can be obtained by uniformly sampling a continuous-time signal at \( t_n = nT_s \), i.e., \( x[n] = x(nT_s) \)

- The values \( x[n] \) are samples of \( x(t) \)
- The time interval between samples is \( T_s \)
- The sampling rate is \( f_s = 1/T_s \)
- Note, we could write \( x[n] = x(n/f_s) \)

A system which performs the sampling operation is called a \textit{continuous-to-discrete} (C-to-D) converter

\[
x(t) \quad \text{Ideal C-to-D Converter} \quad x[n] = x(nT_s)
\]

Simple Sampler
Switch Model

\[
x(t) \quad \text{Momentarily close to take a sample} \quad x[n] = x(nT_s)
\]

Electronic Subsystem:
ADC or A-to-D

\[
x(t) \quad \text{Analog to Digital Converter} \quad x[n] = x(nT_s)
\]

- SAR = successive approximation register
- \( \Delta\Sigma = \) delta-sigma modulator (oversampling)
• A real C-to-D has imperfections, with careful design they can be minimized, or at least have negligible impact on overall system performance

• For testing and simulation only environments we can easily generate discrete-time signals on the computer, with no need to actually capture and C-to-D process a live analog signal

• In this course we depict discrete-time signals as a sequence, and plot the corresponding waveform using MATLAB’s stem function, sometimes referred to as a lollypop plot

```matlab
>> n = 0:20;
>> x = 0.8.^n;
>> stem(n,x,'filled','b','LineWidth',2)
>> grid
>> xlabel('Time Index (n)')
>> ylabel('x[n]')
```

\[ x[n] = \begin{cases} 
0, & n < 0 \\
0.8^n, & n \geq 0 
\end{cases} \]
Sampling Sinusoidal Signals

• We will continue to find sinusoidal signals to be useful when operating in the discrete-time domain

• When we sample (4.2) we obtain a sinusoidal sequence
  \[ x[n] = x(nT_s) \]
  \[ = A \cos(\omega n T_s + \phi) \]
  \[ = A \cos(\hat{\omega} n + \phi) \] (4.3)

• Notice that we have defined a new frequency variable
  \[ \hat{\omega} \equiv \omega T_s = \frac{\omega}{f_s} \text{ rad}, \] (4.4)

  known as the *discrete-time frequency* or normalized continuous-time frequency

  – Note that \( \hat{\omega} \) has units of radians, but could also be called radians/sample, to emphasize the fact that sampling is involved

  – Note also that many values of \( \omega \) map to the same \( \hat{\omega} \) value by virtue of the fact that \( T_s \) is a system parameter that is not unique either

  – Since \( \omega = 2\pi f \), we could also define \( \hat{f} \equiv fT_s \) as the discrete-time frequency in cycles/sample

Example: Sampling Rate Comparisons

• Consider \( x(t) = \cos(2\pi \cdot 60 \cdot t) \) at sampling rates of 240 and 1000 samples per second
The corresponding sample spacing values are

\[ T_s = \frac{1}{240} = 4.1666 \text{ ms} \quad T_s = \frac{1}{1000} = 1 \text{ ms} \]

>> t = 0:1/2000:.02;
>> x = cos(2*pi*60*t); % approx. to continuous-time
>> t240 = 0:1/240:.02;
>> n240 = 0:length(t240)-1;
>> x240 = cos(2*pi*60/240*n240); % fs = 240 Hz
>> axis([0 4.8 -1 1]) % axis scale since .02*240 = 4.8
>> t1000 = 0:1/1000:.02;
>> n1000 = 0:length(t1000)-1;
>> x1000 = cos(2*pi*60/1000*n1000); % fs = 1000 Hz

![Graphs showing sampling](image-url)
• The analog frequency is $2\pi \cdot 60 \text{rad/s}$ or 60 Hz
• When sampling with $f_s = 240$ and 1000 Hz
  \[ \hat{\omega}_{240} = 2\pi \cdot 60/240 = 2\pi(0.25) \text{ rad} \]
  \[ \hat{\omega}_{1000} = 2\pi \cdot 60/1000 = 2\pi(0.06) \text{ rad} \]
respectively
• The sinusoidal sequences are
  \[ x_{240}[n] = \cos(0.5\pi n) \]
  \[ x_{1000}[n] = \cos(0.12\pi n) \]
respectively
• It turns out that we can reconstruct the original $x(t)$ from either sequence
• Are there other continuous-time sinusoids that when sampled, result in the same sequence values as $x_{240}$ and $x_{1000}$?
• Are there other sinusoid sequences of different frequency $\omega$ that result in the same sequence values?

The Concept of Aliasing

• In this section we begin a discussion of the very important signal processing topic known as aliasing
• **Alias** as found in the Oxford American dictionary: *noun*
  – A false or assumed identity: a spy operating under an alias.
  – Computing: an alternative name or label that refers to a file, command, address, or other item, and can be used to locate or access it.
– **Telecommunications**: each of a set of signal frequencies that, when sampled at a given uniform rate, would give the same set of sampled values, and thus might be incorrectly substituted for one another when reconstructing the original signal.

- Consider the sinusoidal sequence
  \[ x_1[n] = \cos(0.4\pi n) \quad (4.5) \]

- Clearly, \( \hat{\omega} = 0.4\pi \)

- We know that cosine is a mod \( 2\pi \) function, so
  \[ x_2[n] = \cos(2.4\pi n) \]
  \[ = \cos[(2 + 0.4)\pi n] = \cos(0.4\pi n + 2\pi n) \quad (4.6) \]
  \[ = \cos(0.4\pi n) = x_1[n] \]

- We see that \( \hat{\omega} = 2.4\pi \) gives the same sequence values as \( \hat{\omega} = 0.4\pi \), so \( 2.4\pi \) and \( 0.4\pi \) are aliases of each other

- We can generalize the above to any \( 2\pi \) multiple, i.e.,
  \[ \hat{\omega}_l = \omega_0 + 2\pi l, \quad l = 0, 1, 2, 3, \ldots \quad (4.7) \]
  result in identical frequency samples for \( \cos(\hat{\omega}_l n) \) due to the mod \( 2\pi \) property of sine and cosine

- We can take this one step further by noting that since \( \cos(\theta) = \cos(-\theta) \), we can write
  \[ x_3[n] = \cos(1.6\pi n) \]
  \[ = \cos[(2 - 0.4)\pi n] = \cos(2\pi n - 0.4\pi n) \quad (4.8) \]
  \[ = \cos(-0.4\pi n) = \cos(0.4\pi n) \]
– We see that \( \hat{\omega} = 1.6\pi \) gives the same sequence values as
\( \hat{\omega} = 0.4\pi \), so \( 1.6\pi \) and \( 0.4\pi \) are aliases of each other

- We can generalize this result to saying
\[
\hat{\omega}_l = 2\pi l - \hat{\omega}_0, \ l = 0, 1, 2, 3, \ldots 
\] (4.9)
result in identical frequency samples for \( \cos(\hat{\omega}_l n) \) due to the
mod \( 2\pi \) property and evenness property of cosine

- This result also holds for sine, expect the amplitude is
inverted since \( \sin(-\theta) = -\sin(\theta) \)

- In summary, for any integer \( l \), and discrete-time frequency
\( \hat{\omega}_0 \), the frequencies
\[
\hat{\omega}_0, \hat{\omega}_0 + 2\pi l, 2\pi l - \hat{\omega}_0, \ l = 1, 2, 3, \ldots 
\] (4.10)
all produce the same sequence values with cosine, and with
sine may differ by the numeric sign

- A generalization to handle both cosine and sine is to con-
sider the inclusion of an arbitrary phase \( \phi \),
\[
A \cos[(\hat{\omega} + 2\pi l)n + \phi] = A \cos[\hat{\omega}n + 2\pi l \cdot n + \phi]
= A \cos(\hat{\omega}n + \phi)
\]
\[
A \cos[(2\pi l - \hat{\omega})n - \phi] = A \cos[2\pi l \cdot n - \hat{\omega}n - \phi] 
= A \cos(-\hat{\omega}n - \phi)
= A \cos(\hat{\omega}n + \phi) 
\] (4.11)

- Note in the second grouping the sign change in the phase

- The frequencies of (4.10) are \textit{aliases} of each other, in terms
of discrete-time frequencies
• The smallest value, $\hat{\omega} \in [0, \pi)$, is called the principal alias

• These aliased frequencies extend to sampling a continuous-time sinusoid using the fact that $\hat{\omega} = \omega T_s$ or $\omega = \hat{\omega} / T_s = \hat{\omega} f_s$, thus we may rewrite (4.10) in terms of the continuous-time frequency $\omega_0$

$$\omega_0, \omega_0 + 2\pi l f_s, 2\pi l f_s - \omega_0, l = 1, 2, 3... \quad (4.12)$$

– In terms of frequency in Hz we also have

$$f_0, f_0 + l f_s, l f_s - f_0, l = 1, 2, 3... \quad (4.13)$$

• When viewed in the continuous-time domain, this means that sampling $A \cos(2\pi f_0 t + \phi)$ with $t \rightarrow n T_s$ results in

$$A \cos[2\pi f_0(n T_s) + \phi] = A \cos[2\pi(f_0 + l f_s)(n T_s) + \phi] = A \cos[2\pi(l f_s - f_0)(n T_s) - \phi] \quad (4.14)$$

being equivalent sequences for any $n$ and any $l$

**Example:** Input a 60 Hz, 340 Hz, or 460 Hz Sinusoid with $f_s = 400$ Hz

• The analog signal is

$$x_1(t) = \cos(2\pi 60t + \pi/3)$$

$$x_2(t) = \cos(2\pi 340t - \pi/3)$$

$$x_3(t) = \cos(2\pi 460t + \pi/3)$$

• We sample $x_i(t)$, $i = 1, 2, 3$ at rate $f_s = 400$ Hz

```matlab
>> ta = 0:1/4000:2/60; % analog time axis
>> xa1 = cos(2*pi*60*ta+pi/3);
>> xa2 = cos(2*pi*340*ta-pi/3);
```
We have used (4.14) to set this example up, so we expected the sample values for all three signals to be identical. This shows that 60, 340, and 460, are aliased frequencies, when the sampling rate is 400 Hz.
– **Note**: $400 - 340 = 60$ Hz and $460 - 400 = 60$ Hz

- The discrete-time frequencies are $\omega_i = 0.3\pi, 1.7\pi, 2.3\pi$
  – **Note**: $2\pi - 1.7\pi = 0.3\pi$ rad and $2.3\pi - 2\pi = 0.3\pi$ rad

### Example: $5\cos(7.3\pi n + \pi/4)$ versus $5\cos(0.7\pi n + \pi/4)$

- To start with we need to see if either
  \[
  7.3\pi = 0.7\pi + 2\pi l
  \]
  or $7.3\pi = 2\pi l - 0.7\pi$

  for $l$ a positive integer

- Solving the first equation, we see that $l = 3.3$, which is not an integer

- Solving the second equation, we see that $l = 4$, which is an integer

What are some other valid alias frequencies?

- The phase does not agree with (4.11), so we will use MATLAB to see if $5\cos(0.7\pi n + \pi/4) \to 5\cos(0.7\pi n - \pi/4)$ to make the samples agree in a time alignment sense

\[
\begin{align*}
&\text{>> } n = 0:10; \text{ % discrete time axis} \\
&\text{>> } x1 = 5*\cos(7.3*\pi*n+\pi/4); \\
&\text{>> } x2 = 5*\cos(0.7*\pi*n+\pi/4); \\
&\text{>> } x3 = 5*\cos(0.7*\pi*n-\pi/4); \\
&\text{>> } na = 0:1/200:10; \text{ % continuous time axis} \\
&\text{>> } x1a = 5*\cos(7.3*\pi*na+\pi/4); \\
\end{align*}
\]
The Spectrum of a Discrete-Time Signal

- As alluded to in the previous example, a spectrum plot can be helpful in understanding aliasing.
- From the earlier discussion of line spectra, we know that for each real cosine at $\hat{\omega}_0$, the result is spectral lines at $\pm \hat{\omega}_0$.
- When we consider the aliased frequency possibilities for a single real cosine signal, we have spectral lines not only at $\pm \hat{\omega}_0$, but at all $\pm 2\pi l$ frequency offsets, that is...
The principal aliases occur when \( l = 0 \), as these are the only frequencies on the interval \([-\pi, \pi]\).

**Example:** \( x[n] = \cos(0.4\pi n) \)

The line spectra plot of this discrete-time sinusoid is shown below.

A particularly useful view of the alias frequencies is to consider a folded strip of paper, with folds at integer multiples of \( \pi \), with the strip representing frequencies along the \( \omega \)-axis.

All of the alias frequencies are on the same line when the paper is folded like an accordion, hence the term *folded frequencies.*
The Sampling Theorem

- The lowpass sampling theorem states that we must sample at a rate, $f_s$, at least twice that of the highest frequency in the analog signal $x(t)$. Specifically, for $x(t)$ having spectral content extending up to $B$ Hz, we must choose $f_s = 1/T_s > 2B$.

Example: Sampling with $f_s = 2000$ Hz

- When we sample an analog signal at 2000 Hz the maximum usable frequency range (positive frequencies) is 0 to $f_s/2$ Hz.
- This is a result of the sampling theorem, which says that we must sample at a rate that is twice the highest frequency to avoid aliasing; in this case 1000 Hz is that maximum.
- If the signal being sampled violates the sampling theorem, aliasing will occur (see the figure below).
• An input frequency of 1500 Hz aliases to 500 Hz, as does an input frequency of 2500 Hz
• The behavior of input frequencies being converted to principle value alias frequencies, continues as \( f \) increases
• Notice also that the discrete-time frequency axis can be displayed just below the continuous-time frequency axis, using the fact that \( \hat{\omega} = 2\pi f / f_s \text{rad} \)
• We can just as easily map from the \( \omega \)-axis back to the continuous-time frequency axis via \( f = \frac{\hat{\omega} f_s}{2\pi} \)
• Working this in MATLAB we start by writing a support function

```matlab
function f_out = prin_alias(f_in,fs)
% f_out = prin_alias(f_in,fs)
% Mark Wickert, October 2006

f_out = f_in;

for n=1:length(f_in)
    while f_out(n) > fs/2
        f_out(n) = abs(f_out(n) - fs);
    end
end
```

• We now create a frequency vector that sweeps from 0 to 2500 and assume that \( f_s = 2000 \text{Hz} \)

```matlab
>> f = 0:5:2500;
>> f_alias = prin_alias(f,2000);
>> plot(f,f)
>> hold
```
Example: Compact Disk Digital Audio

- Compact disk (CD) digital audio uses a sampling rate of \( f_s = 44.1 \text{ kHz} \)
- From the sampling theorem, this means that signals having frequency content up to 22.05 kHz can be represented
- High quality audio signal processing equipment generally has an upper frequency limit of 20 kHz
  - Musical instruments can easily produce harmonics above 20 kHz, but human’s cannot hear these signals
• The fact that aliasing occurs when the sampling theorem is violated leads us to the topic of reconstructing a signal from its samples.

• In the previous example with \( f_s = 2000 \text{Hz} \), we see that taking into account the principle alias frequency range, the usable frequency band is only \([0, 1000]\) Hz.

**Ideal Reconstruction**

• Reconstruction refers using just the samples \( x[n] = x(nT_s) \) to return to the original continuous-time signal \( x(t) \).

• *Ideal* reconstruction refers to exact reconstruction of \( x(t) \) from its samples so long as the sampling theorem is satisfied.

• In the extreme case example, this means that a sinusoid having frequency just less than \( f_s/2 \), can be reconstructed from samples taken at rate \( f_s \).

• The block diagram of an ideal discrete-to-continuous (D-to-C) converter is shown below.

\[
y[n] \xrightarrow{\text{Ideal D-to-C Converter}} y(t) \quad \text{with} \quad f_s = \frac{1}{T_s}
\]

• In very simple terms the D-to-C performs interpolation on the sample values \( y[n] \) as they are placed on the time axis at spacing \( T_s \) s.
There is an ideal interpolation function that is discussed in detail in Chapter 12 of the text.

- Consider placing the sample values directly on the time axis.

- The D-to-C places the $y[n]$ values on the time axis and then must interpolate signal waveform values in between the sequence (sample) values.

- Two very simple interpolation functions are zero-order hold and linear interpolation.

- With zero-order hold each sample value is represented as a rectangular pulse of width $T_s$ and height $y[n]$.
  - Real world digital-to-analog converters (DACs) perform this type of interpolation.

- With linear interpolation the continuous waveform values between each sample value are formed by connecting a line.
between the \( y[n] \) values

- Both cases introduce errors, so it is clear that something better must exist
- For D-to-C conversion using pulses, we can write
  \[
y(t) = \sum_{n = -\infty}^{\infty} y[n] p(t - n T_s)
  \] (4.16)
  where \( p(t) \) is a rectangular pulse of duration \( T_s \)
- A complete sampling and reconstruction system requires both a C-to-D and a D-to-C

\( x(t) \)
\[
\begin{array}{c|c|c|c|c}
Ideal & \text{Ideal} & x[n] & y[n] & y(t) \\
C-to-D & \text{C-to-D} & \text{Connection} & \text{Connection} & \text{Connection} \\
Converter & \text{Converter} & \text{DSP System} & \text{DSP System} & \text{DSP System} \\
\end{array}
\]

\( f_s \)
• With this system we can sample analog signal $x(t)$ to produce $x[n]$, and at the very least we may pass $x[n]$ directly to $y[n]$, then reconstruct the samples $y[n]$ into $y(t)$

- The DSP system that sits between the C-to-D and D-to-C, should do something useful, but as a starting point we consider how well a direct connection system does at returning $y(t) \cong x(t)$

- As long as the sampling theorem is satisfied, we expect that $y(t)$ will be close to $x(t)$ for frequency content in $x(t)$ that is less than $f_s/2$ Hz

- What if some of the signals contained in $x(t)$ do not satisfy the sampling theorem?

- Typically the C-to-D is designed to block signals above $f_s/2$ from entering the C-to-D (antialiasing filter)

- A practical D-to-C is designed to reconstruct the principle alias frequencies that span

$$\hat{\omega} \in [-\pi, \pi] \Leftrightarrow f \in [-f_s/2, f_s/2]$$

(4.17)
Spectrum View of Sampling and Reconstruction

- We now view the spectra associated with a cosine signal passing through a C-to-D/D-to-C system.
- Assume that \( x(t) = \cos(2\pi f_0 t) \)
- The sampling rate will be fixed at \( f_s = 2000 \text{ Hz} \)

\[ f_s = 2000 \text{ Hz} \]

**Spectrum of 500 Hz Cosine**

\( x(t) \)

**Spectrum of Sampled 500 Hz Cosine**

\( x[n] = y[n] \)

**Spectrum of Reconstructed 500 Hz Cosine**

\( y(t) \)

**Reconstruction Band**

\( \frac{1}{2} \)

\( \frac{1}{2} \)

\( \frac{1}{2} \)

\( \frac{1}{2} \)
We see that the 1500 Hz sinusoid is aliased to 500 Hz, and when it is output as $y(t)$, we have no idea that it arrived at the input as 1500 Hz.

What are some other inputs that will produce a 500 Hz output?

The Ideal Bandlimited Interpolation

In Chapter 12 of the text it is shown that ideal D-to-C conversion utilizes an interpolating pulse shape of the form

$$p(t) = \frac{\sin(\pi t/T_s)}{(\pi t/T_s)}, \quad -\infty < t < \infty$$  \hspace{1cm} (4.18)
• The function $\sin(\pi x)/(\pi x)$ is known as the sinc function

• Note that interpolation with this function means that all samples are required to reconstruct $y(t)$, since the extent of $p(t)$ is doubly infinite
  
  – In practice this form of reconstruction is not possible

A Mathematica animation showing that when the sinc() pulses are weighted by the sample values, delayed, and then summed, high quality reconstruction (interpolation) is possible

– The code used to create the animation

Manipulate[
  Show[Plot[\[Cos](2 \pi \phi + \phi), \{t, 0, 10\}], PlotStyle \rightarrow \{Dashing[0.01], RGBColor[1, 0, 0]\}],
  Plot[\[Sum](\[Cos](2 \pi \phi + \phi) Sinc[\pi (t - n)], \{n, 0, 10\}), \{t, 0, 10\}],
  PlotStyle \rightarrow \{Thick, RGBColor[0, 1, 0]\}], DiscretePlot[\[Cos](2 \pi \phi + \phi),
  \{n, 0, 10\}, Filling \rightarrow \text{Axis}, PlotStyle \rightarrow \{PointSize[0.02], RGBColor[1, 0, 0]\}],
  Plot[\[Table](\[Cos](2 \pi \phi + \phi) Sinc[\pi (t - n)], \{n, 0, 10\}), \{t, 0, 10\}, PlotRange \rightarrow \text{All}],
  PlotRange \rightarrow \text{All}],
  {{\phi, 1/10}, {1/2, 1/3, 1/4, 1/10, 1/20}}, {{\phi, 0}, {0, \pi / 4, \pi / 2}}]
• The final display showing an interpolated output for a single sinusoid
  – The input signal is \( x(t) = \cos(2\pi ft + \phi) \) and we assume that \( T_s = 1 \), so in sampling we let \( t \to n \)
  – With \( f = 1/4 \) (4 samples per period) and \( \phi = \pi/4 \) we have the following display:

![Diagram](image-url)