Applications of Power Spectra in Discrete Signal Analysis

Bryson Rogers
Jenna Matus

University of Southern California AME 341a
Mechoptronics

Lab Day: Wednesday October 28, 2020

Abstract

Applications of power spectra in discrete data analysis were studied. A power spectrum is a numerically represents a discrete signal by decomposing it into constituent frequencies at various intensities, or powers. The study was conducted by measuring the time traces and power spectra of various sine, square, and triangle waves using a Digital Spectrum Analyzer and various sampling settings. The power spectrum was used to identify constituent frequencies of a signal, investigate aliasing effects, and identify an unknown signal. The power spectrum complements the time trace as a tool of fundamental importance in discrete signal analysis.

Introduction

Time traces and power spectra are two means of numerically representing a signal, both of which are used extensively in digital signal acquisition and processing.

The time trace of a signal is its amplitude as a function of time. The time trace of a pure 100Hz, 1V sine wave is shown in Figure 1.

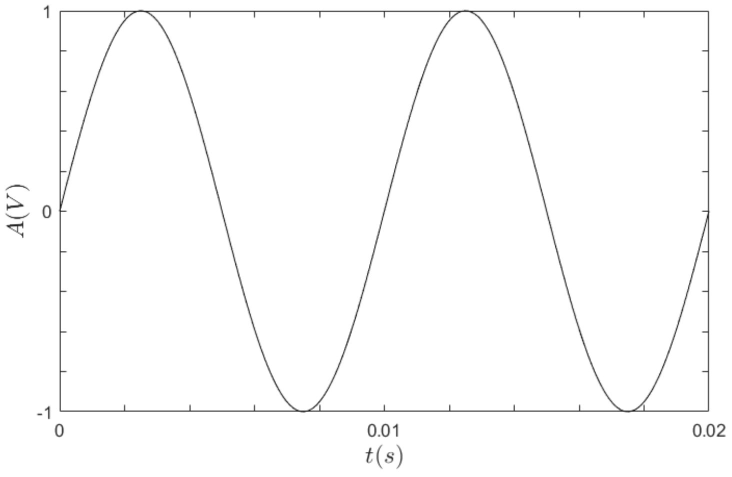


Figure 1. Amplitude of a 1V, 100Hz sine wave over time, a basic example of a time trace

A power spectrum represents a signal by decomposing it into constituent frequencies of varying intensities, or powers. Plotted in Figure 2 is the power spectrum of the same 100Hz sine wave whose time trace is plotted in Figure 1. This power spectrum has a single peak at 100Hz, because a pure sine wave is composed of a single trigonometric basis function with a single frequency, and the entirety of the power of the signal is carried at this frequency.

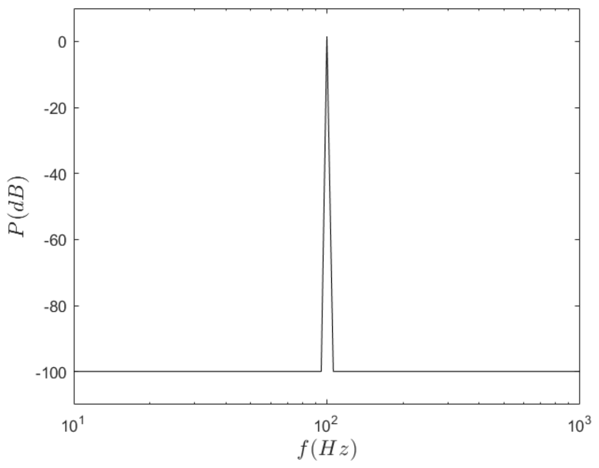


Figure 2. A sine wave is composed entirely of one frequency, and thus has one peak in its power spectrum.

The power spectrum of an electric potential signal is generated by transforming it from the time domain into the frequency domain using the Fourier Transform, defined in Equation (1)

$$P(f) = \int_{-\infty}^{\infty} e(t)\epsilon^{-2\pi i f t} dt \tag{1}$$

where $P(f)$ is power as a function of frequency (the power spectrum), $e(t)$ is signal amplitude as a function of time (the time trace), f is frequency in Hz, and ϵ is the natural logarithm base. $P(f)$ is a complex-valued function of f whose output encodes signal power and phase.

Equation (1) is applicable only if $e(t)$ is a continuous function. Because real-world data is discrete, real power spectra are generated with a modified FT, the Discrete Fourier Transform (DFT) [1], defined in Equation (2).

$$P(f) = \sum_{k=0}^{n-1} e(t)\epsilon^{-2\pi i f k/n} \tag{2}$$

where k is the index of the k^{th} datum[1], and n is the total number of data sampled. There are mathematical conditions for the existence of the FT (and DFT) of a function, but they are not of interest in this study, as the FT exists for all real-world signals **Error! Reference source not found.**.

The power spectrum is a useful tool in signal processing and analysis. It is frequently used to decompose a known or unknown signal into constituent frequencies, compactly encode signal information, and complement time traces in the investigation of signal aliasing.

Signal Decomposition

The DFT can be used to discern the frequency and power of each trigonometric basis function from which a signal is composed. Figure 1 and Figure 2 illustrated this concept with a pure sine wave, but this analysis may be applied to arbitrarily complicated signals. In this study, the DFT was used to decompose sine, square, triangle, and unknown waves into their constituent frequencies.

Integral to the usefulness of the power spectrum is its equivalence to a time trace; traces and spectra are alternate mathematical representations of the *same* signal. A power spectrum therefore does not need to be generated as a signal transpires, it can be produced from a time trace using the DFT at any time. A power spectrum can also be transformed back into a time trace using the Inverse Discrete Fourier Transform (IDFT) [1], defined in Equation (3).

$$e(t) = \frac{1}{n}\sum_{k=0}^{n-1} P(f)\epsilon^{2\pi i t k/n} \tag{3}$$

The IDFT is a powerful data processing tool because a power spectrum typically requires less digital storage space than a time trace. Compare, for example, Figure 1 and Figure 2. Many data are required to construct the 100Hz sine wave trace, depending on the resolution and duration of interest. The power spectrum requires a single datum, the 100Hz peak, which could be stored,

[1] Equation (2) assumes that the data are equally spaced in time.

transmitted, then transformed with the IDFT to reconstruct the time trace on a machine that did not measure the signal in the first place.

Aliasing

Aliasing is an effect inherent in discrete data collection, in which signal information is not entirely resolved if sampling frequency f_s is not sufficiently high compared to signal frequency f. Because aliasing is a limitation of data collection itself, it manifests in both time traces and power spectra. However, analysis of both a trace and spectrum can help identify if aliasing has occurred and to what extent.

When $f_s \gg f$, the observed shape, amplitude, and frequency match the actual signal characteristics, i.e. $A_{obs} = A$ and $f_{obs} = f$. As f_s decreases, signal resolution becomes coarser, and some shape and amplitude information is lost. At $f_s = 2f$, signal shape and amplitude are not resolved, but $f_{obs} = f$. The maximum f that can be resolved at a given f_s is dictated by the Nyquist frequency, defined in Equation (4)

$$f_{Ny} = f_s/2 \tag{4}$$

If $f_s < 2f$, such that $f > f_{Ny}$, the frequency of the signal is not resolved. When this occurs, and f_{obs} is described as “aliased”, and its value is dictated by Equation (5).

$$f_{obs} = |f_s - f| \tag{5}$$

Figure 3 provides an example of unaliased and aliased time traces; both curves were sampled from the same hypothetical 100Hz sine signal, but at different f_s. The effects of f_s on A_{obs} and f_{obs} are summarized in **Error! Reference source not found.**. The effects of aliasing on time traces and power spectra are explored in Procedure and Results.

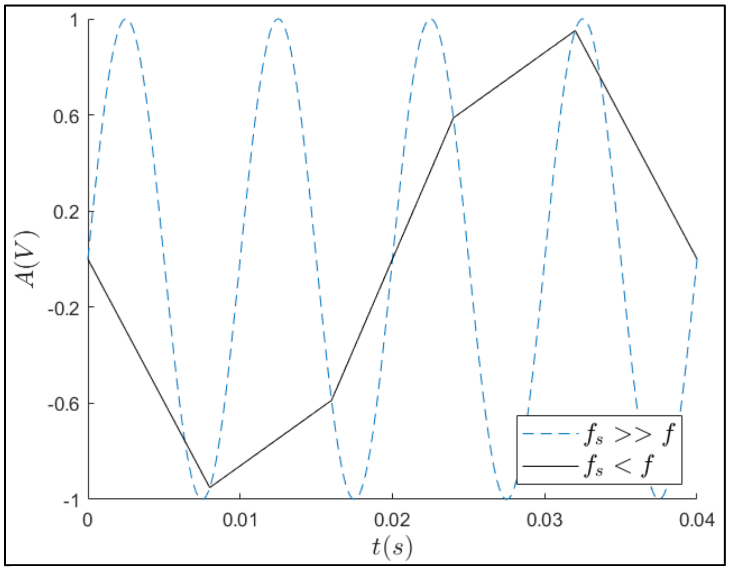


Figure 3. Two traces of the same hypothetical 100Hz sine wave yield different results depending on f_s

Table 1. Summary of aliasing effects on observed signal characteristics

	f_{obs}	A_{obs}	Shape
$f_s \gg f$	🟩 $=$	🟩 $=$	🟩 $=$
$f_s = 2f$	🟩 $=$	🟥 $\neq$	🟥 $\neq$
$f_s < 2f$	🟥 $f_{obs} = \lvert f_s - f \rvert$	🟥 $\neq$	🟥 $\neq$

Green: signal characteristic resolved
Red: signal characteristic not resolved

Signal Identification

A power spectrum isolates each constituent frequency from a signal, as discussed above. This property can be used to identify the components of a complicated signal. A single peak in the frequency domain corresponds to a sine wave in the time domain, as was shown in Figure 1 and Figure 2. Square and triangle waves also exhibit characteristic frequency domain peaks.

All functions can be represented as an infinite sum of trigonometric basis functions known as a Fourier Series[3]. The Fourier Series of ideal square and triangle waves are given in Equations (6) and (7), respectively[2]

$$e_{sq}(t) = \sum_{N=1}^{\infty} B_N \sin(2\pi f_0 N t) \qquad B_N = \frac{4}{N\pi} \tag{6}$$

$$e_{tri}(t) = \sum_{N=1}^{\infty} A_N \cos(2\pi f_0 N t) \qquad A_N = \frac{8}{N^2\pi^2} \tag{7}$$

where N is an odd index 1, 3, 5…, and f_0 is the fundamental frequency of the wave. The frequencies of the terms in the square and triangle Fourier Series are the same: f_0, $3f_0$, $5f_0$, … . The square and triangle Fourier Series differ, however, in their respective rates of decay. The coefficients in the square wave Fourier Series are inversely proportional to N ($B_N \propto 1/N$), whereas the triangle wave coefficients are inversely proportional to N^2, ($A_N \propto 1/N^2$).

Each term in a Fourier Series is a trigonometric basis function, which appears as a peak in the frequency domain. The power spectra of square and triangle waves thus consist of a series of peaks at f_0, $3f_0$, $5f_0$… . The rate of power decay of these peaks is determined by the coefficients of the terms in each wave's Fourier Series. Square wave peaks decay as $P_{sq}(Nf_0) \propto 1/N$, triangle peaks decay as $P_{tri}(Nf_0) \propto 1/N^2$.

Sine, square, and triangle waves each exhibit the characteristic peak behavior described above in power spectra. These signals can be identified as components of an unknown signal via measurement of power spectrum peaks.

Procedure

A function generator (FG) was attached across Channel 1 of a digital spectrum analyzer (DSA). An unknown electric potential signal was attached across Channel 2 of the DSA. The DSA was used to measure time traces and generate power spectra for various signals in order to investigate the characteristics of power spectra discussed in Introduction.

I. Signal Decomposition

[2] For derivations of Equations (6) and (7), see [4] and **Error! Reference source not found.**

A sine wave, square wave, and triangle wave[3] were produced with the FG. Each wave had frequency 100Hz and amplitude 2 V_{RMS}. The time trace and power spectrum of each wave were produced with the DSA. The frequencies of each wave's time trace and spectral peak(s) were compared with expected values. The frequency of each time trace, f_{trace}, was calculated with Equation (8)

$$f_{trace} = \frac{1}{t_{max2} - t_{max1}} \tag{8}$$

where t_{max1} and t_{max2} are the times at which the first and second local maxima occur in the trace. Δt_{max1},$\Delta t_{max2} \cong 0$, so Δf_{trace} was assumed to be 0. The frequency of each spectrum peak, $f_p \pm \Delta f_p$, was calculated with Equations (9) and (10)

$$f_p = \frac{f_p{}' + f_p{}''}{2} \tag{9}$$

$$\Delta f_p = \frac{|f_p{}' - f_p{}''|}{2} \tag{10}$$

where $f_p{}'$ and $f_p{}''$ are the highest-power frequencies at the peak of interest.

Results I

The time traces and power spectra of the 100Hz 2V_{RMS} sine, square, and triangle waves are presented in Figure 4.

The time trace frequency f_{trace} of each wave was 100Hz, equal to the input setting of the FG. The frequency of the first spectrum peak, $f_{p1} \pm \Delta f_{p1}$, was equal to f_{trace} for each signal. f_{p1} corresponds to the fundamental frequency of each signal f_0 in Equations (6) and (7).

The sine spectrum had only one peak, at f_{p1}, which was expected because the signal itself was a trigonometric basis function. The smaller "peaks" to the right of f_{p1} in the sine spectrum are noise, not peaks. The power of the second highest "peak" is roughly $7 * 10^{-8} P(f_{p1})$, and is thus insignificant.

The square and triangle spectra had many peaks beyond f_{p1}, which was expected due to their Fourier Series having infinite terms. The first four peak frequencies of each spectrum are presented in Table 2. Frequencies of first several peaks in each power spectrum.

[3] Square wave had a 50% duty cycle, and triangle wave had 50% symmetry.

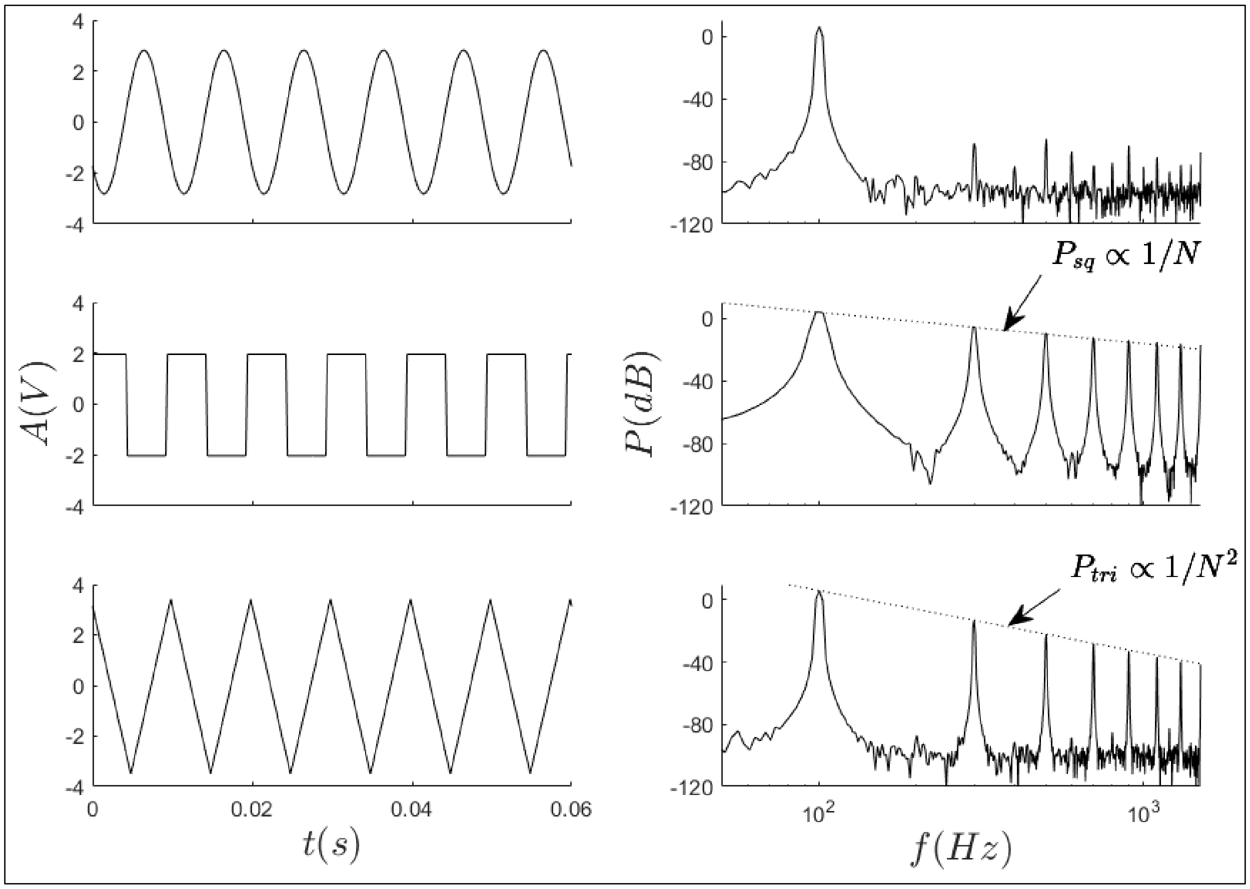


Figure 4. Time traces (left) and power spectra (right) of (from top to bottom) sine, square, and triangle waves. Power vs. frequency relationships predicted by square and triangle wave Fourier Series are plotted for comparison with measured results.

Table 2. Frequencies of first several peaks in each power spectrum

	f_{p1}	Δf_{p1}	f_{p3}	Δf_{p3}	f_{p5}	Δf_{p5}	f_{p7}	Δf_{p7}
	[Hz]		[Hz]		[Hz]		[Hz]	
Sine	99	1	N/A		N/A		N/A	
Square	100	2	300	2	500	2	701	2
Triangle	99	1	299	1	499	1	699	1

Table 2 shows that $f_{pN} \pm \Delta f_{pN} = N f_{p1}$ for the square and triangle spectra. This relationship between f_{pN} and the fundamental frequency f_{p1} was predicted by the Fourier Series of ideal square and triangle waves. This relationship held true for all $N \leq 23$, beyond which frequency data was not collected.

The power of each peak in the square and triangle spectra agreed with each wave's respective Fourier Series. The dotted lines plotted over the square and triangle spectra in Figure 4 are anchored at (f_{p1}, P_{p1}), and decay proportionally to $1/N$ and $1/N^2$, as predicted for ideal square and triangle waves, respectively.

II. Aliasing

Aliasing occurs when $f_s < 2f$, as discussed in introduction. Aliasing prevents signal shape and amplitude from being resolved and causes $f_{obs} \neq f$. To investigate the effects of aliasing on time traces and power spectra, a 1500Hz 2V$_{\mathrm{RMS}}$ sine wave was produced with the FG. Its time trace and power spectra were observed first at $f_s = 40$kHz, such that $f_s \gg f$. The trace and spectrum of the same signal were then observed at decreasing values of f_s: 20kHz, 10kHz, 5000Hz, 2500Hz, and 1500Hz. At each value of f_s, trace and spectrum characteristics were observed and f_{obs} was compared to f.

Results II

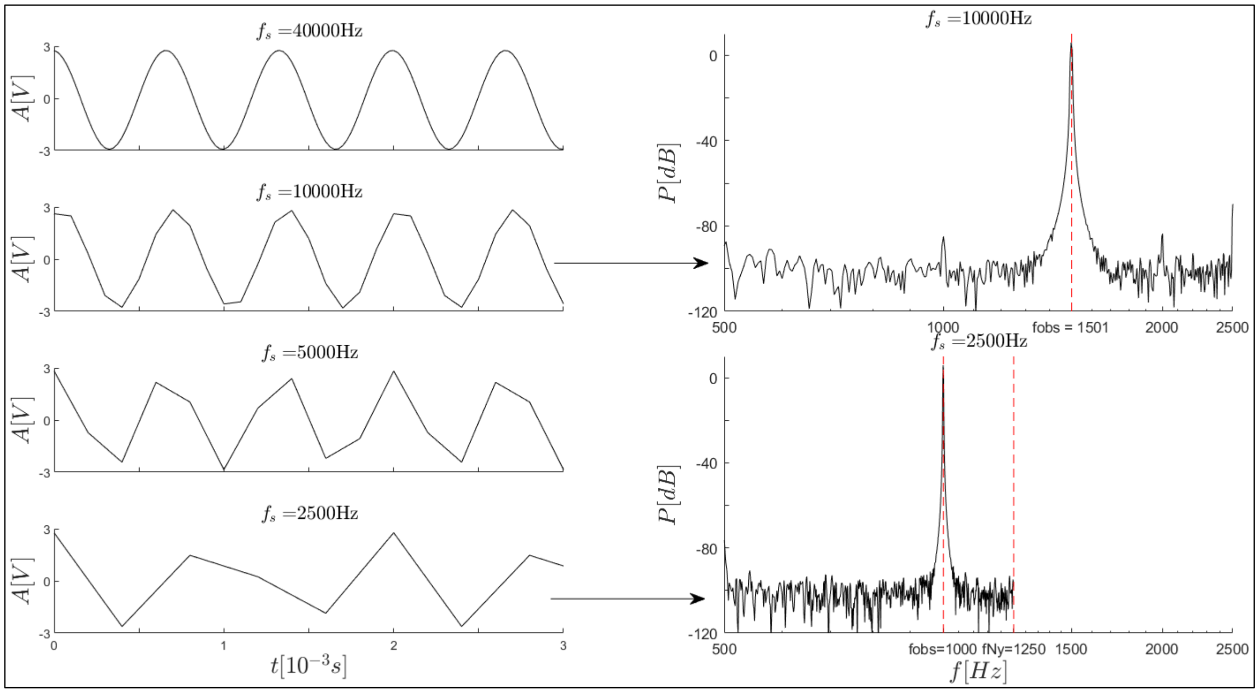


Figure 5. Demonstration of the effects f_s on time traces (left) and power spectra (right) of a 1500Hz sine wave

Traces and spectra generated at various values of f_s for the 1500Hz sine wave were plotted in Figure 5. The peak frequency was labeled in both spectra, and the Nyquist frequency $f_{Ny} = f_s/2$ was labeled in the $f_s = 2500$Hz spectrum.

The time traces in Figure 5 became coarser as f_s decreased. At $f_s = 40$kHz, signal shape and amplitude A are fully resolved and $f_{obs} = f$. At $f_s = 10$kHz, signal shape is largely retained, but A is not fully resolved at signal peaks. At $f_s = 5$kHz and 2.5kHz, the observed signal is too coarse for signal shape or A to be resolved.

$f_{obs} \pm \Delta f_{obs} = f$ for all $f_s \geq$ 5kHz. This was expected, because 5kHz $> f_{Ny}$ (for the 1500Hz sine wave, $f_{Ny} =$ 3000Hz). In the $f_s =$ 10kHz power spectrum in Figure 5, the peak frequency $f_{obs} = 1501 \pm 2$Hz $= f$, as expected[4].

At $f_s = 2500$Hz, the observed frequency was aliased: $f_{obs} = 1000 \pm 1$Hz $\neq f$. As predicted by Equation (5), $f_{obs} = |f_s - f|$. The $f_s = 2500$Hz spectrum in Figure 5 contains no information for $f > f_{Ny} =$ 1250Hz because all signals with frequencies greater than the Nyquist frequency are aliased back into the observable spectrum, where $f_{obs} < f_{Ny}$.

It is impossible to determine if a signal has been aliased given a single time trace or power spectrum. However, multiple traces and spectra at various f_s can be used to identify which frequencies have and have not been aliased. If f_s is changed and f_{obs} of a spectrum peak changes, that frequency is aliased.

III. Unknown Signal Identification

An electric potential signal with unknown frequency and amplitude was connected across Channel 2 of the DSA. A time trace and two power spectra of the signal were produced with the DSA at $f_s =$ 30kHz, 2kHz, and 1kHz, respectively. The findings of I.Signal Decomposition and II.Aliasing were used to identify the signals of which the unknown signal was composed.

Results III

The unknown signal was determined to be the sum of a 44 ± 1Hz sine wave and a 65 ± 1Hz triangle wave.

The time trace and power spectrum of the unknown signal are presented in Figure 6. The time trace shape indicated that the mystery signal was neither a pure sine, square, nor triangle wave. Equivalently, the signal power spectrum did not correspond to the pure sine, square, or triangle spectrum in Figure 4.

The first five peak frequencies, of which f_{p1} and f_{p2} are indicated on the spectrum in Figure 6, are presented in

Table 3.

[4] $f_{obs} \pm \Delta f_{obs}$ were determined with the same method as $f_p \pm \Delta f_p$, using Equations (9) and (10)

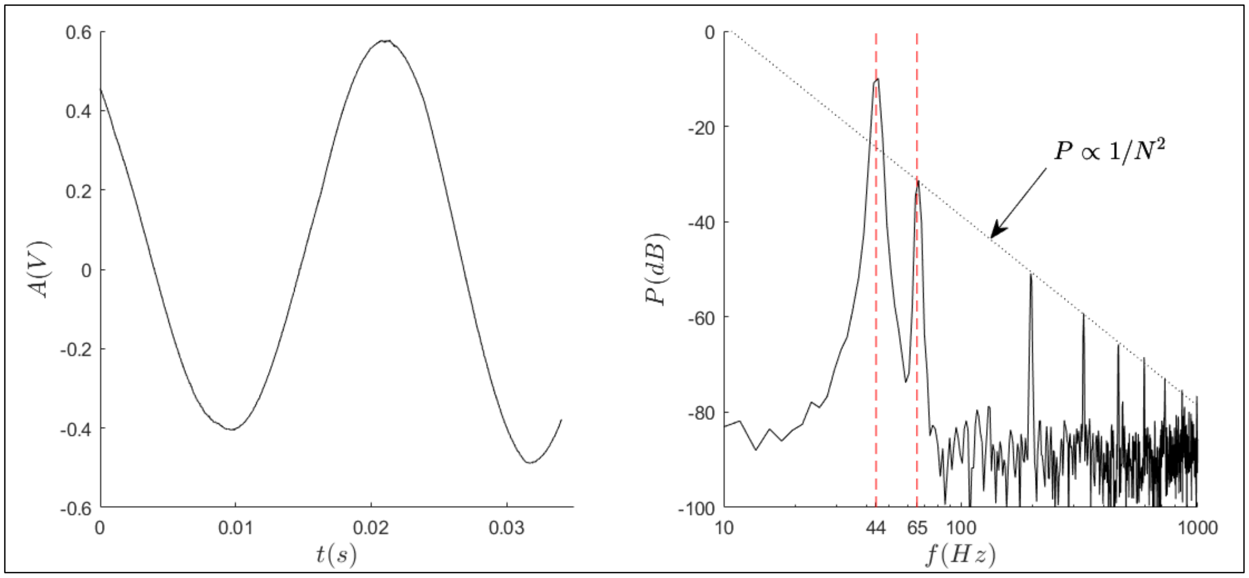
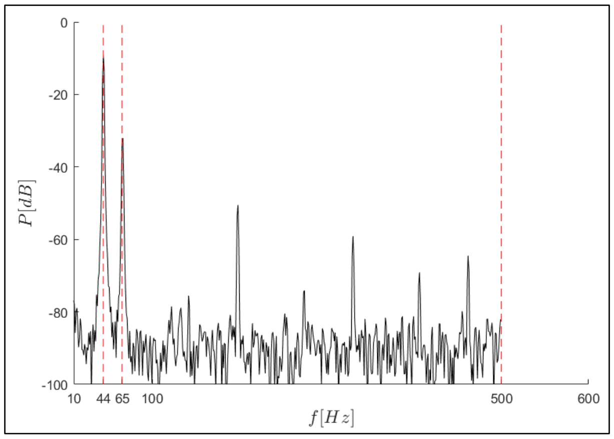


Figure 6. An unknown signal measured with respect to time (left) and decomposed into is constituent frequencies (right)

Table 3. First five frequencies at which peaks occur in the unknown signal spectrum

f_{p1} Δf_{p1} [Hz]	f_{p2} Δf_{p2} [Hz]	f_{p3} Δf_{p3} [Hz]	f_{p4} Δf_{p4} [Hz]	f_{p5} Δf_{p5} [Hz]
44 1	65 1	198 1	329 1	463 1

The spectrum at $f_s = 1$kHz, plotted in Figure 7, was used to identify aliased frequencies. Peaks with greater frequency than $f_{Ny} = 500$Hz were shifted with respect to the $f_s = 1$kHz peaks. The peak frequencies listed in Table 3 did not shift, and are therefore not aliased at $f_s = 2$kHz.



When divided by f_{p2}, the values of f_{p3}, f_{p4}, and f_{p5} become 3.05 ± 0.05, 5.06 ± 0.08, and 7.1 ± 0.1, respectively. This distribution of peak frequencies is equal to the $f_{pN} = Nf_0$ frequency distribution predicted for ideal square and triangle waves, as discussed in Introduction.

Figure 7. Unknown signal sampled at $f_s =$ 1kHz to cross-check $f_s = 2$kHz spectrum for aliased peaks

Plotted over the spectrum in Figure 6 is a curve $P \propto 1/N^2$, anchored at $(f_{p2}, P(f_{p2}))$, where $N = 1, 3, 5\ldots$ corresponds to $f_{p2}, f_{p3}, f_{p4}\ldots$. This curve is the expected rate of peak decay for a triangle wave of fundamental frequency $f_0 = f_{p2} =$ 65Hz. The power of each peak beyond f_{p2} was approximately $\propto\ 1/N^2$: the discrepancy between measured and predicted peak power at f_{p3} and f_{p4} was 7% and 3%, respectively[5]. The frequency and power observations suggest that the peaks $f_p \geq f_{p2}$ were due to a 65Hz triangle wave in the unknown input signal.

No observations suggested that the first peak in Figure 6, $f_{p1} = 44 \pm 1$Hz, was associated with any other peaks, as the triangle waves peaks did. Therefore, a 44Hz sine wave was present in the input signal.

Conclusion

This study discussed the power spectrum as a means of discrete signal representation, some fundamental properties of power spectra, and applications in discrete signal analysis. The time trace and power spectrum can equivalently be used to represent a single signal, but the power spectrum reveals frequency and power information that the time trace cannot. As seen in I.Signal Decomposition, a power spectrum can be used to identify and/or isolate trigonometric basis functions that make up a periodic signal. Similarly, in III.Signal Identification, the power spectrum was used to identify constituent signals within a composite signal. In II.Aliasing, power spectra were used to determine whether the frequency of a signal had been aliased.

Each task performed in this study was fundamental to discrete data collection and analysis, yet would have been computationally and visually cumbersome without the use of power spectra. Power spectra are thus a fundamentally important tool in discrete data analysis.

[5] These percent discrepancy values were found via the difference between the $P(f)$ of an ideal triangle wave and actual measured $P(f_p)$ values

Works Cited

[1] Weisstein, Eric W. "Discrete Fourier Transform." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/DiscreteFourierTransform.html

[2] "Existence of Fourier Transform", *ccrma.stanford.edu*, 2020. [Online]. Available: [2] https://ccrma.stanford.edu/~jos/mdft/Existence_Fourier_Transform.html. [Accessed: 11-Nov- 2020].

[3] Weisstein, Eric W. "Fourier Series." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/FourierSeries.html

[4] Weisstein, Eric W. "Fourier Series--Square Wave." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/FourierSeriesSquareWave.html

[5] "Fourier Series--Triangle Wave -- from Wolfram MathWorld", *Mathworld.wolfram.com*, 2020. [Online]. Available: https://mathworld.wolfram.com/FourierSeriesTriangleWave.html. [Accessed: 11- Nov- 2020].

Appendix: Matlab Script

```
% AME341a Mechoptronics
% E7
% Bryson Rogers
clear; clc; close all;
% NOTE file naming conventions:
% shape_domain(time or freq)_f(Hz)_VRMS(V)_n_fs(Hz).txt
% mystery_domain(time or freq)_n_fs(Hz).txt

%% Sample Plots
clear;clc;close all
% 100Hz Sine Trace
x = linspace(0,0.02,500);
y = sin(2*pi*100*x);
plot(x,y,'k')

xticklabels({0,'','','','',0.01,'','','','',0.02});
xlabel('$t(s)$','interpreter','latex','FontSize',14);
yticklabels({-1,'','','','',0,'','','','',1});
ylabel('$A(V)$','interpreter','latex','FontSize',14);

% 100Hz sine spectrum
xf = logspace(1,3,100);
xf(50:52) = [95,100,106];
yf = -100*ones(1,length(xf));
yf( xf==100 ) = 1.5;
semilogx(xf,yf,'k');

xlabel('$f(Hz)$','interpreter','latex','FontSize',14);
ylim([-110 10]);
ylabel('$P(dB)$','interpreter','latex','FontSize',14)
%yticklabels({-100,'',-60,'',-20,'',20})

% 100Hz Sine Trace
figure
x = linspace(0,0.04,1000);
xAlias = linspace(0,0.04,6);
y = sin(2*pi*100*x);
yAlias = sin(2*pi*100*xAlias);

hold on
plot(x,y,'--')
plot(xAlias,yAlias,'k')
```

```
hold off
xticks(0:0.01:0.04); yticks(-1:0.4:1)
xlabel('$t(s)$','interpreter','latex','FontSize',14);
ylabel('$A(V)$','interpreter','latex','FontSize',14);
legend({'$f_s>>f$','$f_s<f$'},'interpreter','latex','Locati
on','southeast','FontSize',14)

%% Basic Sine, Square, Triangle Plots
shapes = {'sine','square','tri'};
nSpec = [2048,1024,2048];
nTrace = [2048,1024,512];
figure

for i=1:length(shapes)
    % Retrieve the power spectrum data for each wave shape
    fileName =
fullfile('E7_Raw_Data',[shapes{i},'_freq_100_2_',num2str(nS
pec(i)),'_5000.txt']);
    shapeSpec = dlmread(fileName,'\t',2,0);

    % Determine Peak Frequencies
    [peaks,peakIndices] = findpeaks(shapeSpec(:,2));
    peakFilter = peaks>-60;
    peakIndices = peakIndices( peakFilter );
    peaks = peaks( peakFilter ); % Consider only peaks of
significant power
    peakFreq = shapeSpec(peakIndices,1);

    for peaki = 1:length(peaks)
        neighborPower = [shapeSpec(peakIndices(peaki)+1,2),
shapeSpec(peakIndices(peaki)-1,2)];
        neighborIndex = shapeSpec(:,2) ==
max(neighborPower);
        fpAB = [peakFreq(peaki),
shapeSpec(neighborIndex,1)];
        fp(i,peaki) = sum(fpAB)/2;
        dfp(i,peaki) = abs( fpAB(2) - fpAB(1) )/2;
    end

    % Plot each spectrum in a subplot
    subplot(3,2,2*i)
    semilogx(shapeSpec(:,1),shapeSpec(:,2),'k');
    % Plot fourier series lines for square and triangle
peaks
    if i==2 || i==3
        hold on
        fRef = logspace(1,4,200);
        PRefDb = zeros(1,length(fRef));
        for j=1:length(fRef)
            if i==2
                PRefDb(j) = peaks(1) + 20*log10(
peakFreq(1) / fRef(j));
            elseif i==3
                PRefDb(j) = peaks(1) + 20*log10(
(peakFreq(1) / fRef(j))^2 );
            end
        end
        semilogx(fRef,PRefDb,'k:')
        hold off
    end

    % Format Power Spectra
    box off
    labelSq = annotation('textarrow',[0.79375
0.767708333333333],[0.654982905982906 0.595085470085547],...
        'String','$P_{sq}\propto 1/N$
','interpreter','latex','FontSize',14);
    labelTri = annotation('textarrow',[0.806770833333333
0.770833333333333],[0.320581196581197
0.282051282051282],...
        'String','$P_{tri}\propto
1/N^2$','interpreter','latex','FontSize',14);
    xlim([50 1.5*10^3]); xticklabels('');
    ylim([-120 10]); yticks(-120:40:0);
yticklabels('auto');
    if i==2

ylabel('$P(dB)$','interpreter','latex','FontSize',16);
    elseif i==3

xlabel('$f(Hz)$','interpreter','latex','FontSize',16);
        xticklabels('auto')
    end

    % Load in time trace data
    subplot(3,2,2*i-1)
    fileName =
fullfile('E7_Raw_Data',[shapes{i},'_time_100_2_',num2str(nT
race(i)),'_5000.txt']);
    shapeTrace = dlmread(fileName,'\t',2,0);

    % Determine trace frequencies
    [tracePeaks, traceIndices] =
findpeaks(shapeTrace(:,2));
    tMax2 = shapeTrace(traceIndices(2),1); tMax1 =
shapeTrace(traceIndices(1),1);
    shapePeriod(i) = tMax2 - tMax1;
    if i==2
        shapePeriod(i) = 0.0142 - 0.0042;
    end
    fTrace(i) = 1/shapePeriod(i);

    % Plot Time traces
    plot(shapeTrace(:,1),shapeTrace(:,2),'k')

    % Format traces
    box off
    xlim([0 0.06]); xticks(0:0.02:0.06); xticklabels('');
    if i == 3
        xticklabels(0:0.02:0.06)

xlabel('$t(s)$','interpreter','latex','FontSize',16);

    end
    ylim([-4 4]); yticks(-4:2:4); yticklabels('auto');
    if i==2

ylabel('$A(V)$','interpreter','latex','FontSize',16);
    end
end

%% Aliasing
fs = [40000,10000,5000,2500];
n = [256,128,128,128];

figure
for i = 1:length(fs)
    % Retrieve data of interest from its .txt file
    fileName =
fullfile('E7_Raw_Data',['sine_time_1500_2_',num2str(n(i)),'
_',num2str(fs(i)),'.txt']);
    aliasData = dlmread(fileName,'\t',2,0);

    % Align each trace on first peak
    [aliasTracePeaks,aliasTracePeaksIndices] =
findpeaks(aliasData(:,2));
    peak1 = aliasTracePeaksIndices(1);
    shiftedAliasData = aliasData(peak1:end,2);
    shiftedAliasData = [shiftedAliasData;
zeros(length(aliasData(:,1)) -
length(shiftedAliasData),1)];

    % Plot each time trace in a subplot
    subplot(4,2,2*i-1);
    plot(aliasData(:,1),shiftedAliasData,'k');

    % Format the traces
    box off
    ylim([-3,3]); yticks([-3,0,3]); yticklabels([-3,0,3]);
ylabel('$A[V]$','FontSize',13,'interpreter','latex','FontSi
ze',16);
    title({['$f_s =
$',num2str(fs(i)),'Hz']},'FontSize',13,'interpreter','latex
');
    xlim([0 3e-3]); xticks(10^-3*[0:0.5:5]);
xticklabels('');
    if i==4
        xticklabels({'0' ''
'1','','2','','3','','4','','5'});
        xlabel('$t[10^{-3}
s]$','FontSize',16,'interpreter','latex');
    end

    % Plot spectra for fs = 10kHz and 2.5kHz

    if i==2 || i==4
        fileName =
fullfile('E7_Raw_Data',['sine_freq_1500_2_2048_',num2str(fs
(i)),'.txt']);
        aliasData = dlmread(fileName,'\t',2,0);

        subplot(4,2,[2*(i-1),2*i])
        semilogx(aliasData(:,1),aliasData(:,2),'k');

        % determine peak freq
        maxInd = aliasData(:,2) == max(aliasData(:,2));
        neighborPower = [aliasData(find(maxInd)-1,2)
,aliasData(find(maxInd)+1,2)];
        fABInd = aliasData(:,2) == max(neighborPower);
        fAB = [aliasData(find(maxInd),1),
aliasData(find(fABInd),1)];
        fpAlias(i) = mean(fAB);
        dfpAlias(i) = abs(fAB(1) - fAB(2))/2;

        hold on
        plot([fpAlias(i), fpAlias(i)], [-150, 100],'r--');

        % annotate nyquist frequency
        if i==4
            plot([1250, 1250], [-150, 100],'r--');
        end
        hold off

        % Format spectra
        box off
        title({['$f_s =
$',num2str(fs(i)),'Hz']},'FontSize',13,'interpreter','latex
');
        xlim([500 2.5e3]); xticks('auto');
        ylim([-120 10]); yticks(-120:20:140);
yticklabels({-120,'',-80,'',-40,'',0});
ylabel('$P[dB]$','interpreter','latex','FontSize',16)
        if i==2
            xticklabels({500,1000,['fobs =
',num2str(round(fpAlias(i),0))],2000,2500,3000})
        elseif i == 4

xlabel('$f[Hz]$','interpreter','latex','FontSize',16);
            xticks([500,1000,1250,1500,2000,2500,3000]);

xticklabels({500,['fobs=',num2str(round(fpAlias(i),0))],'fN
y=1250',1500,2000,2500,3000})
            hold on
        end
    end
    % add arrows
    annotation('arrow',[0.45875 0.56],[0.613598673300166
0.613598673300166]);
    annotation('arrow',[0.456458333333333
0.557708333333333],[0.157131011608624 0.157131011608624]);
end

%% Plot mystery trace and spectrum
figure
domain = {'time','freq','freq'};
n = [1024,1024,1024];
fs = [30000,2000,1000];

for i = 1:length(domain)
    % Retrieve mystery signal time trace and power spectrum
data
    fileName =
fullfile('E7_Raw_Data',['mystery_',domain{i},'_',num2str(n(
i)),'_',num2str(fs(i)),'.txt']);
    mystery = dlmread(fileName,'\t',2,0);
    mysX = mystery(:,1);
    mysY = mystery(:,2);

    % Plot mystery signal time trace and power spectrum
    if i==1
        subplot(1,2,i)
        plot(mysX,mysY,'k');

        box off

xlabel('$t[s]$','interpreter','latex','FontSize',14);

ylabel('$A[V]$','interpreter','latex','FontSize',14);
        xticklabels(0:0.01:0.04);

    elseif i==2
        subplot(1,2,i)
        semilogx(mysX,mysY,'k')

        % Determine Peak Frequencies
        [peaks,peakIndices] = findpeaks(mysY);
        peakFilter = peaks>-60;
        peakIndices = peakIndices( peakFilter );
        peaks = peaks( peakFilter ); % Consider only peaks
of significant power
        peakFreq = mysX(peakIndices);

        for peaki = 1:length(peaks)
            neighborPower = [mysY(peakIndices(peaki)+1),
mysY(peakIndices(peaki)-1)];
            neighborIndex = mysY == max(neighborPower);
            mysfpAB = [peakFreq(peaki),
mysX(neighborIndex)];
            mysfp(i,peaki) = sum(mysfpAB)/2;
            mysdfp(i,peaki) = abs( mysfpAB(2) - mysfpAB(1)
)/2;
        end

        % Triangle reference line
        hold on
        fRef = logspace(1,4,200);
        PRefDb = zeros(1,length(fRef));
        for j=1:length(fRef)
            PRefDb(j) = peaks(2) + 20*log10( (mysfp(2,2) /
fRef(j))^2 );
        end
        semilogx(fRef,PRefDb,'k:')

        box off
        xlim([10 1000]); ylim([-100 0]);
        xticks([10,mysfp(2,1),mysfp(2,2),100,1000]);
xticklabels({10,44,65,100,1000});
        yticks(-100:20:0); yticklabels(-100:20:0)

xlabel('$f[Hz]$','interpreter','latex','FontSize',14);

ylabel('$P[dB]$','interpreter','latex','FontSize',14);

        plot([mysfp(2,1), mysfp(2,1)],[-200,200],'r--');
        plot([mysfp(2,2), mysfp(2,2)],[-200,200],'r--');
        hold off

        annotation('textarrow',[0.797916666666667
0.75875],[0.691371475953566 0.571310116086236],...
        'String','$P\propto 1/N^2 $
','interpreter','latex','FontSize',14);
```

```
    elseif i==3
        figure
        hold on
        semilogx(mysX,mysY,'k')
        % Determine Peak Frequencies
        [peaks,peakIndices] = findpeaks(mysY);
        peakFilter = peaks>-60;
        peakIndices = peakIndices( peakFilter );
        peaks = peaks( peakFilter ); % Consider only peaks
of significant power
        peakFreq = mysX(peakIndices);

        for peaki = 1:length(peaks)
            neighborPower = [mysY(peakIndices(peaki)+1),
mysY(peakIndices(peaki)-1)];
            neighborIndex = mysY == max(neighborPower);
            mysfpAB = [peakFreq(peaki),
mysX(neighborIndex)];
            mysfp(i,peaki) = sum(mysfpAB)/2;
            mysdfp(i,peaki) = abs( mysfpAB(2) - mysfpAB(1)
)/2;
        end

        % Triangle reference line
        fRef = logspace(1,4,200);
        PRefDb = zeros(1,length(fRef));
        for j=1:length(fRef)
            PRefDb(j) = peaks(2) + 20*log10( (mysfp(2,2) /
fRef(j))^2 );
        end
        semilogx(fRef,PRefDb,'k:')

        box off
        xlim([10 600]); ylim([-100 0]);
        xticks([10,mysfp(2,1),mysfp(2,2),100,500,600]);
xticklabels({10,44,65,100,500,600});
        yticks(-100:20:0); yticklabels(-100:20:0)

xlabel('$f[Hz]$','interpreter','latex','FontSize',14);

ylabel('$P[dB]$','interpreter','latex','FontSize',14);
%          plot([mysfp(2,1), mysfp(2,1)],[-200,200],'r--');
%          plot([mysfp(2,2), mysfp(2,2)],[-200,200],'r--');
%          plot([500,500],[-200,200],'r--');
```

```
        hold off
    end


end
```