Communication Systems

Calculating Signal Power

Farnam Adelkhani

Abstract

The objective of digital signal processing is to represent analog data as an approximated digital version or visa-versa. The potential advantages of converting an analog signal to a digital form that is close enough for use are that noise in the signal may be removed, the amplitude of the signal can be varied and the signal can be coded amongst other advantages to using digital. The idea is that using digital data allows the user to transmit and receive signals accurately, easily and reliably. The key is to sample the frequency at an appropriate rate when dealing with a continuous function with known amplitude, frequency and phase.

Introduction

The objective of this communication systems project is to make a variety of calculations to find the average signal power of a continuous random signal by different methods and comparing the results. The continuous signal is given as:

$$x(t) = Acos(\omega_0 t + \varphi) + n \quad \text{for } 0 \leq t \leq 8s,$$

The calculations for average power will be made using various methods and all the values should be the same. The fact that all the calculated values are the same regardless of the method used would prove the underlying theoretical fact that the average signal power found by means of autocorrelation function, statistical variance or integral of power spectral density function all lead to the same value of average power.

Sampling theorem; the concepts of aliasing and the Nyquist theorem will be applied. We can avoid aliasing by using the sampling rate specified by the Nyquist theorem which says we must sample at twice the maximum frequency.

Background Theory

Old tapes and vinyl are storing an analog representation of how the air moves over time. Now, if we want to do something more complicated like grabbing that sound and storing it digitally on a computer, ipod or whatever electronic device have you. There is just no way of capturing the entire signal and saving it exactly as is if you are familiar with how computers work and binary. The solution is that we will have to discritize the signal first and then save the discrete samples; this technique is essentially sampling. We are essentially storing the amplitude of the wave at a particular time interval.

Creating summations of sin waves we can regenerate desired signals of any desired shape and amplitude. For example a square pulse can be generated using an idealized sin wave along with a dc component. Human speech is audible in a particular range which is around 20Hz - 20kHz so what would be the point of sampling anything outside of that audible range? These are the ideas this project is set around.

Although aliasing can occur, considering two signals at frequency f and f + 1/T we can apply trigonometry and find that the sampled version of the two signals will be exactly the same:

$$\cos\left(2\pi\left(f+\frac{1}{T}\right)kT\right) = \cos\left(2\pi k + 2\pi\, f\, k\, T\right) = \cos\left(2\pi\, f\, k\, T\right)$$

Procedure

The sample code below was given as a foundation to build upon with given variables in the parameter table.

```
Fs = 1000; % Sampling frequency
Ts = 1/Fs; % Sampling period
Wo=100*pi  % Signal Frequency
Ph= pi/2   % Signal Phase
t = (0:Ts:8); % Time axe
L = size(t,2); % Total Number of Sample
Sn=2;      % Standard Deviation of AWGN
A=2;        % Amplitude
x = Sn*randn(size(t))+ A*cos(Wo*t+Ph);
```

Parameter	Value
A	2
ω_0	100π
φ	$\pi/2$
σ_n	2

The following lines complete the sequence to plot the two-sided PSD for x(t):

```
nfft = 2^nextpow2(length(x));
Pxx = abs(fft(x,nfft)).^2/length(x)/Fs;
Hpsd = dspdata.psd(Pxx,'Fs',Fs,'SpectrumType','twosided')
figure;
plot(Hpsd);
axis([0 Fs-Fs/L -60 10]);
```

- Since FFT is faster if it is an exponential of two: '2^nextpow2' to generate more points and increase the spectral resolution. This could have been further increased by a greater power.

- The periodogram function is represented by 'pxx' in matlab, with output arguments present to plot the periodogram PSD estimate.

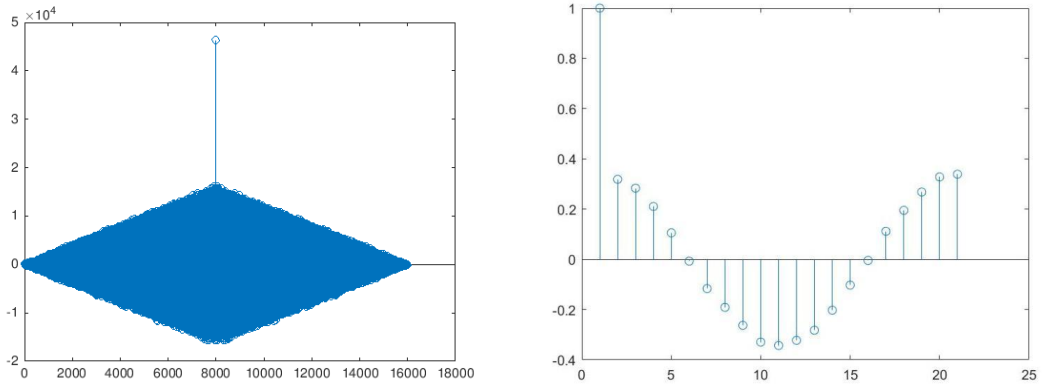
- Hpsd is the power spectral density function used to compute the average power in a signal over the frequency band.

The directions for part 2 are given below:

Calculate and plot the autocorrelation function $R_x(\tau)$ (Note: the plot should be a symmetrical function with respect to the origin)

... the user is required to plot the autocorrelation function for the continuous random signal in matlab.

My first sense in this section was to use the 'autocorr' function but after doing some research I found that the student version of matlab does not include the necessary toolbar so I downloaded a trial version that did offer that piece of software and then implemented the autocorrelation plot two ways:



The first plot was generated using the 'xcorr' function and the second plot was generated using the 'autocorr' matlab function.

Part 3 and 4

- For part 3 and then used again in part 4 of the project, the 'bandpower' function is used to calculate the average power and display it into the command window. As required, the 'var' command is also applied in order to calculate the variance.

- Part 4 utalized the functions used in part 1-3 in order to create a series of data plots that can be discussed in the appropriate sections of this technical report.

Results and Discussion of the Results

This section of the report will focus on critical discussions about outcomes and variations within the resulting data.

In part 3 the project calls for the average power of x(t) to be calculated and shown to be the same value whether using the 'bandpower()' matlab function or calculating using the variance by means of the 'var()' matlab function. The Average power and variance were found to be:

```
p =

    5.8116

v =

  5.8115
```

.... which are the same value.

The table below is of the results for calculating average power by three different means; all of which arrive at the same value. Using the PSD function the value for average power was found to be the same as the value calculated using the variance.

Average Power, Statistical Variance and integral of PSD:

$w_o = 500\pi$	$w_o = 1000\pi$	$w_o = 1500\pi$
Wo =1.5708e+03	Wo =3.1416e+03	Wo =4.7124e+03
Ph =1.5708	Ph =1.5708	Ph =1.5708
p =1.9998	p =1.1349e-23	p =1.9998
v = 2	v =1.1350e-23	v = 2
q=2	q=1.1350e-23	q=2

** p = average power ** v = variance

Looking at the above output we can see that at 1000π the values for average power are different from what is expected. If the frequency of the signal is modified just slightly to 999π or 1001π, the power output is again the values theoretically expected. We need to start thinking about the Nyquist theorem being fulfilled and in this case it helps to visualize why and how the issue at 1000π is occurring if we take into account the period being sampled.

The problem here is that power should not have anything to do with frequency but in this situation, the ▲ in frequency is disturbing the total average power calculation. We

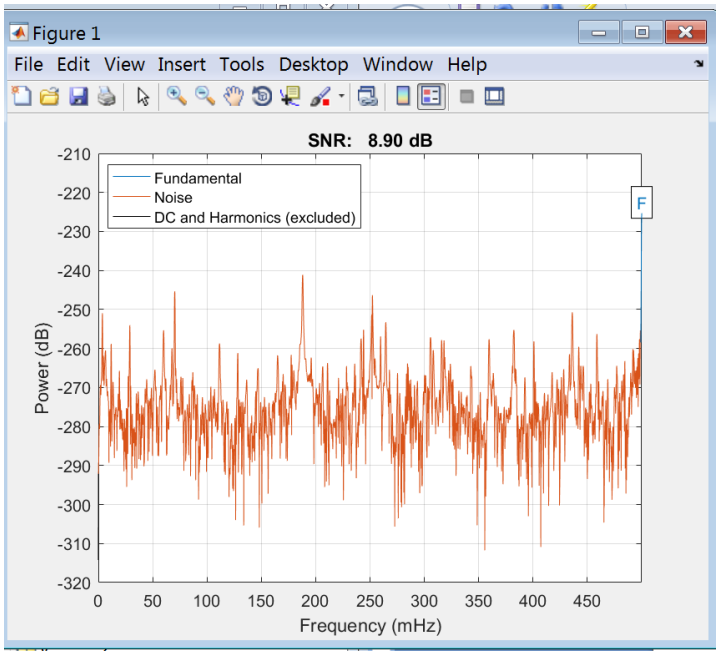
will discuss why this is in-depth but simply put, the version being sampled is not continuous at that particular frequency.

In order to fulfill the Nyquist theory the signal has to be sampled at the minimum rate necessary, which is twice the highest frequency present in the signal. Signals are represented as sin waves with a particular frequency that will then repeat over the coming periods. The idea is to choose the right thing to convolve the signals with in order to reconstruct the signal. The actual interpolation response is going to be a sinc function in the time domain. Another issue that can come to fruition and be a problem is aliasing; it may be the case that the signal is too high frequency to be reconstructed using too low of frequency sampling. In our case, the sampling frequency is the same as the signal frequency which creates an ambiguity. If the signal is sampled along the same path as the initiated frequency there is going to be a discrepancy in the data as seen here where at 1000π the value of average power is found to be a small value of about 1.135×10^{-23} watts.

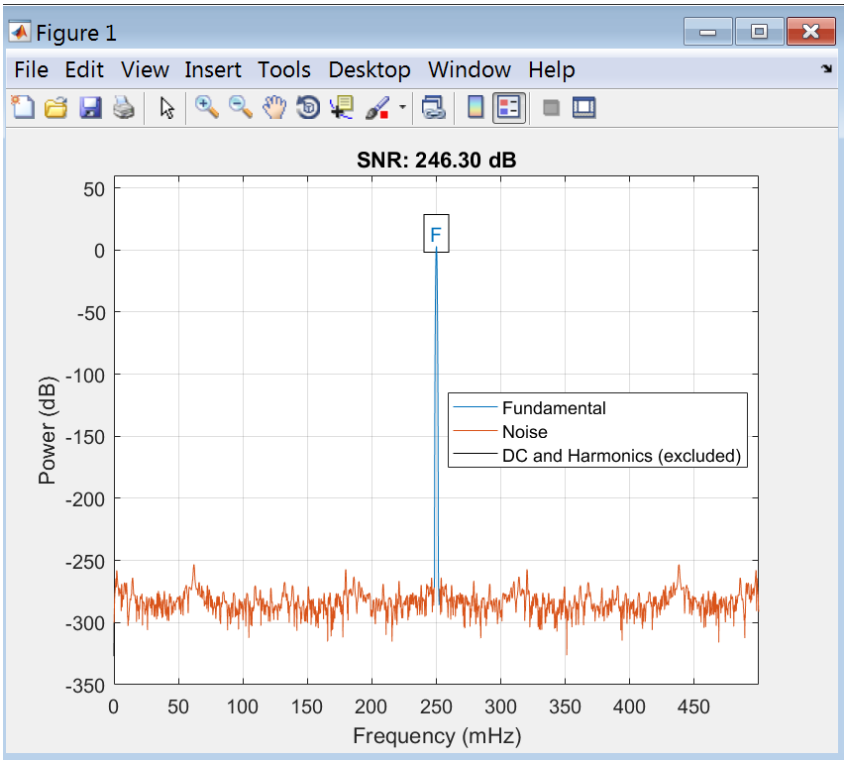
I will use the matlab function for calculating and plotting the signal to noise ratio (SNR) of the 1000π case so we can better observe and discuss why this frequency is generating a low value of average power.

$$SNR_{db} = 10\log_{10}(SNR_P)\ dB$$

So, if we have a signal/N_P ratio of 1, we can plug 1 into the formula and see that it will return a value of 0 dB. We want to have a higher signal to noise ratio in order to have something better to work with. There is not much gain in the case of sampling the continuous signal at 1000π as seen below with a small value of only about 9dB:



If we evaluation the SNR for the case of $w_o = 500\pi$ we can see that the rate is much greater at 246.3 dB. Providing an excellent amount of gain to work with. This reinforces the importance of the Nyquist theorem to generate the most useful signal.



Another thing is that looking at the amount of noise present, indicated by the SNR plots, the case of 1000 π is unusually low. This is another alert that the signal is not being sampled as desired.

It is important to note that in part 4 the noise in the system has been reduced so the effect on the detection process has been reduced or diminished. Additive white noise is superimposed onto the signal without any multiplicative mechanisms at work. Thermal noise is present in all communications systems but for the sake of experimentation it was reduced in this section.

Conclusion

The real conclusions to draw from the data is discussed in the previous section but as a more general takeaway this project was an exercise in using matlab to calculate and plot the PSD function over a set range that fulfills the Nyquist theorem; the plot can be manipulated to find discrepancies that the user will wish to avoid during real-life implementation. The Nyquist theorem must be fulfilled when sampling the signal. Other principles of the sampling theorem such and aliasing and white noise must be accounted for. Manipulating the data and analyzing why certain conditions were created as a result was a powerful method of gaining insight and understanding about sampling theorem and signal processing techniques.

Works Cited

DIGITAL. **COMMUNICATIONS**. **Fundamentals and Applications**. Second Edition. BERNARD SKLAR. **Communications** Engineering Services, Tarzana, California. Prentice Hall (January 21, 2001)

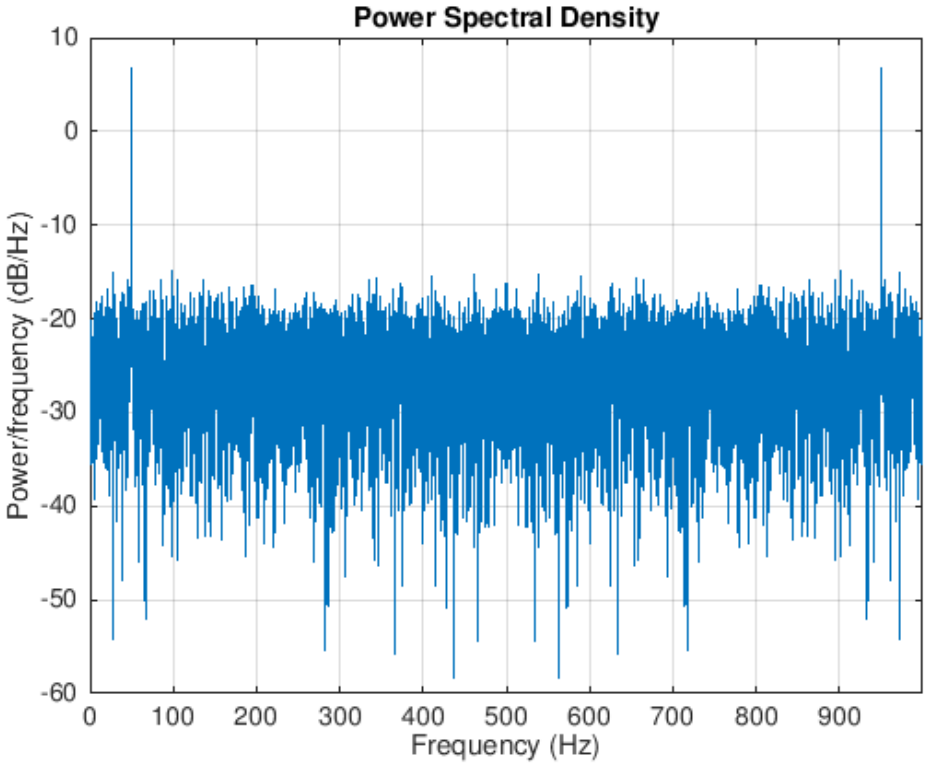
Appendix

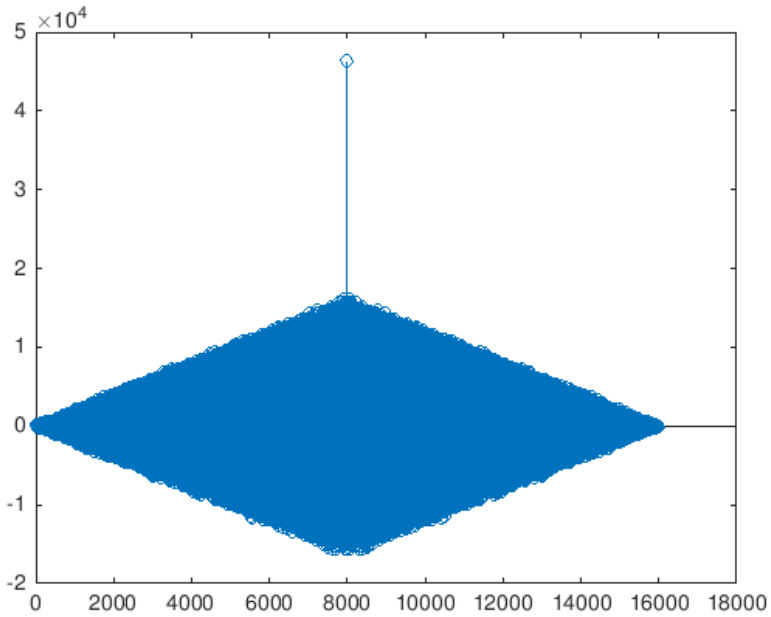
*** These first two pages were implemented using the matlab "xcorr" function...

*** I then go on to redo part 1 and 2 using the "autocorr" function.

```
clc;

clear all;
close all;
Fs = 1000;              % Sampling frequency
Ts = 1/Fs;              % Sampling period
Wo = 100*pi             % Signal Frequency
Ph = pi/2               % Signal Phase
t = (0:Ts:8);           % Time axis
L = size(t,2);          % Total Number of Sample
Sn=2;                   % Standard Deviation of AWGN
A=2;                    % Amplitude
x = Sn*randn(size(t))+ A*cos(Wo*t+Ph);
nfft = 2^nextpow2(length(x));
Pxx = abs(fft(x,nfft)).^2/length(x)/Fs;
Hpsd = dspdata.psd(Pxx,'Fs',Fs,'SpectrumType','twosided');
figure;
plot(Hpsd);
axis([0 Fs-Fs/L -60 10]);

y = xcorr(x,x);
figure;
stem(y);
```

```
Wo =

  314.1593



Ph =



    1.5708
```


Power Spectral Density
Power/frequency (dB/Hz)
Frequency (Hz)
10
0
-10
-20
-30
-40
-50
-60
0
100
200
300
400
500
600
700
800
900


×10^4
5
4
3
2
1
0
-1
-2
0
2000
4000
6000
8000
10000
12000
14000
16000
18000

Published with MATLAB® R2016b

Part 1 16

Part 2 17

Part 3 18

Part 4a 4

Part 4b 5

Part 4c 6

Avg Power -- Part 4a 7

Avg Power -- Part 4b 7

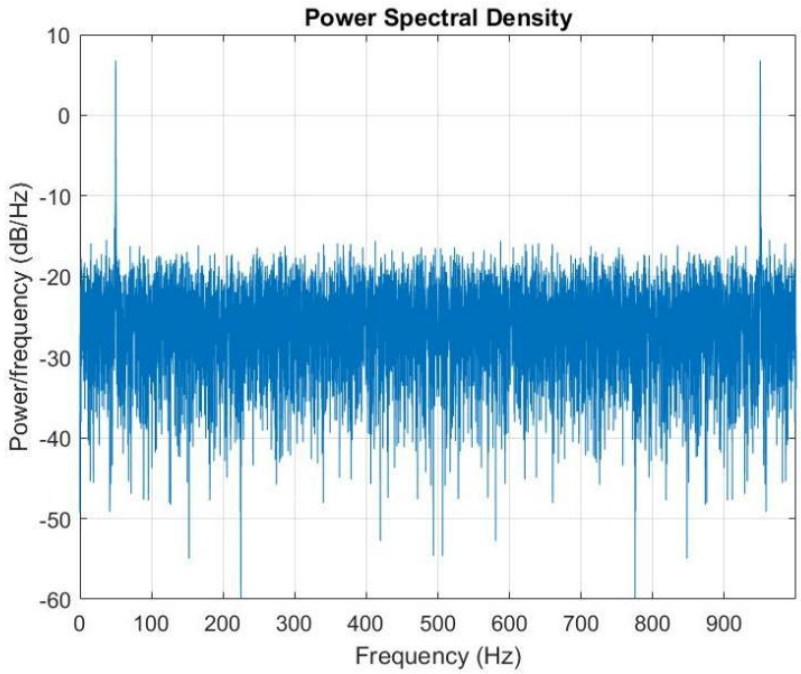
Avg Power -- Part 4c 8

Part 1

```
clc;
clear all;
close all;
Fs = 1000;              % Sampling frequency
Ts = 1/Fs;              % Sampling period
Wo=100*pi               % Signal Frequency
Ph= pi/2                % Signal Phase
t = (0:Ts:8);           % Time axe
L = size(t,2);          % Total Number of Sample
Sn=2;                   % Standard Deviation of AWGN
A=2;                    % Amplitude
x = Sn*randn(size(t))+ A*cos(Wo*t+Ph);

nfft = 2^nextpow2(length(x));
Pxx = abs(fft(x,nfft)).^2/length(x)/Fs;
Hpsd = dspdata.psd(Pxx,'Fs',Fs,'SpectrumType','twosided');
figure;
```

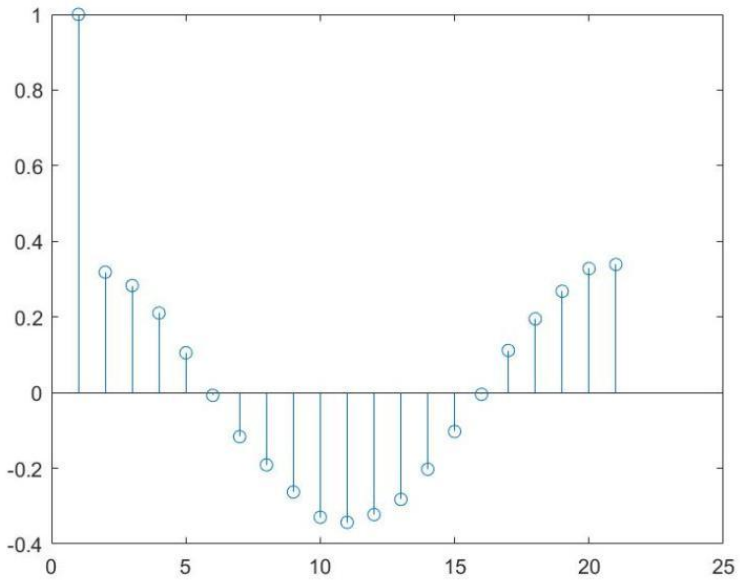
```
plot(Hpsd);
axis([0 Fs-Fs/L -60 10]);
```

```
Wo =

  314.1593


Ph =

    1.5708
```



Part 2

```
y = autocorr(x);
figure;
stem(y);
```



Part 3

```
Fs = 1000;              % initialization of frequency
Ns = 1/Fs;              % period
Wo = 100*pi             % formula for frequency of signal
Ph = pi/2               % formula for phase of signal
t = (0:Ns:8);
L = size(t,2);          % sample count
Std=2;
A=2;                    %it is amplitude
x = Std*randn(size(t))+ A*cos(Wo*t+Ph);
p = bandpower(x)        % for calculation of average power of x(t)

v = var(x)      % calculation of variance of x(t)
```

```
Wo =

  314.1593


Ph =

    1.5708


p =
```

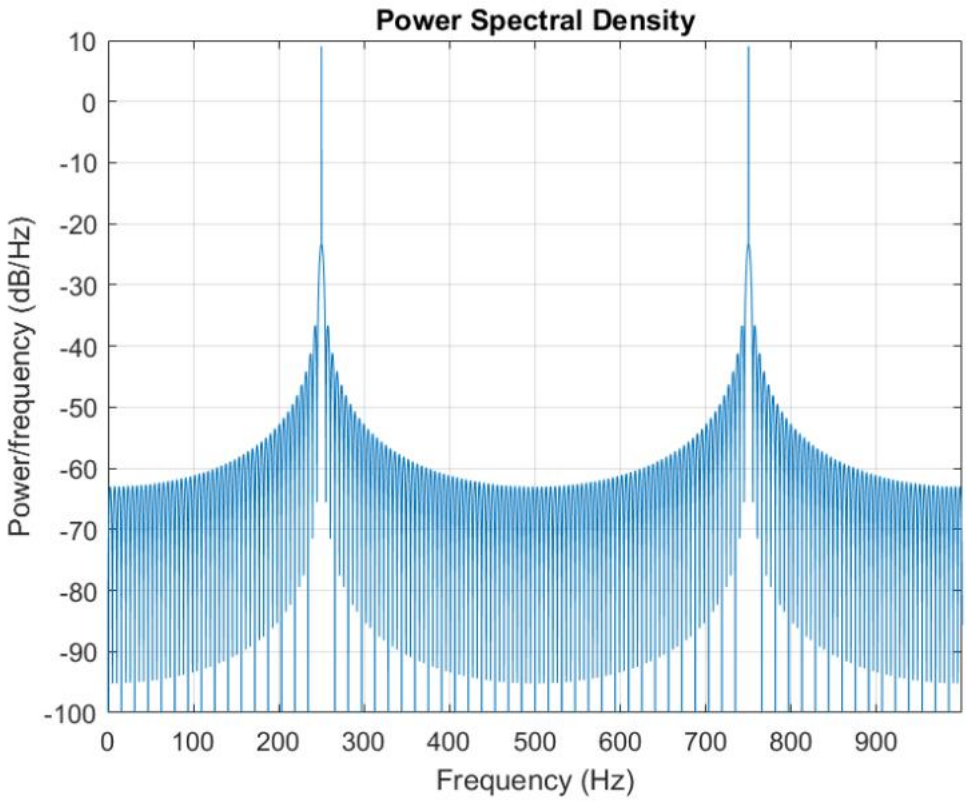
```
    5.8116

v =


    5.8115
```

Part 4a

Sn=0 and Omega0=500pi

```
clc;
clear all;
close all;
Fs = 1000;            % Sampling frequency
Ts = 1/Fs;            % Sampling period
Wo=500*pi             % Signal Frequency
Ph= pi/2              % Signal Phase
t = (0:Ts:8);         % Time axe
L = size(t,2);        % Total Number of Sample
Sn=0;                 % Standard Deviation of AWGN
A=2;                  % Amplitude
x = Sn*randn(size(t))+ A*cos(Wo*t+Ph);

nfft = 2^nextpow2(length(x));
Pxx = abs(fft(x,nfft)).^2/length(x)/Fs;
Hpsd = dspdata.psd(Pxx,'Fs',Fs,'SpectrumType','twosided');
figure;
plot(Hpsd);
axis([0 Fs-Fs/L -100 10]);
```

```
Wo =

   1.5708e+03


Ph =

    1.5708
```



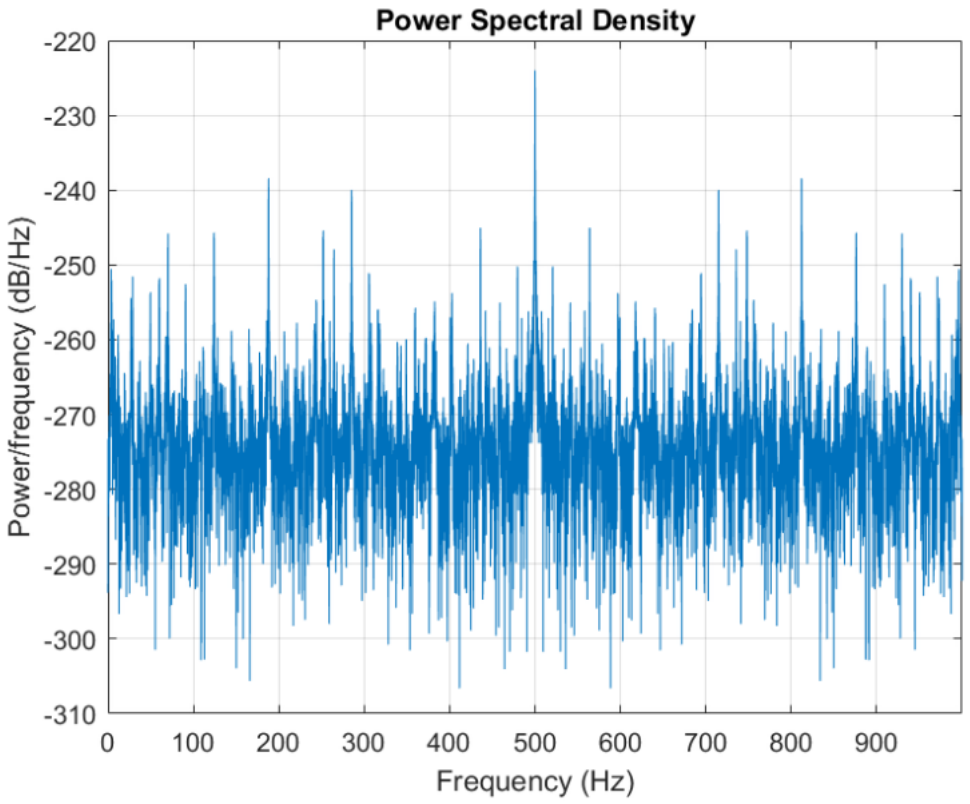
Part 4b

Sn=0 and Omega0=500pi

```
clc;
clear all;
close all;
Fs = 1000;      % Sampling frequency
Ts = 1/Fs;      % Sampling period
Wo=1000*pi      % Signal Frequency
Ph= pi/2        % Signal Phase
t = (0:Ts:8);  % Time axis
L = size(t,2); % Total Number of Sample
Sn=0;           % Standard Deviation of AWGN
A=2;            % Amplitude
x = Sn*randn(size(t))+ A*cos(Wo*t+Ph);

nfft = 2^nextpow2(length(x));
Pxx = abs(fft(x,nfft)).^2/length(x)/Fs;
Hpsd = dspdata.psd(Pxx,'Fs',Fs,'SpectrumType','twosided');
figure;
```

```
plot(Hpsd);
%axis([0 Fs-Fs/L -60 10]);
```

```
Wo =

   3.1416e+03


Ph =

    1.5708
```



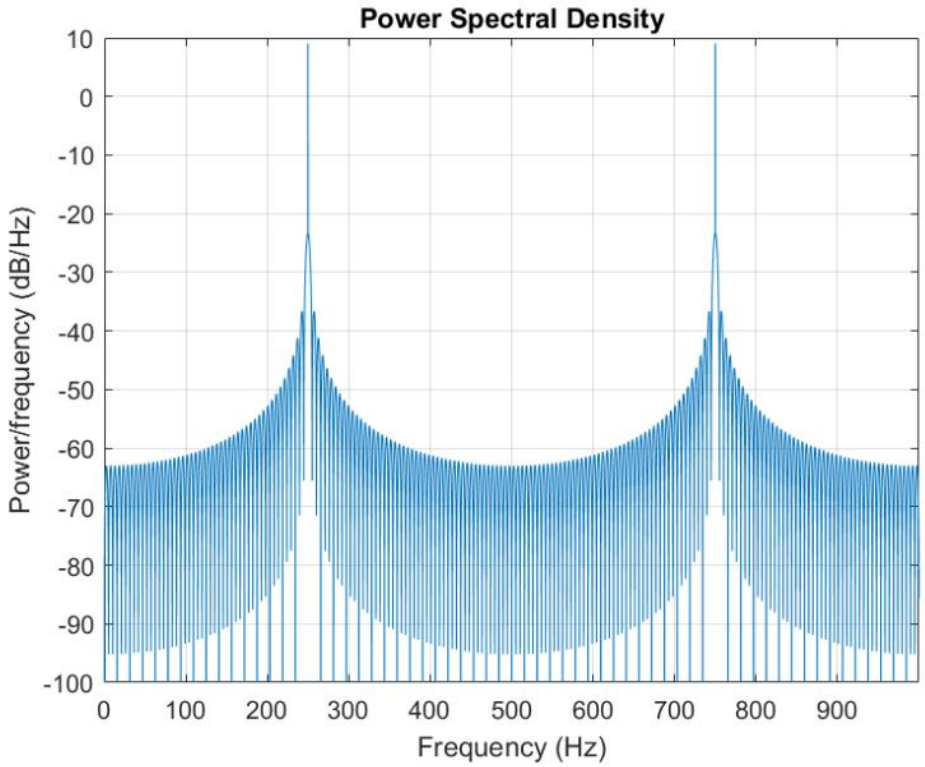
Part 4c

Sn=0 and Omega0=500pi

```
clc;
clear all;
close all;
```

```
Fs = 1000;              % Sampling frequency
Ts = 1/Fs;              % Sampling period
Wo=1500*pi              % Signal Frequency
Ph= pi/2                % Signal Phase
t = (0:Ts:8);           % Time axe
L = size(t,2);          % Total Number of Sample
Sn=0;                   % Standard Deviation of AWGN
A=2;                    % Amplitude
x = Sn*randn(size(t))+ A*cos(Wo*t+Ph);

nfft = 2^nextpow2(length(x));
Pxx = abs(fft(x,nfft)).^2/length(x)/Fs;
Hpsd = dspdata.psd(Pxx,'Fs',Fs,'SpectrumType','twosided');
figure;
plot(Hpsd);
axis([0 Fs-Fs/L -100 10]);
```

```
Wo =

   4.7124e+03


Ph =

    1.5708
```



Avg Power -- Part 4a

```
Fs = 1000;      % initialization of frequency
Ns = 1/Fs;      % period
Wo = 500*pi     % formula for frequency of signal
Ph = pi/2       % formula for phase of signal
t = (0:Ns:8);
L = size(t,2);  % sample count
Std=0;
A=2;            % amplitude
x = Std*randn(size(t))+ A*cos(Wo*t+Ph);
p = bandpower(x) % for calculation of average power of x(t)
v = var(x)      % calculation of variance of x(t)
%snr(x)
```

```
Wo =

   1.5708e+03


Ph =

    1.5708


p =

    1.9998


v =

     2
```

Avg Power -- Part 4b

```
Fs = 1000;      % initialization of frequency
Ns = 1/Fs;      % period
Wo = 1000*pi    % formula for frequency of signal
Ph = pi/2       % formula for phase of signal
t = (0:Ns:8);
L = size(t,2);  % sample count
```

```
Std=0;
A=2;            %it is amplitude
x = Std*randn(size(t))+ A*cos(Wo*t+Ph);
p = bandpower(x) % for calculation of average power of x(t)
v = var(x)       % calculation of variance of x(t)
%snr(x)
```

```
Wo =

   3.1416e+03


Ph =

    1.5708


p =

   1.1349e-23


v =

   1.1350e-23
```

Avg Power -- Part 4c

```
Fs = 1000;      % initialization of frequency
Ns = 1/Fs;      % period
Wo = 1500*pi    % formula for frequency of signal
Ph = pi/2       % formula for phase of signal
t = (0:Ns:8);
L = size(t,2);  % sample count
Std=0;
A=2;            %it is amplitude
x = Std*randn(size(t))+ A*cos(Wo*t+Ph);
p = bandpower(x) % for calculation of average power of x(t)
v = var(x)       % calculation of variance of x(t)
%snr(x)
```

```
Wo =

   4.7124e+03
```

```
Ph =

    1.5708


p =

    1.9998


v =

     2
```

Published with MATLAB® R2016b