LAB 2: AUTOCORRELATION AND POWER SPECTRAL DENSITY

ELG4176 – Comms – Nov 3rd, 2022

Nick Cardamone(300060019)
Niki Galagedara (8728488)
Nickolas Reepschlager(300069614)

Introduction

In this lab, several different pseudo random bit sequences (PBRS) are generated. After generation, the autocorrelation and power spectral density are computed to demonstrate the effects of different encoding schemes has on the data. In particular, the probability of receiving on one is changed and dependence is added between adjacent bits. Both of these changes are compared against the vanilla PBRS. In addition to that, the length of the PBRS is changed, however since built in MATLAB functions are used, the effect of PBRS length is minimal. Proper coding conventions are also discussed at the end as the example code in the lab manual is thrown out.

Note: All three group members have either had covid, currently suspect they have covid or have been bouncing in and out of the hospital as the work has been completed for this lab. If it is disjunct/ missing parts, that is why. Also the code for the lab is on github (https://github.com/ncardamone10/ELG4176-Labs)

Results

Part One

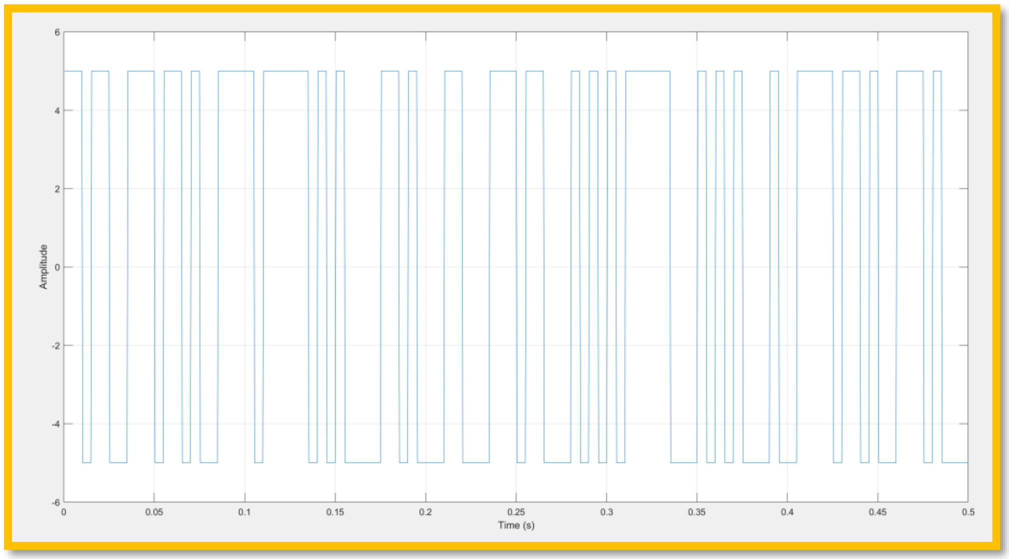

Figure 1: Pseudo Random Bit Sequence

Part Two

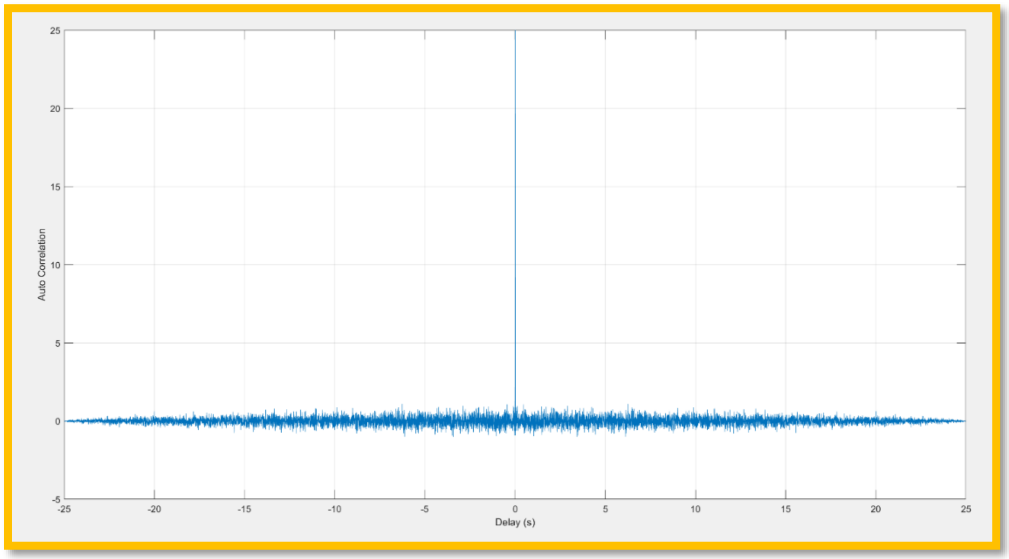

Figure 2: Autocorrelation of PBRS

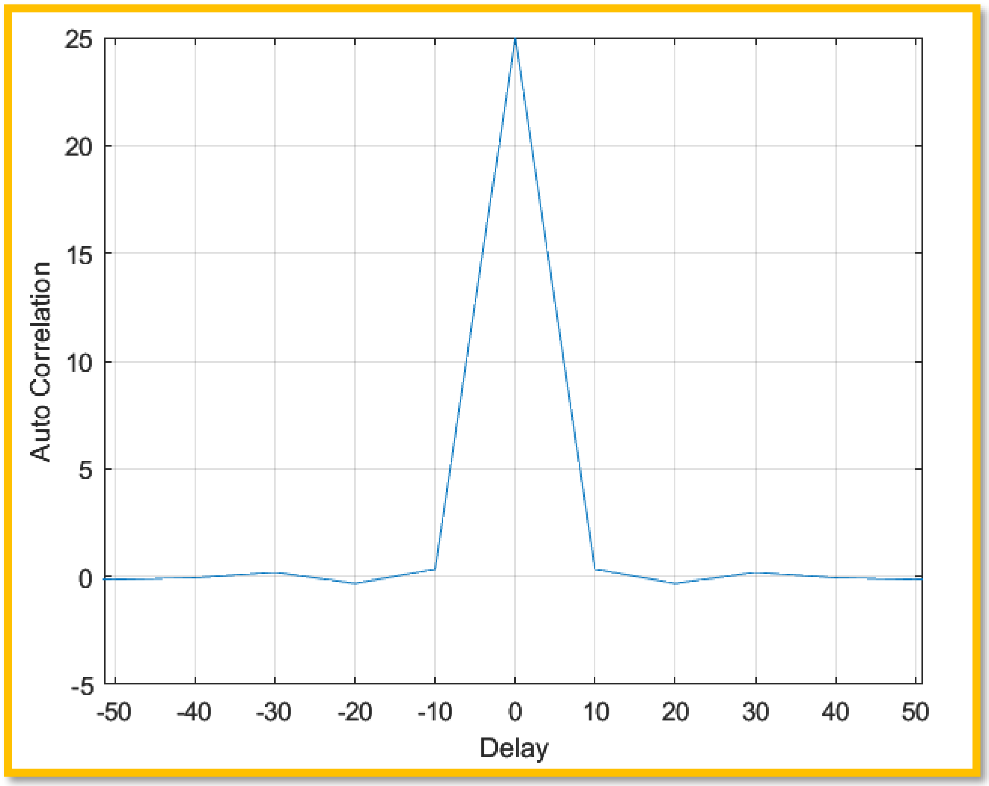

Figure 3: Zoomed In Autocorrelation of PBRS

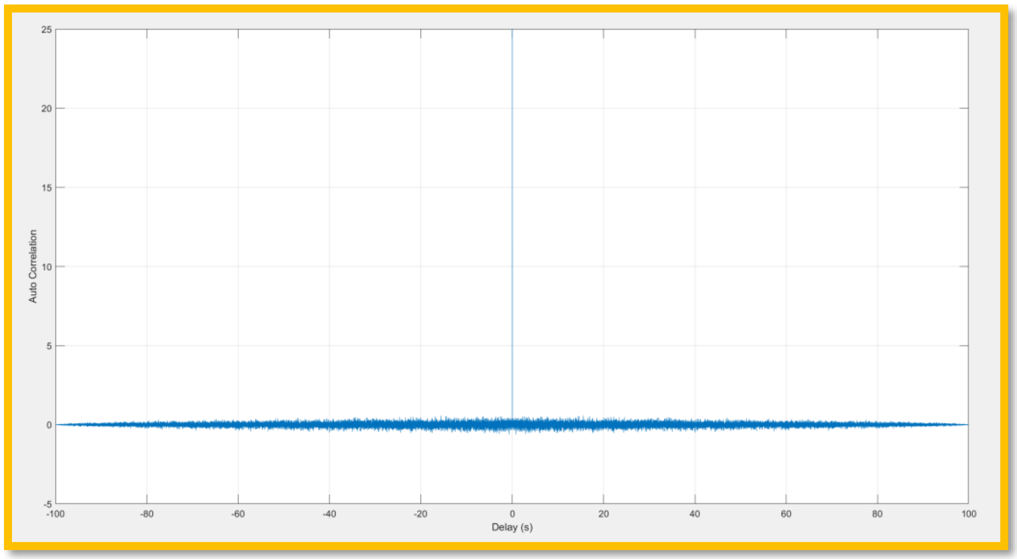

Figure 4: Autocorrelation of longer PBRS

Part Three

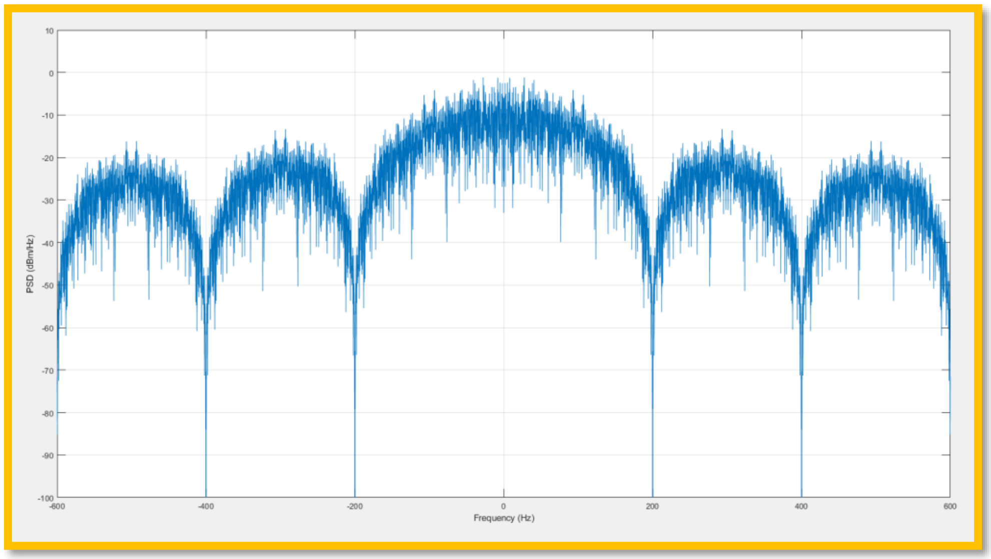

Figure 5: PSD of 1k PBRS

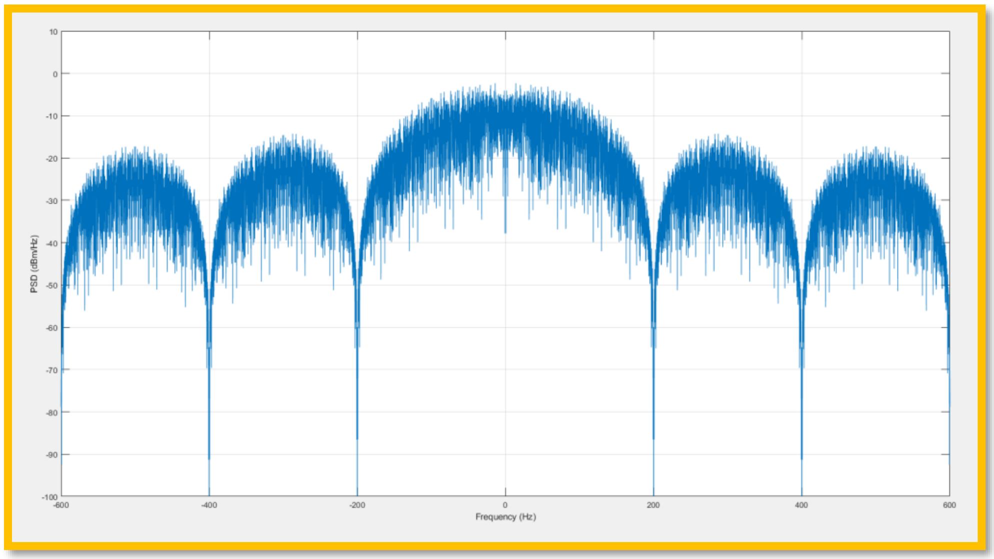

Figure 6: PSD of 2k PBRS

Part Four

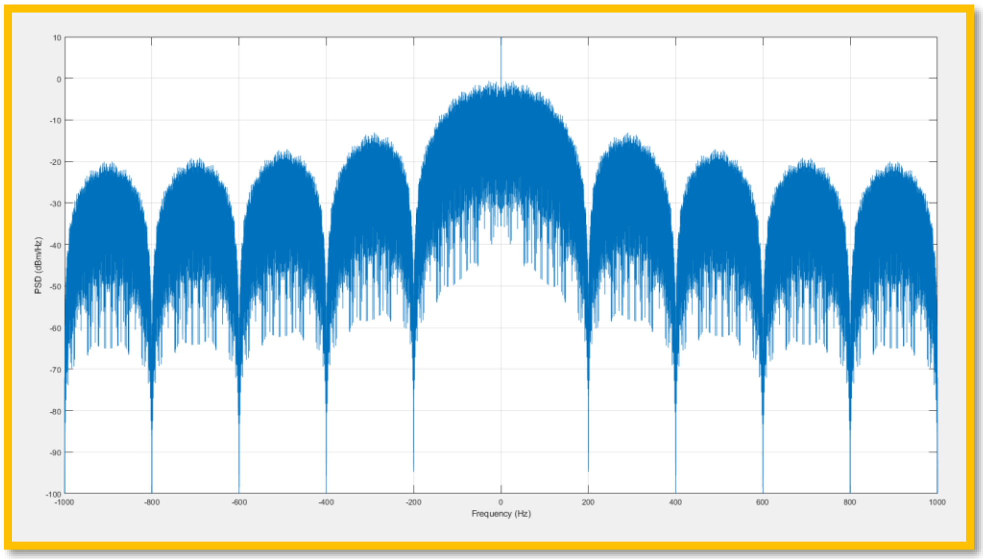

Figure 7: PSD of Weighted Bit Probability PBRS

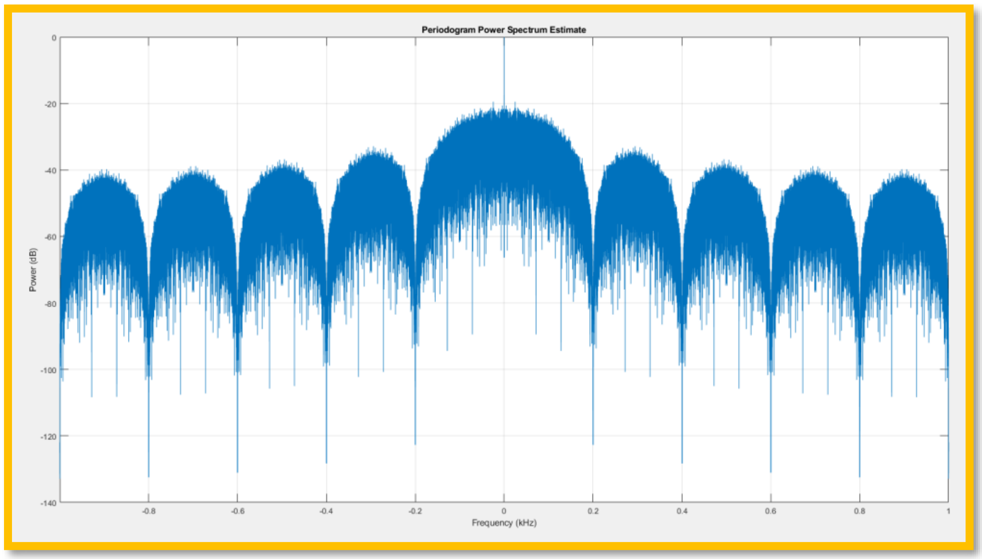

Figure 8: Power Spectrum of Weighted Bit Probability PBRS

DC Component Strength (PSD)	**20 dBm**
DC Component Strength (Power Spectrum)	-0.0698 dBm

Part Five

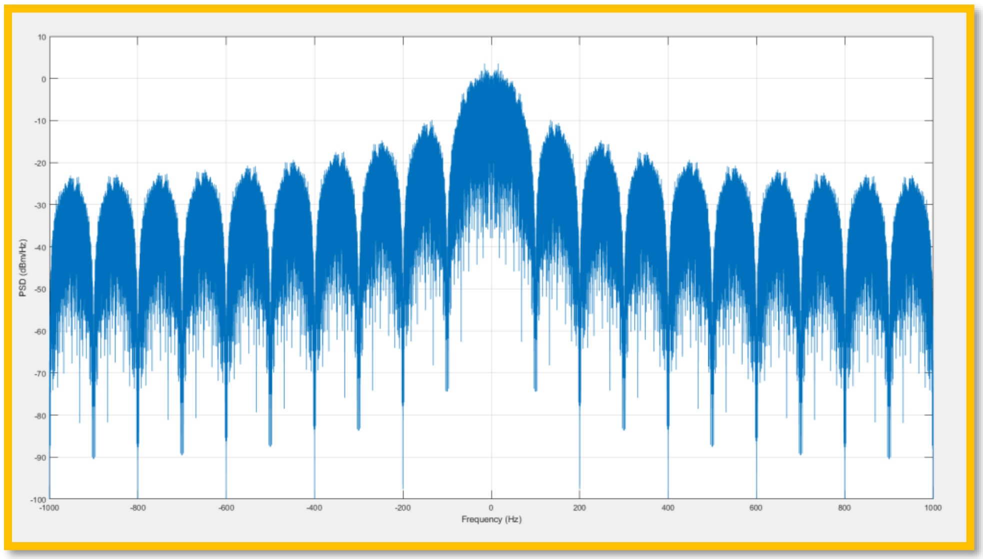

Figure 9: PSD of BW Limited PBRS

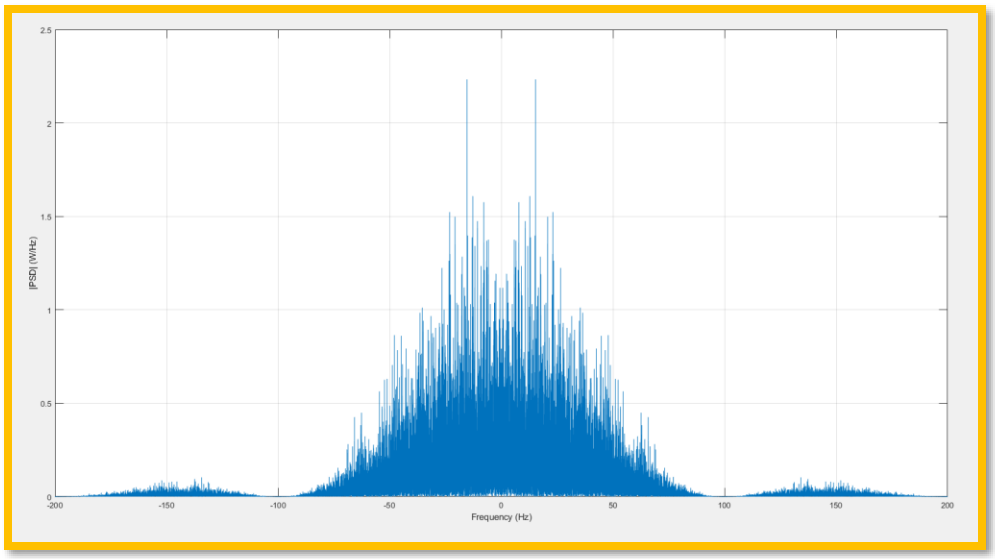

Figure 10: PSD of BW Limited PBRS (linear scale)

Discussion

Part One

The code used to generate this part is as follows.

```
% Signal Params
numberOfBits = 100;
samplesPerBit = 10;
bitRate = 200;
fsampling = bitRate*samplesPerBit;
A = 5;


% Sample Time (s)
Ts = 1/fsampling;


% Generate PBRS between -A and A
x = rand([1,numberOfBits]);
bitArray = round(x);
for k=1:numberOfBits*samplesPerBit
    index = floor((k-1)/samplesPerBit)+1
    x(k)=A*(2*bitArray(index)-1);
    t(k)=k/(fsampling);
end


% Plotting
plot(t,x)
axis([0 t(numberOfBits*samplesPerBit) -A-1 A+1])
```

```
grid on
xlabel('Time (s)')
ylabel('Amplitude')
```

Part Two

The plotted autocorrelation function looks as expected. There is very high correlation at τ = 0 as with no delay between a signal and itself, the signals are identical. Since there are 10 samples per bit used, the autocorrelation function should be a triangle function from τ = –10 to 10 and 0 otherwise. Calculating the autocorrelation, we get $R_x(\tau) = 25\ \Lambda(\tau/T)$. This was observed in the plot with the signals reaching 25 at τ=0, linearly changing from 25 to approximately zero at around +/- 10, and for any other value of τ outside these bounds, the signal is approximately zero.

This is because since our signal is random, there is no repeating pattern, and so shifting the signal any value of τ will not lead to any significant repetition of a pattern. The pattern is a triangle function on the interval [-10,10] centered at 0 because there are 10 samples per bit, so as |τ |increases up to 10, part of each bit will still overlap between the signal and its time shifted version, and so there still is some correlation.

There are still peaks and the autocorrelation is not flat at 0 for values of τ outside of the interval [-10,10]. This is because since the signal is random, and since the signal is binary and therefore has only two possible values, there are likely to be some sections of the signal that repeat. These sections are likely only small sections though among large sections where there is no overlap between the signal and the shifted signal, and so the spikes are only small.

By increasing the number of bits, there are fewer spikes and the non-zero values of τ are closer to 0. This is because by using more bits, for each section of bits that line up, there are likely more which do not.

The code used to generate this part is as follows.

```
% Signal Params
numberOfBits = 20000;
samplesPerBit = 10;
bitRate = 200;
fsampling = bitRate*samplesPerBit;
A = 5;


% Sample Time
Ts = 1/fsampling;


% Create the PBRS
x = rand([1,numberOfBits]);
bitArray = round(x);
for k=1:numberOfBits*samplesPerBit
    index = floor((k-1)/samplesPerBit)+1;
    x(k)=A*(2*bitArray(index)-1);
    t(k)=k/(fsampling);
end


% Find the autocorrelation (cross corr overloaded to auto corr)
[Rx, tau] = xcorr(x, 'biased');


% Plotting
plot(tau*Ts, Rx)
grid on
xlabel('Delay (s)')
ylabel('Auto Correlation')
```

Part Three

$$S_x(f) = \frac{25}{200} sinc^2(\frac{f}{200})$$

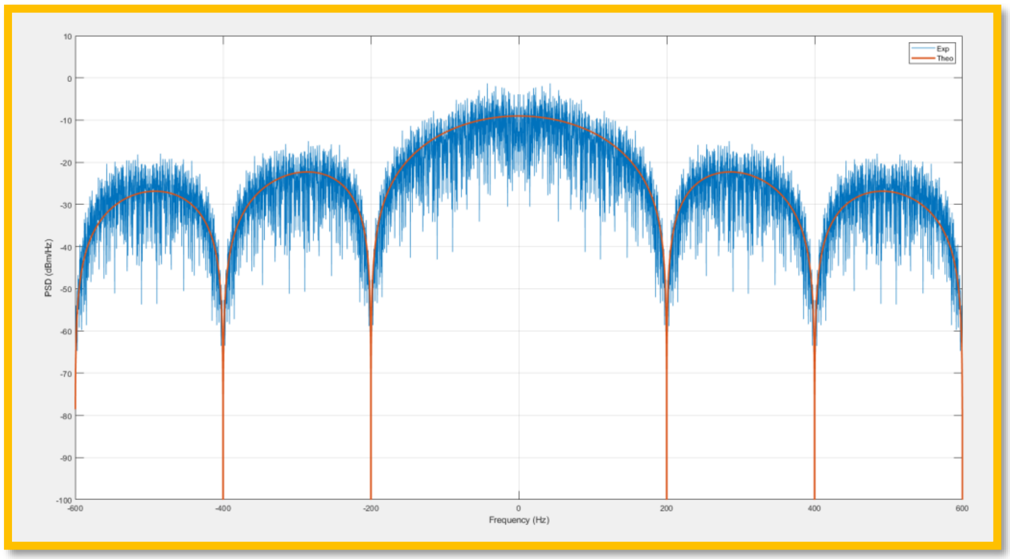

Figure 11: Part Three Exp and Theo Comparison

Name	Calculation Method	Result
Total Power (Theo PSD)	trapz(fp, PSD)	24.2
Total Power (Exp PSD)	trapz(fp, Sxp)	25.0
Total Power (Time Domain)	var(x)	24.9

The empirical PSD calculated is very similar to the ideal PSD. The major differences between the two are that the empirical PSD is not a smooth line, but rather there are many spikes and dips which make up this pattern. As noted in part 2, because the empirical model used random samples, there were some small values of positive autocorrelation outside the interval of [-10,10] where ideally the autocorrelation should have been 0.

Since the PSD is the Fourier transform of the autocorrelation function, the non-ideal nature of the autocorrelation function will carry through to the PSD. We know the autocorrelation function is the expected value of a function multiplied by itself with a delay, and so a non-zero autocorrelation outside of [-10,10] means that there is similarity outside of the expected range, and from this similarity we get a greater probability that the two bits are the same at that point.

By increasing the number of samples, the fluctuations provide a smoother looking curve with more points to get a better autocorrelation function and therefore a better correlation function as by the law of large numbers, increasing the number of random samples increases the accuracy of the empirical model towards the theoretical model.

The code used to generate this part is as follows.

```
% Signal Params
numberOfBits = 1000;
samplesPerBit = 6;
bitRate = 200;
fsampling = bitRate*samplesPerBit;
A = 5;


% Useless Params
% fMax = 400;
% deltaF = 2;


% Sample Period
Ts = 1/fsampling;


% Create the PBRS
x = rand([1,numberOfBits]);
bitArray = round(x);
for k=1:numberOfBits*samplesPerBit
    index = floor((k-1)/samplesPerBit)+1;
    x(k)=A*(2*bitArray(index)-1);
    t(k)=k/(fsampling);
end
% Calculate PSD with periodogram
[Sxp, fp] = periodogram(x,rectwin(length(x)),length(x),fsampling, "centered");
Sxp = Sxp';
fp = fp';

% Plotting
plot(fp, 10*log10(Sxp))
grid on
xlabel('Frequency (Hz)')
```

```
ylabel('PSD (dBm/Hz)')
ylim([-100 10])


% Comparing Total Powers
trapz(fp, PSD)
trapz(fp, Sxp)
var(x)
```

Part Four

In this part, the relative probability of receiving a one vs a zero is shifted to 40% and 60% respectively. Since the adjacent bits are independent, the overall shape of the power spectrum does not change. However, since it is more likely to receive a one than zero, there is now a DC component or non zero average to the PBRS. This non zero average shows us as an impulse at DC in the power spectrum.

$$S_x(f) = \frac{25}{200} sinc^2\left(\frac{f}{200}\right) + \delta(f)$$

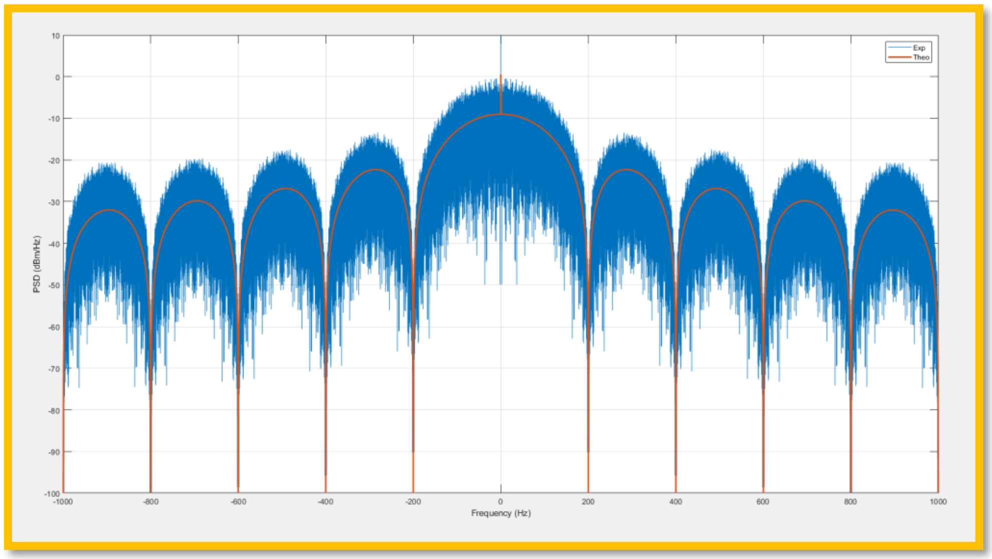

Figure 12: Part Four Exp and Theo Comparison

Name	Calculation Method	Result
Total Power (Theo PSD)	trapz(fp, PSD)	24.5
Total Power (Exp PSD)	trapz(fp, Sxp)	25.0
Total Power (Time Domain)	var(x)	23.9

The code used to generate this part is as follows.

```
% Signal Params
numberOfBits = 20000;
samplesPerBit = 10;
bitRate = 200;
```

```
fsampling = bitRate*samplesPerBit;
A = 5;


% Sample Period
Ts = 1/fsampling;


% Create the PBRS
x = randi([1,10], [1,numberOfBits]);
bitArray = zeros(size(x));
bitArray(x > 10-6) = 1;
for k=1:numberOfBits*samplesPerBit
    index = floor((k-1)/samplesPerBit)+1;
    x(k)=A*(2*bitArray(index)-1);
    t(k)=k/(fsampling);
end


% Calculate PSD
%[Sxp, fp] = periodogram(x,rectwin(length(x)),length(x),fsampling, "centered",
'power');
[Sxp, fp] = periodogram(x,rectwin(length(x)),length(x),fsampling, "centered");
Sxp = Sxp';
fp = fp';


% Plotting
plot(fp, 10*log10(Sxp))
grid on
xlabel('Frequency (Hz)')
ylabel('PSD (dBm/Hz)')
ylim([-100 10])


% DC Power
idx = find(fp == 0)
DC = Sxp(idx)
DC = 10*log10(DC)
```

Part Five

In this part, dependency between adjacent bits is added. This dependency has the effect of decreasing the bandwidth. If the bandwidth of the signal is taken from DC to the first null in the power spectrum, this dependency slashes the overall bandwidth by half.

$$S_x(f) \;=\; \frac{25}{200}\operatorname{sinc}^2\left(\frac{f}{200}\right)\left(1+\cos\left(\frac{2\pi f}{200}\right)\right)$$

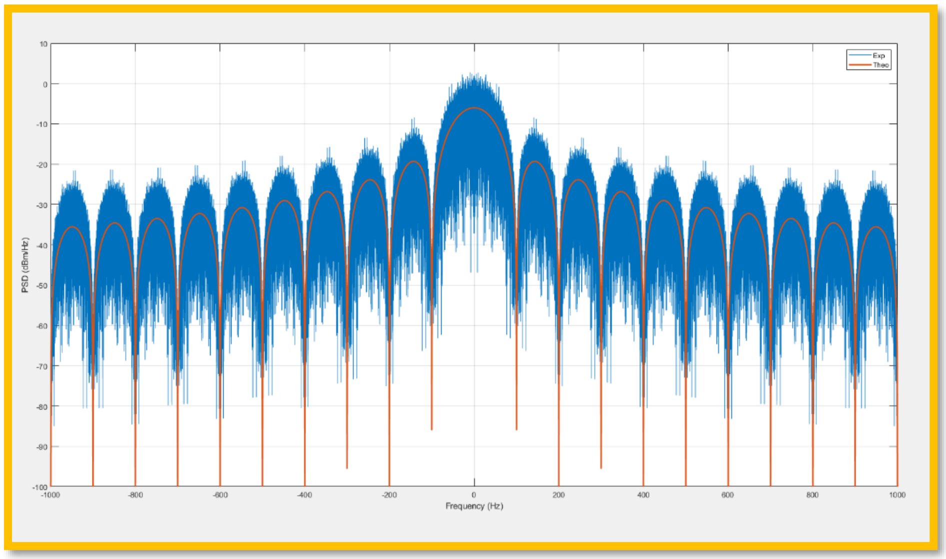

Figure 13: Part Five Exp and Theo Comparison

Name	Calculation Method	Result
Total Power (Theo PSD)	trapz(fp, PSD)	24.7
Total Power (Exp PSD)	trapz(fp, Sxp)	24.9
Total Power (Time Domain)	var(x)	24.9

The code used to generate this part is as follows.

```
% Signal Params
numberOfBits = 20000;
samplesPerBit = 10;
bitRate = 200;
fsampling = bitRate*samplesPerBit;
A = 5;


% Sample Period
Ts = 1/fsampling;




% Generate PBRS
%   Generate Bits
x = rand([1,numberOfBits]);
bitArray = round(x);


%   Generate Encoded Bits
encodedBitArray = zeros(size(bitArray));
```

```
encodedBitArray(1) = bitArray(1);
for k = 2:length(bitArray)
    An = A*(2*bitArray(k)-1);
    An_prev = A*(2*bitArray(k-1)-1);
    encodedBitArray(k) = An/sqrt(2) + An_prev/sqrt(2);
end


%   Upsample Bit Array
for k=1:numberOfBits*samplesPerBit
    index = floor((k-1)/samplesPerBit)+1;
    x(k)=encodedBitArray(index);
    t(k)=k/(fsampling);
end


% Calculate PSD
[Sxp, fp] = periodogram(x,rectwin(length(x)),length(x),fsampling, "centered");
Sxp = Sxp';
fp = fp';


% Plotting
plot(fp, 10*log10(Sxp))
grid on
xlabel('Frequency (Hz)')
ylabel('PSD (dBm/Hz)')
ylim([-100 10])
```

Coding Conventions

As mentioned in the introduction and seen through out this report, the example code provided was disregarded. This is done due to the example code having poor form and efficiency. Instead, built in MATLAB functions are used to compute the autocorrelation and the power spectral density.

To compute the autocorrelation, the cross correlation function is used (xcorr()). When this function is passed one argument, it overloads to the autocorrelation function. The trade off to using this function is there is an offset between the calculated autocorrelation values and the actual values. To get around this, the ‘biased’ keyword is passed in to the xcorr function as well. To verify that the xcorr function produces similar results to the example code, one text case is presented below.

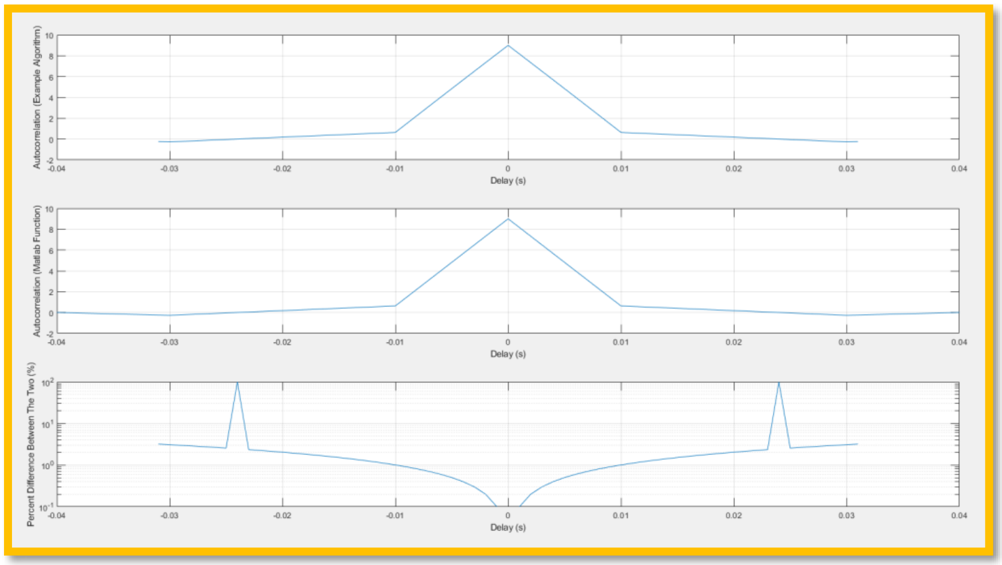

Figure 14: Autocorrelation Comparison between Built in xcorr() and Example Code

To calculate the power spectral density, the periodogram function is used instead of the discrete Fourier transform algorithm given in the example program. This is done purely for efficiency as the periodogram is based on the fast fourier transform with complexity O(nlogn) instead of the much slower discrete Fourier transform with complexity O(n^2). The following plot compares the PSD of a PBRS computed using both methods as well as a simple fft of the autocorrelation using xcorr().

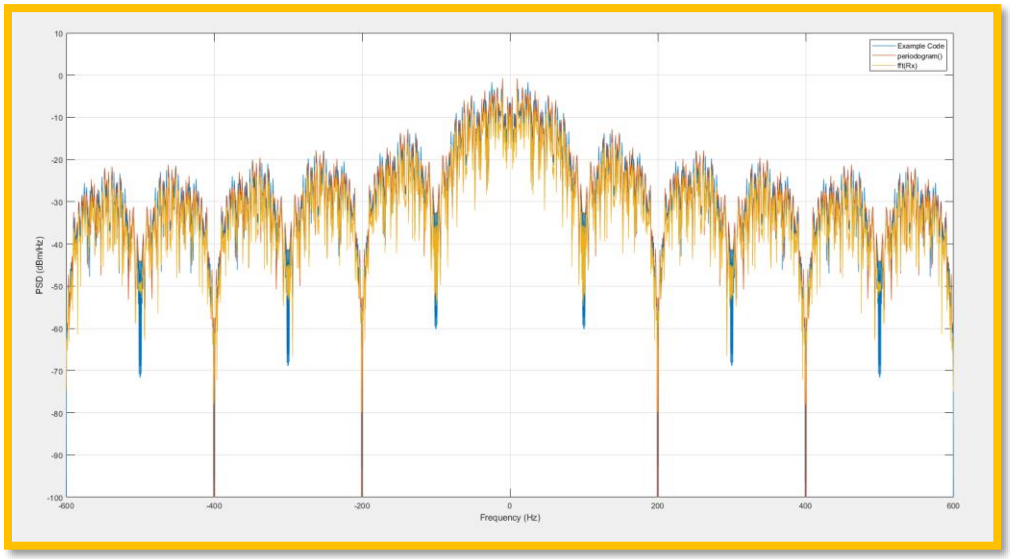

Figure 15: Comparison of PSDs with different computation algorithms

In addition to the visual comparison, the total power given from each PSD is tabulated below.

Name	Calculation Method	Result
Total Power (Example PSD)	trapz(f, PSDofSig)	22.8
Total Power (fft() PSD)	trapz(fx, Sx)	1e-11
Total Power (periodogram() PSD)	trapz(fp, Sxp)	22.8
Total Power (Time Domain)	var(x)	22.7

From the visual comparison and the tabulated total powers, the periodogram and example algorithm produce very similar powers to the time domain signal. In addition to that, the periodogram runs significantly faster than the example algorithm. Therefore, there is no reason to ever use the algorithm presented in the example and said example algorithm should be avoided.

Comparing the periodogram to the naive fft of the autocorrelation, the overall shape is the same. However, there is an offset as can be seen below.

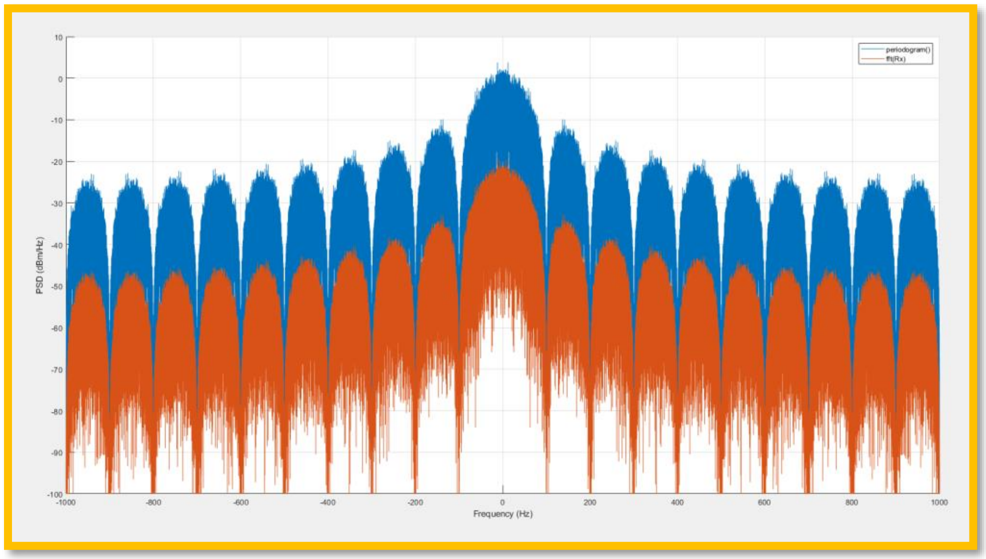

Figure 16: Comparison of Periodogram and fft(Rx) based PSDs

Conclusion

To conclude, in this report various pseudo random bit sequences are generated and characterized. The auto correlation function and power spectral densities are computed using built in MATLAB function and then further analyzed. The properties of the PRBS are altered (such as bit independence/ relative frequency) and those effects are quantified. Major take aways are when one bit is more likely than another, a tone at DC appears in the power spectrum and when independence is introduced between bits, the overall bandwidth can decrease.