Digital Modulation Simulation Project

ECE 300: Communication Theory

Professor Frost

Allister Liu, Bonny Wang, Amy Leong

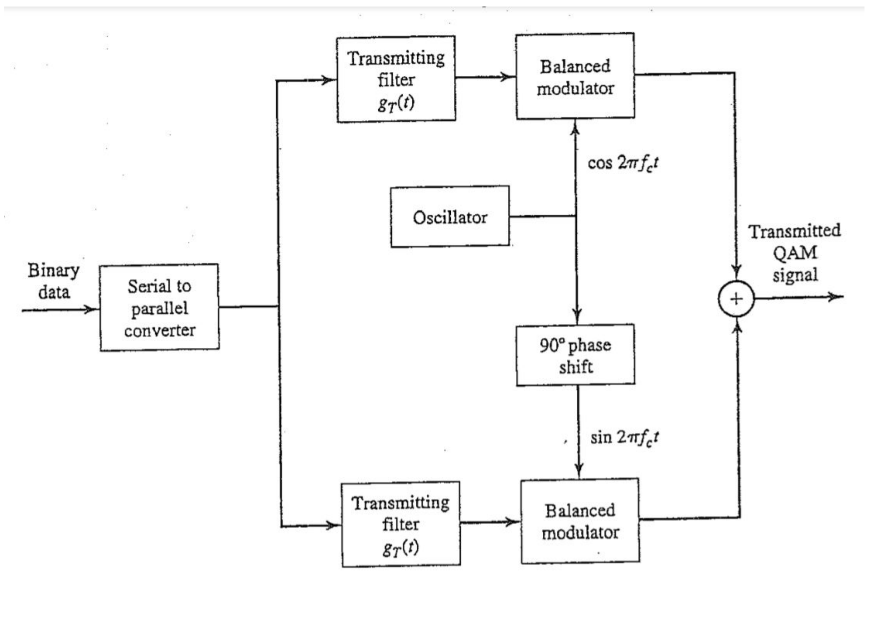
Table of Contents

Introduction

Digital modulation encodes digital information into the amplitude, phase, frequency or a combination of those listed so that the information can be transmitted between two or more locations. Generally, several bits are encoded into one symbol and the rate of symbol transmission determines the bandwidth of the transmitted signal[1]. This project sets out to explore the probability of error for QAM, PSK, and DPSK digital modulation methods for symbols from a constellation that is corrupted by additive white Gaussian noise (AGWN).

QAM

Quadrature amplitude modulation or QAM uses amplitude and phase variations to represent binary data. QAM uses two message signals with the same carrier frequency by changing the amplitude and phase shifting 90 degrees[2]. Cos ($2\pi f_c t$) and sin ($2\pi f_c t$) are used on each quadrature carrier to construct the signal waveforms that are not constrained to fall on a circle. Below is a block diagram for the QAM modulator.



[1]https://www.sciencedirect.com/topics/computer-science/digital-modulation#:~:text=Digital%20modulation%20is%20the%20process,its%20robustness%20to%20channel%20impairments.
[2] https://arxiv.org/ftp/arxiv/papers/2002/2002.07392.pdf

Figure 1: Functional block diagram for QAM modulator (Diagram from *Communication Systems Engineering* 2nd edition)

An advantage of the QAM modulation is that there are more bits per carrier, so that there is more efficient usage of bandwidth. However, a disadvantage is that the QAM modulation is more prone to the noise. Another disadvantage is that the amplifier would use more power. The amplifier is from the amplitude component of the signal. Since there is an amplitude component, there needs to be a linear amplifier[3].

PSK

Phase shift keying or PSK is a digital modulation process that changes the phase of the carrier signal to transmit data. The change in the phase is determined by the binary inputs and the changes can be random. In comparison to the QAM modulation, PSK only modulates the phase of the symbol instead of both phase and amplitude. This results in a better transmitter efficiency because the transmitter will always be operating at the peak power and the transmitted signal will be at the maximum strength[4]. PSK is good for long range communications.

DPSK

Differential phase shift keying or DPSK is a special type of PSK that changes the phase based on the previous value. Since the phase shift can be measured by using the phase of the previous symbol, DPSK is simpler to implement than PSK. However, since this modulation uses two

[3]https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-QAM-types.html
[4]https://www.researchgate.net/post/why_QAM_is_better_than_PSK_and_DPSK_modulation_techniques_what_do_you_mean_by_jakes_and_dent_channel_model_and_where_it_is_used

consecutive bits, an error in the first bit will lead to an error in the second bit. Also, the bit error rate for DPSK is usually higher in comparison to BPSK[5].

[5] https://www.elprocus.com/differential-phase-shift-keying/

Methods

- symbolGen

```
function output = symbolGen(constellation, num)
    output = constellation(randsample(length(constellation),num,'true'));
end
```

A simple one-line function that generates symbols from a given constellation. The built in MATLAB function constellation samples from a vector equiprobably.

- LSestimation

```
function output = LSestimation(constellation, symbolNoisy)
    % Find the distance of each symbol to every constellation
    distance = abs(constellation - symbolNoisy.');

    % Find the index number of min value in each row
    [minDistance, iMin] = min(distance,[],2);

    % Return the closest constellation
    output = constellation(iMin);
end
```

To avoid loops, simple vector subtraction was used to find the distance of each symbol to every constellation. Using `[minDistance, iMin] = min(distance,[],2);` syntax, the minimum distance was found with its corresponding index number. Then, the constellation corresponding to the index will be returned. (Although the actual minimum distance was not useful therefore meaningless to find)

- scale

```
function [nf, received] = scale(constellation, symbol, N0, SNRpBit, differential)

    M = length(constellation);
    N = size(symbol, 2);

    % desired average symbol energy
    % (from the SNR per bit, the noise power
    %  and the number of symbols in the constellation)
    desiredAvgEnergy = 10^(SNRpBit/20) * N0/2 * log2(M);
    % average energy of constellation
    avgEnergy = mean(abs(constellation).^2);

    % scaling
    scaleFactor = sqrt(desiredAvgEnergy / avgEnergy);

    % scale base constellation to true constellation
    constellationScaled = constellation .* scaleFactor;

    % scale symbol to true transmitted scaled symbol
    symbolScaled = symbol .* scaleFactor;

    % generate AWGN
    %    (variance = N0/2)
    noise = sqrt(N0/2)/2 * (randn(1,N) + 1j*randn(1,N));

    % adding noise to symbol to get noisy symbol
    noisy = symbolScaled + noise;

    % differential scheme
    if differential == true
        %fprintf("in true\n");

        % phase of scaled symbol
        symbolScaledAng = atan2(imag(symbolScaled), real(symbolScaled));

        % return phase differential of noise-free scaled symbol
        nf = diff(symbolScaledAng);

        % phase of constellation
        constellationAng = atan2(imag(constellationScaled),real(constellationScaled));
        constellationAng = [constellationAng constellationAng+2*pi];
        constellationAngle = ([constellationAng-min(constellationAng) 4*pi])-2*pi; %min phase 0

        % phase of noisy symbol
        noisyAng = atan2(imag(noisy),real(noisy));

        % differential noisy
        noisyAng = diff(noisyAng);

        % call LSestimation and return estimated received vector
        received = LSestimation(constellationAngle, noisyAng);

    % non-differential scheme
    elseif differential == false
        %fprintf("in false\n");

        % return noise-free scaled symbol
        nf = symbolScaled;

        % call LSestimation and return estimated received vector
        received = LSestimation(constellationScaled, noisy);
    end

end
```

The scale function takes in a base constellation, a list of N noise-free transmitted symbols, noise power N0, desired SNR per bit in dB, and a boolean variable differential (differential scheme or not) as input. The function will find the desired average symbol energy from the SNR per bit, N0 and the number of symbols in the constellation (M), and the average energy of the base constellation; therefore, scaling factor is determined, and the function will scale the transmitted symbols and the base constellation to the true constellation with correct average energy based on the scaling factor. Then noise is generated with variance = N0/2 and added to the scaled noise-free symbols to produce noisy symbols.

If in a differential scheme, the function will translate both the noisy symbols and noise-free symbols into their phase differentials, and call LSestimation to find the estimated received vector. Finally, the phase differentials of noise-free symbols and the estimated received vector are returned.

Otherwise, if in a non-differential scheme, no translation is needed. The function will simply call LSestimation to find the estimated received vector, and return that along with the noise-free scaled symbols.

- **calcError**

```
function output = calcError(estSym, trueSym)
    N = size(estSym, 2);
    threshold = 0.001;
    output = nnz(abs(estSym-trueSym) > threshold)/N;
end
```

First the calcError function will find N, number of symbol transmission. By utilizing the nnz() function, calcError will find the number of errors by comparing the difference of estimated symbol with the true symbol to a threshold of 1e-3. Finally, dividing numbers of errors by N, the function will return the probability of error.

Results

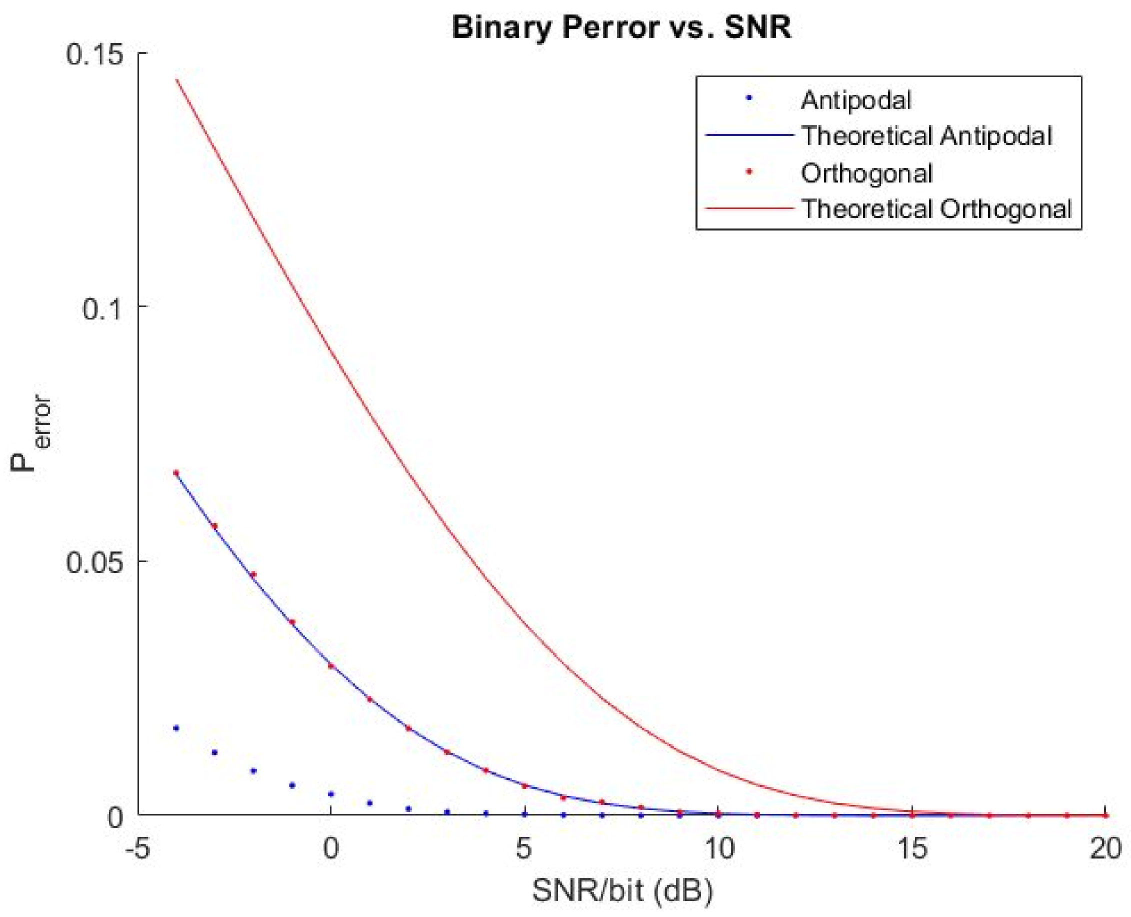


Figure 2: The figure shows the P_{error} vs SNR/bits for antipodal and orthogonal as well as its theoretical values.

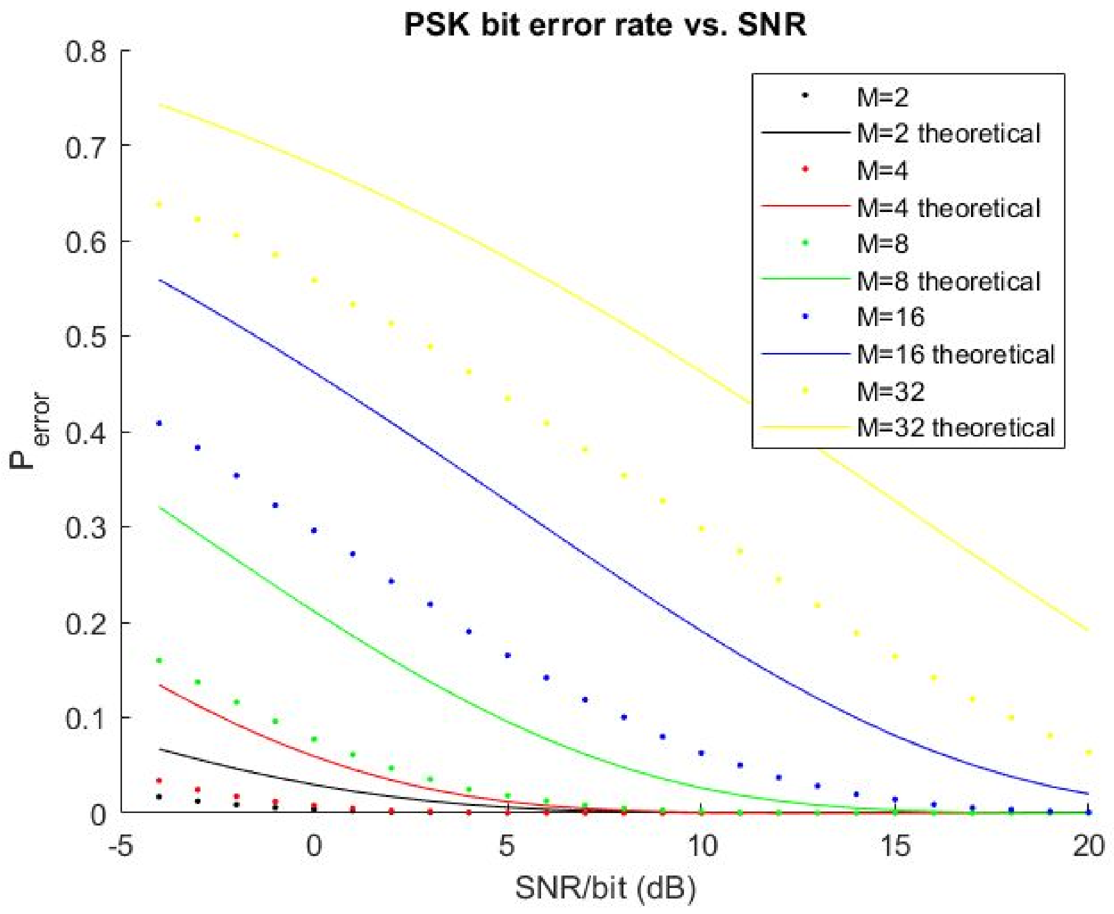


Figure 3: This figure shows the PSK bit error rate with its SNR value for different M values. The theoretical values are also shown. The binary antipodal case was used for M=2.

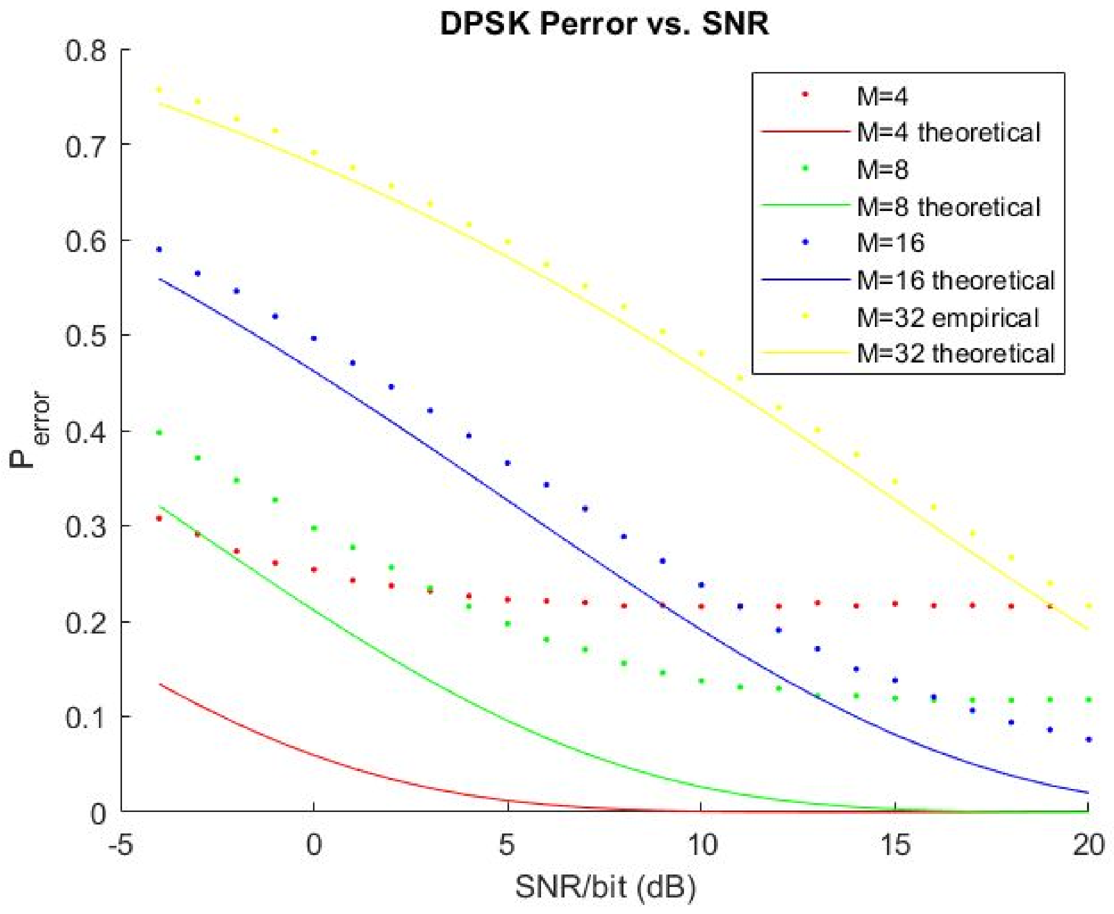


Figure 4: This figure shows the DPSK bit error rate with its SNR value for different M values. The theoretical values are also shown.

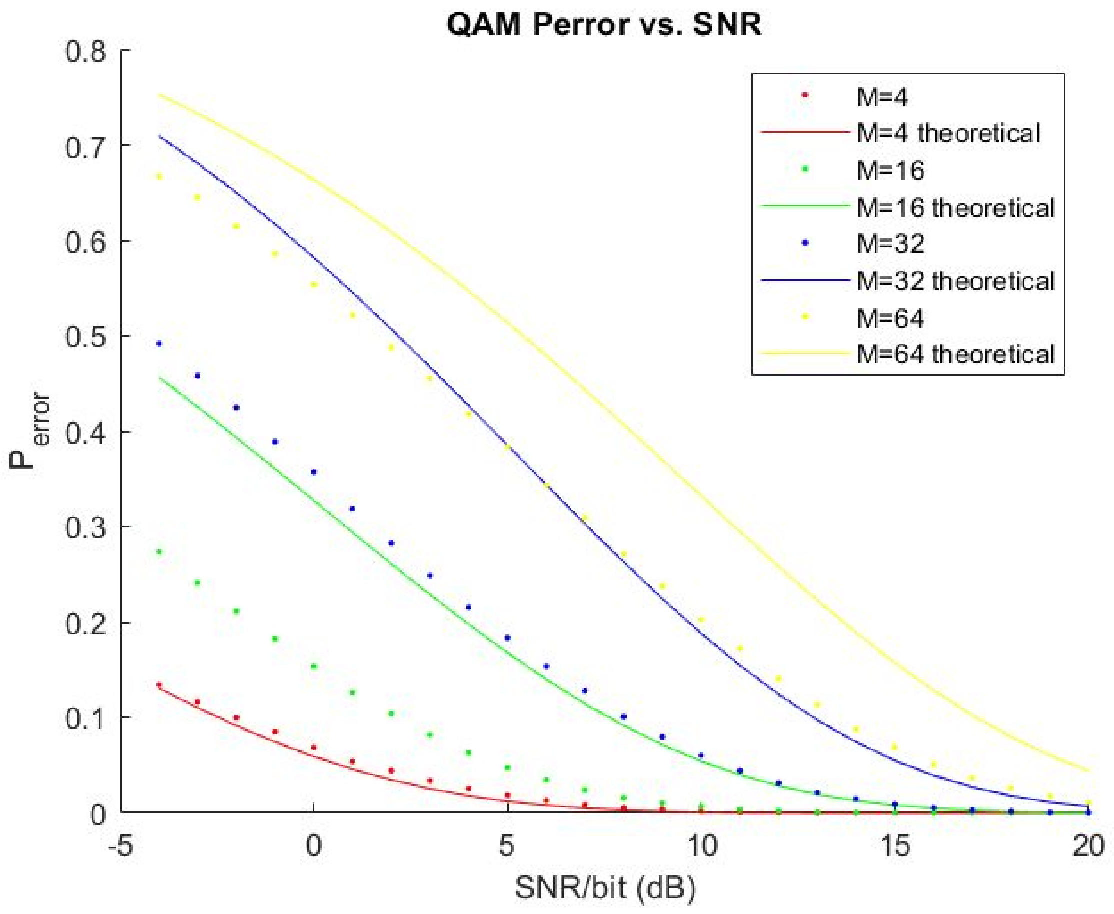


Figure 5: This figure shows the QAM bit error rate with its SNR value for different M values. The theoretical values are also shown.

	N	Time taken (s)
Binary Antipodal	1000000	0.19
Binary Orthogonal	1000000	0.16
PSK	1000000	1.87
DPSK	1000000	1.51
QAM	1000000	3.08

Figure 6 : The table shows the time values for each part of the simulation. The commands tic toc were used to calculate these time values

Discussion

The probability of error of binary antipodal and orthogonal is lower than theoretical value as seen in Figure 2. Therefore, binary antipodal and orthogonal signaling have relatively good performance. Noticeably, the difference between the binary antipodal and orthogonal in the experiment is the same as the theoretical difference - 3dB. Also, orthogonal signaling requires twice as much bit energy than antipodal when the same error rate is needed.

Generally, increasing M results in a higher bit rate to bandwidth ratio, but this comes with a cost of a higher SNR/bit. The probability of error for QAM increased as the value of M increased. This can be confirmed in Figure 5, where the curves are steeper as each M value increases. Since there is a negative slope, the effect of M value on the probability of error is greater with lower M value. This increase in error due to an increase in M is because the symbols are closer together in the constellation space. The decoding region for each symbol decreases as there are more number of symbols in the signal space. This trend of increasing M leading to an increase in error can also be said for PSK and DPSK. As seen in Figures 3 and 4, the curves are also steeper as M increases. QAM has comparable performance with PSK when M = 4 and better performance than PSK when M is larger than 4. In practical use, QAM is also more energy efficient than PSK when M is greater than 4. There was a weird behavior for M=4 in the DPSK case where the P_{error} exceeded M=8, M=16 and M=32 after SNR/bit =3 dB.

Generally, DPSK has a much higher probability of error than PSK, especially with lower M value. When looking at Figures 3 and 4, DPSK generally has a higher error for the same SNR value than PSK. For example, for M=32 and SNR/bit is -4dB, the P_{error} value for PSK was .6445

,while the P_{error} value for DPSK was .7545. Therefore, DPSK is more affected by noise since it depends on the previous transmitted signal. This dependency on the previous bit can cause errors in pairs. However, since DPSK does not need a coherent demodulation, it might have better performance in more practical situations and save a large amount of power.

As seen in Figure 6, the reported time values were all a couple of seconds or less. The lowest time values were for binary antipodal and binary orthogonal, which were 0.19 seconds and 0.16 seconds respectively. The time needed to produce the graph of binary antipodal and binary orthogonal was less than the other graphs because there was less calculation needed. The time values were measured using tic toc and the time values for binary antipodal and binary orthogonal were generally under 1 second. The time it took for DPSK and PSK were relatively the same compared to the time for QAM. The graph for DPSK took 1.51 seconds and the graph for PSK took 1.87 seconds, which was lower than the 3.08 seconds for QAM. The reason for this is because it includes the M = 64 situation so there are more operations to be performed. A larger matrix calculation was needed and the generation for QAM took longer. However, using this method, overall, the time needed is relatively short as expected even with large amounts of symbol transmissions.

Conclusion

The simulation generated symbols, generated Gaussian noise, added the noise, applied least squares, and calculated the error for QAM, PSK, and DPSK digital modulation. Overall, the runtime for each part of the simulation was relatively fast. The runtimes were roughly a couple of seconds or less. When the value of M increases for those 3 digital modulations, the error also increases since the symbols are closer together in the constellation space. For the differential and non differential modulation, DPSK generally had a higher error rate than PSK. This is due to DPSK's reliance on the previous bit.

Bibliography

Bharati, Subrato, et al. Bit Error Rate Analysis of M-Ary PSK and M-Ary QAM over Rician Fading Channel. arxiv.org/ftp/arxiv/papers/2002/2002.07392.pdf.

Manfredi, Albert.
www.researchgate.net/post/why_QAM_is_better_than_PSK_and_DPSK_modulation_techniques_what_do_you_mean_by_jakes_and_dent_channel_model_and_where_it_is_used.

“Advantages of QAM | Disadvantages of QAM | 16QAM,64QAM,256QAM.” RF Wireless World, RF Wireless World, www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-QAM-types.html.

“High-Performance Communication Networks (Second Edition).” Edited by Jean Walrand and Pravin Varaiya, Digital Modulation , www.sciencedirect.com/topics/computer-science/digital-modulation.

“What Is Differential Phase Shift Keying : Modulation and Demodulation.” ElProCus, www.elprocus.com/differential-phase-shift-keying/.