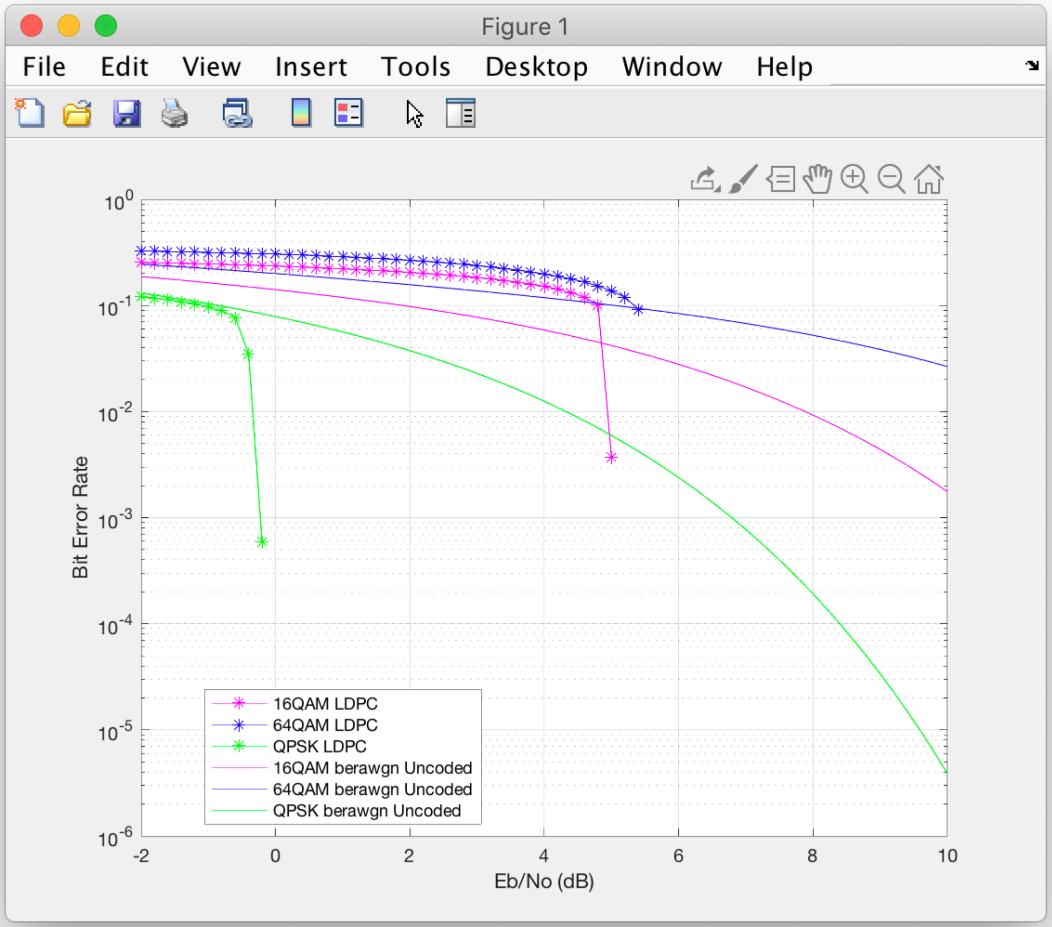
Measuring BER vs. EbNo or SNR for LDPC, Turbo, and Convolutional Codings in MATLAB Simulations

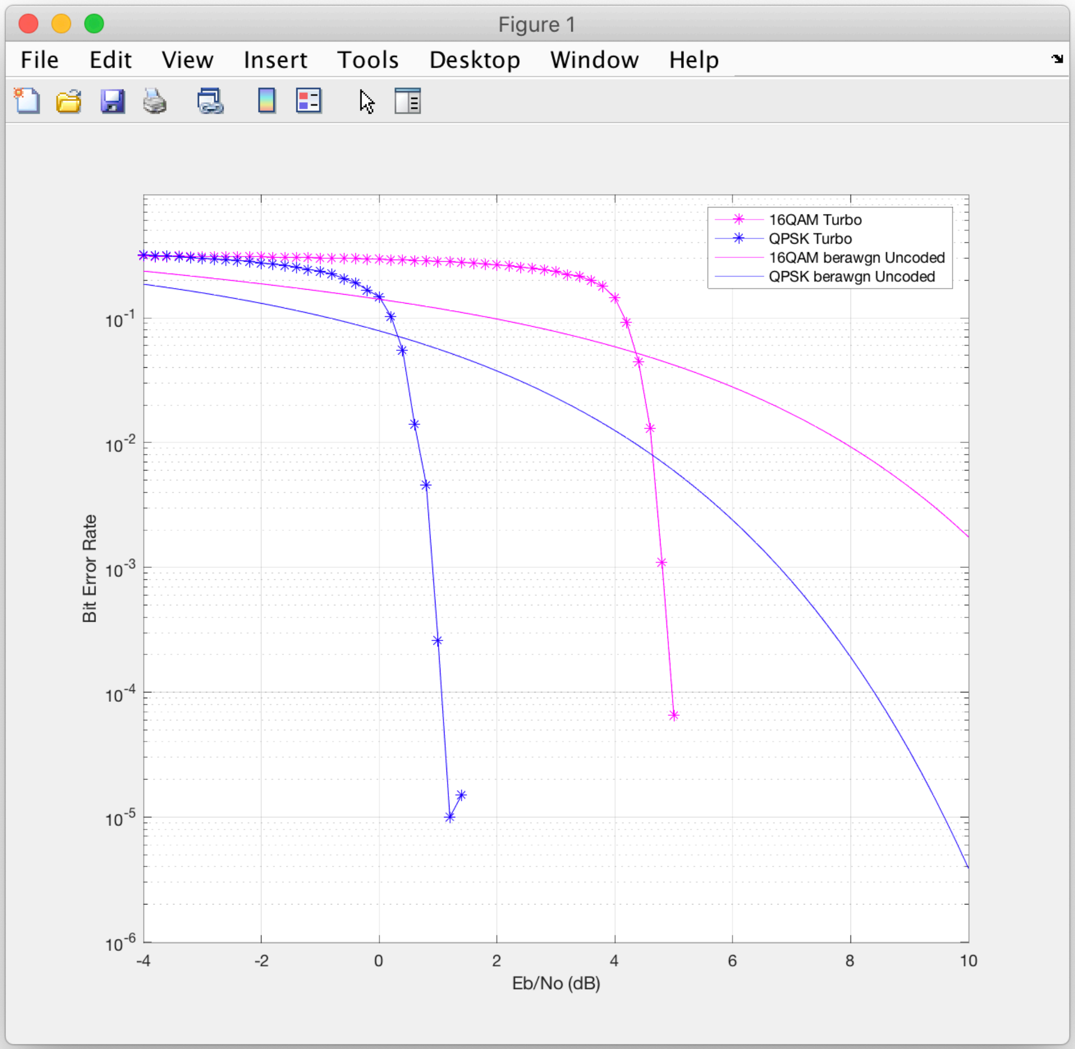
LDPC

The BER vs EbNo values were measured using LDPC coding with QPSK, 16QAM, and 64QAM mappings. The EbNo values range from -2 : 10, with a step size of 0.2. Each combination of LDPC and mapping was ran for 100 frames per EbNo value, yielding a total of 6.48e6 encoded symbols per EbNo value. The code rate for all sets of data is 1/2, and the demodulation decision for the coded channels is approx. llr. The encoded data points represented by stars, are plotted against the MATLAB berawgn function outputs for uncoded mappings. Each mapping is designated a different color: blue for 64QAM, magenta for 16QAM, and green for QPSK.

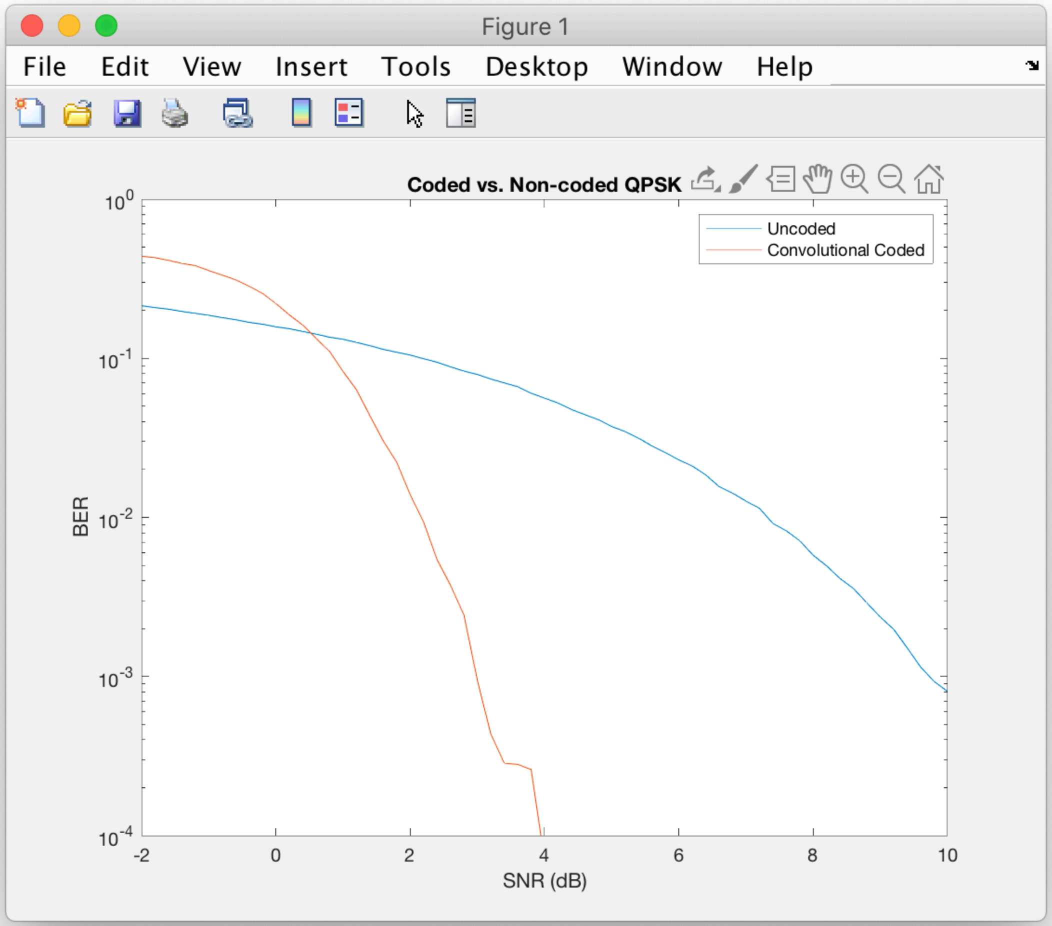


The results for the encoded signals as the EbNo values increased were initially surprising. While, the trend looked normal, the BER values reach 0 quicker than what was expected. Our theory as to why we see BER values reach 0 too early deals with the transmit limit imposed on MATLAB's LDPC Encoder. MATLAB's LDPC encoder uses by default the dvbs2 (digital video broadcasting) for its parity check matrix which accepts a total of 64,800 symbols, allowing the input to the LDPC encoder to be max. 32,400 symbols. Our theory hinges upon the assumption that since we can give a max. Of 32,400 symbols to the input of the LDPC encoder, we are limiting the theoretical amount of error possible under simulated AWGN conditions. This phenomenon would explain why after reaching fixed low-EbNo value, the BER of the signal remains at 0. Since the LDPC encoder has a fixed input, there will be an EbNo value which the LDPC coder/decoder will yield 100% fidelity to the system under simulated conditions. Having a higher EbNo value would only decrease the noise, therefore any EbNo value after the value yielding 100% fidelity will also yield 100% fidelity. To counter this EbNo threshold, we attempt to increase the transmitted symbols, but since there is a max. amount of encoded symbols per MATLAB LDPC encoded transmission we use frames. Each frame contains 64800 encoded symbols, and we summate the error for each frame per EbNo value to calculate the total BER for that EbNo value. However, this summation essentially acts as an averaging calculation across all frames, and since each frame can only produce a BER up to an EbNo threshold the summation does not represent a BER for a longer transmission, but the average BER across multiple 64800 encoded-symbol transmissions. Thus, by increasing the number of frames, we are measuring the max. BER for an LDPC encoded signal with 64800 encoded symbols only, regardless of how many frames we add.

Turbocode
BER vs. EbNo for Turbocoded QPSK and 16QAM mappings. The EbNo ranges from -2 : 10 with a step size of 0.2. Both mappings are ran using 100 frames per EbNo value; however, the QPSK mapped code uses 2e3 input bits per EbNo value, while the 16QAM uses 4e3 input bits per EbNo value. The code rate for both sets of data is 1/3, and the decision method for the coded channels is llr. The coded data is represented by stars, while the uncoded data is generated using MATLAB's berawgn function for standardization purposes. 16QAM is colored magenta, while QPSK is colored blue.



Convolutional Coding
BER vs. SNR for a convolutionally coded QPSK mapping. The SNR ranges from -2 : 10 with a step size of 0.2. Both the coded and uncoded signals ran 2e5 input bits per SNR value. The coded signal uses a code rate of 1/2 with a constraint length of 7, and the decision method for the coded channel is llr. The coded signal is colored orange, while the uncoded signal is colored blue.



The convolutionally encoded signal quickly approaches 0 BER for a relatively low SNR Value. Right around when the SNR passes 1dB is when the convolutional encoding really starts surpassing the uncoded QPSK. This is likely because the added redundancy is more exploitable when it is more easily distinguished from the noise.