Communication Channel Report

James Henrik Middleton (2785209M)

December 2025

1 Introduction

This document details the process of creating a Binary Amplitude Shift Keying Simulation in Python. I chose BASK because of its simplicity compared to other forms of modulation. Due to it being binary, the modulation is very simple. To create a BASK modulated all you need is to create a binary array, and multiply it by a sinusoidal carrier. Although BASK is simple, it is more sensitive to noise and amplitude distortion than phase-based methods such as BPSK. In this simulation, additive noise is introduced to evaluate how the modulated signal degrades under different noise conditions.

2 Method

2.1 Converting Text to Binary

The BASKModulation.py file starts by extracting data from a text file. It converts the string to binary using UTF-8 coding. This means that each character in the txt file is represented by 8 binary digits. Upon encoding the recieved signal must maintain this structure or the decoded message would be illegible.

2.2 Modulation using a Binary Vector

To create a modulated waveform we must first create a carrier wave. This is done within the modulatebask() function. This function takes three parameters: vector, samples per bit, and frequency. The vector is the binary array. The samples per bit are the number of samples taken per bit of data. The frequency component in this case is the number of full carrier cycles per bit. The np.tile function is used to match the total number of samples in the sin wave to that of the vector, creating the "full-carrier" wave. The np.repeat function is used to create repetitions of binary elements in the vector, so it can be multiplied by the full carrier, creating the modulation signal. the multiplication of the full carrier and modulation signals produces the modulated waveform which is used to carry our data.

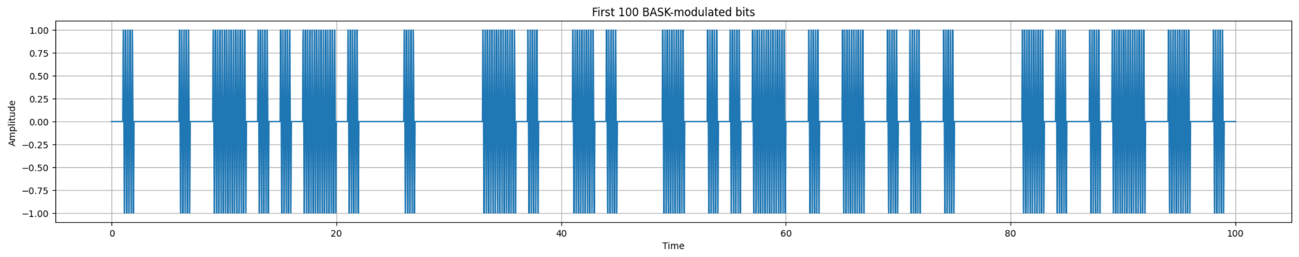


Figure 1: Pure BASK Signal

2.3 Adding Noise to the Signal

To Simulate noise in the signal I used code based on Experimental lab 2. The three Signal to Noise Ratio (SNR) values experimented with here are 6dB, 18dB, and 24dB. In order to add noise into our signal we must calculate the rms amplitude of our modulated wave. To do this we simply use numpy functions to take the root of the mean of the square of our carrier wave (which is represented by a vector). This value is the passed into the additive white gaussian noise function. This adds noise in the correct ratio to the signal such that it conforms to the provided SNR. It converts the SNR to dB into a linear ratio and uses this to compute the required noise RMS value into the carrier wave's RMS value. This is then used to generate gaussian noise using np.random.randn. It then adds noise directly into the modulated wave and returns the new "noisy" signal. This simulates background radiation and white noise which can affect the clarity of communication signals.

$$\text{RMS of a signal } x(t) = \sqrt{\frac{1}{T}\int_0^T x(t)^2\,dt}$$

$$\text{Discrete RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

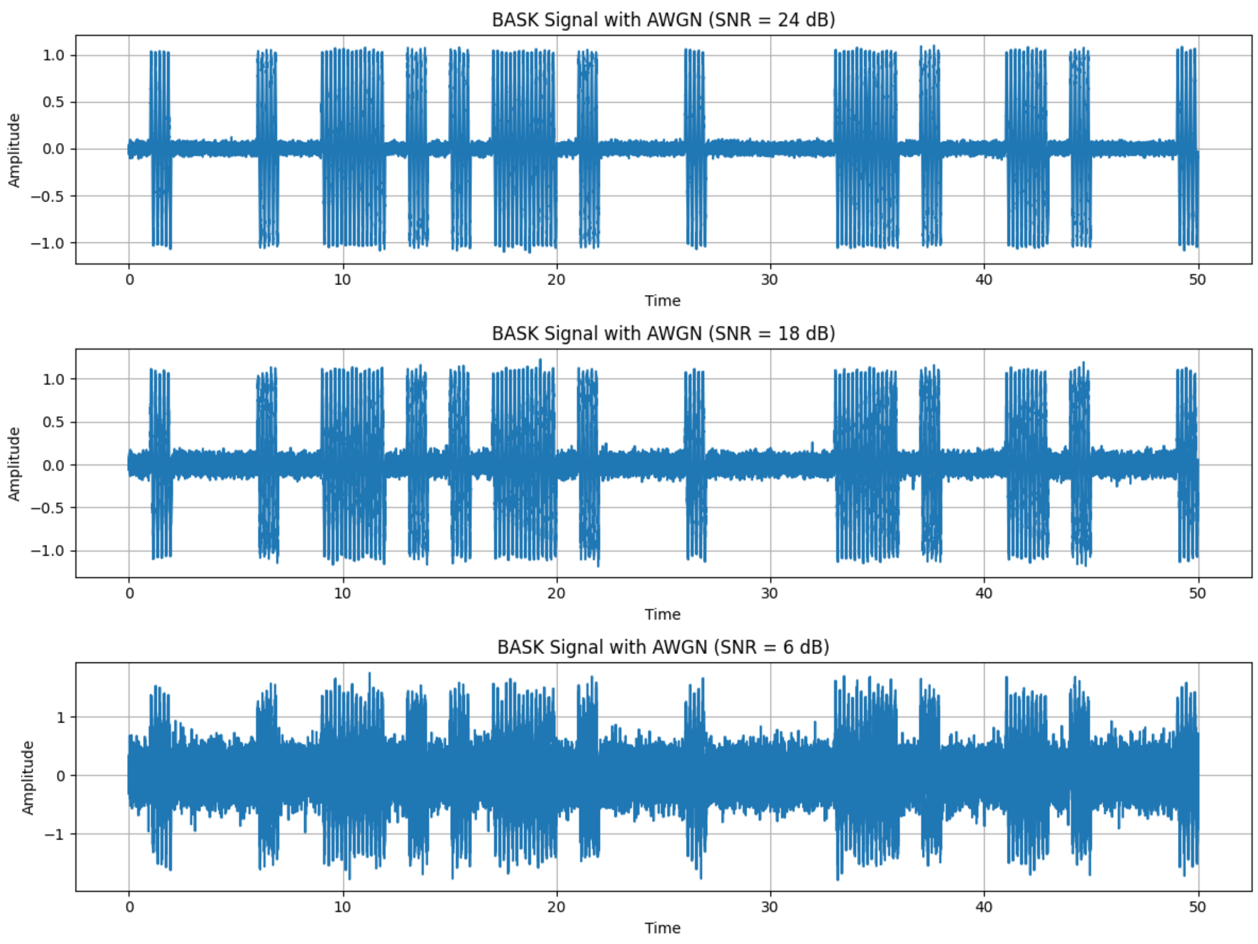


Figure 2: Noisy Signals

2.4 Bandpass Filtering

In order to improve the clarity of the noisy signal we have to perform filtering. Filters can be high pass, low pass, bandpass, or band stop. For this Simulation I have implemented a bandpass filter. A bandpass filter removes edge frequencies from a signal whilst allowing a selected middle band to pass. As per the specifications of the pass I have attempted to design a filter that only allows 20% of the carrier frequenct through. I set the carrier frequency to 5Hz and set the bandpass filter to only pass a bandwidth of between 4.5Hz and 5.5Hz ($\frac{1}{5} = 20\%$).
The bandpass filter function has 4 parameters: lowcut, highcut, fs, and numtaps. Lowcut and highcut are the cutoff frequencies. Within the function lowcut and highcut are normalized through division by the sampling rate (fs) to give cycles per bit - which is how this system measure frequency. Numtaps is the number of coefficients within the filter. It determines the filter's length, sharpness, and computational cost. Larger numtap values typically give a narrower transition band but with greater delay. Numtaps is typically chosen to be an odd number so the band can be symmetric around a central point. This band is then given a size by calculating the difference between the high pass and low pass. This value is then multiplied by a hamming window to produce a cleaner frequency with reduced ripple. The normalization calculation at the end of the function evaluates the filter's frequenct response and rescales the coefficients so the filter has unity gain, ensuring that the passband amplitude is correctly normalized. The passband is then returned from the function.

$$n = [\,0, 1, 2, \ldots, (\text{numtaps} - 1)\,] \;-\; \frac{\text{numtaps} - 1}{2}$$

$$H(\omega) = \sum_{n=0}^{\text{numtaps}-1} h[n]\, e^{-j\omega n}$$

$$h[n] \leftarrow \frac{h[n]}{|H(\omega)|}$$

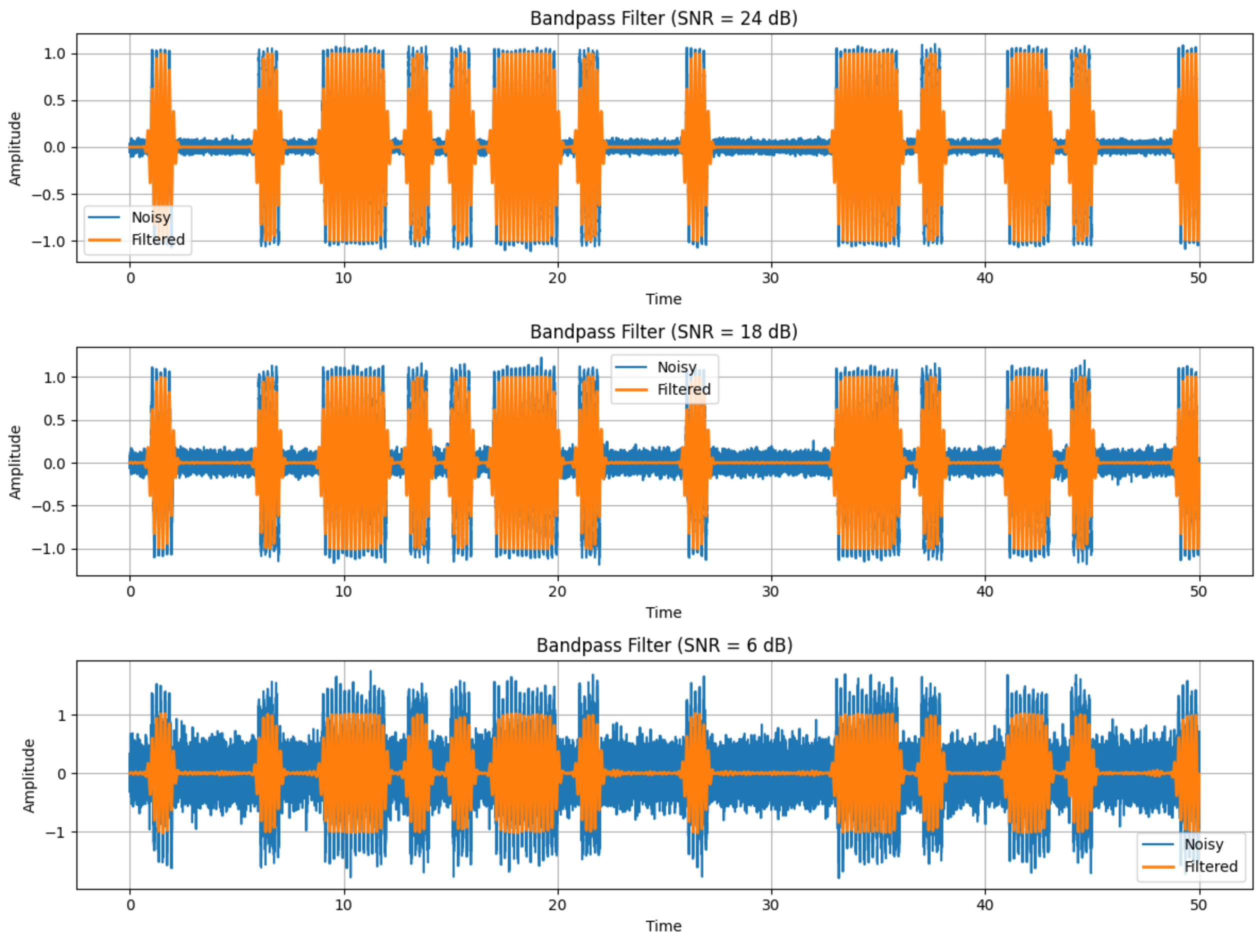
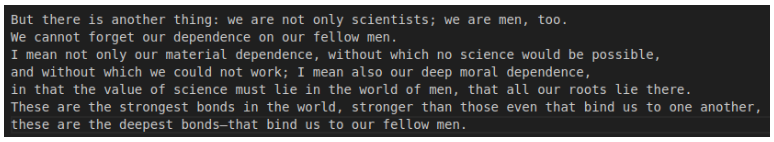


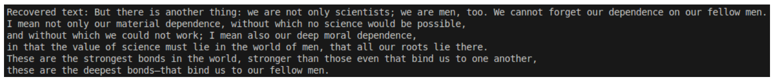
Figure 3: Filtered Signals

2.5 Recovering Signal Data

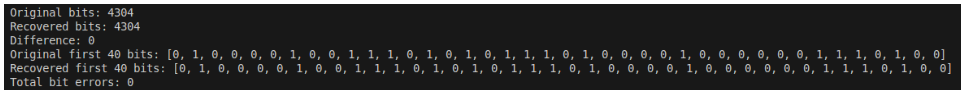
In order to retrieve data from the transmitted signal it must first be demodulated into its constituent frequencies. The demodulate bask() function demodulates the signal by dividing the input waveform into blocks corresponding to individual bit intervals, each containing "samples per bit" samples. For each block, it performs simple envelope detection by computing the mean absolute amplitude, which estimates the signal's strength for that bit period. It then compares this amplitude to a threshold (default 0.4): amplitudes above the threshold are interpreted as bit 1, and those below as bit 0, just like the previous functions. The function returns the sequence of recovered binary values. These binary values are then put through UTF-8 decoding to reproduce our text file.


```
But there is another thing: we are not only scientists; we are men, too.
We cannot forget our dependence on our fellow men.
I mean not only our material dependence, without which no science would be possible,
and without which we could not work; I mean also our deep moral dependence,
in that the value of science must lie in the world of men, that all our roots lie there.
These are the strongest bonds in the world, stronger than those even that bind us to one another,
these are the deepest bonds—that bind us to our fellow men.
```

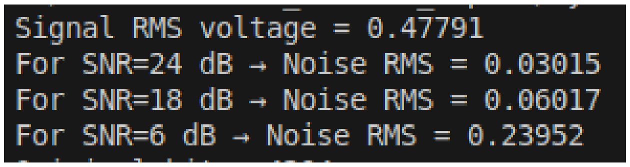
Original Text


```
Recovered text: But there is another thing: we are not only scientists; we are men, too. We cannot forget our dependence on our fellow men.
I mean not only our material dependence, without which no science would be possible,
and without which we could not work; I mean also our deep moral dependence,
in that the value of science must lie in the world of men, that all our roots lie there.
These are the strongest bonds in the world, stronger than those even that bind us to one another,
these are the deepest bonds—that bind us to our fellow men.
```

Reconstructed Text


```
Original bits: 4304
Recovered bits: 4304
Difference: 0
Original first 40 bits: [0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
Recovered first 40 bits: [0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
Total bit errors: 0
```

Total Bit Error (The same for all SNR)


```
Signal RMS voltage = 0.47791
For SNR=24 dB → Noise RMS = 0.03015
For SNR=18 dB → Noise RMS = 0.06017
For SNR=6 dB → Noise RMS = 0.23952
```

SNR Values

3 Conclusion

This Simulation of BASK seems technically robust. However, there are several issues. The noise implementation was insufficient to simulate bit error. This may be corrected by increasing the SNR values. Another issue is that only AGWN was attempted and no other forms of signal loss or disruption as would be present in an urban environment. With that being said the actual implementation of BASK is clean. From a technical standpoint the simulation is successful, but from a testing point of view it could be improved.