Monte Carlo simulations in wireless communication

BER estimation and SNR envelope

Giacomo Calabria – 12^{th} April 2024

Introduction

Why we need Monte Carlo simulation?

Signal in Wireless Channels

- Wireless communications systems facilitate the transmission of information through radio waves. These radio waves travel through various channels, including air, buildings, and foliage.
- Channel impairments like fading, noise, and interference can significantly degrade the transmitted signal.
  - **Fading**: Signal strength fluctuates due to multipath propagation and environmental factors.
  - **Noise**: Unwanted electrical signals introduced during transmission, corrupting the data.
  - **Interference**: Overlapping radio signals from other sources disrupt the desired signal.

Monte Carlo simulation

- Traditional analytical methods for analysing communication systems can be unmanageable or even intractable in complex scenarios.
- Monte Carlo simulations offer a powerful and versatile approach to overcome these limitations. They rely on repeated random sampling to statistically evaluate system performance under various channel conditions.
- Advantages:
    - Handles complex channel models like Rayleigh fading and Rician fading.
    - Accounts for non-linearities and system imperfections.
    - Provides valuable insights into system behaviour for diverse scenarios.

The wireless channel

Statistical model of the wireless channel

Wireless AWGN Channels

We will assume an **Additive White Gaussian Noise (AWGN)** channel, in this environment the error events arising from channel noise are **independent** and its described by a Gaussian distribution.

Using Monte Carlo simulation to estimate a performance parameter of a communications system, such as **Bit Error Rate (BER)**, if an estimator is *unbiased*, we know that, on the average, Monte Carlo simulation provides the correct result.

We use Monte Carlo simulation in communication system with these assumptions:

- The channel is assumed AWGN, noise is the only source of error.
- Data symbols at the source are independent and equally probable.
- No intersymbol interference.

Evaluate the Channel: SNR

The Signal-to-Noise Ratio (SNR) at the receiver is usually a major factor in determining the performance of the system. It is defined as the ratio of the power of a signal to the power of background noise.

In order to compare the BER performance we use a normalized SNR, also known as the «SNR per bit», which is defined as:

$$SNR = \frac{E_b}{N_0}$$

Where E_b is the energy associated with each user data bit and N_0 is the noise spectral density (noise power).

Evaluate the Channel: BER

The parameter we use to analyse our communication system is the **Bit Error Rate (BER)** which is defined as the ratio of the number of error bits to the total number of bits transmitted during a specific period.

$$\mathrm{BER} = \frac{N_e}{N}$$

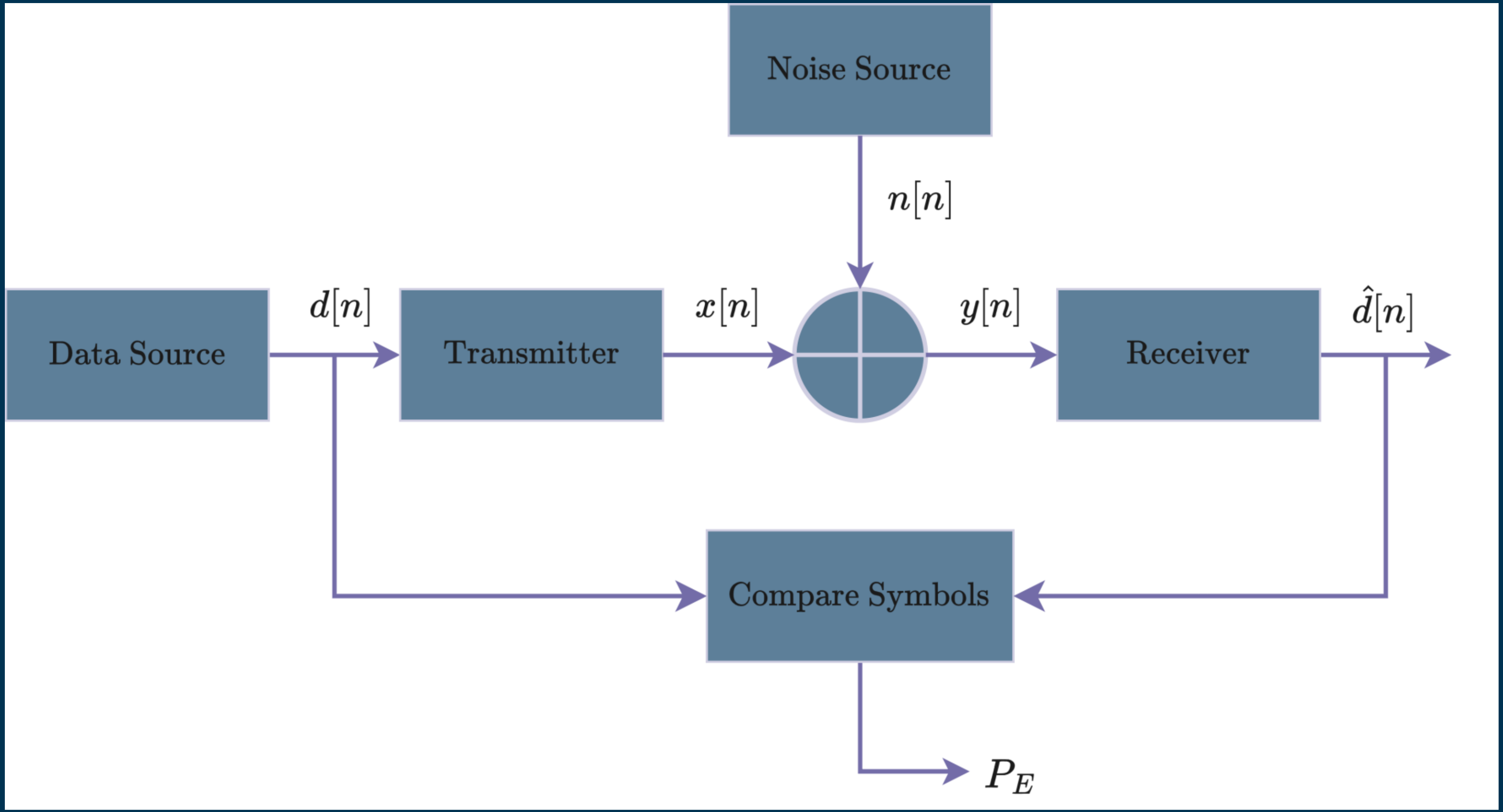
It is the likelihood that a single error bit will occur within received bits, independent of the rate of transmission.

The theoretical relationship between the SNR and the BER for a BPSK scheme is given by:

$$BER = Q\left(\sqrt{2SNR}\right)$$

where Q is the Q-function

AWGN channel



Simple modulation scheme

Binary Phase Shift Keying, BPSK

The simplest modulation scheme. The two symbols of the constellation are:

$$x_d[n] = \cos\left(\pi d[n]\right) = \begin{cases} 1, & d[n] = 0 \\ -1, & d[n] = 1 \end{cases}$$

Which delineates two decision regions. The receiver threshold is zero, the case for equally probable signals. The decision rule is:

$$\hat{d}[n] = \begin{cases} 0, & y_d[n] > 0 \\ 1, & y_d[n] < 0 \end{cases}$$

Binary Frequency Shift Keying, BFSK

In this scheme, the two information symbols has direct and quadrature components are defined as:

$$x_d[n] = \cos\left(\frac{\pi}{2} d[n]\right) \quad x_q[n] = \sin\left(\frac{\pi}{2} d[n]\right)$$

The signal space is two dimensional, so the decision rule becomes:

$$\hat{d}[n] = \begin{cases} 0, & y_d[n] > y_q[n] \\ 1, & y_d[n] < y_q[n] \end{cases}$$

Monte Carlo simulation setup

Step in the Monte Carlo simulation

Set up

- Modulation scheme
- SNR steps
- #TX symbols

Random data

Generate the random input data to be encoded.

Generate a sample of the AWGN noise with a given σ

Evaluate

Compare IN data with OUT data and check errors.

We count the total number of error to estimate BER.

Repeat 2 & 3

On each step we vary the SNR, and we store the total number of errors.

Analyse results

Analyse results so on each step of SNR compute the BER.

Plot the graph

SNR vs BER

MC simulation parameter

Vary the SNR form 0 dB to 10 dB with steps of 0.5 dB.

$$\sigma = \sqrt{\frac{1}{2}\frac{1}{SNR}} \quad (*)$$

We perform *Nsymbol* iteration of the transmission on each SNR step and we account how many times the channel decodes the information wrongly, in order to compute the BER. We choose to simulate with various number of iteration:

$$10^3;\ 10^4;\ 10^5;\ 10^6$$

Then we will compare the resulting BER using both BPSK and BFSK modulation.

(*) Energy and sampling frequancy are normalized.

Python code

Here some snaps of the main parts of the code:

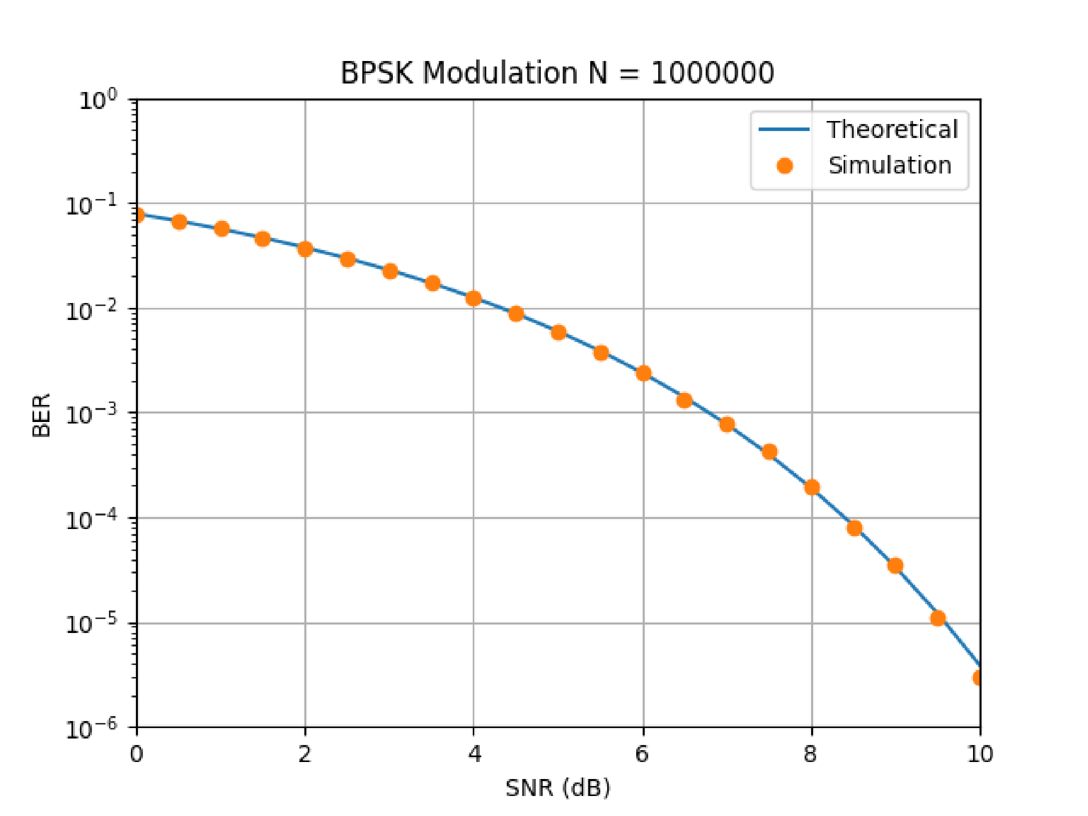
```python
def qfunc(x):
    return 0.5-0.5*sp.erf(x/math.sqrt(2))

# Define SNR parameters
snrdb_min = 0
snrdb_max = 10
# SNR steps
snrdb = np.arange(snrdb_min, snrdb_max +
0.5, 0.5)
# SNR values in linear scale
snr = 10**(snrdb / 10)
# Number of symbols to simulate
Nsymbols = 10000000
```

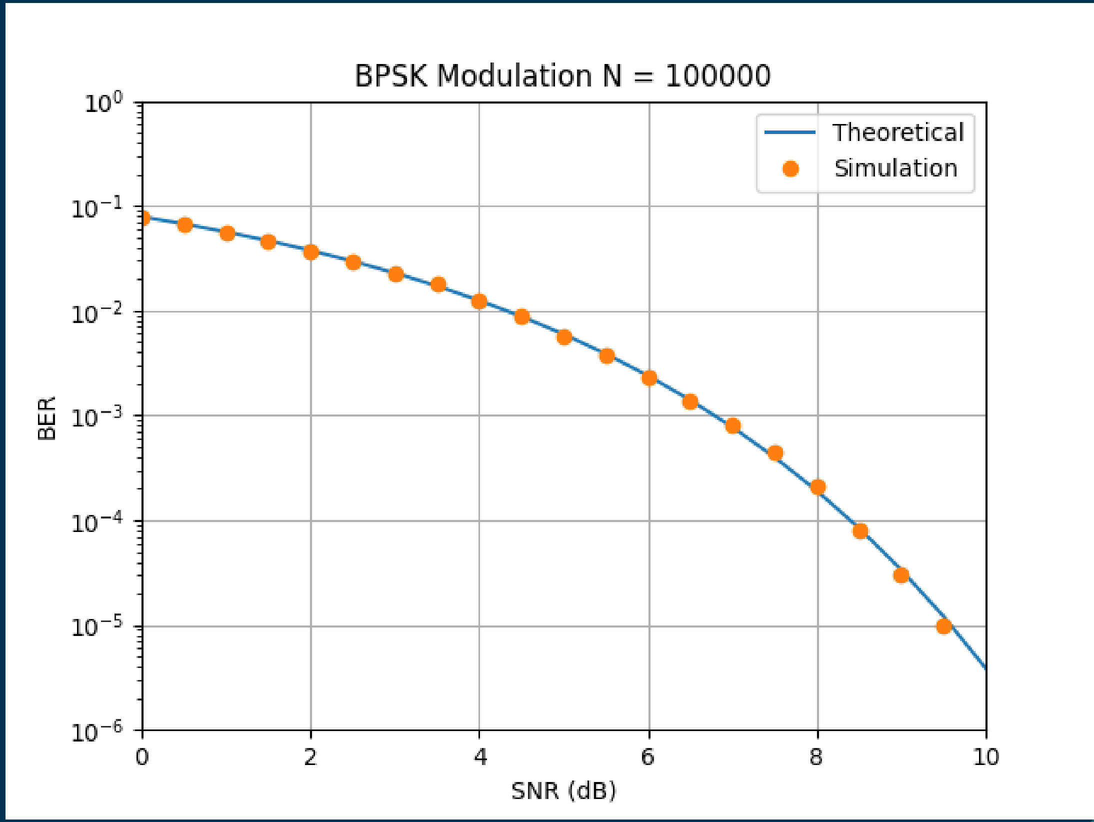
```python
for j, snr_db in enumerate(snrdb):
    sigma = np.sqrt(1 / (2 * snr[j]))  # Noise standard deviation
    error_count = 0

    for _ in range(Nsymbols):        # Simulation loop
        d = np.random.randint(0, 2) # Generate random data bit (0 or 1)
        x_d = 2 * d - 1             # BPSK modulation: -1 for 0, 1 for 1
        n_d = sigma * np.random.randn() # Add Gaussian noise
        y_d = x_d + n_d
        if y_d > 0:                 # Make decision
            d_est = 1
        else:
            d_est = 0
        if d != d_est:              # Count errors in decoding
            error_count += 1

    errors[j] = error_count  # Store error count for each SNR level

ber_sim = errors / Nsymbols           # Simulated BER
ber_theor = qfunc(np.sqrt(2 * snr))  # Theoretical BER (Q-function)
```

BER simulation results

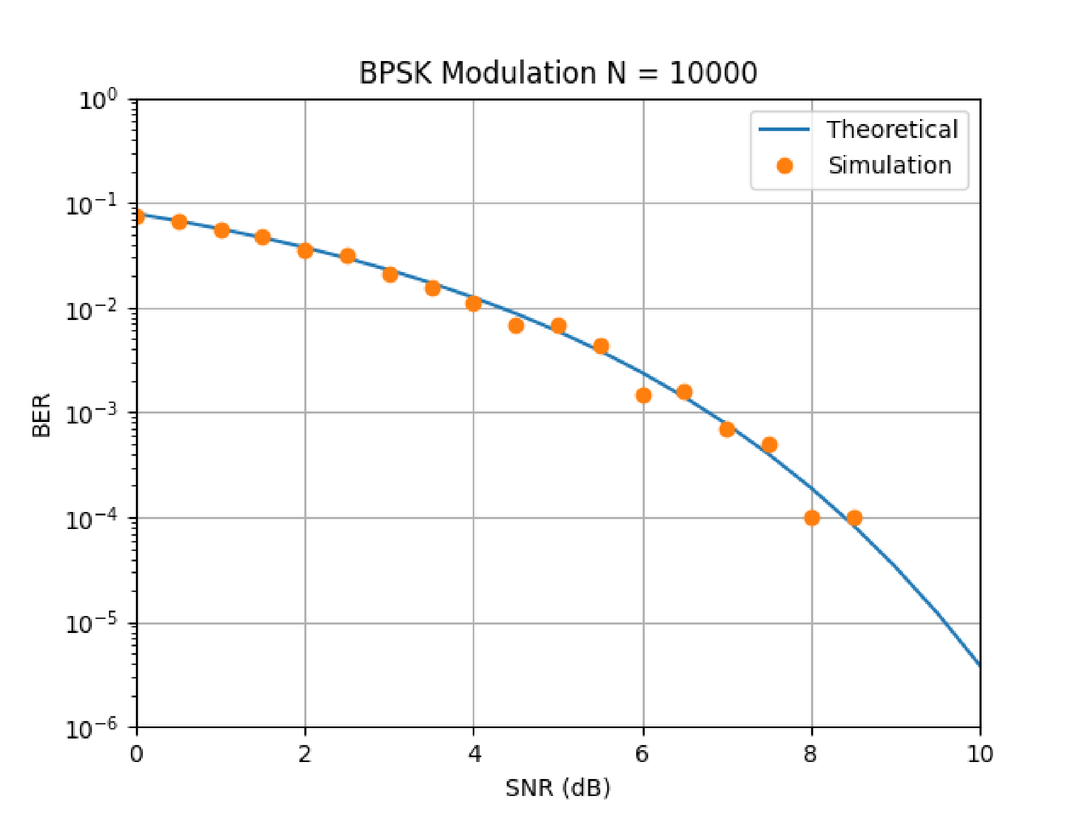
Comparing different modulation schemes and number of iterations

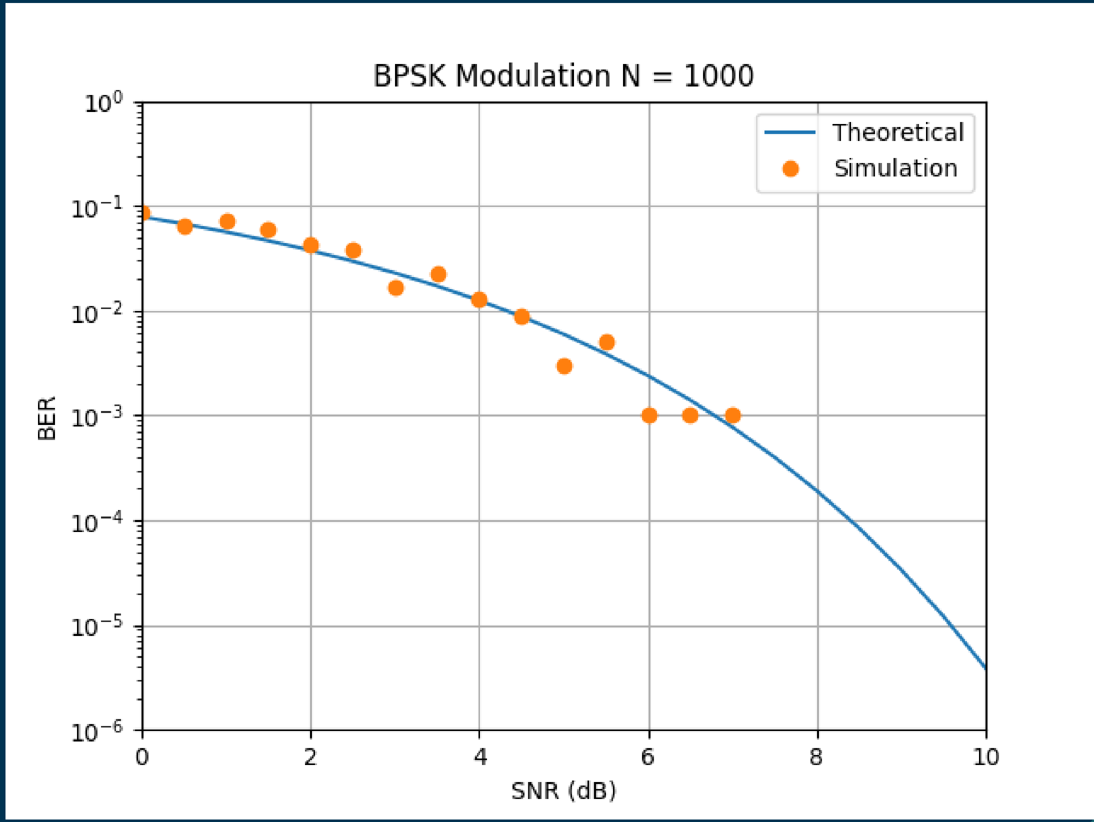
Varying the number of iteration





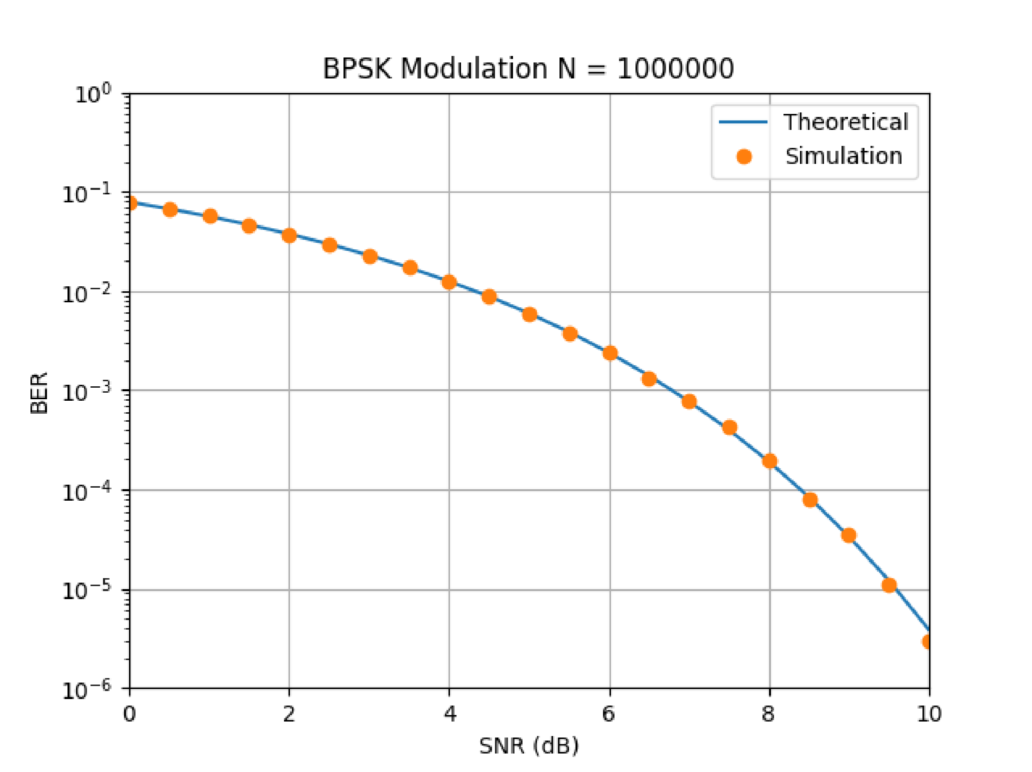
Varying the number of iteration



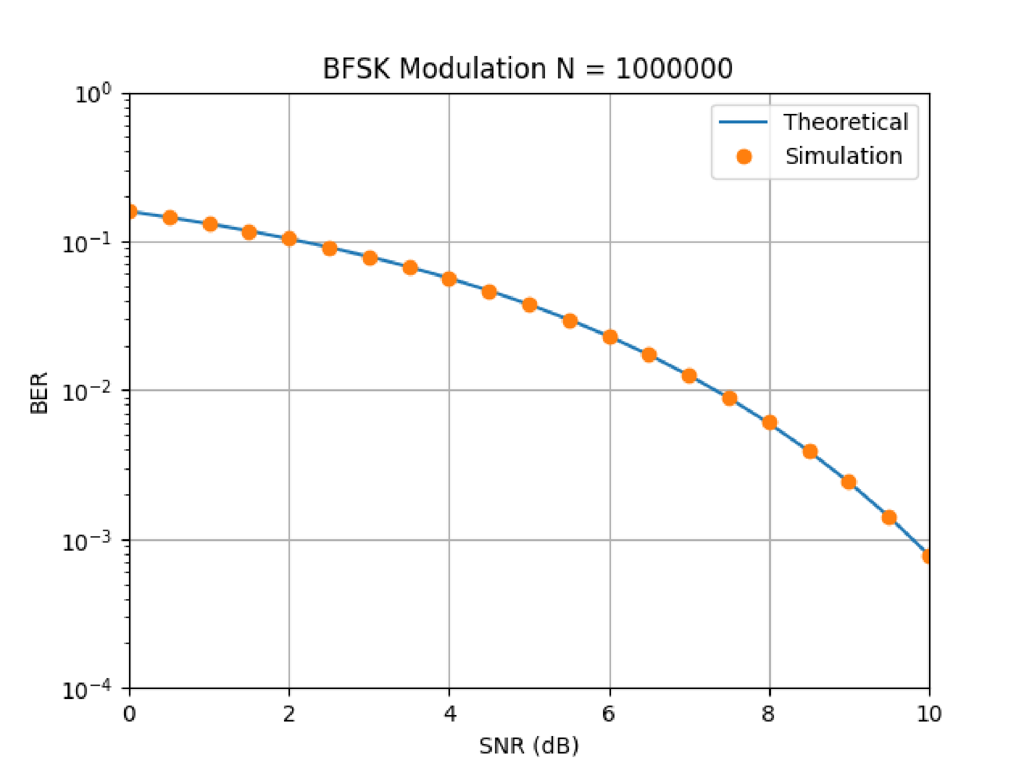


Compare two modulation schemes

Binary Phase Shift Keying, BPSK

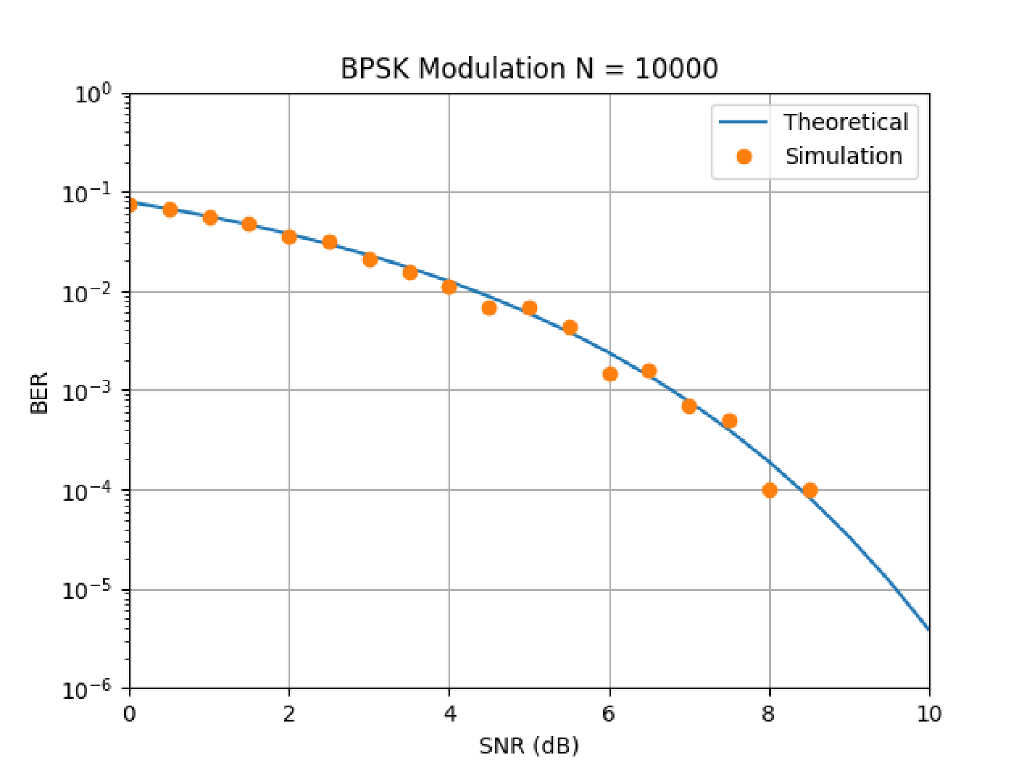


Binary Frequency Shift Keying, BFSK

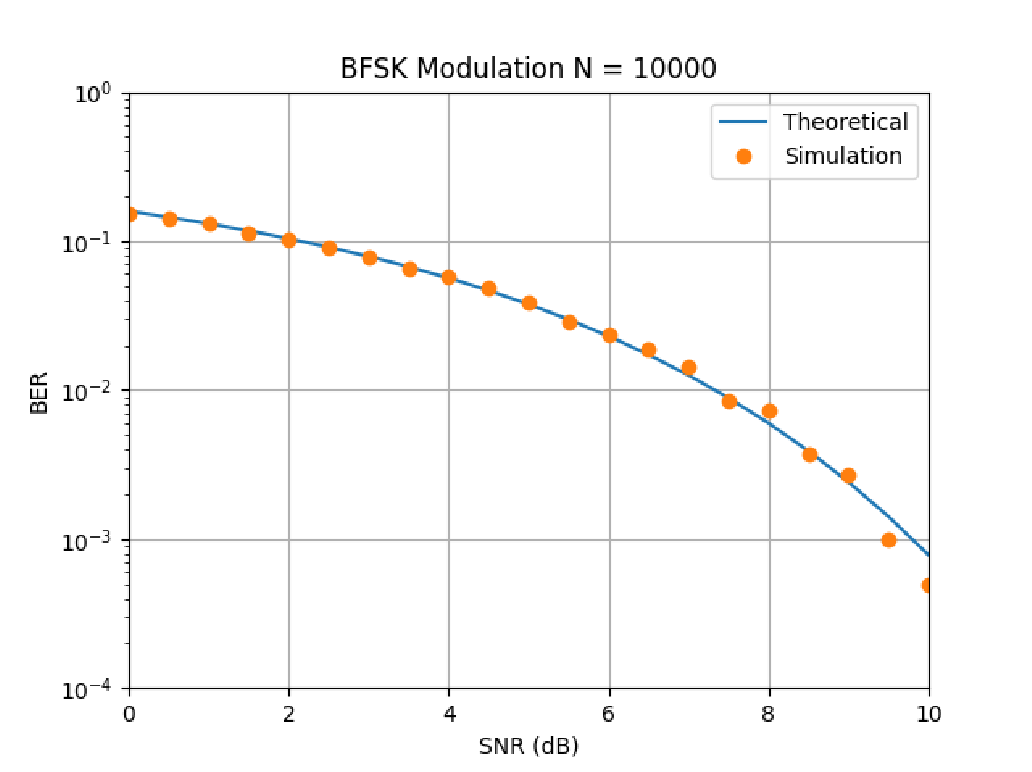


Compare two modulation schemes

Binary Phase Shift Keying, BPSK



Binary Frequency Shift Keying, BFSK



Monte Carlo simulation of SNR envelope

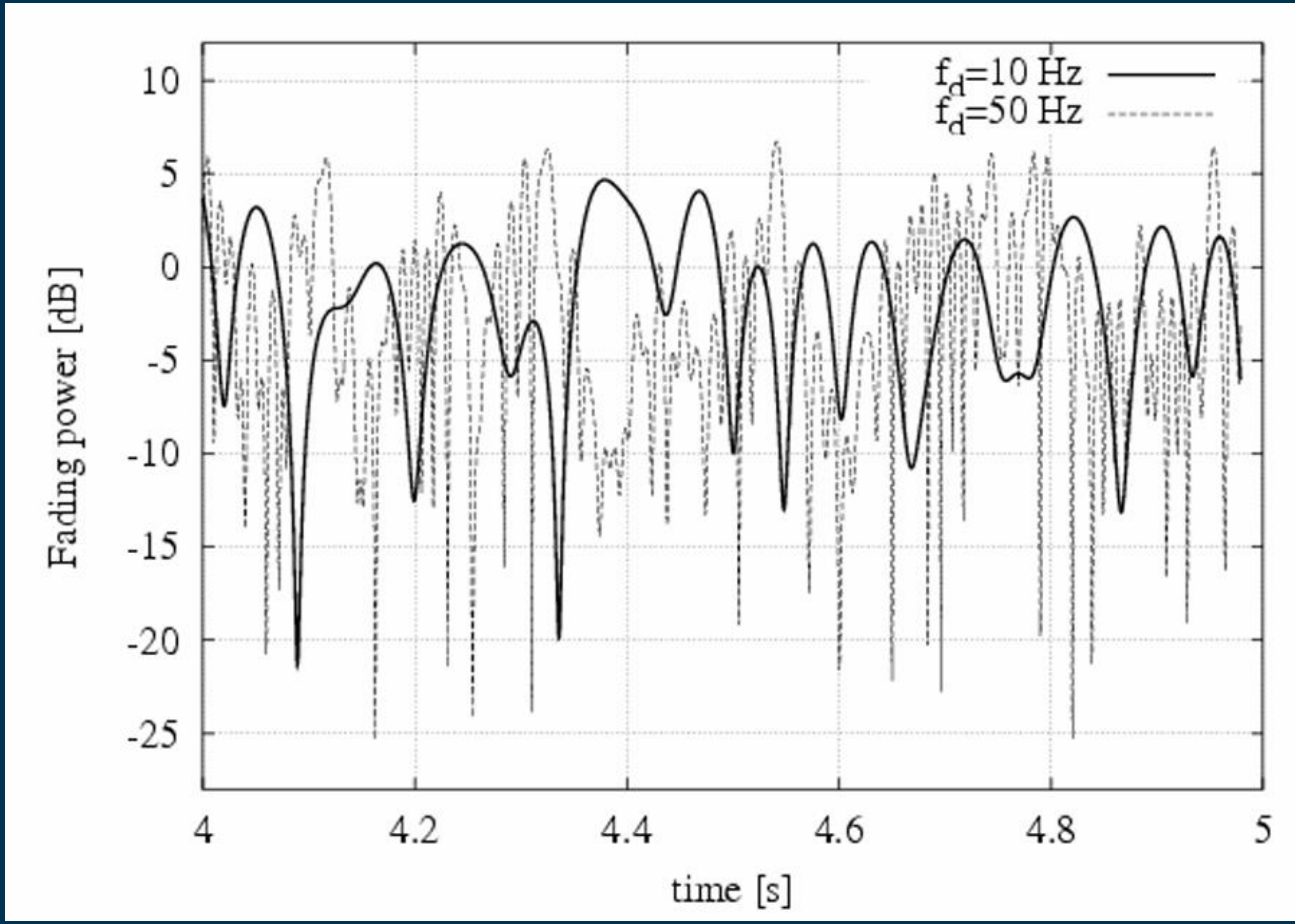
It is right to consider SNR constant over iterations?

SNR envelope

As the wireless environment is not static, using just the average SNR is not enough to track the temporal evolution of the fading process: temporal dynamics (correlation).

The SNR envelope is affected by fluctuations over time.

These fluctuations can be described by an Finite State Markov Chain Monte Carlo model.

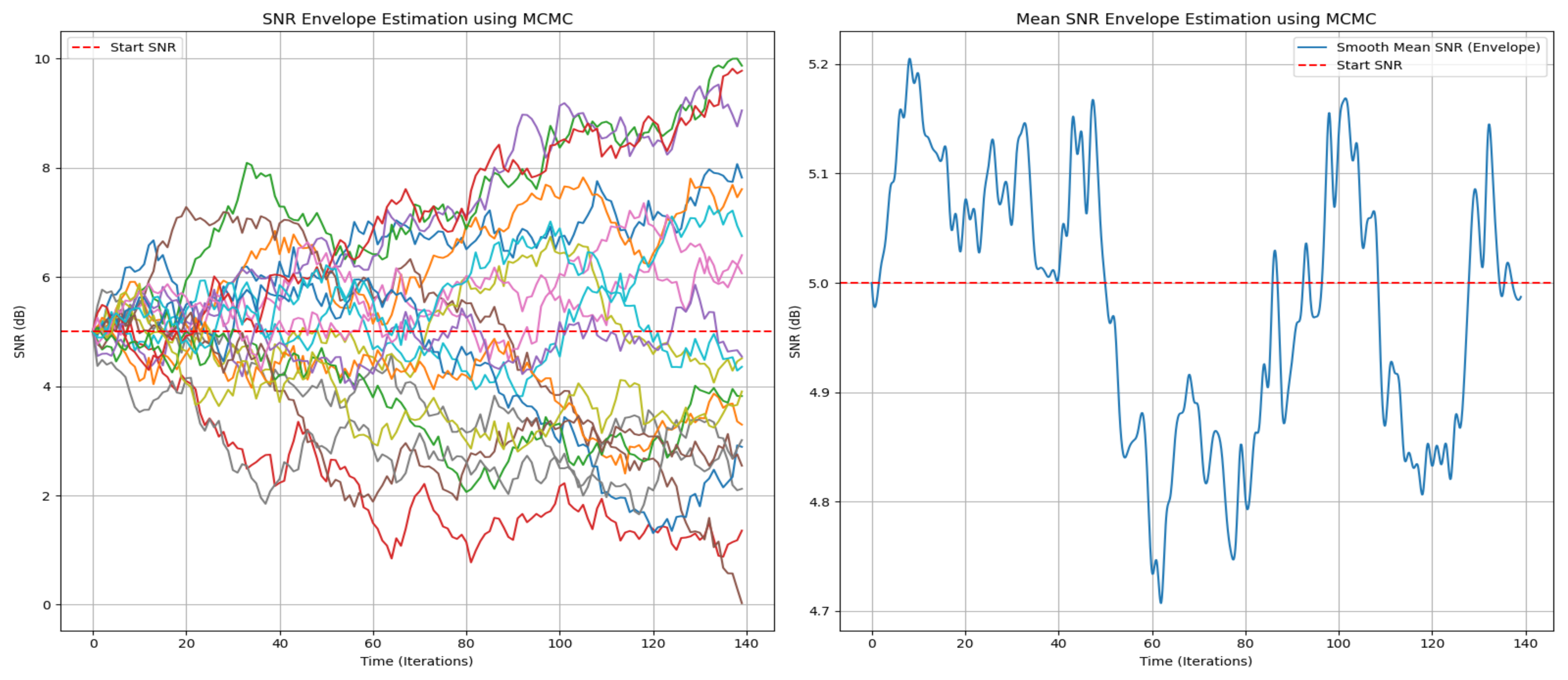


Python code

```python
for j in range(num_averaging):
    for i in range(1, num_iterations):
        sigma = np.sqrt(1 / (2 * (10**(snr_db_levels[i-1,j] / 10))))
        noise = sigma * np.random.randn(num_samples)
        [...]
        # Random walk proposal
        snr_db_proposed = snr_db_levels[i - 1,j] + np.random.normal(scale=0.2)

        likelihood_current = np.exp(-0.5 * np.sum((noise) ** 2) / (10 ** (-snr_db_levels[i-1,j] / 10)))
        likelihood_proposed = np.exp(-0.5 * np.sum((noise) ** 2) / (10 ** (-snr_db_proposed / 10)))

        # Aacceptance criterion
        acceptance_prob = min(1, likelihood_proposed / likelihood_current)
        if np.random.rand() < acceptance_prob:
            snr_db_levels[i,j] = snr_db_proposed
        else:
            snr_db_levels[i,j] = snr_db_levels[i - 1,j]
```

$$\mathcal{L} = \exp\left(-\frac{1}{2}\frac{\|\mathbf{n}\|^2}{\mathrm{SNR}}\right)$$

SNR envelope results



Conclusions

Is it worth using Monte Carlo simulation in wireless communication?

Using MC simulation in wireless communication systems

Monte Carlo technique leverage on the performance of stochastic, or random, experiments.

- Traditional mathematical analysis may be ineffective in obtaining a description of a system;
- Monte Carlo techniques provides a results that are represented as random variable; the properties of this random variable, such as the mean, convey valuable information about the quantity being simulated.

While Monte Carlo techniques can yield fast simulation results, they can also produce inaccurate outcomes if not executed with care.

However, by integrating mathematical analytical methods with Monte Carlo simulations, it's possible to achieve fast and accurate results.

References

- Poniran, Mazlina. *"Monte Carlo simulation in communication system."* (2009)
- Rossi, Michele. *"Link Layer Errors Models"*, lecture notes of Wireless Networks, University of Padua. 2023
- William H.Tranter, K.Sam Shanmugam, *"Communication system simulation with wireless Application,"* Prentice Hall

Thank you!
Gracias!
Grazie!

Giacomo Calabria