

# Tutorial

## Codificação Turbo no Simulink

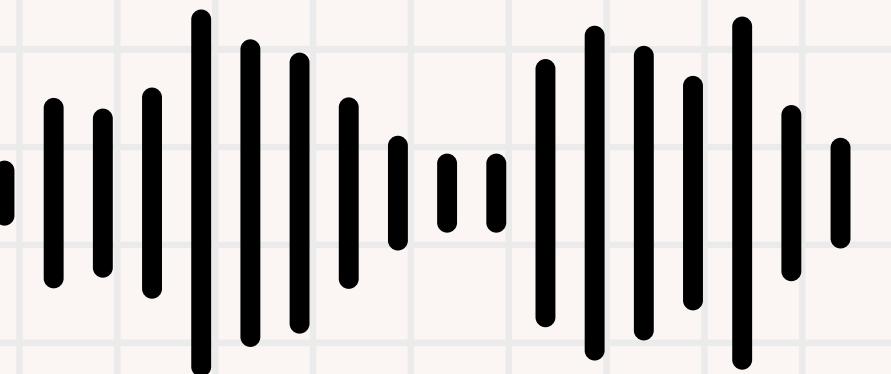
Laboratórios Didáticos para Ensino de Sistemas de  
Comunicação

Materiais

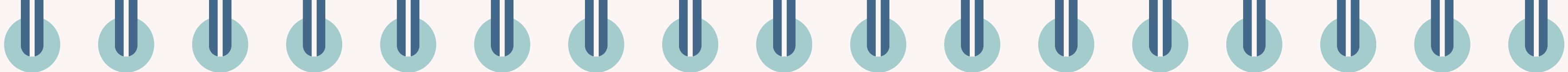
# Pré-requisitos



**MATLAB + Simulink  
instalados**



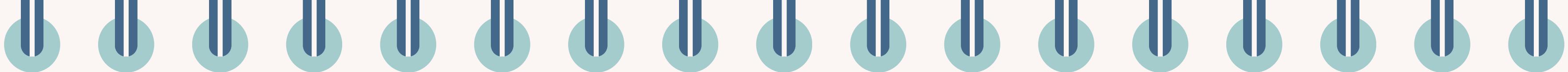
**Conceito rápido de  
codificação TURBO**



## Introdução

# Simulink

A codificação turbo combina dois ou mais codificadores convolucionais e utiliza um processo de decodificação iterativo para melhorar a correção de erros. No Simulink, é possível modelar facilmente o sistema de codificação turbo, realizar simulações para validar o desempenho, garantindo a eficiência na transmissão de dados, mesmo em canais ruidosos.



## Introdução

# Fluxo de trabalho com ModelSim



Modelagem no Simulink (Codificação turbo)



Simulação



Desenvolvimento



# Passo a Passo



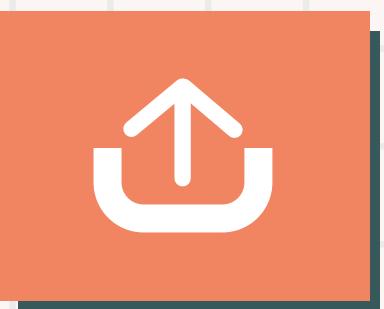


Desenvolvimento

# Simulink

## Arquivo do projeto

Para facilitar, é disponibilizado arquivo do modelo em Simulink no Github:

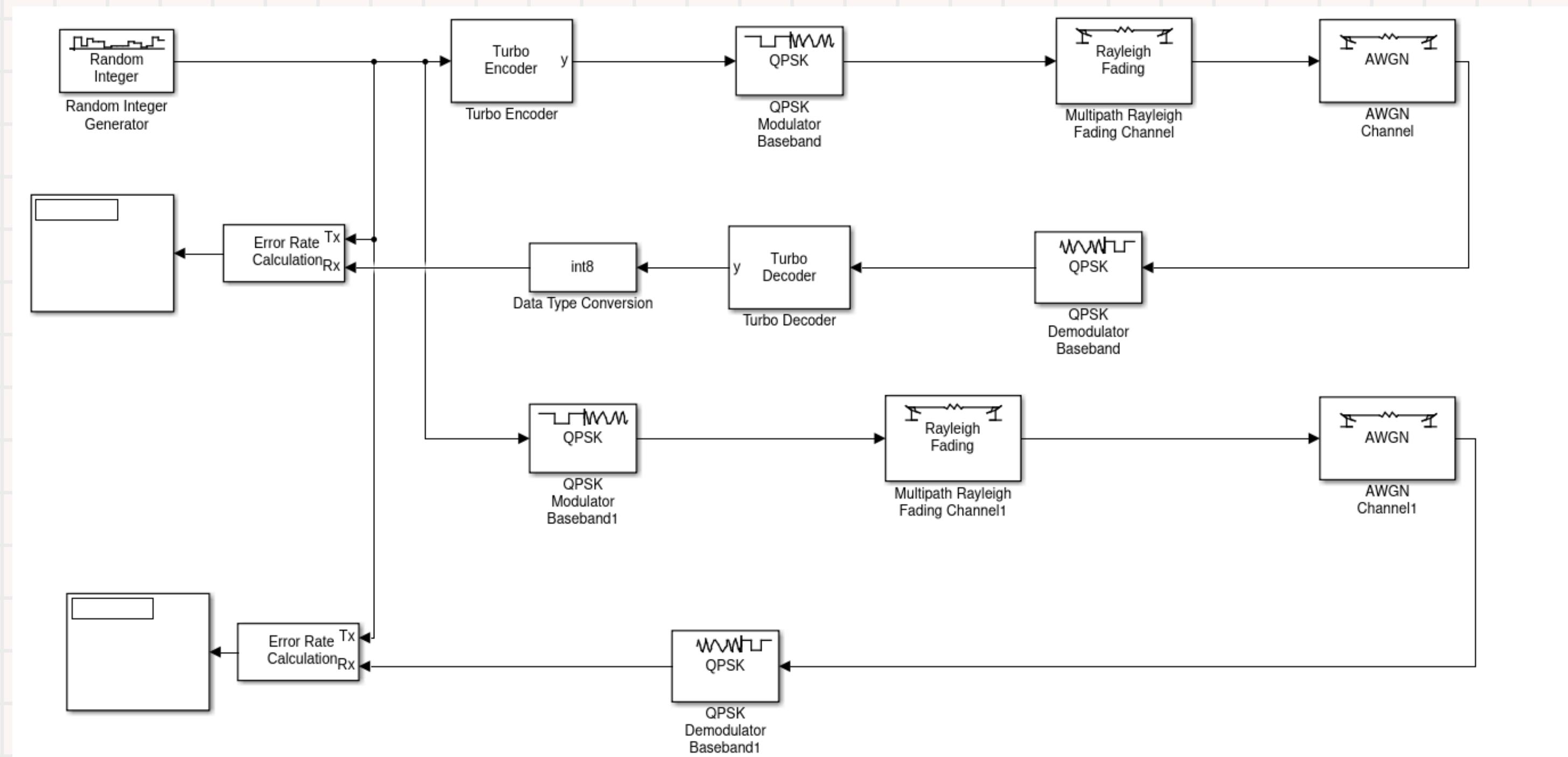


[Link Download](#)



Desenvolvimento

# Codificação TURBO - Sistema





Desenvolvimento

# Codificação TURBO - Sistema

No projeto você já encontra:

- **Fonte de bits (Random Integer Generator):** Gerando bits aleatórios de 0 a 1.
- **Codificador Turbo:** Consiste em dois codificadores convolucionais em série e um intercalação entre eles. Após a codificação, teremos a sequência de bits codificada, que será transmitida.
- **Modulação QPSK:** Utilizado para mapear os bits para símbolos QPSK.
- **Canal Multipath Rayleigh:** Simula a propagação de ondas de rádio que chegam ao receptor por diferentes caminhos, com diferentes atrasos.
- **Canal AWGN:** Adiciona ruído branco gaussiano ao sinal transmitido.
- **Demodulação QPSK:** Transforma os símbolos de volta em bits.
- **Decodificador Turbo:** Responsável por corrigir os erros usando o processo de decodificação iterativa.
- **Cálculo da Taxa de Erro:** Compara os bits transmitidos com os bits recebidos, calculando a taxa de erro de bits (BER).



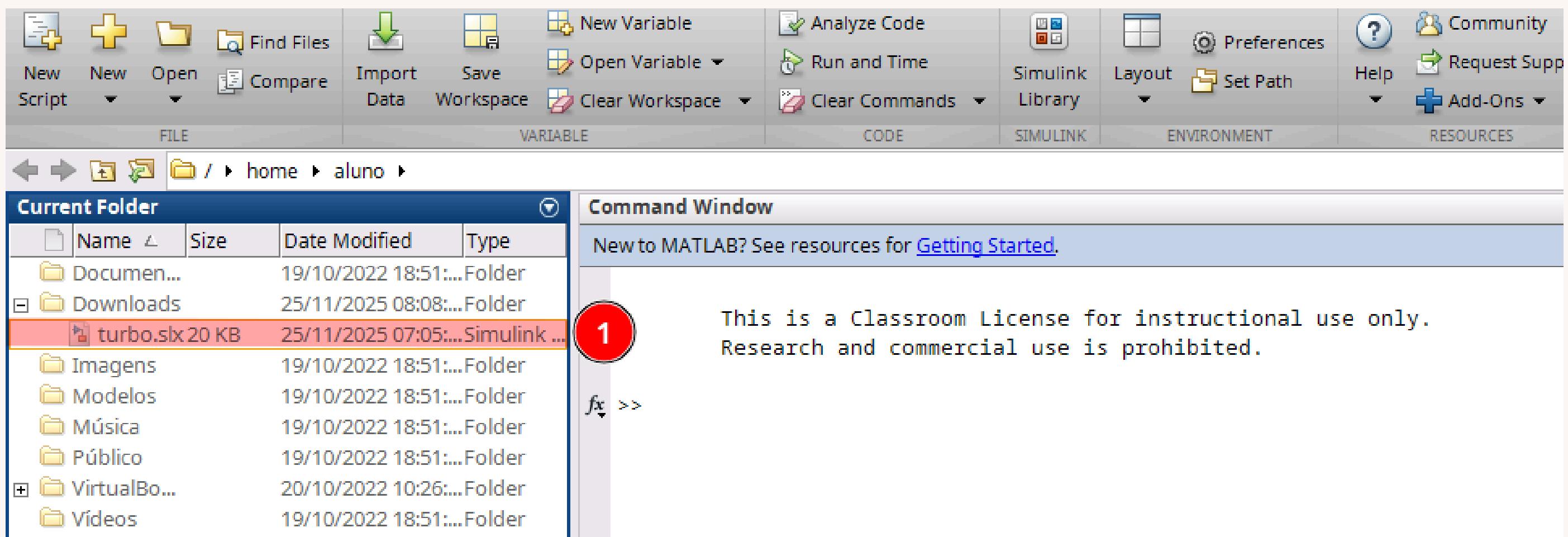
## Desenvolvimento



A seguir a demonstração de como usar o Simulink...



## Desenvolvimento



Abra o Matlab

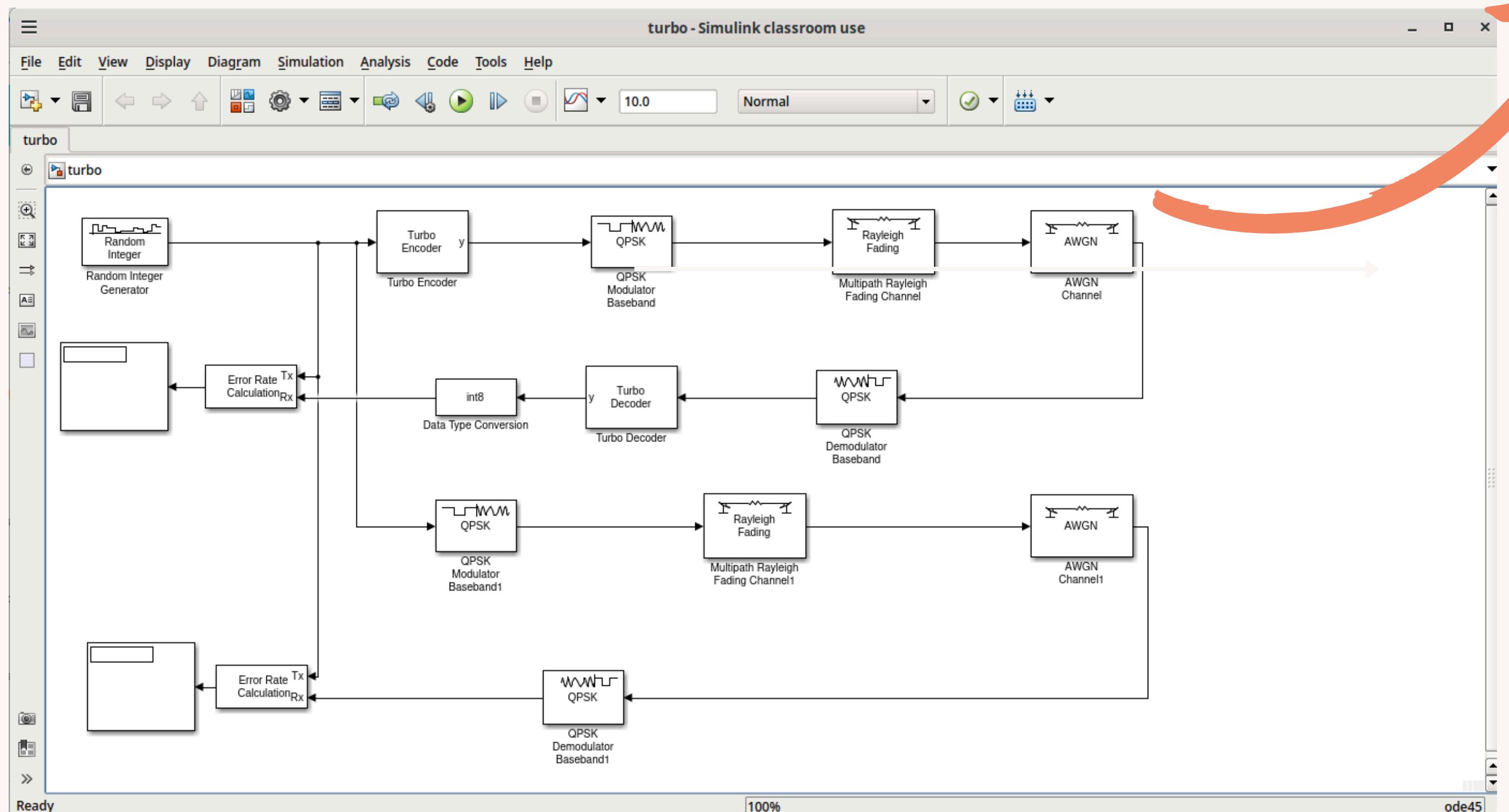
Vá até ao diretório onde foi salvado o arquivo **tubo.slx**

1 - Dê um duplo click



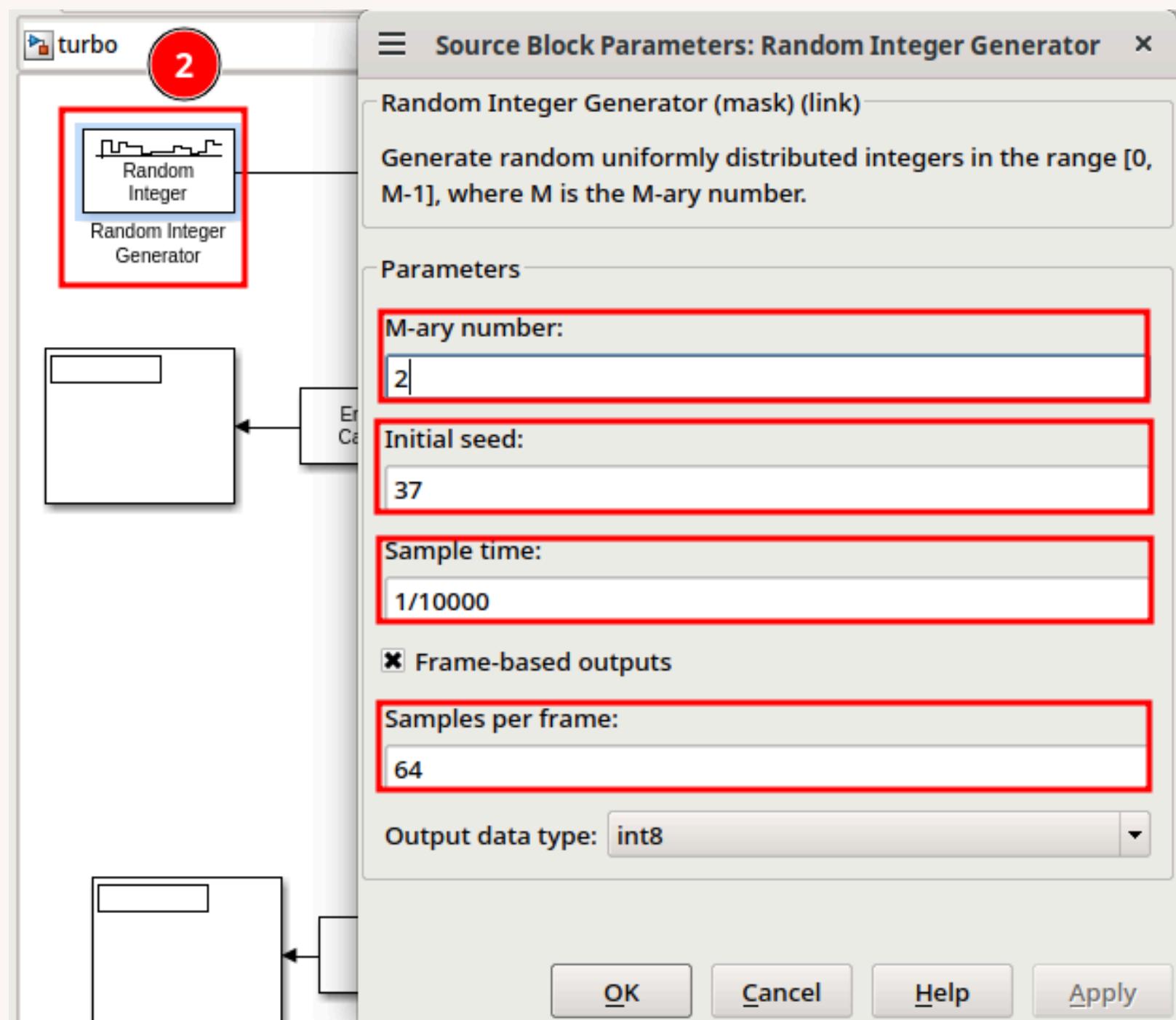
## Desenvolvimento

## turbo - Simulink classroom use



Abrirá essa aba  
do simulink

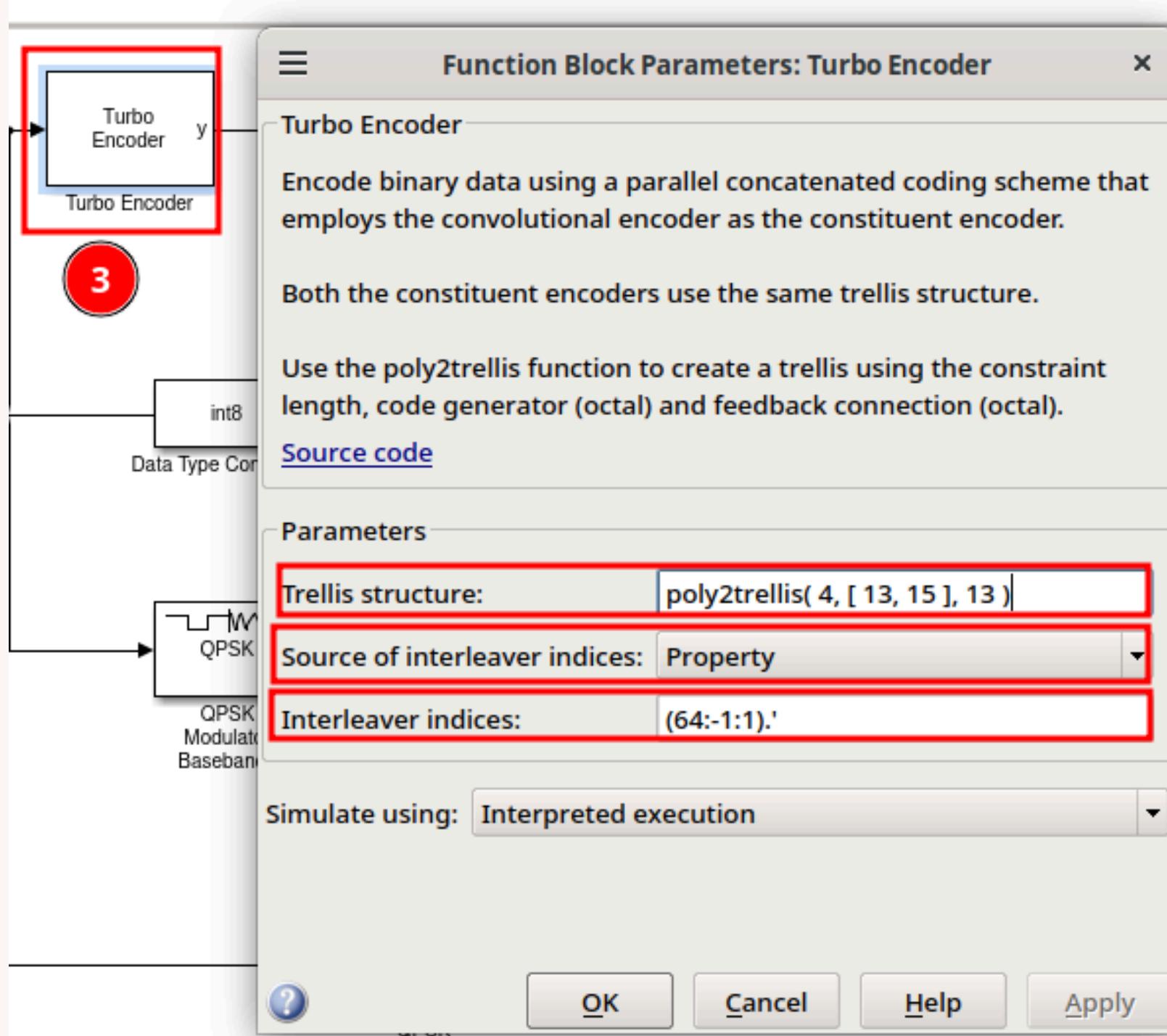
## Desenvolvimento



2 - **Random Integer Generator** é responsável por gerar a sequência de bits que servirá de entrada para o codificador Turbo.

- **M-ary number = 2:** define que os valores gerados serão binários, que serão somente **0** ou **1**.
- **Initial seed = 37:** A semente do gerador aleatório define o ponto inicial da sequência, garantindo que a simulação produza sempre a mesma sequência.
- **Sample time = 1/10000:** Representa o intervalo entre amostras; quanto menor o tempo, maior a taxa de geração de bits, ideal para sistemas Turbo.
- **Samples per frame = 64:** Cada saída gera um quadro de 64 bits por período de amostragem, fornecendo um pacote de 64 amostras, formato para o Turbo Encoder.

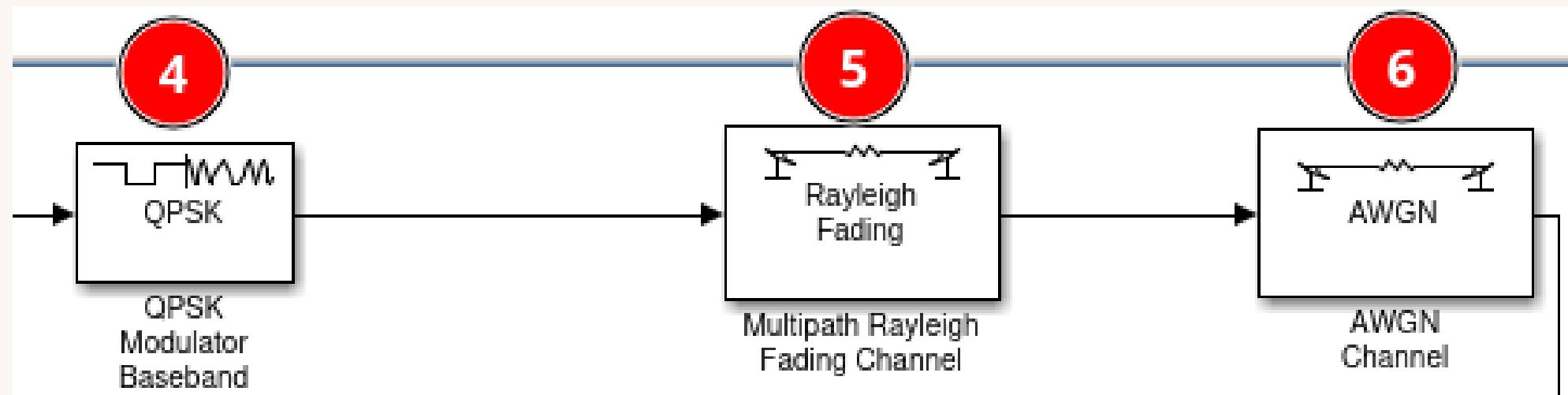
## Desenvolvimento



3 - **Turbo Encoder:** O bloco já implementa o esquema clássico de codificação turbo: dois codificadores convolucionais em paralelo + um interleaver interno.

- **Trellis structure = `poly2trellis(4, [13 15], 13)`:** Define o codificador RSC dos dois ramos do Turbo Encoder - “4” é o constraint length, [13 15] são os polinômios geradores, e o 13 é o polinômio de realimentação. Esse conjunto forma o codificador padrão de taxa 1/2 usado em Turbo Codes (3GPP).
- **Source of interleaver indices = Property:** Indica que o padrão de interleaving será definido manualmente
- **Interleaver indices = `(64:-1:1)'`:** Define a permutação dos bits antes do 2º codificador. `(64:-1:1)'` cria a ordem reversa de 64 posições em formato de coluna. Assim, o segundo codificador recebe os bits invertidos, aumentando a aleatoriedade. O tamanho do interleaver deve coincidir com o frame (64)

## Desenvolvimento

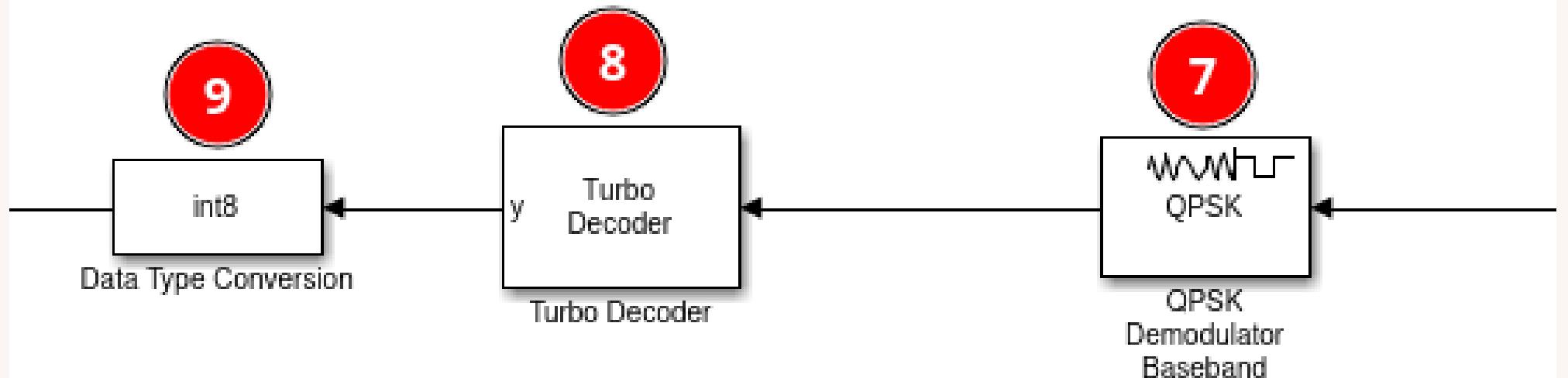


4 - **QPSK Modulator Baseband:** O modulador QPSK converte os bits do codificador turbo em símbolos capazes de serem transmitidos pelo canal. Ele mapeia pares de bits nos pontos da constelação QPSK, gerando um sinal complexo e permitindo transmitir 2 bits por símbolo, tornando o sinal adequado para propagação física.

5 - **Multipath Rayleigh Fading Channel:** Simula a propagação real com multipercorso e variações rápidas → representa o canal sem linha de visada direta.

6 - **AWGN Channel:** Adiciona ruído térmico gaussiano ao sinal → permite analisar o desempenho em diferentes SNR.

## Desenvolvimento



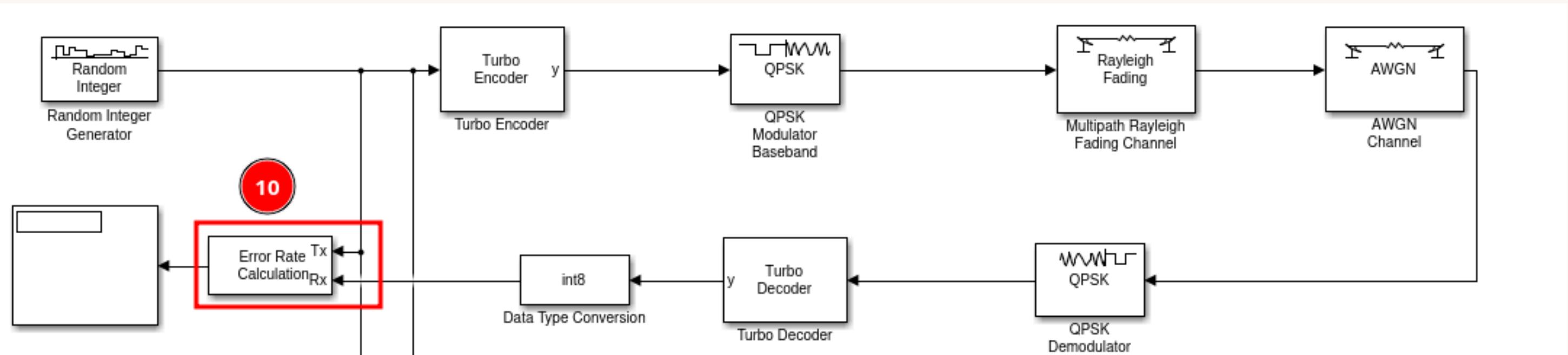
7 - **QPSK Demodulator Baseband:**  
Recupera os bits a partir do sinal ruidoso  
recebido.

Converte símbolos complexos novamente  
em informações digitais.

8 - **Turbo Decoder:** Recebe os bits da  
demodulação QPSK e usa algoritmos  
iterativos (MAP/Log-MAP) para corrigir  
erros de ruído e desvanecimento,  
combinando as informações dos dois  
codificadores.

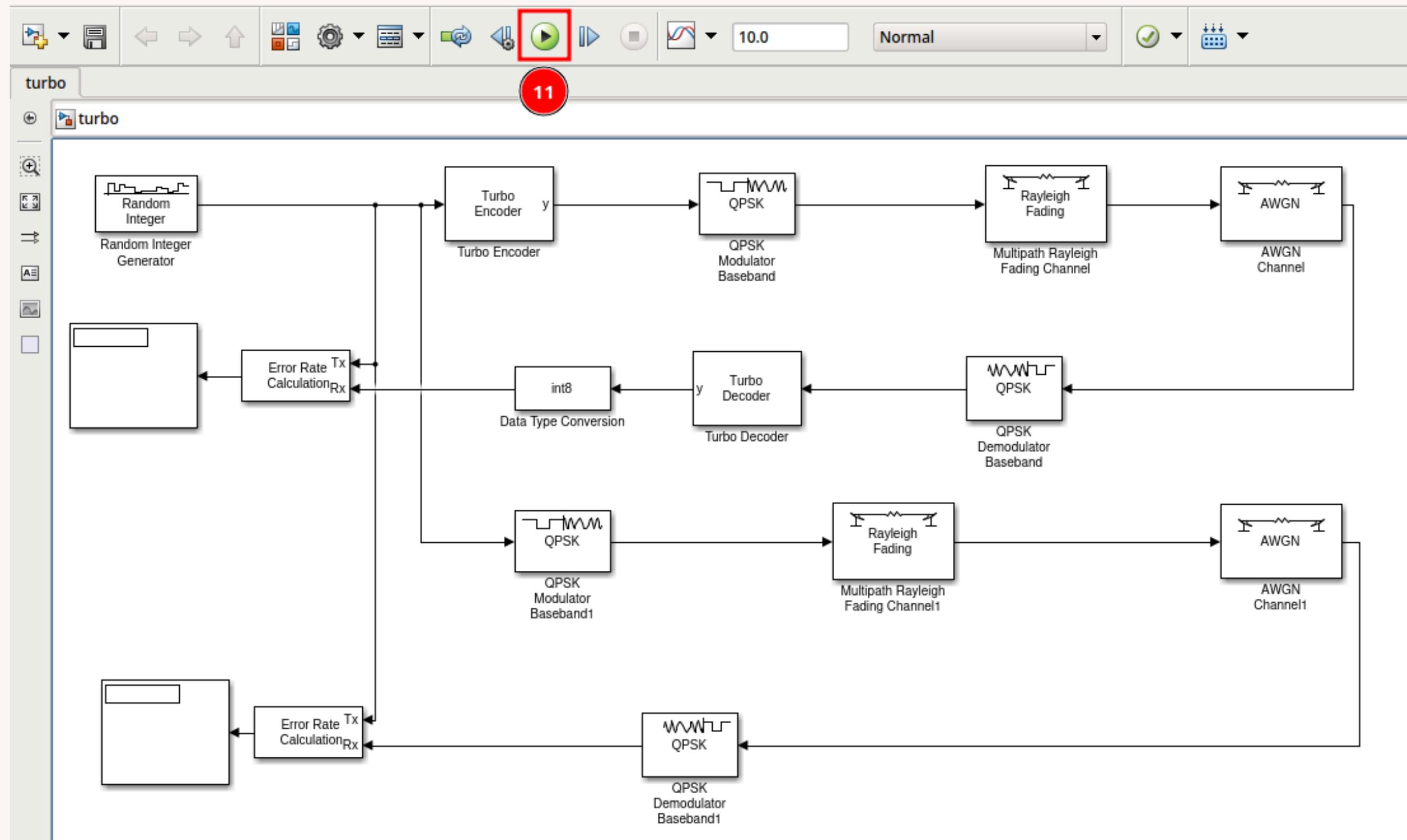
9 - **Data Type Conversion:** Ajusta o tipo de  
dado para int8, igual ao dos bits  
transmitidos. Necessário para comparar o  
resultado final com a sequência transmitida.

## Desenvolvimento



10 - **Error Rate Calculation:** O bloco compara os bits transmitidos (Tx) com os bits decodificados (Rx) e calcula automaticamente o BER, o número de erros e o total de bits analisados. Ele mede o desempenho do sistema sob ruído e fading, permitindo avaliar o efeito da modulação QPSK, do canal e do Turbo Decoder. É essencial para validar a transmissão e gerar análises como BER × SNR.

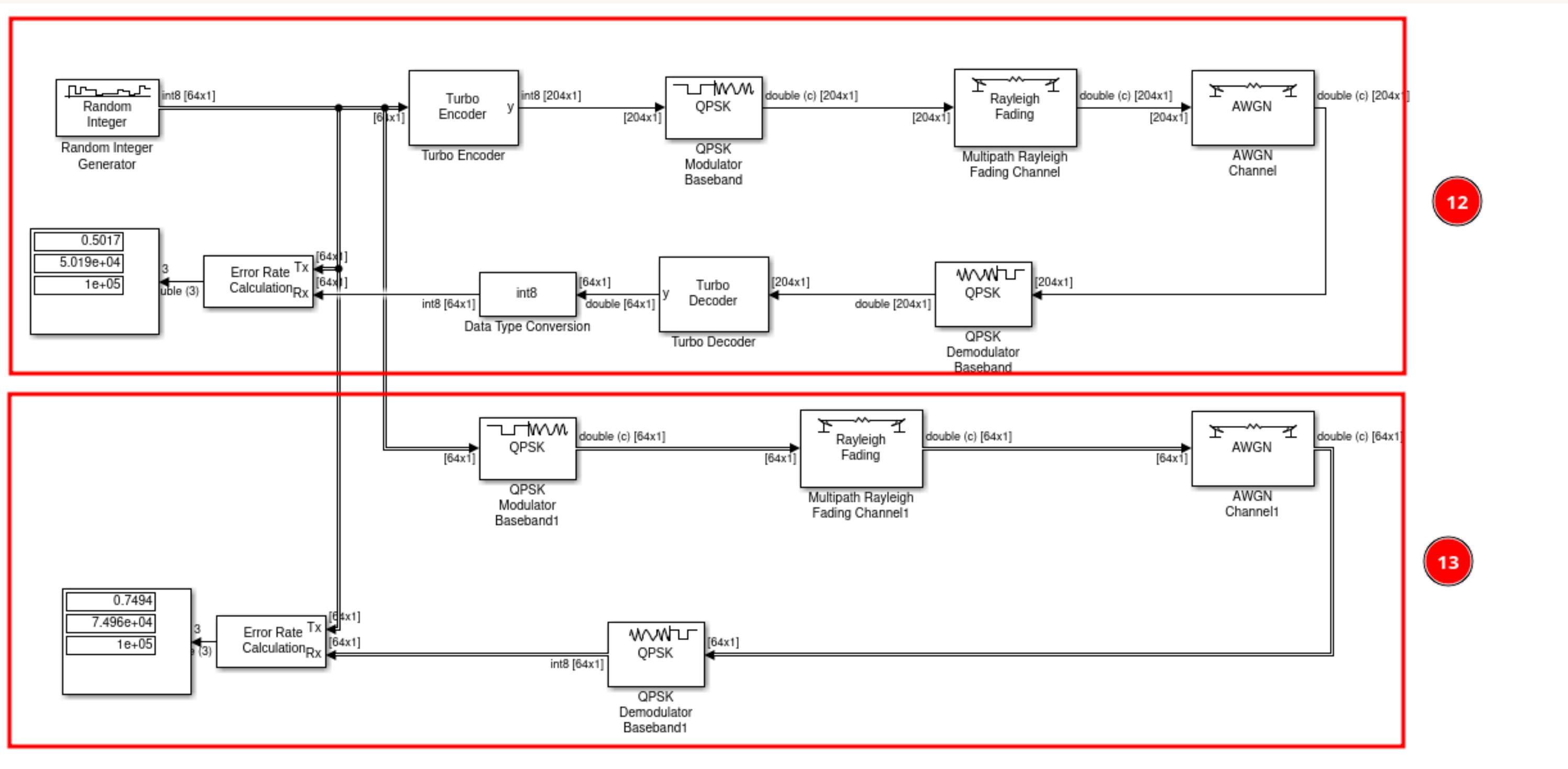
## Desenvolvimento



11 - Clique em **run**

## Desenvolvimento

12 - Com Codificação Turbo  
13 - Sem Codificação Turbo





## Desenvolvimento

### 12 - Com Codificação Turbo:

- BER  $\approx 0.5017$
- Erros  $\approx 50.199$  (aprox)
- Bits comparados  $\approx 100.000$

### 13 - Sem Codificação Turbo:

- BER  $\approx 0.7494$
- Erros  $\approx 74.964$  (aprox)
- Bits comparados  $\approx 100.000$

## Desenvolvimento

## Interpretação dos Resultados

A presença do Turbo Decoder melhora a comunicação mesmo em um canal severamente degradado:

- Reduz o BER de 0.75 → 0.50
- Diminui os erros absolutos de ~75.000 → ~50.000
- Melhora global de aproximadamente 33%

Isso confirma que:

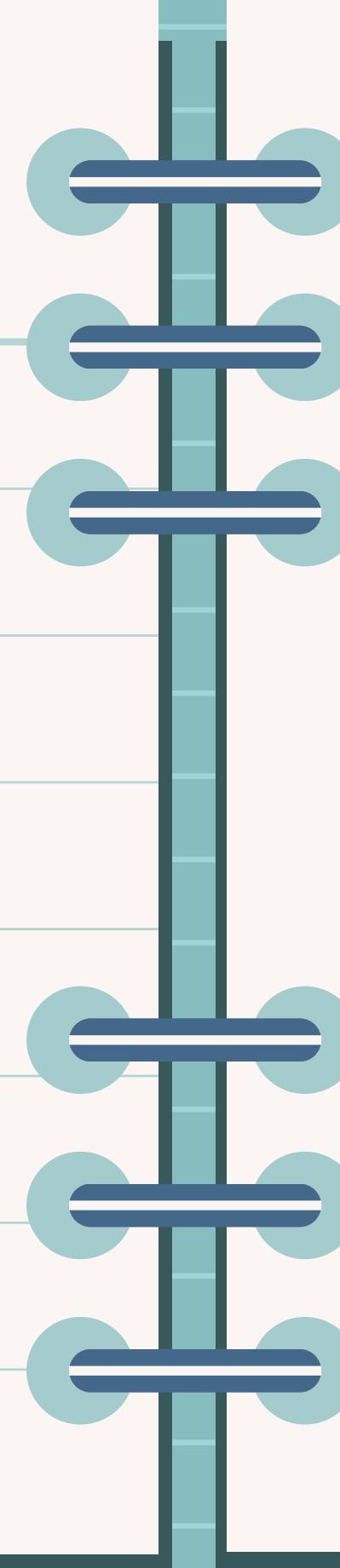
O sistema com turbo coding consegue corrigir parte dos erros introduzidos pelo canal Rayleigh + AWGN, enquanto o sistema sem turbo fica muito mais vulnerável ao ruído e ao desvanecimento.



## Atividades

# Impacto do tamanho do interleaver

**Objetivo:** Ver como diferentes tamanhos de interleaver afetam o BER



### Tarefa:

- Aumente o interleaver ( $100 \rightarrow 1000 \rightarrow 5000$  bits).

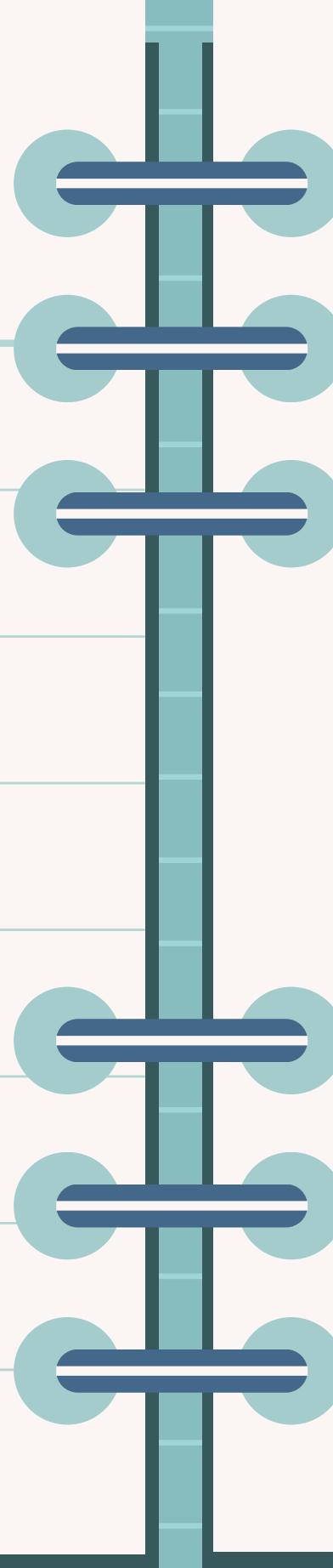
### Pergunta:

Interleavers maiores melhoram a decodificação turbo? Por quê?

## Atividades

# Treliça

**Objetivo:** Comparar o desempenho de diferentes configurações de poly2trellis em um encoder turbo.



### Tarefa:

- Troque a treliça para `poly2trellis(3, [7 0])`;
- Troque a treliça para `poly2trellis(3, [7 7])`.

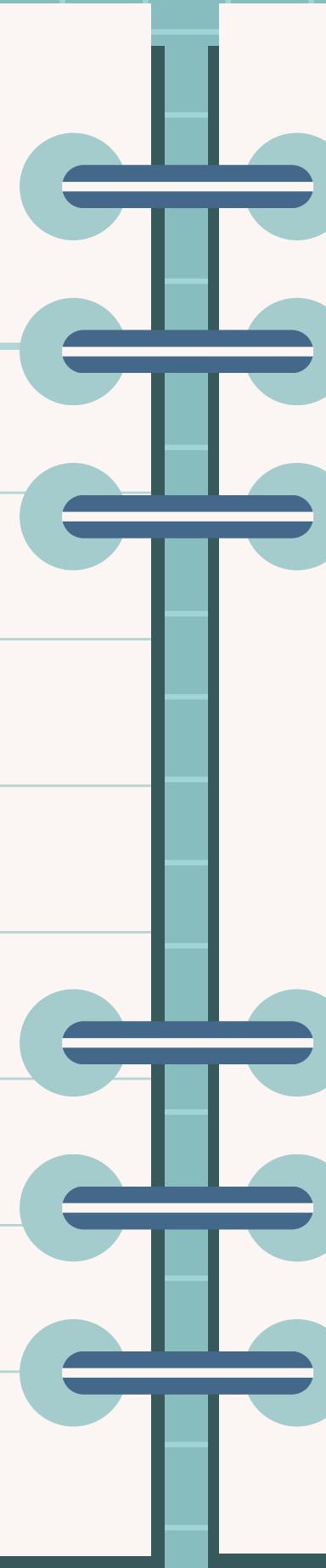
### Pergunta:

O BER melhora ou piora em relação à treliça original?

## Atividades

# Efeito do Doppler

**Objetivo:** Comparar o desempenho de diferentes configurações de poly2trellis em um encoder turbo.

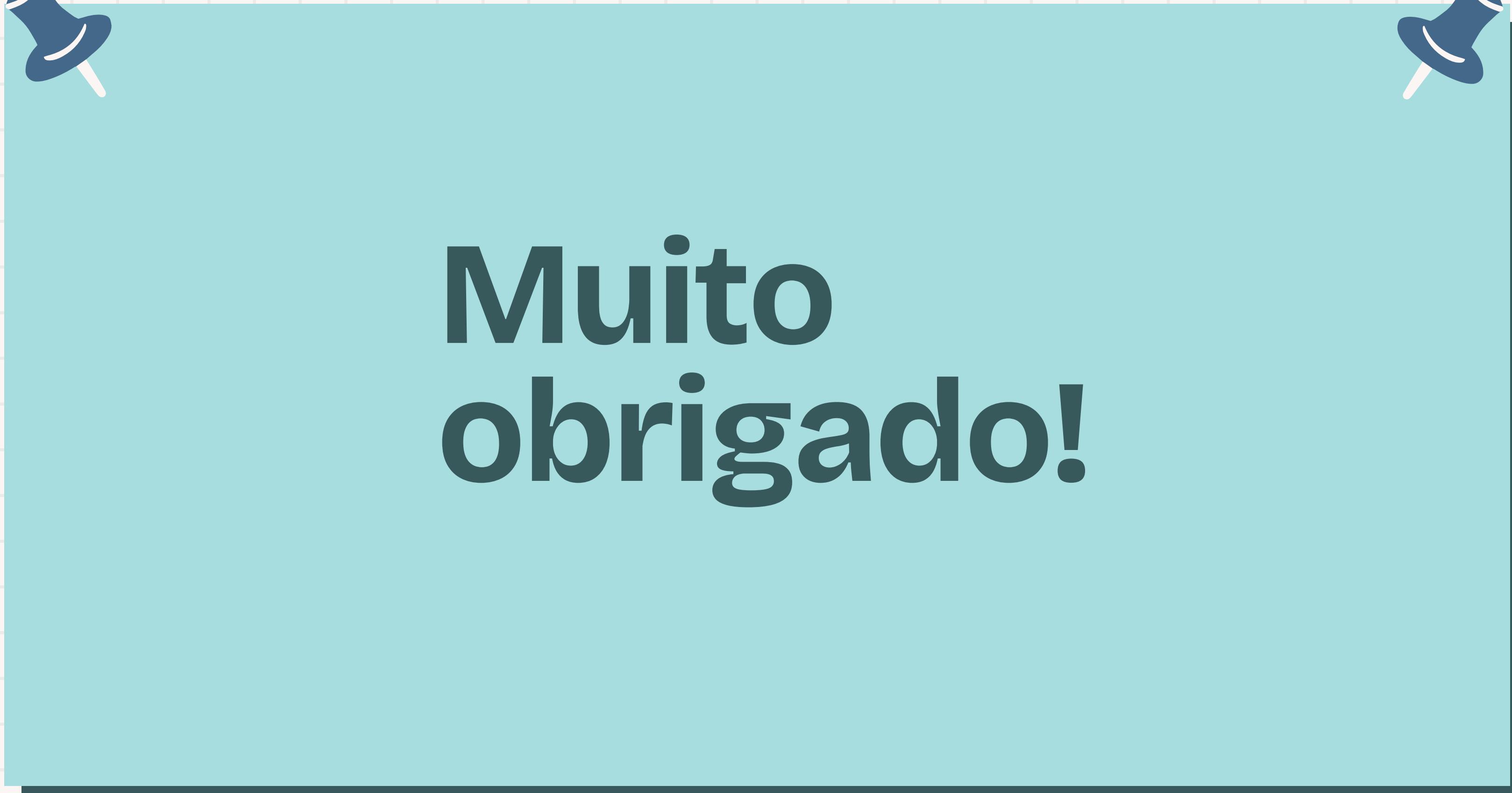


### Tarefa:

- Itere a frequência Doppler do canal Rayleigh para diferentes valores: 0 Hz, 5 Hz, 30 Hz e 100 Hz

### Pergunta:

Quando o Doppler aumenta, o canal varia mais rápido. Isso melhora ou piora o BER? Por quê?



**Muito  
obrigado!**