

Emerging Network Technologies

Exercise 2

Faculty of Architecture, Computing and Engineering (FACE)

School of Applied Computing

Mount Pleasant Campus

Swansea

SA1 6ED

Module Leader: Dr Carlene Campbell

Simulink: Building a Channel Noise Model

In the previous tutorial you built a simple model which involved generating a sine wave and observing how noise impacted the signal. This week will expand upon this by building a simple model of a communications system with a means of detecting and measuring the errors caused by noise, and then looking at how we can reduce the error rate.

Getting Started

Start Matlab and open a new Simulink model and the Simulink library as usual. Remember to issue the **commstartup** command to set the appropriate simulation parameters.

Blocks

As with any communications system, there are three fundamental parts: a transmitter, a communication medium, and a receiver. Table 1 equates these three components with the Simulink blocks we will be using.

Transmitter	Bernoulli Binary Generator (Comm. Sources library)	This block generates a random binary sequence (0s and 1s) for transmission
Communication Medium	Binary Symmetric Channel (Comm. Channels library)	This block is a simulation of a channel affected by noise, and introduces random errors in the transmitted signal by changing a 0 to a 1 or vice versa.
Receiver	Error Rate Calculation (Comm. Sinks library)	This block calculates the error rate of the channel by comparing the transmitted signal (Tx) and the received signal (Rx).

Additionally, we also need to use a **Display** block (Simulink > Sinks library) so that we can see the output of the Error Rate Calculation Block.

To build the model, drag the above blocks into the Simulink model window.

Setting block parameters

Binary Symmetric Channel

Double click on the channel block and configure the following parameters as shown in Figure 1.

Error probability: 0.01

Uncheck the “output error vector” box

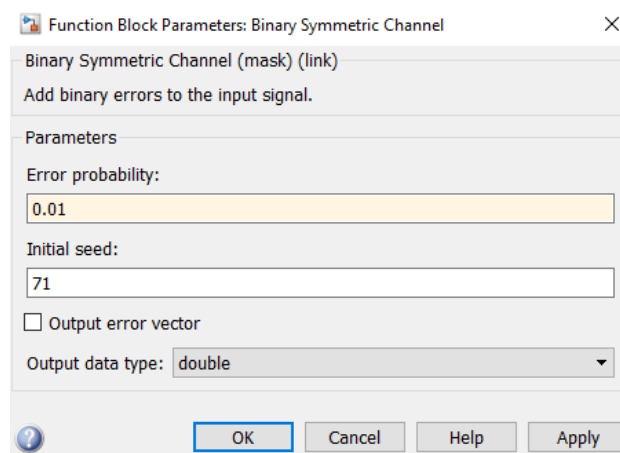


Figure 1 - Channel Block Parameters

Once you’ve configured these parameters the channel block should only have one output port. Make sure this is the case.

Error Rate Calculation

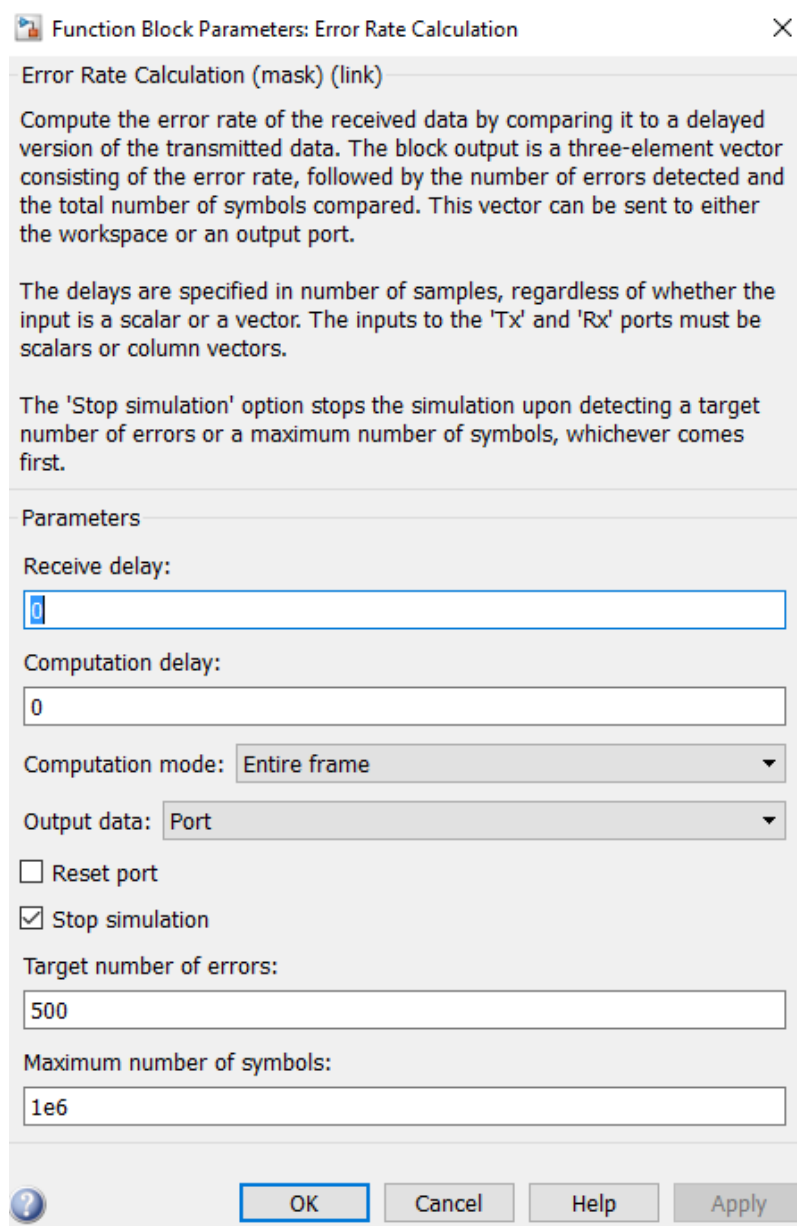


Figure 2 - Error Rate Block Parameters

Using the “Output data” drop-down menu, ensure “Port” is selected.

Make sure the “Stop simulation” checkbox is checked and that the “Target number of errors” is set to 500.

Connecting the blocks

Based on the function of each block that is explained in Table 1, connect the blocks together. Notice that the Error Rate Calculation block has two inputs: the transmitted signal (Tx) and the received signal (Rx); you need to make sure both of these are connected.

Hint: for the Tx input you may need to refer back to tutorial 1, specifically the bit about one line tapping off another.

Hint 2: if your error rate calculation block does not have an output port, check the block parameters and make sure they reflect Figure 2.

Running the model

Once you've connected the blocks together, go ahead and run the simulation. Because of the block parameters we configured previously for the Error Rate Calculator, the simulation will run continually until 500 errors have occurred (see Figure 2). Once the simulation is complete, the **Display** block should look something like this (you may need to enlarge the size of the block to see all three displays):

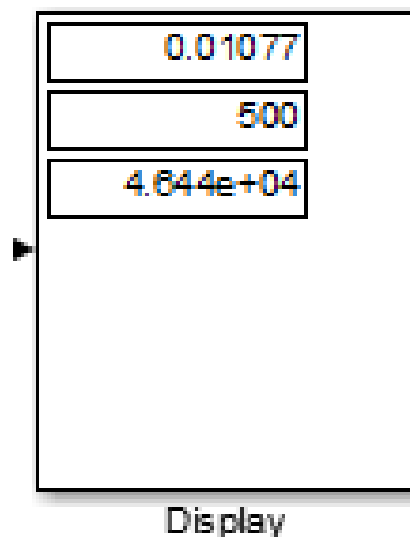


Figure 3 - Error Rate Calculation Display

The display block is showing the following:

- Bit Error Rate (BER)
- Number of errors
- Total number of transmitted bits. By default, these are shown in scientific notation, however, this can be changed in the block parameters.

Error Correction – Adding Blocks

The main purpose of studying the level of errors in a transmission is to determine how the error rate can be lessened. This is done using a technique called error-correction coding, or channel coding. There are many different correcting-codes that can be used; for this exercise we will focus on one of the simpler ones: Hamming code.

Further details can be found at - [https://en.wikipedia.org/wiki/Hamming\(7,4\)](https://en.wikipedia.org/wiki/Hamming(7,4)). It is suggested you read this first in order to establish a basic understanding before proceeding.

For this task, two additional blocks are required, both of which are found in the Communication Systems Toolbox under **Error Detection and Correction / Block:**

- Hamming Encoder
- Hamming Decoder

The Hamming Encoder encodes the data prior to it being sent over the channel, whilst the Hamming Decoder decodes the same after it has passed through the channel. With this in mind, find these two blocks in the Simulink library determine where they need to be placed.

Configuring Error Correction

The default hamming code is [7,4], meaning that an input of four bits will be converted to seven bits. This adds three additional “check” bits to every four bits of a message.

Because the input requirement of the Hamming Encoder Block is a message word of 4, we need to ensure that this is configured in the Bernoulli Binary Generator block. As shown in Figure 4 below, ensure that **Samples per frame** is set to **4**.

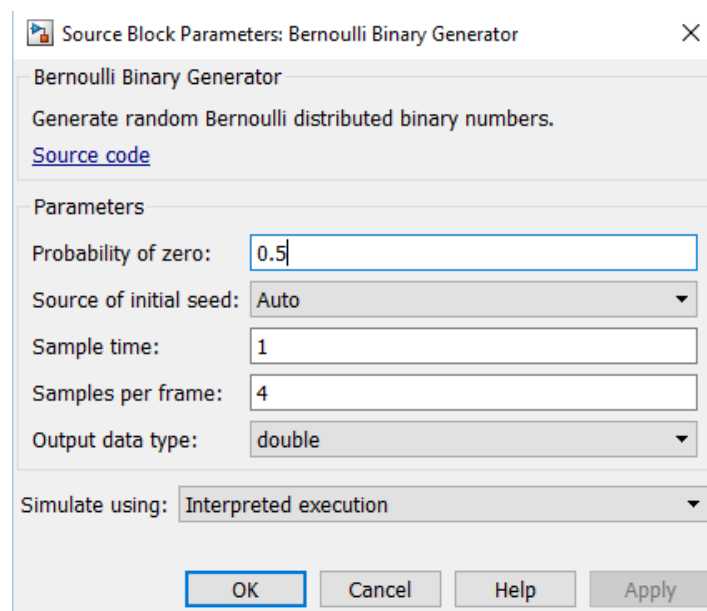


Figure 4 - Samples per frame

You can visualize this more easily by clicking **Display > Signals & Ports** and checking **Signal Dimensions** in the Simulink toolbar. Notice how the output of the Binary Generator block and the input of the Hamming block is labelled **4x1**. However, because the Hamming

encoder adds three bits, the output for this block is **7x1**. Once it has been decoded on the other side, the three additional bits are removed.

Now re-run the simulation and observe the difference in the error calculations as a result of the correction techniques that you have employed.