****

**Matlab, simulink**

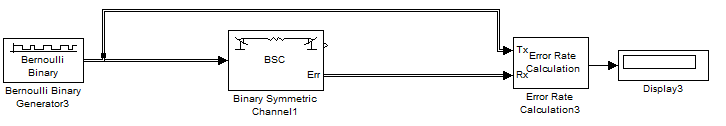
**Building a Channel Noise Model**

**&**

**Reducing the Error Rate Using a Hamming Code**

1. **Building a Channel Noise Model**

This section shows how to build a simple model of a communication system. The model, shown in the following figure, contains the most basic elements of a communication system: a source for the signal, a channel with noise, and means of detecting errors caused by noise.



**Overview of the Model**

The channel noise model generates a random binary signal, and then switches the symbols 0 and 1 in the signal, according to a specified error probability, to simulate a channel with noise. The model then calculates the error rate and displays the result. The model contains the following components.

**Source (Bernoulli Binary Generator)**

The source for the signal in this model is the Bernoulli Binary Generator block, which generates a random binary sequence.

**The Channel (BSC)**

The Binary Symmetric Channel block simulates a channel with noise. The block introduces random errors to the signal by changing a 0 to a 1 or the reverse, with a probability specified by the **Error probability** parameter in the block’s dialog.

**Error Rate Calculation**

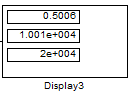
The Error Rate Calculation block calculates the error rate of the channel. The block has two input ports, labeled Tx, for the transmitted signal, and Rx, for the received signal. The block compares the two signals and checks for errors. The output of the block is a vector with three entries:

**•** Bit error rate, which you expect to be approximately .01, since this is the probability of error in the channel

**•** Number of errors

**•** Total number of bits that are transmitted

**Display:** The Display block displays the output of the Error Rate Calculation block. This block, labeled "Error Rate Display," displays the number of errors introduced by the channel noise. When you run the simulation, three small boxes appear in the block, as shown in the following figure, displaying the vector output from the Error Rate Calculation block.

****

**Error Rate Display**

The block displays the output as follows:

**•** The first entry is the symbol error rate (SER).

**•** The second entry is the total number of errors.

**•** The third entry is the total number of comparisons made. The notation 1e+004 is shorthand for 104.

**II.I Selecting Blocks for the Channel Noise Model**

To build the model, first move its blocks into a new model window, as follows:

**1** Type commstartup at the MATLAB prompt to set simulation parameters for the model.

**2** Type simulink at the MATLAB prompt to open the Simulink Library Browser.

**3** From the **File** menu, select **New**, and then **Model**. This opens a new model window.

**4** Drag the following blocks from the Simulink Library Browser into the model window:

**•** Bernoulli Binary Generator block, from the Data Sources sublibrary of the Comm Sources library

**•** Binary Symmetric Channel block, from the Channels library

**•** Error Rate Calculation block, from the Comm Sinks library

**•** Display block, from the Simulink Sinks library

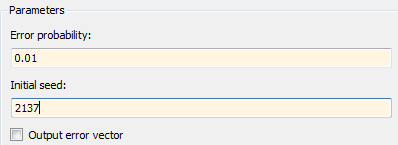
**II.2 Setting Parameters in the Channel Noise Model**

To set block parameters in the channel noise model, do the following:

**1** Double-click the Binary Symmetric Channel block and make the following changes to the default parameters in the block’s dialog:

**•** Set **Error probability** to 0.01.

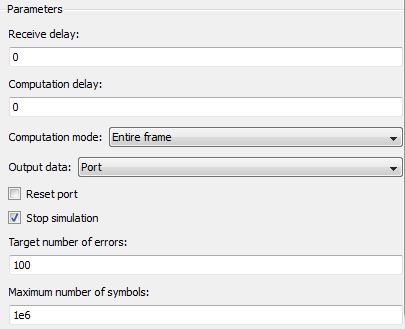
**•** Clear the **Output error vector** box. This removes the block’s lower output port, which is not needed for this model.



**2** Double-click the Error Rate Calculation block and make the following changes to the default parameters in the block’s dialog:

**•** Set **Output data** to Port to create an output port for the block.

**•** Select **Stop simulation**.



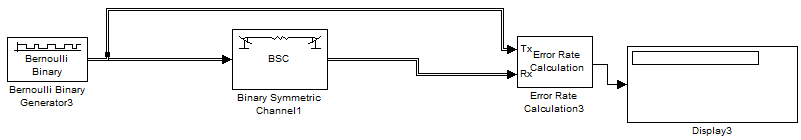
Selecting **Stop simulation** causes the simulation to stop after the target number of errors occurs or the maximum number of symbols is reached.

**Initial Seeds**

The Bernoulli Binary Generator block and the Binary Symmetric Channel block both uses a random number generator to generate random sequences of bits. In both blocks, the **Initial seed** parameter initializes the random sequence. The initial seeds in the two blocks should have different values to ensure that the source signal and the channel noise are statistically independent. In general, initial seeds should have different values in all blocks that have an **Initial seed** parameter.

**II.3 Connecting the Blocks**

Next, connect the blocks as shown in the following figure. Make sure to connect the arrow from the Binary Symmetric Channel block to the input port labeled Rx on the Error Rate Calculation block.



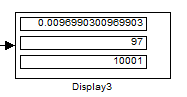
**II.4 Running the Channel Noise Model**

To run the model, select **Start** from the **Simulation** menu. After a few seconds, the model will stop automatically. To see all three boxes in the Display block, you must enlarge the block slightly, as follows:

**1** Select the Display block and move the mouse pointer to one of the lower corners of the block, so that a diagonal arrow appears on the corner, as shown.



**2** Drag the corner of the block down with the mouse until three windows appear, as shown.



The Display block displays the following information:

**•** The bit error rate

**•** The number of errors

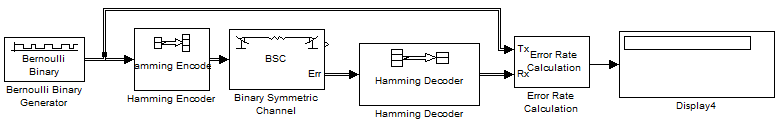
**•** The total number of bits that are transmitted

Note that the exact values that appear will vary, depending on the **Initial seed** parameters in the Bernoulli Binary Generator block and the Binary Symmetric Channel block.

Since the **Target number of errors** in the dialog for the Error Rate Calculation block is set to 100, the simulation stops when 100 errors have been detected. To save the model, select **Save** from the **File** menu, type a name for the model, such as channel noise, in the **File name** field, and click **Save**.

1. **Reducing the Error Rate Using a Hamming Code**

We will now describe how to reduce the error rate in the model shown in the figure Channel Noise Model by adding an error-correcting code. The following figure shows an example that uses a Hamming code.



**III.1 Building the Hamming Code Model**

You can build the Hamming code model by adding blocks to the model shown in the figure Channel Noise Model

**1** Type channeldoc at the MATLAB prompt to open the channel noise model. Then save the model as my\_hamming in the directory where you keep your work files.

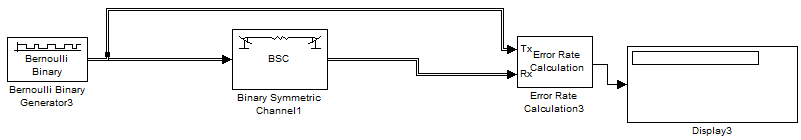
**2** Drag the following two Communications Blockset blocks from the Simulink Library Browser into the model window:

**•** Hamming Encoder block, from the Block sublibrary of the Error Detection and Correction library

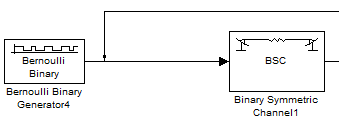
**•** Hamming Decoder block, from the Block sublibrary of the Error Detection and Correction library

**3** Click the right border of the model and drag it to the right to widen the model window.

**4** Move the Binary Symmetric Channel block, the Error Rate Calculation block, and the Display block to the right by clicking and dragging. This creates more space between the Binary Symmetric Channel block and the blocks next to it. The model should now look like the following figure.



**5** Click the Hamming Encoder block and drag it on top of the line between the Bernoulli Binary Generator block and the Binary Symmetric Channel block, to the right of the branch point, as shown in the following figure. Then release the mouse button. The Hamming Encoder block should automatically connect to the line from the Bernoulli Binary Generator block to the Binary Symmetric Channel block.



**6** Click the Hamming Decoder block and drag it on top of the line between the Binary Symmetric Channel block and the Error Rate Calculation block.

**Using the Hamming Encoder and Decoder Blocks**

The Hamming Encoder block encodes the data before it is sent through the channel. The default code is the [7,4] Hamming code, which encodes message words of length 4 into codewords of length 7. As a result, the block converts frames of size 4 into frames of size 7. The code can correct one error in each transmitted codeword.

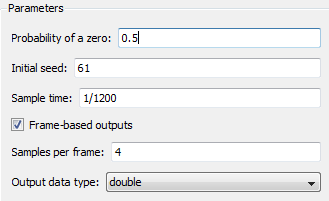
For an [n,k] code, the input to the Hamming Encoder block must consist of vectors of size k. In this example, k=4. The Hamming Decoder block decodes the data after it is sent through the channel. If at most one error is created in a codeword by the channel, the block decodes the word correctly. However, if more than one error occurs, the Hamming Decoder block might decode incorrectly.

**Setting Parameters in the Hamming Code Model**

Double-click the Bernoulli Binary Generator block and make the following changes to the parameter settings in the block’s dialog, as shown in the following figure:

**1** Select the box next to **Frame-based outputs** in the dialog for the Bernoulli Binary Generator block.

**2** Set **Samples per frame** to 4. This converts the output of the block into frames of size 4, in order to meet the input requirement of the Hamming Encoder Block.

****

**Note** Many Communications Blockset blocks, such as the Hamming Encoder block, require their input to be a vector of a specific size. If you connect a source block, such as the Bernoulli Binary Generator block, to one of these blocks, you should select the box next to **Frame-based outputs** in the dialog for the source, and set **Samples per frame** to the required value.

**III.2 Running the Hamming Code Model**

To run the model, select **Start** from the **Simulation** menu. The model terminates after 100 errors occur. The error rate, displayed in the top window of the Display block, is approximately .001. Note that you get slightly different results if you change the **Initial seed** parameters in the model or run a simulation for a different length of time (10 and 20 exemple). You expect an error rate of approximately .001 for the following reason.

**Questions : Building 2 models**

* **Building a Channel Noise Model (model 1)**
* **Reducing the Error Rate Using a Hamming Code (model 2)**
* **Comment the difference between model 1 and 2 in document.txt**
* **Make all this (.mdl et .txt) in a file with your name**

**Matlab, simulink**

**Building a Channel Noise Model**

**&**

**Reducing the Error Rate Using a Cyclic Code**

1. **Modeling a Channel with Modulation**

The Binary Symmetric Channel block, which simulates a channel with noise, is useful for building models of channel coding. For other types of applications, you might want to construct a more realistic model of a channel.

For example, you can add modulation and demodulation, and replace the Binary Symmetric Channel block with an AWGN Channel block, which adds white Gaussian noise to the channel. This following figure shows an example that uses binary phase shift keying (BPSK).

**II.1 Building the BPSK Model**

You can build the BPSK model from the one shown in the figure Channel Noise Model. To build the model, follow these steps:

**1** Enter channeldoc at the MATLAB prompt to open the channel noise model, and then save the model as my\_bpsk in the directory where you keep your work files.

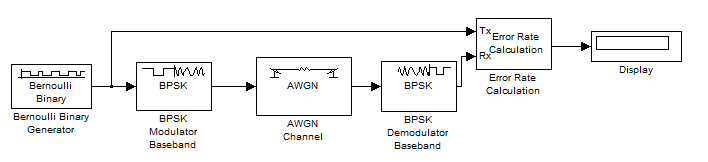
**2** Move the following blocks from the Simulink Library Browser into the model window, and insert them into the model as shown in the following figure:

**•** BPSK Modulator Baseband block, from PM in the Digital Baseband Modulation sublibrary of the Modulation library

**•** AWGN Channel block, from the Channels library

**•** BPSK Demodulator Baseband block, from PM in the Digital Baseband Modulation sublibrary of the Modulation library

The model should now appear as in the figure:



**Binary Phase Shift Keying**

The BPSK Modulator and Demodulator Baseband blocks implement binary phase shift keying (BPSK) modulation. BPSK is a method for modulating a binary signal onto a complex waveform by shifting the phase of the complex signal. In digital baseband BPSK, the symbols 0 and 1 are modulated to the complex numbers exp(jt) and -exp(jt), respectively, where t is a fixed angle. In this example, t = 0, so these numbers are just 1 and -1.

You can set the value of t in the **Phase offset** parameter in the dialogs for the BPSK Modulator Baseband block and the BPSK Demodulator Baseband block. The default value is 0.

**Setting Parameters in the BPSK Model**

To set block parameters in the BPSK model, do the following:

**1** Double-click the AWGN Channel block and set **Es/No** to 4.2.

**2** Double-click the Error Rate Calculation block and make the following changes to the default parameters in the block’s dialog:

**•** Set **Output data** to Port.

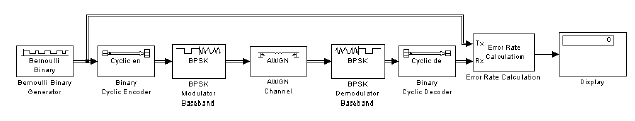
**•** Check the box next to **Stop simulation**.

**Running the BPSK Model**

When you run the model, the Display block shows an error rate of approximately 0.01, the same as in the channel noise model. The BPSK model uses the BPSK Modulator Baseband, the AWGN Channel, and the BPSK Demodulator Baseband blocks to simulate a channel with noise. This provides a more realistic model of a channel than using just the Binary Symmetric Channel block. You can also model other types of channel noise using blocks from the Communications Blockset Channels library.

1. **Reducing the Error Rate with a Cyclic Code**

You can improve the error rate in the model shown in the figure BPSK Modulation Model, for certain noise levels, by adding channel coding. An example that uses a binary cyclic code is shown below.



**Building the Cyclic Code Model**

You can build the cyclic code model by adding blocks to the BPSK model shown in the figure BPSK Modulation Model. To build the model, follow these steps:

**1** Open the BPSK model by entering bpskdoc at the MATLAB prompt. Save the model as my\_cyclic in the directory where you keep your work files.

**2** Double-click the AWGN Channel block and set **Es/No** to 7. This value is chosen for this example to illustrate the benefit of using a code. If Es/N0 were much lower than the value chosen here, then the Es/N0 reduction caused by the additional coded bits would be greater than the coding gain provided by the code.

**3** Run the simulation and note its error rate (for later comparison in “Running the Cyclic Code Model”. The error rate appears in the top entry of the Display block.

**4** To introduce cyclic coding, drag these Communications Blockset blocks from the Simulink Library Browser into the model window:

**•** Binary Cyclic Encoder block, from the Block sublibrary of the Error Detection and Correction library

**•** Binary Cyclic Decoder block, from the Block sublibrary of the Error Detection and Correction library

**5** Widen the model window and connect the blocks as below.

**6** Double-click the Bernoulli Binary Generator block and change these parameters:

**•** Select **Frame-based outputs**.

**•** Set **Samples per frame** to 21 to match the input requirement of the Binary Cyclic Encoder block.

**7** Double-click the Binary Cyclic Encoder block and change these parameters:

**•** Set **Codeword length N** to 31.

**•** Set **Message length K** to 21.

Make the same changes in the Binary Cyclic Decoder block.

**8** Double-click the AWGN Channel block and change these parameters:

**•** Set **Es/No** to 7+10\*log10(21/31). The second term in this sum accounts for the difference in symbol period compared to the original BPSK model.

**•** Set **Symbol period** to 21/31. For more information on setting **Symbol period**, see “Verifying the Symbol Period” below.

With these parameter values, the AWGN channel block produces the same amount of noise per symbol as in the BPSK model (without coding) in which **Es/No** is 7. The equivalence of the noise enables you to determine how much the cyclic code improves the bit error rate.

**9** Double-click the Error Rate Calculation block and change this parameter:

**•** Set **Maximum number of symbols** to 1e7. This extends the simulation, producing a more reliable estimate for the error rate.

**Binary Cyclic Encoder and Decoder**

The Binary Cyclic Encoder block implements a binary cyclic code. In this example, the block has the following parameter settings:

**• Codeword length N** = 31

**• Message length K** = 21

The code rate is given by :



This example uses a rate 21/31 code. The codeword length N must have the form 2M - 1, where M is an integer greater than or equal to 3. The input to the block must be a vector whose length is **Message length**.

The Binary Cyclic Decoder block decodes the demodulated signal. This block must have the same parameter settings as the Binary Cyclic Encoder block.

**Running the Cyclic Code Model**

When you run the simulation, the bit error rate is less than one-tenth of the error rate in the model that does not have channel coding.

**Verifying the Symbol Period**

When you compare the cyclic code model to the BPSK model, which does not have channel coding, the ratio of energy per information symbol to noise spectral density, Eb/N0, should be the same in both models. You can use the **Symbol period** and **Es/No** parameters in the AWGN Channel block to adjust the amount of channel noise so that Eb/N0 is the same as in the model without coding. Because the cyclic code has rate 21/31, you set **Symbol period** to 21/31. For a BPSK simulation with a rate K / N code, set **Symbol period** to K/N because there are K information symbols for each N channel symbols.

Note that **Es/No**, the ratio of energy per channel symbol to noise spectral density, is not the same as Eb/N0. To convert between the two, use the formula



: where K / N is the ratio of information symbols to channel symbols. Changing the **Symbol period** to K/N has the same effect as subtracting 10log(K / N) from the **Es/No** parameter.

**Simulation Parameters:**

Bernoulli Generator:

Probability of a zero = 0.5

Sample time = 1/1200

AWGN Channel

Mode = Signal to Noise Ratio (Eb/No) = 10

Symbol Period = 1/1200

Set the Simulation Stop Time parameter to 15 and Run the Model.

**Questions : Building 2 models**

* **Building the BPSK Model (model 3)**
* **Reducing the Error Rate Using a cyclic Code (model 4)**
* **Comment the difference between model 3 and 4 in the precedent document.txt**
* **Make all this (.mdl et .txt) in a file with your name**