

Goal

The goal of this implementation is to generate realistic bit error patterns using the Gilbert-Elliot model.

The manipulated output stream is collected together with the bit error rate, the gap sizes between consecutive errors, and a compressed list of visited channel states.

The Gilbert-Elliot model is useful because it does not assume independent errors only.

Instead, it models a channel with two states: a good state and a bad state. This allows the simulation of burst-like error behavior, where errors can appear close to each other during bad channel phases.

Fixed Constants

$p_{gg} = 0.99$ is the probability of staying in the good state.

$p_{bb} = 0.1$ is the probability of staying in the bad state.

$p_{bg} = 1 - p_{bb} = 0.9$ is the probability of switching from the bad state back to the good state.

Following the Gilbert-Elliot model shown in the lecture, where e_g and e_b are mentioned I use them as, $e_g = 10^{-4}$ is the bit error probability in the good state, and $e_b = 10^{-1}$ is the bit error probability in the bad state.

$bits_array_len = 10^6$ is the length of the input bitstream is.

Variables

`bits` is a random integer array of length `bits_array_len`. It contains only zeros and ones and simulates the input bitstream.

`channel_state` stores the current channel state. The simulation starts with `channel_state = true`, which means that the channel starts in the good state.

`result_stream` is a preinitialized array of length `bits_array_len`. It stores the received bitstream after possible bit flips.

`bit_error_rate_counter` counts how many bits were changed during transmission.

`gap_begin` stores the index of the last detected error. It is initialized with 1. This is a design choice to avoid an additional condition inside the main loop.

`gap_tracker` is an empty array that is filled during execution. It contains a 3 x n matrix. The first row stores the beginning indices of gaps, the second row stores the ending indices of gaps, and the third row stores the gap size between two consecutive errors.

`state_sequence` is an empty string array. It stores a compressed version of the visited states.

`state_visit_counter` counts how often the current state was visited consecutively. When the state changes, the counter is stored in `state_sequence`, for example as "12x Good" or "4x Bad".

Implementation

I started by defining the fixed probabilities and generating the random input bitstream. For this demonstration, the channel always starts in the good state.

To apply the Gilbert-Elliot model to the bitstream, I iterate over the array. In each iteration, two random values are generated. The first random value is used for the channel state transition. The second random value is used to decide whether the current bit is flipped. Separating them is used to prevent unwanted coupling between channel state switching and bit flipping.

The main loop consists of two channel cases: the good state and the bad state.

If the channel is in the good state, it stays in the good state with probability p_{gg} .

Otherwise, it switches to the bad state for the next iteration. In the good state, the current bit is flipped with probability e_g .

If the channel is in the bad state, it switches back to the good state with probability p_{bg} . Otherwise, it remains in the bad state. In the bad state, the current bit is flipped with probability e_b .

The resulting bit of each iteration is written into the output array `result_stream`.

At the same time, the implementation compares the original input bit with the received bit. If both bits are different, an error occurred. In that case, `bit_error_rate_counter` is incremented.

The gap tracking is also updated when an error occurs. The current gap is stored with its start index, end index, and size. After that, `gap_begin` is set to the current error index, so the next gap can be measured from this position.

The `state_visit_counter` is incremented at the beginning of each loop iteration. Whenever the channel state changes, the current value of the counter is added to `state_sequence`. This creates a compact state history, for example: "12x Good", "4x Bad", "2x Good", "1x Bad".

This avoids storing every single state separately and gives a basic compressed overview of the visited states.

Postprocessing

After the main loop has finished, the first entry of `gap_tracker` is removed. This first entry exists because `gap_begin` was initialized with 1, before the first real error position was known. Removing this entry corrects the data while keeping the main loop simpler and avoiding an additional runtime condition.

The bit error rate is calculated as $BER = \text{bit_error_rate_counter} / \text{bits_array_len}$.

After that, the gap sizes are extracted from the third row of `gap_tracker`.

For the histogram, only gap sizes smaller than or equal to 500 are used. This filtering was done on behalf of Christos to keep the histogram readable and focused on the most relevant gap sizes.

Finally, a histogram of the gap sizes is created. MATLAB makes this part very convenient, because the histogram can be generated directly from the collected gap-size array.

At the end, the compressed state sequence and the calculated bit error rate are printed to the Command Window.

Results

The simulation produced a bit error rate of $BER = 0.0012$.

This result is plausible for the selected parameters. The channel is in the good state most of the time, and the error probability in the good state is very low. Therefore, most bits are transmitted correctly.

However, the bad state has a much higher error probability. When the channel enters the bad state, errors become more likely. Because of this, errors are not evenly distributed over the whole bitstream. Instead, they can occur in short clusters.

An excerpt from the compressed visited states looks like this:

```
"246x Good"    "1x Bad"      "240x Good"   "1x Bad"
"42x Good"     "1x Bad"      "103x Good"   "1x Bad"
"112x Good"    "1x Bad"      "130x Good"   "1x Bad"
"4x Good"      "1x Bad"      "49x Good"    "1x Bad"
"76x Good"     "1x Bad"      "17x Good"    "1x Bad"
"66x Good"     "1x Bad"
```

This excerpt shows that the channel usually stays in the good state for much longer periods, while the bad state is often only visited for a short time. This matches the selected probabilities: p_{gg} is very high, while p_{bg} is also high, so the channel tends to stay good and quickly returns from bad to good.

The histogram of the gap sizes shows the distribution of distances between consecutive errors. Small gap sizes mean that errors happened close to each other. These small gaps are mainly caused by bad-state phases, where the error probability is high. Larger gap sizes occur when the channel stays in the good state for longer periods. In these phases, errors are rare because e_g is very small. Overall, the result shows the intended behavior of the Gilbert-Elliot model: the channel is mostly reliable, but short bad phases can create burst-like error patterns.

Conclusion

The implementation successfully simulates a Gilbert-Elliot channel with two states. The good state produces only a small number of errors, while the bad state creates a much higher error probability.

The calculated bit error rate gives a general measurement of the transmission quality. The gap-size histogram gives additional information about how the errors are distributed over time.

The observed bit error rate of 0.0012 and the compressed state sequence are consistent with the chosen model parameters. The channel spends most of its time in the good state, while the bad state appears only shortly but can still cause clustered errors.

Therefore, the simulation fulfills its goal of generating more realistic bit error patterns than a simple memoryless error model.