Author:

- Elsayed Mohammed Elsayed Mostafa elsayedmmostafa255@gmail.com +201126411339

In this project, a channel coding and viterbi decoder is implemented from scratch.

Convolutional Encoder/Decoder

The implementation of the convolutional Encoder/Decoder enables using any prefered generative polynomials. The dimensions of the generative polynomials used specify implictly the constraint length (K) in addition to the information rate r.

In order to have a generic implementation, a new function named 'ploy_xor' is implemented to be able to apply XORing to the content of a matlab vector.

Implementation Procedure

To use the implemented functions, the generative plynomials must be specified at the first place besides the stream to be encoded/decoded.

The implemented functions do the following steps respectively:

1. Encode the stream into a new stream of bits after using the specified generative polynomials.
2. Decode the stream of bits into the corresponding bits without the parity bits using the vetrbi decoder algorithm.

Application

Here, an applicaiton to the previous procedures is shown with codes.

```
%Define the number of bits to use.
n_bits = 10;
% Get a random binary stream
stream = randi([0 1],[1,n_bits])
```

```
stream = 1×10
     0     0     1     1     1     1     0     1     1     1
```

In order to have an information rate = 1/b, the parity bits to be used is b. Hence, there must be b generative polynomials. In other words, the generative polynomials matrix must have b rows.

The number of columns in the generative polynomials matrix corresponds to the number of bits used to generate each parity bit i.e. the constraint length K.

To have information rate r = 1/3, and contraint length K = 3. Here is the generative polynomials matrix (gs).

```
gs = [1 1 1;
      1 0 1
      0 1 1];
```

Now, the setup is ready to encode the stream.

```
conv_stream  = conv_encoder(stream,gs)
```

```
conv_stream = 1×30
     0     0     0     0     0     0     1     1     1     0     1     0     1⋯
```

The decoding step also needs to have the generative polynomials matrix

```
decoded_conv_stream = viterbi_decoder(conv_stream,gs)
```

```
decoded_conv_stream = 1×10
     0     0     1     1     1     1     0     1     1     1
```

By comparing the two variables (stream , decoded_conv_stream), they are identical. Hence, our encoding/decoding procedures are working accuratly.

Inspecting the effect of channel encoding with AWGN.

In this section, AWGN is added to the stream of bits with/without using convolutional coding to simulate the effect of convoultional channel coding on error correction.

The following script uses a randomly generated binary 1000 bits in 2 different ways. The first one is applying the same previous encoding/decoding procedures with the second includes no application for this convolutional encoding/decoding. In both cases, the AWGN is added and the bit error rate (BER) is calculated at the decoder side.

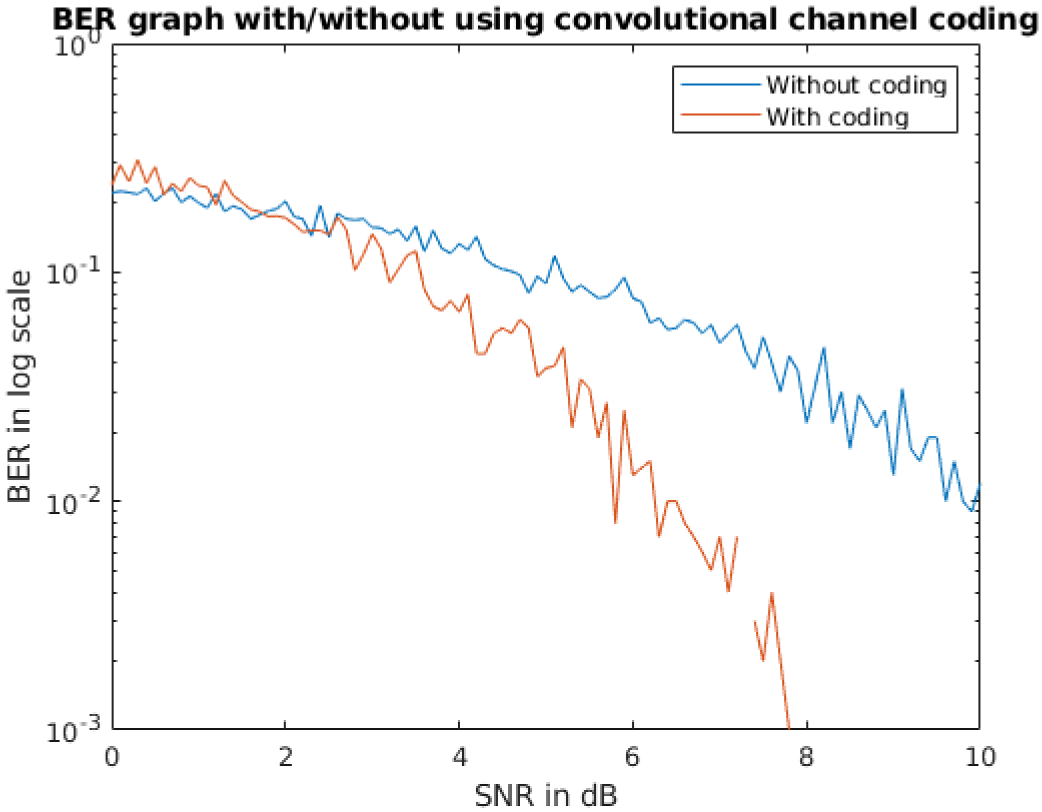
```
snr=0:0.1:10; %SNR in dB
stream = randi([0 1],[1,1000]);
with_coding=[];
without_coding=[];
 for iSNR = snr
 % ---------- With coding--------------
   conv_stream = conv_encoder (stream,gs);                     %Channel Encoding
   conv_stream = awgn_binary(conv_stream,iSNR);                %Noise added
   decoded_conv_stream = viterbi_decoder(conv_stream,gs);      %Channel Decoding
   BIRs = sum(decoded_conv_stream ~= stream);                  %Bit Errors
   BER = BIRs/length(stream);                                  %BER
   with_coding=cat(2,with_coding,BER);                         %Concatination
 %----------- Without Coding----------
   noisy_stream = awgn_binary(stream,iSNR);                    %Noise added
   BIRs = sum(noisy_stream ~= stream);                         %Bit Errors
   BER = BIRs/length(stream);                                  %BER
   without_coding=cat(2,without_coding,BER);                   %Concatination

 end
```

Hence, the convolutional coding effect on BER is ready to be plotted.

```
figure
semilogy(snr,without_coding);
hold on;
```

```
semilogy(snr,with_coding);
legend('Without coding','With coding');
xlabel('SNR in dB'); ylabel('BER in log scale');
title('BER graph with/without using convolutional channel coding');
```



The plot reveals the high effect of the channel coding in reducing the BER compared to the case when no channel coding is performed. The more SNR value the much more difference in performance between both cases (with/without channel coding) and the advantage is always to the existance of channel encoding.