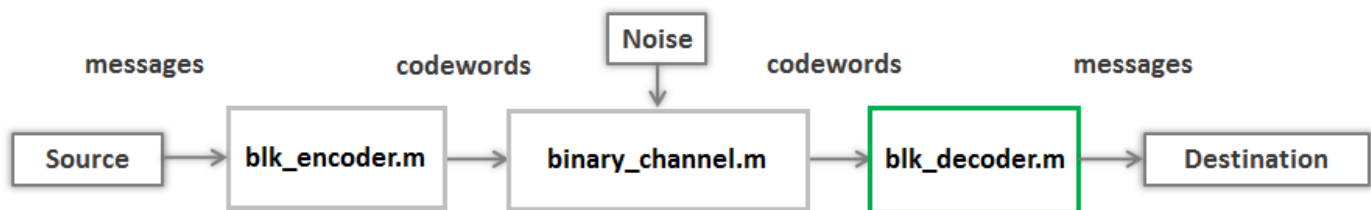


## LAB 10 TASK 2 - (8,4,3) REPETITION DECODER (1/1 point)

In this task, you will write code implementing the MATLAB function, `blk_decoder.m`, which returns the four message bits from an eight bit sequence, where we assume that the eight bit sequences were generated according to the (8,4,3) code discussed in lecture and then possibly corrupted by errors.



```
1 codewords = gen_messages(8); % cell array of all eight bit codewords
2 messages = cell(256,1);      % cell array to hold codewords
3
4 for i=1:256, % loop over all messages
5     codeword = codewords{i}; % take the i'th codeword from the array
6
7     % Modify the code below to return the 4 bit message after error
8     % correction assuming that the 8 bit codeword is received with at most
9     % one bit error.
10
11     message = codeword(1:4);
12     parity = codeword(5:8);
13     s1 = message(1) + message(2) + parity(1);
14     s2 = message(3) + message(4) + parity(2);
15     s3 = message(1) + message(3) + parity(3);
```

Correct

```
% assume no error at first
message = codeword(1:4);

% compute syndrome bits
% If we rearrange the codeword bits 1 to 8 as
%   1 2 5
%   3 4 6
%   7 8
% The parity checks correspond to checking the parity of the bits indexed
% by the columns of the matrix below
ind = [1 3 1 2;...
       2 4 3 4;...
       5 6 7 8 ];
% We can check parity by summing down the rows and then taking the modulus
% after division by two.
S = mod(sum(codeword(ind)),2);

% check for one bit errors in the message block only
% the function ISEQUAL(A,B) returns TRUE if A and B are equal
% the function NOT(X) returns 0 if X=1 and 1 if X=0.
% There are four possible one bit errors in the message block
if isequal(S,[1 0 1 0]),
    message(1)=not(message(1));
elseif isequal(S,[1 0 0 1]),
    message(2)=not(message(2));
elseif isequal(S,[0 1 1 0]),
    message(3)=not(message(3));
elseif isequal(S,[0 1 0 1]),
    message(4)=not(message(4));
end
```

Congratulations! All codewords correctly decoded.





*You have used 1 of 10 submissions*

## INSTRUCTIONS

### Step 1: Run the code as presented

The code above creates a 256x1 cell array called **codewords**, which contains all 256 possible eight bit sequences, e.g. [0 0 0 0 0 0 0], [0 0 0 0 0 0 1], etc. It also initializes a 256x1 cell array called **messages** to store the corresponding four bit messages contained within these codewords.

The **for** loop cycles through each of the possible codewords. It first extracts each vector stored in the cell array **codewords** as a 1x8 vector of binary digits called **codeword**. It then extracts the first four bits from **codeword** and assigns them to the vector **message**. Recall from lecture that the first four bits do contain the message bits, but these may be corrupted by bit errors. It then stores the vector **message** in the cell array **messages** for later checking.

After the for loop has extracted the messages from each of the 256 possible codewords, it passes the cell arrays **codewords** and **messages** to the function **check\_blk\_decoder**, which checks to see whether each message was correctly extracted with error correction from the corresponding codeword.

Click on the **Run Code** button. MATLAB will return a message indicating that one of the messages is incorrect. This is because we did not try to correct any errors. Even though there are many incorrect messages, the function **check\_blk\_decoder** only indicates the first one it encounters.

### Step 2: Implement the (8,4,3) decoder

#### Help

To complete this task, you should replace the line under the comments starting with

```
% Modify the code below
```

to return the four bit message *after error correction* for each of the codewords and return it as a 1x4 vector **message**.

Remember that in order to perform error correction with the (8,4,3) code, we must make an assumption that the number of bit errors in the codeword is at most one. However, since we have considered all possible 8 bit sequences, many of the received codewords may be a Hamming distance more than one from the valid codewords. (Recall that there are only 16 valid codewords, but 256 possible 8 bit sequences.) If our assumption of only one bit error is correct, we should never observe these sequences. However, we want our decoder to work for all possible bit sequences, so we should handle these cases anyway. In this case, your code should just return the first four bits of the codeword.

Hints:

1. In most cases, you will be returning the first four codeword bits unaltered, so it is a good idea to leave the assignment of message to the first four codeword bits in the initial code unchanged, and then change this if you detect an error.
2. Remember that in order to detect and correct errors, you must compute the four syndrome bits, one for each parity bit.
3. Since you are only interested in returning the four message bits and we assume that at most one bit error occurs, you are only interested in four error cases, one for each message bits. An error in each message bit corresponds to a particular combination of syndrome bits. Thus, you can just check if each of these four combinations occurs, and make the corresponding correction in the message bit.
4. Other combinations of syndrome bits either correspond to no error [0 0 0 0] or more than one error (which we've assumed never occur, but we need to handle just in case). In this cases, we should just return the first four bits of the codeword unaltered. This is handled already by the initialization.

### Step 3: Submit your work

If you have correctly computed all the messages, then running the MATLAB code should return a message indicating all messages are correct.

Once you have completed your work, click on the **Check** button to submit your answer.





EdX offers interactive online classes and MOOCs from the world's best universities. Online courses from MITx, HarvardX, BerkeleyX, UTx and many other universities. Topics include biology, business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

© 2014 edX, some rights reserved.

[Terms of Service and Honor Code](#)

[Privacy Policy \(Revised 4/16/2014\)](#)

#### About & Company Info

[About](#)

[News](#)

[Contact](#)

[FAQ](#)

[edX Blog](#)


[Donate to edX](#)

[Jobs at edX](#)

#### Follow Us

 [Twitter](#)

 [Facebook](#)

 [Meetup](#)

 [LinkedIn](#)

 [Google+](#)