



# Engineering Assignment Coversheet

Student Number(s)

817532

807780

Group Code (if applicable):

Please note that you:

- Must keep a full copy of your submission for this assignment
- Must staple this assignment
- Must NOT use binders or plastic folders except for large assignments

Assignment Title:	Convolutional Coding
Subject Number:	ELEN90051
Subject Name:	Advanced Communication Systems
Student Name:	Yuxuan Li Yuntao Wu
Lecturer/Tutor:	
Due Date:	

## For Late Assignments Only

Has an extension been granted? Yes / No (circle)

A per-day late penalty may apply if you submit this assignment after the due date/extension. Please check with your Department/coordinator for further information.

## Plagiarism

Plagiarism is the act of representing as one's own original work the creative works of another, without appropriate acknowledgment of the author or source.

## Collusion

Collusion is the presentation by a student of an assignment as his or her own which is in fact the result in whole or in part of unauthorised collaboration with another person or persons. Collusion involves the cooperation of two or more students in plagiarism or other forms of academic misconduct.

Both collusion and plagiarism can occur in group work. For examples of plagiarism, collusion and academic misconduct in group work please see the University's policy on Academic Honesty and Plagiarism:

<http://academichonesty.unimelb.edu.au/>

Plagiarism and collusion constitute cheating. Disciplinary action will be taken against students who engage in plagiarism and collusion as outlined in University policy. Proven involvement in plagiarism or collusion may be recorded on my academic file in accordance with Statute 13.1.18.

## STUDENT DECLARATION

Please sign below to indicate that you understand the following statements:

I declare that:

- This assignment is my own original work, except where I have appropriately cited the original source.
- This assignment has not previously been submitted for assessment in this or any other subject.

For the purposes of assessment, I give the assessor of this assignment the permission to:

- Reproduce this assignment and provide a copy to another member of staff; and
- Take steps to authenticate the assignment, including communicating a copy of this assignment to a checking service (which may retain a copy of the assignment on its database for future plagiarism checking).

Student signature

李雨萱 吴天宇

Date

20/05/2018

# Convolutional Coding

Yutao Wu 807780 & Yuxuan Li 817532

19/05/2018

## Task 1

The Python code of ConvEncoder are shown below:

```
# implement the encoder
def ConvEncoder(SourceSequence, GenPoly):
    # The SourceSequence is a stream of bits with {0,1} in string type
    # The GenPoly is the matrix. In this case GenPoly[0] = g1 = [1,1,1]
    GenPoly[1] = g2 = [1,1,0]
    # matrix multiply are using for calculation efficiency
    EncodedCodeSequence = list()
    reg = np.zeros((1,3)) # The initial state of register are [0,0,0]
    #print(reg.shape)
    for curBit in SourceSequence: # for every bit in the sequence
        # Insert the curBit into the reg
        print('curbit_{}'.format(curBit))
        reg = np.array([curBit, reg[0][0], reg[0][1]])
        print('Registeris_{}'.format(reg))
        print(reg.shape)
        y = np.dot(reg, GenPoly) # y = [x0+x1+x2, x0+x1]
        y = np.mod(y, 2)
        EncodedCodeSequence = EncodedCodeSequence + [int(y[0][0]), int(y[0][1])]
    return np.array(EncodedCodeSequence)
```

With source code to be 1001001, the output are shown below, which is correct.

```
In [164]: GenPoly = np.array([[1,1],[1,1],[1,0]])
          source = np.array([1,0,0,1,0,0,1])
          ConvEncoder(source, GenPoly)

Out[164]: array([1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1])
```

Figure 1: Output of CovEncoder

## Task 2

1.

We assume the initial state is 00, and the source sequence is of length 5 using the encoder above. Also, the noise of the channel is no greater than the signals. There are  $2^5 = 32$  possible inputs. The most possible one is **11000**, which has only 1 bits error.

2.

The Figure are shown below of the value of path and branch of the optimized path indicated in yellow pen. We can see the optimum path is **11000** the same as previous question.

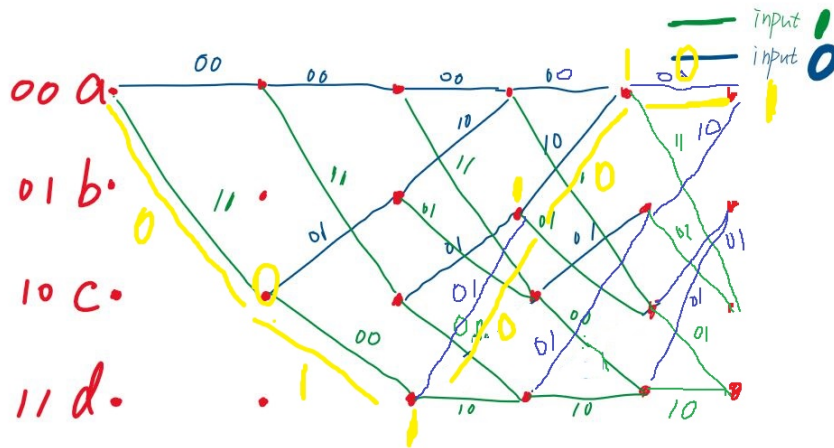


Figure 2: Vertb with Trellis Diagram

3.

When applied Soft Decision, the value of the branch and path should be changed from the hamming distance (# of different bits) to the Euclid Distance  $\sqrt[n]{\sum^n (y_i - y_{1i})^2}$

## Task 3

1.

The Python code are shown below:

```
def ConvDecoder(ReceivedSequence, Type):
    F = np.ones([4, ReceivedSequence.shape[0]//2+2])*0xFFFF # for state's F[00]
    F[01] F[10] F[11] set the large value for init;
    G = np.zeros([4, ReceivedSequence.shape[0]//2+2, 2]) # to store the decision
    # make. to recover the path to get the decode code.
    #Reg = np.zeros(4)
    F[0,0] = 0
    for i in range(1, (ReceivedSequence.shape[0]//2)+1, 1):
        for cur_state in range(4):
            for pre_state in range(4): # find the min value for
                # previous state to current state
                trans = str(pre_state)+'to'+str(cur_state)
                if(trans in stateTrs.keys()):
                    #if the state transform is exists()
                    predOut = np.array(stateTrs[trans][1])
                    realOut = ReceivedSequence[(i-1)*2:((i-1)
                        *2+1)+1]
                    # The state diagram are stored in a
                    # stateTrs dictionary
```

```

dis = Distance(predOut,realOut,Type)
# determine the distance between predict
sequence and received sequence

if dis+F[pre_state,i-1]<=F[cur_state,i]:
    F[cur_state,i] = dis+F[pre_state,i-1]
# update the min Value.
G[cur_state,i][1] = pre_state
G[cur_state,i][0] = stateTrs[trans][0]
# store the cur_state is from
pre_state and input.

# Find the min values of the final node.
minState = 0
minVal = 0xFFFF
for item in range(4):
    if (F[item,i]<minVal):
        minState = item
        minVal = F[item,i]

EncodedCodeSequence = list()
# use a recursive function to generate the predicted input sequence using G
array
findCode(G,minState,i,EncodedCodeSequence)

return np.array(EncodedCodeSequence[::-1])

```

2.

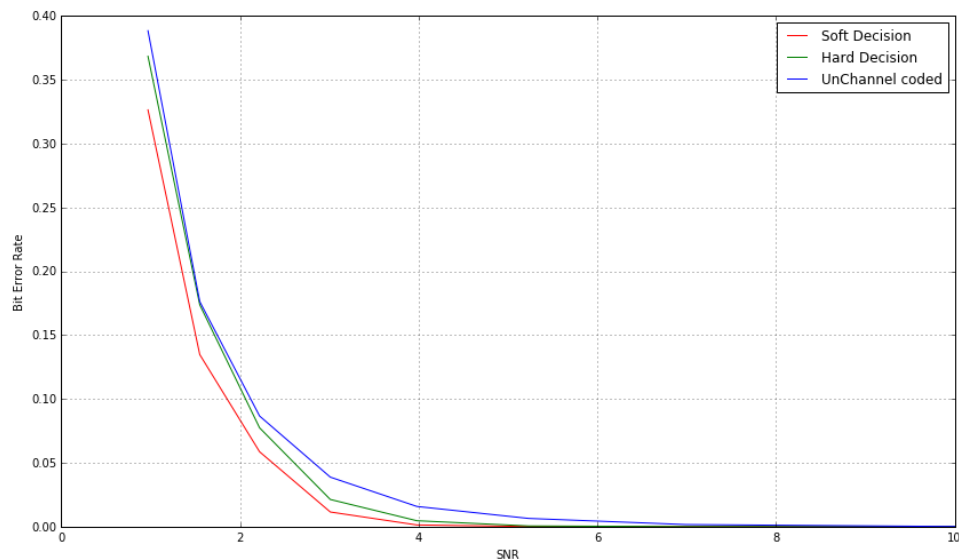


Figure 3: BER vs SNR Performance

### 3.

We could see from the Fig.3 that with high SNR,even with no channel coding,the ML detection with bpsk has a good error performance to be nearly 0 which is same as the one with channel coding.However,with noise increasing,the channel coding shows it better error performance compared with un-channel coding. The Soft and Hard decision method shows little error performance in high SNR,with no error.However,in low SNR,the soft decision method shows better error performance which is consistent with the theory.

## Reflections and Comparisons

### 1.

Error Correcting codes are used to provide efficient transmission of data over the noisy channel. Convolutional codes are one of the FEC (Forward error correction) codes which are being used in wireless communication from early 1970s[2].

Convolutional codes were introduced in 1955 by Peter Elias. It was thought that convolutional codes could be decoded with arbitrary quality at the expense of computation and delay.[1] Convolutional codes transform a whole sequence of information bits into a sequence of encoded bits by convolving the information bits with a set of generator coefficients.[2] In 1967 Andrew Viterbi determined that convolutional codes could be maximum-likelihood decoded with reasonable complexity.[1].The Viterbi algorithm is just a dynamic programming method which reduced the time complexity from exponential to polynomial.

In computer science,the dynamic programming algorithm has the property of **optimal substructure** and **overlapping sub-problems**.With these two properties,the algorithm can determine the global optimal value in polynomial time complexity with more space consumptions.Therefore,it is a time and space trade-off.

### 2.

If we need to change a Viterbi equalizer to the Viterbi decoder,the branch's metrics need to be changed from the conditional probability to the distance.For Hard decision,the distance is hamming distance while for Soft decision,the distance is Euclidean distance.

Also the state of the optimized function need to be changed from only two state 0,1 in equalizer to more states depending on the coding method.

## References

- [1] Convolutional code. Website, 2018. [https://en.wikipedia.org/wiki/Convolutional\\_code#History](https://en.wikipedia.org/wiki/Convolutional_code#History).
- [2] Geetanjali Baghel and Ms Mamta Arora. Review paper convolutional codes in a nutshell. *IJRIT International Journal of Research in Information Technology*, Volume 3, Issue 3, March 2015, Pg. 77-83, 2015.