

ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ

1ο σετ Εργαστηριακών Ασκήσεων

Λουδάρος Ιωάννης - AM 1067400

```
% Μεταφορά του Current Folder στον κατάλογο του αρχείου ώστε να υπάρχει  
% πρόσβαση στις απαιτούμενες συναρτήσεις.
```

```
clear;clc;close all
```

```
cd(fileparts(matlab.desktop.editor.getActiveFilename))  
addpath './functions'  
addpath './sources'
```

Table of Contents

1ο σετ Εργαστηριακών Ασκήσεων	1
Μέρος Α.....	1
Η πηγή	1
1.a. Συχνότητα Εμφάνισης τιμών στο Μητρώο	2
1.b. Κωδικοποίηση Huffman.....	4
2.a. Σύμβολα Δεύτερης Τάξης Επέκτασης Πηγής.....	5
2.b. Κωδικοποίηση Εικόνας.....	6
3.a. Υπολογισμός Εντροπίας	9
3.b. Φράγμα Μέσου Μήκους Κώδικα	9
4. Επαλήθευση Κωδικοποίησης και Λόγος Συμπίεσης	10
5. Μετάδοση Σήματος μέσα από Κανάλι	10
Μέρος Β.....	12
Η πηγή	12
1.Κωδικοποίηση και Αποκωδικοποίηση.....	13
2.Πως Επηρεάζουν οι Παράμετροι p και N το Σφάλμα Πρόβλεψης;.....	15
3.Πως Επηρεάζουν οι Παράμετροι p και N το Μέσο Τετραγωνικό Σφάλμα Πρόβλεψης;.....	18
4.Επανακατασκευή του Σήματος.....	19

Μέρος Α

Η πηγή

```
image = imread('parrot.png');
```

```
image_stream = reshape(image,1,[])
```

```
image_stream = 1x30000 uint8 row vector
```

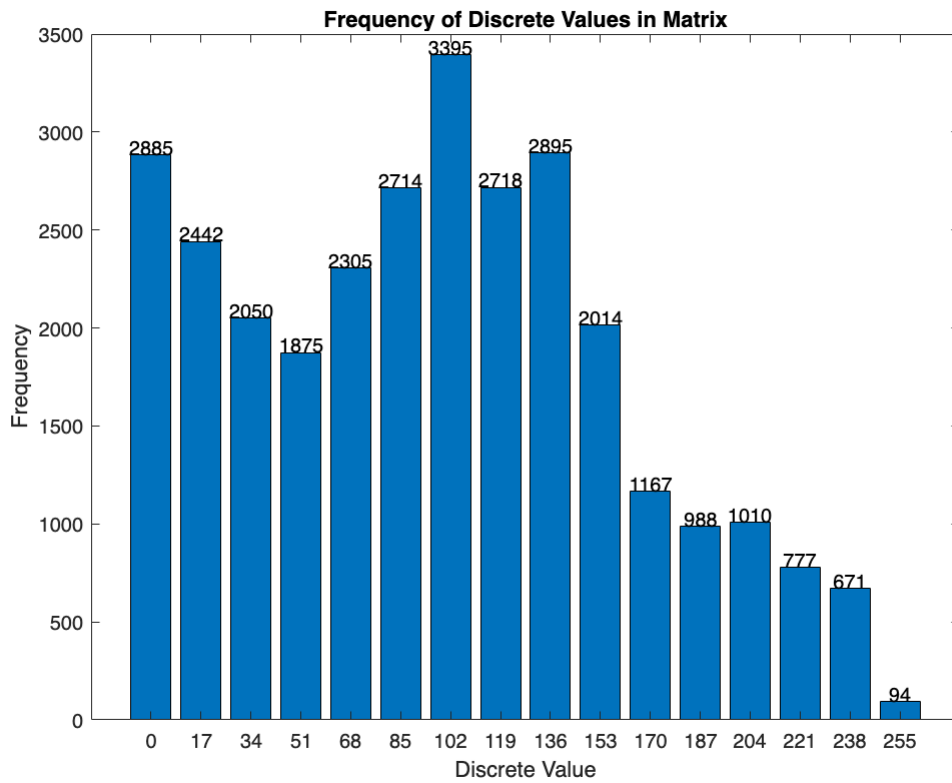
```
0 0 0 17 0 0 0 0 0 0 0 0 0 0 ...
```

```
binary_image_stream =  
reshape((dec2bin(typecast(image(:),'uint8'),4)-'0').',1,[])
```

```
binary_image_stream = 1x240000  
0 0 0 0 0 0 0 0 0 0 0 0 0 ...
```

1.a. Συχνότητα Εμφάνισης τιμών στο Μητρώο

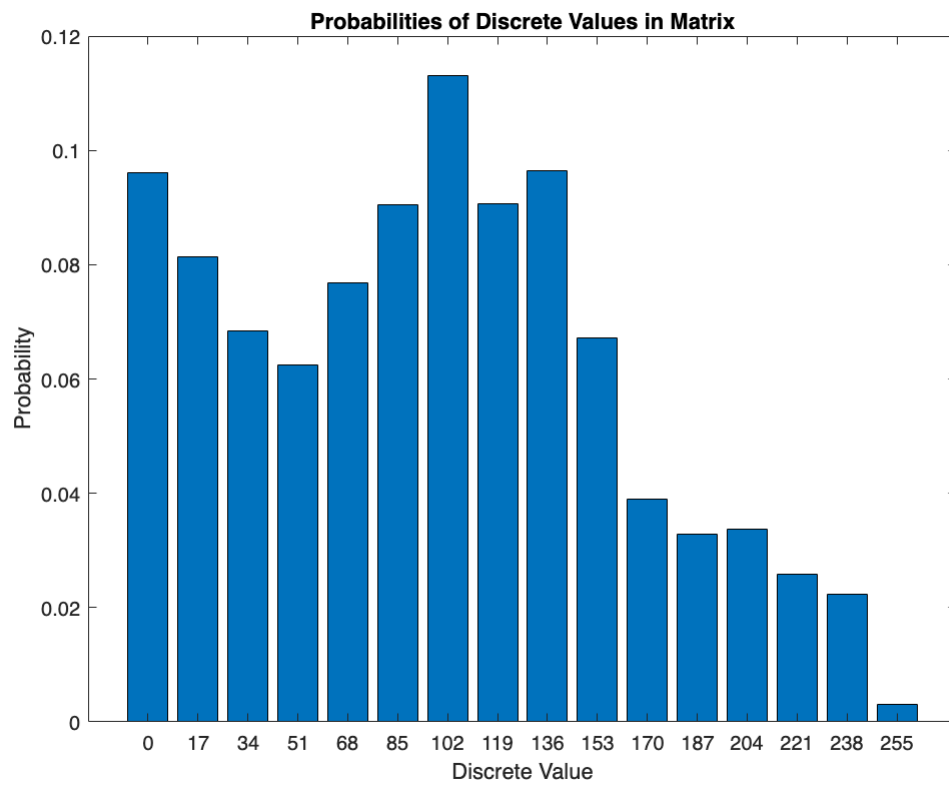
```
% Get the unique values in the matrix  
unique_values = unique(image);  
  
% Count the occurrences of each unique value  
frequency = zeros(size(unique_values));  
for i = 1:size(image, 1)  
    for j = 1:size(image, 2)  
        frequency(unique_values == image(i, j)) = frequency(unique_values  
== image(i, j)) + 1;  
    end  
end  
  
% Create a bar graph to make the frequencies easily readable  
figure;  
bar(unique_values, frequency);  
xticks(unique_values);  
xticklabels(unique_values);  
xlabel('Discrete Value');  
ylabel('Frequency');  
title('Frequency of Discrete Values in Matrix');  
  
% Add labels to the top of each bar  
for i = 1:size(unique_values,1)  
    text(unique_values(i), frequency(i)+100, num2str(frequency(i)), ...  
        'VerticalAlignment', 'top', 'HorizontalAlignment', 'center');  
end
```



```
% Get the probability of each integer for the huffman dict later
probability = frequency./ (200*150);
```

```
% Create a bar graph to make the probabilities easily readable
```

```
figure;
bar(unique_values, probability);
xticks(unique_values);
xticklabels(unique_values);
xlabel('Discrete Value');
ylabel('Probability');
title('Probabilities of Discrete Values in Matrix');
```



1.b. Κωδικοποίηση Huffman

```
% Create a dictionary for the encoding
[dict, avglen] = huffmandict(unique_values, probability)
```

dict = 16x2 cell

	1	2
1	0	[1,1,0]
2	17	[0,0,1,0]
3	34	[0,1,0,0]
4	51	[0,1,1,1]
5	68	[0,0,1,1]
6	85	[0,0,0,0]
7	102	[1,0,0]
8	119	[1,1,1]
9	136	[1,0,1]
10	153	[0,1,0,1]
11	170	[0,0,0,1,1]

	1	2
12	187	[0,1,1,0,1]
13	204	[0,1,1,0,0]
14	221	[0,0,0,1,0,0]

⋮

avglen = 3.8374

% Encode the Source

```
encoded_image_stream = huffmanenco(image_stream, dict);
```

% Compute the entropy

```
entropy = -sum(probability .* log2(probability))
```

entropy = 3.7831

% Compute efficiency

```
efficiency = entropy/avglen
```

efficiency = 0.9859

2.a. Σύμβολα Δεύτερης Τάξης Επέκτασης Πηγής

% Finding the symbols

% (all the possible permutations of unique_values).

```
new_symbols = zeros(size(unique_values,1)^2,2);
```

```
new_symbol_prob = zeros(size(unique_values,1)^2,1);
```

```
for j = 1 : size(unique_values)
```

```
    for k = 1 : size(unique_values)
```

```
        new_symbols((j-1)*size(unique_values,1)+k, :) = [unique_values(j)  
unique_values(k)];
```

```
        new_symbol_prob((j-1)*size(unique_values,1)+k) =  
probability(j)*probability(k);
```

```
    end
```

```
end
```

```
new_symbols
```

```
new_symbols = 256x2
```

```
0     0  
0    17  
0    34  
0    51  
0    68  
0    85  
0   102  
0   119
```

```
0 136
0 153
⋮
```

new_symbol_prob

```
new_symbol_prob = 256x1
0.0092
0.0078
0.0066
0.0060
0.0074
0.0087
0.0109
0.0087
0.0093
0.0065
⋮
```

2.b. Κωδικοποίηση Εικόνας

```
% Find the symbols that are actually used.
unique_pairs = [];
for i = 1 : 2 : size(image_stream,2)
    temp = [image_stream(i) image_stream(i+1)];

    % Does this pair already exists?
    flag = 0;
    for j = 1:size(unique_pairs,1)
        if j == 0
            break;
        else
            if temp == unique_pairs(j,:)
                flag = 1;
            end
        end
    end

    % If not, add it
    if (~flag)
        unique_pairs = [unique_pairs; temp];
    end
end

unique_pairs = sortrows(unique_pairs,[1,2])
```

```
unique_pairs = 178x2 uint8 matrix
0 0
```

```

0    17
0    34
0    51
0    68
0    85
0   102
17    0
17   17
17   34
:

```

```

% Find the frequency of the new symbols.
unique_pair_frequency = zeros(size(unique_pairs,1),1);
for i = 1:2:size(image_stream, 2)
    temp = [image_stream(i) image_stream(i+1)];

    for j = 1:size(unique_pairs,1)
        if unique_pairs(j,:) == temp
            unique_pair_frequency(j) = unique_pair_frequency(j)+1;
            break;
        end
    end
end

unique_pair_frequency

```

```

unique_pair_frequency = 178x1
    1232
     207
      18
       8
       2
       1
       2
     170
     752
     220
      :

```

```

% Get the probability of each pair for the huffman dict later
unique_pair_probability = unique_pair_frequency ./ ((200*150)/2)

unique_pair_probability = 178x1
    0.0821
    0.0138
    0.0012
    0.0005
    0.0001
    0.0001

```

```

0.0001
0.0113
0.0501
0.0147
⋮

```

```

% Convert the previous stream to use the new symbols
image_stream_converted = zeros(1,size(image_stream,2)/2);
for i = 1:2:size(image_stream, 2)
    temp = [image_stream(i) image_stream(i+1)];

    for j = 1:size(unique_pairs,1)
        if unique_pairs(j,:) == temp
            image_stream_converted(ceil(i/2)) = j;
            break;
        end
    end
end
end

image_stream_converted

```

```

image_stream_converted = 1×15000
    1     2     1     1     1     1     1     1     1     1     1     1     9 ...

```

```

% Create a dictionary for the encoding
[dict2, avglen2] = huffmandict(1:178, unique_pair_probability)

```

```
dict2 = 178×2 cell
```

	1	2
1	1	[0,0,0,1]
2	2	[1,0,1,0,0,1]
3	3	[0,0,0,0,1,1,1,1,1,0]
4	4	1×11 double
5	5	1×13 double
6	6	1×14 double
7	7	1×13 double
8	8	[0,0,0,0,0,1,1]
9	9	[1,1,0,1]
10	10	[0,1,1,1,0,1]

	1	2
11	11	[0,0,0,0,1,0,0,0,1]
12	12	[0,1,1,1,1,0,1,0,0,0]
13	13	1×12 double
14	14	1×12 double

⋮

avglen2 = 5.6412

% Encode the Source

```
encoded_image_stream_converted = huffmanenco(image_stream_converted, dict2);
```

% Compute the entropy

```
entropy2 = -sum(unique_pair_probability .* log2(unique_pair_probability))
```

entropy2 = 5.6147

% Compute efficiency

```
efficiency2 = entropy2/avglen2
```

efficiency2 = 0.9953

3.a. Υπολογισμός Εντροπίας

Όπως θίξαμε ήδη από την δεύτερη παράγραφο του ερωτήματος 2.a, η διαδικασία που ακολουθήσαμε δεν δημιούργησε ακριβώς την δεύτερης τάξης επέκταση πηγής του ερωτήματος 1, αλλά μια μικρότερη που περιέχει ένα υποσύνολο των συμβόλων της. Έτσι από τα 256 που θα έπρεπε να είναι κανονικά τα σύμβολα, συναντήσαμε μόνο 178.

3.b. Φράγμα Μέσου Μήκους Κώδικα

Το φράγμα για το μέσο μήκος όπως διατυπώνεται στις σημειώσεις του μαθήματος ισχύει και στις δύο πηγές.

$$H(X) \leq \bar{L} < H(X) + 1$$

Δηλαδή :

$3.7831 \leq 3.8374 < 4.7831$, που είναι αληθές, και

$5.6147 \leq 5.6412 < 6.6147$, που είναι επίσης αληθές.

4. Επαλήθευση Κωδικοποίησης και Λόγος Συμπίεσης

```
% Decode and validate
decoded_image_stream = huffmandeco(encoded_image_stream,dict);

check = all(image_stream == decoded_image_stream)

check = logical
      1
```

```
% Calculate Compression Rate
Compression_Rate = size(encoded_image_stream,2) / (size(image_stream,2)*4)

Compression_Rate = 0.9593
```

```
Compression_Rate_uint8 = size(encoded_image_stream,2) /
(size(image_stream,2)*8)

Compression_Rate_uint8 = 0.4797
```

```
%ή ισοδύναμα --> Compression_Rate_uint8 = size(encoded_image_stream,2) /
(size(binary_image_stream,2))
```

5. Μετάδοση Σήματος μέσα από Κανάλι

```
% Transmit through the channel
received = binary_symmetric_channel(encoded_image_stream);
```

```
% Calculate the p parameter for this transmission
% (and the joint probabilities pxy)
counter = 0;
pxy = zeros(2,2);
px = zeros(2,1);
py = zeros(2,1);
for i = 1:size(received,2)
    if received(i) == encoded_image_stream(i)
        counter = counter + 1;
    end
    received_bit = received(i);
    sent_bit = encoded_image_stream(i);

    pxy(sent_bit+1,received_bit+1) = pxy(sent_bit+1,received_bit+1) + 1;
```

```

    if encoded_image_stream(i)==0
        px(1)=px(1)+1;
    end

    if received(i)==0
        py(1)=py(1)+1;
    end
end

px(1) = px(1)/size(encoded_image_stream,2);
px(2) = 1 - px(1);

py(1) = py(1)/size(received,2);
py(2) = 1 - py(1);

px

```

```

px = 2x1
    0.5519
    0.4481

```

```
py
```

```

py = 2x1
    0.5381
    0.4619

```

```
pxy= pxy./size(encoded_image_stream,2)
```

```

pxy = 2x2
    0.4851    0.0668
    0.0529    0.3951

```

```
p = counter / size(received,2)
```

```
p = 0.8803
```

```
% Calculate the Capacity of the Channel
```

```
H = - p * log2(p) - (1-p)*log2(1-p)
```

```
H = 0.5286
```

```
C = 1 - H
```

```
C = 0.4714
```

```
% Mutual Information
```

```
MI=0;
```

```
for i = 1:2
```

```
    for j = 1:2
```

```
        MI = MI + pxy(i,j) * log2((pxy(i,j)/px(i))/py(j));
```

```
end  
end
```

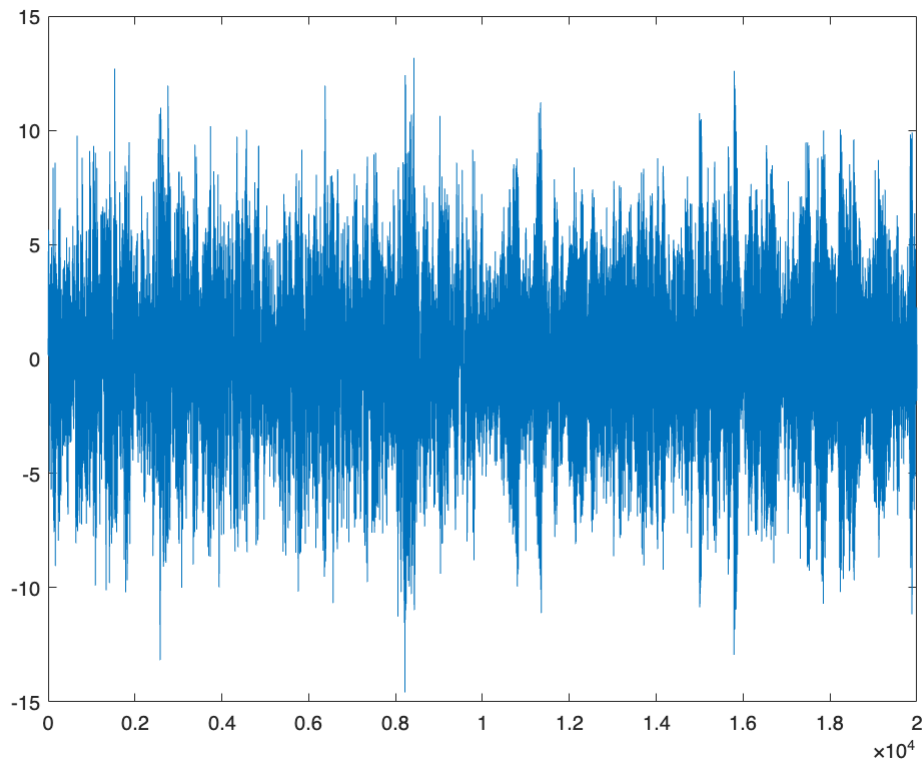
```
MI
```

```
MI = 0.4672
```

Μέρος Β

Η πηγή

```
load source.mat;  
plot(t)
```



```
%sound(t); %Δεν ακούγεται κάτι ευχάριστο ή κατανοήσιμο.
```

Υπολογισμός Στοχαστικών Ποσοτήτων

```
p=5;  
[R,r] = Rx(p,t)
```

```
R = 5x5  
12.4251    5.5417   -4.5025   -5.1474   -0.8877  
 5.5417    12.4251    5.5416   -4.5024   -5.1473
```

```

-4.5025    5.5416   12.4250    5.5416   -4.5023
-5.1474   -4.5024    5.5416   12.4250    5.5416
-0.8877   -5.1473   -4.5023    5.5416   12.4250
r = 5×1
    5.5415
   -4.5025
   -5.1477
   -0.8879
   -0.7998

```

Υπολογισμός και Κβάντιση Συντελεστών Φίλτρου Πρόβλεψης

```
a = R \ r
```

```

a = 5×1
    1.2852
   -1.5856
    0.9901
   -0.5424
   -0.0287

```

```
[a_quantised_areas, a_centers] = iquantizer(a, 8, -2, 2);
```

```
a_quantised = a_centers(a_quantised_areas)
```

```

a_quantised = 5×1
    1.2988
   -1.5723
    0.9980
   -0.5332
   -0.0410

```

1.Κωδικοποίηση και Αποκωδικοποίηση

Σταθερές

```

N = 4;
min_value = -3.5;
max_value = 3.5;

```

Η κωδικοποίηση

```
[encoded, centers, a, y] = idpcm_enco(t, p, N, min_value, max_value)
```

```

encoded = 20000×1
    7
    8
    9
    4
    2

```

```

9
1
1
5
1
:
:
centers = 16x1
3.2812
2.8438
2.4062
1.9688
1.5312
1.0938
0.6562
0.2188
-0.2188
-0.6562
:
:
a = 5x1
1.2988
-1.5723
0.9980
-0.5332
-0.0410
y = 20000x1
0.8147
0.0534
-0.2324
1.7598
2.7576
-0.2853
3.7410
3.1180
1.3783
3.4059
:
:

```

Σημείωση : Είναι προφανές ότι θα μπορούσαμε να χρησιμοποιούμε την Rx καθώς και να παράγουμε τα κέντρα των κβαντισμένων περιοχών έξω από την συνάρτηση κωδικοποίησης. Ο μόνος λόγος που προτιμήσαμε να υπολογίζονται όλα εντός της, είναι για να διευκολυνθούμε στα επόμενα ερωτήματα με τις πολλαπλές επαναλήψεις κωδικοποιήσεων με διαφορετικές παραμέτρους.

Η αποκωδικοποίηση

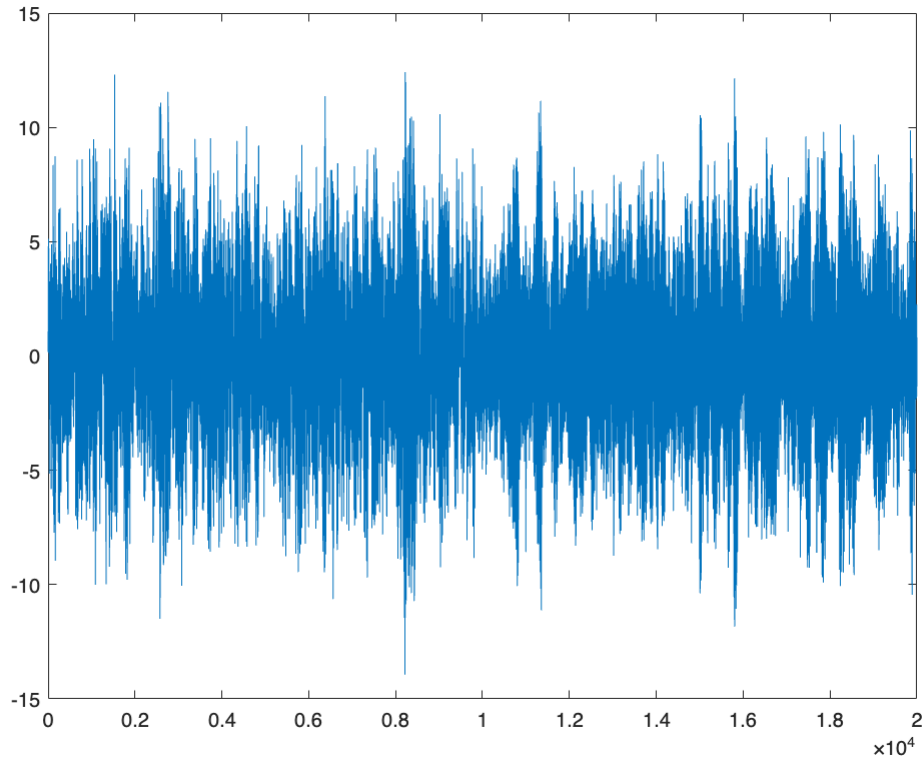
```
reconstructed = idpcm_deco(encoded, a, centers)
```

```
reconstructed = 1x20000
0.6562    1.0711    0.1406    1.1223    4.7994    3.7926    1.6625    3.6633 ...
```

```
t'
```

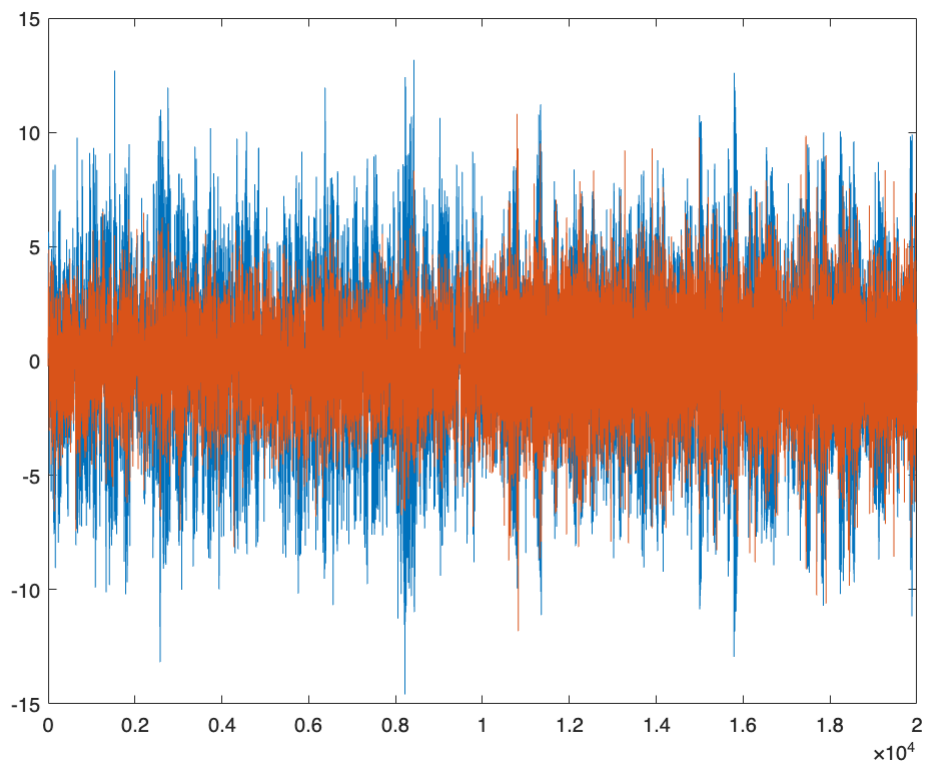
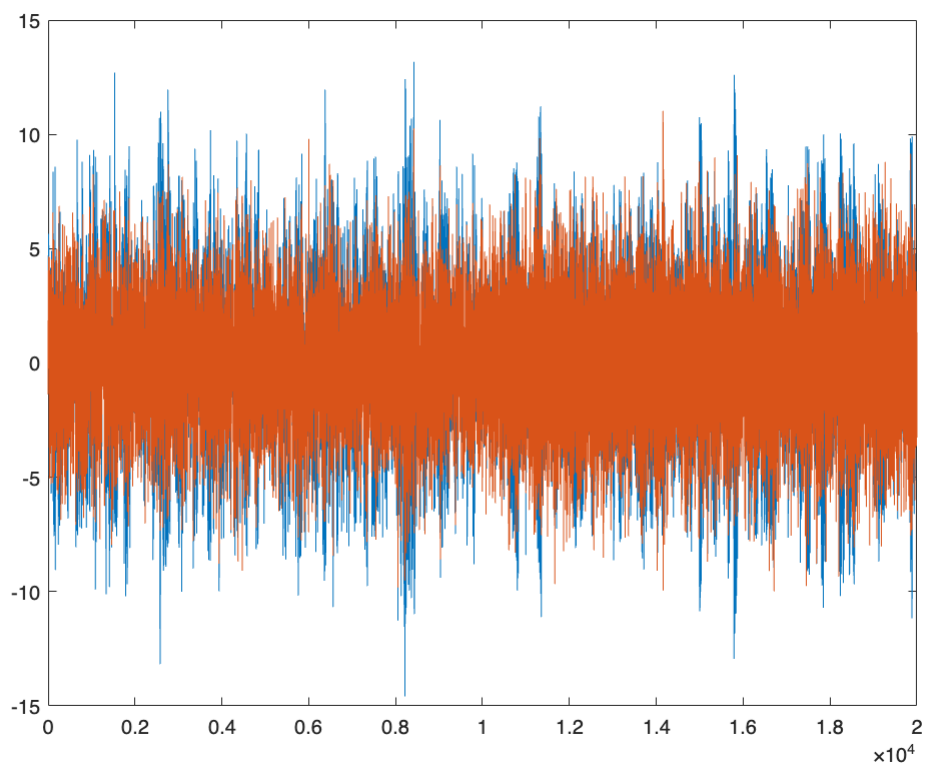
```
ans = 1x20000
0.8147    0.9058    0.1270    0.9134    4.7133    3.7260    2.1223    3.5001 ...
```

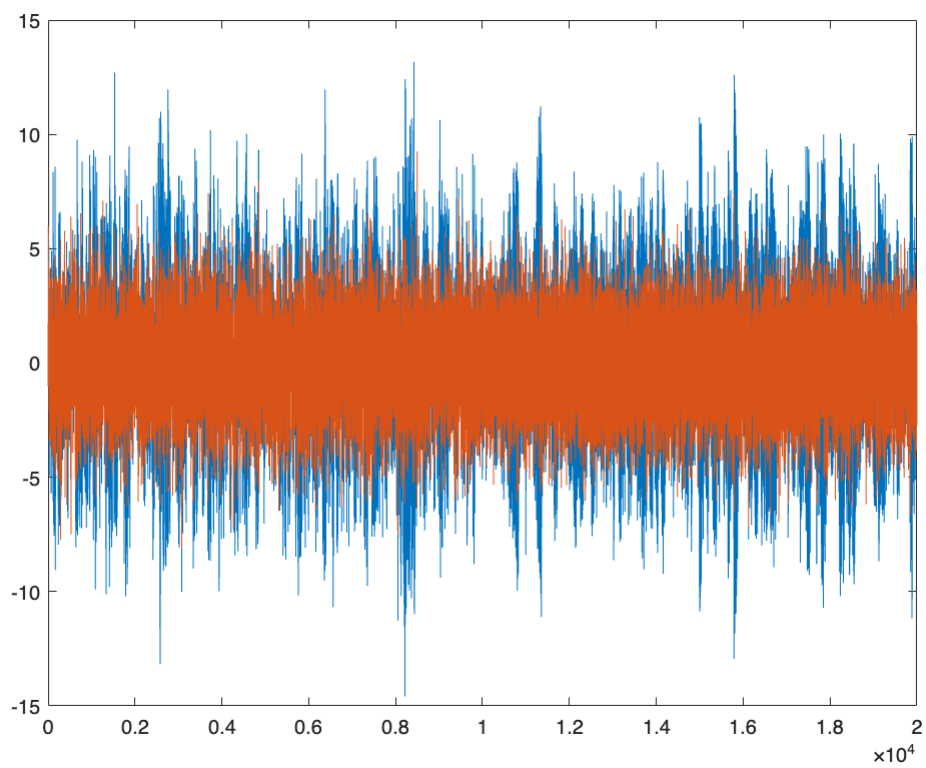
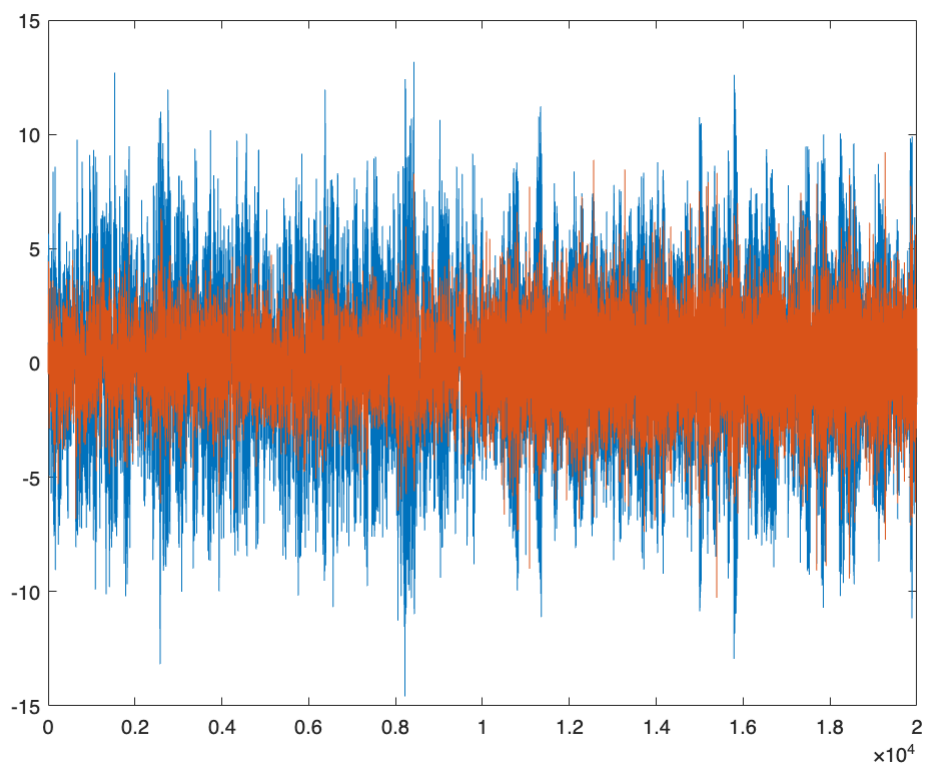
```
plot(reconstructed);
```

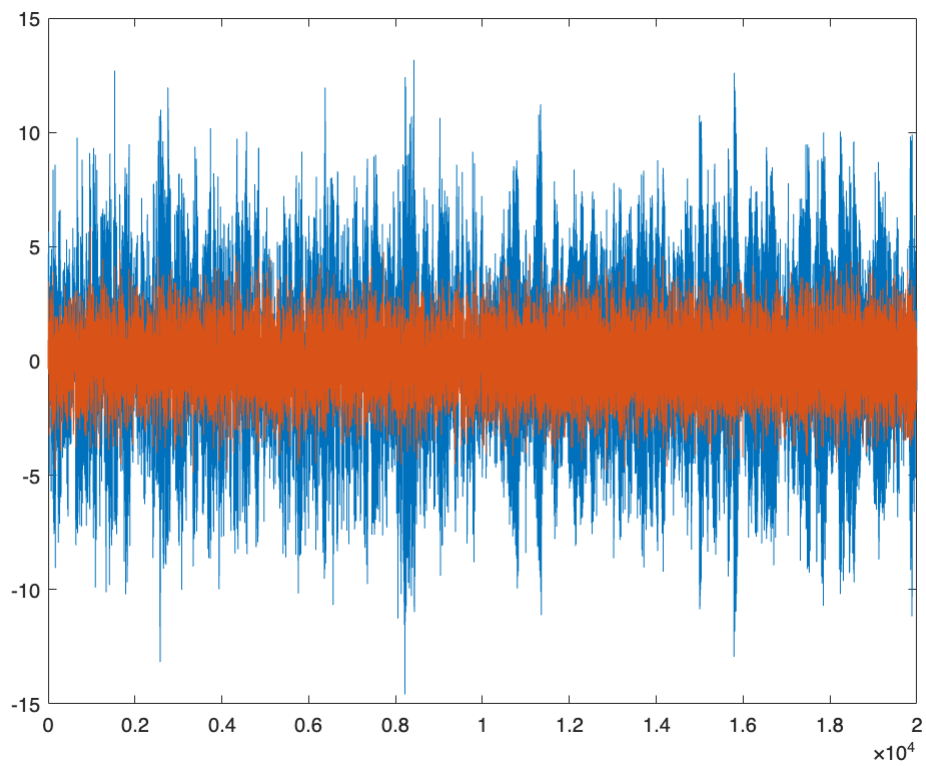
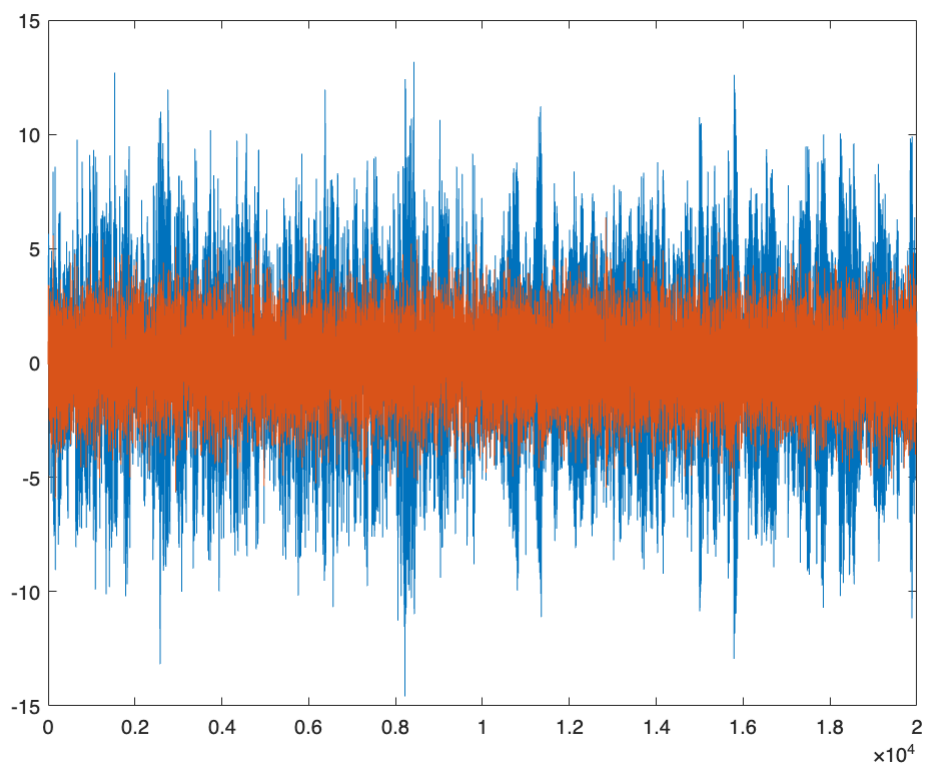


2.Πως Επηρεάζουν οι Παράμετροι p και N το Σφάλμα Πρόβλεψης;

```
for p = [5,10]
    for N = 1:3
        [encoded, centers, a, y] = idpcmenco(t, p, N, min_value,
max_value);
        figure;
        plot(t);
        hold on
        plot(y);
        hold off
    end
end
```







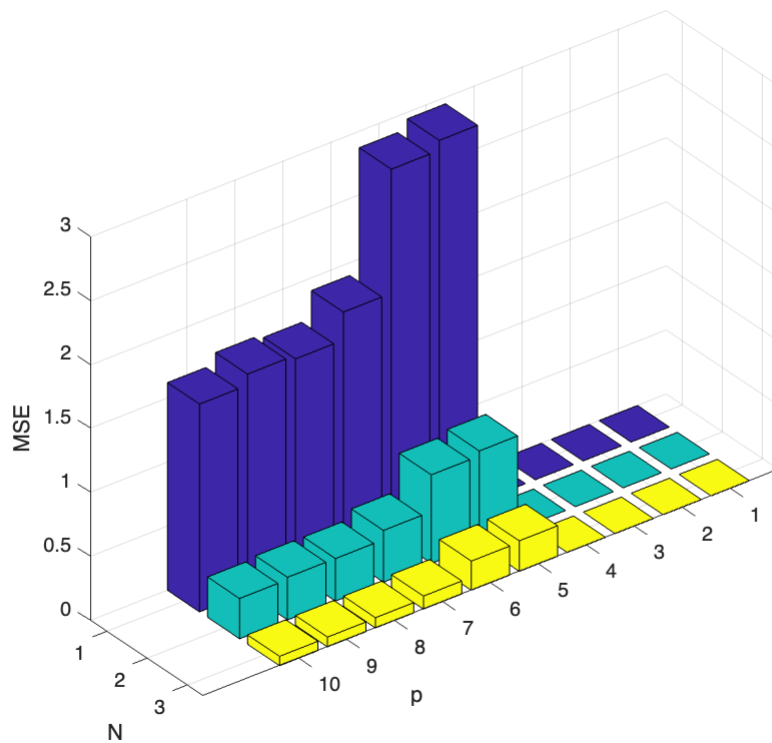
3.Πως Επηρεάζουν οι Παράμετροι ρ και N το Μέσο Τετραγωνικό Σφάλμα Πρόβλεψης;

```

for p = 5:10
    for N = 1:3
        [encoded, centers, a, y] = idpcmenco(t, p, N, min_value,
max_value);
        reconstructed = idpcm_deco(encoded, a, centers);
        MSE(p,N) = 1/size(reconstructed,2) * sum((t-reconstructed').^2);

    end
end
bar3(MSE);
xlabel('N');
ylabel('p');
zlabel('MSE');
view([52.59 30.95])

```



4.Επανακατασκευή του Σήματος

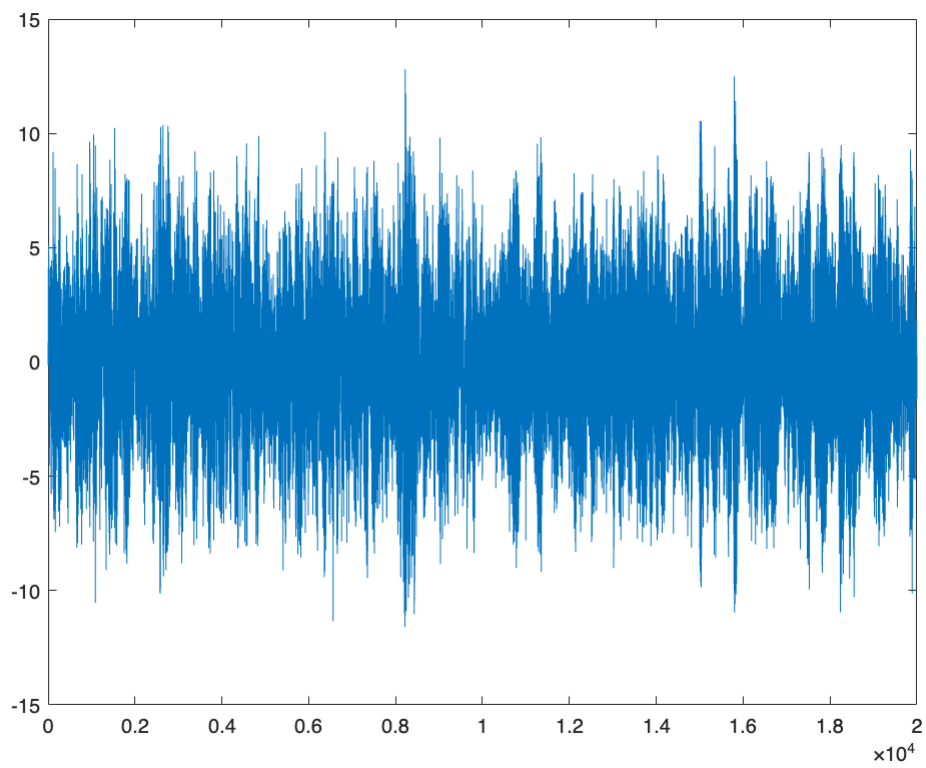
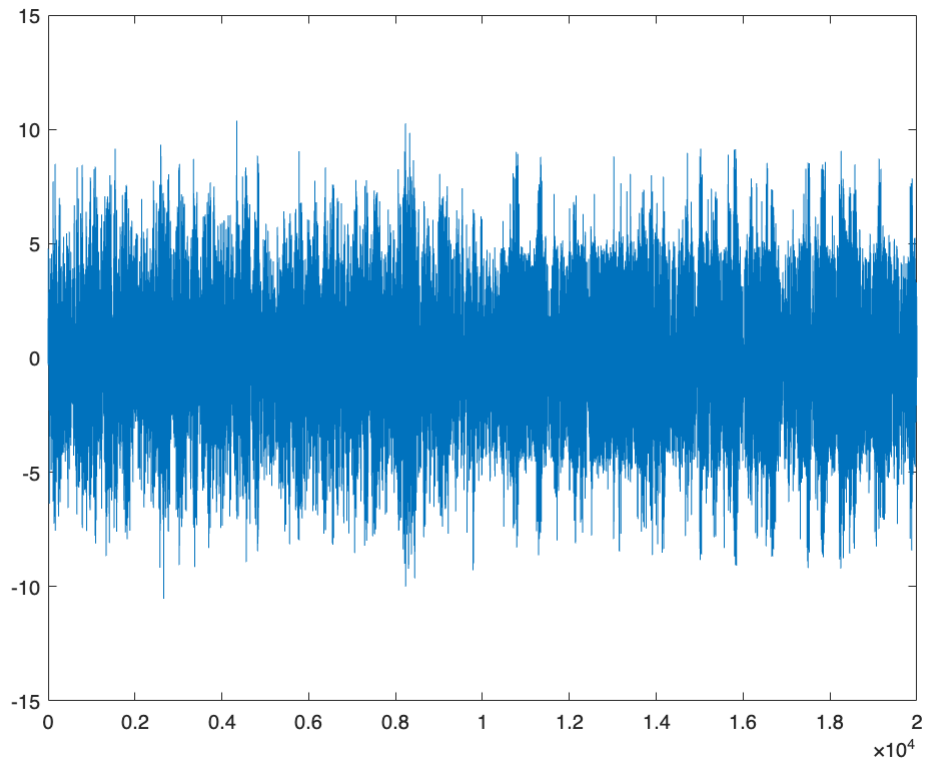
```

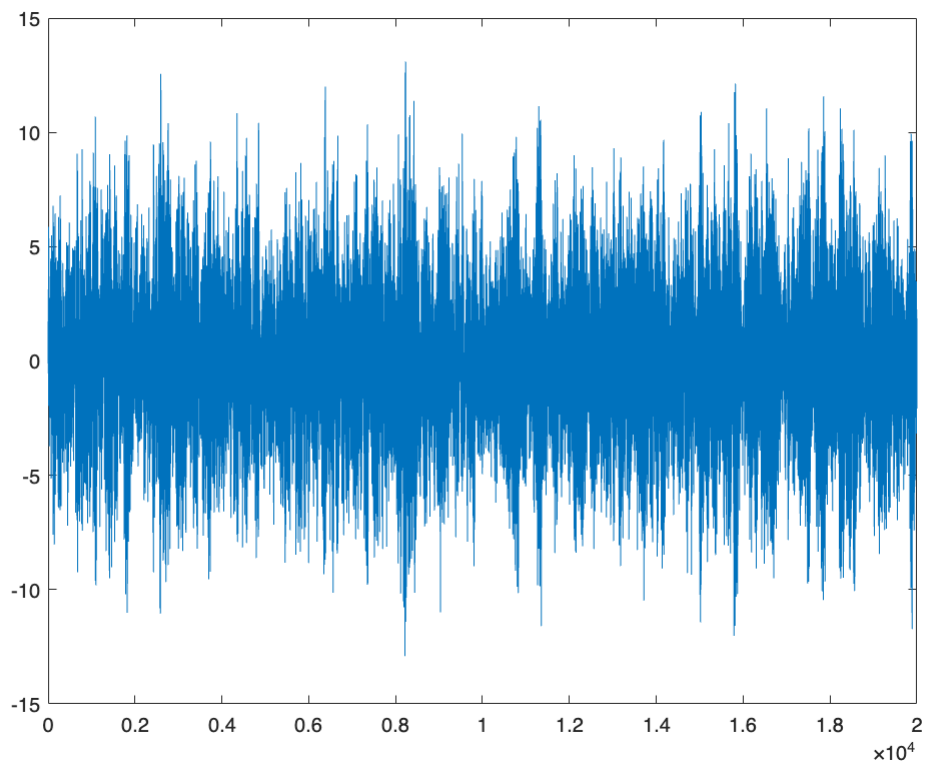
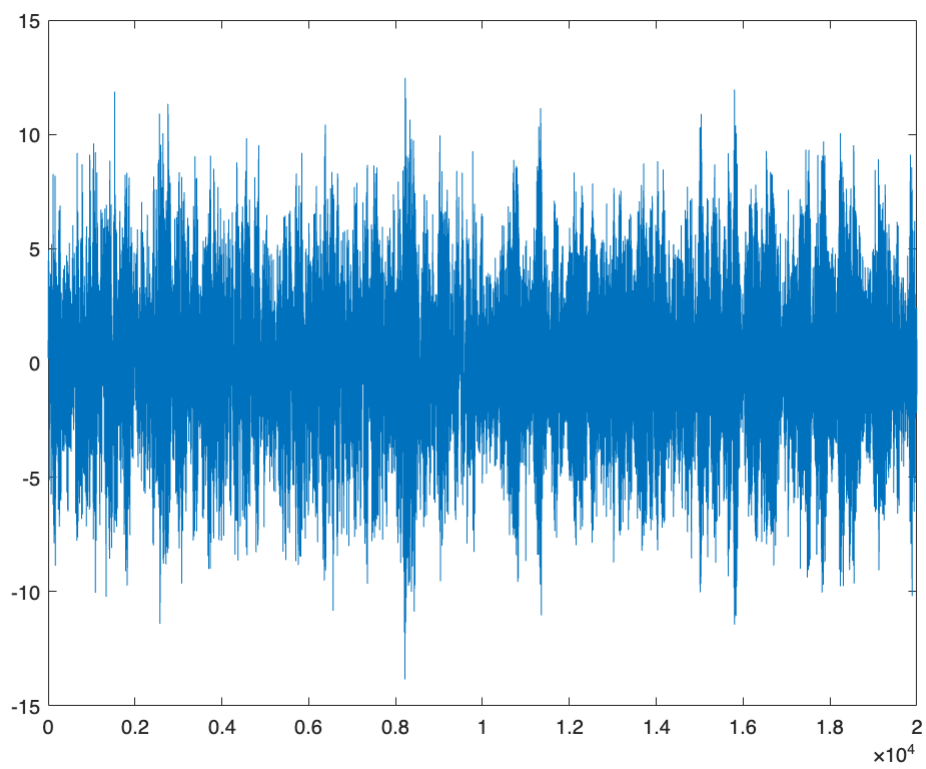
for p = [5,10]
    for N = 1:3
        [encoded, centers, a, y] = idpcmenco(t, p, N, min_value,
max_value);
        reconstructed = idpcm_deco(encoded, a, centers);

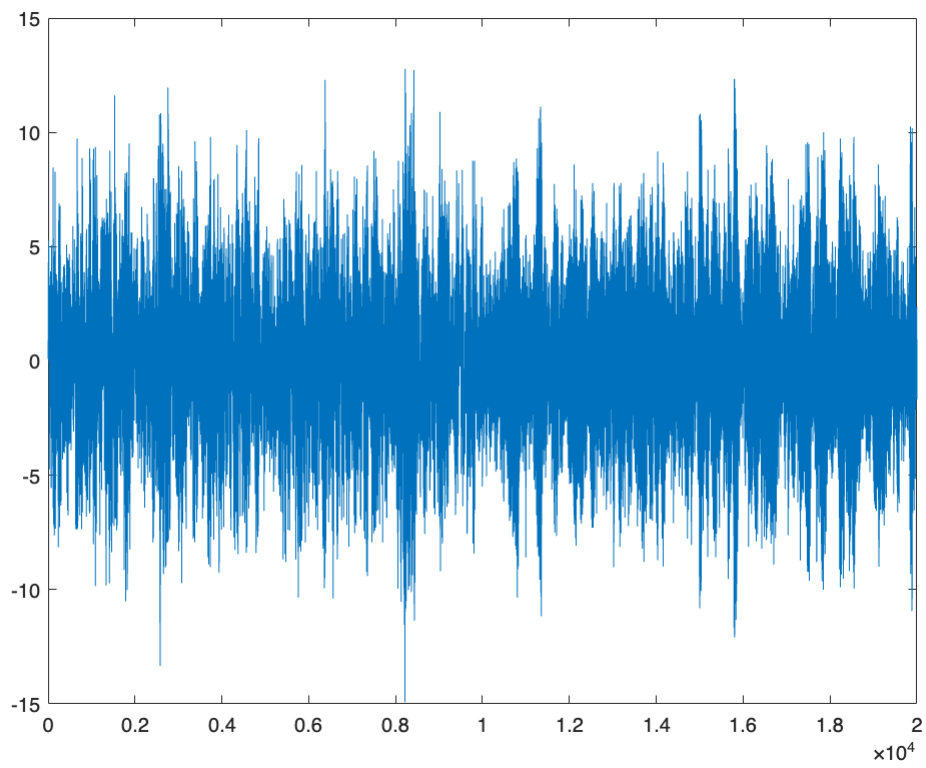
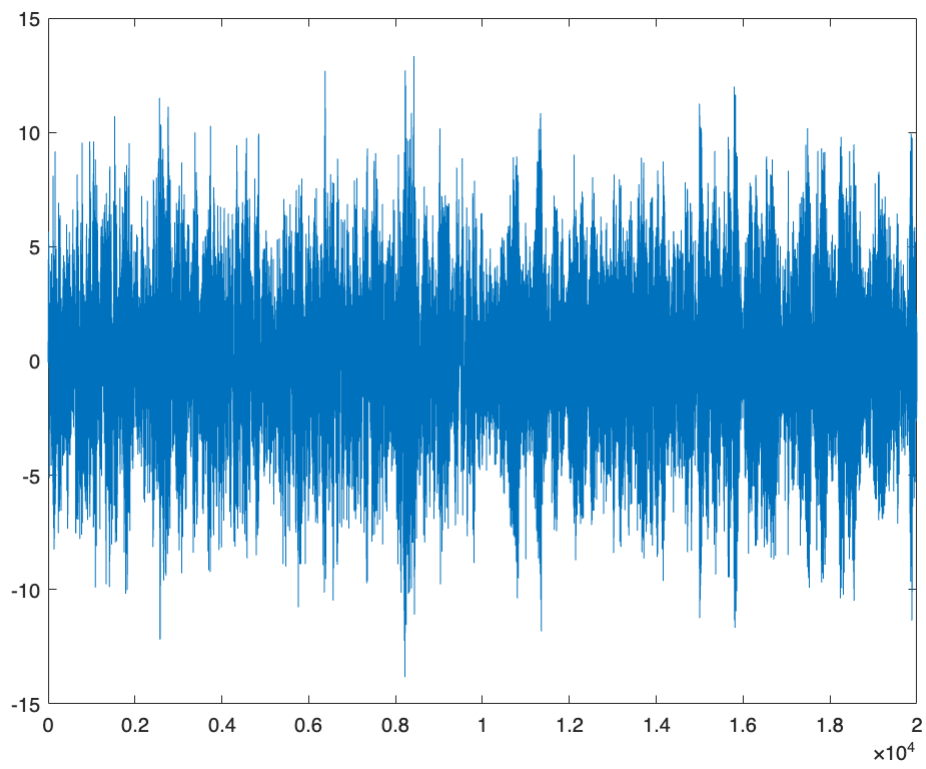
        figure;
        plot(reconstructed);
    end
end

```

end

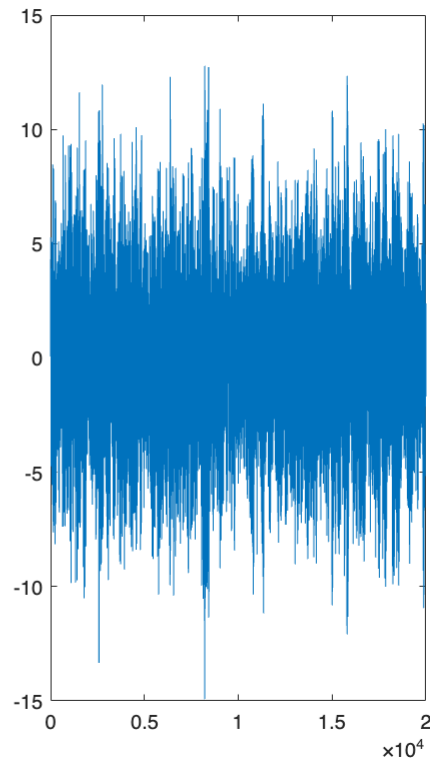
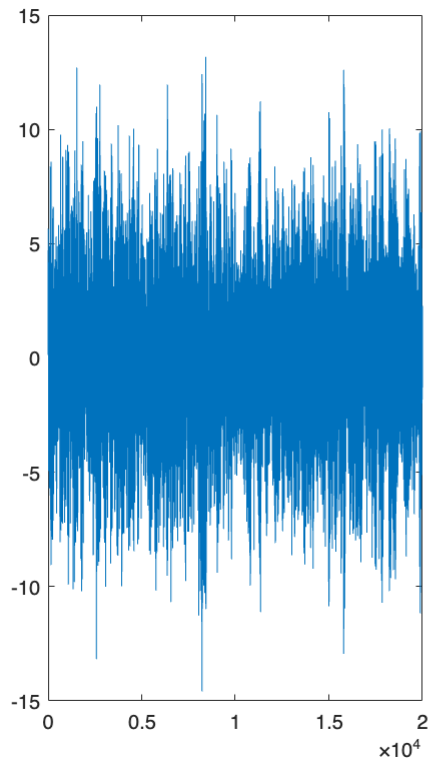






```
figure;  
subplot(1,2,1);  
plot(t);
```

```
subplot(1,2,2);  
plot(reconstructed);
```



Τέλος Άσκησης

Μόλις τελειώσετε το παιχνίδι, μην ξεχάσετε να εκτελέσετε τα ακόλουθα:

```
rmpath './functions'  
rmpath './sources'
```