Εργασία 1

Ψηφιακές Τηλεπικοινωνίες .

Όνομα/επώνυμο : Γρηγόρης Τζωρτζάκης Αμ:1084538

Περιεχόμενα

Μέρος Α΄

Θεωρία Πληροφορίας

Ερωτήσεις :

Ερώτημα 1.

α.

Αρχική ανάλυση του προβλήματος :

Κάθε οθόνη έχει pixel , τα οποία έχουν μέσα τρανζίστορ με λαμπάκια 3 χρωμάτων (κόκκινο ,πράσινο ,μπλε) και έτσι βλέπουμε τα χρώματα. Όμως , πως λέμε στο κάθε pixel τι χρώμα να βγάλει ? Αυτό το πετυχαίνουμε με τα bit. Όπως έχουμε μάθει , τα bit μπορούν να συμβολίσουν 2^N καταστάσεις , οπού N είναι ο αριθμός των δυαδικών ψηφίων. Αρα ,η λύση είναι να κωδικοποιούμε το κάθε χρώμα με την βοήθεια τους . Για παράδειγμα , μπορούμε να πούμε μαύρο=0 και άσπρο=1. Συνεπώς , ο υπολογιστής διαβάζει τις δυαδικές τιμές , καταλαβαίνει το χρώμα χάρη στην κωδικοποίηση που έχει γίνει , το μεταφέρει στα pixel και έτσι τελικά βλέπω την εικόνα.

Τότε , για να βρω τις τιμές που λαμβάνουν τα pixel , θα τα εξετάσω όλα και θα συγκρατήσω όλες τις διαφορετικές τιμές από μια φορά την καθεμιά σε μια λίστα.

Διακριτά σύμβολα της πηγής :

Στην εικόνα που μας έχει δοθεί πραγματοποιώ την εντολή

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

Και παρατηρώ ότι δημιουργείτε στον χώρο εργασίας

Το uint8 σημαίνει unsigned integer 8 bit , άρα κάθε pixel αποτελείτε από 8 bit. Συνεπώς ,καθένα από αυτά μπορεί να λάβει 256 διαφορετικές τιμές . Κανονικά, το κάθε χρώμα rgb περιγράφεται από 8 bit~~.~~ , με αποτέλεσμα να έχω συνολικά 24 bit σε κάθε pixel. Όμως , εφόσον η εικόνα είναι ασπρόμαυρη υπάρχει η σύμβαση πως και τα 3 χρώματα λαμβάνουν τις ιδίες τιμές . Τότε 8 είναι αρκετά για εικόνες χωρίς χρώμα .

Αρα , θεωρητικά θα έχω 256 διακριτές τιμές . Αυτό δεν σημαίνει ότι θα χρησιμοποιούμε όλες τις διαθέσιμες τιμές , μερικές από αυτές είναι αρκετές για μια εικόνα .

Ας βρούμε λοιπόν ποιες χρησιμοποιούνται για τον σχηματισμό της εικόνας.

Χρησιμοποιούμε την συνάρτηση

```
simvola_pigis = unique(I(:));
```

για να δούμε πραγματικά ποιες τιμές έχουν λάβει τα pixel.

Δημιουργείται αυτό

simvola_pigis 16x1 uint8

και έτσι ξέρουμε ότι υπάρχουν τελικά 16 διακριτές τιμές που λαμβάνουν τα pixel.

Για να εμφανίσω ακριβώς τις τιμές χρησιμοποιώ την εντολή

```
disp(simvola_pigis);
```

και έχω

```
    0
   17
   34
   51
   68
   85
  102
  119
  136
  153
  170
  187
  204
  221
  238
  255
```

Πιθανότητα εμφάνισης :

Αφού εντοπίσαμε τα σύμβολα της πηγής , τώρα πρέπει να βρούμε τις πιθανότητες εμφάνισης τους , δηλαδή ποιες τιμές λαμβάνουν τα pixel περισσότερο και ποιες λιγότερο. Για να τις υπολογίσουμε πολύ απλά θα πούμε: αριθμός pixel με συγκεκριμένη τιμή/ αριθμός όλων των pixel της εικόνας. Αρα τελικά έχουμε την εντολή

```
pithanotites = histcounts(I, numel(simvola_pigis)) / numel(I);
```

Το histcounts καταγράφει πόσα πιξελ έχουν την κάθε διακριτή τιμή και το numel(I) μετράει συνολικά πόσα έχει όλη η εικόνα . Τότε με την διαίρεση βρίσκουμε την πιθανότητα εμφάνισης του κάθε συμβόλου .

Αποτέλεσμα εκτέλεσης εντολής :

```
pithanotites = histcounts(I, numel(simvola_pigis)) / numel(I);
disp(pithanotites);
0.0962    0.0814    0.0683    0.0625    0.0768    0.0905    0.1132    0.0906    0.0965    0.0671    0.0389    0.0329    0.0337    0.0259    0.0224    0.0031
```

Τέλος , παρατηρούμε ότι εάν προσθέσουμε όλες τις πιθανότητες μαζί βγάζουν αποτέλεσμα 1 , συνεπώς πράγματι η πηγή λαμβάνει 16 διακριτές τιμές.

b.

Ανάλυση του προβλήματος :

Η κωδικοποίηση Huffman δημιουργεί βέλτιστους προθεματικούς κώδικες. Για την υλοποίηση του , χρειαζόμαστε τις πιθανότητες εμφάνισης του κάθε συμβόλου που βρήκαμε στο προηγούμενο ερώτημα. Επίσης , η εντροπία της πηγής πρέπει να υπολογιστεί επειδή ορίζει το όριο της βέλτιστης συμπίεσης , ώστε τελικά να συγκρίνουμε το μήκος κώδικα με αυτή και να δούμε αν η κωδικοποίηση μας είναι αποδοτική.

Επίλυση :

i) Θα χρησιμοποιήσουμε τον τύπο $\Sigma = p*\log_2(1/p)$.

Ζητείτε η εντροπία κωδικοποίησης , άρα θα βρω της πηγής. Αφού οι πιθανότητες δεν αλλάζουν μετα από την κωδικοποίηση ούτε μειώνεται ο αριθμός διακριτών καταστάσεων τότε είναι σωστό να πούμε ΕΝΤΡΟΠΙΑ Πηγής=ΕΝΤΡΟΠΙΑ Κωδικοποίησης.

Γράφουμε την εντολή

```
entropia_pigis = sum(pithanotites .* log2(1 ./ pithanotites));
```

και έχουμε H(x)= 3.7831

Με γρήγορους υπολογισμούς επαληθεύω το αποτέλεσμα

```
0.0962 * log(10.3950103950104) = 0.0962 * 3.378 = 0.3249636
0.0814 * log(12.28501228501229)= 0.0814 * 3.619 = 0.2945866
0.0683 * log(14.64128843338214)= 0.0683 * 3.872 = 0.2644576
0.0625 * log(16)= 0.0625 * 4 = 0.25
0.0768 * log(13.020833333333330)= 0.0768 * 3.703 = 0.2843904
0.0905 * log(11.04972375690608)= 0.0905 * 3.466 = 0.313673
0.1132 * log(8.833922261484099)= 0.1132 * 3.143 = 0.3557876
0.0906 * log(11.03752759381898)= 0.0906 * 3.464 = 0.3138384
0.0965 * log(10.36269430051813)= 0.0965 * 3.373 = 0.3254945
0.0671 * log(14.90312965722802)= 0.0671 * 3.8975 = 0.26152225
0.0389 * log(25.70694087403599)= 0.0389 * 4.684 = 0.1822076
0.0329 * log(30.3951367781155)=  0.0329 * 4.926 = 0.1620654
0.0337 * log(29.67359050445104)= 0.0337 * 4.891 = 0.1648267
0.0259 * log(38.61003861003861)= 0.0259 * 5.271 = 0.1365189
0.0224 * log(44.64285714285714)= 0.0224 * 5.48 = 0.122752
0.0031 * log(322.5806451612903)= 0.0031 * 8.334 = 0.0258354


0.3249636+0.2945866+0.2644576+0.25+0.2843904+0.313673+0.3557876+0.3138384+0.3254945+0.26152225+0.1822076+0.1620654+0.1648267+0.1365189+0.122752+0.0258354
=
3.78291995
```

ii) Για να υπολογίσω το μήκος κώδικα πρώτα θα κατασκευάσω το δέντρο Huffman .

Πρέπει να κατασκευαστεί ώστε να γίνει ανάθεση κωδικών στα σύμβολα της πηγής . Έπειτα , σύμφωνα με αυτά που βρήκαμε , γίνεται η κωδικοποίηση όλων των πιξελ της εικόνας . Θα έχουμε ως αποτέλεσμα ένα μεγάλο string 0 και 1 . Η κωδικοποίηση (huffmanenco) δεν είναι απαραίτητη για την εύρεση του μήκους αλλά θα χρειαστεί σε επόμενο ερώτημα.

Ας δημιουργήσουμε πρώτα το δέντρο :

```
dedro_huffman= huffmandict(simvola_pigis, pithanotites);
```

```
disp(dedro_huffman);
 {[  0]}    {[        1 1 0]}
 {[ 17]}    {[      0 0 1 0]}
 {[ 34]}    {[      0 1 0 0]}
 {[ 51]}    {[      0 1 1 1]}
 {[ 68]}    {[      0 0 1 1]}
 {[ 85]}    {[      0 0 0 0]}
 {[102]}    {[        1 0 0]}
 {[119]}    {[        1 1 1]}
 {[136]}    {[        1 0 1]}
 {[153]}    {[      0 1 0 1]}
 {[170]}    {[    0 0 0 1 1]}
 {[187]}    {[    0 1 1 0 1]}
 {[204]}    {[    0 1 1 0 0]}
 {[221]}    {[  0 0 0 1 0 0]}
 {[238]}    {[0 0 0 1 0 1 0]}
 {[255]}    {[0 0 0 1 0 1 1]}
```

Και αμέσως μετα η κωδικοποίηση :

```
kodikopoihsh = huffmanenco(I(:), dedro_huffman);
```

όπως είπαμε , χάρη στην λίστα κωδικών που βρήκαμε παραπάνω , αναθέτει τους κωδικούς σε όλα τα πιξελ της εικόνας.

```
-
1
0
0
0
1
0
0
0
1
0
0
0
1
0
0
0
1
0
0
0
1
0
0
1
0
```

Παρατηρούμε πως το αποτέλεσμα είναι μια τεράστια γραμμή από 0 και 1 , αφού η εικόνα έχει πολλά πιξελ θα χρειαστεί τεράστιος αριθμός bit για την κωδικοποίηση της.

Συνεχίζοντας , ας υπολογίσουμε το μήκος του κώδικα . Αυτό θα γίνει με τον τύπο Σ (p*L) .

```
mhkos_huffman = sum(pithanotites .* arrayfun(@(x) length(dedro_huffman{x, 2}), 1:size(dedro_huffman, 1)));
```

Με αυτόν τον κώδικα , δημιούργησα πρώτα μια λίστα που έχει μέσα το δέντρο Huffman. Το arrayfun εξετάζει κάθε νούμερο της λίστας για να δει τελικά τι μήκος έχει το κάθε στοιχείο. Στην συνέχεια πολλ/σιαζει το μήκος του στοιχείου με την πιθανότητα εμφάνισης του και στο τέλος προσθέτει όλα τα αποτελέσματα.

Αρα το μήκος είναι

```
disp(mhkos_huffman);
    3.8374
```

Με την χρήση calculator επιβεβαίωσα τα αποτελέσματα.

Calculation precision
Digits after the decimal point: 5

Weighted path length
3.83730

Shannon entropy
3.78294

Huffman coding

Symbol	Encoding
102	011
136	010
0	001
119	000
85	1111
17	1101
68	1100
34	1011
153	1010
51	1000
170	11100
204	10011
187	10010
221	111011
238	1110101
255	1110100

iii)

Για να βρω την αποδοτικότητα αρκεί να κάνω Εντροπία/μήκος κώδικα .

```
>> apodotikothta= entropia_pigis / mhkos_huffman

apodotikothta =

    0.9859
```

Τότε η αποδοτικότητα είναι η=98.59%.

Ερώτημα 2 .

a. Για την επέκταση δεύτερης τάξης επέκτασης πηγής εξετάζουμε τα σύμβολα ως ζευγάρια και όχι μεμονωμένα , παρατηρώντας τις εξαρτήσεις και σχέσεις των συμβόλων.

Όπως και στο προηγούμενο ερώτημα , πρέπει να εντοπίσουμε όλα τα ζευγάρια που υπάρχουν μέσα στην εικόνα. Αρα θα κοιτάξουμε τον γείτονα του κάθε πιξελ για να δούμε ποια ζευγάρια υπάρχουν.

Εντοπισμός διακριτών συμβόλων :

Θα κοιτάξουμε για οριζόντια και κάθετα ζευγάρια (όλα τα γειτονικά pixel).

Θα χρειαστούν 2 μεταβλητές που θα αποθηκεύουν όλα τα ζευγάρια καθώς και κώδικας για να τα δημιουργεί .

Οι μεταβλητές :

```
orizodia_zeugaria = containers.Map('KeyType', 'char', 'ValueType',
'double');

katheta_zeugaria = containers.Map('KeyType', 'char', 'ValueType',
'double');
```

Ο τρόπος δημιουργίας ζευγαριών :

```
for i = 1:row
    for j = 1:(col - 1)
orizodio_zeugari = [image(i, j), image(i, j + 1)];
```

όπου με το loop παίρνει το pixel που βρίσκεται ο μετρητής και το ζευγαρώνει με το αμέσως επόμενο στην σειρά. Με τον ίδιο τρόπο κάνουμε και το κάθετο ζευγάρι

```
for i = 1:(row - 1)
    for j = 1:col
katheto_zeugari= [image(i, j), image(i + 1, j)];
```

Ας προχωρήσουμε στην εύρεση πιθανοτήτων . Όπως κάναμε και στο ερώτημα 1 , θα διαιρέσουμε το κάθε οριζόντιο ζευγάρι προς όλα τα οριζόντια και αντίστοιχα το κάθε κάθετο προς όλα .

```
orizodies_pithanotites = zeros(1, length(horizontal_keys));
for i = 1:length(horizontal_keys)
    arithos_emfaniseon = orizodia_zeugaria(horizontal_keys{i});
    pithanotita = arithos_emfaniseon / sinolika_orizodia_zeugaria;
    orizodies_pithanotites(i) = pithanotita;
```

Αρχικά δημιουργούμε το array orizodies_pithanotites στο οποίο θα αποθηκεύετε η πιθανότητα του κάθε ζευγαριού. Συνεχίζοντας , το loop μετράει κάθε ζευγάρι πόσες φορές εμφανίζεται στην εικόνα και αποθηκεύει το αποτέλεσμα στην μεταβλητή arithmos_emfaniseon. Τέλος υπολογίζουμε την πιθανότητα όπως έχουμε αναφέρει παραπάνω και αποθηκεύουμε το αποτέλεσμα στο array. Με τον ίδιο τρόπο βρίσκουμε και τις κατακόρυφες πιθανότητες.

Τα αποτελέσματα που λαμβάνουμε :

```
orizodies pithanotites:
0  0: 0.078926
0  119: 3.3557e-05
0  136: 6.7114e-05
0  17: 0.013255
0  34: 0.0020134
0  51: 0.00057047
0  68: 0.00040268
0  85: 0.00026846
102   17: 0.00030201
102   34: 0.00067114
102   51: 0.0018121
102   68: 0.0058725
102   85: 0.018725
102  102: 0.059832
102  119: 0.018926
102  136: 0.004698
102  153: 0.0018121
102  170: 0.0004698
102  187: 0.00020134
102  204: 0.00010067
119   17: 0.00020134
119   34: 0.00040268
119   51: 0.00097315
119   68: 0.0023154
119   85: 0.005302
119  102: 0.016913
119  119: 0.041309
119  136: 0.017416
119  153: 0.0042953
119  170: 0.0010067
119  187: 0.00050336
119  204: 6.7114e-05
119  221: 0.00013423
119  238: 3.3557e-05
119  255: 3.3557e-05
136   17: 6.7114e-05
136   34: 0.00016779
136   51: 0.00063758
136   68: 0.00083893
136   85: 0.0023826
136  102: 0.0060738
136  119: 0.014698
136  136: 0.050805
136  153: 0.017215
136  170: 0.0024161
136  187: 0.0010738
136  204: 0.00030201
136  221: 0.00016779
136  255: 3.3557e-05
153   17: 3.3557e-05
153   34: 0.00010067
153   51: 0.00033557
153   68: 0.00073826
153   85: 0.0007047
153  102: 0.001745
153  119: 0.0051007
153  136: 0.015705
153  153: 0.029966
153  170: 0.010034
153  187: 0.0019463
153  204: 0.00067114
153  221: 0.00020134
17   0: 0.014396
```

```
17  102: 0.00013423
17  119: 0.00010067
17  136: 0.00010067
17  153: 3.3557e-05
17  17: 0.046846
17  170: 6.7114e-05
17  34: 0.014564
17  51: 0.0029195
17  68: 0.0014765
17  85: 0.00067114
170   17: 3.3557e-05
170   51: 0.00016779
170   68: 0.0002349
170   85: 0.00053691
170  102: 0.00067114
170  119: 0.0012752
170  136: 0.0030872
170  153: 0.009094
170  170: 0.015168
170  187: 0.0069799
170  204: 0.0013423
170  221: 0.00033557
170  238: 0.00010067
187   34: 3.3557e-05
187   51: 3.3557e-05
187   68: 6.7114e-05
187   85: 0.00016779
187  102: 0.00040268
187  119: 0.00040268
187  136: 0.0009396
187  153: 0.0018456
187  170: 0.0068792
187  187: 0.014463
187  204: 0.0068121
187  221: 0.00087248
187  238: 0.00010067
204   68: 3.3557e-05
204   85: 0.00013423
204  102: 6.7114e-05
204  119: 0.00020134
204  136: 3.3557e-05
204  153: 0.00073826
204  170: 0.0017785
204  187: 0.0060738
204  204: 0.017685
204  221: 0.0063758
204  238: 0.00050336
204  255: 0.00010067
221   68: 3.3557e-05
221   85: 6.7114e-05
221  119: 0.00013423
221  136: 0.00010067
221  153: 0.00036913
221  170: 0.00030201
221  187: 0.00090604
221  204: 0.0056376
221  221: 0.013926
221  238: 0.0044295
221  255: 0.00013423
238  102: 3.3557e-05
238  119: 6.7114e-05
238  136: 3.3557e-05
238  153: 0.00010067
238  170: 0.00020134
238  187: 0.00030201
238  204: 0.00077181
```

```
238  221: 0.0037919
238  238: 0.015839
238  255: 0.0013758
255  170: 3.3557e-05
255  187: 6.7114e-05
255  221: 0.00016779
255  238: 0.0014094
255  255: 0.0014765
34   0: 0.0013423
34  102: 0.0007047
34  119: 0.00053691
34  136: 0.00020134
34  153: 0.00016779
34  17: 0.015369
34  187: 0.00016779
34  204: 3.3557e-05
34  34: 0.032248
34  51: 0.011913
34  68: 0.0037248
34  85: 0.0015101
51   0: 0.00033557
51  102: 0.0016443
51  119: 0.00050336
51  136: 0.00053691
51  153: 0.00026846
51  17: 0.0035906
51  170: 0.00010067
51  187: 3.3557e-05
51  204: 3.3557e-05
51  221: 3.3557e-05
51  34: 0.013792
51  51: 0.024933
51  68: 0.012315
51  85: 0.0039597
68   0: 6.7114e-05
68  102: 0.005604
68  119: 0.0022148
68  136: 0.00057047
68  153: 0.00036913
68  17: 0.0014094
68  170: 0.0002349
68  187: 3.3557e-05
68  204: 3.3557e-05
68  238: 3.3557e-05
68  34: 0.0033221
68  51: 0.01349
68  68: 0.034933
68  85: 0.014362
85  102: 0.019597
85  119: 0.005302
85  136: 0.0020134
85  153: 0.00057047
85  17: 0.00040268
85  170: 0.00040268
85  187: 3.3557e-05
85  204: 0.00016779
85  221: 3.3557e-05
85  238: 3.3557e-05
85  34: 0.0011409
85  51: 0.004698
85  68: 0.014128
85  85: 0.04198
```

```
kathetes pithanotites:
0  0: 0.081139
0  102: 0.0001005
0  17: 0.012831
0  34: 0.0012395
0  51: 0.00040201
0  68: 0.00026801
0  85: 3.3501e-05
102   17: 0.000134
102   34: 0.00036851
102   51: 0.0015745
102   68: 0.0036181
102   85: 0.016549
102  102: 0.070117
102  119: 0.015645
102  136: 0.0026466
102  153: 0.001072
102  170: 0.00056951
102  187: 0.00063652
102  204: 0.0001675
102  221: 0.0001005
102  238: 0.0001005
102  255: 3.3501e-05
119   34: 0.000134
119   51: 0.00043551
119   68: 0.0017755
119   85: 0.0047236
119  102: 0.015779
119  119: 0.050151
119  136: 0.013367
119  153: 0.0024456
119  170: 0.00063652
119  187: 0.00056951
119  204: 0.00020101
119  221: 0.0001005
119  238: 0.000134
119  255: 3.3501e-05
136   17: 3.3501e-05
136   34: 3.3501e-05
136   51: 0.000134
136   68: 0.00036851
136   85: 0.001541
136  102: 0.0047236
136  119: 0.015578
136  136: 0.056348
136  153: 0.013568
136  170: 0.0020101
136  187: 0.001005
136  204: 0.00030151
136  221: 0.0001005
136  238: 0.000134
136  255: 0.0001675
153   34: 6.7002e-05
153   51: 3.3501e-05
153   68: 0.000134
153   85: 0.00036851
153  102: 0.00063652
153  119: 0.0035176
153  136: 0.018392
153  153: 0.034405
153  170: 0.0068342
153  187: 0.001474
153  204: 0.00093802
153  221: 0.00023451
153  238: 0.0001675
153  255: 6.7002e-05
```

```
17   0: 0.012261
17  102: 0.00043551
17  119: 0.00020101
17  136: 0.00026801
17  153: 0.0001005
17  17: 0.050452
17  170: 6.7002e-05
17  187: 3.3501e-05
17  34: 0.013333
17  51: 0.0025461
17  68: 0.001273
17  85: 0.00053601
170   34: 3.3501e-05
170   51: 3.3501e-05
170   68: 0.0001005
170   85: 0.00033501
170  102: 0.000134
170  119: 0.00067002
170  136: 0.0022111
170  153: 0.010519
170  170: 0.016918
170  187: 0.0060302
170  204: 0.0013735
170  221: 0.00040201
170  238: 0.0001675
170  255: 0.0001005
187   85: 3.3501e-05
187  102: 6.7002e-05
187  119: 0.00023451
187  136: 0.00060302
187  153: 0.0024121
187  170: 0.0079732
187  187: 0.013635
187  204: 0.0059296
187  221: 0.001474
187  238: 0.00050251
187  255: 0.00020101
204  102: 6.7002e-05
204  119: 3.3501e-05
204  136: 0.00043551
204  153: 0.001072
204  170: 0.0022446
204  187: 0.0073702
204  204: 0.01675
204  221: 0.0046566
204  238: 0.001005
204  255: 0.00020101
221   85: 6.7002e-05
221  119: 3.3501e-05
221  136: 6.7002e-05
221  153: 0.000134
221  170: 0.00067002
221  187: 0.0013735
221  204: 0.0068342
221  221: 0.012898
221  238: 0.0036516
221  255: 0.00030151
238   85: 3.3501e-05
238  119: 3.3501e-05
238  153: 0.000134
238  170: 6.7002e-05
238  187: 0.00026801
238  204: 0.00087102
238  221: 0.0055946
238  238: 0.014171
238  255: 0.0013065
```

```
255  153: 3.3501e-05
255  170: 6.7002e-05
255  221: 0.0001005
255  238: 0.0022111
255  255: 0.00073702
34   0: 0.001005
34  102: 0.001005
34  119: 0.00040201
34  136: 0.00036851
34  153: 0.00033501
34  17: 0.012998
34  170: 0.00020101
34  187: 0.0001675
34  204: 0.000134
34  34: 0.036147
34  51: 0.01196
34  68: 0.0025796
34  85: 0.0011725
51   0: 0.00020101
51  102: 0.001407
51  119: 0.00073702
51  136: 0.00020101
51  153: 0.00033501
51  17: 0.0029816
51  170: 0.00023451
51  187: 6.7002e-05
51  204: 0.0001005
51  221: 0.0001005
51  238: 3.3501e-05
51  34: 0.012228
51  51: 0.028978
51  68: 0.012127
51  85: 0.0023451
68  102: 0.0025796
68  119: 0.001407
68  136: 0.00073702
68  153: 0.00050251
68  17: 0.00093802
68  170: 0.00020101
68  187: 0.00023451
68  204: 0.000134
68  221: 0.0001675
68  238: 0.0001005
68  34: 0.0024791
68  51: 0.011926
68  68: 0.041876
68  85: 0.013568
85  102: 0.016583
85  119: 0.0024121
85  136: 0.00134
85  153: 0.00040201
85  17: 0.00030151
85  170: 0.00040201
85  187: 0.00023451
85  204: 0.0001005
85  221: 0.0001005
85  238: 0.0001005
85  34: 0.0016415
85  51: 0.0041876
85  68: 0.013032
85  85: 0.049514
```

b.

i)

Έχοντας υπολογίσει τις πιθανότητες , είμαστε σε θέση να βρούμε την εντροπία της κωδικοποίησης. Το μόνο που χρειάζεται να κάνουμε είναι κάθε στοιχείο μέσα στο loop αφού βρεθεί η πιθανότητα του να υπολογίζετε από κάτω η εντροπία του . Αρα θα ορίσουμε απλά μια μεταβλητή που θα υπολογίζει την εντροπία για κάθε ζευγάρι .

```
orizodia_edropia = 0;
```

Έξω από το loop την αρχικοποιουμε με αρχική τιμή 0 για να βγει το σωστό αποτέλεσμα.

```
orizodia_edropia = orizodia_edropia + pithanotita *
log2(1/pithanotita);
```

Μέσα στο λοοπ σε κάθε επανάληψη υπολογίζεται η εντροπία του ζευγαριού και κάθε φορά προσθέτουμε την προηγούμενη τιμή στην τωρινή ώστε να λάβουμε το συνολικό αποτέλεσμα .Αντίστοιχα υπολογίζουμε και την κάθετη εντροπία. Το αποτέλεσμα είναι

```
orizodia edropia: 5.7878 katheti edropia: 5.627
```

Άρα η συνολική εντροπία είναι (5.7878+5.627)/2= 5.7074 .

ii) Θα χρειαστεί να βρούμε το δέντρο Huffman με την χρήση της εντολής huffmandict ώστε να υπολογιστεί το μήκος κώδικα .

```
[orizodia_kodikopoihsh, ~] = huffmandict(keys(orizodia_zeugaria),
orizodies_pithanotites);
```

Στην συνέχεια , όπως κάναμε για τον υπολογισμό της εντροπίας θα κάνουμε ένα loop που υπολογίζει για κάθε ζευγάρι το μήκος του ώστε να βρούμε το συνολικό μήκος.

```
oriz_mhkoskodika = 0;
for i = 1:length(keys(orizodia_zeugaria))
    kodikas =
orizodia_kodikopoihsh{find(strcmp(keys(orizodia_zeugaria){i},
orizodia_kodikopoihsh(:,1))), 2};
    mhkos = length(kodikas);
    oriz_mhkoskodika = oriz_mhkoskodika + orizodies_pithanotites(i) *
mhkos;
end
```

Ανακτάμε τον κώδικα του κάθε ζευγαριού με την εντολή kodikas=orizodia_kodikopoihsh , βρίσκουμε τον αριθμό των bit του κώδικα με την εντολή mhkos= length(kodikas) και τέλος υπολογίζουμε το μέσο μήκος με τον τύπο p*l. Το αποτέλεσμα είναι

```
meso mhkos oriz: 5.8126 bits
meso mhkos kath: 5.6549 bits
```

Άρα το μέσο μήκος είναι (5.8126+5.6549) / 2 = 5.73375 bits/ ζευγάρι .

5.73375/2=2.866875 bit/σύμβολο.

iii) Όπως στο ερώτημα 1 , θα κάνουμε εντροπία/μήκος κώδικα για να βρούμε την αποδοτικότητα η.

```
apodotikothta_oriz = edropia_oriz / meso_mhkos_oriz;
disp(['apodotikothta oriz: ', num2str(apodotikothta_oriz * 100),
'%']);
apodotikothta_kath = edropia_kath / meso_mhkos_kath;
disp(['apodotikothta kath: ', num2str(apodotikothta_kath * 100),
'%']);
```

Και έχουμε ως αποτέλεσμα

```
apodotikothta oriz: 99.5732%
apodotikothta kath: 99.5066%
```

Συνεπώς η ολική αποδοτικότητα είναι (οριζόντια + κάθετα)/2 = 99.5399 %

c.

Παρατηρούμε ότι η αποδοτικότητά είναι καλύτερη με την επέκταση πηγής και μειώνεται το μέσο μήκος του κώδικα. Ο λόγος που συμβαίνει αυτό είναι πως η δημιουργία ζευγαριών αναδεικνύει τις εξαρτήσεις των pixel , με αποτέλεσμα να μειώνεται η αβεβαιότητα αφού το κάθε σύμβολο προσδίδει πληροφορία για το επόμενο. Συνεπώς , με κάθε επέκταση της πηγής θα πετυχαίνουμε μικρότερο μήκος κώδικα και καλύτερη αποδοτικότητά εφόσον υπάρχουν εξαρτήσεις μεταξύ των συμβολών.

Ερώτημα 3.

α. Ο τύπος $H(X^2)= 2H(x)$ υπονοεί ότι αφού έχουμε ζευγάρι pixel θα προσθέσουμε απλά τις 2 εντροπίες. Όμως , αυτό δεν ισχύει καθώς η εντροπία εξαρτάται από την πιθανότητα η οποία αλλάζει. Με την δημιουργία του ζευγαριού η πιθανότητα εμφάνισης είναι πλέον διαφορετική επειδή πρέπει να εντοπίσω και τα 2 pixel μαζί . Συνοψίζοντας , δεν γίνεται να υποθέσουμε ότι η εντροπία διπλασιάζεται αφού και η πιθανότητα αλλάζει λόγο ότι δεν έχω 2 ανεξάρτητα στοιχεία αλλά ένα ζεύγος .

b. Από θεωρία γνωρίζουμε ότι για τους προθεματικούς κώδικες (άρα και για τον Huffman) ισχύει το φράγμα $H(x) \leq L<H(x)+1$. Το κάτω φράγμα πρέπει να ισχύει αλλιώς η συμπίεση είναι με απώλειες και το άνω φράγμα ισχύει επειδή θα βρεθεί κώδικας με βέλτιστο μήκος. Μπορούμε να δούμε ότι ισχύει και για τα 2 ερωτήματα

$3.7831<3.8374<4.7831$

$5.7074<5.73375<6.7074$

Ερώτημα 4. Για να επιβεβαιώσουμε την ορθότητα της κωδικοποίησης , θα συγκρίνουμε την εικόνα που θα λάβουμε μετα από την επεξεργασία Huffman με την αρχική. Θα φροντίσουμε και οι 2 εικόνες να έχουν την ιδιά διάσταση πρώτου γίνει η σύγκριση.

```
apokodikopoihsh = huffmandeco(kodikopoihsh, huffmanDict);
apokodikopoihsh = reshape(apokodikopoihsh, size(I));
elenxos = isequal(I, apokodikopoihsh);
disp('Einai sosti i kodikopoihsh?');
disp(elenxos);
```

Η συνάρτηση isequal κάνει την σύγκριση των 2 εικόνων. Εκτελώντας τον κώδικα παίρνουμε

```
Einai sosti i kodikopoihsh?
   1
```

άρα η κωδικοποίηση είναι σωστή (1=true) , όπως ήταν αναμενόμενο , αφού η Huffman είναι μέθοδος χωρίς απώλειες.

Συνεχίζοντας , θα υπολογίσουμε τον αριθμό bit της αρχικής εικόνας και της κωδικοποίησης.

```
bits_diadikhs_anaparastashs = numel(I) * 8;
bits_huffman = length(kodikopoihsh);
```

Η συνάρτηση numel(I) βρίσκει τον αριθμό των pixel της εικόνας και πολλαπλασιάζεται με 8 επειδή όπως έχουμε αναφέρει κάθε πιξελ αναπαριστάτε από 8 bit.

Τέλος , υπολογίζουμε τον λόγο συμπίεσης

```
j = bits_huffman / bits_diadikhs_anaparastashs;
```

και το αποτέλεσμα είναι

```
j:
    0.4797
```

Ερώτημα 5. Για να υπολογίσουμε την πιθανότητα p , αρκεί να συγκρίνουμε την είσοδο της ακολουθίας x με την ακολουθία y ώστε να παρατηρήσουμε πόσα bit είναι αλλαγμένα και στην συνέχεια να κάνουμε την πράξη αριθμός σωστών μπιτ/ όλα τα μπιτ.

```
x = kodikopoihsh
y = binary_symmetric_channel(x);
sostabit = sum(x == y);
sinolikabit = length(x);
p = sostabit / sinolikabit;
```

Η σύγκριση γίνεται με x==y , δημιουργεί ένα array με 0 και 1 (false , truth) και το άθροισμα το κάνουμε για να δούμε πόσα 1 (σωστά) bit μεταφέρθηκαν στο κανάλι. Τελικά βρίσκουμε

```
p: 0.88
```

Παρακάτω , για τον υπολογισμό τη χωρητικότητας του καναλιού χρησιμοποιούμε τους τύπους $C=1-H(p)$, $H(p)= -p*\log_2(p) – (1-p)* \log_2(1-p)$ και βρίσκουμε

`Xorithkothta kanaliou: 0.47075` bits/sec.

Η αμοιβαία πληροφορία θα βρεθεί με τον τύπο $I(X|Y)= H(X) – H(p)$. Είναι αναγκαίο να υπολογιστεί η κατανομή της πιθανότητας εισόδου , επειδή η τιμή της H(X) επηρεάζετε από αυτή . Ας υπολογίσουμε λοιπόν τις πιθανότητες με βάση την κωδικοποιημένη ακολουθία του ερωτήματος 4.

Για να βρούμε τις πιθανότητες των συμβολών αρκεί να διαιρέσουμε τον αριθμό εμφάνισης τους με όλα τα bit. Στην συνέχεια απλά θα βάλουμε αυτές τις τιμές στον τύπο της H(X).

```
arithmos_0 = sum(kodikopoihsh == '0');
arithmos_1 = sum(kodikopoihsh == '1');
olatabit = length(kodikopoihsh);

pithanotita_0 = arithmos_0 / olatabit;
pithanotita_1 = arithmos_1 / olatabit;
H_X = - (pithanotita_0 * log2(pithanotita_0) + pithanotita_1 *
log2(pithanotita_1));


amoivaia_pliroforia = H_X - H_p;

disp(['Amoivaia Pliroforia: ', num2str(amoivaia_pliroforia)]);
```

Άρα βρίσκουμε `Amoivaia Pliroforia: 0.46296` bits.

Μέρος β

Κωδικοποίηση Διακριτής Πηγής με τη μέθοδο DPCM

1. Καλούμαστε να υλοποιήσουμε το σύστημα DPCM. Αρχικά , θα πρέπει να προσδιορίζουμε τις τιμές που θα χρησιμοποιήσουμε , όπως την τάξη του προβλεπτή (πόσα δείγματα θα έχει για την προβλεψη) και τα επίπεδα του κβαντιστη. Για την αρχική υλοποίηση χρησιμοποιώ p=3 και N=10 καθώς και την δυναμική περιοχή που δίνεται από την άσκηση (-3.5,3.5).

```
function main()

close all;

% posa proigoumena deigmata tha exoume
p = 10;
% ta bit kvadisti
N = 3;
% dinamiki perioxi tou kvadisti
min_value = -3.5;
max_value = 3.5;
load('source.mat')
```

Συνεχίζοντας , πριν κάνουμε την κωδικοποίηση , είναι αναγκαίο να λούσουμε την εξίσωση yule walker ώστε να υπολογιστεί το δείγμα πρόβλεψης .

```
R = zeros(p,p);
r = zeros(p,1);
```

Αρχικοποιουμε τις δυο μεταβλητές r,R (οι οποίες μάλιστα είναι πίνακες) και προχωράμε στον υπολογισμό τους

```
for i=1:p
    sum_r = 0;
    for n=p+1:sinolikos_arithmos_deigmaton
        sum_r = sum_r + x(n)*x(n-i);
    end
    r(i) = sum_r * 1/(sinolikos_arithmos_deigmaton-p);
    for j=1:p
        sum_R = 0;
        for n=p+1:sinolikos_arithmos_deigmaton
            sum_R = sum_R + x(n-j)*x(n-i);
```

```
        end
        R(i,j) = sum_R * 1/(sinolikos_arithmos_deigmaton-p);
```

Τέλος , αρκεί να λούσουμε την εξίσωση

```
a = R\r;
```

και υπολογίζουμε τις κβαντισμένες τιμές των συντελεστών με τις κατάλληλες τιμές κβαντιστη

```
for i=1:p
    a(i) = my_quantizer(a(i), 8, -2, 2);
```

Προχωρώντας , είμαστε πλέον σε θέση να πραγματοποιήσουμε την κωδικοποίηση . Η πρόβλεψη υπολογίζεται με το a που βρήκαμε και τις τιμές των προηγούμενων κβαντισμένων δειγμάτων. Το σφάλμα πρόβλεψης είναι το τωρινό δείγμα – πρόβλεψη , . Τέλος , κβαντίζουμε το σφάλμα πρόβλεψης.

```
for i=1:sinolikos_arithmos_deigmaton
    y(i) = x(i) - y_meto_simvolo_apano_tonismeno;
    y_meto_simvolo_apano(i) =
my_quantizer(y(i),N,min_value,max_value);


 y_meto_simvolo_apano_tonismeno = y_meto_simvolo_apano(i) +
y_meto_simvolo_apano_tonismeno;


 proigoumena_deigmata = [y_meto_simvolo_apano_tonismeno;
proigoumena_deigmata(1:p-1)];


 y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
end
```

Το loop κάνει την παραπάνω διαδικασία και επιπροσθέτως προσαρμόζει την πρόβλεψη για το επόμενο δείγμα με βάση το σφάλμα κβαντισης του τωρινού δείγματος , βελτιώνοντας με αυτό τον τρόπο τον προβλεπτή.

Αμέσως μετα υλοποιούμε τον αποκωδικοποιητή. Με απλά λόγια , θα κάνουμε την αντίθετη διαδικασία από την κωδικοποίηση , δηλαδή από τα κβαντισμένα σφάλματα πρόβλεψης θα ανακατασκευάσουμε το αρχικό σήμα.

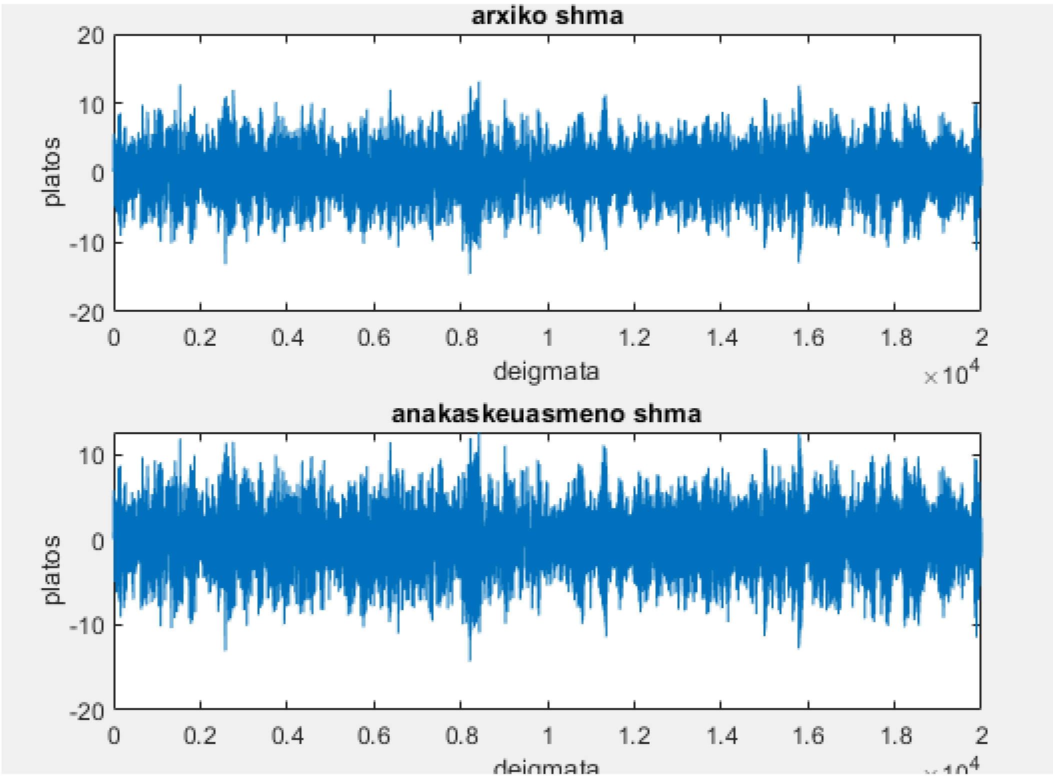
```
% apokodikopoihths
proigoumena_deigmata = zeros(p,1);
y_meto_simvolo_apano_tonismeno = 0;
anakataskeuasmeno_sima = zeros(sinolikos_arithmos_deigmaton,1);
for i=1:sinolikos_arithmos_deigmaton
    anakataskeuasmeno_sima(i) = y_meto_simvolo_apano(i) +
y_meto_simvolo_apano_tonismeno;
```

```
    proigoumena_deigmata = [anakataskeuasmeno_sima(i);
proigoumena_deigmata(1:p-1)];
    y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
end
```

Το μόνο που έχει μείνει να κάνουμε είναι να υλοποιήσουμε τον κβαντιστη. Πρέπει να βρούμε το βήμα Δ , να ορίσουμε τα κέντρα κβαντισης και να αναθέτουμε κάθε τιμή στο κατάλληλο επίπεδο.

```
function y_final = my_quantizer(y, N, min_value, max_value)

% periorizoume to deigma mesa sta oria tou kvadisti
if y < min_value
    y = min_value;
end
if y > max_value
    y = max_value;
end

% to step size tou kvadisth
D = (max_value - min_value) / (2^N - 1);

% kedra
centers = zeros(2^N, 1);
for i = 1:2^N
    centers(i) = min_value + D * (i - 1);
end

% se pio epipedo tha paei to deigma
for i = 1:2^N
    if (y >= centers(i) - D/2) && (y <= centers(i) + D/2)
        y_final = centers(i);
        break;
    end
end

end
```

Με τις ενδεικτικές τιμές Ν και p διακρίνουμε ότι το σύστημα λειτουργεί κανονικά

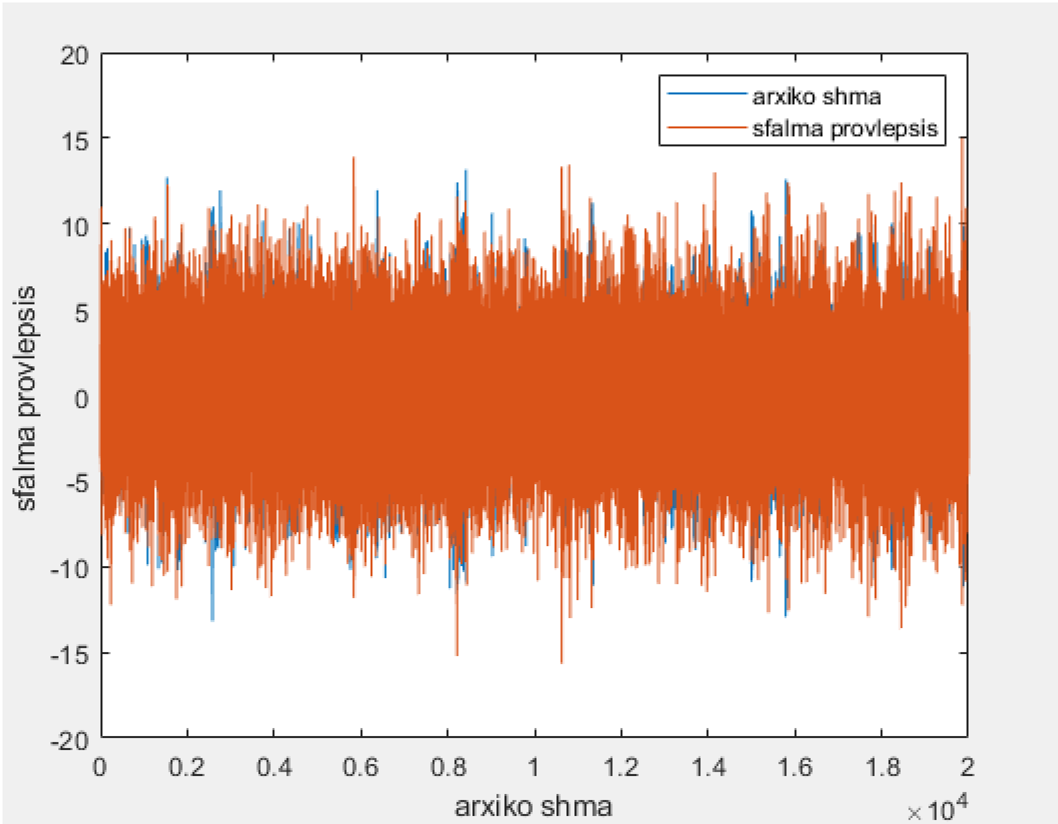


2. Διαλέγω τις τιμές p=6,12. Το μόνο που χρειάζεται να κάνουμε στον κώδικα είναι να βάλουμε κάθε φορά τις κατάλληλες τιμές p και Ν και να προσθέσουμε τα γραφήματα

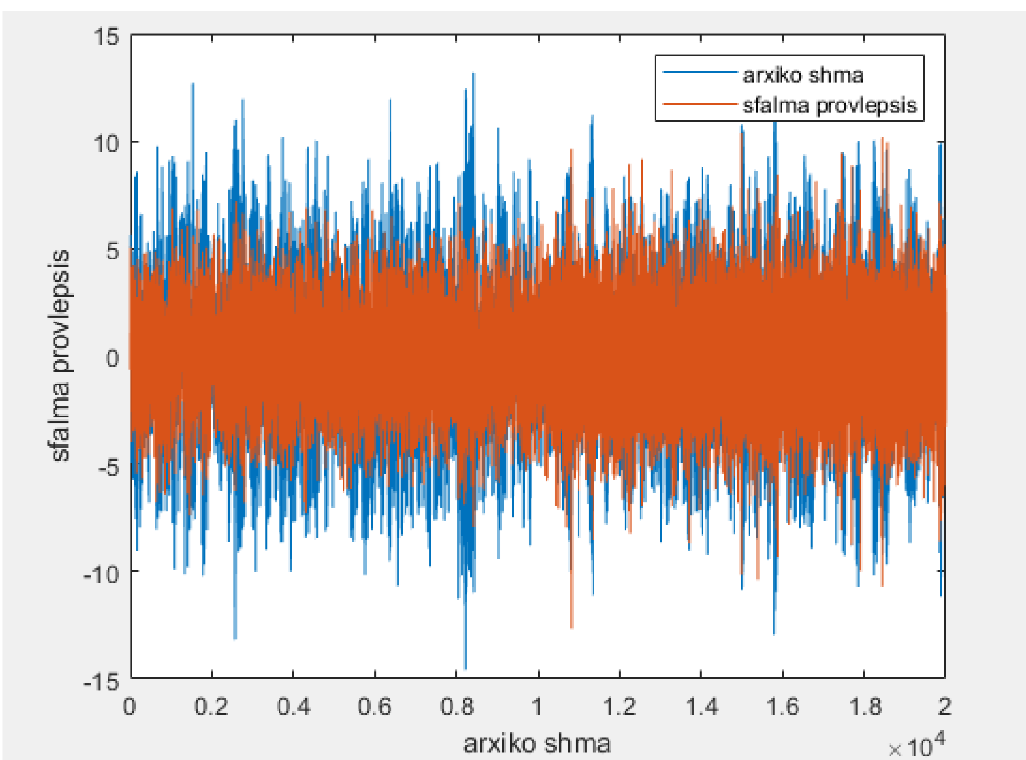
```
plot(t)
     hold on
     plot(y)
     xlabel('arxiko shma')
     ylabel('sfalma provlepsis ')
     legend('arxiko shma', 'sfalma provlepsis')
     hold off
```

P=6 :

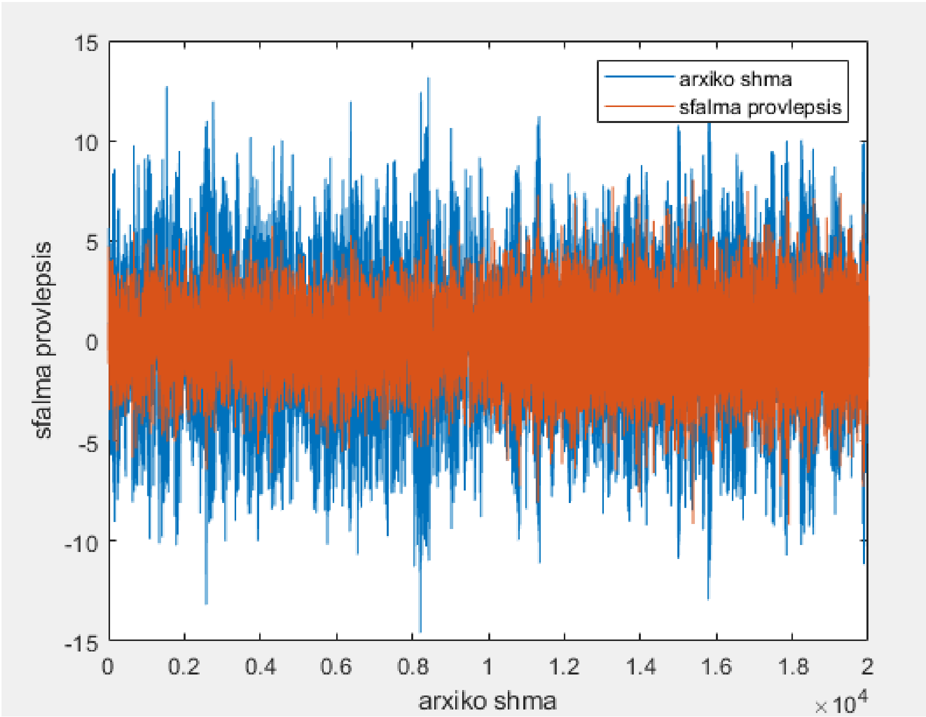
N=1


arxiko shma
sfalma provlepsis
sfalma provlepsis
arxiko shma
$\times 10^4$

N=2


arxiko shma
sfalma provlepsis
sfalma provlepsis
arxiko shma
$\times 10^4$

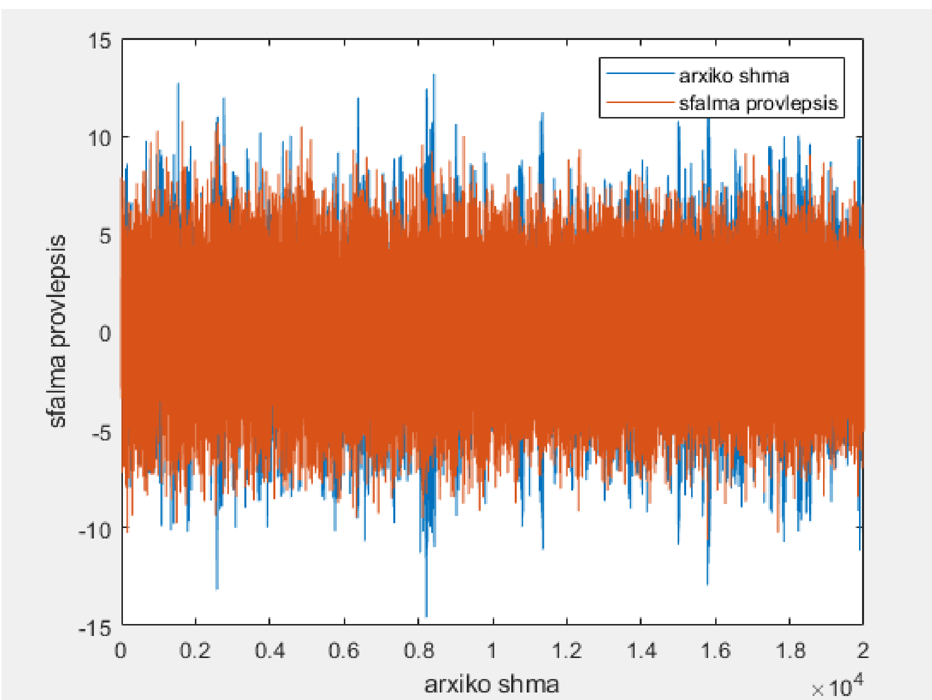
N=3



arxiko shma
sfalma provlepsis
sfalma provlepsis
arxiko shma
15
10
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10^4

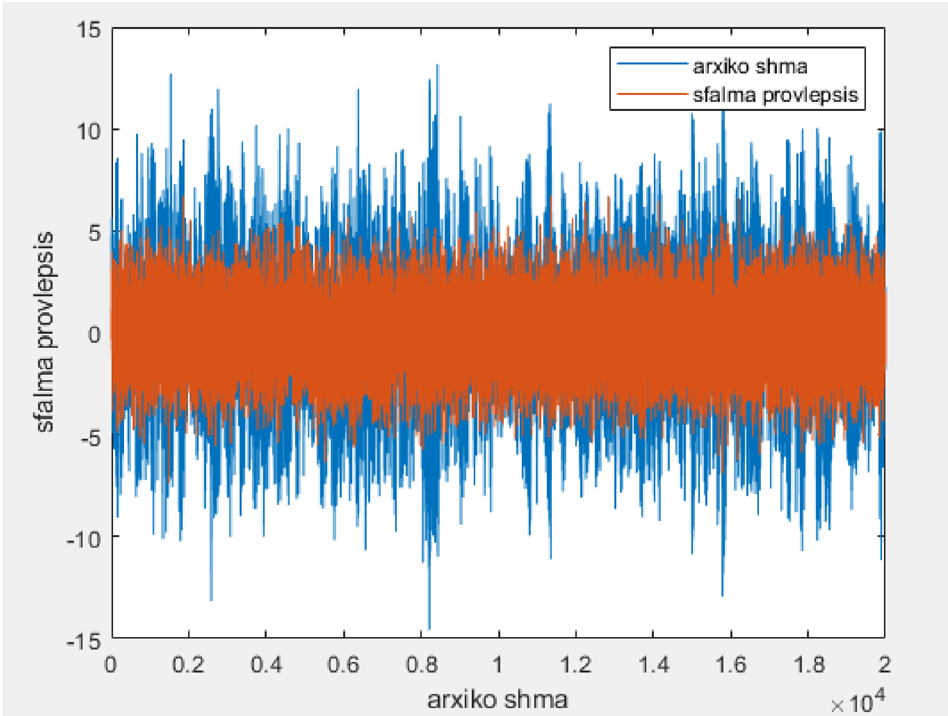
P=12 :

N=1

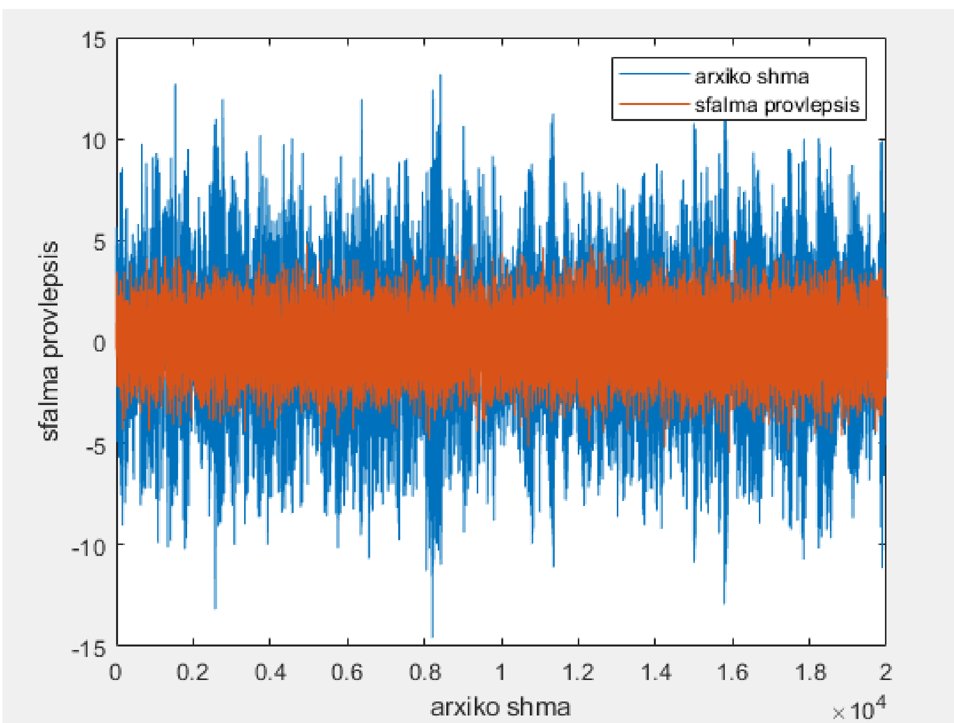


arxiko shma
sfalma provlepsis
sfalma provlepsis
arxiko shma
15
10
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
×10^4

N=2


arxiko shma
sfalma provlepsis
sfalma provlepsis
arxiko shma
15
10
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
$\times 10^4$

N=3


arxiko shma
sfalma provlepsis
sfalma provlepsis
arxiko shma
15
10
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
$\times 10^4$

Μπορούμε εύκολα να διακρίνουμε ότι όσο αυξάνεται το N , τόσο μικρότερο σφάλμα πρόβλεψης έχουμε. Το αποτέλεσμα αυτό είναι λογικό , καθώς με περισσότερα επίπεδα το βήμα Δ του κβαντιστη μειώνεται , συνεπώς μειώνεται και η διαφορά της πρόβλεψης με την τωρινή τιμή. Σε ένα σύστημα DPCM γνωρίζουμε ότι είναι σημαντικό τα δείγματα να μην έχουν μεγάλη απόκλιση για να έχω την βέλτιστη δυνατή απόδοση έναντι του κόστους χώρου , καθώς το κάθε επίπεδο θα αναπαριστάτε από περισσότερα bit. Επίσης παρατηρούμε ότι η αύξηση του p μείωσε το σφάλμα κβαντισης αλλά σε μικρότερο βαθμό απότι το N. Αυτό συμβαίνει καθώς η πρόβλεψη χρησιμοποιεί περισσότερα δείγματα άρα θα είναι καλύτερη έως ένα βαθμό. Συμπερασματικά , η αύξηση του N επιφέρει σίγουρη βελτίωση ενώ του p από ελάχιστη έως καθόλου.

3.

Θα υπολογίσουμε το MSE για καθέναν από τους συνδυασμούς που ζητούνται. Θα τροποποιήσω τον αρχικό κώδικα με loop ώστε να διατρέχει όλα τα ζευγάρια και να υπολογίζει το σφάλμα.

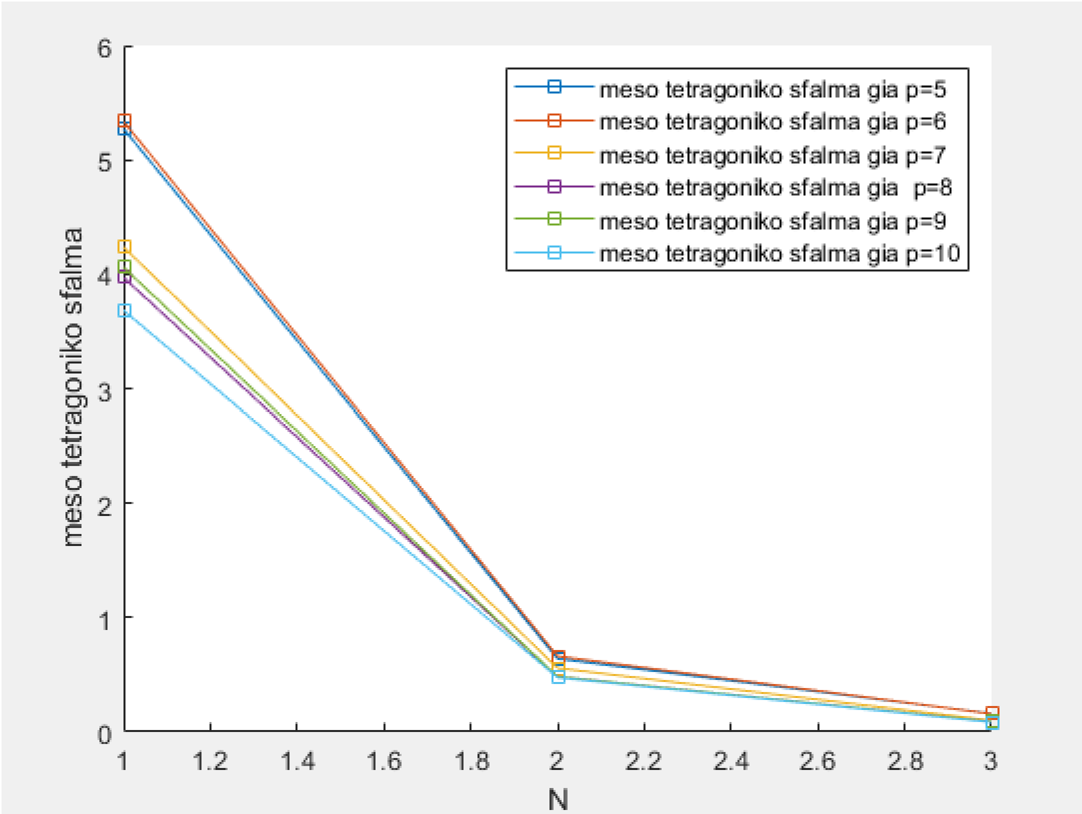
```
% ipologismos mse gia kathe iteration tou loop
            MSE(p-4, N) = mean((x - anakataskeuasmeno_sima).^2);
```

```
mse gia kathe sindiasmo:
    5.2714    0.6409    0.1611
    5.3353    0.6633    0.1586
    4.2377    0.5591    0.1004
    3.9709    0.4855    0.0898
    4.0595    0.4841    0.0892
    3.6841    0.4766    0.0863
```

Στην συνέχεια απλά θα πάρω τα αποτελέσματα που βρήκα και θα δημιουργήσω χειροκίνητα τα γραφήματα

```
times_kvadisti = [1 2 3];
mse_giap5 = [5.2714, 0.6409, 0.1611];
mse_giap6 = [5.3353, 0.6633, 0.1586];
mse_giap7 = [4.2377, 0.5591, 0.1004];
mse_giap8 = [3.9709, 0.4855, 0.0898];
mse_giap9 = [4.0595, 0.4841, 0.0892];
mse_giap10 = [3.6841, 0.4766, 0.0863];
```

Το αποτέλεσμα του κώδικα είναι



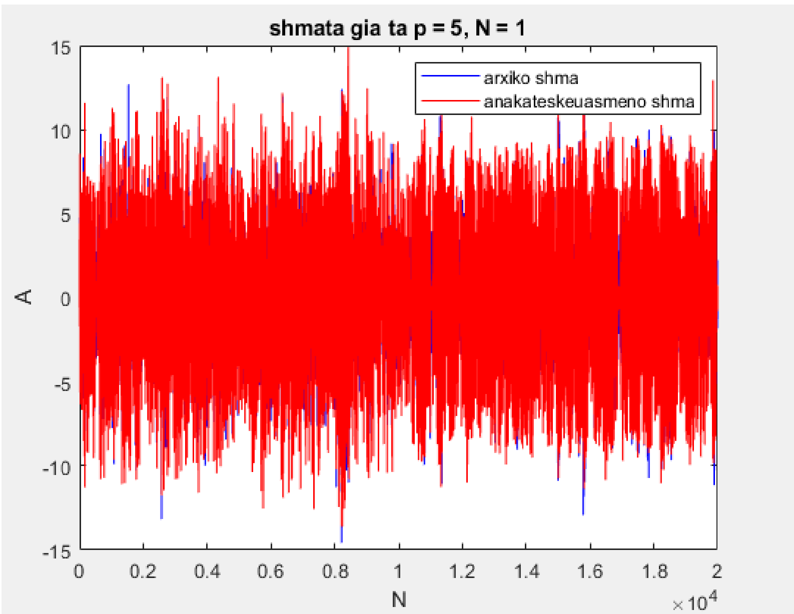
Παρατηρώντας τα αποτελέσματα μπορούμε να διακρίνουμε ότι :

Η αύξηση του p γενικά οδηγεί στην μείωση του σφάλματος , αν και όπως έχουμε αναφέρει παραπάνω αυτό δεν είναι εγγυημένο. Σε μερικές περιπτώσεις (για τις τιμές 6 και 9) το σφάλμα μάλιστα αυξάνεται. Επιπλέον , οι αλλαγές στην πραγματικότητα είναι απειροελάχιστες σε σχέση με την αύξηση του N. Το p στην καλύτερη περίπτωση μειώνει το σφάλμα κατά 1.5 μονάδα , ενώ το N μειώνει το σφάλμα έως και 5 μονάδες. Αυτά τα ευρήματα συμπίπτουν με το συμπέρασμα της προηγουμένης ερώτησης.

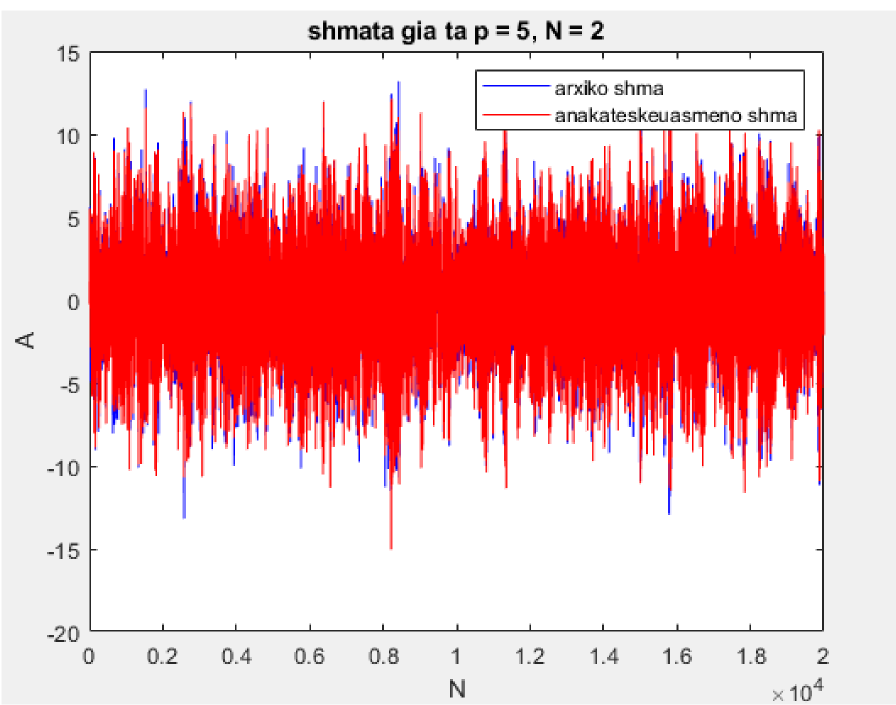
4. Θα κάνω loop όπως στο προηγούμενο ερώτημα αλλά συγκεκριμένα για p=5 και p=10. Στην συνέχεια απλά θα βάλω τις καμπύλες του αρχικού σήματος και του ανακασκευασμενου στο ίδιο σχήμα.

Για p=5 :

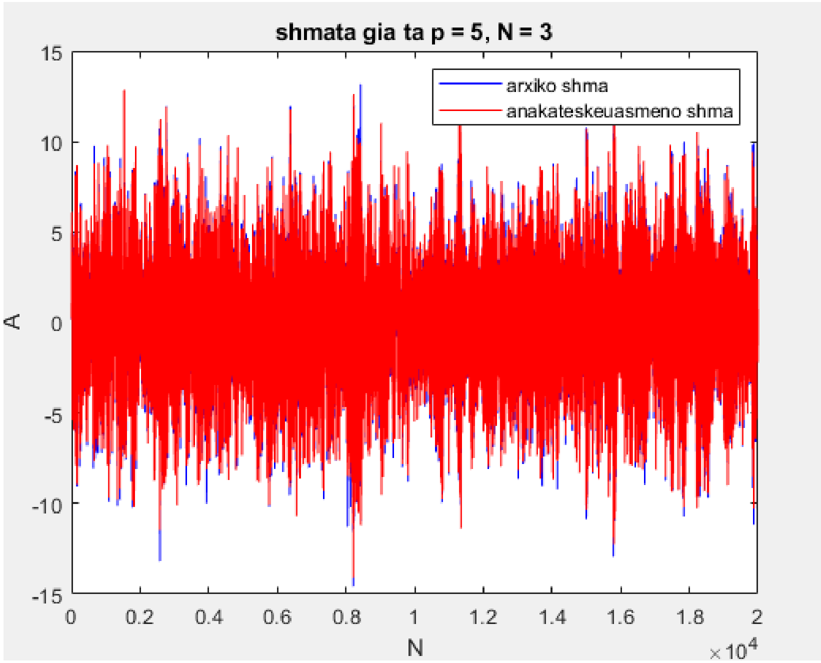
N=1



N=2



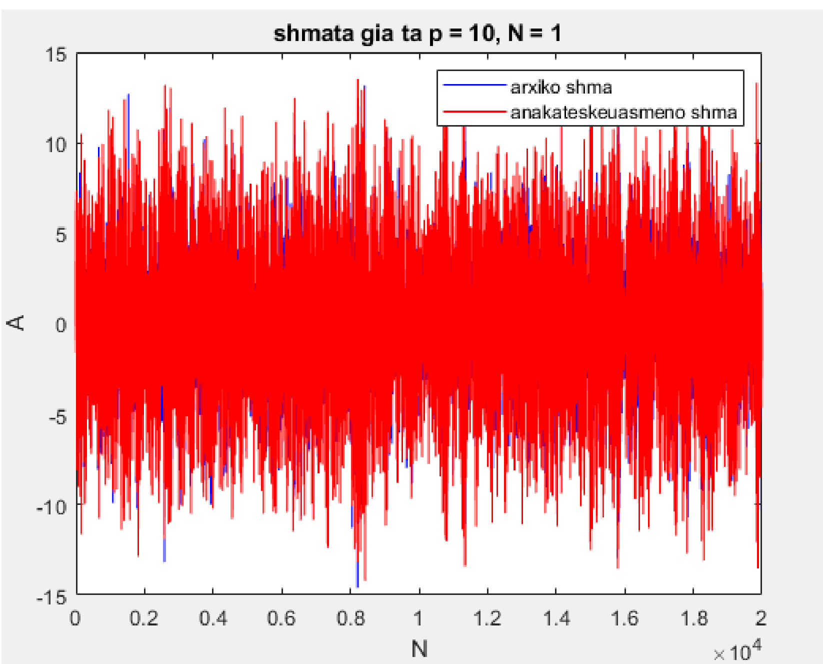
N=3



shmata gia ta p = 5, N = 3
arxiko shma
anakateskeuasmeno shma
A
N
15
10
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
$\times 10^4$

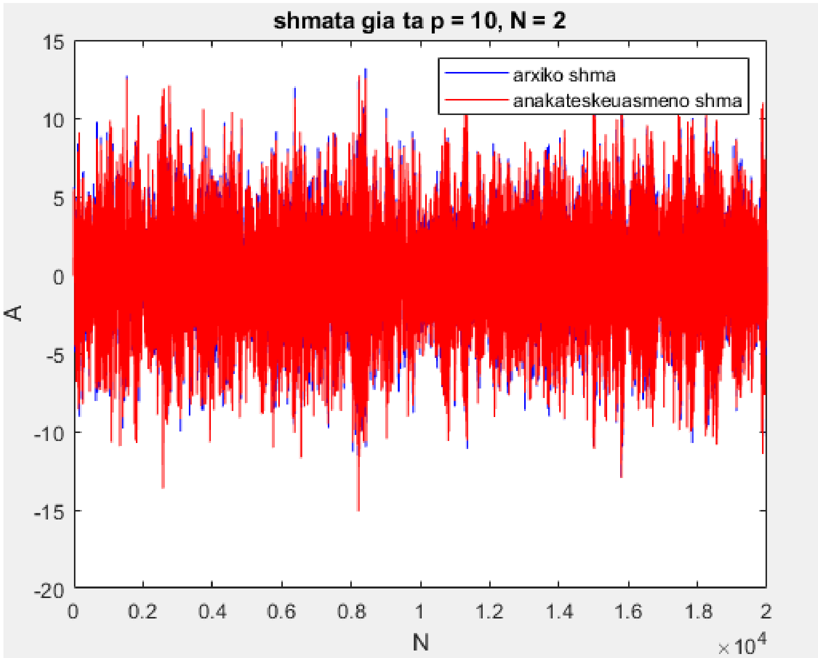
Για p=10:

N=1

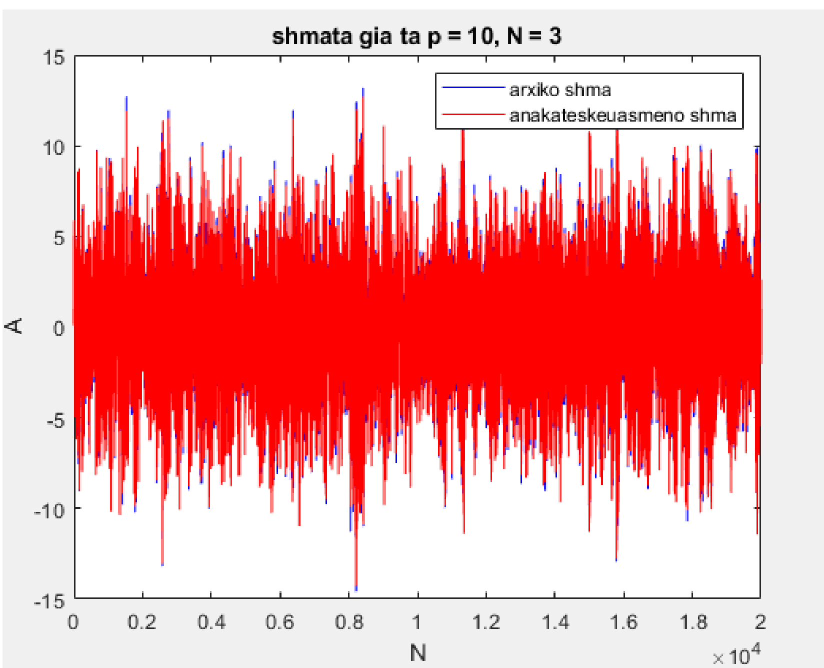


shmata gia ta p = 10, N = 1
arxiko shma
anakateskeuasmeno shma
A
N
15
10
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
$\times 10^4$

N=2


shmata gia ta p = 10, N = 2
arxiko shma
anakateskeuasmeno shma
A
N
×10^4
15
10
5
0
-5
-10
-15
-20
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

N=3


shmata gia ta p = 10, N = 3
arxiko shma
anakateskeuasmeno shma
A
N
×10^4
15
10
5
0
-5
-10
-15
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Μπορούμε να διακρίνουμε ότι το p βοηθάει στην καλύτερη ποιότητα του σήματος ανακατασκευής , αλλά όχι όσο η αύξηση του N. Βλέπουμε κατανοητή με το μάτι ότι το ανακατασκευασμενο σήμα ακολουθεί καλυτέρα το αρχικό όσο αυξάνουμε το N , ενώ με το p οι αλλαγές είναι ελάχιστες και δύσκολο να εντοπιστούν. Τα ευρήματα αυτά συμπίπτουν με αυτά από τα προηγούμενα ερωτήματα .

Κώδικας

Μέρος α

1. a

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


%  vrisko ta monadika simvola pigis
simvola_pigis = unique(I(:));

%  ipologismos pithanotiton
pithanotites = histcounts(I, numel(simvola_pigis)) / numel(I);


disp('Simvola Pigis:');
disp(simvola_pigis);

disp('Pithanotites:');
disp(pithanotites);
```

b.

```
% ipologismos edropias
entropia_pigis = sum(pithanotites .* log2(1 ./ pithanotites));
disp('edropia pigis:');
disp(entropia_pigis);

% dimiourgia dedrou huffman
huffmanDict = huffmandict(simvola_pigis, pithanotites);

% kodikopoihsh huffman
kodikopoihsh = huffmanenco(I(:), huffmanDict);
```

```
disp('dedrohuffman:');
disp(huffmanDict);

% meso mhkos kodika
mhkos_huffman = sum(pithanotites .* arrayfun(@(x) length(huffmanDict{x, 2}),
1:size(huffmanDict, 1)));
disp('Meso mhkos huffman:');
disp(mhkos_huffman);

% apodotikothta
apodotikothta = entropia_pigis / mhkos_huffman;
disp('apodotikothta:');
disp(apodotikothta);
```

2. a.

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

% megethos eikonas
[row, col] = size(I);

% metavlites pou kratane ta monadika zeugaria pou vrisko
orizodia_zeugaria = containers.Map('KeyType', 'char', 'ValueType', 'double');
katheta_zeugaria = containers.Map('KeyType', 'char', 'ValueType', 'double');


sinolika_orizodia_zeugaria = 0;
sinolika_katheta_zeugaria = 0;

% euresi orizodion zeugarion
for i = 1:row
    for j = 1:(col - 1)
        orizodio_zeugari = [I(i, j), I(i, j + 1)];
        pair_str = num2str(orizodio_zeugari);
        if isKey(orizodia_zeugaria, pair_str)
            orizodia_zeugaria(pair_str) = orizodia_zeugaria(pair_str) + 1;
        else
            orizodia_zeugaria(pair_str) = 1;
        end
        sinolika_orizodia_zeugaria = sinolika_orizodia_zeugaria + 1;
    end
end

% euresi katheton zeugarion
for i = 1:(row - 1)
    for j = 1:col
        katheto_zeugari = [I(i, j), I(i + 1, j)];
        pair_str = num2str(katheto_zeugari);
```

```
        if isKey(katheta_zeugaria, pair_str)
            katheta_zeugaria(pair_str) = katheta_zeugaria(pair_str) + 1;
        else
            katheta_zeugaria(pair_str) = 1;
        end
        sinolika_katheta_zeugaria = sinolika_katheta_zeugaria + 1;
    end
end
```

b.

```
% ipologimos edropias gia orizodia zeugaria
disp('orizodia zeugaria kai pithanotites:');
edropia_oriz = 0;
orizodies_pithanotites = [];
horizontal_keys = keys(orizodia_zeugaria);
for k = 1:length(horizontal_keys)
    key = horizontal_keys{k};
    arithos_emfaniseon = orizodia_zeugaria(key);
    pithanotita = arithos_emfaniseon / sinolika_orizodia_zeugaria;
    orizodies_pithanotites(end+1) = pithanotita;
    edropia_oriz = edropia_oriz - pithanotita * log2(pithanotita);
    disp([key, ' pithanotita: ', num2str(pithanotita)]);
end
disp(['edropia oriz: ', num2str(edropia_oriz)]);

% ipologismos edropias gia katheta zeugaria
disp('katheta zeugaria kai pithanotites:');
edropia_kath = 0;
kathetes_pithanotites = [];
vertical_keys = keys(katheta_zeugaria);
for k = 1:length(vertical_keys)
    key = vertical_keys{k};
    arithos_emfaniseon = katheta_zeugaria(key);
    pithanotita = arithos_emfaniseon / sinolika_katheta_zeugaria;
    kathetes_pithanotites(end+1) = pithanotita;
    edropia_kath = edropia_kath - pithanotita * log2(pithanotita);
    disp([key, ' pithanotita: ', num2str(pithanotita)]);
end
disp(['edropia kath: ', num2str(edropia_kath)]);

% dedro huffman gia orizodia
[orizodio_dedro, ~] = huffmandict(horizontal_keys, orizodies_pithanotites);


disp('dedro oriz:');
disp(orizodio_dedro);

% dedro huffman gia katheta
[katheto_dedro, ~] = huffmandict(vertical_keys, kathetes_pithanotites);


disp('dedro kath:');
disp(katheto_dedro);
```

```
% meso mhkos kodika gia orizodia
meso_mhkos_oriz = 0;
for k = 1:length(horizontal_keys)
    key = horizontal_keys{k};
    kodikas = orizodio_dedro{find(strcmp(key, orizodio_dedro(:,1))), 2};
    mhkos_kodika = length(kodikas);
    meso_mhkos_oriz = meso_mhkos_oriz + orizodies_pithanotites(k) * mhkos_kodika;
end


disp(['meso mhkos oriz: ', num2str(meso_mhkos_oriz), ' bits']);

% meso mhkos kodika gia katheta
meso_mhkos_kath = 0;
for k = 1:length(vertical_keys)
    key = vertical_keys{k};
    kodikas = katheto_dedro{find(strcmp(key, katheto_dedro(:,1))), 2};
    mhkos_kodika = length(kodikas);
    meso_mhkos_kath = meso_mhkos_kath + kathetes_pithanotites(k) * mhkos_kodika;
end


disp(['meso mhkos kath: ', num2str(meso_mhkos_kath), ' bits']);

% apodotikothta gia orizodia
apodotikothta_oriz = edropia_oriz / meso_mhkos_oriz;
disp(['apodotikothta oriz: ', num2str(apodotikothta_oriz * 100), '%']);

% apodotikothta gia katheta
apodotikothta_kath = edropia_kath / meso_mhkos_kath;
disp(['apodotikothta kath: ', num2str(apodotikothta_kath * 100), '%']);
```

4.

```
% sigrisi ton eikonon
apokodikopoihsh = huffmandeco(kodikopoihsh, huffmanDict);
apokodikopoihsh = reshape(apokodikopoihsh, size(I));
elenxos = isequal(I, apokodikopoihsh);
disp('Einai sosti i kodikopoihsh?');
disp(elenxos);

% vrisko ta bit tis kanonikis eikonas kai meta tis kodikopoihmenhs
bits_diadikhs_anaparastashs = numel(I) * 8;
bits_huffman = length(kodikopoihsh);


disp('arithmos bit original eikonas:');
disp(bits_diadikhs_anaparastashs);
disp('arithmos bit kodikopoihmenhs eikonas:');
disp(bits_huffman);
```

```
% logos sibieshs
j = bits_huffman / bits_diadikhs_anaparastashs;
disp('j:');
disp(j);
```

5.

```
% apo to erotima 4
x = kodikopoihsh;

% i sinartisi pou mas dinete sto zip
y = binary_symmetric_channel(x);

% ipologismos p sostis metavasis
sostabit = sum(x == y);
sinolikabit = length(x);
p = sostabit / sinolikabit;


disp(['p: ', num2str(p, '%.2f')]);

% diadiki edropia kai xoritikothta kanaliou
H_p = -p * log2(p) - (1 - p) * log2(1 - p);


xorithkothta_kanaliou = 1 - H_p;


disp(['Xorithkothta kanaliou: ', num2str(xorithkothta_kanaliou)]);

% ipologismos katanomis pithanotiton tis eisodou
arithmos_0 = sum(kodikopoihsh == 0);
arithmos_1 = sum(kodikopoihsh == 1);
olatabit = length(kodikopoihsh);


pithanotita_0 = arithmos_0 / olatabit;
pithanotita_1 = arithmos_1 / olatabit;

% edropia eisodou
H_X = - (pithanotita_0 * log2(pithanotita_0) + pithanotita_1 * log2(pithanotita_1));



% ipologismos amoivaias pliroforias
amoivaia_pliroforia = H_X - H_p;


disp(['Amoivaia Pliroforia: ', num2str(amoivaia_pliroforia)]);
```

Μέρος β

1.

```
function main()

close all;

% posa proigoumena deigmata tha exoume
p = 10;
% ta bit kvadisti
N = 3;
% dinamiki perioxi tou kvadisti
min_value = -3.5;
max_value = 3.5;
load('source.mat')
%diavasa to arxeio pou exei dothei kai i t einai i metavliti pou exei ta
%deigmata
x = t;
sinolikos_arithmos_deigmaton = length(x);
R = zeros(p,p);
r = zeros(p,1);

% yule walker
for i=1:p
    sum_r = 0;
    for n=p+1:sinolikos_arithmos_deigmaton
        sum_r = sum_r + x(n)*x(n-i);
    end
    r(i) = sum_r * 1/(sinolikos_arithmos_deigmaton-p);
    for j=1:p
        sum_R = 0;
        for n=p+1:sinolikos_arithmos_deigmaton
            sum_R = sum_R + x(n-j)*x(n-i);
        end
        R(i,j) = sum_R * 1/(sinolikos_arithmos_deigmaton-p);
    end
end

a = R\r;
for i=1:p
    a(i) = my_quantizer(a(i), 8, -2, 2);
end

% kodikopoiths
proigoumena_deigmata = zeros(p,1);
y = zeros(sinolikos_arithmos_deigmaton,1);
y_meto_simvolo_apano = zeros(sinolikos_arithmos_deigmaton,1);
y_meto_simvolo_apano_tonismeno = 0;
%ipologismos sfalmatos provepsis
for i=1:sinolikos_arithmos_deigmaton
    y(i) = x(i) - y_meto_simvolo_apano_tonismeno;
    y_meto_simvolo_apano(i) = my_quantizer(y(i),N,min_value,max_value);
    y_meto_simvolo_apano_tonismeno = y_meto_simvolo_apano(i) +
y_meto_simvolo_apano_tonismeno;
```

```
    proigoumena_deigmata = [y_meto_simvolo_apano_tonismeno; proigoumena_deigmata(1:p-1)];
    y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
end

% apokodikopoihths
proigoumena_deigmata = zeros(p,1);
y_meto_simvolo_apano_tonismeno = 0;
anakataskeuasmeno_sima = zeros(sinolikos_arithmos_deigmaton,1);
for i=1:sinolikos_arithmos_deigmaton
    anakataskeuasmeno_sima(i) = y_meto_simvolo_apano(i) + y_meto_simvolo_apano_tonismeno;
    proigoumena_deigmata = [anakataskeuasmeno_sima(i); proigoumena_deigmata(1:p-1)];
    y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
end

% arxiko shma kai shma anakataskeuhs
figure;
subplot(2,1,1);
plot(x);
title('arxiko shma');
xlabel('deigmata');
ylabel('platos');

subplot(2,1,2);
plot(anakataskeuasmeno_sima);
title('anakaskeuasmeno shma');
xlabel('deigmata');
ylabel('platos');

end
```

2. plot(t)

```
hold on
plot(y)
xlabel('arxiko shma')
ylabel('sfalma provlepsis ')
legend('arxiko shma', 'sfalma provlepsis')
hold off
```

3.
```
function main()

    close all;
    load('source.mat')
    x = t;
    sinolikos_arithmos_deigmaton = length(x);

    % meso tetragoniko sfalma
    MSE = zeros(6, 3);

    % kano to loop adi gia mia mia fora opos to 2
    for p = 5:10
        for N = 1:3
            min_value = -3.5;
            max_value = 3.5;

            R = zeros(p,p);
            r = zeros(p,1);


            for i=1:p
                sum_r = 0;
                for n=p+1:sinolikos_arithmos_deigmaton
                    sum_r = sum_r + x(n)*x(n-i);
                end
                r(i) = sum_r * 1/(sinolikos_arithmos_deigmaton-p);
                for j=1:p
                    sum_R = 0;
                    for n=p+1:sinolikos_arithmos_deigmaton
                        sum_R = sum_R + x(n-j)*x(n-i);
                    end
                    R(i,j) = sum_R * 1/(sinolikos_arithmos_deigmaton-p);
                end
            end


            a = R\r;
            for i=1:p
                a(i) = my_quantizer(a(i), 8, -2, 2);
            end

            % kodikopoihths
            proigoumena_deigmata = zeros(p,1);
            y = zeros(sinolikos_arithmos_deigmaton,1);
            y_meto_simvolo_apano = zeros(sinolikos_arithmos_deigmaton,1);
            y_meto_simvolo_apano_tonismeno = 0;
            for i=1:sinolikos_arithmos_deigmaton
                y(i) = x(i) - y_meto_simvolo_apano_tonismeno;
                y_meto_simvolo_apano(i) = my_quantizer(y(i),N,min_value,max_value);
                y_meto_simvolo_apano_tonismeno = y_meto_simvolo_apano(i) +
y_meto_simvolo_apano_tonismeno;
                proigoumena_deigmata = [y_meto_simvolo_apano_tonismeno;
proigoumena_deigmata(1:p-1)];
```

```
                y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
            end

            % apokodikopoiths
            proigoumena_deigmata = zeros(p,1);
            y_meto_simvolo_apano_tonismeno = 0;
            anakataskeuasmeno_sima = zeros(sinolikos_arithmos_deigmaton,1);
            for i=1:sinolikos_arithmos_deigmaton
                anakataskeuasmeno_sima(i) = y_meto_simvolo_apano(i) +
y_meto_simvolo_apano_tonismeno;
                proigoumena_deigmata = [anakataskeuasmeno_sima(i);
proigoumena_deigmata(1:p-1)];
                y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
            end

            % ipologismos mse gia kathe iteration tou loop
            MSE(p-4, N) = mean((x - anakataskeuasmeno_sima).^2);
        end
    end
    disp('mse gia kathe sindiasmo:');
    disp(MSE);
    % apla vazo ta mse pou vrika xeirokinita
times_kvadisti = [1 2 3];
mse_giap5 = [5.2714, 0.6409, 0.1611];
mse_giap6 = [5.3353, 0.6633, 0.1586];
mse_giap7 = [4.2377, 0.5591, 0.1004];
mse_giap8 = [3.9709, 0.4855, 0.0898];
mse_giap9 = [4.0595, 0.4841, 0.0892];
mse_giap10 = [3.6841, 0.4766, 0.0863];


hold on;
plot1 = plot(times_kvadisti, mse_giap5);
plot2 = plot(times_kvadisti, mse_giap6);
plot3 = plot(times_kvadisti, mse_giap7);
plot4 = plot(times_kvadisti, mse_giap8);
plot5 = plot(times_kvadisti, mse_giap9);
plot6 = plot(times_kvadisti, mse_giap10);

set(plot1, 'Marker', 'square');
set(plot2, 'Marker', 'square');
set(plot3, 'Marker', 'square');
set(plot4, 'Marker', 'square');
set(plot5, 'Marker', 'square');
set(plot6, 'Marker', 'square');

hold off;
xlabel('N');
ylabel('meso tetragoniko sfalma');
legend('meso tetragoniko sfalma gia p=5', 'meso tetragoniko sfalma gia p=6', 'meso
tetragoniko sfalma gia p=7', 'meso tetragoniko sfalma gia  p=8', 'meso tetragoniko
sfalma gia p=9', 'meso tetragoniko sfalma gia p=10');

end
```

4.

```
function main()

    close all;
    load('source.mat');
    x = t;
    sinolikos_arithmos_deigmaton = length(x);

    % times p pou zitoude
    p_sigekrimena = [5, 10]; % Only p = 5 and p = 10

    %times N pou zitoude
    N_values = 1:3; % N = 1, 2, 3

    % to idio me prin apla kano loop mono ta 2 p pou zitaei
    for p = p_sigekrimena
        for N = N_values
            min_value = -3.5;
            max_value = 3.5;

            % yule walker
            R = zeros(p,p);
            r = zeros(p,1);

            for i=1:p
                sum_r = 0;
                for n=p+1:sinolikos_arithmos_deigmaton
                    sum_r = sum_r + x(n)*x(n-i);
                end
                r(i) = sum_r * 1/(sinolikos_arithmos_deigmaton-p);
                for j=1:p
                    sum_R = 0;
                    for n=p+1:sinolikos_arithmos_deigmaton
                        sum_R = sum_R + x(n-j)*x(n-i);
                    end
                    R(i,j) = sum_R * 1/(sinolikos_arithmos_deigmaton-p);
                end
            end

            a = R\r;
            for i=1:p
                a(i) = my_quantizer(a(i), 8, -2, 2);
            end

            % kodikopoiths
            proigoumena_deigmata = zeros(p,1);
            y = zeros(sinolikos_arithmos_deigmaton,1);
            y_meto_simvolo_apano = zeros(sinolikos_arithmos_deigmaton,1);
            y_meto_simvolo_apano_tonismeno = 0;

            for i=1:sinolikos_arithmos_deigmaton
                y(i) = x(i) - y_meto_simvolo_apano_tonismeno;
                y_meto_simvolo_apano(i) = my_quantizer(y(i),N,min_value,max_value);
```

```
                y_meto_simvolo_apano_tonismeno = y_meto_simvolo_apano(i) +
y_meto_simvolo_apano_tonismeno;
                proigoumena_deigmata = [y_meto_simvolo_apano_tonismeno;
proigoumena_deigmata(1:p-1)];
                y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
            end

            % apokodikopoiths
            proigoumena_deigmata = zeros(p,1);
            y_meto_simvolo_apano_tonismeno = 0;
            anakataskeuasmeno_sima = zeros(sinolikos_arithmos_deigmaton,1);

            for i=1:sinolikos_arithmos_deigmaton
                anakataskeuasmeno_sima(i) = y_meto_simvolo_apano(i) +
y_meto_simvolo_apano_tonismeno;
                proigoumena_deigmata = [anakataskeuasmeno_sima(i);
proigoumena_deigmata(1:p-1)];
                y_meto_simvolo_apano_tonismeno = a'*proigoumena_deigmata;
            end

           % ta plot pou zitaei na doume
            figure;
            plot(x, 'b', 'DisplayName', 'arxiko shma');
            hold on;
            plot(anakataskeuasmeno_sima, 'r', 'DisplayName', 'anakateskeuasmeno
shma');
            hold off;
            title(['shmata gia ta p = ' num2str(p) ', N = ' num2str(N)]);
            xlabel('N');
            ylabel('A');
            legend show;
        end
    end
end
```