Ψηφιακές Τηλεπικοινωνίες

Εργαστηριακό Σετ 1

Χειμερινό Εξάμηνο 2023-2024

Βιλλιώτης Αχιλλέας 1084567

4ο έτος

Μέρος Α′

Ερώτημα 1

a.

probabilities	symbols
0.096167	0
0.0814	17
0.068333	34
0.0625	51
0.076833	68
0.090467	85
0.11317	102
0.0906	119
0.0965	136
0.067133	153
0.0389	170
0.032933	187
0.033667	204
0.0259	221
0.022367	238
0.0031333	255

b. Παρατηρούμε πως το μέσο μήκος του κώδικά μας (3.8374 bit/symbol) είναι πολύ κοντά στην εντροπία (3,7831 bit/symbol) το οποίο μας λέει πως έχουμε κάνει σχεδόν την τέλεια κωδικοποίηση, το οποίο είναι αναμενόμενο από την κωδικοποίηση Huffman.

Ερώτημα 2

Αρχικά να σημειωθεί πως, λόγω του πως λειτουργεί η `huffencode`, δηλαδή δέχεται ένα μονοδιάστατο πίνακα, θεωρήθηκε πως ο σωστός τρόπος να γίνει επέκταση πηγής είναι να χωρίσουμε το αρχικό σήμα σε δυάδες. Αυτό μπορεί να διαφέρει από τον συμβατικό τρόπο που κάποιος θα έπαιρνε συνδυασμούς με όλα τα γειτονικά πίξελ, όμως, το αποτέλεσμα αυτής της διαδικασίας θα ήταν να έχουμε πολλά

αχρησιμοποίητα σύμβολα-δυάδες που θα αύξαναν το μέσο μήκος του κώδικα.

a. Βλ. παράρτημα.

b. Παρατηρούμε πως το μέσο μήκος του κώδικά μας (5.6412 bit/symbol) είναι ακόμα πιο κοντά στην εντροπία (5.6412 bit/symbol), αναμενόμενο αφού χρησιμοποιήσαμε επέκταση πηγής δεύτερης τάξης, άρα ο κώδικας γίνεται πιο αποδοτικός.

c. Η επίδοση της δεύτερης κωδικοποίησης με την επέκταση πηγής είναι σαφώς πιο αποδοτική.

Ερώτημα 3

a. Παρατηρούμε $H(X^2) = 5.6147 < 2H(X) = 2 \times 3.7831$. Αυτό ισχύει διότι η πηγή μας έχει μνήμη. Με άλλα λόγια, το κάθε πίξελ είναι εξαρτημένο από τα γειτονικά του πίξελ, το οποίο είναι λογικό αφού η εικόνα έχει συγκεκριμένη δομή, με περιοχές που ακολουθούν κανόνες, όπως μόνο μαύρο, τις διαγώνιους κλπ. Εφόσον η επέκταση πηγής θεωρεί πηγή χωρίς μνήμη, προφανώς το αποτέλεσμα δεν θα είναι το θεωρητικό.

b. Μπορούμε από την θεωρία να θέσουμε τα εξής φράγματα (για πηγή DMS)

Για την πρώτης τάξης: $H(X) < L(X) < H(X) + 1$..

Για την δεύτερης τάξης: $H(X^2) < L(X^2) < H(X^2) + 1$.

Με αριθμούς:

Πρώτης: $3.7831 < L=3.8374 < 4.7831$
Δεύτερης: $5.6147 < L=5.6412 < 6.6147$

Προφανώς η πηγή μας δεν είναι DMS, οπότε το ότι πλησιάζουμε στα όρια αυτά είναι προφανώς πολύ θετικό.

Ερώτημα 4

Κωδικοποιώντας βγάζουμε λόγο συμπίεσης J = 47,967%.

Ερώτημα 5

Εκτελώντας 1.000 μεταδόσεις της κωδικοποιημένης εικόνας, μπορούμε να υπολογίσουμε τον παράγοντα λάθους p=0.12. Έτσι η χωρητικότητα του καναλιού C= 0.470472 bits.

Την αμοιβαία πληροφορία την υπολογίζουμε μέσω του τύπου I=H(Y) – H(Y|X) = 0,462696 bits, το οποίο σημαίνει πως χρησιμοποιούμε το κανάλι με σχεδόν βέλτιστο τρόπο.

Μέρος Β’

Ερώτημα 1

Δημιουργήθηκαν οι εξής συναρτήσεις: `dpcm_encode`, `dpcm_decode` και `my_quantizer`. Παρακάτω εξηγούμε την λειτουργία τους.

dpcm_encode:

Κωδικοποίηση από την μεριά του πομπού. Δέχεται το σήμα καθώς και τις παραμέτρους p, N, min_val και max_val (για την κβάντιση). Υπολογίζεται η αυτοσυσχέτιση του σήματος, λύνεται το σύστημα εξισώσεων Yule-Walker και κβαντίζονται οι συντελεστές a με N=8 στο [-2,2]. Μετά κωδικοποιείται το σήμα, χρησιμοποιώντας μια μνήμη μεγέθους p, σύμφωνα με την θεωρία. Η έξοδος της συνάρτησης είναι το κωδικοποιημένο σήμα και οι κβαντισμένοι συντελεστές a.

dpcm_decode:

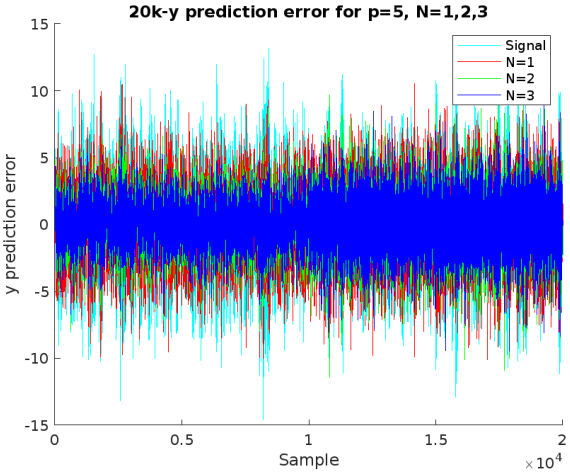
Αντίστοιχα με τον encode, σύμφωνα με την θεωρία, δέχεται το κωδικοποιημένο σήμα καθώς και τους συντελεστές a. Το σήμα αποκωδικοποιείται και επιστρέφεται.

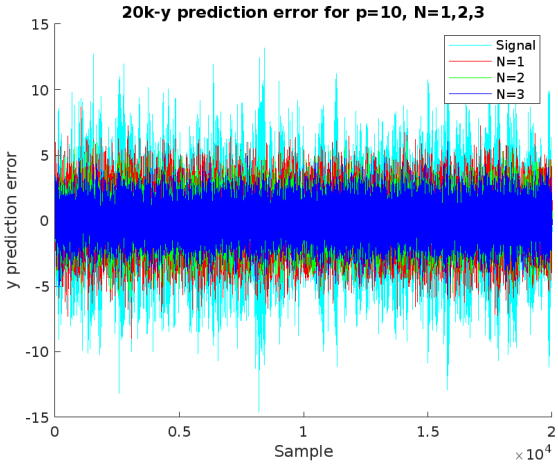
my_quantizer:

Σύμφωνα με τα ζητούμενα, υπολογίζεται το δέλτα, και μέσω αυτού ο δείκτης ο οποίος χρησιμοποιείται εκτός της συνάρτησης για να βρεθεί η κβαντισμένη τιμή, σέ πίνακα που παράγεται από την εντολή `delta = (max_val - min_val) / (2^N); centers = (max_val - delta/2):-delta:(min_val + delta/2);`.

Ερώτημα 2

Επιλέγουμε τιμές για το p=5 και 10. Παρατηρούμε πως για p=10 το σφάλμα είναι μικρότερο (άρα μικρότερη δυναμική περιοχή) ενώ και στις δύο περιπτώσεις, για N=2 και 3 παρατηρείται μεγάλη μείωση του σφάλματος σε σχέση με N=1.





Ερώτημα 3

Παρατηρούμε μείωση όσο αυξάνεται το p, ενώ για p=9 μια μικρή αύξηση. Επίσης, για N=2 και 3 το σφάλμα μειώνεται σημαντικά και λόγω του είναι περίπου ίσο με 5, αυτό σημαίνει πως κατά την μέση περίπτωση, κατά την κβάντιση του δεν χάνεται πολλή πληροφορία, αφού έχει απόλυτη τιμή μικρότερη του 3,5 που είναι η κβάντιση.

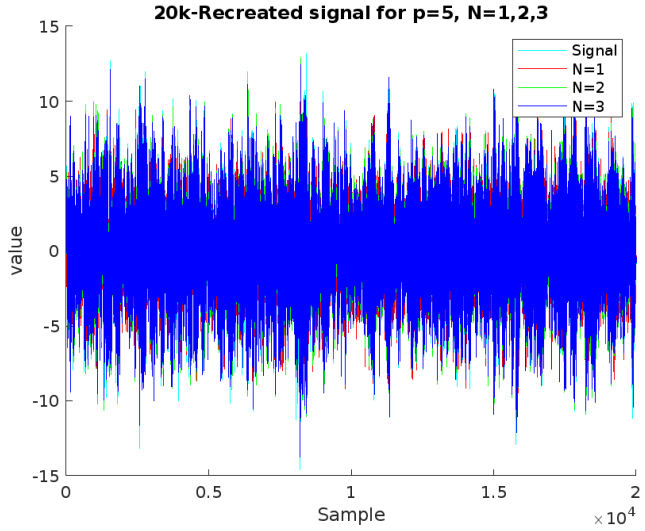

```
{[                                  1.2941 -1.5922 0.9961 -0.5412 -0.0235]}
{[                            1.2941 -1.5608 0.9490 -0.4784 -0.0863 0.0392]}
{[                      1.2627 -1.5294 1.1843 -0.9647 0.7137 -0.6039 0.5098]}
{[                1.0902 -1.3098 0.9490 -0.6196 0.2902 -0.0706 0.0706 0.3529]}
{[       1.1373 -1.2941 0.9333 -0.5725 0.1961 0.0706 -0.1490 0.5255 -0.1647]}
{[1.1059 -1.1843 0.9020 -0.5569 0.2431 -0.0549 0.0706 0.2275 0.1020 -0.2275]}
```

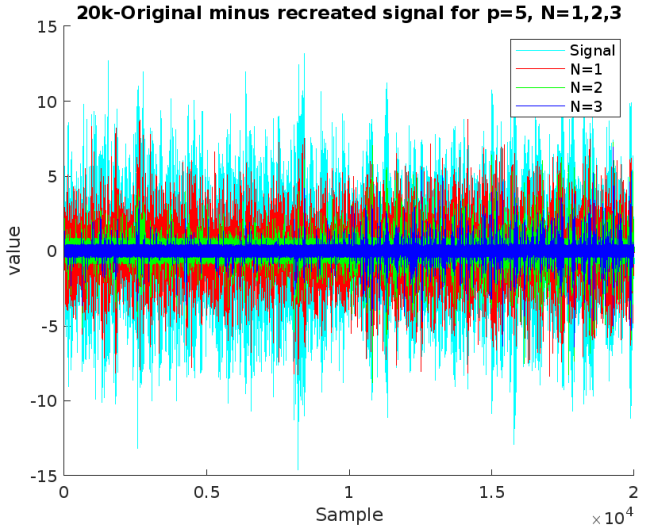
Οι συντελεστές a παρατηρούμε πως εναλλάσσουν πρόσημο (οι περισσότεροι) και επίσης όσο μεγαλώνει το μέγεθος της μνήμης p, όλοι οι συντελεστές γίνονται μικρότεροι σε απόλυτη τιμή που σημαίνει πως το αποτέλεσμα θα έχει λιγότερες ακραίες τιμές.

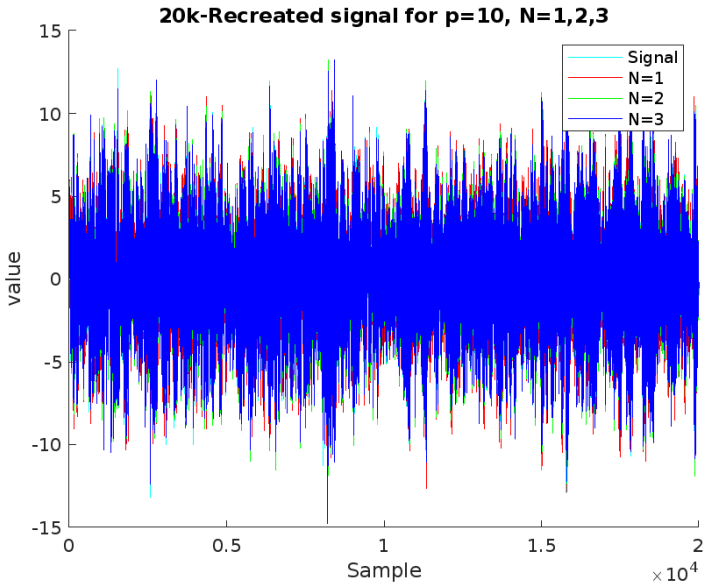
Ερώτημα 4

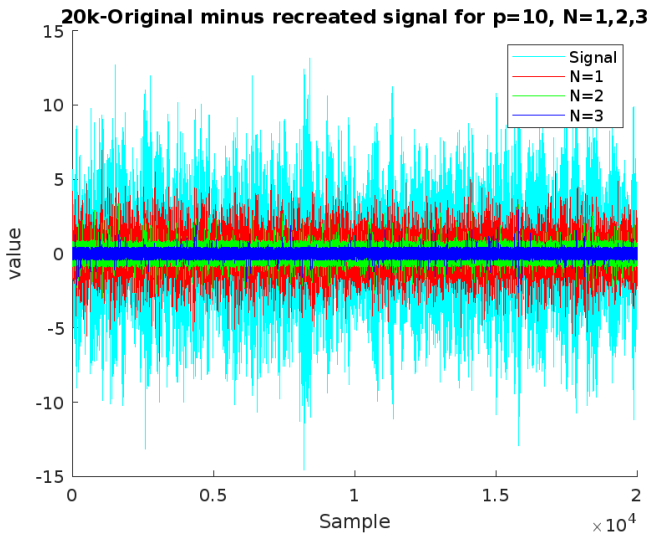
Με σκοπό να διακρίνονται οι διαφορές πιο εύκολα, εκτός από τα γραφήματα των ανακατασκευασμένων σημάτων, δημιουργήσαμε και τα γραφήματα που δείχνουν την απόκλιση του ανακατασκευασμένου από το αρχικό.

Αναμενόμενα, τα σήματα για N=2 και 3 είναι πολύ καλύτερης ποιότητας από για N=1. Επίσης για p=10, το λάθος παραμένει σχετικά σταθερό, σε σχέση με για p=5, το οποίο οφείλεται στην αυξημένη μνήμη η οποία αναγνωρίζει μεγαλύτερα μοτίβα στην εικόνα (ειδικά στο τέλος της εικόνας).



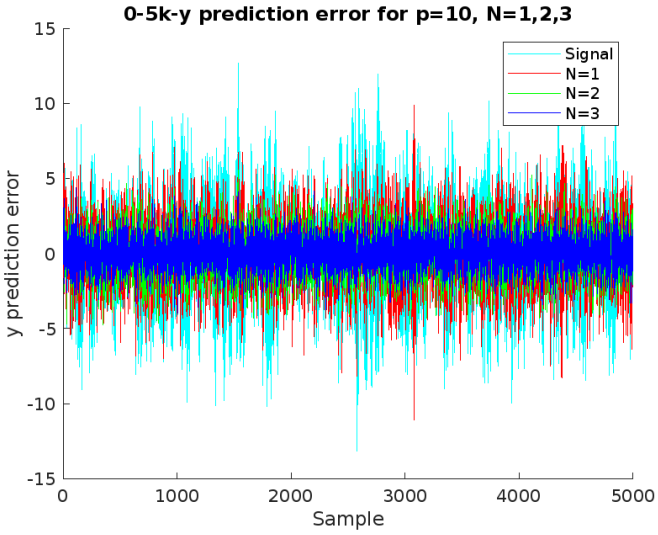




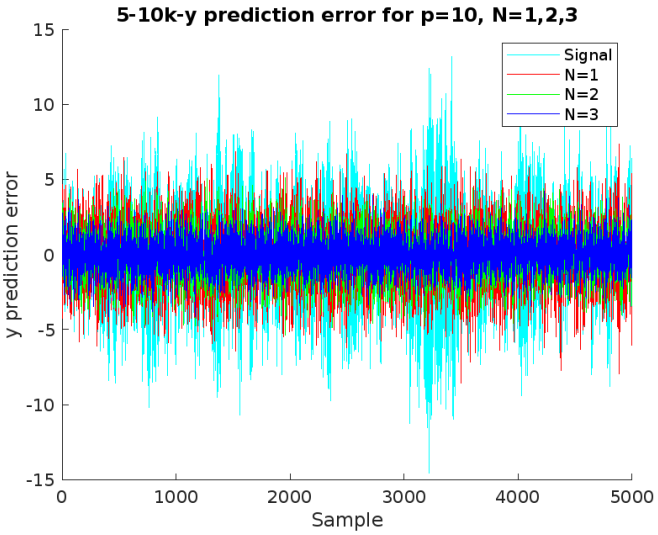


Για κομμάτια μήκους 5.000 του αρχικού σήματος

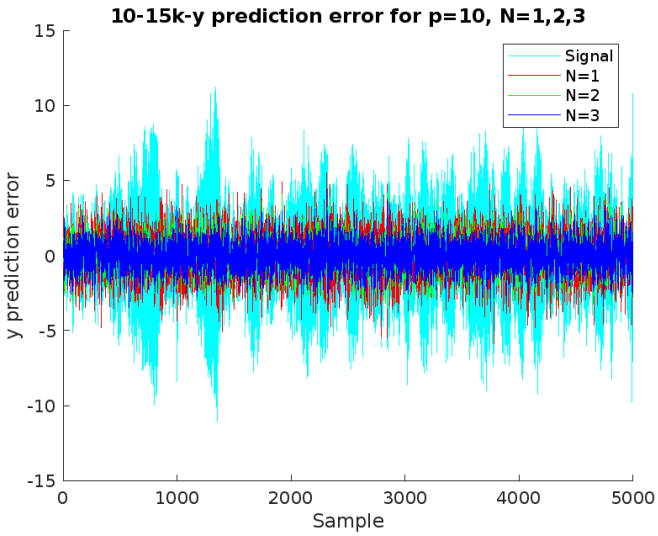
Επειδή «τεμαχίσαμε» το σήμα σε 4 μέρη, τα οποία κατά πάσα πιθανότητα παρουσιάζουν άλλη δομή, ο υπολογισμός των συντελεστών α γίνεται με πιο αποδοτικό τρόπο και καταλήγουμε ανακατασκευή του σήματος. Παραθέτουμε μερικά παραδείγματα, ενώ τα υπόλοιπα βρίσκονται στο παράρτημα.



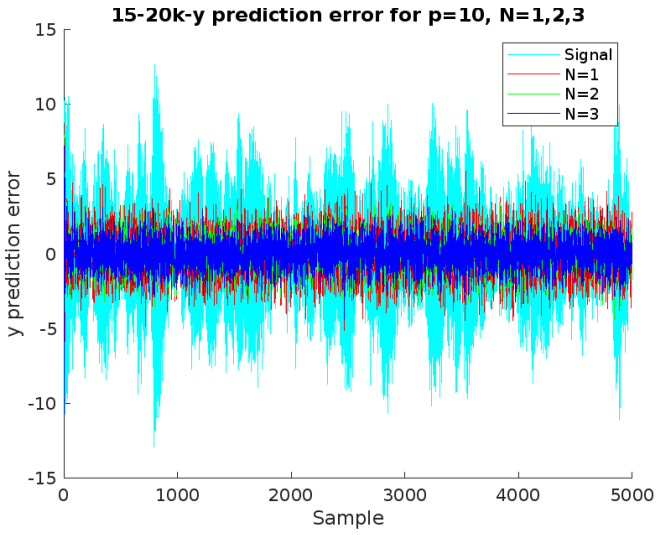
Εικόνα 4 0-5k



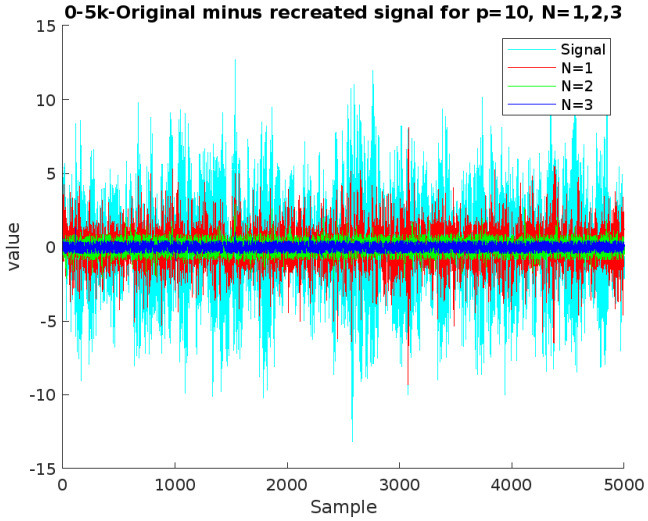
Εικόνα 3 5-10k



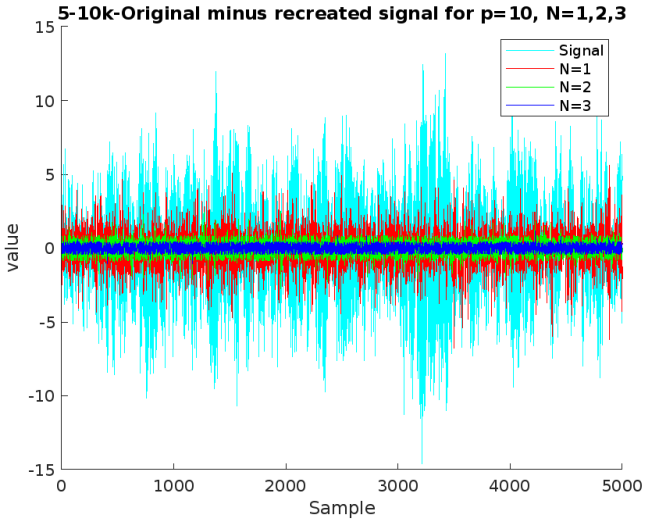
Εικόνα 2 10-15k



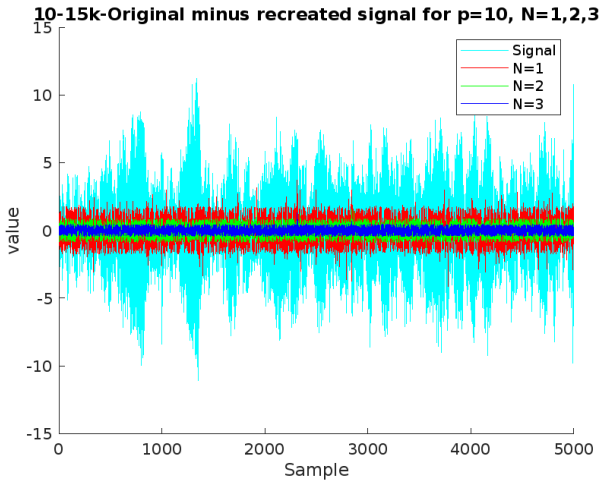
Εικόνα 1 15-20k



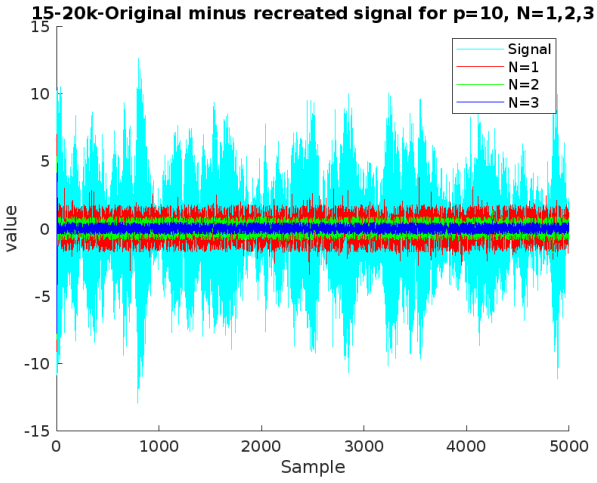
Εικόνα 8 0-5k



Εικόνα 7 5-10k



Εικόνα 6 10-15k



Εικόνα 5 15-20k

Παρατηρούμε πως για συγκεκριμένα κομμάτια της εικόνας, όπως 10-15κ και 15-20κ, το N=2 δίνει απόδοση πολύ κοντινή αυτής για N=3. Επίσης για p=5 βλέπουμε καλύτερη απόδοση σε σχέση με τα ολόκληρα κομμάτια, ειδικά όσον αφορά τις ακραίες τιμές (παράρτημα). Επίσης, οι συντελεστές α είναι μεγαλύτεροι, αφού μιλάμε για μικρότερα κομμάτια κώδικα τα μοτίβα γίνονται πιο συγκεκριμένα και συνεισφέρουν περισσότερο το κάθε κοντινό πίξελ.

Παράρτημα

1) Πιθανότητες Δεύτερης Τάξης

pair_probabilities	pair_symbols	
0.082133	0	0
0.0138	0	17
0.0012	0	34
0.00053333	0	51
0.00013333	0	68
6.6667e-05	0	85
0.00013333	0	102
0.011333	17	0
0.050133	17	17
0.014667	17	34
0.0027333	17	51
0.00093333	17	68
0.0002	17	85
0.00033333	17	102
0.00026667	17	136
0.00013333	17	153
0.00013333	17	170
6.6667e-05	17	187
0.0006	34	0
0.0136	34	17
0.035333	34	34
0.012867	34	51
0.0022667	34	68
0.0013333	34	85
0.00073333	34	102
0.00033333	34	119
0.0004	34	136
0.00046667	34	153
0.00013333	34	170
0.0002	34	187
0.00026667	51	0
0.0028	51	17
0.012467	51	34
0.028667	51	51
0.012267	51	68
0.0024	51	85
0.0011333	51	102
0.00053333	51	119
0.0002	51	136
0.00046667	51	153
0.00026667	51	170
6.6667e-05	51	187
0.00013333	51	204
6.6667e-05	51	221
6.6667e-05	51	238
0.0012	68	17
0.0022667	68	34
0.012533	68	51
0.041333	68	68

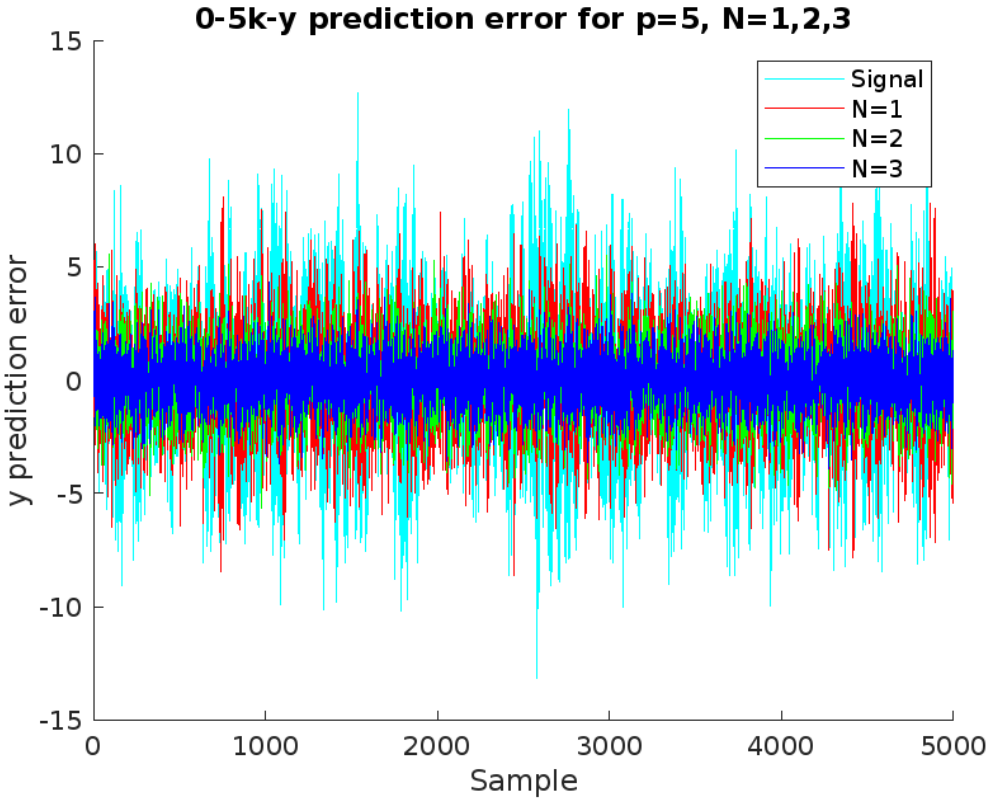
0.014333	68	85
0.0025333	68	102
0.0012	68	119
0.0008	68	136
0.00073333	68	153
0.00026667	68	170
0.00013333	68	187
0.0002	68	204
0.00026667	68	221
0.00013333	68	238
0.00026667	85	17
0.0018	85	34
0.0036	85	51
0.013067	85	68
0.0498	85	85
0.015933	85	102
0.0021333	85	119
0.0012	85	136
0.00053333	85	153
0.00033333	85	170
0.00033333	85	187
0.00013333	85	204
0.0002	85	221
0.00013333	85	238
6.6667e-05	102	17
0.00033333	102	34
0.0016667	102	51
0.0032667	102	68
0.016467	102	85
0.0702	102	102
0.017133	102	119
0.0028	102	136
0.0012667	102	153
0.00073333	102	170
0.00073333	102	187
0.0002	102	204
6.6667e-05	102	238
6.6667e-05	102	255
0.0002	119	34
0.00033333	119	51
0.0019333	119	68
0.0047333	119	85
0.015	119	102
0.049267	119	119
0.014333	119	136
0.0029333	119	153
0.0006	119	170
0.00046667	119	187
0.0002	119	204
6.6667e-05	119	221
0.00026667	119	238
6.6667e-05	119	255
6.6667e-05	136	34
0.0002	136	51
0.0004	136	68
0.0012667	136	85
0.0043333	136	102
0.016067	136	119

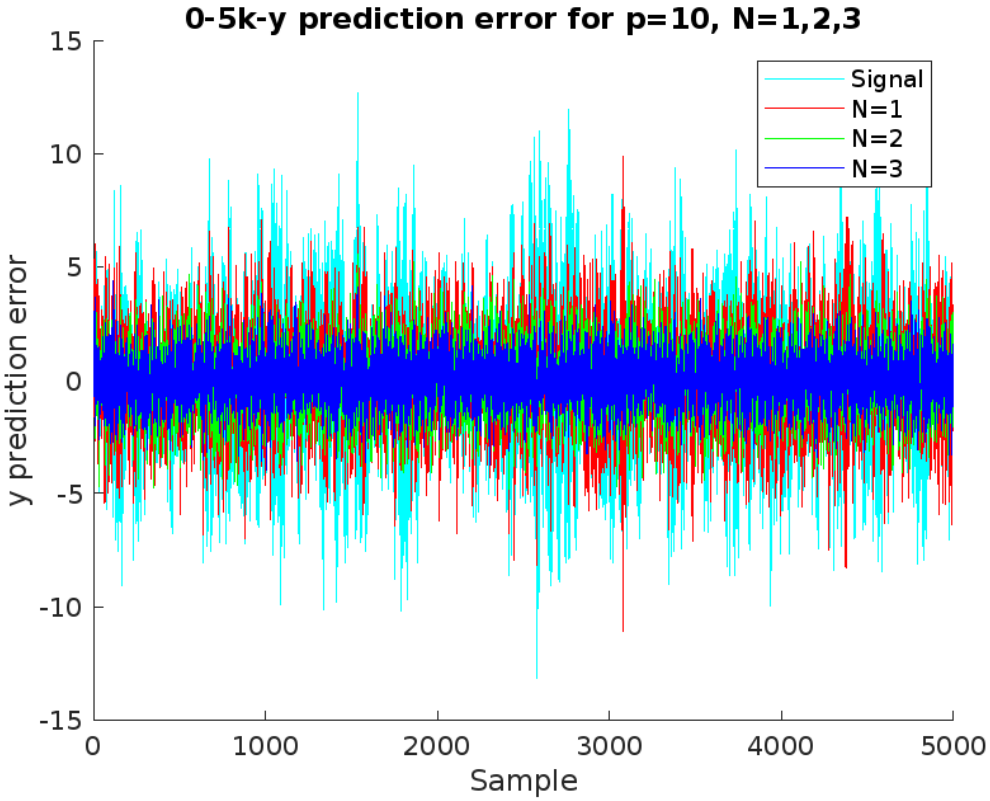
0.056467	136	136
0.0128	136	153
0.0019333	136	170
0.0010667	136	187
0.0004	136	204
6.6667e-05	136	221
0.00013333	136	238
0.0002	136	255
6.6667e-05	153	51
6.6667e-05	153	68
0.0004	153	85
0.0008	153	102
0.0034667	153	119
0.0176	153	136
0.034533	153	153
0.0073333	153	170
0.0011333	153	187
0.0008	153	204
0.00013333	153	221
6.6667e-05	153	255
6.6667e-05	170	34
6.6667e-05	170	68
0.0004	170	85
6.6667e-05	170	102
0.00033333	170	119
0.0022	170	136
0.010667	170	153
0.016733	170	170
0.0059333	170	187
0.0012667	170	204
0.00033333	170	221
0.00013333	170	238
0.00013333	170	255
0.0002	187	119
0.00066667	187	136
0.0019333	187	153
0.0079333	187	170
0.013733	187	187
0.0063333	187	204
0.0016	187	221
0.00046667	187	238
0.00013333	187	255
0.00013333	204	102
0.00066667	204	136
0.001	204	153
0.0024	204	170
0.0072667	204	187
0.0164	204	204
0.0046	204	221
0.00093333	204	238
0.00026667	204	255
6.6667e-05	221	119
0.0002	221	153
0.00066667	221	170
0.0014	221	187
0.0066667	221	204
0.0132	221	221
0.0032	221	238

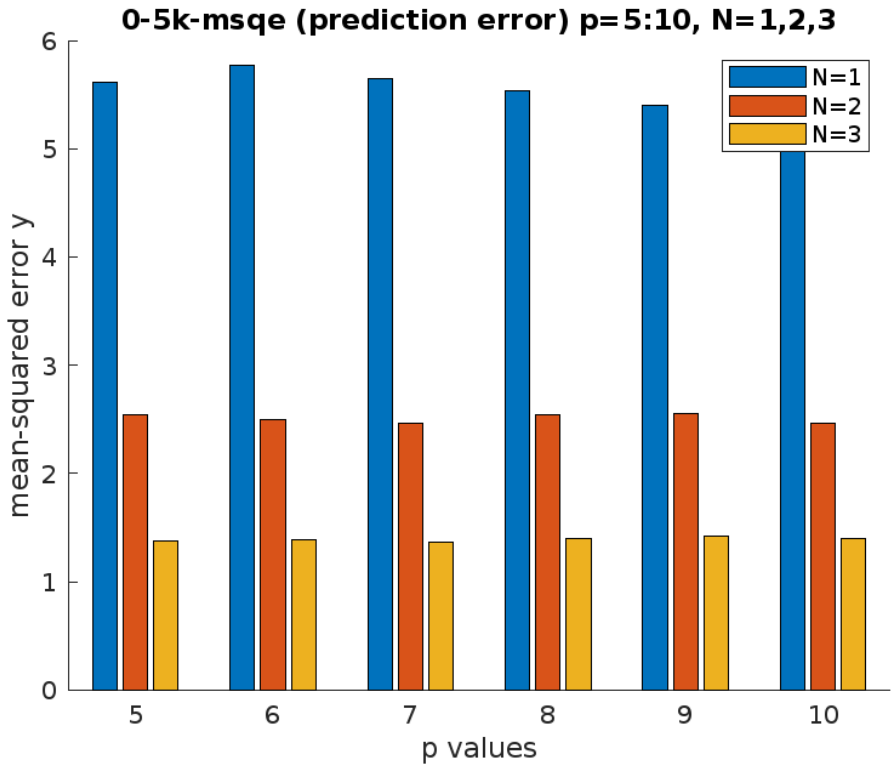
```
0.00026667         221    255
6.6667e-05         238     85
6.6667e-05         238    119
0.00013333         238    153
0.00033333         238    187
0.00093333         238    204
    0.0054         238    221
  0.014267         238    238
0.00073333         238    255
6.6667e-05         255    153
    0.0002         255    221
     0.003         255    238
0.00053333         255    255
```

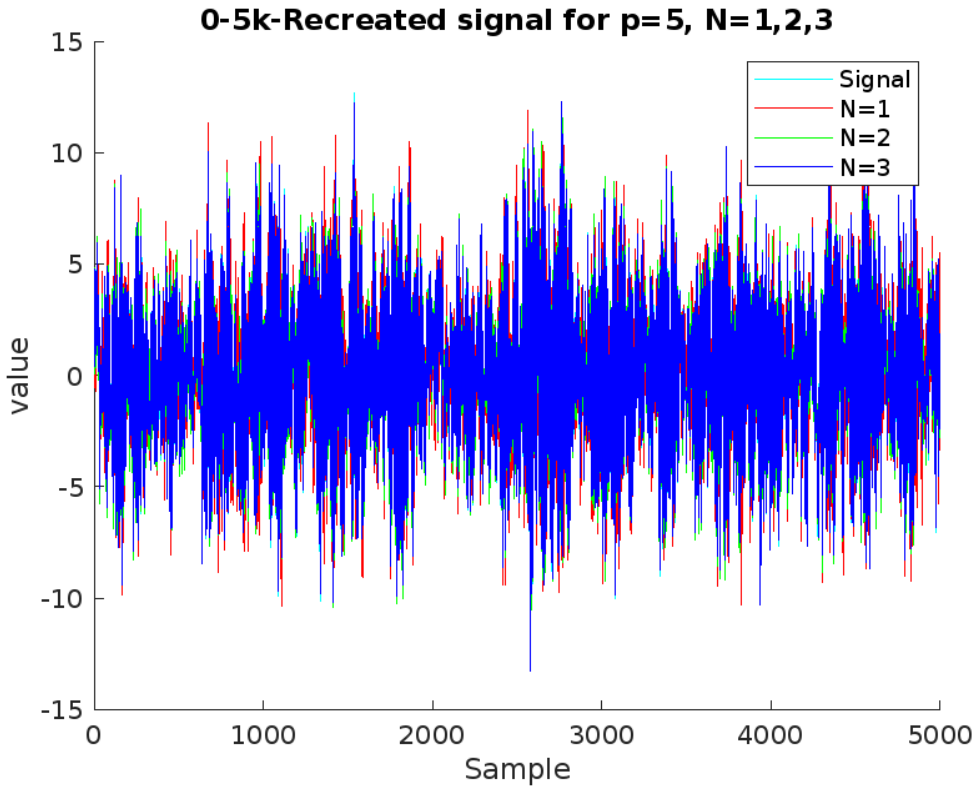
2) Λοιπές Εικόνες DPCM

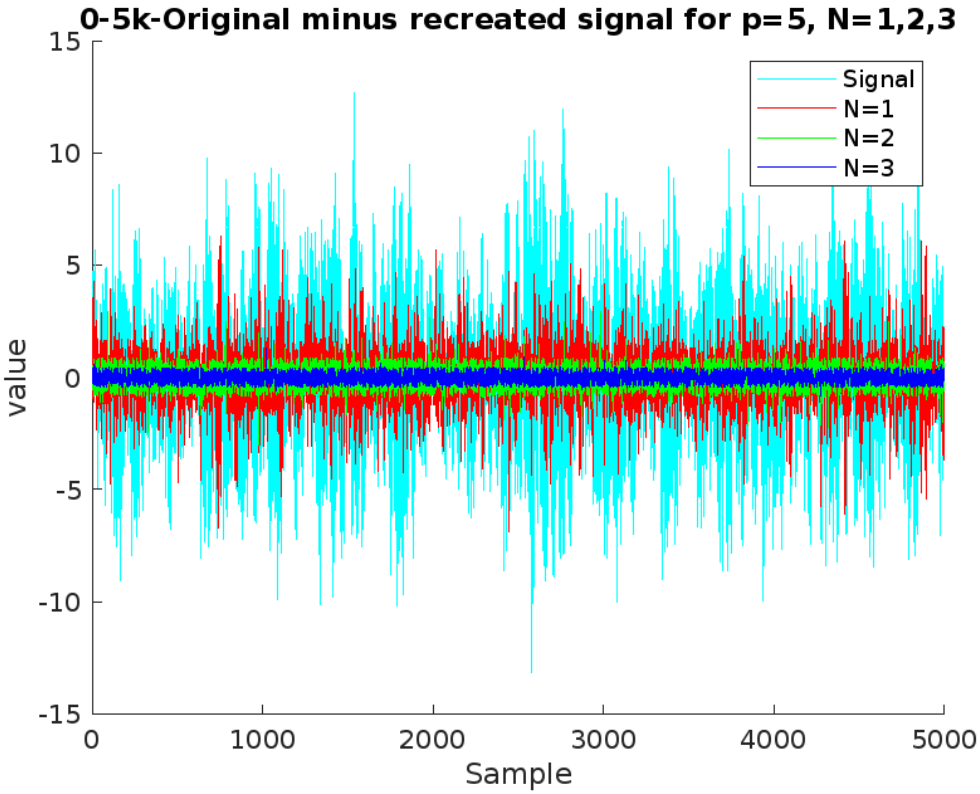
0-5k

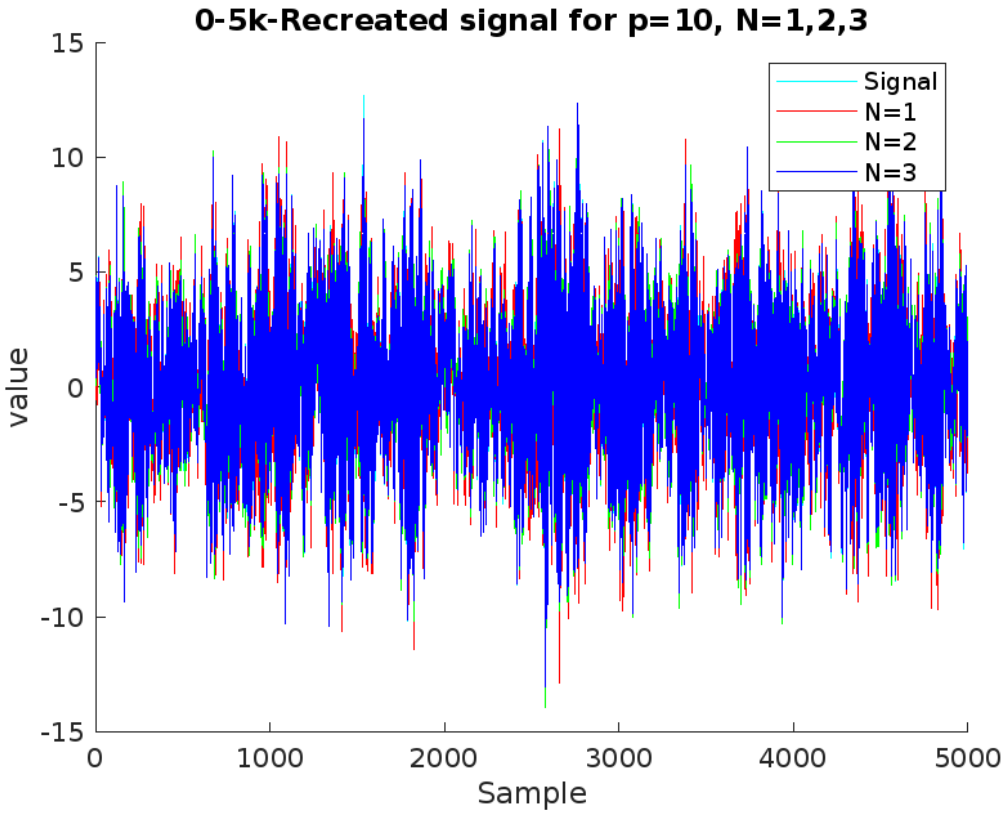


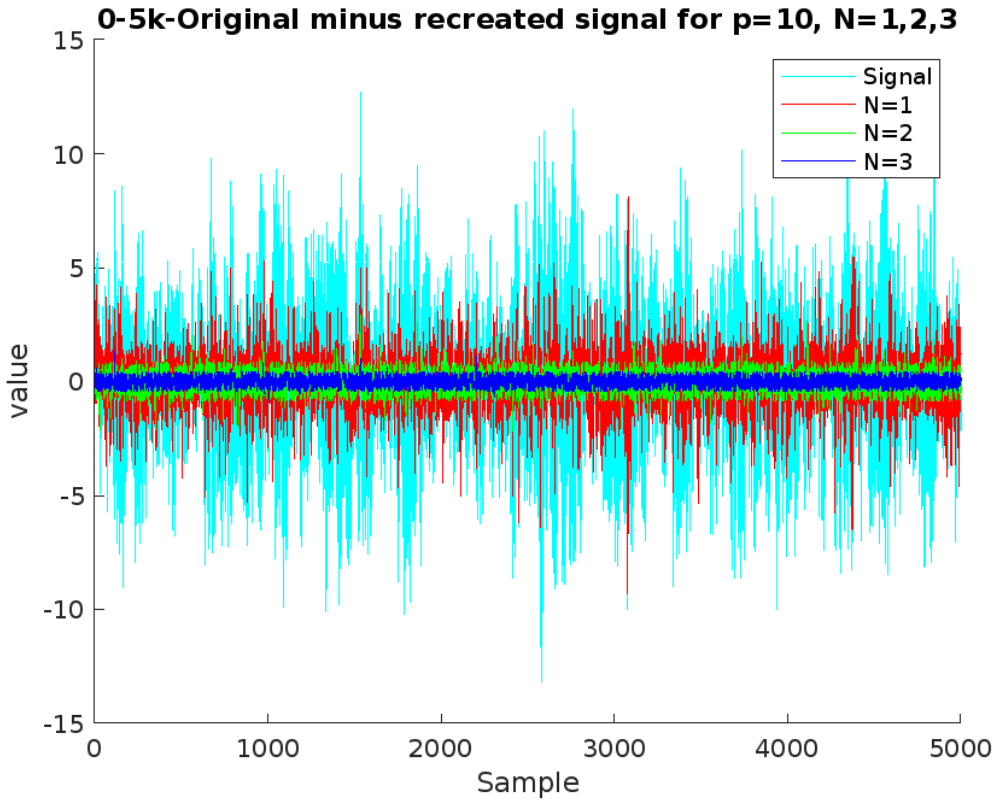



0-5k-msqe (prediction error) p=5:10, N=1,2,3
mean-squared error y
p values
N=1
N=2
N=3
0
1
2
3
4
5
6
5
6
7
8
9
10


0-5k-Recreated signal for p=5, N=1,2,3
value
Sample
Signal
N=1
N=2
N=3
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

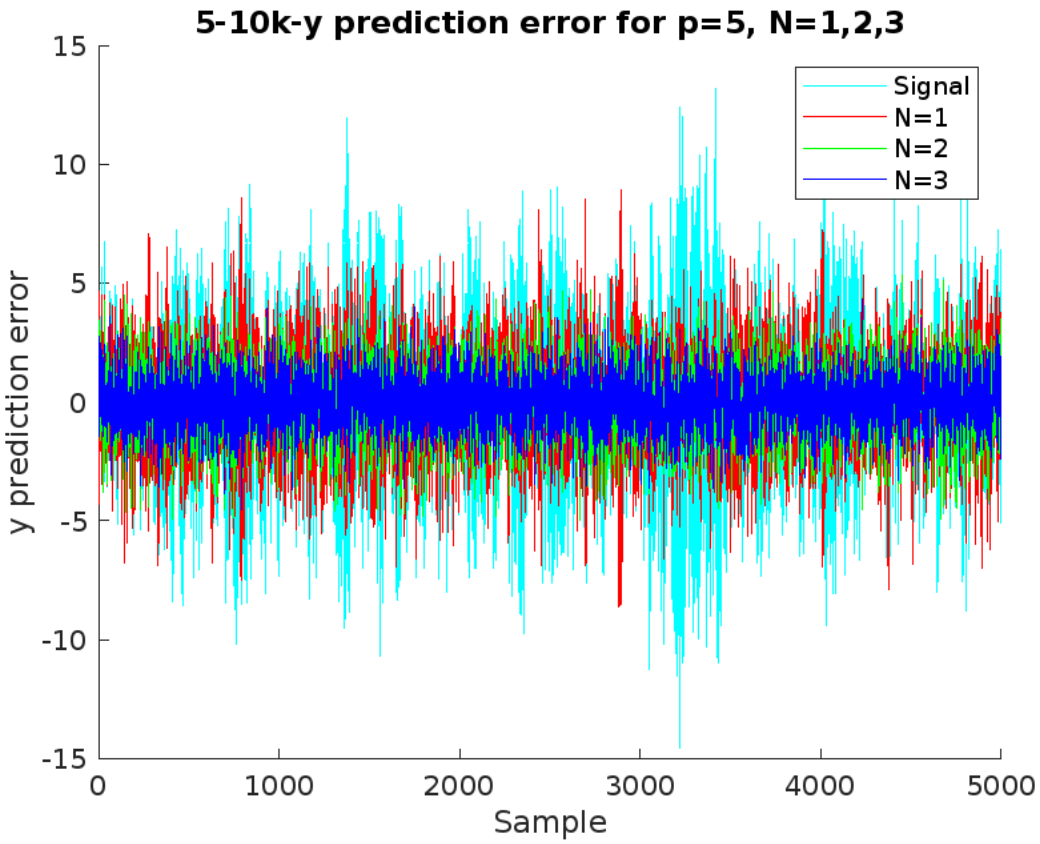

0-5k-Original minus recreated signal for p=5, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

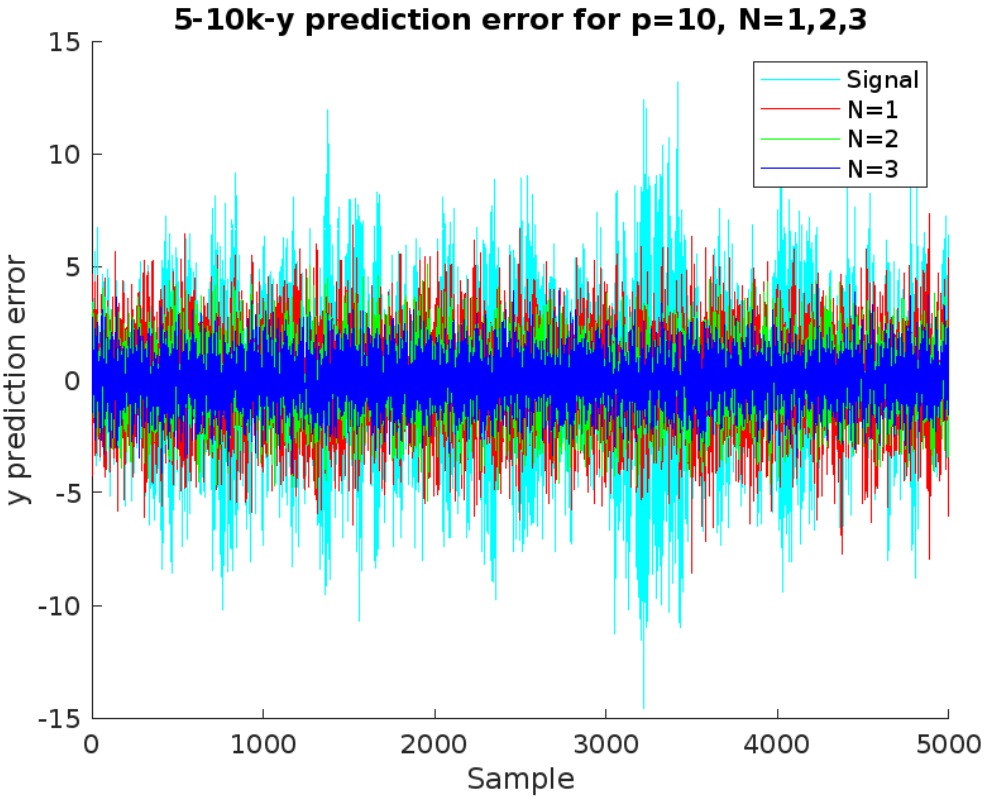

0-5k-Recreated signal for p=10, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

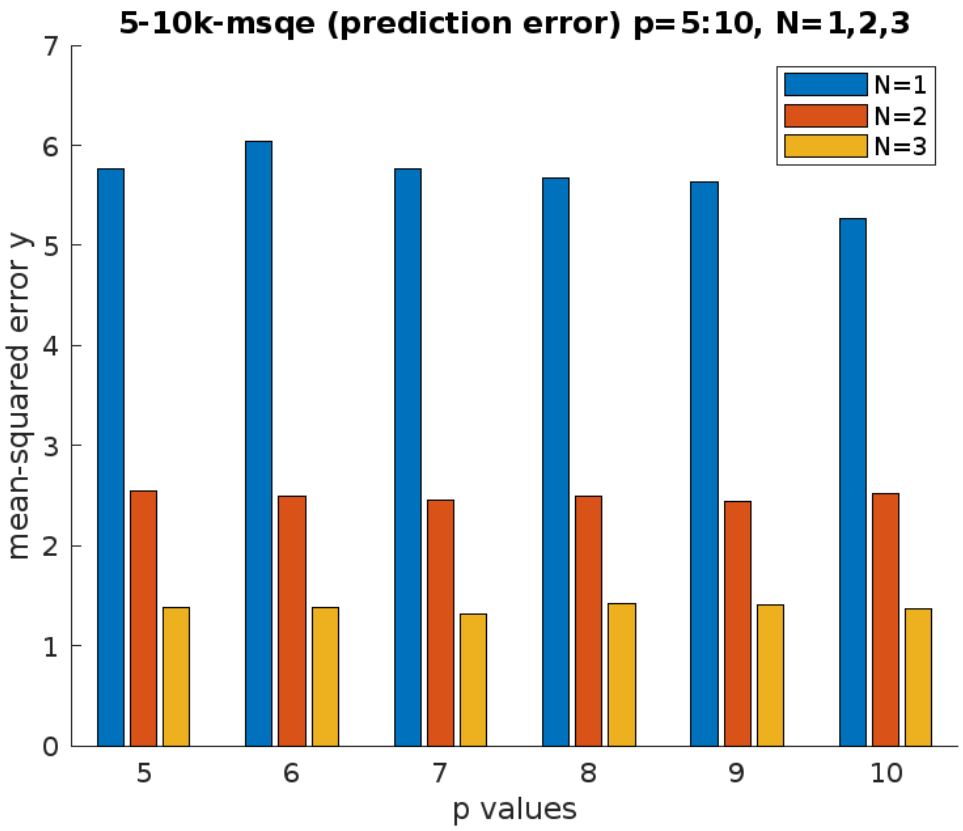


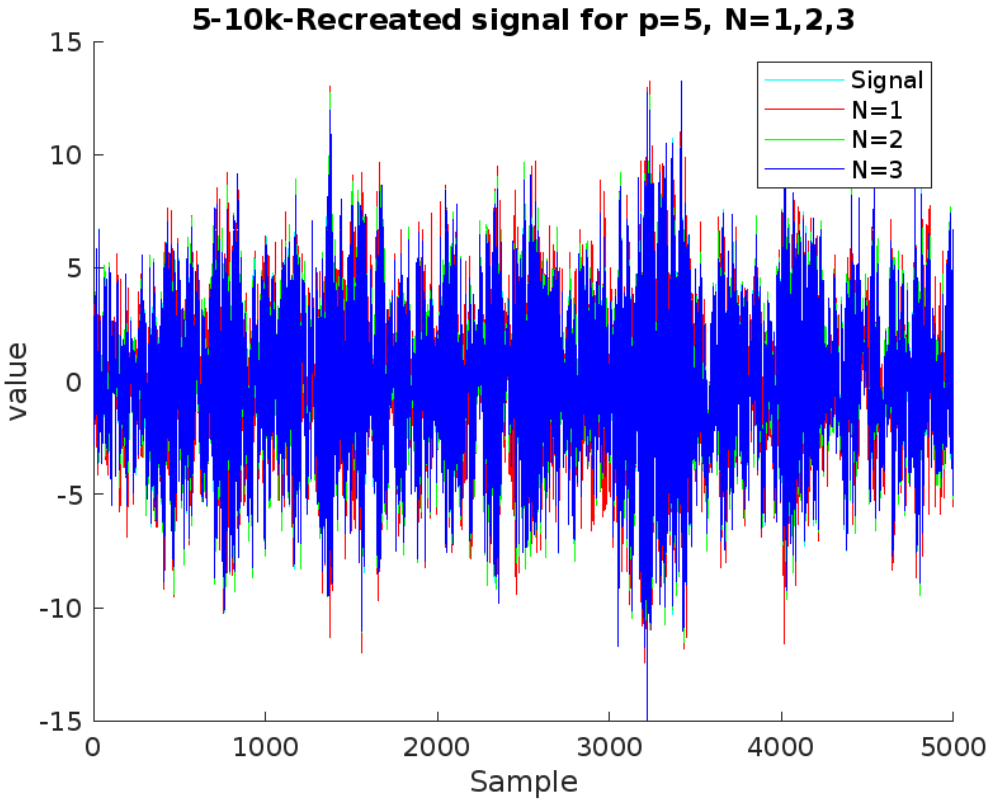

```
a_coeff =

  6x1 cell array

    {[                                    1.4297 -1.5703 1.2891 -0.3516 0.0234]}
    {[                                1.4297 -1.5703 1.2734 -0.3516 0.0078 0.0078]}
    {[                            1.4297 -1.5703 1.2891 -0.3672 0.0234 -0.0078 0.0078]}
    {[                        1.4297 -1.5703 1.2891 -0.3672 0.0391 -0.0234 0.0234 -0.0078]}
    {[                    1.4297 -1.5703 1.2891 -0.3672 0.0234 0.0078 -0.0078 0.0234 -0.0234]}
    {[1.4297 -1.5703 1.2891 -0.3672 0.0234 0.0078 -0.0234 0.0234 -0.0234 0.0078]}
```

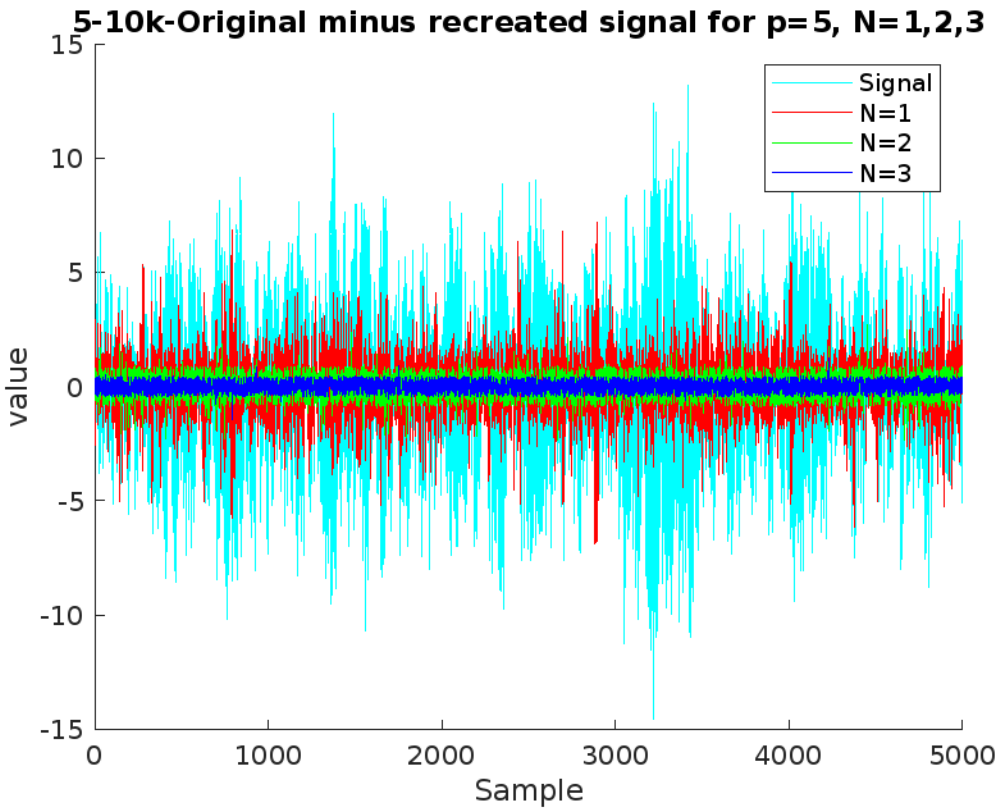
5-10k

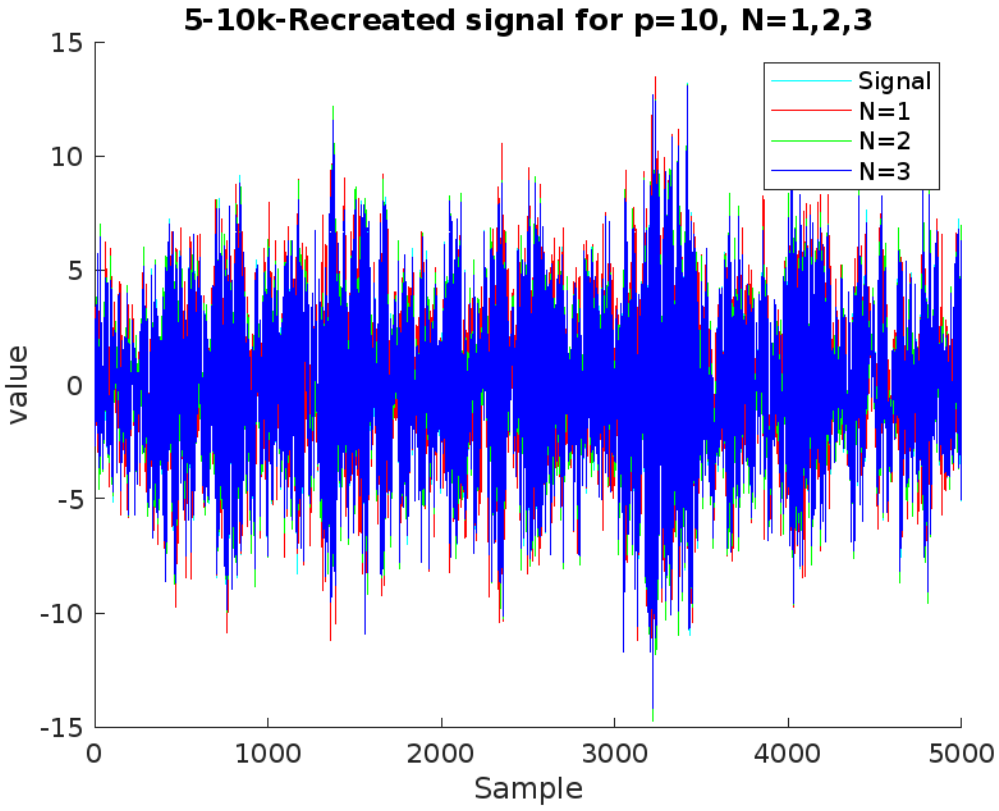

5-10k-y prediction error for p=5, N=1,2,3
Signal
N=1
N=2
N=3
y prediction error
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

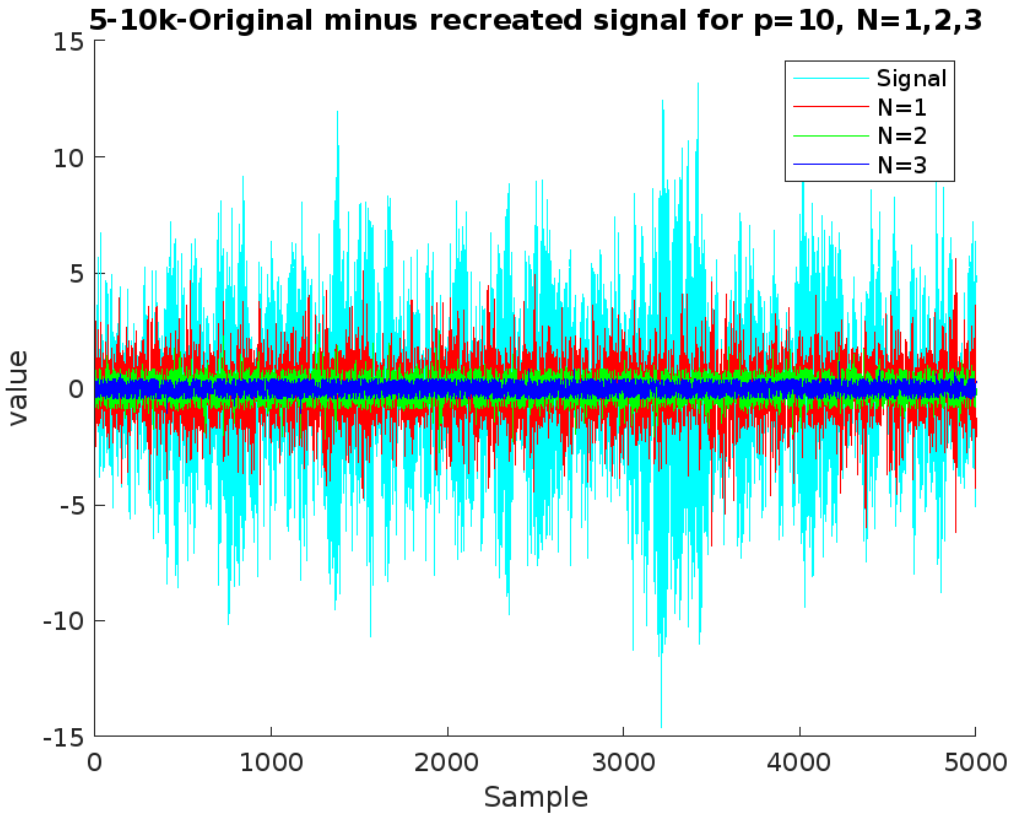

5-10k-y prediction error for p=10, N=1,2,3
Signal
N=1
N=2
N=3
y prediction error
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000


5-10k-msqe (prediction error) p=5:10, N=1,2,3
N=1
N=2
N=3
mean-squared error y
p values
0
1
2
3
4
5
6
7
5
6
7
8
9
10


5-10k-Recreated signal for p=5, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

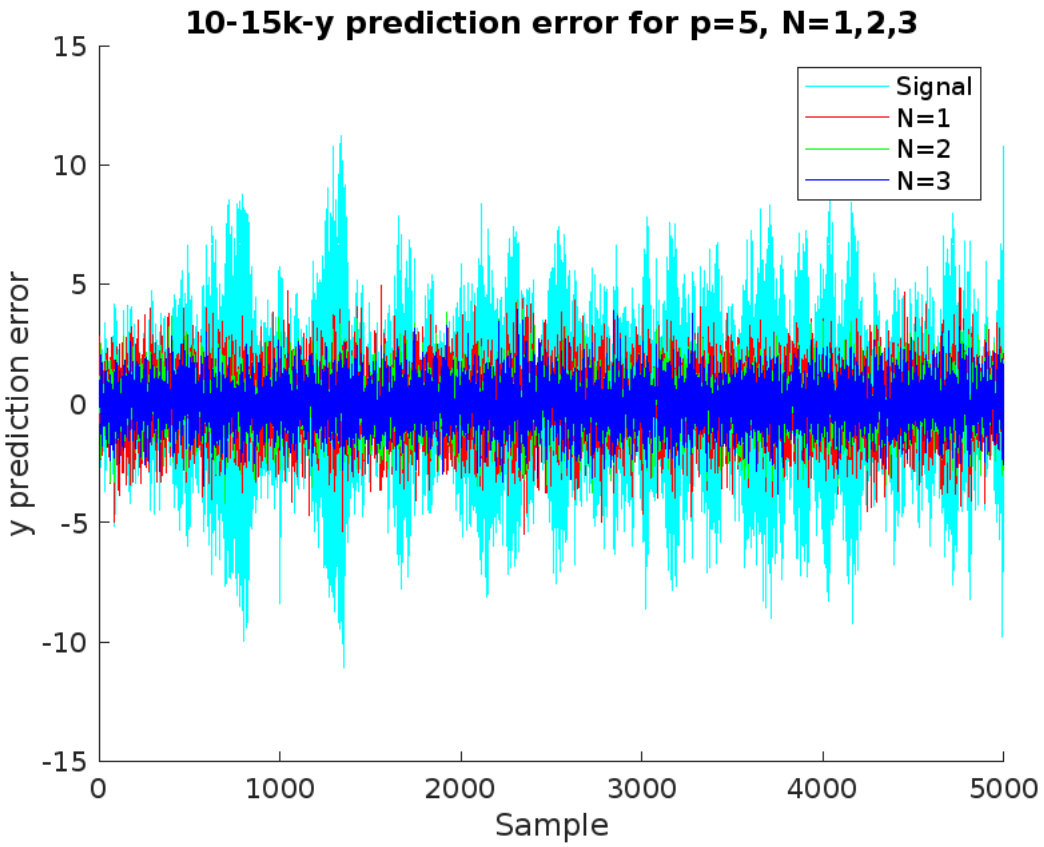

5-10k-Original minus recreated signal for p=5, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

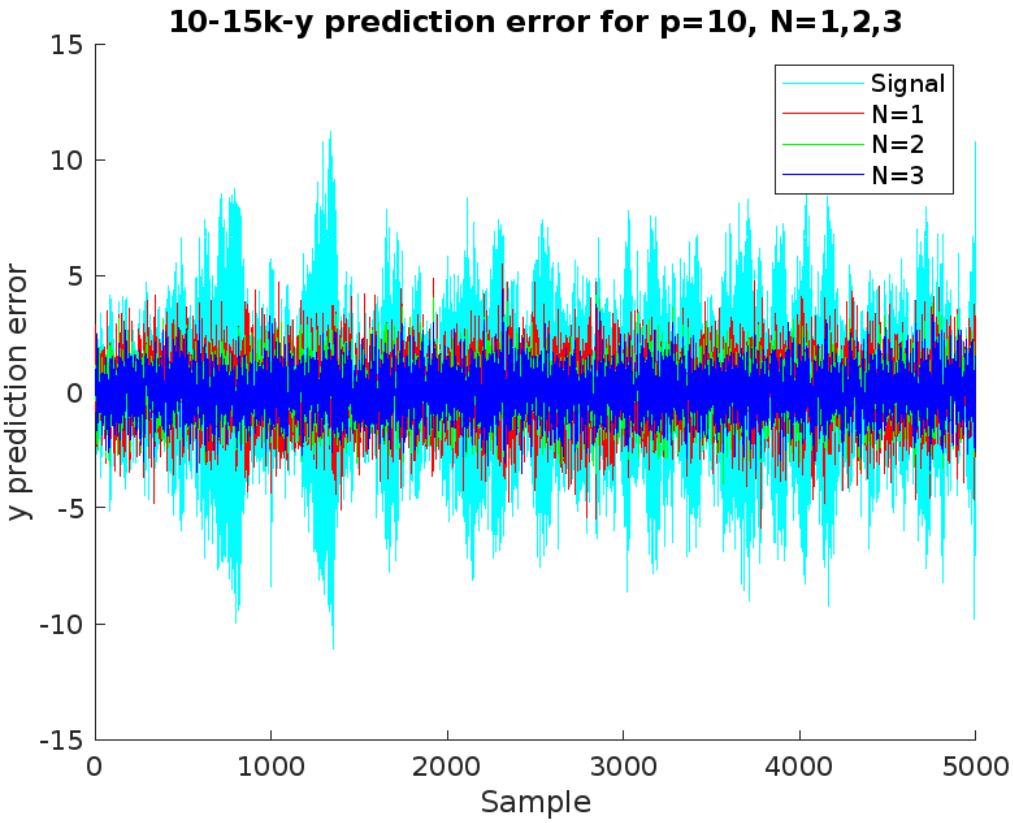

5-10k-Recreated signal for p=10, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

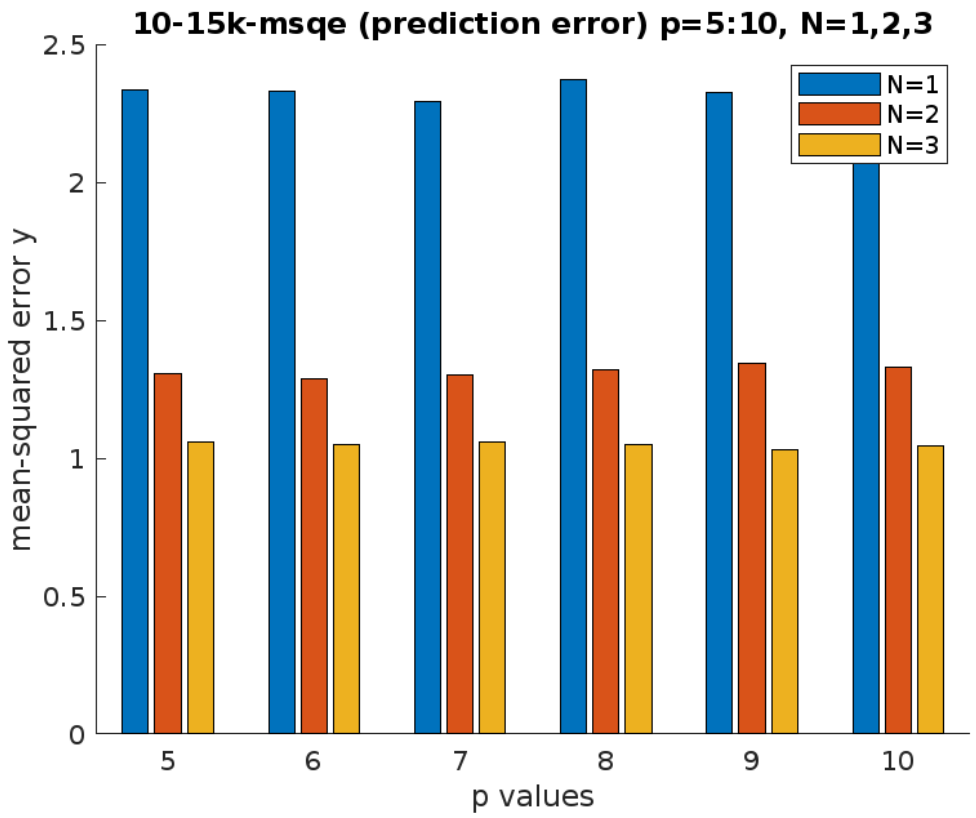


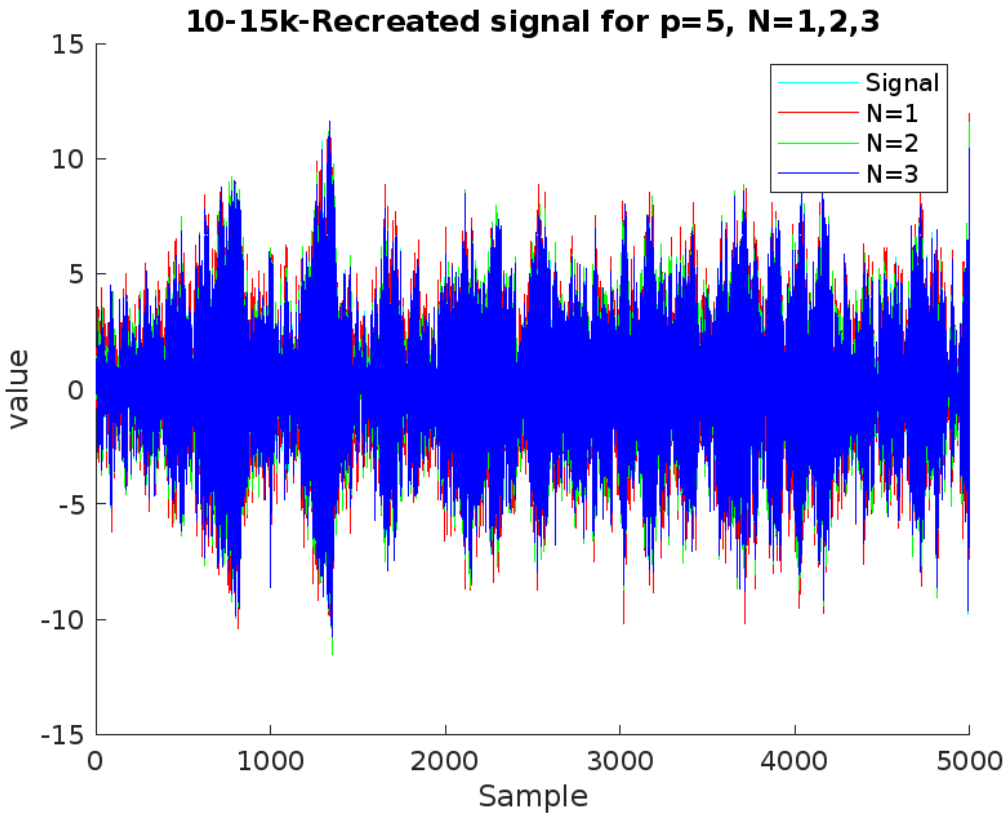

```
a_coeff =

  6x1 cell array

    {[                                    1.4453 -1.5859 1.2578 -0.3359 -0.0078]}
    {[                              1.4453 -1.5859 1.2734 -0.3516 -0.0078 -0.0078]}
    {[                        1.4453 -1.5859 1.2891 -0.3984 0.0547 -0.0547 0.0234]}
    {[                  1.4453 -1.5859 1.2891 -0.3984 0.0547 -0.0703 0.0391 -0.0078]}
    {[           1.4453 -1.5859 1.2891 -0.3984 0.0547 -0.0391 0.0078 0.0234 -0.0078]}
    {[1.4453 -1.5859 1.2891 -0.3984 0.0547 -0.0391 0.0234 0.0078 -0.0078 -0.0078]}
```

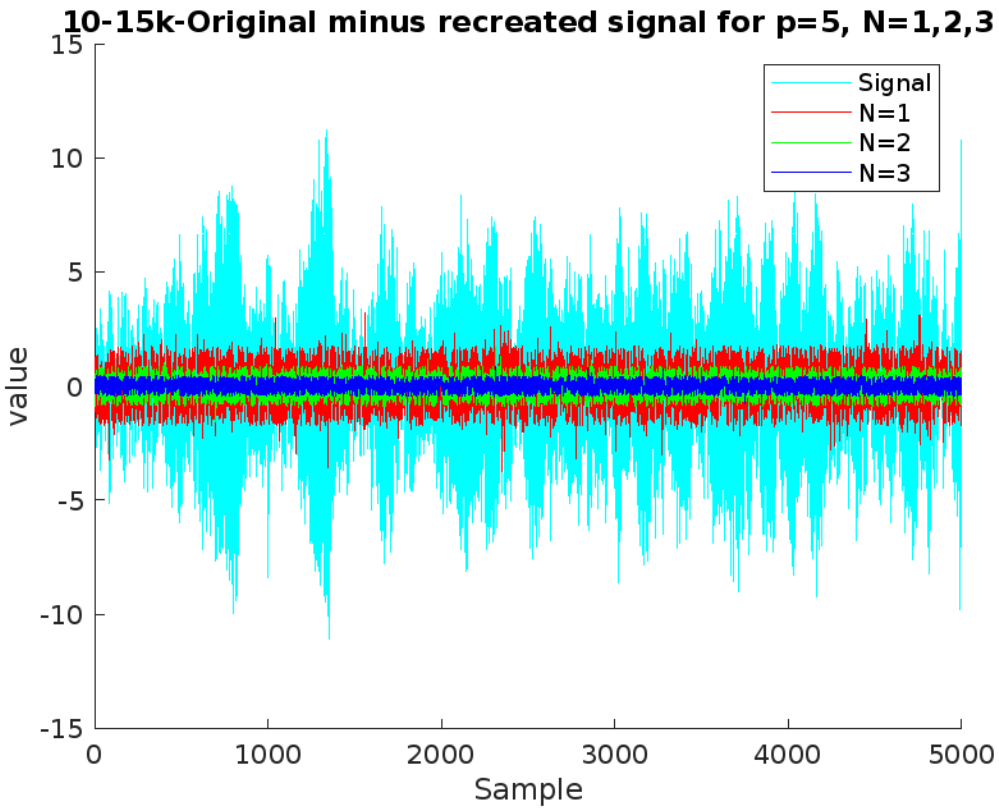
10-15k

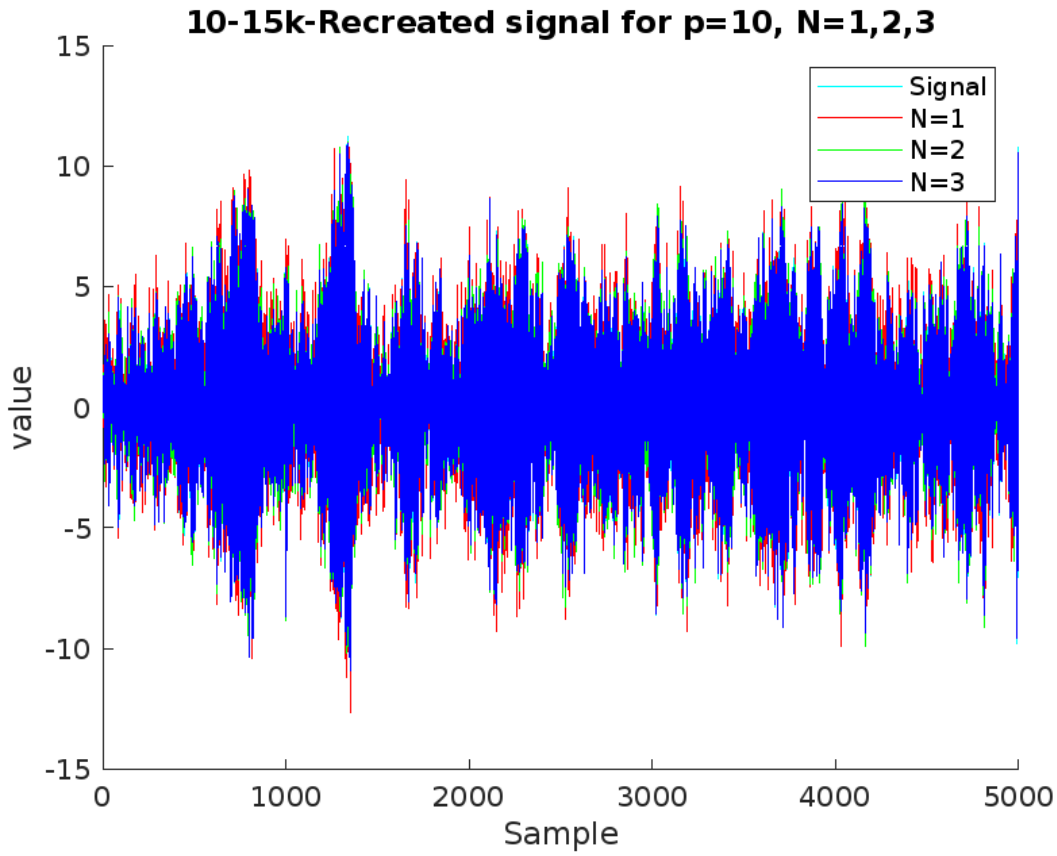

10-15k-y prediction error for p=5, N=1,2,3
Signal
N=1
N=2
N=3
y prediction error
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

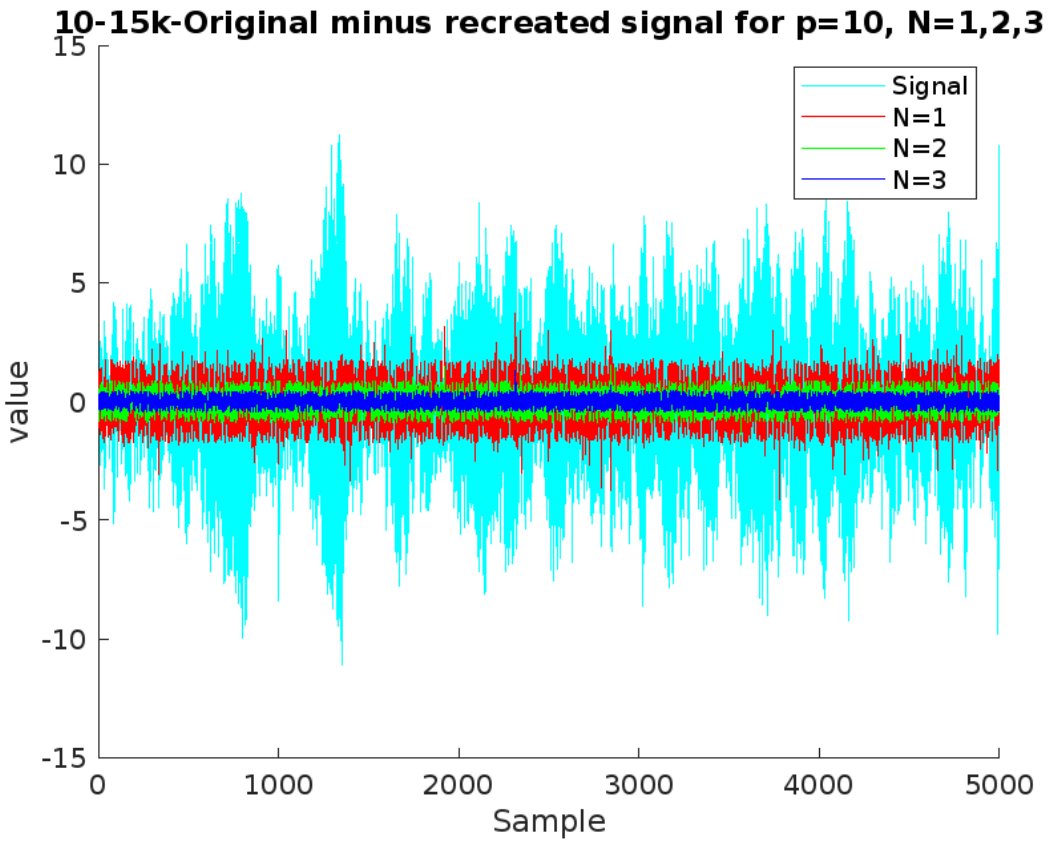

10-15k-y prediction error for p=10, N=1,2,3
Signal
N=1
N=2
N=3
y prediction error
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000


10-15k-msqe (prediction error) p=5:10, N=1,2,3
mean-squared error y
p values
N=1
N=2
N=3
0
0.5
1
1.5
2
2.5
5
6
7
8
9
10


10-15k-Recreated signal for p=5, N=1,2,3
value
Sample
Signal
N=1
N=2
N=3
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

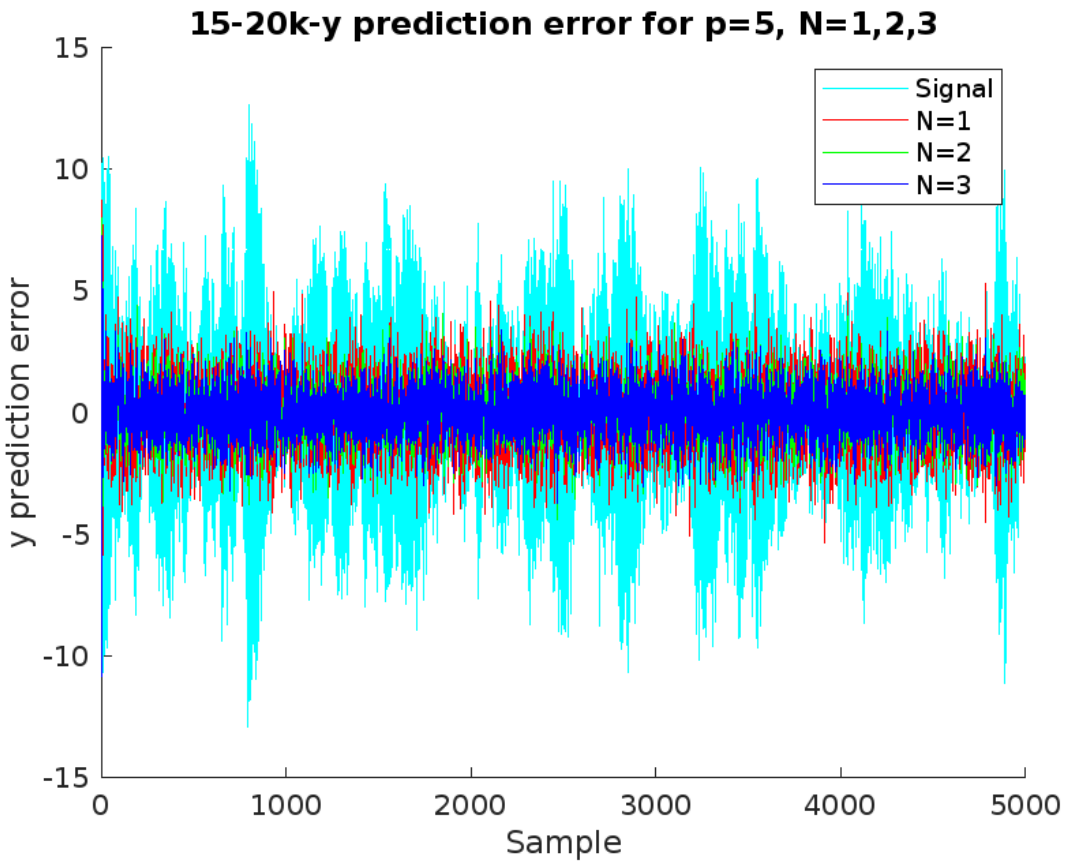

10-15k-Original minus recreated signal for p=5, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

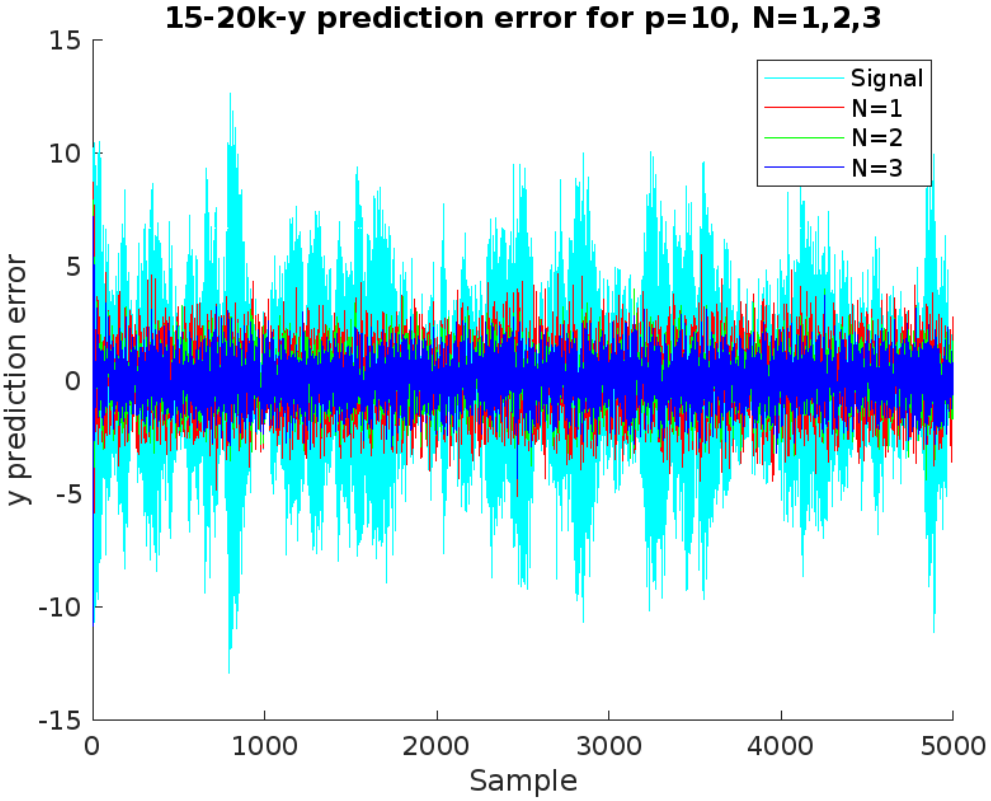

10-15k-Recreated signal for p=10, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

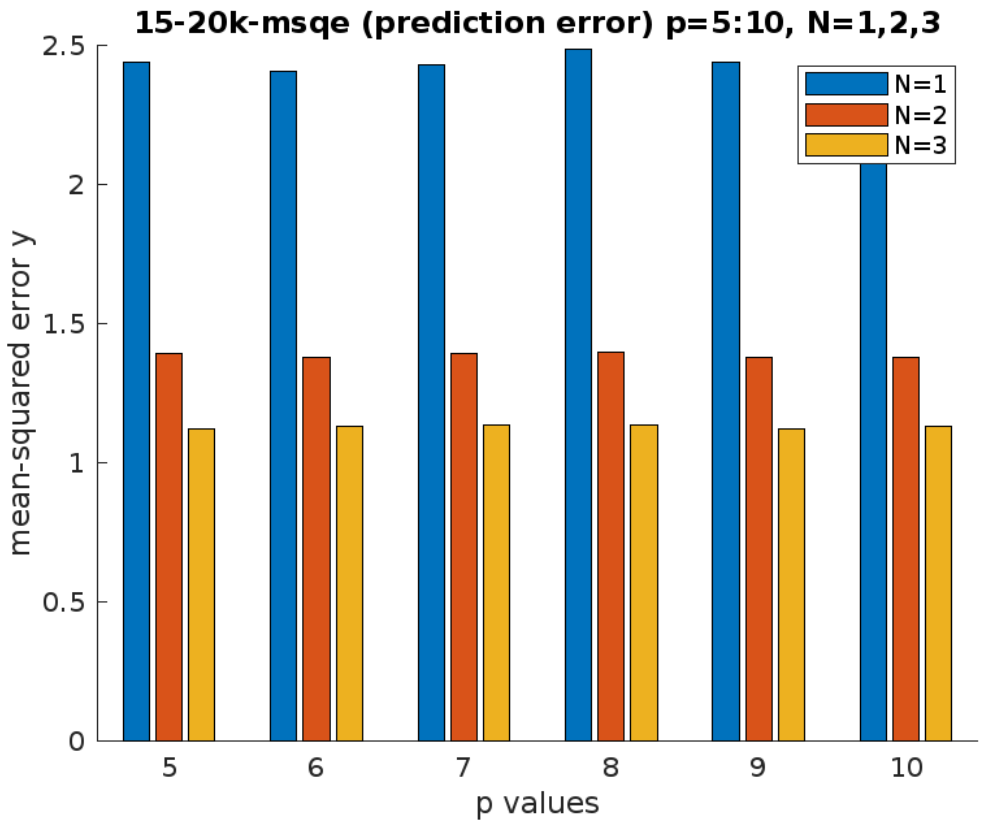


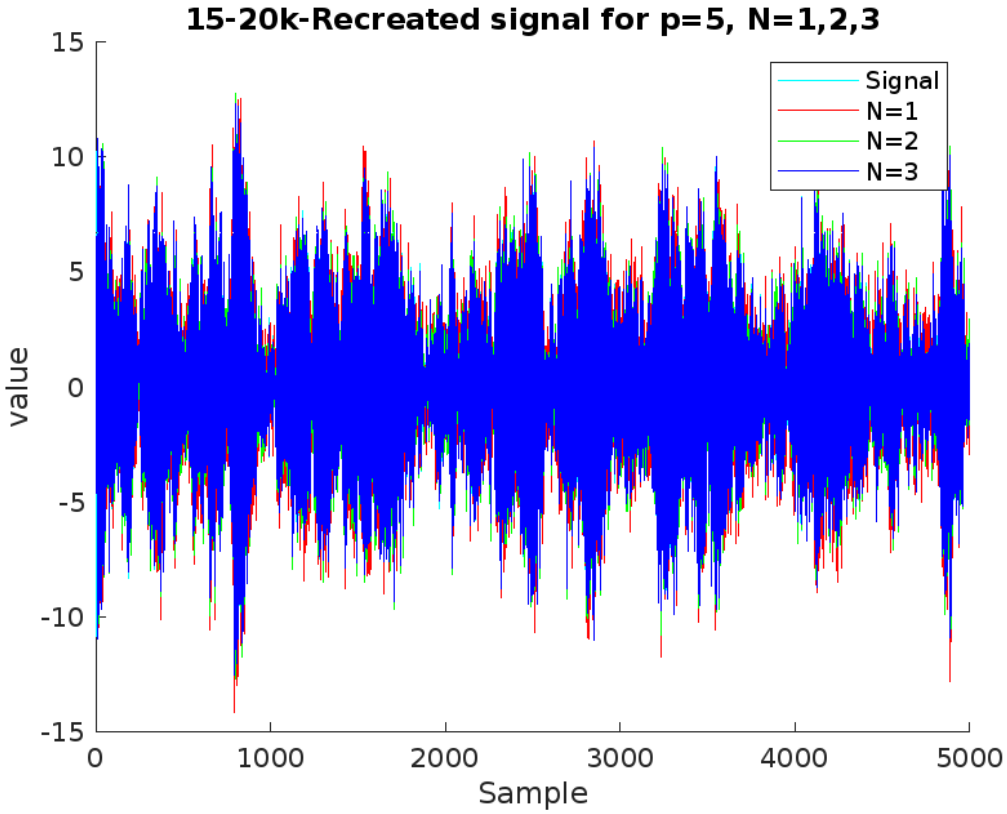

```
6x1 cell array

  {[                                    0.5234 -0.7109 0.2578 -0.7109 -0.0859]}
  {[                               0.5234 -0.7109 0.2578 -0.7109 -0.1016 0.0078]}
  {[                          0.5234 -0.7109 0.2422 -0.7109 -0.1172 0.0234 -0.0234]}
  {[                     0.5234 -0.7109 0.2422 -0.7266 -0.1016 0.0078 -0.0234 -0.0078]}
  {[            0.5234 -0.7109 0.2422 -0.7266 -0.1172 0.0078 -0.0234 -0.0078 -0.0078]}
  {[0.5234 -0.7109 0.2422 -0.7266 -0.1172 0.0078 -0.0234 -0.0078 -0.0078 0.0078]}
```

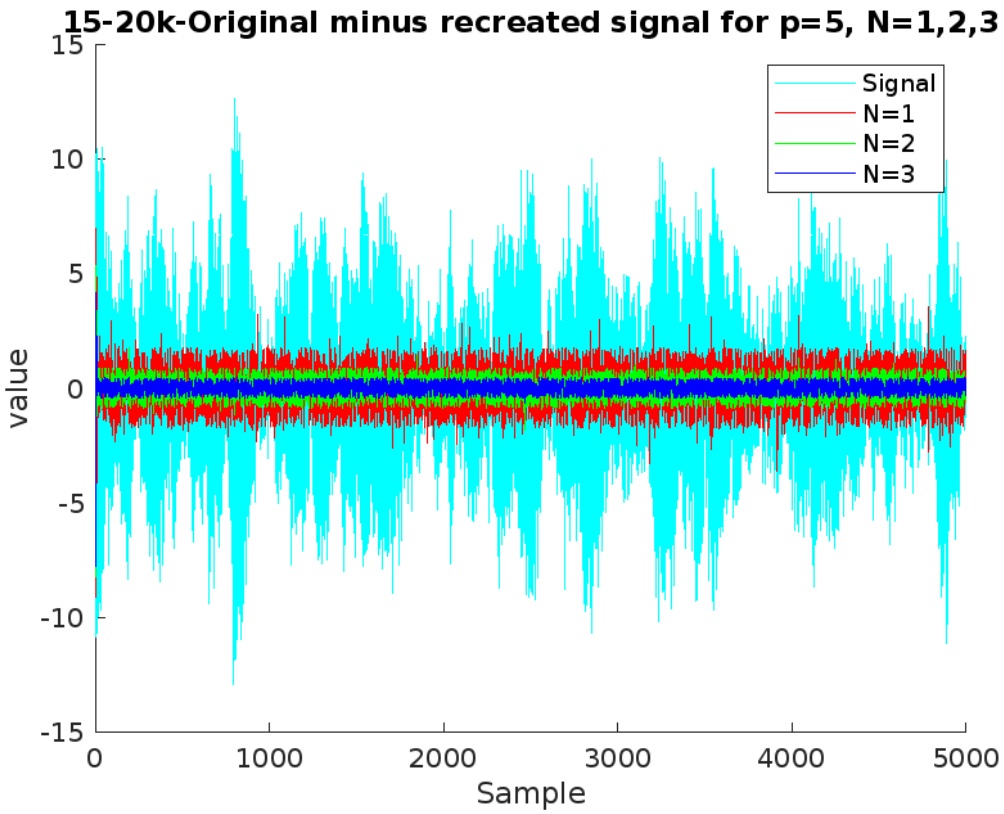
15-20k

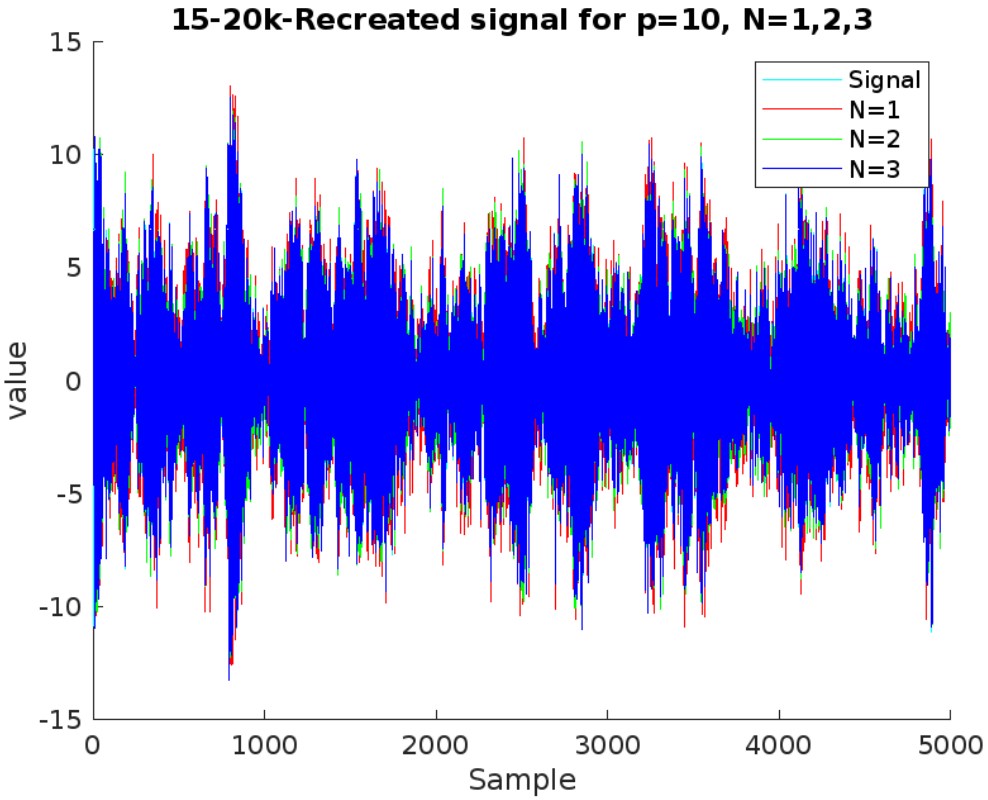

15-20k-y prediction error for p=5, N=1,2,3
Signal
N=1
N=2
N=3
y prediction error
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000

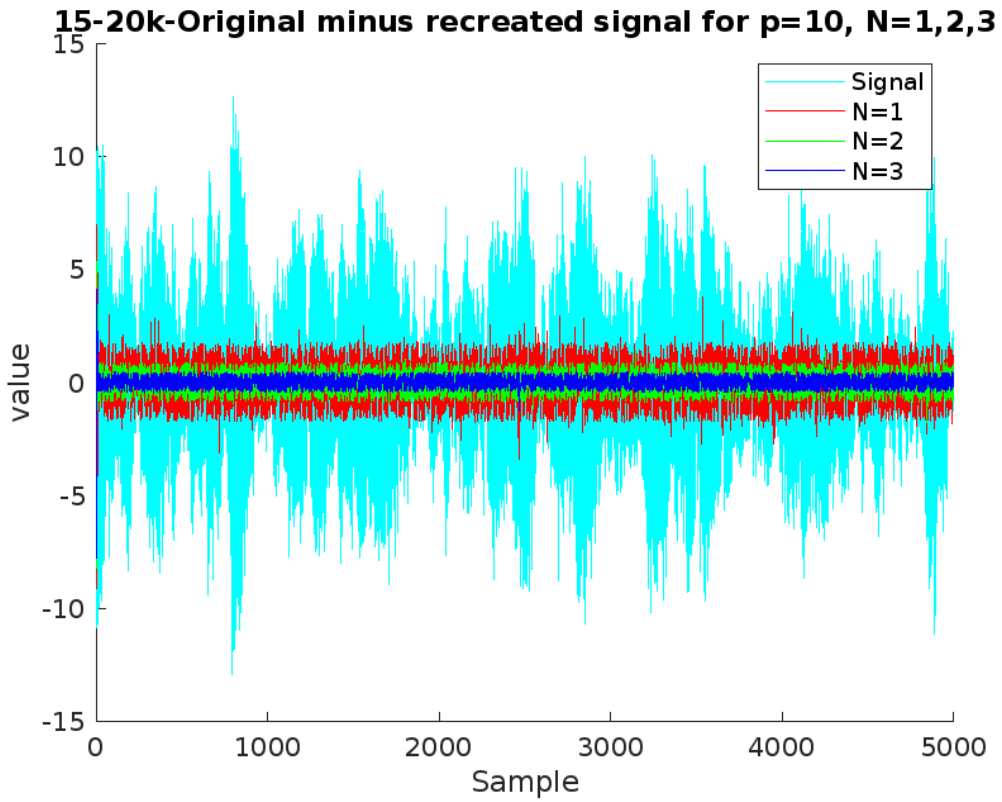

15-20k-y prediction error for p=10, N=1,2,3
Signal
N=1
N=2
N=3
y prediction error
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000


15-20k-msqe (prediction error) p=5:10, N=1,2,3
mean-squared error y
p values
N=1
N=2
N=3
0
0.5
1
1.5
2
2.5
5
6
7
8
9
10


15-20k-Recreated signal for p=5, N=1,2,3
value
Sample
Signal
N=1
N=2
N=3
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000


15-20k-Original minus recreated signal for p=5, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000


15-20k-Recreated signal for p=10, N=1,2,3
Signal
N=1
N=2
N=3
value
Sample
15
10
5
0
-5
-10
-15
0
1000
2000
3000
4000
5000




```
a_coeff =

  6x1 cell array

    {[                                    0.5078 -0.6953 0.2422 -0.7109 -0.1172]}
    {[                             0.5078 -0.6953 0.2422 -0.7109 -0.1172 -0.0078]}
    {[                      0.5078 -0.6953 0.2266 -0.6953 -0.1328 0.0078 -0.0234]}
    {[               0.5078 -0.6953 0.2266 -0.6953 -0.1328 0.0078 -0.0234 0.0078]}
    {[        0.5078 -0.6953 0.2266 -0.6953 -0.1328 0.0078 -0.0234 0.0078 -0.0078]}
    {[0.5078 -0.6953 0.2266 -0.6953 -0.1328 0.0078 -0.0234 0.0078 -0.0078 -0.0078]}
```

3) Κώδικας

part_a.m

```
%TODO: Sxoliasmos 1b, 2b, ++?
%%% Part A %%%
clear;
clc;

%% Question 1.a
image = imread('parrot.png');
[symbol_counts, symbols] = groupcounts(image(:));
probabilities = symbol_counts./numel(image);

disp('1.a) Probabilities: ');
disp(table(probabilities, symbols));

%% Question 1.b
huff_dict = huffmandict(symbols, probabilities);
entropy = -sum(arrayfun(@(x) x*log2(x), probabilities));

dict_lengths = cellfun('length', huff_dict(:,2));
average_length = sum(probabilities .* dict_lengths);

efficiency = entropy/average_length;

disp(['1.b.i) Entropy: ' num2str(entropy)]);
disp(['1.b.ii) Average Length: ' num2str(average_length)]);
disp(['1.b.iii) Efficiency: ' num2str(efficiency)]);

%% Question 2.a
pairs = image(:);
pairs = [pairs(1:2:end), pairs(2:2:end)];

[pair_symbol_counts, pair_symbols] = groupcounts(pairs);
pair_probabilities = pair_symbol_counts./(numel(pairs)/2);
pair_symbols = [pair_symbols{1}, pair_symbols{2}];

disp('2.a) Probabilities: ');
disp(table(pair_probabilities, pair_symbols));

%% Question 2.b
% We need to convert pair_symbols to cell (178x1)
huff_dict_2nd = huffmandict(num2cell(pair_symbols, 2),
pair_probabilities);
entropy_2nd = -sum(arrayfun(@(x) x*log2(x), pair_probabilities));

dict_lengths_2nd = cellfun('length', huff_dict_2nd(:,2));
average_length_2nd = sum(pair_probabilities .* dict_lengths_2nd);

efficiency_2nd = entropy_2nd/average_length_2nd;

disp(['1.b.i) Entropy: ' num2str(entropy_2nd)]);
disp(['1.b.ii) Average Length: ' num2str(average_length_2nd)]);
disp(['1.b.iii) Efficiency: ' num2str(efficiency_2nd)]);

%% Question 3.a
```

```
disp(['3.a) H(X²) = ' num2str(entropy_2nd) ' < 2H(X) = 2 × '
num2str(entropy)])

%% Question 3.b
disp('3.b)')
disp(['For 1st order, entropy: ' num2str(entropy) ' < L='
num2str(average_length) ' < ' num2str(entropy+1)])
disp(['For 2nd order, entropy: ' num2str(entropy_2nd) ' < L='
num2str(average_length_2nd) ' < ' num2str(entropy_2nd+1)])

%% Question 4
encoded_image = huffmanenco(image(:), huff_dict);

decoded_image = huffmandeco(encoded_image, huff_dict);
decoded_image = reshape(decoded_image, 200, []);
total_error = sumabs(image - decoded_image);

bI = reshape((dec2bin(typecast(image(:),'uint8'),4)-'0').',1,[]);
J_ratio = numel(encoded_image) / numel(bI);

disp(['4) Total error in decoded image = ' num2str(total_error)])
disp(['J = ' num2str(J_ratio)])

%% Question 5
% test the channel
TEST_TRANSMISSIONS = 1000;
transmission_errors = 0;
for i = 1:TEST_TRANSMISSIONS
    transmitted_image = binary_symmetric_channel(encoded_image);
    temp_errors = sum(transmitted_image ~= encoded_image);
    transmission_errors = transmission_errors + temp_errors;
end
p = transmission_errors / (numel(encoded_image) * TEST_TRANSMISSIONS);

% Calculations for input X
encoded_occurances = groupcounts(encoded_image);
encoded_probabilities = encoded_occurances./numel(encoded_image);

% Joint and conditional calculations
joint_probabilities = [encoded_probabilities .* [1-p; p] ...
                       encoded_probabilities .* [p; 1-p]];

Y_given_X_entropy = -sum(joint_probabilities .* log2([1-p, p; p, 1-p]),
"all");

% Calculation for output Y
channel_probabilities = [joint_probabilities(1,1) +
joint_probabilities(1,2);
                         joint_probabilities(2,1) +
joint_probabilities(2,2)];

channel_entropy =  -sum(arrayfun(@(x) x*log2(x), channel_probabilities));

% Calculate entropy Hb and capacity
Hb = -p*log2(p) - (1-p)*log2(1-p);
capacity = 1 - Hb;
mutual_information = channel_entropy - Y_given_X_entropy;
```

```
fprintf('5) p=%.2f, capacity=%f bits, mutual information = %f bits', p,
capacity, mutual_information)

%% Clear utility variables
clear symbol_counts dict_lengths;               % Q1
clear pair_symbol_counts dict_lengths_2nd;      % Q2
clear temp_errors;                              % Q5
```

part_b.m

```
%%% Part B %%%
clear;
clc;
load source.mat
% Here you can change t=t(1:5000) etc.
%t=t(15001:20000);
%% Calculations
y_errors = num2cell(zeros(3,6));

encoded_p5 = {0, 0, 0};
encoded_p10 = {0, 0, 0};
a_coeff_p5 = {0, 0, 0};
a_coeff_p10 = {0, 0, 0};
decoded_p5 = {0, 0, 0};
decoded_p10 = {0, 0, 0};

msqerrors = num2cell(zeros(3,6));

for N=1:3
    for p=5:10
        % Calculate encoded signals, coefficients and y errors.
        if p ~=5 && p ~= 10
            [~, ~, y_errors{N, p-4}] = dpcm_encode(t, p, N,-3.5, 3.5);
        end

        if p == 5
            [encoded_p5{N}, a_coeff_p5{N}, y_errors{N, p-4}] =
dpcm_encode(t, p, N,-3.5, 3.5);
        end

        if p == 10
            [encoded_p10{N}, a_coeff_p10{N}, y_errors{N, p-4}] =
dpcm_encode(t, p, N,-3.5, 3.5);
        end

        % Calculate mean square errors y.
        msqerrors{N,p-4} = sumsqr(y_errors{N,p-4})/numel(y_errors{N,p-
4});
    end
end

for N=1:3
    decoded_p5{N} = dpcm_decode(encoded_p5{N}, a_coeff_p5{N});
    decoded_p10{N} = dpcm_decode([encoded_p10{N}], a_coeff_p10{N});
```

```
end

%% Question 2
%Plot for p = 5
figure
hold on
plot(t, 'c')
plot(y_errors{1,1}, 'r-')
plot(y_errors{2,1}, 'g-')
plot(y_errors{3,1}, 'b-')
hold off
title("y prediction error for p=5, N=1,2,3")
legend('Signal', 'N=1', 'N=2', 'N=3')
xlabel('Sample')
ylabel('y prediction error')

%Plot for p = 10
figure
hold on
plot(t, 'c')
plot(y_errors{1,6}, 'r-')
plot(y_errors{2,6}, 'g-')
plot(y_errors{3,6}, 'b-')
hold off
title("y prediction error for p=10, N=1,2,3")
legend('Signal', 'N=1', 'N=2', 'N=3')
xlabel('Sample')
ylabel('y prediction error')

%% Question 3
figure
hold on
bar(5:10, cell2mat(msqerrors)')
legend('N=1', 'N=2', 'N=3')
xlabel('p values')
ylabel('mean-squared error y')
hold off

%% Question 4
%Plots for p = 5
figure
hold on
plot(t, 'c')
plot(decoded_p5{1}, 'r-')
plot(decoded_p5{2}, 'g-')
plot(decoded_p5{3}, 'b-')
hold off
title("Recreated signal for p=5, N=1,2,3")
legend('Signal', 'N=1', 'N=2', 'N=3')
xlabel('Sample')
ylabel('value')

figure
hold on

plot(t, 'c')
plot(t - decoded_p5{1}, 'r-')
plot(t - decoded_p5{2}, 'g-')
plot(t - decoded_p5{3}, 'b-')
```

```
hold off
title("Original minus recreated signal for p=5, N=1,2,3")
legend('Signal', 'N=1', 'N=2', 'N=3')
xlabel('Sample')
ylabel('value')

%Plots for p = 10
figure
hold on
plot(t, 'c')
plot(decoded_p10{1}, 'r-')
plot(decoded_p10{2}, 'g-')
plot(decoded_p10{3}, 'b-')
hold off
title("Recreated signal for p=10, N=1,2,3")
legend('Signal', 'N=1', 'N=2', 'N=3')
xlabel('Sample')
ylabel('value')

figure
hold on
plot(t, 'c')
plot(t - decoded_p10{1}, 'r-')
plot(t - decoded_p10{2}, 'g-')
plot(t - decoded_p10{3}, 'b-')
hold off
title("Original minus recreated signal for p=10, N=1,2,3")
legend('Signal', 'N=1', 'N=2', 'N=3')
xlabel('Sample')
ylabel('value')

%% DPCM Encoding %%
function [enc_signal, a_coeff_quant, y_error] = dpcm_encode(signal, p, N,
min_val, max_val)
    % Calculate a coefficients based on the input signal
    auto_corr = xcorr(signal, p, 'unbiased');
    R = toeplitz(auto_corr(p+1:end-1));
    r = auto_corr(p+2:end);

    a_coeff = R\r;

    % Alternative: Levinson way (requires multiplying the final array
with -1!!)
    %a_coeff = levinson(auto_corr(p+1:end), p);
    %a_coeff = a_coeff(2:end); % Drop a(0)

    % Convert the indices into the actual values for the coefficients a
    a_coeff_indices = arrayfun(@(x) my_quantizer(x, 8, -2, 2), a_coeff);

    % Calculate the quantizing centers
    %delta = (max_value - min_value) / (2^N);
    %centers = (max_value - delta/2):-delta:(min_value + delta/2);
    delta_center_coeff = (2 - (-2)) / (2^8);
    centers_coeff = (2 - delta_center_coeff/2):-delta_center_coeff:(-2 +
delta_center_coeff/2);

    a_coeff_quant = centers_coeff(a_coeff_indices);

    % Initialize arrays
```

```
    enc_signal = zeros(size(signal));
    memory = zeros(p, 1);

    % Calculate the quantizer centers
    delta = (max_val - min_val) / (2^N);
    centers = (max_val - delta/2):-delta:(min_val + delta/2);

    y_error = zeros(size(signal)); % So it's plotable

    for n=1:length(signal)
        % Predict y
        y_predicted = a_coeff_quant * memory;
        y_error(n) = signal(n) - y_predicted;

        % Calc y_quant
        y_quant_ind = my_quantizer(y_error(n), N, min_val, max_val);

        y_quant = centers(y_quant_ind);
        enc_signal(n) = y_quant;

        % update memory
        y_new_memory = y_quant + y_predicted;
        memory = [y_new_memory; memory(1:end-1)];

    end

end

%% DPCM Decoding %%
function [dec_signal] = dpcm_decode(enc_signal, a_coeff_quant)
    % Initialize arrays
    dec_signal = zeros(size(enc_signal));
    memory = zeros(numel(a_coeff_quant), 1);

    for n=1:length(enc_signal)
        % Reconstruct signal
        y_predicted = a_coeff_quant * memory;
        y = enc_signal(n) + y_predicted;
        dec_signal(n) = y;

        % update memory
        memory = [y; memory(1:end-1)];

    end

end

%% Quantizer %%
function index = my_quantizer(value, N, min_value, max_value)
    % Calculate delta and find the centers
    delta = (max_value - min_value) / (2^N);
    centers = (max_value - delta/2):-delta:(min_value + delta/2);

    % Find the difference between quantized values and input value
    differences = abs(centers - value);

    % Find quantized value with minimum distance from input value
    [~, index] = min(differences);
end
```

part_b_exports.m (για εξαγωγή των γραφημάτων)

```
%%% Part B %%%
clear;
clc;
load source.mat

%% Draw and Save
FolderName = "./exports/";

% Save 20k figures
draw_figures(t, "20k-");
FigList = findobj(allchild(0), 'flat', 'Type', 'figure');

for iFig = 1:length(FigList)
  FigHandle = FigList(iFig);
  FigName   = strcat("20k-", num2str(get(FigHandle, 'Number')));
  saveas(FigHandle, fullfile(FolderName, strcat(FigName,'.png')));
end
close all;

% Save 5k parts figures
FilePrefixes = ["0-5k-", "5-10k-", "10-15k-", "15-20k-"];

for i=0:3
    draw_figures(t((1 + 5000*i):(5000 + 5000*i)), FilePrefixes(i+1));
    FigList = findobj(allchild(0), 'flat', 'Type', 'figure');
    for iFig = 1:length(FigList)
      FigHandle = FigList(iFig);
      FigName   = strcat(FilePrefixes(i+1), num2str(get(FigHandle,
'Number')));
      saveas(FigHandle, fullfile(FolderName, strcat(FigName,'.png')));
    end
    close all;
end

%% Draw function
function draw_figures(t, graph_prefix)
    % Calculations
    y_errors = num2cell(zeros(3,6));

    encoded_p5 = {0, 0, 0};
    encoded_p10 = {0, 0, 0};
    a_coeff_p5 = {0, 0, 0};
    a_coeff_p10 = {0, 0, 0};
    decoded_p5 = {0, 0, 0};
    decoded_p10 = {0, 0, 0};

    msqerrors = num2cell(zeros(3,6));

    for N=1:3
        for p=5:10
            % Calculate encoded signals, coefficients and y errors.
            if p ~=5 && p ~= 10
                [~, ~, y_errors{N, p-4}] = dpcm_encode(t, p, N,-3.5,
3.5);
            end
```

```
            if p == 5
                [encoded_p5{N}, a_coeff_p5{N}, y_errors{N, p-4}] =
dpcm_encode(t, p, N,-3.5, 3.5);
            end

            if p == 10
                [encoded_p10{N}, a_coeff_p10{N}, y_errors{N, p-4}] =
dpcm_encode(t, p, N,-3.5, 3.5);
            end

            % Calculate mean square errors y.
            msqerrors{N,p-4} = sumsqr(y_errors{N,p-
4})/numel(y_errors{N,p-4});

        end
    end

    for N=1:3
        decoded_p5{N} = dpcm_decode(encoded_p5{N}, a_coeff_p5{N});
        decoded_p10{N} = dpcm_decode([encoded_p10{N}], a_coeff_p10{N});
    end

    % Question 2
    %Plot for p = 5
    figure
    hold on
    plot(t, 'c')
    plot(y_errors{1,1}, 'r-')
    plot(y_errors{2,1}, 'g-')
    plot(y_errors{3,1}, 'b-')
    hold off
    title(graph_prefix + "y prediction error for p=5, N=1,2,3")
    legend('Signal', 'N=1', 'N=2', 'N=3')
    xlabel('Sample')
    ylabel('y prediction error')

    %Plot for p = 10
    figure
    hold on
    plot(t, 'c')
    plot(y_errors{1,6}, 'r-')
    plot(y_errors{2,6}, 'g-')
    plot(y_errors{3,6}, 'b-')
    hold off
    title(graph_prefix + "y prediction error for p=10, N=1,2,3")
    legend('Signal', 'N=1', 'N=2', 'N=3')
    xlabel('Sample')
    ylabel('y prediction error')

    % Question 3
    figure
    hold on
    bar(5:10, cell2mat(msqerrors)')
    title(graph_prefix + "msqe (prediction error) p=5:10, N=1,2,3")
    legend('N=1', 'N=2', 'N=3')
    xlabel('p values')
    ylabel('mean-squared error y')
    hold off
```

```
    % Question 4
    %Plots for p = 5
    figure
    hold on
    plot(t, 'c')
    plot(decoded_p5{1}, 'r-')
    plot(decoded_p5{2}, 'g-')
    plot(decoded_p5{3}, 'b-')
    hold off
    title(graph_prefix + "Recreated signal for p=5, N=1,2,3")
    legend('Signal', 'N=1', 'N=2', 'N=3')
    xlabel('Sample')
    ylabel('value')

    figure
    hold on
    plot(t, 'c')
    plot(t - decoded_p5{1}, 'r-')
    plot(t - decoded_p5{2}, 'g-')
    plot(t - decoded_p5{3}, 'b-')
    hold off
    title(graph_prefix + "Original minus recreated signal for p=5,
N=1,2,3")
    legend('Signal', 'N=1', 'N=2', 'N=3')
    xlabel('Sample')
    ylabel('value')

    %Plots for p = 10
    figure
    hold on
    plot(t, 'c')
    plot(decoded_p10{1}, 'r-')
    plot(decoded_p10{2}, 'g-')
    plot(decoded_p10{3}, 'b-')
    hold off
    title(graph_prefix + "Recreated signal for p=10, N=1,2,3")
    legend('Signal', 'N=1', 'N=2', 'N=3')
    xlabel('Sample')
    ylabel('value')

    figure
    hold on
    plot(t, 'c')
    plot(t - decoded_p10{1}, 'r-')
    plot(t - decoded_p10{2}, 'g-')
    plot(t - decoded_p10{3}, 'b-')
    hold off
    title(graph_prefix + "Original minus recreated signal for p=10,
N=1,2,3")
    legend('Signal', 'N=1', 'N=2', 'N=3')
    xlabel('Sample')
    ylabel('value')
end

%% DPCM Encoding %%
function [enc_signal, a_coeff_quant, y_error] = dpcm_encode(signal, p, N,
min_val, max_val)
    % Calculate a coefficients based on the input signal
    auto_corr = xcorr(signal, p, 'unbiased');
```

```
    R = toeplitz(auto_corr(p+1:end-1));
    r = auto_corr(p+2:end);

    a_coeff = R\r;

    % Alternative: Levinson way (requires multiplying the final array
with -1!!)
    %a_coeff = levinson(auto_corr(p+1:end), p);
    %a_coeff = a_coeff(2:end); % Drop a(0)

    % Convert the indices into the actual values for the coefficients a
    a_coeff_indices = arrayfun(@(x) my_quantizer(x, 8, -2, 2), a_coeff);

    % Calculate the quantizing centers
    %delta = (max_value - min_value) / (2^N);
    %centers = (max_value - delta/2):-delta:(min_value + delta/2);
    delta_center_coeff = (2 - (-2)) / (2^8);
    centers_coeff = (2 - delta_center_coeff/2):-delta_center_coeff:(-2 +
delta_center_coeff/2);

    a_coeff_quant = centers_coeff(a_coeff_indices);

    % Initialize arrays
    enc_signal = zeros(size(signal));
    memory = zeros(p, 1);

    % Calculate the quantizer centers
    delta = (max_val - min_val) / (2^N);
    centers = (max_val - delta/2):-delta:(min_val + delta/2);

    y_error = zeros(size(signal)); % So it's plotable

    for n=1:length(signal)
        % Predict y
        y_predicted = a_coeff_quant * memory;
        y_error(n) = signal(n) - y_predicted;

        % Calc y_quant
        y_quant_ind = my_quantizer(y_error(n), N, min_val, max_val);

        y_quant = centers(y_quant_ind);
        enc_signal(n) = y_quant;

        % update memory
        y_new_memory = y_quant + y_predicted;
        memory = [y_new_memory; memory(1:end-1)];

    end

end

%% DPCM Decoding %%
function [dec_signal] = dpcm_decode(enc_signal, a_coeff_quant)
    % Initialize arrays
    dec_signal = zeros(size(enc_signal));
    memory = zeros(numel(a_coeff_quant), 1);

    for n=1:length(enc_signal)
        % Reconstruct signal
```

```
        y_predicted = a_coeff_quant * memory;
        y = enc_signal(n) + y_predicted;
        dec_signal(n) = y;

        % update memory
        memory = [y; memory(1:end-1)];

    end

end

%% Quantizer %%
function index = my_quantizer(value, N, min_value, max_value)
    % Calculate delta and find the centers
    delta = (max_value - min_value) / (2^N);
    centers = (max_value - delta/2):-delta:(min_value + delta/2);

    % Find the difference between quantized values and input value
    differences = abs(centers - value);

    % Find quantized value with minimum distance from input value
    [~, index] = min(differences);
end
```