



ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ 1 ΣΕΤ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ

2023-2024



ΔΗΜΗΤΡΙΟΣ ΤΣΑΜΠΡΑΣ

1072467

[Διεύθυνση εταιρείας]

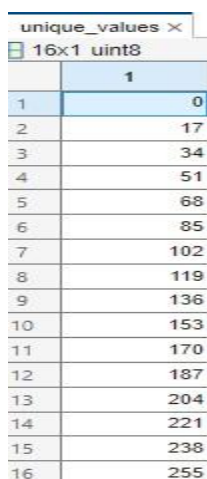
Περιεχόμενα

ΜΕΡΟΣ Α΄	2
ΕΡΩΤΗΜΑ 1	2
ΕΡΩΤΗΜΑ 2	4
ΕΡΩΤΗΜΑ 3	6
ΕΡΩΤΗΜΑ 4	6
ΜΕΡΟΣ Β΄	7
ΕΡΩΤΗΜΑ 1	7
ΕΡΩΤΗΜΑ 2	8
ΕΡΩΤΗΜΑ 3	9
ΕΡΩΤΗΜΑ 4	10
Η ΠΕΡΙΠΤΩΣΗ ΤΟΥ ΟΜΟΙΟΜΟΡΦΟΥ ΚΒΑΝΤΙΣΤΗ	11
ΣΥΝΟΛΟ ΚΩΔΙΚΑ	15

ΜΕΡΟΣ Α΄

ΕΡΩΤΗΜΑ 1

- a) Οι διακριτές τιμές οι οποίες λαμβάνουν τα pixels της εικόνας είναι **16** στο πλήθος και είναι οι εξής:



unique_values x	
16x1 uint8	
	1
1	0
2	17
3	34
4	51
5	68
6	85
7	102
8	119
9	136
10	153
11	170
12	187
13	204
14	221
15	238
16	255

Για να βρω τα ζητούμενα, χρησιμοποίησα τις συναρτήσεις **unique** και **histcounts**: Η συνάρτηση **unique** χρησιμοποιείται για να βρει τις μοναδικές τιμές στην εικόνα, ενώ η συνάρτηση **histcounts** χρησιμοποιείται για να υπολογίσει το ιστόγραμμα των τιμών, δηλαδή πόσες φορές εμφανίζεται κάθε διακριτή τιμή pixel και να επιστρέψει τις πιθανότητες για κάθε σύμβολο, οι οποίες φαίνονται παρακάτω:

	1	2	3	4	5	6	7	8	9
1	0.0962	0.0814	0.0683	0.0625	0.0768	0.0905	0.1132	0.0906	0.0965

10	11	12	13	14	15	16
0.0671	0.0389	0.0329	0.0337	0.0259	0.0224	0.0031

b) Η κωδικοποίηση Huffman φαίνεται στο παρακάτω screenshot:

Dictionary ×		
16x2 cell		
	1	2
1	0	[1,1,0]
2	17	[0,0,1,0]
3	34	[0,1,0,0]
4	51	[0,1,1,1]
5	68	[0,0,1,1]
6	85	[0,0,0,0]
7	102	[1,0,0]
8	119	[1,1,1]
9	136	[1,0,1]
10	153	[0,1,0,1]
11	170	[0,0,0,1,1]
12	187	[0,1,1,0,1]
13	204	[0,1,1,0,0]
14	221	[0,0,0,1,0,0]
15	238	[0,0,0,1,0,1,0]
16	255	[0,0,0,1,0,1,1]

- I. Η εντροπία της κωδικοποίησης προκύπτει ίση με **3.7831** και στη matlab υπολογίζεται από τον τύπο: **entropy = -sum(probabilities .* log2(probabilities));**
- II. Το μέσο μήκος κώδικα προκύπτει ίσο με **3.8374**.
- III. Η αποδοτικότητα του κώδικα προκύπτει ίση με **0.9859** ή αλλιώς περίπου **98.5%** και στη matlab υπολογίζεται από τον τύπο: **efficiency = entropy / mean_code_length;**

The entropy is:
3.7831

The mean code length is:
3.8374

The efficiency is:
0.9859

Σύμφωνα με τα αποτελέσματα, η εντροπία και το μέσο μήκος κώδικας βρίσκονται πολύ κοντά σε θέμα τιμών. Εξαιτίας αυτού, η αποδοτικότητα του κώδικα βγαίνει πάρα πολύ υψηλή, καθώς αντιπροσωπεύει το πόσο πλησιάζει η εντροπία το μέσο μήκος κώδικα. Συνολικά, τα αποτελέσματα υποδεικνύουν ότι η συγκεκριμένη κωδικοποίηση είναι αποτελεσματική και η πληροφορία στην εικόνα μπορεί να αναπαρασταθεί αποτελεσματικά.

ΕΡΩΤΗΜΑ 2

α) Τα ζεύγη χαρακτήρων που προκύπτουν είναι **202** στο πλήθος και ένα κομμάτι αυτών των ζευγών φαίνεται παρακάτω:

202x1 cell	
	1
1	' 0 0'
2	' 0 17'
3	' 0 34'
4	' 0 51'
5	' 0 68'
6	' 0 85'
7	' 0 102'
8	' 17 0'
9	' 17 17'
10	' 17 34'
11	' 17 51'
12	' 17 68'
13	' 17 85'
14	' 17 102'
15	' 17 119'
16	' 17 136'
17	' 17 153'
18	' 17 170'
19	' 17 187'
20	' 34 0'
21	' 34 17'
22	' 34 34'
23	' 34 51'
24	' 34 68'

Η πιθανότητα κάθε ζεύγους υπολογίζεται στη matlab από το:

```
[unique_values,~, biSymbolIndices]= unique(symbols);
```

```
probabilities = histcounts(biSymbolIndices, numel(unique_values)) / numel(symbols);
```

Ένα κομμάτι των πιθανοτήτων φαίνεται παρακάτω:

	1	2	3	4	5	6	7	8	9
1	0.0810	0.0129	0.0013	0.0005	0.0003	0.0001	0.0001	0.0123	0.0502

- Εδώ, το **symbols** είναι ένα διάνυσμα που περιέχει σύμβολα.
- Η συνάρτηση **unique** χρησιμοποιείται για να εντοπίσει τις μοναδικές τιμές του **symbols**.
- Οι μοναδικές τιμές αποθηκεύονται στη μεταβλητή **unique_values**.
- Το **biSymbolIndices** περιέχει τους δείκτες των μοναδικών τιμών στο αρχικό διάνυσμα **symbols**.
- Η συνάρτηση **histcounts** χρησιμοποιείται για να υπολογίσει το ιστόγραμμα των δεικτών **biSymbolIndices** με βάση τον αριθμό των μοναδικών τιμών.
- Το **numel(unique_values)** αναφέρεται στον αριθμό των μοναδικών τιμών.
- Οι πιθανότητες υπολογίζονται διαιρώντας το ιστόγραμμα με τον συνολικό αριθμό των συμβόλων (**numel(symbols)**).

b) Ένα κομμάτι της κωδικοποίησης Huffman φαίνεται στο παρακάτω screenshot:

```
{' 0 34'} {[ 0 0 0 0 1 1 1 0 1 1]}
{' 0 51'} {[ 1 0 0 1 0 0 1 0 0 1 0]}
{' 0 68'} {[ 0 0 1 0 1 0 1 0 0 1 1 1]}
{' 0 85'} {[ 0 1 0 1 1 0 0 0 1 0 1 1 0 0]}
{' 0 102'} {[ 1 0 1 0 1 1 1 1 1 0 0 1 1]}
{' 17 0'} {[ 1 1 1 0 1 0]}
{' 17 17'} {[ 1 1 0 0]}
{' 17 34'} {[ 1 0 1 0 1 0]}
{' 17 51'} {[ 0 0 0 0 1 0 1 1 0]}
```

- I. Η εντροπία της κωδικοποίησης προκύπτει ίση με **5.6520** και στη matlab υπολογίζεται ομοίως με το ερώτημα 1.
- II. Το μέσο μήκος κώδικα προκύπτει ίσο με **10.5297**.
- III. Η αποδοτικότητα του κώδικα προκύπτει ίση με **0.5368** ή αλλιώς περίπου **53.6%** και στη matlab υπολογίζεται ομοίως με το ερώτημα 1.

```
The entropy is:
3.7831
```

```
The mean code length is:
3.8374
```

```
The efficiency is:
0.9859
```

Σύμφωνα με τα αποτελέσματα, η εντροπία και το μέσο μήκος κώδικας βρίσκονται πολύ μακριά σε θέμα τιμών. Συγκεκριμένα, η εντροπία είναι περίπου η μισή του μέσου μήκους κώδικα. Εξαιτίας αυτού, η αποδοτικότητα του κώδικα βγαίνει πολύ χαμηλή, καθώς αντιπροσωπεύει το πόσο πλησιάζει η εντροπία το μέσο μήκος κώδικα. Συνολικά, τα αποτελέσματα υποδεικνύουν ότι η συγκεκριμένη κωδικοποίηση δεν είναι αποτελεσματική.

- c) Συγκριτικά, η εντροπία στη 2^η περίπτωση είναι υψηλότερη. Αυτό σημαίνει ότι η δεύτερη τάξη επέκτασης προσφέρει μεγαλύτερη πολυπλοκότητα και πληροφορία στα δεδομένα. Επίσης, το μέσο μήκος κώδικα στη 2^η περίπτωση είναι μεγαλύτερο. Αυτό υποδηλώνει ότι, λόγω της αυξημένης εντροπίας, ο κώδικας απαιτεί μεγαλύτερο μήκος για την αναπαράσταση της πληροφορίας. Τέλος, η αποδοτικότητα στη 2^η περίπτωση είναι αρκετά χαμηλότερη, αφού η δεύτερη τάξη επέκτασης εισάγει πιο περίπλοκη δομή στα δεδομένα.

ΕΡΩΤΗΜΑ 3

a) Ο τύπος $H(X^2) = 2H(X)$ δεν ισχύει στη συγκεκριμένη περίπτωση. Για να ισχύει, πρέπει τα συμβάντα να είναι ανεξάρτητα. Όμως, στην δεύτερη τάξη επέκτασης, δεν μπορώ να θεωρήσω ανεξαρτησία των συμβάντων. Στη δεύτερη τάξη, καθένα από τα διπλά σύμβολα εξαρτάται από το προηγούμενο σύμβολο. Συνεπώς, ο τύπος που αναφέρεται δεν ισχύει σε αυτήν την περίπτωση. Ο τρόπος υπολογισμού της εντροπίας για τη δεύτερη τάξη είναι πιο σύνθετος και περιλαμβάνει την εξέταση των πιθανοτήτων εμφάνισης όλων των δυνατών διπλών συμβόλων.

b) Για το ερώτημα 1, ένα φράγμα της μορφής $a \leq \mathcal{L} < b$ θα μπορούσε να είναι το εξής:

$$3.8 \leq 3.8374 < 4.0$$

Για το ερώτημα 2, ένα φράγμα της μορφής $a \leq \mathcal{L} < b$ θα μπορούσε να είναι το εξής:

$$10.0 \leq 10.5297 < 11.0$$

Αυτές είναι κάποιες ενδεικτικές τιμές που μπορεί να πάρουν τα a και b . Γενικότερα το φράγμα παρέχει μια συνοπτική εικόνα του εύρους των τιμών που μπορεί να λάβει το μέσο μήκος κώδικα. Αυτό μπορεί να βοηθήσει στην κατανόηση της απόδοσης του κώδικα και στην εκτίμηση του πόσο καλά λειτουργεί σε σχέση με τα προσδοκώμενα εύρη.

ΕΡΩΤΗΜΑ 4

Ο αριθμός bits της δυαδικής αναπαράστασης της εικόνας ισούται με **240000**, ενώ τα bits της κωδικοποίησης Huffman, ισούται με **115121**. Άρα, ο λόγος συμπίεσης ισούται με 0.47967. Γενικότερα, ο χαμηλός λόγος συμπίεσης μπορεί να εξοικονομήσει χώρο στην αποθήκευση εικόνων και να επιταχύνει τη μετάδοση δεδομένων.

```
Huffman Encoding and Decoding Results:  
Number of bits Huffman: 115121  
Number of bits Binary Representation: 240000  
Compression Ratio (J): 0.47967
```

ΜΕΡΟΣ Β΄

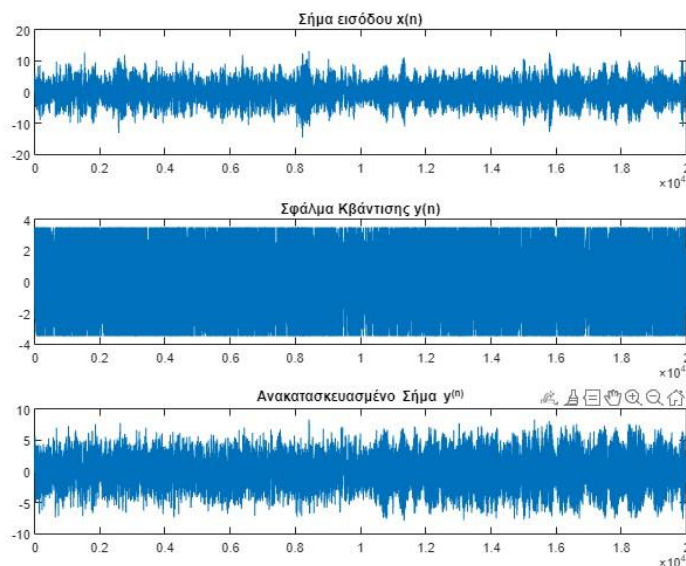
ΕΡΩΤΗΜΑ 1

Ο κώδικας που παρατίθεται στο τέλος, υλοποιεί το σύστημα κωδικοποίησης/αποκωδικοποίησης DPCM. Πιο συγκεκριμένα, ο κώδικας κάνει τα εξής:

- Φορτώνει τα δεδομένα από το αρχείο **source.mat**.
- Καθορίζει τις μεταβλητές **max_value** και **min_value** που αντιπροσωπεύουν τα άνω και κάτω όρια για τον κβαντισμό με τιμές 3.5 και -3.5 αντίστοιχα.
- Αντιγράφει τα δεδομένα της μεταβλητής **t** στη μεταβλητή **x**.
- Δημιουργεί τρία μηδενικά μητρώα **y_pred**, **y_quant**, και **y_reconstructed** για την αποθήκευση προβλέψεων, σφαλμάτων κβάντισης και ανακατασκευής αντίστοιχα.
- Εκτελεί ένα βρόγχο για κάθε χρονική στιγμή **n** από το δεύτερο δείγμα και μετά.
- Υπολογίζει την πρόβλεψη **y_pred(n)** βασισμένη στην προηγούμενη ανακατασκευή.
- Υπολογίζει το σφάλμα κβάντισης **y_quant(n)** ως διαφορά μεταξύ της εισόδου **x(n)** και της πρόβλεψης **y_pred(n)**.
- Κβαντίζει το σφάλμα κβάντισης στα όρια **max_value** και **min_value**.
- Ανακατασκευάζει το σήμα **y_reconstructed(n)** ως άθροισμα του κβαντισμένου σφάλματος και της πρόβλεψης.

Συνοπτικά, ο κώδικας προβλέπει τις τιμές της εισόδου, εφαρμόζει κβάντιση στα σφάλματα πρόβλεψης και ανακατασκευάζει το σήμα βασισμένο στα κβαντισμένα σφάλματα και τις προβλέψεις του προηγούμενου βήματος.

Τα αποτελέσματα φαίνονται και γραφικά στο παρακάτω screenshot:

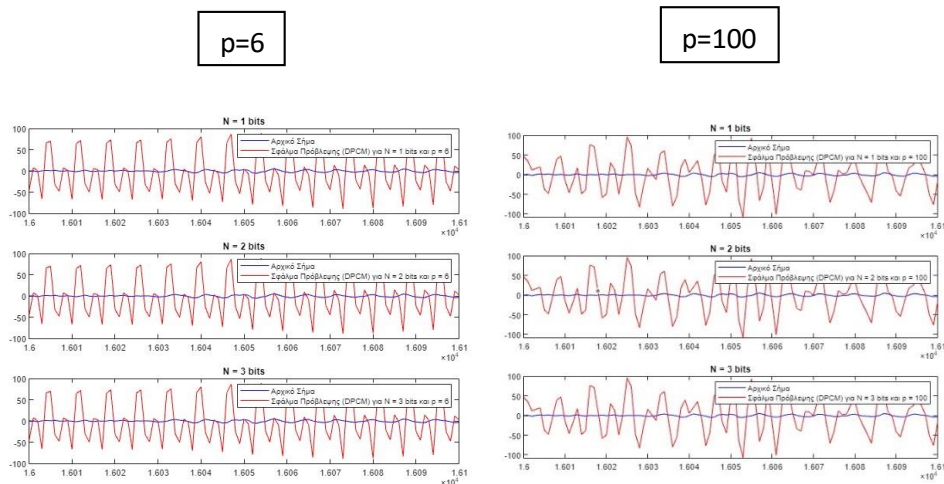


ΕΡΩΤΗΜΑ 2

Ο κώδικας που παρατίθεται στο τέλος, κάνει τα εξής:

- Φορτώνεται το σήμα **t** από το αρχείο **source.mat** και αποθηκεύεται στη μεταβλητή **x**.
- Ορίζεται η παράμετρος **p**, η οποία αντιπροσωπεύει τον αριθμό των προηγούμενων δειγμάτων που χρησιμοποιούνται για τον υπολογισμό της πρόβλεψης.
- Ο κώδικας υπολογίζει την πρόβλεψη **prediction** χρησιμοποιώντας τον αριθμό των προηγούμενων δειγμάτων που ορίζεται από την παράμετρο **p**. Στη συνέχεια, υπολογίζεται το σφάλμα πρόβλεψης **y** και αποθηκεύεται στη μεταβλητή **y_hat**. Τέλος, το αρχικό σήμα και το σφάλμα πρόβλεψης παρουσιάζονται σε διάφορα υπογραφήματα για διάφορες τιμές της παραμέτρου **N**.

Για $p=6$ και $p=100$, τα ζητούμενα γραφήματα φαίνονται παρακάτω:



Παρατηρώ, ότι όσο αυξάνει η τιμή του p , αλλάζει η αρχική φάση του σήματος. Στην πρώτη περίπτωση, το σήμα ξεκινάει από τα αρνητικά, ενώ στην δεύτερη περίπτωση ξεκινάει από τα αρνητικά.

Επίσης, αλλάζει η μέγιστη τιμή του πλάτους. Συγκεκριμένα, στη δεύτερη περίπτωση αυξάνει.

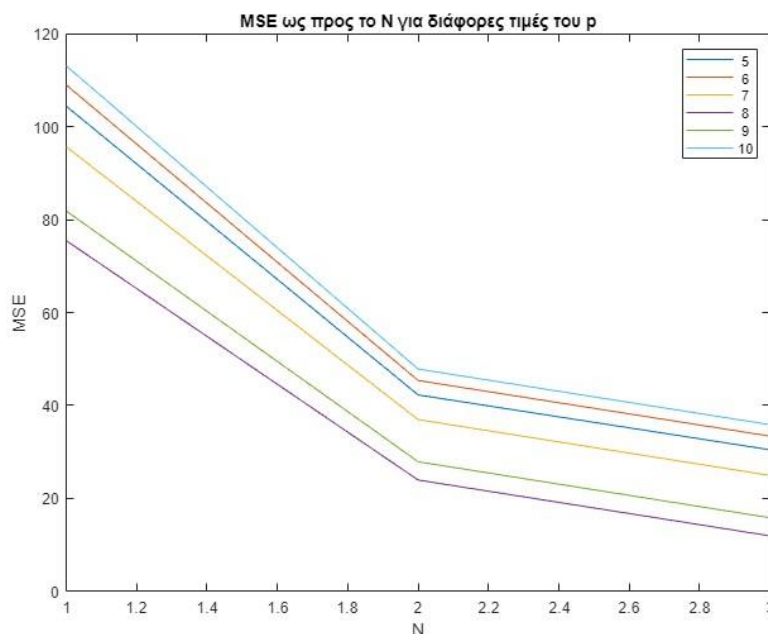
Συνήθως, όταν αυξάνει η τιμή του p , η πρόβλεψη γίνεται πιο ακριβής, αλλά το κόστος της αποθήκευσης των προηγούμενων δειγμάτων και της υπολογιστικής πολυπλοκότητας αυξάνεται. Συνεπώς, υπάρχει ένα trade-off μεταξύ ακρίβειας πρόβλεψης και πολυπλοκότητας.

ΕΡΩΤΗΜΑ 3

Ο κώδικας που παρατίθεται στο τέλος, κάνει τα εξής:

- Η συνάρτηση **predict_DPCM** προβλέπει το σήμα χρησιμοποιώντας την κβαντισμένη έξοδο και τον βαθμό **p**. Η πρόβλεψη βασίζεται στην προηγούμενη κβαντισμένη τιμή.
- Η συνάρτηση **quantize** εκτελεί τον κβαντισμό του σήματος με **N** bits, χρησιμοποιώντας ίσα βήματα μεταξύ της ελάχιστης και μέγιστης τιμής του σήματος.
- Η συνάρτηση **calculate_MSE** υπολογίζει το μέσο τετραγωνικό σφάλμα (MSE) μεταξύ του αρχικού σήματος **t** και του προβλεπόμενου σήματος.
- Ο κώδικας εκτελεί έναν εμπειρικό υπολογισμό του MSE για διάφορες τιμές των παραμέτρων **N** και **p**. Οι αξιολογήσεις αυτές αποθηκεύονται σε έναν πίνακα **MSE_values**.
- Το τμήμα κώδικα **coefficients = zeros(max_p - min_p + 1, max_p + 1);** δημιουργεί ένα μητρώο **coefficients** με μηδενικές τιμές, όπου οι γραμμές αντιστοιχούν στις διάφορες τιμές του **p** και οι στήλες αντιστοιχούν στους συντελεστές του προβλέπτη.
- Το τμήμα κώδικα **temp_coefficients = aryule(x, p);** χρησιμοποιεί τη συνάρτηση **aryule** για να υπολογίσει τους συντελεστές AR για την τρέχουσα τιμή του **p**.

Το ζητούμενο γράφημα φαίνεται στο παρακάτω screenshot και οι τιμές του **p** κυμαίνονται στο διάστημα [5,10] και του **N** στο [0,3]:



Με βάση το γράφημα, μπορώ να εξάγω τα εξής συμπεράσματα:

- Με την αύξηση του **N**, παρατηρείται μείωση του MSE. Αυτό είναι αναμενόμενο, καθώς περισσότερα bits για τον κβαντισμό σημαίνουν μεγαλύτερη ακρίβεια στην αναπαράσταση των τιμών και ενδεχομένως καλύτερη πρόβλεψη.
- Υπάρχει μια βέλτιστη τιμή του **p(p=8)**, κατά την οποία το MSE γίνεται ελάχιστο. Κατά τα άλλα, το MSE ενδέχεται να μην μειώνεται σημαντικά.

Οι συντελεστές που υπολογίζονται από τη συνάρτηση **aryule** αντιπροσωπεύουν τους συντελεστές του μοντέλου AR (Autoregressive), το οποίο χρησιμοποιείται για την πρόβλεψη των μελλοντικών τιμών του σήματος.

Κάθε συντελεστής αντιπροσωπεύει το βάρος που αντιστοιχεί σε μία προηγούμενη τιμή του σήματος. Τα μοντέλα AR χρησιμοποιούν αυτά τα βάρη για να εκτιμήσουν την επόμενη τιμή του σήματος, λαμβάνοντας υπόψη την ιστορία των προηγούμενων τιμών.

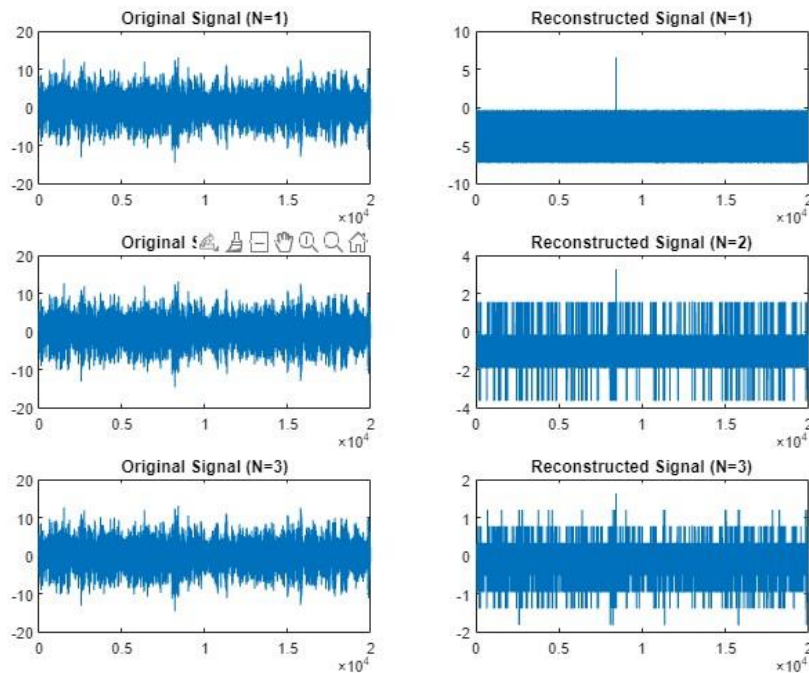
ΕΡΩΤΗΜΑ 4

Ο κώδικας που παρατίθεται στο τέλος κάνει τα εξής:

- Ορίζονται οι παράμετροι **p_values** και **N_values** για τον βαθμό πρόβλεψης και τις τιμές bits κβάντισης αντίστοιχα.
- Γίνεται επανάληψη για κάθε τιμή του **N(N_values)**. Καλείται η συνάρτηση **predict_DPCM** για την υλοποίηση της Διακριτικής Πρόβλεψης Κβαντισμένου Σήματος και εμφανίζονται τα αρχικά και τα ανακατασκευασμένα σήματα. Χρησιμοποιεί το κβαντισμένο σήμα για να υπολογίσει την πρόβλεψη για την επόμενη τιμή του σήματος.
- Η συνάρτηση **quantize** υλοποιεί τον κβαντισμό του σήματος, χρησιμοποιώντας έναν καθορισμένο αριθμό bits **N**.
- Η συνάρτηση **dequantize** ανακατασκευάζει το σήμα από κβαντισμένα δεδομένα, αντιστρέφοντας τη διαδικασία του κβαντισμού.

Κατά συνέπεια, ο κώδικας υλοποιεί τη DPCM για διάφορες τιμές bits κβάντισης **N** και παρουσιάζει τα αρχικά και τα ανακατασκευασμένα σήματα για κάθε τιμή του **N**.

Η ζητούμενη απεικόνιση, φαίνεται στο παρακάτω screenshot:



Σύμφωνα με τα παραπάνω γραφήματα, παρατηρώ ότι με υψηλότερο N , ο κβαντισμός είναι πιο λεπτομερής, αλλά μπορεί να υπάρχει μεγαλύτερη παραμόρφωση στην ανακατασκευή, ενώ με μικρότερο N , το ανακατασκευασμένο σήμα θα είναι περισσότερο εξομαλυνμένο και λιγότερο λεπτομερές.

Η ΠΕΡΙΠΤΩΣΗ ΤΟΥ ΟΜΟΙΟΜΟΡΦΟΥ ΚΒΑΝΤΙΣΤΗ

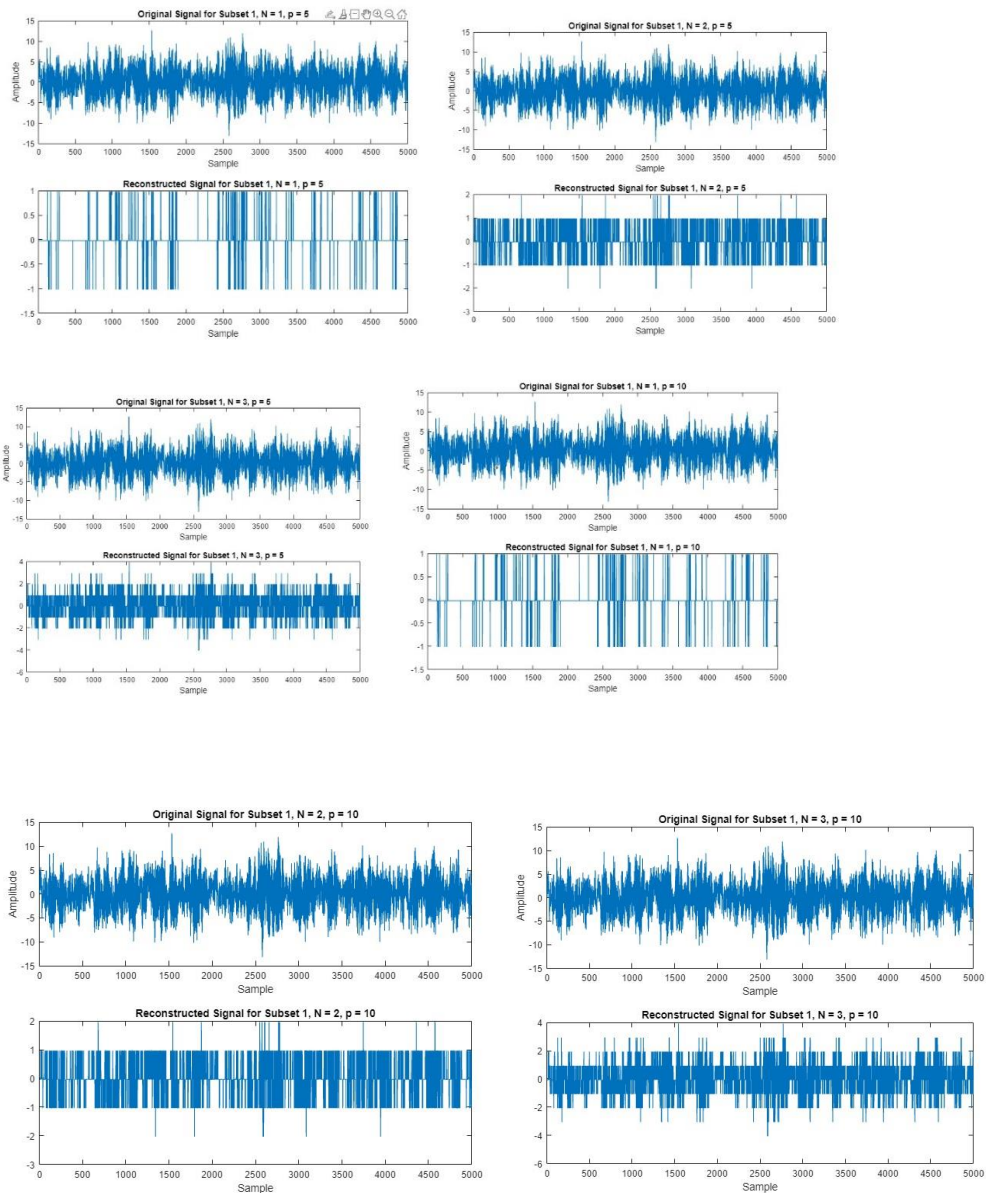
Ο κώδικας στο τέλος, είναι κοινός και για το 3^ο και για το 4^ο ερώτημα. Πιο συγκεκριμένα:

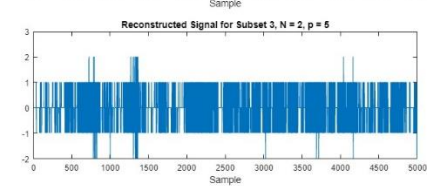
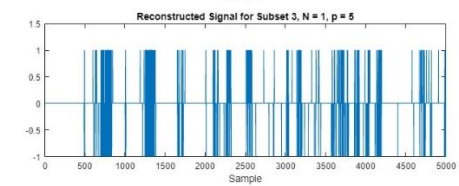
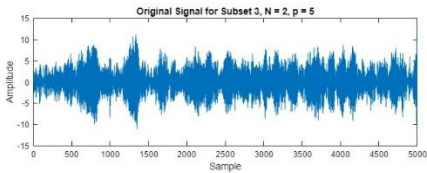
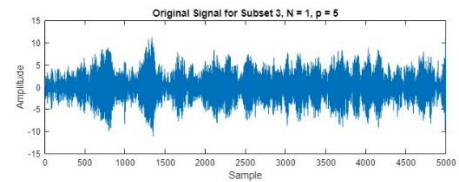
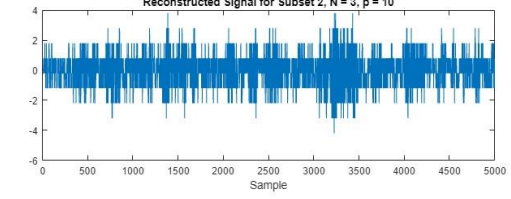
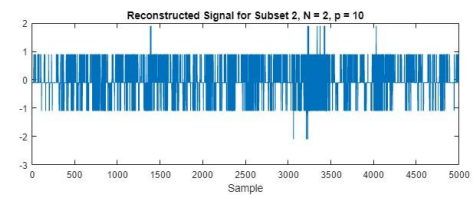
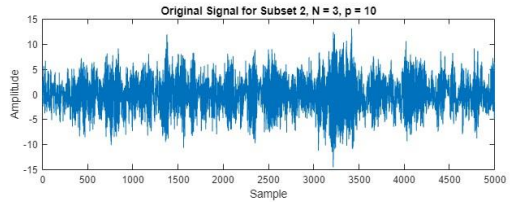
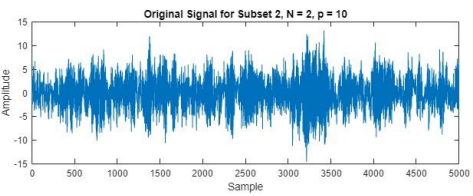
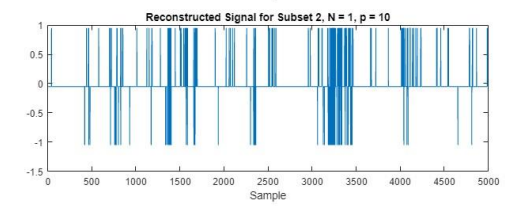
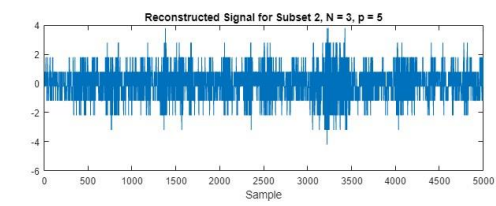
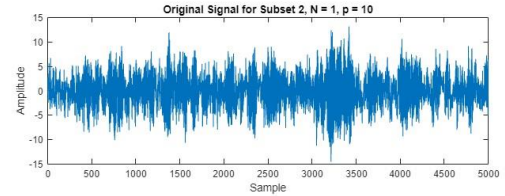
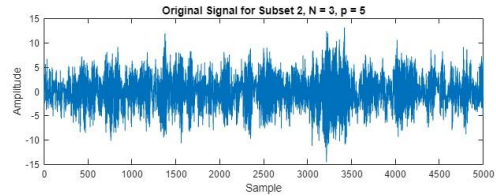
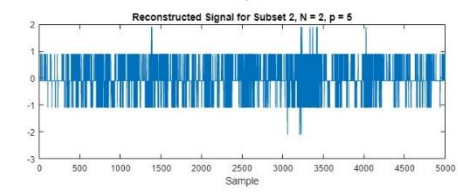
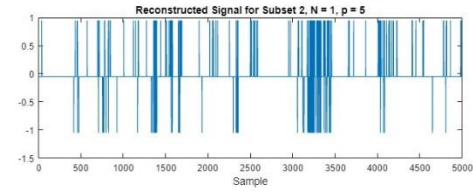
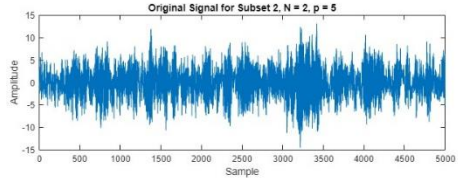
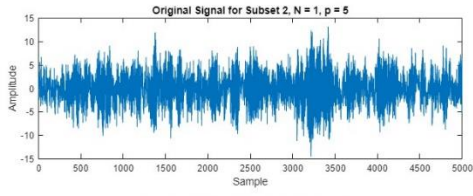
- Για το ερώτημα 3, με τον κώδικα εκτυπώνεται το MSE για κάθε p και N . Παρατίθεται ένα απόσπασμα παρακάτω:

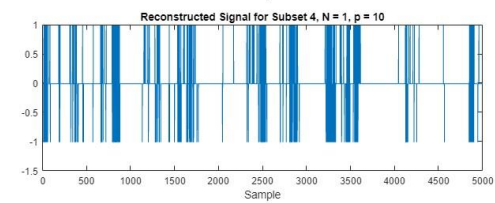
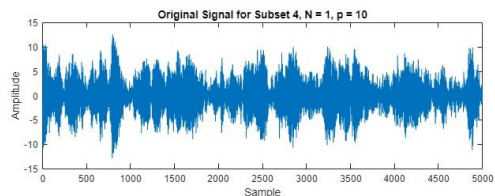
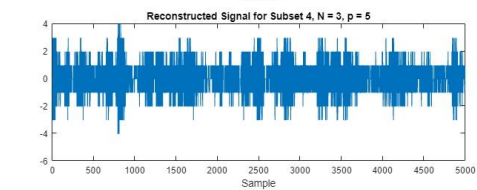
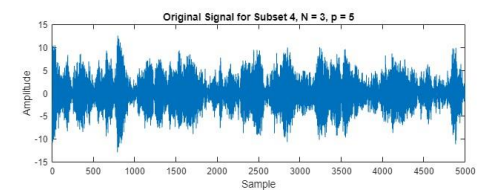
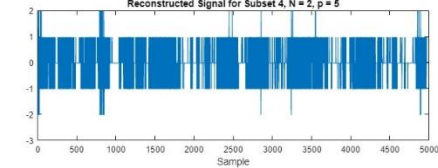
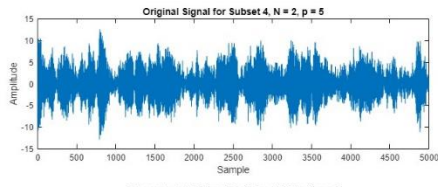
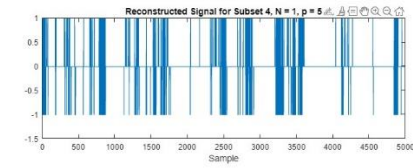
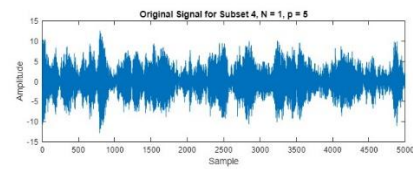
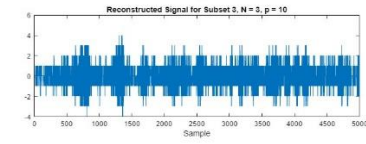
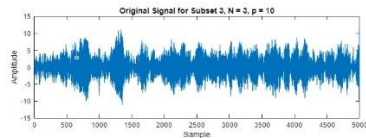
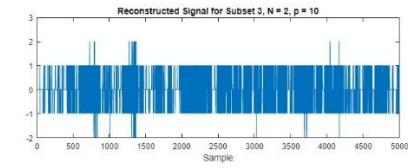
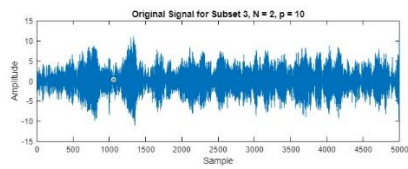
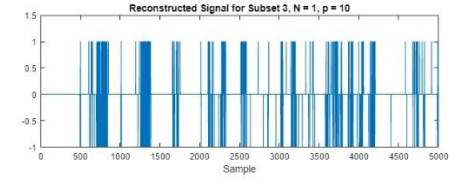
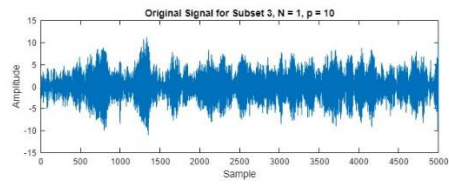
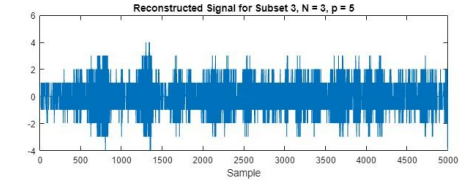
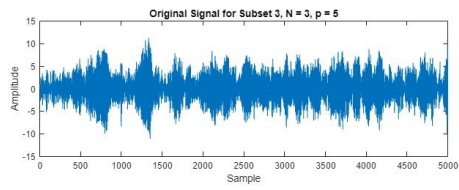
```
Uniform Quantizer Analysis for p = 5, N = 3
Subset 1, MSE = 10.195
Uniform Quantizer Analysis for p = 6, N = 1
Subset 1, MSE = 12.882
Uniform Quantizer Analysis for p = 6, N = 2
Subset 1, MSE = 14.7861
Uniform Quantizer Analysis for p = 6, N = 3
Subset 1, MSE = 17.7597
Uniform Quantizer Analysis for p = 7, N = 1
```

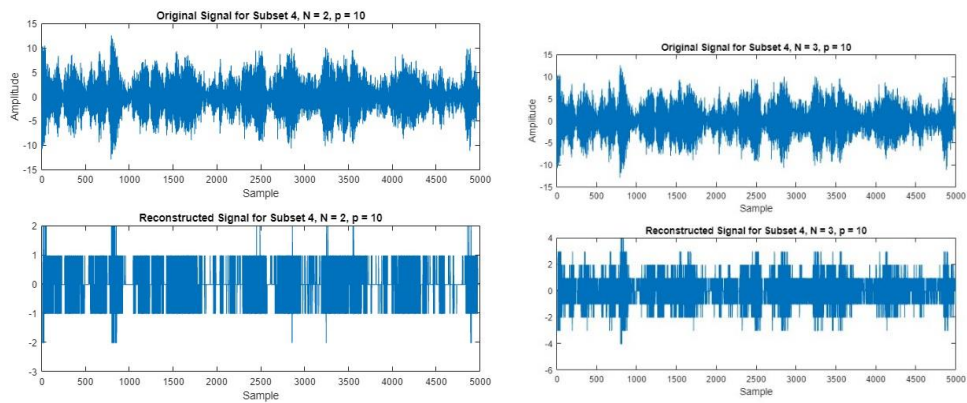
Όπως και στην περίπτωση του DPCM, έτσι και στον ομοιόμορφο κβαντιστή, όσο αυξάνει το N , τόσο μικρότερο είναι το mse . Βέβαια, για κάποιες τιμές του p , το mse αυξάνει για μεγάλο N , ενώ για άλλες τιμές του p , μειώνεται.

- Για το ερώτημα 4, χώρισα τα 20000 δείγματα που είχα σε 4 πακέτα των 5000. Παραθέτω για κάθε subplot και για κάθε τιμή του N και του p τα ακόλουθα γραφήματα:









ΣΥΝΟΛΟ ΚΩΔΙΚΑ

ΜΕΡΟΣ Α

ΕΡΩΤΗΜΑ 1

```

I = imread('parrot.png');

unique_values = unique(I);
probabilities = histcounts(I, length(unique_values)) / numel(I);

symbols = unique_values;

Dictionary = huffmandict(symbols, probabilities);
HuffmanCodes = huffmanenco(symbols, Dictionary);

entropy = -sum(probabilities .* log2(probabilities));

mean_code_length = 0;
for i = 1:length(symbols)

```



```

        symbol = symbols(i);
        code = Dictionary{i, 2};
        probability = probabilities(i);
        mean_code_length = mean_code_length + length(code) * probability;
    end

```

```

efficiency = entropy / mean_code_length;

```

```

disp(Dictionary);
disp('The entropy is:');
disp(entropy);
disp('The mean code length is:');
disp(mean_code_length);
disp('The efficiency is:');
disp(efficiency);

```

EPQTHMA2

```

I = imread('parrot.png');
values_of_pixels = double(I(:));

symbols = cellstr(num2str([values_of_pixels(1:end-1),
values_of_pixels(2:end)]));

[unique_values, ~, biSymbolIndices] = unique(symbols);
probabilities = histcounts(biSymbolIndices, numel(unique_values)) /
numel(symbols);

Dictionary = huffmandict(unique_values, probabilities);
HuffmanCodes = huffmanenco(unique_values, Dictionary);

entropy = -sum(probabilities .* log2(probabilities));

mean_code_lengthOfAll = sum(probabilities .* cellfun(@length, Dictionary(:,
2)));
mean_code_length = sum(mean_code_lengthOfAll)/202;

efficiency = entropy / mean_code_length;

disp(Dictionary);
disp('The entropy is:');
disp(entropy);

```

```

disp('The mean code length is:');
disp(mean_code_length);
disp('The efficiency is:');
disp(eficiency);

```

ΕΡΩΤΗΜΑ 4

```

I = imread('parrot.png');
bI = reshape((dec2bin(typecast(I(:), 'uint8'),4)-'0').',1,[])

values_of_pixels = double(I(:));

[symbols, ~, symbolIndices] = unique(values_of_pixels);
probabilities = histcounts(symbolIndices, numel(symbols)) /
numel(values_of_pixels);

Dictionary = huffmandict(symbols, probabilities);
HuffmanEncodedData = huffmanenco(values_of_pixels, Dictionary);

decodedData = huffmandeco(HuffmanEncodedData, Dictionary);

numBitsHuffman = length(HuffmanEncodedData);

numBitsBinary = numel(bI);

compressionRatio = numBitsHuffman / numBitsBinary;

disp('Huffman Encoding and Decoding Results:');
disp(['Number of bits Huffman: ', num2str(numBitsHuffman)]);
disp(['Number of bits Binary Representation: ', num2str(numBitsBinary)]);
disp(['Compression Ratio (J): ', num2str(compressionRatio)]);

```

ΜΕΡΟΣ Β

ΕΡΩΤΗΜΑ 1

```

load source.mat;

max_value = 3.5;
min_value = -3.5;
x = t;

y_pred = zeros(size(x));
y_quant = zeros(size(x));
y_reconstructed = zeros(size(x));

for n = 2:length(x)
    y_pred(n) = y_reconstructed(n-1);
    y_quant(n) = x(n) - y_pred(n);

```

```

    y_quant(n) = max(min(y_quant(n), max_value), min_value);
    y_reconstructed(n) = y_quant(n) + y_pred(n);
end

subplot(3,1,1), plot(x), title('Αρχικό Σήμα x(n)');
subplot(3,1,2), plot(y_quant), title('Σφάλμα Κβάντισης y(n)');
subplot(3,1,3), plot(y_reconstructed), title('Ανακατασκευασμένο Σήμα y^(n)');

```

ΕΡΩΤΗΜΑ 2

```

load source.mat;

x = t;

p = 10;

figure;
for N = 1:3
    y_hat = zeros(size(x));
    for n = p+1:length(x)
        prediction = sum(y_hat(n-1:-1:n-p));

        y = x(n) - prediction;
        y_hat(n) = y;
    end

    subplot(3, 1, N);
    plot(x, 'b', 'DisplayName', 'Αρχικό Σήμα');
    xlim([1.6*(10^4), 1.61*(10^4)]);
    hold on;
    plot(y_hat, 'r', 'DisplayName', ['Σφάλμα Πρόβλεψης (DPCM) για N = '
num2str(N) ' bits και p = ' num2str(p)]);
    xlim([1.6*(10^4), 1.61*(10^4)]);
    hold off;
    title(['N = ' num2str(N) ' bits']);
    legend;
end

```

ΕΡΩΤΗΜΑ 3

```

load source.mat;

p_values = 5:10;
N_values = 1:3;
x = t;
MSE_values = zeros(length(N_values), length(p_values));
for i = 1:length(N_values)
    for j = 1:length(p_values)
        N = N_values(i);
        p = p_values(j);

        predicted_signal = predict_DPCM(t, p, N);
    end
end

```

```

        MSE = calculate_MSE(t, predicted_signal);

        MSE_values(i, j) = MSE;
    end
end
min_p = 5;
max_p = 10;
coefficients = zeros(max_p - min_p + 1, max_p + 1);
for p = min_p:max_p
    temp_coefficients = aryule(x, p);
    coefficients(p - min_p + 1, 1:length(temp_coefficients)) =
temp_coefficients;
end

figure;
plot(N_values, MSE_values);
legend(cellstr(num2str(p_values')), 'Location', 'Best');
xlabel('N');
ylabel('MSE');
title('MSE ως προς το N για διάφορες τιμές του p');
function predicted_signal = predict_DPCM(x, p, N)
    predicted_signal = zeros(size(x));

    quantized_signal = quantize(x, N);

    for i = p+1:length(x)
        predicted_signal(i) = quantized_signal(i - p);
    end
end
function quantized_signal = quantize(signal, N)

    max_value = max(signal);
    min_value = min(signal);
    delta = (max_value - min_value) / (2^N);

    quantized_signal = floor((signal - min_value) / delta) * delta +
min_value;
end
function mse = calculate_MSE(t, predicted_signal)

    mse = mean((t - predicted_signal).^2);
end

```

ΕΡΩΤΗΜΑ 4

```

load source.mat;
x = t;

p_values = 10;
N_values = [1, 2, 3];
for i = 1:length(N_values)
    N = N_values(i);

```

```

predicted_signal = predict_DPCM(x, p_values, N);

quantized_signal = quantize(predicted_signal, N);

reconstructed_signal = dequantize(quantized_signal, N);

subplot(length(N_values), 2, (i-1)*2+1);
plot(x);
title(sprintf('Original Signal (N=%d)', N));

subplot(length(N_values), 2, i*2);
plot(reconstructed_signal);
title(sprintf('Reconstructed Signal (N=%d)', N));
end
function predicted_signal = predict_DPCM(x, p_values, N)
predicted_signal = zeros(size(x));

quantized_signal = quantize(x, N);
for i = p_values+1:length(x)

    predicted_signal(i) = quantized_signal(i - p_values);
end
end
function quantized_signal = quantize(signal, N)

max_value = max(signal);
min_value = min(signal);
delta = (max_value - min_value) / (2^N);

quantized_signal = floor((signal - min_value) / delta) * delta +
min_value;
end
function reconstructed_signal = dequantize(quantized_signal, N)

reconstructed_signal = quantized_signal * (2^(-N));
end

```

Η ΠΕΡΙΠΤΩΣΗ ΤΟΥ ΟΜΟΙΟΜΟΡΦΟΥ ΚΒΑΝΤΙΣΤΗ

```

load source.mat;
x = t;

signal_length = 20000;
signal = x;

num_subsets = signal_length / 5000;
subsets = reshape(signal, 5000, num_subsets);

ps = [5, 6, 7, 8, 9, 10];
Ns = [1, 2, 3];
for i = 1:size(subsets, 2)
    subset = subsets(:, i);

```

```

        for p = ps
            for N = Ns
                disp(['Uniform Quantizer Analysis for p = ', num2str(p), ',
N = ', num2str(N)]);

                predicted_signal = predict_DPCM(subset, p);

                quantized_signal = uniformQuantizer(predicted_signal, N);

                reconstructed_signal = dequantize(quantized_signal, N);

                plotSignals(subset, reconstructed_signal, N, p, i);

                mse = calculate_MSE(subset, reconstructed_signal);
                disp(['Subset ', num2str(i), ', MSE = ', num2str(mse)]);
            end
        end
    end
function predicted_signal = predict_DPCM(signal, p)
    predicted_signal = zeros(size(signal));
    for i = p+1:length(signal)

        predicted_signal(i) = signal(i - p);
    end
end
function quantized_signal = uniformQuantizer(predicted_signal, N)
    min_val = min(predicted_signal);
    max_val = max(predicted_signal);
    step = (max_val - min_val) / (2^N);
    quantized_signal = round((predicted_signal - min_val) / step) * step +
min_val;
end
function dequantized_signal = dequantize(quantized_signal, N)
    min_val = min(quantized_signal);
    max_val = max(quantized_signal);
    step = (max_val - min_val) / (2^N);
    dequantized_signal = quantized_signal / step;
end
function mse = calculate_MSE(signal, reconstructed_signal)
    mse = mean((signal - reconstructed_signal).^2);
end
function plotSignals(signal, reconstructed_signal, N, p, subset_number)
    if p == 5 || p == 10
        figure;
        subplot(2, 1, 1);
        plot(signal);
        title(['Original Signal for Subset ', num2str(subset_number), ', N =
', num2str(N), ', p = ', num2str(p)]);
        xlabel('Sample');
        ylabel('Amplitude');

        subplot(2, 1, 2);
        plot(reconstructed_signal);
        title(['Reconstructed Signal for Subset ', num2str(subset_number),
', N = ', num2str(N), ', p = ', num2str(p)]);
        xlabel('Sample');
    end
end
end

```