ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ

1η Εργαστηριακή Ασκηση

Καλάργαρης Χαράλαμπος, ΑΜ:3929

25/1/2011

Contents

1 Κωδικοποίηση PCM

Η κωδικοποίηση PCM είναι μια μέθοδος κωδικοποίησης κυματομορφής, η οποία μετατρέπει ένα αναλογικό σήμα σε ψηφιακά δεδομένα. Τυπικά, το σχήμα PCM αποτελείται από τρία βασικά τμήματα: έναν δειγματολήπτη, έναν κβαντιστή, και έναν κωδικοποιητή. Η έξοδος του κωδικοποιητή είναι μια ακολουθία από κωδικές λέξεις (σύμβολα) σταθερού μήκους N bits. Σε αυτο το κεφαλαιο, θα επικεντρωθούμε στη λειτουργία του κβαντιστή.

1.1 Ομοιόμορφος Κβαντιστής

Ενας ομοιόμορφος κβαντιστής παίρνει σαν είσοδο το σήμα του δειγματολήπτη και το κβαντιζει.Το χαρακτηριστικό του **ομοιόμορφου** κβαντιστή είναι οτι όλες οι περιοχές κβάντισης έχουν το ίδιο εύρος, ένω οι στάθμες κβάντισης επιλέγονται ως κέντρα των περιοχών κβάντισης. Αύτο διασφαλίζει την ελαχιστοποίηση του θορύβου κβάντισης αρα και την καλύτερη λειτουργία του κβαντιστή.

Ο κβαντιστής που έπρεπε να υλοποιήσουμε για την άσκηση σε Matlab φαίνεται παρακάτω:

```
function [xq,centers] = my_quantizer(x,N,min_value,max_value)

% Limit to dymanic area

for p = 1 : length(x)
    if x(p)>max_value
        x(p)=max_value;
    end
    if x(p)<min_value
        x(p)=min_value;
    end
end

% levels of quantization .
levels = 2^N;
% length of quantization's area
Δ = (max_value - min_value) / levels;
% centres of quantization's area
centers(1) = min_value + Δ/2;

for i=1:levels-1,
    centers(i+1) = Δ + centers(i);
end
for i=1:length(x)
    for p = 1: levels
        if x(i) ≤ min_value + Δ*p
            xq(i)=centers(p); % Final output signal
            break;
        end
    end
end

end
```

Σύμφωνα με τα δεδομένα της άσκησης ο κβαντιστής που υλοποιήσαμε εχει τα παρακάτω δεδομένα:

```
[xq,centers] = my_quantizer(x,N,min_value,max_value);
x: το σήμα εισόδου υπό μορφή διανύσματος
N: ο αριθμός των bits που θα χρησιμοποιηθούν
max_value: η μέγιστη αποδεκτή τιμή του σήματος εισόδου
min_value: η ελάχιστη αποδεκτή τιμή του σήματος εισόδου
xq: το διάνυσμα του σήματος εξόδου κωδικοποιημένο ως εξής: τα
επίπεδα κβάντισης αναπαρίστανται με τους ακεραίους 1,2,..,2N,όπου
το μεγαλύτερο θετικό επίπεδο κβάντισης αντιστοιχεί στον ακέραιο 1.
Οι ακέραιοι αυτοί μπορούν να αναπαρασταθούν δυαδικά με N bits.
centers: τα κέντρα των περιοχών κβάντισης.
```

Οπως φαινεται και στον κώδικα που παρέθεσα παραπάνω το μήκος της κβάντισης υπολογίζεται απο τον τύπο $\Delta = \frac{x_{max}-x_{min}}{2^N}$, ενω τα άκρα των περιοχών κβάντισης υπολογίζονται $T_1 = x_{min}, T_2 = x_{min} + \Delta, T_3 = x_{min} + 2\Delta$ κοκ.Τελος τα κέντρα περιοχών κβάντισης υπολογίζονται ως $\hat{x_1} = T_1 + \Delta/2, \hat{x_2} = \hat{x_1} + \Delta, \hat{x_3} = \hat{x_1} + 2\Delta$ κοκ.

1.2 Μη Ομοιόμορφος Κβαντιστής

Το πρόγραμμα Matlab που υλοποοιήθηκε για τον μη ομοιόμορφο κβαντιστή συμφωνμα με τον αλγόριθμο του Lloyd-Max φαινεται παρακάτω:

```
function [xq,centers,p,D] = Lloyd_Max(x,N,min_value, max_value)

j_max = 100;
e = 10^-5;
levels = 2^N; % levels of quantization .
D = zeros(j_max,1);
T = zeros(levels+1,1);



% centres of uniform's quantization area
Δ = (max_value - min_value) / levels;

centers(1) = min_value + Δ/2;

for i=1:levels-1,
    centers(i+1) = centers(i)+Δ;
end

j = 1;
D_old = 0;
D_new = 1;

while abs(D_new-D_old) > e

   T(1) = min_value;
    for h = 2 : levels
        T(h) = (centers(h-1)+centers(h)) / 2;
    end
    T(levels+1) = max_value;

shows = zeros(levels,1);
sum = zeros(levels,1);
```

```
p = zeros(levels ,1);
    for i=1:length(x)
        for l = 2: levels+1
            if x(i) ≤ T(l)
                xq(i)=centers(l-1);
                sum(l-1) = sum(l-1) + x(i);
                shows(l-1) = shows(l-1) + 1;
                p(l-1) = p(l-1) + 1;
                break;
            end
        end
        D(j) = D(j) + (x(i) - xq(i))^2; % deformation computation
    end

D(j) = D(j)/length(x); % final deformation

  for g=1:levels
      p(g) = p(g)/length(x); % final probabilities
   end

   for w = 1 : levels
       if shows(w) > 0
           centers(w) = sum(w)/shows(w); %new centres
       end
   end

D_old = D_new;
D_new = D(j);
j = j + 1;
end

D = D(1:j-1);
end
```

Να σημειωθεί οτι για την υλοποίηση του παραπάνω κώδικα συνεργάστηκα με την συμφοιτήτριά μου,Μαρία Παπαδουράκη ΑΜ:3999.

2 Κωδικοποίηση Huffman

Ο αλγόριθμος Huffman αποτελεί έναν κωδικοποιητή διακριτών πηγών, που αντιστοιχεί τα σύμβολα εισόδου σε κωδικές λέξεις μεταβλητού αριθμού bits. Μαζί με την εκφώνηση της άσκησης, μας δίνεται το αρχείο huffman.m με το οποίο θα υλοποιήσουμε τον αλγόριθμο Huffman:

```
[code,len] = huffman(p);
p: διάνυσμα πιθανοτήτων εμφάνισης κάθε συμβόλου
code: ο κώδικας καθενός συμβόλου (μεταβλητή τύπου string)
len: το μήκος της κωδικοποίησης κάθε συμβόλου σε bits.
```

3 Πήγες

Σύμφωνα με τη εκφώνηση της άσκησης θα πρέπει να κωδικοποιήσουμε δύο πηγες, A και B.

1. **Πηγη A**

Η έξοδος της πρώτης πηγής A είναι μια τυχαία διαδικασία, που ακολουθεί την εκθετική κατανομή. Η συνάρτηση πυκνότητας πιθανότητας $f_x(x)$ της εκθετικής κατανομής δίνεται παρακάτω:

$$f_x(x) = \begin{cases} e^{-x} & : \quad x > 0 \\ 0 & : \quad other \end{cases}$$

Για να παράγουμε τα M δείγματα που μας ζηταει η ασκηση εκτελούμε τον παρακάτω κώδικα στην Matlab:

```
function [x] = Source_A
% Source's A output signal
t = (randn(10000,1)+j*randn(10000,1))/sqrt(2);
x = abs(t).^2;
end
```

2. **Πηγή B**

Η δεύτερη πηγή B είναι τα εικονοστοιχεία (pixels) μιας grayscale εικόνας. Μια τέτοια εικόνα μπορεί να θεωρηθεί σαν ένα πίνακας με εικονοστοιχεία όπου κάθε εικονοστοιχείο αντιστοιχεί σε ένα byte πληροφορίας. Κάθε εικονοστοιχείο επομένως λαμβάνει μια τιμή στο δυναμικό εύρος [0:255] η οποία αντιστοιχεί σε ένα από τα 256 επίπεδα φωτεινότητας (το 0 αντιστοιχεί στο μαύρο και το 255 στο λευκό).

Ο κώδικας σε Matlab για την πηγη B ειναι :

```
function [x] = Source_B
%  Source's B output signal
load lenna.mat;
x = x_le(:);
x=(x-128)/128;
end
```

Τρέχοντας τον παρακάτω κώδικα σε Matlab παίρνουμε σαν έξοδο την εικόνα παρακάτω:

```
function Source_B_image

load lenna.mat;
imshow(uint8(x_le));
end
```

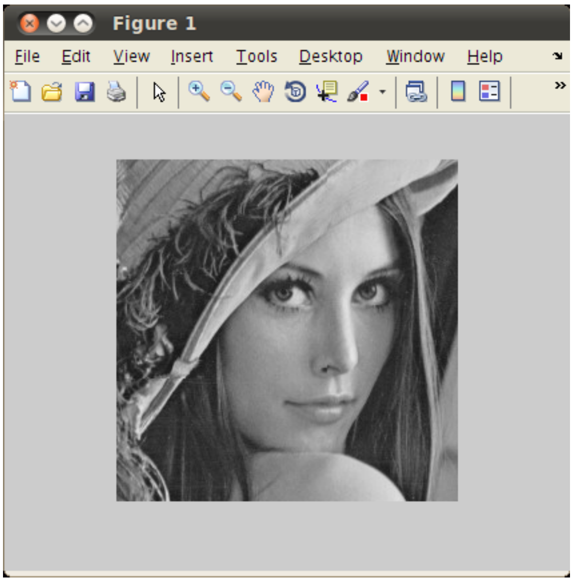

Figure 1: Εικόνα απο Matlab

4 Ερωτήσεις - Ζητούμενα

4.1 Πρώτο Ερώτημα

Χρησιμοποιώντας τον ομοιόμορφο κβαντιστή που υλοποιήσατε, κωδικοποιήστε την **πηγή A** για $min\_value = 0$ και $max\_value = 4$, και $N = 2$, $N = 4$ και 6 bits.

4.1.1 A Υποερώτημα

Υπολογίστε το SQNR (dB) στην έξοδο του κβαντιστή. Υπολογίστε τη θεωρητική τιμή του SQNR. Συγκρίνετε και σχολιάστε τα αποτελέσματα της θεωρητικής και της πειραματικής τιμής του SQNR που υπολογίσατε.

Για το θεώρητικο κομματι υλοιποιήσαμε τον παρακάτω κωδικα σε Matlab:

```
function [sqnr] = theoritikoSQNR_A
[x] = Source_A;
min_value = 0;
max_value = 4;
for k=1:3

    N=2*k;

    % uniform quantizer
    [xq,centers] = my_quantizer(x,N,min_value,max_value);
    levels = 2^N;
    ∆ = (max_value - min_value) / levels;

    % Compute signal's power
    syms z;
    Psignal(k) = int(z^2*exp(-z),z,0,inf);
    Psignal(k)=double(Psignal(k));

```

```
    % Compute noise's power
    Pnoise(k) = 0;

    for i=1:levels-1
        Pnoise(k) = Pnoise(k) + int((z-centers(i))^2*exp(-z),z,(i-1)*Δ,i*Δ);
    end
    Pnoise(k) = Pnoise(k) + int((z-centers(levels))^2*exp(-z),z,(levels-1)*Δ,inf);
    Pnoise(k) = double(Pnoise(k));

    % SQNR
    sqnr(k)= 10*log10(double(Psignal(k)) / double(Pnoise(k)));
end
end
```

Τύποι για τους Θεωρητικούς υπολογισμους:

- **Ισχύς θορύβου:** $P_{noise} = E[(X-\hat{X})^2] = \int_{-\infty}^{+\infty}(x-\hat{x})^2 f_X(x)\,\mathrm{d}x = \int_0^{+\infty}(x-\hat{x})^2 e^{-x}\,\mathrm{d}x$
- **Ισχύς σήματος:** $P_{signal} = E[X^2] = \int_{-\infty}^{+\infty} x^2 f_X(x)\,\mathrm{d}x = \int_0^{+\infty} x^2 e^{-x}\,\mathrm{d}x = 2$
- **SQNR(dB)**$=10 * \log_{10} \frac{P_{signal}}{P_{noise}}$

Για το πειραματικό κομματι υλοιποιήσαμε τον παρακάτω κωδικα σε Matlab:

```
function [sqnr] = peiramatikoSQNR_A
[x] = Source_A;
min_value = 0;
max_value = 4;
for i=1:3
    N=2*i;

    % uniform quantizer
    [xq,centers] = my_quantizer(x,N,min_value,max_value);

    % Signal and noise power
    Pnoise(i) = mean((x-xq').^2);
    Psignal(i) = mean (x.^2);

    %SQNR
    sqnr(i) = 10*log10(Psignal(i) / Pnoise(i));
end
end
```

Παρακάτω φαινονται τα αποτελέσματα που πήραμε απο το Matlab τρέχοντας τις εξής εντολες

```
theoritikoSQNR_A;
peiramatikoSQNR_A;
```

N	SQNR Θεωρητικο	SQNR Πειραματικο
2	11.4267	11.3652
4	16.3245	16.3868
6	17.1995	17.2913

Αναλύοντας τα αποτελέσματα του παραπάνω πινακα εξάγουμε τα εξής συμπεράσματα:

1. Τα πειραματικά αποτελέσματα ειναι πολύ κοντά στα θεωρητικά.

2. Παρατηρούμε οτι όσο αυξάνονται τα επίπεδα κβάντισης, αυξάνεται και το SQNR.Αρα ειναι μικροτερη η παραμόρφωση οταν έχουμε περισσότερα επίπεδα.Κατι που είναι πολύ λογικό.

4.1.2 B Υποερώτημα

Ποια είναι η πιθανότητα να βρεθεί η είσοδος του κβαντιστή εκτός της δυναμικής περιοχής του (πιθανότητα υπερφόρτωσης); Υπολογίστε θεωρητικά και πειραματικά την πιθανότητα αυτή.

Για το πειραματικό κομματι υλοιποιήσαμε τον παρακάτω κωδικα σε Matlab:

```
function [pith] = overload_probability
[x] = Source_A;
k = 0;
for i = 1 : length(x)
    % times ektos tis dunamikhs perioxhs
     if (x(i)>4) | (x(i)<0)
         k = k +1;
     end
end
% diairoume me to synoliko plithos
k = k/length(x);

pith=k;
end
```

Υπολογισμός θεωρητικής τιμής για την πιθανότητα υπερφόρτωσης:

$$\int_{4}^{+\infty} f_X(x)\,\mathrm{d}x = \int_{4}^{+\infty} e^{-x}\,\mathrm{d}x = e^{-4} = 0.0183$$

Θεωρητικό	Πειραματικό
0,0183	0,0180

Παρατηρώντας τον παραπάνω πίνακα βλεπουμε οτι υπάρχει πολύ μικρή απόκλιση μεταξύ του θεωρητικού και πειραματικού σκέλους.

4.1.3 C Υποερώτημα

Υλοποιήστε μια συνάρτηση που να υπολογίζει (θεωρητικά) την πιθανότητα εμφάνισης κάθε στάθμης του κβαντιστή. Για να επαληθεύσετε τους υπολογισμούς σας μετρήστε κάθε πιθανότητα εμφάνισης και συγκρίνετε την με τη θεωρητική τιμή που υπολογίσατε. Ποια είναι η αποδοτικότητα της κωδικοποίησης PCM;

Για το θεωρητικό υπολογισμο της πιθανότητας εμφάνισης κάθε στάθμης του κβαντιστή υλοποιήσαμε σε Matlab το παρακάτω κωδικα:

```
function [prob] = theoritikh_1C(N)
min_value = 0;
max_value = 4;
levels = 2^N;
Δ = (max_value - min_value) / levels;
prob = zeros(levels,1);
syms z
```

```
for i=1:levels-1
    prob(i)=int(exp(-z),z,(i-1)*Δ,i*Δ);
end
    prob(levels)=int(exp(-z),z,(levels-1)*Δ,inf);
end
```

Για το πειραματικό υπολογισμο της πιθανότητας εμφάνισης κάθε στάθμης του κβαντιστή υλοποιήσαμε σε Matlab το παρακάτω κωδικα:

```
function [prob] = peiramatikh_1C(N)
[x] = Source_A;
min_value = 0;
max_value = 4;
levels = 2^N;
%uniform quantizer
[xq,centers] = my_quantizer(x,N,min_value,max_value);
prob=zeros(levels,1)
%pithanothta kathe level
for i=1:length(x);
    for k = 1: levels;
            if xq(i)==centers(k);
        prob(k) = prob(k) + 1;
        break;
        end
    end
end
for t=1:levels
        prob(t) = prob(t)/length(x);
end
end
```

Παρακάτω σας παραθέτω τα αποτελεσματα απο το Matlab τρέχοντας τις εντολες:

```
theoritikh_1C(N);
peiramatikh_1C(N);
%οπου N=2, 4, 6.
```

	Θεωρητικο	Πειραματικο
	0.6321	0.6294
$N = 2$	0.2325	0.2277
	0.0855	0.0915
	0.0497	0.0497

	Θεωρητικο	Πειραματικο
	0.2212	0.2244
	0.1723	0.1691
	0.1342	0.1319
	0.1045	0.1067
	0.0814	0.0800
	0.0634	0.0657
	0.0494	0.0454
$N = 4$	0.0384	0.0397
	0.0299	0.0322
	0.0233	0.0235
	0.0182	0.0190
	0.0141	0.0129
	0.0110	0.0120
	0.0086	0.0094
	0.0067	0.0069
	0.0235	0.0212

	Θεωρητικο	Περαματικο
$N = 6$	0.0606	0.0642
	0.0569	0.0557
	0.0535	0.0528
	0.0502	0.0499
	0.0472	0.0453
	0.0443	0.0425
	0.0416	0.0431
	0.0391	0.0392
	0.0367	0.0372
	0.0345	0.0339
	0.0324	0.0322
	0.0305	0.0281
	0.0286	0.0297
	0.0269	0.0280
	0.0253	0.0251
	0.0237	0.0245
	0.0223	0.0228
	0.0209	0.0228
	0.0197	0.0217
	0.0185	0.0180
	0.0174	0.0167
	0.0163	0.0145
	0.0153	0.0127
	0.0144	0.0150
	0.0135	0.0141
	0.0127	0.0126
	0.0119	0.0122
	0.0112	0.0111
	0.0105	0.0125
	0.0099	0.0078
	0.0093	0.0110
	0.0087	0.0086
	0.0082	0.0086
	0.0077	0.0089
	0.0072	0.0065
	0.0068	0.0070
	0.0064	0.0066
	0.0060	0.0051
	0.0056	0.0057
	0.0053	0.0052
	0.0050	0.0056
	0.0047	0.0038
	0.0044	0.0041
	0.0041	0.0040
	0.0039	0.0054
	0.0036	0.0041
	0.0034	0.0034

0.0032	0.0032
0.0030	0.0031
0.0028	0.0025
0.0027	0.0022
0.0025	0.0020
0.0023	0.0025
0.0022	0.0022
0.0021	0.0022
0.0019	0.0021
0.0018	0.0014
0.0017	0.0019
0.0016	0.0013
0.0015	0.0015
0.0014	0.0014
0.0013	0.0016
0.0013	0.0014
0.0195	0.0180

Πραγματικά κοιτώντας τα παραπάνω αποτελεσματα βλέπουμε οτι οι πειραματικες τιμες επαληθεύουν τις θεωρητικές.

Για να βρούμε ποια είναι η αποδοτικότητα της κωδικοποίησης PCM σχεδιάσαμε το παρακάτω προγραμμα Matlab λαμβάνοντας υποψίν τους τύπους που εχει το βιβλίο για την εντροπία και για την αποδοτικότητα:

```
function [effec] = effeciency
for j=1:3

    N=j*2;
    % Edw mporoume na baloume kai thn sunarthsh theoritikh_1C
    [prob] = peiramatikh_1C(N);
    levels = 2^N;
    entrop = 0;

    for w = 1: levels
        if prob(w)>0
            entrop = entrop - prob(w)*log2(prob(w));
        end
    end

    effec(j) = entrop/N;
end
end
```

Παρακάτω ειναι τα αποτελέσματα για την αποδοτικότητα της κωδικοποίησης PCM για N=2 ,4 ,6

N	Αποδοτικοτητα
2	0.7111
4	0.8452
6	0.8874

Απο τον παραπάνω πινακα γίνετε φανερο οτι η αύξηση του N οδηγεί και σε αύξηση της αποδοτικότητας.

4.1.4 D Υποερώτημα

Χρησιμοποιήστε τη ρουτίνα Huffman για να κωδικοποιήσετε την πηγή A2 που προέκυψε από το κβαντισμό της πηγής A (για N=2, N=4 και 6 bits). Ποια είναι η αποδοτικότητα του κώδικα Huffman;

Για βρούμε ποια είναι η αποδοτικότητα του κώδικα Huffman υλοποιήσαμε σε Matlab το παρακάτω κωδικα:

```
function [h_effic] = huffman_efficiency
for y=1:3

    N=y*2;
    [prob] = peiramatikh_1C(N);
    levels = 2^N;
    entrop = 0;

    for w = 1: levels
        if prob(w)>0
            entrop = entrop - prob(w)*log2(prob(w));
        end
    end

    p = prob;

    % Huffman
    [code,len]=huffman(p);

    % meso mhkos Huffman.
    L = 0;

    for i = 1 : levels;
        L = L + len(i)*p(i);
    end

    % apodotikothta Huffman
    h_effic(y) = entrop/L;
end
end
```

Τα αποτελέσματα του παραπάνω κώδικα φαινονται στον πινακα παρακάτω:

N	Αποδοτικοτητα
2	0.9525
4	0.9912
6	0.9942

Απο τον παραπάνω πινακα γίνετε φανερο οτι η αύξηση του N οδηγεί και σε αύξηση της αποδοτικότητας.Αν και γενίκα μπορούμε να πουμε οτι και στις τρείς περιπτώσεις η αποδοτικότητα ειναι πάρα πολύ καλή.

4.2 Δεύτερο Ερώτημα

Κωδικοποιήστε την πηγή B χρησιμοποιώντας τον α) τον ομοίομορφο και β) τον μη ομοιόμορφο κβαντιστή που υλοποιήσατε για min_value=-1 και max_value=1 και N=2,4,6 bits

4.2.1 Α Υποερώτημα

Παραθέστε γραφική παράσταση που θα παρουσιάζει το SQNR σε συνάρτηση με το N για τους δύο κβαντιστές. Σχολιάστε τα αποτελέσματα.

Παρακάτω παραθέτω τις δυο συναρτήσεις που υλοποιήσα για υπολογίζω SQNR στην έξοδο του ομοίομορφου και μη ομοίομορφου κβαντιστή για την πηγη B:

```
function [sqnr] = not_uniform_SQNR_B

min_value = -1;
max_value = 1;

[x] = Source_B;

for d=1:3
    N=2*d;

    % not uniform quantizer.
    [xq,centers,p,D] = Lloyd_Max(x,N,min_value, max_value);

    %signal and noise power
    Psignal(d) = mean (x.^2);
    Pnoise(d) = mean((x-xq').^2);

    %SQNR
    sqnr(d) = 10*log10(Psignal(d) / Pnoise(d));
end

end
```

```
function [sqnr] = uniform_SQNR_B
min_value = -1;
max_value = 1;

[x] = Source_B;

for d=1:3
    N=2*d;
    % uniform quantizer
    [xq,centers] = my_quantizer(x,N,min_value,max_value);

    %Signal and noise power
    Psignal(d) = mean (x.^2);
    Pnoise(d) = mean((x-xq').^2);

    %SWNR
    sqnr(d) = 10*log10(Psignal(d) / Pnoise(d));
end

end
```

Ο παρακάτω κώδικας μας δινει την γραφική παρασταση που μας ζητάει το ερώτημα:

```
%sunarthseis apo tis opoies tha kanoume thn grafiki parastash
[sqnr_1] = not_uniform_SQNR_B
[sqnr_2] = uniform_SQNR_B
```

```
%grafiki parastash gia not_uniform_SQNR_B
plot([2,4,6],sqnr_1,'rs');
hold on

%grafiki parastash gia uniform_SQNR_B
plot([2,4,6],sqnr_2,'gs');
%legends
legend('Not uniform quantizer','Uniform quantizer');
%aksones
xlabel('N level');
ylabel('SQNR(dB)');
%titlos
title('Graphs SQNR Source B');
hold off
```

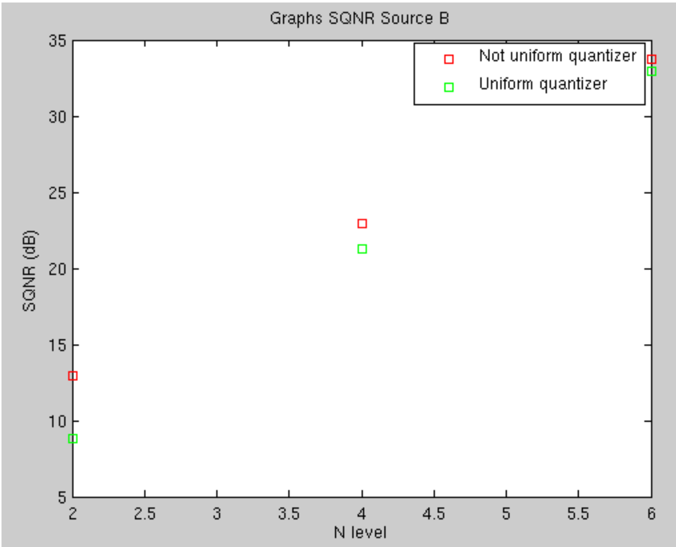

Figure 2: Γραφικη παρασταση

N	Uniform SQNR	NOT Uniform SQNR
2	8.8291	12.9289
4	21.2825	22.9472
6	32.9253	33.7278

Βλεπουμε οτι ο μη ομοιόμορφος κβαντιστής (κοκκινο χρώμα) εχει υψηλότερο SQNR κατι που δικαιολογείται απο το γεγονος οτι δεν ακολουθεί ομοιόμορφη κατανομή.

4.2.2 Β Υποερώτημα

Παραθέστε την προκύπτουσα εικόνα για κάθε περίπτωση.Σχολιάστε τα αποτελέσματα.

Ο παρακάτω κώδικας μας δινει εικόνες που μας ζητάει το ερώτημα:

```
%stoixeia shmatos eisodou
```

```
x = Source_B;
max_value = 1;
min_value = -1;

for f=1:3
    N=2*f;
    %uniform quantizer
    [xq,centers] = my_quantizer(x,N,min_value,max_value);
    %image
    xq2 = reshape(128*xq +128,256,256);
    figure(f);
    imshow(uint8(xq2));

    %not uniform quantizer
    [xq,centers,p,D] = Lloyd_Max(x,N,min_value, max_value);
    %image
    xq3 = reshape(128*xq +128,256,256);
    figure(f+3); %f+3 gia na vgoun me swsth seira
    imshow(uint8(xq3));
end
```

Συμπεράσματα:

1. Παρατηρουμέ οτι οσο αυξάνεται το N τοσο για τον ομοιόμορφο κβαντιστη οσο και για τον μη ομοιόμορφο η ποιότητα της εικονας γινεται ολο και καλύτερη.

2. Παρατηρουμε οτι ο μη ομοιόμορφος κβαντιστής παράγει καλύτερες εικονες απο οτι ο ομοιόμορφος.αυτο οφειτελεται στο γεγονος οτι ο μη ομοιόμορφος κβαντιστής έχει καλύτερο SQNR.

Παρακάτω σας παραθέτω τις εικόνες

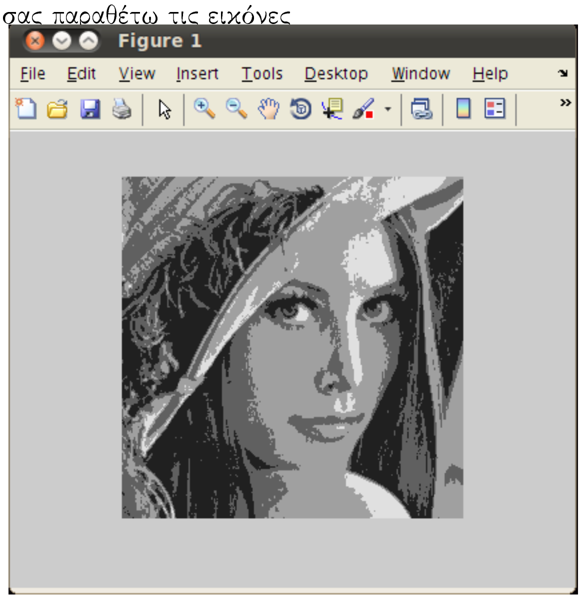

Figure 3: Ομοιόμορφος Κβαντιστης N=2

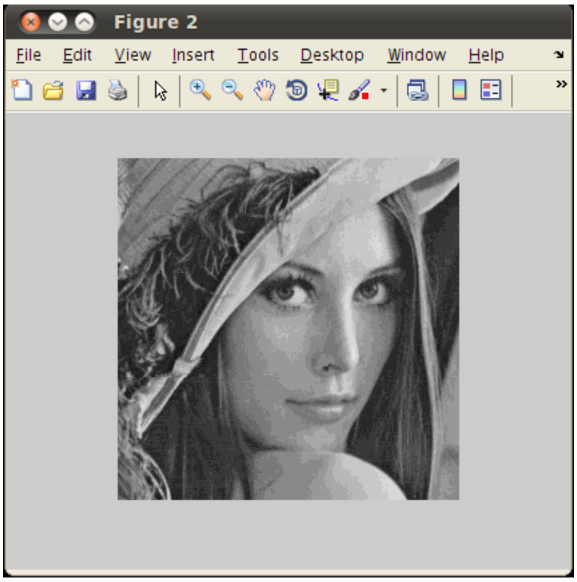

Figure 4: Ομοιόμορφος Κβαντιστης N=4

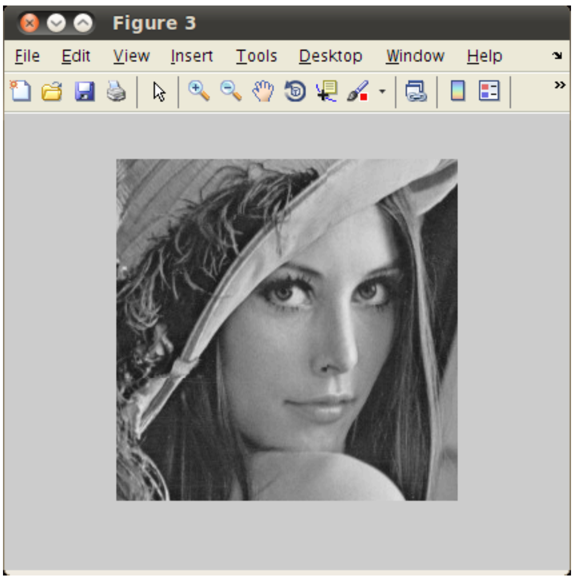

Figure 5: Ομοιόμορφος Κβαντιστης N=6

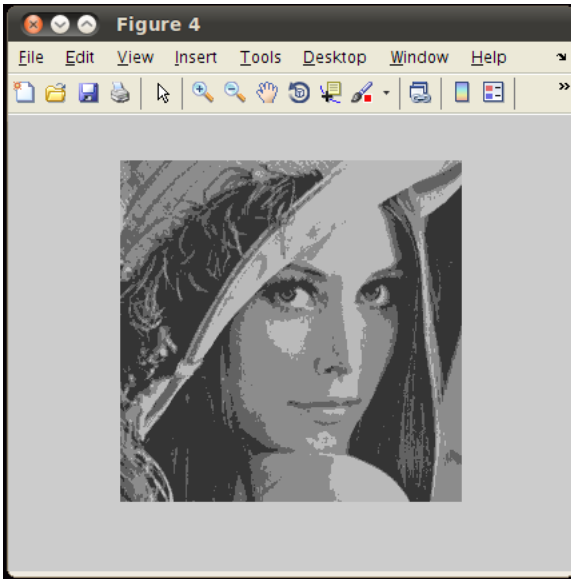

Figure 6: Μη Ομοιόμορφος Κβαντιστης N=2

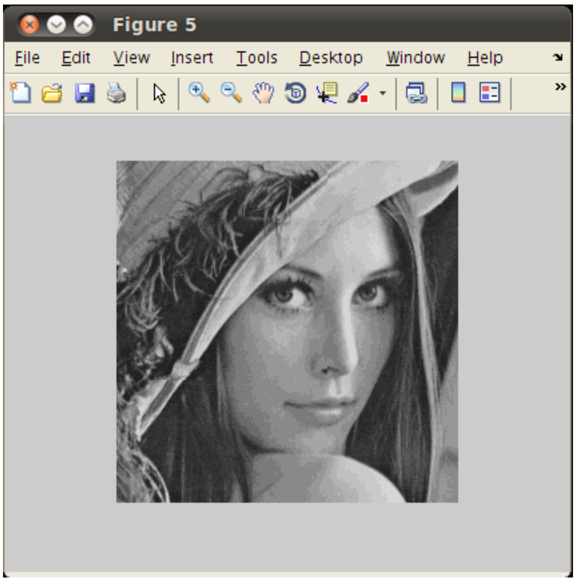

Figure 7: Μη Ομοιόμορφος Κβαντιστης N=4

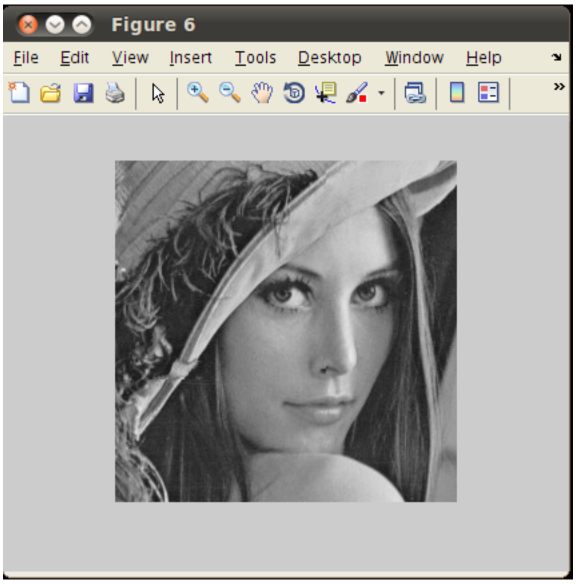

Figure 8: Μη Ομοιόμορφος Κβαντιστης N=6