

ΑΝΑΦΟΡΑ 1ης ΕΡΓΑΣΙΑΣ ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΟΙΚΙΝΩΝΙΕΣ

ΣΠΕΝΤΖΑΡΗΣ ΠΑΝΑΓΙΩΤΗΣ 1071110

Ακαδημαϊκό έτος 2021-22



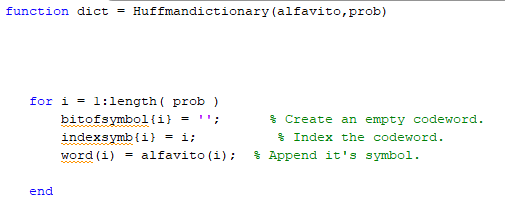
**ΑΣΚΗΣΗ 1: ΚΩΔΙΚΟΠΟΙΗΣΗ HUFFMAN**

* 1. **Huffmandictionary**

**Aρχείο:Huffmandictionary.m**

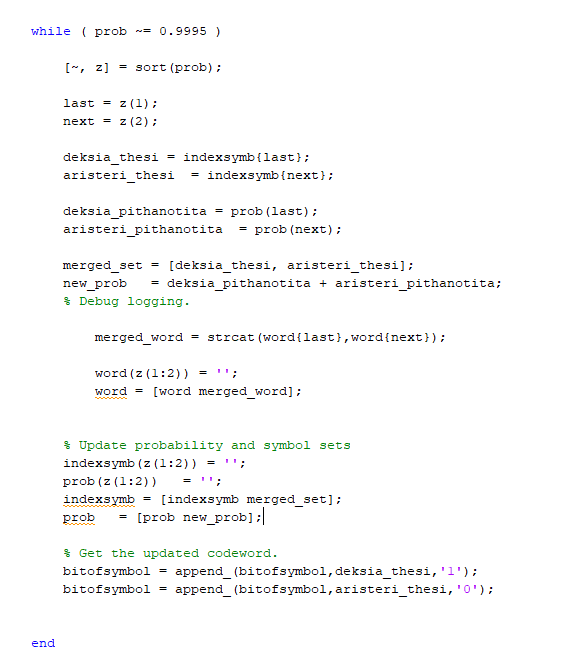
Βήμα 1:

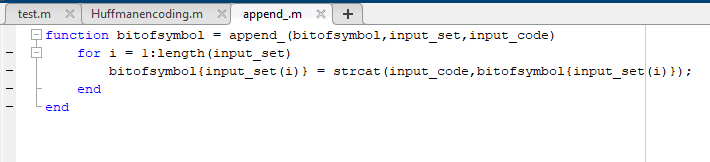
Η συνάρτηση που κατασκεύασα παίρνει ως είσοδο το αλφάβητο με τα σύμβολα από a έως το space και ως δεύτερη είσοδο τις πιθανότητες των συμβόλων που μας δόθηκε.Η πάρακατω for τρέχει όσο είναι το μέγεθος των συμβόλων και μέσα σε αυτήν δημιουργώ το bitofsymbol για την δυαδική κωδικοποίηση,το indexsymb που θα αντιστοιχεί την κωδικοποίηση στο αντίστοιχο σύμβολο καθώς και το word(i) οπου θα παίρνει όλα τα σύμβολα.



Βήμα 2:

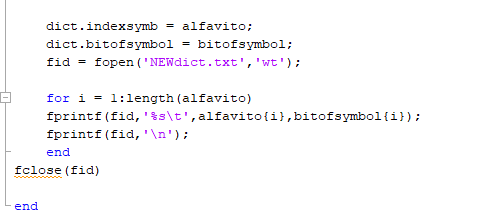
Η while θα τρέξει μέχρι να φτάσει το άθροισμα όλων των πιθανοτήτων (sum(prob)) ,ως πρώτο βήμα κάνουμε sort τον πίνακα με τις πιθανότητες ,αποθηκεύουμε στο index 1,2 της μεταβλητής z τα 2 πρώτα στοιχεία από τον sorting πίνακα.Υπολογίζω την συνολική πιθανότητα των δύο αυτών συμβόλων καθώς και την θέση τους.Και κάθε φορά κάνω update το prob και indexsymb στην γραμμή 40 και 41 και αδειάζουν ώστε να επαναληφθεί ξανά η λούπα για τίς υπόλοιπες πιθανότητες του νέου ταξινομημένου πίνακα prob**.Η append\_** **είναι μία επιπρόσθετη συνάρτηση** που έφτιαξα ώστε να δίνω την κωδικοποίηση 0 και 1 ανάλογα την θέση που βρίσκεται ο κάθε κόμβος.



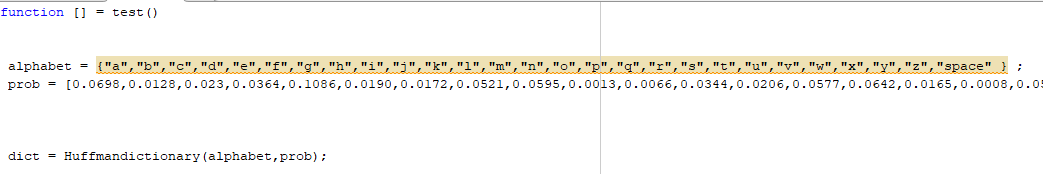


Βήμα 3:

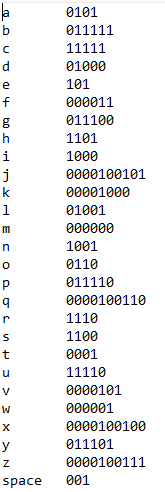
Στο cell dict.indexsymb αποθηκεύεται το αλφάβητο ενώ στο dict.bitofsymbol η δυαδική κωδικοποίηση τους.Τα αποθηκεύω σε ένα αρχείο txt σε 2 στήλες αντίστοιχα.



Βήμα 4:Δημιουργία ενός αρχείου.m όπου θα τρέχω τους κώδικες για να εξάγουμε τα αποτελέσματα μας.



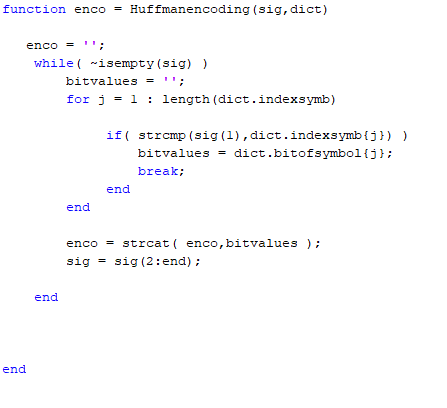
ΑΠΟΤΕΛΕΣΜΑΤΑ:



**HUFFMANENCODING**

**Αρχείο: Huffmanencoding.m**

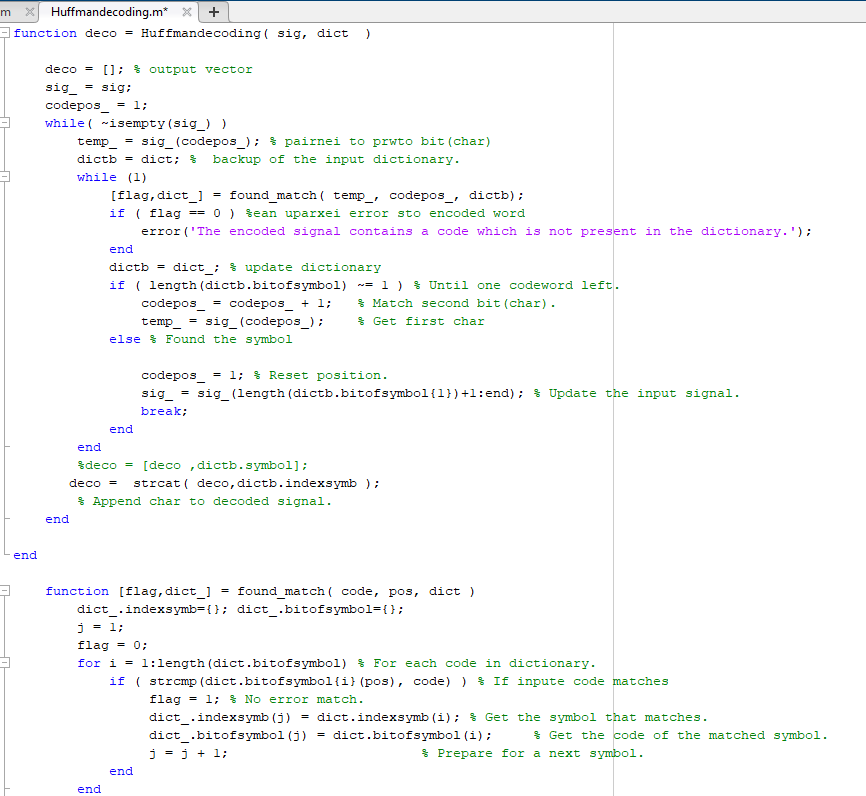
Η συνάρτηση που έφτιαξα παίρνει ως είσοδο ένα αρχείο που περιέχει μία ακολουθία από σύμβολα(όπως το cvxopt.txt αρχείο που μας δώσατε),καθώς και ως δεύτερη είσοδο το dict ώστε να αντλούμε δεδομένα από το huffmandictionary.Η for θα τρέξει έως το μέγεθος των συμβόλων που του έχουμε δώσει στο huffmandict,παίρνει το πρώτο γράμμα του κειμένου και αν βρει ταίριασμα με κάποιο σύμβολο αναθέτει στο bitvalues την δυαδική τιμή του. Βγαίνει από την λουπα αποθηκεύεται στο enco ,οπου στην συνέχεια γίνεται το ίδιο για τα υπόλοιπα στοιχεία του κειμένου μέχρι τέλους.

****

**HUFFMANDECODING**

**Αρχείο: Huffmandecoding.m**

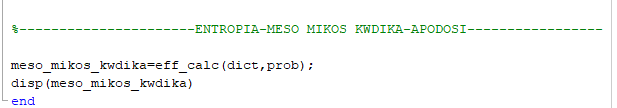
Παίρνει ως είσοδο το αρχείο με την δυαδική ακολουθία και ως δεύτερη είσοδο το dict.Η while τρέχει όσο το αρχείο έχει πληροφορία μέσα.Η δεύτερη while εκτελεί μέσα της την συνάρτηση found\_match μέσα στην οποία γίνεται ο έλεγχος και το ταίριασμα της κάθε δυαδικής ακολουθίας με το σύμβολο της.Η for στην γραμμή 36 εκτελείται όσο είναι το μέγεθος του dict.bitofsymbol από το huffmandictionary.Εάν βρεί ταίριασμα τότε στα dictionaries που έχουν δημιουργηθεί εντός αυτής δίνουμε το σύμβολο που ταιρίαζει με την δυαδική ακολουθία , την δυαδική ακολουθία και αυξάνουμε τον μετρήτη j ώστε να βρεί και τα υπόλοιπα σύμβολα.



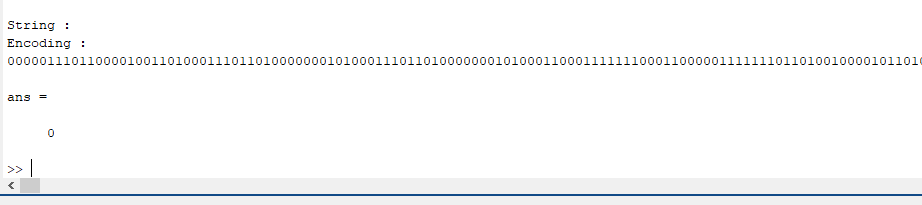
**ΕΡΩΤΗΜΑ 2:**

Τρέχω τα αποτελέσματα εντός του αρχείου **testerwtima2.m** που παραθέτω και με screen παρακάτω και εξάγω τα αποτελέσματα:

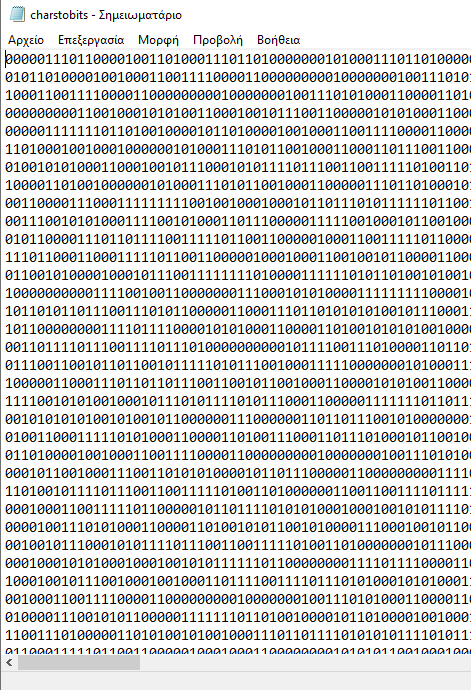




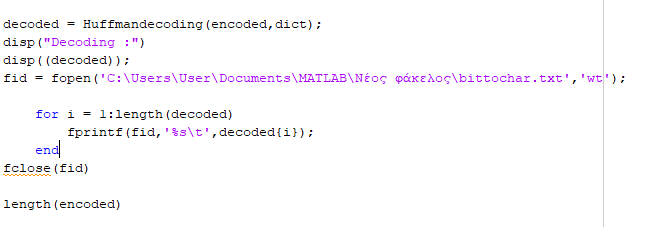
Τα αποτελέσματα της κωδικοποίησης της πηγής Α:



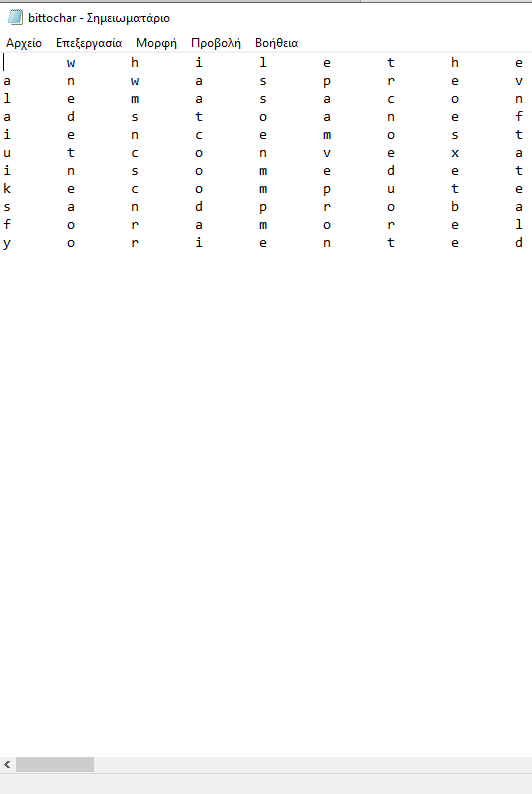
Τα οποία επίσης βρίσκονται στο charstobits.txt όπου τα αποθήκευσα:



α)Καλώ την συνάρτηση huffmandecoding η οποία παίρνει σαν πρώτο όρισμα, την έξοδο της huffmanencoding δηλαδή την κωδικοποιημένη δυαδική πηγή και σαν δεύτερο όρισμα την έξοδο της huffmandictionary.



Τα αποτελέσματα αποθηκεύονται στο txt και εμφανίζονται με αυτό τον τρόπο:



(β) Έχω δημιουργήσει 2 συναρτήσεις την

[out\_] = eff\_calc( dict, prob )

out\_ = average\_cdl( dict, prob )

Η πρώτη συνάρτηση υπολογίζει το μέσο μήκος κώδικα και την εντροπία της πηγής.Μέσα στην συνάρτηση eff\_calc καλώ την average\_cdl.

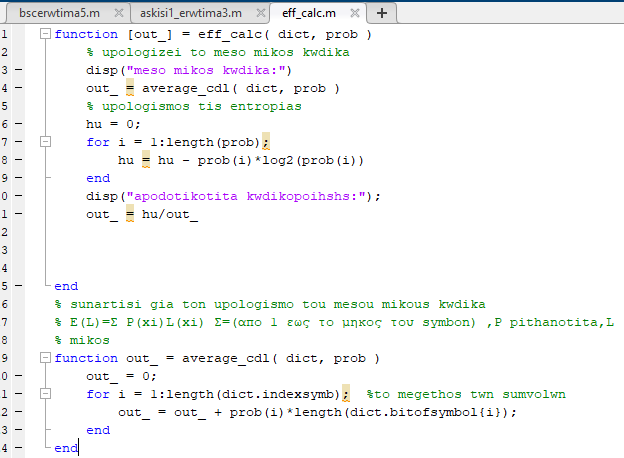
Τρέχω την testerwtima2.m και τα αποτελέσματα:

Εντροπία της πηγής:

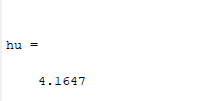


Μέσο Μήκος Κώδικα:

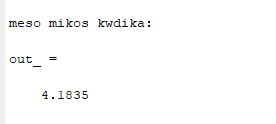




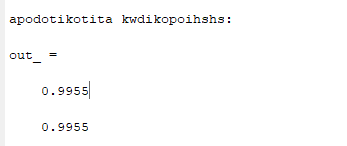
ΤΗΣ ΕΝΤΡΟΠΙΑΣ ΤΗΣ ΠΗΓΗΣ:



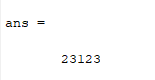
ΜΕΣΟ ΜΗΚΟΣ ΚΩΔΙΚΑ:



ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΚΩΔΙΚΟΠΟΙΗΣΗΣ:



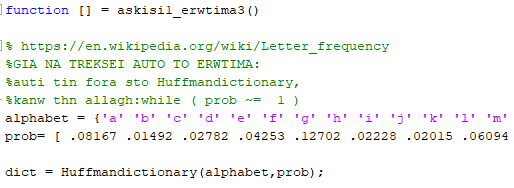
Μήκος της κωδικοποιημένης πηγής σε bits.



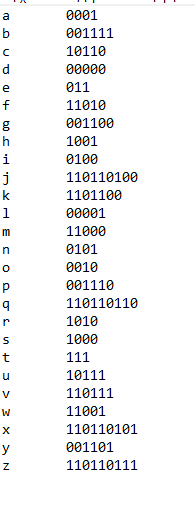
Συμπεραίνουμε , λοιπόν, από τα παραπάνω αποτελέσματα ότι η τιμή της εντροπίας της πηγής είναι αρκετά κοντά με την τιμή του μέσου μήκους κώδικα. Για να είναι το μήκος κάθε κωδικής λέξης ακέραιος,θα πρέπει σε όλες η κατανομή των πιθανοτήτων να είναι της μορφής P(i) ~ 2^k οπου κ θετικος,όμως στην πράξη δεν ισχύει πάντα αυτή η ισότητα οπότε το μέσο μήκος των κωδικολέξεων όπως και στην περιπτωσή μας είναι λίγο μεγαλύτερο από την εντροπία.

**ΕΡΩΤΗΜΑ 3:**

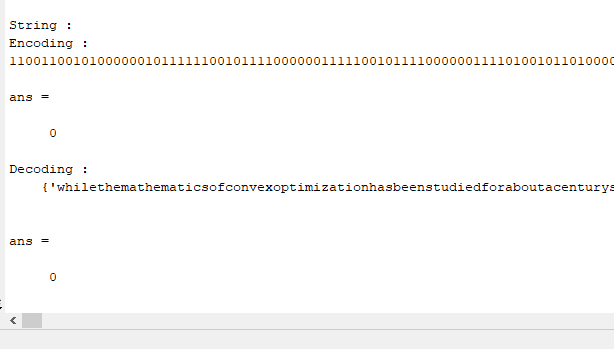
Τρέχω την **askisi1\_erwtima3.m** ,για να τρέξει σωστά χωρίς να πετάξει σφάλμα:Αφού το άθροισμα τον πιθανοτήτων αυτή την φορά είναι ακριβώς 1.



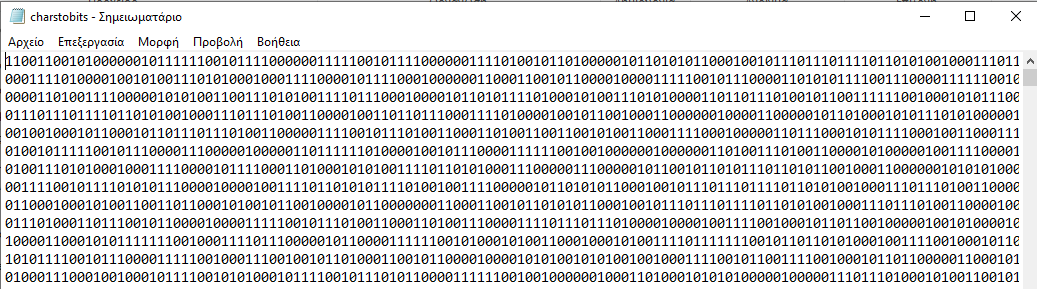
Αποτελέσματα Κωδικοποίησης των αγγλικών χαρακτήρων:

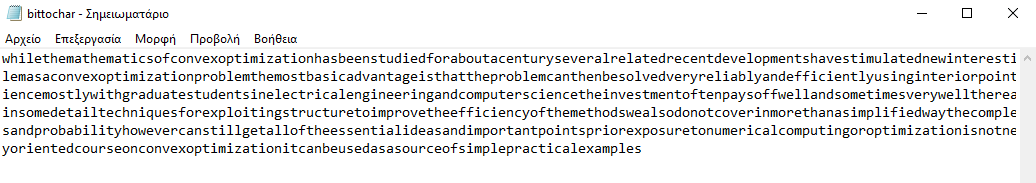


Αποτελέσματα Συμπίεσης/Αποσυμπίεσης της πηγής Α:

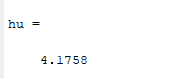


Αποθηκεύονται στα παρακάτω txt και τα αποτελέσματα τους:

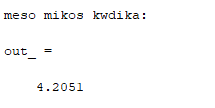




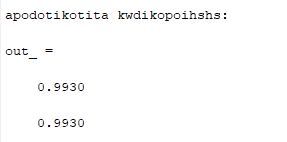
Εντροπία της Πηγής:



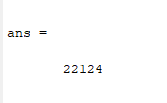
Μέσο μήκος κώδικα:



Απόδοση:



Μήκος της κωδικοποιημένης πηγής σε bits.



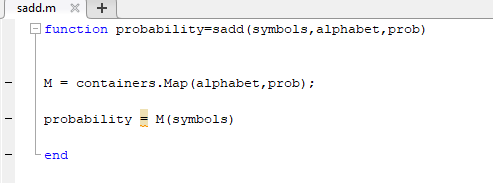
Παρατήρηση: Βλέπουμε ότι αυτή την φορά η κωδικοποίηση στο δυαδικό είναι μικρότερη σε σχέση με το ερώτημα2,αυτό συμβαίνει καθώς στο ερώτημα αυτό έχουμε χρησιμοποιήσει πιο αποδοτικές πιθανότητες των συμβόλων οι οποίες αναπαραστούν κάθε γράμμα σε μικρότερη δυαδική ακολουθία άρα και το μήκος της κωδικοποιημένης πηγής σε bit είναι μικρότερο.

Για αυτό στο ερώτημα2 το μέγεθος της κωδικοποιημένης πηγής είναι 522.142KB

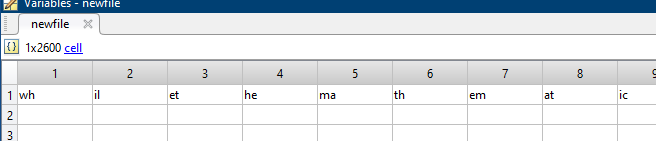


Ενώ στο ερώτημα 3 το μέγεθος μειώνεται σε 478.000KB συνεπώς έχουμε πιο αποδοτική σε χώρο κωδικοποίηση.

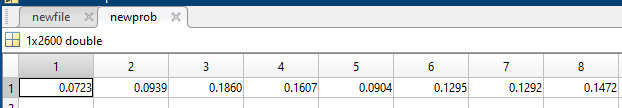
**ΕΡΩΤΗΜΑ 4:**

****

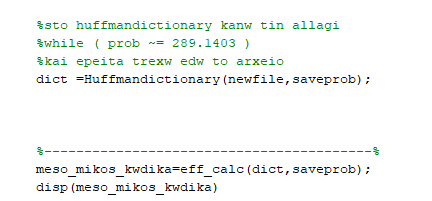
Μέσα στο ερώτημα **erwtima4\_test.m** καλώ την συνάρτηση **sadd.m** μέσω της οποίας υπολογίζω τις πιθανότητες που αναφέρω παρακάτω.Χρησιμοποιώ σε αυτό το ερώτημα την **erwtima4\_test.m** .Αρχικά ανοίγω το cvxopt δημιουργώ ένα άδειο cell (το newfile) όπου θα αποθηκευτούν τα στοιχεία του cvxopt.Μετά την εκτέλεση της for loop σπάω σε δυάδες τα σύμβολα όπως φαίνεται εδώ:

****

Στην συνέχεια,δημιουργώ έναν άδειο πίνακα τον saveprob η for loop τρέχει όσο είναι το μέγεθος της newfile.Το cell2mat παίρνει όλα τα στοιχεία σε κάθε θέση και για κάθε index υπολογίζει την πιθανότητα τους και τα αποθηκεύει στον πίνακα newprob.

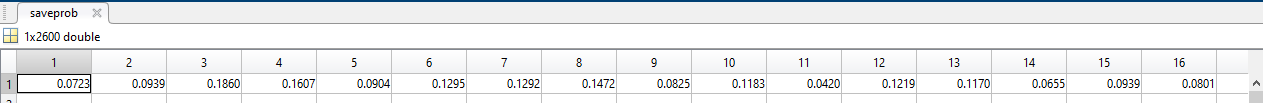
****

Τέλος υπολογίζω την εντροπία,το μέσο μήκος κώδικα και την αποδοτικότητα της πηγής:

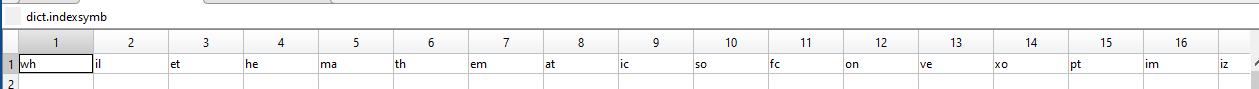
****

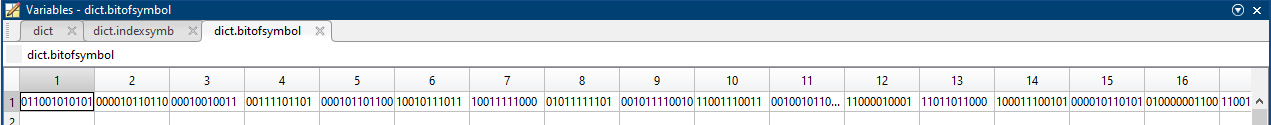
**ΑΠΟΤΕΛΕΣΜΑΤΑ:**

Η ΠΙΘΑΝΟΤΗΤΑ ΤΩΝ ΖΕΥΓΩΝ ΤΩΝ ΣΥΜΒΟΛΩΝ:

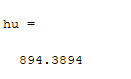


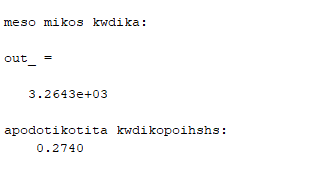
Η ΚΩΔΙΚΟΠΟΙΗΣΗ ΤΩΝ ΣΥΜΒΟΛΩΝ ΤΗΣ ΠΗΓΗΣ ΑΝΑ ΖΕΥΓΑΡΙ:

****

****

ΕΝΤΡΟΠΙΑ:



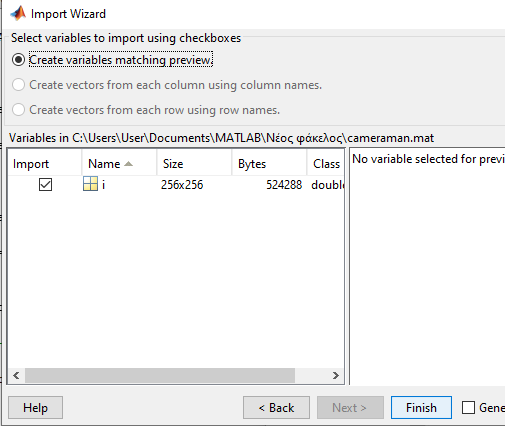


Παρατήρηση:Παρατηρούμε ότι σε αυτή την περίπτωση η αποδοτικότητα της κωδικοποίησης μας είναι σαφώς καλύτερη σε σχέση με το ερώτημα2.Επίσης το μέσο μήκος κώδικα που έχουμε είναι καλύτερο στο ερώτημα 4, καθώς έχουμε κωδικοποίηση ανά ζεύγη συμβόλων.

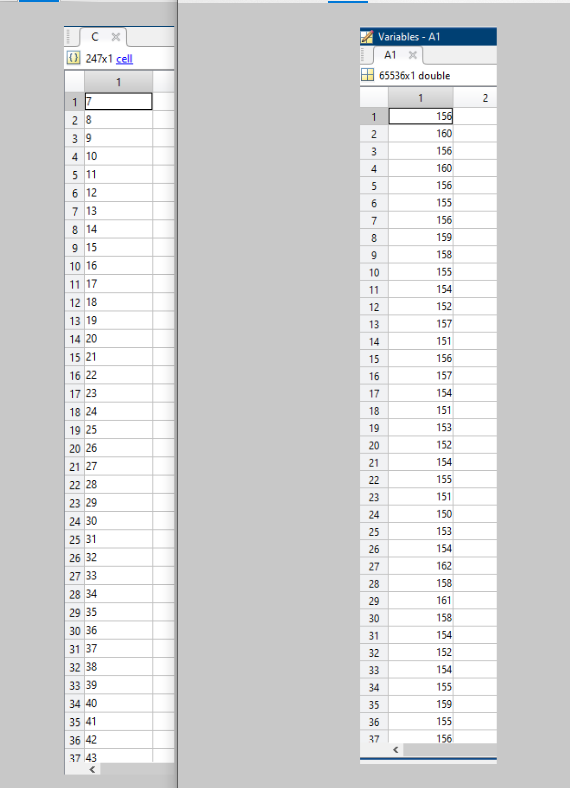
**ΕΡΩΤΗΜΑ 5: ΚΩΔΙΚΟΠΟΙΗΣΗ ΠΗΓΗΣ Β**

Το ερώτημα 5 το έχω υλοποιήση στο **bcserwtima5.m** και στην **introduce\_bsc\_noise.m.**

**ΒΗΜΑ 1:** Κάνω import το cameraman.mat



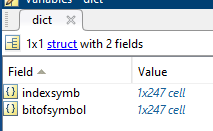
**ΒΗΜΑ 2:** Το Α1 εμφανίζει σε μία στήλη όλους τους αριθμούς που υπάρχουν μέσα στο i=256x256,στόχος είναι να βρόυμε τον αριθμό εμφάνισης του κάθε ‘’αριθμού’’ με σκοπό να βγάλουμε την πιθανότητα τους.Στον πίνακα C υπάρχουν ταξινομημένα οι αριθμοί που εμφανίζονται.

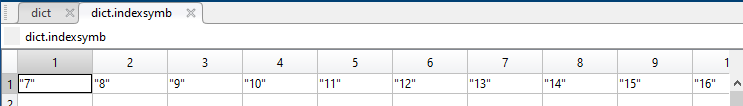


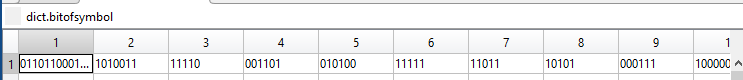
**ΒΗΜΑ 3:** Κάνω transpoC = reshape(C,[1 247]); ώστε να κάνω transpose τον πίνακα c.Εκτελώ την for και μετατρέπω κάθε αριθμό μέσα στον πίνακα C σε μορφή string.Τα αποθηκεύω αυτά σε ένα άδειο cell και έτσι καταλήγω σε cell 1x247.



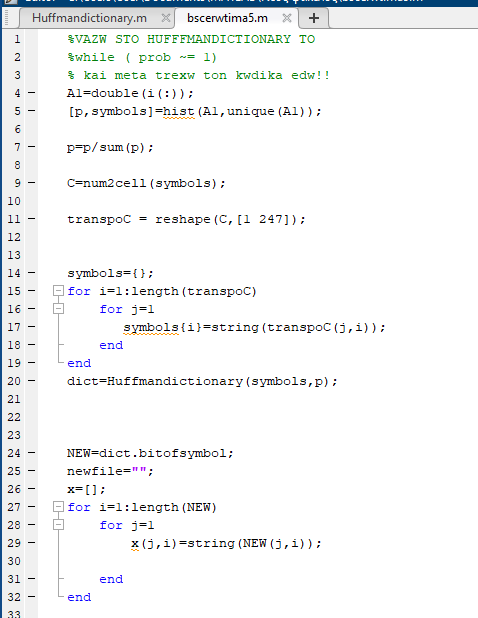
**ΒΗΜΑ 4:** Τρέχω την dict= huffmandictionary(symbols,p) οπου το symbols η πρώτη είσοδος ενώ η δεύτερη είσοδος είναι η πιθανότητα εμφάνισης του κάθε string(“αριθμού”) και παίρνω τα αποτελέσματα:όπου το indexsymb περιέχει την θέση των string(αριθμών) και το bitofsymbol(την δυαδικη κωδικοποιηση τους).

****

****

****

Παραθέτω ένα screenshot του κώδικα:

****

**ΟΙ ΚΩΔΙΚΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΣΑ ΓΙΑ ΤΗΝ ΑΣΚΗΣΗ 1:**

**Huffmandictionary.m**

function dict = Huffmandictionary(alfavito,prob)

for i = 1:length( prob )

bitofsymbol{i} = '';

indexsymb{i} = i;

word(i) = alfavito(i);

end

while ( prob ~= 0.9995 )

[~, z] = sort(prob);

last = z(1);

next = z(2);

deksia\_thesi = indexsymb{last};

aristeri\_thesi = indexsymb{next};

deksia\_pithanotita = prob(last);

aristeri\_pithanotita = prob(next);

merged\_set = [deksia\_thesi, aristeri\_thesi];

new\_prob = deksia\_pithanotita + aristeri\_pithanotita;

% Debug logging.

merged\_word = strcat(word{last},word{next});

word(z(1:2)) = '';

word = [word merged\_word];

% Update probability and symbol sets

indexsymb(z(1:2)) = '';

prob(z(1:2)) = '';

indexsymb = [indexsymb merged\_set];

prob = [prob new\_prob];

% Get the updated codeword.

bitofsymbol = append\_(bitofsymbol,deksia\_thesi,'1');

bitofsymbol = append\_(bitofsymbol,aristeri\_thesi,'0');

end

dict.indexsymb = alfavito;

dict.bitofsymbol = bitofsymbol;

fid = fopen('C:\Users\User\Documents\MATLAB\Νέος φάκελος\NEWdict.txt','wt');

for i = 1:length(alfavito)

fprintf(fid,'%s\t',alfavito{i},bitofsymbol{i});

fprintf(fid,'\n');

end

fclose(fid)

end

**Huffmanencoding.m**

function enco = Huffmanencoding(sig,dict)

enco = '';

while( ~isempty(sig) )

bitvalues = '';

for j = 1 : length(dict.indexsymb)

if( strcmp(sig(1),dict.indexsymb{j}) )

bitvalues = dict.bitofsymbol{j};

break;

end

end

enco = strcat( enco,bitvalues );

sig = sig(2:end);

end

end

**Huffmandecoding.m**

function deco = Huffmandecoding( sig, dict )

deco = []; % output vector

sig\_ = sig;

codepos\_ = 1;

while( ~isempty(sig\_) )

temp\_ = sig\_(codepos\_);

dictb = dict;

while (1)

[flag,dict\_] = found\_match( temp\_, codepos\_, dictb);

if ( flag == 0 )

error('The encoded signal contains a code which is not present in the dictionary.');

end

dictb = dict\_; % update dictionary

if ( length(dictb.bitofsymbol) ~= 1 )

codepos\_ = codepos\_ + 1;

temp\_ = sig\_(codepos\_);

else

codepos\_ = 1;

sig\_ = sig\_(length(dictb.bitofsymbol{1})+1:end);

break;

end

end

%deco = [deco ,dictb.symbol];

deco = strcat( deco,dictb.indexsymb );

% Append char to decoded signal.

end

end

function [flag,dict\_] = found\_match( code, pos, dict )

dict\_.indexsymb={}; dict\_.bitofsymbol={};

j = 1;

flag = 0;

for i = 1:length(dict.bitofsymbol)

if ( strcmp(dict.bitofsymbol{i}(pos), code) )

flag = 1; % No error match.

dict\_.indexsymb(j) = dict.indexsymb(i);

dict\_.bitofsymbol(j) = dict.bitofsymbol(i);

j = j + 1; % Prepare for a next symbol.

end

end

end

**append\_.m**

function bitofsymbol = append\_(bitofsymbol,input\_set,input\_code)

for i = 1:length(input\_set)

bitofsymbol{input\_set(i)} = strcat(input\_code,bitofsymbol{input\_set(i)});

end

end

**eff\_calc.m**

function [out\_] = eff\_calc( dict, prob )

% upologizei to meso mikos kwdika

disp("meso mikos kwdika:")

out\_ = average\_cdl( dict, prob )

% upologismos tis entropias

hu = 0;

for i = 1:length(prob);

hu = hu - prob(i)\*log2(prob(i));

end

disp("apodotikotita kwdikopoihshs:");

out\_ = hu/out\_;

end

% sunartisi gia ton upologismo tou mesou mikous kwdika

% E(L)=Σ P(xi)L(xi) Σ=(απο 1 εως το μηκος του symbon),

%P pithanotita,L

% mikos

function out\_ = average\_cdl( dict, prob )

out\_ = 0;

for i = 1:length(dict.indexsymb);%to megethos twn sumvolwn

out\_ = out\_ + prob(i)\*length(dict.bitofsymbol{i});

end

end

**testerwtima2.m**

function [] = testerwtima2()

alphabet = {"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","space" } ;

prob = [0.0698,0.0128,0.023,0.0364,0.1086,0.0190,0.0172,0.0521,0.0595,0.0013,0.0066,0.0344,0.0206,0.0577,0.0642,0.0165,0.0008,0.0512,0.0541,0.0774,0.0236,0.0084,0.0202,0.0013,0.0169,0.0006,0.1453];

dict = Huffmandictionary(alphabet,prob);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp("String :");

fileI=fopen('cvxopt.txt','r');

x=fscanf(fileI,"%s");

counter=1;

folder={};

enco="";

for i=1

for j=1:length(x)

folder{i,j} = x(i,j) ;

counter = counter + 1 ;

end

enco=strcat(enco,folder);

end

encoded = Huffmanencoding(folder,dict);

disp("Encoding :")

disp(encoded);

fid = fopen('C:\Users\User\Documents\MATLAB\Νέος φάκελος\charstobits.txt','wt');

for i = 1:length(encoded)

fprintf(fid,'%s',encoded);

end

fclose(fid)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

decoded = Huffmandecoding(encoded,dict);

disp("Decoding :")

disp((decoded));

fid = fopen('C:\Users\User\Documents\MATLAB\Νέος φάκελος\bittochar.txt','wt');

for i = 1:length(decoded)

fprintf(fid,'%s\t',decoded{i});

end

fclose(fid)

length(encoded)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

meso\_mikos\_kwdika=eff\_calc(dict,prob);

disp(meso\_mikos\_kwdika)

end

**askisi1\_erwtima3.m**

function [] = askisi1\_erwtima3()

% https://en.wikipedia.org/wiki/Letter\_frequency

%GIA NA TREKSEI AUTO TO ERWTIMA:

%auti tin fora sto Huffmandictionary,

%kanw thn allagh:while ( prob ~= 1 )

alphabet = {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z'};

prob= [ .08167 .01492 .02782 .04253 .12702 .02228 .02015 .06094 .06966 .00153 .00772 .04025 .02406 .06749 .07507 .01929 .00095 .05987 .06327 .09056 .02758 .00978 .02361 .00150 .01974 .00074 ];

dict = Huffmandictionary(alphabet,prob);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp("String :");

fileI=fopen('cvxopt.txt','r');

x=fscanf(fileI,"%s");

counter=1;

folder={};

for i=1

for j=1:length(x)

folder{i,j} = x(i,j) ;

counter = counter + 1 ;

end

end

encoded = Huffmanencoding(folder,dict);

disp("Encoding :")

disp(encoded);

fid = fopen('C:\Users\User\Documents\MATLAB\Νέος φάκελος\charstobits.txt','wt');

for i = 1:length(encoded)

fprintf(fid,'%s',encoded);

end

fclose(fid)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

decoded = Huffmandecoding(encoded,dict);

disp("Decoding :")

disp((decoded));

fid = fopen('C:\Users\User\Documents\MATLAB\Νέος φάκελος\bittochar.txt','wt');

for i = 1:length(decoded)

fprintf(fid,'%s\t',decoded{i});

end

fclose(fid)

length(encoded)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

meso\_mikos\_kwdika=eff\_calc(dict,prob);

disp(meso\_mikos\_kwdika)

end

**erwtima4\_test.m**

alphabet = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','space' } ;

prob = [0.0698,0.0128,0.023,0.0364,0.1086,0.0190,0.0172,0.0521,0.0595,0.0013,0.0066,0.0344,0.0206,0.0577,0.0642,0.0165,0.0008,0.0512,0.0541,0.0774,0.0236,0.0084,0.0202,0.0013,0.0169,0.0006,0.1453];

fid=fopen("cvxopt.txt",'r');

txt=fscanf(fid,"%s");

n=1;

enco="";

newfile={};

for j=1

for i=1:2:size(txt,2)

newfile{n} =txt(i:min(i+1,size(txt,2)));

n=n+1;

end

end

%A=cell2mat(newfile(i)); //vriskei kathe

y=0;

saveprob = [];

for j=1:length(newfile)

A=cell2mat(newfile(j));

for i=1:length(A)

y=y+ sadd(A(i),alphabet,prob);

end

newprob(j)=y;

y=0;

end

saveprob = [saveprob newprob ] ;

%sto huffmandictionary kanw tin allagi

%while ( prob ~= 289.1403 )

%kai epeita trexw edw to arxeio

dict =Huffmandictionary(newfile,saveprob);

meso\_mikos\_kwdika=eff\_calc(dict,saveprob);

disp(meso\_mikos\_kwdika)

**bscerwtima5.m**

%VAZW STO HUFFFMANDICTIONARY TO

%while ( prob ~= 1)

% kai meta trexw ton kwdika edw!!

A1=double(i(:));

[p,symbols]=hist(A1,unique(A1));

p=p/sum(p);

C=num2cell(symbols);

transpoC = reshape(C,[1 247]);

symbols={};

for i=1:length(transpoC)

for j=1

symbols{i}=string(transpoC(j,i));

end

end

dict=Huffmandictionary(symbols,p);

NEW=dict.bitofsymbol;

newfile="";

x=[];

for i=1:length(NEW)

for j=1

x(j,i)=string(NEW(j,i));

end

end

%ndata=introduce\_bsc\_noise(data,newp)

%ndata = bsc(x,p)

%for snr=1:70 % in dB

%BW=7000 % ch BW in Hz

%c(snr)=(BW\*log10(1+10^(snr/10)))/log10(2)

%end

%hold on

%plot(c,'r')

**Sadd.m**

function probability=sadd(symbols,alphabet,prob)

M = containers.Map(alphabet,prob);

probability = M(symbols)

end

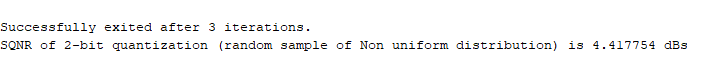
**ΑΣΚΗΣΗ 2: ΚΩΔΙΚΟΠΟΙΗΣΗ PCM**

Η άσκηση ζητάει την υλοποίηση ενός non-uniform βαθμωτού κβαντιστή καθώς και ένα διανυσματικό κβαντιστή.Η υλοποίηση του κβαντιστή(βαθμωτού) βρίσκεται στο αρχείο PCM.m και η υλοποίηση του κβαντιστή(διανυσματικού) βρίσκεται στο αρχείο vecQuant.m

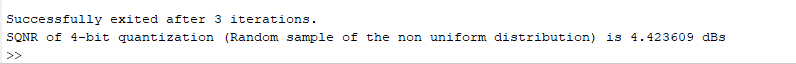
Ερώτημα 1.

a)Για το ερώτημα αυτό εκτελώ το αρχείο **calc\_sqnrA\_nonuniformQuantization.m** καθως και **calc\_sqnrA\_vectorquantization.m** τα οποία καλούν τις αντίστοιχες συναρτήσεις του βαθμωτού και διανυσματικού κβαντιστή.

SQNR-ΒΑΘΜΩΤΟΣ:







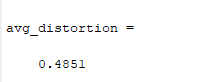
SQNR-ΔΙΑΝΥΣΜΑΤΙΚΟΥ:

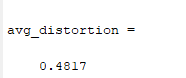


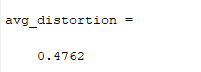




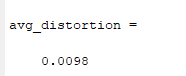
α) ΜΕΣΗ ΠΑΡΑΜΟΡΦΩΣΗ ΒΑΘΜΩΤΟΥ:

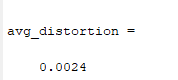


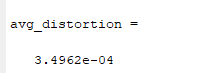


****

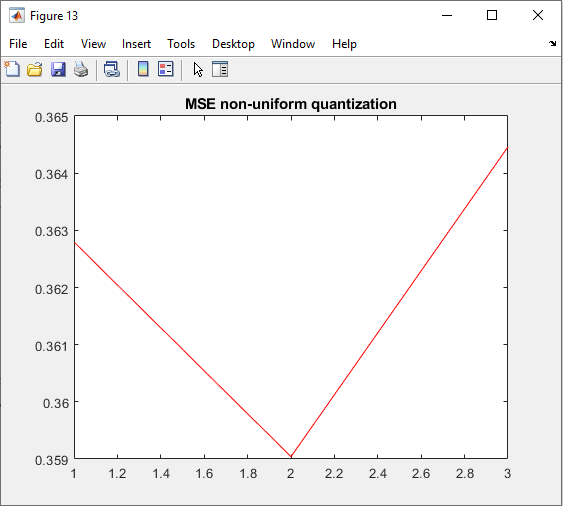
β) ΜΕΣΗ ΠΑΡΑΜΟΡΦΩΣΗ ΔΙΑΝΥΣΜΑΤΙΚΟΥ:

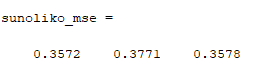




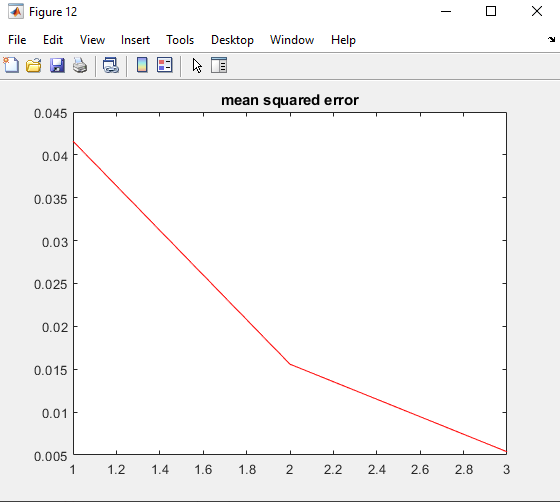


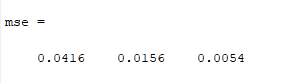
b) ΑΠΟΔΟΤΙΚΟΤΗΤΑ MSE ΒΑΘΜΩΤΟΥ:





ΑΠΟΔΟΤΙΚΟΤΗΤΑ MSE ΔΙΑΝΥΣΜΑΤΙΚΟΥ:



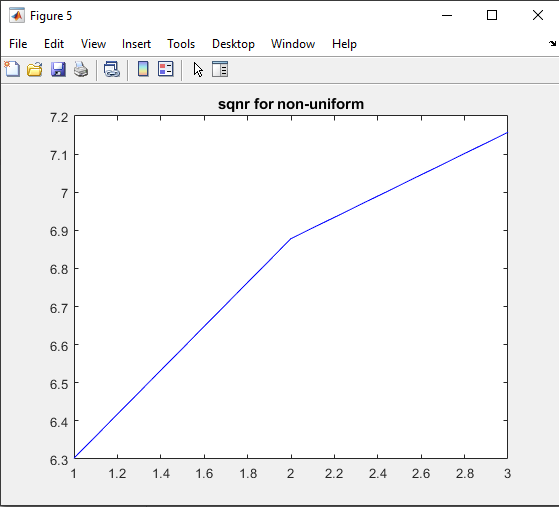


ΕΡΩΤΗΜΑ 2)

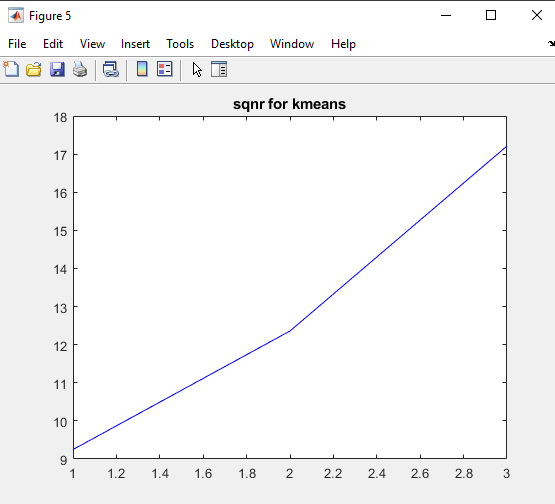
Τρέχω το αρχείο **calc\_sqnrB\_KmeansQuantization.m** και **calc\_sqnrB\_nonuniformQuantization.m**

Η μεταβολή του sqnr γραφικά είναι στην ουσία η ισχυς του σήματος προς την ισχύ θορύβου.

Βαθμωτού:



Διανυσματικού(kmeans):



1. SQNR ΒΑΘΜΩΤΟΥ:







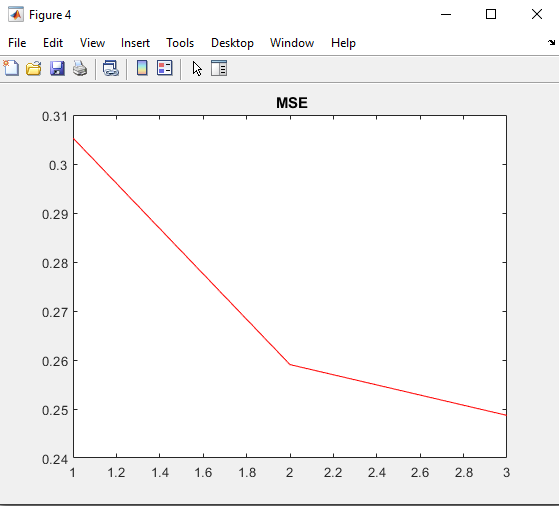
SQNR ΔΙΑΝΥΣΜΑΤΙΚΟΥ:





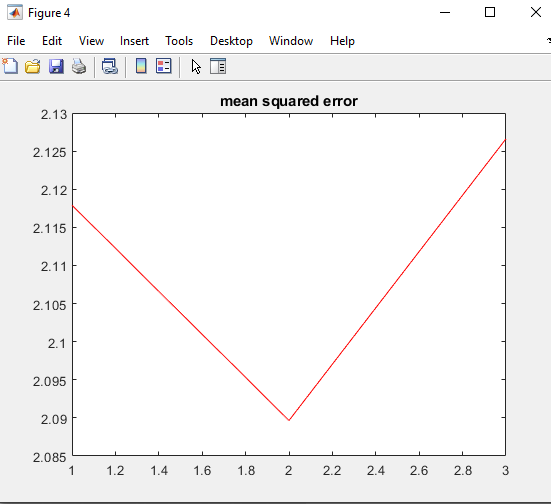


MSE ΒΑΘΜΩΤΟΥ:





MSE ΔΙΑΝΥΣΜΑΤΙΚΟΥ:





**ΑΣΚΗΣΗ 3: M-FSK**