ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ψηφιακές Επικοινωνίες

Εργαστηριακή Άσκηση

Ονοματεπώνυμο: Δημήτριος Λεονταρίδης

Αριθμός Μητρώου: 711 171029

1) Να φτιαχτεί συνάρτηση που να υπολογίζει το φάσμα πλάτους ημιτονοειδούς σήματος με χρήση της συνάρτησης fft(). Ορίσματα της συνάρτησης να θεωρηθούν το σήμα και η περίοδός του.

Κώδικας:

ask1.m

```
f0=30;
T0=1/f0;
Tw = 3*T0;
dt = T0/100;
t=dt:dt:Tw;

y=sin(2*pi*f0*t);
[Yf, f] = AmplitudeSpectrum(y, T0);

plot(f, abs(Yf))
title('Spectrum Amplitude of sin')
xlabel('frequency Hz')
ylabel('Amplitude Volt')
```

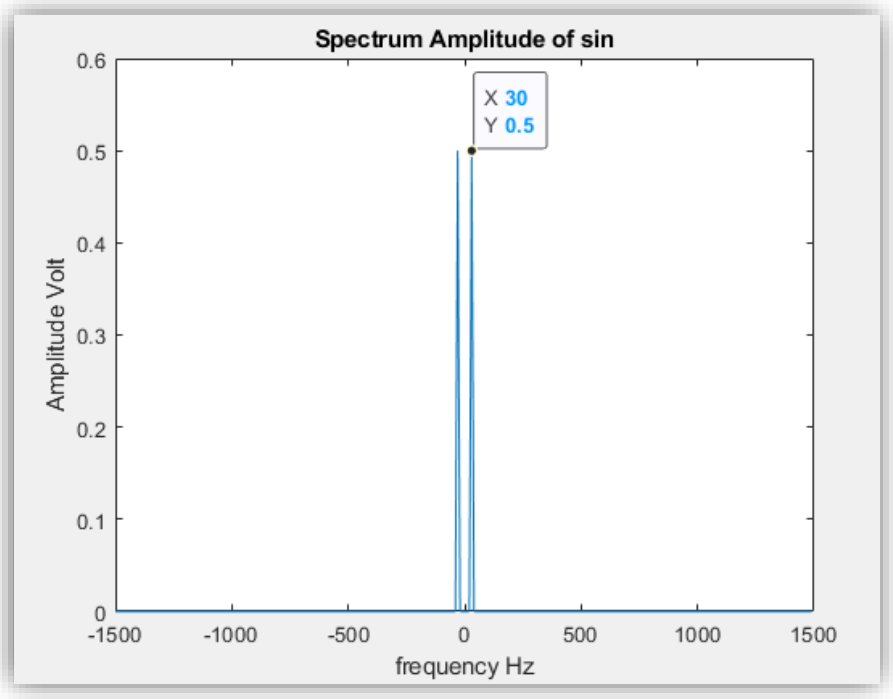
Αρχικά αποθηκεύουμε στην μεταβλητή `f0` την τιμή της συχνότητας (30 Hz) του ημιτονοειδούς σήματος.
Ύστερα υπολογίζουμε την περίοδο `T0` και στην συνέχεια ορίζουμε το συνολικό διάστημα παρατήρησης `Tw` να είναι ίσο με 3 περιόδους του σήματος.
Η διακριτική ικανότητα `dt` είναι το βήμα, κατά το οποίο υπολογίζονται οι τιμές του σήματος `y` στο διάστημα του χρόνου. Ισούται με το ένα εκατοστό στης περιόδου του σήματος.
Στο διάστημα `y` αποθηκεύονται οι τιμές του ημιτονοειδούς σήματος για το χρονικό διάστημα `t`.

AmplitudeSpectrum.m

```
function [Yf,f] = AmplitudeSpectrum(x,T0)

Tw=3*T0;
dt = T0/100;
N=Tw/dt;

Yf = fft(x);
Yf=fftshift(Yf)/N;

BW=1/dt;
df=BW/N;
f=-BW/2:df:BW/2-df;
End
```

Η συνάρτηση `AmplitudeSpectrum()` υπολογίζει το φάσμα πλάτους του ημιτονοειδούς σήματος `x`.
Δέχεται ως παραμέτρους τις τιμές του ημιτονοειδούς σήματος `x` και την περίοδό του `T0`.
Υπολογίζει τον μετασχηματισμό fourier του σήματος `x` και επιστρέφει τους συντελεστές $A_κ$ των συνιστωσών της σειράς Fourier αποθηκευμένους στο διάστημα `Yf`. Επίσης επιστέφει το διάστημα των συχνοτήτων `f`.

<u>Αποτελέσματα:</u>



Στο παραπάνω plot φαίνονται οι τρεις συντελεστές της σειράς fourier $A_0 = 0$, $A_1 = 0.5$, $A_{-1} = -0.5$ στις αντίστοιχες συχνότητες $f_0 = 0$, $f_1 = 30$, $f_{-1} = -30$.

2) Για ένα σήμα ομιλίας αποθηκευμένο σε αρχείο wav να γίνουν τα ακόλουθα α) να διαβαστεί το σήμα (από το πρόγραμμα που χρησιμοποιείτε, π.χ. Octave) και να αποθηκευτεί σε ένα διάνυσμα. Να γίνει γραφική παράσταση της κυματομορφής του σήματος συναρτήσει του χρόνου, β) να ακούσετε το σήμα γ) να αναπαρασταθεί το φάσμα πλάτους.

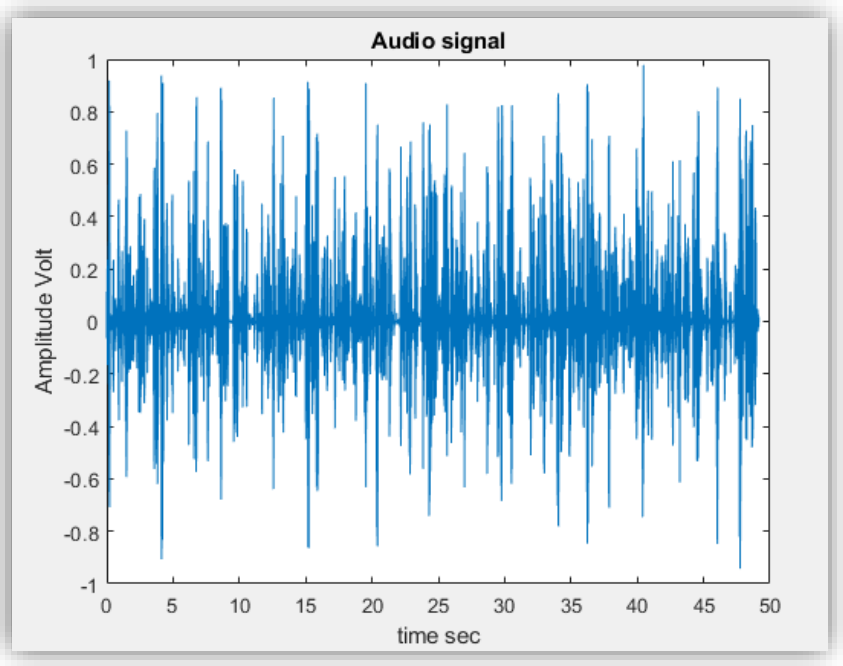
Κώδικας:

ask2.m

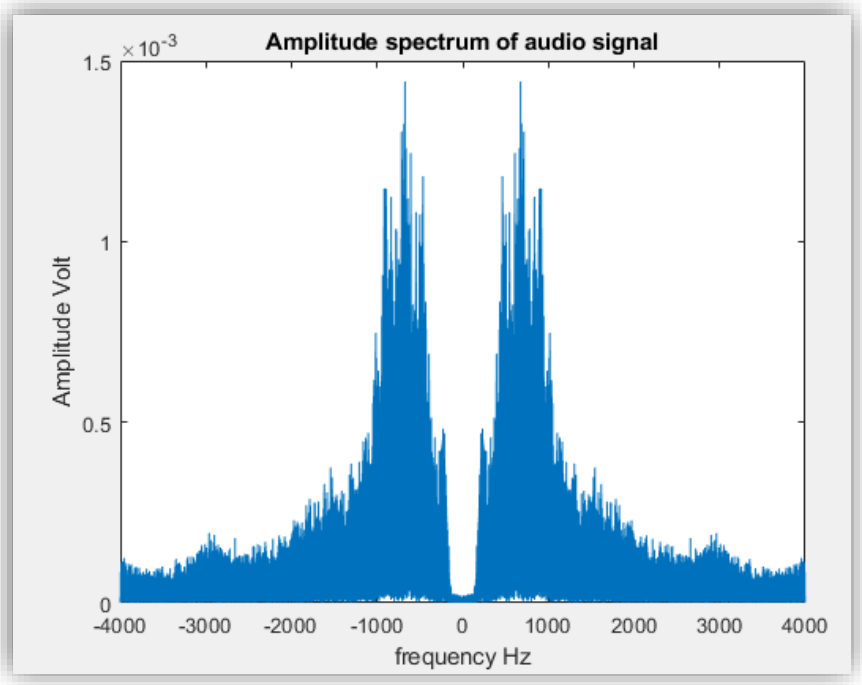
```
[y, Fs] = audioread('female.wav');

N = length(y);
dt = 1/Fs;
Tw = N*dt;
t = dt:dt:Tw;
plot(t, y)
title('female.wav')
xlabel('time sec')
ylabel('Amplitude Volt')
sound(y, Fs);

BW=1/dt;
df=BW/N;
f=-BW/2:df:BW/2-df;
Yf=fftshift(fft(y))/N;
figure
plot(f, abs(Yf))
title('Spectrum Amplitude of female.wav')
xlabel('frequency Hz')
ylabel('Amplitude Volt')
```

Αρχικά διαβάζουμε το .wav αρχείο με την συνάρτηση `audioread()`. Η συνάρτηση επιστρέφει τα δείγματα του σήματος ήχου τα οποία αποθηκεύονται στο διάστημα `y`.
Επίσης επιστρέφει την συχνότητα δειγματοληψίας του σήματος `Fs`.
Στην συνέχεια υπολογίζουμε την διακριτική ικανότητα η οποία είναι αντιστρόφως ανάλογη της `Fs` και το συνολικό διάστημα παρατήρησης `Tw`, ως το γινόμενο του αριθμού των δειγμάτων `N` με την `dt`.
Ύστερα ορίζουμε το διάστημα του χρόνου `t` και κάνουμε plot τις τιμές του σήματος `y` στο πεδίο του χρόνου.
Εφαρμόζουμε μετασχηματισμό fourier στο σήμα `y`, υπολογίζουμε το διάστημα των συχνοτήτων `f` και κάνουμε plot του συντελεστές της σειράς fourier στο πεδίο των συχνοτήτων.

Αποτελέσματα:



Στο παραπάνω Plot φαίνονται οι τιμές του σήματος ήχου y, για τις διάφορες τιμές του διαστήματος χρόνου t.



Στο παραπάνω Plot φαίνονται οι τιμές των συντελεστών της σειράς fourier του σήματος ήχου, για τις διάφορες τιμές του διαστήματος των συχνοτήτων f.

3) Να δημιουργηθεί σήμα πληροφορίας που να αποτελείται από την υπέρθεση συνημιτονοειδών σημάτων από 800Hz με βήμα 50Hz εως 1000Hz.
Να δημιουργηθεί ο άξονας του χρόνου ορίζοντας τη στοιχειώδη χρονική μεταβολή, dt, βάση της πιο γρήγορης συνιστώσας και η διάρκεια παρατήρησης βάση της πιο αργής συνιστώσας.
Να δημιουργηθεί η γραφική παράσταση του σήματος.
Μπορείτε επίσης να ακούσετε το σήμα με τη sound(x). Να προστεθεί στο σήμα λευκός προσθετικός Gaussian θόρυβος 15 dB με χρήση της awgn(). Να δημιουργηθεί επίσης και 2ο ενθόρυβο σήμα με προσθήκη θορύβου 5dB. Να γίνουν σε νέα γραφικά παράθυρα οι αντίστοιχες γραφικές παραστάσεις. (Ακούστε επίσης τα δύο ενθόρυβα σήματα).
Να γίνει αναπαρασταση του φάσματος πλάτους των σημάτων

Κώδικας:

ask3.m

```
f1 = 800;
T1 = 1/f1;
f2 = 1000;
T2 = 1/f2;

dt = T1/100;
Tw = 50*T2;
t=dt:dt:Tw;

S = 0;
Fstart = 800;
Fend = 1000;
step = 50;

for fv = Fstart:step:Fend
    S = S + cos(2*pi*fv*t);
end

% S = Superposition of cosines
plot(t, S)
title('Superposition of cosine signals')
xlabel('time sec')
ylabel('Amplitude Volt')
%sound(S);

% SG1 = Superposition of cosines with adwn, SNR = 15
SGN1 = awgn(S, 15);
figure;
```

```
plot(t, SGN1)
title('SNR = 15db');
xlabel('time sec')
ylabel('Amplitude Volt')
%sound(SGN1);

% SG2 = Superposition of cosines with adwn, SNR = 5
SGN2 = awgn(S, 5);
figure;
plot(t, SGN2)
title('SNR = 5db');
xlabel('time sec')
ylabel('Amplitude Volt')
%sound(SGN2);

BW=1/dt;
N=Tw/dt;
df=BW/N;
f=-BW/2:df:BW/2-df;

% Fourier transform of SG1
Yf1=fftshift(fft(SGN1))/N;
figure
plot(f, abs(Yf1))
title('Amplitude Spectrum, SNR = 15db');
xlabel('frequency Hz')
ylabel('PSD')

% Fourier transform of SG2
Yf2=fftshift(fft(SGN2))/N;
figure
plot(f, abs(Yf2))
title('Amplitude Spectrum, SNR = 5db');
xlabel('frequency Hz')
ylabel('PSD')
```

Αρχικά για τον υπολογισμού του τελικού σήματος πληροφορίας θα προσθέτουμε επαναληπτικά ένα συνημίτονο στο σήμα πληροφορίας, για τις διάφορες τιμές της συχνότητας `fv = 800:50:1000`.

```
for fv = Fstart:step:Fend
    S = S + cos(2*pi*fv*t);
end
```

Το πεδίο του χρόνου `t`, στο οποίο ορίζονται οι τιμές του σήματος του τελικού σήματος `S`, θα έχει διάστημα παρατήρησης `Tw = 50*T2`, όπου `T2 = 1/f2 = 1/1000`. Δηλαδή η συνολική παρατήρηση θα διαρκέσει 50 περιόδους `T2`.

Η χρονική μεταβολή υπολογίζεται βάση της πιο αργής συνιστώσας `dt = T1/100`.
Όπου η συνιστώσα `T1 = 1/f1 = 1/800`.

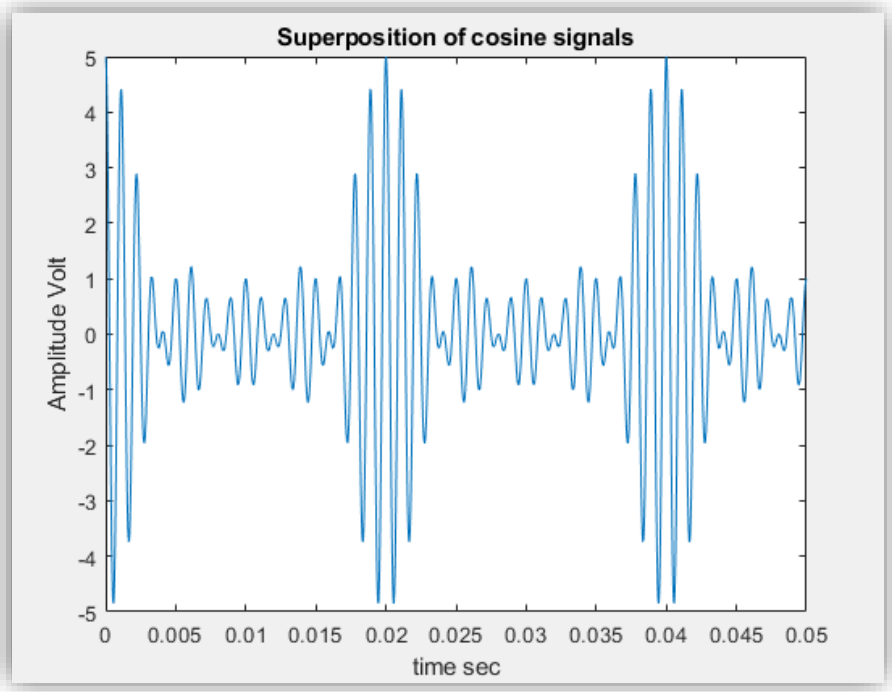
Επομένως το διάστημα του χρόνου για το τελικό σήμα πληροφορίας ορίζεται ως εξής, `t=dt:dt:Tw`.

Ύστερα, δημιουργούμε το σήμα `SG1`, το οποίο προκύπτει από την προσθήκη λευκού Γκαουσιανού θορύβου στο σήμα πληροφορίας `S`, με λόγο σήματος/θορύβου `SNR = 15db`.

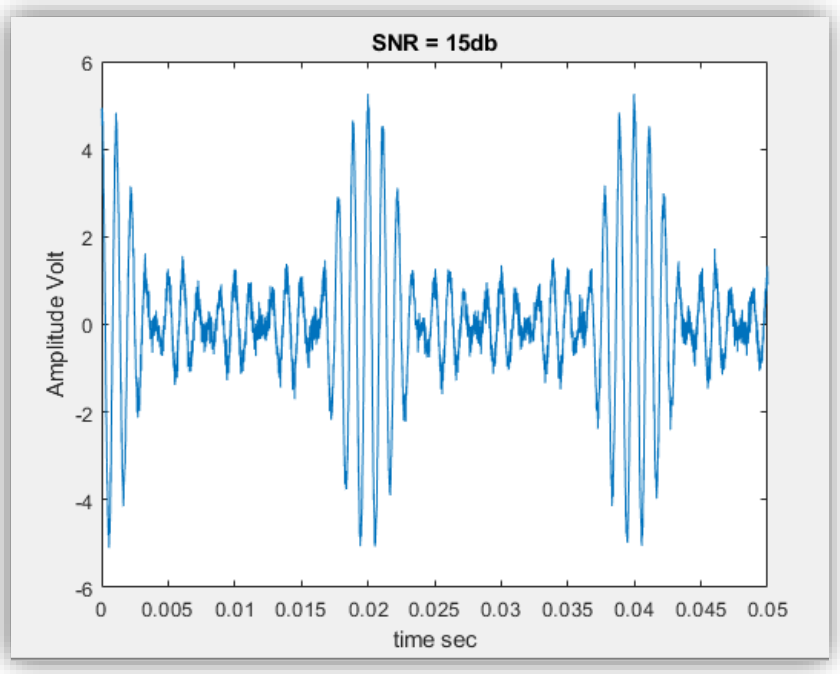
Επίσης, δημιουργούμε και ένα δεύτερο σήμα `SG2`, προσθέτοντας λευκό Γκαουσιανό θορύβο, με λόγο σήματος/θορύβου `SNR = 5db`.

Έπειτα, υπολογίζουμε τους συντελεστές των συνιστωσών της σειράς Fourier για κάθε ένα από τα δύο σήματα `SG1, SG2` καθώς και το διάστημα των συχνοτήτων `f`.

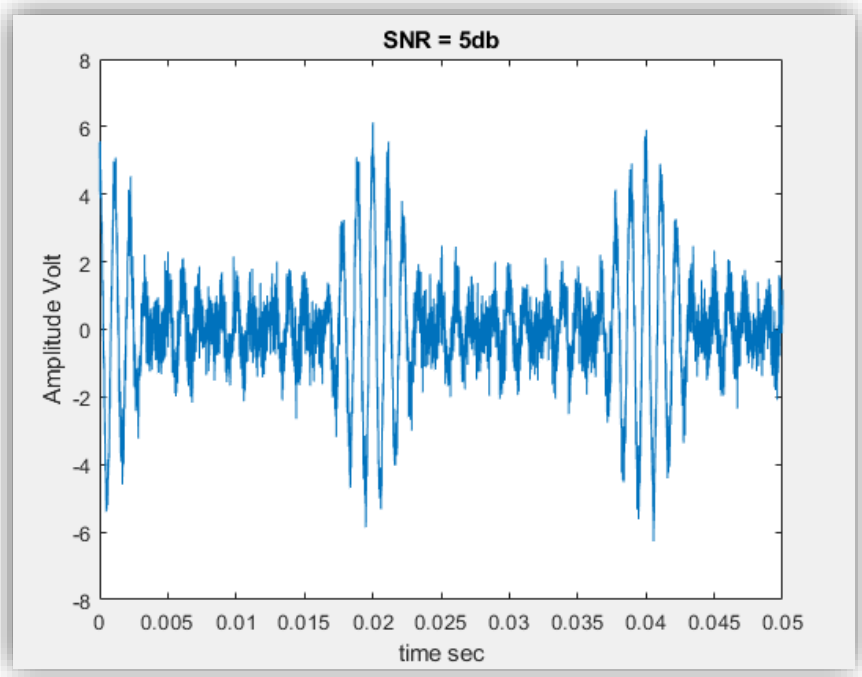
<u>Αποτελέσματα:</u>



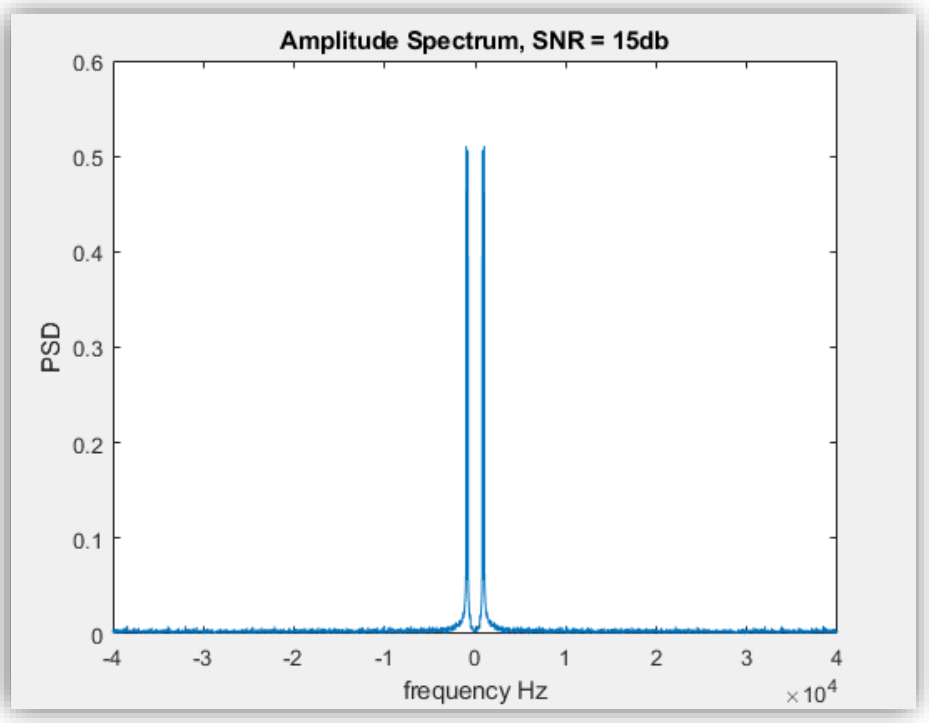
Στο παραπάνω Plot φαίνεται η κυματομορφή του τελικού σήματος πληροφορίας `S`.



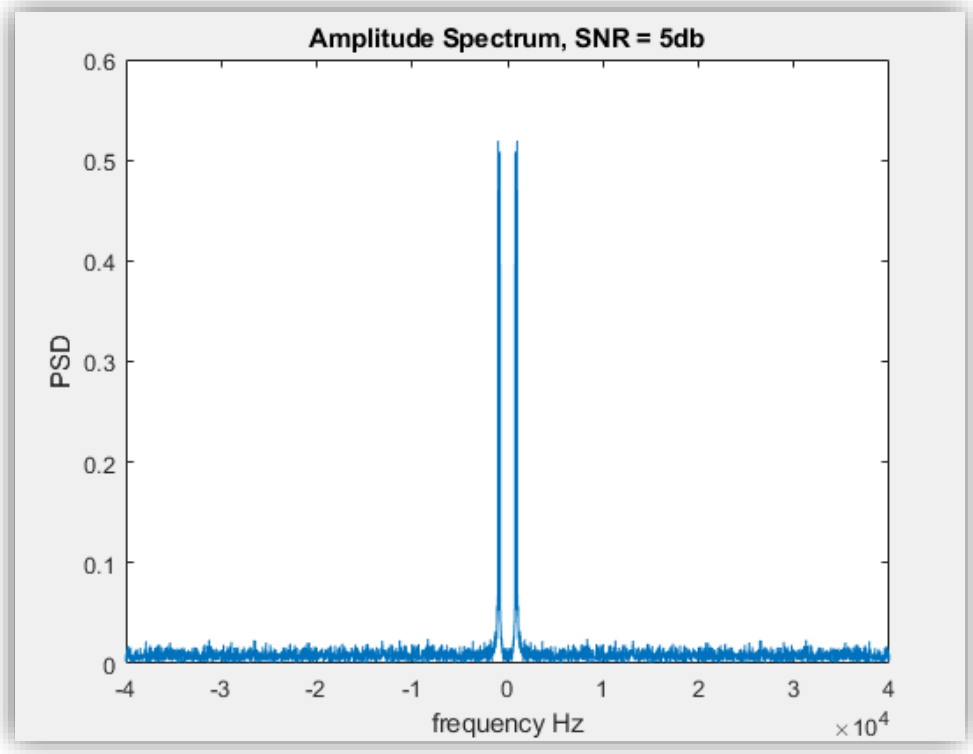
Στο παραπάνω Plot φαίνεται η κυματομορφή του τελικού σήματος πληροφορίας `S` μετά από την προσθήκη λευκού Γκαουσιανού θορύβου, με λόγο `SNR = 15db`.



Στο παραπάνω Plot φαίνεται η κυματομορφή του τελικού σήματος πληροφορίας `S` μετά από την προσθήκη λευκού Γκαουσιανού θορύβου, με λόγο `SNR = 5db`. Η αλλοίωση του αρχικού σήματος είναι περισσότερο εμφανής σε σχέση με την περίπτωση όπου `SNR = 15db`.



Στο παραπάνω Plot φαίνεται το φάσμα πλάτους για `SNR = 15db`.



Στο παραπάνω Plot φαίνεται το φάσμα πλάτους για `SNR = 5db`.

4) Να δημιουργηθεί σήμα πληροφορίας που να αποτελείται από την υπέρθεση συνημιτονοειδών σημάτων από 500Hz με βήμα 100Hz έως 3000Hz.
Φέρον συχνότητας 100KHz. Πραγματοποιήστε διαμόρφωση DSB.
Στη συνέχεια χρησιμοποιήστε τη συνάρτηση που υλοποιεί ένα ιδανικό (τετραγωνικό) απλοποιημένο ζωνοπερατό φίλτρο : function y = bpfilt(signal, f1, f2,fs) που παρατίθεται παρακάτω και τα ορίσματά της είναι signal : το σήμα που θέλουμε να φιλτράρουμε, f1, f2 το κάτω και άνω όριο συχνότητας του ζωνοπερατού φίλτρου και fs η συχνότητα δειγματοληψίας.
Με χρήση της συνάρτησης παράγετε
α) ένα USB – άνω πλευρικής ζώνης και
β) ένα LSB – κάτω πλευρικής ζώνης διαμορφωμένο σήμα.
Απεικονίστε τα φάσματα πλάτους όλων των σημάτων.

Κώδικας:

ask4.m (Η συνάρτηση bpfilt.m βρίσκεται στο τέλος του αρχείου)

```
f1 = 500;
T1 = 1/f1;
f2 = 3000;
T2 = 1/f2;

dt = T2/100;
Tw = T1;
t=dt:dt:Tw;
N=Tw/dt;

S = 0;
Fstart = 500;
Fend = 3000;
step = 100;

% Sum of cosines
for fv = Fstart:step:Fend
    S = S + cos(2*pi*fv*t);
end
% ym = Information signal
ym = S ;

% yc = Carrier signal
fc=100000;
Tc=1/fc;
yc = cos(2*pi*fc*t);
```

```
% y = Modulated signal
y = yc.*(ym + 10);
plot(t, ym)
title('Information signal');
xlabel('Time sec')
ylabel('Ampplitude Volt')

figure
plot(t, yc)
title('Carrier signal');
xlabel('Time sec')
ylabel('Ampplitude Volt')


figure
plot(t, y)
title('Modulated signal');
xlabel('Time sec')
ylabel('Ampplitude Volt')

% Information signal spectrum
W=1/dt;
df=W/N;
f=-W/2:df:W/2-df;
F_ym=fftshift(fft(ym))/N;
figure
plot(f, abs(F_ym))
title('Information signal spectrum');
xlabel('Frequency Hz')
ylabel('PSD')

% Carrier signal spectrum
F_yc=fftshift(fft(yc))/N;
figure
plot(f, abs(F_yc))
title('Carrier signal spectrum');
xlabel('Frequency Hz')
ylabel('PSD')

% DSB modulated signal spectrum
F_y=fftshift(fft(y))/N;
figure
plot(f, abs(F_y))
title('DSB Modulated signal spectrum');
xlabel('Frequency Hz')
ylabel('PSD')
```

```
% USB modulated signal spectrum
yusb = bpfilt(y, Fstart+1500, fc/2+500, W);
F_yusb = fftshift(fft(yusb))/N;
figure
plot(f, abs(F_yusb))
title('USB Modulated signal spectrum')
xlabel('Frequency Hz')
ylabel('PSD')

% LSB modulated signal spectrum
ylsb = bpfilt(y, 0, fc+ 500, W);
F_ylsb = fftshift(fft(ylsb))/N;
figure
plot(f, abs(F_ylsb))
title('LSB Modulated signal spectrum')
xlabel('Frequency Hz')
ylabel('PSD')
```

Αρχικά δημιουργούμε το σήμα πληροφορίας `ym` το οποίο αποτελείται από το άθροισμα συνημίτονων στις διάφορες συχνότητες του διαστήματος `fv = Fstart:step:Fend`

`Fstart = 500 Hz, step = 100 Hz Fend = 3000 Hz.`

Ύστερα δημιουργούμε φέρον σήμα `yc` με συχνότητα `fc = 100kHz.`

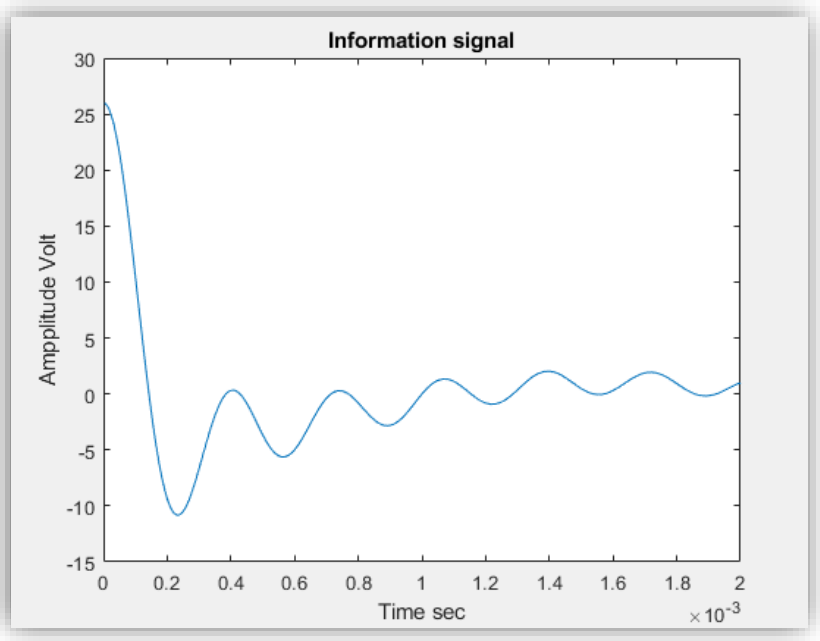
Από το γινόμενο του φέροντος και του σήματος πληροφορίας προκύπτει το διαμορφωμένο τελικό σήμα `y`. Στο αρχικό σήμα πληροφορίας έχει προστεθεί μία σταθερά με τιμή 10, με σκοπό να αποφευχθεί η παραμόρφωση της περιβάλλουσας του διαμορφωμένου `y` από τις αρνητικές τιμές του σήματος πληροφορίας `ym`.

Στην συνέχεια υπολογίζουμε τα αμφίπλευρα φάσματα του σήματος πληροφορίας `ym`, του φέροντος `yc` και του διαμορφωμένου `y`.

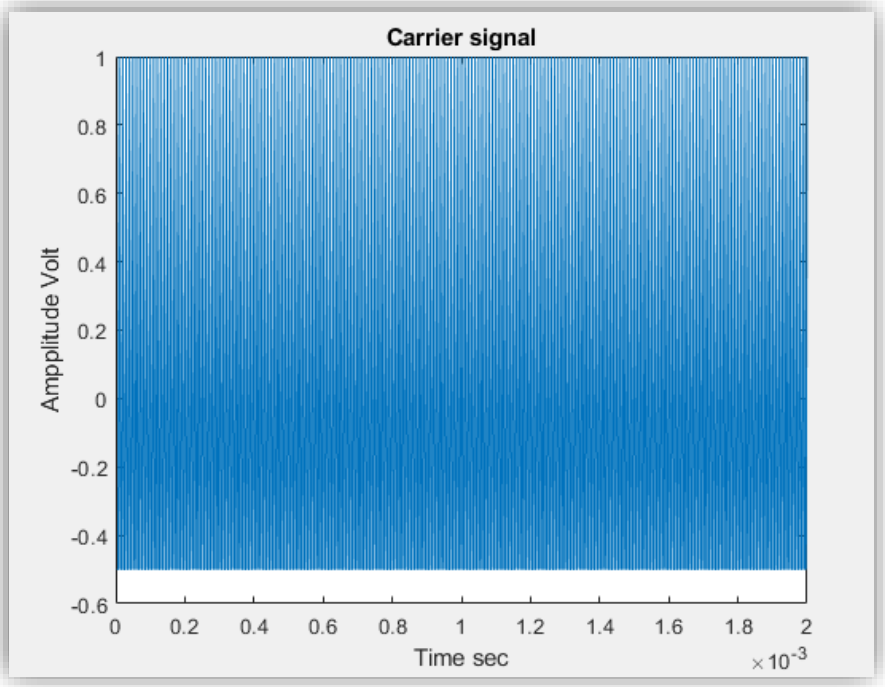
Για την αναπαράσταση του φάσματος της άνω πλευρικής ζώνης (USB), έγινε φιλτράρισμα του διαμορφωμένου σήματος και κρατήθηκαν οι τιμές για από τα `Fstart + 50 = 450 Hz` έως τα `fc/2 + 500 = 50500 Hz.`

Επίσης, για την αναπαράσταση του φάσματος της άνω πλευρικής ζώνης (LSB), έγινε φιλτράρισμα του διαμορφωμένου σήματος και κρατήθηκαν οι τιμές για από τα `0 Hz` έως τα `fc + 500 = 100500 Hz.`

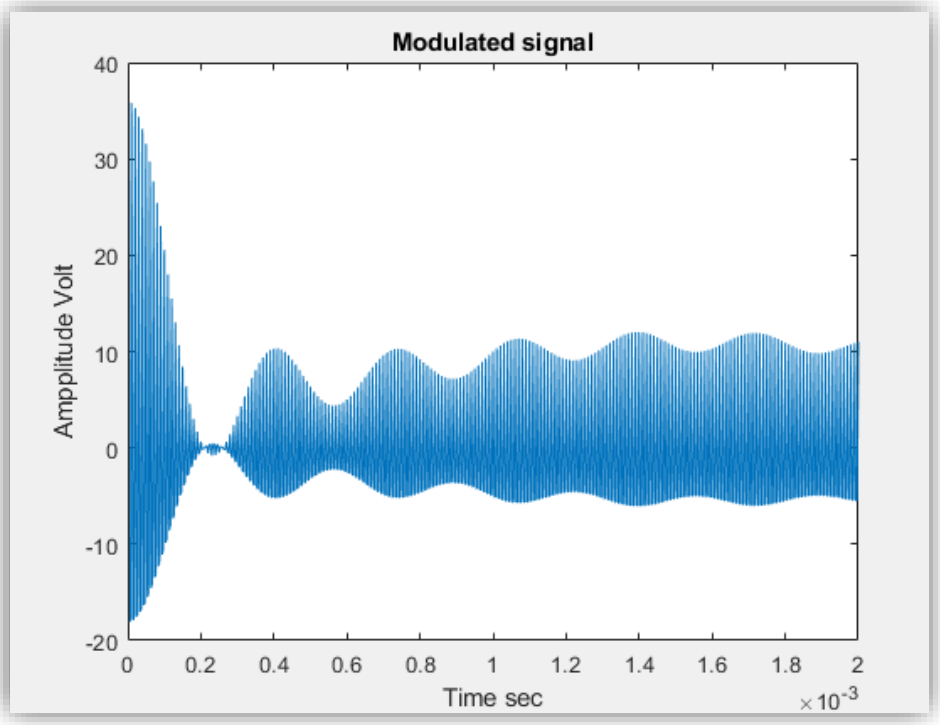
Αποτελέσματα:



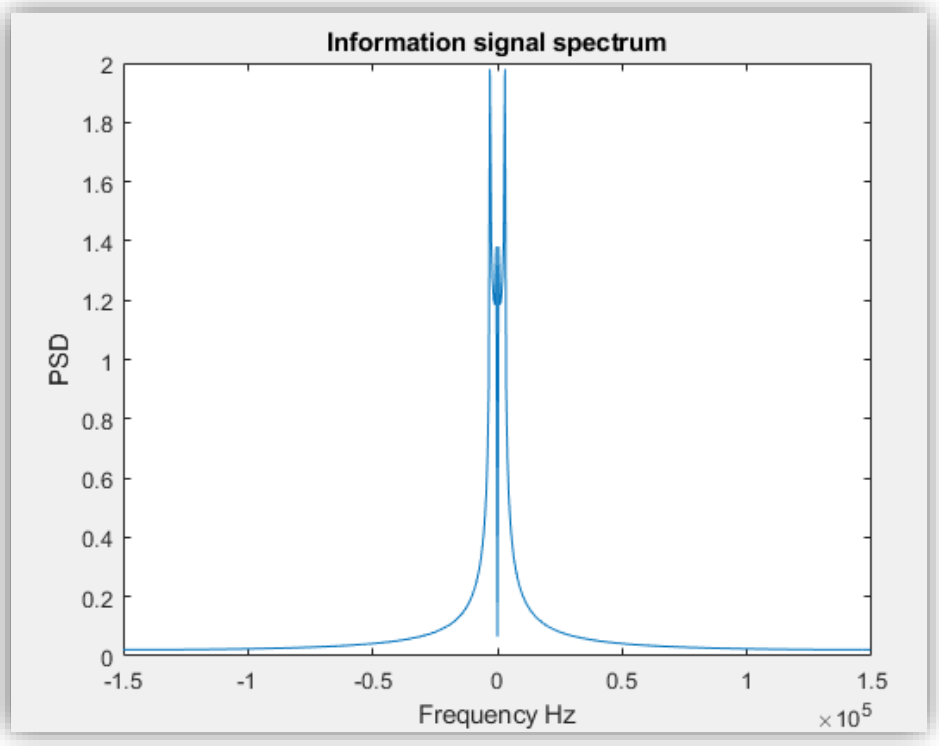
Παραπάνω φαίνεται το plot του σήματος πληροφορίας `ym`. Η συνολική διάρκεια παρατήρησης είναι μία περίοδος `T1`, όπου `T1 = 1/f1, f1 = 500 Hz`. Η συνιστώσα στην συχνότητα `f1` είναι και η πιο αργή.



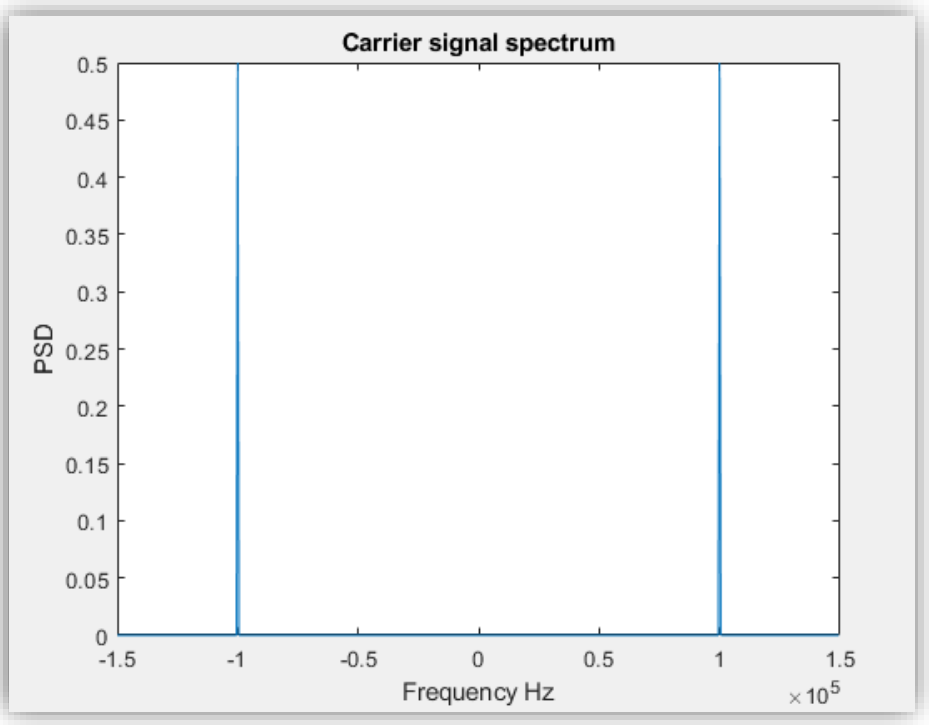
Παραπάνω φαίνεται το plot του φέροντος `yc` με συχνότητα `fc = 100kHz`.



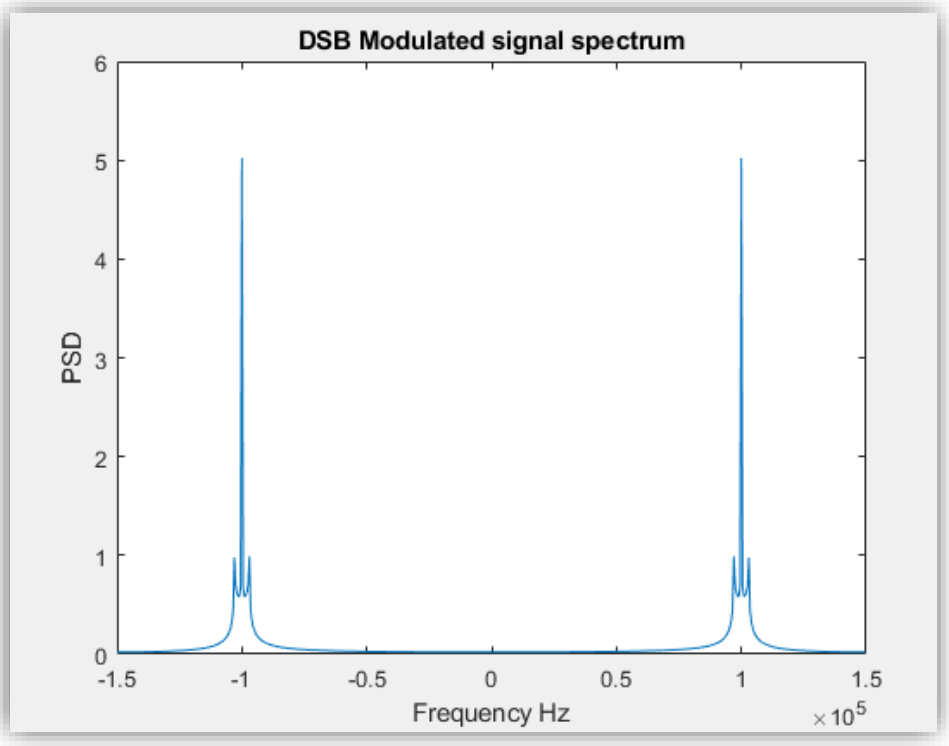
Παραπάνω φαίνεται το plot του διαμορφωμένου σήματος `y`.



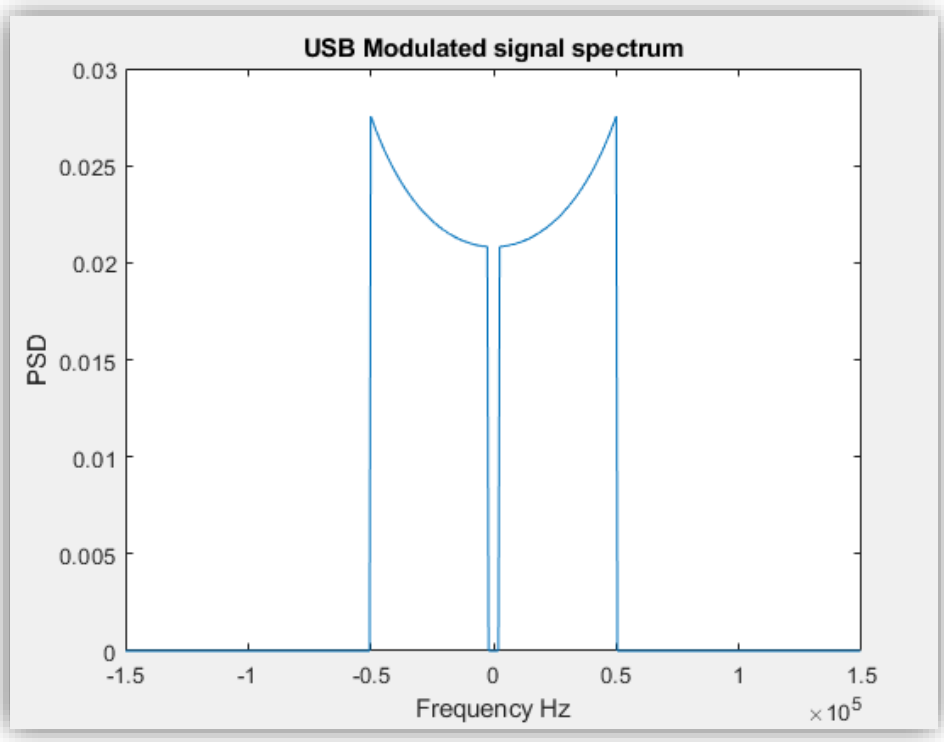
Παραπάνω φαίνεται το φάσμα του σήματος πληροφορίας `ym`.



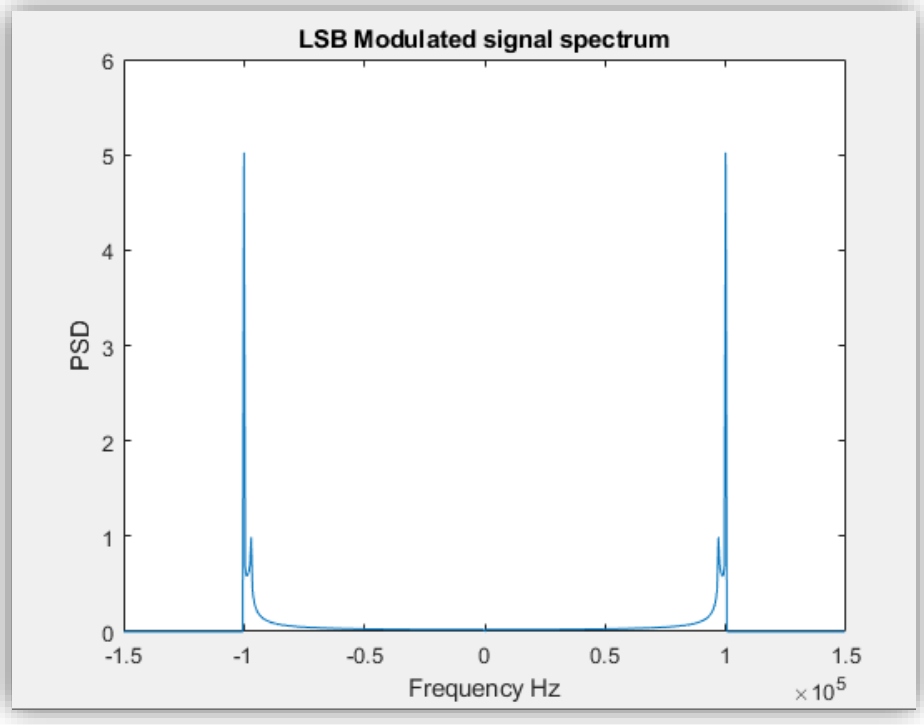
Παραπάνω φαίνεται το φάσμα του φέροντος `yc`.



Παραπάνω φαίνεται το φάσμα διπλής πλευρικής ζώνης του διαμορφωμένου σήματος.



Παραπάνω φαίνεται το φάσμα άνω πλευρικής ζώνης του διαμορφωμένου σήματος.



Παραπάνω φαίνεται το φάσμα άνω πλευρικής ζώνης του διαμορφωμένου σήματος.

5) Αναπαράσταση του φαινομένου aliasing λόγω υποδειγματοληψίας.

α) Δημιουργήστε ημιτονοειδές σήμα 1KHz, διάρκειας 2 sec.

Να δειγματοληπτηθεί το σήμα με 20KHz και έπειτα με 1.5KHz.

Να αναπαρασταθούν και τα δύο σήματα στο ίδιο γράφημα.

β) Ακούστε τα δύο σήματα με τη συνάρτηση soundsc(x,fs). Τι παρατηρείτε?

γ) Δημιουργήστε ένα ημιτονοειδές σήμα που θα παράγει τον ίδιο ήχο με αυτό του δειγματοληπτημένου ημιτόνου με 1.5KHz.

Κώδικας:

ask5.m

```
% y at 1kHz
f=1e+3;
T=1/f;
dt=T/100;
Tw=6*T;
t=dt:dt:Tw;
y=sin(2*pi*f*t);

% y1, fs = 20khZ
fs1=20e+3;
Ts1=1/fs1;
t1=Ts1:Ts1:Tw;
y1=sin(2*pi*f*t1);
plot(t,y,'g')
hold on
stem(t1,y1,'b')
hold on

% y2, fs = 1.5khZ
fs2=1.5e+3;
Ts2=1/fs2;
t2=Ts2:Ts2:Tw;
y2=sin(2*pi*f*t2);

stem(t2,y2,'r')
title('\color{green}Sinusoidial of 1 KHz, \color{black}sampling
at \color{red}1.5 KHz, \color{blue} 20 KHz');
```

```
soundsc(y1, fs1);
soundsc(y2, fs2);

% y3 same output with y2
t2=Ts2:Ts2:Tw;
y3=sin(pi*f*t2+pi);
figure
stem(t2,y2,'r')
hold on
plot(t2,y3,'b')
title('\color{red}sin(2*pi*f*t2) at fs = 1.5 KHz, \color{blue}
sin(pi*f*t2+pi) of 1 KHz \color{black}(they sound the same)')
soundsc(y3, fs2);
```

Αρχικά δημιουργούμε το ημιτονοειδές σήμα πληροφορίας `y` με συχνότητα `f = 1 kHz.`

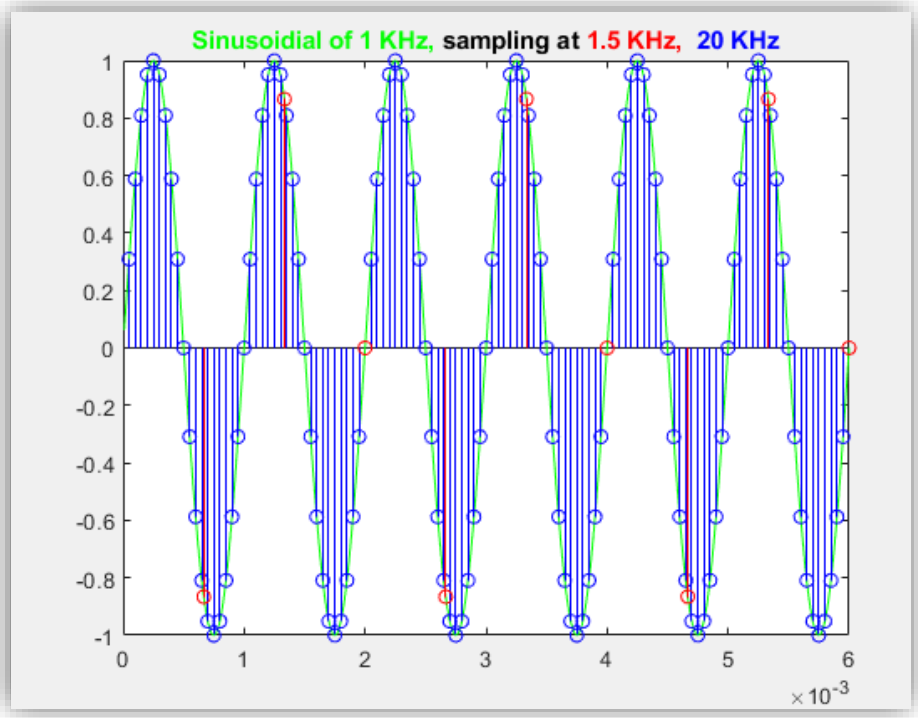
Στην συνέχεια δημιουργούμε το νέο διάστημα χρόνου `t1,` το οποίο υπολογίζεται με βάση την συχνότητα δειγματοληψίας `fs1 = 20 kHz.` Έτσι προκύπτει το νέο σήμα `y1` ως το επαρκώς δειγματοληπτιμένο σήμα `y.`

Επίσης, δημιουργούμε το νέο διάστημα χρόνου `t2,` το οποίο υπολογίζεται με βάση την συχνότητα δειγματοληψίας `fs2 = 1.5 kHz.` Έτσι προκύπτει το νέο σήμα `y2`. Το σήμα `y2` είναι το υποδειγματοληπτιμένο σήμα `y.`

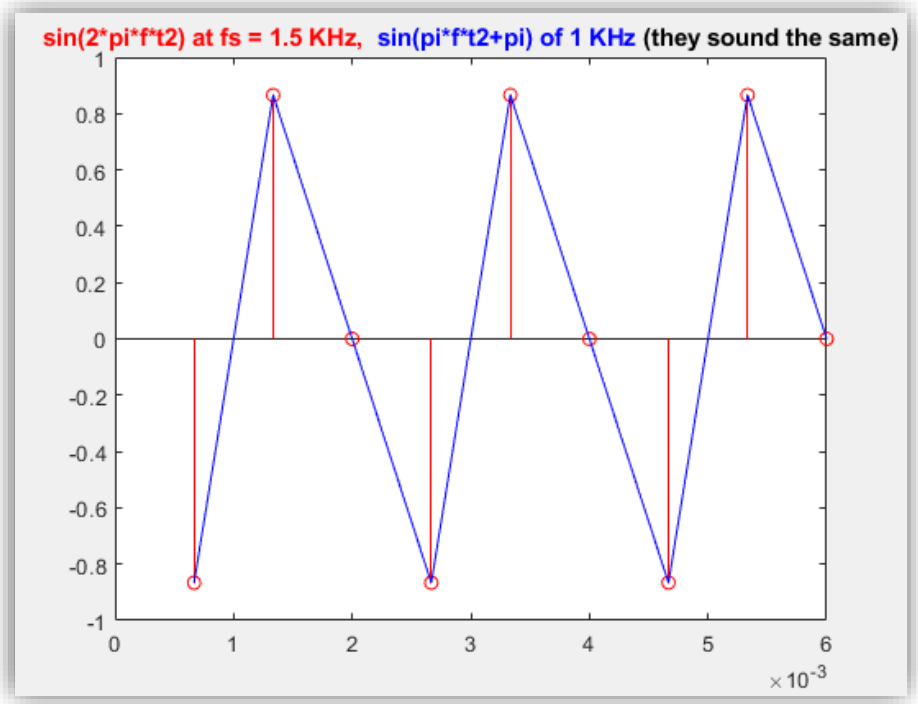
Η διαφορά στην συχνότητα παρατηρείται και στο άκουσμα. Το σήμα y1 βρίσκεται σε υψηλή συχνότητα (20 kHz), συνεπώς βγάζει λεπτό ήχο .Ενώ το σήμα `y2`, βρίσκεται σε πιο χαμηλή συχνότητα (1.5 kHz) και βγάζει βαρύ ήχο.

Το σήμα `y3` είναι ένας περιοδικός τριγωνικός παλμός ο οποίος αντιστοιχεί στις τιμές του ημιτονοειδούς σήματος `y2`.

Αποτελέσματα:



Παραπάνω φαίνονται το σήμα `y` (πράσινο) και το αποτέλεσμα της δειγματοληψίας στην συχνότητα `fs1 = 1kHz` (κόκκινο) και αντίστοιχα στην συχνότητα `fs2 = 20 kHz` (μπλε).



Το υποδειγματοληπτιμένο σήμα `y2` είναι το ίδιο με ένα αντίστοιχο ημιτονοειδές `y3` στην μισή συχνότητα του `y2` και με μία διαφορά φάσης π. Επίσης ακούγονται το ίδιο.

6) Δημιουργήστε ένα AM σήμα x1(t) με τις εξής παραμέτρους: Ac =1 Volt, Am=0.5 Volt,fm=2 Hz, fc=100Hz, όταν το σήμα βασικής ζώνης είναι s(t)= Am sin(2πfmt). Θεωρήστε ότι t=[0:0.001:1].

Αναπαραστήστε γραφικά το διαμορφωμένο σήμα με την plot(t,x1). Αναπαραστήστε γραφικά επίσης και την περιβάλλουσα (envelope) του σήματος.

Διαμορφώστε το ίδιο σήμα s(t) με την συνάρτηση ammod() του communication toolbox του Matlab και ονομάστε το σήμα x2(t).

Ποιο είδος διαμόρφωσης υλοποιεί η ammod?

Σύνταξη ammod: y = ammod(s, fc, fs)

y: διαμορφωμένο σήμα s: σήμα πληροφορίας

fc : συχνότητα φέροντος

fs : συχνότητα δειγματοληψίας- θέσετε fs=400

Κώδικας:

ask6.m

```
Am = 0.5;
fm = 2;
Tm = 1/fm;
Tw = 3*Tm;

Ac = 1;
fc = 100;
Tc = 1/fc;
dt = Tc/100;
t=dt:dt:Tw;

% sm = base band signal
sm = Am*sin(2*pi*fm*t);
% sc = carrier signal
sc = Ac*cos(2*pi*fc*t);

% x1 =  AM signal
x1 = sm.*sc + Ac*sc;
plot(t, x1, 'b')
hold on
plot(t, sm, 'r')
title('\color{blue} Modulated signal, \color{red}Base Band
```

```
signal');

% x2 = ammod signal
fs = 400;
x2 = ammod(sm, fc, fs);
figure
plot(t, x2)
title('ammod, Am modulation (Hypermodulation)');
```

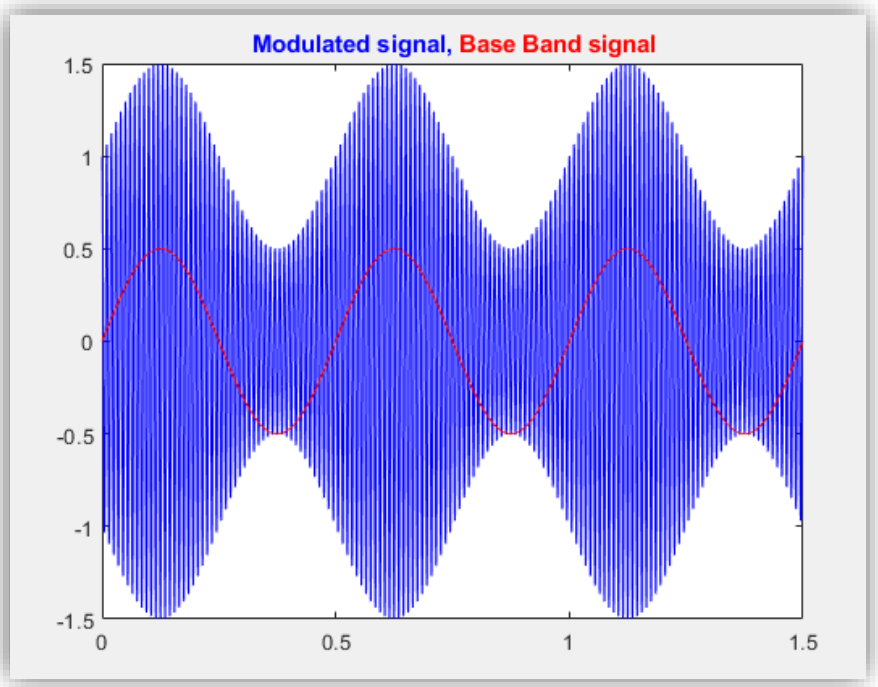
Αρχικά δημιουργούμε το σήμα βασικής ζώνης `sm` με συχνότητα `fm = 2 Hz` και πλάτος `Am = 0.5 Volt`.
Στην συνέχεια δημιουργούμε το φέρον σήμα `sc` με συχνότητα `fc = 100 Hz` και πλάτος `Ac = 1 Volt`.
Το διαμορφωμένο σήμα `x1` προκύπτει ως το γινόμενο του `sm` και `sc` με την προσθήκη ακόμη ενός φέροντος `sc`.

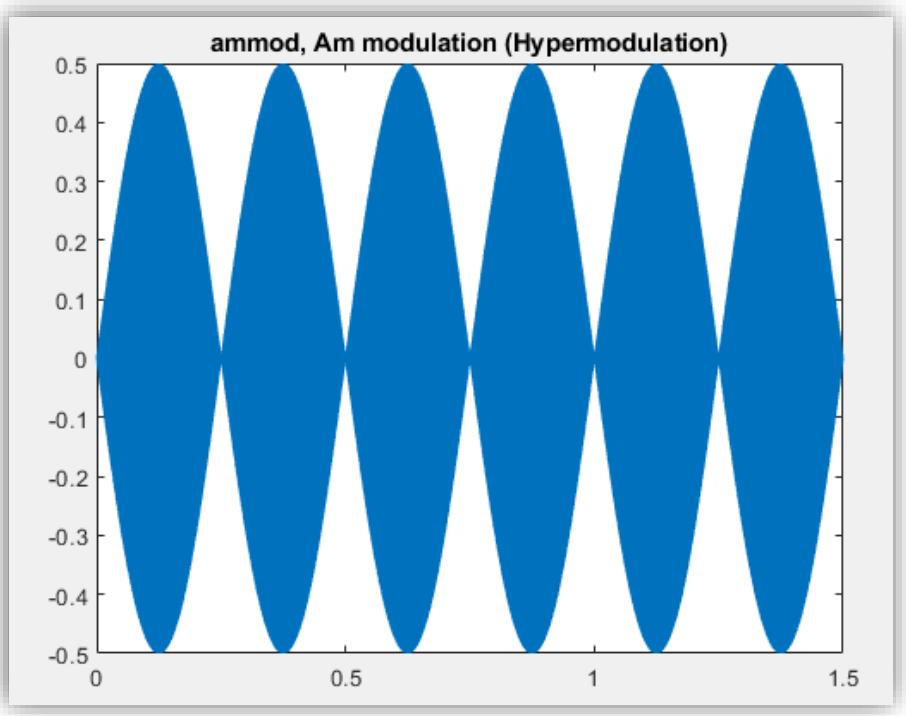
```
x1 = sm.*sc + Ac*sc;
```

Με την προσθήκη ακόμη ενός φέροντος αποφεύγεται η παραμόρφωση της περιβάλλουσας από τις αρνητικές τιμές του σήματος βασικής ζώνης `sm` (υπερδιαμόρφωση).

Τέλος δημιουργούμε το σήμα `x2` το οποίο προκύπτει ως το αποτέλεσμα της διαμόρφωσης με την χρήση της συνάρτησης `ammod()`. Το φέρον και το σήμα βασικής ζώνης παραμένουν ίδια. Η συχνότητα δειγματοληψίας είναι ίση με `fs = 400 Hz`. Η συνάρτηση `ammod()` εκτελεί αναλογική διαμόρφωση πλάτους.

<u>Αποτελέσματα:</u>





Στο παραπάνω Plot φαίνεται το αποτέλεσμα της διαμόρφωσης πλάτους, με την χρήση της συνάρτησης ammod(). Η περιβάλλουσα έχει παραμορφωθεί από τις αρνητικές τιμές του σήματος βασικής ζώνης.

7) Να δημιουργηθεί δειγματοληπτημένο σήμα θορύβου, κανονικής κατανομής (να χρησιμοποιηθεί η randn) με χρονικό άξονα t=0:0.001:.5. Να κβαντιστεί α) σε 8 και β) σε 16 επίπεδα κβάντισης. Να γίνουν στο ίδιο γραφικό παράθυρο οι γραφικές παραστάσεις του αρχικού σήματος, του κβαντισμένου, των ορίων των επιπέδων κβάντισης, των ζωνών κβάντισης και του σφάλματος κβάντισης.

Κώδικας:

ask7.m

```
% x = noise signal
t = 0:0.001:.5;
N = size(t);
x = randn(N);
Amp = max(abs(x));


% 8 levels quantization
Nq = 8;
step = Amp/(Nq/2);
partition = (-Amp+step:step:Amp-step);
```

```
codebook = (-Amp+(step/2):step:Amp-(step/2));

[~,quants,~] = quantiz(x,partition,codebook);
plot(t,x,'g')
hold on
stem(t,quants,'m')


hold on
qerr=x-quants;
max(abs(qerr))
plot(t, qerr,'k');
title('\color{green} Noise Signal Sampled, \color{magenta} Noise
Signal Quantized at 8 Levels, \color{black} Quant. Error')
grid on

% 16 levels quantization
Nq = 16;
step = Amp/(Nq/2);
partition = (-Amp+step:step:Amp-step);
codebook = (-Amp+(step/2):step:Amp-(step/2));

[~,quants,~] = quantiz(x,partition,codebook);
figure
plot(t,x,'g')
hold on
stem(t,quants,'m')
hold on

% qerr = Quntization error
qerr=x-quants;
max(abs(qerr))
plot(t, qerr,'k');
title('\color{green} Noise Signal Sampled, \color{magenta} Noise
Signal Quantized at 16 Levels, \color{black} Quant. Error')
grid on
```

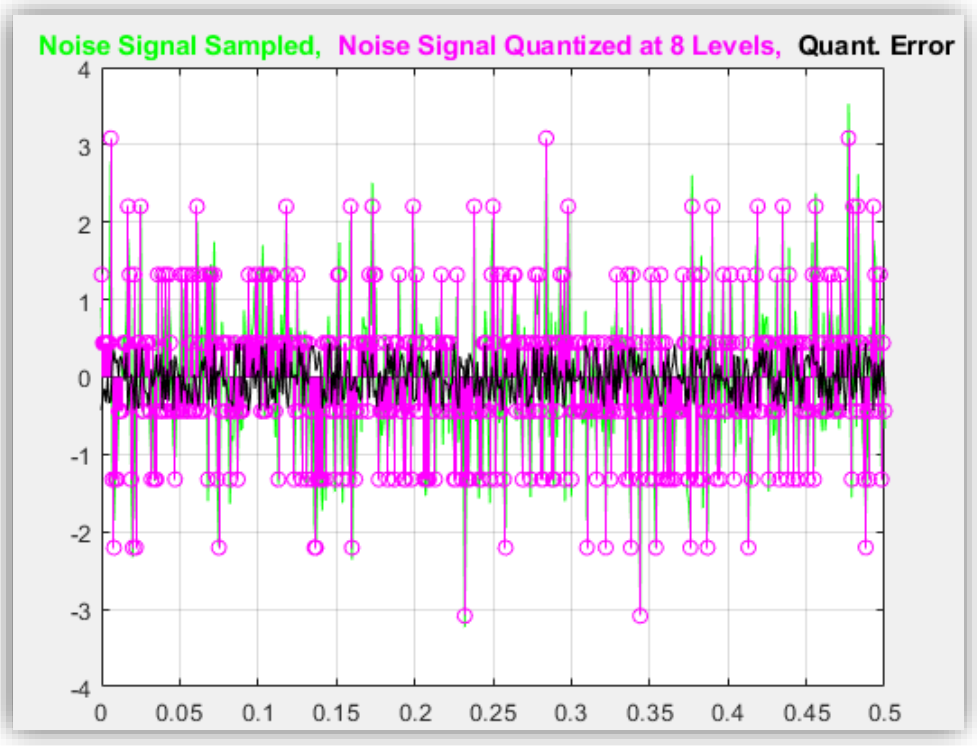
Αρχικά δημιουργούμε το σήμα θορύβου `x` με πλάτος `Amp = max(abs(x))`.
Έπειτα υπολογίζουμε το βήμα κβάντισης `step = Amp/(Nq/2)`, όπου `Nq/2` είναι τα χωρία κβάντισης. Το κάθε χωρίο αποτελείται από δύο επίπεδα κβάντισης.

Αποθηκεύουμε στο διάστημα `partition` όλα τα επιμέρους επίπεδα κβάντισης.
Αποθηκεύουμε, επίσης, στο διάστημα `codebook` όλα τα επιμέρους χωρία κβάντισης.

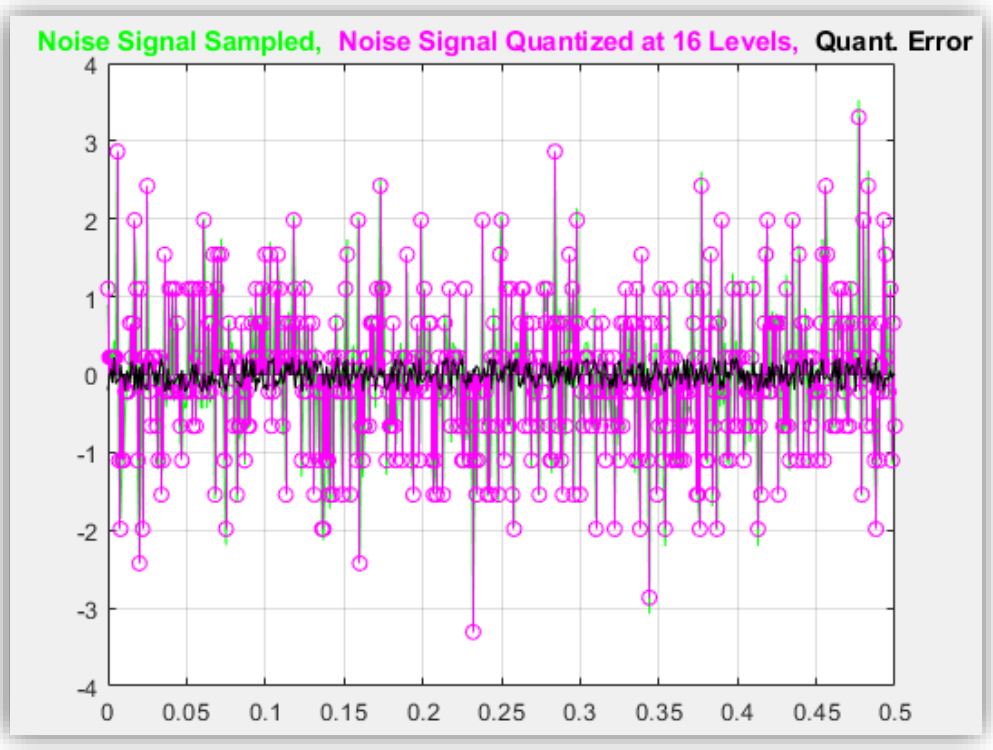
Ύστερα υπολογίζεται το κβαντισμένο σήμα `quants` με την χρήση της συνάρτησης `quantiz()` για 8 και 16 επίπεδα κβάντισης.

Τέλος υπολογίζουμε τις αποστάσεις μεταξύ των τιμών του κβασντισμένου σήματος θορύβου και του αρχικού, ως το σφάλμα κβάντισης `qerr`.

Αποτελέσματα:



Στο παραπάνω Plot φαίνεται το αρχικό σήμα θορύβου (πράσινο), το κβαντισμένο σήμα θορύβου σε 8 επίπεδα(magenta) και το σφάλμα κβάντισης (μαύρο).



Στο παραπάνω Plot φαίνεται το αρχικό σήμα θορύβου (πράσινο), το κβαντισμένο σήμα θορύβου σε 16 επίπεδα(magenta) και το σφάλμα κβάντισης (μαύρο).
Παρατηρούμε ότι το σφάλμα μειώθηκε στην περίπτωση της κβάντισης σε 16 επίπεδα.

8) Παλμοκωδική διαμόρφωση Να φτιαχτεί πρόγραμμα που θα διαμορφώνει κατά PCM ένα ημιτονικό σήμα πλάτους 4V και συχνότητας 10 Hz. Ο κβαντιστής θα χρησιμοποιεί 4 bit κωδικής λέξης.
Να σχεδιαστεί το αρχικό σήμα, το κβαντισμένο και το σφάλμα κβάντισης.

Κώδικας:

ask8.m

```
% y = Initial signal
A=4;
f=10;
T=1/f;
Tw=2*T;
dt=T/100;
t=dt:dt:Tw;
```

```
y=A*sin(2*pi*f*t);
plot(t,y,'r')
hold on

% quants = quantized signal
bits=4;
Nq = 2^bits;
step = A/(Nq/2);
partition = (-A+step:step:A-step);
codebook = (-A+(step/2):step:A-(step/2));
[~,quants,~] = quantiz(y,partition,codebook);
plot(t,quants, 'b')
hold on

% qerr = Quantization error
qerr=y-quants;
plot(t, qerr, 'k');
legend('Sin(2?10t)', 'Quantized signal', 'Qerror')
title('\color{red}Sinusoidial signal of 10 Hz, \color{blue}
Quantized at 2^4 Levels')
grid on
```

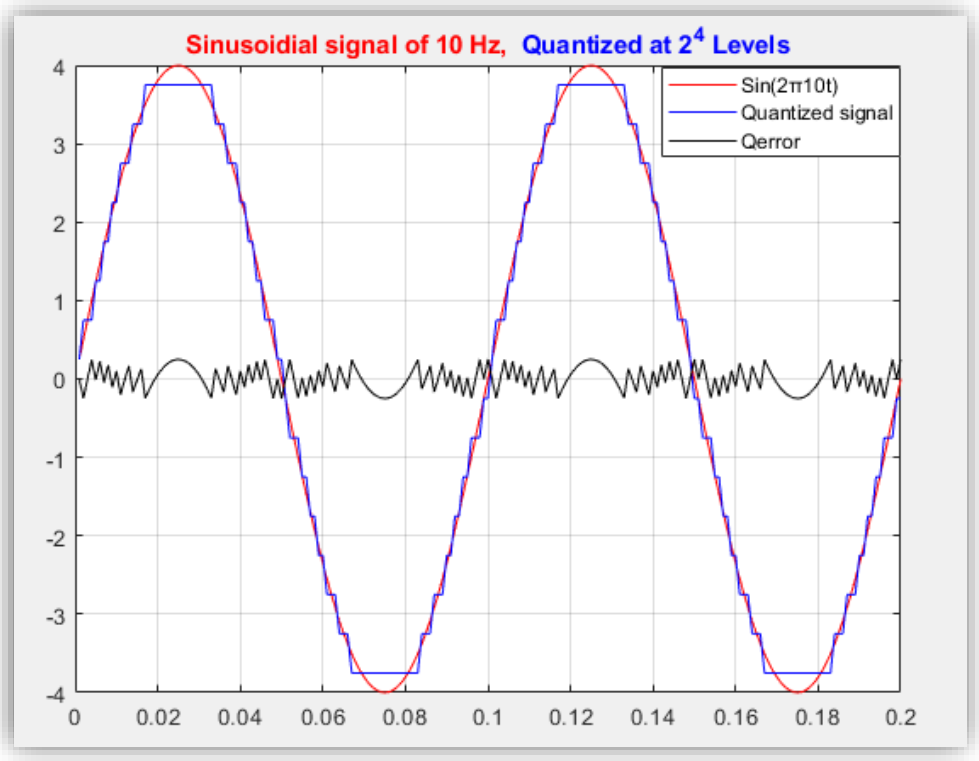
Αρχικά δημιουργούμε το σήμα `y` με συχνότητα `f = 10 Hz` και πλάτος `A = 4 Volt`. Προκειμένου να εφαρμόσουμε παλμοκωδική διαμόρφωση (PCM), τα επίπεδα κβάντισης θα είναι 2^{bits}.
Ο κβαντιστής θα χρησιμοποιεί `4 bits` κωδικής λέξης, οπότε `Nq = 16` επίπεδα.
Έπειτα υπολογίζουμε το βήμα κβάντισης `step = A/(Nq/2)`, όπου `Nq/2` είναι τα χωρία κβάντισης. Σε κάθε ένα χωρίο αντιστοιχούν δύο επίπεδα κβάντισης.

Αποθηκεύουμε στο διάστημα `partition` όλα τα επιμέρους επίπεδα κβάντισης.
Αποθηκεύουμε, επίσης, στο διάστημα `codebook` όλα τα επιμέρους χωρία κβάντισης.

Ύστερα υπολογίζεται το κβαντισμένο σήμα `quants` με την χρήση της συνάρτησης `quantiz()`.

Τέλος υπολογίζουμε τις αποστάσεις μεταξύ των τιμών του κβασντισμένου σήματος και του αρχικού, ως το σφάλμα κβάντισης.

Αποτελέσματα:



Στο παραπάνω Plot φαίνεται το αρχικό σήμα `y` με κόκκινο χρώμα, το κβαντισμένο σήμα `quants` με μπλέ χρώμα και το σφάλμα κβάντισης `qerr` με μαύρο χρώμα.

9) A) Δημιουργήστε σήμα λευκού Gaussian θορύβου μήκους 100000 δειγμάτων με χρήση της randn και κάντε το γράφημά του. Ας θεωρηθεί ότι η Gaussian pdf έχει μέση τιμή 0 και τυπική απόκλιση σ= 2 (διασπορά σ^2 =4)

Σε νέο γράφημα δημιουργήστε το ιστόγραμμα του σήματος (θεωρήστε 100 bins) και διαπιστώστε τη χαρακτηριστική μορφή της Gaussian pdf.

Δημιουργήστε δεύτερο σήμα λευκού Gaussian θορύβου μήκους 100000 δειγμάτων με χρήση της randn και κάντε το γράφημά του. Ας θεωρηθεί ότι η Gaussian pdf έχει μέση τιμή 2 και τυπική απόκλιση σ= 1 (διασπορά σ^2 =1)

Δημιουργήστε το ιστόγραμμα του σήματος και απεικονίστε το στο ίδιο γράφημα με το προηγούμενο ιστόγραμμα.

Κώδικας:

ask9a.m

```
% noise1, σ = 2, samples = 100000, μ = 0
tw = 1;
bins=100;
samples = 1e+5;
dt=tw/samples;
t=dt:dt:tw;
mu=0;sigma=2;
noise1= sigma *randn(1,1e+5)+mu;
plot(t,noise1)
title('Gaussian signal Noise1, ?=2, ?=0, samples=100000')
xlabel('Time sec')
ylabel('PSD')

% noise2, σ = 1, samples = 100000, μ = 2
mu=2;sigma=1;
noise2= sigma *randn(1,1e+5)+mu;
figure
plot(t,noise2)
title('Gaussian signal Noise2, ?=1, ?=2, samples=100000')
xlabel('Time sec')
ylabel('PSD')

% Histograms
figure
histogram(noise1,bins)
hold on
histogram(noise2,bins)
title('Histogram plot (100bins)')
legend('Noise1', 'Noise2')
xlabel('PSD')
ylabel('Number of Elements')
```

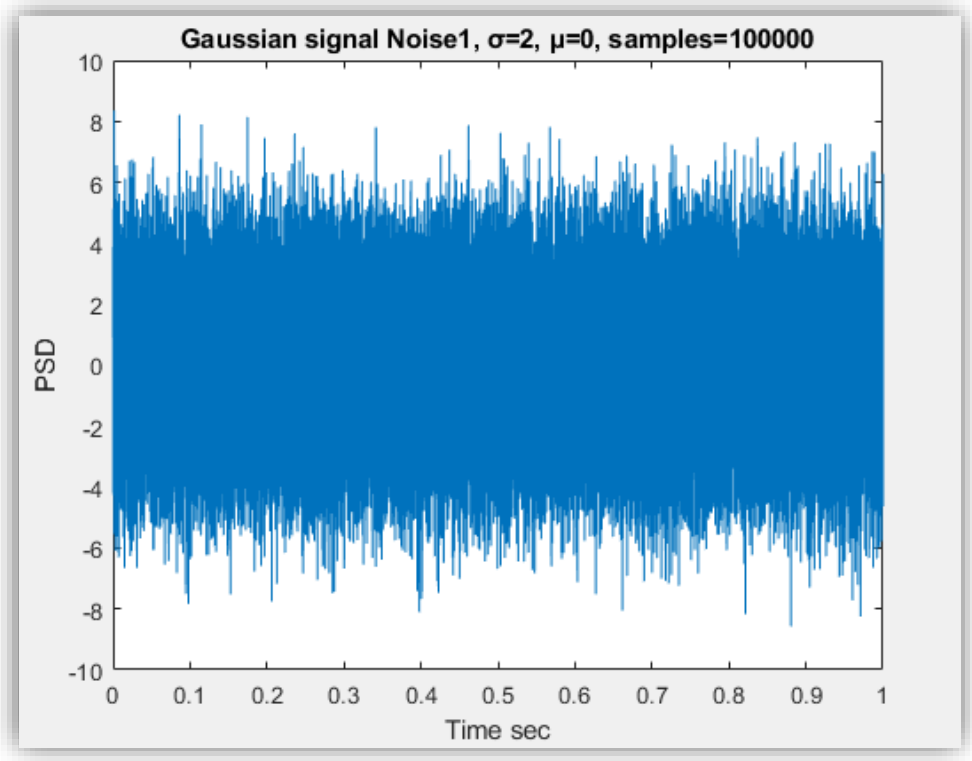
Αρχικά δημιουργούμε το σήμα θορύβου `noise1` 1000 δειγμάτων. Στην συνάρτηση κατανομής η τυπική απόκλιση θα είναι ίση με σ = 2 και μέση τιμή μ = 0.

Στην συνέχεια δημιουργούμε το σήμα θορύβου `noise2` 1000 δειγμάτων με τυπική απόκλιση σ = 1 και μέση τιμή μ = 2.

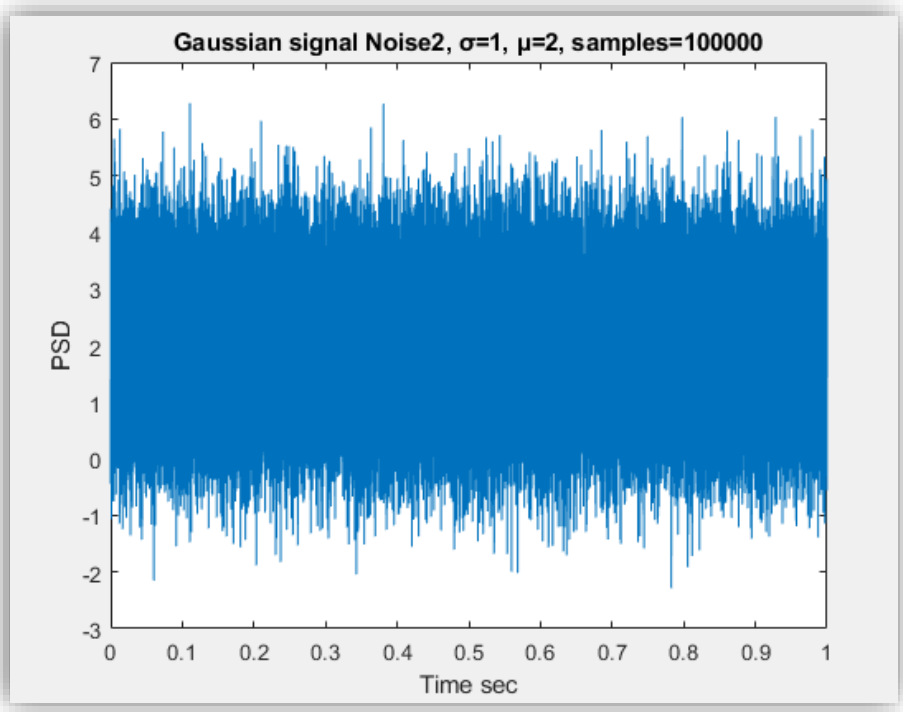
Έπειτα δημιουργούμε για κάθε ένα σήμα θορύβου και το αντίστοιχο ιστόγραμμα. Στα ιστογράμματα το πλήθος των χωρίων θα είναι ίσο με `bins = 100`.

Στα ιστογράμματα, για κάθε χωρίο της φασματικής πυκνότητας ισχύος αντιστοιχεί το πλήθος των δειγμάτων, που η τιμή τους βρίσκεται εντός του συνόλου τιμών του εκάστοτε χωρίου.

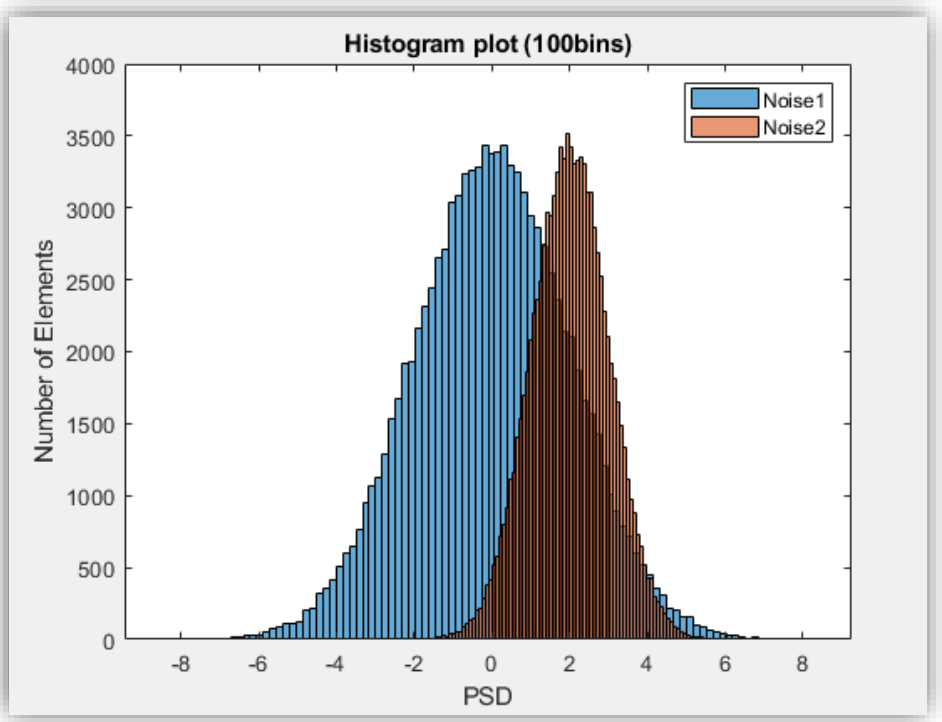
<u>Αποτελέσματα:</u>



Στο παραπάνω Plot φαίνεται το σήμα θορύβου `noise1`.



Στο παραπάνω Plot φαίνεται το σήμα θορύβου `noise2`.



Στο παραπάνω Plot φαίνονται τα ιστογράμματα των σημάτων θορύβου `noise1` και `noise2`. Παρατηρούμε ότι η καμπύλη της κατανομής του `noise2` είναι πιο στενή, λόγω της μικρότερης τυπικής απόκλισης.

9) Β) Δημιουργήστε μία τυχαία διαδικασία λευκού Gaussian θορύβου αποτελούμενη από 1000 τυχαίες ακολουθίες 512 δειγμάτων η καθεμιά που ακολουθούν Gaussian κατανομή (π.χ. σε ένα πίνακα: v 1000x512).
Η κάθε ακολουθία θα έχει μηδενική μέση τιμή και τυπική απόκλιση σ=2.
Για κάθε μία ακολουθία κάντε χρήση του ακόλουθου κώδικα για εύρεση της psd

V(i,:) = abs(fft(v(i,:))).^2/512; % periodogram

Βρείτε το μέσο των psd και απεικονίστε γραφικά το αποτέλεσμα. Επιβεβαιώστε ότι η τιμή της psd είναι σταθερή και ισούται με σ^2.

Κώδικας:

ask9b.m

```
% 1000 sequences of 512 samples, ? = 2, ? = 0
seq=1e+3;
samples=512;
mu=0;sigma=2;
v(1:seq,1:samples) = sigma*randn(seq,samples)+mu;

% psd vetor calculation
psd_v(1:seq,1:samples) = 0;
for i = 1:seq
    psd_v(i,:) = abs(fft(v(i,:))).^2/512;
end
size(psd_v)

% mean vector calulation
mean(1:seq)=sum(psd_v,2);
for i = 1:seq
    mean(i) = psd_v(i)/samples;
end
plot((1:1:seq), mean);
title('PSD mean values from sequence 1 to 1000')

% Mean = Total mean value
Mean = abs(sum(mean)/seq)
```

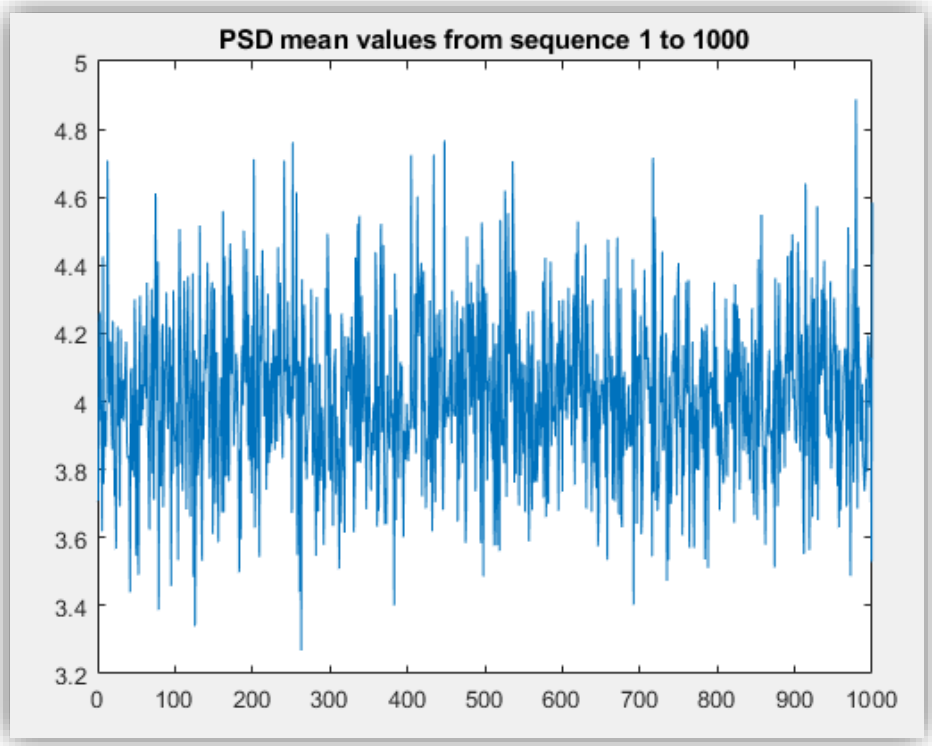
Αρχικά θα αποθηκεύσουμε τις 1000 ακολουθίες των τυχαίων κατανομών, στον πίνακα

`v(1:seq,1:samples)`

Η κάθε ακολουθία θα έχει σύνολο `samples = 512` δείγματα. Ύστερα για κάθε ένα δείγμα θα υπολογίζουμε την φασματική πυκνότητά του για κάθε ακολουθία. Οι τιμές που θα προκύψουν, θα αποθηκευτούν στον πίνακα 1000x512, `psd_v(1:seq,1:samples)`.

Στην συνέχεια, υπολογίζουμε την μέση psd τιμή για κάθε ακολουθία και τα αποτελέσματα αποθηκεύονται, στο διάνυσμα 1x1000, `mean(1:seq)`.
Τέλος υπολογίζουμε την συνολικά μέση τιμή των τιμών psd και αποθηκεύουμε το αποτέλεσμα στην μεταβλητή `Mean`.

Αποτελέσματα:



Στο παραπάνω Plot φαίνονται οι τιμές της μέσης φασματικής πυκνότητας ισχύος καθεμίας από τις 1000 ακολουθίες.

```
Mean =

    4.0146
```

Παρατηρούμε ότι η συνολική μέση φασματική πυκνότητας ισχύος είναι ίση με την διασπορά σ^2.

10) Να δημιουργηθούν τα σήματα που δίνονται από την ακόλουθη σχέση

$$m1(t) = sinc(2tf1)$$
$$m2(t) = sinc(4tf1)$$

Οπου f1=100Hz.
Να γίνει η γραφική απεικόνιση των σημάτων από -0.05 ως 0.05 sec με βήμα δειγματοληψίας dt=0.0001.
Να αναπαρασταθεί επάνω στα δύο γραφήματα και ο άξονας των x. Σε ποια σημεία συμβαίνουν οι μηδενισμοί των σημάτων και γιατί?
Να γίνει η απεικόνιση του φάσματος πλάτους των σημάτων αλλά το πλήθος των σημείων του fft (και του άξονα f) να βρεθεί από την εντολή:

```
Nf=2^ceil(log2(N));
```

Όπου N το πλήθος των σημείων του άξονα t.
Γιατί χρησιμοποιείται η εντολή αυτή?
Ποιο είναι θεωρητικά το φάσμα των σημάτων m1(t) και m2(t)? Συγκρίνετε το εύρος των δύο φασμάτων και εξηγείστε.

Κώδικας:

ask10.m

```
dt=0.0001;
t=-0.05:dt:0.05;
f1=100;

% m1 plot
m1 = sinc(2*t*f1);
plot(t,m1)
yline(0);
title('m1(t)=sinc(2tf1)')
grid on

% m2 plot
m2 = sinc(4*t*f1);
figure
plot(t,m2)
title('m2(t)= sinc(4tf1)')
yline(0);
grid on

N=length(t);
Nf=2^ceil(log2(N));
BW=1/dt;
df=BW/Nf;
f=-BW/2:df:BW/2-df;
```

```
% m1f = Amplitude spectrum of m1
m1f=fftshift(fft(m1,Nf))/Nf;
figure
plot(f, abs(m1f))
title('FT of m1(t)')
xlabel('Frequency Hz');
ylabel('Spectrum');

% m2f = Amplitude spectrum of m2
m2f=fftshift(fft(m2,Nf))/Nf;
figure
plot(f, abs(m2f))
title('FT of m2(t)')
xlabel('Frequency Hz');
ylabel('Spectrum');
```

Αρχικά δημιουργούμε τα σήματα `m1` και `m2` και στην συνέχεια τα κάνουμε plot στο πεδίο του χρόνου. Για την εμφάνιση του άξονα x χρησιμοποιούμε την εντολή `yline(0)`.

Ο μηδενισμός των σημάτων γίνεται για τις τιμές του χρόνου ≠ 0. Για t = 0, τα σήματα έχουν τιμή 1. Συγκεκριμένα τα σήματα έχουν τιμή 0 σε μη μηδενικά ακέραια πολλαπλάσια της περιόδου, εφόσον η συνάρτηση $sinct$ για τιμές του χρόνου t ≠ 0, ορίζεται ως $\frac{sint}{t}$. Επομένως μηδενίζεται στις χρονικές στιγμές t, που μηδενίζεται το ημίτονο.

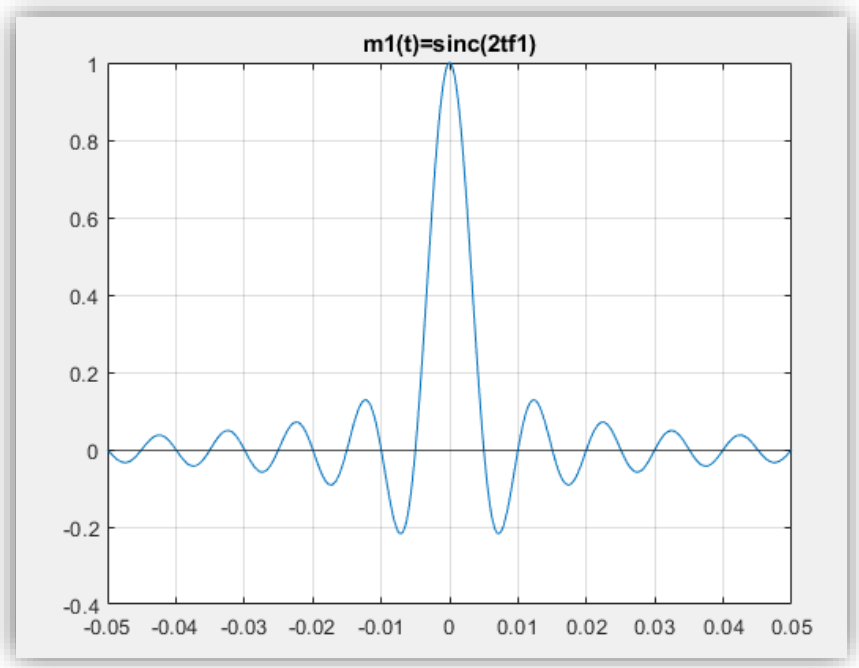
Στην συνέχεια υπολογίζουμε τα φάσματα `m1f` και `m2f` για τα σήματα `m1` και `m2` αντίστοιχα.

Για τον υπολογισμό του πλήθους `Nf` των συχνοτήτων χρησιμοποιούμε την εντολή

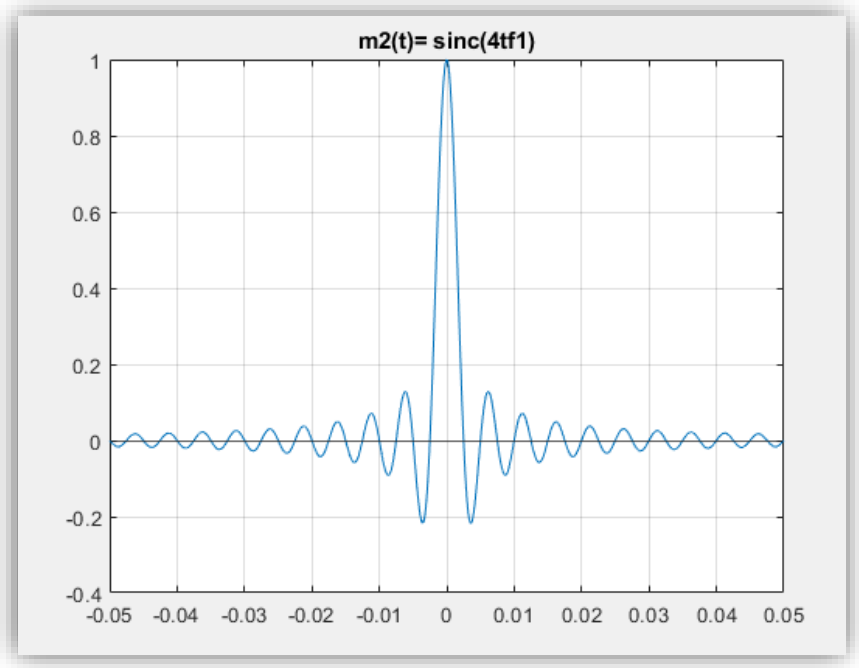
```
Nf = 2^ceil(log2(N))
```

Η εντολή αυτή μετατρέπει το πλήθος των δειγμάτων του διαστήματος του χρόνου t σε τιμή, η οποία είναι δύναμη του 2.

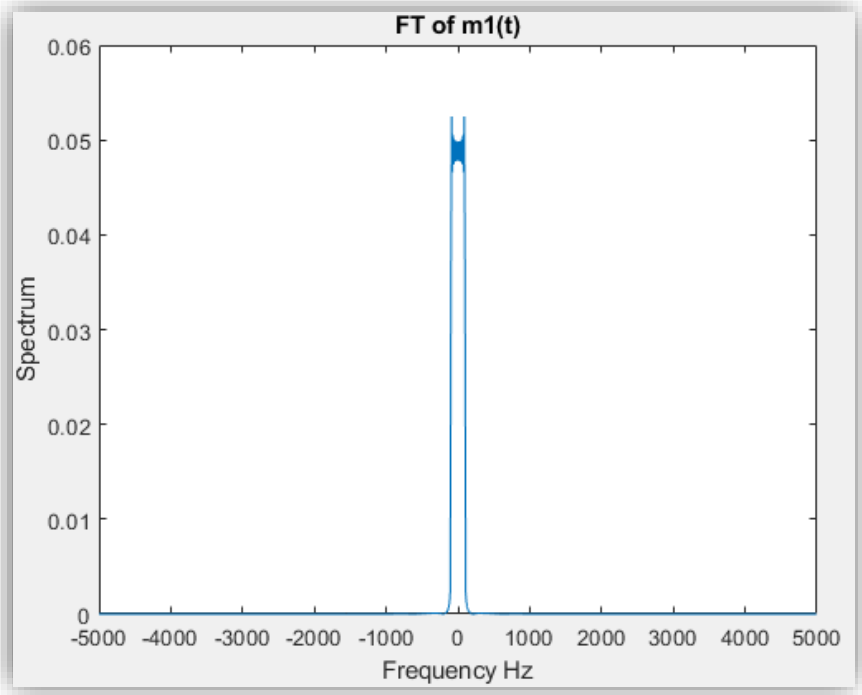
Αποτελέσματα:



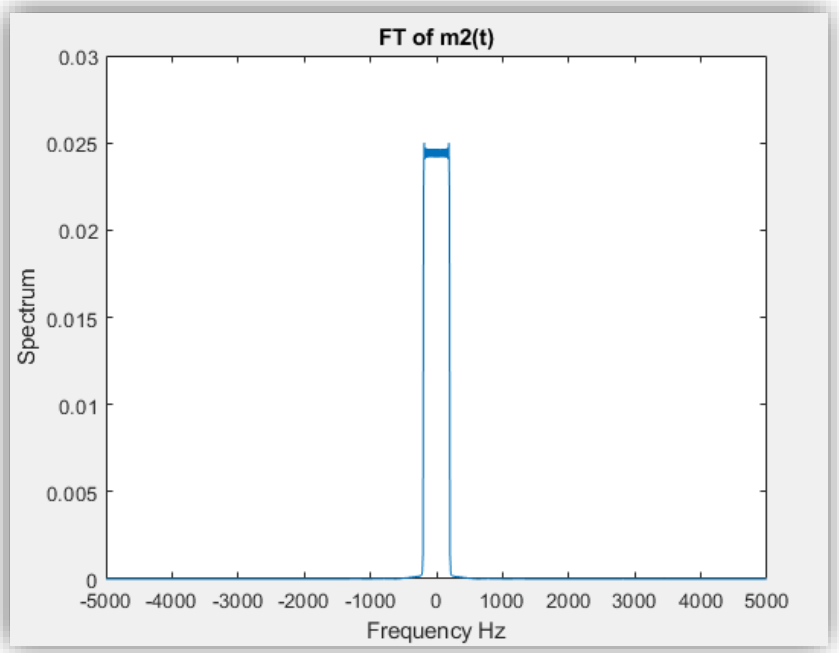
Στο παραπάνω Plot φαίνεται η αναπαράσταση του σήματος `m1,` στην συχνότητα `2*f1 = 2*100 = 200 Hz.`



Στο παραπάνω Plot φαίνεται η αναπαράσταση του σήματος `m2,` στην συχνότητα `4*f1 = 4*100 = 400 Hz.`



Στο παραπάνω Plot φαίνεται η αναπαράσταση του φάσματος του σήματος `m1` .



Στο παραπάνω Plot φαίνεται η αναπαράσταση του φάσματος του σήματος `m2` .
Το αποτέλεσμα του υπολογισμού των φασμάτων είναι το αναμενόμενο, εφόσον ο μετασχηματισμός Fourier της συνάρτησης `sinct` είναι ο τετραγωνικός παλμός `rect`. Το φάσμα του `m2` είναι δύο φορές ευρύτερο, αφού έχει την διπλάσια συχνότητα σε σχέση με το `m1` .

11) Να δημιουργηθεί σήμα πληροφορίας που δίνεται από την ακόλουθη σχέση

$$m(t) = \mathrm{sinc}(2tf1)$$

Οπου f1=100Hz.
Να γίνει η απεικόνιση του φάσματος του σήματος αλλά το πλήθος των σημείων του fft (και του άξονα f)
να βρεθεί από την εντολή

```
Nf=2^ceil(log2(N));
```

Όπου N το πλήθος των σημείων του άξονα t.
Να διαμορφωθεί το σήμα κατά DSB με φέρον συχνότητας 600Hz.
Να γίνει η γραφική απεικόνιση του διαμορφωμένου σήματος και του φάσματός του.

Κώδικας:

ask11.m

```
% m = Information signal
f1=100;
T=1/f1;
dt=T/100;
Tw=10*T;
t=dt:dt:Tw;
m = sinc(2*t*f1);

% F_m = Amplitude spectrum of information signal m
N=length(t);
Nf=2^ceil(log2(N));
BW=1/dt;
df=BW/Nf;
f=-BW/2:df:BW/2-df;
F_m=fftshift(fft(m,Nf))/Nf;
plot(f, abs(F_m))
title('FT of m(t)')
xlabel('Frequency Hz');
ylabel('PSD');

% yc = Carrier signal
fc=600;
yc=cos(2*pi*fc*t);

% ym = Modulated signal
ym = m.*yc + yc;
figure
plot(t, ym, 'b')
hold on
plot(t, m +1, 'r')
title('\color{blue}Modulated signal, \color{red}Signal m(t)
```

```
Envelope ')

% F_ym = Amplitude spectrum of modulated signal ym
F_ym=fftshift(fft(ym,Nf))/Nf;
figure
plot(f, abs(F_ym))
title('DSB Modulated signal spectrum');
xlabel('Frequency Hz');
ylabel('PSD');
```

Αρχικά δημιουργούμε το σήμα πληροφορίας `m` στην συχνότητα `2*f1 = 2*100= 200 Hz.`

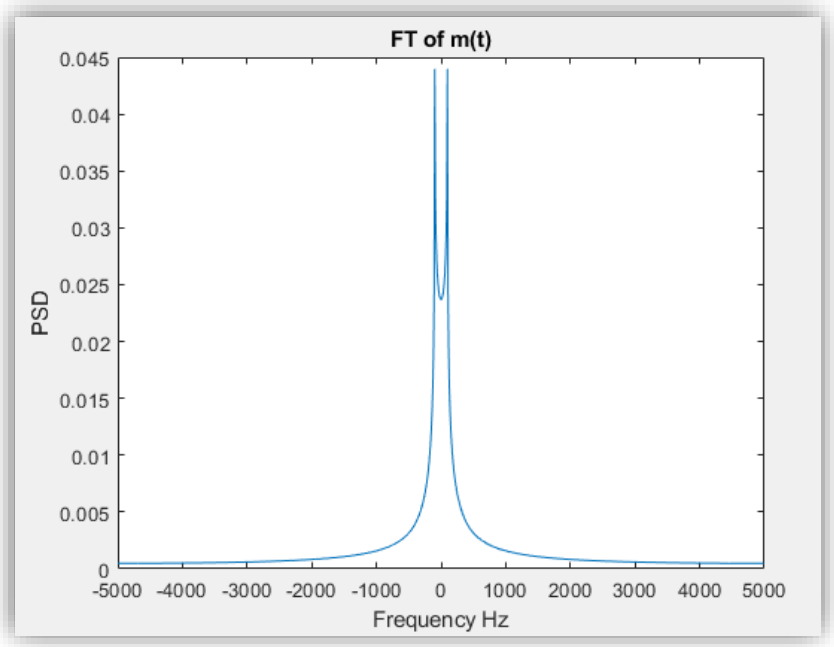
Έπειτα, υπολογίζουμε το φάσμα `F_m` του σήματος πληροφορίας, με τιμή του πλήθους συχνοτήτων σε δύναμη του 2.

```
Nf=2^ceil(log2(N))
```

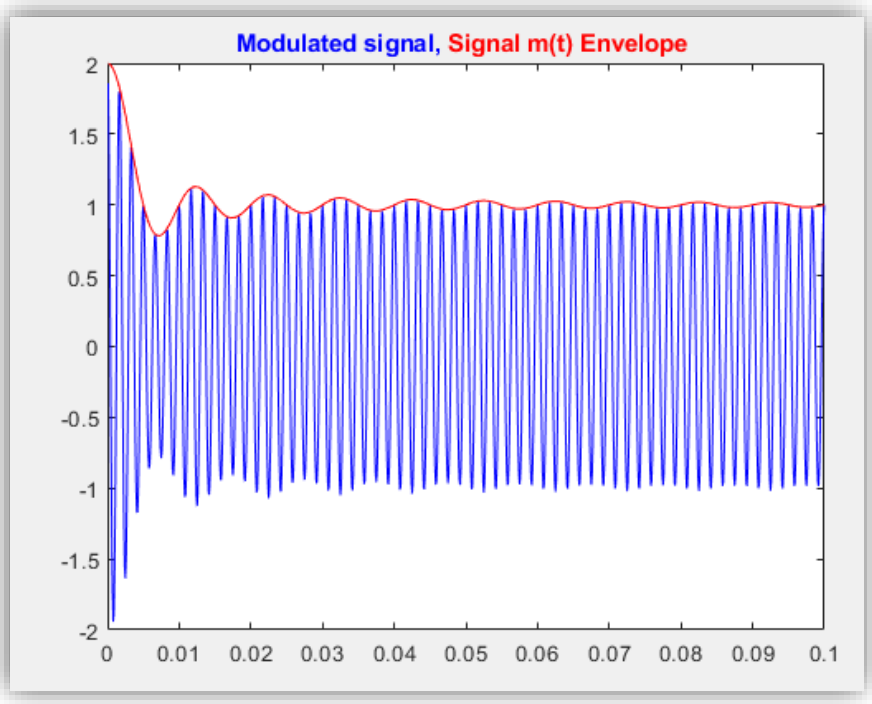
Ύστερα, δημιουργούμε το φέρον `yc` στην συχνότητα `fc=600 Hz` και υπολογίζουμε το διαμορφωμένο σήμα `ym`, το οποίο προκύπτει ως αποτέλεσμα του γινομένου του σήματος πληροφορίας με το φέρον. Στο διαμορφωμένο σήμα προστίθεται και μία συνιστώσας του φέροντος, με σκοπό την αποφυγή της παραμόρφωση της περιβάλλουσας (envelope).

Τέλος, υπολογίζουμε τα αμφίπλευρο γραμμικό φάσμα `F_ym` του διαμορφωμένου σήματος `ym`.

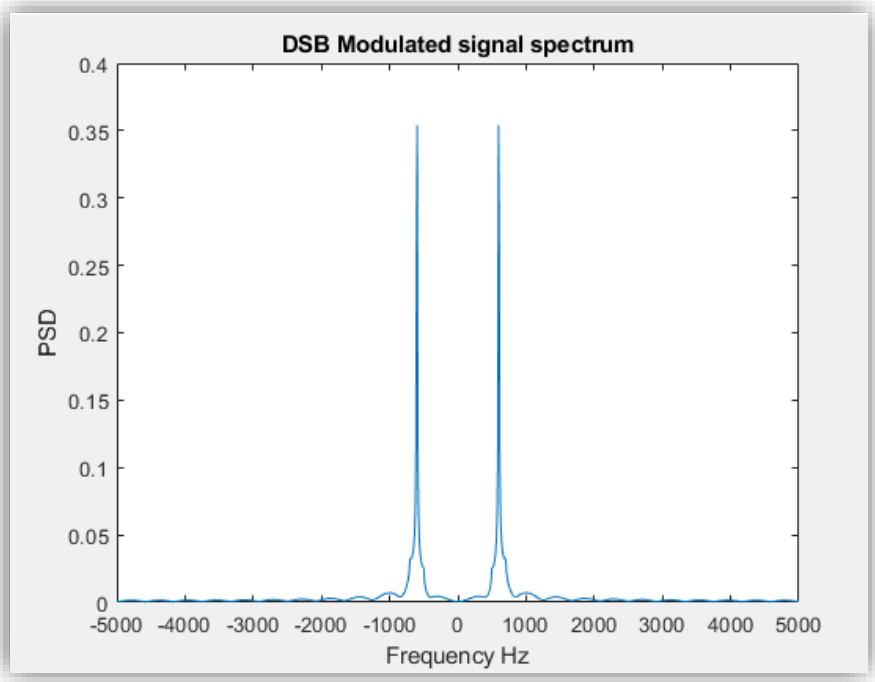
<u>Αποτελέσματα:</u>



Στο παραπάνω Plot φαίνεται το φάσμα `F_m` του σήματος πληροφορίας `m`.



Στο παραπάνω Plot φαίνεται το διαμορφωμένο σήμα `ym` και η περιβάλλουσα (envelope) με κόκκινο χρώμα.



Στο παραπάνω Plot φαίνεται το φάσμα `F_ym` διπλής πλευρικής ζώνης (DSB), του διαμορφωμένου σήματος `ym`.

12) α) Να φτιαχτεί συνημιτονοειδές σήμα πλάτους Am=0.5 Volt, συχνότητας fa=1KHz. Ο άξονας του χρόνου να οριστεί από 0 ως 5 περιόδους του σήματος. Δημιουργείστε ένα γραφικό παράθυρο και χωρίστε το σε 3 υποπαράθυρα. Στο 1ο υποπαράθυρο κάνετε τη γραφική παράσταση του x(t) ως προς t.

β) Να φτιαχτεί συνημιτονοειδές σήμα πλάτους Ac=2.5 Volt, συχνότητας fc δεκαπλάσιας της συχνότητας του σήματος πληροφορίας Ο άξονας του χρόνου παραμένει ο ίδιος.

Εμφανίστε τη γραφική παράσταση του y(t) ως προς το t στο 2ο υποπαράθυρο.

γ) Να διαμορφωθεί το φέρον σήμα, y(t), από το σήμα πληροφορίας, x(t), κατά DSB.

Το διαμορφωμένο σήμα, z(t), να αναπαρασταθεί ως προς t, στο 3ο υποπαράθυρο.

δ) Να γίνει σύμφωνη αποδιαμόρφωση του σήματος, με χρήση τοπικού ταλαντωτή, σύμφωνου με το φέρον σήμα του πομπού και στη συνέχεια να περαστεί το σήμα από χαμηλοπερατό φίλτρο butterworth_filter(in, dt, order, fcut, fcenter), με χρήση του αντίστοιχου αρχείου (επιλέξτε τις κατάλληλες παραμέτρους).

Σε νέο γραφικό παράθυρο να αναπαρασταθούν ταυτόχρονα (με πάγωμα του παραθύρου) το αρχικό σήμα πληροφορίας και το αποδιαμορφωμένο (μετά από το φίλτρο). Να επιλεχθούν διαφορετικά χρώματα για τις δύο γραφικές παραστάσεις.

ε) Να επαναλάβετε τη διαδικασία **(δ)** για ένα μη ιδανικό τοπικό ταλαντωτή με κάποια μικρή διαφορά φάσης από το φέρον σήμα (π.χ. 0.1π). Σε νέο γραφικό παράθυρο να αναπαρασταθούν ταυτόχρονα το αρχικό σήμα πληροφορίας και το αποδιαμορφωμένο (μετά από το φίλτρο).

ζ) Τέλος, να επαναληφθεί η διαδικασία για διαφορά φάσης τέτοια ώστε να υπάρχει ολική απώλεια του αποδιαμορφωμένου σήματος. Σε νέο γραφικό παράθυρο να αναπαρασταθούν ταυτόχρονα το αρχικό σήμα πληροφορίας και το αποδιαμορφωμένο (μετά από το φίλτρο).

Κώδικας:

ask12.m (Η συνάρτηση butterworth_filter.m βρίσκεται στο τέλος του αρχείου)

```
% x = Information signal
Am=0.5;
fa=1e+3;
Ta=1/fa;
dt=Ta/100;
Tw=5*Ta;
t=dt:dt:Tw;
x=Am*cos(2*pi*fa*t);
subplot(3,1,1);
plot(t,x)
```

```
title('x(t)=Am*cos(2?*fa*t)')
xlabel('time (sec)')
ylabel('x(t)')

% y = Carrier signal
Ac=2.5;
fc=10*fa;
y=Ac*cos(2*pi*fc*t);
subplot(3,1,2);
plot(t,y)
title('y(t)=Ac*cos(2?*fc*t)')
xlabel('time (sec)')
ylabel('y(t)')

% z = Modulated signal
z = x.*y + y;
subplot(3,1,3);
plot(t,z)
title('Modulated signal z(t)=y(t)*[1+x(t)]')
xlabel('time (sec)')
ylabel('z(t)')

% demod = Demodulated signal
LO=cos(2*pi*fc*t);
Signal_out=z.*LO;
demod=butterworth_filter(Signal_out, dt, 10, 1.6e3, 0);
figure
plot(t, abs(demod), 'b')
hold on
plot(t, x, 'r')
title('\color{blue}Demodulated signal d(t), \color{red}Initial
signal x(t)')
xlabel('time (sec)')
ylabel('\color{blue}d(t), \color{red}x(t)')

%Demodulation, phase shift 0.1π at demodulator
LO=cos(2.1*pi*fc*t);
Signal_out=z.*LO;
demod=butterworth_filter(Signal_out, dt, 10, 1.6e3, 0);
figure
plot(t, abs(demod), 'b')
hold on
plot(t, x, 'r')
title('\color{blue}Demodulated signal d(t) (Phase shift 0.1? at
demodulator), \color{red}Initial signal x(t)')
xlabel('time (sec)')
ylabel('\color{blue}d(t), \color{red}x(t)')
```

```
%Demodulation, phase shift π at demodulator
LO=cos(3*pi*fc*t);
Signal_out=z.*LO;
demod=butterworth_filter(Signal_out, dt, 10, 1.6e3, 0);
figure
plot(t, abs(demod), 'b')
hold on
plot(t, x, 'r')
title('\color{blue}Demodulated signal d(t) (Phase shift ? at
demodulator), \color{red}Initial signal x(t)')
xlabel('time (sec)')
ylabel('\color{blue}d(t), \color{red}x(t)')
```

Αρχικά δημιουργούμε το σήμα πληροφορίας `x(t)` στην συχνότητα `fa = 1 kHz`, με πλάτος `Am = 0.5 Volt`.

Στην συνέχεια δημιουργούμε το φέρον `y(t)` στην συχνότητα `fc = 10*fa = 10 kHz`, με πλάτος `Ac = 2.5 Volt`.

Ύστερα υπολογίζουμε το διαμορφωμένο σήμα `z(t)`, ως το γινόμενο του φέροντος `y(t)` με το σήμα πληροφορίας `x(t)`. Στο διαμορφωμένο σήμα, προσθέτουμε και μία συνιστώσα του φέροντος, έτσι ώστε να μην επηρεάσουν την περιβάλλουσα οι αρνητικές τιμές του σήματος πληροφορίας.

Στην συνέχεια, δημιουργούμε ένα σήμα `LO` το οποίο είναι ίδιο με το φέρον. Το σήμα αυτό στην πραγματικότητα, θα το δημιουργούσε ο σύμφωνος τοπικός ταλαντωτής του δέκτη, με σκοπό να αποδιαμορφώσει το λαμβανόμενο σήμα. Πολλαπλασιάζοντας το λαμβανόμενο σήμα `z(t)` με το σήμα `LO`, παίρνουμε το σήμα `Singal_out`. Για να απομονώσουμε το σήμα πληροφορίας από το σήμα `Signal_out`, χρησιμοποιούμε ένα χαμηλοπερατό φίλτρο με τις παρακάτω παραμέτρους.

```
butterworth_filter(Signal_out, dt, 10, 1.6e3, 0);
```

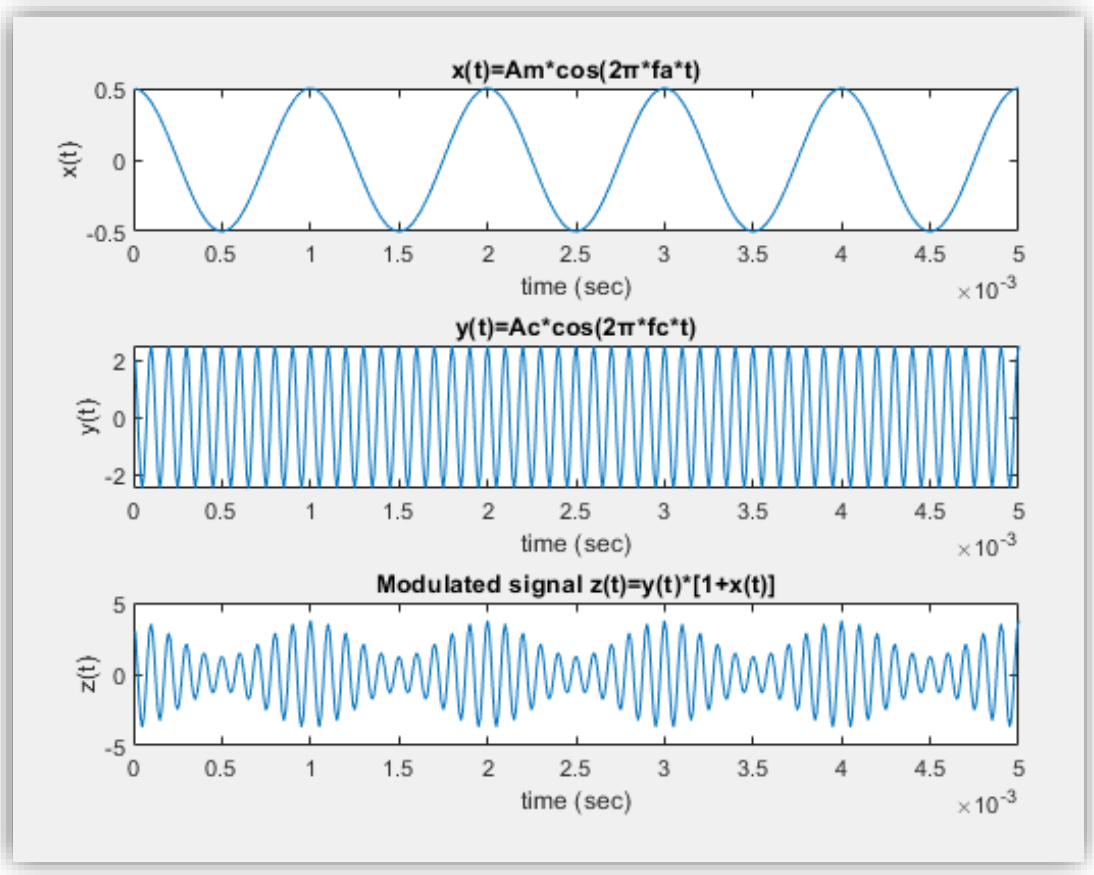
Η μεταβολή `dt = 1/Ta` υπολογίζεται από την περιόδο, `Ta = 1/fa`, του σήματος πληροφορίας `x(t)`.

Η τάξη του φίλτρου είναι ίση με `order = 10`.

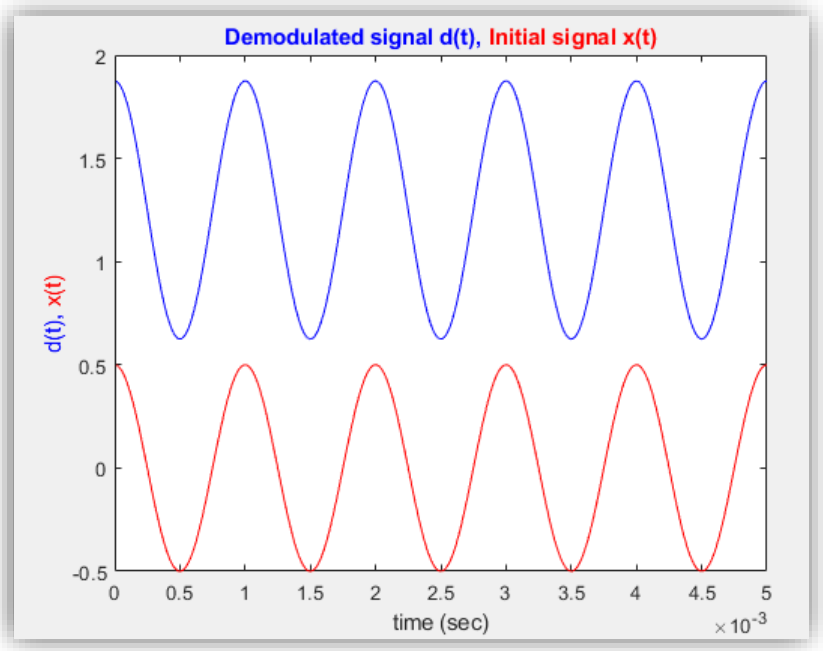
Η συχνότητα αποκοπής είναι ίση με `fcutoff = fa + 600 = 1600 Hz`, έτσι ώστε το φέρον σήμα με συχνότητα `10 kHz` να αποκοπεί.

Τέλος, εκτελούμε την ίδια διαδικασία της αποδιαμόρφωσης στον δέκτη, αλλά με διαφορά φάσης `0.1π` και π στο σήμα `LO` που παράγει ο τοπικός ταλαντωτής.

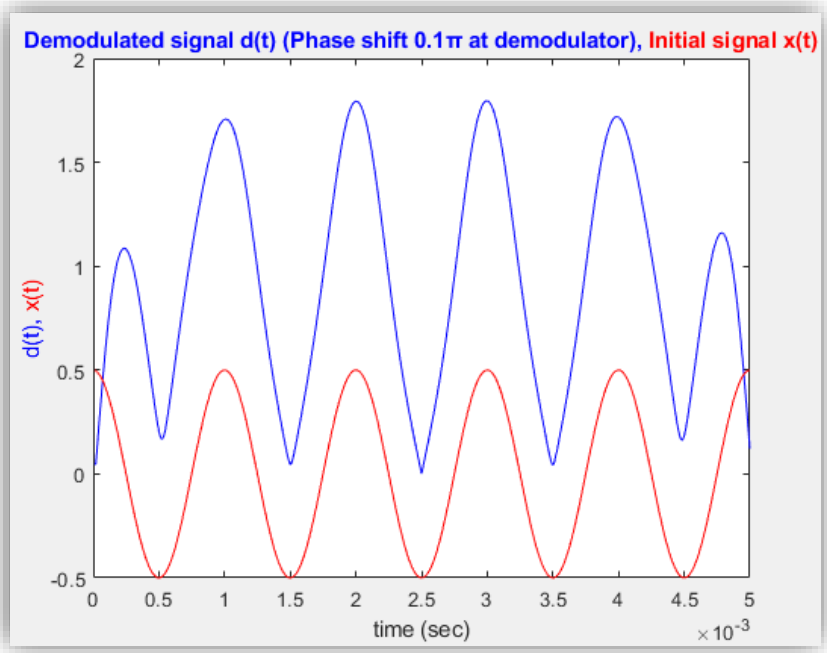
Αποτελέσματα:



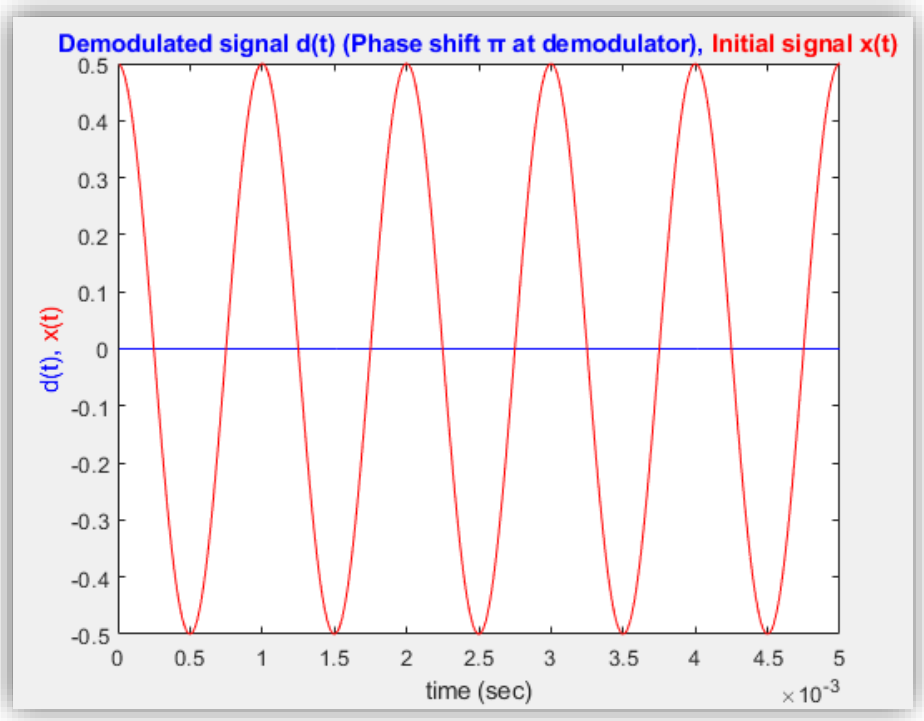
Στο παραπάνω Plot φαίνονται οι κυματομορφές του σήματος πληροφορίας `x(t)`, του φέροντος `y(t)` και του διαμορφωμένου `z(t)`.



Στο παραπάνω Plot φαίνονται το αρχικό σήμα `x(t)` (κόκκινο) και το αποδιαμορφωμένο σήμα `d(t)` (μπλε) που προκύπτει, όταν ο σύμφωνος τοπικός ταλαντωτής είναι ιδανικός.



Στο παραπάνω Plot φαίνονται το αρχικό σήμα `x(t)` (κόκκινο) και το αποδιαμορφωμένο σήμα `d(t)` (μπλε) που προκύπτει, όταν ο σύμφωνος τοπικός ταλαντωτής παρουσιάζει διαφορά φάσης `0.1π`. Παρατηρούμε ότι το αποδιαμορφωμένο σήμα δεν είναι ίδιο με το αρχικό σήμα `x(t)`.



Στο παραπάνω Plot φαίνονται το αρχικό σήμα `x(t)` (κόκκινο) και το αποδιαμορφωμένο σήμα `d(t)` (μπλε) που προκύπτει, όταν ο σύμφωνος τοπικός ταλαντωτής παρουσιάζει διαφορά φάσης `π`. Παρατηρούμε ότι το αποδιαμορφωμένο σήμα έχει αλλοιωθεί εντελώς και δεν έχει καμία ομοιότητα με το αρχικό σήμα `x(t)`.

13) Ας θεωρηθεί το θεώρημα Shannon Hartley :

$$C = B\ log2\ (1 + \frac{S}{N})\ bits/sec$$

Αν αντί της μέσης ισχύος N του θορύβου θεωρήσουμε τη φασματική πυκνότητα ισχύος N0/2 τότε η παραπάνω σχέση παίρνει τη μορφή:

$$C = B\ log2\ (1 + \frac{S}{BN_o})\ bits/sec$$

Οταν ο λόγος S/N τείνει στο άπειρο, τότε και η χωρητικότητα του καναλιού C τείνει στο άπειρο. Αντίστοιχα, όταν το εύρος ζώνης του καναλιού B, τείνει στο άπειρο η χωρητικότητα του καναλιού C τείνει σε συγκεκριμένη τιμή :

$$\lim_{B\to\infty} C = 1.44\ \frac{S}{N_o}$$

Να γράψετε πρόγραμμα υπολογισμού και αναπαράστασης της χωρητικότητας καναλιού C με εύρος ζώνης συχνοτήτων B=3000Hz σε συνάρτηση με το λόγο S/N0 (να δώσετε τιμές στο λόγο S/N0 από –20 έως 30dB). Για σύμπτυξη τιμών χρησιμοποιείστε την εντολή `semilogx` αντί για την plot.

Κώδικας:

ask13.m

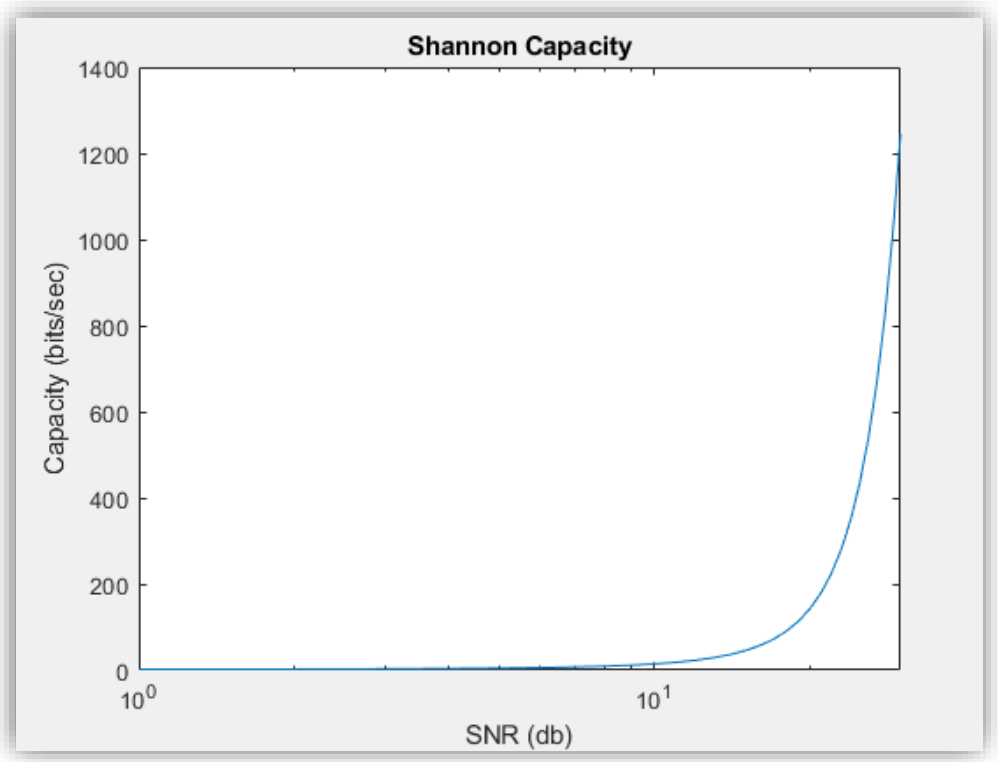
```
% B = Bandwidth
B = 3e+3;
% trasnsform db to snr value
snr_db=-20:1:30;
snr=snr_db/10;
snr=10.^snr;

% C = Channel Capacity calculation
C=B*log2(1+((1/B)*snr));
semilogx(snr_db, C)
title('Shannon Capacity')
xlabel('SNR (db)')
ylabel('Capacity (bits/sec)')
```

Αρχικά αποθηκεύουμε στο διάνυσμα `snr_db=-20:1:30` τις διάφορες τιμές του λόγου `SNR`. Στην συνέχεια, μετατρέπουμε τις τιμές του λόγου `SNR` από `db` σε καθαρή αριθμητική τιμή και τις αποθηκεύουμε στο διάνυσμα `snr`.

Τέλος, υπολογίζουμε την χωρητικότητα του καναλιού `C` για τις διάφορες τιμές του διανύσματος `snr` και για εύρος ζώνης, `B = 3000 Hz`.

Με την εντολή `semilogx(snr_db, C)` γίνεται αναπαράσταση των τιμών της χωρητικότητας `C` και του διανύσματος `snr_db` σε λογαριθμική κλίμακα με βάση το 10.

Αποτελέσματα:



Στο παραπάνω Plot φαίνεται η γραφική παράσταση της χωρητικότητας C του καναλιού για τις διάφορες τιμές του λόγου Σήματος/Θόρυβο. `snr_db=-20:1:30`
Παρατηρούμε ότι η χωρητικότητα αυξάνεται εκθετικά. Επομένως όταν ο λόγος SNR τείνει στο άπειρο, δηλαδή ο θόρυβος μειώνεται, τόσο η χωρητικότητα τείνει και αυτή στο άπειρο, για τιμή Bandwidth < ∞.

14) Να γράψετε πρόγραμμα υπολογισμού και αναπαράστασης της χωρητικότητας καναλιού C με λόγο S/N0 ίσο με 25dB σε συνάρτηση με το εύρος ζώνης συχνοτήτων του καναλιού B (τιμές από 1 έως100000). Για σύμπτυξη τιμών χρησιμοποιείστε την εντολή semilogx αντί για την plot. Η τιμή στην οποία τείνει η χωρητικότητα είναι η αναμενόμενη με βάση τη Θεωρία?

Κώδικας:

ask14.m

```
% B = Bandwidth
B = 1:1:1e+5;
% trasnsform db to snr value
snr_db=25;
snr=snr_db/10;
snr=10.^snr;

% C = Channel Capacity calculation
C=B.*log2(1+((1./B).*snr));
semilogx(B, C)
title('Shannon Capacity ')
xlabel('SNR (db)')
ylabel('Capacity (bits/sec)')

% Capacity value as Bandwidth tends to infinity
C_lim = 1.44*snr
```

Αρχικά αποθηκεύουμε τις διάφορες τιμές του εύρους ζώνης στο διάνυσμα

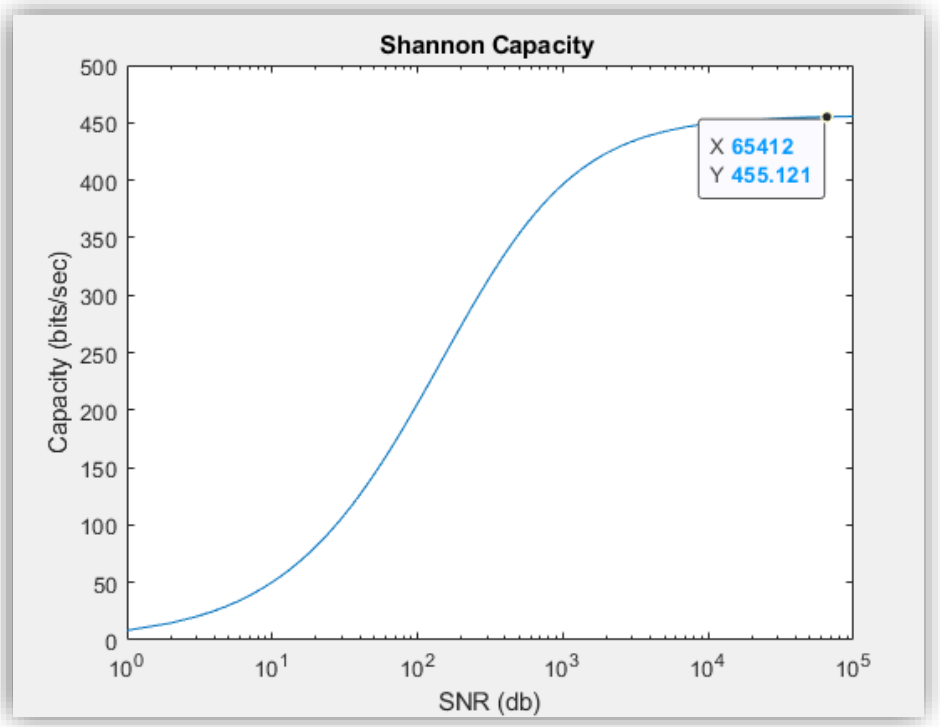
`B = 1:1:1e+5`

Ύστερα μετατρέπουμε τον λόγο Σήματος/Θόρυβο `snr_db = 25 db,` από `db` σε καθαρή αριθμητική τιμή `snr`.

Έπειτα, υπολογίζουμε την χωρητικότητα του καναλιού `C` και αναπαριστούμε τις τιμές της συναρτήσει του λόγου `snr_db` σε κλίμακα log_{10}.

Τέλος, υπολογίζουμε την θεωρητική τιμή της χωρητικότητας `C` καθώς $B \to \infty$.

Αποτελέσματα:



Στο παραπάνω Plot φαίνεται η γραφική παράσταση της χωρητικότητας `C` του καναλιού για τις διάφορες τιμές του εύρους ζώνης. `B = 1:1:1e+5`
Παρατηρούμε ότι η χωρητικότητα του καναλιού `C` συγκλίνει σε συγκεκριμένη τιμή `C = 455.121` καθώς το εύρος ζώνης τείνει στο άπειρο.
Για την επαλήθευση της τιμής για την οποία φράσετε η χωρητικότητα, χρησιμοποιούμε τον τύπο: $\lim_{B\to\infty} C = 1.44\frac{S}{N_o}$
Το θεωρητικό αποτέλεσμα της χωρητικότητας `C,` καθώς το εύρος ζώνης τείνει στο άπειρο παρατίθεται παρακάτω.

```
C_lim =

  455.3680
```

Παρατηρούμε ότι το θεωρητικό αποτέλεσμα όσο και το αποτέλεσμα της γραφικής παράστασης είναι όμοια, για την περίπτωση που $B \to \infty$.

15) Να δημιουργηθεί η ακολουθία μηνύματος:

'my name is <insert here>'

Να αποθηκευτεί το μήνυμα σε μεταβλητή με όνομα myname. Να δημιουργηθεί το αλφάβητο της πηγής και το διάνυσμα των πιθανοτήτων εμφάνισης των συμβόλων. Να δημιουργηθεί το Huffman codebook. Να γίνει κωδικοποίηση Huffman του μηνύματος. Να υπολογιστεί η εντροπία της πηγής, η απόδοση του κώδικα και ο πλεονασμός του.

Κώδικας:

ask15.m

```
% Store string str into myname variable
myname = 'my name is Dimitris';
str = myname;

% prob = probability vector
prob=[];
syms=[];
i=1;
while ~isempty(str)
    [char,no_occurrances]=mode(str);
    syms(i)=char;
    prob(i)=no_occurrances;
    indices=find(str==char);
    str(indices)='';
    i=i+1;
end
n=length(myname);
prob=prob/n;

% Huffman Encoding using the huffman dictionary
[dict,avglength]= huffmandict(syms, prob);
binstream = huffmanenco(myname, dict);
msgdeco= huffmandeco(binstream, dict);

% Check the if the initial and the decoded message are the
same
isequal(myname,msgdeco)

% Source entropy calculation
Hi=prob.*log2(1./(prob+1e-9));
entropy=sum(Hi)

% Code efficiency percentage
code_eff=(entropy/avglength)*100
% Huffman code redundancy percentage
code_red=(1-(entropy/avglength))*100
```

Αρχικά αποθηκεύουμε το αλφαριθμητικό `'my name is Dimitris'` στην μεταβλητή `myname`. Αντιγράφουμε το περιεχόμενο της `myname` στην μεταβλητή `str`.
Με τον παρακάτω κώδικα αποθηκεύονται τα σύμβολα του αλφαβήτου για το αλφαριθμητικό `str`, στο διάνυσμα `syms`. Σε κάθε επανάληψη διαβάζεται ένα σύμβολο από το αλφαριθμητικό `str`. Ύστερα, το κάθε σύμβολο που διαβάζεται με την συνάρτηση `mode()`, αποθηκεύεται στην μεταβλητή `char`.

```
while ~isempty(str)
    [char,no_occurrances]=mode(str);
    syms(i)=char;
    prob(i)=no_occurrances;
    indices=find(str==char);
    str(indices)='';
    i=i+1;
end
```

Για κάθε σύμβολο επιστρέφεται ο αριθμός των εμφανίσεών του στο αλφαριθμητικό `str`. Το πλήθος των εμφανίσεων αποθηκεύεται στην μεταβλητή `no_occurrences`. Το πλήθος των εμφανίσεων αποθηκεύεται στην αντίστοιχη θέση `i` του διανύσματος των πιθανοτήτων `prob`. Έπειτα, κάθε επανεμφάνιση του συμβόλου char αντικαθίσταται με το white space (‘ ’). Η διαδικασία αυτή, εκτελείται μέχρι το αλφαριθμητικό str να είναι άδειο, δηλαδή να περιέχει μόνο white spaces.

Στην συνέχεια υπολογίζουμε το μήκος του αλφαριθμητικού `myname` και το αποθηκεύουμε στην μεταβλητή `n`.

```
n=length(myname);
```

Το τελικό διάνυσμα πιθανοτήτων `prob` προκύπτει από την διαίρεση των εμφανίσεων `prob(i)` κάθε συμβόλου `i`, με το μήκος `n` του αλφαριθμητικού `myname`.

```
prob=prob/n;
```

Έχοντας το διάνυσμα των συμβόλων, `syms`, και των αντίστοιχων πιθανοτήτων τους, `prob`, υπολογίζουμε τις κωδικές λέξεις του αλφαβήτου με την συνάρτηση `huffmandict()`.

```
[dict,avglength] = huffmandict(syms, prob);
```

Στο διάνυσμα `dict` αποθηκεύονται οι κωδικές λέξεις όλων των συμβόλων, που περιέχονται στο διάνυσμα `syms`. Το διάνυσμα `syms` περιέχει το αλφάβητο του αλφαριθμητικού `myname`.
Στην μεταβλητή `avglength` αποθηκεύεται το μέσο μήκος των κωδικών λέξεων.

Η δυαδική ακολουθία για το αλφαριθμητικό `myname` υπολογίζεται με την χρήση της συνάρτησης `huffmanenco()`.

```
binstream = huffmanenco(myname, dict);
```

Στο διάνυσμα `binstream` αποθηκεύονται οι τιμές των `bits` της δυαδικής ακολουθίας. Η αποκωδικοποίηση της δυαδικής ακολουθίας `binstream` εκτελείται με την χρήση της συνάρτησης `huffmandeco()`.

```
msgdeco= huffmandeco(binstream, dict);
```

Η αποκωδικοποιημένη ακολουθία συμβόλων αποθηκεύεται στο διάνυσμα `msgdeco`.
Με την εντολή `isequal(myname,msgdeco)` ελέγχουμε εάν το αλφαριθμητικό `myname` έχει την ίδια τιμή με την αποκωδικοποιημένη ακολουθία `msgdeco`.

```
ans =

  logical

   1
```

Πράγματι, τα διανύσματα `msgdeco` και `myname` έχουν ίδιο περιεχόμενο. Οπότε η κωδικοποίηση κατά Huffman, ολοκληρώθηκε επιτυχώς.

Τέλος, υπολογίζουμε την εντροπία της πηγής, την απόδοση του κώδικα και τον πλεονασμό του.

Αποτελέσματα:

Εντροπία πηγής	`entropy = 3.2211`
Απόδοση του κώδικα %	`code_eff = 98.7110`
Πλεονασμός του κώδικα %	`code_red = 1.2890`

16) Δυαδική Διαμόρφωση PSK. Έστω η ακολουθία μηνύματος m=[1 1 0 0 0 1 1 0 1]. Έστω ότι κάθε παλμός – bit έχει διάρκεια T=1ms και το διάστημα δειγματοληψίας είναι dt=10-7 s. Έστω ότι το φέρον έχει συχνότητα fc=5000Hz. Διαμορφώστε το μήνυμα κατά δυαδικό PSK και στη συνέχεια φτιάξτε τη γραφική παράσταση του φάσματος του σήματος πληροφορίας και του διαμορφωμένου.

Κώδικας:

ask16.m

```
% m = bit sequence
m = [1 1 0 0 0 1 1 0 1];
n = length(m);

% Tb = Period of bit
Tb = 1e-3;

dt = 10e-7;
phase = [];
% msig = m bit sequence sampled with fs = 5000 Hz
msig = [];
for i = 1:1:n
    for j = dt:dt:Tb
        phase = [phase pi*m(i)];
        msig = [msig m(i)];
    end
end

% psk_sig = PSK modulated signal
fc = 5000;
Tw = Tb*n;
tm = dt:dt:Tw;
psk_sig = cos(2*pi*fc*tm + phase);
figure
plot(tm, psk_sig, 'b')
hold on
plot(tm, msig, 'r')
title('\color{blue}PSK modulated signal, \color{red}Bit sequence
')
xlabel('Time sec')
ylabel('Amplitude')

% f vector and BW calculation
BW=1/dt;
N=Tw/dt;
```

```
df=BW/N;
f=-BW/2:df:BW/2-df;


% F_psk = psk_sig Spectrum
F_psk=fftshift(fft(psk_sig))/N;
figure
plot(f, abs(F_psk))
title('PSK modulated signal Spectrum');
xlabel('Frequency Hz')
ylabel('PSD')

% F_msig = msig Spectrum
F_msig=fftshift(fft(msig))/N;
figure
plot(f, abs(F_msig))
title('Bit sequence signal Spectrum');
xlabel('Frequency Hz')
ylabel('PSD')
```

Αρχικά αποθηκεύουμε την ακολουθία από bits στο διάνυσμα m. Έπειτα υπολογίζουμε το πλήθος των bits, `n = 9`, στην δυαδική ακολουθία `m`.
Με τον παρακάτω κώδικα υπολογίζουμε τις τιμές της φάσης και τις αποθηκεύουμε στο διάνυσμα `phase`.

```
for i = 1:1:n
    for j = dt:dt:Tb
        phase = [phase pi*m(i)];
        msig = [msig m(i)];
    end
end
```

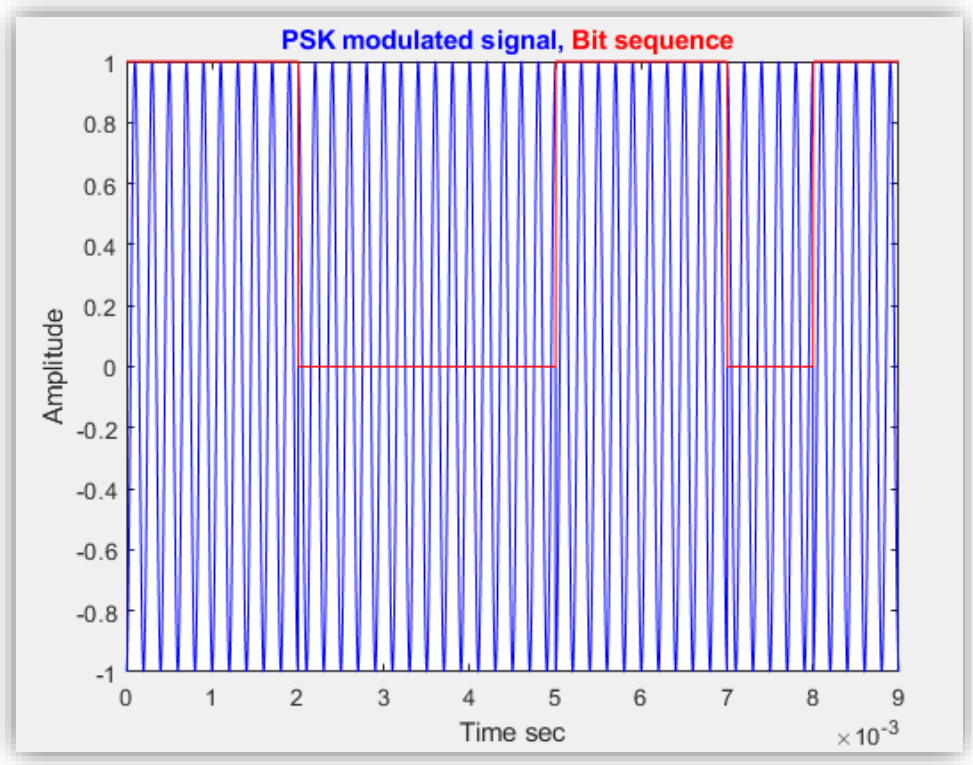
Για κάθε bit με τιμή ίση με 0 η τιμή της φάσης του θα είναι και αυτή ίση με 0. Για τα bit με τιμή 1 η τιμή της φάσης θα είναι ίση με π. Το σήμα `msig` είναι η ακολουθία `m` δειγματοληπτιμένη με συχνότητα ίση με `fc = 5000 Hz`. Δηλαδή κάθε περίοδος `Tb` της ακολουθίας `m`, δειγματοληπτείται κατά `dt = 1/Tc`, όπου `Tc = 1/fc`.
Ύστερα υπολογίζουμε το διαμορφωμένο κατά PSK σήμα, `psk_sig`.

```
psk_sig = cos(2*pi*fc*tm + phase);
```

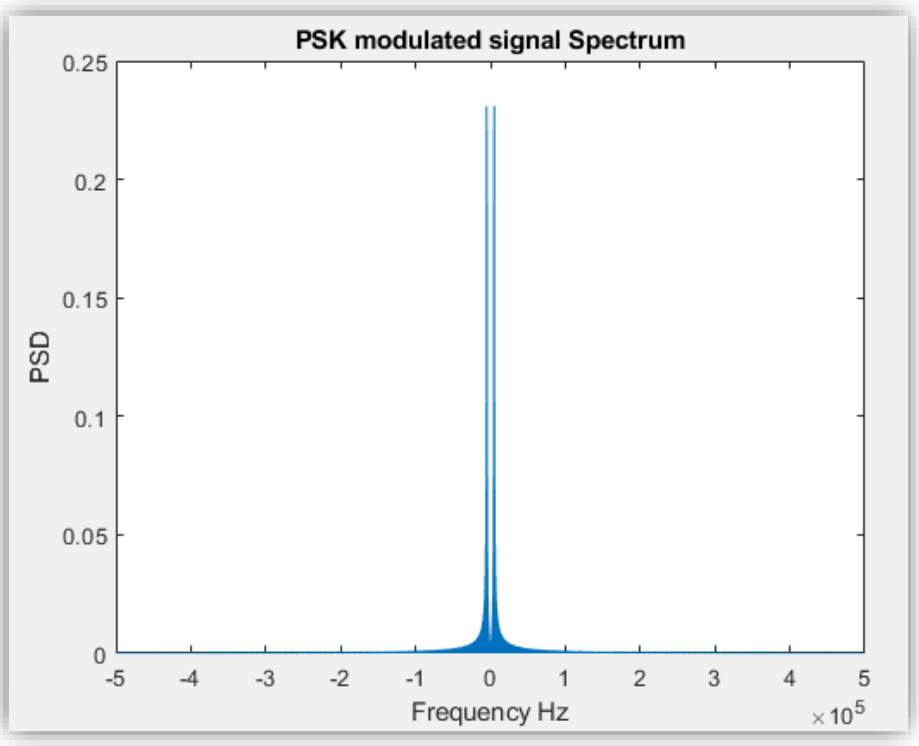
Κάθε τιμή του διανύσματος `psk_sig` προκύπτει από την τιμή του παραπάνω συνημίτονου στην αντίστοιχη χρονική στιγμή του διανύσματος `tm` και στην αντίστοιχη τιμή φάσης του διανύσματος `phase`.

Στην συνέχεια υπολογίζουμε τα φάσμα `F_psk` του διαμορφωμένου σήματος `psk_sig`. Τέλος υπολογίζουμε το φάσμα `F_msig` του δειγματοληπτιμένου σήματος `msig` της bit ακολουθίας.

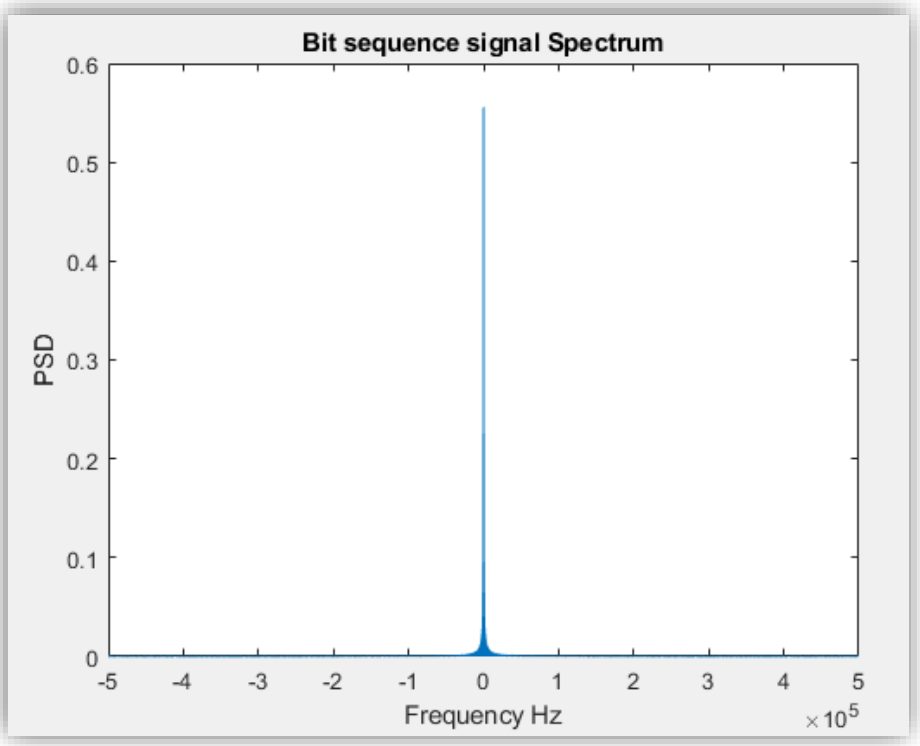
Αποτελέσματα:



Στο παραπάνω Plot φαίνεται το σήμα `msig` της ακολουθίας από bit με κόκκινο χρώμα. Επίσης, φαίνεται με μπλε χρώμα το διαμορφωμένο σήμα κατά PSK.
Παρατηρούμε ότι εκεί που εναλλάσσεται η τιμή των bits στην ακολουθία, εναλλάσσεται και η φάση του συνημίτονου του σήματος `psk_sig`. Συγκεκριμένα η φάση εναλλάσσεται στις χρονικές στιγμές `2*Tb, 5*Tb, 7*Tb` και `8*Tb`.



Στο παραπάνω Plot φαίνεται το φάσμα του διαμορφωμένου σήματος `psk_sig`.



Στο παραπάνω Plot φαίνεται το φάσμα του δειγματοληπτιμένου σήματος `msig` της bit ακολουθίας `m`.

17) Εκτελέστε τον ακόλουθο κώδικα Matlab για δημιουργία και γραφική απεικόνιση δεδομένων διαμορφωμένων κατά 16QAM. Εξηγείστε αναλυτικά τι κάνει η κάθε εντολή. Γράψτε τον αντίστοιχο κώδικα για a) 16QAM και SNR=5dB b)4 QAM και 20dB. Χρησιμοποιείστε τη συνάρτηση scatterplot() εναλλακτικά για την παραγωγή των αστερισμών των σημάτων.

Κώδικας:

ask17.m

```
M =16;
data = randi([0 M-1],1000,1);
sym = qammod(data,M);
rcv = awgn(sym, 5);
scatter(real(rcv),imag(rcv),'.')
hold on
plot(real(sym),imag(sym),'r*')
```

Αρχικά, αποθηκεύουμε στο διάνυσμα στήλη, `data`, 1000 τυχαίους αριθμούς από τον αριθμό 0 μέχρι το `M-1`.

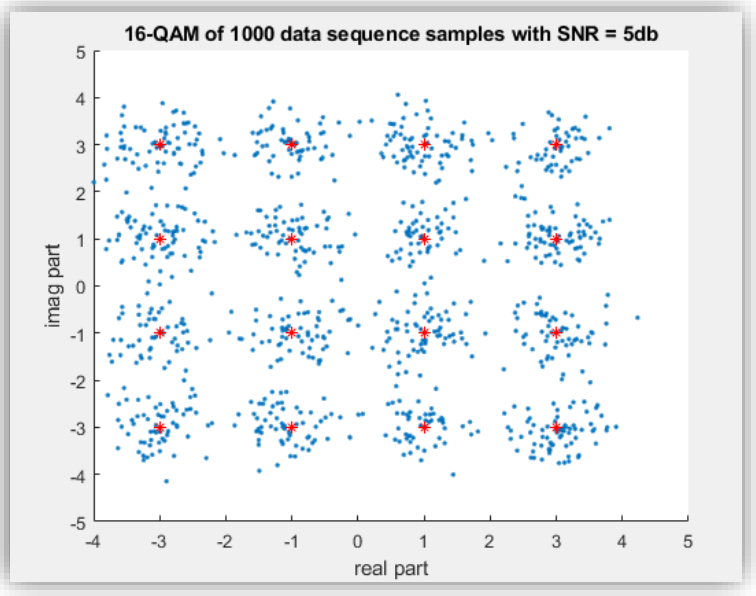
Ύστερα, εκτελούμε διαμόρφωση πλάτους τετραγώνου (QAM) με την χρήση της συνάρτησης `qammod()`. Η συνάρτηση δέχεται παράμετρο το διάνυσμα `data` και το πλήθος `M` των τιμών του συνόλου του. Η επιστρεφόμενη τιμή είναι ένα διάνυσμα το οποίο περιέχει την δυαδική ακολουθία των 1000 τιμών του διανύσματος `data`. Κάθε δείγμα του διανύσματος `data` αντιστοιχεί σε μία κωδική λέξη από το σύνολο των `M` κωδικών λέξεων. Τα κωδικοποιημένα δείγματα, αποθηκεύονται στο διάνυσμα `sym`.

Ακολούθως, προσθέτουμε λευκό Gaussian θόρυβο λόγου `SNR = 5db` στην ακολουθία, `sym`, των κωδικών λέξεων.

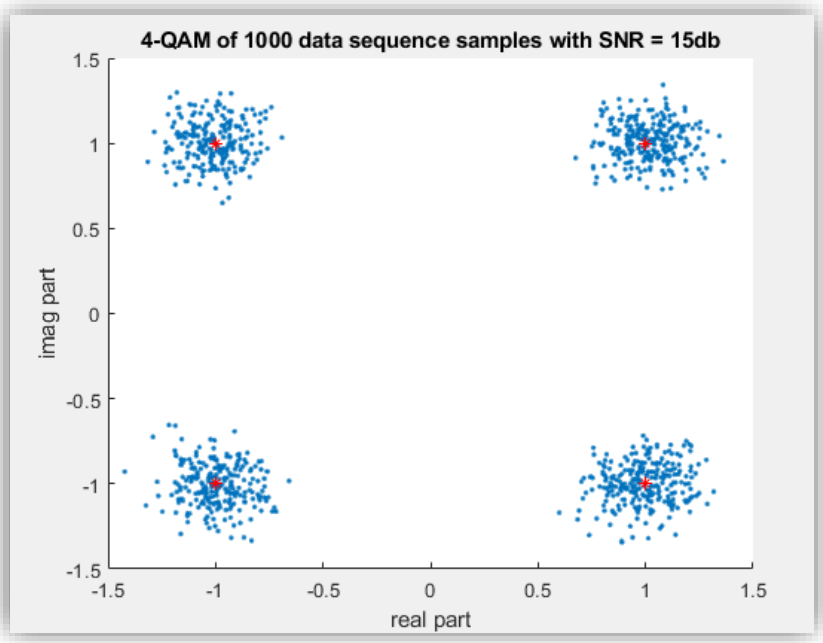
Ύστερα με την εντολή scatter αναπαριστούμε τις τιμές της ενθόρυβης ακολουθίας δυαδικών λέξεων, `rcv`, υπό την μορφή σημείων με πολικές συντεταγμένες.

Στην συνέχεια με πάγωμα παραθύρου, εμφανίζουμε στο ίδιο γράφημα και τις `M` τιμές των δυαδικών λέξεων με τις οποίος κωδικοποιήθηκε η ακολουθία δειγμάτων `data`.

Αποτελέσματα:



Στο παραπάνω plot φαίνονται με κόκκινο χρώμα οι `M = 16` δυαδικές λέξεις με τις οποίες κωδικοποιήθηκαν τα 1000 δείγματα της ακολουθίας δεδομένων `data`. Η αναπαράσταση των σημείων έχει γίνει με βάση της πολικές τους συντεταγμένες. Η κωδικοποίηση κατά φάση και πλάτος στην QAM γίνεται βάση της τριγωνομετρικής αναπαράσταση των τιμών στο επίπεδο.



Στο παραπάνω γράφημα ακολουθεί το αποτέλεσμα της διαμόρφωσης QAM με πλήθος δυαδικών λέξεων `M = 4`, για ακολουθία 1000 ενθόρυβων δειγμάτων λόγου `SNR = 25db`.

18) Να γραφτεί κώδικας για τον υπολογισμό της Εντροπίας ενός αγγλικού κειμένου

Υπόδειξη: μπορείτε να εντοπίσετε τα διαφορετικά γράμματα που εμφανίζονται στο κείμενο και στη συνέχεια να υπολογίσετε την πιθανότητα εμφάνισής τους (συχνότητα εμφάνισης του χαρακτήρα/ σύνολο χαρακτήρων στο κείμενο). Στη συνέχεια υπολογίζετε την εντροπία με βάση τον ορισμό. (Για λόγους απλότητας μπορείτε να θεωρήσετε ότι το κείμενο αποτελείται μόνο από γράμματα και κενά. Επίσης, μπορείτε να θεωρήσετε ένα μικρό κείμενο και να το αποθηκεύσετε σε μία μεταβλητή π.χ. `text='This is a short text';`) Ποιες είναι οι πιθανότητες εμφάνισης των χαρακτήρων του κειμένου?

Κώδικας:

ask18.m

```
str=' Missiles are thus also called guided missiles or guided
rockets (when in rocket form). Missiles have five system
components: targeting, guidance system, flight system, engine
and warhead.';
message=str;
syms=[];

prob=[];
n=length(message);
i=1;
while ~isempty(message)
    [char,no_occurrances]=mode(message);
    syms(i)=char;
    prob(i)=no_occurrances;
    indices=find(message==char);
    message(indices)='';
    i=i+1;
end
prob=prob/n;
disp('probabilities: ')
display(transpose(prob))
s = sum(prob);
disp(['pob sum: ' num2str(s)])


% Source entropy calculation
Hi=prob.*log2(1./(prob));
entropy=sum(Hi)
```

Αρχικά, αποθηκεύουμε το κείμενο στο διάνυσμα str. Αντιγράφουμε το περιεχόμενο του str στο διάνυσμα message.

Υπολογίζουμε το πλήθος n των γραμμάτων του κειμένου message.
Βρίσκουμε το αλφάβητο syms με τον κώδικα του βρόγχου while (ο κώδικας επεξηγείται στην λύση της άσκ. 16), καθώς και την πιθανότητα εμφάνισης κάθε συμβόλου. Η τιμές της πιθανότητας εμφάνισης αποθηκεύονται στο διάνυσμα prob.
Τέλος υπολογίζουμε την εντροπία κάθε συμβόλου του αλφαβήτου και
ύστερα την συνολική εντροπία του κειμένου str.

Αποτελέσματα:

```
probabilities:
    0.1436
    0.1117
    0.1011
    0.0745
    0.0532
    0.0479
    0.0479
    0.0426
    0.0372
    0.0372
    0.0372
    0.0372
    0.0319
    0.0266
    0.0266
    0.0213
    0.0160
    0.0160
    0.0160
    0.0106
    0.0106
    0.0106
    0.0106
    0.0106
    0.0053
    0.0053
    0.0053
    0.0053

pob sum: 1
```

Ο πίνακας των πιθανοτήτων περιέχει όλες τις πιθανότητες των συμβόλων του αλφαβήτου για το κείμενο πληροφορίας str.

Το άθροισμα των πιθανοτήτων είναι ίσο με 1, επομένως οι πιθανότητες των συμβόλων έχουν υπολογιστεί σωστά.

Επίσης το άθροισμα των επιμέρους εντροπιών των συμβόλων είναι ίσο με την παρακάτω τιμή.

```
entropy =

    4.2646
```

(Συνολική εντροπή κειμένου)

19) Δίνεται μία DMS πηγή με πιθανότητες εκπεμπόμενων συμβόλων:
0.2, 0.05, 0.03, 0.1, 0.3, 0.02, 0.22, 0.08.
Να εφαρμόσετε κωδικοποίηση πηγής κατά Huffman και στη συνέχεια να δημιουργηθεί μία ακολουθία 200 συμβόλων της πηγής. Η ακολουθία να κωδικοποιηθεί κατά Huffman. Το πρόγραμμα να εμφανίζει για κάθε ένα σύμβολο της πηγής (x1,x2,….) την πιθανότητα εμφάνισής του και την κωδική λέξη που του αντιστοιχίζεται, η εκτύπωση θα είναι ως εξής:

```
x1 probability : 0.3 codeword :--->
 0 0
x2 probability : 0.25 codeword :--->
 0 1
.....
```

Στη συνέχεια να γίνεται υπολογισμός της εντροπίας της πηγής και του μέσου μήκος κωδικής λέξης. Να υπολογίζεται επίσης, η απόδοση και ο πλεονασμός του κώδικα.

Κώδικας:

ask19.m

```
prob=[0.2 0.05 0.03 0.1 0.3 0.02 0.22 0.08];
syms = ['k' 'g' 'h' 'd' 'v' 'e' 't' 'q'];
n = length(syms);
seq = [];

% Calculating the Symbol Sequence from symbol probabilities
for i = 1:1:n
    t = prob(i)*200;
    for j = 1:1:t
        seq = [seq syms(i)];
    end
end
u = uint8(syms);

% Huffman Encoding using the huffman dictionary
[dict,~]= huffmandict(u, prob);

for i = 1:1:n
    bin = huffmanenco(syms(i), dict);
    bin = transpose(bin);
    output = ['x', num2str(i), ' probability: ',
num2str(prob(i)), '     codeword:--->'];
    disp(output)
    disp(num2str(bin))
end
```

Για αρχή, αποθηκεύουμε τις δοθείσες τιμές των πιθανοτήτων στο διάνυσμα prob. Αρχικοποιούμε το διάνυσμα sym με τα σύμβολα του αλφαβήτου της πηγής.
Έπειτα, υπολογίζουμε τις εμφανίσεις t των συμβόλων του αλφαβήτου στην ακολουθία των 200 συμβόλων μέσω της πιθανότητας εμφάνισής του.

```
t = prob(i)*200
```

Με τους παρακάτω επαναληπτικούς βρόγχους προσθέτουμε στην ακολουθία συμβόλων, t φορές το κάθε ένα σύμβολο syms(i) από τα n σύμβολα.

```
for i = 1:1:n
    t = prob(i)*200;
    for j = 1:1:t
        seq = [seq syms(i)];
    end
end
```

Ύστερα, αποθηκεύουμε στο διάνυσμα u, την ascii τιμή του καθενός συμβόλου του διανύσματος syms. Υπολογίζουμε τις κωδικές λέξεις του αλφαβήτου κατά Huffman και τις αποθηκεύουμε στο διάνυσμα dict. Ύστερα, έχοντας το δυαδικό αλφάβητο dict, υπολογίζουμε την δυαδική κωδική λέξη bin, του ακολουθίας συμβόλων syms.

```
bin = huffmanenco(syms(i), dict);
```

Έπειτα τυπώνουμε την δυαδική λέξη bin και την πιθανότητα εμφάνισης prob(i), του κάθε συμβόλου από το αλφάβητο syms.

<u>Αποτελέσματα:</u>

```
>> ask19
x1 probability: 0.2     codeword:--->
1  1
x2 probability: 0.05     codeword:--->
0  1  0  0  0
x3 probability: 0.03     codeword:--->
0  1  0  0  1  0
x4 probability: 0.1     codeword:--->
0  1  1
x5 probability: 0.3     codeword:--->
0  0
x6 probability: 0.02     codeword:--->
0  1  0  0  1  1
x7 probability: 0.22     codeword:--->
1  0
x8 probability: 0.08     codeword:--->
0  1  0  1
```

Στο παραπάνω Plot φαίνονται οι κωδικές λέξεις, του αλφαβήτου syms και η πιθανότητα εμφάνισής τους. Παρατηρούμε ότι όσο πιο μικρή είναι η πιθανότητα εμφάνισης, τόσο περισσότερα bits αποδίδονται στην κωδική λέξη του συμβόλου.

20) Οι κώδικες Hamming είναι γραμμικοί κώδικες block με n=2m-1, k=2m-1-m και ελάχιστη απόσταση d_{min}=3 και που έχουν έναν πολύ απλό πίνακα ελέγχου ισοτιμίας. Ο πίνακας ελέγχου ισοτιμίας ο οποίος είναι ένας πίνακας m×(2^m-1) πίνακας, έχει σαν στήλες όλες τις δυαδικές ακολουθίες με μήκος m, εκτός από τη μηδενική ακολουθία.

Να δημιουργηθεί πρόγραμμα για τον κώδικά Hamming (15,11). Για το σκοπό αυτό να χρησιμοποιηθεί η εντολή [h,g,n,k] = hammgen().
Πόσα bits έχει η κωδική λέξη και πόσα bits έχει το μήνυμα πληροφορίας για το συγκεκριμένο κώδικα;

Ποιος είναι ο πίνακας ελέγχου ισοτιμίας και ποιος ο γεννήτορας? Συμφωνεί η μορφή του πίνακα ελέγχου ισοτιμίας με τις θεωρητικές προδιαγραφές?
Να κωδικοποιηθεί το μήνυμα

[1 0 0 1 0 1 1 1 1 1 0]

με κατάλληλη σύνταξη της εντολής encode. Ποια είναι η κωδική λέξη που προκύπτει?

Κώδικας:

ask20.m

```
[H,G,n,k] = hammgen(4);
G = wshift(2,G,[0 -k])
H = wshift(2,H,[0 -k])

% n = codeword bits= 15 bits
% k = message bits = 11 bits
% n - k = check bits = 4 bits

M = [1 0 0 1 0 1 1 1 1 1 0];
C = encode(M,n,k);
C = wshift(2,C,[0 -k])

Ht = transpose(H);
error = mod(C*Ht, 2);
error
```

Με την χρήση της συνάρτησης hammgen() δημιουργούμε τον κώδικα Hamming. Η συνάρτηση δέχεται ως παράμετρο, το πλήθος n-k = 4 των bit ελέγχου. Όπου n=15 είναι το συνολικό πλήθος bits της κάθε κωδικής λέξης και k = 11 το πλήθος bit του δυαδικού μηνύματος.

```
[H,G,n,k] = hammgen(4);
```

Η συνάρτηση επιστρέφει τον γεννήτορα πίνακα G και τον πίνακα ελέγχου ισοτιμίας H. Η μορφή των πινάκων δεν είναι ίδια με την θεωρητική. Συγκεκριμένα ο επιμέρους μοναδιαίος πίνακας και στους δύο πίνακς G, H είναι στην αντίθετη θέση.

```
G = wshift(2,G,[0 -k])
H = wshift(2,H,[0 -k])
```

Με τις παραπάνω εντολές, μετατοπίζουμε τις γραμμές τους κατά k θέσεις αριστερά με σκοπό να έχουν ίδια μορφή με την θεωρητική.

Έπειτα, κωδικοποιούμε την παρακάτω ακολουθία M με την συνάρτηση encode().

```
M = [1 0 0 1 0 1 1 1 1 1 0];
C = encode(M,n,k);
```

Η δυαδική λέξη που επιστρέφει η συνάρτηση αποθηκεύεται στο διάνυσμα C.

Τέλος με την παρακάτω εντολή ελέγχομε για τυχόν σφάλματα.

```
error = mod(C*Ht, 2);
```

Η συνάρτηση mod() εκτελεί πολλαπλασιασμό πινάκων μεταξύ 2 δυαδικών πινάκων. Στον πολλαπλασιασμό πινάκων η πρόσθεση είναι η λογική πράξη AND και ο πολλαπλασιασμός η XOR. Όπου C είναι το διάνυσμα της κωδικής λέξης και Ht αντίστροφος πίνακας του πίνακα ελέγχου ισοτιμίας H.

Αποτελέσματα:

```
G =

     1     0     0     0     0     0     0     0     0     0     0     1     1     0     0
     0     1     0     0     0     0     0     0     0     0     0     0     1     1     0
     0     0     1     0     0     0     0     0     0     0     0     0     0     1     1
     0     0     0     1     0     0     0     0     0     0     0     1     1     0     1
     0     0     0     0     1     0     0     0     0     0     0     1     0     1     0
     0     0     0     0     0     1     0     0     0     0     0     0     1     0     1
     0     0     0     0     0     0     1     0     0     0     0     1     1     1     0
     0     0     0     0     0     0     0     1     0     0     0     0     1     1     1
     0     0     0     0     0     0     0     0     1     0     0     1     1     1     1
     0     0     0     0     0     0     0     0     0     1     0     1     0     1     1
     0     0     0     0     0     0     0     0     0     0     1     1     0     0     1
```

Παραπάνω φαίνεται ο γεννήτορα πίνακας G, στον οποίο τα n-k = 4 δεξιότερα bits κάθε γραμμής είναι τα bit ελέγχου. Τα υπόλοιπα bits συνθέτουν τον μοναδιαίο πίνακα I.

```
H =

     1     0     0     1     1     0     1     0     1     1     1     1     0     0     0
     1     1     0     1     0     1     1     1     1     0     0     0     1     0     0
     0     1     1     0     1     0     1     1     1     1     0     0     0     1     0
     0     0     1     1     0     1     0     1     1     1     1     0     0     0     1
```

Παραπάνω φαίνεται ο πίνακας ελέγχου ισοτιμίας Η, στον οποίο τα k = 11 αριστερότερα bits κάθε γραμμής είναι τα bit ελέγχου ισοτιμίας για το πληροφοριακό μέρος του μηνύματος. Τα υπόλοιπα bits συνθέτουν τον μοναδιαίο πίνακα Ι.

```
C =

     1     0     0     1     0     1     1     1     1     1     0     1     0     0     1


error =

     0     0     0     0
```

Παραπάνω βλέπουμε την κωδική λέξη που παράγεται από την ακολουθία πληροφορίας Μ. Δεν υπάρχει κανένα σφάλμα στην λήψη της κωδικής λέξης.

bpfilt.m

```
function y = bpfilt(signal, f1, f2,fs)
%% Bandpass filtering
%
% Syntax:
% y = bpfilt(signal, f1, f2, fs)
%
% Description:
% This function performs bandpass filtering of a time series
% with rectangle window.
%
% Input Arguments:
% signal - a column vector of time series.
% f1 - the lower bound of frequencies (in Hz).
% f2 - the upper bound of frequencies (in Hz).
%
% Options:
% fs - the sampling frequency in Hz.
%
% Output Arguments:
% y - the filtered time series.
%
if isrow(signal)
 signal = signal';
end
N = length(signal);
dF = fs/N;
f = (-fs/2:dF:fs/2-dF)';
%% Band-Pass Filter:
if isempty(f1) || f1==-Inf
 BPF = (abs(f) < f2);
elseif isempty(f2) || f2==Inf
 BPF = (f1 < abs(f));
else
 BPF = ((f1 < abs(f)) & (abs(f) < f2));
end
%% Power spectrum of the original signal
%signal = signal-mean(signal);
spektrum = fftshift(fft(signal))/N;
%% Power spectrum of the band-pass filtered signal
spektrum = BPF.*spektrum;
%% The band-pass filtered time series
y = ifft(ifftshift(spektrum))*N; %inverse ifft
y = real(y);
```

butterworth_filter.m

```
function [B, C]=butterworth_filter(A, dt,N,fc,f0);
% B=butterworth_filter(A, dt,N,fc,f0);
% B: output of the filter
% A:  input signal to be filtered
% dt:  smaller time particle
% N: filter order (1-12) the higher the order the more ideal the
filter
% fc: cutoff frequency
% f0:  central frequency of the filter

samples=length(A);
j=-(samples/2):1:(samples/2)-1;
f=j*(1/samples);
f=f/dt;
Hpar=1+i*(((f-f0)/fc).^N);
H=Hpar.^(-1);
A=fft(A);
A=fftshift(A);
A=H.*A;
C=A;%%%%%%%%%%%%%%%
A=ifftshift(A);
A=ifft(A);
B=A;
```