Ερωτήματα

**1) Ημιτονοειδές σήμα.** Να φτιαχτεί συνάρτηση που να δημιουργεί ημιτονοειδές σήμα. Ορίσματα της συνάρτησης να θεωρούνται η συχνότητα του σήματος, η συχνότητα δειγματοληψίας του, το πλάτος του, η φάση του και η χρονική του διάρκεια. Εξοδοι της συνάρτησης να είναι το ημιτονοειδές σήμα και ο χρονικός άξονας στον οποίο ορίστηκε, π.χ. *[x,t]=mine\_sin(f0,fs,A,ph,T1).*

f0 = 5;

fs = 1000;

A = 2;

ph = pi/2;

T1 = 3;

[x,t] = mine\_sin(f0,fs,A,ph,T1);

function[x,t] = mine\_sin(f0,fs,A,ph,T1)

t = 0:1/fs:T1-1/fs;

x = A \* sin(2 \* pi \*f0 \* t+ph);

%Plot the signal and time axis

plot(t,x);

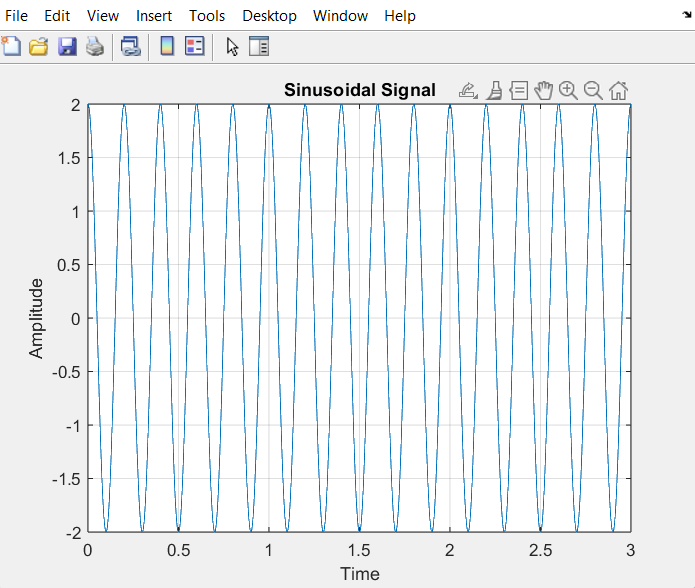
xlabel("Time");

ylabel("Amplitude");

title("Sinusoidal Signal");

grid on;

end



**2) Συνάρτηση φάσματος.** 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.

function num = ask2(f,T) %Η περίοδος δίνεται σε ms

dt = (T / 100) \* exp(1) - 3; %Δειγματοληψία στο χρόνο

Tw = T \* exp(1) - 3; %Χρονική διάρκεια

t = dt:dt:Tw;

y = sin(2 \* pi \* f \* t);

N = length(y);

y = fft(y);

num = fftshift(y)/N

end

**3) Φάσμα ημιτονοειδούς σήματος.** Χρησιμοποιώντας τις συναρτήσεις των ερωτημάτων 1) και 2) να δημιουργηθεί ημίτονο συχνότητας 10 Hz με συχνότητα δειγματοληψίας 1KHz, το πλάτος και η φάση του ημιτόνου να είναι 5 Volt και 00 αντίστοιχα και η χρονική διάρκεια 0.5 sec. Να γίνει η γραφική παράσταση του ημιτόνου, να βρεθεί το φάσμα του και να γίνει και η γραφική παράσταση του φάσματος πλάτους. Συγκρίνετε το φάσμα πλάτους που απεικονίζεται στη γραφική παράσταση με τη θεωρητική έκφραση του φάσματος πλάτους του ημιτόνου.

function ask3()

% Define the signal parameters

amplitude = 5; % Amplitude of the sine wave (in volts)

frequency = 10; % Frequency of the sine wave (in Hz)

duration = 0.5; % Duration of the signal (in seconds)

fs = 1000; % Sampling frequency (in Hz)

% Generate the time vector

t = 0:1/fs:duration-1/fs;

% Generate the sinusoidal signal

phase = 0; % Phase of the sine wave (in radians)

signal = amplitude \* sin(2\*pi\*frequency\*t + phase);

% Plot the sine wave

figure;

plot(t, signal);

xlabel('Time (s)');

ylabel('Amplitude');

title('Sine Wave');

% Call the function to calculate the amplitude spectrum

period = 1/frequency;

spectrum = calculateAmplitudeSpectrum(signal, period);

% Calculate the theoretical amplitude spectrum of the sine wave

theoretical\_spectrum = amplitude/2 \* ones(size(spectrum));

% Plot the amplitude spectrum

figure;

plot(frequency, spectrum);

hold on;

plot(frequency, theoretical\_spectrum, 'r--');

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title('Amplitude Spectrum');

legend('Measured Spectrum', 'Theoretical Spectrum');

end

function spectrum = calculateAmplitudeSpectrum(signal, period)

% Calculate the length of the signal

N = length(signal);

% Apply the FFT to the signal

signal\_fft = fft(signal);

% Calculate the single-sided amplitude spectrum

amplitude\_spectrum = abs(signal\_fft(1:N/2+1));

% Normalize the spectrum by dividing by the length of the signal

amplitude\_spectrum = amplitude\_spectrum/N;

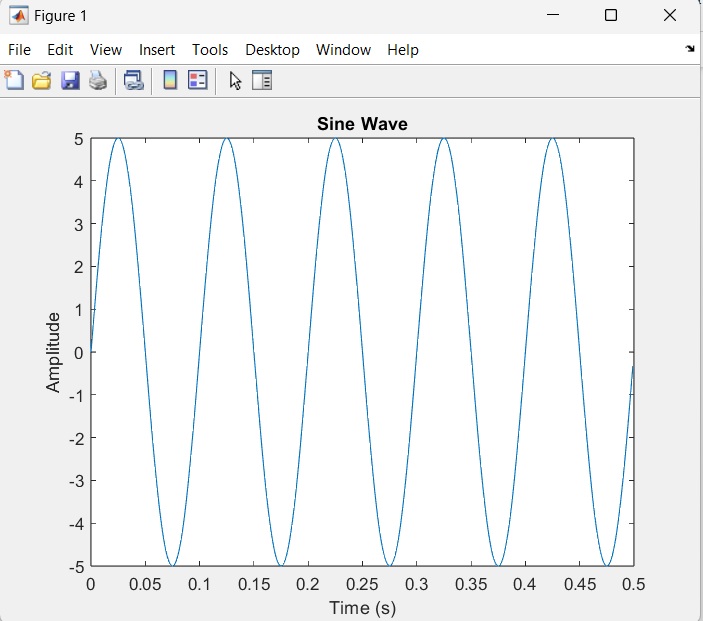
% Generate the frequency axis for the spectrum

frequency = (0:(N/2))/period;

% Return the amplitude spectrum

spectrum = amplitude\_spectrum;

end



**4) Φάσμα σήματος ομιλίας**. Κατεβάστε και αποθηκεύστε το σήμα ομιλίας 3WORDS.WAV που θα βρείτε στο https://booksite.elsevier.com/9780080993881/ (κατάλογος data). Διαβάστε το σήμα με τη συνάρτηση audioread και αποθηκεύστε το σε ένα διάνυσμα. Ακούστε το σήμα με χρήση της συνάρτησης sound. Να γίνει η γραφική παράσταση της κυματομορφής του σήματος συναρτήσει του χρόνου, και να αναπαρασταθεί το φάσμα πλάτους του σήματος.

Kάνω 2 αρχεία πρώτα το calculateAmplitudeSpectrum.m

function spectrum = calculateAmplitudeSpectrum(signal, period)

% Calculate the length of the signal

N = length(signal);

% Apply the FFT to the signal

signal\_fft = fft(signal);

% Calculate the single-sided amplitude spectrum

amplitude\_spectrum = abs(signal\_fft(1:N/2+1));

% Normalize the spectrum by dividing by the length of the signal

amplitude\_spectrum = amplitude\_spectrum/N;

% Return the amplitude spectrum

spectrum = amplitude\_spectrum;

end

Και μετά το ask4.m

% File path or filename of the speech signal

file\_path = 'C:\Users\arist\Downloads\data\3WORDS.WAV';

% Read the speech signal

[speech\_signal, sample\_rate] = audioread(file\_path);

% Listen to the speech signal

sound(speech\_signal, sample\_rate);

% Plot the waveform of the signal as a function of time

time = (0:length(speech\_signal)-1)/sample\_rate;

figure;

plot(time, speech\_signal);

xlabel('Time (s)');

ylabel('Amplitude');

title('Speech Signal Waveform');

% Calculate the amplitude spectrum of the signal

period = 1/sample\_rate;

spectrum = calculateAmplitudeSpectrum(speech\_signal, period);

% Plot the amplitude spectrum

frequency = (0:(length(spectrum)-1))\*(1/period)/length(spectrum);

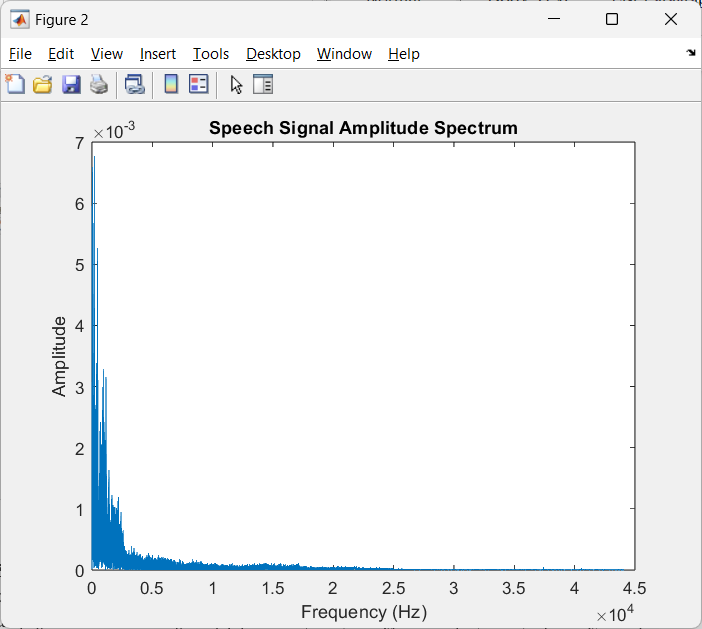
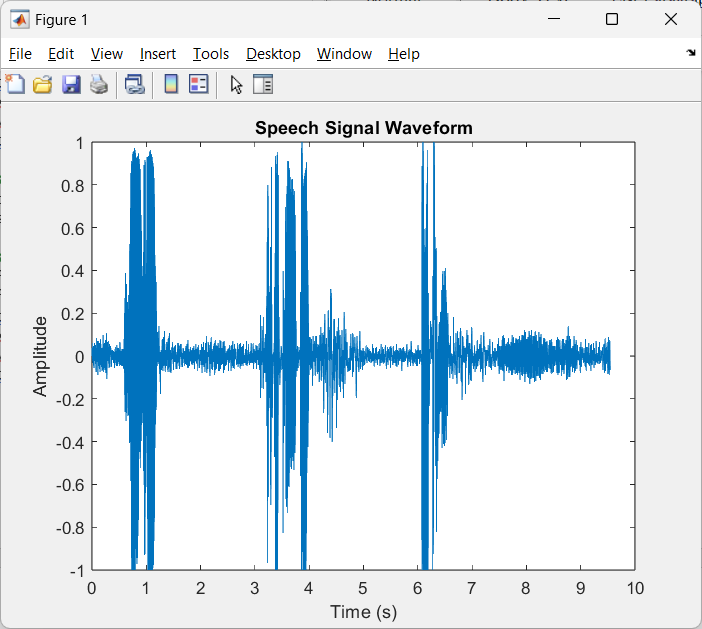
figure;

plot(frequency, spectrum);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title('Speech Signal Amplitude Spectrum');



To ηχητικό αποτέλεσμα είναι ΔΕΝΤΡΟ ΑΥΤΟΚΙΝΗΤΟ ΠΑΤΩΜΑ

**5) Φάσμα ενθόρυβου σήματος.** Να δημιουργηθεί σήμα πληροφορίας που να αποτελείται από την υπέρθεση συνημιτονοειδών σημάτων από 500Hz εως 3000Hz, με βήμα 100Hz. Να δημιουργηθεί ο άξονας του χρόνου ορίζοντας τη στοιχειώδη χρονική μεταβολή, dt, βάση της πιο γρήγορης συνιστώσας και τη 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 10 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 όλων των σημάτων.

% Signal parameters

f\_start = 500; % Starting frequency

f\_end = 3000; % Ending frequency

f\_step = 100; % Frequency step

% Time parameters

f\_max = f\_end; % Fastest component frequency

T\_min = 1/f\_max; % Minimum period

duration = 10\*T\_min; % Duration of observation (10 times the minimum period)

dt = 1/(10\*f\_end); % Elementary time variation (based on fastest component)

% Time axis

t = 0:dt:duration;

% Information signal generation

info\_signal = zeros(size(t));

frequencies = f\_start:f\_step:f\_end;

for f = frequencies

info\_signal = info\_signal + cos(2\*pi\*f\*t);

end

% Plot the information signal

figure;

plot(t, info\_signal);

xlabel('Time (s)');

ylabel('Amplitude');

title('Information Signal');

ylim([-length(frequencies), length(frequencies)]);

% Listen to the information signal

sound(info\_signal);

% Add white Gaussian noise to the signal

SNR\_db = 5; % Signal-to-noise ratio (in dB)

noisy\_signal = awgn(info\_signal, SNR\_db, 'measured');

% Plot the noisy signal

figure;

plot(t, noisy\_signal);

xlabel('Time (s)');

ylabel('Amplitude');

title('Noisy Signal');

ylim([-length(frequencies), length(frequencies)]);

% Listen to the noisy signal

sound(noisy\_signal);

% Plot the amplitude spectrum of the information signal

info\_spectrum = calculateAmplitudeSpectrum(info\_signal, dt);

frequency\_info = (0:(length(info\_spectrum)-1))/(duration);

figure;

plot(frequency\_info, info\_spectrum);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title('Amplitude Spectrum - Information Signal');

% Plot the amplitude spectrum of the noisy signal

noisy\_spectrum = calculateAmplitudeSpectrum(noisy\_signal, dt);

frequency\_noisy = (0:(length(noisy\_spectrum)-1))/(duration);

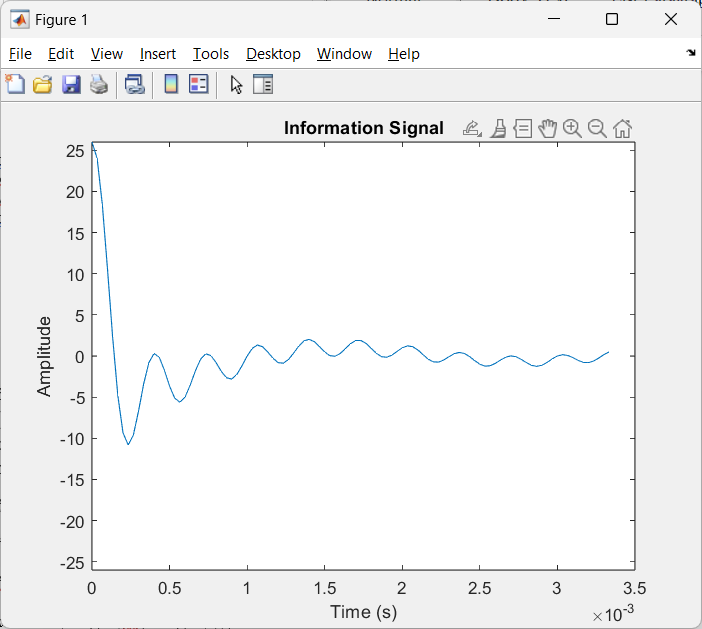
figure;

plot(frequency\_noisy, noisy\_spectrum);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title('Amplitude Spectrum - Noisy Signal');



**6) Φάσμα μελωδίας.** Δίνεται το ακόλουθο πρόγραμμα Matlab. Ακούστε το σήμα (xx(t)), σε ποιες μουσικές νότες αντιστοιχούν οι συχνότητες από τις οποίες αποτελείται και σε ποια γνωστή μελωδία αντιστοιχεί το σήμα? Φτιάξτε το φάσμα του σήματος μελωδίας.

*fs=6000;ts=1/fs;*

*t=0:ts:1;*

*fr=[1567 1760 1975 1046];*

*x=zeros(length(fr), length(t));*

*for i=1:length(fr)*

*x(i,:)=cos(2\*pi\*fr(i)\*t);*

*end*

*xx=[x(1,:) x(2,:) x(3,:) x(4,:)];*

*plot([0:ts:4+3\*ts],xx)*

*sound(xx,fs)*

fs = 6000;

ts = 1/fs;

t = 0:ts:1;

fr = [1567 1760 1975 1046];

x = zeros(length(fr), length(t));

for i = 1:length(fr)

x(i, :) = cos(2\*pi\*fr(i)\*t);

end

xx = [x(1, :) x(2, :) x(3, :) x(4, :)];

% Plot the melody signal

time = 0:ts:(length(xx)-1)\*ts;

figure;

plot(time, xx);

xlabel('Time (s)');

ylabel('Amplitude');

title('Melody Signal');

% Listen to the melody signal

sound(xx, fs);

% Calculate and plot the amplitude spectrum of the melody signal

spectrum = calculateAmplitudeSpectrum(xx, ts);

frequency = (0:(length(spectrum)-1))/(length(spectrum)\*ts);

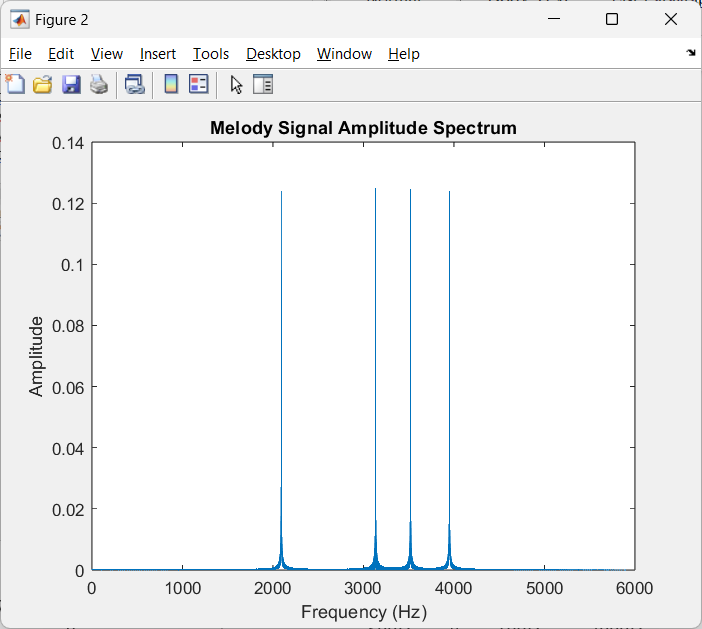
figure;

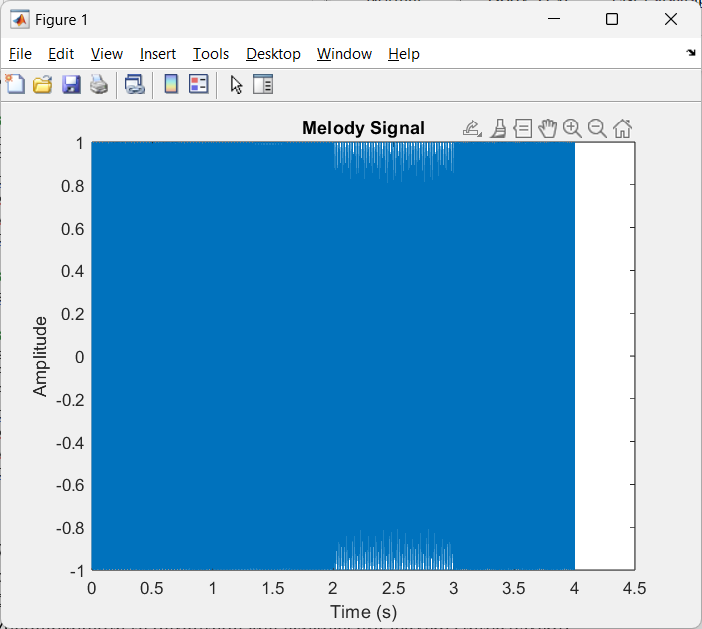
plot(frequency, spectrum);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title('Melody Signal Amplitude Spectrum');





**7) Διαμόρφωση DSB.** Να δημιουργηθεί σήμα πληροφορίας που να αποτελείται από την υπέρθεση συνημιτονοειδών σημάτων από 500Hz με βήμα 100Hz εως 3000Hz. Φέρον συχνότητας 100ΚΗz. 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.

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

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);

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

% Information signal parameters

info\_start\_freq = 500; % Starting frequency of the information signal (Hz)

info\_freq\_step = 100; % Frequency step of the information signal (Hz)

info\_end\_freq = 3000; % Ending frequency of the information signal (Hz)

% Carrier signal parameters

carrier\_freq = 100000; % Carrier frequency (Hz)

% Sampling frequency

fs = 10 \* carrier\_freq; % Choose a sampling frequency at least 10 times the carrier frequency

% Time axis

t = 0:(1/fs):1; % Time duration of 1 second

% Generate the information signal

info\_signal = zeros(1, length(t));

freqs = info\_start\_freq:info\_freq\_step:info\_end\_freq;

for freq = freqs

info\_signal = info\_signal + cos(2\*pi\*freq\*t);

end

% Perform DSB modulation

usb\_modulated\_signal = info\_signal .\* cos(2\*pi\*carrier\_freq\*t); % Upper sideband modulation

lsb\_modulated\_signal = info\_signal .\* cos(2\*pi\*carrier\_freq\*t + pi); % Lower sideband modulation

% Bandpass filter parameters

f1 = carrier\_freq - info\_end\_freq; % Lower bound of the bandpass filter

f2 = carrier\_freq + info\_end\_freq; % Upper bound of the bandpass filter

% Apply bandpass filtering to the USB modulated signal

usb\_filtered\_signal = bpfilt(usb\_modulated\_signal, f1, f2, fs);

% Apply bandpass filtering to the LSB modulated signal

lsb\_filtered\_signal = bpfilt(lsb\_modulated\_signal, f1, f2, fs);

% Plot the amplitude spectra of all signals

figure;

subplot(4,1,1);

plot\_spectrum(info\_signal, fs, 'Information Signal');

subplot(4,1,2);

plot\_spectrum(usb\_modulated\_signal, fs, 'USB Modulated Signal');

subplot(4,1,3);

plot\_spectrum(lsb\_modulated\_signal, fs, 'LSB Modulated Signal');

subplot(4,1,4);

plot\_spectrum(usb\_filtered\_signal, fs, 'USB Filtered Signal');

% Function for plotting the amplitude spectrum

function plot\_spectrum(signal, fs, title\_str)

spectrum = abs(fftshift(fft(signal)))/length(signal);

freq = (-fs/2):(fs/length(signal)):(fs/2 - fs/length(signal));

plot(freq, spectrum);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title(title\_str);

end

% Function for bandpass filtering

function y = bpfilt(signal, f1, f2, fs)

% bandpass filtering

if isrow(signal)

signal = signal.';

end

N = length(signal);

dF = fs / N;

f = (-fs/2 : dF : fs/2 - dF).';

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

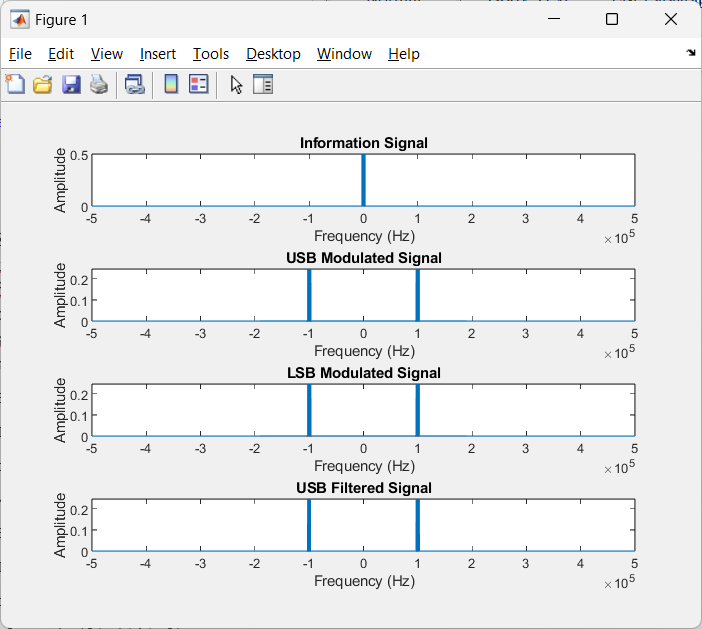
spectrum = fftshift(fft(signal)) / N;

spectrum = BPF .\* spectrum;

y = ifft(ifftshift(spectrum)) \* N;

y = real(y);

end



**8) Φαινόμενο Aliasing.** Aναπαράσταση 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.

F1= 20; %sampling rate

T1 = 1/F1; %sampling interval

t1 = 0:T1:2;

x1 = sin(2\*pi\*1\*t1);

F2= 1.5; %sampling rate

T2 = 1/F2; %sampling interval

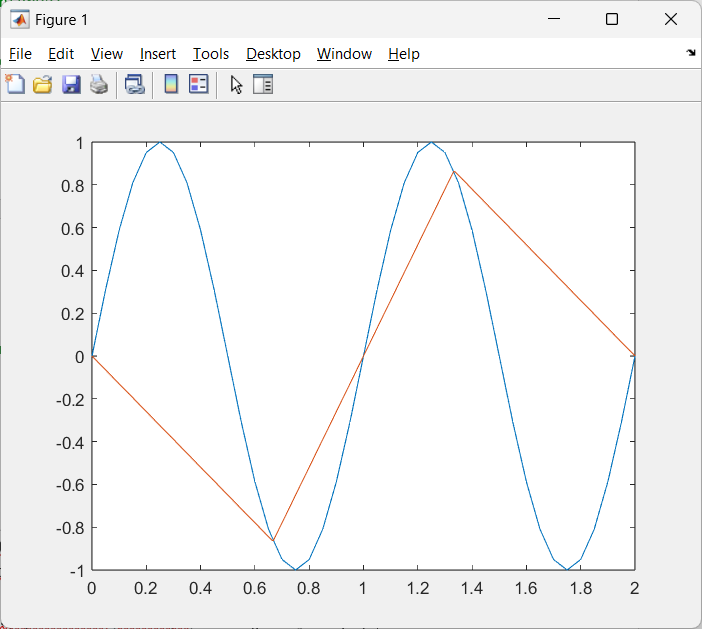
t2 = 0:T2:2;

x2 = sin(2\*pi\*1\*t2);

plot(t1,x1,t2,x2);

%sound(x1,F1);

%sound(x2,F2);



**9) Διαμόρφωση ΑΜ.** Δημιουργήστε ένα ΑΜ σήμα x1(t) με τις εξής παραμέτρους: Ac =1 Volt, Αm=0.5 Volt,fm=2 Hz, fc=100Hz, όταν το σήμα βασικής ζώνης είναι s(t)= Αm sin(2πfmt). Θεωρήστ 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 δειγγματοληψίας- θέσετε fs=400

% Signal parameters

Am = 0.5; % Amplitude of the baseband signal (V)

fm = 2; % Frequency of the baseband signal (Hz)

fc = 100; % Carrier frequency (Hz)

fs = 400; % Sampling frequency (Hz)

t = 0:0.001:1; % Time axis

% Generate the baseband signal

s = Am \* sin(2\*pi\*fm\*t);

% Generate the AM signal manually

x1 = (1 + s) .\* cos(2\*pi\*fc\*t);

% Modulate the baseband signal using ammod() function

x2 = ammod(s, fc, fs);

% Plot the modulated signal and its envelope

figure;

plot(t, x1, 'b', t, abs(x1), 'r');

xlabel('Time (s)');

ylabel('Amplitude');

title('AM Signal (x1(t)) and Envelope');

legend('x1(t)', 'Envelope');

% Plot the modulated signal obtained from ammod() function

figure;

plot(t, x2);

xlabel('Time (s)');

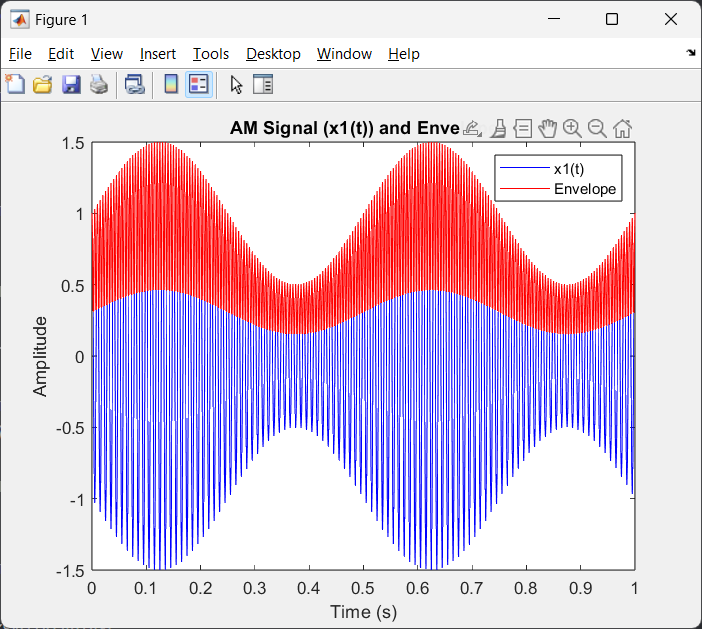
ylabel('Amplitude');

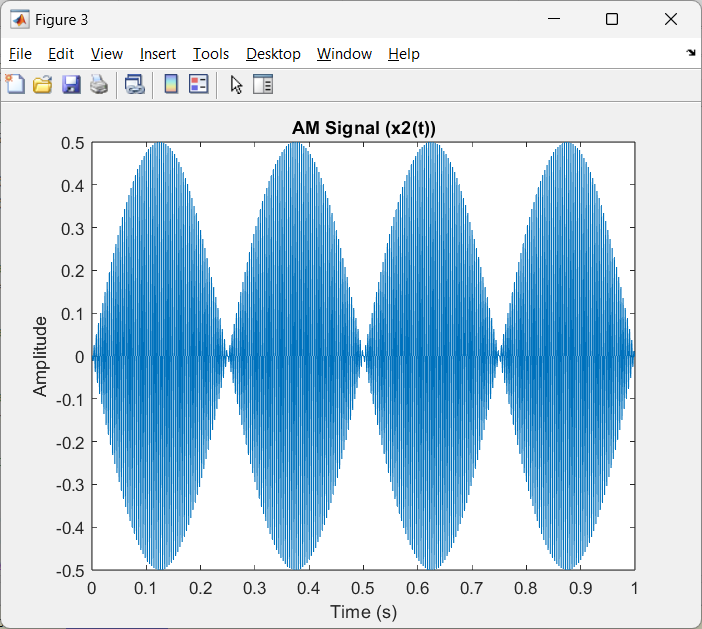
title('AM Signal (x2(t))');

% Listen to the modulated signals

sound(x1, fs);

sound(x2, fs);





**10) Κβάντιση**. 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.

% Signal parameters

t = 0:0.001:0.5; % Time axis

% Generate the noise signal

noise\_signal = randn(size(t));

% Quantization levels

quant\_levels\_8 = linspace(min(noise\_signal), max(noise\_signal), 8);

quant\_levels\_16 = linspace(min(noise\_signal), max(noise\_signal), 16);

% Quantize the noise signal to 8 levels

quantized\_signal\_8 = quantize(noise\_signal, quant\_levels\_8);

% Quantize the noise signal to 16 levels

quantized\_signal\_16 = quantize(noise\_signal, quant\_levels\_16);

% Plot the original signal, quantized signals, quantization levels, and quantization error

figure;

subplot(2,1,1);

plot(t, noise\_signal, 'b');

hold on;

plot(t, quantized\_signal\_8, 'r');

plot(t, quantized\_signal\_16, 'g');

xlabel('Time (s)');

ylabel('Amplitude');

title('Original Signal and Quantized Signals');

legend('Original Signal', 'Quantized (8 levels)', 'Quantized (16 levels)');

hold off;

subplot(2,1,2);

plot(t, noise\_signal - quantized\_signal\_8, 'r');

hold on;

plot(t, noise\_signal - quantized\_signal\_16, 'g');

xlabel('Time (s)');

ylabel('Amplitude');

title('Quantization Error');

legend('Error (8 levels)', 'Error (16 levels)');

hold off;

% Function to perform quantization

function quantized\_signal = quantize(signal, levels)

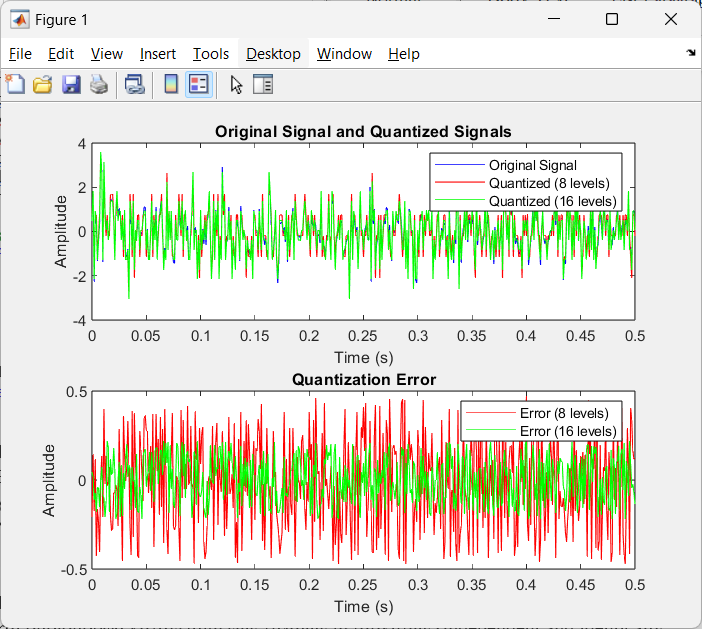
% Calculate the quantization step size

step\_size = levels(2) - levels(1);

% Quantize the signal to the nearest quantization level

quantized\_signal = levels(round((signal - min(levels)) / step\_size) + 1);

end



**11) 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.

% Signal parameters

amplitude = 4; % Amplitude of the sine wave (V)

frequency = 10; % Frequency of the sine wave (Hz)

duration = 1; % Duration of the signal (s)

sampling\_rate = 1000; % Sampling rate (samples per second)

% Time axis

t = linspace(0, duration, duration \* sampling\_rate);

% Generate the sine wave signal

signal = amplitude \* sin(2 \* pi \* frequency \* t);

% Quantization parameters

bit\_depth = 4; % Number of bits for quantization

quantization\_levels = 2^bit\_depth; % Number of quantization levels

% Perform quantization

quantized\_signal = quantize(signal, quantization\_levels);

% Calculate quantization error

quantization\_error = signal - quantized\_signal;

% Plot the original signal, quantized signal, and quantization error

figure;

subplot(3,1,1);

plot(t, signal);

xlabel('Time (s)');

ylabel('Amplitude');

title('Original Signal');

subplot(3,1,2);

plot(t, quantized\_signal);

xlabel('Time (s)');

ylabel('Amplitude');

title('Quantized Signal');

subplot(3,1,3);

plot(t, quantization\_error);

xlabel('Time (s)');

ylabel('Amplitude');

title('Quantization Error');

% Function to perform quantization

function quantized\_signal = quantize(signal, levels)

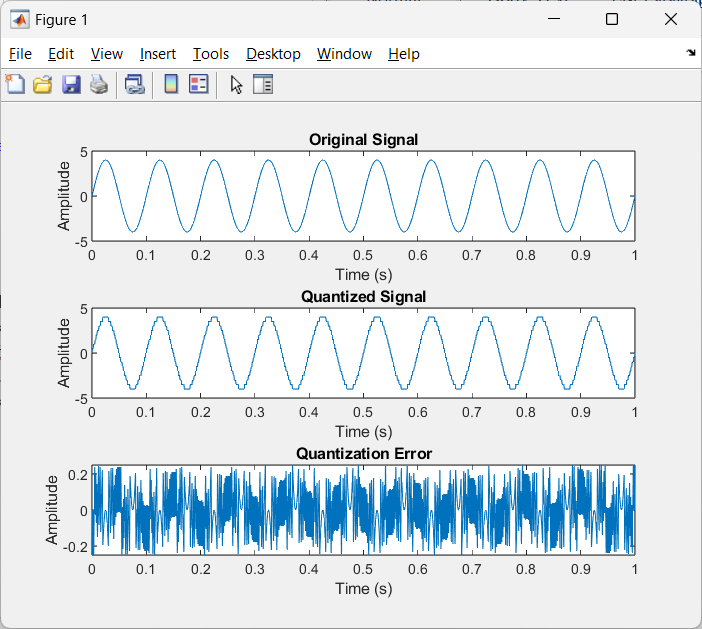
% Calculate the quantization step size

step\_size = (max(signal) - min(signal)) / levels;

% Quantize the signal to the nearest quantization level

quantized\_signal = round(signal / step\_size) \* step\_size;

end



**12) AWGN**. Ένα σήμα λευκού θορύβου αποτελείται από ένα σύνολο ανεξάρτητων και ομοιόμορφα κατανεμημένων τυχαίων μεταβλητών (independent and identically distributed (i.i.d)). Στη διακριτή περίπτωση το σήμα λευκού θορύβου αποτελείται από μία ακολουθία δειγμάτων που είναι ανεξάρτητα και παράγονται από την ίδια κατανομή πιθανότητας (π.χ. ομοιόμορφη ή Gaussian).

Λευκός Gaussian θόρυβος μπορεί να παραχθεί στο Matlab με τη συνάρτηση randn. Για παράδειγμα με τις παρακάτω εντολές παράγεται Λευκός Gaussian θόρυβος (μήκος 30 δειγμάτων) με μηδενική μέση τιμή και τυπική απόκλιση σ=1.

*mu=0;sigma=1;*

*noise= sigma \*randn(1,30)+mu*

Η φασματική πυκνότητα ισχύος Power Spectral Density function (PSD) δείχνει την ποσότητα ισχύος που περιέχεται ανά μονάδα φάσματος.   
Η φασματική πυκνότητα ισχύος μίας τυχαίας διαδικασίας μπορεί να υπολογιστεί από το μετασχηματισμό Fourier της συνάρτησης αυτοσυσχέτισής της.   
Θεωρείται ότι η τιμή της είναι σταθερή για όλο το φάσμα συχνοτήτων και ίση με τη διακύμανση της ισχύος του σήματος θορύβου.

Ενας απλός εκτιμητής της psd είναι το περιοδόγραμμα (periodogram). Συνίσταται στο να ληφθεί ο DTFT των δειγμάτων του σήματος και μετά να υψωθεί στο τετράγωνο το μέτρο του αποτελέσματος. Μπορεί να χρησιμοποιηθεί η συνάρτηση periodogram του Matlab για να υπολογιστεί και σχεδιαστεί το περιοδόγραμμα.

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.

**13) Φασματική πυκνότητα ισχύος AWGN.** ( psd, power spectral density). Δημιουργήστε μία τυχαία διαδικασία λευκού Gaussian θορύβου αποτελούμενη από 1000 τυχαίες ακολουθίες 512 δειγμάτων η καθεμία 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 .

% Parameters

num\_sequences = 1000; % Number of random sequences

sequence\_length = 512; % Length of each sequence

sigma = 2; % Standard deviation

% Generate random white Gaussian noise processes

v = sigma \* randn(num\_sequences, sequence\_length);

% Calculate the periodogram for each sequence

V = zeros(num\_sequences, sequence\_length);

for i = 1:num\_sequences

V(i,:) = abs(fft(v(i,:))).^2 / sequence\_length;

end

% Calculate the average PSD

average\_psd = mean(V, 1);

% Plot the average PSD

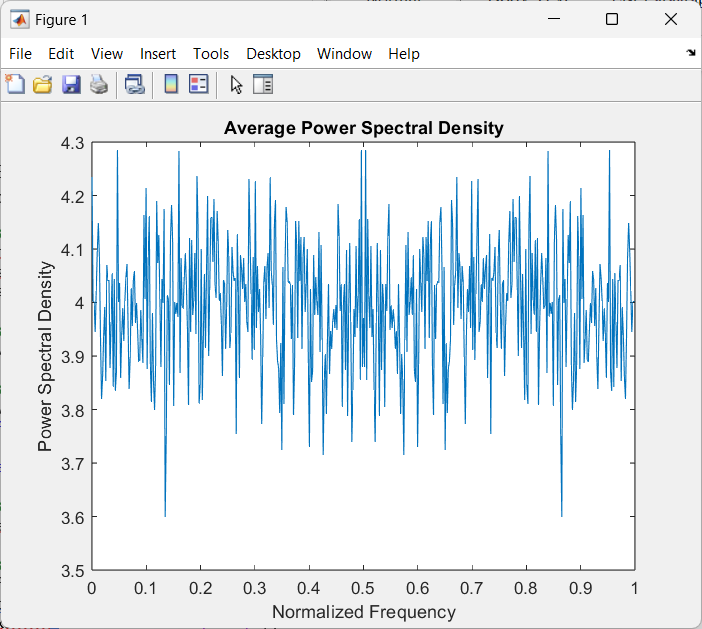
frequency = (0:sequence\_length-1) / sequence\_length;

plot(frequency, average\_psd);

xlabel('Normalized Frequency');

ylabel('Power Spectral Density');

title('Average Power Spectral Density');



**14) Συνάρτηση sinc.** 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 Ν 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.

% Parameters

f1 = 100; % Frequency in Hz

t = -0.05:0.0001:0.05; % Time axis

% Generate m1(t) and m2(t)

m1 = sinc(2\*f1\*t);

m2 = sinc(4\*f1\*t);

% Plot the signals

figure;

subplot(2,1,1);

plot(t, m1);

xlabel('Time (s)');

ylabel('Amplitude');

title('Signal m1(t)');

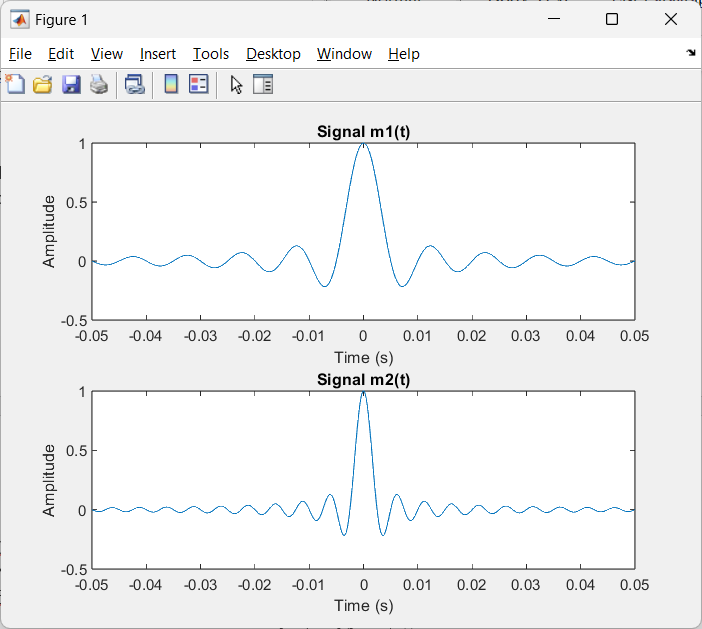
subplot(2,1,2);

plot(t, m2);

xlabel('Time (s)');

ylabel('Amplitude');

title('Signal m2(t)');



**15) Διαμόρφωση DSB. Φάσμα.**Να δημιουργηθεί σήμα πληροφορίας που δίνεται από την ακόλουθη σχέση

*m(t)= 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 Ν 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.

% Parameters

f1 = 100; % Frequency in Hz

fc = 600; % Carrier frequency in Hz

t = -0.05:0.0001:0.05; % Time axis

% Generate m(t)

m = sinc(2\*f1\*t);

% Calculate the number of points for FFT

N = length(t);

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

% Compute the spectrum of m(t)

spectrum = abs(fftshift(fft(m, Nf)));

% Generate the frequency axis

df = 1 / (t(2) - t(1));

f = (-Nf/2 : Nf/2 - 1) \* df / Nf;

% Plot the spectrum of m(t)

figure;

plot(f, spectrum);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title('Spectrum of m(t)');

% Modulate the signal

modulated\_signal = m .\* cos(2\*pi\*fc\*t);

% Compute the spectrum of the modulated signal

modulated\_spectrum = abs(fftshift(fft(modulated\_signal, Nf)));

% Plot the modulated signal and its spectrum

figure;

subplot(2,1,1);

plot(t, modulated\_signal);

xlabel('Time (s)');

ylabel('Amplitude');

title('Modulated Signal');

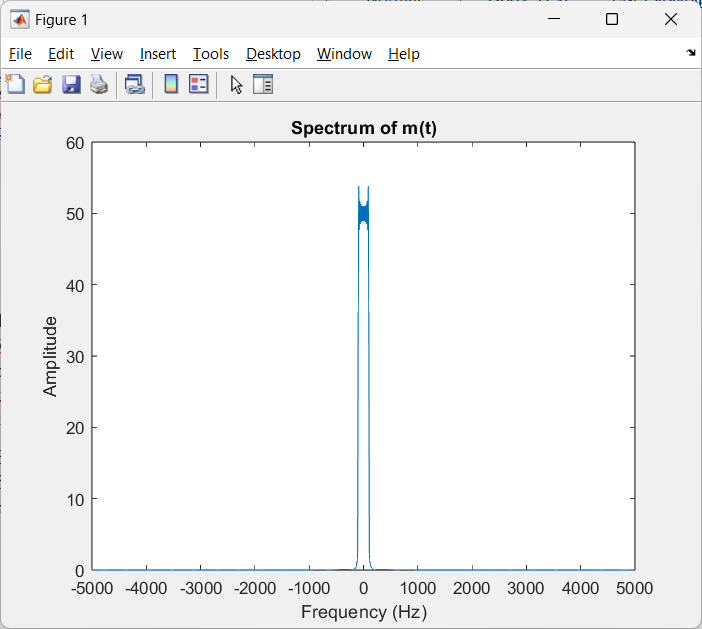
subplot(2,1,2);

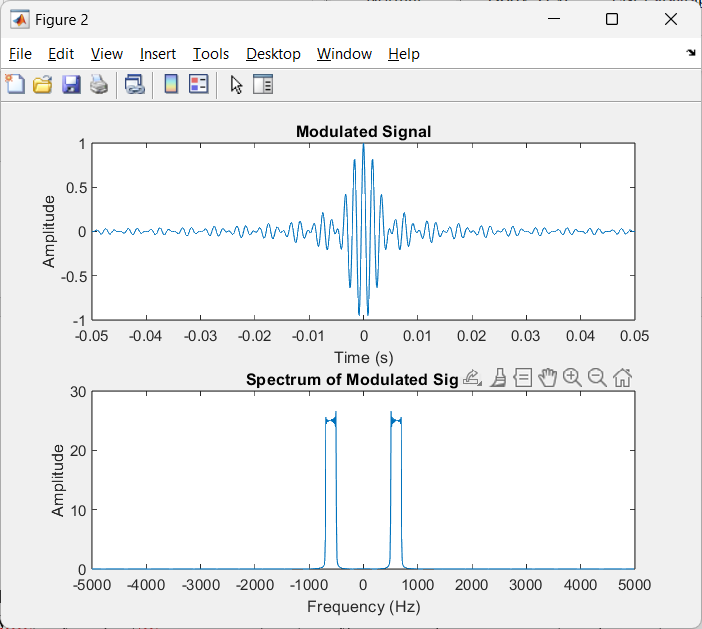
plot(f, modulated\_spectrum);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

title('Spectrum of Modulated Signal');





**16) Αποδιαμόρφωση DSB.** α) Να φτιαχτεί συνημιτονοειδές σήμα πλάτους Αm=0.5 Volt, συχνότητας fa=1KHz. O 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 Αc=2.5 Volt, συχνότητας fc δεκαπλάσιας της συχνότητας του σήματος πληροφορίας O 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.

To 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.

% Parameters

Am = 0.5; % Amplitude of information signal

fa = 1000; % Frequency of information signal (Hz)

AC = 2.5; % Amplitude of carrier signal

fc = 10 \* fa; % Frequency of carrier signal (Hz)

T = 5 \* (1/fa); % Time duration

% Time axis

dt = 0.0001; % Time step

t = 0:dt:T; % Time vector

% Generate information signal (x(t))

x = Am \* sinc(2\*fa\*t);

% Generate carrier signal (y(t))

y = AC \* cos(2\*pi\*fc\*t);

% Modulate carrier signal with information signal (z(t))

z = y .\* x;

% Perform DSB demodulation

lo = cos(2\*pi\*fc\*t); % Local oscillator

demodulated\_signal = butterworth\_filter(z .\* lo, dt, 5, 2\*fa, fc);

% Plotting

figure

% Subplot 1: Original information signal (x(t))

subplot(3,1,1)

plot(t, x)

title('Original Information Signal')

xlabel('Time (s)')

ylabel('Amplitude')

% Subplot 2: Carrier signal (y(t))

subplot(3,1,2)

plot(t, y)

title('Carrier Signal')

xlabel('Time (s)')

ylabel('Amplitude')

% Subplot 3: Modulated signal (z(t))

subplot(3,1,3)

plot(t, z)

title('Modulated Signal')

xlabel('Time (s)')

ylabel('Amplitude')

% New figure for demodulated signal

figure

hold on

% Plot original information signal

plot(t, x, 'b', 'LineWidth', 1.5)

% Plot demodulated signal after filter

plot(t, demodulated\_signal, 'r', 'LineWidth', 1.5)

title('Demodulated Signal')

xlabel('Time (s)')

ylabel('Amplitude')

legend('Original Signal', 'Demodulated Signal')

hold off

% Function for Butterworth filter

function filtered\_signal = butterworth\_filter(signal, dt, order, fcut, fcenter)

% Design Butterworth filter

fs = 1 / dt;

nyquist = fs / 2;

normalized\_fc = fcut / nyquist;

[b, a] = butter(order, normalized\_fc);

% Filter the signal

filtered\_signal = filtfilt(b, a, signal);

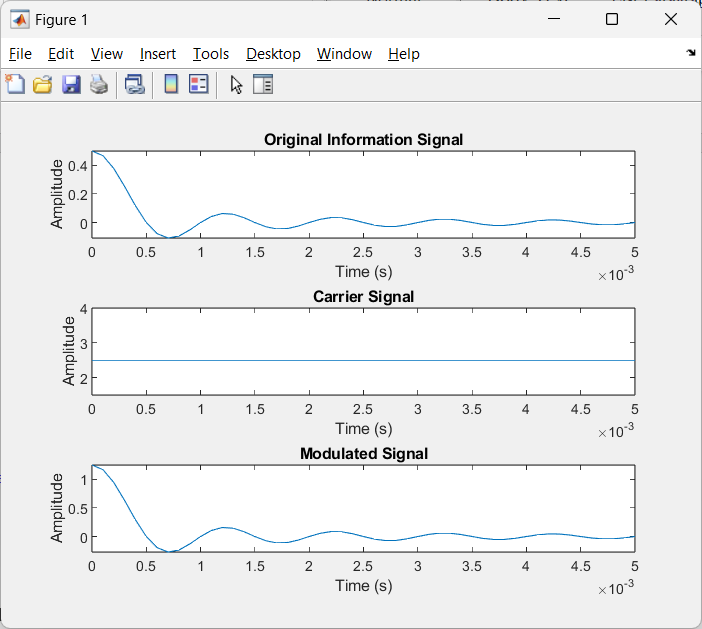
% Shift the filtered signal to the center frequency

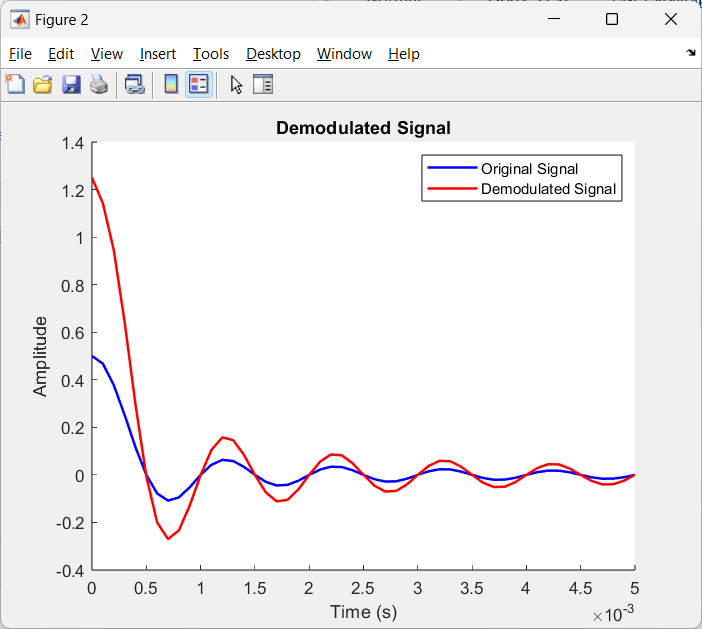
t = (0:length(signal)-1) \* dt;

carrier = cos(2\*pi\*fcenter\*t);

filtered\_signal = filtered\_signal .\* carrier;

end





**17) Αποδιαμόρφωση DSB με μη ιδανικό τοπικό ταλαντωτή**. Να επαναλάβετε τη διαδικασία του ερωτήματος (δ) της άσκησης 16, 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).

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).

% Parameters

Am = 0.5; % Amplitude of information signal

fa = 1000; % Frequency of information signal (Hz)

AC = 2.5; % Amplitude of carrier signal

fc = 10 \* fa; % Frequency of carrier signal (Hz)

T = 5 \* (1/fa); % Time duration

% Time axis

dt = 0.0001; % Time step

t = 0:dt:T; % Time vector

% Generate information signal (x(t))

x = Am \* cos(2\*pi\*fa\*t);

% Generate carrier signal (y(t))

y = AC \* cos(2\*pi\*fc\*t);

% Modulate carrier signal with information signal (z(t))

z = y .\* x;

% Subplot 1: Original information signal (x(t))

subplot(3,1,1)

plot(t, x)

title('Original Information Signal')

xlabel('Time (s)')

ylabel('Amplitude')

% Subplot 2: Carrier signal (y(t))

subplot(3,1,2)

plot(t, y)

title('Carrier Signal')

xlabel('Time (s)')

ylabel('Amplitude')

% Subplot 3: Modulated signal (z(t))

subplot(3,1,3)

plot(t, z)

title('Modulated Signal')

xlabel('Time (s)')

ylabel('Amplitude')

% Perform DSB demodulation

lo = cos(2\*pi\*fc\*t); % Local oscillator

demodulated\_signal = butterworth\_filter(z .\* lo, dt, 5, fa, fc);

% New figure for demodulated signal

figure

hold on

% Plot original information signal

plot(t, x, 'b', 'LineWidth', 1.5)

% Plot demodulated signal after filter

plot(t, demodulated\_signal, 'r', 'LineWidth', 1.5)

title('Demodulated Signal')

xlabel('Time (s)')

ylabel('Amplitude')

legend('Original Signal', 'Demodulated Signal')

hold off

% Function for Butterworth filter

function filtered\_signal = butterworth\_filter(signal, dt, order, fcut, fcenter)

% Design Butterworth filter

fs = 1 / dt;

nyquist = fs / 2;

normalized\_fc = fcut / nyquist;

[b, a] = butter(order, normalized\_fc);

% Filter the signal

filtered\_signal = filtfilt(b, a, signal);

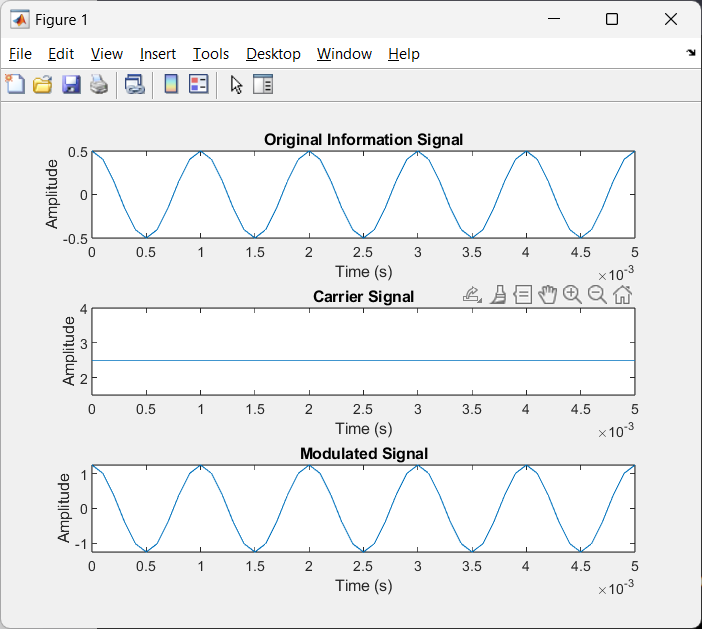
% Shift the filtered signal to the center frequency

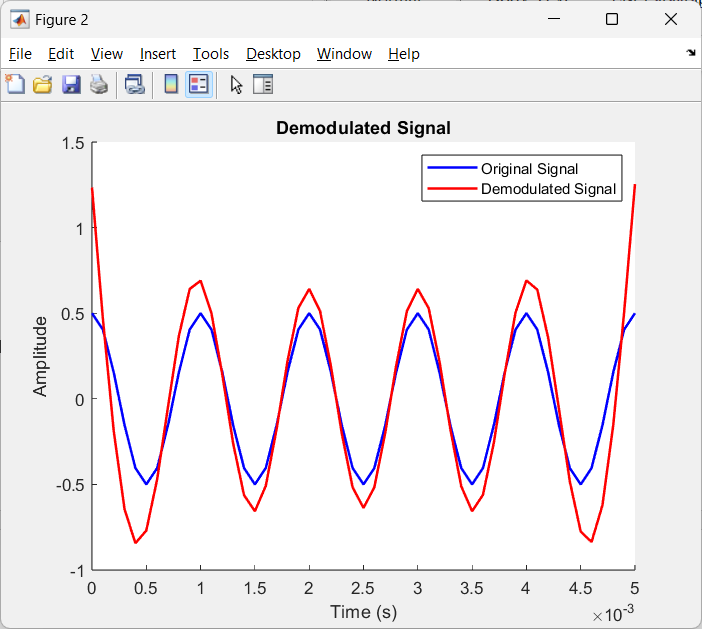
t = (0:length(signal)-1) \* dt;

carrier = cos(2\*pi\*fcenter\*t);

filtered\_signal = filtered\_signal .\* carrier;

end

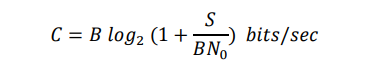




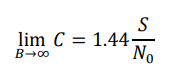
**18) Θεώρημα Shannon Hartley.** Ας θεωρηθεί το θεώρημα Shannon Hartley :



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



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 :



Να γράψετε πρόγραµµα υπολογισµού και αναπαράστασης της χωρητικότητας καναλιού C µε εύρος ζώνης συχνοτήτων Β=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.

**19) Χωρητικότητα καναλιού.** 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?

% Parameters

SNRdB = 25; % Signal-to-Noise Ratio (dB)

SNR = 10^(SNRdB/10); % Convert SNR from dB to linear scale

% Bandwidth frequency range

B = 1:100000; % Bandwidth frequency range

% Calculate channel capacity

C = B \* log2(1 + SNR);

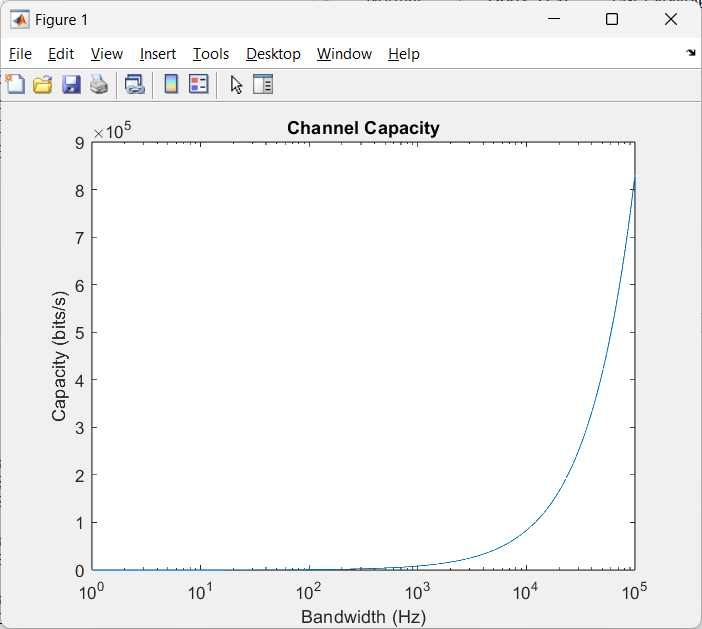
% Plot channel capacity using semilogx

semilogx(B, C)

title('Channel Capacity')

xlabel('Bandwidth (Hz)')

ylabel('Capacity (bits/s)')



**20) Κωδικοποίηση Huffman.** Να δημιουργηθεί η ακολουθία μηνύματος:

'my name is ' <insert name 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. Nα υπολογιστεί η εντροπία της πηγής, η απόδοση του κώδικα και ο πλεονασμός του.

% Define the message sequence

myname = 'my name is <insert name here>';

% Create source alphabet and calculate symbol probabilities

alphabet = unique(myname); % Unique symbols in the message

probabilities = histcounts(double(myname), 0.5:numel(alphabet)+0.5) / numel(myname); % Calculate symbol probabilities

% Create Huffman codebook

huffDict = huffmandict(alphabet, probabilities);

% Perform Huffman encoding of the message

huffEncoded = huffmanenco(myname, huffDict);

% Calculate entropy of the source

entropySource = -sum(probabilities .\* log2(probabilities));

% Calculate performance of the code

bitrateSource = entropySource \* numel(myname); % Number of bits needed to represent the source

bitrateCode = numel(huffEncoded); % Number of bits needed to represent the Huffman-encoded message

performance = bitrateSource / bitrateCode;

% Calculate redundancy

redundancy = 1 - (1 / performance);

% Display results

disp('Huffman Coding Results:')

disp(['Entropy of the source: ' num2str(entropySource) ' bits'])

disp(['Performance of the code: ' num2str(performance)])

disp(['Redundancy of the code: ' num2str(redundancy)])

**21) Δυαδική Διαμόρφωση PSK.** Έστω η ακολουθία μηνύματος m=[1 1 0 0 0 1 1 0 1]. Έστω ότι κάθε παλμός – bit έχει διάρκεια Τ=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.

**22) Διαμόρφωση QAM.** 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.

% Generate and display 16QAM modulated data with SNR=5dB

data = randi([0 15], 1000, 1);

modulatedData = qammod(data, 16);

SNR\_dB = 5;

noisyData = awgn(modulatedData, SNR\_dB, 'measured');

figure;

subplot(1, 2, 1);

scatterplot(modulatedData);

title('16QAM Modulated Data');

xlabel('In-Phase');

ylabel('Quadrature');

subplot(1, 2, 2);

scatterplot(noisyData);

title('16QAM Modulated Data with SNR=5dB');

xlabel('In-Phase');

ylabel('Quadrature');

% Generate and display 4QAM modulated data with SNR=20dB

data = randi([0 3], 1000, 1);

modulatedData = qammod(data, 4);

SNR\_dB = 20;

noisyData = awgn(modulatedData, SNR\_dB, 'measured');

figure;

subplot(1, 2, 1);

scatterplot(modulatedData);

title('4QAM Modulated Data');

xlabel('In-Phase');

ylabel('Quadrature');

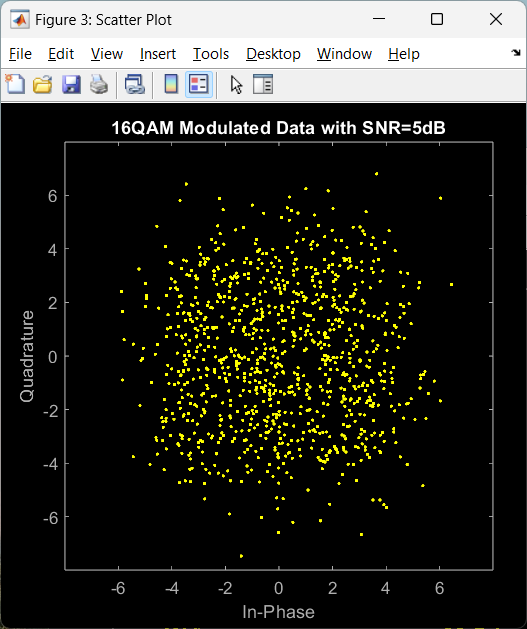
subplot(1, 2, 2);

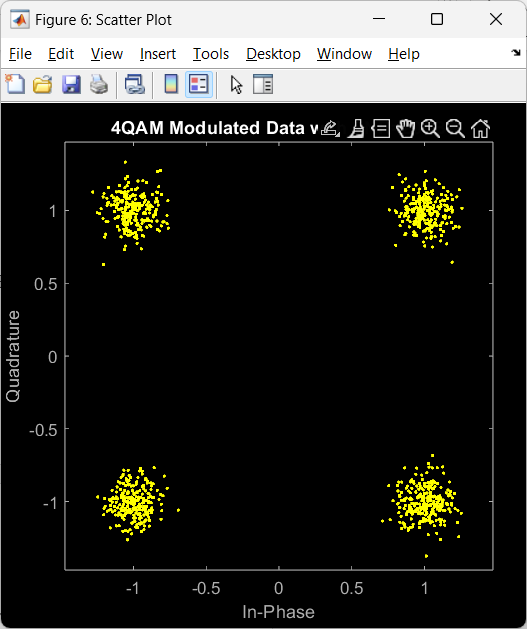
scatterplot(noisyData);

title('4QAM Modulated Data with SNR=20dB');

xlabel('In-Phase');

ylabel('Quadrature');





**23) Εντροπία κειμένου.** 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?

text = 'This is a short text';

% Remove spaces and convert to lowercase

text = lower(strrep(text, ' ', ''));

% Get unique characters in the text

uniqueChars = unique(text);

% Count occurrences of each character

charCount = zeros(1, numel(uniqueChars));

for i = 1:numel(uniqueChars)

charCount(i) = sum(text == uniqueChars(i));

end

% Calculate probabilities of occurrence

totalChars = sum(charCount);

probabilities = charCount / totalChars;

% Calculate entropy

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

% Display probabilities of occurrence

fprintf('Probabilities of occurrence of characters:\n');

for i = 1:numel(uniqueChars)

fprintf('%c: %.4f\n', uniqueChars(i), probabilities(i));

end

% Display entropy

fprintf('Entropy: %.4f\n', entropy);

ask23

Probabilities of occurrence of characters:

a: 0.0625

e: 0.0625

h: 0.1250

i: 0.1250

o: 0.0625

r: 0.0625

s: 0.1875

t: 0.2500

x: 0.0625

Entropy: 2.9528

**24) Εντροπία πηγής.** 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.

**25) Κώδικας Hamming.** Οι κώδικες Hamming είναι γραµµικοί κώδικες block με n=2^m -1, k=2^m - 1-m και ελάχιστη απόσταση dmin=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 σύνταξη της εντολής encode: code=encode(mes,n, k, 'hamming'). Επαληθεύστε θεωρητικά την κωδική λέξη που προκύπτει.

**26) Συνάρτηση Q του Marcum.** Kατά τη μελέτη της επίδρασης του θορύβου στη μετάδοση ψηφιακού σήματος ενδιαφέρουν πιθανότητες 𝑃(𝑋 > 𝑎). To αντίστοιχο ολοκλήρωμα δεν υπολογίζεται αναλυτικά και για τον υπολογισμό του χρησιμοποιείται ο πίνακας της συνάρτησης Q του Marcum. Η τιμή της συνάρτησης Q(x) αντιστοιχεί στο εμβαδόν της Γκαουσιανής καμπύλης από 𝑥 → ∞. .

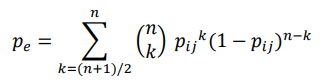
Στο Matlab υπάρχει η έτοιμη συνάρτηση qfunc(x) η οποία υπολογίζει την τιμή της Q(x).

Υπολογίστε της τιμές της συνάρτησης Q (με χρήση της qfunc) για x=[0, 1, 2, 3, 4, 5, 6, 7]. Συγκρίνετε με τις τις αντίστοιχες τιμές του πίνακα της συνάρτησης Q του Marcum. Στη συνέχεια φτιάξτε τη γραφική παράσταση των τιμών αυτών και διαπιστώστε οτι η Q είναι ραγδαίως φθίνουσα συνάρτηση.

**27) Ενθόρυβο σήμα, SNR**. Δημιουργείστε ημιτονοειδές σήμα συχνότητας 100Hz, χρονικής διάρκειας 4sec, πλάτους √2 και με συχνότητα δειγματοληψίας 8KHz. Υπολογίστε την ισχύ του ημιτονοειδούς σήματος.  Στη συνέχεια, χρησιμοποιήστε τη συνάρτηση wgn, ώστε να παράγετε λεύκο Gaussian θόρυβο με SNR=15dB, (σύνταξη : wgn(1,N,Pn,'linear'), όπου, Pn η ισχύς του θορύβου, θα υπολογιστεί από τον τύπο snr=10log10(px/pn)). Προσθέστε το θόρυβο στο σήμα. Αναπαραστήστε γραφικά το ενθόρυβο σήμα.

**28) Μ-αδική διαμόρφωση PSK**. Ανοίξτε την εφαρμογή bertool (Bit Error Rate Analysis) του Matlab. Δημιουργήστε, με τη βοήθεια της εφαρμογής, το κατάλληλο γράφημα για να δείξετε γραφικά ότι στην Μ-αδική διαμόρφωση PSK, η ανοχή στο θόρυβο μειώνεται όσο το Μ αυξάνει. Χρησιμοποιήστε τη θεωρητική ανάλυση (theoritical), τύπο καναλιού AWGN, τύπο διαμόρφωσης PSK, κωδικοποίηση καναλιού καμία και τέλειο συγχρονισμό. Σχολιάστε το γράφημα που δημιουργήσατε.

**29) Κωδικοποίηση με κώδικα επανάληψης.** Μία πολύ απλή περίπτωση μπλοκ κωδικοποίησης είναι ο απλός κώδικας επανάληψης (simple repetition code). Στον κώδικα αυτό η κωδική λέξη αποτελείται από την απλή επανάληψη του 0 (ή του 1) n φορές (n περιττός). Η διαδικασία της αποκωδικοποίησης βασίζεται σε μία πλειοψηφική απόφαση: Στη διαδικασία της αποκωδικοποίησης, λάθος παρατηρείται όταν τουλάχιστον (n+1)/2 από τα εκπεμπόμενα bit έχουν ληφθεί λάθος. Αν το κανάλι επικοινωνίας είναι ένα Δυαδικό Συμμετρικό Κανάλι, BSC, κανάλι, το οποίο εμφανίζει πιθανότητα σφάλματος στο εκπεμπόμενο bit ίση με pij, τότε η πιθανότητα λάθους αποκωδικοποίησης για έναν απλό κώδικα επανάληψης (n,k) αποδεικνύεται ότι δίνεται από την παρακάτω σχέση,



Να γράψετε πρόγραμμα υπολογισμού και αναπαράστασης της πιθανότητας λάθους 𝑝𝑒 για κώδικα επανάληψης, στον οποίο δίνεται ότι η πιθανότητα λάθους σε ένα εκπεμπόμενο bit στο κανάλι επικοινωνίας είναι ίση με pij=0.3 και να γίνει γραφική παράσταση της πιθανότητας λάθους αποκωδικοποίησης pe σε συνάρτηση με το μήκος του κώδικα (μήκος του μπλοκ) n. Να δώσετε τιμές στο n όλες τις περιττές τιμές από 1 έως και 61. Τι πιστεύετε οτι πληρώνουμε για τη μείωση της πιθανότητας σφάλματος με την αύξηση του μήκους κώδικα.

**30) Κωδικοποίηση με κυκλικό κώδικα.** Θεωρήστε κυκλικό κώδικα με n=7 και k=4. Να παράγετε ένα πολυώνυμο γεννήτορα με χρήση της εντολής genpoly = cyclpoly(n,k). Ας σημειωθεί ότι το πολυώνυμο που παράγεται έχει τους συντελεστές του κατά αύξουσα σειρά. Στη συνέχεια, θεωρήστε το μήνυμα msg=[1 1 1 0]. Χρησιμοποιείστε την εντολή code=encode(msg, n, k, 'cyclic', genpoly) για να κωδικοποιήσετε το μήνυμα. Ποια κωδική λέξη προκύπτει? Κωδικοποιήστε και θεωρητικά το μήνυμα με βάση το πολυώνυμο γεννήτορα και συγκρίνετε τα αποτελέσματα.