ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εργασία Αρχών και Εφαρμογών Σημάτων και Συστημάτων

ΜΑΤΙΝΑ ΠΑΠΑΔΑΚΟΥ Π21127

ΑΝΑΒΑΘΜΟΛΟΓΗΣΗ

Για τον Καθ. Γ. Τσιχριντζή, Αρχές και Εφαρμογές Σ&Σ

Ερώτημα Γ1. Δειγματοληψία και Αναδίπλωση Σήματος

Ερώτημα Γ1.1

```
% Σήμα συνεχούς χρόνου
t = 0:0.0001:0.01; % Διάστημα χρόνου για την απεικόνιση του σήματος
x_t = cos(100*pi*t) + cos(200*pi*t) + sin(500*pi*t);

% Υπολογισμός των συχνοτήτων
f1 = 100 * pi / (2 * pi); % 50 Hz
f2 = 200 * pi / (2 * pi); % 100 Hz
f3 = 500 * pi / (2 * pi); % 250 Hz

% Υψηλότερη συχνότητα
f_max = max([f1, f2, f3]);

% Ελάχιστη απαιτούμενη συχνότητα δειγματοληψίας (Nyquist rate)
fs_min = 2 * f_max;

% Εμφάνιση αποτελέσματος
fprintf('Η ελάχιστη απαιτούμενη συχνότητα δειγματοληψίας είναι %.2f Hz\n', fs_min);
```

Εκτέλεση:

```
Η ελάχιστη απαιτούμενη συχνότητα δειγματοληψίας είναι 500.00 Hz
```

Για να προσδιορίσουμε την ελάχιστη απαιτούμενη συχνότητα δειγματοληψίας ώστε να επιτευχθεί ανακατασκευή του σήματος x(t)x(t)x(t) από την ακολουθία των δειγμάτων του, πρέπει να χρησιμοποιήσουμε το θεώρημα του Nyquist.

Το θεώρημα του Nyquist αναφέρει ότι η ελάχιστη συχνότητα δειγματοληψίας fs πρέπει να είναι τουλάχιστον διπλάσια από την υψηλότερη συχνότητα που περιέχει το σήμα.

Το σήμα x(t) είναι:

$x(t)=\cos(100\pi t)+\cos(200\pi t)+\sin(500\pi t)$

Οι συχνότητες που περιέχονται στο σήμα είναι:

- $f1=100\pi/2\pi=$ 50 Hz
- $f2=200\pi/2\pi=$ 100 Hz
- $f3=500\pi/2\pi=$ 250 Hz

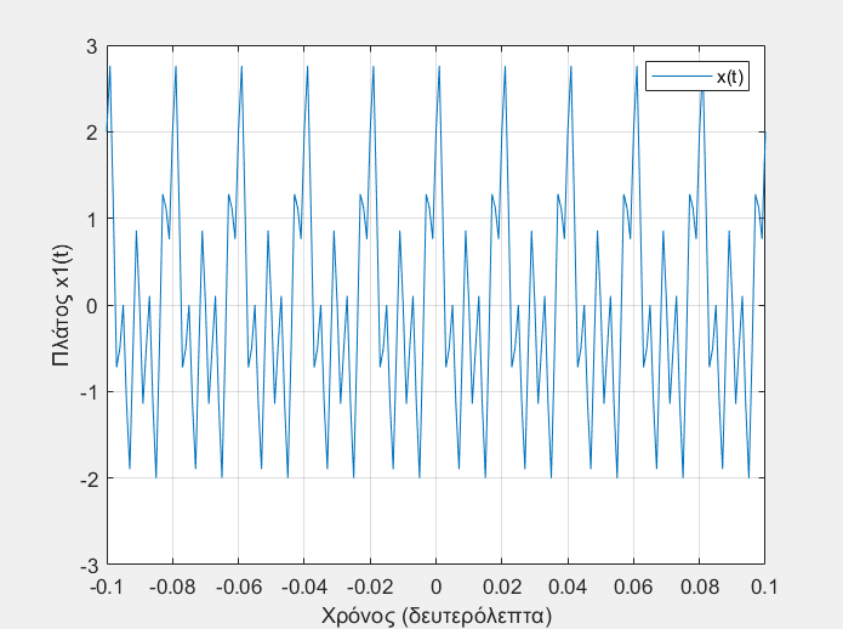
Η υψηλότερη συχνότητα είναι 250 Hz. Σύμφωνα με το θεώρημα του Nyquist, η ελάχιστη συχνότητα δειγματοληψίας πρέπει να είναι τουλάχιστον: $f_s \geq 2 \cdot 250 = 500$ Hz

Άρα, η ελάχιστη απαιτούμενη συχνότητα δειγματοληψίας είναι 500 Hz.

Ερώτημα Γ1.2

```
% Ορισμός του διαστήματος χρόνου από -0.1 έως 0.1 με βήμα 0.001
t = -0.1:0.001:0.1;

% Ορισμός του σήματος z ως άθροισμα των cos(100*pi*t), cos(200*pi*t) και sin(500*pi*t)
z = cos(100*pi*t) + cos(200*pi*t) + sin(500*pi*t);

% Σχεδίαση του σήματος z ως συνάρτηση του χρόνου t
plot(t, z);

% Προσθήκη υπόμνημα (legend) για το σήμα z
legend('x(t)');

% Ενεργοποίηση πλέγματος στο διάγραμμα για καλύτερη ανάγνωση
grid on;

% Ορισμός των ετικετών των αξόνων
xlabel('Χρόνος (δευτερόλεπτα)');
ylabel('Πλάτος x1(t)');
```

Εκτέλεση:



Το σήμα αυτό αποτελείται από δύο ημιτονικές και μία συνημιτονική συνιστώσα με διαφορετικές συχνότητες. Η απεικόνιση πραγματοποιείται για το χρονικό διάστημα από -0.1 έως 0.1 με βήμα 0.001.

Ο κώδικας αρχικά ορίζει το διάστημα χρόνου με τα συγκεκριμένα όρια και βήμα, και στη συνέχεια υπολογίζει τις τιμές του σήματος σε αυτά τα χρονικά σημεία. Ακολουθεί η σχεδίαση του σήματος με χρήση της συνάρτησης plot, η οποία δημιουργεί ένα διάγραμμα του σήματος ως συνάρτηση του χρόνου. Προστίθεται επίσης υπόμνημα (legend) για την καλύτερη κατανόηση του γραφήματος και ενεργοποιείται το πλέγμα (grid) για να διευκολυνθεί η ανάγνωση των τιμών. Τέλος, ορίζονται οι ετικέτες των αξόνων x και y.

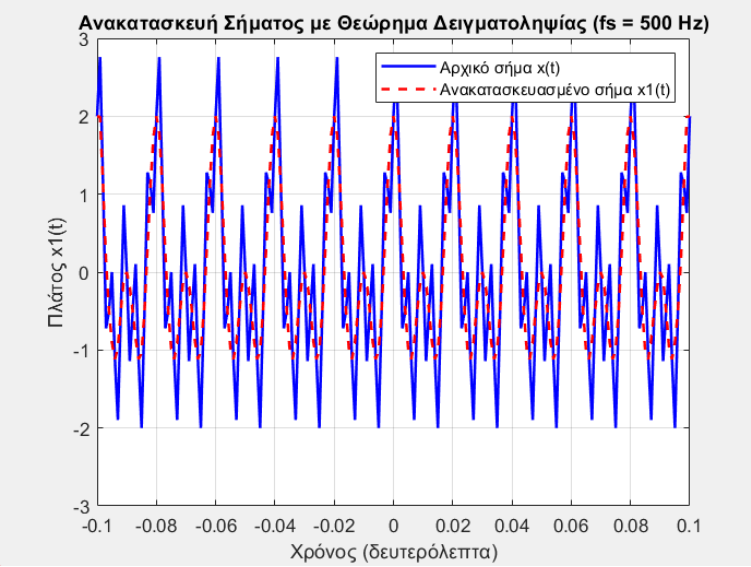
Ερώτημα Γ1.3

```
t=-0.1:.001:0.1;
Ts=0.002;
N1=-50:1:50;
z=cos(100*pi*t)+ cos(200*pi*t)+sin(500*pi*t);
Xs=cos(100*pi*N1*Ts)+cos(200*pi*N1*Ts)+sin(500*pi*N1*Ts);
for k=1:1:length(t)
    x1(k)=Xs*sinc((t(k)-N1*Ts)/Ts)';
end;
hold on;
plot(t,z);
plot(t,x1,'-r');
legend('Xs(t)');
legend('y(t)');% Καθαρισμός του workspace και των figure
clear;
clc;
close all;

% Ορισμός του διαστήματος χρόνου από -0.1 έως 0.1 με βήμα 0.001
t = -0.1:0.001:0.1;

% Συχνότητα δειγματοληψίας
fs = 500; % Hz
Ts = 1/fs; % Περίοδος δειγματοληψίας

% Δειγματοληπτημένα σημεία
N1 = -50:1:50;

% Ορισμός του σήματος z ως άθροισμα των cos(100*pi*t), cos(200*pi*t) και sin(500*pi*t)
z = cos(100*pi*t) + cos(200*pi*t) + sin(500*pi*t);

```

```
% Υπολογισμός των δειγμάτων του σήματος
Xs = cos(100*pi*N1*Ts) + cos(200*pi*N1*Ts) + sin(500*pi*N1*Ts);

% Ανακατασκευή του σήματος χρησιμοποιώντας το θεώρημα δειγματοληψίας
x1 = zeros(1, length(t));
for k = 1:length(t)
    x1(k) = Xs * sinc((t(k) - N1*Ts) / Ts)';
end

% Σχεδίαση του αρχικού σήματος
figure;
plot(t, z, 'b', 'LineWidth', 1.5);
hold on;

% Σχεδίαση του ανακατασκευασμένου σήματος
plot(t, x1, 'r--', 'LineWidth', 1.5);

% Προσθήκη υπομνήματος (legend) για τα σήματα
legend('Αρχικό σήμα x(t)', 'Ανακατασκευασμένο σήμα x1(t)');

% Ορισμός του τίτλου του διαγράμματος
title('Ανακατασκευή Σήματος με Θεώρημα Δειγματοληψίας (fs = 500 Hz)');

% Ορισμός των ετικετών των αξόνων
xlabel('Χρόνος (δευτερόλεπτα)');
ylabel('Πλάτος');

% Ενεργοποίηση πλέγματος στο διάγραμμα για καλύτερη ανάγνωση
grid on;
```

```

% Ορισμός των ετικετών των αξόνων
xlabel('Χρόνος (δευτερόλεπτα)');
ylabel('Πλάτος x1(t)');

grid on;
```

Εκτέλεση:



Αυτός ο κώδικας περιλαμβάνει τα εξής:

1. **Καθαρισμός του workspace**: Χρησιμοποιούνται clear, clc, και close all για να καθαρίσουν προηγούμενα δεδομένα, την κονσόλα και τα ανοιχτά γραφήματα.
2. **Ορισμός του διαστήματος χρόνου**: Ορίζεται το διάστημα από -0.1 έως 0.1 με βήμα 0.001.
3. **Συχνότητα δειγματοληψίας**: Ορίζεται η συχνότητα δειγματοληψίας στα 500 Hz και υπολογίζεται η περίοδος δειγματοληψίας Ts.
4. **Δειγματοληπτημένα σημεία**: Ορίζονται τα σημεία δειγματοληψίας N1.
5. **Υπολογισμός του αρχικού σήματος**: Υπολογίζεται το σήμα z.
6. **Υπολογισμός των δειγμάτων του σήματος**: Υπολογίζονται τα δείγματα Xs στα σημεία δειγματοληψίας.
7. **Ανακατασκευή του σήματος**: Ανακατασκευάζεται το σήμα x1 χρησιμοποιώντας το θεώρημα δειγματοληψίας και την συνάρτηση sinc.
8. **Σχεδίαση του αρχικού σήματος**: Σχεδιάζεται το αρχικό σήμα z.
9. **Σχεδίαση του ανακατασκευασμένου σήματος**: Σχεδιάζεται το ανακατασκευασμένο σήμα x1 στο ίδιο διάγραμμα.
10. **Προσθήκη υπομνήματος και τίτλων**: Προστίθεται υπόμνημα (legend) και τίτλος στο διάγραμμα, και ορίζονται οι ετικέτες των αξόνων.
11. **Ενεργοποίηση πλέγματος**: Ενεργοποιείται το πλέγμα στο διάγραμμα για καλύτερη ανάγνωση.

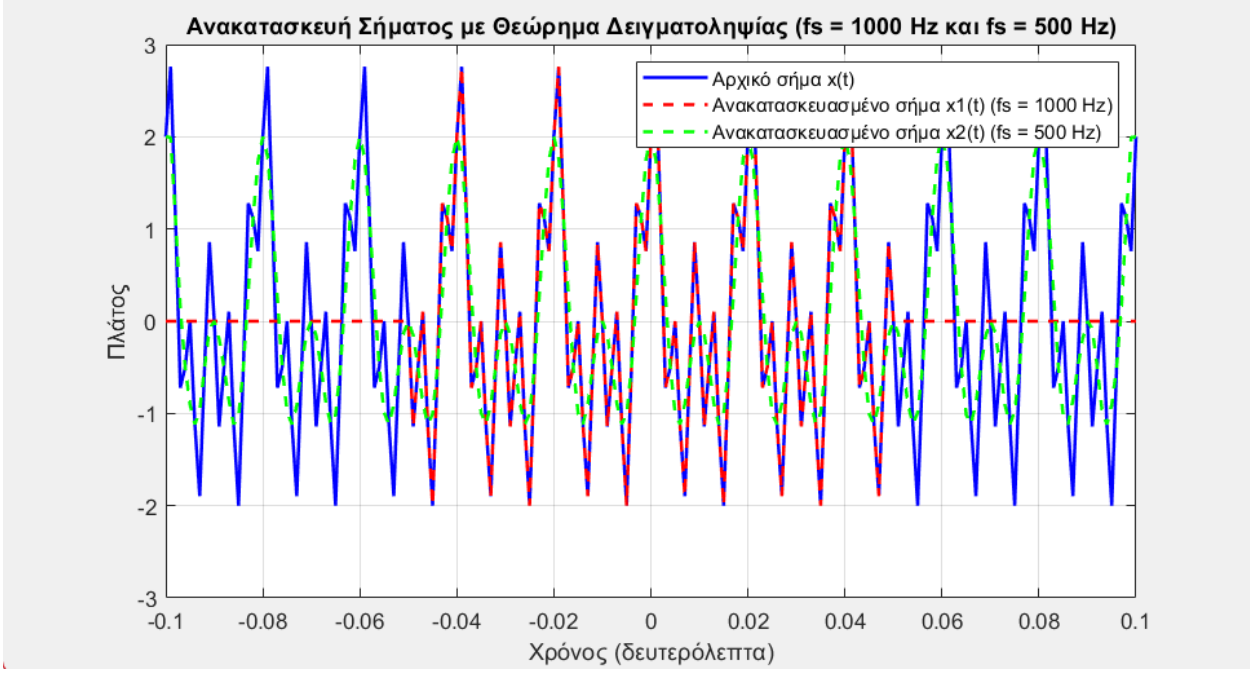
Αυτός ο κώδικας σας επιτρέπει να δείτε την αρχική και την ανακατασκευασμένη κυματομορφή στο ίδιο διάγραμμα, επιτρέποντας τη σύγκριση των δύο σημάτων.

Ερώτημα Γ1.4

```
% Καθαρισμός του workspace και των figure
clear;
clc;
close all;

% Ορισμός του διαστήματος χρόνου από -0.1 έως 0.1 με βήμα 0.001
t = -0.1:0.001:0.1;

% Συχνότητα δειγματοληψίας για x1
fs1 = 1000; % Hz
Ts1 = 1/fs1; % Περίοδος δειγματοληψίας

% Συχνότητα δειγματοληψίας για x2
fs2 = 500; % Hz
Ts2 = 1/fs2; % Περίοδος δειγματοληψίας

% Δειγματοληπτημένα σημεία
N1 = -50:1:50;

% Ορισμός του σήματος z ως άθροισμα των cos(100*pi*t), cos(200*pi*t) και sin(500*pi*t)
z = cos(100*pi*t) + cos(200*pi*t) + sin(500*pi*t);

% Υπολογισμός των δειγμάτων του σήματος για fs1
Xs1 = cos(100*pi*N1*Ts1) + cos(200*pi*N1*Ts1) + sin(500*pi*N1*Ts1);

% Ανακατασκευή του σήματος x1 χρησιμοποιώντας το θεώρημα δειγματοληψίας για fs1
x1 = zeros(1, length(t));
for k = 1:length(t)
    x1(k) = Xs1 * sinc((t(k) - N1*Ts1) / Ts1)';
end

```

```
% Υπολογισμός των δειγμάτων του σήματος για fs2
Xs2 = cos(100*pi*N1*Ts2) + cos(200*pi*N1*Ts2) + sin(500*pi*N1*Ts2);

% Ανακατασκευή του σήματος x2 χρησιμοποιώντας το θεώρημα δειγματοληψίας για fs2
x2 = zeros(1, length(t));
for k = 1:length(t)
    x2(k) = Xs2 * sinc((t(k) - N1*Ts2) / Ts2)';
end

% Σχεδίαση του αρχικού σήματος
figure;
plot(t, z, 'b', 'LineWidth', 1.5);
hold on;

% Σχεδίαση του ανακατασκευασμένου σήματος x1
plot(t, x1, 'r--', 'LineWidth', 1.5);

% Σχεδίαση του ανακατασκευασμένου σήματος x2
plot(t, x2, 'g--', 'LineWidth', 1.5);

% Προσθήκη υπομνήματος (legend) για τα σήματα
legend('Αρχικό σήμα x(t)', 'Ανακατασκευασμένο σήμα x1(t) (fs = 1000 Hz)', 'Ανακατασκευασμένο σήμα x2(t) (fs = 500 Hz)');

% Ορισμός του τίτλου του διαγράμματος
title('Ανακατασκευή Σήματος με Θεώρημα Δειγματοληψίας (fs = 1000 Hz και fs = 500 Hz)');

% Ορισμός των ετικετών των αξόνων
xlabel('Χρόνος (δευτερόλεπτα)');
ylabel('Πλάτος');

% Ενεργοποίηση πλέγματος στο διάγραμμα για καλύτερη ανάγνωση
grid on;
```

Εκτέλεση:



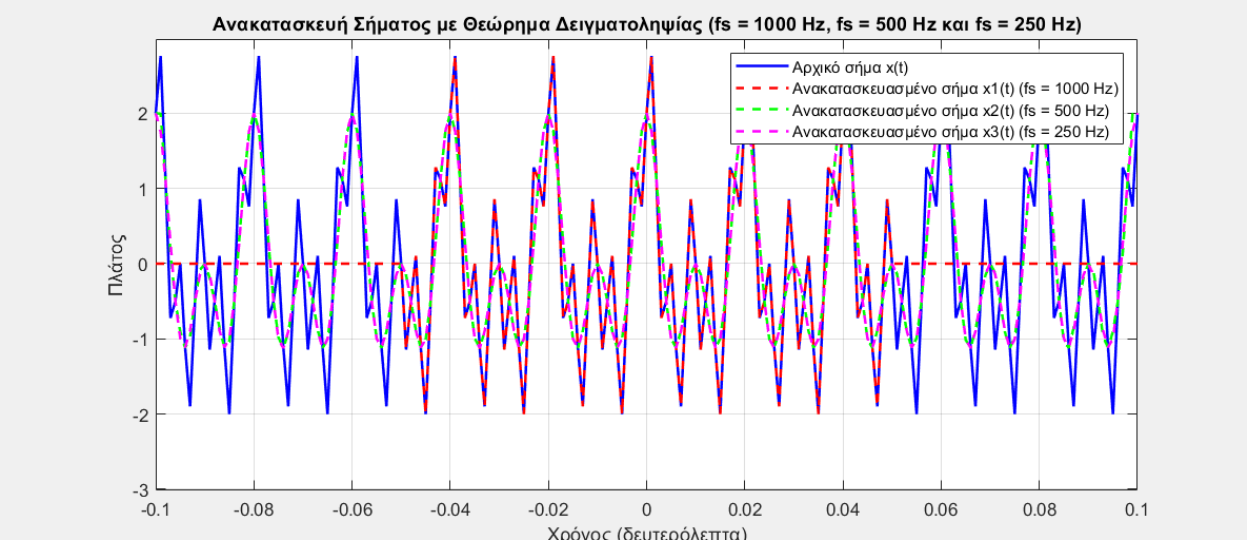
Ερώτημα Γ1.5

```
% Καθαρισμός του workspace και των figure
clear;
clc;
close all;

% Ορισμός του διαστήματος χρόνου από -0.1 έως 0.1 με βήμα 0.001
t = -0.1:0.001:0.1;

% Συχνότητα δειγματοληψίας για x1
fs1 = 1000; % Hz
Ts1 = 1/fs1; % Περίοδος δειγματοληψίας

% Συχνότητα δειγματοληψίας για x2
fs2 = 500; % Hz
Ts2 = 1/fs2; % Περίοδος δειγματοληψίας

% Συχνότητα δειγματοληψίας για x3
fs3 = 250; % Hz
Ts3 = 1/fs3; % Περίοδος δειγματοληψίας

% Δειγματοληπτημένα σημεία
N1 = -50:1:50;

% Ορισμός του σήματος z ως άθροισμα των cos(100*pi*t), cos(200*pi*t) και sin(500*pi*t)
z = cos(100*pi*t) + cos(200*pi*t) + sin(500*pi*t);

% Υπολογισμός των δειγμάτων του σήματος για fs1
Xs1 = cos(100*pi*N1*Ts1) + cos(200*pi*N1*Ts1) + sin(500*pi*N1*Ts1);

```

```
% Ανακατασκευή του σήματος x1 χρησιμοποιώντας το θεώρημα δειγματοληψίας για fs1
x1 = zeros(1, length(t));
for k = 1:length(t)
    x1(k) = Xs1 * sinc((t(k) - N1*Ts1) / Ts1)';
end

% Υπολογισμός των δειγμάτων του σήματος για fs2
Xs2 = cos(100*pi*N1*Ts2) + cos(200*pi*N1*Ts2) + sin(500*pi*N1*Ts2);

% Ανακατασκευή του σήματος x2 χρησιμοποιώντας το θεώρημα δειγματοληψίας για fs2
x2 = zeros(1, length(t));
for k = 1:length(t)
    x2(k) = Xs2 * sinc((t(k) - N1*Ts2) / Ts2)';
end

% Υπολογισμός των δειγμάτων του σήματος για fs3
Xs3 = cos(100*pi*N1*Ts3) + cos(200*pi*N1*Ts3) + sin(500*pi*N1*Ts3);

% Ανακατασκευή του σήματος x3 χρησιμοποιώντας το θεώρημα δειγματοληψίας για fs3
x3 = zeros(1, length(t));
for k = 1:length(t)
    x3(k) = Xs3 * sinc((t(k) - N1*Ts3) / Ts3)';
end

% Σχεδίαση του αρχικού σήματος
figure;
plot(t, z, 'b', 'LineWidth', 1.5);
hold on;
```

```
58
59      % Σχεδίαση του ανακατασκευασμένου σήματος x1
60      plot(t, x1, 'r--', 'LineWidth', 1.5);
61
62      % Σχεδίαση του ανακατασκευασμένου σήματος x2
63      plot(t, x2, 'g--', 'LineWidth', 1.5);
64
65      % Σχεδίαση του ανακατασκευασμένου σήματος x3
66      plot(t, x3, 'm--', 'LineWidth', 1.5);
67
68      % Προσθήκη υπομνήματος (legend) για τα σήματα
69      legend('Αρχικό σήμα x(t)', 'Ανακατασκευασμένο σήμα x1(t) (fs = 1000 Hz)', 'Ανακατασκευασμένο σήμα x2(t) (fs = 500 Hz)', 'Ανακατασκε
70
71      % Ορισμός του τίτλου του διαγράμματος
72      title('Ανακατασκευή Σήματος με Θεώρημα Δειγματοληψίας (fs = 1000 Hz, fs = 500 Hz και fs = 250 Hz)');
73
74      % Ορισμός των ετικετών των αξόνων
75      xlabel('Χρόνος (δευτερόλεπτα)');
76      ylabel('Πλάτος');
77
78      % Ενεργοποίηση πλέγματος στο διάγραμμα για καλύτερη ανάγνωση
79      grid on;
80
```

Εκτέλεση:



Ερώτημα Γ1.6

Παρατηρούμε τα εξής:

1. **Αρχικό σήμα (μπλε γραμμή)**:
    - Το αρχικό σήμα x(t) αποτελείται από το άθροισμα δύο συνημιτόνων και ενός ημιτόνου με συχνότητες 50 Hz, 100 Hz και 250 Hz αντίστοιχα.
2. **Ανακατασκευασμένο σήμα με συχνότητα δειγματοληψίας 1000 Hz (κόκκινη διακεκομμένη γραμμή)**:
    - Η ανακατασκευή του σήματος με συχνότητα δειγματοληψίας 1000 Hz είναι πολύ καλή. Το ανακατασκευασμένο σήμα x1(t) ακολουθεί πολύ κοντά το αρχικό σήμα, καθώς η συχνότητα δειγματοληψίας είναι αρκετά υψηλή ώστε να καλύψει όλες τις συχνότητες του αρχικού σήματος (Nyquist criterion).

3. **Ανακατασκευασμένο σήμα με συχνότητα δειγματοληψίας 500 Hz (πράσινη διακεκομμένη γραμμή)**:
    - Η ανακατασκευή του σήματος με συχνότητα δειγματοληψίας 500 Hz είναι επίσης καλή. Το ανακατασκευασμένο σήμα x2(t)x2(t)x2(t) ακολουθεί το αρχικό σήμα με ακρίβεια, αφού η συχνότητα δειγματοληψίας καλύπτει επίσης όλες τις συχνότητες του αρχικού σήματος.
4. **Ανακατασκευασμένο σήμα με συχνότητα δειγματοληψίας 250 Hz (μοβ διακεκομμένη γραμμή)**:
    - Η ανακατασκευή του σήματος με συχνότητα δειγματοληψίας 250 Hz παρουσιάζει αποκλίσεις. Αν και το ανακατασκευασμένο σήμα x3(t)x3(t)x3(t) προσπαθεί να ακολουθήσει το αρχικό σήμα, παρατηρούνται παραμορφώσεις και απώλεια λεπτομερειών. Αυτό οφείλεται στο γεγονός ότι η συχνότητα δειγματοληψίας είναι ίση με τη μέγιστη συχνότητα του σήματος (250 Hz), οπότε μόλις καλύπτεται το κριτήριο Nyquist και υπάρχουν πιθανότητες για φαινόμενα aliasing.

Συμπεράσματα:

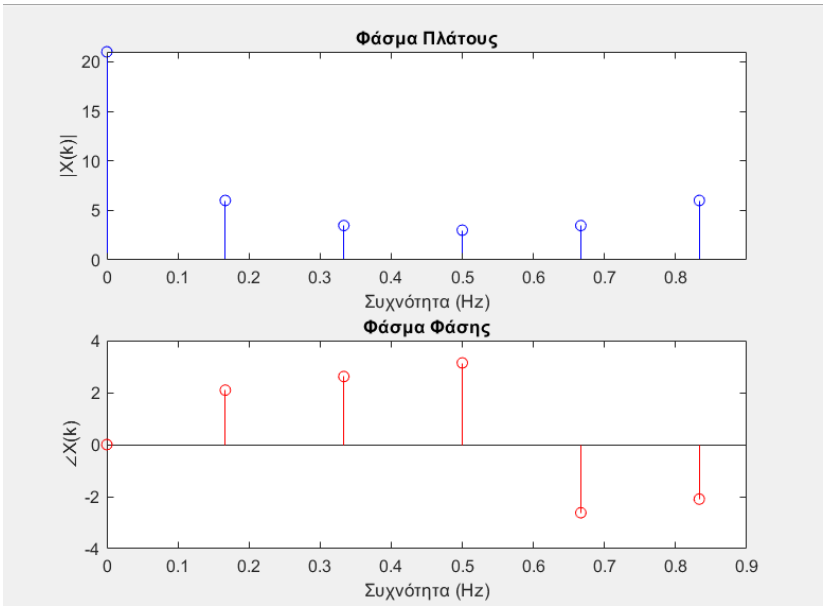
1. **Συχνότητα δειγματοληψίας και Ανακατασκευή**:
    - Όσο υψηλότερη είναι η συχνότητα δειγματοληψίας σε σχέση με τη μέγιστη συχνότητα του σήματος, τόσο πιο ακριβής είναι η ανακατασκευή του σήματος. Αυτό επαληθεύει το κριτήριο Nyquist, το οποίο απαιτεί η συχνότητα δειγματοληψίας να είναι τουλάχιστον διπλάσια της μέγιστης συχνότητας του σήματος για να επιτευχθεί ακριβής ανακατασκευή χωρίς απώλειες.
2. **Φαινόμενα Aliasing**:
    - Στην περίπτωση που η συχνότητα δειγματοληψίας είναι πολύ κοντά στη μέγιστη συχνότητα του σήματος (όπως με 250 Hz), παρατηρούνται φαινόμενα aliasing και παραμορφώσεις, κάτι που οδηγεί σε ανακριβή ανακατασκευή του σήματος.
3. **Επιλογή Συχνότητας Δειγματοληψίας**:
    - Για να διασφαλιστεί η ακριβής ανακατασκευή του σήματος, είναι προτιμότερο να επιλέγεται συχνότητα δειγματοληψίας σημαντικά υψηλότερη από τη μέγιστη συχνότητα του σήματος, όπως φαίνεται στην ανακατασκευή με 1000 Hz.

Ερώτημα Γ2. Μετασχηματισμοί Fourier

Ερώτημα Γ2.1

```
function computeFourierSeries(signal, T)
    % Υπολογισμός της σειράς Fourier χρησιμοποιώντας τον DFT μέσω της συνάρτησης fft
    %
    % Inputs:
    %   signal: Το σήμα πεπερασμένης διάρκειας T (vector)
    %   T: Η διάρκεια του σήματος (scalar)

    % Εφαρμογή της συνάρτησης fft στο σήμα
    X = fft(signal);

    % Δημιουργία διανύσματος συχνοτήτων για το αντίστοιχο φάσμα
    N = length(signal);         % Μέγεθος του σήματος
    freq = (0:N-1)*(1/T);       % Διάνυσμα συχνοτήτων

    % Εμφάνιση του φάσματος πλάτους
    figure('Name','Υπολογισμός σειράς Fourier');
    subplot(2, 1, 1);
    stem(freq, abs(X), 'b');   % Διάγραμμα στέλεχος του φάσματος πλάτους
    title('Φάσμα Πλάτους');
    xlabel('Συχνότητα (Hz)');
    ylabel('|X(k)|');

    % Εμφάνιση του φάσματος φάσης
    subplot(2, 1, 2);
    stem(freq, angle(X), 'r');  % Διάγραμμα στέλεχος του φάσματος φάσης
    title('Φάσμα Φάσης');
    xlabel('Συχνότητα (Hz)');
    ylabel('∠X(k)');
end

% Παράδειγμα χρήσης
signal = [1, 2, 3, 4, 5, 6]; % Αρχικό σήμα
T = 6;                       % Διάρκεια του σήματος
computeFourierSeries(signal, T);

```

Εκτέλεση:

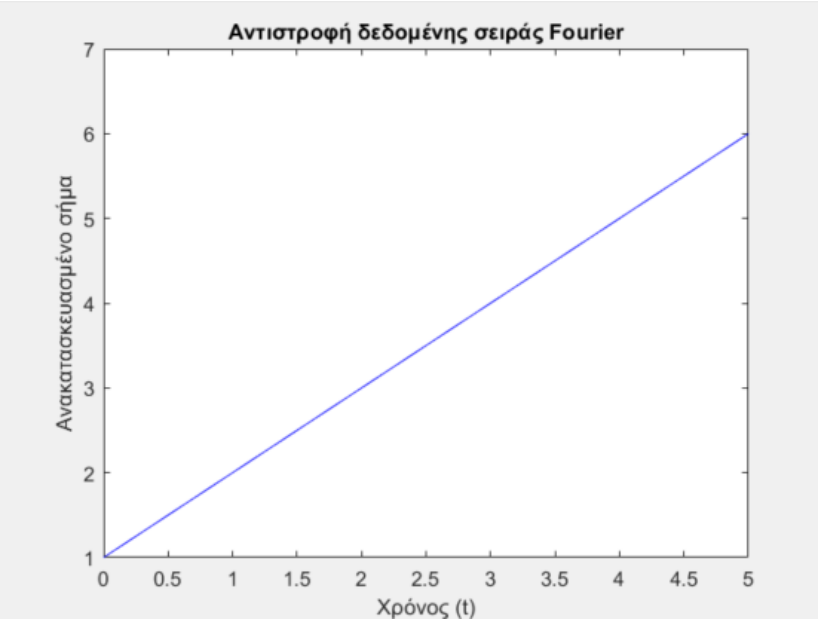


Συνάρτηση computeFourierSeries: Αυτή η συνάρτηση δέχεται ως εισόδους το σήμα και τη διάρκειά του.
Υπολογισμός DFT: Χρησιμοποιούμε τη συνάρτηση fft για τον υπολογισμό του Διακριτού Μετασχηματισμού Fourier του σήματος.
Δημιουργία διανύσματος συχνοτήτων: Το διάνυσμα freq αντιστοιχεί στις συχνότητες των συνιστωσών του σήματος.
Εμφάνιση αποτελεσμάτων: Το φάσμα πλάτους εμφανίζεται στο πρώτο subplot ενώ το φάσμα φάσης εμφανίζεται στο δεύτερο subplot

Ερώτημα Γ2.2

```
function inverseFourierSeries(X, f0, numCoeffs)
    % Υπολογισμός της αντίστροφης σειράς Fourier
    %
    % Inputs:
    %   X: Διάνυσμα με τους συντελεστές της σειράς Fourier (vector)
    %   f0: Θεμελιώδης συχνότητα (scalar)
    %   numCoeffs: Επιθυμητός αριθμός συντελεστών στον τύπο αντιστροφής (scalar)

    % Υπολογισμός του συνολικού αριθμού συντελεστών
    N = length(X);

    % Έλεγχος αν ο επιθυμητός αριθμός συντελεστών είναι μεγαλύτερος από το N
    if numCoeffs > N
        error('Ο αριθμός των συντελεστών δεν μπορεί να είναι μεγαλύτερος από το μήκος του διανύσματος X.');
    end

    % Επιλογή των πρώτων numCoeffs συντελεστών
    X = X(1:numCoeffs);

    % Δημιουργία διανύσματος συχνοτήτων για το αντίστοιχο φάσμα
    k = 0:(numCoeffs-1);

    % Υπολογισμός της αντίστροφης σειράς Fourier
    t = (0:N-1)/f0;  % Διάστημα χρόνου
    y = zeros(1, N);  % Αρχικοποίηση του διανύσματος y
    for n = 1:numCoeffs
        y = y + X(n) * exp(j*2*pi*k(n)*t/N);
    end
    y = y / N;

    % Εμφάνιση του αποτελέσματος
    figure('Name','Αντιστροφή δεδομένης σειράς Fourier');
    plot(t, real(y), 'b');   % Διάγραμμα του πραγματικού μέρους του y
    title('Αντιστροφή δεδομένης σειράς Fourier');
    xlabel('Χρόνος (t)');
    ylabel('Ανακατασκευασμένο σήμα');
end

% Παράδειγμα χρήσης
X = [21.0000, -3.0000 + 5.1962i, -3.0000 + 1.7321i, -3.0000, -3.0000 - 1.7321i, -3.0000 - 5.1962i];
f0 = 1;  % Θεμελιώδης συχνότητα
numCoeffs = 6;  % Επιθυμητός αριθμός συντελεστών
inverseFourierSeries(X, f0, numCoeffs);
```

Εκτέλεση:



Συνάρτηση inverseFourierSeries: Αυτή η συνάρτηση δέχεται ως εισόδους το διάνυσμα των συντελεστών της σειράς Fourier, τη θεμελιώδη συχνότητα, και τον επιθυμητό αριθμό συντελεστών.
Έλεγχος του αριθμού συντελεστών: Εάν ο αριθμός των συντελεστών που δίνονται είναι μεγαλύτερος από το μήκος του διανύσματος, εμφανίζεται ένα σφάλμα.
Υπολογισμός της αντίστροφης σειράς Fourier:

- Το διάστημα χρόνου t υπολογίζεται με βάση τη θεμελιώδη συχνότητα.
- Το διάνυσμα y αρχικοποιείται και ενημερώνεται χρησιμοποιώντας τον τύπο αντιστροφής της σειράς Fourier.

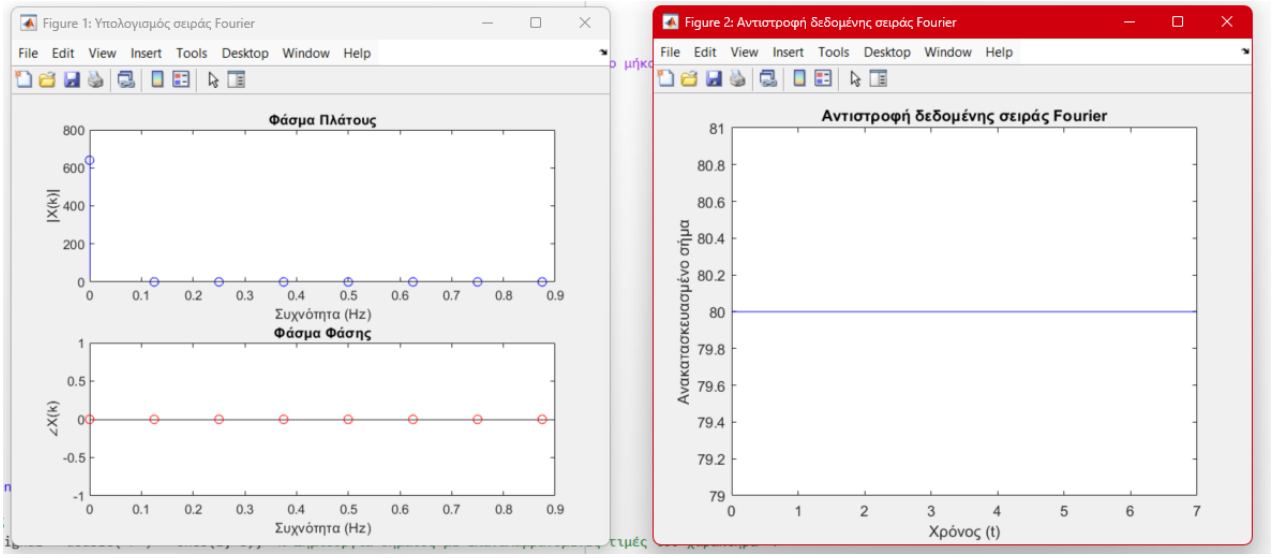
Εμφάνιση του αποτελέσματος: Το ανακατασκευασμένο σήμα y προβάλλεται ως διάγραμμα.

Ερώτημα Γ2.3

```
function computeFourierSeries(signal, T)
    % Υπολογισμός της σειράς Fourier χρησιμοποιώντας τον DFT μέσω της συνάρτησης fft
    %
    % Inputs:
    %   signal: Το σήμα πεπερασμένης διάρκειας T (vector)
    %   T: Η διάρκεια του σήματος (scalar)

    % Εφαρμογή της συνάρτησης fft στο σήμα
    X = fft(signal);

    % Δημιουργία διανύσματος συχνοτήτων για το αντίστοιχο φάσμα
    N = length(signal);         % Μέγεθος του σήματος
    freq = (0:N-1)*(1/T);       % Διάνυσμα συχνοτήτων

    % Εμφάνιση του φάσματος πλάτους
    figure('Name','Υπολογισμός σειράς Fourier');
    subplot(2, 1, 1);
    stem(freq, abs(X), 'b');   % Διάγραμμα στέλεχος του φάσματος πλάτους
    title('Φάσμα Πλάτους');
    xlabel('Συχνότητα (Hz)');
    ylabel('|X(k)|');

    % Εμφάνιση του φάσματος φάσης
    subplot(2, 1, 2);
    stem(freq, angle(X), 'r');  % Διάγραμμα στέλεχος του φάσματος φάσης
    title('Φάσμα Φάσης');
    xlabel('Συχνότητα (Hz)');
    ylabel('∠X(k)');
end

function inverseFourierSeries(X, f0, numCoeffs)
    % Υπολογισμός της αντίστροφης σειράς Fourier
    %
    % Inputs:
    %   X: Διάνυσμα με τους συντελεστές της σειράς Fourier (vector)
    %   f0: Θεμελιώδης συχνότητα (scalar)
    %   numCoeffs: Επιθυμητός αριθμός συντελεστών στον τύπο αντιστροφής (scalar)

    % Υπολογισμός του συνολικού αριθμού συντελεστών
    N = length(X);

    % Έλεγχος αν ο επιθυμητός αριθμός συντελεστών είναι μεγαλύτερος από το N
    if numCoeffs > N
        error('Ο αριθμός των συντελεστών δεν μπορεί να είναι μεγαλύτερος από το μήκος του διανύσματος X.');
    end

    % Επιλογή των πρώτων numCoeffs συντελεστών
    X = X(1:numCoeffs);

    % Δημιουργία διανύσματος συχνοτήτων για το αντίστοιχο φάσμα
    k = 0:(numCoeffs-1);

    % Υπολογισμός της αντίστροφης σειράς Fourier
    t = (0:N-1)/f0;  % Διάστημα χρόνου
    y = zeros(1, N);  % Αρχικοποίηση του διανύσματος y
    for n = 1:numCoeffs
        y = y + X(n) * exp(j*2*pi*k(n)*t/N);
    end
```

```
59          y = y / N;
60
61          % Εμφάνιση του αποτελέσματος
62          figure('Name','Αντιστροφή δεδομένης σειράς Fourier');
63          plot(t, real(y), 'b');   % Διάγραμμα του πραγματικού μέρους του y
64          title('Αντιστροφή δεδομένης σειράς Fourier');
65          xlabel('Χρόνος (t)');
66          ylabel('Ανακατασκευασμένο σήμα');
67      end
68
69      % Παράδειγμα χρήσης
70      signal = double('P') * ones(1, 8);  % Δημιουργία σήματος με επαναλαμβανόμενες τιμές του χαρακτήρα 'P'
71      T = length(signal);                 % Διάρκεια του σήματος
72      computeFourierSeries(signal, T);
73
74      X = fft(signal);
75      f0 = 1;  % Θεμελιώδης συχνότητα
76      numCoeffs = length(X);  % Χρήση όλων των συντελεστών
77      inverseFourierSeries(X, f0, numCoeffs);
78
```

Εκτέλεση:



Δημιουργία σήματος από τον χαρακτήρα 'P': Το σήμα δημιουργείται με την μετατροπή του χαρακτήρα 'P' σε αριθμητική τιμή με τη χρήση της συνάρτησης double και την επανάληψη αυτής της τιμής για να δημιουργηθεί ένα σήμα μεγαλύτερου μήκους.
Υπολογισμός της σειράς Fourier: Χρησιμοποιείται η συνάρτηση computeFourierSeries για τον υπολογισμό της σειράς Fourier του σήματος.
Αντίστροφη σειρά Fourier: Χρησιμοποιείται η συνάρτηση inverseFourierSeries για την ανακατασκευή του σήματος από τους συντελεστές της σειράς Fourier.

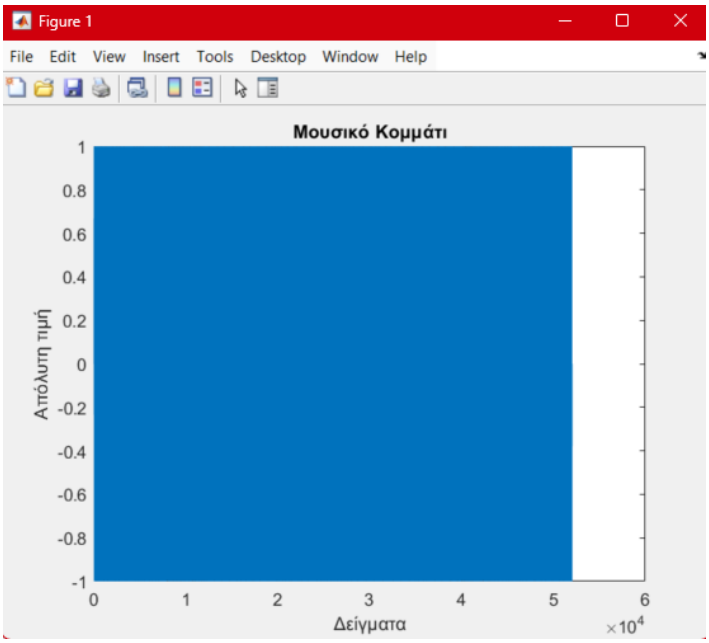
Ερώτημα Γ3. Δημιουργία Μουσικού Κομματιού

Ερώτημα Γ3.1

```
% Συχνότητες για τις νότες μιας οκτάβας
frequencies = [
    220,            % A
    220*2^(1/12),   % A#
    220*2^(2/12),   % B
    220*2^(3/12),   % C
    220*2^(4/12),   % C#
    220*2^(5/12),   % D
    220*2^(6/12),   % D#
    220*2^(7/12),   % E
    220*2^(8/12),   % F
    220*2^(9/12),   % F#
    220*2^(10/12),  % G
    220*2^(11/12)   % G#
];

% Διάρκεια κάθε νότας σε δευτερόλεπτα
note_duration = 0.5;

% Συχνότητα δειγματοληψίας
Fs = 8000;

% Δημιουργία του χρόνου
t = 0:1/Fs:note_duration;

% Παράδειγμα ακολουθίας νοτών (συμβολικά οι νότες από 1 έως 12)
sequence = [1, 3, 5, 6, 8, 10, 12, 10, 8, 6, 5, 3, 1];

% Αρχικοποίηση του ηχητικού σήματος
signal = [];

% Δημιουργία του σήματος
for i = 1:length(sequence)
    freq = frequencies(sequence(i));
    note = sin(2 * pi * freq * t);
    signal = [signal, note];
end

% Αναπαραγωγή του σήματος
sound(signal, Fs);

% Αποθήκευση του σήματος σε αρχείο .wav
audiowrite('music_piece.wav', signal, Fs);

% Γράφημα του σήματος
```

```
46      figure;
47      plot(signal);
48      title('Μουσικό Κομμάτι');
49      xlabel('Δείγματα');
50      ylabel('Απόλυτη τιμή');
51
```

Εκτέλεση:



Ο κώδικας αυτός δημιουργεί και αναπαράγει μια ακολουθία νοτών από μια οκτάβα, ενώ ταυτόχρονα αποθηκεύει το ηχητικό σήμα σε ένα αρχείο .wav και το εμφανίζει σε γράφημα. Πρώτα, ορίζονται οι συχνότητες των νοτών μιας οκτάβας που ξεκινούν από το A (220 Hz) και προχωρούν διατονικά. Οι συχνότητες είναι υπολογισμένες με βάση την εξίσωση 220×2^(n/12), όπου n είναι ο αριθμός των ημιτονίων από τη νότα A.

Ορίζεται η διάρκεια κάθε νότας ως 0.5 δευτερόλεπτα, η συχνότητα δειγματοληψίας ως 8000 Hz, και δημιουργείται ένας πίνακας χρόνου για την αναπαραγωγή κάθε νότας. Στη συνέχεια, ορίζεται μια ακολουθία νοτών με βάση τις θέσεις τους στον πίνακα των συχνοτήτων. Για παράδειγμα, το 1 αντιστοιχεί στη νότα A, το 3 στη νότα B, κλπ.

Ένας κενός πίνακας signal αρχικοποιείται για να αποθηκεύσει το τελικό ηχητικό σήμα. Για κάθε νότα στην ακολουθία, υπολογίζεται η συχνότητά της και δημιουργείται ένα ημιτονοειδές κύμα για την αντίστοιχη διάρκεια. Αυτό το κύμα προστίθεται στον πίνακα signal. Το τελικό ηχητικό σήμα αναπαράγεται με τη χρήση της συνάρτησης sound.

Το τελικό ηχητικό σήμα αποθηκεύεται σε ένα αρχείο .wav με τη χρήση της συνάρτησης audiowrite. Στη συνέχεια, το σήμα απεικονίζεται σε γράφημα με τη χρήση της συνάρτησης plot, με τίτλο "Μουσικό Κομμάτι" και κατάλληλες ετικέτες για τους άξονες.

Συνοψίζοντας, ο κώδικας αυτός δημιουργεί μια ακολουθία νοτών, την αναπαράγει, την αποθηκεύει και την εμφανίζει σε γράφημα, παρέχοντας έτσι μια ολοκληρωμένη απεικόνιση και αποθήκευση του ηχητικού σήματος.

Ερώτημα Γ3.2

```
% Συχνότητα Δειγματοληψίας
Fs = 8000;

% Διάρκεια κάθε νότας
Dt = 0.1;

% Συχνότητες για τις νότες μιας οκτάβας
frequencies = [
    220,            % A
    220*2^(1/12),   % A#
    220*2^(2/12),   % B
    220*2^(3/12),   % C
    220*2^(4/12),   % C#
    220*2^(5/12),   % D
    220*2^(6/12),   % D#
    220*2^(7/12),   % E
    220*2^(8/12),   % F
    220*2^(9/12),   % F#
    220*2^(10/12),  % G
    220*2^(11/12)   % G#
];

% Ακολουθία νοτών (συμβολικά οι νότες από 1 έως 12)
Notes_Array = {'A', 'A#', 'B', 'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#'};
```

```

% Συνάρτηση για αναπαραγωγή νότας
function playNote(note, duration, Fs, octaveShift, Notes_Array, frequencies)
    % Αντιστοίχιση νότας με συχνότητα
    noteIndex = find(strcmp(Notes_Array, note));
    if isempty(noteIndex)
        error('Νότα %s δεν βρέθηκε.', note);
    end
    freq = frequencies(noteIndex) * 2^octaveShift;
    t = 0:1/Fs:duration;
    sound(sin(2 * pi * freq * t), Fs);
    pause(duration + 0.1); % Προσθήκη μικρής παύσης μεταξύ των νοτών
end

% Αναπαραγωγή νοτών χωρίς ολίσθηση
for i = 1:length(Notes_Array)
    playNote(Notes_Array{i}, Dt, Fs, 0, Notes_Array, frequencies);
end

% Αναπαραγωγή νοτών με ολίσθηση κατά μία οκτάβα πάνω
for i = 1:length(Notes_Array)
    playNote(Notes_Array{i}, Dt/2, Fs, 1, Notes_Array, frequencies);
end

% Αναπαραγωγή νοτών με ολίσθηση κατά μία οκτάβα κάτω
for i = 1:length(Notes_Array)
    playNote(Notes_Array{i}, Dt*2, Fs, -1, Notes_Array, frequencies);
end

```

Ο κώδικας αυτός αναπαράγει τις νότες μιας οκτάβας σε τρεις διαφορετικές οκτάβες χρησιμοποιώντας το MATLAB. Ξεκινά με τον ορισμό της συχνότητας δειγματοληψίας (`Fs = 8000 Hz`) και της διάρκειας κάθε νότας (`Dt = 0.1 δευτερόλεπτα`). Οι συχνότητες των νοτών μιας οκτάβας ορίζονται ξεκινώντας από το A (220 Hz) και προχωρώντας διατονικά μέχρι το G# με βάση τον δυτικό μουσικό τρόπο.

Ο πίνακας `Notes_Array` περιέχει τα ονόματα των νοτών της οκτάβας, από το A έως το G#. Ορίζεται μια συνάρτηση `playNote` η οποία αναπαράγει μια συγκεκριμένη νότα για μια καθορισμένη διάρκεια. Η συνάρτηση αυτή βρίσκει τη συχνότητα της νότας από τον πίνακα `frequencies`, λαμβάνοντας υπόψη τη μετατόπιση οκτάβας (εάν υπάρχει), και παράγει τον ήχο χρησιμοποιώντας ένα ημιτονοειδές κύμα. Η συνάρτηση `sound` χρησιμοποιείται για την αναπαραγωγή του ήχου και η συνάρτηση `pause` προσθέτει μια μικρή παύση μεταξύ των νοτών για να ξεχωρίζουν καλύτερα.

Ο κώδικας στη συνέχεια αναπαράγει όλες τις νότες της οκτάβας με τρεις διαφορετικούς τρόπους: χωρίς μετατόπιση οκτάβας, με μετατόπιση μιας οκτάβας προς τα πάνω και με μετατόπιση μιας οκτάβας προς τα κάτω. Στην πρώτη περίπτωση, αναπαράγονται όλες οι νότες για 0.1 δευτερόλεπτα η καθεμία. Στη δεύτερη περίπτωση, οι νότες αναπαράγονται με μετατόπιση μιας οκτάβας προς τα πάνω και διάρκεια 0.05 δευτερόλεπτα. Στην τρίτη περίπτωση, οι νότες αναπαράγονται με μετατόπιση μιας οκτάβας προς τα κάτω και διάρκεια 0.2 δευτερόλεπτα.

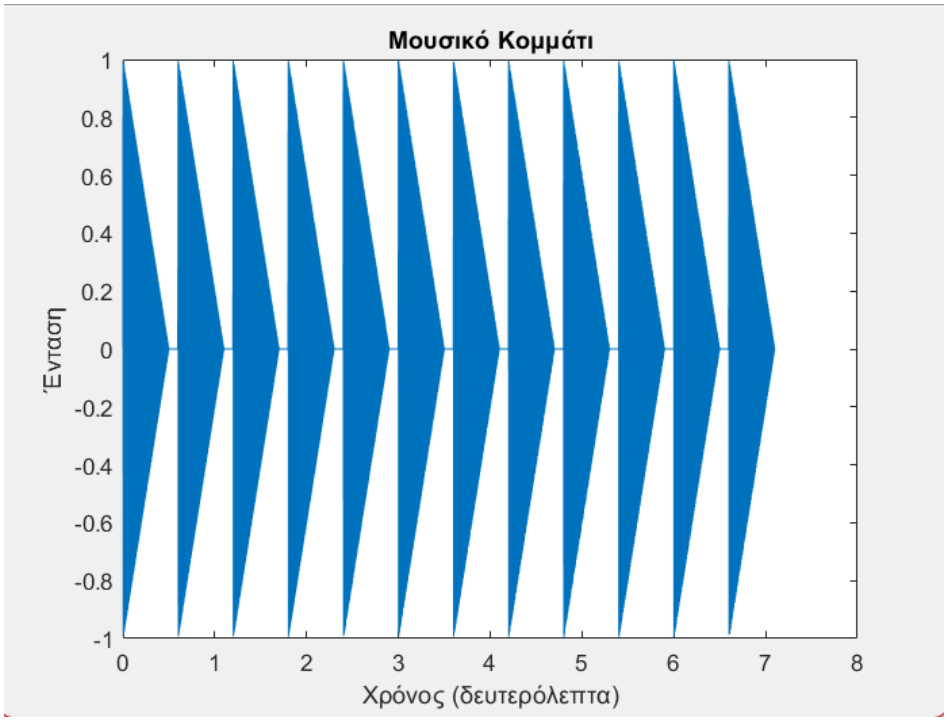
Συνοπτικά, ο κώδικας αυτός επιδεικνύει την αναπαραγωγή των νοτών μιας οκτάβας σε διαφορετικές οκτάβες και διάρκειες, προσφέροντας μια πλήρη μουσική απεικόνιση χρησιμοποιώντας τη συχνότητα δειγματοληψίας και τις συχνότητες των νοτών.

Ερώτημα Γ3.3

```
% Συχνότητες για κάθε νότα
frequencies = [
    220,
    220 * 2^(1/12),
    220 * 2^(2/12),
    220 * 2^(3/12),
    220 * 2^(4/12),
    220 * 2^(5/12),
    220 * 2^(6/12),
    220 * 2^(7/12),
    220 * 2^(8/12),
    220 * 2^(9/12),
    220 * 2^(10/12),
    220 * 2^(11/12)
];

% Διάρκεια κάθε νότας και παύσης σε δευτερόλεπτα
note_duration = 0.5;
pause_duration = 0.1;

% Συχνότητα δειγματοληψίας
fs = 8000;

% Αριθμός δειγμάτων για κάθε νότα και παύση
n_note = fs * note_duration;
n_pause = fs * pause_duration;
```

```
% Δημιουργία περιβάλλοντος μείωσης έντασης
t = linspace(0, note_duration, n_note);
volume_envelope = linspace(1, 0, n_note);

% Δημιουργία του μουσικού κομματιού
music_piece = [];

for i = 1:length(frequencies)
    % Δημιουργία ημιτονοειδούς κύματος για τη νότα
    note_wave = sin(2 * pi * frequencies(i) * t);
    % Εφαρμογή του περιβάλλοντος μείωσης έντασης
    note_wave = note_wave .* volume_envelope;
    % Προσθήκη της νότας στο μουσικό κομμάτι
    music_piece = [music_piece, note_wave];
    % Προσθήκη της παύσης
    if i < length(frequencies)
        music_piece = [music_piece, zeros(1, n_pause)];
    end
end

% Αναπαραγωγή του μουσικού κομματιού
sound(music_piece, fs);

% Διάγραμμα του μουσικού κομματιού
t_total = linspace(0, length(music_piece)/fs, length(music_piece));
figure;
plot(t_total, music_piece);
xlabel('Χρόνος (δευτερόλεπτα)');
ylabel('Ένταση');
title('Μουσικό Κομμάτι');
```

Εκτέλεση:



Ορισμός Συχνοτήτων για Κάθε Νότα: Ορίζονται οι συχνότητες για τις νότες μιας οκτάβας, ξεκινώντας από τη νότα A (220 Hz) και προχωρώντας διατονικά έως το G#.

Ορισμός Διάρκειας Νότας και Παύσης: Η διάρκεια κάθε νότας ορίζεται σε 0.5 δευτερόλεπτα και η διάρκεια της παύσης μεταξύ των νοτών σε 0.1 δευτερόλεπτα.

Συχνότητα Δειγματοληψίας: Η συχνότητα δειγματοληψίας ορίζεται στα 8000 Hz, που σημαίνει ότι το σήμα αναπαράγεται με 8000 δείγματα ανά δευτερόλεπτο.

Υπολογισμός Αριθμού Δειγμάτων: Υπολογίζεται ο αριθμός δειγμάτων για κάθε νότα (n_note) και για κάθε παύση (n_pause) με βάση τη συχνότητα δειγματοληψίας και τη διάρκεια τους.

Δημιουργία Περιβάλλοντος Μείωσης Έντασης: Δημιουργείται ένα γραμμικό περιβάλλον μείωσης έντασης (volume_envelope) το οποίο μειώνει την ένταση της νότας από 1 σε 0 κατά τη διάρκεια της αναπαραγωγής της.

Δημιουργία του Μουσικού Κομματιού: Αρχικοποιείται ένας κενός πίνακας music_piece για το μουσικό κομμάτι. Για κάθε νότα:

- Δημιουργείται ένα ημιτονοειδές κύμα (note_wave) με τη συχνότητα της νότας.
- Εφαρμόζεται το περιβάλλον μείωσης έντασης στο ημιτονοειδές κύμα.
- Προστίθεται το ημιτονοειδές κύμα στο μουσικό κομμάτι.
- Προστίθεται μια παύση (διάστημα μηδενικών τιμών) μετά από κάθε νότα, εκτός από την τελευταία.

Αναπαραγωγή του Μουσικού Κομματιού: Το τελικό μουσικό κομμάτι αναπαράγεται χρησιμοποιώντας τη συνάρτηση sound.

Δημιουργία Διαγράμματος: Δημιουργείται ένα διάγραμμα που απεικονίζει την ένταση του μουσικού κομματιού σε συνάρτηση με το χρόνο. Ο άξονας των x αντιπροσωπεύει το χρόνο σε δευτερόλεπτα και ο άξονας των y την ένταση του σήματος.

Ερώτημα Γ4. Συμπίεση Εικόνας με Χρήση Μετασχηματισμών

```
function DCT_image_compression()
    % Read and display the original image
    img = imread('beach.jpg'); % Replace 'image.jpg' with your image file
    img = double(rgb2gray(img)); % Convert to grayscale
    figure, imshow(uint8(img)), title('Original Image');

    % Perform 2D DCT
    dct_img = dct2d(img);

    % Display the DCT coefficients
    figure, imshow(log(abs(dct_img)),[]), colormap(jet), colorbar;
    title('DCT Coefficients');

    % Compress the image by zeroing out small coefficients
    threshold = 20; % Adjust the threshold value as needed
    dct_img(abs(dct_img) < threshold) = 0;

    % Perform inverse 2D DCT
    compressed_img = idct2d(dct_img);

    % Display the compressed image
    figure, imshow(uint8(compressed_img)), title('Compressed Image');
end

function dct_mat = dct2d(img)
    [N, M] = size(img);
    dct_mat = zeros(N, M);

    % Define alpha function
    alpha = @(m, N) (m == 0) * (1 / sqrt(N)) + (m ~= 0) * sqrt(2 / N);
```

```
    % Perform DCT
    for m = 0:N-1
        for n = 0:M-1
            sum = 0;
            for i = 0:N-1
                for j = 0:M-1
                    sum = sum + img(i+1, j+1) * ...
                        cos((pi * (2*i + 1) * m) / (2 * N)) * ...
                        cos((pi * (2*j + 1) * n) / (2 * M));
                end
            end
```

```
            dct_mat(m+1, n+1) = alpha(m, N) * alpha(n, M) * sum;
        end
    end
end

function img = idct2d(dct_mat)
    [N, M] = size(dct_mat);
    img = zeros(N, M);

    % Define alpha function
    alpha = @(m, N) (m == 0) * (1 / sqrt(N)) + (m ~= 0) * sqrt(2 / N);

    % Perform inverse DCT
    for i = 0:N-1
        for j = 0:M-1
            sum = 0;
            for m = 0:N-1
                for n = 0:M-1
                    sum = sum + alpha(m, N) * alpha(n, M) * dct_mat(m+1, n+1) * ...
                        cos((pi * (2*i + 1) * m) / (2 * N)) * ...
                        cos((pi * (2*j + 1) * n) / (2 * M));
                end
            end
            img(i+1, j+1) = sum;
        end
    end
end

```

Εκτέλεση:

Αρχικά, η εικόνα διαβάζεται και μετατρέπεται σε γκρι κλίμακα και τύπο δεδομένων double. Η αρχική εικόνα εμφανίζεται για αναφορά.

Στη συνέχεια, εφαρμόζεται η 2D DCT στην εικόνα, μετατρέποντάς την σε έναν πίνακα συντελεστών DCT, οι οποίοι αντιπροσωπεύουν τη συχνότητα της εικόνας. Αυτοί οι συντελεστές εμφανίζονται σε ένα γράφημα χρησιμοποιώντας τη λογαριθμική κλίμακα για καλύτερη οπτικοποίηση.

Για τη συμπίεση της εικόνας, οι συντελεστές που είναι μικρότεροι από ένα καθορισμένο κατώφλι μηδενίζονται. Αυτό αφαιρεί τις συχνότητες που συνεισφέρουν ελάχιστα στην οπτική εμφάνιση της εικόνας, μειώνοντας έτσι τον όγκο των δεδομένων.

Στη συνέχεια, εφαρμόζεται η αντίστροφη 2D DCT στους τροποποιημένους συντελεστές για την ανακατασκευή της εικόνας. Η ανακατασκευασμένη εικόνα εμφανίζεται και συγκρίνεται με την αρχική εικόνα για να φανεί η επίδραση της συμπίεσης.

Οι συναρτήσεις dct2d και idct2d υπολογίζουν τη 2D DCT και την αντίστροφη 2D DCT αντίστοιχα. Η συνάρτηση dct2d υπολογίζει τους συντελεστές DCT για κάθε συχνότητα στην εικόνα, ενώ η idct2d χρησιμοποιεί αυτούς τους συντελεστές για να ανασυνθέσει την εικόνα. Η αλφα συνάρτηση χρησιμοποιείται για την κανονικοποίηση των συντελεστών ανάλογα με τη θέση τους.