ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος ***Αρχές και Εφαρμογές Σημάτων και Συστημάτων***

Αριθμός εργασίας – Τίτλος εργασίας	***Τελική Εργασία***
Ονόματα φοιτητών ομάδας	Κρόιτορ Καταρτζίου Ιωάν Βασιλείου Αλέξιος Ρούτσης Αλέξιος
Αρ. Μητρώου	Π21077 Π21009 Π21145
Ημερομηνία παράδοσης	30/06/2023

Εκφώνηση εργασίας

Η εκφώνηση της εργασίας αφορά τα ερωτήματα Γ'1, Γ'2, Γ'3, Γ'4 των σελίδων 357-361 του συγγράμματος του μαθήματος.

1 Εισαγωγή

Στην τρέχουσα εργασία καλεστήκαμε να επιλύσουμε μία σειρά από ασκήσεις και εφαρμογές, πάνω στην θεωρία των σημάτων και συστημάτων. Ως εργαλείο χρησιμοποιήσαμε το matlab, το οποίο μας παρέχει μία σειρά από βιβλιοθήκες που διευκολύνουν την υλοποίηση των ασκήσεων.

Μια γενική μεθοδολογία που χρησιμοποιήθηκε στα ερωτήματα Γ2 και Γ3 είναι ότι εισάγαγε τον κώδικα σε κλάσεις ώστε να μπορεί να επαναχρησιμοποιηθεί εύκολα, αλλά και επειδή με αυτόν τον τρόπο καθίσταται παραμετροποιήσιμος. Επιπλέον, οι κλάσεις εντός των προγραμμάτων είναι στατικές ώστε να μπορούμε να δημιουργήσουμε (instantiate) αντικείμενα τους χωρίς να τα εκχωρήσουμε σε κάποια μεταβλητή (anonymous object).

Σημειώνουμε ότι όπου αναγράφεται εντός παρενθέσεων διαδικτυακή πηγή, αυτή περιλαμβάνεται στη βιβλιογραφία.

2 Επίλυση και περιγραφή του προγράμματος

Κάθε ένα από τα παρακάτω ερωτήματα αντιστοιχεί στο ανάλογο ερώτημα της εκφώνησης.

Γ'.1

Μας δίνεται το σήμα συνεχούς χρόνου x(t)=cos(10πt) + cos(200πt) + sin(500πt)

Γ'.1.1

Σύμφωνα με το θεώρημα δειγματοληψίας του Whittaker για να επιτευχθεί ανακατασκευή του σήματος x(t) από την ακολουθία περιοδικών δειγμάτων θα πρέπει να ληφθούν δείγματα με συχνότητα μεγαλύτερη ή ίση από 2So, άρα η ελάχιστη συχνότητα είναι 2So, τέτοιο ώστε X(S)=0 για S >= So, όπου X(S) είναι ο μετασχηματισμός Fourier x(t)

|So|>=500 => 1/Ts >= 500

ωmax = 500π => fmax = 250 Hz

άρα fs >= 2fmax = 500Hz, ελάχιστη συχνότητα δειγματοληψίας

Γ'.1.2

Αρχικά, ορίσαμε την συχνότητα δειγματοληψίας(fs), καθώς και το χρονικό διάστημα στο οποίο αυτή θα εφαρμοστεί (t) [-10, 10] με βήμα Δt = 0.001. Ύστερα, αφού εισάγουμε το δοσμένο σήμα, το κάνουμε plot (χρώμα μαύρο).

Γ'.1.2

Ορίζουμε εκ νέου το χρονικό διάστημα για το οποίο θα ανακατασκευαστεί το σήμα t_reconstructed=-10:Ts:10 . Συνεχίζοντας, εφαρμόζουμε τη γραμμική παρεμβολή (linear interpolation), η οποία αποτελεί μία μαθηματική τεχνική για να εκτιμήσουμε τιμές μεταξύ γνωστών σημείων δεδομένων (η μέθοδος βρέθηκε από διαδικτυακή πηγή που αναφέρεται στη βιβλιογραφία). Συγκεκριμένα, η συνάρτηση interp1() στο matlab χρησιμοποιείται για για γραμμική παρεμβολή μονοδιάστατων δεδομένων και λαμβάνει ως είσοδο t, x, t_reconstructed, 'linear' (γραμμική παρεμβολή). Τέλος, κάνουμε plot το αρχικό (χρώμα black) και το ανακατασκευασμένο σήμα (χρώμα blue).

Γ'.1.4

Ακολουθώντας την ίδια ακριβώς διαδικασία με το Γ'1.2, με διαφορά ότι η συχνότητα δειγματοληψίας (fsHigh) ήταν μεγαλύτερη (3000), προκύπτει το αντίστοιχο σήμα (χρώμα cyan).

Γ'.1.5

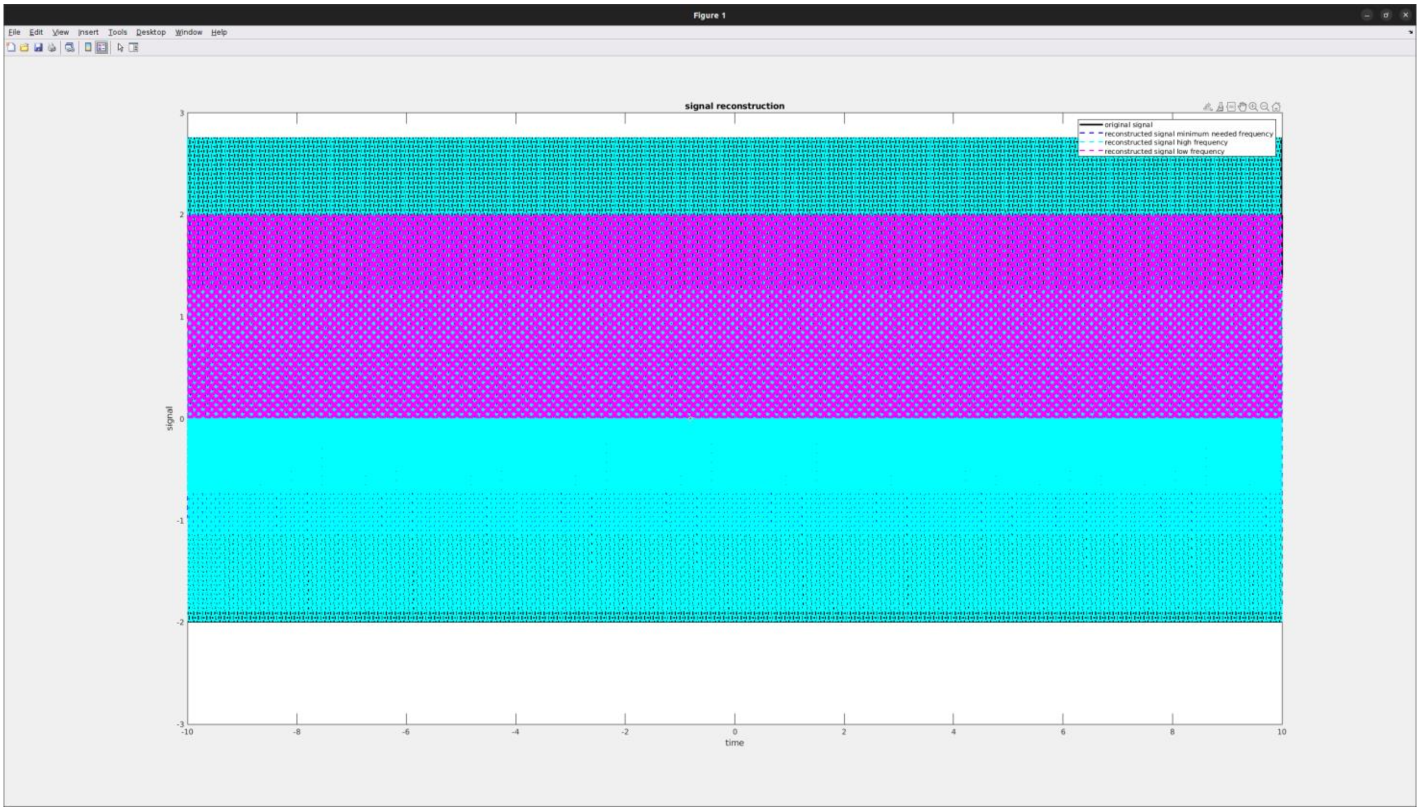
Πράττοντας ομοίως με το Γ'.1.3 με συχνότητα δειγματοληψίας (fsLow) μικρότερη (100), προκύπτει το αντίστοιχο σήμα (χρώμα magenda).

Τα ερωτήματα Γ'.1.2, Γ'.1.3, Γ'.1.4, Γ'.1.5 υλοποιήθηκαν όλα στο ίδιο αρχείο κώδικα για να εμφανίζονται όλα τα σήματα στο ίδιο γράφημα ώστε να είναι ευκολότερη η μεταξύ τους σύγκριση αλλά και για επειδή έτσι αναφέρεται στην εκφώνηση («..υπέρθεσατε (με διαφορετικό χρώμα) το γράφημα του ανακατασκευασμένου σήματος.»). Παρακάτω παραθέτουμε screenshots από την εκτέλεση του προγράμματος.

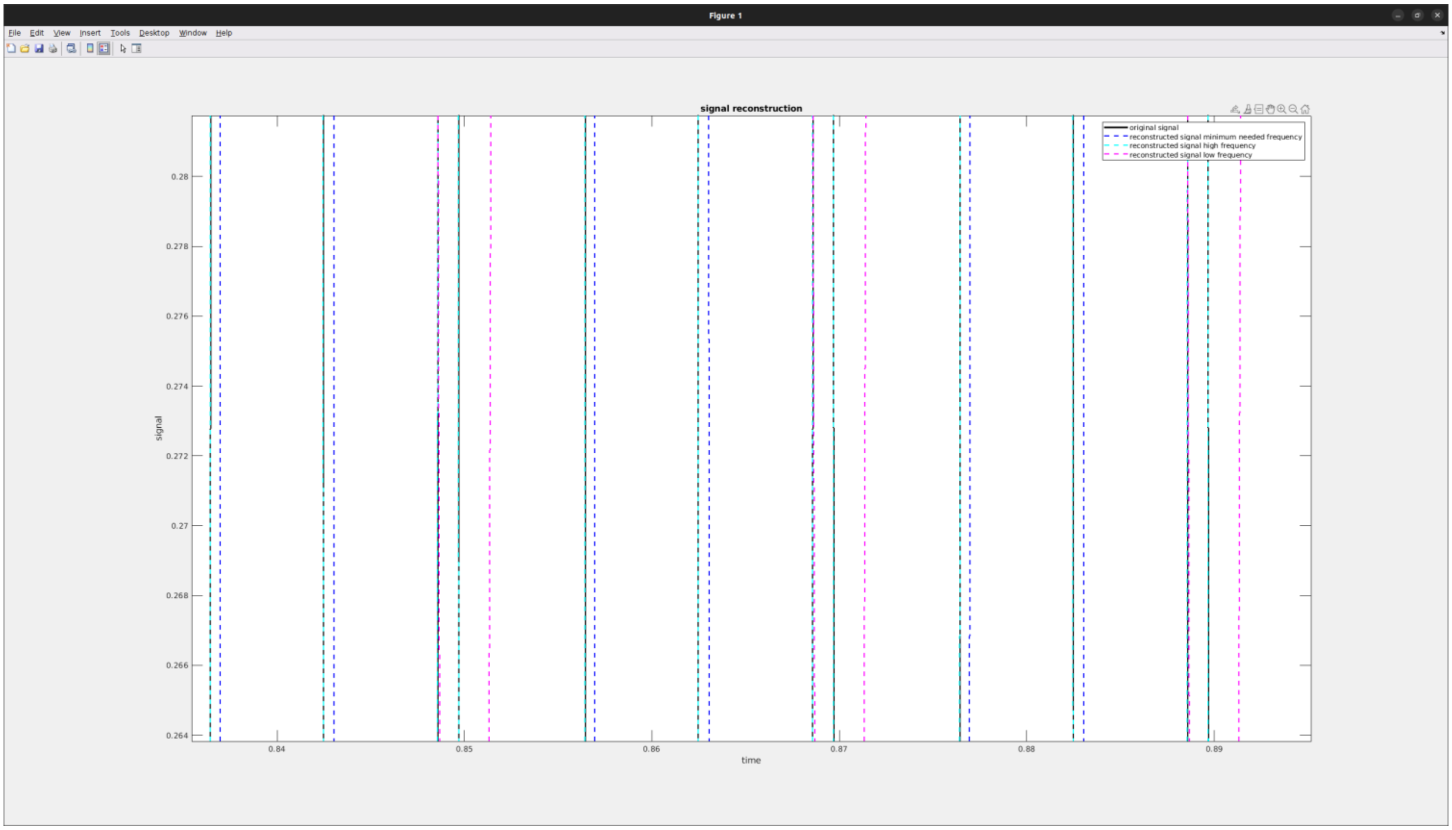
Editor - /home/ioannis/ioannis/university/semester4/matlab/g1_4.m

```matlab
% define the time range and sampling frequency
fs = 500;
fsHigh = 1000;
fsLow = 100;

Ts = 1/fs;
TsHigh = 1/fsHigh;
TsLow = 1/fsLow;

t = -10:0.001:10;

% generate the sampled signal
x = cos(100*pi*t) + cos(200*pi*t) + sin(500*pi*t);

% define the reconstructed time range
t_reconstructed=-10:Ts:10;
t_reconstructedHigh=-10:TsHigh:10;
t_reconstructedLow=-10:TsLow:10;

% perform linear interpolation to reconstruct the signal
x_reconstructed = interp1(t, x, t_reconstructed, 'linear');
x_reconstructedHigh = interp1(t, x, t_reconstructedHigh, 'linear');
x_reconstructedLow = interp1(t, x, t_reconstructedLow, 'linear');

% plot the original and reconstructed signals
figure;
plot(t, x, 'k', 'LineWidth', 2);
hold on;
plot(t_reconstructed, x_reconstructed, 'b--', 'LineWidth', 1.5);
plot(t_reconstructedHigh, x_reconstructedHigh, 'c--', 'LineWidth', 1.5);
plot(t_reconstructedLow, x_reconstructedLow, 'm--', 'LineWidth', 1.5);
xlabel('time');
ylabel('signal');
legend('original signal', 'reconstructed signal sampling frequency', ...
    'reconstructed signal high frequency','reconstructed signal low frequency');
title('signal reconstruction');
```

Εικόνα 1-1. Κώδικας g1_4



Εικόνα 2-2. Σήματα ερωτημάτων Γ'.1.2, Γ'.1.3, Γ'.1.4, Γ'.1.5 σε όλο το χρονικό εύρος



Εικόνα 3-3. Σήματα ερωτημάτων Γ'.1.2, Γ'.1.3, Γ'.1.4, Γ'.1.5 σε μικρό χρονικό εύρος (zoom). Εδώ μπορούμε να παρατηρήσουμε αναλυτικότερα την συμπεριφορά του κάθε ανακατασκευασμένου σήματος σε σχέση με το αρχικό. Η πιστότερη ανακατασκευή αποδίδεται από αυτό με τη μεγαλύτερη συχνότητα (cyan), ενώ η λιγότερο πιστή από αυτό με την χαμηλότερη συχνότητα (magenda). Το σήμα με fs ακριβώς 2fmax (blue) αποδίδει μία ενδιάμεσης (σε σχέση με την υψηλότερη και την χαμηλότερη πιστότητα) ακρίβειας πιστότητας.

Γ'.1.5

Μπορούμε να παρατηρήσουμε πως με την αύξηση της συχνότητας δειγματοληψίας, η ανακατασκευή του σήματος είναι πιο πιστή και πιο ακριβής εφόσον σε μικρότερο χρονικό διάστημα (1/fs > 1/fsHigh) λαμβάνονται περισσότερα δείγματα (fs > fsHigh). Αντίστοιχα, στην περίπτωση της μείωσης της συχνότητας δειγματοληψίας, ανακατασκευή του σήματος θα είναι λιγότερο ακριβής εφόσον σε μαγαλύτερο χρονικό διάστημα (1/fs < 1/fsLow) λαμβάνονται λιγότερα δείγματα (fs > fsHigh). Άρα, η κατανομή των δειγμάτων θα είναι πιο αραιή.

Γ’.2

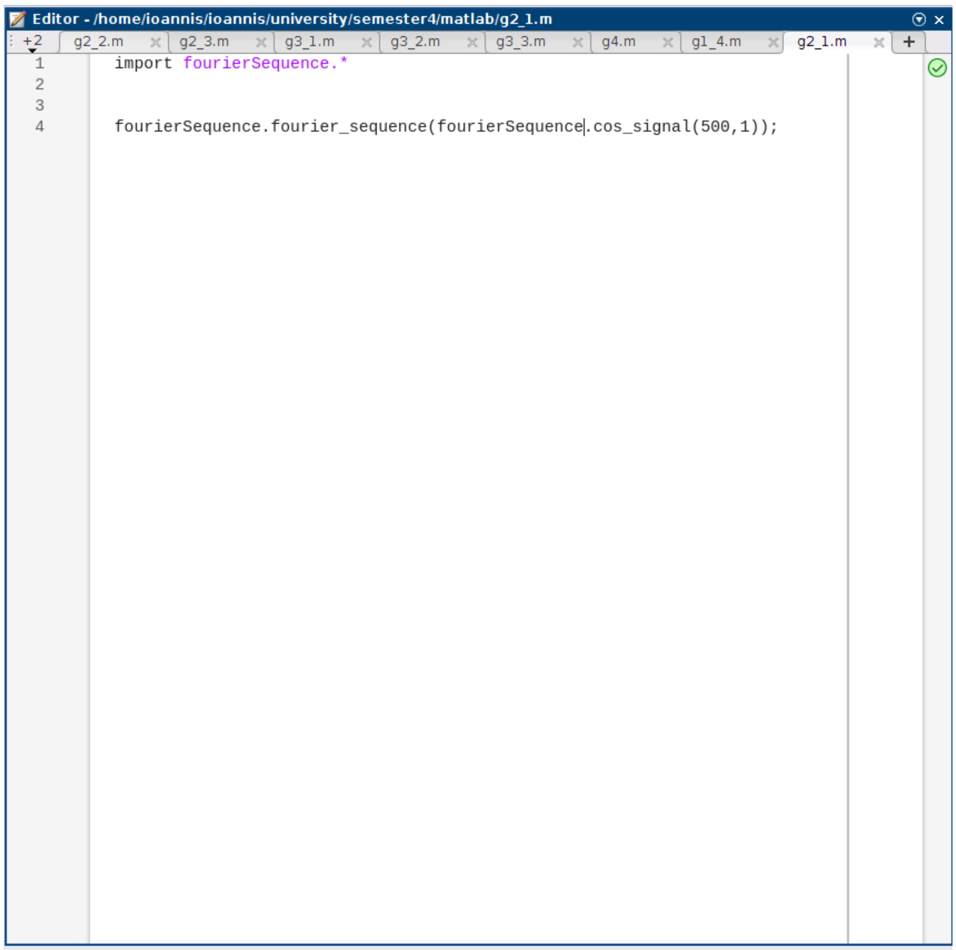
Γ’.2.1

Αρχικά, αποφασίσαμε να ενσωματώσουμε τον κώδικα μας σε κλάσεις για τους λόγους τους οποίους παραθέσαμε στην εισαγωγή. Η κλάση fourierSequence αποτελείται από δύο στατικές μεθόδους, η πρώτη εκ των οποίων είναι η cos_signal που παράγει ένα πεπερασμένο σήμα διακριτού χρόνου. Λαμβάνουμε ως είσοδο την συχνότητα, καθώς και τη διάρκειά του, ενώ παράγουμε και έναν μονοδιάστατο πίνακα που περιλαμβάνει 100 ισαπέχουσες, διακριτές τιμές (linespace). Η δεύτερη αφορά την ακολουθία Fourier στην οποία, αφού ορίσουμε τις κατάλληλες παραμέτρους (fs, T, t), υπολογίζουμε την ακολουθία Fourier μέσω της συνάρτησης fft (Discrete Fourier Transform), που αναπαριστά το σήμα στο πεδίο συχνοτήτων. Επιπλέον, πραγματοποιούμε την διαίρεση ‘X = X / length(signal)’ ώστε να κάνουμε normalize την ακολουθία (ουσιαστικά μειώνουμε τα πλάτη για να εξασφαλίσουμε ότι είναι συνεπή και συγκρίσιμα σε διαφορετικά μήκη σήματος, βήμα απαραίτητο για την περεταίρω ανάλυση). Στη συνέχεια, υπολογίζουμε ένα διάνυσμα συχνοτήτων που αντιστοιχεί στον άξονα συχνοτήτων για το σήμα (ώστε να το κάνουμε plot). Ύστερα, εμφανίζουμε την απόλυτη τιμή του διανύσματος X (απόλυτη τιμή φάσματος), δηλαδή απεικονίζει το μέγεθος (πλάτος) των συχνοτήτων στο σήμα. Έτσι, μπορούμε να εξάγουμε πληροφορίες για τις συχνότητες του σήματος και για την ένταση τους.

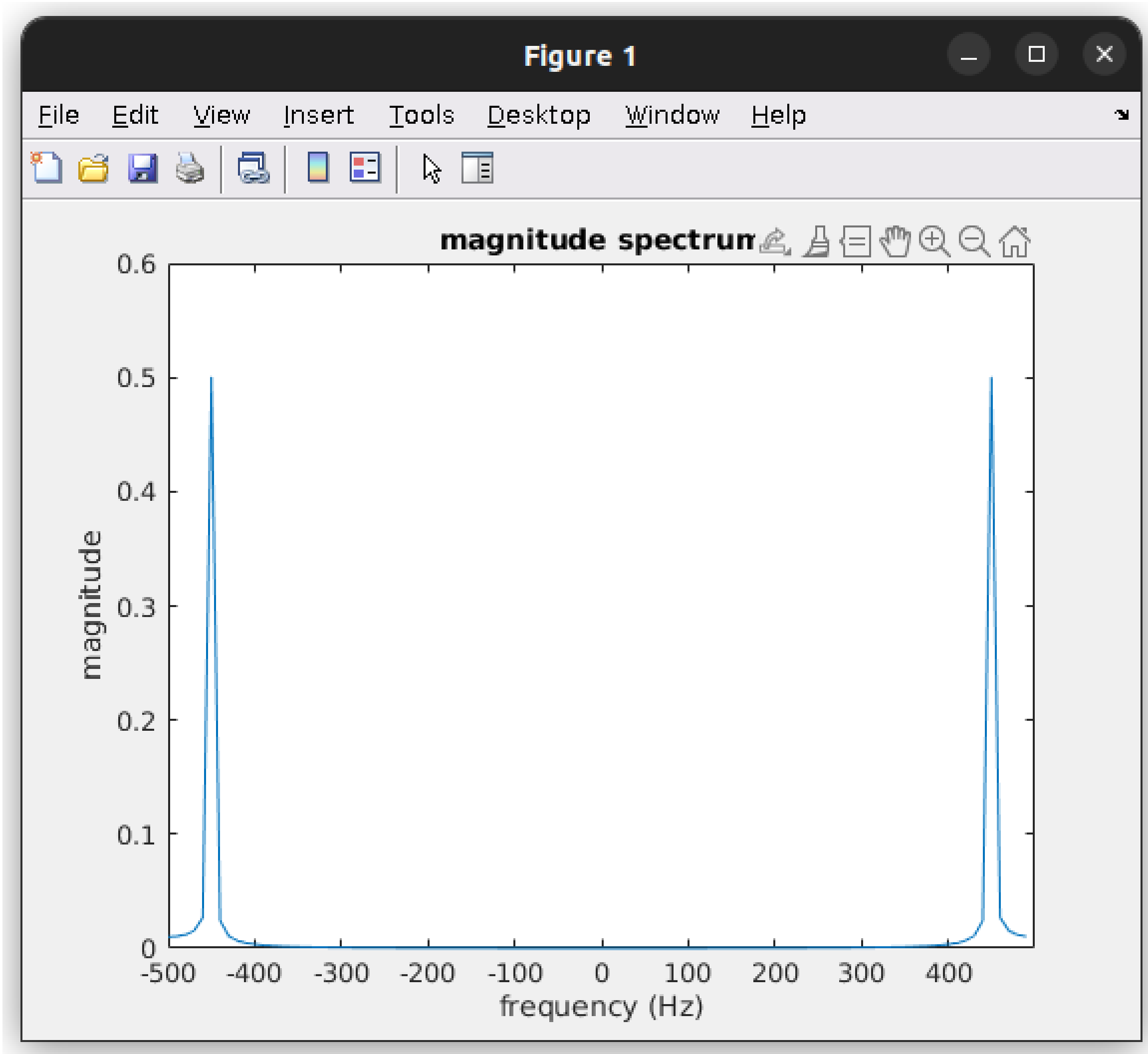
```
classdef fourierSequence
    methods (Static)
        function x = cos_signal(f, T)
            % signal
            fs = 2*f; % sampling frequency (Hz)
            t = linspace(0, T, 100); % 100 time values so that the signal is
            % finite and produces an exact set of values

            % calculate the function
            x = cos(2*pi*f*t);
        end

        function fourier_sequence(signal)
            % define parameters
            T = 1;  % signal duration
            fs = 1000;  % sampling frequency (Hz)
            t = linspace(0, T, length(signal)); % time values;
            disp(t);

            % compute Fourier sequence
            X = fft(signal);
            X = X / length(signal);

            % compute frequency vector
            f = (-fs/2 : fs/length(t) : fs/2 - fs/length(t));

            % plot magnitude spectrum
            plot(f, abs(X));
            xlabel('frequency (Hz)');
            ylabel('magnitude');
            title('magnitude spectrum');
        end
    end
end
```

Εικόνα 2-4. Κώδικας κλάσης fourierSequence

Για να τρέξουμε την συνάρτηση, δημιουργούμε ένα νέο αρχείο, το g2_1 στο οποίο κάνουμε import fourierSequence.* (την κλάση του προηγούμενου αρχείου) και καλούμε την συνάρτηση fourier_sequence() και την εφαρμόζουμε πάνω στο σήμα της συνάρτησης cos_signal() με τις παραμέτρους f = 250 και T = 1.



Εικόνα 2-2. Κώδικας g2_1



Εικόνα 2-3. Το φάσμα συχνοτήτων του μετασχηματισμού Fourier του σήματος cos(2πft) για f = 500, T = 1

```
>> g2_1
  Columns 1 through 14

         0    0.0101    0.0202    0.0303    0.0404    0.0505    0.0606    0.0707    0.0808    0.0909    0.1010    0.1111    0.1212    0.1313

  Columns 15 through 28

    0.1414    0.1515    0.1616    0.1717    0.1818    0.1919    0.2020    0.2121    0.2222    0.2323    0.2424    0.2525    0.2626    0.2727

  Columns 29 through 42

    0.2828    0.2929    0.3030    0.3131    0.3232    0.3333    0.3434    0.3535    0.3636    0.3737    0.3838    0.3939    0.4040    0.4141

  Columns 43 through 56

    0.4242    0.4343    0.4444    0.4545    0.4646    0.4747    0.4848    0.4949    0.5051    0.5152    0.5253    0.5354    0.5455    0.5556

  Columns 57 through 70

    0.5657    0.5758    0.5859    0.5960    0.6061    0.6162    0.6263    0.6364    0.6465    0.6566    0.6667    0.6768    0.6869    0.6970

  Columns 71 through 84

    0.7071    0.7172    0.7273    0.7374    0.7475    0.7576    0.7677    0.7778    0.7879    0.7980    0.8081    0.8182    0.8283    0.8384

  Columns 85 through 98

    0.8485    0.8586    0.8687    0.8788    0.8889    0.8990    0.9091    0.9192    0.9293    0.9394    0.9495    0.9596    0.9697    0.9798

  Columns 99 through 100

    0.9899    1.0000
```

Εικόνα 2-4. Οι διακριτές τιμές του διανύσματος t (100 σε πλήθος)

Γ’.2.2

Ομοίως με το Γ’.2.1, κατασκευάσαμε την κλάση reverseFourierTransform που περιέχει την στατική μέθοδο reverse_fourier_transfrom(coefficients, base_frequency, num_coefficients), με ορίσματα τους συντελεστές της σειράς (διάνυσμα), τη θεμελιώδη συχνότητα και τον επιθυμητό αριθμό συντελεστών στον τύπο αντιστροφής. Αρχικά, βρίσκουμε την συχνότητα δειγματοληψίας (fs) καθώς και την διάρκεια του σήματος (T), αλλά και το διάνυσμα που περιέχει τις διακριτές χρονικές στιγμές (t). Ύστερα, μέσω ενός βρόχου που διατρέχει κατά μήκος όλου του διανύσματος με τους συντελεστές της σειράς, υπολογίζουμε το αντίστοιχο όρο της εκθετικής συνάρτησης X = X + coefficients(k) * exp(1i * 2 * pi * (k-1) * base_frequency (συμβουλευτήκαμε το matlab community). Στη συνέχεια, κάνουμε normalize το σήμα και κάνουμε plot το πραγματικό μέρος του.

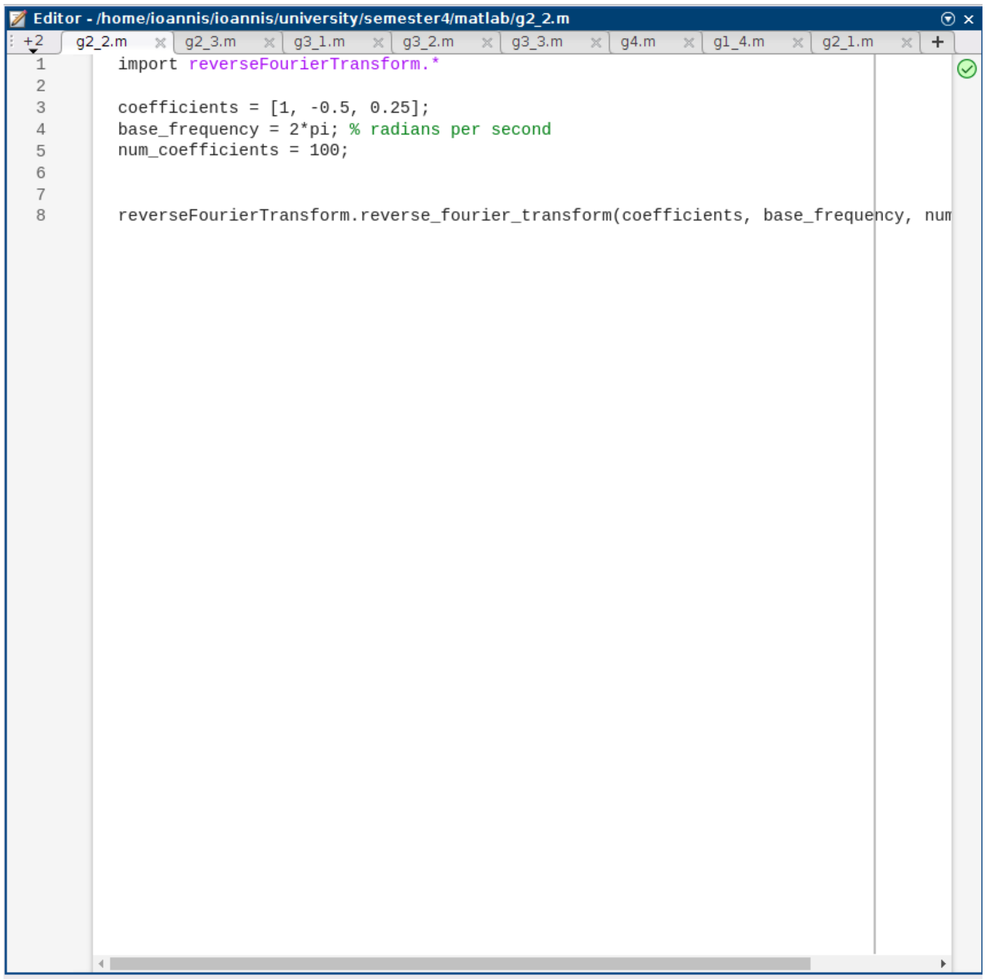
Editor - /home/ioannis/ioannis/university/semester4/matlab/reverseFourierTransform.m

+4 | fourierSequence.m | reverseFourierTransform.m | g2_2.m | g2_3.m | g3_1.m | g3_2.m | +

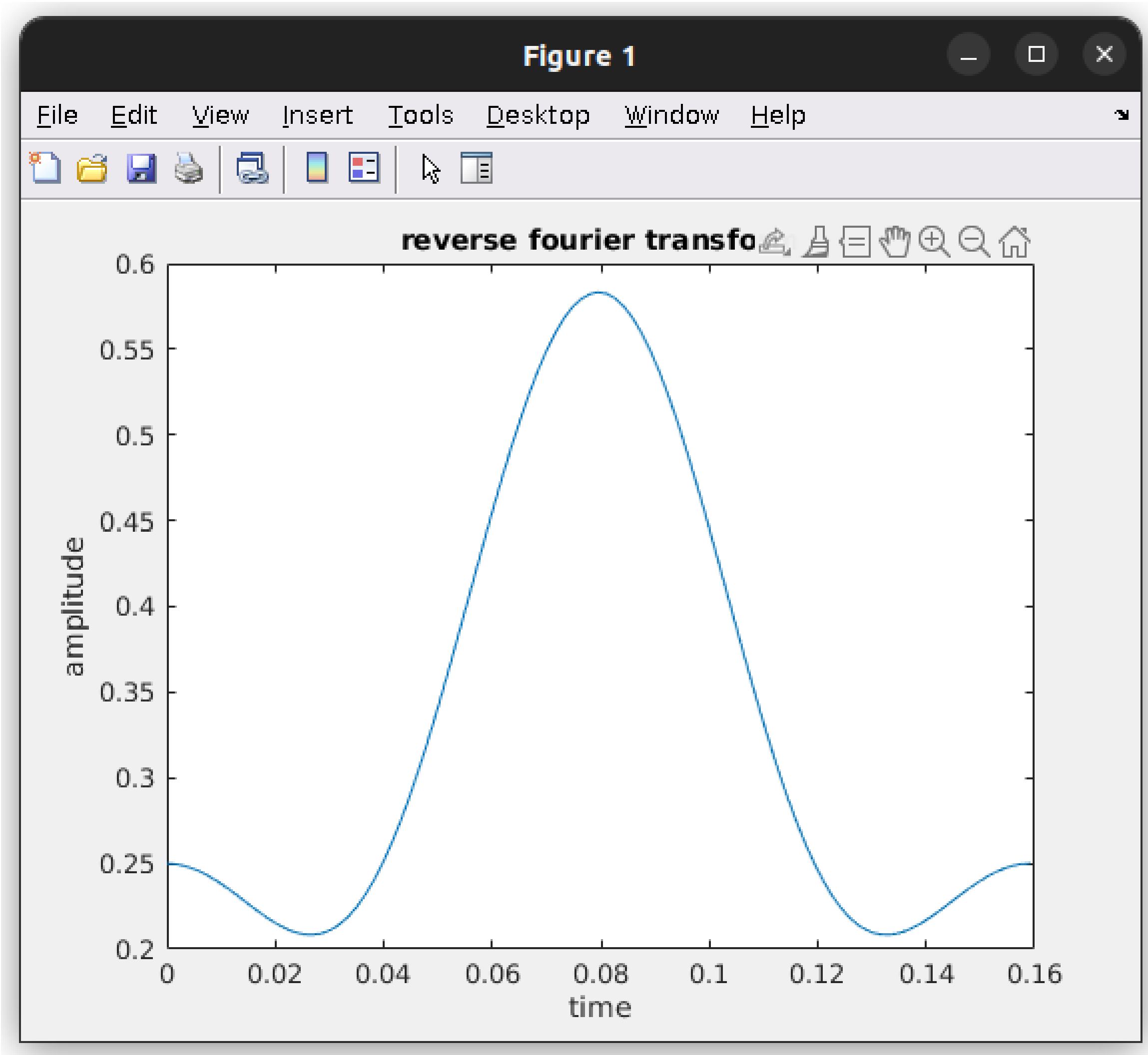
```matlab

classdef reverseFourierTransform
    methods (Static)
        function x = reverse_fourier_transform(coefficients, base_frequency, num_coeff
            % calculate the time values
            fs = base_frequency * num_coefficients; % sampling frequency
            T = num_coefficients / fs; % signal duration
            t = linspace(0, T, num_coefficients+1); % time values

            % calculate the reverse fourier transform
            X = zeros(size(t)); % initialize the transformed signal with a matrix of z
            for k = 1:length(coefficients) %iterates over the fourier coefficients
                X = X + coefficients(k) * exp(1i * 2 * pi * (k-1) * base_frequency * t
                % the coresponding complex exponential term is calculated
                % X represents the reverse fourier transform of the signal
            end

            % scale the transformed signal in order to noramlize
            x = X / length(coefficients);

            % plot the reverse fourier transform
            plot(t, real(x));
            xlabel('time');
            ylabel('amplitude');
            title('reverse fourier transform');
        end
    end
end

```

Εικόνα 2-5. Κώδικας κλάσης reverseFourierTransform

Για να τρέξουμε την συνάρτηση, δημιουργούμε ένα νέο αρχείο, το g2_2 στο οποίο κάνουμε import reverseFourierTransform.* (την κλάση του προηγούμενου αρχείου) και καλούμε την συνάρτηση reverse_fourier_transform() και την εφαρμόζουμε πάνω στο σήμα της συνάρτησης cos_signal() με τις παραμέτρους coefficients = [1, -0.5, 0.25], base_frequency = 2*pi, num_coefficients = 100.



Εικόνα 2-6. Κώδικας reverseFourierTransform



Εικόνα 2-7. Ο αντίστροφος μετασχηματισμός Fourier για τη σειρά με συντελεστές [1, -0.5, 0.25], συχνότητα 2π και 100 συντελεστές

```
Editor - /home/ioannis/ioannis/university/semester4/matlab/g2_2.m
Command Window
  Columns 1 through 7

   0.2500 + 0.0000i   0.2497 - 0.0000i   0.2487 - 0.0002i   0.2471 - 0.0006i   0.2449 - 0.0013i   0.2422 - 0.0025i   0.2391 - 0.0043i

  Columns 8 through 14

   0.2356 - 0.0068i   0.2319 - 0.0099i   0.2281 - 0.0139i   0.2242 - 0.0187i   0.2205 - 0.0244i   0.2171 - 0.0309i   0.2140 - 0.0383i

  Columns 15 through 21

   0.2115 - 0.0466i   0.2096 - 0.0556i   0.2085 - 0.0653i   0.2084 - 0.0757i   0.2093 - 0.0866i   0.2112 - 0.0979i   0.2144 - 0.1095i

  Columns 22 through 28

   0.2189 - 0.1213i   0.2246 - 0.1330i   0.2317 - 0.1446i   0.2402 - 0.1559i   0.2500 - 0.1667i   0.2611 - 0.1768i   0.2735 - 0.1861i

  Columns 29 through 35

   0.2871 - 0.1944i   0.3018 - 0.2016i   0.3174 - 0.2075i   0.3339 - 0.2120i   0.3512 - 0.2150i   0.3690 - 0.2164i   0.3872 - 0.2161i

  Columns 36 through 42

   0.4055 - 0.2141i   0.4240 - 0.2103i   0.4422 - 0.2047i   0.4601 - 0.1973i   0.4774 - 0.1881i   0.4939 - 0.1772i   0.5095 - 0.1647i

  Columns 43 through 49

   0.5240 - 0.1507i   0.5373 - 0.1352i   0.5490 - 0.1184i   0.5593 - 0.1005i   0.5678 - 0.0816i   0.5745 - 0.0619i   0.5794 - 0.0416i

  Columns 50 through 56

   0.5823 - 0.0209i   0.5833 - 0.0000i   0.5823 + 0.0209i   0.5794 + 0.0416i   0.5745 + 0.0619i   0.5678 + 0.0816i   0.5593 + 0.1005i

  Columns 57 through 63

   0.5490 + 0.1184i   0.5373 + 0.1352i   0.5240 + 0.1507i   0.5095 + 0.1647i   0.4939 + 0.1772i   0.4774 + 0.1881i   0.4601 + 0.1973i

  Columns 64 through 70

   0.4422 + 0.2047i   0.4240 + 0.2103i   0.4055 + 0.2141i   0.3872 + 0.2161i   0.3690 + 0.2164i   0.3512 + 0.2150i   0.3339 + 0.2120i

  Columns 71 through 77

   0.3174 + 0.2075i   0.3018 + 0.2016i   0.2871 + 0.1944i   0.2735 + 0.1861i   0.2611 + 0.1768i   0.2500 + 0.1667i   0.2402 + 0.1559i

  Columns 78 through 84

   0.2317 + 0.1446i   0.2246 + 0.1330i   0.2189 + 0.1213i   0.2144 + 0.1095i   0.2112 + 0.0979i   0.2093 + 0.0866i   0.2084 + 0.0757i

  Columns 85 through 91

   0.2085 + 0.0653i   0.2096 + 0.0556i   0.2115 + 0.0466i   0.2140 + 0.0383i   0.2171 + 0.0309i   0.2205 + 0.0244i   0.2242 + 0.0187i

  Columns 92 through 98

fx 0.2281 + 0.0139i   0.2319 + 0.0099i   0.2356 + 0.0068i   0.2391 + 0.0043i   0.2422 + 0.0025i   0.2449 + 0.0013i   0.2471 + 0.0006i
Zoom: 100%   UTF-8   LF
```

Εικόνα 2-8. Οι αντίστροφος μετασχηματισμός Fourier για τη σειρά με συντελεστές [1, -0.5, 0.25], συχνότητα 2π και 100 συντελεστές

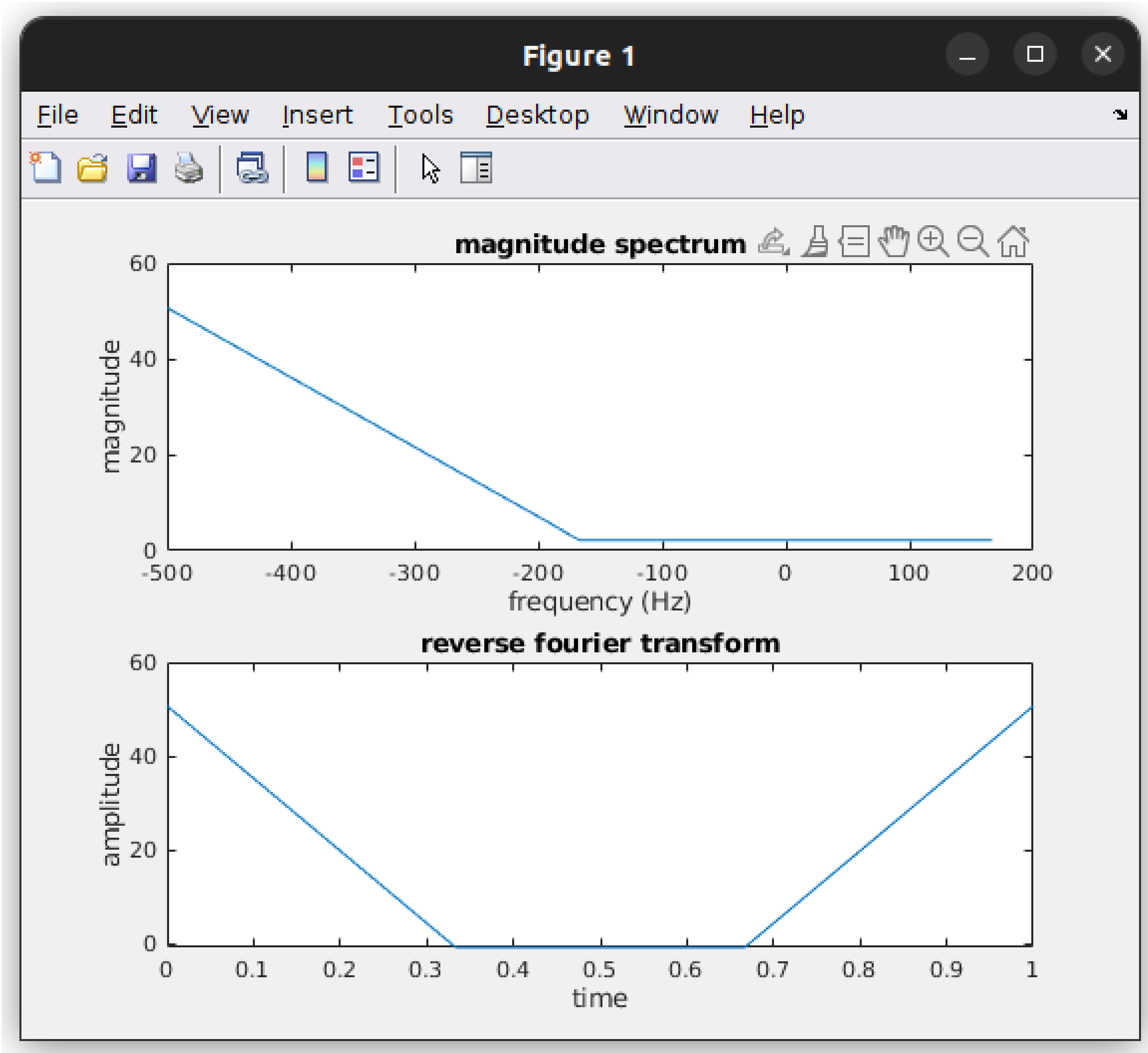
Γ’.2.3

Σε αυτό το ερώτημα καλούμαστε να εφαρμόσουμε τις συναρτήσεις των ερωτημάτων Γ’.2.1, Γ’.2.2, με σήμα εισόδου τον χαρακτήρα ASCII που αντιστοιχεί στο πρώτο γράμμα του επωνύμου (επιλέξαμε το ‘k’). Οπότε απλώς καλούμε τις συναρτήσεις με όρισμα το συγκεκριμένο χαρακτήρα όπως φαίνεται παρακάτω.

Editor - /home/ioannis/ioannis/university/semester4/matlab/g2_3.m

```matlab
import fourierSequence.*
import reverseFourierTransform.*

% convert 'k' to its ASCII value
k_ascii = '107';
k_ascii_code = double(k_ascii);
x_ascii = k_ascii_code;
%X_ascii = fft(x_ascii);
%subplot(2,1,2);
%plot(X_ascii);

subplot(2,1,1);
fourierSequence.fourier_sequence(x_ascii);

subplot(2,1,2);
reverseFourierTransform.reverse_fourier_transform(x_ascii,1,length(x_ascii));

hold on;
```

Εικόνα 2-9. Κώδικας g2_3



Εικόνα 2-10. Ο μετασχηματισμός (πάνω) και ο αντίστροφος μετασχηματισμός Fourier του ‘k’ ASCII χαρατήρα.

Γ'.3

Αρχικά σκεφτήκαμε να δημιουργήσουμε την στατική κλάση notes_function, η οποία περιέχει την συνάρτηση generate_note_waveform(selected_note, duration, fs, volume), με ορίσματα την νότα που θέλουμε να συνθέσουμε, την διάρκεια, την συχνότητα δειγματοληψίας αλλά και την ένταση. Το διάνυσμα του χρόνου έχει βήμα Ts = 1/ fs. Μετά, σύμφωνα με την νότα που λήφθηκε ως όρισμα, την αναθέτουμε σε μία συγκεκριμένη συχνότητα (μέσω ενός switch statement) όπως περιγράφεται από την εκφώνηση. Τέλος, παράγεται η επιθυμητή νότα μέσω του ημιτονοειδούς σήματος sin(2πft) πολλαπλασιασμένο με την ένταση που δόθηκε ως όρισμα.

Editor - /home/ioannis/ioannis/university/semester4/matlab/notes_function.m

```
classdef notes_function
    methods (Static)
        function note = generate_note_waveform(selected_note, duration, fs,volume)
        % takes four input arguments
        t = 0:(1/fs):duration; % time vector

            switch selected_note
                case 'A ', f = 220;
                case 'A#', f = 220*2^(1/12);
                case 'B ', f = 220*2^(2/12);
                case 'C ', f = 220*2^(3/12);
                case 'C#', f = 220*2^(4/12);
                case 'D ', f = 220*2^(5/12);
                case 'D#', f = 220*2^(6/12);
                case 'E ', f = 220*2^(7/12);
                case 'F ', f = 220*2^(8/12);
                case 'F#', f = 220*2^(9/12);
                case 'G ', f = 220*2^(10/12);
                case 'G#', f = 220*2^(11/12);
            end

        note = volume*sin(2*pi*f*t); % the selected note's waveform sin

        end
    end
end
```

Εικόνα 3-1. Κώδικας κλάσης notes_function

Γ'.3.1

Σε αυτό το ερώτημα, αφού κάναμε import notes_function.*, ορίσαμε fs = 8000 και dt = 0.4 (παύσεις μεταξύ των νοτών), καθώς και το διάνυσμα των νοτών που φαίνεται στην παρακάτω εικόνα (που είναι μια προσέγγιση της 41ης συμφωνίας 'Jupiter' του Μότσαρντ). Μετά, διατρέχοντας μία-μία της νότες του πίνακα, παράγουμε τον αντίστοιχο ήχο με την συνάρτηση generate_note_waveform() (που περιγράφηκε παραπάνω), η οποία δίνεται ως όρισμα στην συνάρτηση sound() του matlab (βρέθηκε σε διαδικτυακή πηγή). Επιπλέον ενδεικτικά προσθέσαμε και έναν πίνακα με τυχαίες νότες (notes)

Editor - /home/ioannis/ioannis/university/semester4/matlab/g3_1.m

```
import notes_function.*

fs = 8000;     % define sampling frequency
dt = 0.4;      % pauses between the notes (signals)
symphony = [
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '
];

notes = ['C ';'C#';'D ';'D#';'E ';'F ';'F#';'G ';'G#';'A ';'A#';'B '];

for i = 1:1:length(symphony) % iterate through each element of notes array
        soundsc(generate_note_waveform(symphony(i,1:end),dt,fs,1));
        soundsc(0);
        % here we use the built in soundsc() function of matlab in order to
        % generate the sounds
end

for i = 1:1:length(notes) % iterate through each element of notes array
        soundsc(generate_note_waveform(notes(i,1:end),dt,fs,1));
        soundsc(0);
        % here we use the built in soundsc() function of matlab in order to
        % generate the sounds
end
```

Εικόνα 3-2. Κώδικας g3_1

Γ'.3.2

Εργαζόμενοι όμοια με το προηγούμενο ερώτημα αλλάζοντας τη διάρκεια του σήματος πολλαπλασιάζοντας τη διάρκεια με το i σε κάθε διάτρεξη, ώστε να επιτύχουμε την ψηφιακή ολίσθηση προς τα άνω και προς τα κάτω κατά μία οκτάβα.

Editor - /home/ioannis/ioannis/university/semester4/matlab/g3_2.m

+2 | g2_2.m | g2_3.m | g3_1.m | g3_2.m | g3_3.m | g4.m | g1_4.m | g2_1.m | notes_function.m

```matlab
import notes_function.*

fs = 8000;     % sampling frequency
dt = 0.1;      % pauses between the signal

symphony = [
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '
];

notes = ['C ';'C#';'D ';'D#';'E ';'F ';'F#';'G ';'G#';'A ';'A#';'B '];

for i = 1:1:length(symphony)
        soundsc(generate_note_waveform(symphony(i,1:end),i*dt,fs,1));
        % the i*dt is the shifting at each iteration
        soundsc(0);
end

for i = 1:1:length(notes)
        soundsc(generate_note_waveform(notes(i,1:end),i*dt,fs,1));
        % the i*dt is the shifting at each iteration
        soundsc(0);
end
```

Εικόνα 3-3. Κώδικας g3_2

Γ'.3.2

Ομοίως με τα προηγούμενα δύο ερωτήματα, μειώνουμε σε κάθε διάτρεξη την ένταση, κάνοντας χρήση του i (μετρητής) στον τύπο (13-i)/10.

Editor - /home/ioannis/ioannis/university/semester4/matlab/g3_3.m

```matlab
import notes_function.*

fs = 8000;      % sampling frequency
dt = 0.2;       % pauses between signals

symphony = [
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'G '; 'C '; 'D '; 'E ';...
    'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'F#'; 'G#'; 'B '; 'C#'; 'D#'; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E '; 'E ';...
    'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '; 'G '
];

notes = ['C ';'C#';'D ';'D#';'E ';'F ';'F#';'G ';'G#';'A ';'A#';'B '];

for i = 1:1:length(symphony)
        soundsc(generate_note_waveform(symphony(i,1:end),dt,fs,(13-i)/10));
        soundsc(0);
        % each time the volume is decreased using the i component of
        % the iteration
end

for i = 1:1:length(notes)
        soundsc(generate_note_waveform(notes(i,1:end),dt,fs,(13-i)/10));
        soundsc(0);
        % each time the volume is decreased using the i component of
        % the iteration
end
```

Εικόνα 3-4. Κώδικας g3_3

Γ’.4

Αφού ορίσαμε μία εικόνα, καθώς και τον αριθμό των συντελεστών του μετασχηματισμού που θα συγκρατηθούν κατά τη διάρκεια της συμπίεσης, και στη συνέχεια θα εξάγουμε την εικόνα ως έναν πίνακα από pixels. Μετά, ελέγχουμε ένα η εικόνα είναι σε rgb ή grayscale με τη χρήση της συνάρτησης size() (κάνουμε discard τις δύο πρώτες τιμές της με τη χρήση του ~). Εφόσον η εικόνα είναι rgb (p=3) καλείται η συνάρτηση rgb2gray ώστε να την μετατρέψει σε grayscale. Συνεχίζοντας, μετατρέπουμε από integer σε double τον πίνακα με την εικόνα και εφαρμόζουμε τον συνημιτονικό μετασχηματισμό δισδιάστατης εικόνας με χρήση της συνάρτησης dct2 (από διαδικτυακή πηγή), διαδικασία η οποία θα επιστρέψει τον πίνακα στο πεδίο των συχνοτήτων. Επιπροσθέτως, τετραγωνίζουμε το κάθε στοιχείο του πίνακα ώστε να βρούμε τη ισχύ της κάθε συχνότητας, ενώ στη συνέχεια μετατρέπουμε τον πίνακα σε πίνακα στήλη, ώστε να τα ταξινομήσουμε τα στοιχεία του κατά αύξουσα σειρά στο επόμενο βήμα. Ύστερα, αντιστρέφουμε την σειρά των στοιχείων του πίνακα ώστε να έχουν προτεραιότητα συντελεστές με μεγαλύτερες συχνότητες, πράγμα χρήσιμο για την επακόλουθη συμπίεση.

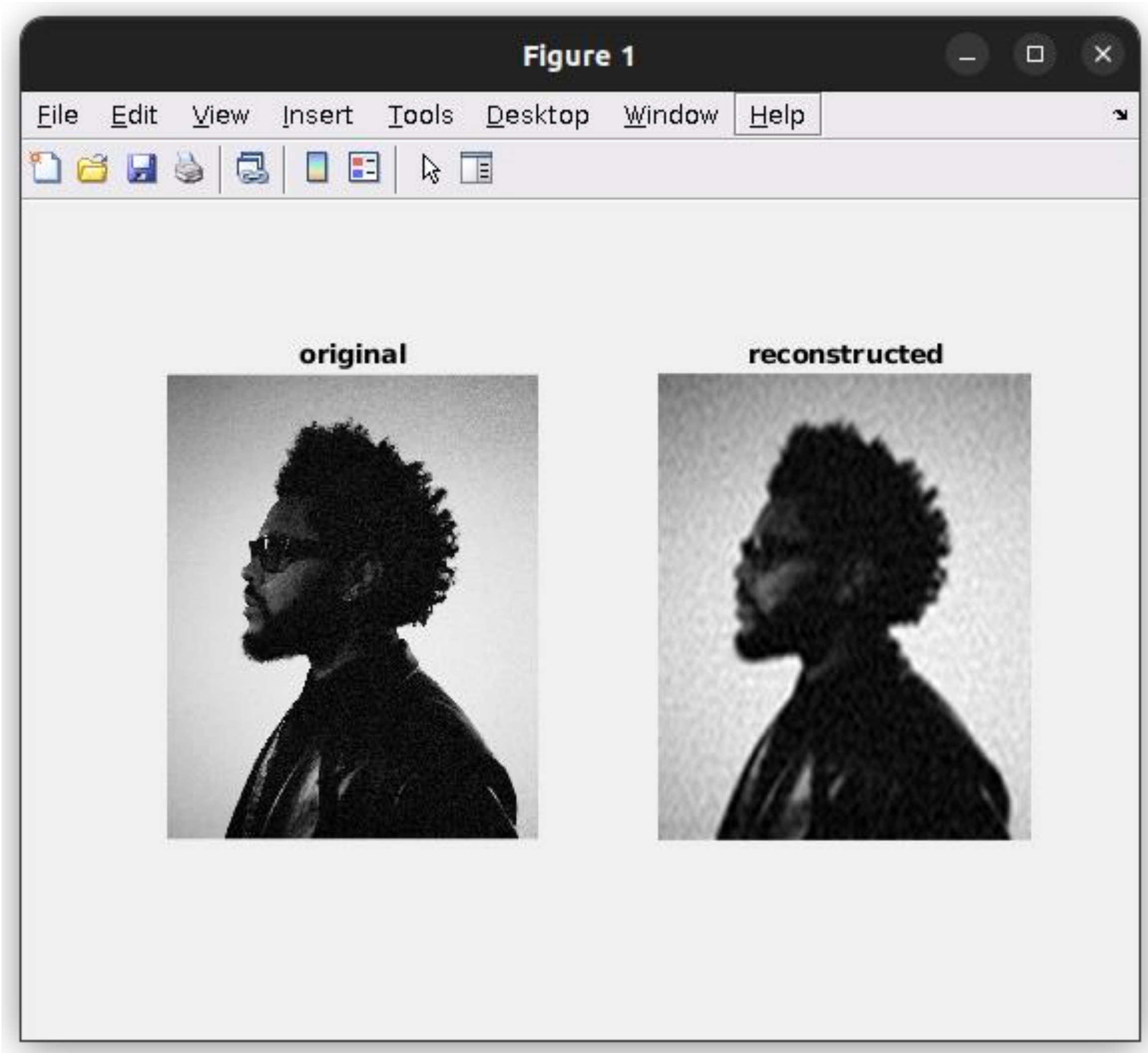
Για την συμπίεση, αφού αρχικοποιήσαμε έναν πίνακα compressed_dft με μηδενικά (μήκους όσο το μήκος του πίνακα της εικόνας), στην κάθε θέση του οποίου αντιστοιχήσαμε στην συνέχεια την αντίστοιχη θέση του πίνακα dft (που περιέχει τον μετασχηματισμό της κάθε τιμής του πίνακα που περιέχει την εικόνα), κρατώντας μόνο τις υψηλές συχνότητες του και μηδενίζοντας τις υπόλοιπες. Ύστερα, ανακατασκευάζουμε τον συμπιεσμένο πίνακα μέσω της idct2 (αντίστροφος δισδιάστατος συνημιτονικός μετασχηματισμός, από διαδικτυακή πηγή) και εισάγουμε το αποτέλεσμα ως όρισμα στην συνάρτηση uint8, η οποία μας εξασφαλίζει ότι οι τιμές των pixel βρίσκονται εντός του εύρους [0, 255]. Τέλος, αποθηκεύουμε την ανακατασκευασμένη εικόνα, ενώ την εμφανίζουνε μαζί με την αρχική

Editor - /home/ioannis/ioannis/university/semester4/matlab/g4.m

+4 | g3_1.m | g3_2.m | g3_3.m | g4.m | g1_4.m | g2_1.m | notes_function.m | +

```matlab
% in the code below the ~ is used as a placeholder to discard some
% values from being assigned

image = 'weeknd.jpeg';
num_coeff = 2000;

array = imread(image);
[~, ~, p] = size(array);      % we check if the image is gray scale or rgb

if p == 3
    array = rgb2gray(array);  % convert the rgb image to grayscale
end

dbl = double(array);         % convert the array with the grayscale image
                             % into double data type

dft = dct2(dbl);             % perform the 2 dimensional discrete cosine
                             % transform which returns frequency

square2 = (dft).^2;          % claculate the squared magnitude of each element in
                             % dft array

square2 = square2(:);        % reshape the array into a cloumn vector so
                             % that it will be 1 dimesion

[~,index] = sort(square2);    % sort the elements in ascending order
                             % and returns the sorted indeces

index = flipud(index);       % flip the order of elements in order to
                             % prioritize coefficients with larger
                             % magnitudes at beggining of the array

compressed_dft = zeros(size(dbl));   % initialize a matrix of zeros

for i = 1:num_coeff
    compressed_dft(index(i)) = dft(index(i));
    % assign the dct coefficient in the index(i) position in the dft matrix
end

output = idct2(compressed_dft); % perform inverse dct on the comprssed
output = uint8(output); % convert to uint8 data type to ensure that the
                        % pixel values of the reconstructed image are [0, 255]

imwrite(output, 'weeknd_compressed.jpg'); % save the recontructed image
subplot 121; imshow(array); title('original');
subplot 122; imshow(output); title('reconstructed');
```

Εικόνα 4-5. Κώδικας g4



Εικόνα 4-6. Αρχική (αριστερά) και ανακατασκευασμένη εικόνα (δεξιά)

3 Βιβλιογραφικές Πηγές

Στο τέλος της εργασίας θα πρέπει ***<u>να περιλάβετε οπωσδήποτε</u>****, όλες τις αναφορές των βιβλιογραφικών πηγών που χρησιμοποιήσατε για τη λύση του προβλήματος (βιβλία, ιστοσελίδες κτλ). Ακολουθεί ενδεικτικό παράδειγμα.*

1. Explained: Linear Interpolation [Math] - https://www.youtube.com/watch?v=Cvc-XalN_kk
2. How to Generate Program & Plot Complex Exponential Sequence in Matlab #56 - https://www.youtube.com/watch?v=00lNv_zTqY4
3. sound() matlab - https://www.mathworks.com/help/matlab/ref/sound.html
4. dst2 2-D discrete sine transform - https://www.mathworks.com/matlabcentral/fileexchange/49572-dst2-2-d-discrete-sine-transform
5. idct2 - https://www.mathworks.com/help/images/ref/idct2.html
6. Image Processing Toolbox