A blue and white logo

Description automatically generated

AIN SHAMS UNIVERISTY FACULTY OF ENGINEERING i-CREDIT HOURS ENGINEERING PROGRAMS COMPUTER ENGINEERING AND SOFTWARE SYSTEMS PROGRAM

ECE 251: Signals and Systems Fundamentals

Project Report - Fall 2024

Presented By:

Omar Mohamed Mostafa 22P0197

Ahmed Abbady Mohamed 22P0308

Ahmed Wael Raafat 22P0221

Ezzeldin ismail 22p0141

Anas Mansour 22u0005

Table of Contents

[Abstract 3](#_Toc186141272)

[Introduction 3](#_Toc186141273)

[Project 4](#_Toc186141274)

[Step 1: 4](#_Toc186141275)

[Step 2: 4](#_Toc186141276)

[Step 3: 5](#_Toc186141277)

[Step 4: 6](#_Toc186141278)

[Step 5 + 6: 7](#_Toc186141279)

[Step 7: 8](#_Toc186141280)

[Step 8: 9](#_Toc186141281)

[Step 9: 9](#_Toc186141282)

[Step 10 + 11: 10](#_Toc186141283)

[Step 12: 11](#_Toc186141284)

[Step 13: 11](#_Toc186141285)

[Step 14 + 15: 12](#_Toc186141286)

[Step 16: 12](#_Toc186141287)

[Step 17 + 18: 13](#_Toc186141288)

[Step 19 + 20: 14](#_Toc186141289)

[Step 21: 14](#_Toc186141290)

[Step 22: 15](#_Toc186141291)

[Step 23+24: 16](#_Toc186141292)

[Step 25: 16](#_Toc186141293)

[Conclusion: 17](#_Toc186141294)

|  |  |
| --- | --- |
| Omar Mohamed Mostafa 22p0197 | Steps (1-5) |
| Ahmed Abbady Mohamed 22P0308 | Steps (6-10) |
| Ahmed Wael Raafat 22P0221 | Step(11-15) |
| Ezzeldin ismail 22p0141 | Steps(16-20) |
| Anas Mansour 22u0005 | Steps(21-25) |

# Contribution Table

# Abstract

This project aims to give a practical approach to principles taught in the “ECE 251: Signals and Systems Fundamentals” course in GNU Octave (MATLAB) with a more computation approach allowing for illustration of analyzes of signals in time domain, frequency domain with figures, and filtering of those signals to demonstrate Butterworth filters (low and high) application for frequency separation.

# Introduction

This project focuses on the generation, analysis, and filtering of a signal composed of four distinct frequency components (500 Hz, 1000 Hz, 1500 Hz, and 2000 Hz). The signal is sampled at 10 kHz (Fs) and processed using various digital signal processing techniques.

Key objectives include:

1. Signal generation and analysis in both time and frequency domains
2. Implementation of Butterworth filters for frequency separation
3. Energy analysis and verification of Parseval's theorem
4. Practical application through audio file generation and visualization

# Project

A white background with black text

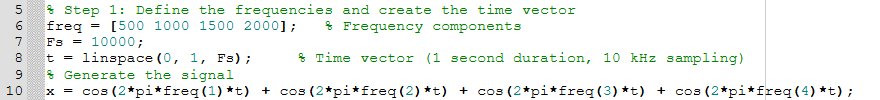
Description automatically generated

* This allows compiler to clear Command Window and variables to allow for clean compilation.
* In command window a command of “pkg load signal” is needed on old version of matlab or any version of GNU Octave. (if error is given write “pkg install -forge signals) which downloads necessary libraries for some functions required in this project)
* Headphone warning before listening to any .wav files.

## Step 1:

A white background with black text

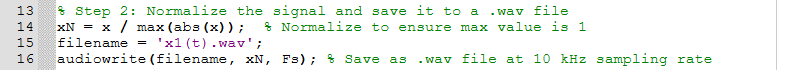
Description automatically generated



* A freq array is made which stores all the required frequencies indicated in the project requirements which is then each frequency in array is retrieved to each respective cosine functions to then sum up the 4 cosine functions.
* Using a sample rate of 10KHz (Fs), a vector of t is created with 10000 values made from 0 to 1, which is them used in the summation of the 4 cosine functions to create signal x(t) which is stored in a vector (x) which takes sum with each t.

## Step 2:





First normalization of signal to ensure maximum amplitude is 1 as .wav files require values between -1 and 1 when saving an audio file, then a file with the name of “x1(t).wav” is made as a reservation for audio which is then wrote using audiowrite taking filename (destination = “x1(t).wav”), xN (normalized signal) and sampling rate as arguments (which is required in audiowrite() as it tells audio player how many samples to play per second, and without it the player wouldn’t know the correct playback speed).



## Step 3:



A white background with black text

Description automatically generated

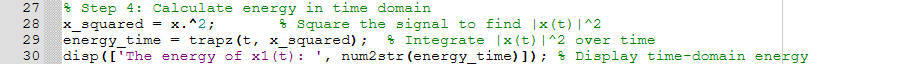
Figure is used to create a new window which then contains a graph with t being the x-axis and x (summation of the 4 cosine functions) being the y-axis with it being generated using plot() function. Labels on x-axis and y-axis are made using xlabel and ylabel , and title is made to describe the figure all for clarity of figure.

A screen shot of a graph

Description automatically generated

## Step 4:





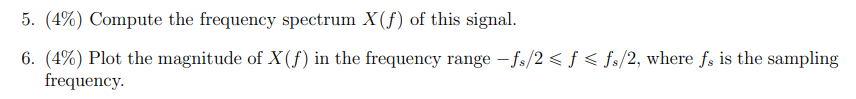
A black background with white text

Description automatically generated

* With consideration of this formula, firstly x is squared and stored in x\_squared which is a vector taking the square of each sum in the vector x and to square every element a “.^2” is used for code clarity, then trapz calculates the area under the signal curve using trapezoidal method which approximates the area by connecting points with straight lines and summing the areas of resulting trapezoids, the result is stored in a variable (energy\_time) that is then displayed in command window after casting it into a string.



## Step 5 + 6:



A screenshot of a computer program

Description automatically generated

* fft() function (Fast Fourier Transform) computes the fourier transform of x converting it from a time domain to frequency domain stored in a vector of the same size (X\_f) (size = 10000).
* A mathematical equation with numbers and symbols

  Description automatically generatedBy default FFT puts zero frequency at start of array, but theoretically there are negative frequencies as well as:
* There is a negative counterpart in cosine’s euler identity which would have negative frequency, if frequencies start from 0 plot wouldn’t be accurate as there would be no representation of negative frequencies, so to solve this ffshift() is used to shift the zero frequency to the center which helps visualize symmetry of frequencies.
* Lastly when plotting frequency spectrum x axis is the frequency and with constraint given in requirement. We need to create frequency axis (f = Fs\*(-N/2: N/2-1)/N) , N being the length of x which is used for frequency resolution (Δf=Fs/N) as the spacing between frequency points is determined by number of samples (N), so dividing by N ensures correct scaling of frequency axis. N is also used in computation of X[k] (X\_f\_magnitude) as in MATLAB theoretically even if signal is CT (continuous time), as results are all stored and plotted in points therefore it is turned into a discrete signal. So DFT is performed with fft, dividing by N ensures amplitude of signal in frequency domain is consistent with the time domain signal, so that magnitude is accurate with Parseval’s theorem:

A black background with white text

Description automatically generated

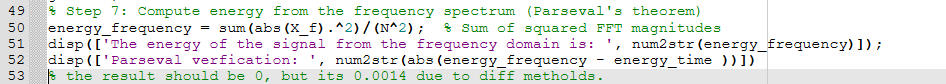
* Then a plot is made similar to step 3 but this time f being the x-axis and magnitude of x\_f (X[K]) as y-axis and f as x axis with xlim limiting the x axis values represented in plot from -Fs/2 to Fs/2

A screenshot of a computer

Description automatically generated

## Step 7:





To compute the energy of frequency domain signal, Parseval’s theorem is used as mentioned in step 5+6 and dividing by N^2 is necessary as without it the frequency domain energy would be scaled incorrectly relative to time domain energy then for debugging the value is displayed in command window, then to verify Parseval’s theorem the difference between energy in both domains should be very close to 0 with the difference being in numerical method.

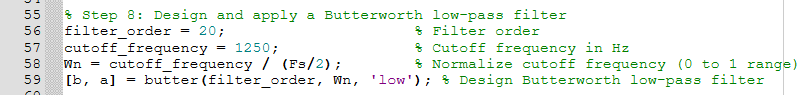
A black and white text

Description automatically generated

As value is close to 0 we can verify that Parseval theorem is correct, and frequency and time domain signals are consistent with each other with correct scaling.

## Step 8:





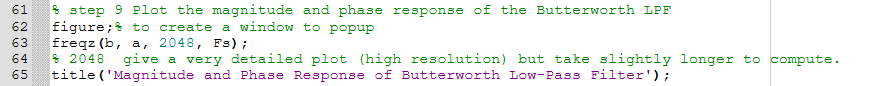
The butter function takes filter order, (Wn) cutoff frequency (which has to be between 0 and 1) , and filter type (‘low’, ‘high’, ‘bandpass’, or ‘bandstop’).

Constraints defined in requirements were put in separate variables (filter\_order), (cutoff\_frequency) then in Wn it Is Normalized to scale between 0 and 1 and in our case Wn is 0.25. A low-pass Butterworth filter is used to allow signal with frequencies below 1250 to pass through while reducing amplitude of frequencies above 1250 which removes them from the signal. b stores the numerator coefficients of the filter transfer function, while a A mathematical equation with numbers

Description automatically generatedstores denominator coefficients of the filter transfer function.

## Step 9:



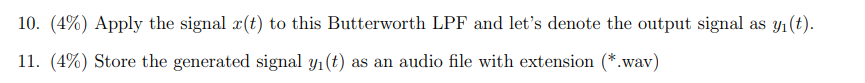


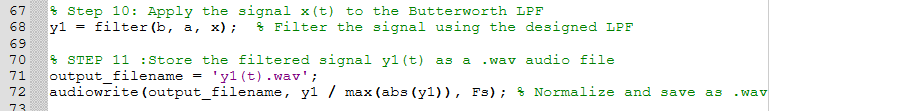
As mentioned before, a figure is made to create a window and a title is used for clarity, this time instead of plot we use freq() which is used to compute frequency response of filter which takes usually 3 arguments but in this case we use 4 arguments which is b and a both from butter function, 2048 which is the number of points to use in sampling the frequency domain (as 2^n number increases , a more detailed frequency response is produced but as compute time cost), and lastly Fs which is the sampling frequency which is optional (syntax wise).

A screenshot of a computer

Description automatically generated

## Step 10 + 11:





filter() function takes 3 arguments (b as defined in step 8) , (a as defined in step 8), and lastly x which is our signal in time domain, this filtered signal is stored is y1 which is a vector.

Then similar to step 2 an audio file is generated and stored in y1(t).wav



## Step 12:

A close up of a word

Description automatically generated

A white background with blue text

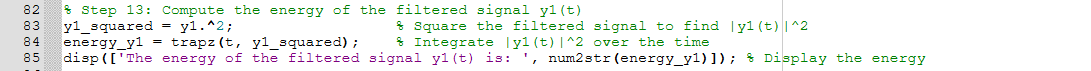
Description automatically generated

A screen shot of a computer

Description automatically generatedAs previously mentioned, here we used plot with t being the x-axis and y1 (filtered signal) as y-axis.

## Step 13:

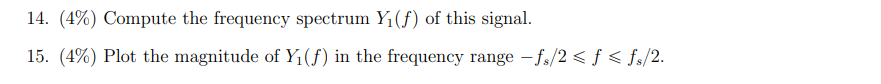




Similar to step 4, energy of y1(t) was computed.



## Step 14 + 15:



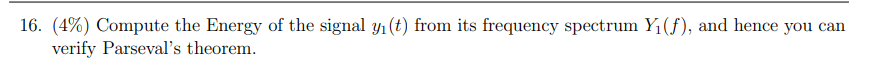
A computer screen shot of a program

Description automatically generated

A screen shot of a graph

Description automatically generatedSimilar to step 5 + 6 , Fourier transform of y1(t) was computed (Y1\_f)and after correct scaling it was used to plot magnitude of Y1\_f\_magnitude after it was computed with it being the y axis and f which was previously declared in step 5 + 6 as constraints are the same, being the x-axis.

## Step 16:



A close-up of a computer screen

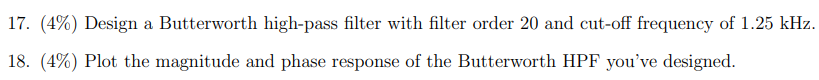
Description automatically generated

Similar to step 7 , energy in frequency domain is done using Parseval’s theorem and then after correct scaling the difference between energy in time and frequency domain is computed to verify Parseval's theorem.



9.8143e-05 is equivalent to 0.000098143 which means Parseval’s theorem is valid and the difference is due to using different numerical methods.

## Step 17 + 18:



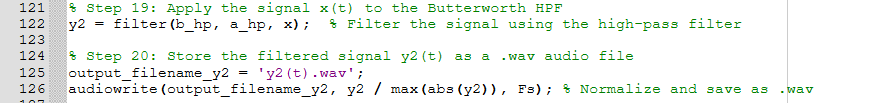
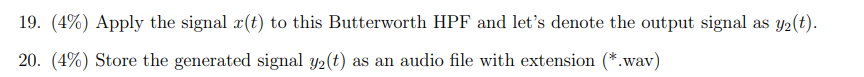
A close-up of a text

Description automatically generated

A screenshot of a computer

Description automatically generatedExact same constraints and steps as steps 8 + 9 but difference being that in butter function the filter type is high which would allow passing for frequencies above 1250 and reduces amplitude of frequencies below 1250 (as frequency decrease reduction increase).

## Step 19 + 20:



Same steps as 10 + 11 where filtered signal of x but this time filtered with nominator and denominator of Butterworth high pass filter stored in y2 then audio file like in step 2 is generated while doing the normalization inside the audiowrite function.



## Step 21:



A white background with text

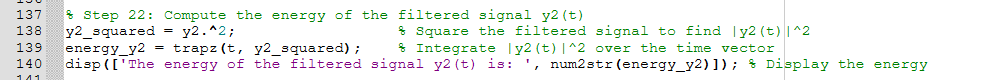
Description automatically generated

A screen shot of a computer screen

Description automatically generatedSimilar to step 3 a graph is made with y2 being the y-axis and t being the x-axis.

## Step 22:

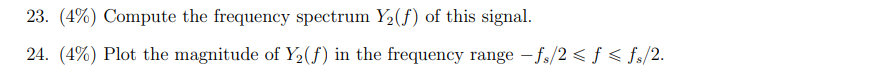




Similar to step 4, energy of y2(t) is computed.



## Step 23+24:



A screenshot of a computer screen

Description automatically generated

Similar to step 14+ 15, with y2\_f\_magnitude being the y axis and x-axis being the f from step 5+ 6 , as there are the same constraints.

A screen shot of a graph

Description automatically generated

## Step 25:



A close-up of a white background

Description automatically generated

Lastly, Similar to step 7 , energy in frequency domain is done using Parseval’s theorem and then after correct scaling the difference between energy in time and frequency domain is computed to verify Parseval's theorem.

A black text with yellow and black numbers

Description automatically generated with medium confidence

7.5314e-05 is equivalent to 0.000075314 which means Parseval’s theorem is valid and the difference is due to using different numerical methods.

# Conclusion:

The project demonstrated the implementation of principles taught in the course through practical application. The signal was effectively separated into low and high-frequency components using 20th-order Butterworth filters with 1250 Hz cutoff frequency. Energy calculations in both time and frequency domains showed minimal difference, validating Parseval's theorem with differences on the order of 10^-4 or less. The generated audio files and spectral analyses provide verification of the filtering effectiveness, while the figures offer insights into the signal's characteristics at each processing stage. This implementation serves as a practical benefit for understanding and applying filtering techniques in real-world applications.