UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

LABORATORIO 1:
SEÑALES ANÁLOGAS Y DIGITALES

DANY RUBIANO JIMENEZ

Profesores: Carlos González Cortés
Ayudantes: Pablo Reyes Díaz
Maximiliano Pérez Rodríguez

Santiago - Chile
18 de abril de 2016

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS .... **iv**

CAPÍTULO 1. INTRODUCCIÓN **5**
1.1 MOTIVACIÓN Y ANTECEDENTES 5
1.2 OBJETIVOS 5
1.3 ORGANIZACIÓN DEL DOCUMENTO 5

CAPÍTULO 2. DESARROLLO DE LA EXPERIENCIA **7**
2.1 HERRAMIENTAS 7
2.2 PROCEDIMIENTO 7
2.2.1 Importación de la señal de audio 7
2.2.2 Gráfico de la función de audio en el tiempo 7
2.2.3 Aplicación de la transformada de Fourier 8
2.2.3.1 Gráfico en el dominio de la frecuencia 8
2.2.3.2 Transformada de Fourier inversa 10
2.2.4 Estudio en el dominio de la frecuencia 11
2.2.4.3 Componentes de mayor amplitud 11
2.2.4.4 Truncamiento al 15 % 11
2.2.4.5 Transformada de Fourier inversa sobre lo truncado 13

CAPÍTULO 3. ANÁLISIS DE RESULTADOS **15**

CAPÍTULO 4. CONCLUSIONES **17**

CAPÍTULO 5. BIBLIOGRAFíA **19**

CAPÍTULO 6. APÉNDICE (CÓDIGO: SCRIPT) **21**

ÍNDICE DE FIGURAS

Figura 2-1: Gráfico de la función de audio en el tiempo 8
Figura 2-2: Gráfico de frecuencias 9
Figura 2-3: Gráfico de frecuencias normalizado 10
Figura 2-4: Gráfico de la función de audio en el tiempo obtenido con la transformada de Fourier inversa 11
Figura 2-5: Gráfico de la función de audio en el tiempo, normalizado 11
Figura 2-6: Gráfico de frecuencias con el truncamiento al 15 % 12
Figura 2-7: Gráfico de frecuencias con el truncamiento al 15 % normalizado 13
Figura 2-8: Gráfico de la función de audio en el tiempo truncado 14
Figura 2-9: Gráfico de la función de audio en el tiempo truncado, normalizado 14
Figura 3-1: Ilustración de la Propiedad de Similitud de la Transformada de Fourier 15
Figura 3-2: Ilustración de la Propiedad de Modulación de la Transformada de Fourier 16

CAPÍTULO 1. INTRODUCCIÓN

1.1 MOTIVACIÓN Y ANTECEDENTES

La noción de señal es bastante amplia y aparece en diferentes situaciones en las cuales ciertas cantidades varían en el tiempo o el espacio de una magnitud física o de otra naturaleza. Por tanto está ligada al concepto de función, he allí que se busca su procesamiento.

El Procesamiento de Señales es una disciplina de las ciencias de la Ingeniería que desarrolla las técnicas de procesamiento, análisis e interpretación de señales. Entre las operaciones posibles con las señales se tienen el control, filtrado, compresión de datos, convolución, predicción, entre otras.

Se pueden procesar señales analógicas (representadas por funciones continuas) o señales digitales (dadas por funciones discretas). En el procesamiento de señales existen diferentes ramas dependiendo de la naturaleza de las señales consideradas (audio, voz, imagen, vídeo).El procesamiento de señales puede tener diferentes objetivos:detección de una señal, estimación de los valores de una señal,codificación, compresión para su almacenamiento y transmisión. Sus aplicaciones son amplias en telecomunicaciones, audio, vídeo, imagen (médica, satelital), geofísica.

1.2 OBJETIVOS

Para este laboratorio se tiene como objetivo reforzar los contenidos vistos en clases sobre señales análogas y digitales, así mismo de la serie y transformada de Fourier, todo esto en base a su aplicación en el estudio de un audio.

1.3 ORGANIZACIÓN DEL DOCUMENTO

El presente documento distribuye su contenido de la siguiente forma, en primer lugar se encuentra un capítulo dedicado al desarrollo de la experiencia en donde se describe el uso de las herramientas pedidas, a continuación se da el lugar para el análisis de los resultados obtenidos, y en último se dan las conclusiones respectivas al desarrollo de esta experiencia.

CAPÍTULO 2. DESARROLLO DE LA EXPERIENCIA

2.1 HERRAMIENTAS

Entre los recursos utilizados, se encuentra el lenguaje de programación Python en su versión 3 (en este caso se utilizó la versión 3.4), juntos algunos módulos para poder trabajar con el procesamiento de audio y algunas operaciones matemáticas, estos son:

Numpy: NumPy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices. El ancestro de NumPy, Numeric, fue creado originalmente por Jim Hugunin con algunas contribuciones de otros desarrolladores. En 2005, Travis Oliphant creó NumPy incorporando características de Numarray en NumPy con algunas modificaciones. NumPy es open source.

Matplotlib: Matplotlib es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una API, pylab, diseñada para recordar a la de MATLAB.

Scipy: SciPy es una biblioteca open source de herramientas y algoritmos matemáticos para Python. SciPy contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales y de imagen, resolución de ODEs y otras tareas para la ciencia e ingeniería.

2.2 PROCEDIMIENTO

2.2.1 Importación de la señal de audio

Importe la señal de audio utilizando la función read de scipy.

2.2.2 Gráfico de la función de audio en el tiempo

Dado el audio entregado en base a la importación realizada con la función 'read' de Scipy, se obtiene un arreglo que contiene toda la información en lo respectivo a las señales que este proporciona. Además, se puede obtener la frecuencia de muestreo que en este caso es de 870912, es decir que este es el número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta, durante el proceso necesario para convertirla de analógica en digital. Para poder realizar el gráfico del audio en función del tiempo, es necesario obtener valga la redundancia, el tiempo total que dura este. Es por esto que dada la frecuencia de muestreo y el tamaño total del arreglo que contiene la información del audio, se deriva que:

$$tiempoTotal = \frac{tamanioArreglo}{frecuenciaMuestreo} = \frac{870912}{44100} = 19{,}748571428571427[s]$$

Ahora, con la función linspace de Numpy, la cual retorna una cierta cantidad de números con igual separación en el intervalo dado, se genera una lista de números del mismo tamaño de la muestra total de datos, entre los $0[s]$ y los $19{,}748[s]$. Una vez obtenido esto, se procede a graficar con la función plot de Matpolib.

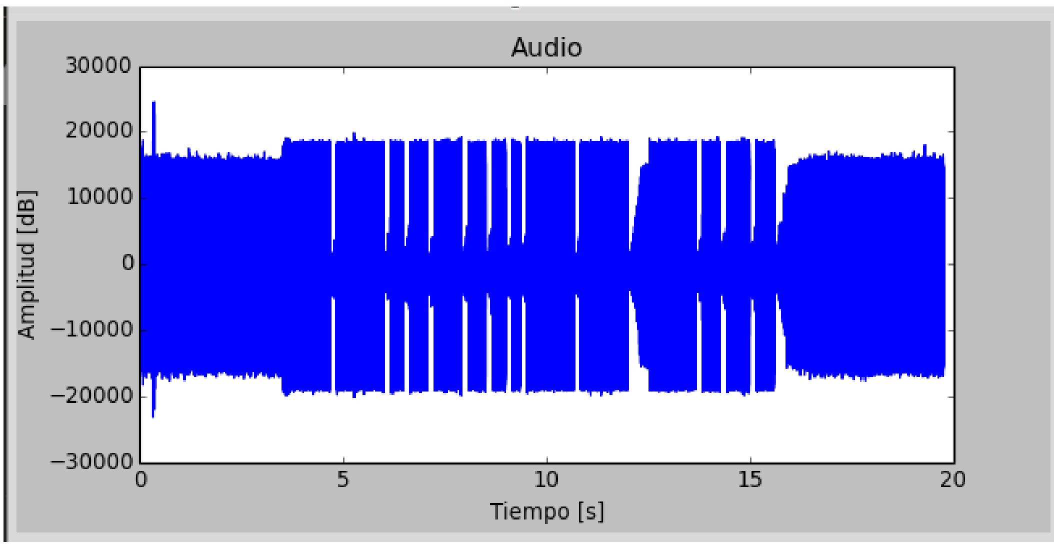

Figura 2-1: Gráfico de la función de audio en el tiempo

2.2.3 Aplicación de la transformada de Fourier

2.2.3.1 Gráfico en el dominio de la frecuencia

Para proceder a obtener la transformada de Fourier y su gráfico respectivo, la función fft de Scipy entrega el resultado a esta, pero para poder realizar el gráfico es necesario realizar la transformación en el dominio del tiempo al dominio de la frecuencia, asimilándose a lo que se hizo en el paso anterior con el tiempo.

Por definición, $f = \frac{1}{T}$, por lo tanto, se puede tener que dado cada elemento del arreglo que contiene la información del dato, dividido por el tiempo de cada señal presente, corresponde a la frecuencia. De modo que con la función $arange$ de Scipy, la cual tiene la misma funcionalidad de la función $linspace$ pero con número enteros, se tienen todos los elementos, vasta con hacer la división por el tiempo ya obtenido anteriormente. Ahora, se procede a realizar el gráfico correspondiente.

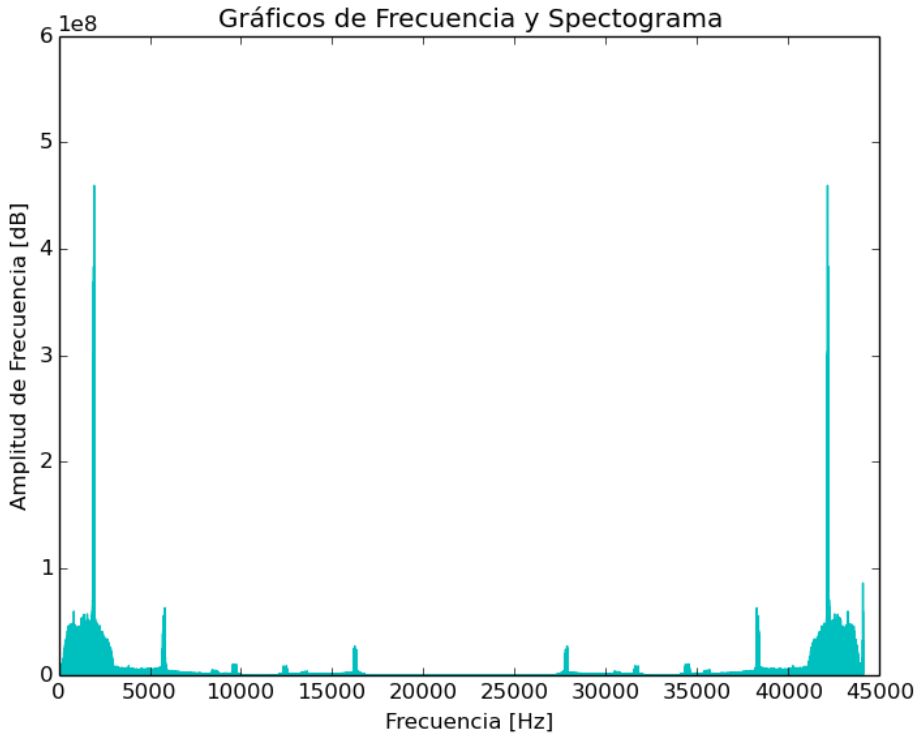

Figura 2-2: Gráfico de frecuencias

Normalizando la transformada de Fourier del audio, se obtiene lo siguiente:

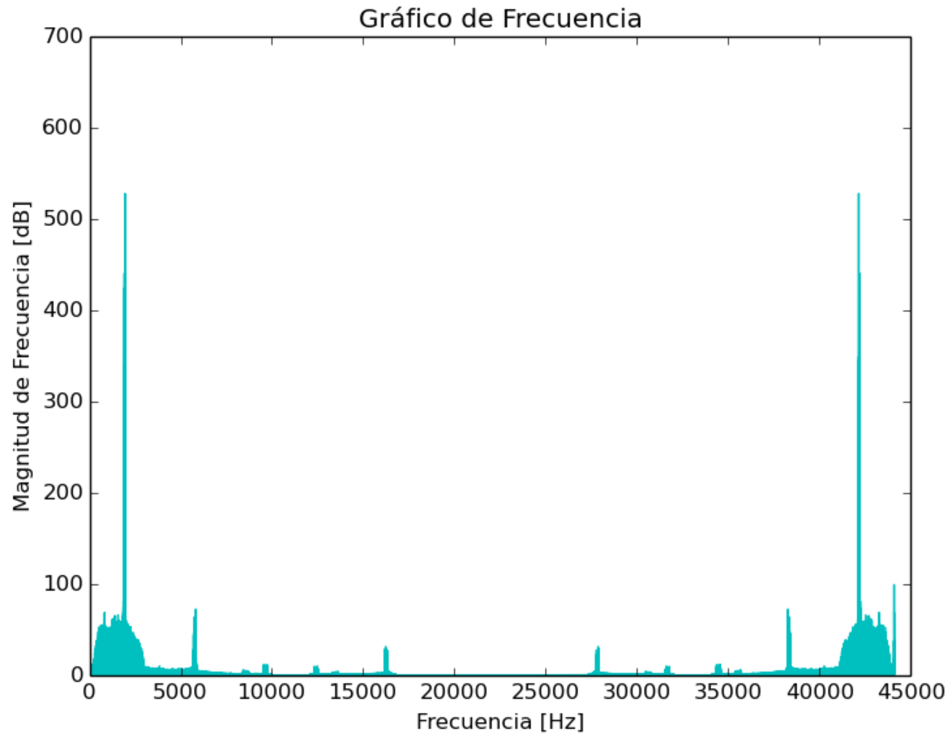

Figura 2-3: Gráfico de frecuencias normalizado

2.2.3.2 Transformada de Fourier inversa

Al aplicar la inversa de la transformada a lo obtenido anteriormente, del dominio de la frecuencia se cambia nuevamente al dominio del tiempo, y con la función $ifft$ de Scipy, se obtiene el resultado de ello. Ahora basta graficar lo aplicado con respecto al tiempo.

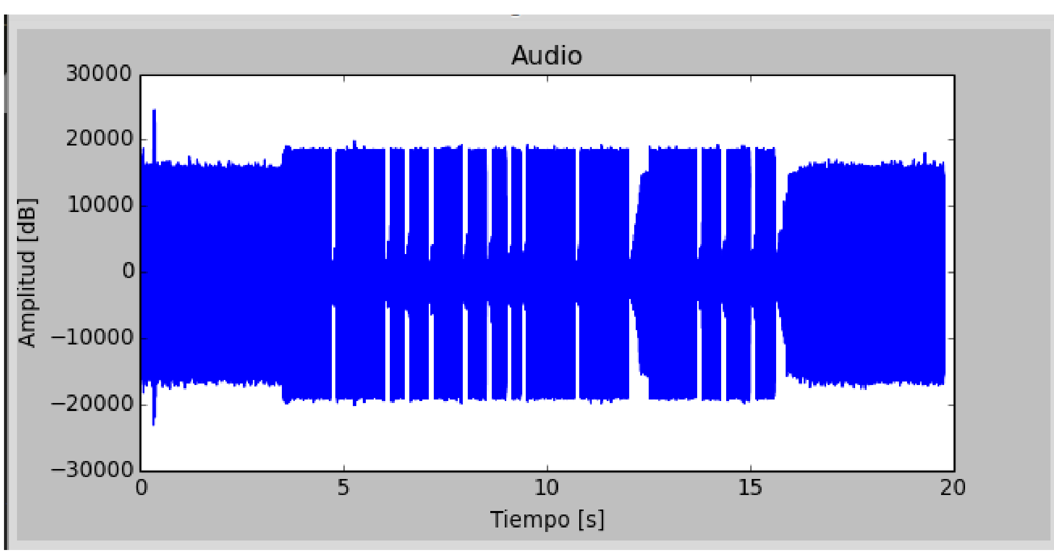

Figura 2-4: Gráfico de la función de audio en el tiempo obtenido con la transformada de Fourier inversa

Haciendo la inversa a la transformada de Fourier Normalizada se obtiene lo siguiente:

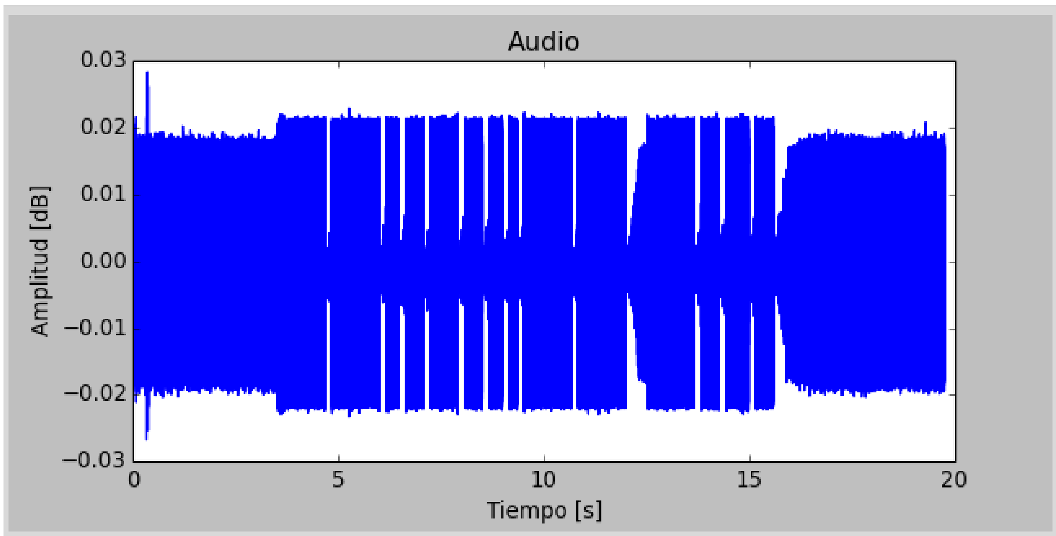

Figura 2-5: Gráfico de la función de audio en el tiempo, normalizado

2.2.4 Estudio en el dominio de la frecuencia

2.2.4.3 Componentes de mayor amplitud

Para obtener los componentes de mayor amplitud en el dominio de la frecuencia, se implementó una función que busca el numero mayor sobre la lista obtenida en la aplicación de la transformada de Fourier, retornando este número y la posición en la que se encuentra. Esta función se encuentra en el anexo del presente informe. El resultado obtenido es que el valor de la amplitud máxima es $525{,}125896829 + 52{,}1761169452j$

2.2.4.4 Truncamiento al 15 %

La función implementada para buscar el componente de mayor amplitud, retorna también la posición del arreglo en la cual se encuentra, por lo tanto, como se requiere un truncamiento con un margen del 15 % en torno al componente, se puede trabajar sobre el arreglo de datos obtenido. De esta forma, dado el tamaño del arreglo se puede obtener el valor del margen pedido, para luego obtener los valores del intervalo que debe resultar.

- Posición de la amplitud máxima = 832508
- Tamaño del Arreglo = 870912
- Margen = $870912 * 0{,}15 = 130636, 8$
- Intervalo = $[701871, 2, 870912]$, debido a que la cota máxima es el tamaño total del arreglo.

Dado el intervalo calculado, se genera una copia del arreglo de la transformada de Fourier de la información, para poder trabajar sobre el. Entonces, para las frecuencias $0701871, 2$, el valor de la amplitud se hace nulo, de esta forma se genera el truncamiento pedido.

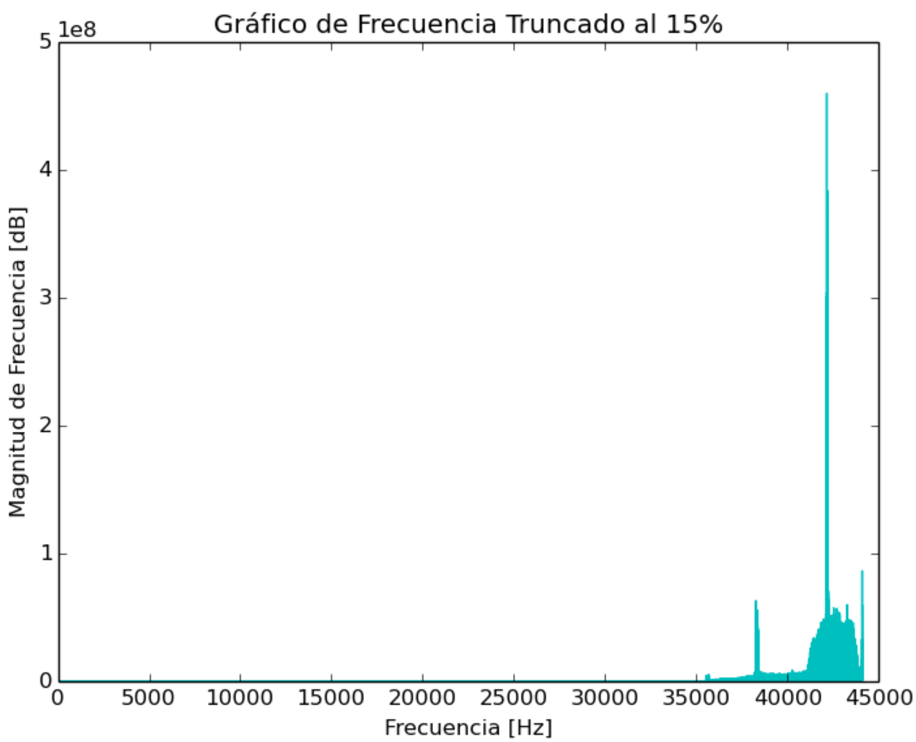

Figura 2-6: Gráfico de frecuencias con el truncamiento al 15 %

Normalizado:

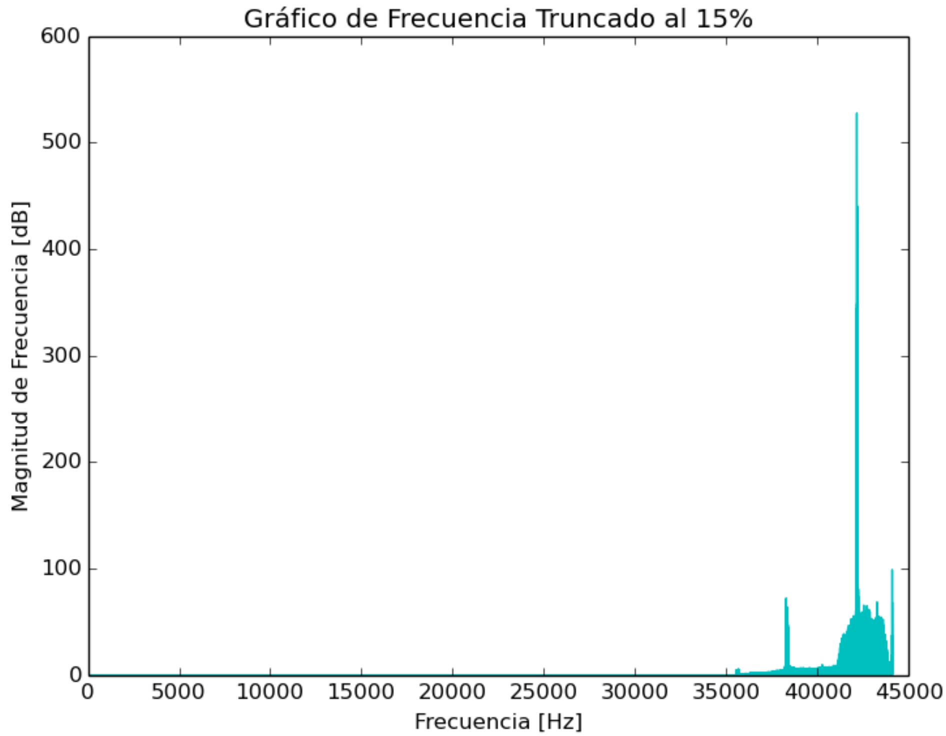

Figura 2-7: Gráfico de frecuencias con el truncamiento al 15 % normalizado

2.2.4.5 Transformada de Fourier inversa sobre lo truncado

Se hace el mismo procedimiento que se hizo en secciones anteriores, pero trabajando sobre el arreglo truncado.

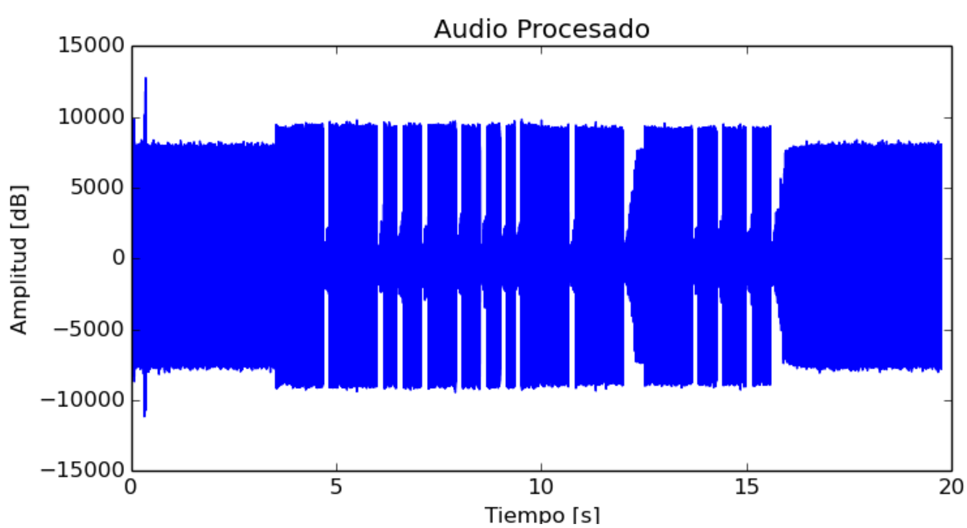

Figura 2-8: Gráfico de la función de audio en el tiempo truncado

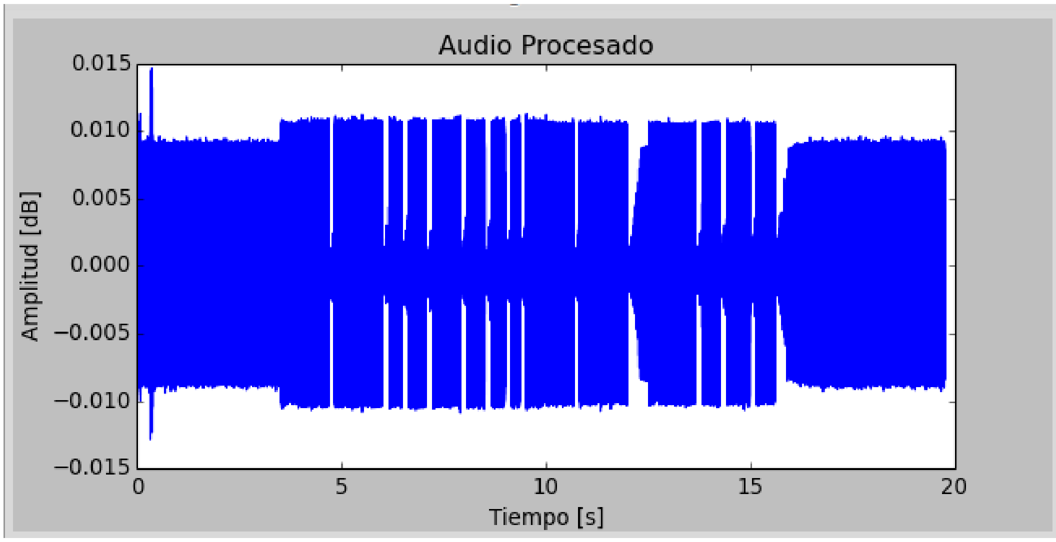

Figura 2-9: Gráfico de la función de audio en el tiempo truncado, normalizado

CAPÍTULO 3. ANÁLISIS DE RESULTADOS

A partir del gráfico de la función de audio en el dominio del tiempo y del gráfico de su transformada de Fourier, se puede comprobar la propiedad de la similitud de la transformada, en la que la ecuación $f(t) \rightleftharpoons F()$ conduce a una relación de escala:

$$f(at) = \frac{1}{a}F(\frac{v}{a})$$

Un agrandamiento de la escala de tiempo conduce a una contracción de la escala de frecuencias.

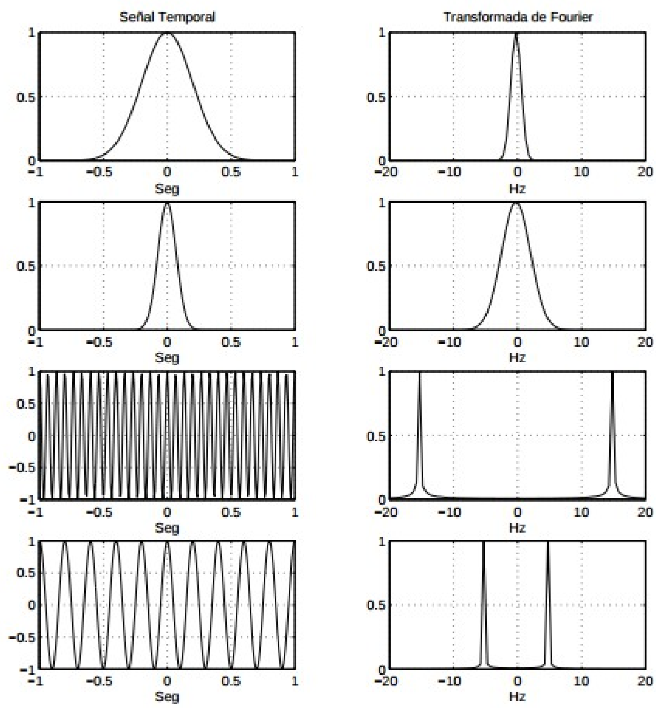

Figura 3-1: Ilustración de la Propiedad de Similitud de la Transformada de Fourier

En el caso de la experiencia desarrollada, se puede observar que en el dominio del tiempo la función de audio hay una distribución muy contraida en el lapso de tiempo debido a la multiplicidad de datos existentes, por lo que hay un agrandamiento a escala de las frecuencias.

Así mismo, se puede comprobar la propiedad de la translación en frecuencia y modulación, la cual dice que la relación $f(t) \rightleftharpoons F()$ conduce a:

$$f(t)e^{+2j_p t} \rightleftharpoons F(vp)$$

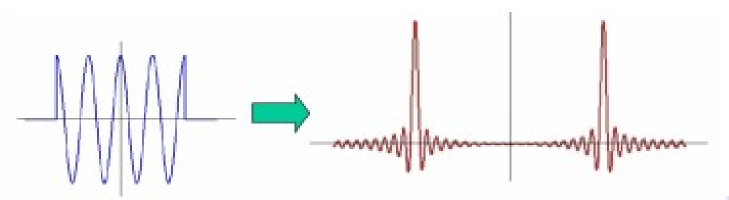

Figura 3-2: Ilustración de la Propiedad de Modulación de la Transformada de Fourier

En el gráfico de frecuencias, se puede observar que hay una modulación, transladando las amplitudes hacia la derecha.

Teniendo en cuenta la transformación inversa de Fourier aplicada, se puede detallar que en comparación al primer gráfico de audio obtenido, estos son casi exactamente iguales, lo que comprrueba que con la transformada de Fourier se pueden realizar cambios en el dominio de tiempo a frecuencia y viceversa. Los unicos aspectos que cambian son en el caso de la amplitud cuando se hace la inversa a la transformada de Fourier que esta normalizada, en donde estos valores son menores.

Los componentes de amplitud maxima son dos, debido a la propiedad de similitud revisada anteriormente, pero en este caso se toma el ultimo componente de la amplitud maxima para los efectos de cálculo.

Tomando el nuevo espectro obtenido a partir del truncamiento del 15 % en torno al componente de máxima amplitud, como resultado posterior al aplicar la transformada de Fourier inversa, se obtiene que hay de cierta manera menor ruido en la función, ya que se acotaron las frecuencias que toma esta función, por lo tanto se hizo de alguna manera una filtración de los datos. Así mismo, se puede inferir que el otro impulso que se observaba en el dominio de la frecuencia, era era el reflejo de la otra. Y se observa que producto de la filtración y disminución del ruido hay una baja en la magnitud de la amplitud.

CAPÍTULO 4. CONCLUSIONES

Una vez finalizada la experiencia se puede concluir que:

La transformada de Fourier es muy util para obtener información que no es evidente en el dominio temporal, por ejemplo, para la experiencia desarrollada, es más facil denotar el ancho de banda en el que se concentra el audio a partir de su análisis en el dominio de la frecuencia, el cual comprendía principalmente entre los 40000 y 45000 [Hz]. Así mismo, a partir de la observación de los graficos resultantes y su comparación, se pudieron denotar algunas de las comprobaciones de las propiedades que tiene la transformada de Fourier, tales como la similitud y la modulación, además se pudo comprobar que cada señal o sistema que puede ser transformado tiene una única transformada de Fourier, solo hay una señal de tiempo y una señal de frecuencia que se correspondan vis a vis.

Se pudo desarrollar una forma de filtración semi-formal del audio, a partir del acotamiento del ancho de banda que presentan las frecuencias resultantes, para obtener menos aspectos de ruido y poder estudiar mejor las señales obtenidas. Este es un proceso muy importante para el trabajo con las señales, como por ejemplo en el sonido donde se pueden aplicar filtraciones para quietar las frecuencias más agudas o las más graves.

CAPÍTULO 5. BIBLIOGRAFÍA

Cano, J. L. (2012). Transformada de fourier discreta en python con scipy. http://pybonacci.org/2012/09/29/transformada-de-fourier-discreta-en-python-con-scipy/.

Ruiz, I. V. M. (2013). Audiolab - audio en python. http://turing.iimas.unam.mx/~ivanvladimir/es/post/audio_en_python/.

USACH, I. I. (2012). Proyectos del laboratorio de redes de computadores. https://github.com/redes-usach.

Wikipedia. (s.f.-a). Frecuencia de muestreo. https://es.wikipedia.org/wiki/Frecuencia_de_muestreo.

Wikipedia. (s.f.-b). Transformada de fourier. https://es.wikipedia.org/wiki/Transformada_de_Fourier.

CAPÍTULO 6.   APÉNDICE (CÓDIGO: SCRIPT)

```
import numpy as np
from numpy import sin, linspace, pi
from scipy.io.wavfile import read,write
from scipy import fft, arange, ifft
import matplotlib.pyplot as plt
from pylab import plot, show, title, xlabel, ylabel

def leerAudio(audio):
    rate,info=read(audio)
    dimension = info[0].size
    if dimension==1:
        data = info
    else:
        data = info[:,dimension-1]

    return data, rate


def plotTime(data, rate):
    large = len(data)
    T = large/rate
    t = linspace(0,T,large)             #linspace(start,stop,
    time1 = plot(t, data)
    title('Audio')
    xlabel('Tiempo [s]')
    ylabel('Amplitud [dB]')
    return time1, t


def plotFrecuece(data, rate):
    large = len(data)
    k = arange(large)
    T = large/rate
    frq = k/T
    Y1 = fft(data)
    Y2 = fft(data)/large
    frq1 = plot(frq,abs(Y1),'c')
    title('Gráficos de Frecuencia y Spectograma')
    ylabel('Amplitud de Frecuencia [dB]')
    xlabel('Frecuencia [Hz]')
    return Y1, frq, large, frq1


def plotFrecueceNormalizada(data, rate, Y, large, frq):
    Y2 = Y/large
    frq2 = plot(frq,abs(Y2),'c')
    title('Gráficos de Frecuencia y Spectograma')
    ylabel('Amplitud de Frecuencia [dB]')
    xlabel('Frecuencia [Hz]')
    return frq2
```

```python

def plotTimeAgain(data, rate, Y, t):
    otro = ifft(Y)
    time2 = plot(t, otro)
    title('Audio')
    xlabel('Tiempo [s]')
    ylabel('Amplitud [dB]')
    return time2

def plotTimeAgainNormalizado(data, rate, Y, t, large):
    otro = ifft(Y)/large
    time3 = plot(t, otro)
    title('Audio')
    xlabel('Tiempo [s]')
    ylabel('Amplitud [dB]')
    return time3

def buscarMayor(lista):
    if lista ==[]:
        return("error")
    elif len(lista) == 1:
        return(lista)
    mayor = 0
    index = 0
    i = 0
    while lista != []:
        primero = lista[0]
        if mayor > primero:
            mayor = mayor
        else:
            mayor =primero
            index = i
        lista = lista[1:]
        i = i+1
    return mayor, index


def truncar(data, rate, Y, indexMayor):
    aux = Y
    b = indexMayor
    vmax = len(aux)
    cota = vmax * 0.15

    for i in range(0, vmax):
        if i < (b - cota):
            aux[i] = 0
        elif i > (b + cota):
            aux[i] = 0

    return aux


```

```python
def plotTimeAgain(data, rate, Y, t):
    otro = ifft(Y)
    time2 = plot(t, otro)
    title('Audio')
    xlabel('Tiempo [s]')
    ylabel('Amplitud [dB]')
    return time2

def plotTimeAgainNormalizado(data, rate, Y, t, large):
    otro = ifft(Y)/large
    time3 = plot(t, otro)
    title('Audio')
    xlabel('Tiempo [s]')
    ylabel('Amplitud [dB]')
    return time3

def buscarMayor(lista):
    if lista ==[]:
        return("error")
    elif len(lista) == 1:
        return(lista)
    mayor = 0
    index = 0
    i = 0
    while lista != []:
        primero = lista[0]
        if mayor > primero:
            mayor = mayor
        else:
            mayor =primero
            index = i
        lista = lista[1:]
        i = i+1
    return mayor, index


def truncar(data, rate, Y, indexMayor):
    aux = Y
    b = indexMayor
    vmax = len(aux)
    cota = vmax * 0.15

    for i in range(0, vmax):
        if i < (b - cota):
            aux[i] = 0
        elif i > (b + cota):
            aux[i] = 0

    return aux
```

```python

################################################################
################################################################
##Funcion Main

data, rate = leerAudio("beacon.wav")

time1, t = plotTime(data, rate)
show()

Y, frq, large, frq1 = plotFrecuece(data, rate)
show()

plotFrecueceNormalizada(data, rate, Y, large, frq)
show()

plotTimeAgain(data, rate, Y, t)
show()

plotTimeAgainNormalizado(data, rate, Y, t, large)
show()

mayor, indexMayor = buscarMayor(Y)

aux = truncar(data, rate, Y, indexMayor)

plotFrqTruncada(aux, frq)
show()

plotFrqTruncadaNormalizado(aux, frq, large)
show()

plotTimeTruncado(aux, t)
show()

plotTimeTruncadoNormalizado(aux, t, large)
show()

```