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

Laboratorio N 1

Integrantes: Juan Fernandez Muñoz
Vicente Vega Toro

Curso: Redes de Computadores

Profesor(a): Carlos González Cortés

20 de Abril de 2019

Tabla de contenidos

1. Introducción

Las señales son un medio que permite la transmisión de datos en grandes distancias, es por esto que son bastantes usadas en la actualidad en diversas áreas, tales como la radio, el internet, la telefonía, entre otros.

Son diversas las transformaciones que se le pueden realizar a una señal, una de estas es la transformada de Fourier, la que permite transformar una señal en el dominio del tiempo al dominio de la frecuencia. Esta se puede representar mediante la siguiente expresión.

$$F(w) = \int_{-\infty}^{\infty} f(t)e^{-jwt}dt \tag{1}$$

Mediante la utilización del lenguaje de programación Python junto a sus bibliotecas numpy, scipy y matplotlib, se aplica la transformada de Fourier a un archivo de audio, en formato WAV, para analizar como esta permite el estudio de la señal. Todo esto con el objetivo de comprender mejor como una señal representada en función de la frecuencia nos permite obtener información que estando en función del tiempo no se puede obtener.

El presente informe cuenta con un desarrollo, en donde se presentan las acciones que se realizan para llevar a cabo esta experiencia, un análisis de resultados y finalmente una conclusión, donde se muestran apreciaciones finales respecto al trabajo realizado.

Por lo tanto, en el presente informe se busca detallar el trabajo realizado para el desarrollo del primer laboratorio de la asignatura Redes de Computadores de la Universidad de Santiago de Chile, junto a sus respectivos análisis y resultados.

2. Desarrollo de la experiencia

Para el desarrollo de esta experiencia, como se menciona anteriormente, se utiliza el lenguaje de programación Python, con las respectivas bibliotecas numpy, scipy y matplotlib para poder .

```
import numpy as np
import matplotlib.pyplot as py
import scipy.io.wavfile as wavfile
from scipy.fftpack import fft
```

Figura 1: Importe de bibliotecas.

A continuación se describe cómo se utiliza Python junto a sus bibliotecas para el análisis del sonido.

1. Lo primero que se hace es leer el archivo de sonido .wav con la función **read** de la biblioteca scipy, a la cual se le entrega el nombre del archivo del sonido. De esta función se obtienen un arreglo que contiene la cantidad de muestras de este sonido que se obtienen en un segundo, y un vector de la biblioteca numpy con el total de muestras.

   ```
   sound = wavfile.read(nombre)
   return sound
   ```

   Figura 2: Función read.

   Luego para graficar la amplitud del sonido (datos obtenidos con la función **read**) en función del tiempo, se usa la función **plot** de la librería matplotlib, a la cual se le ingresa como eje y, el arreglo con el total de datos obtenidos de la función **read** y en el eje x se ingresa el tiempo total de la duración del sonido, el cual se de la siguiente forma:

   $$TiempoTotal = \frac{Cantidad\ total\ de\ datos\ obtenidos}{Cantidad\ de\ datos\ obtenidos\ en\ un\ segundo} \tag{2}$$

2. Una vez obtenido los datos, se realiza la transformada de Fourier de las muestras obtenidas del archivo de sonido. Para realizar lo anterior, se utiliza la función que provee la biblioteca numpy, llamada **fft** (nombre abreviado de "$FastFourierTransform$"), el

```
def graficoTiempo(sound):
    x = np.linspace(0,sound[1].size/sound[0],sound[1].size)
    y=sound[1]
    py.figure(1)
    py.plot(x,y)
    py.xlabel("Tiempo (s)")
    py.ylabel("Muestra")
```

Figura 3: Código gráfico datos de archivo de sonido.

cual implementa el algoritmo de la "*transformada rápida de Fourier*", que corresponde a un algoritmo óptimo para el cálculo de esta de manera discreta. A esta función se le ingresa la cantidad total de muestras obtenidas del archivo de sonido. Posterior a lo dicho anteriormente, para obtener la frecuencia sobre la que se proyectan los puntos de la transformada, se utiliza una función de la biblioteca numpy llamada **fftfreq**, a la cual se le entrega la cantidad total de datos del muestreo y a su vez, el la distancia que hay entre cada muestra, la cual se calcula de la siguiente forma:

$$Distancia = \frac{1}{Cantidad\ de\ muestras\ en\ un\ segundo} \tag{3}$$

Luego, la función retorna un arreglo con todos los puntos correspondientes a la frecuencia con la que se proyecta cada punto del arreglo con los puntos de la transformada de Fourier. Nuevamente para graficar se utiliza matplotlib, utilizando en el eje X el arreglo con las frecuencias y en el eje Y el arreglo con los puntos de la transformada de Fourier.

```
def frecuenciaFourier(sound):
    transformada = np.fft.fft(sound[1])
    freq = np.fft.fftfreq(sound[1].size,1/sound[0])
    py.figure(2)
    py.plot(freq,np.absolute(transformada))
    py.xlabel('Frecuencia')
    py.ylabel('Amplitud')
    #py.show()
    return transformada,freq
```

Figura 4: Código transformada de Fourier y gráfico respecto a frecuencia.

3. Se procede a truncar ciertos valores de la transformada de Fourier asignándoles el valor 0. Para esto, se escogen ciertos valores que se consideran no significativos para la función

(ruido), y se les asigna el valor 0. La elección se realiza mirando en el gráfico obtenido en el punto 2 aquellos puntos que deslindan con los puntos más altos. Luego se procede a graficar la nueva transformada de Fourier (con los valores truncados) utilizando el mismo rango de frecuencias que en el gráfico del punto 2.

```
def frecuenciaFourierTruncado(transformada):
    copyT = transformada.copy()
    freq = np.fft.fftfreq(sound[1].size,1/sound[0])
    for x in range(2000):
        copyT[x] = 0
        copyT[73112 - x] = 0
    for x in range(12000,36556):
        copyT[x] = 0
        copyT[73112 - x] = 0
    py.figure(3)
    py.plot(freq,np.absolute(copyT),'m')
    py.title("Transformada De Fourier truncada")
    py.xlabel('Frecuencia(Hz)')
    py.ylabel('Amplitud')
    #py.show()
    return copyT
```

Figura 5: Código para truncamiento de valores de la transformada de Fourier.

4. Se realiza la transformada de Fourier inversa al nuevo arreglo con la transformada truncada, utilizando la función **ifft** (nombre abreviado de "$InverseFastFourierTransform$"), la cual implementa un algoritmo óptimo para el cálculo de la función inversa de una transformada de Fourier discreta. Para esto, simplemente se le ingresa el arreglo con los datos de la transformada de Fourier truncada obtenido en el punto 3. Luego, el nuevo arreglo obtenido es la función de la amplitud del sonido en función del tiempo, la cual es utilizada para comparar con el arreglo de datos obtenidos en el punto uno. A su vez, se crea un nuevo sonido con los datos obtenidos de la inversa truncada para comparar de manera práctica las diferencias sonoras.

```
def inversaTruncada(copyT, sound):
    inversa = np.fft.ifft(copyT).real
    x = np.linspace(0,sound[1].size/sound[0],sound[1].size)
    py.figure(4)
    py.plot(x,inversa)
    py.title("Frecuencia truncada inversa")
    py.xlabel('Tiempo')
    py.ylabel('Amplitud')
    #py.show()
    wavfile.write("test.wav",sound[0],inversa)
    return inversa
```

Figura 6: Código para sacar inversa truncada y gráfico de esta respecto al tiempo.

5. Finalmente, se calcula el error cuadrático medio entre los datos obtenidos originalmente del archivo de sonido con la función **read** y los que se obtienen realizando la transformada de Fourier inversa truncada. También, se compara con los datos que se obtienen de la transformada de Fourier inversa sin truncar, para ver la precisión de la función **ifft**

```
def errorCuadtratico(real,calculado):
    sumatoria = 0
    for x in range(73113):
        sumatoria = sumatoria + (calculado[x]-real[x])**2
    print(sumatoria)
    div = sumatoria / 73113.0
    raiz = np.sqrt(div)
    return raiz
```

Figura 7: Cálculo error cuadrático medio.

3. Análisis de los resultados

El audio utilizado para trabajar en este laboratorio fue leído y transformado en datos numéricos, los cuales fueron graficados obteniendo así el siguiente resultado.

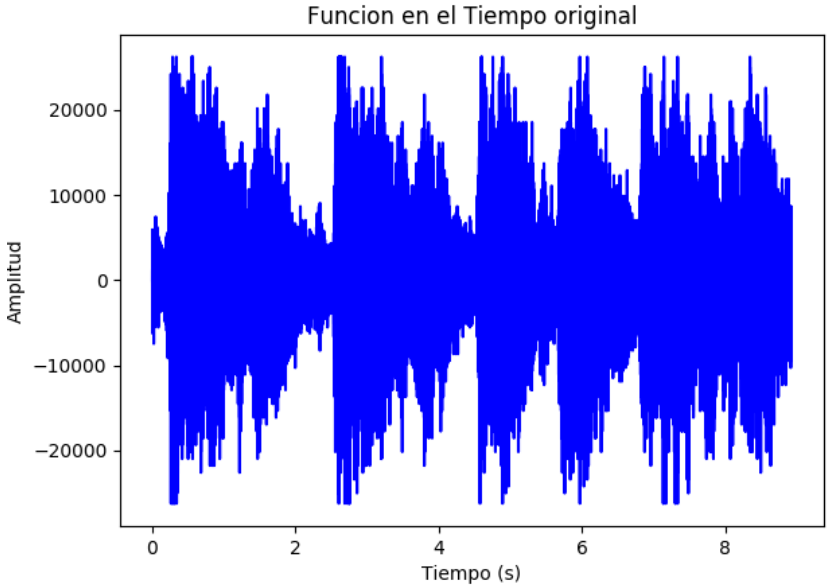

Figura 8: Gráfico de señal original en el tiempo.

En este se puede observar como las amplitudes se distribuyen a través del tiempo, siendo este los 8 segundos que dura el audio y es indicado en el eje x del gráfico. En el gráfico cada ciertos tramos se pueden observar ciertas amplitudes mas altas, las que se deben a la palabra 'Aleluya' que se menciona en audio, por lo que es así posible saber los puntos en el que esta palabra comienza a ser pronunciada. Las amplitudes altas observadas van decayendo debido a que el sonido escuchado comienza a ser menor y por lo tanto los datos obtenidos tienen una amplitud menor.

Aplicando la transformada de Fourier a la señal mencionada anteriormente y graficando los puntos obtenidos se puede ver el siguiente gráfico.

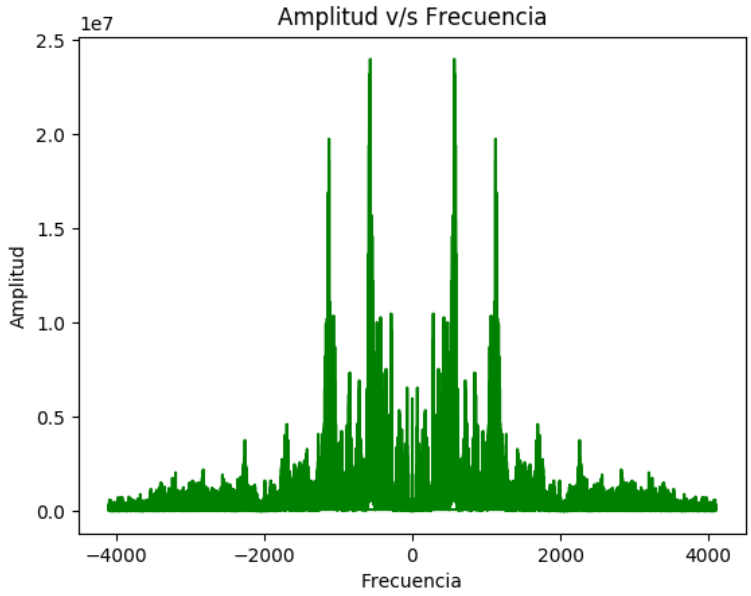

Figura 9: Transformada de Fourier de señal original.

Los datos utilizados en el gráfico anterior se encuentran en valor absoluto, es debido a esto que solo se muestra la parte positiva de la transformada, además esto se debe ya que en el contexto de este laboratorio los valores negativos no aportan al resultado. Por otro lado, se puede observar que este tiene cuatro picos importantes donde se concentran las frecuencias mas altas, en cambio, en el resto se concentran frecuencias mas bajas que pueden ser interpretadas como ruido.

Al eliminar las frecuencias menores y mantener las cuatro amplitudes mas altas, se obtiene el siguiente gráfico.

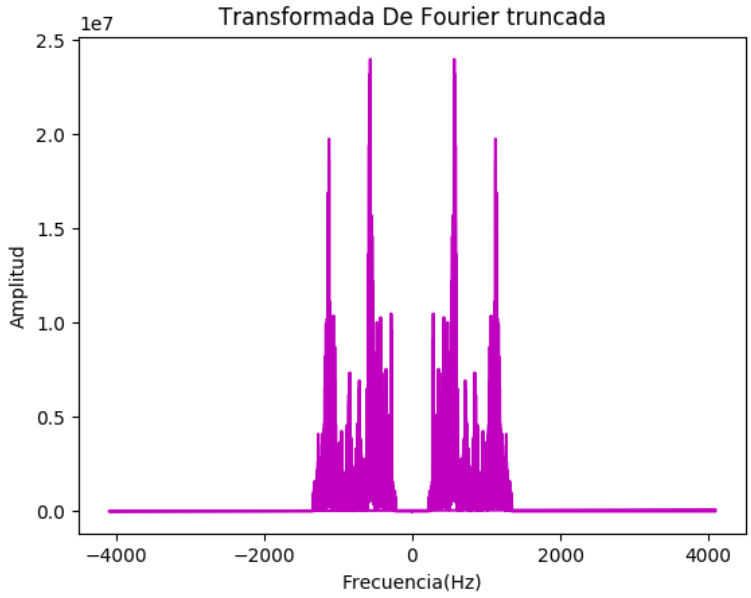

Figura 10: Transformada de Fourier truncada.

A estos datos se aplica la inversa de la transformada de Fourier para así obtener una señal en función del tiempo. En la cual graficando los datos obtenidos se tiene lo siguiente.

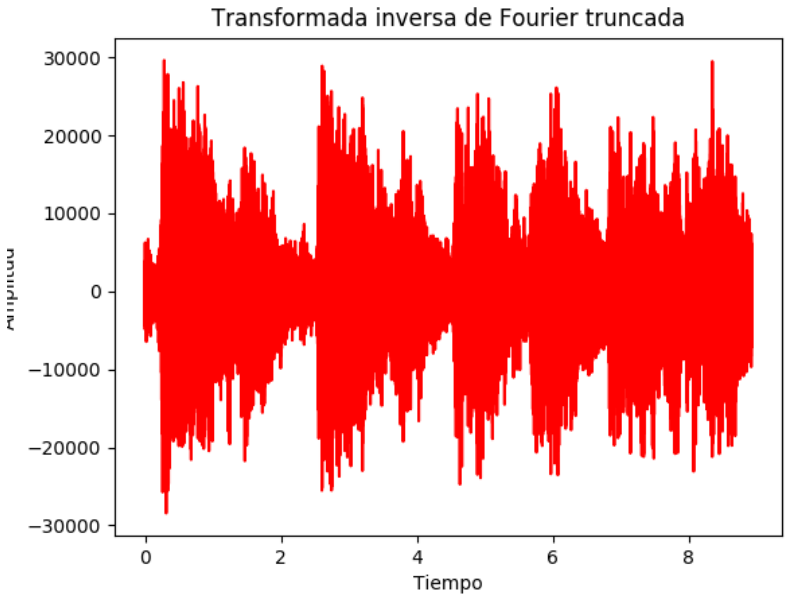

Figura 11: Transformada inversa.

Comparando el gráfico que se obtiene al aplicar la inversa con el gráfico de la

señal original, se puede observar que estas mantienen una estructura similar pero no iguales, donde la primera diferencia se puede observar en el alcance de las amplitudes máximas, en la que la primera alcanza un alto de aproximadamente 30000 y en cambio la original presenta un alto de aproximadamente 26000.

A pesar de que las amplitudes máximas aumentaron, la mayoría de las amplitudes en la transformada inversa disminuyeron en comparación a la original, esto se debe a que se debe mantener el teorema de Parseval el cual dice que la integral del cuadrado de la función en el tiempo es igual a la integral del cuadrado de su transformada. Es por esto que al disminuir las frecuencias se debe disminuir las amplitudes en la función con respecto al tiempo para mantener este teorema.

Al transformar los datos de la inversa de la transformada de Fourier en audio, se puede escuchar las mismas voces que el audio original diciendo 'Aleluya', pero la diferencia es que esto se escucha distorsionado y con un volumen mucho mas alto. Lo primero se debe a que las frecuencias que fueron quitadas son importantes para la calidad del audio, por lo que al quitarlas el audio se escucha en una calidad inferior a la original. Por otro lado, este se escucha mas alto debido a que al disminuir las frecuencias se acentúa mejor las voces.

El error cuadrático medio sirve para comparar los resultados con los datos originales, por lo que calculando este entre los datos de la señal original en el tiempo y la transformada inversa de la señal truncada se puede ver que se tiene un error de 2043.4837, en cambio, calculando el error entre la señal original en el tiempo con la transformada inversa sin truncar se tiene que el error es de 3.0223e-11. Claramente se puede notar la diferencia entre los errores, siendo el primero el mas alto debido a que las frecuencias son alteradas, logrando así que las ondas presentes en el tiempo no sean las mismas y en consecuencia la onda obtenida por la inversa es parecida a la onda original pero no igual. Por otro lado, el segundo es bajo debido a que las frecuencias no son alteradas, el error que se produce es debido al algoritmo utilizado por la función **ifft**, la cual siempre tendrá un margen de error ya sea por como se implementa el algoritmo o las capacidades de la maquina en la que se ejecute el programa. Pero este margen de error nos permite determinar que la función **ifft** si es apropiada al momento de realizar análisis experimentales ya que este error no conlleva a a un escalamiento de error mayor.

4. Conclusiones

En esta experiencia, se puede apreciar la diferencia existente entre el análisis de una señal en función del tiempo, y la misma en función de la frecuencia utilizando la transformada de Fourier. La primera solo nos permite visualizar en qué instantes del tiempo la energía es mayor (el volumen del sonido), mientras que la segunda, nos da la posibilidad de ver bajo que frecuencias se concentra la mayor energía en el sonido, permitiendo visualizar frecuencias bajas que podrían ser consideradas en un principio como ruido, y ver también cuales frecuencias tienen menor relevancia en el sonido mismo y así poder eliminarlas.

Cabe destacar que a pesar de lo dicho anteriormente respecto de las frecuencias que pueden ser consideradas ruido, el hecho de que al generar un nuevo archivo de sonido se haya obtenido un sonido más distorsionado y con menor calidad al truncar los datos obtenidos por la transformada de Fourier, indica que las frecuencias eliminadas en su totalidad no necesariamente son ruido. Esto se debe a que el archivo de sonido analizado es una mezcla tanto de voz como de instrumentos musicales, por lo que se pudo haber alterado alguno de estos componentes en vez de haber eliminado "ruido"propiamente tal.

Además, como apreciación, se puede observar uno de los campos donde se puede aplicar la transformada de Fourier, permitiendo así ver como se aplican los conceptos vistos en clases y como estos afectan a objetos de la vida cotidiana.

También se considera que con esta experiencia, se ha podido aprender a trabajar con tecnologías que permiten el análisis de señales, como lo es python y las bibliotecas mencionadas al principio de este informe. Además, que se ha aprovechado al máximo los recursos que esta tecnología nos brinda y así evitar reinventar la rueda.

Bibliografía

(2018). Transformada de fourier. [Online] https://es.wikipedia.org/wiki/Transformada_de_Fourier.

(2019). scipy.fftpack.fft. [Online] https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.fft.html.

(2019). scipy.fftpack.ifft. [Online] https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.ifft.html.

(2019). scipy.io.wavfile.read. [Online] https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.read.html.

(2019). scipy.io.wavfile.write. [Online] https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.write.html.