Fourier Algorithm

Erick Yary, Emmanuel Galdos

20 de junio de 2022

1. Introducción

La transformada de Fourier, denominada así por Joseph Fourier, es una transformación matemática empleada para transformar señales entre el dominio del tiempo (o espacial) y el dominio de la frecuencia, que tiene muchas aplicaciones en la física y la ingeniería. Es reversible, siendo capaz de transformarse en cualquiera de los dominios al otro.

La transformada rápida de Fourier (FFT) es un algoritmo eficiente que permite calcular la transformada de Fourier discreta (DFT) y su inversa. Ayudando asi a reducir las complejidades de la informática

Por otro lado la transformada discreta de Fourier o (DFT), obtiene una representación en el dominio de la frecuencia al transformar una función matemática en otra, siendo la función original una función en el dominio del tiempo. Requiere que la función de entrada sea una secuencia discreta y de duración finita como puede ser la voz humana. DFT funciona en aplicaciones como osciladores LC para ver cuánto ruido está presente en una onda sinusoidal producida.

Siendo una herramienta sumamente importante en la computacion existen multiples aplicaciones basadas en el algoritmo de Fourier como lo son la multiplicación rápida de polinomios y enteros grandes, algoritmos de filtrado, algoritmos para transformadas discretas de coseno o seno, cálculos de distribuciones isotópicas entre otras.

2. La transformada de Fourier

2.1. La transformada discreta de Fourier

La transformada de Fourier ocurre en muchas versiones diferentes a lo largo de la computación clásica, en áreas que van desde el procesamiento de señales hasta la compresión de datos y la teoría de la complejidad.

En el caso de una función periódica en el tiempo como puede ser un sonido musical continuo pero no necesariamente sinusoidal, la transformada de Fourier puede ser empleada para la simplificacion del cálculo de un conjunto discreto de amplitudes complejas, que representan el espectro de frecuencia de la señal del dominio-tiempo original.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i \frac{i2\pi kn}{N}}$$

Donde n = 0 a N-1 representan el índice de la señal y k = 0 a N-1 representan el índice del espectro.

Para ilustrar las mejoras que propone una FFT, consideramos el conteo de multiplicaciones y sumas complejas para N=4096 puntos de datos. La evaluación de las sumas de la DFT implica directamente N elevado a 2 multiplicaciones complejas y $N(N1)$ sumas complejas, de las cuales $\mathcal{O}(N)$ operaciones

se pueden ahorrar eliminando operaciones triviales como las multiplicaciones por 1, dejando alrededor de 30 millones de operaciones, lo cual representa una mejora considerable.

2.2. La transformada discreta inversa de Fourier

La transformada de Fourier es una aplicacion lineal definida por una serie de propiedades de continuidad y se define mediante una integral, a esta integral se le llama integral de contorno. La transformada inversa de Fourier busca recuperar la funcion inicial (nucleo de transformacion) mediante otra tranformacion integral.

$$X_n = -\frac{1}{N} \cdot \sum_{n=0}^{N-1} x_k \cdot e^{i\frac{i2\pi kn}{N}}$$

Para una compresion mas clara consideremos a la Transformada de Fourier inversa (IFT) como el músico que lee las notas (frecuencias) en una partitura musical y las convierte en tonos (señales en el dominio del tiempo).

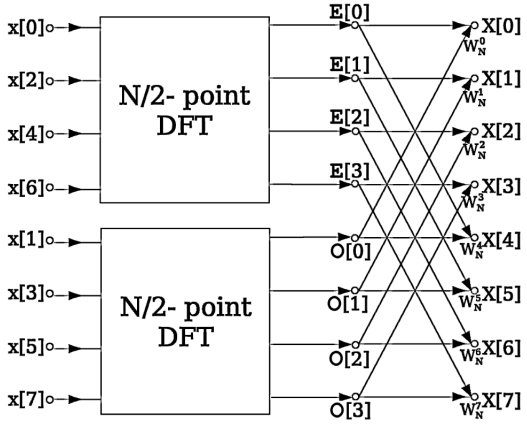
2.3. La transformada rapida de Fourier

La tranformada rapida de Fourier busca hacer eficiente el calculo de la transformada de Fourier discreta y su inversa, reduciendo asi las complejidad presentes en estas.

El cálculo de la DFT requiere mucho tiempo y requiere del orden de N^2 multiplicaciones de punto flotante. Sin embargo, muchas de las multiplicaciones se repiten cuando (i) y (k) varían. La FFT es una colección de rutinas que están diseñadas para reducir la cantidad de cálculos redundantes, siendo necesario que cada implementación diferente de la FFT contenga diferentes características y ventajas. El algoritmo utilizado en algunos lenguajes de programación se conoce como algoritmo de (raíz dividida) que esta basada en el paradigma (divide y venceras) y requiere aproximadamente $NlogN$ operaciones asi como todos los algoritmos FFT requieren de $\theta(NlogN)$ operaciones, sin embargo no hay pruebas de que sea imposible una menor complejidad.

Los algoritmos FFT más conocidos dependen de la factorización de N, pero existen FFT con complejidad O(N log N) para todo N, incluso para N primos. Muchos algoritmos FFT dependen únicamente del hecho de que $e^{\frac{2\pi i}{N}}$ es una N-ésima raíz primitiva de la unidad y, por lo tanto, se puede aplicar a transformadas análogas sobre cualquier campo finito, como las transformadas de teoría de números. Dado que la DFT inversa es lo mismo que la DFT, pero con el signo opuesto en el exponente y un factor 1/N, cualquier algoritmo de FFT puede adaptarse fácilmente a ella.

Una estructura de algoritmo FFT utilizando una descomposición en FFT basada en el paradigma "divide y venceras"de tamaño medio:



2.4. Precision de los algoritmo FFTs

Los algoritmos FFT tienen errores cuando se usa la aritmética de punto flotante de precisión finita, pero estos errores suelen ser bastante pequeños; la mayoría de los algoritmos FFT, Cooley-Tukey, tienen excelentes propiedades numéricas como consecuencia de la estructura de suma por pares de los algoritmos. El límite superior del error relativo para el algoritmo de Cooley-Tukey es $O(\epsilon log N)$, en comparación con $O(\epsilon N^{\frac{3}{2}})$ para la fórmula DFT ingenua, donde ϵ es la precisión relativa de punto flotante de la máquina.

Para verificar la exactitud de una implementación de FFT, se pueden obtener garantías rigurosas en tiempo $O(NlogN)$ mediante un procedimiento simple que verifica la linealidad, la respuesta de impulso y las propiedades de cambio de tiempo de la transformada en entradas aleatorias.

2.5. FFTs multidimencional

Las FFT bidimensionales son particularmente importantes en el campo del procesamiento de imágenes. Una imagen generalmente se representa como una matriz bidimensional de intensidades de píxeles, números reales (y generalmente positivos). Comúnmente se desea filtrar componentes espaciales de alta o baja frecuencia de una imagen, o para convolucionar o desconvolucionar la imagen con alguna función instrumental de dispersión de puntos, y el uso de la FFT es la técnica más eficiente. En tres dimensiones, un uso común de la FFT es resolver la ecuación de Poisson para un potencial (por ejemplo, electromagnético o gravitacional) en una red tridimensional que representa la discretización del espacio tridimensional. Aquí los términos fuente (distribución de masa o carga) y los potenciales deseados también son reales. En dos y tres dimensiones, con arreglos grandes, la memoria suele ser escasa. Por lo tanto, es importante realizar las FFT, en la medida de lo posible, en los datos ïn situ".

3. Aplicaciones para el algoritmo de Fourier

La FFT se utiliza en software de grabación digital, muestreo, síntesis aditiva y corrección de tono y su importancia deriva del hecho de que ha logrado que trabajar en el dominio de la frecuencia sea igual de factible desde el punto de vista computacional que trabajar en el dominio temporal o espacial.

3.1. Convolucion rapida en redes neuronales

Suele ser un problema recurrente el tener tiempo de ejecucion demasiado altos al momento de definir una red neuronal convolucional.

El algoritmo FFT resulta especialmente util al momento de la operacion de la convolucion de la matriz en el dominio del espacio equivalente a la multiplicaion de elementos correspondientes de las dos matrices en el dominio de frecuencia.

El método de superposición y adición se utiliza para dividir señales largas en segmentos más pequeños para facilitar el procesamiento. La convolución FFT utiliza el método de superposición y suma junto con la transformada rápida de Fourier, lo que permite que las señales se convolucionen multiplicando sus espectros de frecuencia. Para núcleos de filtro de más de 64 puntos, la convolución FFT es más rápida que la convolución estándar y produce exactamente el mismo resultado.

El unico problema es que el tamaño de nuestra característica suele ser mucho mayor que el tamaño del núcleo de convolución, si la transformada rápida de Fourier se realiza directamente en los dos, las dos matrices obtenidas no son del mismo tamaño por lo que es necesario expandir el kernel de convolucion al mismo tamaño de la caracteristica haciendo uso de 0´s.

Supongamos que nuestra imagen de entrada y nuestro núcleo de convolución son los siguientes:

input 8x7

5	8	9	1	4	0	3
8	4	3	1	4	2	1
3	2	1	0	4	2	8
7	2	4	8	2	1	2
2	1	0	4	5	7	2
8	4	2	4	0	1	6
3	2	1	4	1	1	5
7	2	4	2	1	1	4

kernel:3x3

1	5	2
1	6	1
7	7	2

Si se utiliza el método completo para la convolución, el relleno de ceros debe ser el siguiente. Cuando se realiza la convolución, el tamaño de la imagen original es de 8x7 y se convertirá en 10x9, por lo que debemos rellenar la imagen original como se describe anteriormente. De la misma manera, necesitamos llenar el kernel de convolución hasta un tamaño de 10x9.

full

5	8	9	1	4	0	3	0	0
8	4	3	1	4	2	1	0	0
3	2	1	0	4	2	8	0	0
7	2	4	8	2	1	2	0	0
2	1	0	4	5	7	2	0	0
8	4	2	4	0	1	6	0	0
3	2	1	4	1	1	5	0	0
7	2	4	2	1	1	4	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

1	5	2	0	0	0	0	0	0
1	6	1	0	0	0	0	0	0
7	7	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

3.2. Tratamiento de imagenes(JPEG) y audio(MP3)

Es posible hacer uso del algoritmo FFT en el tratamiento de imagenes, siendo una herramienta importante en este campo pues esta permite la descomposicion de una imagen en sus componente seno y coseno. La salida de la transformacion representa la imagen en el dominio de Fourier o dominio de frecuencia, mientras que la imagen de entrada esta en el dominio espacial. En este tipo de apliacion se suele utilizar variaciones del algoritmo de Fourier como la transformada de Fourier Bidimensional.

Al ilustrar imagenes sus espectros de Fourier son capaces de representar las funciones de la intesnsidad con niveles de gris.

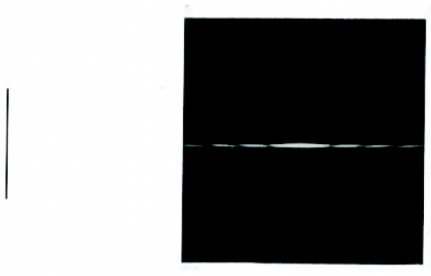


Figura 1: Transformada de Fourier en una linea

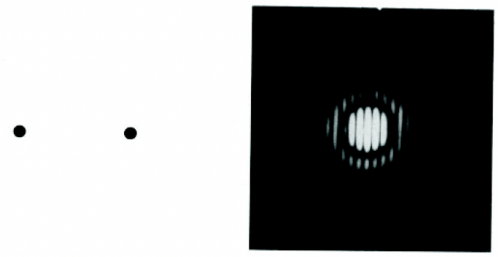
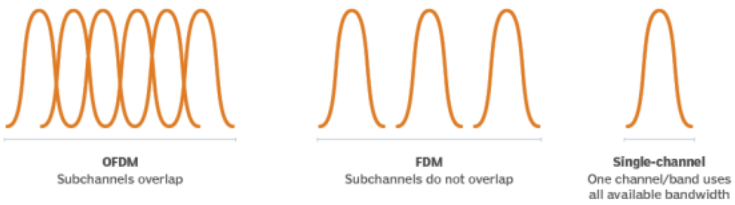


Figura 2: Transformada de Fourier en dos cirulos

3.3. Modulación y demodulación de símbolos de datos complejos para 5G, LTE, Wi-Fi, DSL y otros modernos sistemas de comunicacion

La multiplexación por división de frecuencia ortogonal es un método de transmisión de datos en el que un único flujo de información se divide entre varias frecuencias de subcanal de banda estrecha estrechamente espaciadas en lugar de una única frecuencia de canal de banda ancha. Se utiliza principalmente en la transmisión inalámbrica de datos, pero también se puede emplear en comunicaciones por cable y fibra óptica.



Siendo la base de estas tecnologias la trasmision de frecuencia, es indispensable el uso de los algoritmos FFTs para su manipulacion e interpretacion, pero sobretodo para lograr una optimizacion en este campo, ya que cuando hablamos de redes 5G, Wi-fi, etc, la latencia que pueden sufrir este tipo de conecciones es de vital importancia para conseguir mejor resultados.

3.4. Análisis de vibraciones

Cuando se toma una lectura de vibración, el objetivo es crear una lectura que representa toda la vibración de la máquina, y debe ser una medida repetible. Esto significa que si se tomara una medida y despues de un rato se toma la medida de nuevo, debería ser la misma. Aunque existan variaciones las lecturas deben ser muy similares, si no lo son, entonces la máquina ha cambiado drásticamente entre las lecturas o no está tomando suficientes promedios.

Esto toma una porción de tiempo, puesto que tiene que pasar por algo llamado ventanas, y a partir de eso calcula el FFT. Así que tenemos un espectro. Si hiciéramos sólo un promedio y repitiéramos las mediciones un momento después, definitivamente veríamos variación en el espectro.

4. Implementacion del algoritmo y presentacion del trabajo

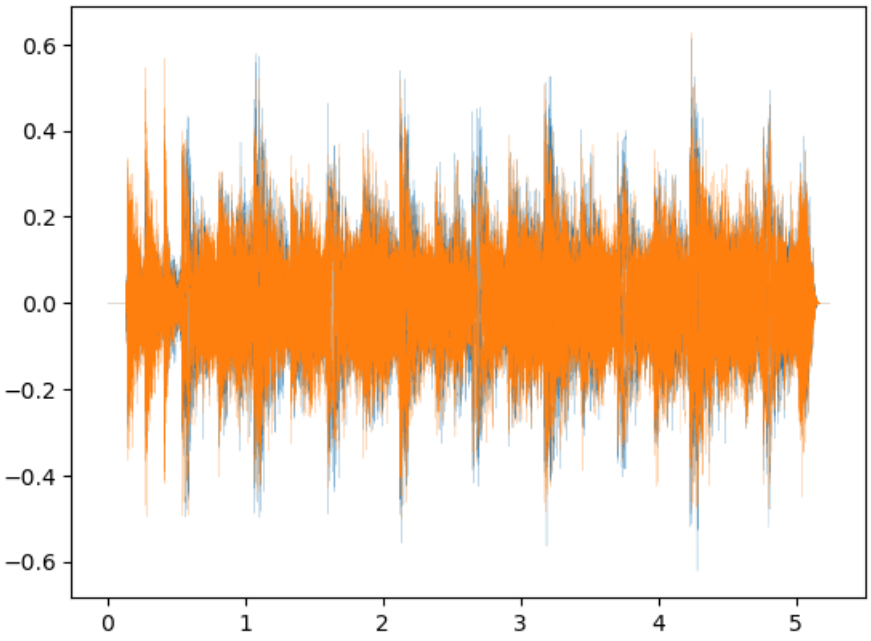
El lenguaje de programacion que se utilizo en esta implementacion es Julia ya que al momento de intentar realizar la implementacion en el lenguaje c++ hubo problemas al usar la libreria que nos permitia manipular los archivos de audio.

4.1. Desarrollo

EL algoritmo admite solo valores que tengan potencias de 2, por que si el vector o matriz no cumple estos requisitos es necesaria una tranformacion para que contenga n^2 elementos.

Para el trabajo utilizamos un audio en formato "wav", ya que el formato wav es un estándar de formato de archivo de audio, desarrollado por IBM y Microsoft, para almacenar un flujo de bits de audio en PC. Seguido de leer el archivo, procesdemos a realizar una grafica del audio seleccionado.

```
using PyPlot, WAV
s, fs = wavread("test1.wav")
aux = length(0:1/fs:(length(s)-1)/(fs))-length(s)
plot(aux/fs:1/(fs):(length(s)-1)/(2*fs), s,linewidth = 0.1)
wavplay(s,fs)
```



El comportamiento de los algoritmos FFT puede variar pero principalmente estan basados en el (paradigma divide y venceras) diviendo los datos recibidos en forma par e impar, llamandose nuevamente de forma recursiva para repetir el proceso hasta finalmente volver a combinar los datos.

Funcion empleada en el trabajo e implementada en legunaje Julia:

```
function myfft(a)
    temp = 1
    if typeof(a)==Matrix{Float64}
        a = vec(a)
    end
    while temp < length(a)
        temp*=2
    end
    resize!(a, temp)
    y1 = ComplexF64[]; y2 = ComplexF64[]
    n = length(a)
    if n ==1 return a end
```

```
    wn(n) = exp(-2*  *im/n)
    y_even = myfft(a[1:2:end])
    y_odd = myfft(a[2:2:end])
    w = 1
    for k in 1:Int(n/2)
        push!(y1, y_even[k] + w*y_odd[k])
        push!(y2, y_even[k] - w*y_odd[k])
        w = w*wn(n)
    end
    return vcat(y1,y2)
end
using FFTW
```

Al realizar la implementacion de este algoritmo, encontramos que tenia cierto fallos pues este solo funciona con archivo de un tamaño menor a 2 digitos, por lo cual obtamos utilizar el algortirmo ya implementado en las librerias para ser capaces de continuar con el trabajo requerido.

```
d=fft(s)
```

```
231424×2 Matrix{ComplexF64}:
 -42.2543-4.04329e-14im    -1.47151-1.45023e-14im
  3.10936-1.92242im          9.06348+0.416182im
  18.8111+23.6331im       -0.0466413+4.53529im
 -16.7973-12.3328im          9.81953-2.00071im
  1.68121+13.8415im          2.18645+9.01973im
  14.5935-24.8043im         -2.94689+6.6323im
  5.22395-0.914097im         19.6308+8.97008im
  3.94821+18.2277im         -11.0502+6.71307im
 -19.8169-23.4001im          10.7854-10.4987im
 -9.35293+8.32999im       -0.0274592+20.2041im
  2.84418+19.6367im           -5.0525+4.2867im
   34.189+13.7986im         -4.15567+3.00117im
  7.03233+8.3171im            3.58284-0.838796im
         ⋮
  7.03233-8.3171im            3.58284+0.838796im
   34.189-13.7986im         -4.15567-3.00117im
  2.84418-19.6367im           -5.0525-4.2867im
 -9.35293-8.32999im       -0.0274592-20.2041im
 -19.8169+23.4001im          10.7854+10.4987im
  3.94821-18.2277im         -11.0502-6.71307im
  5.22395+0.914097im         19.6308-8.97008im
  14.5935+24.8043im         -2.94689-6.6323im
  1.68121-13.8415im          2.18645-9.01973im
 -16.7973+12.3328im          9.81953+2.00071im
  18.8111-23.6331im       -0.0466413-4.53529im
  3.10936+1.92242im          9.06348-0.416182im
```

Mostramos los resultados al aplicar dichas funciones.

```
e = sqrt.(real(d).^2+imag(d).^2)
```

```
231424×2 Matrix{Float64}:
 42.2543    1.47151
  3.65566   9.07303
 30.2057    4.53553
 20.8386   10.0213
 13.9432    9.28095
 28.7788    7.25752
  5.30332  21.5831
 18.6504   12.9295
 30.6639   15.0515
 12.5246   20.2041
 19.8416    6.62598
 36.8685    5.12608
 10.8916    3.67972
  ⋮
 10.8916    3.67972
 36.8685    5.12608
 19.8416    6.62598
 12.5246   20.2041
 30.6639   15.0515
 18.6504   12.9295
  5.30332  21.5831
 28.7788    7.25752
 13.9432    9.28095
 20.8386   10.0213
 30.2057    4.53553
  3.65566   9.07303
```

Ahora presentamos el espectro generado al hacer uso de la funcion FFT en el audio elegido, para generar la grafica hacemos uso de la herramienta plot indicandoles los paremetros necesarios.

En la siguiente grafica es posible observar 2 espectros, uno de color azul y color anaranjado, las cuales corresponden a los audios definidos para una salida en especial que fue definida al momento de generar dichos valores, para hacer mas facil su comprension podemos decir que cada color representa en que lado de un audifono o bocina se va a reproducir dicho contenido, generando asi lo que conocemos comunmente como audio Stereo, la cual genera una mayor inmersion por parte de los usuarios al escuchar contenido multimedia.

Esta grafica es importante para poder entender de forma visual el cambio que se realizara en los datos y que se veran reflejados en los espectros mostrados.

```
plot(aux/fs:1/fs:(length(s)-1)/(2*fs), e, linewidth = 0.1)
```

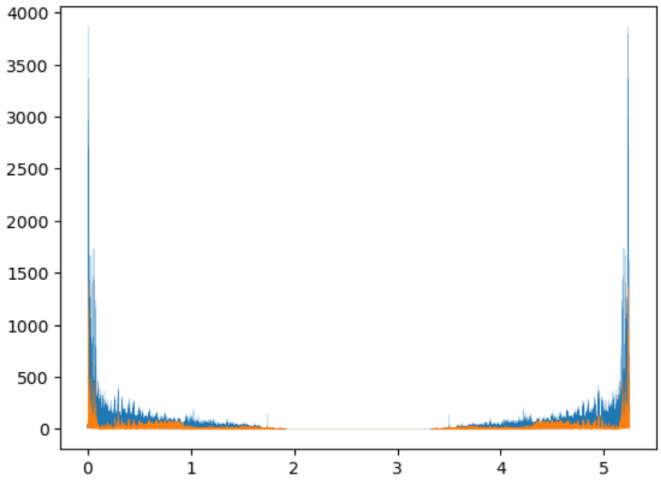


Figura 3: Espectro original

Realizamos modificaciones al spectro haciendo uso de la funcion FFT y mostramos su grafica.

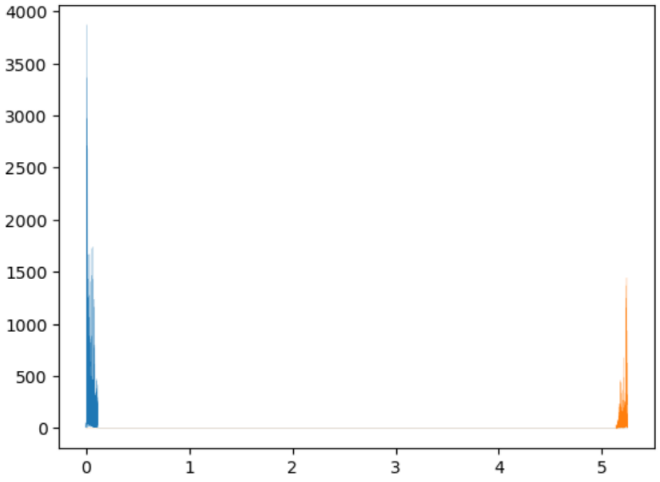


Figura 4: Espectro modificado

Procedimos a implementar la funcion Inversa ya que esta es muy similar a la funcion original, solo es necesrio aplicar algunos cambios, aunque en este caso hicimos uso de las funciones definidas para la implementacion de la funcion inversa.

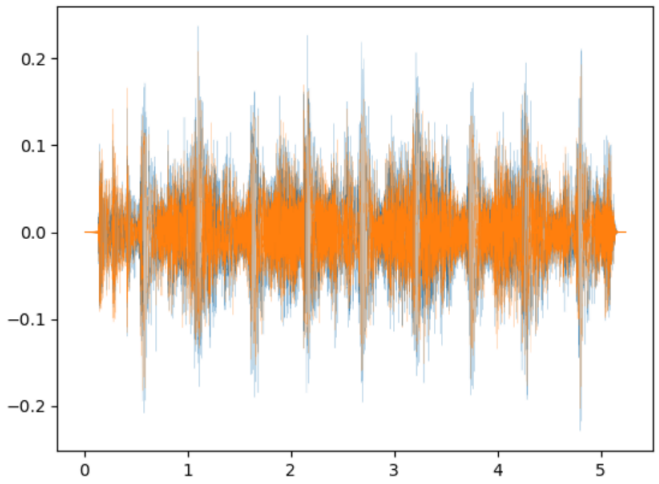
```
function myifft(a)
    a = conj(a)
    a = fft(a)
    a = conj(a)
    a ./= length(a)
end
```

Una vez teniendo la inversa implementada procedimos a usarla en el espectro previamente modificado para obtener una representacion grafica similar a la original y asi poder realizar comparacion visuales, pero sobretodo para ser capacer de reproducir el audio generado y escuchar los resultado generado por las modificaciones realizadas por el algoritmo de Fourier.

```
f = myifft(d)
```

```
231424×2 Matrix{ComplexF64}:
  0.000158977-0.000387359im   0.000218759-0.000201538im
  0.000198902-0.000347919im   0.000267068-0.000171239im
  0.000234359-0.000302295im   0.000309598-0.00013581im
  0.000264693-0.000251328im   0.000345563-9.59021e-5im
  0.000289346-0.000195955im   0.000374301-5.22499e-5im
   0.00030786-0.000137195im    0.00039528-5.65724e-6im
  0.000319895-7.61291e-5im    0.000408112+4.3018e-5im
  0.000325227-1.38815e-5im    0.000412559+9.28794e-5im
  0.000323757+4.84014e-5im    0.000408538+0.000143008im
  0.000315511+0.000109573im   0.000396121+0.000192481im
  0.000300638+0.000168505im   0.000375535+0.000240386im
  0.000279412+0.000224114im   0.000347157+0.00028584im
  0.000252223+0.000275373im   0.000311508+0.000328005im
                ⋮
 -0.000363633-0.000255964im  -0.000436439-7.71742e-5im
 -0.000335771-0.000306444im   -0.00039997-0.000118783im
  -0.00030253-0.000351525im  -0.000356996-0.000156239im
  -0.00026452-0.000390377im  -0.000308307-0.000188852im
 -0.000222441-0.000422281im    -0.0002548-0.000216021im
 -0.000177067-0.00044665im   -0.000197458-0.000237243im
 -0.000129235-0.000463031im  -0.000137338-0.000252127im
  -7.98237e-5-0.000471122im   -7.55458e-5-0.000260398im
  -2.97442e-5-0.000470772im   -1.32195e-5-0.000261902im
   2.00811e-5-0.000461985im    4.84929e-5-0.00025661im
   6.87344e-5-0.000444921im   0.000108455-0.000244619im
  0.000115319-0.000419893im   0.000165561-0.000226149im
```

Como resultado de aplicar la funcion inversa obtenemos datos que nos permiten generar un archivo .wav el cual somos capaces de repodrucir para ser capaces de apreciar la diferencia en el aspecto de audio, y tambien obtenemos la siguiente grafica que representa dicho audio.



5. Áreas de investigación

5.1. Big FFTs

En esta area la transformada rápida de Fourier FFT resulta de vital importancia puesto que reduce el tiempo de cálculo de n^2 pasos a $n \log_2(n)$. El único requisito es que el número de puntos en la serie tiene que ser una potencia de 2 (2 n puntos), por ejemplo 32, 1024, 4096, etc. Sim embargo es posible hacer uso de el algoritmo de Fourier de otras maneras sin la necesidad que el número de puntos sea

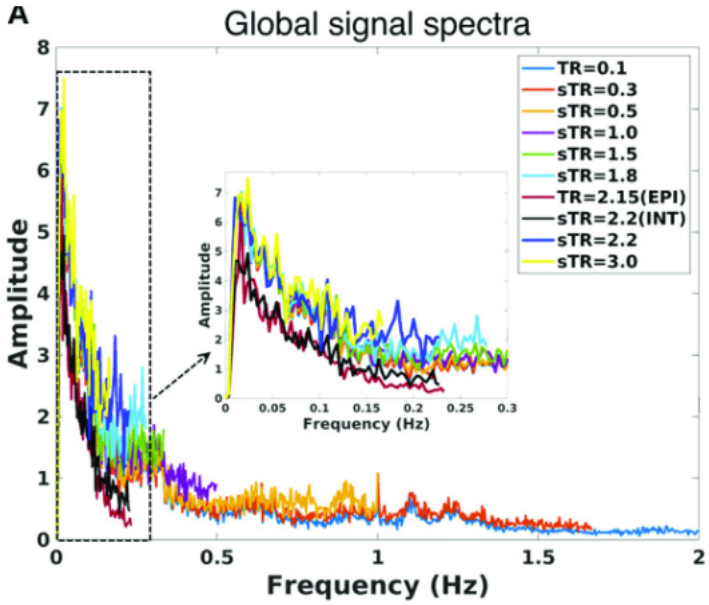
una potencia de 2, como el rellenar los datos incompletos con 0s.

5.2. Approximate FFTs

Una FFT aproximada para un diseño de área limitada en comparación con el diseño fijo convencional, el segundo algoritmo tiene como objetivo el rendimiento para lograr una frecuencia operativa más alta. Ambos algoritmos propuestos muestran que es posible un equilibrio eficiente entre la utilización del hardware y el rendimiento a nivel de etapa. Los diseños FFT aproximados propuestos se implementan en FPGA (Field-programmable gate array). Los resultados experimentales muestran que la utilización del hardware con el primer algoritmo aproximado que utiliza FFT se reduce en al menos casi un 40 %. el segundo algoritmo aumenta el rendimiento de los diseños en más de un 20 %. También se investiga el diseño de granularidad fina, donde los recursos de FPGA para un cálculo de FFT de 256 puntos se pueden reducir aún más en casi un 10 % en comparación con un diseño grueso.

5.3. Group FFTs

La FFT de Cooley-Tukey se puede interpretar como un algoritmo para el cálculo eficiente de la transformada de Fourier para grupos cíclicos finitos, un grupo compacto (el círculo) o el grupo no compacto de la línea real. Todas estas son instancias conmutativas de una "FFT de grupo".



5.4. Quantum FFTs

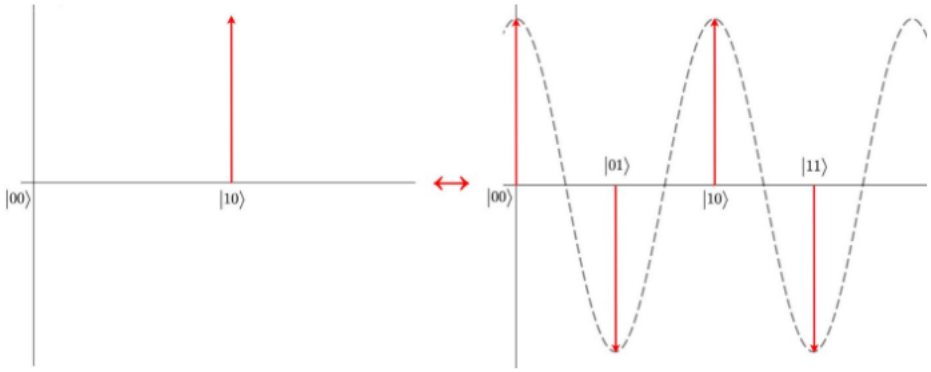
Es una transformación lineal en bits cuánticos y es el análogo cuántico de la transformada discreta de Fourier . La transformada cuántica de Fourier es parte de muchos algoritmos cuánticos , en particular el algoritmo de Shor para factorizar y calcular el logaritmo discreto , el algoritmo de estimación de fase cuántica para estimar los valores propios de un operador unitario y algoritmos para el problema del subgrupo oculto.

La transformada cuántica de Fourier se define como:

$$\sum_j a_j(j) \rightarrow \sum_k a'_k(k)$$

$$a'_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi i j k}{N_{aj}}}$$

La cual es capaz de convertir una función delta en una función sinusoidal.



6. Conclusiones

La Transformada de Fourier juega un papel muy importante en el PDI(Portable Database Image) y en cualquier campo en el que se utilicen frecuencias, ya que es una herramienta que nos permite obtener la representación de información en el espacio de frecuencias y aplicando un operador en éste dominio, ademas ya que la señal discreta (digital) es más fácil de transmitir, almacenar o manipular y tiene un bajo costo a través de internet, puede ser reenviada a su remitente o a algún otro destino. Esto sin que la señal sufra variaciones o alteraciones de calidad severas, pudiendo usarse sobre la imagenes, para detectar y realizar bordes, eliminar ruido, etc.