UNIVERSIDAD DE GRANADA

Facultad de Ciencias

Grado en Matemáticas

TRABAJO DE FIN DE GRADO

# Título del trabajo

Presentado por:
Nombre apellidos

Curso académico 2023-2024

# Título del trabajo

Nombre apellidos

Nombre apellidos *Título del trabajo*.
Trabajo de fin de Grado. Curso académico 2023-2024.

**Responsable de tutorización**

Nombre del tutor 1
*Departamento del tutor 1*

Nombre del tutor 2
*Departamento del tutor 2*

Grado en Matemáticas

Facultad de Ciencias

Universidad de Granada

## Declaración de originalidad

D./Dña. Nombre apellidos

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 13 de febrero de 2024

Fdo: Nombre apellidos

*Dedicatoria (opcional)*
*Ver archivo* `preliminares/dedicatoria.tex`

# Índice general

# Agradecimientos

Agradecimientos (opcional, ver archivo `preliminares/agradecimiento.tex`).

# Summary

An english summary of the project (around 800 and 1500 words are recommended).
File: `preliminares/summary.tex`

# Introducción

Estamos rodeados de señales.

Al aplicar la transformada de Fourier a una señal en el dominio del tiempo, obtenemos su representación en el dominio de la frecuencia. Esto nos permite examinar la señal original como una composición en términos de componentes sinusoidales.

Una imagen no es más que una señal visual y es por tanto susceptible de ser representada en el dominio de la frecuencia donde trabajando con la magnitud y la fase, no solo podremos recuperar la imagen original sino que ciertas tareas como extracción de características, compresión o eliminación de ruido se verán significativamente simplificadas.**¿?**

Dado que partimos de datos discretos y que los ordenadores solo pueden manejar sumas finitas, surge la imperiosa necesidad de definir en este ambiente una 'aproximación' de la Transformada de Fourier.

no se si deberia ir aqui esto la verdad

**Falta quizas poner ejemplos aunque ns si aqui o al final., y lo de 1 variable**

# Parte I.

# Parte Matemática

# 1. Transformada de Fourier

**Definición 1.1.**

# 2. Introducción

# Parte II.

# Segunda parte

# 3. Introducción

# 4. Transformada de Fourier Discreta

## 4.1. Motivación

En este capítulo, deduciremos la Transformada Discreta de Fourier (DFT), una herramienta crucial en el procesamiento digital de señales. Existen diversas aproximaciones que derivan en la expresión de la DFT.

Esta definición podría darse directamente y luego tratar de enlazarla con el concepto que teníamos para funciones integrables en $\mathbb{R}$, sin embargo, se opta por tratar de desarrollar una intuición partiendo del ámbito continuo. Esto se debe a que proporcionar una conexión conceptual con el dominio continuo contribuye a una comprensión más profunda y coherente de la transformada discreta de Fourier.

En cierta manera el lector puede pensar que trataremos de "transformar lo continuo en discreto".

## 4.2. 1-DFT

Sea $f : \mathbb{R} \to \mathbb{R}$ una función integrable en $\mathbb{R}$ que representará una señal. Haremos dos suposiciones acerca de esta.

- Suponemos que $f$ es de soporte compacto en $[0, L]$, $\mathrm{L} \in \mathbb{N}$
- Suponemos que $f$ es de soporte compacto en $[0, 2B]$, $\mathrm{B} \in \mathbb{N}$

Estamos considerando por tanto $f$ tanto limitada en el tiempo como limitada en banda, **sabiendo que esto solo puede ser aproximadamente cierto**. Sin embargo, finalmente vamos a llegar a una definición de una transformada discreta de Fourier que tendrá sentido por sí misma, independientemente de estas precarias suposiciones iniciales.

*Paso1*

Tomamos $N$ muestras equiespaciadas de $f$ en $[0, L]$ con

$$N = \frac{L}{\frac{1}{2B}} = 2BL$$

.

Obteniendo los puntos

$$t_0 = 0, \quad t_1 = \frac{1}{2B}, \quad t_2 = \frac{2}{2B}, \ldots, t_{N-1} = \frac{N-1}{2B}$$

.

Consideramos la versión discreta de $f$ como la lista $[f(t_0), f(t_1), \ldots, f(t_{N-1})]$.

En el contexto del **Teorema de Muestreo** es una buena elección como tasa de muestreo:

Paso2

Sea $g : \mathbb{R} \to \mathbb{R}$ definida como

$$g(t) = \sum_{n=0}^{N-1} \delta(t - t_n) \cdot f(t_n)$$

es la versión discreta de $f$ definida en $\mathbb{R}$

Construimos ahora $\widehat{g}$ como:

$$\widehat{g}(s) = \sum_{n=0}^{N-1} f(t_n) \cdot \widehat{\delta}(t - t_n) = \sum_{n=0}^{N-1} f(t_n) \cdot e^{-2\pi i s t_n}$$

Paso3

**Necesito justificar transformada de g**

Discretizamos $\widehat{g}$

Tomamos $N$ muestras equiespaciadas de $f$ $[0, 2B]$ con

$$N = \frac{2B}{\frac{1}{L}} = 2BL$$

.

Obteniendo los puntos

$$s_0 = 0, \quad s_1 = \frac{1}{L}, \quad s_2 = \frac{2}{L}, \ldots, s_{N-1} = \frac{N-1}{L}$$

.

Consideramos la versión discreta de $\widehat{g}$ como la lista $[\widehat{g}(s_0), \widehat{g}(s_1), \ldots, \widehat{g}(s_{N-1})]$.

Obteniendo finalmente:

$$\widehat{g}(s_n) = \sum_{k=0}^{N-1} f(t_k) e^{-2\pi i s_n t_k}, \quad n = 0, 1, \ldots, N-1$$

Paso4

$$Ff(s) = \int_0^L e^{-2\pi i s t} f(t)\, dt.$$

Así, en los puntos de muestra $s_m$,

$$Ff(s_m) = \int_0^L e^{-2\pi i s m t} f(t)\, dt,$$

y conocer los valores de $Ff(s_m)$ es conocer $Ff(s)$ razonablemente bien. Ahora usa los puntos de muestra $t_k$ para $f(t)$ y escribe una aproximación de suma de Riemann para la integral. El espaciado $\Delta t$ de los puntos es $1/2B$, entonces

$$Ff(s_m) = \int_0^L f(t) e^{-2\pi i s m t}\, dt \approx \frac{1}{2B} \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i s m t_n} \Delta t = \frac{1}{2B} \sum_{n=0}^{N-1} f(t_n) e^{-2\pi i s m t_n} = \frac{1}{2B} F(s_m).$$

**Definición 4.1.** Sea $f = (f[0], f[1], \ldots, f[N-1])$ una N-tupla. Definimos la Transformada de Fourier Discreta de $f$ como la N-tupla $F = (F[0], F[1], \ldots, F[N-1])$ definida como:

$$F[m] = \sum_{n=0}^{N-1} f[n] e^{-2\pi i \frac{mn}{N}}, \quad m = 0, 1, \ldots, N-1$$

De donde se sigue inmediatamente la siguiente definición.

**Definición 4.2.** Sea $F = (F[0], F[1], \ldots, F[N-1])$ una N-tupla. Definimos la Inversa de la Transformada de Fourier Discreta de $F$ como la N-tupla $f = (f[0], f[1], \ldots, f[N-1])$ definida como:

$$f[n] = \frac{1}{N} \sum_{m=0}^{N-1} F[m] e^{2\pi i \frac{mn}{N}}, \quad n = 0, 1, \ldots, N-1$$

Recibe el nombre de Inversa de la Transformada de Fourier Discreta ya que a partir de una comprobación directa obtenemos la identidad al componer ambas. **(hacer una dirección ) escala**

# 5. Transformada Rápida de Fourier

## 5.1. Motivación

La Transformada Rápida de Fourier (FFT) es un algoritmo eficiente para calcular la Transformada de Fourier Discreta y su inversa.
Su impacto en la era moderna es incalculable, prueba de ello es su inclusión por la revista IEEE de Computación en Ciencia e Ingeniería como uno de los 10 algoritmos más influyentes en el desarrollo y la práctica de la ciencia y la ingeniería en el siglo XX [ref]
El capítulo comienza contextualizando al lector con algunos referencias históricas del algoritmo.Posteriormente avanzaremos con el desarrollo teórico que sustenta la FFT donde usaremos fundamentalmente la teoría desarrollada previamente. Este análisis culminará con el estudio de la mejora en eficiencia en comparación con el método directo y una implementación recursiva para un caso particular.
Para finalizar el capítulo se abordarán algunas de las aplicaciones en diversos campos, particularmente en el ámbito de visión por computador.Con esto se pretende convencer al lector de que la FFT se encuentra prácticamente en cualquier aplicación que involucre el análisis y procesamiento de señales.

## 5.2. Historia

Nos remontamos a 1945, año en el que finaliza la la Segunda Guerra Mundial tras el lanzamiento de las primeras bombas atómicas usadas directamente sobre una población civil en Japón por parte de Estados Unidos. Este acontecimiento desencadenó en una imperiosa necesidad por parte de las naciones de controlar la proliferación nuclear al comprobar el grado de destrucción que podían provocar.
Como resultado, Estados Unidos celebró conversaciones con los soviéticos y otras naciones con capacidad nuclear con el objetivo final de detener la carrera de armamento nuclear.
Para hacerlo se trató de llegar a determinados acuerdos que prohibía la experimentación con energía nuclear.Sin embargo debido a la esperable desconfianza entre países, era necesario que cada nación tuviera la tecnología para identificar las pruebas nucleares realizadas por otros.
Los hidrófonos podían detectarlas bajo el mar y los átomos residuales podían identificarse en el cielo desde pruebas en tierra. Pero detectar las pruebas subterraneas era todo un desafío.
Una idea era analizar series temporales sismológicas obtenidas de sismómetros situados en distintos lugares estratégicos para estudiar estas señales y deducir si se trataba de un exprimento nuclear. Dentro de una señal compleja, hay muchas ondas sinusoidales de diferentes frecuencias, todas contribuyendo a formar el resultado final, y esto puede descifrarse si podemos aislar cada frecuencia. Para ello recurrieron a la DFT que permitía descomponer la señal en distintas frecuencias, sin embargo debido a la gran complejidad computacional que se necesitaba este método era imposible de poner en práctica incluso con los ordenadores actuales.

No fue hasta 1960,cuando James Cooley y John Tukey publicaron el algoritmo de FFT que reducía considerablemente el orden de complejidad de la DFT, pudiendo reducir el tiempo de cálculo de más de tres años a 35 minutos.
Trágicamente, era demasiado tarde para firmar una prohibición de pruebas completa, y las pruebas nucleares fueron forzadas bajo tierra, donde las pruebas aumentaron a una notable tasa de alrededor de una vez por semana.
Desde entonces, la FFT ha encontrado nuevos usos en casi todas las aplicaciones de comunicaciones y señalización de datos Sin embargo, tenía el potencial de ser mucho mayor: casi fue el algoritmo que detuvo la carrera armamentista nuclear, si hubiera llegado solo unos años antes.
Lo más fascinante es que la primera aparición de la Transformada Rápida de Fourier (FFT) data de 1807 del matemático Gauss. Sus intereses estaban en ciertos cálculos astronómicos (un área recurrente de aplicación de la FFT) relacionados con la interpolación de órbitas asteroidales a partir de un conjunto finito de observaciones equidistantes. Seguramente, la perspectiva de un cálculo manual laborioso y extenso era una buena motivación para el desarrollo de un algoritmo rápido. Sin embargo su trabajo no fue publicado y apareció únicamente en sus obras recopiladas como un manuscrito inédito [ref].
Algunos trabajos previos en los Cooley y Tukey se basaron se mencionan a continuación.
Un método eficiente para el cálculo de las interacciones de un experimento factorial $2^m$ fue introducido por Yates[1]. La generalización a $3^m$ fue proporcionada por Box et al. [1]. Good [2] generalizó estos métodos y proporcionó algoritmos elegantes, para los cuales una clase de aplicaciones es el cálculo de series de Fourier.
Otro precursor importante es el trabajo de Danielson y Lanczos, realizado en el servicio de la cristalografía de rayos X, otro usuario frecuente de la tecnología FFT. [ref]
A lo largo de la historia, estos y numerosos otros nombres han estado vinculados al algoritmo FFT culminado con Cooley y Tukey. .Esto puede servir como motivación para explorar no solo enfoques novedosos, sino también para revisitar de vez en cuando viejos documentos y descubrir la diversidad de trucos e ideas ingeniosas que se empleaban en una época en la que el cálculo era una tarea laboriosa. Quizás entre las ideas descartadas antes de la era de las computadoras electrónicas, podamos encontrar valiosas semillas para nuevos algoritmos."[ref]

**Definición 5.1.** Orden de Complejidad, definir y decir clases de complejodad y llamarle T y presentar los ordenes de complejidad que vienen a continaución importante. Y difernecia entre complejidad del problema y del algoritmo y tal

## 5.3. Método Directo

Si atendemos a la expresión de la DFT.El enfoque directo implica utilizar el método de fuerza bruta, mediante el cual debemos realizar cada sumatoria para cada elemento de la N-tupla cuya transformada deseamos calcular.
De este modo, dada una N-tupla $x = [x_0, .., x_{N-1}]$ calcular la DFT de $x$ implica para cada $j \in \{0, ..., N-1\}$ realizar $N$ operaciones. Por tanto para calcular la salida completa se necesitarían $N$ operaciones por cada índice $j \in \{0, ..., N-1\}$ lo que nos lleva a un total de $N^2$ operaciones. Y por tanto tiene un orden de complejidad cuadrático $T = O(N^2)$

**Ejemplo 5.1.** Si disponemos de una imagen de $2048 \times 2048$ (tamaño habitual), esto significaría realizar del orden de 17 mil millones de operaciones para calcular una única 2DFT excluyendo

las exponenciales que podrían calcularse una única vez y almacenarse.
Mientras que calcular la FFT 2-D de una imagen de $2048 \times 2048$ requeriría del orden de 92 millones de operaciones, lo cual representa una reducción significativa respecto a los mil millones de cálculos mencionados anteriormente.

*Observación* 5.1. En ambos casos se ha tenido en cuenta la factorización de una 2DFT en 1DFT para realizar una comparación justa.[insertar imagen grande ¿?]

*Observación* 5.2. Una operación consiste en una multiplicación y una suma de dos números complejos.

## 5.4. FFT

Sea $W_N = e^{\frac{-2\pi i}{N}}$ la $N$-ésima raíz de la unidad. La 1-DFT de $f = (f[0], f[1], \ldots, f[N-1])$ puede reescribirse como $F = (F[0], F[1], \ldots, F[N-1])$ donde:

$$F[j] = \sum_{k=0}^{N-1} f[k] W_N^{jk}, \quad j \in \{0, 1, \ldots, N-1\}$$

**Teorema 5.1.** *Sea $N$ compuesto con factorización $N = r_1 \cdot r_2$ con $r_1, r_2$ divisores naturales no triviales de $N$. La DFT de una $N$-tupla $f$ puede ser calculada en $T = O(N(r_1 + r_2))$*

*Demostración.* Reescribimos los índices $j, k$ como sigue:

$$j = j_1 r_1 + j_0 \quad j_0 \in \{0, 1, \ldots r_1 - 1\}, j_1 \in \{0, 1, \ldots r_2 - 1\}$$

$$k = k_1 r_1 + k_0 \quad k_0 \in \{0, 1, \ldots r_2 - 1\}, k_1 \in \{0, 1, \ldots r_1 - 1\}$$

Esto nos permite escribir la DFT en función de $j_0, j_1, k_0, k_1$.

$$\begin{aligned} F[(j_1, j_0)] &= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{(j_1 r_1 + j_0)(k_1 r_2 + k_0)} \\ &= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{(j_1 r_1 + j_0) k_1 r_2} W_N^{(j_1 r_1 + j_0) k_0} \\ &= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{j_0 k_1 r_2} W_N^{(j_1 r_1 + j_0)(k_0)} \end{aligned} \tag{5.1}$$

La última igualdad se justifica con:

$$W_N^{(j_1 r_1 + j_0) k_1 r_2} = W_N^{j_1 k_1 r_1 r_2} W_N^{j_0 k_1 r_2} = W_N^{j_1 k_1 N} W_N^{j_0 k_1 r_2} = W_N^{j_0 k_1 r_2}$$

donde se ha usado que $W_N$ es la raíz enésima de la unidad. Notamos que $f(k_1, k_0) W_N^{j_0 k_1 r_2}$ depende únicamente de $j_0$ y no de $k_0$. Esto nos permite definir la N-tupla $f_1$ como

$$f_1(j_0, k_0) = \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{j_0 k_1 r_2} \tag{5.2}$$

y podemos escribir (5.1) en función de $f_1$

$$F[(j_1, j_0)] = \sum_{k_0=0}^{r_2-1} f_1(j_0, k_0) W_N^{(j_1 r_1 + j_0)(k_0)} \tag{5.3}$$

Observamos que reescribiendo los índices hemos podido desacoplar las dos sumatorias. Esta es la clave del cálculo de la FFT.

Como $f_1$ es una $N$-tupla y cada elemento se calcula usando $r_1$ operaciones,necesitamos por un lado $Nr_1$ operaciones para el cáculo de $f_1$. Una vez obtenido $f_1$, $F$ es calculado en $r_2$ operaciones por cada elemento lo que nos deja un total de $Nr_2$ operaciones. En total el algoritmo regido por las ecuaciones (5.9), (5.8) es del orden de $O(N(r_1 + r_2))$ □

**Teorema 5.2.** *Sea $N$ compuesto con factorización $N = \gamma_1 \cdot \gamma_2 \cdot \ldots \cdot r_m$ con $\gamma_1, \gamma_2, \ldots, \gamma_m$ divisores naturales no triviales de $N$. La DFT de una $N$-tupla $f$ puede ser calculada en $T = O(N(\gamma_1 + \gamma_2 + \ldots + \gamma_m))$.*

*Demostración.* La demostración consiste en un razonamiento recursivo. Se tiene que si $N = \gamma_1 \cdot \gamma_2 \cdot \ldots \cdot \gamma_m$ con $\gamma_1, \gamma_2, \ldots, \gamma_m$ dando $m$ pasos en el algoritmo anteriormente descrito se deduce que $T = O(N(\gamma_1 + \gamma_2 + \ldots + \gamma_m))$. **NO ESTÁ BIEN**. Aplicaremos por tanto m veces el algoritmo descrito en [ref teorema anterior].Consideraremos en cada paso las variables $r_1$ y $r_2$ para una mayor claridad estas se irán renombrando en cada paso del algoritmo.Los índices $j_O, j_1, k_0, k_1$ serán los relativos al cálculo seguido en el caso anterior para la factorización $N = r_1 r_2$.

*Paso1*

Sea $N_1 = \underbrace{\gamma_1}_{r_1} \cdot \underbrace{\gamma_2 \cdot \ldots \cdot \gamma_m}_{r_2}$.

$$F[(j_1, j_0)] = \sum_{k_0=0}^{r_2-1} f_1(j_0, k_0) W_N^{(j_1 r_1 + j_0)(k_0)} = \sum_{k=0}^{r_2-1} f_1(k) W_N^{jk}$$

donde $f_2$ es calculado usando $\gamma_1 N$ operaciones mediante la siguiente expresión:

$$f_1(j_0, k_0) = \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{j_0 k_1 r_2}$$

*Paso2*

Sea $N_2 = \underbrace{\gamma_2}_{r_1} \cdot \underbrace{\gamma_3 \cdot \ldots \cdot \gamma_m}_{r_2}$. Aplicamos el proceso de nuevo ahora para la ecuación

$$F[j] = \sum_{k=0}^{N_2-1} f_1(k) W_N^{jk} \tag{5.4}$$

Lo que nos deja con

$$F[j0, j_1] = \sum_{k_0=0}^{r_2-1} f_2(j_0, k_0) W_N^{(j_1 r_1 + j_0)(k_0)} = \sum_{k=0}^{r_2-1} f_2(k) W_N^{jk} \tag{5.5}$$

donde $f_2$ es calculado usando $\gamma_2 N$ operaciones mediante la siguiente expresión:

$$f_2(j_0, k_0) = \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{j_0 k_1 r_2} \tag{5.6}$$

*Pasom*

Sea $N_m = \underbrace{\gamma_{m-1}}_{r_1} \cdot \underbrace{\gamma_m}_{r_2}$. Aplicamos el proceso de nuevo ahora para la ecuación

$$F[j] = \sum_{k=0}^{r_2-1} f_m(k) W_N^{jk} \tag{5.7}$$

Lo que nos deja con

$$F[(j_1, j_0)] = \sum_{k_0=0}^{r_2-1} f_2(j_0, k_0) W_N^{(j_1 r_1 + j_0)(k_0)} = \sum_{k=0}^{r_2-1} f_1(k) W_N^{jk} \tag{5.8}$$

donde $f_m$ es calculado usando $\gamma_{m-1} N$ operaciones mediante la siguiente expresión:

$$f_m(j_0, k_0) = \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{j_0 k_1 r_2} \tag{5.9}$$

Finalmente concluimos con $r_m N$ operaciones el cálculo de $F$.

Lo que nos deja un total de $O(N(r_1 + \dots r_m))$

Lo dejo apartado y ya está.

□

*Observación* 5.3. Si queremos minimizar el orden de complejidad $T$, debemos usar tantos factores de $N$ como sea posible. Es decir si $N = r_1 \cdot r_2 \cdot \ldots \cdot r_m$ y dado $j \in \{1, \dots m\}$ se tiene que $r_j = p_1 p_2$ con $p_1, p_2 > 1$ entonces $p_1 + p_2 < r_j$ salvo que $p_1 = p_2 = 2$. Y por consiguiente $O(N(r_1 + r_2 + \ldots + r_j + \ldots + r_m))$ tiene mayor orden de complejidad que $T = O(N(r_1 + r_2 + \ldots + p_1 + p_2 + \ldots r_m))$. La igualdad se da para $p_1 = p_2 = 2$, de donde deducimos que el factor 2 puede combinarse de manera indistinta sin pérdida alguna.

**hablar sobre la memoria tb**

Luego, al descomponer un número en factores, observamos una disminución del orden de complejidad. Por lo tanto, podemos inferir que los números con un mayor número de divisores proporcionarán una ganancia en eficiencia mayor.Esto motiva la siguiente definición:

**Definición 5.2.** Un número antiprimo o altamente compuesto es un entero positivo con más divisores que cualquier entero positivo más pequeño

Ramanujan (1915) enumeró 102 números altamente compuestos hasta 6746328388800, pero omitió algunos de ellos. Robin (1983) proporciona los primeros 5000 números altamente compuestos, y una encuesta exhaustiva se da por Nicholas (1988). Flammenkamp da una lista de los primeros 779674 números altamente compuestos.

**Ejemplo 5.2.** Son números antiprimos el 1, 2, 4, 6, 12, 24, 36, 48, 60.

**Proposición 5.1.** *Si $N = 2^{a_2}3^{a_3}\dots p^{a_p}$ es la factorización prima de un número altamente compuesto,* ***entonces****:*

- Los primos 2, 3, ..., p forman una secuencia de primos consecutivos,
- Los exponentes son no decrecientes, por lo que $a_2 \geq a_3 \geq \dots \geq a_p$, y

**No lo se bien falta buscar mas datos sobre estos numeros**

**Proposición 5.2.** *El algoritmo FFT proporciona ganancias significativas cuando el tamaño de entrada N es un número altamente compuesto.*

*Demostración.* El orden de complejidad de la FFT con una entrada de tamaño $N$ con $N = p_1^{a_1}p_2^{a_2}\dots p_m^{a_m}$ es $T = O(N(a_1 \cdot p_1 + a_2 \cdot p_2 + \dots a_m \cdot p_m))$
Luego,

$$\frac{T}{N} = a_1 \cdot p_1 + a_2 \cdot p_2 + \dots a_m \cdot p_m$$

□

Aplicando log en base 2 **(por qué)**

$$log_2 N = log_2(a_1) \cdot p_1 + log_2(a_2) \cdot p_2 + \dots log_2(a_m) \cdot p_m$$

De donde obtenemos

$$\frac{T}{Nlog(N)} = \frac{a_1 \cdot p_1 + a_2 \cdot p_2 + \dots a_m \cdot p_m}{log_2(a_1) \cdot p_1 + log_2(a_2) \cdot p_2 + \dots log_2(a_m) \cdot p_m}$$

Cuando el número $N$ es altamente compositivo el término de la derecha se acota por una constante $C$, y por tanto $\frac{T}{Nlog(N)} = C$, lo que implica que el orden de complejidad es de $T = O(Nlog(N)$. **esto deberíamos comprobarlo y quizás hablar sobre el tema de las ganancias distinguie para los numeros para los que hay mas ganancia y hacer alguna grafica** comprobar porque no tengo claro lo de los números Y grafica

**Proposición 5.3.** *Si tomamos $r_1 = r_2 = \dots = r_m = r$ entonces $m = log_r N$ y el número total de operaciones para el cálculo de la FFT con tamaño de entrada $N = r^m$ es $T(r) = rNlog_r N$*

*Demostración.* Se deduce del teorema 5.2 [ref]

□

El caso $r = 2$ y $r = 4$ es de especial interés a nivel computacional, ya que el algoritmo ofrece ciertas ventajas añadidas tanto en el direccionamiento como en las operaciones realizadas como consecuencia de la naturaleza binaria inherente a los ordenadores.

Detallaremos el caso $r = 2$. La lógica seguida durante el proceso es similar a la ya aplicada anteriormente.

Sea $N = 2^m$, expresamos los índices $j, k$ en [ref] como

$$j = j_{m-1} \cdot 2^{m-1} + \dots j_1 \cdot 2 + j_0, \quad j_i \in \{0, 1\} \quad \forall i \in \{0, \dots m-1\}$$

$$k = k_{m-1} \cdot 2^{m-1} + \dots k_1 \cdot 2 + k_0, \quad k_i \in \{0, 1\} \quad \forall i \in \{0, \dots m-1\}$$

La clave está en que esta expresión de $j, k$ coincide con la descomposición binaria de cada índice. De tal modo que $j_i, k_i$ refieren al bit que ocupa la posición $i$ de la representación

binaria de $j$ y $k$ respectivamente, y por tanto solo pueden tomar el valor $0, 1$. Escribiremos entonces [ref def] en función de sus índices.Luego, **no lo tengo claro**

$$F[(j_1, j_0)] = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \cdots \sum_{k_{m-1}=0}^{1} f(k_{m-1}, \ldots, k_0) W_N^{jk_{m-1}2^{m-1}+\ldots+jk_0}$$

Siguiendo un razonamiento similar a [ref]. Como consecuencia de $W_N^N = 1$.

$$W_N^{jk_{m-1}2^{m-1}} = W_N^{(j_{m-1}\cdot 2^{m-1}+\ldots j_1\cdot 2+j_0)k_{m-1}2^{m-1}} = W_N^{j_0 k_{m-1}2^{m-1}}$$

Esto nos permite desacoplar la suma interior $f_1$ y calcularla directamtente como

$$f_1(j_0, k_{m-2}, \ldots k_0) = \sum_{k_{m-1}=0}^{1} f(k_{m-1}, \ldots, k_0) W_N^{j_0 k_{m-1}2^{m-1}}$$

ya que sólo depende de $j_0$.
Se procede de igual forma con el resto de índices desacoplando las distintas sumatorias.
De manera que en general para $l \in \{1, \ldots m\}$ tendríamos

$$f_l(j_0, \ldots, j_{l-1}, k_{m-l-1}, \ldots k_0) = \sum_{k_{m-l}=0}^{1} f_{l-1}(j_0, \ldots, j_{l-2}, k_{m-l}, \ldots k_0) W_N^{(j_{l-1}2^{l-1}+\ldots+j_0)k_{m-l}2^{m-l}}$$

donde hemos usado

$$W_N^{jk_{m-l}2^{m-l}} = W_N^{(j_{m-1}\cdot 2^{m-1}+\ldots j_1\cdot 2+j_0)k_{m-l}2^{m-l}} = W_N^{j_0 k_{m-l}2^{m-l}}$$

Podemos desarrollar la sumatoria y usar que $e^{i\pi} = -1$ $e^{i\frac{\pi}{2}=i}$

$$\begin{aligned} &f_l(j_0, \ldots, j_{l-1}, k_{m-l-1}, \ldots, k_0) \\ &\quad = f_{l-1}(j_0, \ldots, j_{l-2}, 0, k_{m-l-1}, \ldots, k_0) \\ &\qquad + (-1)^{j_{l-1}} i^{j_{l-2}} f_{l-1}(j_0, \ldots, j_{l-2}, 1, k_{m-l-1}, \ldots, k_0) W_N^{(j_{l-3}2^{l-3}+\ldots+j_0)2^{m-l}} \qquad j_{l-1} \in \{0, 1\} \end{aligned}$$

De acuerdo con la convención de indexación, esto se almacena en una ubicación cuyo índice es

$$j_0 2^{m-1} + \cdots + j_{l-1}2^{m-l} + k_{m-l-1}2^{m-l-1} + \cdots + k_0$$

Se puede observar en [ref] que solo están involucradas en el cálculo las dos ubicaciones de almacenamiento con índices que tienen 0 y 1 en la posición del bit $2^{m-1}$.
Además la computación paralela está permitida, ya que la operación descrita por [ref] se puede llevar a cabo con todos los valores de $j_0, \cdots, j_{i-2}$ y $k_0, \cdots, k_{m-i-1}$ simultáneamente. En algunas aplicaciones, es conveniente usar (20) para expresar $A_i$ en términos de $A_{i-2}$, lo que equivale a un algoritmo con $r = 4$.
El último array $f_m$ calculado proporciona el resultado deseado de la siguiente manera.

$$F(j_{m-i}, \ldots, j_0) = f_m(j_0, \ldots, j_{m-1})$$

## 5.5. Algoritmo

Se habla de loque en general se busca y del padding. Hemos visto ya que el orden de complejidad $T = O(NlogN)$. Presentamos ahora una implementación recursiva del algoritmo concreto para $2^m$ que se utiliza mucho. y que la vimo aquí.

## 5.6. Rader

El algoritmo de Cooley-Tukey de la FFT no contempla para $N$ primo ninguna mejora. La mayoría de algoritmos de la FFT aplican en este caso un zero-padding a la potencia de 2 más cercana para aproximar la transformada de fourier continua. Sin embargo si lo que se quiere es calcular la DFT de una secuencia de elementos podemos recurrir al algoritmo de Rader que utiliza la teoría de números para proporcionar una alternativa en este caso.

Recordamos que $\mathbb{Z}_p = \mathbb{Z}/p$, esto es, el anillo cociente $\mathbb{Z}$ módulo el ideal $p\mathbb{Z}$, que es un cuerpo gracias a que $p\mathbb{Z}$ es un ideal maximal. Denotamos por $\mathbb{Z}_p^\times$ las unidades de $\mathbb{Z}_p$.

**Teorema 5.3.** *$\mathbb{Z}_p^\times$ es un grupo cíclico.*

**Teorema 5.4.** *(Teorema de Fermat)*

El algoritmo de Rader consiste por tanto en :
bla bla
primos de sophie germain y cadena ytal. ————————————————-

# 6. Aplicaciones

## 6.1. Motivación

**quizas esto deberia ser otro capitulo o deberia estar al final de lo de DFT** "Si aceleras cualquier algoritmo no trivial en un factor de un millón o más, el mundo encontrará rápidamente aplicaciones útiles para él "[ref]
Mostramos a continuación algunas de las posibles aplicaciones de la FFT en el campo de visión por computador.

## 6.2. Cálculo de Polinomios

## 6.3. Tratamiento de Imágenes

- Reducción de Ruido
- Compresión de Imágenes
- Pegado de Imágenes

## 6.4. Detección de Características

- Detección de Bordes
- Detección de puntos de interés

## 6.5. Aplicaciones en Redes Neuronales

El culmen del tfga teorema de convolucion y ta,

# A. Ejemplo de apéndice

Los apéndices son opcionales.

Este fichero `apendice-ejemplo.tex` es una plantilla para añadir apéndices al TFG. Para ello, es necesario:

- Crear una copia de este fichero `apendice-ejemplo.tex` en la carpeta `apendices` con un nombre apropiado (p.e. `apendice01.tex`).
- Añadir el comando `\input{apendices/apendice01}` en el fichero principal `tfg.tex` donde queremos que aparezca dicho apéndice (debe de ser después del comando `\appendix`).

# Glosario

La inclusión de un glosario es opcional.
Archivo: `glosario.tex`

$\mathbb{R}$ Conjunto de números reales.

$\mathbb{C}$ Conjunto de números complejos.

$\mathbb{Z}$ Conjunto de números enteros.

# Bibliografía