Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Triennale in Matematica

Analisi degli Algoritmi di FFT ed Applicazioni alla Moltiplicazione Rapida di Polinomi

Relatore
Prof. Wolfgang Erb
Università degli Studi di Padova

Correlatrice
Prof.ssa Michela Redivo-Zaglia
Università degli Studi di Padova

Laureando
Matteo Girardin

Matricola
1146104

Anno Accademico 2023/2024

19/04/2024

ad Alessia, Anna, Francesco e a tutti gli amici che mi hanno accompagnato in questo viaggio

Abstract

La Trasformata di Fourier, una delle scoperte scientifiche più rivoluzionarie del 19-esimo secolo, è l'operatore matematico fondamentale per poter analizzare segnali nel tempo. Se nella teoria quest'ultimi sono rappresentati da funzioni continue e periodiche, nella pratica l'analisi si basa su singoli campioni discreti: lo strumento matematico adeguato a tale studio è la Trasformata Discreta di Fourier (DFT).
Se da un lato tale trasformazione risolve il problema di implementare la Trasformata di Fourier al calcolatore, dall'altro presenta un grande difetto: l'enorme complessità computazionale.
L'obiettivo del seguente elaborato è di analizzare vari algoritmi ottimizzati che risolvono questa criticità, ovvero gli algoritmi di FFT, con implementazioni in Matlab, contestualizzando le varie scoperte nel periodo storico di riferimento. Inoltre, anche se questi metodi sono principalmente utilizzati in ambiti pratici, come, ad esempio, nell'elaborazione di audio ed immagini, nelle telecomunicazioni, nello spettroscopia, si conclude descrivendo una loro originale applicazione algebrica alla moltiplicazione rapida di polinomi, e il conseguente impatto che tale procedimento ha avuto nell'ottimizzazione del calcolo di prodotti di numeri interi.

Sommario

Lista degli Acronomi

FT . . . . . Fourier Transform (Trasformata di Fourier)

IFT . . . . . Inverse Fourier Transform (Antitrasformata di Fourier)

DFT . . . . . Discrete Fourier Transform (Trasformata Discreta di Fourier)

IDFT . . . . . Inverse Discrete Fourier Transform (Antitrasformata Discreta di Fourier)

FFT . . . . . Fast Fourier Transform (Trasformata Veloce di Fourier)

IFFT . . . . . Inverse Fast Fourier Transform (Antitrasformata Veloce di Fourier)

PFA . . . . . Prime Factor Algorithm (Algoritmo dei Fattori Primi)

CRT . . . . . Chinese Remainder Theorem (Teorema Cinese del Resto)

Introduzione

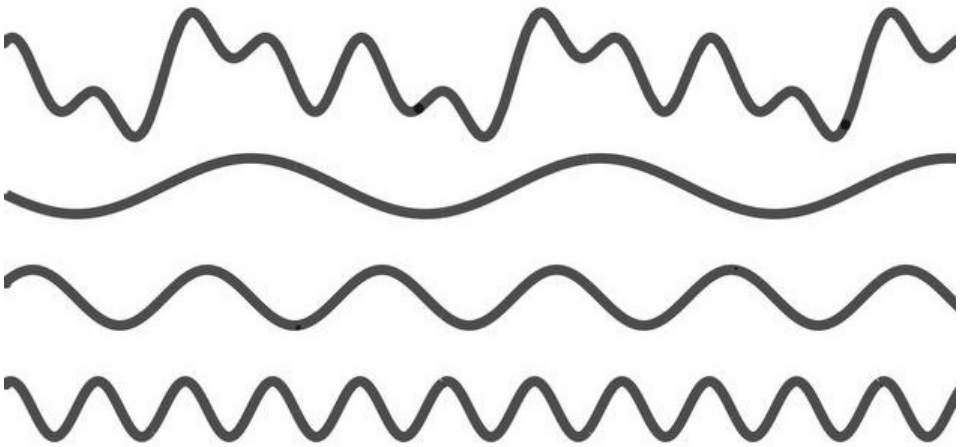
Cosa collega un brano musicale, il wi-fi, un elettrocardiogramma, un'immagine digitale e lo scoppio delle bombe atomiche? Tutti questi elementi, per essere funzionali, ottimizzati, o semplicemente esplorati, richiedono l'analisi di segnali nel tempo. Lo strumento che si pone al servizio di tale scopo è una delle scoperte matematiche più importanti e rivoluzionarie del 19-esimo secolo: la Trasformata di Fourier (FT).
Se nella teoria tale operazione non lascia spazio a rivali, nella pratica è di insidioso utilizzo. Infatti, mentre in astratto i segnali si possono pensare come delle perfette onde continue, nella realtà essi sono finiti e costituiti da singoli campioni. Ecco che, in una qualsiasi applicazione, si ricorre all'utilizzo della Trasformata Discreta di Fourier (DFT).
Tale trasformazione risponde all'esigenza di implementare al calcolatore la FT di una funzione del tempo, con un limite non trascurabile: la grande complessità computazionale.
Per risolvere questo problema, nella seconda metà del '900, due scienziati che lavoravano all'identificazione di esplosioni atomiche, scoprirono un algoritmo che ottimizzava il calcolo della DFT. Ci si riferisce a James William Cooley e John Wilder Tukey, e al primo algoritmo per ridurre drasticamente il numero di operazioni necessarie all'elaborazione del metodo: la Trasformata Veloce di Fourier (FFT).
L'obiettivo del seguente elaborato è di analizzare vari algoritmi di FFT, con implementazioni in Matlab, contestualizzando le varie scoperte nel periodo storico di riferimento, e di mostrarne un'applicazione algebrica.
Più precisamente, nel primo capitolo viene esposta la teoria degli spazi $\mathcal{L}^p$, l'ambiente funzionale ideale in cui definire la Trasformata di Fourier ed enunciare il Teorema di Inversione, fondamentale per provare l'esistenza dell'Antitrasformata di Fourier.
Nel secondo capitolo, in primo luogo vengono presentate la Trasformata Discreta di Fourier e l'Antitrasformata Discreta di Fourier, trasformazioni che consentono di implementare al calcolatore l'Analisi di Fourier, operando su determinati valori discreti assunti dalla funzione in particolari punti, sottolineando come esse rappresentino un'approssimazione di FT e IFT utilizzando la Formula dei Trapezi. In secondo luogo, dopo aver stimato la complessità computazionale della DFT, si introduce la Trasformata Veloce di Fourier, l'algoritmo ottimizzato per calcolare la DFT, che abbassa il numero di operazioni da $\mathcal{O}(N^2)$ a $\mathcal{O}(N\log_2 N)$. Segue

l'esposizione di tre algoritmi di FFT: l'algoritmo di Cooley-Tukey *radix-2*, in cui si computa la FFT con un approccio *divide et impera*; l'algoritmo di Gentleman-Sande *radix-2*, in cui ci si discosta dal primo anteponendo la fase di calcolo a quella di riordinamento; l'algoritmo di Good-Thomas, in cui si utilizza il Teorema Cinese del Resto.
Nel terzo capitolo si analizza l'applicazine algebrica dell'algoritmo di Cooley-Tukey alla moltiplicazione rapida di polinomi, sfruttando le proprietà di simmetria intrinseche di ogni polinomio, in seguito alla valutazione strategica nelle varie radici complesse dell'unità. Si conclude mostrando come questo approccio sia stato decisivo nell'ottimizzare la moltiplicazione tra numeri interi, percorrendo i vari risultati ottenuti da Karatsuba, Schönhage e Strassen, fino alla recente pubblicazione di Harvey e van der Hoeven.

1

La Trasformata di Fourier

«Les causes primordiales ne nous sont point connues; mais elles sont assujetties à des lois simples et constantes, que l'on peut découvrir par l'observation, et dont l'étude est l'objet de la philosophie naturelle.»[1] [7]



[1] J. B. J. Fourier, Théorie analytique de la chaleur (1822), p.i
«Le cause primordiali non ci sono note; ma sono soggette a leggi semplici e costanti, che si possono scoprire mediante l'osservazione, e il cui studio è oggetto della filosofia della natura.» trad.

1.1 Cenni storici

Jean-Baptiste Joseph Fourier nacque ad Auxerre nel 1768. Orfano dall'età di 9 anni, egli fu cresciuto ed educato presso una scuola militare dell'ordine dei Benedettini, dove si appassionò parallelamente alla Matematica e alla vita militare. Sebbene fosse destinato a diventare monaco, lo scoppio della Rivoluzione Francese lo liberò da una vita che non lo appassionava e gli consentì di partecipare attivamente all'attività politica, rischiando anche di essere ghigliottinato. L'allora generale Napoleone riconobbe in lui una figura di spicco, tanto che lo nominò docente di Matematica all'École Polytechnique, cattedra che era stata del suo maestro Pierre-Simon Laplace, finchè non decise di portarlo con sé nella *campagna di civilizzazione* d'Egitto del 1798. Tra il sole cocente e le sabbie ardenti del deserto egiziano, in quei tre anni che lo videro impegnato in incarichi diplomatici, Fourier maturò un interesse quasi maniacale nei confronti del calore. Egli, infatti, riteneva che avesse proprietà curative e, per questo motivo, una volta tornato in Francia, era solito impostare la temperatura del riscaldamento dei suoi uffici a livelli alti.

Nei 21 anni successivi, in cui ricoprì il ruolo di prefetto della regione di Grenoble, conciliò sempre i suoi doveri amministrativi con gli studi sulla propagazione del calore, finchè, nel 1822, pubblicò la *Thèorie analytique de la chaleur*.

In questo trattato presentò un resoconto completo delle sue ricerche, dove introdusse l'idea che una qualsiasi funzione, anche discontinua, potesse essere rappresentata come somma di funzioni sinusoidali: uno strumento che utilizzò per analizzare la diffusione del calore nella materia.

La sua intuizione non lasciò indifferenti i grandi matematici dell'epoca. Da un lato, Fourier fu molto contestato dai membri dell'accademia di Parigi, tra cui il suo stesso maestro Pierre-Simon Laplace, soprattutto per le dimostrazioni poco rigorose; ma, d'altra parte, ci fu chi vide in lui un visionario: Lord Kelvin definì la sua opera *un grande poema matematico*. Ma Fourier non si fermò, e continuò ad ampliare questa teoria, espandendo il suo studio anche alle funzioni non periodiche.

Da questa storia travagliata iniziò a prendere vita uno degli strumenti matematici più influenti e più importanti nello sviluppo tecnologico dell'ultimo secolo, essendo alla base della Teoria dell'Informazione: la *Trasformata di Fourier*.

1.2 La Teoria degli Spazi $\mathcal{L}^p$

Prima di procedere con l'analisi della *Trasformata di Fourier*, è opportuno illustrare brevemente gli spazi $\mathcal{L}^p$, l'ambiente ideale per definire tale operazione, nonchè la struttura matematica appropriata per formulare e dimostrare il *Teorema di Inversione* per le loro proprietà di completezza e convergenza.

Definizione 1.1 (Spazi $\mathcal{L}^p$) *Sia $X \subseteq \mathbb{R}^\nu$ un insieme misurabile, e sia $p \in [1; +\infty]$. Si definisce lo spazio funzionale $\mathcal{L}^p$ come lo spazio della funzioni a p-esima potenza sommabile, ovvero:*

$$\mathcal{L}^p := \left\{ f : X \to \mathbb{R} \vee \mathbb{C} : f \text{ misurabile}, \int_X |f|^p < +\infty \right\}. \tag{1.1}$$

Le varie funzioni f possono considerarsi a valori reali o complessi: la teoria resta invariata.
Inoltre, si noti che gli spazi $\mathcal{L}^p$, con le usuali operazioni di somma e prodotto scalare, sono spazi vettoriali su $\mathbb{C}$.
In tali spazi viene definita la semi-norma[2] $\| \cdot \|_{\mathcal{L}^p}$ in modo naturale come segue:

Definizione 1.2 (Semi-norma $\mathcal{L}^p$) *Siano $X \subseteq \mathbb{R}^\nu$ un insieme misurabile e $f \in \mathcal{L}^p(X)$. Allora la semi-norma $\| \cdot \|_{\mathcal{L}^p}$ è:*

$$\|f\|_p \equiv \|f\|_{\mathcal{L}^p} := \left(\int_X |f|^p \right)^{\frac{1}{p}}. \tag{1.2}$$

Osservazione 1.1: Il fatto che sia una semi-norma viene dalla *disuguaglianza di Minkowsky*:

$$\|f + g\|_p \leq \|f\|_p + \|g\|_p$$

Diventa quindi immediato identificare gli spazi $\mathcal{L}^p$ come gli spazi delle funzioni con semi-norma $\mathcal{L}^p$ finita.

[2] Considerato X spazio vettoriale, una semi-norma è una funzione $\| \cdot \| : X \to [0; +\infty]$ tale che:

1. $\|x\| \geq 0 \ \ \forall x \in X$ (ma $\|x\| = 0 \not\Rightarrow x = 0$);
2. $\|\lambda x\| = \lambda \|x\| \ \ \forall x \in X, \forall \lambda \in \mathbb{C}$;
3. $\|x + y\| \leq \|x\| + \|y\| \ \ \forall x, y \in X$.

Inoltre, si osservi che per ogni $1 \leq p \leq +\infty$ i suddetti sono spazi di Banach[3], e, in particolare, $\mathcal{L}^2$ è uno spazio di Hilbert[4].
La trattazione prosegue con l'esposizione dei risultati utili per definire le prime proprietà della *Trasfromata di Fourier* e per dimostrare il *Teorema di Inversione*.

Teorema 1.1 (Continuità in media in $\mathcal{L}^p$) *Sia $f \in \mathcal{L}^p(\mathbb{R}^\nu)$, con $1 \leq p \leq +\infty$. Allora:*

$$\| f(x+h) - f(x) \|_p \xrightarrow[h \to 0]{} 0 \tag{1.3}$$

Dimostrazione: Sia $\varepsilon > 0$, allora esiste $g \in \mathcal{L}^p(\mathbb{R}^\nu)$ tale che $\|f - g\|_p < \varepsilon$.
Si ottiene:

$$\|f(x+h) - f(x)\|_p \leq \|f(x+h) - g(x+h)\|_p + \|g(x+h) - g(x)\|_p + \|g(x) - f(x)\|_p < 3\varepsilon$$

dove:

$$\|g(x+h) - g(x)\|_p < \varepsilon$$

segue dalla compattezza del supporto di g e dal *Teorema di Heine-Cantor*. □

Teorema 1.2 (di Lebesgue in $\mathcal{L}^p$) *Sia $\{f_n\}_{n \in \mathbb{N}}$ una successione di funzioni misurabili in X tale che $f_n \xrightarrow[n \to +\infty]{} f$ e $|f_n(x)| \leq g(x)$ per q.o. $x \in X$, dove $g \in \mathcal{L}^p$. Allora $\{f_n\}_{n \in \mathbb{N}} \subset \mathcal{L}^p, f \in \mathcal{L}^p$ e si ha $\|f_n - f\|_p \xrightarrow[n \to +\infty]{} 0$.*

Teorema 1.3 (Completezza di $\mathcal{L}^p$) *Sia $\{f_n\}_{n \in \mathbb{N}} \subset \mathcal{L}^p$ successione di Cauchy[5] e sia $\varepsilon > 0$ fissato. Allora:*

1. *esiste $\{f_{n_j}\}_{j \in \mathbb{N}}$ sottosuccesione tale che $f_{n_j} \xrightarrow[j \to +\infty]{} f$ quasi ovunque;*
2. *esiste $X_\varepsilon \subseteq X$ misurabile tale che $\mu(X \setminus X_\varepsilon) < \varepsilon$ e $f_{n_j} \xrightarrow[j \to +\infty]{} f$ uniformemente su X_ε;*
3. *$\|f_n - f\|_p \xrightarrow[n \to +\infty]{} 0$.*

[3] Uno spazio di Banach è uno spazio normato completo rispetto alla metrica indotta dalla norma.
[4] Uno spazio di Hilbert è uno spazio vettoriale completo secondo la norma indotta da un certo prodotto scalare.
[5] Si dice che una successione $\{f_n\}_{n \in \mathbb{N}} \subset \mathcal{L}^p$ è di Cauchy se e solo se $\forall \varepsilon > 0 \, \exists \overline{n} \in \mathbb{N} : \forall n, m > \overline{n} \Rightarrow \|f_n - f_m\|_p < \varepsilon$

Con questa breve introduzione si sono definite le varie proprietà analitiche degli spazi $\mathcal{L}^p$: si può quindi procedere nel definire la *Trasformata di Fourier* in $\mathcal{L}^1$ ed analizzarne i principali risultati.

1.3 La Trasformata di Fourier

La *Trasformata di Fourier* è un operatore lineare integrale che, ad una funzione di partenza, associa un'altra funzione a valori reali o complessi. Nella pratica, tale operazione consente di scomporre un'onda qualsiasi, anche particolarmente complessa o affetta da *rumore*, in sotto-componenti di facile analisi: delle funzioni sinusoidali. Di tali componenti, infatti, risulta più agevole studiarne le caratteristiche principali, ovvero l'ampiezza, la fase e la frequenza.
Dato che un'infinità di fenomeni fisici, geologici, chimici, sono descritti o modellizzati da onde[6], la *Trasformata di Fourier* risulta essere una delle intuizioni matematiche più importanti e più applicate nell'analisi di segnali e grandezze periodiche nel tempo.

Definizione 1.3 (Trasformata di Fourier) *Sia $f \in \mathcal{L}^1(\mathbb{R}^\nu)$. La Trasformata di Fourier di f è la funzione $\widehat{f}$ definita come:*

$$\widehat{f}(\xi) := \int_{\mathbb{R}^\nu} e^{-i\langle x,\xi\rangle} f(x)\, dx \tag{1.4}$$

L'integrale in (1.4) è ben definito, infatti si ha:

$$|\widehat{f}(\xi)| \leq \int_{\mathbb{R}^\nu} \left|e^{-i\langle x,\xi\rangle} f(x)\right|\, dx = \int_{\mathbb{R}^\nu} |f(x)|\, dx = \|f\|_1$$

In particolare, così facendo si dimostra che $|\widehat{f}(\xi)| < +\infty$, poichè $f \in \mathcal{L}^1(\mathbb{R}^\nu)$ per ipotesi e, ulteriormente, che $|\widehat{f}|$ è una funzione globalmente limitata.
Le principali proprietà analitiche di tale operatore sono di seguito riportate.

Proposizione 1.1 *Sia $f \in \mathcal{L}^1(\mathbb{R}^\nu)$. Allora la sua Trasformata di Fourier $\widehat{f}$ è uniformemente continua.*

[6] Ad esempio la luce, i suoni, i fluidi, i terremoti, le onde elettromagnetiche, le trasmissioni radio.

Dimostrazione:

$$\begin{aligned}|\widehat{f}(\xi+h)-\widehat{f}(\xi)| &= \left|\int_{\mathbb{R}^\nu}(e^{-i\langle x,\xi+h\rangle}-e^{-i\langle x,\xi\rangle})f(x)\,dx\right| \\ &\leq \int_{\mathbb{R}^\nu}|e^{-i\langle x,\xi\rangle}||e^{-i\langle x,h\rangle}-1||f(x)|\,dx \\ &= \int_{\mathbb{R}^\nu}|e^{-i\langle x,h\rangle}-1||f(x)|\,dx \xrightarrow[h\to 0]{} 0.\end{aligned}$$

poichè $|e^{-i\langle x,\xi\rangle}| = 1$ e $|e^{-i\langle x,h\rangle}-1| \xrightarrow[h\to 0]{} 0$. □

Teorema 1.4 (di Riemann-Lebesgue) *Sia $f \in \mathcal{L}^1(\mathbb{R}^\nu)$. Considerata la sua Trasformata di Fourier $\widehat{f}$ si ha che:*

$$\widehat{f}(\xi) \xrightarrow[|\xi|\to+\infty]{} 0.$$

Dimostrazione: Considerata la definizione di Trasformata di Fourier come in (1.4), si ha che:

$$\begin{aligned}\widehat{f}(\xi) = \int_{\mathbb{R}^\nu} e^{-i\langle x,\xi\rangle}f(x)\,dx &\overset{x=x+h}{=} \int_{\mathbb{R}^\nu} e^{-i\langle x+h,\xi\rangle}f(x+h)\,dx = \\ &= e^{-i\langle h,\xi\rangle}\int_{\mathbb{R}^\nu} e^{-i\langle x,\xi\rangle}f(x+h)\,dx\end{aligned}$$

Si scelga $h \in \mathbb{R}^\nu$ in modo che $e^{-i\langle h,\xi\rangle} = -1$, ossia $\langle h,\xi\rangle = \pi$.
Se $h,\xi \neq 0$, si ottiene che

$$h = \pi\frac{\xi}{|\xi|^2}$$

Allora:

$$\begin{aligned}\widehat{f}(\xi) &= -\int_{\mathbb{R}^\nu} e^{-i\langle x,\xi\rangle}f\left(x+\pi\frac{\xi}{|\xi|^2}\right)\,dx \\ \Rightarrow 2\widehat{f}(\xi) &= \int_{\mathbb{R}^\nu} e^{-i\langle x,\xi\rangle}\left[f(x)-f\left(x+\pi\frac{\xi}{|\xi|^2}\right)\right]\,dx \\ \Rightarrow \left|2\widehat{f}(\xi)\right| &\leq \int_{\mathbb{R}^\nu}\left|e^{-i\langle x,\xi\rangle}\left[f(x)-f\left(x+\pi\frac{\xi}{|\xi|^2}\right)\right]\right|\,dx \xrightarrow[|\xi|\to+\infty]{} 0.\end{aligned}$$

dove si è usato il risultato (1.3) del Teorema (1.1) della Continuità in media di $\mathcal{L}^1$. □

1.4 L'Antitrasformata di Fourier

L'*Antitrasformata di Fourier* è una trasformazione integrale che associa una funzione f alla propria *Trasformata di Fourier* $\widehat{f}$, mediante un'integrazione. Concettualmente, si ottiene ripristinando la funzione originale a partire dalla sua trasformata.
L'importanza della *Trasformata Inversa di Fourier* è cruciale, riflettendo il valore fondamentale della sua controparte diretta. Infatti, essa consente di ricostruire le funzioni di partenza, precedentemente divise in sotto-componenti per analizzarne le caratteristiche.

Definizione 1.4 (Antitrasformata di Fourier) *Sia $\widehat{f} \in \mathcal{L}^1(\mathbb{R}^\nu)$. L'Antitrasformata di Fourier di $\widehat{f}$ è la funzione f definita come:*

$$f(x) := \frac{1}{2\pi} \int_{\mathbb{R}^\nu} e^{i\langle x,\xi\rangle} \widehat{f}(\xi)\, d\xi \tag{1.5}$$

L'esistenza e la rigorosità di tale operazione è assicurata dal *Teorema di Inversione*, uno dei risultati principali dell'analisi di Fourier. Tale Teorema vale negli *spazi di Schwartz*[7], ma può essere di conseguenza applicato allo spazio $\mathcal{L}^1$, centrale nella trattazione di questo elaborato.

Teorema 1.5 (di Inversione in $\mathcal{L}^1$) *Siano $f,\ \widehat{f} \in \mathcal{L}^1(\mathbb{R}^\nu)$. Allora, quasi ovunque, vale la formula di inversione:*

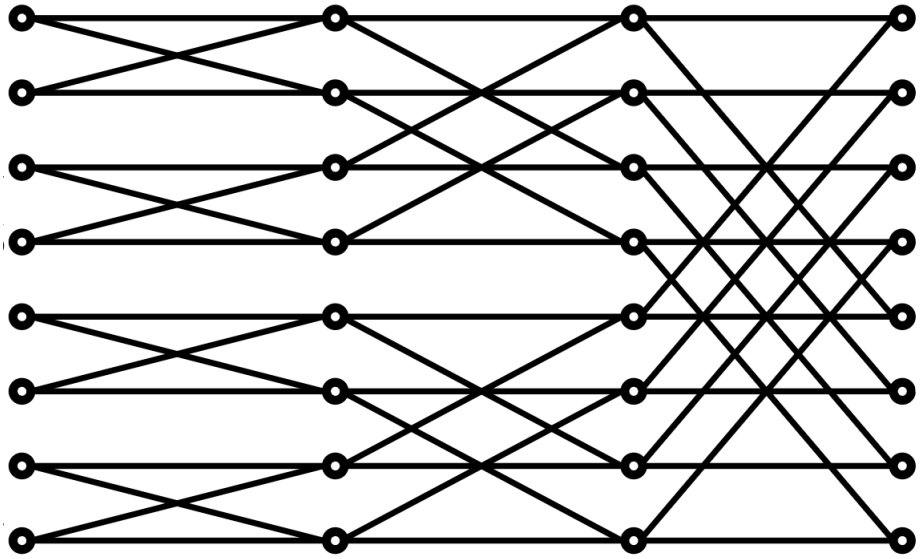
$$\widehat{\widehat{f}}(x) = (2\pi)^\nu f(-x).$$

[7] Lo spazio di Schwartz (o spazio delle funzioni a decrescenza rapida) è lo spazio delle funzioni lisce che decrescono, assieme alle proprie derivate, più velocemente di una qualsiasi potenza di $\frac{1}{x}$

2

La Trasformata Veloce di Fourier

«The most important numerical algorithm in our lifetime.»[1] [20]



[1] G. Strang, "Wavelets", *American Scientist*, vol. 82, no. 3, 1994, pp. 250–255. JSTOR
«L'algoritmo numerico più importante della nostra epoca.» trad.

2.1 Cenni storici

Correva l'anno 1945, quando nelle mattine del 6 e del 9 Agosto l'aeronautica militare statunitense sganciò le bombe atomiche *Little Boy* e *Fat Man* sulle città di Hiroshima e Nagasaki, causando centinaia di migliaia di vittime e provocando danni, direttamente e indirettamente, senza precedenti. Per questi motivi fu la prima e unica volta che in una guerra furono utilizzati ordigni di tale portata. Tuttavia, questo episodio scatenò la ricerca e lo sviluppo di bombe atomiche da parte delle altre potenze mondiali. Per placare la corsa agli armamenti, Canada e Regno Unito richiesero un incontro al fine di discutere quanto sarebbe stato predisposto in ambito nucleare, e, contrariamente ad ogni aspettativa, gli Stati Uniti si dimostrarono aperti al dialogo. Compresero, infatti, che non sarebbero stati l'unico Paese a possedere armi nucleari e, pertanto, proposero di smantellarle tutte, a condizione che anche le altre nazioni si impegnassero a non fabbricarne mai. Questa proposta fu denominata il *Piano Baruch* e suggeriva l'introduzione di un organismo internazionale che avrebbe avuto il controllo completo di tutti i materiali radioattivi sulla Terra, dalle fasi di estrazione e raffinazione fino all'utilizzo di tali materiali a scopi pacifici, come la produzione di energia nucleare. Ad ogni modo, i Sovietici respinsero la proposta, considerandola come un ulteriore tentativo per mantenere la supremazia nucleare americana, e così iniziò la corsa globale alle armi di tale natura.
Tra gli ordigni di distruzione di massa si fecero ben presto spazio le bombe ad idrogeno, altrimenti dette termonucleari, che rappresentarono un notevole avanzamento rispetto alle prime bombe atomiche, proprio come queste lo furono rispetto agli esplosivi convenzionali. Lo sviluppo di queste nuove armi richiedeva costosi test, la maggior parte dei quali veniva condotta in luoghi remoti come l'Artico o le Isole del Sud Pacifico.
Dopo il test fallimentare denominato *Castle Bravo* da parte degli USA, che ebbe luogo nell'Atollo Bikini, che causò la morte di diversi civili e per il quale ancora oggi, dopo 60 anni, si registrano livelli di radioattività allarmanti, iniziò la protesta pubblica contro i test nucleari.
Anche Albert Einstein ne contemplò la capacità distruttiva, giungendo ad affermare, davanti agli spaventosi progressi di sviluppo delle *H-bomb*[2]:

> «[...] the annihilation of all life on Earth would be brought within the range of technical possibilities.»[3] [6]

[2] Bombe all'idrogeno

[3] A. Einstein, "Arms can bring no security", *Bulletin of the Atomic Scientists*, vol. 6, no. 3, 1950, p. 71. Educational Foundation for Nuclear Science, Inc.
«[...] l'annientamento di ogni forma di vita sulla Terra diventerà tecnicamente possibile.» trad.

La disperata richiesta di porre fine alla corsa agli armamenti nucleari fu presa effettivamente in considerazione dalle varie potenze mondiali: tra il 1958 e il 1959 furono avviati diversi negoziati, quali la *Conference of the Discontinuance of Nuclear Weapon Tests* tenutasi a Ginevra, con l'obiettivo di sottoscrivere un trattato internazionale sulla messa al bando degli esperimenti nucleari. Questi incontri sembrarono funzionare, tanto che il 1959 fu uno dei pochi anni in cui non furono fatte esplodere bombe nucleari. Il dilemma che affliggeva le diverse superpotenze non tardò però a palesarsi: gli Stati Uniti temevano che i sovietici avrebbero continuato i test di nascosto, superandoli tecnologicamente, e viceversa. Per affrontare queste preoccupazioni, fu convocata una conferenza di esperti per studiare la possibilità di rilevare violazioni di un possibile accordo sulla sospensione dei test nucleari.

Individuare i test atmosferici era abbastanza semplice: gli isotopi radioattivi da loro prodotti si disperdono nell'atmosfera e possono essere rilevati a migliaia di chilometri di distanza. I test subacquei producono suoni distintivi che viaggiano circa un chilometro sotto la superficie dell'oceano e possono essere captati dagli idrofoni. Al contrario, i test sotterranei sono molto più difficili da rilevare da lontano perché le loro radiazioni sono contenute, e, in aggiunta, i sovietici si rifiutarono di consentire visite di conformità in loco, che consideravano spionaggio. Per questi motivi, quando nel 1963 fu firmato un trattato sul divieto dei test, si trattava solo di un divieto parziale: furono vietati i test sott'acqua, nell'atmosfera e nello spazio, luoghi in cui la conformità poteva essere verificata, ma non furono aboliti i test sottoterra, per il semplice motivo che era quasi impossibile rilevarli.

Gli scienziati, tuttavia, non si diedero per vinti: dopo l'incontro di Ginevra, formarono diversi gruppi per risolvere il problema delle rilevazioni. L'idea principale che ne conseguì fu di utilizzare sismometri situati al di fuori dei paesi in cui venivano testate le armi nucleari per rilevare le deboli vibrazioni del terreno causate dalle esplosioni. La sfida che si trovarono ad affrontare fu come distinguere un test nucleare da un terremoto, e, se si fosse trattato di una bomba, quanto potente era e quanto in profondità era stata fatta esplodere. Sapevano che tutte queste informazioni erano contenute nel segnale del sismometro, ma non potevano essere derivate direttamente dal sismogramma. Avevano bisogno di sapere quante frequenze diverse vi erano presenti, il che significava che doveva essere effettuata una *Trasformata di Fourier*.

Nella delegazione americana al convegno di Ginevra, nel team di ricerca per la risoluzione di tale problema spiccarono le figure di Richard Garwin e di John Tukey: essi capirono ben presto che il vero ostacolo della rilevazione di esplosioni sotterranee non era la precisione dei sismometri, bensì l'enorme quantità di calcoli necessari per trasformare i segnali secondo Fourier. Infatti, al contrario di quanto accade nella teoria, in cui i segnali si possono pensare come delle onde

continue infinte, dalle quali con la trasformata di Fourier si ottiene una spettro di frequenza continuo infinito, nella realtà tali segnali sono finiti e costituiti da singoli campioni. Anche se il segnale di un sismometro appare fluido e continuo, non registra il movimento del suolo con infinita precisione. C'è una certa granulosità fondamentale nei dati, ovvero ciò che si sta studiando sono dati finiti discreti. Non si può quindi utilizzare la trasformata di Fourier idealizzata, ma lo strumento matematico adeguato è la *Trasformata Discreta di Fourier* (DFT).
L'ostacolo principale che si trovarono a dover oltrepassare è che per completare una DFT è necessario moltiplicare N punti dati per N onde di frequenza diverse: N^2 moltiplicazioni complesse. Alla velocità dei computer degli anni '60, ci sarebbero voluti più di tre anni per completare una singola trasformazione.
La svolta arrivò nel 1963, quando Tukey trovò un modo di calcolare la DFT con molti meno calcoli, ovvero in un tempo estremamente ridotto: questo procedimento prese il nome di *Fast Fourier Transform*. Per rendere tale algoritmo computabile, Garwin si rivolse a James Cooley, un ricercatore dell'*IBM* [4], con lo scopo di programmare un calcolatore per eseguire l'algoritmo FFT.
I risultati non tardarono ad arrivare: nel 1965, Cooley e Tukey pubblicarono l'algoritmo in un articolo chiamato "*An algorithm for the Machine Calculation of Complex Fourier Series*" [5] [5]. Da allora, l'utilizzo dell'algoritmo ottimizzato per il calcolo della FFT decollò immediatamente, anche se risultava ormai inutile a garantire un divieto totale dei test nucleari. A quel punto, il Regno Unito, la Francia e la Cina si erano uniti all'Unione Sovietica e agli Stati Uniti come potenze nucleari. Il trattato sulla messa al bando parziale dei test, lungi dal rallentare la corsa agli armamenti nucleari, l'aveva semplicemente resa una manovra clandestina.
Cooley e Tukey non furono però i primi a scoprire la FFT. Nel lontano 1805, il matematico Carl Friedrich Gauss stava studiando gli asteroidi appena scoperti di Pallade, Cerere e Giunone. Per determinare l'orbita di quest'ultimo, ideò un nuovo approccio all'analisi armonica e lo annotò nei suoi appunti, che decise di non pubblicare. Durante tali studi possiamo notare che Gauss aveva calcolato la trasformata discreta di Fourier prima che lo stesso Joseph Fourier la pubblicasse nel 1807. Inoltre, aveva sviluppato lo stesso algoritmo di FFT di Cooley e Tukey un secolo e mezzo prima.
Concludendo, in seguito alla pubblicazione dell'articolo di Cooley e Tukey, la ricerca e l'analisi degli algoritmi di FFT, e i conseguente utilizzi, crebbero in modo esponenziale, vedendoli appli-

[4] *International Business Machines Corporation*, azienda statunitense che opera nel settore informatico, tra le più antiche e grandi del mondo.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the Machine Calculation of Complex Fourier Series", *Mathematics of Computation*, vol. 19, no. 90, pp. 297-301. JSTOR

cati in una miriade di vari settori: compressione di audio ed immagini, risoluzione di equazioni differenziali, studio delle strutture cristalline, sviluppo di radar, sonar, wi-fi e 5G. Fondamentalmente, ogniqualvolta l'elaborazione di segnali e dati risulta essenziale, diversi algoritmi di FFT vengono coinvolti.

2.2 La Trasformata Discreta di Fourier

La *Trasformata Discreta di Fourier*, comunemente nota con l'acronimo DFT, risponde all'esigenza di implementare la Trasformata (e Antitrasformata) di Fourier al calcolatore, operando in determinati valori discreti assunti dalla funzione in particolari punti. A tale proposito, è importante notare che la conoscenza di N valori assunti da una funzione f in un intervallo $[0, T]$, in alcuni casi può essere sufficiente a determinare univocamente tale funzione, a patto di considerare un'opportuna distribuzione di punti nell'intervallo preso in considerazione. Sussiste il seguente:

Teorema 2.1 (di campionamento di Shannon) *Sia f una funzione la cui Trasformata di Fourier sia nulla al di fuori dell'intervallo $[-f_{N_y}; +f_{N_y}]$. Siano assegnati i punti $x_n = nh$ con:*

$$h \leq \frac{1}{2f_{N_y}}$$

Allora f può essere univocamente ricostruita dai suoi campioni:

$$f(x) = \sum_{n=-\infty}^{+\infty} f(x_n) \frac{\sin\left(\pi(x - x_n)/h\right)}{\pi(x - x_n)/h}$$

La quantità f_{N_y} è detta *frequenza di Nyquist*.
Il teorema stabilisce che quanto più rapidamente la funzione f oscilli nell'intervallo $[0, T]$, ovvero quanto sia più grande la sua frequenza, tanto più piccola deve essere la distanza dei punti considerati. Inoltre, per ricostruire l'andamento di una funzione che nel suo periodo T ha un unico ciclo $f_{N_y} = \frac{1}{T}$, occorre conoscere il valore assunto dalla funzione in almeno due punti con $h < \frac{T}{2}$. In caso contrario, nel ricostruire la funzione f, le frequenze maggiori della *frequenza di Nyquist* si sovrapporrebbero alle altre. Tale fenomeno, nell'analisi di Fourier, prende il nome di *aliasing*.

Si può dunque procedere esponendo le seguenti:

Definizione 2.1 (Trasformata Discreta di Fourier) *Dato un vettore di numeri complessi* $f = (f_0, \ldots, f_{N-1})$ *di lunghezza N, si definisce la Trasformata Discreta di Fourier (DFT) di f il vettore di numeri complessi* $F = (F_0, \ldots, F_{N-1})$ *ove:*

$$F_k = \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N} jk} \quad \textit{con } k = 0, \ldots, N-1 \tag{2.1}$$

Definizione 2.2 (Trasformata Discreta Inversa di Fourier) *Dato un vettore di numeri complessi* $F = (F_0, \ldots, F_{N-1})$ *di lunghezza N, si definisce la Trasformata Discreta Inversa di Fourier (IDFT) di F il vettore di numeri complessi* $f = (f_0, \ldots, f_{N-1})$ *ove:*

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} F_j \, e^{\frac{2\pi i}{N} jk} \quad \textit{con } k = 0, \ldots, N-1 \tag{2.2}$$

In letteratura, è possibile trovare la DFT e la IDFT definite in un modo leggermente diverso da quello adottato in questa trattazione. Gli unici requisiti di queste convenzioni sono che DFT e IDFT devono avere gli esponenziali di segno opposto e che il prodotto dei coefficienti davanti alle sommatorie deve risultare $\frac{1}{N}$. Un fattore di normalizzazione $\sqrt{\frac{1}{N}}$ sia per la DFT che per la IDFT rende le trasformate unitarie, cui conseguono alcuni vantaggi nella trattazione teorica, mentre sul piano pratico nelle operazioni numeriche è preferibile effettuare le normalizzazioni un'unica volta. In questo saggio si è scelto di utilizzare gli esponenziali negativi per definire la DFT e il fattore di normalizzazione $\frac{1}{N}$ compare nella definizione della IDFT.
Si noti inoltre che in (2.1) e (2.2) gli esponenziali $e^{\pm \frac{2\pi i}{N} jk}$ sono le *radici N-esime dell'unità*[6], che d'ora in avanti verranno indicate con W_N^{jk}.
Un risultato di ortogonalità importante delle *radici N-esime dell'unità* che tornerà utile in seguito è il seguente:

Lemma 2.1 *Le radici N-esime dell'unità sono tali che:*

$$\sum_{k=0}^{N-1} W_N^{jk} = \begin{cases} N & \textit{se } j = 0 \bmod N \\ 0 & \textit{se } j \neq 0 \bmod N \end{cases}$$

[6] Le *radici N-esime dell'unità* sono tutti i numeri complessi la cui potenza N-esima è pari a 1, ovvero tutte le radici del polinomio $x^N - 1$.

Dimostrazione: Dall'identità

$$1 - x^N = (1 - x)(1 + x + x^2 + \cdots + x^{N-1})$$

ponendo $x = W_N^r$, se ne deduce che, se $r \neq 0 \bmod N$ allora $1 - x \neq 0$, mentre $1 - x^N = 0$. Ne segue, essendo in un campo, che $1 + W_N^r + W_N^{2r} + \cdots W_N^{(N-1)r} = 0$. Altrimenti, se $r = 0 \bmod N$, la sommatoria $\sum_{k=0}^{N-1} W_N^{rj}$ diventa una somma di N unità. □

2.3 La DFT a partire dalla Trasformata di Fourier

Data una funzione f, si consideri la sua *Trasformata di Fourier* $\widehat{f}$ definita come in (1.4), calcolata in $\mathbb{R}$ e al cui esponente si aggiunga un fattore 2π per coerenza con le definizioni.
Per $T \in \mathbb{R}$, si ottiene:

$$\widehat{f}(\xi) = \int_{\mathbb{R}} f(x)e^{-2\pi i\xi x}\,dx = \sum_{k=-\infty}^{+\infty} \int_{kT}^{(k+1)T} f(x)e^{-2\pi i\xi x}\,dx =$$
$$= \sum_{k=-\infty}^{+\infty} \int_0^T f(x + kT)e^{-2\pi i\xi(x+kT)}\,dx = \int_0^T \sum_{k=-\infty}^{+\infty} f(x + kT)e^{-2\pi i\xi(x+kT)}\,dx$$

Introducendo la funzione periodica f_p di periodo T, definita come:

$$f_p(x) = \sum_{k=-\infty}^{+\infty} f(x + kT)$$

segue che:

$$\widehat{f}(\xi) = \int_0^T f_p(x)e^{-2\pi i\xi(x+kT)}\,dx \tag{2.3}$$

Si deduce che la *Trasformata di Fourier* di f coincide con la *Trasformata di Fourier* di f_p.
Si richiami ora la formula di quadratura[7] dei *trapezi*, con lo scopo di utilizzarla per discretizzare l'integrale (2.3) in $N + 1$ nodi, e, successivamente, sfruttando la periodicità dell'integrando, di

[7] Una formula di quadratura è un metodo utilizzato per approssimare l'integrale definito di una funzione in un intervallo chiuso e limitato: un classico problema di Analisi Numerica.

valutare la funzione $\widehat{f}(\xi)$ in $\xi_n = \frac{n}{T}$ con $n = 0, \ldots, N-1$:

$$\widehat{f}(\xi_n) = \int_0^T f_p(x) e^{-2\pi i \xi_n (x+kT)}\, dx \tag{2.4}$$

Definizione 2.3 (Formula dei Trapezi) *Sia f una funzione continua definita in un intervallo $[a; b]$.*
Siano $x_k = a + kh$ nodi distinti equispaziati appartenenti all'intervallo $[a; b]$ con $h = \frac{b-a}{N}$ e $k = 0, \ldots, N$.
Allora $I(f) = \int_a^b f(x)\, dx$ è approssimato dalla seguente formula di quadratura, detta dei trapezi (o del trapezio composta):

$$S(f) = h\left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{N-1}) + \frac{f(x_N)}{2}\right]$$

con errore di discretizzazione:

$$E(f) = I(f) - S(f) = -\frac{b-a}{12} h^2 f^{(2)}(\xi) \tag{2.5}$$

per un qualsiasi $\xi \in [a; b]$

Considerando quindi la funzione $\widehat{f}(\xi_n)$ definita in (2.4), e posto $h = \frac{T}{N}$, si ottiene la formula di quadratura in $N+1$ punti $x_k = kh$, con $k = 0, \ldots, N$:

$$S\left(f_p(x) e^{-2\pi i \cdot x \cdot \frac{n}{T}}\right) = h\left[\frac{f_p(0) e^{-2\pi i \cdot 0 \cdot \frac{T}{N} \cdot \frac{n}{T}}}{2} + f_p\left(\frac{T}{N}\right) e^{-2\pi i \cdot \frac{T}{N} \cdot \frac{n}{T}} + \ldots + \frac{f_p(T) e^{-2\pi i \cdot N \cdot \frac{T}{N} \cdot \frac{n}{T}}}{2}\right]$$

in cui si può notare che $f_p(0) e^{-2\pi i \cdot 0 \cdot \frac{T}{N} \cdot \frac{n}{T}}$ e $f_p(T) e^{-2\pi i \cdot N \cdot \frac{T}{N} \cdot \frac{n}{T}}$, ovvero i numeratori del primo e dell'ultimo termine della sommatoria, coincidono. Infatti, essendo f_p periodica di periodo T, si ha che $f_p(0) = f_p(T)$, ed inoltre $e^{-2\pi i \cdot 0} = e^{-2\pi i \cdot n}$.
Da quest'ultima considerazione si può dedurre quindi che:

$$\frac{f_p(0) e^{-2\pi i \cdot 0 \cdot \frac{T}{N} \cdot \frac{n}{T}}}{2} + \frac{f_p(T) e^{-2\pi i \cdot N \cdot \frac{T}{N} \cdot \frac{n}{T}}}{2} = f_p(0) e^{-2\pi i \cdot 0 \cdot \frac{T}{N} \cdot \frac{n}{T}}$$

Allora la formula di quadratura può essere riscritta nel seguente modo:

$$F(\xi_n) = h \sum_{k=0}^{N-1} f_p(kh)\, e^{-2\pi i k h \frac{n}{T}} = h \sum_{k=0}^{N-1} f_p(kh)\, e^{-\frac{2\pi}{N} i k n}$$

Considerati i vettori $F = F(\xi_n)_{n=0,\dots,N-1}$ e $f_p = f_p(kh)_{k=0,\dots,N-1}$ segue che la *Trasformata Discreta di Fourier F*, costruita a partire dal vettore f_p, fornisce un'approssimazione dei valori assunti dalla *Trasformata di Fourier* della funzione f_p in un insieme di punti equidistanziati. In particolare poi, se f è nulla al di fuori dell'intervallo $[0; T]$, allora $f_p(v) = f(v)$ per ogni $v \in [0; T]$, e quindi la DFT di f_p fornisce un'approssimazione della *Trasformata di Fourier* di f.
In altre parole, la FT e la DFT coincidono nei punti ξ_n a meno di un errore. L'errore dipende dell'errore di discretizzazione, dovuto alla formula di quadratura utilizzata, e dall'errore di troncamento introdotto dall'aver assunto $f_p = f$. Per quanto riguarda l'errore di discretizzazione, tale quantità tende a zero al tendere a zero di h come descritto in (2.5). L'ordine di convergenza della formula di quadratura trapezoidale dipende dalla regolarità della funzione f. In particolare, a partire dalla stima dell'errore delle formule di quadratura di Newton-Cotes, se $f \in C^{2M+2}$, utilizzando lo sviluppo dell'errore fornito dalla formula di Eulero-McLaurin[8], l'errore di discretizzazione ha ordine $2M + 2$.

[8] Se l'integranda $f \in C^{2M+2}([a; b])$, allora:
$I(f) = S(f) - \sum_{k=1}^{M} \frac{B_{2k}}{(2k)!} h_N^{2k} \left(f^{(2k-1)}(b) - f^{(2k-1)}(a)\right) - \frac{B_{2M+2}}{(2M+2)!} h_N^{(2M+2)} (b-a) f^{(2M+2)}(\xi)$, dove B_k sono i numeri di Bernoulli

2.4 Matrice di Fourier

La *Matrice di Fourier* è una matrice complessa di Vandermonde[9] che può essere utilizzata per esprimere in forma matriciale la DFT.

Definizione 2.4 (Matrice di Fourier) *La matrice* $\Omega_N = (W_N^{jk})_{j,k=0,\dots,N-1}$*, le cui entrate sono le radici N-esime dell'unità, che si presenta come segue:*

$$\Omega_N = \begin{pmatrix} W_N^{0\cdot 0} & W_N^{0\cdot 1} & W_N^{0\cdot 2} & \cdots & W_N^{0\cdot(N-1)} \\ W_N^{1\cdot 0} & W_N^{1\cdot 1} & W_N^{1\cdot 2} & \cdots & W_N^{1\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^{(N-1)\cdot 0} & W_N^{(N-1)\cdot 1} & W_N^{(N-1)\cdot 2} & \cdots & W_N^{(N-1)\cdot(N-1)} \end{pmatrix}$$

è detta Matrice di Fourier.

Tale matrice gode di diverse proprietà interessanti ai fine di questa trattazione. Citiamo a tal proposito il seguente:

Teorema 2.2 *La matrice* Ω_N *è tale per cui:*

1. $\Omega_N = \Omega_N^T$*;*
2. $\Omega_N^H \Omega_N = NI$*, dove* Ω_N^H *è la matrice coniugata trasposta di* Ω_N *e* I *la matrice identità;*
3. $\Omega_N^2 = N\Pi_N$ *dove* Π_n *è la matrice di permutazione corrispondente alla permutazione* $\pi_0 = 0, \pi_h = N - h$ *con* $h = 0, \dots, N-1$

Dimostrazione:

1. La matrice è banalmente simmetrica, essendo $W_N^{jk} = W_N^{kj}$.
2. Per mostrare l'ortogonalità delle colonne di Ω_N, devo verificare che il prodotto tra l'i-esima riga di Ω_N^H e la j-esima riga colonna di Ω_N vale zero se $i \neq j$ e vale N se $i = j$, ovvero:

$$\sum_{k=0}^{N-1} W_N^{-ik}\, W_N^{jk} = \begin{cases} N & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases}$$

   Segue la tesi per il Lemma (2.1).

[9] In algebra lineare, una Matrice di Vandermonde è un matrice le cui righe (o colonne), hanno elementi che, partendo da 1, sono in progressione geometrica, ovvero: $a_{ij} = \alpha_i^{j-1}$

3. L'elemento di posto (i,j) di Ω_N^2 è dato da:

$$\sum_{k=0}^{N-1} W_N^{ik} W_N^{jk} = \sum_{k=0}^{N-1} W_N^{k(i+j)}$$

e per il Lemma (2.1), vale zero se $i+j \neq 0 \bmod N$ e vale N se $i+j = 0 \bmod N$. □

Il Teorema sopracitato permette di scrivere l'inversa di Ω_N in forme computazionalmente convenienti. Ne consegue il seguente:

Corollario 2.1 *Sia Ω_N Matrice di Fourier. Vale allora:*

$$\Omega_N^{-1} = \frac{1}{N}\Omega_N^H = \frac{1}{N}\Omega_N \Pi_N = \frac{1}{N}\Pi_N \Omega_N^H$$

Mantenendo le definizioni come in (2.1) e (2.2), possiamo quindi definire la DFT come la trasformazione lineare che associa il vettore F ad f, e la IDFT la sua inversa, nel seguente modo:

$$F = \mathrm{DFT}(f) = \Omega_N^H f \tag{2.6}$$

$$f = \mathrm{IDFT}(F) = \frac{1}{N}\Omega_N F \tag{2.7}$$

2.5 Complessità computazionale della DFT

La *Trasformata Discreta di Fourier* è direttamente implementabile su un calcolatore, in quanto l'algoritmo richiede un numero finito di operazioni, al contrario della *Trasformata di Fourier*, che consiste nella risoluzione di un integrale. Tuttavia, il calcolo della DFT non viene mai implementato secondo le definizioni classiche, ma si preferisce utilizzare algoritmi ottimizzati che richiedono degli sforzi e, soprattutto, dei tempi computazionali molto minori. Si noti, infatti, che per calcolare la DFT, supponendo precalcolati, ossia già disponibili in memoria, i termini esponenziali presenti nella formula, si richiedono N moltiplicazioni complesse e $N-1$ addizioni complesse. Se si considera che un'addizione complessa richieda in realtà due addizioni reali, e che una moltiplicazione complessa richieda quattro moltiplicazioni reali e due addizioni, per ognuno dei k nodi sono richieste complessivamente $8N-2$ operazioni reali. Poichè k assume

valori tra 0 e $N-1$, il numero complessivo di operazioni da compiere per il calcolo della DFT di una sequenza periodica di periodo N è, al crescere di N, di circa $8N^2$. Dunque, a parte la valutazione degli esponenziali complessi, la complessità computazionale del calcolo di una DFT è equivalente a quella di un prodotto matrice- vettore. Semplificando, il costo computazionale di una DFT è dell'ordine di $\mathcal{O}(N^2)$, cioè richiede un numero di operazioni proporzionali a N^2, e per un valore di N elevato ben si comprende la necessità di ricorrere a strumenti di calcolo meno onerosi.
L'idea di Cooley e Tukey fu di semplificare opportunamente i calcoli decomponendo il problema, ovvero il calcolo di una DFT di lunghezza N, in sottoproblemi dello stesso tipo ma di dimensione non solo più piccola, ma anche tale da consentire di eliminare buona parte delle operazioni inutili e ridondanti: nacquero così i primi algoritmi FFT.

2.6 La Trasformata Veloce di Fourier

La *Trasformata Veloce di Fourier* (spesso indicata come FFT, ovvero "*Fast Fourier Transform*") è un algoritmo ottimizzato per calcolare la DFT. Vale a dire che tale algoritmo determina esattamente la DFT corrispondente, al netto di errori dovuti ai vari troncamenti in virgola mobile, ma con un numero di calcoli ridotto in senso aritmetico. Il grosso vantaggio della FFT si esprime quindi come la riduzione della complessità computazionale nell'eseguire la DFT, che risulta, infatti, essere proporzionale a $\mathcal{O}(N\log_2 N)$. Risulta immediato notare il vantaggio in termini di velocità di calcolo; si osservi, a titolo esemplificativo, che per $N = 1024$, si ha che $N^2 = 1048576$, mentre $N\log_2 N = 10240$. In altre parole, per N dell'ordine delle migliaia, il numero di operazioni necessario viene ridotto di un ordine 100.
La trattazione procede quindi esponendo diversi algoritmi FFT, evidenziandone i rispettivi limiti e punti di forza, mostrando un'implementazione in Matlab.
Concludendo questa prima fase introduttiva, è doveroso sottolineare che qualsiasi algoritmo FFT può essere facilemente invertito per ottimizzare il calcolo della IDFT, essendo quest'ultima molto simile come forma alla DFT (differisce infatti, come precedentemente evidenziato, solo dal segno degli esponenti e dal fattore di normalizzazione).

2.7 Algoritmo di Cooley-Tukey radix-2

L'algoritmo FFT più diffuso è l'algoritmo di Cooley-Tukey. Questo metodo si basa sul principio di *divide et impera*, spezzando ricorsivamente una DFT di qualsiasi dimensione N, con $N = N_1 N_2$, in DFT di dimensioni più piccole N_1 ed N_2, più $\mathcal{O}(N)$ moltiplicazioni per l'unità immaginaria, dette *fattori twiddle*.
La variante più conosciuta dell'algoritmo di Cooley-Tukey consiste nel dividere l'operazione corrente in due parti di dimensione $\frac{N}{2}$ ad ogni passo, ed è quindi ottimizzata solo per dimensioni che siano potenze di due, ma una sua formulazione più generale può essere utilizzata con qualsiasi fattorizzazione, com'era noto sia a Gauss che a Cooley e Tukey. Questi casi sono chiamati rispettivamente *radix-2* e *radice mista*, ma esistono anche altri approcci con nomi specifici. Anche se l'idea di base è ricorsiva, gran parte delle implementazioni tradizionali consente una riorganizzazione dell'algoritmo per evitare la ricorsione esplicita. Inoltre, poiché l'algoritmo di Cooley-Tukey spezza la DFT corrente in DFT più piccole, esso può essere arbitrariamente combinato con qualsiasi altro algoritmo per la DFT, a seconda della convenienza e delle particolari esigenze.
Di seguito proponiamo l'algoritmo di fattorizzazione di Cooley-Tukey *radix-2*.
Sia $N = 2^s$. Considerando $F_k = \text{DFT}(f_j)$, come definito in (2.1), dato che N è un intero pari, si può approcciare il calcolo di F decomponendo il vettore f in una parte contenente le componenti di indice pari, ovvero il vettore formato da $\{f_{2j}\}_{j=0,\dots,\frac{N}{2}}$ e in una parte contenente le componenti di indice dispari, ovvero il vettore formato da $\{f_{2j+1}\}_{j=0,\dots,\frac{N}{2}-1}$.
Attuando tale ripartizione, dall'equazione:

$$F_k = \sum_{j=0}^{N/2-1} f_{2j}\, W_N^{-2jk} + \sum_{j=0}^{N/2-1} f_{2j+1}\, W_N^{-(2j+1)k}$$

si ottiene:

$$F_k = \sum_{j=0}^{N/2-1} f_{2j}\, W_{N/2}^{-jk} + W_N^{-k} \sum_{j=0}^{N/2-1} f_{2j+1}\, W_{N/2}^{-jk} \quad \textit{con } k = 0, \dots, N-1 \tag{2.8}$$

Chiamate

$$F_k^E = \sum_{j=0}^{N/2-1} f_{2j}\, W_{N/2}^{-jk}$$

$$F_k^O = \sum_{j=0}^{N/2-1} f_{2j+1}\, W_{N/2}^{-jk}$$

le due DFT di dimensione $\frac{N}{2}$, ottenute a partire dai vettori con componenti pari e dispari, la (2.8) diventa:

$$F_k = F_k^E + W_N^{-k} F_k^O \tag{2.9}$$

e sfruttando la periodicità dell'esponenziale complesso $W_{N/2}$ le altre $\frac{N}{2}$ componenti si ricavano a partire dalle prime.

Per semplificare un pò il conto, si può considerare una moltiplicazione complessa seguita da un'addizione complessa come una unica operazione complessa: in questo modo si passa da N^2 operazioni svolte nel caso del calcolo diretto, a $N + 2\left(\frac{N}{2}\right)^2$ operazioni con una sola suddivisione. Ciò significa che se si volesse calcolare la trasformata su 8 campioni, si passerebbe da 64 operazioni complesse a 40.

Ora, essendo N una potenza di 2, si può continuare a dividere F^E e F^O a metà, seguendo lo stesso criterio adottato in precedenza. In generale è possibile iterare questo procedimento in modo da suddividere ulteriormente il calcolo in DFT di dimensione $\frac{N}{8}$, $\frac{N}{16}$, ecc. Si può effettuare una suddivisione per un totale di $s = \log_2(N)$ stadi successivi fino ad ottenere lo stadio base in cui la DFT è calcolata su solo due punti. In termini di operazioni complesse svolte, al più si avrà un numero di successivi dimezzamenti pari a s, che sono proprio il numero di passi necessari per eseguire la FFT. Dato che ad ogni passo al massimo saranno necessarie $\frac{N}{2}$ moltiplicazioni, allora la FFT richiederà $\frac{N}{2}\log_2(N)$ operazioni, contro le N^2 richieste dalla DFT. Perciò questo tipo di algoritmo ha una complessità computazionale logaritmica, anzichè quadratica come il calcolo diretto. Tutto ciò, come già affermato, comporta un enorme risparmio in termini di complessità computazionale.

Si riporta di seguito un esempio compilativo dell'applicazione di tale algoritmo.

Esempio 2.1 Sia $f = \{f_j\}_{j=0,\dots,N-1}$ un vettore di lunghezza $N = 16 = 2^4$, e si consideri il calcolo della DFT di f, ovvero si determini $F = \{F_k\}_{k=0,\dots,N-1}$ tale che:

$$F_k = \sum_{j=0}^{15} f_j\, W_{16}^{-jk} \quad k = 0, \dots, 15 \tag{2.10}$$

Essendo N una potenza di 2, si procede applicando l'algoritmo di Cooley-Tukey *radix-2*.
Per k fissato, con $k = 0, \ldots, 15$, dalla (2.10) si ottiene:

$$\begin{aligned} F_k &= \sum_{j=0}^{7} f_{2j}\, W_{16}^{-2jk} + \sum_{j=0}^{7} f_{2j+1}\, W_{16}^{-(2j+1)k} \\ &= \sum_{j=0}^{7} f_{2j}\, W_8^{-jk} + W_{16}^{-k} \sum_{j=0}^{7} f_{2j+1}\, W_8^{-jk} \end{aligned}$$

Chiamate:

$$\begin{aligned} F_k^E &= \sum_{j=0}^{7} f_{2j}\, W_8^{-jk} \\ F_k^O &= \sum_{j=0}^{7} f_{2j+1}\, W_8^{-jk} \end{aligned}$$

l'algoritmo, al primo passo, porta alla decomposizione del vettore F come:

$$F_k = F_k^E + W_{16}^{-k}\, F_k^O \quad k = 0, \ldots, 7$$

Ora si può procedere ricorsivamente considerando i due vettori $\{F_k^E\}_{k=0,\ldots,7}$ e $\{F_k^O\}_{k=0,\ldots,7}$. Entrambe le DFT di lunghezza 8 si possono decomporre in 2 DFT di lunghezza 4.
Si ha:

$$\begin{aligned} F_k^E = \sum_{j=0}^{7} f_{2j}\, W_8^{-jk} &= \sum_{j=0}^{3} f_{4j}\, W_8^{-2jk} + \sum_{j=0}^{3} f_{4j+1}\, W_8^{-(2j+1)k} = \\ &= \sum_{j=0}^{3} f_{4j}\, W_4^{-jk} + W_8^{-k} \sum_{j=0}^{3} f_{4j+1}\, W_4^{-jk} \end{aligned}$$

Chiamate:

$$\begin{aligned} F_k^{EE} &= \sum_{j=0}^{3} f_{4j}\, W_4^{-jk} \\ F_k^{EO} &= \sum_{j=0}^{3} f_{4j+1}\, W_4^{-jk} \end{aligned}$$

le due DFT di lunghezza 4 costruite utilizzando i vettori $\{f_{4j}\}_{j=0,\dots,3}$ e $\{f_{4j+1}\}_{j=0,\dots,3}$, il secondo passo dell'algoritmo porta alla decomposizione di $\{F_k^E\}_{k=0,\dots,7}$ come:

$$F_k^E = F_k^{EE} + W_8^{-k} F_k^{EO} \quad k = 0, \dots, 3$$

e ad una decomposizione di $\{F_k^O\}_{k=0,\dots,7}$ del tutto analoga.
Ciò ha semplificato il problema iniziale al calcolo di 4 DFT di lunghezza 4.
Reiterando il procedimento, ciascuna delle DFT sopracitate si può riscrivere come somma di 2 DFT di lunghezza 2. Si ottiene:

$$\begin{aligned} F_k^{EE} = \sum_{j=0}^{3} f_{4j}\, W_4^{-jk} &= \sum_{j=0}^{1} f_{8j}\, W_4^{-2jk} + \sum_{j=0}^{1} f_{8j+1}\, W_4^{-(2j+1)k} = \\ &= \sum_{j=0}^{1} f_{8j}\, W_2^{-jk} + W_4^{-k} \sum_{j=0}^{1} f_{8j+1}\, W_2^{-jk} \end{aligned}$$

Siano:

$$F_k^{EEE} = \sum_{j=0}^{1} f_{8j}\, W_2^{-jk}$$
$$F_k^{EEO} = \sum_{j=0}^{1} f_{8j+1}\, W_2^{-jk}$$

le 2 DFT di lunghezza 2 applicando l'algoritmo a $\{F_k^{EE}\}_{k=0,\dots,3}$.
Concludendo, tale esempio ha esplicato come, per $N = 16 = 2^4$, l'algoritmo di Cooley-Tukey *radix-2* abbia portato in 3 step il problema iniziale $F_k = \text{DFT}(f_j)$ alla decomposizione:

$$\begin{aligned} F_k &= F_k^E + F_k^O = \\ &= (F_k^{EE} + F_k^{EO}) + (F_k^{OE} + F_k^{OO}) = \\ &= (F_k^{EEE} + F_k^{EEO}) + (F_k^{EOE} + F_k^{EOO}) + (F_k^{OEE} + F_k^{OEO}) + (F_k^{OOE} + F_k^{OOO}) \end{aligned}$$

ovvero ridotto alla somma di 8 DFT di lunghezza 2.
Si noti la natura ricorsiva di tale metodo e la grande semplificazione a livello di calcolo e complessità computazionale ottenuta. Inoltre, è da evidenziare che l'algoritmo di Cooley-Tukey *radix-2*, da un punto di vista implementativo, prevede una prima fase di riordinamento, seguita da una seconda fase di calcolo sul vettore ordinato. Ne consegue che il vettore risultante

sarà ordinato secondo l'ordine naturale.

∞

2.7.1 FFT in Matlab

Gli algoritmi FFT sono implementati in Matlab nelle funzioni fft e ifft, utilizzando le notazioni adottate in questa trattazione nel definire DFT e IDFT come in (2.1) e (2.2).
Di seguito si riporta un'implementazione in Matlab dell'algoritmo FFT di Cooley-Tukey *radix-2*.

```
function F = FFT_cooley_tukey_radix2(f)             %sia f il vettore di partenza
    N = length(f);                                  %sia N la sua lunghezza
    if N == 1                                       %caso base, se N=1
        F = f;
        return;
    end

    F_e = FFT_cooley_tukey_radix2(f(1:2:end));      %passo 'divide'
    F_o = FFT_cooley_tukey_radix2(f(2:2:end));

    for k = 1:N/2
        twiddle = exp(-2*pi*1i*(k-1)/N);            %passo 'impera'
        F(k) = F_e(k) + twiddle * F_o(k);
        F(k+N/2) = F_e(k) - twiddle * F_o(k);
    end
end
```

2.8 Algoritmo di Gentleman-Sande radix-2

L'algoritmo FFT di Gentleman-Sande *radix-2*, calcola in modo ottimizzato la DFT di un vettore di dimensione N, con N potenza di 2, decomponendo ripetutamente il vettore DFT iniziale, per poi separare le componenti di indici pari e dispari dalle varie DFT ottenute, riconducendosi così a dover risolvere $\frac{N}{2}$ DFT di lunghezza 2.
A differenza del metodo di Cooley-Tukey, tale strategia antepone la fase di calcolo a quella di riordinamento. Infatti tale algoritmo prevede di calcolare le DFT di un certo passo utilizzando il vettore costruito al passo precedente, fino ad ottenere $\frac{N}{2}$ DFT di lunghezza 2. Segue poi una necessaria fase di riordinamento, dato che il vettore risultante si presenterà nell'ordine indotto dall'*ordinamento del bit inverso*.

Definizione 2.5 (Bit Reversal) *L'ordinamento del bit inverso (bit reversal) è una permutazione di N interi indicizzati da 0 a $N-1$ in rappresentazione binaria, che mette in relazione ogni elemento con l'elemento ottenuto invertendo l'ordine delle cifre binarie.*
Un vettore è ordinato secondo l'ordinamento del bit inverso se tale risulta il sottoinsieme dei numeri naturali costituito dai valori assunti dall'indice della generica componente del vettore.

Di seguito si espone l'algoritmo FFT di Gentleman-Sande *radix-2*.
Sia $N = 2^s$. Considerando F_k =DFT(f_k), come definito in (2.1), decomponiamo la DFT iniziale come segue:

$$F_k = \sum_{j=0}^{N-1} f_j \, W_N^{-jk} = \sum_{j=0}^{N/2-1} f_j \, W_N^{-jk} + \sum_{j=0}^{N/2-1} f_{j+N/2} \, W_N^{-(j+N/2)k}$$

da cui, utilizzando il fatto che $W_N^{-N/2} = -1$ e $W_N^{-jk} = W_{N/2}^{-jk/2}$, si ottiene:

$$\begin{aligned} F_k &= \sum_{j=0}^{N/2-1} \left[f_j \, W_N^{-jk} + f_{j+N/2} \, W_N^{-(j+N/2)k} \right] \\ &= \sum_{j=0}^{N/2-1} \left[f_j + (-1)^k f_{j+N/2} \right] W_{N/2}^{-jk/2} \end{aligned}$$

Si procede quindi dividendo gli indici k tra pari e dispari nel seguente modo:

$$F_{2h} = \sum_{j=0}^{N/2-1} \left[f_j + f_{j+N/2} \right] W_{N/2}^{-jh} \quad h = 0, \ldots, N/2-1$$

$$F_{2h+1} = \sum_{j=0}^{N/2-1} \left[\left(f_j - f_{j+N/2} \right) W_N^{-j} \right] W_{N/2}^{-jh} \quad h = 0, \ldots, N/2-1$$

Introdotti i vettori $\{y_j\}_{j=0,\ldots,N/2-1}$ e $\{z_j\}_{j=0,\ldots,N/2-1}$ di componenti:

$$y_j = f_j + f_{j+N/2}$$
$$z_j = (f_j - f_{j+N/2}) W_N^{-j}$$

i due vettori di lunghezza $\frac{N}{2}$ $\{F_{2h}\}_{h=0,\ldots,N/2}$ e $\{F_{2h+1}\}_{h=0,\ldots,N/2}$ possono essere considerati come DFT costruite rispettivamente su $\{y_j\}_{j=0,\ldots,N/2-1}$ e $\{z_j\}_{j=0,\ldots,N/2-1}$, e quindi:

$$F = \mathrm{DFT}(f) = (\{F_{2h}\}, \{F_{2h+1}\})_{h=0,\ldots,N/2} = (\mathrm{DFT}(y), \mathrm{DFT}(z))$$

Al primo passo, l'algoritmo di Gentleman-Sande determina i due vettori y e z, separando in ciascuna DFT le componenti di indice pari da quelle di indice dispari, ottenendo 4 DFT di lunghezza $\frac{N}{4}$.
A partire dal vettore y si ottiene:

$$\mathrm{DFT}(y) = \sum_{j=0}^{N/2-1} \left[f_j + f_{j+N/2} \right] W_{N/2}^{-jh} = \sum_{j=0}^{N/2-1} y_j W_{N/2}^{-jh} =$$
$$= \sum_{j=0}^{N/4-1} \left[y_j + y_{j+N/4} \right] W_{N/4}^{-jh} + \sum_{j=0}^{N/4-1} \left[(y_j - y_{j+N/4}) W_{N/2^{-j}} \right] W_{N/4}^{-jh}$$

Considerando ora le componenti con indice pari e dispari, definite, per $\hat{h} = 0, \ldots, N/4-1$ nel seguente modo:

$$\mathrm{DFT}(y)_{2\hat{h}} = \sum_{j=0}^{N/4-1} \left[y_j + y_{j+N/4} \right] W_{N/4}^{-j\hat{h}}$$

$$\mathrm{DFT}(y)_{2\hat{h}+1} = \sum_{j=0}^{N/4-1} \left[(y_j - y_{j+N/4}) W_{N/2^{-j}} \right] W_{N/4}^{-j\hat{h}}$$

e introdotti i vettori di lunghezza $\frac{N}{4}$ $\{y'\}_{j=0,\ldots,N/4-1}$ e $\{y''\}_{j=0,\ldots,N/4-1}$ di componenti:

$$y_j^{'} = (y_j + y_{j+N/4})$$
$$y_j^{''} = (y_j - y_{j+N/4})\, W_{N/2}^{-j}$$

segue che:

$$\mathrm{DFT}(y) = (\mathrm{DFT}(y^{'}), \mathrm{DFT}(y^{''}))$$

Analogamente, per il vettore (z), introducendo i vettori $\{z^{'}\}_{j=0,\ldots,N/4-1}$ e $\{z^{''}\}_{j=0,\ldots,N/4-1}$ di componenti:

$$z_j^{'} = (z_j + z_{j+N/4})$$
$$z_j^{''} = (z_j - z_{j+N/4})\, W_{N/2}^{-j}$$

segue che:

$$\mathrm{DFT}(z) = (\mathrm{DFT}(z^{'}), \mathrm{DFT}(z^{''}))$$

Seguendo questi passi, il calcolo della DFT di lunghezza N iniziale viene ricondotto a quello di 4 DFT di lunghezza $\frac{N}{4}$:

$$F = (\mathrm{DFT}(y^{'}),\mathrm{DFT}(y^{''}),\mathrm{DFT}(z^{'}),\mathrm{DFT}(z^{''}))$$

Iterando questo procedimento, dopo $s = \log_2(N)$ passi, l'algoritmo di Gentleman-Sande restituisce $\frac{N}{2}$ vettori di lunghezza 2, i quali costituiscono una coppia di componenti della $\mathrm{DFT}(f)$. Per concludere l'algoritmo, è necessaria la fase finale di riordinamento delle varie componenti del vettore F secondo l'ordinamento *bit reversal* come da Definizione (2.5).

2.9 Algoritmo di Good-Thomas (PFA)

Il *Prime Factor Algorithm* (PFA), conosciuto anche come l'algoritmo di Good-Thomas, è un algoritmo di FFT che calcola la DFT di dimensione N nel caso in cui $N = N_1N_2$ con N_1 e N_2 coprimi[10].
Considerato il problema F_k =DFT(f_j), con $j, k = 0, \ldots, N-1$, tale algoritmo inizia riordinando gli indici di input j con un metodo di ordinamento conosciuto come *Good's mapping* e gli indici di output k con il *Teorema Cinese del Resto (CRT)*. Il vantaggio che questo algoritmo presenta rispetto agli algoritmi *mixed-radix* è che non richiede moltiplicazioni extra di radici dell'unità (i *twiddle factors*). D'altra parte, però, ha lo svantaggio di poter essere applicato solo quando N si può fattorizzare in fattori primi tra loro. In ogni caso, ogni passo dell'algoritmo può essere calcolato utilizzando il PFA ricorsivamente o applicando altri algoritmi FFT, come ad esempio quelli di Cooley-Tukey.
Iniziamo la trattazione di tale metodo citando il:

Teorema 2.3 (Cinese del Resto) *Siano $n_1, \ldots, n_k$ interi positivi a due a due coprimi e siano $a_1, \ldots, a_k$ interi qualsiasi. Allora il sistema di congruenze:*

$$\begin{aligned} x &\equiv a_1 \mod n_1 \\ &\vdots \\ x &\equiv a_k \mod n_k \end{aligned}$$

ha una soluzione x unicamente determinata modulo $N = n_1n_2 \cdots n_k$.[11]

Dimostrazione: Per ogni $i = 1, \ldots, k$, siano definiti $m_i = \dfrac{N}{n_i}$. Ora siano $y_i \equiv m_i^{-1} \mod n_i$. Dato che $n_1, \ldots, n_k$ sono a due a due relativamente coprimi, ogni y_i esiste. Definita x come:

$$x = \sum_{i=1}^{k} a_i m_i y_i$$

è una soluzione per il sistema di congruenze. Infatti, per ogni indice i, riducendo x modulo n_i, e ricordando che $m_iy_i \equiv 1 \bmod n_i$, si ottiene:

$$x \equiv a_i m_i y_i \bmod n_i \equiv a_i \bmod n_i$$

[10] Due numeri sono coprimi, o relativamente primi, o primi tra loro, se il loro Massimo Comune Divisore è 1.
[11] Utilizzando il Teorema Cinese del Resto, si può costruire un isomorfismo tra $\mathbb{Z}_{mn}$ e $\mathbb{Z}_m$x$\mathbb{Z}_n$. Per verificare l'implicazione opposta, si può considerare un elemento di $\mathbb{Z}_{mn}$ modulo m e modulo n per trovare il sistema di congruenze.

Per provare l'unicità della soluzione modulo N, consideriamo due soluzioni x_1 e x_2 del sistema di congruenze. Allora $x_1 \equiv x_2 \bmod n_i$, da cui deriva che n_i divide $x_1 - x_2$ per ogni i. Ora, essendo n_i ed n_j coprimi per ogni $i \neq j$, si ha che $N = n_1 n_2 \cdots n_k$ divide $x_1 - x_2$. Si è così dimostrato che $x_1 \equiv x_2 \bmod N$, ovvero che la soluzione x è unica modulo N. □

Per inizializzare il PFA, considerando $j_1 = 0, \ldots, N_1 - 1$ e $j_2 = 0, \ldots, N_1 - 1$, re-indicizziamo la sequenza di input di lunghezza $N = N_1 N_2$ nel seguente modo:

$$j = j_1 N_2 + j_2 N_1 \mod N \tag{2.11}$$

Questa corrispondenza è nota come *Good's mapping*, pubblicata da Irving John Good nell'articolo "*The interaction algorithm and pratical Fourier analysis*"[12] [10]
Usando invece il *Teorema Cinese del Resto*, considerando $k_1 = 0, \ldots, N_1 - 1$ e $k_2 = 0, \ldots, N_2 - 1$, la sequenza in output è re-indicizzata come segue:

$$k = k_1 N_2^{-1} N_2 + k_2 N_1^{-1} N_1 \mod N \tag{2.12}$$

dove $N_1^{-1} N_1 \equiv 1 \mod N_2$ e $N_2^{-1} N_2 \equiv 1 \mod N_1$. [13]
Ora, sostituendo le uguaglianze (2.11) e (2.12) in (2.1), si ottiene:

$$\begin{aligned} F_{k_1 N_2^{-1} N_2 + k_2 N_1^{-1} N_1} &= \sum_{j_1 N_2 + j_2 N_1}^{N-1} f_{j_1 N_2 + j_2 N_1} \, W_N^{(j_1 N_2 + j_2 N_1)(k_1 N_2^{-1} + k_2 N_1^{-1} N_1)} \\ &= \sum_{j_1 N_2 + j_2 N_1}^{N-1} f_{j_1 N_2 + j_2 N_1} \, W_N^{k_1 j_1 N_2 + k_2 j_2 N_1} \\ &= \sum_{k_1=0}^{N_1 - 1} \left(\sum_{k_2=0}^{N_2 - 1} f_{j_1 N_2 + j_2 N_1} \, W_{N_2}^{k_2 j_2} \right) W_{N_1}^{k_1 j_1} \end{aligned}$$

Si noti che sia $\{F\}_{k=0,\ldots,N-1}$ che $\{f\}_{j=0,\ldots,N-1}$ sono sequenze N-periodiche: si può quindi agire sugli indici modulo N. Inoltre, dato che W_N è una *radice N-esima dell'unità*, gli esponenziali con potenza che include il termine $N_1 N_2$ si semplificano a 1.

[12] I. J. Good, "The Interaction Algorithm and Practical Fourier Analysis", *Journal of the Royal Statistical Society*, vol. 20, no. 2, 1958, pp. 361-372. Oxford University Press

[13] N_1^{-1} e N_2^{-1} esistono poichè N_1 e N_2 sono relativamente primi.

Rinominando $j_1 N_2 + j_2 N_1$ con la coppia (j_1, j_2) e $k_1 N_2^{-1} N_2 + k_2 N_1^{-1} N_1$ con la coppia (k_1, k_2), si può riscrivere il risultato dell'algoritmo PFA più elegantemente come:

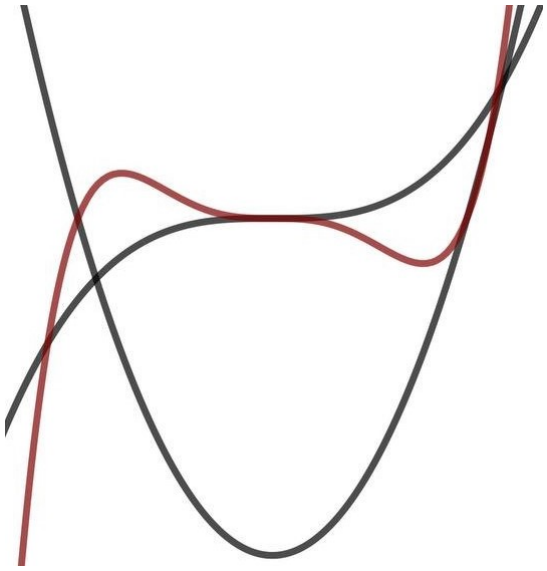
$$F_{k_1,k_2} = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} f_{j_1,j_2} \, W_{N_1}^{k_1 j_1} \, W_{N_2}^{k_2 j_2}$$

Concludendo, il *Prime Factor Algorithm* ha semplificato il calcolo di una DFT di lunghezza $N = N_1 N_2$ con due DFT 2-dimensionali di lunghezza N_1 e N_2. Facendo ciò si è ottenuta un efficienza pari all'algoritmo di Cooley-Tukey *radix-2*, riducendo il numero di operazioni a $\mathcal{O}(N \log_2 N)$ passi.

3

Moltiplicazione Rapida di Polinomi

«[...] it may well become an indispensable tool in the computational mathematician's arsenal.»[1] [11]



[1] D. Harvey, "We've found a quicker way to multiply really big numbers", *The conversation*, 2019.
«[...] potrebbe diventare uno strumento indispensabile nell'arsenale di un matematico computazionale.» trad.

3.1 Applicazioni della DFT

La *Trasformata Discreta di Fourier*, come precedentemente analizzato, è un'operazione che consente di rappresentare vettori complessi nella base speciale di Fourier, dalla quale si possono ricavare importanti informazioni dei vettori originari. Inoltre, i vari algoritmi ottimizzati di FFT per il suo calcolo, rendono il costo computazionale estremamente basso. Per questi motivi, tale trasformazione è di grandissima utilità in molte applicazioni dove è decisivo analizzare segnali, passando dal dominio del tempo al dominio delle frequenze, ad esempio:

1. Elaborazione di audio ed immagini: la DFT è utilizzata per identificare componenti di frequenza dominanti di segnali audio, consentendo di intervenire in fase di registrazione o missaggio, o per identificare bordi, sfocature e pattern periodici per quanto riguarda le immagini. In aggiunta, è alla base degli algoritmi di compressione per estensioni quali MP3 o JPEG, dove componenti di frequenza originali poco significative vengono eliminate per ridurre le dimensioni dei file;

2. Analisi di segnali biologici: in ambito medico, la DFT gioca un ruolo chiave nell'analisi di segnali biologici come elettrocardiogrammi o elettroencefalogrammi, che permette di indagare le funzionalità del cuore e del cervello;

3. Chimica dei materiali: la DFT trova spazio anche nella spettroscopia infrarossa, un metodo per determinare la struttura delle molecole che raccolgono uno spettro vibrazionale molecolare, e per comprendere quindi la chimica superficiale di vari materiali.

Come nei suddetti casi, DFT e FFT vengono utilizzate nei più svariati campi della scienza in applicazioni pratiche, ma sono molto efficaci anche in ambiti matematici, tra cui la teoria dei numeri, la crittografia e, ovviamente, l'analisi numerica.
In questo elaborato si vuole analizzare l'applicazione, ad un primo impatto, un pò naif, dell'algoritmo di Cooley-Tukey alla moltiplicazione di polinomi, e successivamente mostrare come questa sia decisiva nel ridurre significativamente il costo computazionale di un problema classico dell'algebra: la moltiplicazione tra numeri interi.

3.2 Moltiplicazione Rapida di Polinomi

Nell'aritmetica dei polinomi, gioca un ruolo fondamentale poter determinare il prodotto di due o più polinomi di partenza in tempi accettabili, in quanto ciò può essere cruciale per risolvere problemi applicativi di varia natura.
L'ostacolo che ci si trova a dover superare è il costo computazionale elevato che tale operazione richiede al crescere del grado dei polinomi di cui si vuole calcolare il prodotto.
Si considerino due polinomi $A(x)$ e $B(x)$ con $deg(A) = deg(B) = d$:

$$A(x) = a_0 + a_1x + a_2x^2 + \dots a_dx^d$$
$$B(x) = b_0 + b_1x + b_2x^2 + \dots b_dx^d$$

dove a_j e b_j sono i coefficienti, con $a_d \neq 0$ e $b_d \neq 0$.
Il loro prodotto $C(x) = A(x)B(x)$ è un polinomio di grado $deg(C) = deg(A) + deg(B) = 2d$ i cui coefficienti sono calcolati come segue:

$$\begin{aligned}
c_0 &= a_0b_0 \\
c_1 &= a_0b_1 + a_1b_0 \\
&\vdots \\
c_i &= \sum_{j+k=i} a_jb_k \\
&\vdots \\
c_{2d-1} &= a_{d-1}b_d + a_db_{d-1} \\
c_d &= a_db_d
\end{aligned}$$

Esempio 1: Siano $A(x)$ e $B(x)$ due polinomi di grado 2 a coefficienti reali:

$$A(x) = x^2 + 3x + 2$$
$$B(x) = 2x^2 + x + 4$$

Determiniamo il polinomio prodotto $C(x) = A(x)B(x)$, calcolando i coefficienti utilizzando la proprietà distributiva, come illustrato sopra:

$$\begin{aligned} C(x) = A(x)B(x) &= (x^2 + 3x + 2)(2x^2 + x + 4) = \\ &= 2x^4 + x^3 + 4x^2 + 6x^3 + 3x^2 + 12x + 4x^2 + 2x + 8 = \\ &= 2x^4 + 7x^3 + 11x^2 + 14x + 8. \end{aligned}$$

$$\infty$$

Le operazioni richieste per il calcolo dei coefficienti di $C(x)$, utilizzando il metodo classico, sono dell'ordine di $\mathcal{O}(d^2)$.
Tale costo computazionale può essere significativamente ridotto, utilizzando algoritmi di FFT seguendo il seguente procedimento:

1. Valutare i polinomi-fattori $A(x)$ e $B(x)$ in *n radici n-esime dell'unità*, dove n è la più piccola potenza intera di 2 maggiore del grado di $C(x)$ con due DFT;
2. Calcolare i valori per $C(x)$, moltiplicando le valutazioni ottenute al punto 1;
3. Interpolare i valori ottenuti al punto 2, ottenendo i coefficienti di $C(x)$ mediante una IDFT.

Una volta implementato al calcolatore, tale algoritmo richiede il calcolo di 3 FFT/IFFT, n moltiplicazioni come al punto 2 e n moltiplicazioni per il fattore di normalizzazione della IDFT. In totale, il costo complessivo è dell'ordine di $\mathcal{O}\left(\frac{9}{2} n \log_2 n\right)$ operazioni aritmetiche.

3.2.1 Fase di Valutazione e Calcolo

Per ovviare al problema di rappresentare i polinomi al calcolatore, l'approccio più immediato consiste nel considerare i vari coefficienti e riordinarli in un vettore. Così facendo, un polinomio di grado d è rappresentato univocamente da un vettore di dimensione $d + 1$, in cui ogni elemento è un coefficiente. Ad esempio, nella prima entrata è situato il termine noto, nella seconda il coefficiente di x, nella terza il coefficiente di x^2 e così via.
Se da un lato tale rappresentazione è facilmente implementabile e di immediata comprensione, dall'altro risulta totalmente inefficace all'obiettivo prefissato, ovvero calcolare in modo strategico e veloce il prodotto di polinomi.

Dalle suddette osservazioni si può notare che, considerato un polinomio di grado d, esso è univocamente definito come l'interpolatore di $d+1$ punti del piano, in quanto per calcolare i vari coefficienti occorre risolvere un sistema di $d+1$ equazioni in $d+1$ incognite[2]. Ne consegue che, dati $d+1$ valori $(x_i)_{i=0,\dots,d}$, un generico polinomio $P(x)$ di grado d è determinato dalla lista di coppie $\{(x_0; P(x_0)), (x_1; P(x_1)), \dots, (x_d; P(x_d))\}$.
Grazie a tale rappresentazione, dati due polinomi $A(x)$ e $B(x)$ di grado d , risulta semplice individuare il polinomio prodotto $C(x) = A(x)B(x)$.
Si considerino $2d+1$ punti $(x_i)_{i=0,\dots,2d+1}$ in cui valutare i due fattori come segue:

$$A(x) \to \{(x_0; A(x_0)), (x_1; A(x_1)), \dots, (x_{2d+1}; A(x_{2d+1}))\}$$
$$B(x) \to \{(x_0; B(x_0)), (x_1; B(x_1)), \dots, (x_{2d+1}; B(x_{2d+1}))\}$$

Allora $C(x)$ è determinato dalla lista:

$$C(x) \to \{(x_0; A(x_0)B(x_0)), (x_1; A(x_1)B(x_1)), \dots, (x_{2d+1}; A(x_{2d+1})B(x_{2d+1}))\} \tag{3.1}$$

Esempio 2: Siano $A(x)$ e $B(x)$ i polinomi definiti nell'Esempio 1.
Seguendo quanto appena discusso, considerando $x_0 = -2, x_1 = -1, x_2 = 0, x_3 = 1, x_4 = 2$, essi possono essere rappresentati come segue:

$$A(x) \to \{(-2; 0), (-1; 0), (0; 2), (1; 6), (2; 12)\}$$
$$B(x) \to \{(-2; 10), (-1; 5), (0; 4), (1; 7), (2; 14)\}$$

Ne consegue, dalla (3.1), che $C(x)$ è determinato come valutazione in $5 = deg(C) + 1$ punti come:

$$C(x) \to \{(-2; 0), (-1; 0), (0; 8), (1; 42), (2; 168)\}.$$

$$\infty$$

Calcolare in questo modo il polinomio risultante dalla moltiplicazione dei due iniziali richiede, anche ad una prima impressione, molte meno operazioni. Il problema che ci si trova ad affrontare è come effettuare il passaggio da una rappresentazione di polinomi come vettore di coefficienti ad una rappresentazione per valutazione, ovvero come lista di coppie ascissa-valore. Inoltre, ci si chiede se si possono trovare delle ascisse particolari in modo da ridurre di molto il costo computazionale.

[2] Segue direttamente dal Teorema di Rouché-Capelli.

Il metodo più efficace per superare questo ostacolo consiste nello sfruttare le proprietà di simmetria intrinseche nei vari polinomi.
Si consideri a titolo di esempio il polinomio $P(x) = x^2$. Essendo la funzione $y = x^2$ pari, il valore che il polinomio assume in una generica ascissa x_j e nell'opposta $-x_j$ è lo stesso. Analogamente, considerato il polinomio $Q(x) = x^3$, i valori $Q(x_j)$ e $Q(-x_j)$ sono opposti, essendo la funzione $y = x^3$ dispari. In entrambi i casi, considerate n ascisse, si possono ottenere rapidamente $2n$ punti di valutazione per il polinomio considerato.
Quest'ultima idea si può estendere in modo astuto a polinomi più complessi, come esplicato nel seguente:
Esempio 3: Si consideri il polinomio:

$$P(x) = 3x^5 + 2x^4 + x^3 + 7x^2 + 5x + 1$$

Separando i termini dove la x compare con esponenti pari da quelli dove compare con esponenti dispari:

$$P(x) = 2x^4 + 7x^2 + 1 + 3x^5 + x^3 + 5x$$

e raccogliendo la x da quest'ultimi si ottiene:

$$P(x) = (2x^4 + 7x^2 1) + x(3x^4 + x^2 + 5)$$

Si può così scrivere il polinomio come:

$$P(x) = P_e(x^2) + xP_o(x^2)$$

dove

$$P_e(x^2) = 2x^4 + 7x^2 + 1$$
$$P_o(x^2) = 3x^4 + x^2 + 5.$$

$$\infty$$

Così facendo, considerate due coppie di ascisse x_j e $-x_j$, opposte tra loro, si ha che:

$$P(x_j) = P_e(x_j^2) + xP_o(x_j^2) \tag{3.2}$$
$$P(-x_j) = P_o(x_j^2) - xP_o(x_j^2) \tag{3.3}$$

Come mostrato, in generale, dato un polinomio di grado $n - 1$, esso può essere valutato in n punti, a due a due opposti, $\pm x_1, \ldots, \pm x_{n/2}$. I polinomi $P_e(x_j^2)$ e $P_o(x_j^2)$, definiti come in (3.2)

e (3.3), possono quindi essere valutati in $\frac{n}{2}$ punti $x_1^2, \ldots, x_{n/2}^2$.
Ora occorre comprendere come poter reiterare il procedimento applicato a $P(x)$ anche a $P_e(x^2)$ e $P_o(x^2)$, andando a costituire un algoritmo ricorsivo, con lo scopo di abbassare il costo computazionale come prefissato. Per far sì che ciò avvenga, le ascisse considerate nelle iterazioni successive alla prima devono essere necessariamente, a due a due, una opposta all'altra, come avveniva per i punti iniziali $\pm x_1, \ldots, \pm x_{n/2}$. Tutto ciò è realizzabile se inizialmente come ascisse x_j vengono considerati dei numeri complessi, è più precisamente le *radici n-esime dell'unità*.
In generale, dato un polinomio di grado d, considero $n \geq d+1$ valori tali che $n = 2^k$. Le *radici n-esime dell'unità* definite come:

$$\begin{aligned} W_n^0 &= 1 \\ W_n^1 &= e^{\frac{2\pi i}{n}} \\ W_n^2 &= e^{\frac{2\pi i}{n} \cdot 2} \\ &\vdots \\ W_n^{(n-1)} &= e^{\frac{2\pi i}{n} \cdot (n-1)} \end{aligned}$$

risolvono il problema presentatosi, poichè ad ogni iterazione sono sempre identificabili le coppie di segno opposto, in quanto $W_n^j = -W_n^{j+n/2}$ e si dimezzano in numero, mantenendo però la stessa proprietà.
Esempio 4: Si consideri il polinomio:

$$P(x) = x^3 + x^2 - x - 1$$

Per valutarlo sono necessari 4 punti. Se si considerano, al primo passo, le radici 4^e dell'unità:

$$\begin{aligned} x_0 &= W_4^0 = 1 \\ x_1 &= W_4^1 = i \\ x_2 &= W_4^2 = -1 \\ x_3 &= W_4^3 = -i \end{aligned}$$

dove $x_0 = -x_2$ e $x_1 = -x_3$, alla seconda iterazione si dovranno valutare i polinomi P_e e P_o in x_0^2 e in $x_1^2 = -x_0^2$, mantenendo le proprietà di simmetria dei valori iniziali e dando quindi la possibilità di reiterare l'algoritmo.

∞

3.2.2 Fase di Interpolazione

Una volta determinato il polinomio prodotto con il metodo suddetto, esso sarà scritto come lista di coppie ascissa-valore. Per passare ad una rappresentazione a coefficienti, come descritto ad inizio trattazione, occorrerà attuare un processo di interpolazione nei vari punti.
Considerato il polinomio di grado $n-1$:

$$P(x) = p_0 + p_1 x + \ldots + p_{n-1}x^{n-1}$$

e chiamati $x_0, x_1, \ldots, x_{n-1}$ le n ascisse in cui il polinomio è stato valutato, si ha che:

$$\begin{pmatrix} P(x_0) \\ \vdots \\ P(x_{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_{n-1} \end{pmatrix}$$

Nel caso proposto, le varie x_j sono le *radici n-esime dell'unità*, allora si ottiene che:

$$\begin{pmatrix} P(W_n^0) \\ \vdots \\ P(W_n^{n-1}) \end{pmatrix} = \begin{pmatrix} W_n^{0\cdot 0} & W_n^{0\cdot 1} & W_n^{0\cdot 2} & \ldots & W_n^{0\cdot(n-1)} \\ W_n^{1\cdot 0} & W_n^{1\cdot 1} & W_n^{1\cdot 2} & \ldots & W_n^{1\cdot(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_n^{(n-1)\cdot 0} & W_n^{(n-1)\cdot 1} & W_n^{(n-1)\cdot 2} & \ldots & W_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_{n-1} \end{pmatrix}$$

La matrice considerata è una *Matrice di Fourier*, come in Definizione 2.4.
Per trovare quindi i coefficienti $p_0, \ldots, p_{n-1}$, basterà determinare l'inversa di tale matrice, seguendo il Corollario 2.1. Si otterrà una matrice della forma:

$$\frac{1}{n}\begin{pmatrix} W_n^{-0\cdot 0} & W_n^{-0\cdot 1} & W_n^{-0\cdot 2} & \ldots & W_n^{-0\cdot(n-1)} \\ W_n^{-1\cdot 0} & W_n^{-1\cdot 1} & W_n^{-1\cdot 2} & \ldots & W_n^{-1\cdot(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_n^{-(n-1)\cdot 0} & W_n^{-(n-1)\cdot 1} & W_n^{-(n-1)\cdot 2} & \ldots & W_n^{-(n-1)\cdot(n-1)} \end{pmatrix}$$

dove ogni *radice n-esima dell'unità* W_n^{jk} è stata sostituita da W_n^{-jk}. Si è davanti ad una IDFT, che mantiene tutte le proprietà che la rendono un algortimo reiterabile per risolvere il problema di interpolazione, inverso al primo problema di valutazione.[3]

[3] Si noti che in questo capitolo, rispetto alle definizioni 2.6 e 2.7, data la *Matrice di Fourier* Ω_n, si è definita la DFT come l'applicazione di Ω_n e la IDFT come $\frac{1}{n}\Omega_n^H$, per coerenza con le usuali notazioni.

3.2.3 Implementazione in Matlab

Di seguito sono riportate le implementazioni degli algoritmi di Valutazione e Interpolazione descritti nelle sezioni 3.2.1 e 3.2.2 in Matlab.
Nell'algoritmo di Valutazione, considerato un polinomio di grado $n-1$:

$$P(x) = p_0 + p_1 x + \ldots + p_{n-1} x^{n-1}$$

la funzione prende in input un vettore:

$$p = (p_0, p_1, \ldots, p_{n-1})$$

le cui entrate sono i coefficienti di $P(x)$ e restituisce un vettore:

$$y = (y_0, y_1, \ldots, y_{n-1})$$

le cui componenti sono le valutazioni del polinomi nelle *radici n-esime dell'unità*.

```
function y = FFT_eval(p)                          %sia p vettore di coefficienti
    n = length(p);                                %sia n la sua lunghezza
    if n == 1                                     %caso base, se n=1
        y = p;
        return;
    end
    p_e = p(1:2:end);                             %si separino i coefficienti
    p_o = p(2:2:end);                             %dei termini di esponente pari
    y_e = FFT_eval(p_e);                          %dai termini dispari
    y_o = FFT_eval(p_o);

    for k = 1:n/2
        twiddle = exp(2*pi*1i*(k-1)/n);
        y(k) = y_e(k) + twiddle * y_o(k);
        y(k+n/2) = y_e(k) - twiddle * y_o(k);
    end
end
```

Nell'algoritmo di Interpolazione, invece, l'input è un vettore:

$$y = (y_0, y_1, \ldots, y_{n-1})$$

le cui entrate sono i valori assunti dal polinomio prodotto nelle *radici n-esime dell'unità*, mentre l'output è un vettore:

$$p = (p_0, p_1, \ldots, p_{n-1})$$

le cui componenti sono i coefficienti di suddetto polinomio.

```
function p = IFFT_inter(y)                          %sia y vettore di valori
    n = length(y);                                  %sia n la sua lunghezza
    if n == 1                                       %caso base, se n=1
        p = y;
        return;
    end
    y_e = y(1:2:end);                               %si separino i coefficienti
    y_o = y(2:2:end);                               %dei termini di esponente pari
    p_e = IFFT_inter(y_e);                          %dai termini dispari
    p_o = IFFT_inter(y_o);

    for k = 1:n/2
        twiddle = (1/n)*exp(-2*pi*1i*(k-1)/n);
        p(k) = p_e(k) + twiddle * p_o(k);
        p(k+n/2) = p_e(k) - twiddle * p_o(k);
    end
end
```

3.3 Moltiplicazione Rapida di Numeri Interi

Il problema di dover calcolare prodotti di numeri interi enormi con parsimonia di tempo e spazio è diffuso in svariati settori della matematica pura ed applicata. Basti pensare alla crittografia: l'algoritmo di RSA[4] si basa sulla fattorizzazione di interi immensi in numeri primi.
Tutt'al più, trovare metodi efficienti per eseguire moltiplicazioni di tale portata risulta utile per ottimizzare qualsiasi altro dilemma che può essere ricondotto al calcolo di prodotti di interi; ad esempio, approssimare il valore di radici k-esime, di funzioni trascendenti e di costanti come e^x o π, o determinare il massimo comun divisore di numeri interi.
Di seguito si vedrà come lo studio di tale problema trovi palesi analogie con quanto trattato nella sezione 3.2, verificando quanto gli algoritmi di FFT siano stati necessari al raggiungimento dell'obiettivo di abbassare significativamente la complessità computazionale richiesta per il calcolo di prodotti di numeri interi.
Dati due numeri naturali di n cifre, la naturale moltiplicazione tra essi, così come appresa in ambiente scolastico, richiede n^2 operazioni tra le singole cifre. Già alla metà del secolo scorso, il matematico russo Anatoly Karatsuba scoprì l'omonimo algoritmo, il quale, utilizzando un metodo di *divide et impera*, abbassava il costo computazionale ad $n^{\log_2 3}$. Una decina di anni dopo, nel 1971, Arnold Schönhage e Volker Strassen, presentarono un algoritmo alternativo che diminuiva bruscamente la complessità, e che rimase l'algoritmo asintoticamente più veloce in risoluzione a tale problema fino allo scorso decennio: esso richiedeva circa $n \log n \log(\log n)$ passi[5] [18]. Inoltre, questo netto miglioramento suggerì agli autori che tale risultato potesse essere ulteriormente perfezionato. Fu così che formularono una congettura: si ipotizzò che la complessità del problema poteva essere ancora minore, esattamente dell'ordine di $n \log n$.
Questa deduzione rimase tale per ben 48 anni, finchè, nel recente 2019, David Harvey e Joris van der Hoeven riuscirono a confermarla, pubblicando il risultato nell'articolo *Integer multiplication in time $\mathcal{O}(n \log n)$*, dove dimostrarono il seguente:

Teorema 3.1 *Dati due numeri interi di n cifre ciascuno, il tempo richiesto per determinare il loro prodotto è:*

$$M(n) = \mathcal{O}(n \log n)$$

Mentre l'algoritmo di Arnold Schönhage e Volker Strassen consisteva nell'applicazione di FFT unidimensionali, e che si riconduceva quindi ad ottimizzare moltiplicazioni di polinomi, il

[4] Algoritmo di *crittografia asimmetrica*, inventato nel 1977 da Ronald Rivest, Adi Shamir e Leonard Adleman.
[5] A. Schönage, V. Strassen, Schnelle Multiplikation grosser Zahlen. Computing (Arch. Elektron. Rechnen) 7 (1971), 281–292.

metodo di Harvey e van der Hoeven utilizza FFT multidimensionali: nella suddetta pubblicazione, venne utilizzata una FFT di 1729 dimensioni. L'applicazione di tale risultato, prettamente teorico, richiede però numeri estremamente grandi. Infatti, lo stesso Harvey affermò:

«The new algorithm is not really practical in its current form, because the proof given in our paper only works for ludicrously large numbers. Even if each digit was written on a hydrogen atom, there would not be nearly enough room available in the observable universe to write them down.»[6]

L'obiettivo di tale sezione è di mostrare, in via del tutto euristica, come il problema di moltiplicare rapidamente numeri interi si riduca velocemente all'ottimizzazione di moltiplicazioni tra polinomi, seguendo l'idea proposta da Schönage e Strassen e perfezionata da Harvey e van der Hoeven.

3.3.1 Algoritmo

Sia A un numero intero di n cifre.
Esso può essere univocamente determinato dal vettore:

$$a = (a_0, a_1, \ldots, a_{n-1})$$

dove la prima entrata è la cifra delle unità, la seconda delle decine, e così via. Si noti che l'indice della componente j-esima del vettore coincide con la potenza j-esima di 10. Ne consegue che per determinare A partendo dal vettore contenente le sue cifre a basta valutare il polinomio di grado $n-1$:

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

in $x = 10$.
In questo modo ci si è ricondotti a considerare, al posto di un numero intero di n cifre, un polinomi di grado $n-1$ i cui coefficienti sono composti da una sola cifra.
Per di più, come intuito da Schönhage e Strassen, per abbassare il grado dei polinomi, si possono considerare coefficienti da più cifre. Così facendo, il numero intero A può essere il risultato

[6] D. Harvey, "We've found a quicker way to multiply really big numbers", *The conversation*, 2019.
«Il nuovo algoritmo non è molto pratico nella sua attuale forma, poichè la dimostrazione data nel nostro articolo funziona solo per numeri ridicolosamente grandi. Anche se ogni cifra fosse scritta in un atomo di idrogeno, non ci sarebbe abbastanza spazio nell'Universo osservabile per scriverle tutte.» trad.

della valutazione di un polinomio a coefficienti di n cifre in $x = 10^n$.
Esempio 5: Si consideri il numero intero di 9 cifre:

$$A = 123456789$$

Allora esso può essere calcolato valutando in $x = 10$ il polinomio di grado 8:

$$A(x) = 9 + 8x + 7x^2 + 6x^3 + 5x^4 + 4x^5 + 3x^6 + 2x^7 + 1x^8$$

Infatti:

$$A(10) = 9+80+700+6000+50000+400000+3000000+20000000+100000000 = A$$

In alternativa, volendo operare con polinomi di grado minore, A può essere considerato come la valutazione in $x = 10^3$ di:

$$\bar{A} = 789 + 456x + 123x^2$$

in quanto:

$$\bar{A}(10^3) = 789 + 456000 + 123000000 = A$$

$$\infty$$

Considerati, dunque, due numeri interi di cui si vuole calcolare il prodotto, seguendo questa semplice quanto efficace intuizione ci si riconduce a dover ottimizzare la moltiplicazione di polinomi, come trattato nella sezione 3.2.

Conclusione

Il presente studio si è posto l'obiettivo di riorganizzare diverso materiale sull'Analisi di Fourier, contestualizzando i vari risultati raggiunti nel periodo storico di riferimento, con particolare attenzione agli algoritmi di FFT. Inoltre, è stata approfondita un'applicazione marginalmente trattata in letteratura: la moltiplicazione rapida di polinomi.

Infatti, come è stato riportato, gli algoritmi di FFT nacquero per ottimizzare un problema pratico, la grande complessità computazionale dovuta all'interpretazione dei sismografi, e ad oggi trovano la loro quasi completa utilità in ambiti fisici o ingegneristici.

Tuttavia, ciò che si è voluto svelare in tale elaborato è che anche una loro semplice applicazione teorica sull'algebra dei polinomi può risultare cruciale per ottimizzare grandi problemi computazionali tutt'ora argomento di ricerca e di concreto utilizzo, come la moltiplicazione rapida di numeri interi.

In conclusione, questo lavoro di tesi si è posto come un'ulteriore testimonianza di come lo studio e l'approfondimento di temi prettamente teorici ed algebrici possano avere riscontri concreti e decisivi in questioni pratiche e in via di sviluppo.

Bibliografia

[1] L. Ahlfors, *Complex Analysis*, 1st ed. McGraw-Hill, 1953.

[2] K. H. Barth, "Science and politics in early nuclear test ban negotiations," *Physics Today*, vol. 51, no. 3, pp. 34–39, 1998.

[3] D. A. Bini, "La Trasformata Discreta di Fourier e la FFT," 2020, Università di Pisa. [Online]. Available: https://people.dm.unipi.it/bini/Didattica/AnaNum/testi/Dispense/fft.pdf

[4] E. O. Brigham, *The Fast Fourier Transform And Its Applications*, 1st ed. Prentice-Hall Signal Processing Series, 1988.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[6] A. Einstein, "Arms can bring no security," *Bulletin of the Atomic Scientists*, vol. 6, no. 3, p. 71, 1950.

[7] J. B. J. Fourier, *Théorie analytique de la chaleur*, 1st ed. Firmin-Didot, Parigi, 1822.

[8] A. Goldstein, "James W. Cooley, an oral history," *IEEE History Center, Piscataway, NJ, USA*, 1997.

[9] H. H. Goldstine, *A History of Numerical Analysis from the 16th Through the 19th Century*, 1st ed. New York, Heidelberg, and Berlin: Springer-Verlag, 1977.

[10] I. J. Good, "The interaction algorithm and practical fourier analysis," *Journal of the Royal Statistical Society*, vol. 20, pp. 361–372, 1958.

[11] D. Harvey, "We've found a quicker way to multiply really big numbers," *The Conversation*, 2019. [Online]. Available: https://theconversation.com/weve-found-a-quicker-way-to-multiply-really-big-numbers-114923

[12] D. Harvey and J. van der Hoeven, "Integer multiplication in time O(n log n)," *Annals of Mathematics*, vol. 193, pp. 563–617, 2019.

[13] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the Fast Fourier Transform," *Archive for History of Exact Sciences*, vol. 34, no. 3, pp. 265–277, 1985.

[14] F. Montefalcone, "Metodi Matematici," 2021, dispense del corso di Metodi Matematici, Università degli Studi di Padova. [Online]. Available: https://elearning.unipd.it/math/pluginfile.php/92875/mod_resource/content/32/DispenseMetodi.pdf

[15] R. W. Ramirez, *The FFT: fundamentals and concept*, 1st ed. Tektronix, 1975.

[16] M. Redivo-Zaglia, *Calcolo Numerico, metodi ed algoritmi*, 4th ed. Progetto Libreria, 2015.

[17] D. N. Rockmore, "The FFT: an algorithm the whole family can use," *Computing in Science And Engineering*, vol. 2, no. 1, pp. 60–64, 2000.

[18] A. Schönage and V. Strassen, "Schnelle multiplikation grosser zahlen," *Computing (Arch. Elektron. Rechnen)*, vol. 7, pp. 281–292, 1971.

[19] A. Sommariva, "Quadratura Numerica," 2021, dispense del corso di Analisi Numerica, Università degli Studi di Padova. [Online]. Available: https://www.math.unipd.it/~alvise/CN_2015_INFO/LEZIONE_5_QUADRATURA/laboratorio3_quadratura.pdf

[20] G. Strang, "Wavelets," *American Scientist*, vol. 82, no. 3, pp. 250–255, 1994.