Capitolo 5

La Trasformata discreta di Fourier e l'algoritmo FFT

5.1 Introduzione

In questo capitolo, trattiamo algoritmi per il calcolo di somme del tipo:

$$S_j = \sum_{k=0}^{N-1} s_k e^{-\frac{2\pi}{N} ijk} , \qquad i = \sqrt{-1}, \quad j = 0, 1, \ldots, N-1 \tag{5.1}$$

dove $\{s_k\}_{k=0,\ldots,N-1} \in \mathbb{C}^N$ è un vettore le cui componenti sono numeri complessi [1].

[1] Utilizzando l'identità di Eulero:

$$e^{i\theta} = \cos\theta + i\sin\theta \quad , \ \theta \in \Re, \tag{5.2}$$

segue che:

$$S_j = \sum_{k=0}^{N-1} (Re(s_k) + iIm(s_k)) \left[\cos\left(\frac{2\pi jk}{N}\right) - i\sin\left(\frac{2\pi jk}{N}\right)\right] \tag{5.3}$$

e quindi,

$$Re(S_j) = \sum_{k=0}^{N-1} \left[Re(s_k)\cos\left(\frac{2\pi jk}{N}\right) + Im(s_k)\sin\left(\frac{2\pi jk}{N}\right)\right] \tag{5.4}$$

e

$$Im(S_j) = \sum_{k=0}^{N-1} \left[Im(s_k)\cos\left(\frac{2\pi jk}{N}\right) - Re(s_k)\sin\left(\frac{2\pi jk}{N}\right)\right]. \tag{5.5}$$

Se, in particolare, i valori s_k sono numeri reali, allora:

$$Re(S_j) = \sum_{k=0}^{N-1} \left[s_k\cos\left(\frac{2\pi jk}{N}\right)\right] \tag{5.6}$$

e

La somma (5.1), a partire da un vettore $\{s_k\}_{k=0,\ldots,N-1}$, definisce il vettore $\{S_k\}_{k=0,\ldots,N-1}$. A tale operatore, come verrà precisato in seguito, viene dato il nome di **Trasformata Discreta di Fourier (DFT)**.
Come è facile osservare dalla (5.1), il calcolo diretto della DFT ha una complessità dell'ordine di $\mathcal{O}(N^2)$. Negli ultimi anni sono stati messi a punto versioni sempre più specializzate dell' algoritmo **FFT (Fast Fourier Transform)** originariamente proposto da Cooley e Tukey, nel 1965. In questo capitolo, dopo aver introdotto l'operatore DFT e alcune sue proprietà fondamentali, descriviamo l'idea e la metodologia alla base degli algoritmi FFT. Vedremo come, attraverso l'approccio *divide et impera,* il calcolo di una DFT di lunghezza $N = r \cdot q$ possa essere ricondotto a quello di q DFT di lunghezza r, e così proseguendo, ottenendo, ad esempio se $N = 2^m$, una riduzione della complessità di tempo da $\mathcal{O}(N^2)$ a $\mathcal{O}(N \log_2 N)$. Lo stesso approccio, alla base del software di libreria che implementa gli algoritmi FFT, combinato con tecniche di ottimizzazione delle prestazioni implementate dinamicamente in fase di compilazione e di esecuzione (*AEOS - Automated Empirical Optimization of Software*) ha fatto della FFTW (*Fast Fourier Transform in the West)*, la libreria sviluppata nel 1999 da M. Frigo e S. Johnson presso il MIT *(Massachusetts Institute of Technology)* per il calcolo della DFT di un vettore di lunghezza N, il miglior prodotto software in termini di efficienza, accuratezza ed affidabilità (*J. H. Wilkinson Prize for Numerical Software, 2000*).

♣ **Esempio 5.1.** Anche se è consuetudine attribuire a Cooley e Tukey la prima formulazione di un algoritmo della classe FFT per il calcolo della DFT, in effetti già il trattato sull'interpolazione scritto da *Carl Friedrich Gauss* nel 1805, pubblicato postumo nel 1866, contiene un chiaro riferimento all'idea che sottende il calcolo veloce di una DFT mediante algoritmi FFT [16]. A tal proposito, i contributi di Gauss furono citati solo a partire dal 1904 nell'enciclopedia matematica di Burkhardt e successivamente nel 1977 da Goldstine [15].
Consideriamo, quindi, il problema del quale si interessò Gauss [11]. Nella tabella 5.1 si riportano, così come apparsi nel lavoro di Gauss, i valori indicanti la posizione dell'asteroide Pallas, espressa in termini delle variabili (θ, X) che esprimono rispettivamente l'ascensione e la declinazione [2].
A partire da tali misure il problema consisteva nel disegnare la traiettoria dell'asteroide utilizzando un modello interpolante che fosse basato su un polinomio trigonometrico $p(\theta)$, di grado $N+1$, del tipo:

$$p(\theta) = a_0 + \sum_{i=1}^{N} \left[a_k \cos\left(\frac{2\pi k\theta}{360}\right) + b_k \sin\left(\frac{2\pi k\theta}{360}\right)\right] + a_N \cos\left(\frac{2\pi(N+1)\theta}{360}\right).$$

Poiché i dati a disposizione sono 12, come indicato nella tabella, e poiché la costruzione del polinomio $p(\theta)$ comporta la determinazione di $2(N+1)$ coefficienti in questo caso fu necessario fissare $N = 5$. Indicate con (θ_k, X_k), $k = 0, \ldots, 11$ le coppie relative ai valori dell'ascensione e declinazione riportati

$$Im(S_j) = \sum_{k=0}^{N-1} \left[-s_k \sin\left(\frac{2\pi jk}{N}\right)\right]. \tag{5.7}$$

[2]Ascensione e Declinazione sono le coordinate utilizzate in astronomia per localizzare la posizione di un oggetto sulla sfera celeste. Tali coordinate sono analoghe alle coordinate tipicamente utilizzate per definire la posizione di un oggetto sulla sfera terrestre, longitudine e latitudine.

Ascensione	0	30	60	90	120	150
Declinazione	408	89	-66	10	338	807
Ascensione	180	210	240	270	300	330
Declinazione	1238	1511	1583	1462	1183	804

Tabella 5.1: Posizione dell'asteroide Pallas espressa in termini delle coordinate (l'Ascensione è espressa in gradi e la Declinazione in minuti).

nella tabella, e imponendo le condizioni di interpolazione:

$$p(\theta_k) = X_k, \quad k = 0, \ldots, 11$$

si ottiene un sistema di equazioni lineari di ordine 12 la cui soluzione fornisce i coefficienti del polinomio p. Gauss per risolvere tale sistema, utilizzando opportunamente le proprietà di simmetria e periodicità delle funzioni trigonometriche scoprí un modo per *decomporre tale problema in sottoproblemi le cui soluzioni, una volta combinate tra loro, fornirono la soluzione del problema con* $N(3+4) = 5 \cdot 7 = 35$ *operazioni* [3]. Questo procedimento è proprio quello che è alla base dell'algoritmo di FFT. ♣

Da ora in poi indicheremo con w_N l'esponenziale complesso:

$$w_N = e^{\frac{2\pi i}{N}} \quad .$$

Le potenze w_N^k con $k = 0, 1, \ldots, N-1$, dette **radici primitive N-me dell'unità**[4], rappresentano le soluzioni nel campo complesso dell'equazione

$$z^N = 1, \quad \forall\, z \in \mathbb{C}\,,$$

e svolgono un ruolo particolarmente significativo nel calcolo di una DFT poichè la complessità computazionale di una DFT dipende, oltre che dalle operazioni floating point necessarie al calcolo della somma in (5.1), anche dalla loro valutazione. Come vedremo, alla base degli algoritmi FFT vi è una opportuna riformulazione della DFT (derivante dalla fattorizzazione del parametro N) che, sfruttando la periodicità e la simmetria delle funzioni trigonometriche, consente di evitare inutili e ridondanti valutazioni di tali funzioni.

[3] L'algoritmo di eliminazione dello stesso Gauss avrebbe richiesto circa $1210 = \frac{12^3}{3} + \frac{12^2}{2}$ operazioni floating point.

[4] Geometricamente, le radici primitive N -me dell'unità si possono associare ai vertici del poligono regolare a N lati inscritto nel cerchio goniometrico. Tali grandezze si ripetono ciclicamente, ovvero $\omega_N^k = \omega_N^{(k\, modulo\, N)}$. Una utile proprietà delle radici N-me dell'unità è la seguente:

$$1 + \omega^r + \omega^{2r} + \ldots + \omega^{(N-1)r} = 0, \tag{5.8}$$

per ogni intero positivo $r \in \mathcal{N}$. L'appellativo *primitive* deriva dal fatto che ciascuna radice ha la proprietà di riprodurre con le potenze di esponente da 0 a $N-1$ tutte le radici N-me dell'unità.

♣ **Esempio 5.2.** Supponiamo di conoscere in $N = 16$ punti $x_k \in [0, 2\pi]$:

$$x_k = \frac{2\pi}{16}k, \quad k = 0, ..., 15, \tag{5.9}$$

i valori y_k, $k = 0, \ldots, 15$ di una funzione periodica[5].

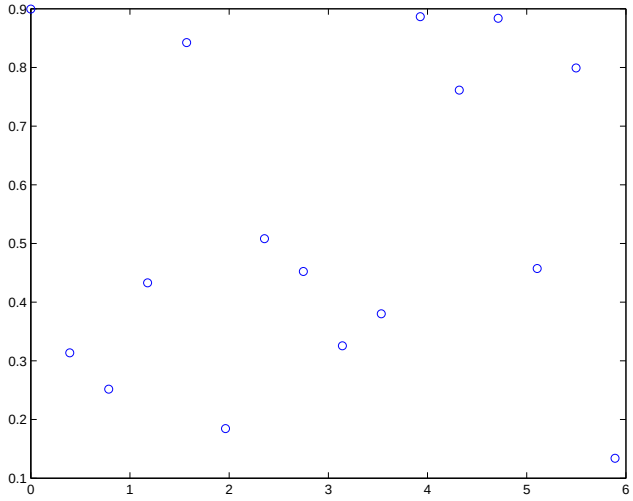

Figura 5.1: Rappresentazione dei punti di coordinate (x_k, y_k), $k = 0, 1, ..., 15$ con $x_k = \frac{2\pi}{N}k$, $y_k \in \mathbb{C}$, $N = 16$.

Al fine di costruire un modello che descriva i dati è ragionevole pensare ad un modello approssimante avendo assunto che le quantità y_k siano affette dall'errore (non trascurabile) introdotto dai dispositivi utilizzati per la loro aquisizione. Inoltre, essendo il fenomeno periodico, per descriverne l'andamento consideriamo come modello un polinomio trigonometrico di grado $N - 1 = 15$, del tipo:

$$p(x) = \frac{a_0}{2} + a_1 cos(x) + \ldots + a_{15} cos(15x)$$

In particolare, il problema è determinare i coefficienti a_j. A tal fine, imponiamo che

$$p^*(x) = \min_p ||y_i - p(x_i)||_2$$

ovvero, imponiamo che p^* sia il polinomio di migliore approssimazione relativo ai punti $(x_k, y_k)_{k=0,N-1}$ nel senso dei minimi quadrati, nello spazio dei polinomi trigonometrici di grado al più 15.
Tale problema conduce alla risoluzione del sistema (vedi **Capitolo 3**, **Parte prima**):

$$A^T \cdot Aa = A^T y$$

essendo:

$$A = \begin{pmatrix} 1 & \cos(x_1) & ... & ... & \cos(15x_1) \\ 1 & \cos(x_2) & ... & ... & \cos(15x_2) \\ 1 & \cos(x_3) & ... & ... & \cos(15x_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cos(x_{15}) & ... & ... & \cos(15x_{15}) \end{pmatrix}$$

[5]Piú in generale, i valori y_k possono essere valori corrispondenti ai nodi $x_k = \frac{T}{N}k$, $k = 0, \ldots, N-1$, con x_k appartenenti ad un generico intervallo $[0, T]$. In tal caso basta utilizzare la funzione $t = \frac{2\pi}{T}x$, che trasforma l'intervallo $[0, T]$ nell'intervallo $[0, 2\pi]$ e risolvere il problema in quest'ultimo intervallo.

$$A^T y = \begin{pmatrix} 2\sum_{i=0}^{15} y_i \\ 2\sum_{i=0}^{15} y_i \cos(x_i) \\ \vdots \\ \vdots \\ 2\sum_{i=0}^{15} y_i \cos(15x_i) \end{pmatrix}$$

e $a = (a_0, \ldots, a_{15})$, $y = (y_1, \ldots, y_{15})$.
Poiché [6]:

$$\sum_{i=0}^{15} \cos(jx_i)\cos(kx_i) = \begin{cases} 0 & j \neq k \\ \frac{16}{2} & j = k \neq 0 \\ 16 & j = k = 0 \end{cases} \tag{5.10}$$

$$\sum_{i=0}^{15} \cos(jx_i) = \begin{cases} 0 & j \neq 16 \\ 16 & j = 16 \end{cases} \tag{5.11}$$

la matrice del sistema diventa:

$$A^T \cdot A = \begin{pmatrix} 16 & 0 & 0 & ... & 0 \\ 0 & 16 & 0 & ... & 0 \\ 0 & 0 & 16 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & ... & 16 \end{pmatrix}$$

la cui soluzione è, ovviamente:

$$a_j^* = \frac{2}{16}\sum_{k=0}^{15} y_k \cos\frac{2\pi kj}{16}, \qquad j = 0, ..., 15$$

Analogamente se:

$$p^*(x) = \frac{b_0}{2} + b_1^* \sin(x) + \ldots + b_{12}^* \sin(15x) \tag{5.12}$$

i coefficienti b_j^* sono:

$$b_j^* = \frac{2}{16}\sum_{k=0}^{15} y_k \sin\frac{2\pi kj}{16}, \qquad j = 0, 1, ..., 15$$

Tenendo conto delle (5.6) e (5.7) i coefficienti del polinomio p^*, a meno del segno e del fattore moltiplicativo 2/16, sono rispettivamente la parte reale e la parte immaginaria delle quantità S_j introdotte nella (5.1).

Il risultato al quale siamo pervenuti è di carattere generale. Posto $z = e^{ix}$, con $i = \sqrt{-1}$, assegnati N punti di coordinate $(z_k, y_k), k = 0, \ldots, N-1$, dove $z_k = e^{ix_k}$ con $x_k = \frac{2\pi}{N}k$ sussiste la seguente:

[6] Queste relazioni si ricavano utilizzando la proprietà (5.8) delle radici N-me dell'unità.

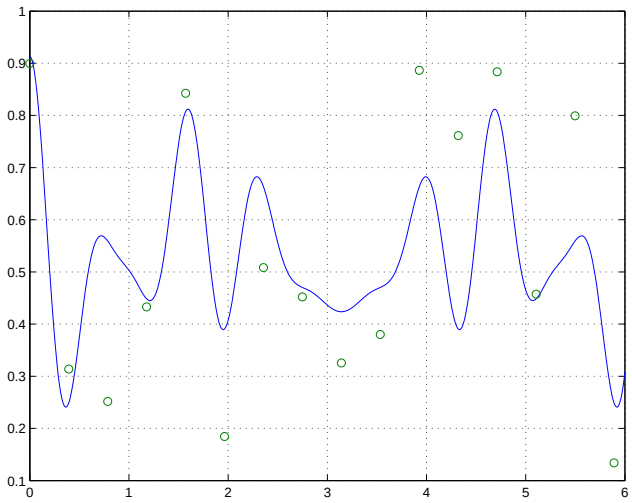

Figura 5.2: Grafico del polinomio trigonometrico p^* di migliore approssimazione nel senso dei minimi quadrati relativo ai punti $(x_i, y_i), \quad i = 0, 1, ..., 15$ dell'**esempio 5.2**.

Proposizione 5.1.1. *Esiste un unico polinomio trigonometrico del tipo*

$$p(x) = a_0 + a_1 z + a_2 z^2 + \ldots + a_{N-1} z^{N-1}, \quad a_j = \frac{2}{N} \sum_{k=0}^{N-1} y_k e^{-\frac{2\pi}{N} ikj}, \qquad j = 0, 1, ..., N-1$$

interpolante gli N punti di coordinate $(z_k, y_k), k = 0, \ldots, N-1$.

e, analogamente per l'approssimazione nel senso dei minimi quadrati:

Proposizione 5.1.2. *Esiste un unico polinomio trigonometrico del tipo*

$$p(x) = b_0 + b_1 z + b_2 z^2 + \ldots + b_{M-1} z^{M-1}, \quad b_j = \frac{2}{N} \sum_{k=0}^{N-1} y_k e^{-\frac{2\pi}{N} ikj}, \qquad j = 0, 1, ..., M-1$$

con $M < N - 1$, di migliore approssimazione nel senso dei minimi quadrati relativo agli N punti $(z_k, y_k), k = 0, \ldots, N-1$.

♣

5.2 La Trasformata discreta di Fourier (DFT)

Come si è visto anche nell'esempio 5.2, un'applicazione che richiede il calcolo di una o più DFT è la costruzione del polinomio interpolante o approssimante per la descrizione di un insieme di dati con andamento periodico. 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 la funzione f a patto di considerare una opportuna distribuzione dei punti nell'intervallo che si sta considerando[7].

[7]Sussiste, infatti, il seguente (**Teorema di campionamento di Shannon** [26]):

Diamo la seguente:

Definizione 5.2.1. *Si definisce* **Trasformata Discreta di Fourier (DFT)** *di un vettore di numeri complessi* $\underline{f} = (f_0, \ldots, f_{N-1})$ *di lunghezza* N*, e si indica con DFT[f], il vettore di numeri complessi* $\underline{F} = (F_0, \ldots, F_{N-1})$ *ove:*

$$F_k = \sum_{j=0}^{N-1} f_j e^{-\frac{2\pi i}{N} jk} \qquad k = 0, \ldots N-1;\ i = \sqrt{-1} \tag{5.13}$$

Definizione 5.2.2. *Si definisce* **Trasformata Discreta Inversa di Fourier (IDFT)** *di un vettore le cui componenti sono numeri complessi* $\underline{F} = (F_0, \ldots, F_{N-1})$ *di lunghezza* N*, e si indica IDFT[F], il vettore di numeri complessi* $\underline{f} = (f_0, \ldots, f_{N-1})$ *ove:*

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} F_j e^{\frac{2\pi i}{N} jk} \qquad k = 0, N-1; i = \sqrt{-1} \tag{5.14}$$

I settori applicativi in cui si fa uso di DFT sono molteplici. D'altra parte, tale operatore si presenta come strumento fondamentale per la risoluzione di molti problemi della matematica numerica, come ad esempio, nella risoluzione di problemi di calcolo matriciale, se la matrice presenta particolari strutture, nella risoluzione di equazioni

Teorema 5.2.1. *Sia* f *una funzione la cui trasformata di Fourier è nulla al di fuori dell'intervallo* $[-f_{Ny}, f_{Ny}]$*. Siano assegnati i punti* $x_n = nh$*, con:*

$$h \leq 1/(2f_{Ny})$$

allora f *può essere univocamente ricostruita dai suoi campioni:*

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

La quantità f_{Ny} è detta **frequenza di Nyquist**.

La serie al secondo membro è anche nota come *serie cardinale di f*, e la funzione f, somma delle serie cardinale, è detta *funzione cardinale di Witthaker*. Il teorema fu introdotto già da Witthaker nel 1915, e successivamente utilizzato da Shannon nel 1948 [26]. In pratica, nelle applicazioni accade che la trasformata di Fourier di una certa funzione si può considerare trascurabile da un certo intervallo in poi. In tal caso è possibile determinare il passo h in modo da controllare l'errore introdotto dalla sua rappresentazione in termini della *funzione cardinale di Witthaker*.

Se fissiamo l'intervallo $[0, T]$ e $h = \frac{T}{N}$, l'ipotesi che $h \leq 1/(2f_{Ny})$ implica che la massima frequenza di f che può essere calcolata è

$$f_{Ny} = \frac{1}{2T}$$

Il teorema stabilisce quindi che quanto più rapidamente la funzione f *oscilla* nell'intervallo $[0, T]$ (ovvero quanto maggiore è la sua frequenza $\omega = \frac{2\pi}{T}$) tanto più piccola deve essere la distanza dei punti. Inoltre, per ricostruire l'andamento di una funzione che nel suo periodo T ha un unico ciclo ($f_{N_y} = \frac{1}{T}$), dobbiamo conoscerla in almeno due punti ($h < \frac{T}{2}$). In caso contrario, nel ricostruire la funzione f utilizzando lo sviluppo in serie cardinale di Witthaker, le frequenze maggiori della frequenza di Nyquist si sovrappongono alle altre. Tale fenomeno nell'analisi di Fourier prende il nome di **aliasing**.

differenziali ordinarie e alle derivate parziali, nell'integrazione numerica di funzioni periodiche o oscillanti. L'algoritmo FFT è oggi uno dei 10 algoritmi con il maggior impatto scientifico e tecnologico [8].

5.2.1 Alcune proprietà della DFT

Descriviamo alcune delle proprietà più significative della DFT, ovvero consideriamo quelle proprietà che sono alla base delle applicazioni della DFT[9]. Le prime proprietà discendono direttamente dalla definizione dell'operatore DFT e riguardano la linearità, la periodicità e la simmetria. La linearità viene utilizzata specialmente nelle applicazioni in cui il vettore di cui bisogna calcolare la DFT si può decomporre nelle sue componenti armoniche e attraverso l'analisi della DFT di tali componenti si possono trarre informazioni significative sul vettore iniziale (questo è ad esempio il principio alla base dell'analisi armonica di funzioni periodiche). La proprietà di periodicità consente di prolungare *periodicamente* il vettore di cui calcolare la DFT senza alterarne il risultato numerico. Infine la simmetria trova applicazione negli schemi di memorizzazione alla base degli algoritmi per il calcolo di una DFT.

Proposizione 5.2.1. *Siano $f, g \in \mathbb{C}^N$ due vettori di numeri complessi, di lunghezza N e siano $\alpha\ , \beta \in \Re$, si ha:*

$$DFT[\alpha f + \beta g] = \alpha DFT[f] + \beta DFT[g]$$

ovvero la DFT è un operatore lineare.

Dimostrazione Dalla definizione di DFT discende che:

$$DFT[\alpha f + \beta g] = \sum_{j=0}^{N-1} (\alpha f + \beta g)_j \omega_N^{-jk} = \sum_{j=0}^{N-1} \alpha f_j \omega_N^{-jk} + \sum_{j=0}^{N-1} \beta g_j w_N^{jk} =$$

$$= \alpha \sum_{j=0}^{N-1} f_j \omega_N^{-jk} + \beta \sum_{j=0}^{N-1} g_j \omega_N^{-jk} = \alpha DFT[f] + \beta DFT[g].$$

■

[8] IEEE, Computing in Science and Engineering, 2000, n. 22.

[9] A tutte le proprietà dell'operatore DFT corrispondono analoghe proprietà dell'operatore di Trasformata di Fourier; d'altra parte, l'operatore DFT si può anche interpretare come discretizzazione della Trasformata di Fourier ottenuta mediante l'applicazione della formula trapezoidale composita all'integrale di Fourier.

♣ **Esempio 5.3.** Assegnati due vettori:

$$f = (1+i, -3, 5+7i, -2), \quad g = (3+4i, 4, 7, 1-i)$$

calcoliamo la DFT dei vettori f e g:

$$DFT[f] = (1+8i, -4-5i, 11+8i, -4-7i), \quad DFT[g] = (15+3i, -3-i, 5-5i, -5-7i)$$

Poiché $f+g = (4+5i, 1, 12+7i, -1-i)$, la DFT di h è:

$$DFT[h] = DFT[f+g] = (16+11i, -7-4i, 16-13i, -9) =$$

$$= (1+8i, -4-5i, 11+8i, -4-7i) + (15+3i, -3-i, 5-5i, -5-7i) = DFT[f] + DFT[g].$$

♣

Proposizione 5.2.2. *Il vettore $F = DFT[f]$, DFT del vettore di numeri complessi $f_0, \ldots, f_{N-1}$ di lunghezza N è N-periodico, ovvero:*

$$F_{N+k} = F_k, \quad \forall k \in \mathcal{N} \cup \{0\}$$

Dimostrazione Questa proprietà discende direttamente dalla periodicità dell'esponenziale complesso:

$$\omega_N^{-(k+N)n} = \omega_N^{-nk}$$

■

Le proprietà seguenti, che enunciamo solamente, lasciando la dimostrazione per esercizio, hanno interessanti implicazioni in termini di complessità di tempo e di spazio nel calcolo di una DFT di un vettore di numeri reali o di numeri immaginari puri[10].

Proposizione 5.2.3. *Valgono le seguenti proprietà* [11] *:*

1. *Se $f \in \Re^N$ è un vettore di numeri reali, allora $F = DFT[f]$ è un vettore hermitiano simmetrico, ovvero:*
$$F_k = \bar{F}_{N-k} \quad k = 1, ..., N/2$$
avendo indicato con $\bar{F}_{N-k}$ il numero complesso coniugato di F_{N-k}.

2. *Se $f \in \mathbb{C}^N$ è un vettore hermitiano simmetrico, ovvero $f_k = \bar{f}_{N-k}$, allora $F = DFT[f]$ è un vettore di numeri reali.*

[10] Nel seguito, all'occorrenza, il vettore DFT si intende prolungato per periodicità

[11] Nel seguito si farà riferimento al numero intero $N/2$, volendo intendere, nel caso in cui N sia dispari, alla parte intera di $N/2$ aumentata di un'unità.

3. *Se $g \in \mathbb{C}^N$ è un vettore le cui componenti sono numeri immaginari allora $G = DFT[g]$ è un vettore hermitiano antisimmetrico, ovvero:*

$$G_k = -\bar{G}_{N-k} \quad k = 1, ..., N/2$$

4. *Se $g \in \mathbb{C}^N$ è un vettore hermitiano antisimmetrico, ovvero $g_k = \bar{g}_{N-k}$, allora $G = DFT[g]$ è un vettore le cui componenti sono numeri immaginari.*

♣ **Esempio 5.4.** Calcoliamo la DFT del vettore $f = (1, 2, 3, 4, 5, 6)$ e del vettore $g = (i, -2i, 5i, -4i, 3i, 6i)$:

$$F = DFT[f] = (21, -3 - 5.1962\,i, -3 - 1.7321\,i, -3, -3 + 1.7321\,i, -3 + 5.1962\,i)$$

$$G = DFT[g] = (9\,i, -5.1962 + 3\,i, -8.6603 + 9\,i, -9i, 8.6603i + 9\,i, 5.1962 + 3\,i)$$

F è un vettore hermitiano simmetrico e G è hermitiano antisimmetrico. ♣

♣ **Esempio 5.5.** Calcoliamo la DFT di due vettori reali $x, y \in \Re^N$.
Costruiamo il vettore di numeri complessi $z = x + iy$, e consideriamo il vettore $Z = DFT[z]$. Poiché l'operatore DFT è lineare, segue che $Z = DFT[z] = DFT[x] + iDFT[y]$. Sia $X = DFT[x]$ e $Y = DFT[y]$, applicando la proprietà 1 della proposizione 5.2.3, risulta

$$X_k = \bar{X}_{N-k}, \quad \bar{X}_k = X_{N-k} \quad k = 1, \ldots, N/2$$

e

$$Y_k = \bar{Y}_{N-k}, \quad \bar{Y}_k = Y_{N-k} \quad k = 1, \ldots, N/2$$

Dall'uguaglianza:

$$Z_k = X_k + iY_k \ , \quad k = 0, \ldots, N-1 \tag{5.15}$$

sostituendo k con $N - k$ e passando ai coniugati segue:

$$\bar{Z}_{N-k} = X_k - iY_k \quad k = 0, \ldots, N-1 \tag{5.16}$$

Addizionando e sottraendo la (5.15) e la (5.16), si ha:

$$X_k = 1/2[\bar{Z}_{N-k} + Z_k], \quad Y_k = i/2[\bar{Z}_{N-k} - Z_k], \quad k = 1, \ldots, N/2$$

ovvero, per ottenere i due vettori X e Y, DFT rispettivamente dei vettori x e y, invece di calcolare due DFT di due vettori di numeri reali, basta calcolare una sola DFT di un vettore di numeri complessi. Inoltre, poiché si tratta di vettori hermitiani simmetrici, si ha anche un risparmio di occupazione di memoria.
Sia ad esempio, $N = 5$ e:

$$x = (5, 2, 1, -1, 3), \quad y(-2, 4, 1, 7, 3)\,;$$

costruiamo il vettore

$$z = x + iy = (5 - 2\,i, 2 + 4\,i, 1 + i, -1 + 7\,i, 3 + 3\,i)\ ;$$

calcoliamo la DFT di z:

$$Z = DFT[z] = (10 + 13\,i, 3.9694 - 6.5335\,i, 7.249 - 2.7011\,i, -5.3392 - 7.6809\,i, 9.1207 - 6.0845i)$$

da cui, applicando le relazioni

$$X_k = 1/2[\bar{Z}_{N-k} + Z_k], \quad Y_k = i/2[\bar{Z}_{N-k} - Z_k]\ , \quad k = 1, \ldots, N/2 \quad (X_0 = Re(Z_0),\, Y_0 = Im(Z_0))$$

si ottiene il vettore DFT di x e y. ♣

♣ **Esempio 5.6.** Calcoliamo la DFT di un vettore $x \in \Re^{2N}$.
Costruiamo i vettori $h = (h_j)_{j=0,\ldots,N-1}$ e $g = (g_j)_{j=0,\ldots,N-1}$ dove

$$h_j = f_{2j} \quad , \quad g_j = f_{2j+1}$$

e

$$z = h + ig$$

Siano $Z = DFT[z]$, $H = DFT[h]$ e $G = DFT[g]$. Osserviamo innanzitutto che, dalla definizione di DFT segue:

$$X_k = DFT[x]_k = DFT[h]_k + e^{-\frac{i\pi}{N}k} DFT[g]_k \quad , k = 0, \ldots, N-1$$

inoltre, poiché il vettore x è reale, il vettore $X = DFT[x]$ è hermitiano simmetrico cioè $X_{2N-k} = \bar{X}_k$. Ció significa che per calcolare la DFT del vettore x basta calcolare le sue prime N componenti. Il problema è stato quindi ricondotto al calcolo delle DFT dei due vettori di numeri reali, h e g, di lunghezza N. Come abbiamo già visto nell'esempio 5.5, questo si puó ottenere effettuando il calcolo della DFT del vettore z. In sintesi, abbiamo calcolato la DFT di x, vettore di numeri reali di lunghezza $2N$, calcolando la DFT di z, vettore di numeri complessi di lunghezza N.

Sia, ad esempio, $N = 3$ e consideriamo il vettore

$$x = (2, -5, 8, -3, -2, 4)$$

di lunghezza $2N = 6$. Siano $h = (2, 8, -2)$ e $g = (-5, -3, 4)$. Costruiamo il vettore

$$z = (2 - 5i, 8 - 3i, -2 + 4i)$$

e calcoliamo la sua DFT:

$$Z = (8 - 4i, -7.06 - 14.16i, 5.06 + 3.16i)$$

da cui si ricava che

$$DFT[h] = (8, -1 - 8.6i, -1 + 8.6i), \quad DFT[g] = (-4, -5.5 + 6.06i, -5.5 - 6.06i)$$

e, infine, ricaviamo le componenti di $X = DFT[x]$:

$$X = (4, 1.5 - 0.8i, -3.5 + 16.4i, 12, -3.5 - 16.4i, 1.5 + 0.8i)$$

♣

Sia:

$$W = \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ \omega_N^0 & \omega_N^{-2} & \omega_N^{-4} & \dots & w_N^{-2(N-1)} \\ \dots & \dots & \dots & \dots & \dots \\ \omega_N^0 & \omega_N^{-(N-1)} & \dots & \dots & \omega_N^{-(N-1)(N-1)} \end{pmatrix}$$

La matrice W è detta **matrice di Fourier**.

Proposizione 5.2.4. *La matrice $\widetilde{W} = \frac{1}{\sqrt{N}} W$ è unitaria , cioè $\widetilde{W} \cdot \widetilde{W}^H = I$, dove $\widetilde{W}^H$ è la matrice trasposta della matrice coniugata di $\widetilde{W}$.*

Dimostrazione

$$\begin{aligned} \tilde{W} \cdot \tilde{W}^T &= \frac{1}{N} \cdot \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ \omega_N^0 & \omega_N^{-2} & \omega_N^{-4} & \dots & \omega^{-2(N-1)} \\ \dots & \dots & \dots & \dots & \dots \\ \omega_N^0 & \omega_N^{-(N-1)} & \dots & \dots & \omega_N^{-(N-1)^2} \end{pmatrix} \cdot \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ \omega_N^0 & \omega_N^{-2} & \omega_N^{-4} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \omega_N^0 & \omega_N^{-(N-1)} & \omega_N^{-2(N-1)} & \dots & \omega_N^{-(N-1)^2} \end{pmatrix} \\ &= \frac{1}{N} \cdot \begin{pmatrix} \sum_{i=0}^{N-1} \omega_N^0 \cdot \omega_N^0 & \sum_{i=0}^{N-1} \omega_N^0 \cdot \omega_N^{-i} & \dots & \sum_{i=0}^{N-1} \omega_N^0 \cdot w^{-(N-1)i} \\ \sum_{i=0}^{N-1} \omega_N^0 \cdot w^{-i} & \sum_{i=0}^{N-1} \omega_N^{-i} \cdot w^{-i} & \dots & \sum_{i=0}^{N-1} \omega_N^{-i} \cdot w^{-(N-1)i} \\ \dots & \dots & \dots & \dots \\ \sum_{i=0}^{N-1} \omega_N^0 \cdot w^{-(N-1)i} & \sum_{i=0}^{N-1} \omega_N^{-i} \cdot w^{-(N-1)i} & \dots & \sum_{i=0}^{N-1} \omega_N^{-(N-1)i} \cdot w^{-(N-1)i} \end{pmatrix} \\ &= \frac{1}{N} \cdot \begin{pmatrix} N & 0 & 0 & \dots & 0 \\ 0 & N & 0 & \dots & 0 \\ 0 & 0 & N & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \end{aligned}$$

■

Il teorema che segue è alla base di tutte le applicazioni della DFT dove si fa uso dell'operatore di convoluzione e dell'operatore inverso di deconvoluzione. In breve, l'-operazione di convoluzione [12] tra due vettori a e b, trasforma il vettore a in un altro in cui ciascuna componente è ottenuta come media pesata della rispettiva componente e di quelle consecutive, laddove il peso è dato dalle componenti del vettore b. Tale operazione

12

Definizione 5.2.3. ***(Prodotto di Convoluzione)***
Dati due vettori $a = (a_0, a_1, ..., a_{N-1})$ e $b = (b_0, b_1, ..., b_{N-1})$ si definisce prodotto di convoluzione il vettore:

$$c = a * b$$

si usa spesso nell'elaborazione di dati che descrivono fenomeni con andamento periodico con l'obiettivo di ridurre *picchi* e brusche variazioni nel loro andamento che spesso denotano la presenza di *rumore*. L'operazione di convoluzione si esprime in termini matriciali come prodotto di una matrice *circolante* [13] per un vettore. Per tale motivo, utilizzando opportunamente la proprietà dell'operatore DFT nel calcolo del prodotto di convoluzione, si derivano in maniera naturale gli algoritmi basati sulla DFT per il calcolo con matrici circolanti.

Teorema 5.2.2. [di Convoluzione]
Siano $F = DFT[f]$ e $G = DFT[g]$ le DFT di due vettori f e g di lunghezza N. La DFT del prodotto di convoluzione tra f e g, $f * g$, è il vettore ottenuto come prodotto puntuale dei due vettori, ovvero G è il vettore le cui componenti sono ottenute effettuando il prodotto componente per componente dei vettori F e G:

$$DFT[f * g] = (F_0G_0, F_1G_1, ..., F_{N-1}G_{N-1}) = F. * G$$

avendo indicato con $.*$ il prodotto puntuale dei vettori F e G.

♣ **Esempio 5.7.** Consideriamo il vettore $\underline{f} = \{f_n\}_{n=0,...,23}$, di $N = 24$ punti ottenuti valutando la funzione

$$f(x) = \cos(2\pi x) + 1/2\cos(10\pi x) + 1/3\cos(12\pi x)$$

in $x_n = n \cdot \Delta x, \Delta x = 1/24$, $n = 0, \ldots, 23$. Assumiamo, inoltre, che tale vettore sia 24-periodico, ovvero $f_{n\pm 24} = f_n$. Consideriamo il vettore $\{g_n\}_{n=0,...,23}$, con

$$g_n = \frac{1}{8}f_{n-2} + \frac{1}{4}f_{n-1} + \frac{1}{4}f_n + \frac{1}{4}f_{n+1} + \frac{1}{8}f_{n+2}$$

Introdotto il vettore

$$\underline{h} = (\ldots, 0, 0, 0, 0, 0, 0, 0, 0, 0, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \ldots)$$

---

la cui $j-esima$ componente è data da:

$$c_j = \sum_{k=0}^{j} a_k \cdot b_{j-k} \quad j = 0, ..., 2N-2.$$

[13]

Definizione 5.2.4. (Matrice Circolante)
Assegnato un vettore $v = (v_0, v_1, ..., v_{N-1})$ si definisce **matrice circolante** *di lunghezza N, generata dal vettore v, una matrice $C := C(v)$*

$$C := C(v) = \begin{bmatrix} v_0 & v_1 & \cdots & v_{N-2} & v_{N-1} \\ v_{N-1} & v_0 & \cdots & v_{N-3} & v_{N-2} \\ v_{N-2} & v_{N-1} & \cdots & v_{N-4} & v_{N-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ v_1 & v_2 & \cdots & v_{N-1} & v_0 \end{bmatrix}.$$

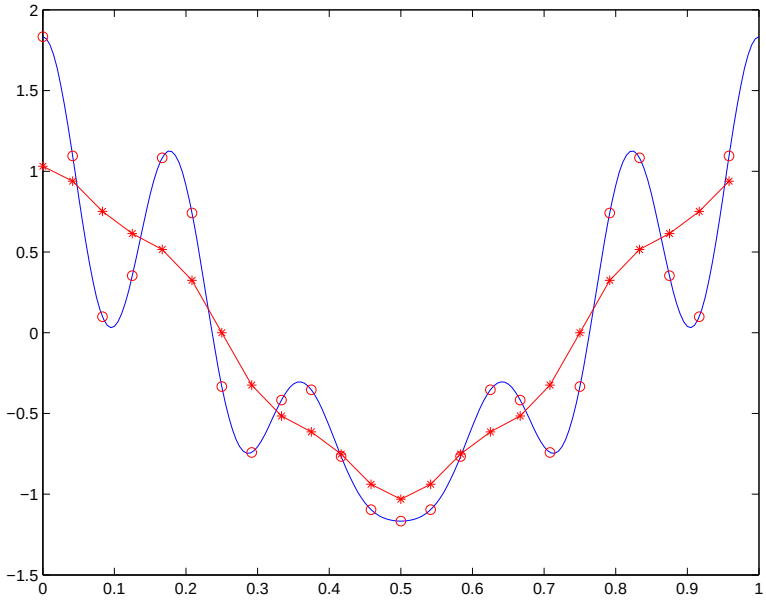

Figura 5.3: Grafico di $f(x) = cos(2\pi x) + \frac{1}{2}cos(10\pi x) + \frac{1}{3}cos(12\pi x)$ in $[0, 1)$. Con il simbolo 'o' si indicano i valori $f_n = f(x_n)$ e con il simbolo '*' si indicano i valori g_n, $n = 0, \ldots, 23$.

24-periodico, con $h_{\pm 2} = 1/8$, $h_{\pm 1} = 1/4$, $h_0 = 1/4$, si ha

$$g_n = \sum_{j=0}^{n} f_j \cdot h_{n-j} = \underline{f} * \underline{h}$$

In altre parole, il vettore $\{g_n\}$ è il prodotto di convoluzione del vettore $\{f_n\}$ con il vettore $\{h_n\}$. Come anche illustrato in Fig.5.3, il vettore $\{f_n\}$ è costituito dai valori assunti dalla funzione f, composta da tre funzioni cosinusoidali con frequenza rispettivamente $1, 5$ e 6; congiungendo a due a due i punti di coordinate (x_n, g_n) si ottiene una curva con meno picchi rispetto al grafico della funzione f e con un andamento molto più *dolce*. Il procedimento appena descritto è un esempio dell'operazione che trasforma un vettore, le cui componenti hanno valori che differiscono tra loro in maniera significativa, in un altro in cui picchi e brusche variazioni vengono smorzati.

♣

♣ **Esempio 5.8.** Consideriamo la matrice circolante C generata dal vettore $v = (5, 2, 7, 9, 4)$:

$$C = \begin{bmatrix} 5 & 2 & 7 & 9 & 4 \\ 4 & 5 & 2 & 7 & 9 \\ 9 & 4 & 5 & 2 & 7 \\ 7 & 9 & 4 & 5 & 2 \\ 2 & 7 & 9 & 4 & 5 \end{bmatrix}.$$

Si vuole utilizzare la DFT per il calcolo del prodotto matrice per vettore

$$C \cdot x' = y'$$

dove $x = (1, 2, 3, 4, 5)$ e $y = (86, 93, 75, 67, 84)$.

Applicando la proprietà delle matrici circolanti:

$$C = W^{-1}DW$$

dove W è la matrice di Fourier e $D = diag(F)$, con $F = DFT[v]$, si ha:

$$Cx' = y' \quad \Leftrightarrow \quad W^{-1}DWx' = y' \quad \Leftrightarrow \quad DWx' = Wy'$$

e

$$Wx' = DFT[x'], \quad Wy' = DFT[y'],$$

posto $X = DFT[x']$, $Y = DFT[y']$ e $F = DFT[v]$ si ha

$$Cx' = y' \quad \Leftrightarrow \quad F. * X' = Y'$$

avendo indicato con $.*$ il prodotto puntuale [14] dei vettori F e X'. Essendo

$$F = (27., -6.0902 + 3.0777i, 5.0902 - 0.7265i, 5.0902 + 0.7265i, -6.0902 - 3.0777i)$$

e

$$X' = (15, -2.5 - 3.441i, -2.5 - 0.8123i, -2.5 + 0.8123i, -2.5 + 3.441i)$$

effettuando il prodotto puntuale $F. * X'$ si ha:

$$Y' = (405, 25.82 + 13.26i, -13.32 - 2.32i, -13.32 + 2.32i, 25.82 - 13.26i)$$

ed applicando la DFT inversa di Y' si ottiene il vettore y':

$$y' = IDFT[Y] = \begin{pmatrix} 86 \\ 93 \\ 75 \\ 67 \\ 84 \end{pmatrix}.$$

♣

Consideriamo una matrice circolante C, generata da un vettore v: $C = C(v)$. Per il calcolo del prodotto matrice per vettore:

$$C \cdot x' = y'$$

si ha:

$$C \cdot x' = y' \Leftrightarrow W^{-1}DW \cdot x' = y' \Leftrightarrow DW \cdot x' = Wy' \tag{5.17}$$

[14] Il prodotto puntuale tra due vettori, a e b, è il vettore le cui componenti sono ottenute moltiplicando le componenti omologhe dei due vettori a e b.

Poiché:

$$Wx' = DFT[x'] \quad \text{e} \quad Wy' = DFT[y'],$$

posto $X = DFT[x']$ e $Y = DFT[y']$ la (5.17) diventa:

$$C \cdot x' = y' \Leftrightarrow F. * X' = Y' \tag{5.18}$$

avendo indicato con $.*$ il prodotto puntuale dei vettori F e X'. Quindi per effettuare il prodotto $Cx' = y'$ si può:

1. calcolare $X = DFT[x']$;
2. calcolare $F = DFT[v]$;
3. calcolare il prodotto puntuale $F. * X' = Y'$;
4. calcolare la DFT inversa $IDFT[Y] = y'$;

e ciò richiede il calcolo di 2 DFT e 1 IDFT.
Analogamente deriva lo schema relativo alla risoluzione di un sistema lineare avente come matrice dei coefficienti una matrice circolante.

♣ **Esempio 5.9.** Consideriamo la seguente matrice:

$$T = \begin{bmatrix} 5 & 2 & 7 \\ 4 & 5 & 2 \\ 9 & 4 & 5 \end{bmatrix}.$$

La matrice T è una matrice di *Toeplitz*[15]. Notiamo che T è una sottomatrice della matrice circolante C:

$$C = \begin{bmatrix} 5 & 2 & 7 & 9 & 4 \\ 4 & 5 & 2 & 7 & 9 \\ 9 & 4 & 5 & 2 & 7 \\ 7 & 9 & 4 & 5 & 2 \\ 2 & 7 & 9 & 4 & 5 \end{bmatrix}$$

introdotta nell'esempio 5.8. Pertanto, viene naturale domandarsi se, anche in questo caso, analogamente a quanto fatto per le matrici circolanti, è possibile utilizzare la DFT per eseguire le operazioni di calcolo matriciale involventi T. Considerato allora il vettore

$$x = (1, 2, 3)\,,$$

[15]

Definizione 5.2.5. *La matrice $T := (t_{ij})$, $i, j = 1, \ldots, n$, è una matrice di Toeplitz se ha elementi costanti lungo ciascuna diagonale, ovvero*

$$t_{ij} = r_{j-i}, \quad \forall i, j = 1, \ldots, n$$

con $r_{-n+1}, \ldots, r_0, \ldots, r_{n-1}$ numeri reali.

calcoliamo ad esempio il vettore y, prodotto di T per x, $T \cdot x = y$. In particolare, si ha $y = (86, 93, 75)$. Si osserva che se introduciamo il vettore $x' = (1, 2, 3, 0, 0)$, il prodotto $C \cdot x'$ fornisce come vettore risultante il vettore y' le cui prime tre componenti coincidono con le componenti del vettore $y = T \cdot x$. Per calcolare y basta calcolare $y' = C \cdot x'$, utilizzando, quindi, la DFT. ♣

In generale, una qualsiasi matrice di Toeplitz $T \in \Re^{n \times n}$, si può *immergere* in una matrice circolante, ovvero T si può riguardare come sottomatrice di una matrice circolante $C \in \Re^{(2n-1)\times(2n-1)}$, ottenuta considerando come prima colonna di C il vettore v di lunghezza $2n-1$ le cui prime n componenti coincidono con le n componenti del vettore relativo alla prima colonna di T, e le ultime $n-1$ componenti sono le componenti del vettore relativo alla prima riga di T, prese nell'ordine inverso. Quanto detto si può esprimere sinteticamente, utilizzando ad esempio la notazione `MATLAB`, come:

$$v = (T(1:n, 1), T(1, n:-1:2))$$

Di conseguenza, per le operazioni di calcolo matriciale involventi una matrice di Toeplitz, di dimensione n, come ad esempio il prodotto matrice per vettore $T \cdot x$, si può utilizzare la DFT a patto di:

- **immergere** la matrice di Toeplitz di dimensione n in una matrice circolante C di dimensione $2n-1$;
- **prolungare** il vettore x aggiungendo $n-1$ componenti nulle;
- calcolare il prodotto $C \cdot x = y$.

♣ **Esempio 5.10.** Consideriamo la matrice tridiagonale a blocchi così definita:

$$M = \begin{bmatrix} T & -I & & & & \\ -I & T & -I & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & -I \\ & & & & -I & T \end{bmatrix}$$

di dimensione $N \times N$, dove $N = n \times n$, I è la matrice identica di dimensione $n \times n$ e T è la matrice tridiagonale di dimensione $n \times n$:

$$T = \begin{bmatrix} 4 & -1 & & & & \\ -1 & 4 & -1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & -1 \\ & & & & -1 & 4 \end{bmatrix}$$

Consideriamo il problema della risoluzione del sistema lineare [16]:

$$Mu = b, \quad M \in \Re^{N \times N}$$

La matrice M è anche simmetrica e a diagonale dominante. La matrice tridiagonale T è anch'essa simmetrica e a diagonale dominante. Inoltre, M è *semplice* e i suoi autovalori sono noti e sono esprimibili come funzioni trigonometriche. In particolare, esiste una matrice ortogonale Q tale che:

$$QMQ^T = S \Leftrightarrow M = Q^T SQ$$

dove:

$$S = diag(\lambda_{jk}), \quad \lambda_{jk} = 4 - 2\cos\left(\frac{j\pi}{n+1}\right) - 2\cos\left(\frac{k\pi}{n+1}\right)$$

e Q è la matrice ortogonale definita come;

$$Q = diag(F_n^T)\ P\ diag(F_n^T) \in \Re^{N \times N}$$

dove:

$$F_n = (f_{ij}), \quad f_{ij} = \sqrt{\frac{2}{\pi}} \sin\frac{\pi ij}{n+1} \quad i, j = 1, \ldots, n$$

e P è una matrice di permutazione.
Se per il calcolo di $u = M^{-1}b$ sostituiamo a M l'espressione $M = Q^T SQ$, si ha:

$$u = Q^T S^{-1} Qb$$

ovvero u si può ottenere:

1. calcolando $b_1 = Qb$,
2. calcolando $b_2 = S^{-1}b_1$,
3. calcolando $b_3 = Q^T b_2$.

Questo procedimento può risultare particolarmente efficiente se si tiene conto che in ciascun passo possiamo utilizzare la DFT, e quindi l'algoritmo FFT. Osserviamo, infatti, che poiché:

$$\sin\left(\frac{\pi kj}{N}\right) = \frac{e^{\frac{\pi kj}{N}} - e^{\frac{-\pi kj}{N}}}{2i} = \frac{w_{N'}^{kj/2} - w_{N'}^{-kj/2}}{2i}$$

dove

$$w = e_{N'}^{i2\pi/N'}, \quad N' = 2N,$$

segue:

$$-2i \sum_{j=1}^{N-1} a_j \sin\left(\frac{\pi kj}{N}\right) = \sum_{j=1}^{N-1} a_j w_{N'}^{-kj} - \sum_{j=1}^{N-1} a_j w_{N'}^{kj}$$

Posto $r = N' - j$ si ha:

$$w^{kj} = w_{N'}^{kN'-kr}$$

[16] Come già visto nell'esempio 4.81 del **Capitolo 4**, questa matrice si ottiene, ad esempio, considerando l'equazione di Poisson, ovvero l'equazione differenziale alle derivate parziali di tipo ellittico:

$$\frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} = f(x, y)$$

su un dominio quadrato $\Omega = (0, L) \times (0, L)$, con condizioni di Dirichlet sulla frontiera $u(x, y) = g(x, y), (x, y) \in \partial\Omega$ e per la sua risoluzione numerica discretizziamo le derivate parziali all'interno del dominio Ω con uno schema alle differenze finite su 5 punti.

da cui:

$$\sum_{j=1}^{N-1} \sin\left(\frac{\pi kj}{N}\right) = \sum_{j=0}^{N'-1} c_j w_{N'}^{-kj}$$

dove

$$c_j = \begin{cases} 0 & j = 0, j = N \\ ia_j & 1 \leq j \leq N-1 \\ -ia_{N'-j} & N+1 \leq j \leq N'-1 \end{cases}$$

In altre parole, il vettore

$$s_k = 2\sum_{j=1}^{N-1} a_j \sin\left(\frac{\pi kj}{N}\right), \quad k = 1, 2, \ldots, N-1$$

che definisce la trasformata discreta di seni del vettore $\{a_j\}_{j=1,2,\ldots,N-1}$ può essere ottenuto calcolando la DFT del vettore $\{c_j\}$, di lunghezza $2N$. Analogo discorso vale per il calcolo della trasformata discreta di soli coseni. Questo significa, ad esempio, che per calcolare il vettore b_1 al primo passo è necessario calcolare $2n$ DFT, analogamente per ottenere b_3 al terzo passo è necessario effettuare $2n$ DFT per un totale complessivo di $4n$ DFT di lunghezza $2N$. ♣

Un'altra applicazione della DFT che fa uso del prodotto di convoluzione riguarda il calcolo dei coefficienti del polinomio prodotto di due polinomi. Se si tiene conto della rappresentazione dei numeri nel sistema di numerazione posizionale (come quello decimale o binario) questa applicazione ha una interessante implicazione negli algoritmi numerici che fanno uso della multiprecisione [17].

♣ **Esempio 5.11.** [**Moltiplicazione di due polinomi**] Siano assegnati due polinomi:

$$p(x) = 1 + 5x + 17x^2, \qquad q(x) = 11 + 6x - 4x^2$$

Definiti i vettori $a, b \in \Re^5$:

$$a = (1, 5, 17, 0, 0) \quad \text{e} \quad b = (11, 6, -4, 0, 0)$$

che contengono nelle prime 3 posizioni i coefficienti di p e q rispettivamente, i coefficienti del polinomio z di quarto grado prodotto di p e di q si ottengono effettuando il prodotto di convoluzione dei vettori a e b, ovvero:

$$c = a * b = (11, 61, 213, 82, -68, 0, 0, 0, 0)$$

Il polinomio z è:

$$z(x) = p(x)q(x) = 11 + 61x + 213x^2 + 82x^3 - 68x^4$$

Per determinare i coefficienti di z utilizziamo l'operatore DFT. Posto:

$$A = DFT[a] = (23, -11.2082 - 14.7476i, 2.082 + 13.229i, 2.2082 - 13.229i, -11.2082 + 14.7476i)$$

$$B = DFT[b] = (13, 16.0902 - 3.3552i, 4.9098 - 7.3309i, 4.9098 + 7.3309i, +16.0902 + 3.3552i)$$

[17] [1]

utilizzando il **Teorema 5.2.2**:

$$C = DFT(c) = DFT[a * b] = A. * B =$$

$$= (299, -229.82 - 199.69i, 107.82 + 48.76i, 107.82 - 48.76i, -229.82 + 199.69i)$$

da cui:

$$c := IDFT(C) = (11, 61, 213, 82, -68)$$

Il calcolo dei coefficienti del prodotto di due polinomi si riduce, quindi, al calcolo di 3 DFT. ♣

Assegnati due polinomi

$$p(x) = \sum_{i=0}^{r} a_i x^i \ \in \Pi_r \,, \qquad q(x) = \sum_{j=0}^{s} b_j x^j \ \in \Pi_s$$

indicato con:

$$z(x) = p(x)q(x) = \sum_{k=0}^{r+s} c_k x^k \quad \in \Pi_{r+s} \tag{5.19}$$

il polinomio prodotto, i coefficienti di z:

$$\begin{cases} c_0 = a_0 \cdot b_0 \\ c_1 = a_0 b_1 + a_1 b_0 \\ \vdots \\ c_k = \sum_{h=0}^{k} a_h b_{k-h} \\ \vdots \end{cases}$$

si possono ottenere effettuando il prodotto di convoluzione tra il vettori a e b:

$$c = a * b$$

avendo indicato con $a = (a_0, a_1, ...a_{r+s})$ il vettore contenente nelle prime $r+1$ posizioni i coefficienti del polinomio p e nelle restanti s posizioni lo zero e analogamente con $b = (b_0, b_1, ...b_{r+s})$ il vettore contenente nelle prime $s+1$ posizioni i coefficienti del polinomio q e nelle restanti r posizioni lo zero, e con $c = (c_0, c_1, ...c_{r+s})$ il vettore contenente i coefficienti del polinomio prodotto z.

Applicando il **Teorema 5.2.2**, si puó ottenere il vettore c utilizzando la DFT. In particolare, per calcolare il vettore c bisogna:

1. calcolare la DFT dei coeffienti del polinomio p:

$$A = DFT[a]$$

2. calcolare la DFT dei coeffienti del polinomio q:

$$B = DFT[b]$$

3. effettuare il prodotto componente per componente di A e di B:

$$A. * B = C$$

4. calcolare i coefficienti di z mediante la DFT inversa:

$$c = IDFT[C]$$

5.3 La DFT come approssimazione della Trasformata di Fourier (FT)

Consideriamo il seguente integrale:

$$\int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt, \quad i = \sqrt{-1}, \tag{5.20}$$

dove $f : \Re \to \Re$ è una funzione per cui tale integrale esiste ed è finito [18]. Al variare di ω l'integrale in (5.20) definisce una funzione

$$F : \omega \to \int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt$$

a cui si dà il nome di **Trasformata di Fourier (FT)** della funzione f. Sia $T \in \Re - \{\infty\}$, si ha:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt = \sum_{k=-\infty}^{+\infty} \int_{kT}^{(k+1)T} f(t)e^{-2\pi i\omega t}dt =$$

$$= \sum_{k=-\infty}^{+\infty} \int_{0}^{T} f(t+kT)e^{-2\pi i\omega(t+kT)}dt = \int_{0}^{T} \sum_{k=-\infty}^{+\infty} f(t+kT)e^{-2\pi i\omega(t+kT)}dt$$

Se introduciamo la funzione f_p, periodica di periodo T, definita come:

$$f_p(t) = \sum_{k=-\infty}^{+\infty} f(t+kT), \quad t \geq 0$$

segue che:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt = \int_{0}^{T} f_p(t)e^{-2\pi i\omega t}dt$$

[18] Ad esempio, tale è una funzione assolutamente integrabile su tutto l'asse reale che in ogni intervallo finito ha al più un numero finito di discontinuità di prima specie ed è a variazione limitata. Una siffatta funzione si dice che soddisfa le *condizioni di Dirichlet*.

ovvero la Trasformata di Fourier di f coincide con la trasformata di Fourier di f_p. Supponiamo di voler valutare la funzione $F(\omega)$ per $\omega = \omega_n = n/T$, $n = 0, 1, 2, \ldots$:

$$F(\omega_n) = \int_0^T f_p e^{-2\pi i \omega_n t} dt$$

e discretizziamo l'integrale con la formula di quadratura trapezoidale composita su N punti. Posto $h = \frac{T}{N}$ si ottiene:

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

Considerati i vettori $F = \{F(\omega_n)\}_{n=0,1,\ldots,N-1}$, e $f_p = \{f_p(kh)\}_{k=0,1,\ldots,N-1}$, segue che

$$F = DFT[f_p]$$

ovvero, la trasformata discreta di Fourier, 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 $f_p(v) = f(v)$, $v \in [0, T]$ e la $DFT[f_p](= DFT[f])$ **fornisce un'approssimazione della trasformata di Fourier di** f. In altre parole la FT e la DFT nei punti ω_n coincidono a meno di un errore. L'errore dipende **dall'errore di discretizzazione**, introdotto dalla 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 abbiamo già notato (vedi **Capitolo 2**, **§2.6)**, l'ordine di convergenza della formula di quadratura trapezoidale dipende dalla regolarità della funzione f. In particolare, a partire dalla stima dell'errore di discretizzazione delle formule di quadratura di Newton-Cotes, che garantisce che se la $f \in C^2([0, T])$ l'errore ha ordine 2, si possono ricavare stime per l'errore di discretizzazione che mostrano un'accuratezza di ordine sempre più alto quanto più la funzione è regolare. Ad esempio, se $f \in C^{2r+1}, r \geq 1$, utilizzando lo sviluppo dell'errore fornito dalla somma di Eulero McLaurin, l'errore di discretizzazione ha ordine $2r + 1$.

Teorema 5.3.1. *[6, 18]*
Fissato $r \geq 1$, sia $f \in C^{2r+1}([0, a])$, integrabile in $[0, +\infty]$ e sia:

$$M = \int_0^\infty |f^{(2r+1)}(x)| dx < \infty$$

Supponiamo inoltre che:

$$f^{(2k-1)}(0) = 0, \quad \lim_{x \to \infty} f^{(2k-1)}(x) = 0, \quad k = 1, \ldots, r$$

Si ha allora, per ogni fissato $h > 0$, per l'errore di discretizzazione E:

$$|E| \leq h^{2r+1} \frac{M \zeta(2r+1)}{2^{2r} \pi^{2r+1}}$$

dove ζ è la funzione zeta di Riemann.

In particolare, poi, se $f \in C^\infty$ e tutte le derivate di ordine dispari tendono a zero agli estremi dell'intervallo di integrazione, l'errore di discretizzazione tende a zero più velocemente di qualsiasi potenza di h. Infine, il teorema seguente riporta una maggiorazione dell'errore di discretizzazione di più semplice utilizzo negli algoritmi numerici, applicabile nel caso in cui la funzione f è la restrizione sull'asse reale di una funzione analitica:

Teorema 5.3.2. *[6, 18]*
Fissato $q \geq 0$. Sia G_q l'insieme delle funzioni g tali che $g(x+iy)$ sia analitica nella striscia $|y| \leq q$ e tali che

$$|g(x+iy)| \leq \frac{A}{|x|^\beta}, \quad |x| > \delta$$

con $A > 0$, $\delta > 0$ e $\beta > 1$. Se $f \in G_q$ e $\omega \in]0, \pi/h[$ allora:

$$E \leq \frac{2e^{-q\omega(p-1)}}{1 - e^{-qp\omega}} M(q)$$

dove $p = \frac{2\pi}{h\omega} > 2$ e

$$M(q) = \max_{y=\pm q} \int_{-\infty}^{\infty} |f(t+iy)| dt$$

Per quanto riguarda l'errore di troncamento, ovvero l'errore introdotto dall'aver assunto che la funzione f sia nulla al di fuori dell'intervallo $[0, T]$, fissato h, tale quantità decresce esponenzialmente al crescere di N (e quindi di T) [19]. In generale, in base a tali considerazioni è possibile una scelta opportuna dei parametri N e h in modo da garantire dei risultati accettabili con un costo computazionale non elevato.

5.4 La Trasformata Veloce di Fourier (FFT)

Dalla definizione di DFT di un vettore di N numeri complessi si deduce che la valutazione diretta di ciascuna componente del vettore DFT richiede il calcolo di N moltiplicazioni complesse e $N-1$ addizioni complesse, ed inoltre la valutazione di N esponenziali complessi. *Supponendo noti gli esponenziali complessi*, il calcolo della DFT di un vettore di lunghezza N richiede, quindi:

$$T(N) = N \cdot [N \quad \text{molt. complesse} + (N-1) \quad \text{add. complesse}] = \mathcal{O}(N^2)$$

operazioni floating point. 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.
Fino al 1965, anno di pubblicazione del primo algoritmo di *Fast Fourier Transform* (FFT) nonostante la già ampia diffusione della DFT, la sua applicazione presentava

[19] [4]

problemi legati al costo computazionale. L'idea di J. Cooley e J. Tukey, alla base degli algoritmi FFT, 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 una buona parte di operazioni inutili e ridondanti. Un algoritmo FFT implementa una metodologia di tipo *divide et impera*, applicandola più volte a ciascuno dei sottoproblemi così ottenuti, in modo da riorganizzare efficacemente le operazioni che coinvolgono gli esponenziali complessi.

Riguardo la valutazione degli esponenziali complessi, la difficoltà principale di un algoritmo per il calcolo di una DFT, e più in generale di tutti gli algoritmi numerici utilizzati nell'analisi di Fourier, è sempre stata la valutazione efficiente delle funzioni trigonometriche, indipendentemente dall'ambiente di calcolo a disposizione. Già nel 1925, Witthaker, avendo a disposizione le macchine calcolatrici per eseguire moltiplicazioni, oltre che le tavole trigonometriche, utilizzando la metodologia *divide et impera*, propose uno schema per il calcolo dei coefficienti di Fourier per $N = 6, 12, 24$, che faceva uso solo delle valutazioni negli angoli notevoli. Tale algoritmo, ad esempio per $N = 12$, impiegava solo 44 operazioni. Nel 1958, Goertzel propose un algoritmo basato su formule di ricorrenza per il calcolo di coefficienti di Fourier per un qualsiasi valore di N che utilizzava solo 2 valutazioni trigonometriche, successivamente reso stabile dalla modifica introdotta da Reinsch [20]. Chiaramente questo algoritmo, a differenza dell'algoritmo poposto da Witthaker, poteva essere eseguito per un *qualsiasi valore di* N, anche se la sua complessità computazionale ($\mathcal{O}(N^2)$) ne impediva l'implementazione effettiva per valori di N grandi. L'introduzione degli algoritmi FFT ha consentito lo sviluppo delle applicazioni dell'analisi di Fourier, tra le quali vi sono quelle caratteristiche delle discipline che prendono il nome di *digital signal processing* e di *digital image processing*. In tali applicazioni N è almeno dell'ordine di $\mathcal{O}(10^3)$.

A partire dall'idea introdotta da Cooley e Tukey, l'anno successivo Gentleman e Sande pubblicarono un articolo che descrive in maniera completa ed esaustiva l'algorimo FFT, corredandolo di analisi della complessità e della stabilità [21]. Oggi sono disponibili molteplici implementazioni dell'algoritmo FFT, ciascuna specializzata in base a vari fattori come il tipo di fattorizzazione di N e l'applicazione o meno del *bit reversal*. Il pacchetto software FFTW contiene tali implementazioni, utilizzabili in maniera del tutto *trasparente* perchè sono generate automaticamente da un compilatore interno alla libreria.

[20] [14]
[21] [12]

5.4.1 L'algoritmo di Cooley e Tukey

L'idea alla base di tutti gli algoritmi FFT è calcolare la DFT di un vettore di lunghezza $N = r \cdot q$ effettuando q DFT di lunghezza r o viceversa.

♣ **Esempio 5.12.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{11})$ di lunghezza $N = 3{\cdot}4 = 12$:

$$DFT[f]_k := F_k = \sum_{j=0}^{11} f_j w_{12}^{-jk} \tag{5.21}$$

posto:

$$\begin{array}{llllll} k = & 3k_1 + k_0 & k_1 = 0,1,2,3 & k_0 = & 0,1,2 \\ j = & 4j_1 + j_0 & j_1 = 0,1,2 & j_0 = & 0,1,2,3 \end{array}$$

allora:

$$F(k) = \sum_{j_0=0}^{3} \sum_{j_1=0}^{2} f(4j_1 + j_0) w_{12}^{-(4j_1+j_0)(3k_1+k_0)} \tag{5.22}$$

Poiché:

$$w_{12}^{-(4j_1+j_0)(3k_1+k_0)} = w_{12}^{-12j_1k_1} w_{12}^{-4j_1k_0} w_{12}^{-3j_0k_1} w_{12}^{-j_0k_0} = w_{12}^{-12j_1k_1} w_3^{-j_1k_0} w_4^{-j_0k_1} w_{12}^{-j_0k_0}$$

e $w_{12}^{-12j_1k_1} = 1$, la (5.22) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_0=0}^{3} w_4^{-j_0k_1} \underbrace{\sum_{j_1=0}^{2} f(4j_1 + j_0) w_3^{-j_1k_0}}_{DFT\ di\ lunghezza\ 3} w_{12}^{-j_0k_0}}_{4\ DFT\ di\ lunghezza\ 3} \tag{5.23}$$

A parte il fattore esponenziale $w_{12}^{-j_0k_0}$, il calcolo di una DFT di lunghezza 12 si riconduce al calcolo di 4 DFT di lunghezza 3. In particolare, si osserva che i 4 vettori di cui si calcola la DFT, ottenuti da $f(4j_1 + j_0)$ fissando $j_0 = 0, 1, 2, 3$ rispettivamente e facendo variare $j_1 = 0, 1, 2$, sono costruiti considerando le componenti del vettore f i cui indici distano di 4 unità. ♣

Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{N-1})$ di lunghezza $N = r_1 \cdot r_2$:

$$F(k) = \sum_{j=0}^{N-1} f_j w_N^{-jk} \tag{5.24}$$

Posto:

$$\begin{array}{llllll} k = & r_1k_1 + k_0 & k_1 = 0,1,...,r_2-1 & k_0 = & 0,1,...,r_1-1 \\ j = & r_2j_1 + j_0 & j_1 = 0,1,...,r_1-1 & j_0 = & 0,1,...,r_2-1 \end{array}$$

allora:

$$F(k) = \sum_{j_0=0}^{r_2-1} \sum_{j_1=0}^{r_1-1} f(r_2 j_1 + j_0) w_N^{-(r_2 j_1 + j_0)(r_1 k_1 + k_0)} \tag{5.25}$$

Poiché:

$$w_N^{-(r_2 j_1 + j_0)(r_1 k_1 + k_0)} = w_N^{-N j_1 k_1} w_{12}^{-r_2 j_1 k_0} w_N^{-r_1 j_0 k_1} w_N^{-j_0 k_0} = w_N^{-N j_1 k_1} w_{r_1}^{-j_1 k_0} w_{r_2}^{-j_0 k_1} w_N^{-j_0 k_0}$$

e $w_N^{-N j_1 k_1} = 1$, la (5.25) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_0=0}^{r_2-1} w_{r_1}^{-j_0 k_1} \underbrace{\left[\sum_{j_1=0}^{r_1-1} f(r_2 j_1 + j_0) w_{r_2}^{-j_1 k_0} \right]}_{DFT\ di\ lunghezza\ r_1} w_N^{-j_0 k_0}}_{r_2\ DFT\ di\ lunghezza\ r_1} \tag{5.26}$$

A meno dell'esponenziale $w_N^{-j_0 k_0}$, il calcolo di una DFT di lunghezza $N = r_1 \cdot r_2$ si riconduce al calcolo di r_2 DFT di lunghezza r_1. Se poi r_1 si fattorizza nel prodotto $r_2 = r_3 \cdot r_4$, si può applicare tale idea alla DFT di lunghezza r_2, pervenendo via via a DFT di lunghezza più piccola.

♣ **Esempio 5.13.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{11})$ di lunghezza $N = 4 \cdot 3 = 2^2 \cdot 3 = 12$:

$$DFT[f]_k := F(k) = \sum_{j=0}^{11} f_j w_{12}^{-jk} \tag{5.27}$$

posto:

$$\begin{array}{llllll} k = & 4k_1 + k_0 & k_1 = 0,1,2 & k_0 = & 0,1,2,3 \\ j = & 3j_1 + j_0 & j_1 = 0,1,2,3 & j_0 = & 0,1,2 \end{array}$$

si ha:

$$F(k) = \sum_{j_0=0}^{2} \sum_{j_1=0}^{3} f(3j_1 + j_0) w_{12}^{-(4j_1 + j_0)(3k_1 + k_0)} \tag{5.28}$$

Poiché:

$$w_{12}^{-(3j_1 + j_0)(4k_1 + k_0)} = w_{12}^{-12 j_1 k_1} w_{12}^{-3 j_1 k_0} w_{12}^{-4 j_0 k_1} w_{12}^{-j_0 k_0} = w_{12}^{-12 j_1 k_1} w_4^{-j_1 k_0} w_3^{-j_0 k_1} w_{12}^{-j_0 k_0}$$

e $w_{12}^{-12 j_1 k_1} = 1$ allora la (5.28) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_0=0}^{2} w_3^{-j_1 k_1} \underbrace{\left[\sum_{j_1=0}^{3} f(3j_1 + j_0) w_4^{-j_1 k_0} \right]}_{DFT\ di\ lunghezza\ 4} w_{12}^{-j_0 k_0}}_{3\ DFT\ di\ lunghezza\ 4} \tag{5.29}$$

Il calcolo di una DFT di lunghezza 12 si riconduce al calcolo di 3 DFT di lunghezza 4. In questo caso gli $r_2 = 3$ vettori di lunghezza $r_1 = 4$ di cui si deve calcolare la DFT, ottenuti per $j = 0, 1, 2$ rispettivamente, sono costruiti considerando le componenti del vettore f che distano di $r_2 = 3$ unità.

Applichiamo lo stesso schema alla DFT più interna di lunghezza 4. Introduciamo il vettore F^* di lunghezza 3 così definito:

$$F^*(j_0) = \sum_{j_1=0}^{3} f(3j_1 + j_0) w_4^{-j_1 k_0} \quad j_1 = 0, 1, 2, 3 \tag{5.30}$$

Sia:

$$j_1 = \quad 2j_1' + j_0' \qquad j_1' = 0, 1, \quad j_0' = \quad 0, 1$$

allora:

$$F^*(j_0) = \underbrace{\sum_{j_1'=0}^{1} w_4^{-j_0' k_0} \underbrace{\left[\sum_{j_0'=0}^{1} f(3j_1 + j_0) w_2^{-j_1' k_0} \right]}_{DFT\ di\ lunghezza\ 2}}_{2\ DFT\ di\ lunghezza\ 2} \tag{5.31}$$

ovvero una DFT di lunghezza 4 è espressa come 2 DFT di lunghezza 2.
Dalle (5.29) e (5.31) segue quindi che:

$$F(k) = \underbrace{\sum_{j_0=0}^{2} w_3^{-j_1 k_1} \underbrace{\sum_{j_1'=0}^{1} w_2^{-j_0' k_0} \underbrace{\left[\sum_{j_0'=0}^{1} f(3j_1 + j_0) w_2^{-j_1' k_0} \right]}_{DFT\ di\ lunghezza\ 2} w_{12}^{-j_0 k_0}}_{2\ DFT\ di\ lunghezza\ 2}}_{3 \cdot 2 = 6\ DFT\ di\ lunghezza\ 2} \tag{5.32}$$

Si nota che la DFT più interna coinvolge coppie di componenti del vettore f che distano di 3 unità, come anche illustrato in Figura 5.4. In particolare, poiché

$$p_0 = 3j_1 + j_0 = 3(2j_1' + j_0') + j_0 = 6j_1' + 3j_0' + j_0$$

le 6 DFT vengono effettuate fissando nelle due sommatorie esterne $j_0 = 0$ e $j_1' = 0$ e quindi utilizzando le coppie di componenti del vettore f in corrispondenza di $p_0 = 0$ e $p_0 = 3$, poi, fissando $j_0 = 0$ e $j_1' = 1$ si ottiene $p_0 = 6$ e $p_0 = 9$, successivamente fissando $j_0 = 1$ e $j_1' = 0$, si ha $p_0 = 1$ e $p_0 = 4$, mentre per $j_0 = 1$ e $j_1' = 1$ si ha $p_0 = 7$ e $p_0 = 10$. Infine, per $j_0 = 2$ e $j_1' = 0$ segue $p_0 = 2$ e $p_0 = 5$ e per $j_0 = 2$ e $j_1' = 1$ si ha l'ultima coppia di componenti ovvero $p_0 = 8$ e $p_0 = 11$. ♣

Se assumiamo che N sia un multiplo di 2, scegliendo $r_1 = N/2$ e $r_2 = 2$, l'espressione (5.26) si scrive come:

$$F(k) = \underbrace{\sum_{j_0=0}^{1} w_{N/2}^{-j_0 k_1} \underbrace{\left[\sum_{j_1=0}^{N/2-1} f(2j_1 + j_0) w_{N/2}^{-j_1 k_0} \right]}_{DFT\ di\ lunghezza\ r_1 = N/2} w_N^{-j_0 k_0}}_{2\ DFT\ di\ lunghezza\ N/2} \tag{5.33}$$

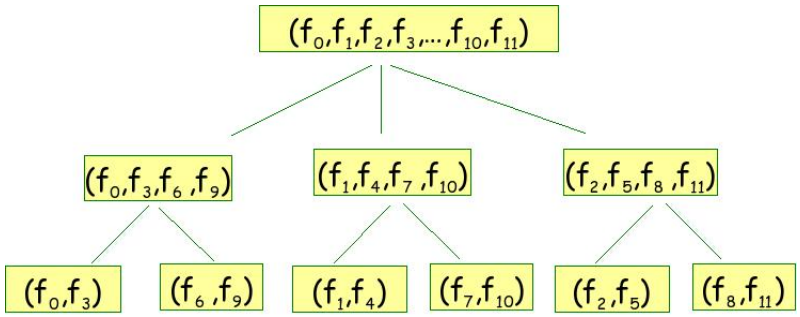

Figura 5.4: Al primo livello dell'albero: il calcolo della DFT di lunghezza 12 viene ricondotto a quello di 3 DFT di lunghezza 4. Al secondo livello dell'albero: il calcolo della DFT di lunghezza 12 viene ricondotto a quello di 6 DFT di lunghezza 2. La distanza fra le componenti è di $r_1 = 3$ unità.

e lo schema appena descritto porta alla decomposizione del vettore f in 2 vettori di lunghezza $N/2$.

Dalla (5.33) si osserva, in particolare, che fissato $j_0 = 0$ nella sommatoria esterna, l'algoritmo di Cooley e Tukey porta alla decomposizione del vettore f in una parte contenente le componenti con indice $2j_1$, con $j_1 = 0, \ldots, N/2$ e quindi le componenti con indice pari, e in un'altra parte, per $j_0 = 1$, contenente le componenti con indice dispari[22]. Siano quindi $\{f_{2j}\}_{j=0,\ldots,N/2}$ e $\{f_{2j+1}\}_{j=0,\ldots,N/2-1}$ i due vettori di lunghezza $N/2$ costruiti a partire dal vettore di componenti $\{f_j\}_{j=0,\ldots,N-1}$ e contenenti, rispettivamente, il primo le componenti di f corrispondenti a valori pari dell'indice j, il secondo le componenti corrispondenti a valori dispari dell'indice j.

♣ **Esempio 5.14.** Consideriamo il calcolo della DFT di un vettore di lunghezza $N = 16 = 2^4$.

$$DFT[f]_k := F(k) = \sum_{j=0}^{15} f(j) w_{16}^{-jk} \quad k = 0, \ldots, 15 \tag{5.34}$$

Fissato k, con $k = 0, \ldots, N-1$ si ha:

$$F(k) = \sum_{j=0}^{15} f(j) w_{16}^{-jk} = \sum_{j=0}^{7} f(2j) w_{16}^{-2jk} + \sum_{j=0}^{7} f(2j+1) w_{16}^{-(2j+1)k} =$$

$$= \sum_{j=0}^{7} f(2j) w_8^{-jk} + w_{16}^{-k} \sum_{j=0}^{7} f(2j+1) w_8^{-jk}$$

Osserviamo che, sfruttando la periodicità dell'esponenziale complesso coinvolto, ovvero dei termini ω_8^{-jk} e ω_8^{-k}, basta calcolare le componenti di $F(k)$ per $k = 0, \ldots, 7$, perché le altre 8 si possono ricavare dalle prime 8. Ad esempio, per $k = 9$ si ha:

$$\omega_8^{-j9} = \omega_8^{-j(8+1)} = \omega_8^{-j\cdot 8} \cdot \omega_8^{-j\cdot 1} = \omega_8^{-j\cdot 1} \quad (\omega_8^r = \omega_8^{r\, mod\, 8}).$$

[22] Questa particolare decomposizione fu evidenziata già nel 1942 da Danielson e Lanczos [5].

e quindi ci si riconduce al calcolo della componente con indice $k = 1$. Siano, allora,

$$F^E(k) = \sum_{j=0}^{7} f(2j) w_8^{-jk}, \quad k = 0, \ldots, 7$$

e

$$F^O(k) = \sum_{j=0}^{7} f(2j+1) w_8^{-jk}, \quad k = 0, \ldots, 7 \quad .$$

L'algoritmo di Cooley e Tukey, al primo passo, porta alla decomposizione del vettore F come:

$$F(k) = F^E(k) + \omega_{16}^{-jk} F^O(k), \quad k = 0, \ldots, 7$$

♣

Dalla (5.24) si ha:

$$F(k) = \sum_{j=0}^{N-1} f(j) \omega_N^{-jk} = \sum_{j=0}^{N/2-1} f(2j) \omega_N^{-2jk} + \sum_{j=0}^{N/2-1} f(2j+1) \omega_N^{-(2j+1)k}. \tag{5.35}$$

Poiché

$$\omega_N^{-(2j+1)k} = \omega_N^{-2jk} \cdot \omega_N^{-k} = \omega_{N/2}^{-jk} \cdot \omega_N^{-k}$$

segue

$$F(k) = \sum_{j=0}^{N/2-1} f(2j) \omega_{N/2}^{-jk} + w_N^{-k} \sum_{j=0}^{N/2-1} f(2j+1) \omega_{N/2}^{-jk}, \quad k = 0, \ldots, N-1$$

Siano

$$F^E(k) = \sum_{j=0}^{N/2-1} f(2j) \omega_{N/2}^{-jk}, \quad k = 0, \ldots, N/2-1$$

e

$$F^O(k) = \sum_{j=0}^{N/2-1} f(2j+1) \omega_{N/2}^{-jk}, \quad k = 0, \ldots, N/2-1$$

le due DFT di lunghezza $N/2$ ottenute a partire dai vettori $\{f_{2j}\}_{j=0,\ldots,N/2-1}$ e $\{f_{2j+1}\}_{j=0,\ldots,N/2-1}$. La (5.35) diventa quindi:

$$F(k) = F^E(k) + \omega_N^{-k} F^O(k) \quad k = 0, \ldots, N/2-1 \tag{5.36}$$

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

Se N è divisibile per 2^2, ovvero $r_1 = N/2$ è divisibile per 2, applicando la stessa idea alle due DFT, $\{F^E(k)\}_{k=0,\ldots,N/2-1}$ e $\{F^O(k)\}_{k=,\ldots,N/2-1}$, di lunghezza $N/2$, l'algoritmo

di Cooley e Tukey decompone ciascuna DFT in 2 DFT di lunghezza $N/4$, ottenute utilizzando le componenti con indice pari e quelle con indice dispari di ciascuno dei due vettori $\{f_{2j}\}_{j=0,\ldots,N/2-1}$ e $\{f_{2j+1}\}_{j=0,\ldots,N/2-1}$. Questo comporta che il calcolo del vettore F al primo passo viene ricondotto a quello di 4 DFT di lunghezza $N/4$.

♣ **Esempio 5.15. (Continua l'esempio 5.14)** Consideriamo i due vettori $\{F^E(k)\}_{k=0,\ldots,7}$ e $\{F^O(k)\}_{k=,\ldots,7}$ ottenute nell'esempio 5.14. Ciascuna DFT di lunghezza 8 si puó decomporre in 2 DFT di lunghezza 4. Ad esempio:

$$F^E(k) = \sum_{j=0}^{7} f(2j) w_8^{-jk} = \sum_{j=0}^{3} f(4j) w_8^{-2jk} + \sum_{j=0}^{3} f(4j+1) w_8^{-(2j+1)k} =$$

$$= \sum_{j=0}^{3} f(4j) w_4^{-jk} + w_8^{-k} \sum_{j=0}^{3} f(4j+1) w_4^{-jk}$$

Siano

$$F^{EE}(k) = \sum_{j=0}^{3} f(4j) w_4^{-jk} \qquad \text{e} \qquad F^{EO}(k) = \sum_{j=0}^{3} f(4j+1) w_4^{-jk}$$

le 2 DFT di lunghezza 4 costruite utilizzando i vettori $\{f_{4j}\}_{j=0,\ldots,3}$ e $\{f_{4j+1}\}_{j=0,\ldots,3}$. Al secondo passo, l'algoritmo di Cooley e Tukey porta alla decomposizione di $\{F^E(k)\}_{k=0,\ldots,N/2-1}$ come:

$$F^E(k) = F^{EE}(k) + w_8^{-k} F^{EO}(k) \quad k = 0, \ldots, 3$$

e quindi il calcolo del vettore F viene ricondotto al calcolo di 4 DFT di lunghezza 4. ♣

In generale, assumendo che N sia esprimibile come una potenza di 2, ad esempio $N = 2^m$, al passo k il calcolo della DFT di lunghezza N viene ricondotto a quello di k DFT di lunghezza 2^{m-k}, ovvero il vettore f viene decomposto in k vettori di lunghezza 2^{m-k}. Dopo $m-1$ passi, l'algoritmo di Cooley e Tukey decompone il vettore f in 2^{m-1} coppie di componenti, laddove ciascuna coppia definisce una DFT di lunghezza 2. L'algoritmo così ottenuto è la variante FFT radix-2 di Cooley e Tukey.

♣ **Esempio 5.16. (Continua l'esempio 5.15)** Riscrivendo ciascuna DFT di lunghezza 4 (ottenuta nell'esempio 5.15) come somma di 2 DFT di lunghezza 2, si ha:

$$F^{EE}(k) = \sum_{j=0}^{3} f(j) w_4^{-jk} = \sum_{j=0}^{1} f(8j) w_4^{-2jk} + \sum_{j=0}^{1} f(8j+1) w_4^{-(2j+1)k} =$$

$$= \sum_{j=0}^{1} f(8j) w_2^{-jk} + w_4^{k} \sum_{j=0}^{1} f(8j+1) w_2^{-jk}$$

Siano

$$F^{EEE}(k) = \sum_{j=0}^{1} f(8j) w_2^{-jk} \qquad \text{e} \qquad F^{EEO}(k) = \sum_{j=0}^{1} f(8j+1) w_2^{-jk}$$

le 2 DFT di lunghezza 2 ottenute condiderando rispettivamente le componenti (f_0, f_8) e (f_1, f_9). In conclusione, per $N = 16 = 2^4$, dopo 3 passi l'algoritmo di Cooley e Tukey porta alla decomposizione seguente:

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

laddove la prima somma coinvolge 2 DFT di lunghezza 8, la seconda somma coinvolge 4 DFT di lunghezza 4 e l'ultima 8 DFT di lunghezza 2. Questo è lo schema dell'algoritmo FFT radix-2, introdotto da Cooley e Tukey. Si nota la natura ricorsiva dell'algoritmo di Cooley e Tukey. ♣

L'algoritmo di Cooley e Tukey decompone ripetutamente il vettore f, di lunghezza $N = 2^m$, separando le componenti con indice pari da quelle con indice dispari fino ad arrivare a 2^{m-1} coppie di componenti. Dopo questa prima fase di *decomposizione* si passa alla fase di *calcolo*: partendo dalla DFT più piccola, ovvero quella di lunghezza 2 si calcolano DFT di lunghezza 4 combinando 2 DFT di lunghezza 2, poi, combinando 2 DFT di lunghezza 4, si calcolano DFT di lunghezza 8 e così proseguendo, fino a ottenere 1 DFT di lunghezza N.

Questa versione dell'algoritmo prevede da un punto di vista implementativo una fase di *riordinamento* iniziale e una seconda fase di *calcolo* sul vettore ordinato. In tal modo, questa versione produce il vettore risultante secondo l'ordine naturale.

In maniera analoga derivano gli algoritmi FFT radix-r. I più comuni sono quelli per $r = 3$ e per $r = 5$.

♣ **Esempio 5.17.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_8)$ di lunghezza $N = 3 \cdot 3 = 9$:

$$DFT[f]_k := F_k = \sum_{j=0}^{8} f_j w_9^{jk} \tag{5.37}$$

Posto:

$$\begin{array}{llll} k = & 3k_1 + k_0 & k_1, k_0 = & 0, 1, 2 \\ j = & 3j_0 + j_1 & j_1, j_0 = & 0, 1, 2 \end{array}$$

si ha

$$F(k) = \sum_{j_0=0}^{2} \sum_{j_1=0}^{2} f(3j_0 + j_1) w_9^{-(3j_0+j_1)(3k_1+k_0)} \tag{5.38}$$

Poiché:

$$w_9^{-(3j_0+j_1)(3k_1+k_0)} = w_9^{-9j_0k_1} w_9^{-3j_0k_0} w_9^{-3j_1k_1} w_9^{-j_1k_0}$$

e $w_9^{-9j_0k_1} = 1$ la (5.38) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_1=0}^{2} w_9^{3j_1k_1} \underbrace{\left[\sum_{j_0=0}^{2} f(3j_0+j_1) w_9^{-3j_0k_0}\right]}_{DFT\ di\ lunghezza\ 3} w_9^{-j_1k_0}}_{3\ DFT\ di\ lunghezza\ 3} \tag{5.39}$$

Il calcolo di una DFT di lunghezza 9 si riconduce al calcolo 3 DFT di lunghezza 3. ♣

La strategia descritta nell'esempio precedente è alla base della classe di algoritmi FFT *radix-3* in cui il vettore f ha lunghezza $N = 3^m$.

5.4.2 L'algoritmo di Gentleman e Sande

Una strategia leggermente diversa da quella alla base dello schema di Cooley e Tukey sottende la versione dell'algoritmo FFT radix-2 introdotta da Gentleman e Sande. Tale variante si basa sul decomporre ripetutamente il vettore DFT da calcolare. Si osserva che le componenti con indice pari della DFT si possono esprimere come una DFT di lunghezza $N/2$ costruita a partire da un vettore ottenuto combinando le componenti del vettore di input che distano di $N/2$. Analogo ragionamento si fa per le componenti con indice dispari della DFT da calcolare.

♣ **Esempio 5.18.** Consideriamo il calcolo della DFT di un vettore di lunghezza $N = 8 = 2^3$.

$$F(k) = \sum_{j=0}^{8} f_j w_8^{-jk} \tag{5.40}$$

Da:

$$F(k) = \sum_{j=0}^{8} f_j w_8^{-jk} = \sum_{j=0}^{3} f_j w_8^{-jk} + \sum_{j=0}^{3} f_{j+4} w_8^{-(j+4)k} \tag{5.41}$$

separiamo le componenti del vettore F con indice pari da quelle con indice dispari, ovvero consideriamo i due vettori:

$$(F_0, F_2, F_4, F_6) \quad e \quad (F_1, F_3, F_5, F_7)\,.$$

Sfruttando la proprietà di periodicità dell'esponenziale complesso, si ha:

$$F(2k) = \sum_{j=0}^{3} [f(j) + f(j+4)] w_4^{-jk} \quad , k = 0, \ldots, 3$$

e

$$F(2k+1) = \sum_{j=0}^{3} [f(j) - f(j+4)] w_4^{-j} w_4^{-jk} \quad , k = 0, \ldots, 3$$

Primo passo: L'espressione di $F(2k)$ e di $F(2k+1)$ suggerisce di costruire i 2 vettori di lunghezza 4:

$$y = (f_0 + f_4, f_1 + f_5, f_2 + f_6, f_3 + f_7)$$

e

$$z = (f_0 - f_4, (f_1 - f_5)w_4^{-1}, (f_2 - f_6)w_4^{-2}, (f_3 - f_7)w_4^{-3})$$

Per cui, i due vettori $\{F(2k)\}_{k=0,1,2,3}$ e $\{F(2k+1)\}_{k=0,1,2,3}$ si ottengono come DFT di lunghezza 4 costruite a partire dai vettori y e z:

$$F(2k) = \sum_{j=0}^{3}[f(j) + f(j+4)]w_4^{-jk} = \sum_{j=0}^{3} y(j)w_4^{-jk} \quad k = 0, \ldots, 3$$

e

$$F(2k+1) = \sum_{j=0}^{3} z(j)w_4^{-jk} \quad k = 0, \ldots, 3$$

Per ciascuno dei due vettori $\{F(2k)\}_{k=0,1,2,3}$ e $\{F(2k+1)\}_{k=0,1,2,3}$, separiamo il calcolo delle componenti con indice pari da quello delle componenti con indice dispari. Ad esempio, per quanto riguarda $\{F(2k)\}_{k=0,1,2,3}$, consideriamo

$$(F(0), F(4)) = \{F(2k)\}_{k=0,2}$$

e

$$(F(2), F(6)) = \{F(2k)\}_{k=1,3}$$

La coppia $(F(0), F(4))$ si può esprimere come DFT di lunghezza 2 costruita a partire dal vettore di componenti $(y_0 + y_2, y_1 + y_3)$. Cioé:

$$\{F(2k)\}_{k=0,2} = \sum_{j=0}^{1}[y(j) + y(j+2)]w_2^{-jk}$$

$$\{F(2k)\}_{k=1,3} = \sum_{j=0}^{1}[y(j) + y(j+2)]w_2^{-jk}w_4^{-j}$$

Analogamente, la coppia $(F(2), F(6))$ si puó esprimere come una DFT di lunghezza 2 costruita a partire dal vettore di componenti $(y_0 - y_2, (y_1 - y_3)w_4^{-1})$.
Secondo passo: questa analisi suggerisce di costruire i due vettori

$$y' = (y_0 + y_2, y_1 + y_3), \quad y'' = (y_0 - y_2, (y_1 - y_3)\omega_4^{-1})$$

per cui il vettore di componenti $\{F(2k)\}_{k=0,1,2,3}$ si puó ottenere calcolando 2 DFT di lunghezza 2 costruite a partire dai vettori y' e y''. Cioè:

$$F(2k)_{k=0,2} = \sum_{j=0}^{2} y'(j)w_2^{-jk}, \qquad F(2k+1)_{k=0,2} = \sum_{j=0}^{2} y''(j)w_2^{-jk}.$$

Analogo ragionamento vale per il vettore di componenti $\{F(2k+1)\}_{k=0,1,2,3}$. Separando le componenti con indice pari da quelle con indice dispari, cioè considerando i due sottovettori: $(F(1), F(5))$ e $(F(3), F(7))$, si trova che

$$(F(1), F(5)) = \{F(2k+1)\}_{k=0,2} = \sum_{j=0}^{1}[z(j) + z(j+2)]w_2^{-jk} = \sum_{j=0}^{1} z'_j w_2^{-jk} \quad k = 0, 2$$

$$(F(3), F(7)) = \{F(2k+1)\}_{k=1,3} = \sum_{j=0}^{1}[z(j) - z(j+2)]w_2^{-jk}w_4^{-j} = \sum_{j=0}^{1} z''(j)w_2^{-jk} \quad k = 1, 3$$

Si costruiscono quindi i vettori:

$$z' = (z_0 + z_2, z_1 + z_3), \quad z'' = (z_0 - z_2, z_1 - z_3)$$

per cui ciascuna coppia di componenti si può esprimere come DFT di lunghezza 2 costruita rispettivamente su z' e z''.
Il terzo passo comporta infine il calcolo delle DFT di lunghezza 2 costruite al secondo passo, e quindi il calcolo delle quantità:

$$y''' = y'_0 + y'_1, \quad y^{iv} = y'_0 - y'_1$$
$$y^{v} = y''_1 + y''_2 \quad y^{vi} = y''_1 - y''_2$$
$$z''' = z'_0 + z'_1, \quad z^{iv} = z'_0 - z'_1$$
$$z^{v} = z''_1 + z''_2 \quad z^{vi} = z''_1 - z''_2$$

Il vettore $F = DFT[f]$ è:

$$F = (y''', y^{iv}, y^{v}, y^{vi}, z''', z^{iv}, z^{v}, z^{vi}) = (F_0, F_4, F_2, F_6, F_1, F_5, F_3, F_7)$$

Come si può osservare, il vettore F ha le componenti non ordinate secondo l'ordine naturale. La componente F_4 si trova al posto della componente F_1 e viceversa, la componente F_6 si trova al posto della componente F_3 e viceversa. L'ordinamento con il quale si presentano le componenti del vettore DFT è detto **ordinamento del bit inverso** (*bit reversal*). ♣

Applicando la proprietà di periodicità dell'esponenziale complesso si ha:

$$F(k) = \sum_{j=0}^{N-1} f(j)w_N^{-jk} = \sum_{j=0}^{N/2-1} f(j)w_N^{-jk} + \sum_{j=0}^{N/2-1} f(j + N/2)w_N^{-(j+N/2)k}$$

da cui:

$$\begin{aligned} F(k) &= \sum_{j=0}^{N/2-1} [f(j)w_N^{-jk} + f(j + N/2)w_N^{(-j+N/2)k}] = \\ &= \sum_{j=0}^{N/2-1} [f(j) + (-1)^k f(j + N/2)]w_{N/2}^{-jk/2} \end{aligned}$$

l'ultima eguaglianza sussiste in quanto $w_N^{-N/2} = -1$ e $w_N^{-jk} = w_N^{-jk/2}$.
Consideriamo le componenti con indice pari:

$$F(2k) = \sum_{j=0}^{N/2-1} [f(j) + f(j + N/2)w_{N/2}^{-jk}], \quad k = 0, \ldots, N/2 - 1$$

e quelle con indice dispari

$$F(2k+1) = \sum_{j=0}^{N/2-1} [(f(j) - f(j+N/2))w_N^{-j}]w_{N/2}^{-jk}, \quad k = 0, \ldots, N/2-1$$

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

$$y_j = f_j + f_{j+N/2}, \quad z_j = (f_j - f_{j+N/2})w_N^{-j}, \quad j = 0, \ldots, N/2-1$$

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

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

Al **primo passo**, l'algoritmo di Gentleman e Sande calcola i due vettori y e z.

Consideriamo ciascuna DFT e separiamo le componenti con indice pari da quelle con indice dispari. I due vettori così costruiti, di lunghezza $N/4$, sono esprimibili in termini di DFT di lunghezza $N/4$:

$$DFT[y] = \sum_{j=0}^{N/2-1} [f(j) + f(j+N/2)w_{N/2}^{-jk}] = \sum_{j=0}^{N/2-1} y(j)w_{N/2}^{-jk} =$$

$$= \sum_{j=0}^{N/4-1} (y(j) + y(j+N/4))w_{N/4}^{-jk} + \sum_{j=0}^{N/4-1} [(y(j) - y(j+N/4))\omega_{N/2}^{-j}]w_{N/4}^{-jk}$$

Se consideriamo le componenti con indice pari del vettore $DFT[y]$:

$$DFT[y](2k) = \sum_{j=0}^{N/4-1} (y(j) + y(j+N/4))\omega_{N/4}^{-jk} \quad k = 0, \ldots, N/4-1$$

e

$$DFT[y](2k+1) = \sum_{j=0}^{N/4-1} (y(j) - y(j+N/4))\omega_{N/2}^{-j}\omega_{N/4}^{-jk} \quad k = 0, \ldots, N/4-1$$

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

$$y'_j = (y_j + y_{j+N/4}), \quad y''_j = (y_j - y_{j+N/4})w_{N/2}^{-j},$$

segue che:

$$\{DFT[y]\} = \{DFT[y'], DFT[y'']\},$$

Analogamente, per il vettore $DFT[z]$ contenente le componenti con indice dispari della DFT di f:

$$DFT[z](k) = \sum_{j=0}^{N/2-1} [(f(j) - f(j+N/4))w_{N/2}^{-j}]w_{N/4}^{-jk} = \sum_{0}^{N/2-1} z(j)w_{N/2}^{-jk}$$

$$= \sum_{j=0}^{N/4-1} [(z(j)+z(j+N/4))w_{N/2}^{j}]w_{N/4}^{jk} + \sum_{j=0}^{N/4-1} [(z(j)-z(j+N/4))w_{N/2}^{-j}]w_{N/4}^{-jk}, \quad k = 0, N/4-1$$

Introdotti i vettori $\{z'\}_{j=0,l...,N/4-1}$ e $\{z''\}_{j=0,l...,N/4-1}$ di componenti

$$z'_j = z_j + z_{j+N/4}, \quad z''_j = (z_j - z_{j+N/4})w_{N/2}^{-j},$$

segue che

$$DFT[z] = \{DFT[z'], DFT[z'']\}$$

Al **secondo passo**, l'algoritmo di Gentleman e Sande calcola i vettori y' , y'', z' e z'' e il calcolo della DFT di lunghezza N viene ricondotto a quello di 4 DFT di lunghezza $N/4$:

$$F = DFT[f] = (DFT[y], DFT[z]) = (DFT[y'], DFT[y''], DFT[z'], DFT[z''])$$

Proseguendo in questo modo, dopo $m = \log_2(N)$ passi, l'algoritmo di Gentleman e Sande costruisce $N/2$ vettori di lunghezza 2. Ciascuna coppia è una DFT e fornisce una coppia di componenti della DFT del vettore f.

A differenza della variante di Cooley e Tukey, si nota che in questa strategia le DFT calcolate in un certo passo sono ottenute *utilizzando il vettore costruito al passo precedente*, per cui l'algoritmo di Gentleman e Sande prevede inizialmente una fase di calcolo, per un certo numero di passi dipendente dalla dimensione del vettore f, e poi, una volta calcolate le $N/2$ DFT di lunghezza 2, è necessaria una fase di riordinamento del vettore DFT, dal momento che tale vettore risulta nell'ordine *scrambled*, ovvero secondo l'ordinamento indotto dal *bit reversal*.
Questa variante è anche nota in letteratura come *decimazione nelle frequenze*, perché la decomposizione viene effettuata sulle componenti del vettore DFT , e nell'analisi armonica tale vettore fornisce informazioni sulle frequenze contenute nel vettore dato [23].

*L'**ordinamento del bit inverso** (bit reversal) è definito nel modo seguente: due interi positivi, j e k, sono in* relazione *secondo l'ordinamento del bit inverso se, considerata la rappresentazione binaria di j e invertendo l'ordine delle cifre binarie, si ottiene il numero intero decimale k. Una coppia di numeri naturali i_1, i_2 si dice* ordinata *secondo il bit inverso se i numeri interi naturali k_1, k_2, in relazione secondo l'ordinamento del bit inverso rispettivamente con i_1 e i_2, sono ordinati secondo l'ordinamento naturale. Un insieme finito di numeri interi risulta ordinato secondo il bit inverso, se una qualsiasi coppia di elementi di tale insieme risulta ordinata secondo il bit inverso. Un vettore, $v \in \Re^N$, 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. In altre parole, considerata la componente di indice j, questa si trova al posto*

[23] [12]

della componente di indice k dove k è l'indice in relazione con j secondo l'ordinamento del bit inverso.

♣ **Esempio 5.19.** Consideriamo 4 numeri interi ordinati (in senso crescente):

$$0, 1, 2, 3.$$

Consideriamo la rappresentazione binaria di ciascuno:

$$\begin{array}{ll} (0)_{base10} = & (00)_{base2} \\ (1)_{base10} = & (01)_{base2} \\ (2)_{base10} = & (10)_{base2} \\ (3)_{base10} = & (11)_{base2} \end{array}$$

e invertiamola. Si ottiene:

$$\begin{array}{ll} (00)_{base2} = & (0)_{base10} \\ (10)_{base2} = & (2)_{base10} \\ (01)_{base2} = & (1)_{base10} \\ (11)_{base2} = & (3)_{base10} \end{array}$$

I numeri interi

$$0\ ,\ 2\ ,\ 1\ ,\ 3$$

sono ordinati in senso crescente secondo la relazione del bit inverso. ♣

5.4.3 Complessità computazionale degli algoritmi FFT

In generale, quindi, gli algoritmi FFT effettuano il calcolo di una DFT di lunghezza N combinando opportunamente DFT di lunghezza inferiore. In base alla fattorizzazione del parametro N si distinguono essenzialmente tre tipologie di algoritmi FFT.

1. Se $N = r_1 r_2$, e r_1 e r_2 sono primi tra loro, gli algoritmi FFT calcolano r_1 DFT di lunghezza r_2 (*algoritmi FFT con fattori primi (Prime Factor Algorithms (PFA))*). Di questi il più noto è l'algoritmo di Rader [24]. La complessità di questi algoritmi è
$$T(N) = \mathcal{O}(N(r_1 + r_2))$$

2. Se $N = r_1^p r_2^q$ gli algoritmi FFT calcolano p DFT di lunghezza r_1 e q DFT di lunghezza r_2 (algoritmi a radici miste *(FFT mixed radix)*. Tipicamente, la radice in questo caso è un numero primo sufficientemente piccolo, ad esempio $r = 3, 5$. La complessità di questa classe di algoritmi è:
$$T(N) = \mathcal{O}(p \log_{r_1} N + q \log_{r_2} N)$$

[24] [24]

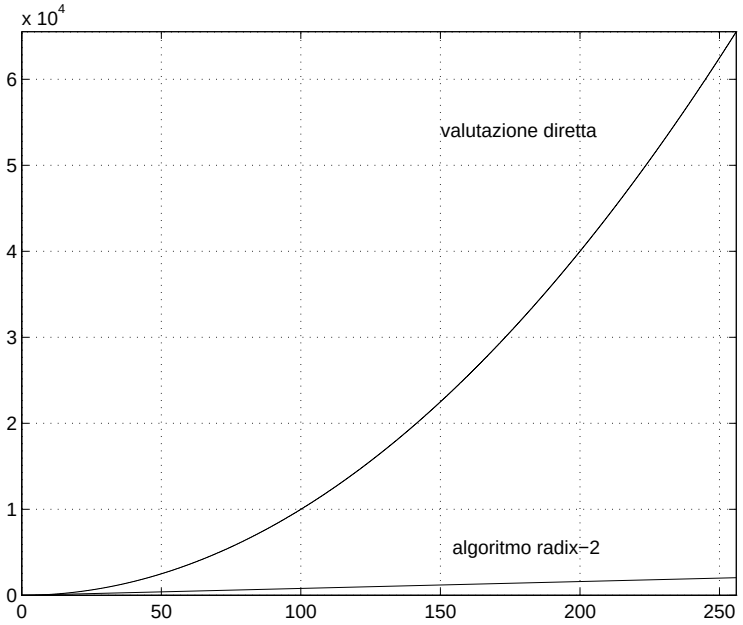

Figura 5.5: Confronto tra la complessità computazionale della valutazione diretta della DFT di una sequenza di numeri complessi di lunghezza N e la FFT della medesima sequenza.

3. Se $N = r^p$ gli algoritmi FFT calcolano p DFT di lunghezza r (algoritmi *radix-r*). In particolare, tra gli algoritmi radix-r, i più efficienti sono quelli del tipo radix-2 a cui appartengono anche quelli in cui r è una potenza di 2 come gli algoritmi radix-4 e radix-8. Ciò deriva dal fatto che in questi casi le funzioni trigonometriche assumono valori uguali a ± 1 e 0. La complessità di tempo $T(N)$ di questi algoritmi è:

$$T_{DFT}(N) = \mathcal{O}(N \log_2 N)$$

Nel grafico in Figura 5.5 sono confrontate la complessità computazionale della valutazione diretta della DFT e quella dell'algoritmo FFT radix-2. Si osserva come la differenza tra le due funzioni, rispettivamente $y = N^2$ e $y = N \log_2 N$, al crescere di N aumenta in maniera molto significativa; ad esempio per $N = 10^8$, risulta $N^2 = 10^{16}$ e $N \log_2 N \simeq 10^8 \cdot 26$ il che significa che, utilizzando un calcolatore con una potenza di calcolo di $1Glop/s = 10^9 flop/s$, nel primo caso occorrono $10^{16} \cdot 10^{-9} sec = 10^7 sec$ corrispondenti a circa 3 mesi, mentre nel secondo caso occorrono appena $26 \cdot 10^8 \cdot 10^{-9} sec = 2.6 sec$

Esiste una stretta relazione tra l'algoritmo radix-2 e gli algoritmi radix-r, infatti la complessità di tali algoritmi è data da:

$$T(N) = \mathcal{O}(Nr log_r N) = \mathcal{O}\left(Nr \frac{log_2 N}{log_2 r}\right) = \mathcal{O}\left(\frac{r}{log_2 r} \cdot N log_2 N\right)$$

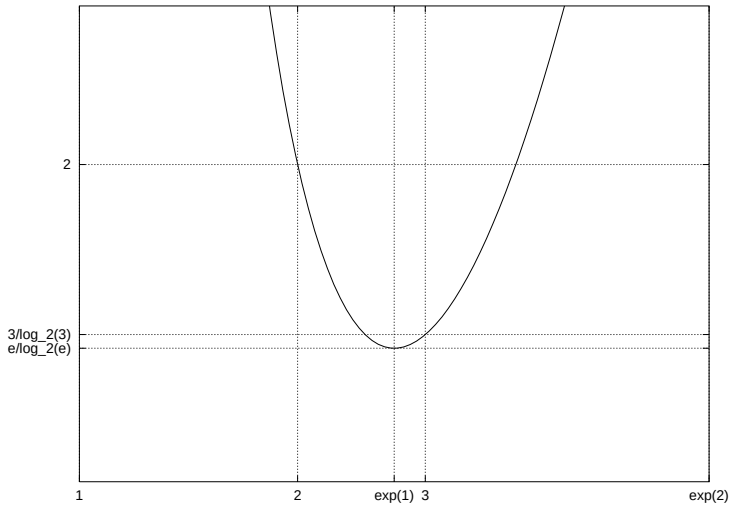

Figura 5.6: Rappresentazione del fattore di proporzionalità $\frac{r}{log_2 r}$.

Il fattore di proporzionalità $r/log_2 r$, come anche mostrato in Figura 5.6, assume il suo valore minimo per $r = e$; infatti:

$$\frac{d}{dr}\frac{r}{log_2 r} = \frac{log_2 r - log_2 e}{(log_2 r)^2} \geq 0 \Longleftrightarrow r \geq e$$

e ha in $r = 3$ un valore minore di quello che assume per $r = 2$:

$$\frac{3}{log_2 3} \simeq 1.893, \quad \frac{2}{log_2 2} = 2$$

quindi l'algoritmo FFT radix-3 è quello che ha la complessità computazionale minima; nonostante ciò l'algoritmo radix-2 è di gran lunga il più utilizzato, sia perché la maggior parte dei calcolatori ha un sistema aritmetico floating point in base 2, sia perché con questa scelta alcune potenze w_N^k sono semplificate (abbiamo visto che il numero complessivo di esponenziali da calcolare è uguale alla metà della lunghezza della sequenza di cui si vuole la FFT).

♣ **Esempio 5.20.** Riprendiamo l'esempio 5.10. Abbiamo visto che la risoluzione del sistema lineare

$$Mu = b$$

attraverso l'uso della DFT comporta il calcolo di $4n$ DFT di lunghezza $2N$. Se $N = 10^6$, questo significa effettuare

$$4n(2N \log_2(2N)) = 4 \cdot 10^3 \cdot 2 \cdot 10^6 \log_2(2 \cdot 10^6) = \mathcal{O}(10^9) \quad flop \tag{5.42}$$

La risoluzione diretta del sistema lineare con l'algoritmo di Cholesky richiede invece

$$T_{chol} = (1/3)N^3 = (1/3) \cdot 10^{18} \quad flop$$

mentre, se si utilizza l'algoritmo specializzato per matrici tridiagonali a blocchi, è necessaria una complessità dell'ordine

$$T_{bandchol} = \mathcal{O}(2N^2) = 2 \cdot 10^{12} \quad flop$$

Infine, dalla (4.42) del **Capitolo 4**, **§4.6**, la risoluzione del sistema mediante un metodo iterativo, in un numero k di iterazioni, richiede una complessità di tempo asintotica:

$$T(n) = k \cdot T_{it}(N) = \mathcal{O}(kN^2(1 - SP)) = \mathcal{O}(k \cdot 10^{12} \cdot (1 - SP)) \quad flop$$

Quest'ultima risulta essere maggiore della (5.42) se il numero di iterazioni k soddisfa la condizione:

$$k(1 - SP) > 10^{-3}$$

Dal confronto tra le complessità si deduce che, utilizzando un calcolatore con una prestazione di 1 Gflop/s, si ottiene una riduzione da 10 anni ($\frac{1}{3}10^{18} \cdot 10^{-9} = \frac{1}{3}10^9$ sec $\simeq$ 10 anni) con l'algoritmo di Cholesky a mezz'ora ($2 \cdot 10^{12} \cdot 10^{-9} = 2 \cdot 10^3$ sec $\simeq$ 30 min) con la versione a banda ed infine a 3 minuti ($8 \cdot 10^9 \cdot \log_2(2 \cdot 10^6) \cdot 10^{-9}$sec $\simeq$ 3 min) utilizzando la FFT.

♣

5.4.4 Aspetti implementativi dell'algoritmo FFT radix-2

Consideriamo l'algoritmo FFT radix-2 che si applica nel caso in cui $N = 2^m$. Come abbiamo visto, sia l'algoritmo di Cooley e Tukey sia la versione proposta da Gentleman e Sande hanno come operazione di base il calcolo di una DFT di lunghezza 2.

♣ Esempio 5.21. DFT di lunghezza 2

Sia $N = 2^1$, si vuole calcolare la DFT del vettore $\underline{f} = (f_0, f_1)$. Per definizione di DFT si ha:

$$F_k = \sum_{j=0}^{1} f_j \omega_2^{-jk} \quad k = 0, 1$$

ovvero:

$$k = 0 \quad \rightarrow \quad F_0 = f_0\omega_2^0 + f_1\omega_2^0 \tag{5.43}$$

$$k = 1 \quad \rightarrow \quad F_1 = f_0\omega_2^0 + f_1\omega_2^{-1} \tag{5.44}$$

il calcolo di F_0 ed F_1 richiede, quindi, 4 moltiplicazioni complesse. D'altra parte osservando che gli esponenziali complessi coinvolti sono:

$$\omega_2^0 = e^{-i\frac{2\pi}{2}0} = cos\left(\frac{2\pi}{2}0\right) - i\, sen\left(\frac{2\pi}{2}0\right) = 1$$

$$\omega_2^{-1} = e^{-i\frac{2\pi}{2}1} = cos\left(\frac{2\pi}{2}1\right) - i\, sen\left(\frac{2\pi}{2}0\right) = -1$$

la (5.43) e la (5.44) si riducono ad una somma ed una sottrazione tra numeri complessi:

$$\begin{array}{lll} k = 0 & \rightarrow & F_0 = f_0 + f_1 \\ k = 1 & \rightarrow & F_1 = f_0 - f_1 \end{array} \tag{5.45}$$

♣

Il calcolo di una DFT di lunghezza 2 richede 2 operazioni (somma e differenza) alle quali si associa lo schema grafico (mostrato in Figura 5.7), detto *schema a farfalla (o butterfly)*.

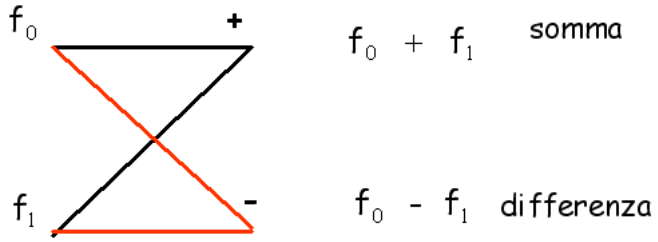

Figura 5.7: Schema butterfly.

Più in generale, gli algoritmi FFT si basano su un'operazione del tipo:

$$\begin{aligned} k = 0 \rightarrow &\quad F_0 = f_0 + f_1 \\ k = 1 \rightarrow &\quad F_1 = w(f_0 - f_1) \end{aligned}$$

dove w è un opportuno esponenziale complesso, come mostrato in Figura 5.8.

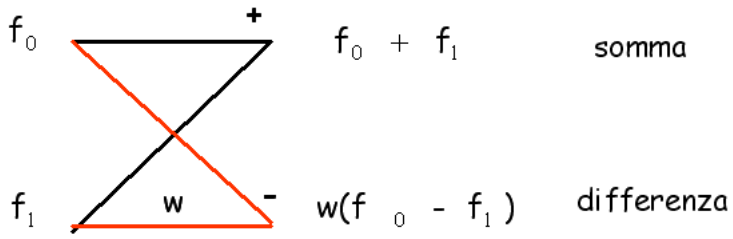

Figura 5.8: Schema generale di una butterfly.

Da un punto di vista implementativo tutti gli algoritmi FFT radix-2 ad ogni passo si compongono di operazioni di tipo *butterfly*.

♣ **Esempio 5.22. DFT di lunghezza 4**
Sia $N = 2^2 = 4$, si vuole calcolare la DFT del vettore:

$$\underline{f} = (f_0, f_1, f_2, f_3).$$

Dalla definizione di DFT si ha:

$$F_k = \sum_{j=0}^{3} f_j \omega_4^{-jk} \quad k = 0, 1, 2, 3$$

Poiché gli esponenziali complessi sono:

$$\begin{array}{l}
\omega_4^{-0} = e^{-i\frac{2\pi}{4}0} = cos(\frac{2\pi}{4}0) - isen(\frac{2\pi}{4}0) = 1 \\
\omega_4^{-1} = e^{-i\frac{2\pi}{4}1} = cos(\frac{2\pi}{4}1) - isen(\frac{2\pi}{4}1) = -i \\
\omega_4^{-2} = e^{-i\frac{2\pi}{4}2} = cos(\frac{2\pi}{4}2) - isen(\frac{2\pi}{4}2) = -1 \\
\omega_4^{-3} = e^{-i\frac{2\pi}{4}3} = cos(\frac{2\pi}{4}3) - isen(\frac{2\pi}{4}3) = i \\
\omega_4^{-4} = e^{-i\frac{2\pi}{4}4} = cos(\frac{2\pi}{4}4) - isen(\frac{2\pi}{4}4) = 1 \\
\omega_4^{-6} = e^{-i\frac{2\pi}{4}6} = cos(\frac{2\pi}{4}6) - isen(\frac{2\pi}{4}6) = -1 \\
\omega_4^{-9} = e^{-i\frac{2\pi}{4}9} = cos(\frac{2\pi}{4}9) - isen(\frac{2\pi}{4}9) = -i
\end{array} \tag{5.46}$$

segue:

$$\begin{array}{llll}
F_0 = f_0 & + f_1 & +f_2 & + f_3 \\
F_1 = f_0 & -if_1 & -f_2 & +if_3 \\
F_2 = f_0 & - f_1 & +f_2 & - f_3 \\
F_3 = f_0 & +if_1 & -f_2 & -if_3
\end{array} \tag{5.47}$$

Se si raccolgono, in maniera opportuna, alcuni termini nella (5.47) si ottiene:

$$\begin{array}{l}
F_0 = \boldsymbol{(f_0 + f_2) + (f_1 + f_3)} \\
F_1 = \boldsymbol{(f_0 - f_2) - i(f_1 - f_3)} \\
F_2 = (f_0 + f_2) - (f_1 + f_3) \\
F_3 = (f_0 - f_2) + i(f_1 - f_3)
\end{array} \tag{5.48}$$

e si separano le componenti del vettore DFT con indice pari da quelle con indice dispari, si ha:

$$\begin{array}{ll}
F_0 = \boldsymbol{(f_0 + f_2) + (f_1 + f_3)} & F_1 = \boldsymbol{(f_0 - f_2) - i(f_1 - f_3)} \\
F_2 = (f_0 + f_2) - (f_1 + f_3) & F_3 = (f_0 - f_2) + i(f_1 - f_3)
\end{array} \tag{5.49}$$

Secondo l'algoritmo di Gentleman e Sande, come descritto nel paragrafo 5.4.2, il calcolo del vettore F viene realizzato attraverso i seguenti passi:

- **passo 1:** si calcolano 2 vettori di lunghezza $2 = N/2$: (y_0, y_1) e (z_0, z_1), con:

$$\begin{array}{ll}
y_0 = (f_0 + f_2) & z_0 = (f_0 - f_2) \\
y_1 = (f_1 + f_3) & z_1 = -i(f_1 - f_3)
\end{array} \tag{5.50}$$

- **passo 2:** si calcolano 4 vettori di lunghezza $1 = N/4$:

$$\begin{array}{ll}
y'_0 = y_0 + y_1 & z'_0 = z_0 + z_1 \\
y'_1 = y_0 - y_1 & z'_1 = z_0 - z_1
\end{array} \tag{5.51}$$

Il vettore (y'_0, y'_1, z'_0, z'_1) è la DFT del vettore F.

Esplicitando le componenti del vettore costruito al secondo passo, si ottiene:

$$(y'_0, y'_1, z'_0, z'_1) = (y_0 + y_1, y_0 - y_1, z_0 + z_1, z_0 - z_1) =$$

$$= f_0 + f_2 + f_1 + f_3, f_0 + f_2 - (f_1 + f_3), (f_0 - f_2) - i(f_1 - f_3), (f_0 - f_2) + i(f_1 - f_3)) = (F_0, F_2, F_1, F_3)$$

cioè si ottengono le componenti del vettore F disposte non nell'ordine naturale, bensì nell'ordine del *bit reversal (scrambled)*, ovvero, ad esempio, la componente di indice $2 = (100)_2$ si trova al posto della componente di indice $1 = (001)_2$.

Le operazioni riportate nella (5.50) si possono descrivere graficamente utilizzando lo schema della Figura 5.9, che corrisponde a due *butterfly* innestate.

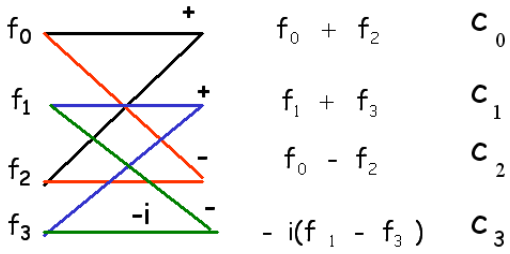

Figura 5.9: Schema del passo 1.

Osserviamo che per ottenere le componenti y_0 e z_0, risultato dell'operazione *butterfly* sulle componenti (f_0, f_2) del vettore iniziale, è necessario utilizzare **esclusivamente** le componenti (f_0, f_2). Analogamente, per ottenere le componenti y_1 e z_1 è necessario utilizzare solo le componenti (f_1, f_3). Questo fatto induce a implementare l'algoritmo *in place*, ovvero di utilizzare le medesime locazioni di memoria del vettore di input f per memorizzare le componenti dei vettori costruiti al primo passo. Naturalemnte, questo si può fare se i due vettori y e z ottenuti al primo passo si costruiscono calcolando le componenti omologhe rispettivamente del primo e del secondo vettore.
Introdotto il vettore

$$c = (y_0, y_1, z_0, z_1)$$

il primo passo dell'algoritmo di Gentleman e Sande equivale alla costruzione di tale vettore effettuando 2 butterly: la prima necessaria al calcolo delle componenti (c_0, c_2) e la seconda per il calcolo delle componenti (c_1, c_3). Inoltre, ciascuna *butterfly* può essere effettuata *in place*, ovvero il risultato può essere memorizzato utilizzando le stesse locazioni di memoria riservate per le componenti di input.
Analogamente al **secondo passo**, le operazioni riportate nella (5.51) si possono descrivere graficamente utilizzando lo schema della Figura 5.10.

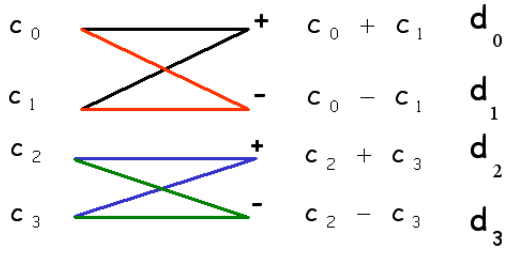

Figura 5.10: Schema del passo 2.

Anche al secondo passo, le componenti (y'_0, y'_1) risultato della butterfly sulle componenti (c_0, c_1) del vettore costruito al primo passo, sono ottenute utilizzando esclusivamente (c_0, c_1) e pertanto possono essere memorizzate nelle locazioni di memoria di queste ultime. Analogo risultato vale per l'altra coppia di componenti, ovvero per (z'_0, z'_1).
Anche in questo caso, introdotto il vettore

$$d = (y'_0, y'_1, z'_0, z'_1)$$

le sue componenti sono ottenute effettuando 2 *butterfly in place*.
In conclusione, per il calcolo di una DFT di lunghezza 4 sono stati effettuati 2 passi, in ciascuno dei quali è richiesto il calcolo di *2 butterfly*, il risultato di ciascuna *butterfly* può essere memorizzato sulle medesime componenti utilizzate per calcolare la *butterfly* stessa (algoritmo *in place*).

Nella Figura 5.11 sono mostrate le *butterfly* necessarie al calcolo di una DFT di lunghezza 4. ♣

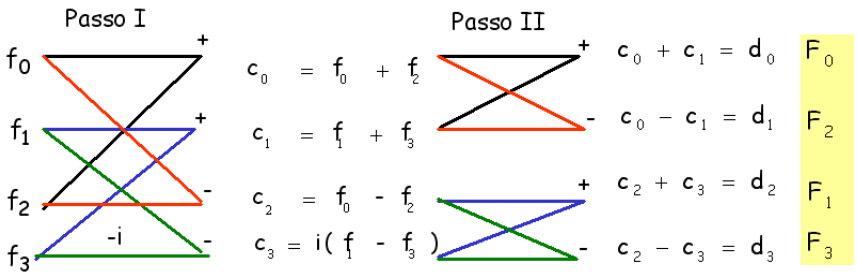

Figura 5.11: Schema di una DFT di lunghezza 4.

♣ **Esempio 5.23. DFT di lunghezza 8**
Supponiamo $N = 2^3 = 8$. Vogliamo effettuare la DFT di un vettore di lunghezza 8:

$$\underline{f} = (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7).$$

Dalla definizione di DFT si ha:

$$F_k = \sum_{j=0}^{7} f_j \omega_8^{-jk} \quad k = 0, 1, ..., 7$$

Seguendo l'algoritmo di Gentleman e Sande si ha che al **passo 1**, si calcolano i due vettori y e z, di lunghezza 4 e componenti:

$$\begin{array}{ll} y_0 = (f_0 + f_4) & z_0 = (f_0 - f_4) \\ y_1 = (f_1 + f_5) & z_1 = (f_1 - f_5) \\ y_2 = (f_2 + f_6) & z_2 = (f_2 - f_6) \\ y_3 = (f_3 + f_7) & z_3 = (f_3 - f_7) \end{array} \tag{5.52}$$

Le componenti omologhe dei due vettori y e z si ottengono rispettivamente come somma e differenza di coppie di componenti del vettore f. In particolare, effettuando $N/2 = 4$ *butterfly* costruite sulle componenti del vettore f che distano di $N/2 = 4$ unità, si ottengono le componenti dei vettori y e z. Introdotto il vettore

$$c = (y_0, y_1, y_2, y_3, z_0, z_1, z_2, z_3)$$

il primo passo dell'algoritmo porta alla costruzione di tale vettore memorizzato su f.
Al **passo 2** si calcolano 4 vettori di lunghezza 2 y', y'', z' e z'', di componenti:

$$\begin{array}{l} y'_0 = c_0 + c_2 \\ y'_1 = c_1 + c_3 \\ y''_0 = c_0 - c_2 \\ y''_1 = c_2 - c_3 \\ z'_0 = c_4 + ic_6 \\ z'_1 = c_5 + ic_7 \\ z''_0 = c_4 - ic_6 \\ z''_1 = c_5 - ic_7 \end{array} \tag{5.53}$$

Al secondo passo, i 4 vettori y', y'', z' e z'' si ottengono effettuando $N/2 = 4$ *butterfly* costruite utilizzando le componenti dei vettori y e z costruiti al passo precedente. In particolare, per ciascuna

butterfly si combinano le componenti che distando di $N/2^2 = 2$ unità. Introdotto il vettore

$$d = (y'_0, y'_1, y''_0, y''_1, z'_0, z'_1, z''_0, z''_1)$$

il secondo passo dell'algoritmo porta alla costruzione di tale vettore memorizzato su c.
Al **passo 3** si calcolano le quantità:

$$\begin{array}{l} y^{'''} = (d_0 + d_1) \\ y^{iv} = (d_0 - d_1) \\ y^{v} = (d_2 + id_3) \\ y^{vi} = (d_2 - id_3) \\ z^{'''} = (d_4 + \omega_8^2 d_5) \\ z^{iv} = (d_4 - \omega_8^2 d_5) \\ z^{v} = (d_6 + \omega_8^6 d_7) \\ z^{vi} = (d_6 - \omega_8^6 d_7) \end{array} \tag{5.54}$$

Al terzo passo, queste 8 quantità si ottengono effettuando $N/2 = 4$ *butterfly* costruite utilizzando le componenti dei vettori costruiti al passo precedente. In particolare, per ciascuna *butterfly* si combinano le componenti che distando di $N/2^4 = 1$ unità. Introdotto il vettore

$$e = (y''', y^{iv}, y^{v}, z^{iii}, z^{iv}, z^{v}, z^{vi})$$

il terzo passo dell'algoritmo porta alla costruzione di tale vettore memorizzato su d.

In generale, al passo $k = 1, 2, 3$ sono necessarie $4 = N/2$ butterfly costruite utilizzando componenti del vettore costruito al passo precedente che distano di $N/2^k = 2^3/2^k = 2^{3-k}$ unità. Tali componenti sono anche dette *nodi duali*.

Osserviamo che se indichiamo con c_k il vettore di $N = 8$ componenti costruito al passo k dell'algoritmo, la relazione che lega la coppia di nodi duali costruita al passo k con la medesima coppia relativa al passo $k-1$ è si esprime in maniera naturale attraverso la formula ricorrente:

$$\begin{aligned} c_k(r) &= c_{k-1}(r) + \omega_8^{q/2^k} c_{k-1}(r + 2^{3-k}) \quad r = 1, N/2 \\ c_k(r + 2^{3-k}) &= c_{k-1}(r) + \omega_8^{q'/2^k} c_{k-1}(r + 2^{3-k}), \quad r = 1, N/2 \end{aligned}$$

Poiché $q' = q + 2^{k-1}$, segue che $\omega_N^{q'/2^k} = -\omega_N^{q/2^k}$ e quindi per ciascuna *butterfly* è necessario valutare un unico esponenziale complesso.

Infine, esplicitando i calcoli ci si accorge che il vettore $\underline{e} = (e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7)$ é la DFT del vettore $\underline{F}$ dove le componenti sono disposte non nell'ordine naturale ma nell'ordine $(F_0, F_4, F_2, F_6, F_1, F_5, F_3, F_7)$ (*scrambled*) ovvero, del bit inverso. ♣

Lo schema generale dell'algoritmo FFT radix-2, nella versione di Gentleman e Sande, per calcolare le componenti del vettore F prevede di combinare a due a due le componenti del vettore costruito al passo precedente in modo da effettuare somme e differenze secondo lo schema *butterfly*. In particolare:

- passo 1: si calcolano 2^{m-1} *butterfly* ottenute combiando le componenti di f a distanza $p = N/2$, come mostrato nella Figura 5.12.

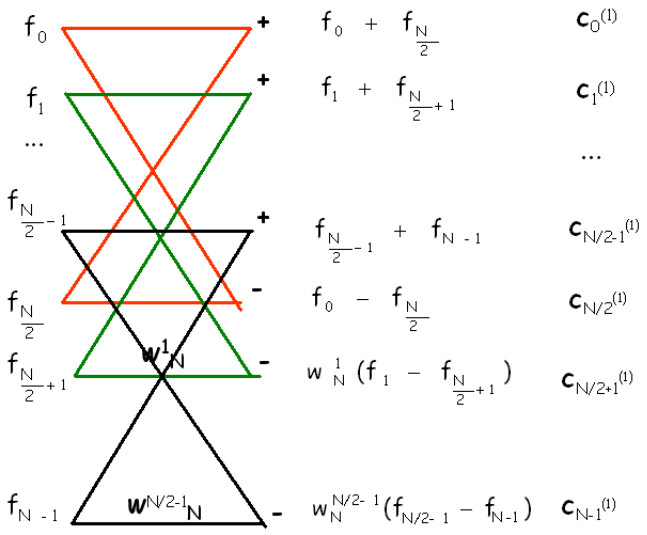

Figura 5.12: Schema del passo 1 di una DFT di lunghezza N.

- passo 2: si calcolano 2^{m-1} *butterfly* ottenute combinando le componenti di f a distanza $p = N/4$, come mostrato nella Figura 5.13.

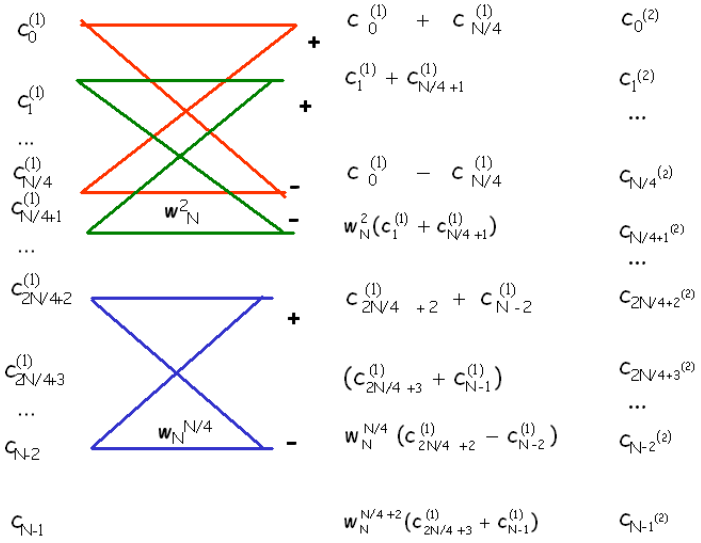

Figura 5.13: Schema del passo 2 di una DFT di lunghezza N.

- passo k: si calcolano 2^{m-1} *butterfly* ottenute combinando le componenti di f a distanza $p = N/2^k$, come mostrato nella Figura 5.14.

- ...

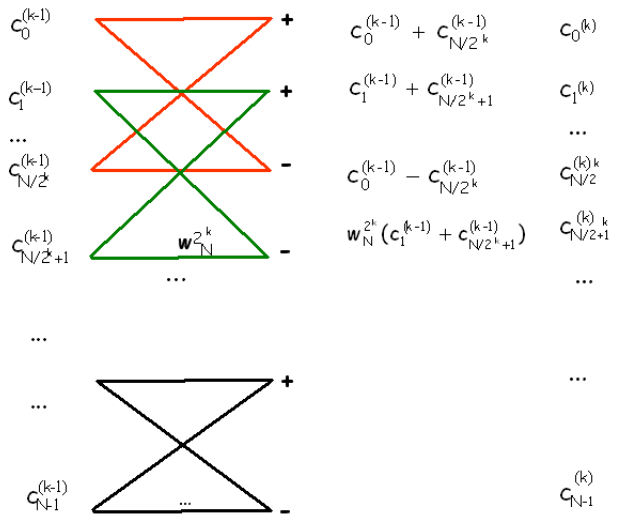

Figura 5.14: Schema del passo k di una DFT di lunghezza N.

- passo m: si calcolano 2^{m-1} *butterfly* ottenute combinando gli elementi di f a distanza $p = N/2^{m-1}$, come mostrato nella Figura 5.15.

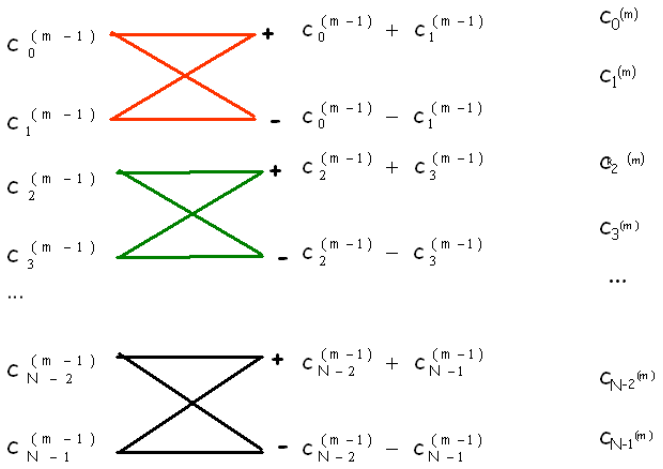

Figura 5.15: Schema del passo m di una DFT di lunghezza N.

Si nota che ad ogni passo ciascuna coppia di componenti ottenuta da un'operazione *butterfly*, può essere memorizzata sulle posizioni di memoria occupate dalle rispettive componenti utilizzate nell'operazione *butterfly*. Ciò è lecito perchè per ciascuna *butterfly*, la coppia di componenti coinvolte viene utilizzata esclusivamente per calcolare le componenti risultanti da quella butterfly. In altre parole, l'algoritmo può essere implementato *in place*, ovvero il vettore calcolato ad ogni passo può essere memorizzato sul vettore di input.
Il vettore c_k costruito al passo k dell'algoritmo si ottiene a partire dal vettore c_{k-1}

attraverso la formula ricorrente:

$$c_k(r) = c_{k-1}(r) + \omega_N^{q/2^k} c_{k-1}(r + 2^{m-k})$$
$$c_k(r + 2^{3-k}) = c_{k-1}(r) + \omega_N^{q'/2^k} c_{k-1}(r + 2^{m-k}) \quad (\omega_N^{q'/2^k} = \omega_N^{-q/2^k})$$

Il vettore costruito all'ultimo passo è la DFT del vettore F disposto nell'ordine *scrambled* (o anche del *bit inverso*).

L'algoritmo di Cooley e Tukey prevede, invece, una fase di *riordinamento* iniziale e una seconda fase di *calcolo* (in particolare, operazioni tipo butterfly tra componenti che distano di 1, poi di 2, poi di 4, poi di 8 componenti, e così via) sul vettore ordinato. In tal modo, questa versione produce il vettore risultante secondo l'ordine naturale.

Un aspetto interessante, da un punto di vista implementativo, è osservare che la decomposizione del vettore di input, nell'algoritmo di Cooley e Tukey, o del vettore DFT nella variante di Gentleman e Sande, si può effettuare utilizzando la rappresentazione binaria degli indici j e k. Infatti notiamo che, ad esempio, nell'algoritmo di Cooley e Tukey, separare le componenti di f con indice pari da quelle con indice dispari significa dimezzare l'indice $j = 0, \ldots, N-1$ e quindi per il numero intero j esiste una rappresentazione del tipo:

$$j = 2l + q, \quad j = 0, \ldots, \underbrace{N/2}_{2^{m-1}} - 1, \quad l, q = 0, 1$$

Analogamente, applicare questo ragionamento ai due sottovettori di lunghezza $N/2$, significa dimezzare l'indice l e quindi:

$$j = 2 \cdot \underbrace{(2r + s)}_{l} + q, \quad r = 0, \ldots, \underbrace{N/4}_{2^{m-2}} - 1, \quad q, s = 0, 1$$

e ancora, riapplicare questo ragionamento ai 4 vettori di lunghezza $N/4$, significa dimezzare r:

$$j = 2 \cdot (2 \cdot \underbrace{(2t + v)}_{r} + s) + q, \quad t = 0, \ldots, \underbrace{N/8}_{2^{m-3}} - 1, \quad v, s, q = 0, 1$$

Dopo $m-1$ passi si ha una rappresentazione di j del tipo:

$$j = 2^{m-1} \cdot j_{m-1} + 2^{m-2} \cdot j_{m-2} \ldots + j_0, \quad k = 0, \ldots, m-1, \quad j_k = 0, 1$$

Ovvero abbiamo rappresentato il numero intero j come numero binario. Applicando questo ragionamento agli indici j e k, la (5.36) diventa:

$$F(k) = \sum_{j_{m-1}=0}^{1} \sum_{j_{m-2}=0}^{1} \ldots \sum_{j_0=0}^{1} f(2^{m-1} \cdot j_{m-1} + 2^{m-2} \cdot j_{m-2} \ldots + j_0) \omega_N^{-jk}$$

Tenendo conto che

$$\omega_N^{-jk} = \omega_N^{-k(2^{m-1}\cdot j_{m-1}+2^{m-2}\cdot j_{m-2}\ldots+j_0)} = \omega_N^{-kj_{m-1}}\omega_{N/2}^{-kj_{m-2}}\cdots\omega_2^{-k_0 j_0}$$

segue:

$$\omega_N^{-jk} = \omega_N^{-kj_{m-1}}\cdot\omega_{2^{m-1}}^{-kj_{m-2}}\cdot\omega_{2^{m-2}}^{-kj_{m-3}}\cdots\omega_2^{-kj_0}$$

Inoltre, se:

$$k = 2^{m-1}\cdot k_{m-1} + 2^{m-2}\cdot k_{m-2}\ldots + k_0$$

si ha che:

$$\omega_2^{-kj_0} = \omega_2^{-j_0 k_{m-1}}$$

e quindi

$$F(k) = \sum_{j_{m-1}=0}^{1}\omega_N^{-kj_{m-1}}\sum_{j_{m-2}=0}^{1}\omega_{2^{m-1}}^{-kj_{m-2}}\cdots\sum_{j_0=0}^{1}f(2^{m-1}\cdot j_{m-1}+2^{m-2}\cdot j_{m-2}\ldots+j_0)\omega_2^{-j_0 k_{m-1}}$$

Fissato l'indice k, il calcolo di ciascuna componente del vettore DFT, espresso in questo modo, si puó interpretare come il calcolo di $m-1$ somme innestate, in cui ciascuna somma è una DFT di lunghezza 2. Questa è l'idea alla base dell'implementazione dell'algoritmo FFT radix-2 di Gentleman e Sande. Al contrario, partendo da somma più esterna, si hanno 2 DFT di lunghezza 2^{m-1} che corrispondono alle rimanenti $m-1$ sommatorie. Considerando le prime due sommatorie esterne, si hanno 4 DFT di lunghezza 2^{m-2} e così proseguendo verso l'interno, arriviamo al calcolo di 2^{m-1} DFT di lunghezza 2, ovvero alla versione dell'algoritmo FFT radix-2 proposta da Cooley e Tukey.

Da un punto di vista implementativo, entrambe le varianti si possono ottenere selezionando il valore (0/1) di un bit della rappresentazione binaria N (ad esempio, nella variante di Cooley e Tukey, il valore 0 del bit meno significativo nella rappresentazione binaria di j corrisponde all' indice pari, il valore 1 corrisponde all'indice dispari.). Questo significa che se consideriamo l'algoritmo di Cooley e Tukey e prima di iniziare i calcoli , si riordinano le componenti del vettore di input secondo l'ordine inverso (del bit reversal), ad ogni passo sarà necessario combinare sempre componenti che nel nuovo ordinamento si trovano in locazioni di memoria adiacenti. Inoltre, il vettore risultante dopo m passi è il vettore DFT ordinato secondo l'ordine naturale delle componenti. Nella variante di Gentleman e Sande, invece, tale riordinamneto iniziale non è necessario. Tale riordinamento dovrà essere applicato unicamente alla componenti del vettore DFT qualora si vogliano tali componenti nell'ordine naturale.

```
procedure fft2(input: f,N, out: f)
/# SCOPO: calcolo del vettore DFT di f
/# .....
     for k=1,m
/# ciclo sul numero di passi
        d := N/2^k;                          distanza nodi duali al passo k
        nesp := 2^(k-1);                     esponenziali diversi al passo k
        s := 0;
        for i=1,nesp                         ciclo sugli esponenziali diversi
         r := s;                             indice prima componente
           for l=0,d-1                       calcolo nodi duali
           r := l + s;                       indice primo elemento della coppia
           r1 := r + d;                      indice secondo elemento della coppia
           t := exp(z/2^k) * f(r1);
           f(r1) := f(r) - t;                calcolo coppia di nodi duali
           f(r) := f(r) + t;
           endfor l;
           s := s + 2d;                      salto
         endfor i;
      endfor k;
```

Procedura 5.1: Algoritmo di FFT radix-2: calcolo del vettore c_k, $k = 1, \ldots, m$.

5.4.5 Formulazione matriciale dell'algoritmo FFT radix-2

L'algoritmo FFT radix-2, nella versione di Gentleman e Sande, applicato ad un vettore di lunghezza N, produce la fattorizzazione della matrice di Fourier nel prodotto di $\log_2(N)$ matrici sparse e strutturate.

♣ **Esempio 5.24.** Sia $N = 2^1$; come nell'**esempio 5.21**, la DFT del vettore:

$$\underline{f} = (f_0, f_1)$$

è:

$$\begin{array}{l} F_0 = f_0 + f_1 \\ F_1 = f_0 - f_1 \end{array} \tag{5.55}$$

Se si considera la matrice di Fourier di dimensione 2:

$$W := A_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

si ha:

$$W \cdot f = F$$

La matrice A_1 è una matrice a blocchi:

$$A_1 = \begin{bmatrix} I_1 & I_1 \\ \Omega_2 & -\Omega_2 \end{bmatrix}$$

dove I_1 è la matrice identica di ordine 1 e Ω_2 è la matrice diagonale di ordine 1 del tipo:

$$\Omega_2 = diag(w_2^0) = diag(1)$$

♣

♣ **Esempio 5.25.** Sia $N = 2^2 = 4$; come nell'**esempio 5.22**, la DFT del vettore:

$$\underline{f} = (f_0, f_1, f_2, f_3)$$

è:

$$\begin{array}{l} F_0 = f_0 + f_1 + f_2 + f_3 \\ F_1 = f_0 - if_1 - f_2 + if_3 \\ F_2 = f_0 - f_1 + f_2 - f_3 \\ F_3 = f_0 + if_1 - f_2 - if_3 \end{array} \tag{5.56}$$

Se si definisce la matrice di Fourier di dimensione 4:

$$W := \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

allora:

$$W \cdot f = F$$

Nell'**esempio 5.22** si è visto anche che, per calcolare le somme (5.56) è necessario:

- al **passo 1** calcolare le componenti del vettore c:

$$\begin{array}{ll} c_0 = (f_0 + f_2) & c_2 = (f_0 - f_2) \\ c_1 = (f_1 + f_3) & c_3 = -i(f_1 - f_3) \end{array} \tag{5.57}$$

Introdotta la matrice:

$$A_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & -i & 0 & i \end{bmatrix},$$

le operazioni (5.57) corrispondono al prodotto matrice per vettore:

$$A_1 \cdot f = c$$

La matrice A_1 è una matrice strutturata a blocchi, del tipo:

$$A_1 = \begin{bmatrix} I_2 & I_2 \\ \Omega_4 & -\Omega_4 \end{bmatrix}$$

essendo I_2 una matrice identica di ordine 2 e Ω_4 una matrice diagonale di ordine 2 definita come

$$\Omega_4 = diag(w_4^0, w_4^1) = diag(1, w_4^1).$$

Se si pone:

$$B_1 = A_1$$

risulta:

$$A_1 = I_1 \otimes B_1$$

dove il simbolo $\otimes$ denota il prodotto tensoriale di due matrici[25] e I_1 la matrice identica di ordine 1.

- Al **passo 2** calcolare le componenti del vettore d:

$$\begin{array}{ll} d_0 = (c_0 + c_1) & d_2 = (c_2 + c_3) \\ d_1 = (c_0 - c_1) & d_3 = (c_2 - c_3) \end{array} \tag{5.58}$$

Introdotta la matrice:

$$A_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

le operazioni (5.58) corrispondono al prodotto matrice per vettore:

$$A_2 \cdot c = d$$

La matrice A_2 è una matrice strutturata a blocchi, del tipo:

$$A_2 = \begin{bmatrix} I_1 & I_1 & 0 & 0 \\ \Omega_2 & -\Omega_2 & 0 & 0 \\ 0 & 0 & I_1 & I_1 \\ 0 & 0 & \Omega_2 & -\Omega_2 \end{bmatrix}$$

[25] Assegnate due matrici $A = (a_{i,j})_{i,j=1,...n} \in \Re^{n \times n}$ e $B = (b_{i,j})_{i,j=1,...m} \in \Re^{m \times m}$ il prodotto tensoriale di A e B è una matrice $C \in \Re^{mn \times mn}$ del tipo:

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & ... & a_{1n}B \\ ... & ... & ... & ... \\ a_{n1}B & a_{n2}B & ... & a_{nn}B \end{bmatrix}$$

essendo I_1 una matrice identica di ordine 1 e

$$\Omega_2 = diag(w_2^0) = diag(1)$$

una matrice diagonale di ordine 1. Se si pone:

$$B_2 = \begin{bmatrix} I_1 & I_1 \\ \Omega_2 & -\Omega_2 \end{bmatrix}$$

risulta che:

$$A_2 = I_2 \otimes B_2,$$

dove I_2 è la matrice identica di ordine due.

In conclusione, la DFT di un vettore lunghezza 4 si ottiene effettuando due prodotti matrice-vettore:

1. $A_1 \cdot f = c$;
2. $A_2 \cdot c = d$.

essendo A_1 ed A_2 matrici strutturate a blocchi.
Si osserva, inoltre, che se:

$$W^* := A_2 \cdot A_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \\ 1 & i & -1 & -i \end{bmatrix}$$

W^* coincide con la matrice di Fourier W *a meno di uno scambio di righe.* In particolare, detta P la matrice di permutazione:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ottenuta dalla matrice identica I_4 di ordine 4 scambiando la seconda e la terza riga, si ha:

$$W^* \cdot P = W$$

In conclusione, l'algoritmo FFT radix-2 applicato ad un vettore di lunghezza 4 produce la fattorizzazione di W nel prodotto di due matrici strutturate a blocchi:

$$W = W^* \cdot P = A_2 \cdot A_1 \cdot P$$

da cui:

$$Wf = F \Leftrightarrow A_2 \cdot A_1 \cdot P\ f = F$$

Si osserva che la permutazione della seconda e terza riga, definita attraverso il prodotto a destra della matrice di Fourier W per la matrice di permutazione P, corrisponde di fatto alla permutazione delle componenti di f indotta dal bit reversal.

♣

In generale, per effettuare la DFT di un vettore di lunghezza $N = 2^m$:

$$\underline{f} = (f_0, f_1, ..., f_{2^m})$$

sono necessari m passi in ciascuno dei quali si effettua un prodotto matrice per vettore. In particolare:

- al **passo 1**, introdotta la matrice:

$$A_1 = I_1 \otimes B_1$$

con:

$$B_1 = \begin{bmatrix} I_{2^{m-1}} & I_{2^{m-1}} \\ \Omega_{2^m} & -\Omega_{2^m} \end{bmatrix}$$

dove $I_{2^{m-1}}$ ed

$$\Omega_{2^m} = (w^0_{2^m}, w^1_{2^m}, ..., w^{2^{m-1}}_{2^m})$$

sono rispettivamente la matrice identica e una matrice diagonale di ordine 2^{m-1} si calcola il vettore:

$$c^{(1)} = A_1 \cdot f$$

- al **passo 2**, introdotta la matrice:

$$A_2 = I_2 \otimes B_1$$

con

$$B_2 = \begin{bmatrix} I_{2^{m-2}} & I_{2^{m-2}} \\ \Omega_{2^{m-1}} & -\Omega_{2^{m-1}} \end{bmatrix}$$

dove $I_{2^{m-2}}$ ed

$$\Omega_{2^{m-1}} = (w^0_{2^{m-1}}, w^1_{2^{m-1}}, ..., w^{2^{m-2}}_{2^{m-1}})$$

sono rispettivamente la matrice identica e una matrice diagonale di ordine 2^{m-2}, si calcola il vettore:

$$c^{(2)} = A_2 \cdot c^{(1)}$$

- al **passo k**, introdotta la matrice:

$$A_k = I_{2^{k-1}} \otimes B_k$$

con

$$B_k = \begin{bmatrix} I_{2^{m-k}} & I_{2^{m-k}} \\ \Omega_{2^{m-k+1}} & -\Omega_{2^{m-k+1}} \end{bmatrix}$$

dove $I_{2^{m-k}}$ ed

$$\Omega_{2^{m-k}} = (w^0_{2^{m-k}}, w^1_{2^{m-k}}, ..., w^{2^{m-k+1}}_{2^{m-k}})$$

sono rispettivamente la matrice identica e una matrice diagonale di ordine 2^{m-k}, si calcola il vettore:

$$c^{(k)} = A_k \cdot c^{(k-1)}$$

In conclusione, la DFT di un vettore lunghezza $N = 2^m$ si ottiene effettuando m prodotti matrice per vettore:

1. $A_1 \cdot f = c^{(1)}$;

2. $A_2 \cdot c^{(1)} = c^{(2)}$;

   ...

m. $A_m \cdot c^{(m-1)} = c^{(m)}$

essendo A_1, A_2,..., A_m matrici strutturate a blocchi. Se si denota con W^* la matrice che si ottiene effettuando il prodotto delle matrici A_1, A_2, ..., A_m ovvero:

$$W^* = A_m \cdot A_{m-1} ... A_1$$

tale matrice coincide con la matrice di Fourier W "a meno di uno scambio di righe", ovvero:

$$W = W^* \cdot P$$

essendo P una matrice di permutazione di ordine 2^m costruita ordinando secondo la relazione del bit inverso le righe della matrice identica I_{2^m}.
In conclusione, l'algoritmo FFT radix-2, applicato ad un vettore di lunghezza $N = 2^m$, fattorizza la matrice di Fourier W nel prodotto di m matrici, ovvero:

$$W = W^* \cdot P = A_m \cdot A_{m-1} \cdots A_1 \cdot P \tag{5.59}$$

da cui:

$$W f = F \Leftrightarrow A_m \cdot A_{m-1} \cdot A_1 \cdots P \; f = F \tag{5.60}$$

Concludiamo questo paragrafo osservando che la matrice di Fourier W gode di una interessante proprietà riguardante l'andamento dei valori singolari. Si dimostra, infatti, che partizionando la matrice di Fourier in p^2 blocchi, di dimensione n/p, tali sottomatrici hanno valori singolari compresi tra zero e uno, e solo una porzione di questi sono vicini all'unità. In altri termini, effettuando una decomposizione in valori singolari "troncata" di siffatti blocchi, si perviene ad una formulazione a blocchi dell'algoritmo FFT radix-2 che, sebbene fornisca il vettore DFT corretto a meno di una prefissata tolleranza, avendo assunto numericamente nulli i valori singolari al di sotto di tale tolleranza, si presta ad una implementazione in ambiente di calcolo parallelo e distribuito più efficiente di quelle basate sugli algoritmi di Cooley e Tukey o Gentleman e Sande [26].

5.4.6 Stabilità dell'algortimo FFT radix-2

Sia $\Im$ un sistema aritmetico a precisione finita e sia $F = W \cdot f$, il vettore DFT di f. Indicato con $\widehat{F}$ il valore di F, calcolato in $\Im$ utilizzando l'algoritmo FFT, il teorema seguente, [17], mostra che la propagazione dell'errore nel sistema aritmetico $\Im$ cresce linearmente con m, ovvero che l'algoritmo FFT è *stabile* nel senso della f.e.a.

[26] [7]

Teorema 5.4.1. *Sia f un vettore di lunghezza $N = 2^m$ esattamente rappresentabile in $\Im$ ovvero $f = fl(f) = \widehat{f}$. Supponiamo che gli esponenziali complessi ω_k siano precalcolati e che su di essi sia stato commesso un errore $|\epsilon_{jk}|$ tale che:*

$$\hat{\omega}_j^k = fl(\omega_j^k) = \omega_j^k + \epsilon_{jk} \quad |\epsilon_{jk}| \leq u \tag{5.61}$$

essendo u la massima accuratezza relativa di $\Im$.
Allora:

$$\frac{||F - \widehat{F}||_2}{||F||_2} \leq \frac{m\eta}{1 - m\eta}, \quad \eta := \mu + \gamma_4(\sqrt{2} + \mu) \tag{5.62}$$

essendo $\gamma_4 = \frac{4u}{1-4u}$.

Dimostrazione Si osserva che per le matrici in (5.60):

$$||A_k||_2 = \sqrt{2} \tag{5.63}$$

e che:

$$|| \ |A_k| \ ||_2 = 2 = \sqrt{2}||A_k||_2$$

Denotata con $\widehat{A}_k$ la matrice definita dai valori degli esponenziali complessi calcolati $\widehat{\omega}_k^j$. Allora:

$$\widehat{F} = fl(\widehat{A}_m....\widehat{A}_1) = (\widehat{A}_m + \Delta\widehat{A}_m)...(\hat{A}_1 + \Delta\widehat{A}_1) \cdot P_r \cdot f \tag{5.64}$$

dove con P_r si indica una matrice di permutazione di ordine r. Poiché ciascuna matrice A_k ha solo due elementi non nulli per riga ed inoltre ciascun elemento è un numero complesso[27] si ha:

$$|\Delta\widehat{A}_k| \leq \gamma_4|\widehat{A_k}|$$

per cui:

$$||\Delta\widehat{A}_k||_2 \leq || \ |\Delta\hat{A}_k| \ ||_2 \leq \gamma_4|| \ |\widehat{A_k}| \ ||_2$$

Dall'ipotesi (5.61) segue che:

$$\widehat{A}_k = A_k + \Delta A_k \quad ||\Delta A_k|| \leq \sqrt{2}\mu = \mu||A_k||_2 \tag{5.65}$$

Utilizzando la (5.63) e la (5.65) si ottiene che:

$$|| \ |\widehat{A}_k| \ ||_2 \leq || \ |A_k| \ ||_2 + || \ |\delta A_k| \ ||_2 \leq (\sqrt{2} + \mu)||A_k||_2. \tag{5.66}$$

La (5.64), mediante la (5.65) si può scrivere come:

$$\hat{F} = (A_m + \Delta A_m + \Delta\widehat{A}_m)...(A_1 + \Delta A_1 + \Delta\widehat{A}_1) \cdot P_r \cdot f \tag{5.67}$$

Posto:

$$E_k = \Delta A_k + \Delta\widehat{A}_k \quad k = 1, ..., m$$

[27]Utilizzando un modello standard di aritmetica a precisione finita [17] se x ed y sono numeri complessi esattamente rappresentabili si assume che:

$$\begin{array}{ll} fl(x \pm y) = (x \pm y)(1 + \delta) & |\delta| \leq u \\ fl(xy) = (xy)(1 + \delta) & |\delta| \leq \sqrt{2}\frac{2u}{1-2nu} \\ fl(x/y) = (x/y)(1 + \delta) & |\delta| \leq \sqrt{2}\frac{4u}{1-4nu} \end{array}$$

si ottiene:

$$\hat{F} = (A_m + E_m)...(A_1 + E_1) \cdot P_r \cdot f \tag{5.68}$$

e dalla (5.65) e (5.66) si ha:

$$||E_k||_2 \leq (\mu + \gamma_4(\sqrt{2} + \mu))||A_k||_2 = \eta||A_k||_2.$$

Applicando il Lemma 5.4.1[28] si trova che:

$$||F - \widehat{F}||_2 \leq [(1+\eta)^m - 1]||A_m||_2...||A_1||_2||P_r||_2||f||_2 \leq$$

$$\frac{m\eta}{1-m\eta} 2^{m/2}||f||_2$$

poiché:

$$||f||_2 = n^{-1/}||F||_2 =^{-m/2} ||F||_2$$

segue la tesi. ■

5.5 Software matematico per la FFT

Esistono numerose implementazioni degli algoritmi FFT, disponibili sia come singoli elementi di software sia come librerie che come software proprietari specializzati per tipologie di processori . La maggior parte di questi software sono scritti in linguaggio C/C++, altri in Fortran 77/90. Le prime implementazioni risalgono al 1969 [29], l'ultima è del 2009 (`FFTW 3.2.1` [9]). Il sito netlib consente di accedere alle più diffuse librerie di software matematico tra cui, ad esempio, `FFTPACK` [8], `MPFUN` [20], `NAPACK` [23] e `SCILIB` [25].

Una delle motivazioni per cui esitono così tanti elementi di software che implementano l'algoritmo FFT nasce dalla necessità di specializzare l'algoritmo FFT in base alle differenti caratteristiche dell'architettura di calcolo e del problema (reale, complesso, simmetrico, hermitiano, radix-r, a radici miste, etc.) al fine di migliorarne le prestazioni. Una rivoluzione nello sviluppo di software matematico efficiente per il calcolo della FFT

[28]

Lemma 5.4.1. *Se le matrici $X_j + \Delta X_j \in \Re^{n\times n}$ soddisfano la relazione $||\Delta X_j|| \leq \delta_j||X_j||$ per ogni j in una qualsiasi norma matriciale, allora:*

$$\left|\left|\prod_{j=0}^{m} X_j + \Delta X_j - \prod_{j=0}^{m} X_j\right|\right| \leq \left(\prod_{j=0}^{m}(1+\delta_j) - 1\right)\prod_{j=0}^{m} X_j$$

[29][27]

si è avuta nel 1998, con il software *Fast Fourier Transform in the West* (`FFTW`)[30]. Il principio alla base del software `FFTW` è quello di *ottenere le massime prestazioni su una qualsiasi architettura e per un qualsiasi problema* (portabilità delle prestazioni) *attraverso una combinazione dinamica di un prefissato insieme di algoritmi elementari.* Ciascuna routine del pacchetto software `FFTW` è una composizione di componenti elementari, detti `codelets`; ciascun `codelet` rappresenta l'implementazione di un algoritmo FFT di base, come ad esempio, il calcolo della DFT per $N = 2, 3, 5, 7, \ldots$, l'algoritmo di Cooley e Tukey, l'algoritmo di Gentleman e Sande, etc.. Il software `FFTW` contiene `codelets` creati automaticamente, in fase di installazione della libreria, da un compilatore (*genfft*) ed utilizzati da un programma (detto `plan`) che manda in esecuzione i `codelets` necessari per il calcolo della DFT richiesta dall'utente. Il programma `plan` viene creato dal `planner`, il cui ruolo è quello di *prendere in input il problema fornito dall'utente e scegliere, attraverso tecniche automatiche e dinamiche, la combinazione migliore e più efficiente per risolvere quel problema*. Tale combinazione viene scelta in base alle prestazioni che i *codelets* raggiungono nell'ambiente di calcolo considerato. La scelta dei *codelets* da utilizzare viene fatta percorrendo l'albero ottenuto dalla fattorizzazione del parametro N.

L'aspetto innovativo della `FFTW` risiede proprio nella presenza dei moduli software, il `plan` e il `planner`, con cui interagisce l'utente. L'obiettivo è coniugare portabilità ed efficienza utilizzando tecniche di generazione automatica di codice (ovvero software che produce a sua volta del software). Tale approccio caratterizza la metodologia `AEOS` *(Automated Empirical Optimization of Software)* alla base dello sviluppo automatico di software efficiente e ottimizzato sulle architetture avanzate.

Nel seguito si descrive in breve un frammento di codice scritto in C, che utilizza codelets di `FFTW`.

[30] [10]

```
#include <fftw3.h>
...
{ fftw_complex *in, *out;
fftw_plan p; /* p contiene le specifiche del problema
/* allocazione delle variabili di input e di output */
in = fftw_malloc(sizeof(fftw_complex)*n);
out =fftw_malloc(sizeof(fftw_complex)*n);
/* Generazione del plan */
p= fftw_plan_dft_1d(n,in,out,FFTW_FORWARD,FFTW_ESTIMATE);
...
/* Esecuzione del plan (executor) */
fftw_execute(p);
...
/* deallocazioni della memoria occupata */
fftw_destroy_plan(p);
fftw_free(in);
fftw_free(out);}
```

Il `plan` definisce l'ordine con il quale devono essere eseguiti i *codelets*. Il plan ha una struttura ad albero corrispondente alla specifica fattorizzazione del parametro N, e può essere memorizzato per successivi impieghi.
Le istruzioni contenute nel `plan` vengono eseguite da un opportuno programma detto `executor`. L'`executor` è la componente software della libreria `FFTW` che si occupa del calcolo effettivo della DFT implementando le istruzioni del plan. L'utente deve solo selezionare il tipo di `plan` per il calcolo di FFT m-dimensionali. In particolare `FFTW` mette a disposizione:

`fftw_plan_dft_1d()`	DFT monodimensionale
`fftw_plan_dft_2d()`	DFT bidimensionale
`fftw_plan_dft_3d()`	DFT tridimensionale
`fftw_plan_dft()`	DFT m-dimensionale

Infine l'header file:

```
#include <fftw3.h>
```

contiene informazioni necessarie ad una corretta compilazione, e utilizza, tra l'altro, routine per l'allocazione ed il rilascio della memoria:

```
fftw_malloc(...)
fftw_free(...)
fftw_destroy_plan(...)
```

5.6 MATLAB e la Trasformata discreta di Fourier

In ambiente MATLAB le routine per la trasformata di Fourier sono collezionate nella directory datafun. Esse sono:

```
fft         - Calcola la Trasformata discreta di Fourier (DFT).
fft2        - Calcola la DFT bidimensionale.
fftn        - Calcola la DFT N-dimensionale.
ifft        - Calcola la DFT inversa (IDFT).
ifft2       - Calcola la IDFT bidimensionale.
ifftn       - Calcola la IDFT N-dimensionale.
fftshift    - Esegue uno shift delle componenti di una DFT.
ifftshift   - Funzione inversa di fftshift.
```

Ciascuna delle routine dedicate al calcolo della DFT (`fft, fft2, fftn, ifft, ifft2, ifftn`) si basa sulla libreria **FFTW**. Inoltre MATLAB mette a disposizione dell'utente anche la funzione `fftw`, che costituisce un'interfaccia alla libreria FFTW.

Dal *prompt* dei comandi è possibile richiamare le funzioni con le relative opzioni; ad esempio con:

```
>> Y = fft(X)
>> Y = fft(X,n)
```

- se X è una matrice, `fft` fornisce la Trasformata di Fourier di ciascuna colonna;
- se X è un array multidimensionale, `fft` lavora lungo la prima dimensione diversa da uno.

Inoltre, con:

```
>> Y = fft(X,[],dim)
>> Y = fft(X,n,dim)
```

si applica l'operatore DFT lungo la dimensione *dim*. Analogamente, con:

```
>> y = ifft(X)
>> y = ifft(X,n)
```

si calcola la *DFT inversa* del vettore X.

♣ **Esempio 5.26.** [31] Si consideri una funzione $x = x(t)$ composta da due funzioni sinusoidali, una di frequenza 50 Hz ed ampiezza 0.7 e l'altra di frequenza 120 Hz ed ampiezza 1; si assuma, inoltre, che la funzione x sia affetta da rumore casuale, $\eta(t)$, distribuito secondo distribuzione normale di media zero (Fig. 5.16).

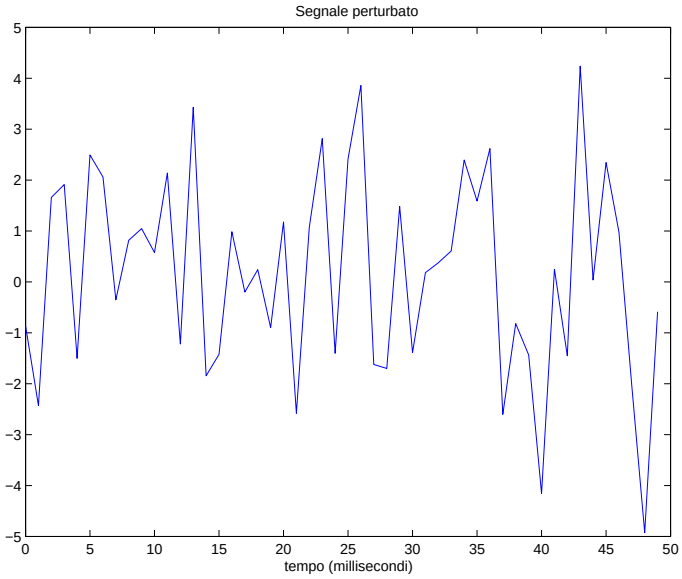

Figura 5.16: Grafico della funzione $y(t) = 0.7\sin(2\pi 50t) + \sin(2\pi 120t) + \eta(t)$.

```
>>Fs = 1000;      % Campionamento delle frequenze
>>T = 1/Fs;       % Campionamento del dominio del tempo
>>L = 1000;       % Lunghezza del segnale
>>t = (0:L-1)*T;  % Vettore del tempo
>>% Somma di una sinusoide a 50 Hz  ed una sinusoide a 120 Hz
>>x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
>>y = x + 2*randn(size(t));  % Aggiunta di rumore
>>plot(Fs*t(1:50),y(1:50))
>>title('Segnale perturbato')
>>xlabel('tempo (millisecondi)')
```

La funzione `randn(N)` genera una matrice di numeri casuali, $N \times N$, appartenenti ad una distribuzione normale di media 0 e varianza 1.
La trasformazione nel dominio delle frequenze è realizzata attraverso il calcolo della DFT del segnale y, mediante la funzione che implementa l'algoritmo FFT:

[31] `http://www.mathworks.com/`

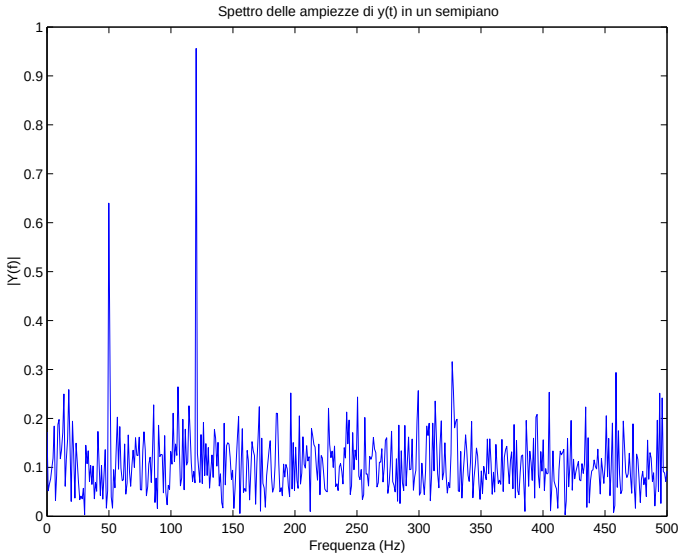

Figura 5.17: Frequenze ed ampiezze della funzione $y(t)$.

```
>>NFFT = 2^nextpow2(L);
>>Y = fft(y,NFFT)/L;
>>%costruzione di un vettore di frequenze equispaziate in [0,Fs/2];
>>f=(Fs/2)*linspace(0,1,NFFT/2);
>>plot(f,2*abs(Y(1:NFFT/2))); %Grafico dello spettro delle ampiezze
>>title('Spettro delle ampiezze di y(t) in un semipiano')
>>xlabel('Frequenza (Hz)')
>>ylabel('|Y(f)|')
```

Poiché la routine `fft` risulta *più veloce* per potenze di due, si calcola il primo p tale che $2^p \geq |L|$ ($2^{10} = 1024$) e si assegna `NFFT` come dimensione alla function `fft` [32].
Il motivo principale per cui le ampiezze non risultano esattamente 0.7 e 1 (Fig.5.17) è la presenza del rumore. Ripetendo le istruzioni (incluso il calcolo del vettore y) si otterranno approssimazioni diverse di 0.7 e di 1. Un ulteriore motivo è la lunghezza finita del segnale; incrementando il valore di L si otterranno approssimazioni mediamente più accurate.

♣

5.6.1 Problemi da risolvere con le routine di MATLAB

Problema 1 Assegnato un vettore x di dimensione $n \geq 1000$, ad esempio con la funzione `MATLAB rand`, $x = rand(n, 1)$, calcolare la DFT di x:

$$y = \texttt{fft}(x)$$

[32] Si osserva che $\texttt{fft}(X, N)$ calcola una DFT di un vettore X di lunghezza N, inserendo zeri se X ha meno di N componenti e troncando se ne ha di più.

Applicare al vettore y la funzione `ifft`:

$$xinv = \texttt{ifft}(\texttt{fft}(x))$$

e stimare l'ordine di grandezza dell'errore:

$$E = \frac{\|x - xinv\|}{\|x\|}$$

Problema 2 Prodotto di due polinomi

Siano

$$a = (1, 5, 17, 0, 0) = (a_0, a_1, ...a_{r+s})$$

e

$$b = (11, 6, -4, 0, 0) = (b_0, b_1, ...b_{r+s})$$

i vettori dei coefficienti, ordinati secondo le potenze crescenti, di due polinomi di grado r e s rispettivamente. A partire dai vettori:

```
>>A=(23,-11.2082+14.7476i,2.082-13.229i,2.2082+13.229i,-11.2082-14.7476i)
>>B=(13,16.0902+3.3552i,4.9098+7.3309i,4.9098-7.3309i,16.0902-3.3552i)
```

con $A = DFT[a]$ e $B = DFT[b]$,

1. calcolare

   ```
   >> ifft(A.*B)=ifft(fft(a).*fft(b))
   ```

2. verificare che il vettore ottenuto coincide con il risultato fornito dalla funzione `conv` di `MATLAB` che, applicata ad a e b, ne calcola il prodotto di convoluzione:

   ```
   >> conv(a,b)
   ```

3. Quante DFT occorrono per il calcolo dei coefficienti del prodotto di due polinomi?
4. Confrontare la complessità di tempo richiesta dal calcolo dei coefficienti di un polinomio prodotto mediante DFT con il numero di operazioni necessarie (asintoticamente) per il calcolo diretto dei coefficienti dello stesso polinomio.

Problema 3 Prodotto matrice circolante per vettore

Si consideri la matrice circolante C generata dal vettore $v = (5, 2, 7, 9, 4, 1, 2, 3)$:

$$C = \begin{bmatrix} 5 & 2 & 7 & 9 & 4 & 1 & 2 & 3 \\ 3 & 5 & 2 & 7 & 9 & 4 & 1 & 2 \\ 2 & 3 & 5 & 2 & 7 & 9 & 4 & 1 \\ 1 & 2 & 3 & 5 & 2 & 7 & 9 & 4 \\ 4 & 1 & 2 & 3 & 5 & 2 & 7 & 9 \\ 9 & 4 & 1 & 2 & 3 & 5 & 2 & 7 \\ 7 & 9 & 4 & 1 & 2 & 3 & 5 & 2 \\ 2 & 7 & 9 & 4 & 1 & 2 & 3 & 5 \end{bmatrix}$$

ed il vettore $x = (1, 2, 3, 4, 5, 6, 7, 8)$. Utilizzare la DFT per il calcolo del prodotto matrice per vettore

$$C \cdot x' = y'$$

1. Calcolare, utilizzando `MATLAB` , il prodotto $y' = Cx'$. Verificare che

   ```
   >>  fft(c).*fft(x)=fft(y)
   ```

   ovvero che attraverso il calcolo di 3 DFT si perviene al vettore soluzione y:

   ```
   >> y=ifft(fft(c).*fft(x))
   ```

2. Confrontare la complessità di tempo richiesta dal calcolo delle 3 DFT con quella del prodotto matrice vettore eseguito righe per colonne.

5.7 Esercizi

5.7.1 Quesiti

Quesito 1 L'algoritmo FFT (Fast Fourier transform) calcola sia la trasformata discreta di Fourier che la sua inversa con la stessa complessità di tempo?

Quesito 2 Fornire due applicazioni della Trasformata Discreta di Fourier (DFT).

Quesito 3 La DFT di un vettore di lunghezza n è legata all'interpolazione trigonometrica mediante un insieme di n funzioni trigonometriche di base.

1. Perché il calcolo della DFT non richiede la risoluzione di un sistema lineare per il calcolo dei coefficienti delle funzioni di base?
2. Qual è il caso peggiore, dal punto di vista della complessità computazionale, per il calcolo della DFT? Per quale valore di n si verifica?
3. Qual è il caso migliore, dal punto di vista della complessità computazionale, per il calcolo della DFT? Per quale valore di n si verifica?
4. Spiegare il motivo della differenza tra le complessità di tempo nei due casi.

Quesito 4 Perché si utilizza l'algoritmo FFT per calcolare il prodotto di convoluzione di due vettori?

5.7.2 Esercizi numerici

Esercizio 1 Verificare, attraverso opportuni esempi, che l'operatore DFT è lineare, ovvero verificare che, se f, $g \in \mathbb{C}^N$ sono vettori di numeri complessi di lunghezza N e se $\alpha, \beta \in \Re$, si ha:

$$DFT[\alpha f + \beta g] = \alpha DFT[f] + \beta DFT[g]$$

al variare dei vettori complessi f e g e degli scalari α e β.

Esercizio 2 Verificare, attraverso opportuni esempi, che, se $f = (f_0, f_1, \ldots, f_{N-1}) \in \mathbb{C}^N$ è un vettore di numeri complessi di lunghezza N e se

$$f_{sim} = (f_0, f_{N-1}, f_{N-2}, ..., f_1)$$

è il suo vettore simmetrico, si ha:

$$\frac{1}{N} DFT[DFT[f]] = f_{sim}.$$

Esercizio 3 Verificare, attraverso opportuni esempi, le seguenti proprietà dell'operatore DFT:

1. Se $f \in \Re^N$ è un vettore di numeri reali, allora $F = DFT[f]$ è un vettore hermitiano simmetrico, ovvero tale che le sue componenti, a meno della prima, soddisfino la condizione:

   $$F_k = \bar{F}_{N-k} \quad k = 1, ..., N/2$$

   avendo indicato con $\bar{F}_{N-k}$ il numero complesso coniugato di F_{N-k}.
2. Se $f \in \mathbb{C}^N$ è un vettore hermitiano simmetrico, ovvero $f_k = \bar{f}_{N-k}$, allora $F = DFT[f]$ è un vettore di numeri reali.
3. Se g è un vettore le cui componenti sono numeri immaginari puri (parte reale nulla) allora $G = DFT[g]$ è un vettore hermitiano antisimmetrico, ovvero:

   $$G_k = -\bar{G}_{N-k} \quad k = 1, ..., N/2.$$

4. Se g è un vettore hermitiano antisimmetrico, ovvero $g_k = \bar{g}_{N-k}$, allora $G = DFT[g]$ è un vettore le cui componenti sono numeri immaginari.

Si considerino, ad esempio, i vettori

$$f = (1, 2, 3, 4, 5, 6) \quad \text{e} \quad g = (i, -2i, 5i, -4i, 3i, 6i).$$

Si calcolino $DFT[f]$ e $DFT[g]$ e si effettuino opportune considerazioni sui risultati.

Esercizio 4 Per un fissato vettore, x, perché la prima componente della sua DFT, y_0, è sempre uguale alla somma delle componenti di x?

Esercizio 5 La matrice di Fourier F_n, definita da: $\{F_n\}_{mk} = w^{mk}$ è simmetrica. Per quali valori di n, se esistono, F_n è Hermitiana?

Esercizio 6 Se y è la DFT di un vettore reale, x, di lunghezza n, dove n è una potenza di due, provare che y_0 e $y_{n/2}$ devono essere reali.

Esercizio 7 Se $y = DFT(x)$, dimostrare che

$$x = \frac{1}{n} \overline{DFT(\overline{y})}$$

dove per $\overline{y}$ si intende il vettore le cui componenti sono i complessi coniugati di ciascuna delle componenti di y.

5.7.3 Problemi da risolvere con il calcolatore

Problema 1 Assegnate due matrici circolanti C_1 e C_2, progettare ed implementare una procedura per il calcolo del prodotto righe per colonne di C_1 per C_2 utilizzando la FFT.
Discutere la riduzione della complessità computazionale.

Problema 2 Descrivere lo schema ad albero in base al quale si esprime il calcolo della FFT mixed-radix nel caso $N = 15$.

Problema 3 Assegnato il vettore di numeri complessi:

$$h = (36, -4-9.6569i, -4-4i, -4-1.6569i-4, -4+1.6569i, -4+4i, -4+9.6569i)$$

si calcoli, utilizzando la libreria **fftw3**, la DFT $u := fft(h)$ del vettore h.
(a) Quale proprietà ha il vettore u?
(b) Indicato con $g = 2h$ il vettore che si ottiene da h moltiplicando le sue componenti per due, calcolare il vettore $v := fft(g)$ DFT del vettore g.
(c) Individuare il legame che sussiste tra u e v ed il calcolo della FFT mixed-radix nel caso $N = 15$.

Problema 4 Assegnato il vettore di numeri reali:

$$h = (1, 2, 3, 4, 5, 6, 7, 8)$$

si calcoli, utilizzando la libreria **fftw3**, la DFT $u := fft(h)$ del vettore h.
(a) Quale proprietà ha il vettore u?
(b) Calcolare il vettore $v := fft(u)$ DFT del vettore u.
(c) Individuare il legame che sussiste tra h e v.

Problema 5 Assegnati i vettori di numeri reali:

$$u = (-1, 3, 4, 8)$$

$$v = (11, -4, 7, 9)$$

utilizzando una opportuna routine della libreria **fftw3**
(a) si calcolino i vettori $r = fft(u)$ ed $s = fft(v)$.
(b) Sia $G = (37, -1+18i, 5, -1-18i)$ si calcoli il vettore $g := ifft(G)$ trasformata inversa di G.
(c) Individuare il legame che sussiste tra g ed i vettori u e v.

Problema 6 Assegnato il vettore di numeri reali:

$$u = (42, 32, 4, 8)$$

utilizzando una opportuna routine della libreria **fftw3**
(a) si calcoli il vettore $r = fft(u)$.
(b) Si illustri la proprietà di cui gode il vettore r.
(c) Descrivere lo schema ad albero in base al quale si esprime il calcolo della FFT mixed-radix nel caso $N = 20$.

Problema 7 Utilizzando i moduli di **FFTPACK**[33] scrivere delle routine che eseguano le seguenti operazioni:

1. assegnato un vettore x di dimensione n, calcolare la FFT di x (verificare la correttezza dei calcoli eseguendo una IFFT del risultato ottenuto e stimando l'ordine di grandezza dell'errore relativo tra x e $IFFT(FFT(x))$);
2. assegnata una matrice quadrata A di dimensione n calcolare la FFT bidimensionale di A (verificare la correttezza dei calcoli eseguendo una IFFT del risultato ottenuto e stimando l'ordine di grandezza dell'errore relativo tra A e $IFFT(FFT(A))$);
3. assegnata una matrice circolante A di dimensione n ed un vettore b di dimensione n eseguire il prodotto matrice vettore.

Problema 8 Ripetere gli stessi esercizi proposti utilizzando la libreria **FFTW**.

Problema 9

1. Per ciascun valore di m, $m = 1, \ldots, 5$, calcolare la DFT di un vettore $x_k = cos(mk\pi), k = 0, \ldots, 7$. Discutere i risultati.
2. Tracciare un grafico delle due funzioni $cos(\pi t)$ e $cos(3t)$ sull'intervallo $0 \leq t \leq 7$. Calcolare la DFT di ciascun vettore $x_k = cos(\pi k)$ e $x_k = cos(3k), k = 0, \ldots, 7$, e confrontare i risultati. Spiegare perché le DFT possono essere così differenti sebbene le funzioni siano così simili.

Problema 10 Gauss analizzò l'orbita dell'asteroide Pallas basata sull'osservazione dei dati:

θ	0	30	60	90	120	150
x	408	89	-66	10	338	807
θ	180	210	240	270	300	330
x	1238	1511	1583	1462	1183	804

dove θ rappresenta l'ascensione, in gradi, e x la declinazione, in minuti.

1. Rappresentare i dati mediante la funzione:

$$f(\theta) = a_0 + \sum_{k=1}^{5}[a_k cos(2\pi k\theta/360) + b_k sin(2\pi k\theta/360)] + a_6 cos(2\pi 6\theta/360),$$
$$\theta \in [0, 400]$$

2. Tracciare il grafico dei dati e della funzione calcolata al punto precedente.
3. Utilizzare una routine che implementi l'algoritmo FFT per calcolare la DFT y del vettore x.

[33] Utilizzare le routine **dfftf**,**dfftb** e **dffti** di FFTPACK.

4. Che relazione si può individuare tra parte reale e coefficiente dell'immaginario di y ed i parametri a_k e b_k calcolati al primo punto? (*Suggerimento*: Potrebbe essere necessario scalare rispetto alla lunghezza del vettore o alla sua radice quadrata, in base alla particolare routine FFT che si utilizza).

Problema 11 Sia x un vettore di numeri casuali di lunghezza n, ad esempio $n = 8$. Utilizzare una routine FFT per calcolare la DFT di x. A questo punto, realizzare uno shift delle componenti di x in modo circolare, ad esempio uno shift *ciclico* (o *end-around*) di *un solo posto* verso destra e calcolare la DFT del vettore shiftato e confrontarla con la DFT del vettore iniziale. Considerare, dunque, il modulo delle componenti di ciascuna delle due trasformate (tale vettore è detto "spettro delle ampiezze") e confrontarli. Provare ad eseguire lo shift di un numero maggiore di componenti, verso destra, e confrontare i vettori trasformate discrete di Fourier, con e senza shift, ed i loro relativi spettri delle ampiezze. Quale conclusione si può trarre?

Problema 12 Sia x il vettore $x_k = 1$, $k = 0, \ldots, n-1$, dove n *non è* una potenza di due. Sia $\hat{x}$ lo stesso vettore prolungato con componenti nulle, in modo da rendere la sua dimensione una potenza di due (i.e., $\hat{x}_k = 1$, $k = 0, \ldots, n-1$ e $\hat{x}_k = 0$, $k = n, \ldots, m-1$, dove m è la più piccola potenza di due maggiore di n). Ponendo $n = 5$ e $m = 8$, utilizzare una routine *FFT mixed-radix* per calcolare la DFT sia di x che di $\hat{x}$ e confrontare i risultati. Le trasformate discrete di Fourier coincidono? Cosa si può concludere sull'inserimento di zeri al fine di rendere la dimensione del vettore una potenza di due? Ripetere l'esercizio con altri valori n e di m per determinare se le considerazioni sui risultati sono coerenti.
Ripetere l'esercizio con altri vettori x (i.e., vettori con componenti non costanti o non periodiche) e confrontare le trasformate discrete di Fourier risultanti.

Problema 13 Usando una routine FFT mixed-radix misurare il tempo necessario per calcolare la DFT di un vettore di lunghezza n, per ciascun valore intero $n = 1, 2, 3, \ldots, 1024$. Tracciare il grafico del tempo di esecuzione in funzione di n, usando la scala logaritmica per l'asse delle ordinate. Noto il numero di operazioni eseguite al secondo, dal calcolatore in cui si implementa la routine, verificare che i limiti superiore ed inferiore dei risultati siano in accordo con i valori teorici della complessità di tempo, compresi tra $\mathcal{O}(n\, log_2 n)$ e $\mathcal{O}(n^2)$.

Problema 14 Utilizzare una routine per il calcolo di tutti gli autovalori della matrice *scalata* $(1/\sqrt{n})F_n$, $n = 1, 2, 3, \ldots, 16$. Quanti sono gli autovalori distinti?

Problema 15 Dimostrare empiricamente che la matrice di Fourier F_n diagonalizza una matrice circolante. A tal fine, generare una matrice circolante random C di ordine n e poi calcolare $(1/n)F_n C F_n^H$, con F_n^H matrice *trasposta coniugata* di F_n. La matrice ottenuta dovrebbe risultare diagonale. Cosa comporta questo risultato, riguardo gli autovalori di C? Provare, ad esempio, con $n = 8$.

Problema 16 Calcolare la DFT utilizzando routine standard che effettuano il prodotto di una matrice per un vettore, mediante matrici di Vandermonde. Confrontare le *prestazioni* con quelle di una routine FFT standard, eseguendo test su vettori di lunghezza potenza di due e su vettori che non abbiano lunghezza potenza di due. In particolare provare ad utilizzare, come dimensione, numeri primi *abbastanza grandi*.

Problema 17 Implementare una routine per il calcolo del prodotto di convoluzione tra due vettori. Usare una routine FFT per trasformare i vettori, calcolare il prodotto puntuale delle trasformate discrete di Fourier e, poi, trasformarle nel dominio originale attraverso la trasformazione inversa.