Capitolo 5

La Trasformata discreta di Fourier e l'algoritmo FFT

5.1 Introduzione

In questo capitolo, trattiamo 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 si ha

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

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 di componenti $\{s_k\}_{k=0,\ldots,N-1}$, definisce un vettore di componenti $\{S_k\}_{k=0,\ldots,N-1}$. 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 $O(N^2)$. 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 $O(N^2)$ a $O(N \log_2 N)$. 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 [14]. 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[2].
Consideriamo, quindi, il problema del quale si interessò Gauss. Nella tabella seguente, si riportano alcuni valori indicanti la posizione dell'asteroide Pallas, espressa in termini delle variabili (θ, X) che esprimono rispettivamente l'ascensione e la declinazione [3]:

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

[2]A tale proposito, Cooley nel ricordare i meriti di Gauss ai giovani ricercatori che aspirano ad una immediato riconoscimento del proprio lavoro, li ammonì raccomandando di *...non scrivere i lavori scientifici in Latino neo-classico.*

[3]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 (longitudine, latidune) tipicamente utilizzate per definire la posizione di un oggetto sulla sfera terrestre. L'Ascensione viene misurata in gradi, la declinazione in minuti.

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

Posizione dell'asteroide Pallas espressa in termini delle coordinate (Ascensione, Declinazione).

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 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* [4]. 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 quantità w_N^k con $k = 0, 1, \ldots, N-1$, dette **radici primitive N-me dell'unità**[5] poiché esse rappresentano le soluzioni nel campo complesso dell'equazione

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

svolgono un ruolo particolarmente significativo nel calcolo di una DFT poichè la complessità computazionale di una DFT dipende, oltre che dalle operazioni floating point

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

[5] 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:

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

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.

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.

♣ **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[6].

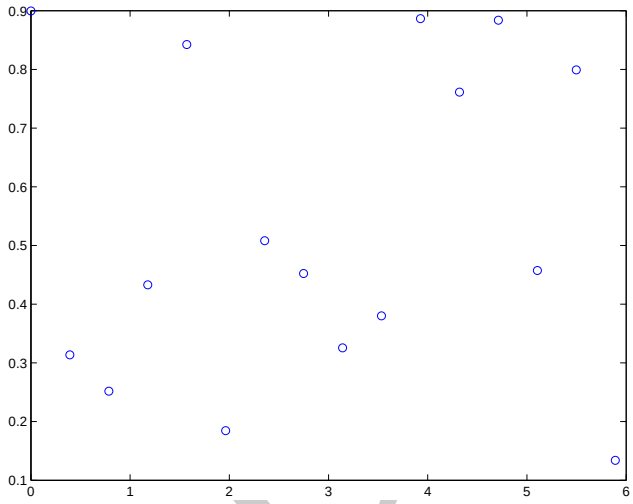

Figura 5.1: rappresentazione dei punti di coordinate (x_k, y_k), $k = 0, 1, ..., 15$ con $x_k = \frac{2\pi}{N}k$, $y_k \in \mathbb{C}$, $N = 16$

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 1**):

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

[6]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 introdurre 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.

essendo:

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

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

e $a = (a_0, \ldots, a_{15})$, $y = (y_1, \ldots, y_{15})$.
Poiché [7]:

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

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

la matrice del sistema diventa:

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

la cui soluzione banale è:

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

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).

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

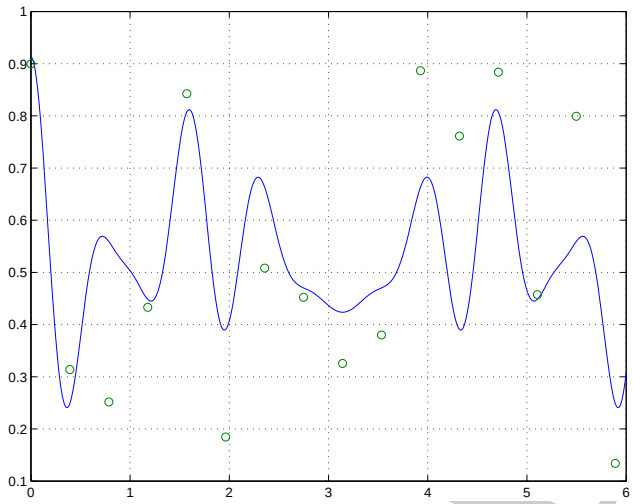

Figura 5.2: Grafico del polinomio trigonometrico p^* di migliore approssimazione nel senso dei minimi quadrati relativo ai punti (x_i, y_i), $\quad i = 0, 1, ..., 15$ dell'**esempio 5.2**.

Il risultato al quale siamo pervenuti è a 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:

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 $(z_k, y_k), k = 0, \ldots, N-1$.

♣

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[8].

[8]Sussiste il seguente teorema (**Teorema di campionamento di Shannon**):

Diamo la seguente:

Definizione 5.1.1. *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 5.1.2. *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 matematico per la risoluzione di problemi in matematica numerica, come ad esempio, nella risoluzione di problemi di calcolo matriciale, se la matrice presenta particolari strutture, nella risoluzione di equazioni differenziali ordinarie e alle derivate parziali con condizioni al contorno di Dirichlet o di Neumann,

Proposizione 5.1.3. *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 *tale che:*

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

allora:

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

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

Lo sviluppo in serie al secondo membro è anche noto 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. 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 *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**.

nell'integrazione numerica di funzioni periodiche o oscillanti. L'introduzione dell'algoritmo FFT per il calcolo veloce di una DFT, ha stimolato lo sviluppo di metodi e algoritmi efficienti per la risoluzione dei problemi della matematica numerica. L'algoritmo FFT è oggi uno dei 10 algoritmi con il maggior impatto scientifico e tecnologico [9].

5.1.1 Alcune proprietà della DFT

Descriviamo alcune delle proprietà più significative della DFT, ovvero consideriamo quelle proprietà che sono alla base delle applicazioni della DFT[10]. 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 numerici per il calcolo di una DFT.

Proposizione 5.1.4. *Siano $f, g \in \mathbb{C}^N$ due vettori di numeri complessi di lunghezza N e $\alpha\ , \beta \in \Re$. 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:

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

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

■

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

[10] A tutte le proprietà dell'operatore DFT corrispondono analoghe proprietà dell'operatore di Trasformata di Fourier perché, come vedremo nel paragrafo 5.2, l'operatore DFT si può anche interpretare come discretizzazione della Trasformata di Fourier ottenuta mediante integrazione numerica dell'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 di $h = f + g$ calcolando 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 5.1.5. *Il vettore $F = DFT[f]$, DFT del vettore di numeri complessi $f_0, \ldots, f_{N-1}$ di lunghezza N è N-periodico, ovvero:*

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

Dimostrazione Questa proprietà discende direttamente dalla peridicità 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[11].

Proposizione 5.1.6. *Valgono le seguenti proprietà* [12] *:*

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.*

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

[12] 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 $g \in \mathcal{C}^N$ 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 $g \in \mathcal{C}^N$ 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 5.1.6, 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:

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

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 5.1.7. *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

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

■

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 [13] tra due vettori a e b, trasforma il vettore a in un altro in

[13] **Definizione (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:

$$c = a * b$$

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

$$c_j = \sum_{k=0}^{j} a_k \cdot b_{j-k} \quad j = 0, ..., N-1.$$

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 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* [14] per un vettore. Per tale motivo, utilizzando opportunamente la proprietà dell'operatore DFT nel calcolo del prodotto di convoluzione, derivano in maniera naturale gli algoritmi basati sulla DFT per il calcolo con matrici circolanti.

Teorema 5.1.1 (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 $\{f_n\}_{n=0,...,23}$, 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

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

si ha

$$g_n = \sum_{j=0}^{23} f_j \cdot h_{n-j} = f_n * h_n$$

14

Definizione 5.1.3. (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 $C := C(v)$*

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

In altre parole, il vettore $\{g_n\}$ è il prodotto di convoluzione del vettore $\{f_n\}$ con il vettore $\{h_n\}$. A differenza del vettore $\{f_n\}$ che, come abbiamo visto, rappresenta i 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 funzione con meno picchi 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 [15] 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:

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

[15] 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.

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:

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

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

e ciò richiede il calcolo di 3 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*, ovvero è una matrice i cui elementi sono costanti lungo le diagonali. 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

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

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 $y = T \cdot x$. Quindi, per calcolare y basta calcolare $y' = C \cdot x'$ utilizzando la DFT. ♣

In generale, una qualsiasi matrice di Toeplitz $T \in \mathbb{R}^{n \times n}$, si può *immergere* in una matrice circolante, ovvero T si può riguardare come sottomatrice di una matrice circolante $C \in \mathbb{R}^{(2n-1)\times(2n-1)}$, 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]:

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

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;

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

dove:

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

[16]Questo sistema si ottiene, ad esempio, se consideriamo 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 $u(x, y) = g(x, y), (x, y) \in \partial\Omega$. Se per la sua risoluzione numerica discretizziamo le derivate parziali all'interno del dominio Ω con uno schema alle differenze finite su 5 punti:

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

e

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

avendo posto $h = L/N$, e introduciamo i punti $x_i = ih$ e $y_j = jh$, $i = 0, \ldots, n$ e $j = 0, \ldots, n$ si ottiene un reticolo Ω_h composto da $(n+1) \times (n+1)$ punti. Numerando i nodi del reticolo costruito nel dominio Ω e quindi le incognite ad essi relative, partendo dal nodo in alto a sinistra e procedendo lungo le righe da sinistra verso destra, prima all'interno del reticolo e poi sulla frontiera in senso orario dal nodo in alto a sinistra, il calcolo della soluzione nei punti del reticolo Ω_h comporta la costruzione del sistema $Mu = b$ dove u indica il vettore di dimensione N costituito da n blocchi ciascuno di dimensione n. Ciascun blocco è ottenuto fissando l'indice di riga $i = 1, \ldots, n$, e facendo variare l'indice di colonna $j = 1, \ldots, n$. In generale $u(x_i, y_j) = u_{i,j}$, $i = 1, \ldots, n-1$, $j = 1, \ldots, n-1$, e $u_{0,j} = f(x_0, y_j)$ $u_{i,0} = f(x_i, y_0)$, $u_{n,j} = g(x_n, y_j)$ e $u_{i,n} = g(x_i, y_n)$.

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:

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

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:

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

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

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

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 $a, b \in \mathbb{R}^5$:

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

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:

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

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

utilizzando il **Teorema 5.1.1**:

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

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

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

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

indicato con:

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

[17] [1]

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 5.1.1**, 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:

$$c = IDFT[C]$$

5.2 La Trasformata discreta di Fourier e la Trasformata di Fourier

Consideriamo il seguente integrale:

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

5.3.3 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 [23]. La complessità di questi algoritmi è
$$T(N) = O(N(r_1 + r_2))$$

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:
$$T(N) = O(p \log_{r_1} N + q \log_{r_2} N)$$

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:
$$T_{DFT}(N) = O(N log_2 N)$$

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 $y = N \log N$, al crescere di N aumenta in maniera molto significativa; ad esempio per $N = 10^8$, risulta $N^2 = 10^{16}$ e $N \log N \simeq 10^8 \cdot 24$ 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 $24 \cdot 10^8 \cdot 10^{-9} sec = 2.4 sec$[21]

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

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

[21]The fundamental law of computer science: as machines become more powerful, the efficiency of algorithms grows more important, not less. [Nick Trefethen, SIAM News, Jan/Feb, 1998]

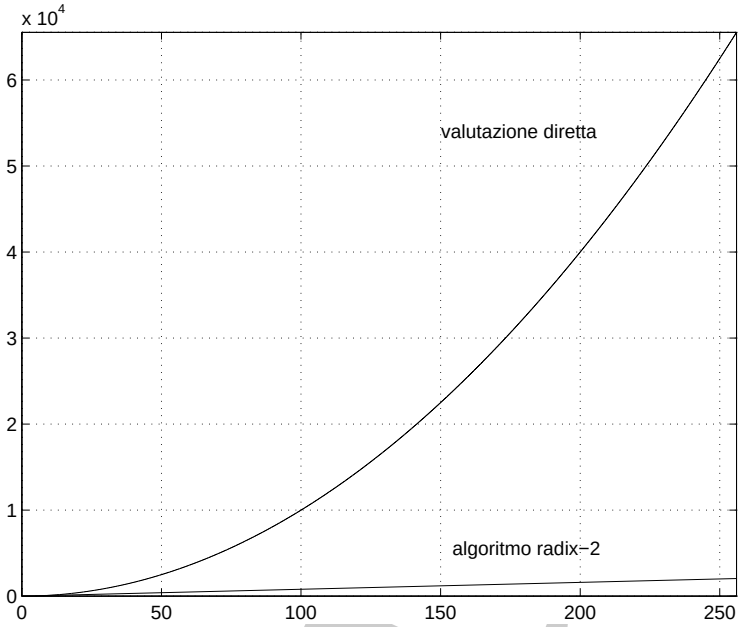

Figura 5.5: confronto tra la complessità computazionale della valutazione diretta della DFT di una sequenza di numeri complessi di lunghezza N e la FFT della medesima sequenza

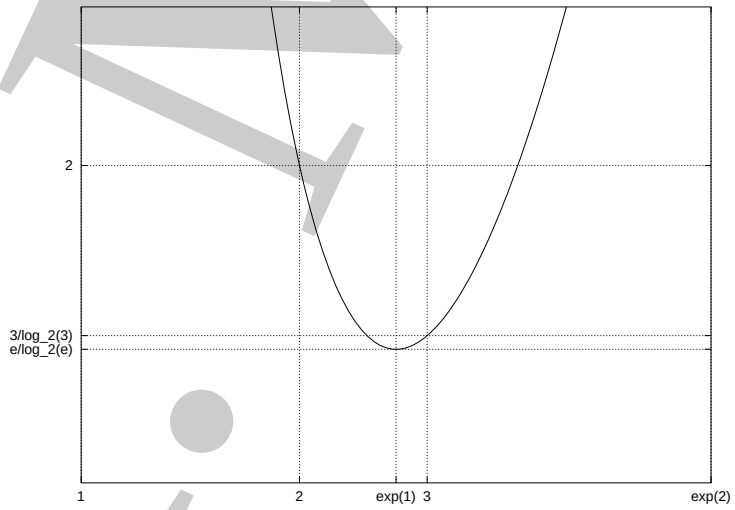

Figura 5.6: rappresentazione del fattore di proporzionalità $\frac{r}{log_2 r}$

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

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

ed ha nel punto $r = 3$ 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 5.19.** 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

$$4n(2N \log(2N)) = 4 \cdot 10^3 \cdot 2 \cdot 10^6 \log(2 \cdot 10^6) = O(10^9) \quad flop$$

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

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

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

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

Utilizzando un calcolatore con una prestazione di 1 Gflop/s, ciò signica una riduzione da 10 anni con l'algoritmo di Cholesky a mezz'ora con la versione a banda e infine a 3 minuti utilizzando la FFT. ♣

5.3.4 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 5.20. 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:

$$k = 0 \quad \rightarrow \quad F_0 = f_0 \omega_2^0 + f_1 \omega_2^0 \tag{5.42}$$

$$k = 1 \quad \rightarrow \quad F_1 = f_0 \omega_2^0 + f_1 \omega_2^{-1} \tag{5.43}$$

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

$$\begin{array}{lll} k = 0 & \rightarrow & F_0 = f_0 + f_1 \\ k = 1 & \rightarrow & F_1 = f_0 - f_1 \end{array} \tag{5.44}$$

♣

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)*.

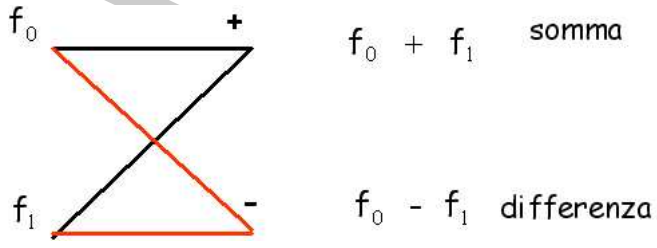

Figura 5.7: Schema butterfly

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

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

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

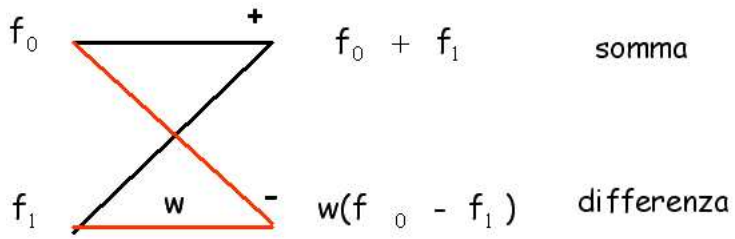

Figura 5.8: Schema generale di una butterfly

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

♣ **Esempio 5.21. 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:

$$\begin{array}{l} \omega_4^{-0} = e^{-i\frac{2\pi}{4}0} = cos(\frac{2\pi}{4}0) - isen(\frac{2\pi}{4}0) = 1 \\ \omega_4^{-1} = e^{-i\frac{2\pi}{4}1} = cos(\frac{2\pi}{4}1) - isen(\frac{2\pi}{4}1) = -i \\ \omega_4^{-2} = e^{-i\frac{2\pi}{4}2} = cos(\frac{2\pi}{4}2) - isen(\frac{2\pi}{4}2) = -1 \\ \omega_4^{-3} = e^{-i\frac{2\pi}{4}3} = cos(\frac{2\pi}{4}3) - isen(\frac{2\pi}{4}3) = i \\ \omega_4^{-4} = e^{-i\frac{2\pi}{4}4} = cos(\frac{2\pi}{4}4) - isen(\frac{2\pi}{4}4) = 1 \\ \omega_4^{-6} = e^{-i\frac{2\pi}{4}6} = cos(\frac{2\pi}{4}6) - isen(\frac{2\pi}{4}6) = -1 \\ \omega_4^{-9} = e^{-i\frac{2\pi}{4}9} = cos(\frac{2\pi}{4}9) - isen(\frac{2\pi}{4}9) = -i \end{array} \tag{5.45}$$

segue:

$$\begin{array}{llll} F_0 = f_0 & + f_1 & + f_2 & + f_3 \\ F_1 = f_0 & - if_1 & - f_2 & + if_3 \\ F_2 = f_0 & - f_1 & + f_2 & - f_3 \\ F_3 = f_0 & + if_1 & - f_2 & - if_3 \end{array} \tag{5.46}$$

Se si raccolgono, in maniera opportuna, alcuni termini nella (5.46) si ottiene:

$$\begin{array}{l} F_0 = \boldsymbol{(f_0 + f_2) + (f_1 + f_3)} \\ F_1 = \boldsymbol{(f_0 - f_2) - i(f_1 - f_3)} \\ F_2 = (f_0 + f_2) - (f_1 + f_3) \\ F_3 = (f_0 - f_2) + i(f_1 - f_3) \end{array} \tag{5.47}$$

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

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

Secondo l'algoritmo di Gentleman e Sande, come descritto nel paragrafo 5.3.2, 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:

$$\begin{array}{ll} y_0 = (f_0 + f_2) & z_0 = (f_0 - f_2) \\ y_1 = (f_1 + f_3) & z_1 = -i(f_1 - f_3) \end{array} \tag{5.49}$$

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

$$\begin{array}{ll} y'_0 = y_0 + y_1 & z'_0 = z_0 + z_1 \\ y'_1 = y_0 - y_1 & z'_1 = z_0 - z_1 \end{array} \tag{5.50}$$

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 (5.49) si possono descrivere graficamente utilizzando lo schema della Figura 5.9, che corrisponde a due *butterfly* innestate.

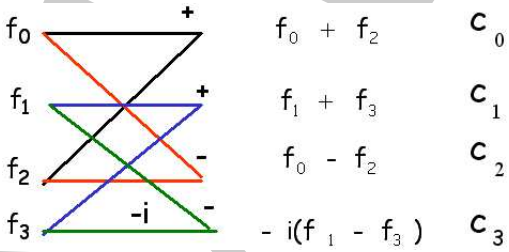

Figura 5.9: Schema del passo 1

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 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 (5.50) si possono descrivere graficamente utilizzando lo schema della Figura 5.10.

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

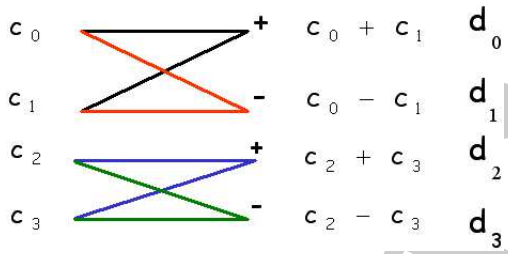

Figura 5.10: Schema del passo 2

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. ♣

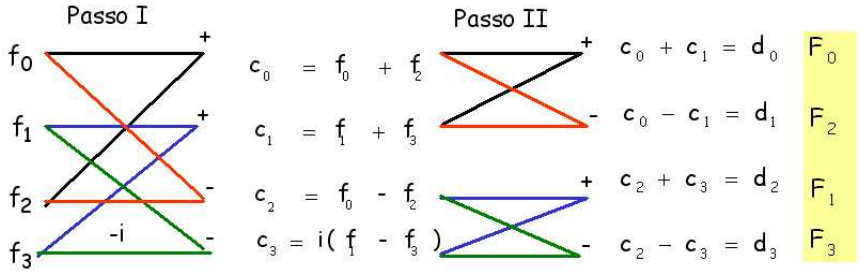

Figura 5.11: Schema di una DFT di lunghezza 4

♣ **Esempio 5.22. 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:

$$\begin{array}{ll} y_0 = (f_0 + f_4) & z_0 = (f_0 - f_4) \\ y_1 = (f_1 + f_5) & z_1 = (f_1 - f_5) \\ y_2 = (f_2 + f_6) & z_2 = (f_2 - f_6) \\ y_3 = (f_3 + f_7) & z_3 = (f_3 - f_7) \end{array} \tag{5.51}$$

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:

$$\begin{array}{l} y'_0 = c_0 + c_2 \\ y'_1 = c_1 + c_3 \\ y''_0 = c_0 - c_2 \\ y''_1 = c_2 - c_3 \\ z'_0 = c_4 + ic_6 \\ z'_1 = c_5 + ic_7 \\ z''_0 = c_4 - ic_6 \\ z''_1 = c_5 - ic_7 \end{array} \tag{5.52}$$

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:

$$\begin{array}{l} y''' = (d_0 + d_1) \\ y^{iv} = (d_0 - d_1) \\ y^{v} = (d_2 + id_3) \\ y^{vi} = (d_2 - id_3) \\ z''' = (d_4 + \omega_8^2 d_5) \\ z^{iv} = (d_4 - \omega_8^2 d_5) \\ z^{v} = (d_6 + \omega_8^6 d_7) \\ z^{vi} = (d_6 - \omega_8^6 d_7) \end{array} \tag{5.53}$$

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:

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

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.

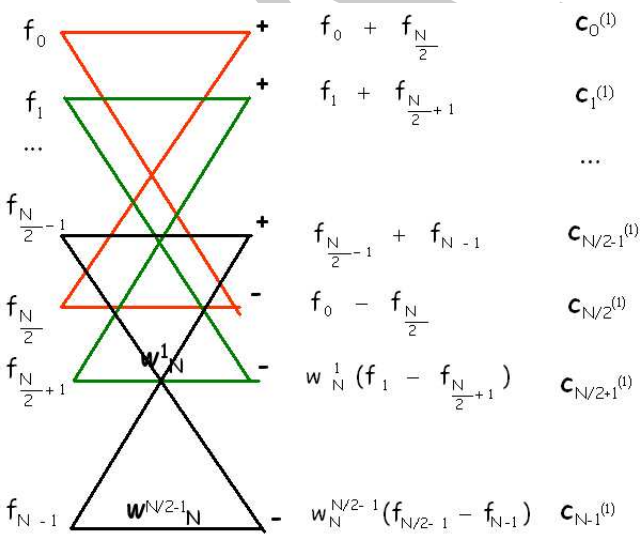

Figura 5.12: Schema del passo 1 di una DFT di lunghezza N

- 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.

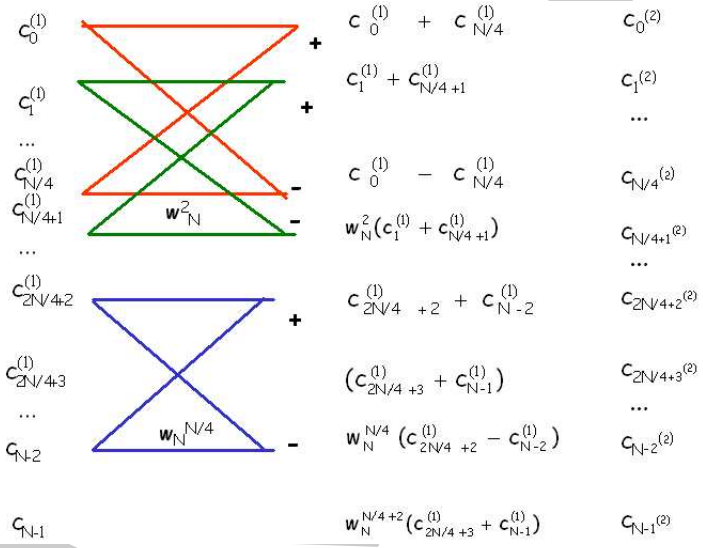

Figura 5.13: Schema del passo 2 di una DFT di lunghezza N

- 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.

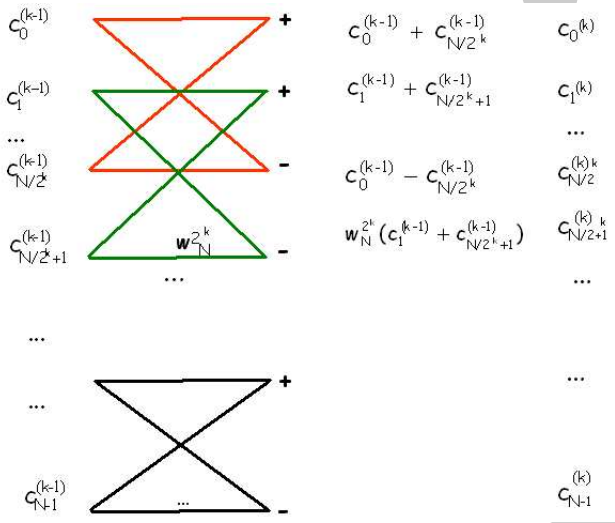

Figura 5.14: Schema del passo k di una DFT di lunghezza N

- ...
- 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.

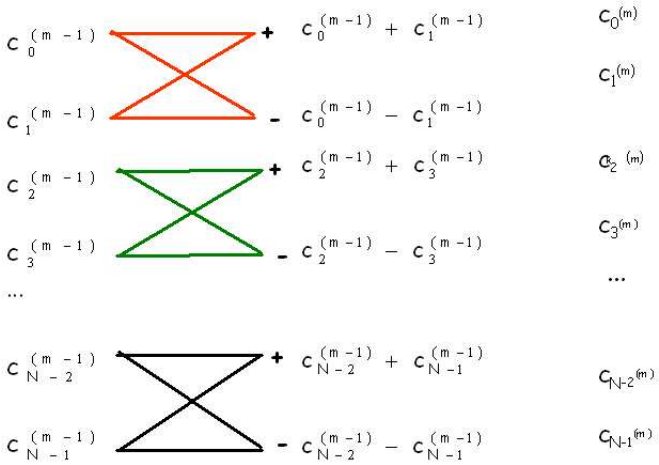

Figura 5.15: Schema del passo m di una DFT di lunghezza N

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*).

La versione proposta nel 1966 da Gentleman e Sande prevede quindi una fase iniziale di *calcolo* (operazioni butterfly) ed una fase di *riordinamento* del vettore ottenuto.

Si nota che l'algoritmo di Cooley e Tukey prevede 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.

L'aspetto più interessante da un punto di vista implementativo è tener conto 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 dividere l'indice $j = 0, \ldots, N-1$ per 2 e quindi per il numero intero j esiste una rappresentazione del tipo:

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

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

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

e ancora, riapplicare questo ragionamento ai 4 vettori di lunghezza $N/4$ ovvero dividere r per 2 si ha:

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

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

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

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

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

Tenendo conto che

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

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, assumendo anche per l'indice k una rappresentazione binaria del tipo:

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

si ha che:

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

e quindi

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

Si nota che, 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, che vedremo nel paragrafo 5.3.5. 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

perché di fatto l'algoritmo combina componenti non adiacenti neanche secondo l'ordinamento del bit reversal. Tale riordinamento dovrà essere applicato unicamente alla componenti del vettore DFT qualora si vogliano tali componenti nell'ordine naturale.

♣ **Esempio 5.23.** Consideriamo 4 interi ordinati secondo la relazione di ordine naturale:

$$0 < 1 < 2 < 3$$

e rappresentiamoli nel sistema binario:

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

invertendo la rappresentazione binaria di ciascun numero, otteniamo:

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

ordinati secondo la relazione di ordine *del bit inverso* $<_{bitinv}$ ovvero:

$$0 <_{bitinv} 2 <_{bitinv} 1 <_{bitinv} 3$$

♣

```
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 : Algoritmo di FFT radix-2: calcolo del veotte c_k, $k = 1, \ldots, m$.

5.3.5 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 $log(N)$ matrici sparse e strutturate. In effetti, il vettore $c_{(k)}$ costruito al passo k dell'algoritmo si può vedere come ottenuto dal prodotto del vettore costruito al passo precedente moltiplicato per una opportuna matrice.

Utilizzando la (5.62) e la (5.64) si ottiene che:

$$|| \ |\hat{A}_k| \ ||_2 \leq || \ |A_k| \ ||_2 + || \ |\delta A_k| \ ||_2 \leq (\sqrt{2} + \mu)||A_k||_2. \tag{5.65}$$

La (5.63), mediante la (5.64) si può scrivere come:

$$\hat{F} = (A_m + \Delta A_m + \Delta\hat{A}_m)...(A_1 + \Delta A_1 + \Delta\hat{A}_1) \cdot P_r \cdot f \tag{5.66}$$

Posto:

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

si ottiene:

$$\hat{F} = (A_m + E_m)...(A_1 + E_1) \cdot P_r \cdot f \tag{5.67}$$

e dalla (5.64) e (5.65) si ha:

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

Applicando il Lemma 5.3.1[25] si trova che:

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

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

poiché:

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

segue la tesi. ■

Il teorema mostra che, utilizzando l'algoritmo FFT, la propagazione dell'errore sulla soluzione, in un sistema aritmetico a precisione finita $\Im$ cresce linearmente con m, ovvero l'algoritmo FFT è *stabile* nel senso della f.e.a.

5.4 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

[25]

Lemma 5.3.1. *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:*

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

C/C++, altri in Fortran 77/90. Le prime implementazioni risalgono al 1969[24], l'ultima è del 2006 (`FFTW 3.1`). Il sito netlib `(url:http://www.netlib.org)` consente di accedere alle più diffuse librerie di software matematico tra cui, ad esempio, `FFTPACK`, `MPFUN`, `NAPACK` e `SCIPORT`.

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 si è avuta nel 1998, con il pacchetto *Fast Fourier Transform in the West* (`FFTW`)[8]. Il principio alla base del software FFTW è quello di *ottenere le massime prestazioni su una qualsiasi architettura e per un qualsiasi problema*, (prestazione portabile) *risolvendo il problema attraverso una opportuna combinazione di un certo numero di algoritmi elementari* . Ciascuna routine del pacchetto software FFTW è ottenuta come compoizione di componenti elementari, ciascuna componente (detta codelets) rappresenta l' implementazione dell'algoritmo FFT di base, come ad esempio, il calcolo della DFT per $N = 2, 3, 5, 7, ..$, l'algoritmo di Cooley e Tukey, l'algoritmo di Gentleman e Sande, etc.. Il software FFTW 3.1 contiene circa 150 codelets. I codelets sono creati automaticamente, in fase di installazione della libreria, da un compilatore (*genfft*) e utilizzati da un programma (detto *plan*) che manda in esecuzione i *codeletes* necessari per il calcolo dela DFT richiesta. Il programma plan viene creato dal *planner*, il cui ruolo è proprio quello di *prendere in input il problema e scegliere, attraverso tecniche automatiche e dinamiche, la strategia migliore e più efficiente per risolvere quel problema*. L'algoritmo migliore viene scelto in base alle prestazioni che i codeletes raggiungono nell'ambiente di calcolo considerato. La ricerca dell'algoritmo migliore viene fatta percorrendo l' albero binario ottenuto dalla fattorizzazione del parametro N.

L'aspetto innovativo della FFTW risiede proprio nella presenza del plan, e del planner, con cui interagisce l'utente. L'obiettivo è quello di coniugare portabilità ed efficienza, che tipicamente sono attributi del software che dipendono in maniera inversamente proporzionale l'uno dall'altro, 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`.

```
#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. Lutente 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, utilizzata, tra l'altro, routine per l'allocazione ed il rilascio di memoria:

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

5.5 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** [26]. 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.** [27] Si consideri una funzione $x = x(t)$ composta da due funzioni sinusoidali, una di

[26] http://www.fftw.org
[27] http://www.mathworks.com/

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).

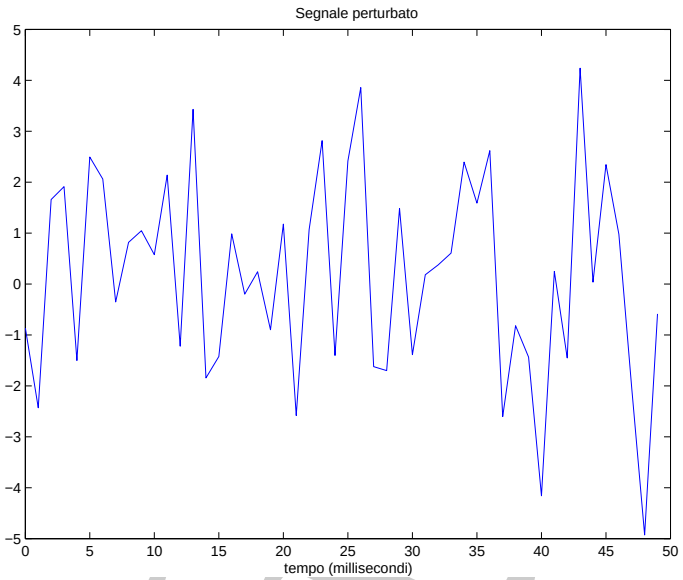

Figura 5.16: Grafico della funzione $y(t) = 0.7\sin(2\pi 50t) + \sin(2\pi 120t) + \eta(t)$

```
>>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:

```
>>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)|')
```

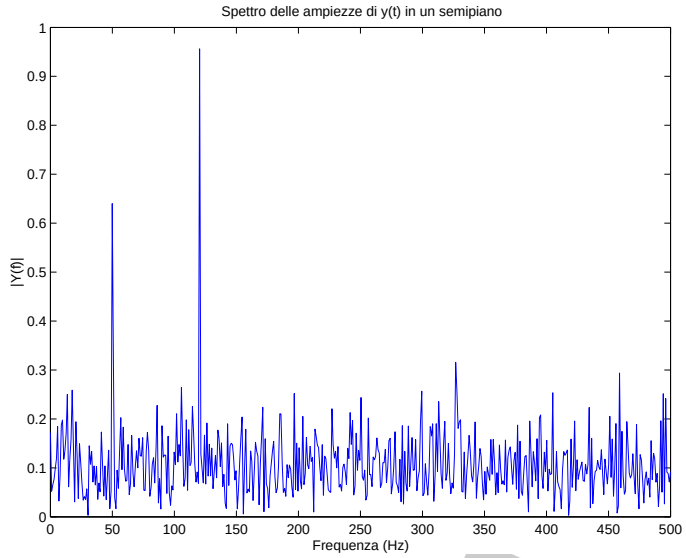

Figura 5.17: Frequenze ed ampiezze della funzione $y(t)$

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` [28].
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.

♣

5.5.1 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:

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

Applicare al vettore y la funzione `ifft`:

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

e stimare l'ordine di grandezza dell'errore:

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

[28] 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.

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.

5.6 Esercizi

5.6.1 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?

5.6.2 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 $\alpha, \beta \in \mathbb{R}$, 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 $f \in \mathbb{R}^N$ 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.

5.6.3 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**[29] 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.

[29] 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 *cielico* (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.

Bibliografia

[1] Bailey D. H. - *Multiprecision Translation and Execution of Fortran Programs* - ACM Trans. on Mathematical Software, vol. 19, no. 3 (Sept. 1993), pp. 288-319.

[2] Briggs W., Henson V. E. - *The DFT, An Owner Manual for the Discrete Fourier Transform* - Society for Industrial and Applied Mathematics, Philadelphia (1995).

[3] Brigham E. - *The Fast Fourier Transform and its Applications* - Prentice Hall Signal Processing Series (1988).

[4] Cooley J. W., Tukey J. W. - *An Algorithm for the Machine Calculation of Complex Fourier Series* - Mathematics of Computation, Vol.19, No. 90, pp. 297-301 (1965).

[5] Danielson G. C., Lanczos C. - *Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids* - J.Franklin Inst., 233, 1942.

[6] De Angelis P., Murli A., Pirozzi M. A. - *Sulla Valutazione dell'Errore nel Calcolo Numerico degli Integrali Trigonometrici* - Calcolo, Vol. XIII, fasc. IV (1976).

[7] Edelman A., MCorquodale P., Toledo S. - *The Future Fast Fourier Transform?* - SIAM J. Sci. Comp., Vol. 20, N. 3, pp. 1094-1114 (1999).

[8] Frigo M. and Johnson S. G. - *FFTW: An Adaptive Software Architecture for the FFT* - ICASSP conference proceedings, vol. 3, pp. 1381-1384, (1998).

[9] Garbow B. S., Giunta G., Lyness J. L., Murli A. - *Algorithm 662. A Fortran Package for the Numerical Inversion of the Laplace Transform Based on Weeks' Method* - ACM Transaction on Mathematical Software, Vol. 14 (3) (1988).

[10] Garbow B. S., Giunta G., Lyness J. L., Murli A. - *Software for an Implementation of Weeks' Method for the Inverse Laplace Transform Problem* - ACM Transaction on Mathematical Software, Vol. 14 (2) (1988).

[11] Gentleman W. M., Sande G. - *Fast Fourier transforms - for fun and profit* - AFIPS Proc., vol. 29 (1966).

[12] Giunta G., Murli A. - *Algorithm 649. A Package for Computing Trigonometric Fourier Coefficients Based on Lyness Algorithm* - ACM Transaction on Mathematical Software, Vol. 13 (1) (1987).

[13] Goertzel G. - *An algorithm for the evaluation of finite trigonometric series* - Amer. Math. Montly, 65, 34 (1958).

[14] Heideman M. T., Johnson D. H., Burrus C. S. - *Gauss and the history of the fast Fourier transform* - Archive for History of Exact Sciences, 34 (1985).

[15] Higham N. - *Accuracy and Stability of Numerical Algorithms* - SIAM 2002.

[16] Maddalena M. R., Murli A. - *Su alcuni algoritmi utilizzabili per il calcolo dell'integrale di Fourier (premesse per la costruzione di un software)* - Quaderni IAC-CNR, N. 101 (1979).

[17] Morris L. R. - *Digital Signal Processing Software* - Toronto, Canada: DSPSW, Inc., 1982, 1983.

[18] Murli A. - *Alcune formule per il calcolo numerico della Trasformata di Fourier* - Calcolo, 4 (1967).

[19] Murli A. - *Alcune formule per il calcolo numerico della Trasformata di Laplace* - Calcolo, Vol. IV, fasc. IV (1967).

[20] Murli A. - *Il calcolo numerico degli integrali trigonometrici*, Calcolo 5 (1968).

[21] Murli A. - *Sull'impiego di un metodo numerico per il calcolo dell'antitrasformata di Laplace* - Rendiconto dell'Accademia di Scienze Fisiche e Matematiche, Vol. IV (1970).

[22] Murli A., Patruno V. - *Un metodo per l'inversione numerica della Trasformata di Laplace* - Calcolo, vol. XV (1978).

[23] Rader C. M. - *Discrete Fourier transforms when the number of data points is prime* - Proc. of IEEE, vol. 56 (1968).

[24] Singleton R. C. - *A method for computing the fast Fourier transform with auxiliary memory and limited high-speed storage* - IEEE Trans. Audio Electroacoustics AU-15, 2 (June 1967) 91-98.

[25] Van Loan C. - *Computational Frameworks for the Fast Fourier Transform* - SIAM, 1992.

[26] Walker J. - *Fast Fourier Transforms* - CRC-Press; 2nd edition (October 22, 1996).