Capitolo 7

Interpolazione trigonometrica e la Trasformata discreta di Fourier

7.1 La Trasformata Discreta di Fourier e la Trasformata Inversa

In molte applicazioni è richiesto il calcolo di somme del tipo:

$$S_j = \sum_{k \in I} e^{-ijk} x_k \ , \qquad i = \sqrt{-1}$$

ove I è un sottoinsieme finito dell'insieme dei numeri naturali, $j \in I$ e $\underline{x} = \{x_k\}_{k \in I} \in \mathbb{C}$ è un vettore le cui componenti sono numeri complessi.

♣ **Esempio 7.1.** Considerati alcuni rilevamenti della marea del golfo di Napoli, come ad esempio quelli riportati nella tabella seguente:

	28 Marzo		29 Marzo		30 Marzo
ore:	0 : 00	13 : 05	1 : 05	14 : 10	0 : 05
centimetri:	2.5	1.9	2.8	1.6	2.6

per descrivere l'andamento della marea costruiamo il polinomio trigonometrico interpolante:

$$p(x) = a_0 + \sum_{k=1}^{m} [a_k cos(kx) + b_k sin(kx)]$$

In particolare, indicando con (x_k, y_k) $k = 0, ..., 4$ le coppie di punti costituite dai rilevamenti orari e dai centimetri della marea della precedente tabella, si richiede che:

$$p(x_k) = y_k \quad k = 0, ..., 4$$

Si dimostra[1] che tali coefficienti sono del tipo:

$$a_0 := \frac{A_0}{2} \quad con \quad A_0 = \frac{1}{5}\sum_{k=0}^{4} y_k = 2.28$$

$$a_1 := \frac{2}{5}\sum_{k=0}^{4} y_k cos(x_k) = 0.1324 \quad , \quad a_2 := \frac{2}{5}\sum_{k=0}^{4} y_k cos(2x_k) = 0.0876$$

$$b_1 := \frac{2}{5}\sum_{k=0}^{4} y_k sin(x_k) = 0.0158 \quad , \quad b_2 := \frac{2}{5}\sum_{k=0}^{4} y_k sin(2x_k) = -0.6211$$

da cui segue che:

$$p(x) = \frac{2.28}{2} + 0.1324\ cos(x) + 0.0876\ cos(2x) + 0.0158\ sin(x) - 0.6211\ sin(2x).$$

In Figura 7.1 è riportato il grafico di $p(x)$.

Utilizzando le identità di Eulero[2] segue che:

$$Z_j = a_j + ib_j = \frac{2}{n}\sum_{k=0}^{n-1}(y_k cos(jx_k) + iy_k sin(jx_k)) = \frac{2}{n}\sum_{k=0}^{n-1} y_k e^{(ijx_k)} \quad i = \sqrt{-1}$$

♣

[1]

Teorema 7.1. *Assegnate le coppie (x_k, y_k), k=0,...,n-1, con $x_k = \frac{2\pi k}{n}$, e posto:*

$$A_j := \frac{2}{n}\sum_{k=0}^{n-1} y_k cos(jx_k), \quad B_j := \frac{2}{n}\sum_{k=0}^{n-1} y_k sin(jx_k)$$

se n è dispari sia:

$$p(x) := \frac{A_0}{2} + \sum_{k=1}^{m}(A_k cos(kx) + B_k sin(kx)) \quad \text{dove } N = 2m+1$$

e se n è pari:

$$p(x) := \frac{A_0}{2} + \sum_{k=1}^{m-1}(A_k cos(kx) + B_k sin(kx)) + \frac{A_m}{2} cos(mx) \quad \text{dove } N = 2m+1$$

allora p è il polinomio trigonometrico interpolante gli n punti, ovvero:

$$p(x_k) = y_k, \quad k = 0, ..., n-1$$

[2]

$$cos(x) = \frac{e^{ix} + e^{ix}}{2}, \quad sin(x) = \frac{e^{ix} - e^{ix}}{2i}$$

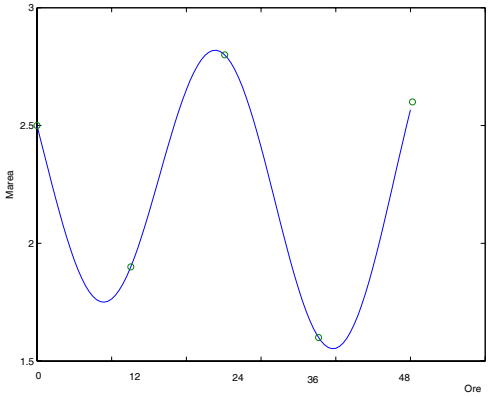

Figura 7.1: Grafico del polinomio trigonometrico $p(x) = 1.14 + 0.1324\,cos(x) + 0.0876\ cos(2x) + 0.0158\ sin(x) - 0.6211\ sin(2x)$

♣ **Esempio 7.2.** Supponiamo di conoscere in $N(=16)$ punti:

$$x_k = \frac{2\pi}{16}k, \quad k = 0, ..., 15; \tag{7.1}$$

i valori di una funzione periodica $y = y(x),\ x \in [0, 1]$.

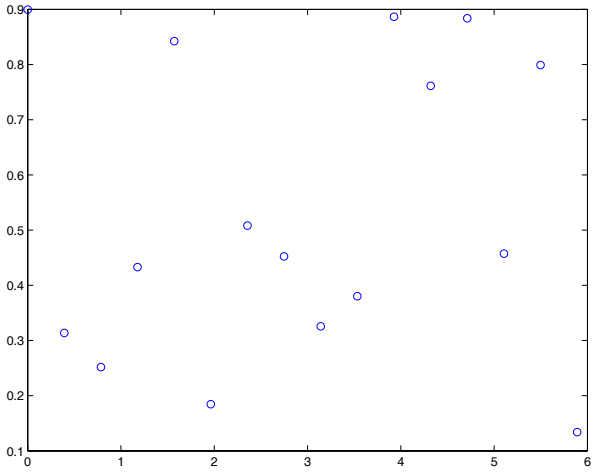

Figura 7.2: rappresentazione dei punti di coordinate $(x_k, y_k),\ k = 0, 1, ..., 15$ con $x_k = \frac{2\pi}{N}k,\ y_k \in \mathbb{C},\ N = 16$

Determiniamo la funzione f^* di migliore approssimazione nel senso dei minimi quadrati dei punti $(x_k, y_k),\quad k = 0, ..., 15$, ovvero tale che:

$$f^*(x) = \min_{f \in \Im} ||y_i - f(x_i)||_2$$

In particolare, il problema è determinare i coefficienti a_j^* della funzione:

$$f^*(x) = \frac{a_0^*}{2} + a_1^* cosx + \ldots + a_{12}^* cos(12x) \tag{7.2}$$

tale che f^* approssimi i punti $(x_k, y_k)_{k=0,N-1}$ nel senso dei minimi quadrati.
Tale problema equivale alla risoluzione del sistema delle *equazioni normali*:

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

essendo:

$$A = \begin{pmatrix} 1 & cos(x_1) & ... & cos(12(x_1)) \\ 1 & cos(x_2) & ... & cos(12(x_2)) \\ 1 & cos(x_3) & ... & cos(12(x_3)) \\ \vdots & & & \\ 1 & cos(x_{15}) & ... & cos(12(x_{15})) \end{pmatrix}$$

$$A^T y = \begin{pmatrix} 2\sum_{i=0}^{15} y_i \\ 2\sum_{i=0}^{15} y_i cos x_i \\ \vdots \\ \vdots \\ 2\sum_{i=0}^{15} y_i cos(12) x_i \end{pmatrix}$$

Poiché:

$$\sum_{i=0}^{N-1} cos(jx_i)cos(kx_i) = \begin{cases} 0 & j \neq k \\ \frac{N}{2} & j = k \neq 0 \\ N & j = k = 0 \end{cases} \tag{7.3}$$

$$\sum_{i=0}^{N-1} cos(jx_i) = \begin{cases} 0 & j \neq mN \\ N & j = mN \end{cases} \quad \text{con} \quad m \in \mathbb{N} \tag{7.4}$$

la matrice del sistema delle equazioni normali è, in realtà, diagonale:

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

e la soluzione è:

$$a_j^* = \frac{2}{16} \sum_{k=0}^{15} y_k cos \frac{2\pi kj}{16}, \qquad j = 0, ..., 12$$

Analogamente se:

$$f^*(x) = \frac{b_0}{2} + b_1^* sin(x) + \ldots + b_{12}^* sin(12x) \tag{7.5}$$

i coefficienti b_j^* sono:

$$b_j^* = \frac{2}{16} \sum_{k=0}^{15} y_k sen \frac{2\pi kj}{16}, \qquad j = 0, 1, ..., 12$$

Utilizzando le identità di Eulero segue:

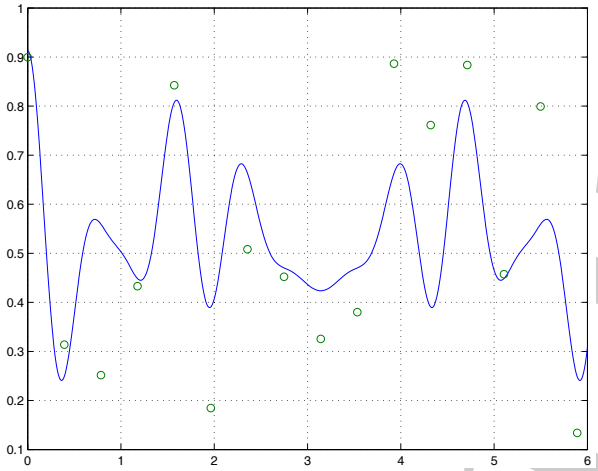

Figura 7.3: Grafico della funzione f^* di migliore approssimazione nel senso dei minimi quadrati dei punti (x_i, y_i), $\quad i = 0, 1, ..., 15$ dell'**esempio 7.2**.

$$z_j = a_j^* + ib_j^* = \frac{2}{n}\left[\sum_{k=0}^{n-1} y_k cos(kx_j) + iy_k sin(kx_j)\right] = \frac{2}{n}\sum_{k=0}^{n-1} y_k e^{(ikx_j)} \qquad i = \sqrt{-1}$$

♣

Diamo la seguente:

Definizione 7.1. *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:*

$$F_k = \sum_{j=0}^{N-1} f_j e^{-\frac{2\pi i}{N} jk} \qquad k = 0, N-1;\ i = \sqrt{-1} \tag{7.6}$$

In particolare:

$$F := DFT[f] = \boldsymbol{W} \cdot \underline{\boldsymbol{f}} \tag{7.7}$$

essendo:

$$W = \begin{pmatrix} w^0 & w^0 & w^0 & \ldots & w^0 \\ w^0 & w^1 & w^2 & \ldots & w^{N-1} \\ w^0 & w^2 & w^4 & \ldots & w^{2(N-1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ w^0 & w^{N-1} & \ldots & \ldots & w^{(N-1)(N-1)} \end{pmatrix}$$

dove si è posto[3] $w = e^{\frac{-2\pi i}{N}}$. La matrice $\boldsymbol{W}$ è detta matrice di Fourier.

Definizione 7.2. *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:*

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} F_j e^{\frac{2\pi i}{N} jk} \qquad k = 0, N-1; i = \sqrt{-1} \tag{7.8}$$

In particolare:

$$f := IDFT[f] = \frac{1}{N} \boldsymbol{W^{-1}} \cdot \underline{\boldsymbol{F}} \tag{7.9}$$

7.1.1 Alcune proprietà della DFT

Proposizione 7.1. *Siano* $f, g \in \mathbb{C}^N$ *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:

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

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

■

♣ **Esempio 7.3.** Assegnati due vettori complessi:

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

calcoliamo la DFT di f e di 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:

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

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

♣

[3] Nel seguito w verrà indicato anche con w_N

Proposizione 7.2. *Sia* $f = (f_0, f_1, ..., f_{N-1}) \in \mathbb{C}^N$ *un vettore di numeri complessi di lunghezza* N *e sia*

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

il suo vettore ***simmetrico****. Si ha:*

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

Dimostrazione Dalla definizione di DFT, fissata una componente di indice k con $k = 1, 2, ...N/2$ del vettore $DFT[DFT[f]]$ si ha che:

$$\frac{1}{N}(DFT[DFT[f]])_k = \frac{1}{N}\sum_{j=0}^{N-1}(DFT[f])_j w_N^{-jk}$$

e per le proprietà dell'esponenziale complesso $w_N^{Nj} = e^{\frac{2\pi N j}{N}} = 1$ si ha:

$$\frac{1}{N}(DFT[DFT[f]])_k = \frac{1}{N}\sum_{j=0}^{N-1}(DFT[f])_j w_N^{Nj} w_N^{-jk} =$$

$$= 1\frac{1}{N}\sum_{j=0}^{N-1}(DFT[f])_j w_N^{(N-k)j} = f_{N-k}$$

■

♣ **Esempio 7.4.** *Sia:*

$$f = (1+i, -3, 5+7i, -2)$$

poniamo:

$$f_{simm} = (1+i, -2, 5+7i, -3)$$

La DFT del vettore f *è data da:*

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

Se calcoliamo $\frac{1}{4}DFT[DFT[f]]$ *si ha:*

$$\frac{1}{4}DFT[F] = (1+i, -2, 5+7i, -3) = f_{simm}$$

♣

Proposizione 7.3. *Valgono le seguenti proprietà:*

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.*
3. *Se g 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 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 7.5.** Calcoliamo la DFT del vettore $f = (1, 2, 3, 4, 5, 6)$ e del vettore $g = (i, -2i, 5i, -4i, 3i, 6i)$:

$$F = DFT[f] = (21, -3 - 5.1962i, -3 - 1.7321i, -3, -3 + 1 - 7321i, -3 + 5.1962i)$$
$$G = DFT[g] = (9i, -5.1962 + 3i, -8.6603 + 9i, -9i, 8.6603i + 9i, 5.1962.3i)$$

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

Proposizione 7.4. *La matrice $\tilde{W} = \frac{1}{\sqrt{N}} W$ è autoaggiunta, cioè $\tilde{W} \cdot \tilde{W}^T = I$.*

Dimostrazione

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

$$= \frac{1}{N} \cdot \begin{pmatrix} \sum_{i=0}^{N-1} w^0 \cdot w^0 & \sum_{i=0}^{N-1} w^0 \cdot w^i & ... & \sum_{i=0}^{N-1} w^0 \cdot w^{(N-1)i} \\ \sum_{i=0}^{N-1} w^0 \cdot w^i & \sum_{i=0}^{N-1} w^i \cdot w^i & ... & \sum_{i=0}^{N-1} w^i \cdot w^{(N-1)i} \\ ..... & & & \\ \sum_{i=0}^{N-1} w^0 \cdot w^{(N-1)i} & \sum_{i=0}^{N-1} w^i \cdot w^{(N-1)i} & ... & \sum_{i=0}^{N-1} w^{(N-1)i} \cdot w^{(N-1)i} \end{pmatrix}$$

$$= \frac{1}{N} \cdot \begin{pmatrix} N & 0 & 0 & ... & 0 \\ 0 & N & 0 & ... & 0 \\ 0 & 0 & N & ... & 0 \\ ..... & & & & \\ 0 & 0 & 0 & ... & N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 1 & ... & 0 \\ ..... & & & & \\ 0 & 0 & 0 & ... & 1 \end{pmatrix}$$

■

Teorema 7.2 (di Convoluzione). *Siano $F = DFT[f]$ e $G = DFT[g]$ le DFT di due vettori f e g di lunghezza N. La DFT del prodotto di convoluzione* [4] *$f * g$ è il prodotto componente per componente dei vettori F e G, ovvero:*

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

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

7.1.2 Alcuni esempi di applicazioni della DFT

♣ **Esempio 7.6.** [**Moltiplicazione di due polinomi**] Siano assegnati due polinomi:

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

Definiti i due vettori:

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

che contengono 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:

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

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:

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

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

utilizzando il **Teorema 7.2**:

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

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

[4] **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:

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

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.

♣

In generale, assegnati due polinomi

$$p(x) = \sum_{i=0}^{p-1} a_i x^i i \in \Pi_r \quad q(x) = \sum_{j=0}^{s-1} b_j x^j \in \Pi_s$$

ed indicato con:

$$z(x) = p(x)q(x) = \sum_{k=0}^{r+s-2} c_k x^k \quad \in \Pi_{r+s-2} \tag{7.10}$$

il polinomio prodotto, i coefficienti di z:

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

sono ottenuti effettuando il prodotto di convoluzione:

$$c = a * b$$

avendo indicato con $a = (a_0, a_1, ...a_{r-1})$ i coefficienti di p, $b = (b_0, b_1, ...b_{s-1})$ i coefficienti di z e con $c = (c_0, c_1, ...c_{r+s-2})$ quelli di z.
Utilizzando il **Teorema 7.2** 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:

$$c = IDFT[C]$$

♣ **Esempio 7.7.** Consideriamo la matrice circolante[5] 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[6]

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

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

Poiché

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

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 dei vettori F e X'. Essendo

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

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

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

$$Y' = (405, 4.63 + 28.6i, -12.1 - 5.951i, -12.1 + 5.951i, 4.63 - 28.6i)$$

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

$$y' = IDFT[Y'] = (86, 93, 75, 67, 84)$$

♣

[5]

Definizione 7.3. (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 $C := C(v)$*

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

[6]

Teorema 7.3. *Sia $C := C(v)$ una matrice circolante di lunghezza N generata dal vettore v ed $F = DFT[v]$ allora vale:*

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

essendo W la matrice di Fourier di dimensione N e $D = diag(F)$ una matrice diagonale con elementi diagonali costituiti dalle componenti di F.

In generale, consideriamo una matrice circolante C, generata da un vettore v: $C = C(v)$. Per tali matrici, indicata con $F = DFT[v]$ la DFT del vettore v, esiste una fattorizzazione del tipo:

$$C = W^{-1}DW \tag{7.11}$$

essendo W la matrice di Fourier e $D = diag(F)$ una matrice diagonale.
Dalla (7.11) segue che se si vuole effettuare il prodotto matrice vettore:

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

si ha:

$$C \cdot x' = y' \Leftrightarrow W^{-1}DW \cdot x' = y' \Leftrightarrow DW \cdot x' = Wy' \tag{7.12}$$

Poiché:

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

posto $X' = DFT[x']$ e $Y' = DFT[y']$ la (7.12) diventa:

$$C \cdot x' = y' \Leftrightarrow F. * X' = Y' \tag{7.13}$$

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

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

e ciò richiede il calcolo di 3 DFT.

7.2 La Trasformata Veloce di Fourier (FFT)

7.2.1 Introduzione

Dalla definizione di DFT di un vettore di N numeri complessi si deduce che la valutazione diretta di ciascuna componente F_k del vettore DFT:

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

richiede il calcolo di N moltiplicazioni complesse e $N-1$ addizioni complesse, ed inoltre la valutazione di N esponenziali complessi. Supponendo noti gli esponenziali complessi, il calcolo della DFT di un vettore di lunghezza N richiede, quindi, una complessità:

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

La complessità computazionale del calcolo di una DFT di lunghezza N è dunque quella di un prodotto matrice-vettore.
Vista la grande varietà di applicazioni per cui è necessario effettuare una DFT, la disponibilità di algoritmi efficienti è fondamentale. Fino al 1965, anno di pubblicazione del primo algoritmo di "*Fast Fourier Transform* (FFT) di Cooley e Tukey, nonostante la già ampia diffusione della DFT, la sua applicazione presentava problemi legati all'alto costo computazionale.
L'idea fondamentale su cui si basa l'algoritmo FFT (di Cooley e Tukey) è la fattorizzazione della matrice W di Fourier nel prodotto di matrici sparse. Nel caso particolare, che sarà trattato nel seguito, in cui N è una potenza di 2, la matrice W viene fattorizzata nel prodotto di $log_2 N$ matrici sparse, e la complessità del calcolo di una DFT si riduce a $O(N log_2 N)$ operazioni tra numeri complessi.

7.2.2 FFT radix-2

Si descrive l'idea alla base della classe di algoritmi **FFT** (Fast Fuorier Transform) per il calcolo della DFT di un vettore di lunghezza uguale ad una potenza di 2.
Supponiamo che $N = 2^r$, ovvero prendiamo in considerazione la classe degli algoritmo FFT *radix-2*.

♣ Esempio 7.8. 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:

$$k = 0 \quad \rightarrow \quad F_0 = f_0 \omega_2^0 + f_1 \omega_2^0 \tag{7.14}$$

$$k = 1 \quad \rightarrow \quad F_1 = f_0 \omega_2^0 + f_1 \omega_2^1 \tag{7.15}$$

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 (7.14) e la (7.15) si riducono a:

$$\begin{array}{lll} k=0 & \rightarrow & F_0 = f_0 + f_1 \\ k=1 & \rightarrow & F_1 = f_0 - f_1 \end{array} \tag{7.16}$$

La DFT di lunghezza 2 richiede allora solo 2 **addizioni** (complesse). ♣

Una DFT di lunghezza 2 si può schematizzare mediante un'operazione di base che consiste di due somme algebriche tra numeri complessi. Ad una DFT di lunghezza 2 associamo lo schema grafico (Figura 7.4), detto *schema a farfalla (o butterfly)* che descrive tale operazione.

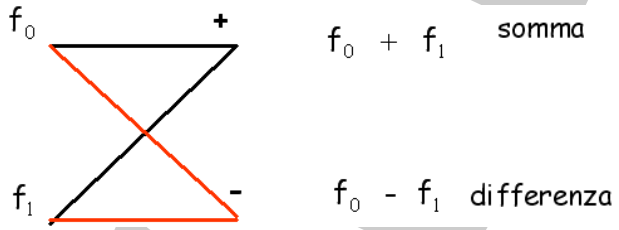

Figura 7.4: Schema butterfly

Più in generale, all'operazione di base del tipo:

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

con w un numero complesso, corrisponde lo schema butterfly mostrato in Figura 7.5.

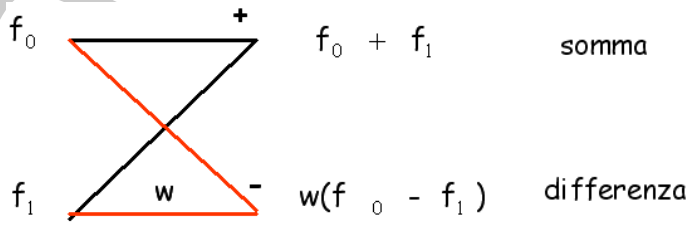

Figura 7.5: Schema generale di una butterfly

♣ **Esempio 7.9. 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:

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

ovvero:

$$\begin{array}{ll} k=0 \rightarrow & F_0 = f_0\omega_4^0 + f_1\omega_4^0 + f_2\omega_4^0 + f_3\omega_4^0 \\ k=1 \rightarrow & F_1 = f_0\omega_4^0 + f_1\omega_4^1 + f_2\omega_4^2 + f_3\omega_4^3 \\ k=2 \rightarrow & F_2 = f_0\omega_4^0 + f_1\omega_4^2 + f_2\omega_4^4 + f_3\omega_4^6 \\ k=3 \rightarrow & F_3 = f_0\omega_4^0 + f_1\omega_4^3 + f_2\omega_4^6 + f_3\omega_4^9 \end{array} \tag{7.17}$$

il calcolo diretto di $F_0,\ F_1,\ F_2,\ F_3$ richiede, quindi, 16 moltiplicazioni complesse. Poiché gli esponenziali complessi sono:

$$\begin{array}{l} \omega_4^0 = e^{-i\frac{2\pi}{4}0} = cos(\frac{2\pi}{4}0) - isen(\frac{2\pi}{4}0) = 1 \\ \omega_4^1 = e^{-i\frac{2\pi}{4}1} = cos(\frac{2\pi}{4}1) - isen(\frac{2\pi}{4}1) = -i \\ \omega_4^2 = e^{-i\frac{2\pi}{4}2} = cos(\frac{2\pi}{4}2) - isen(\frac{2\pi}{4}2) = -1 \\ \omega_4^3 = e^{-i\frac{2\pi}{4}3} = cos(\frac{2\pi}{4}3) - isen(\frac{2\pi}{4}3) = i \\ \omega_4^4 = e^{-i\frac{2\pi}{4}4} = cos(\frac{2\pi}{4}4) - isen(\frac{2\pi}{4}4) = 1 \\ \omega_4^6 = e^{-i\frac{2\pi}{4}6} = cos(\frac{2\pi}{4}6) - isen(\frac{2\pi}{4}6) = -1 \\ \omega_4^9 = e^{-i\frac{2\pi}{4}9} = cos(\frac{2\pi}{4}9) - isen(\frac{2\pi}{4}9) = -i \end{array} \tag{7.18}$$

la (7.17) si riduce a:

$$\begin{array}{lllll} k=0 \rightarrow & F_0 = f_0 & + f_1 & +f_2 & + f_3 \\ k=1 \rightarrow & F_1 = f_0 & -if_1 & -f_2 & +if_3 \\ k=2 \rightarrow & F_2 = f_0 & - f_1 & +f_2 & - f_3 \\ k=3 \rightarrow & F_3 = f_0 & +if_1 & -f_2 & -if_3 \end{array} \tag{7.19}$$

Se si raccolgono, in maniera opportuna, alcuni termini nella (7.19) si ottiene:

$$\begin{array}{ll} k=0 \rightarrow & F_0 = \boldsymbol{(f_0 + f_2) + (f_1 + f_3)} \\ k=1 \rightarrow & F_1 = \boldsymbol{(f_0 - f_2) - i(f_1 - f_3)} \\ k=2 \rightarrow & F_2 = (f_0 + f_2) - (f_1 + f_3) \\ k=3 \rightarrow & F_3 = (f_0 - f_2) + i(f_1 - f_3) \end{array} \tag{7.20}$$

La DFT di lunghezza 4 richiede, allora, solo 8 **addizioni** (complesse).
La riorganizzazione del calcolo delle componenti del vettore F secondo la (7.20), oltre a mettere in evidenza la riduzione della complessità computazionale, induce anche in modo "naturale" il calcolo di tali componenti.
In particolare il calcolo della DFT si può così riorganizzare:

- **passo 1:** si calcolano le quantità:

$$\begin{array}{ll} \boldsymbol{c_0 = (f_0 + f_2)} & \boldsymbol{c_1 = (f_1 + f_3)} \\ \boldsymbol{c_2 = (f_0 - f_2)} & \boldsymbol{c_3 = -i(f_1 - f_3)} \end{array} \tag{7.21}$$

  per cui la (7.20) diventa:

$$\begin{array}{l} F_0 = \boldsymbol{c_0 + c_1} \\ F_2 = \boldsymbol{c_0 - c_1} \\ F_1 = \boldsymbol{c_2 + c_3} \\ F_3 = \boldsymbol{c_2 - c_3} \end{array} \tag{7.22}$$

- **passo 2:** si calcolano le quantità:

$$\begin{aligned} d_0 &= \boldsymbol{c_0} + \boldsymbol{c_1} \\ d_1 &= \boldsymbol{c_0} - \boldsymbol{c_1} \\ d_2 &= \boldsymbol{c_2} + \boldsymbol{c_3} \\ d_3 &= \boldsymbol{c_2} - \boldsymbol{c_3} \end{aligned} \tag{7.23}$$

  Il vettore $\underline{d} = (d_0, d_1, d_2, d_3)$ è la DFT del vettore F a meno dell'ordine delle sue componenti (ordine *scrambled*).

Il calcolo delle quantità (7.21) si può schematizzare utilizzando lo schema grafico descritto nella Figura 7.6, che corrisponde a due butterfly innestate.

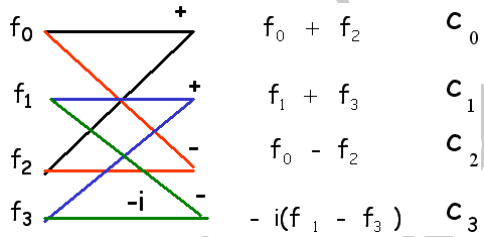

Figura 7.6: Schema del passo 1

Analogamente, il calcolo delle quantità (7.23) si può schematizzare utilizzando lo schema grafico descritto nella Figura 7.7.

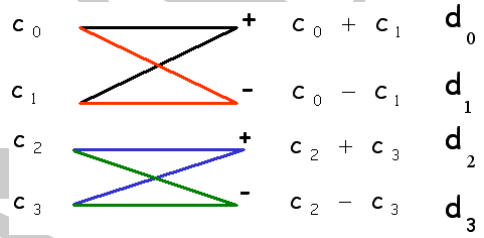

Figura 7.7: Schema del passo 2

Per il calcolo di una DFT di lunghezza 4 si sono effettuati due passi, in ciascuno dei quali è richiesto il calcolo di due DFT di lunghezza 2.
Nella Figura 7.8 sono mostrate le butterfly necessarie al calcolo di una DFT di lunghezza 4. ♣

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

$$(f_0, f_1, ..., f_{2^{m-1}})$$

sfruttando le proprietà degli esponenziali complessi coinvolti nella definizione di DFT:

$$F_k = \sum_{j=0}^{2^{m-1}} f_j \omega_N^{jk} \quad k = 0, 1, ..., N-1$$

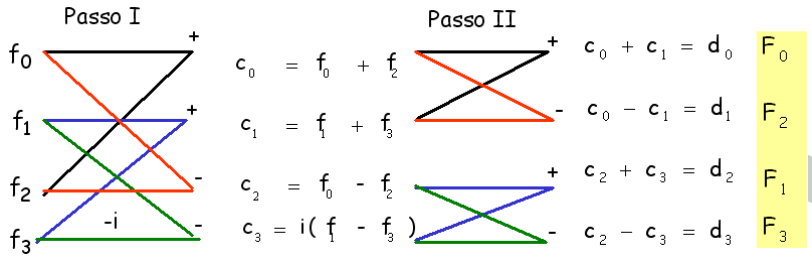

Figura 7.8: Schema di una DFT di lunghezza 4

è possibile riorganizzare le operazioni coinvolte nel calcolo di una DFT nel modo seguente:

$$\begin{aligned} F_0 &= (f_0 + f_{N/2}) + (f_1 + f_{N/2+1}) + \ldots + (f_{N/2-1} + f_{N-1}) \\ F_1 &= (f_0 - f_{N/2}) + w_N^1(f_1 - f_{N/2+1}) + \ldots + w_N^{N/2-1}(f_{N/2-1} - f_{N-1}) \\ &\ldots \\ F_{N-1} &= (f_0 - f_{N/2}) - w_N^1(f_1 - f_{N/2+1}) + \ldots - w_N^{N/2-1}(f_{N/2-1} - f_{N-1}) \end{aligned} \tag{7.24}$$

Per il calcolo della DFT sono necessari esattamente m passi, in ciascuno dei quali si calcolano le DFT di lunghezza 2. In particolare:

- nel passo 1 si utilizzano 2^{m-1} schemi butterfly ottenuti combiando le compenenti di f a distanza $p = N/2$, come mostrato nella Figura 7.9.

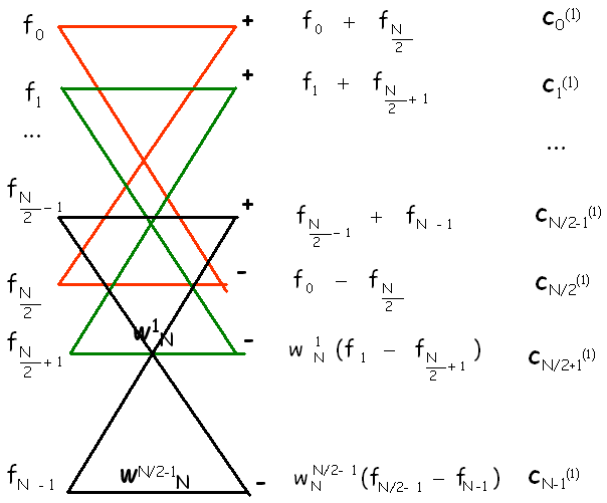

Figura 7.9: Schema del passo 1 di una DFT di lunghezza N

- nel passo 2 si utilizzano 2^{m-1} schemi butterfly ottenuti combiando le compenenti di f a distanza $p = N/4$, come mostrato nella Figura 7.10.

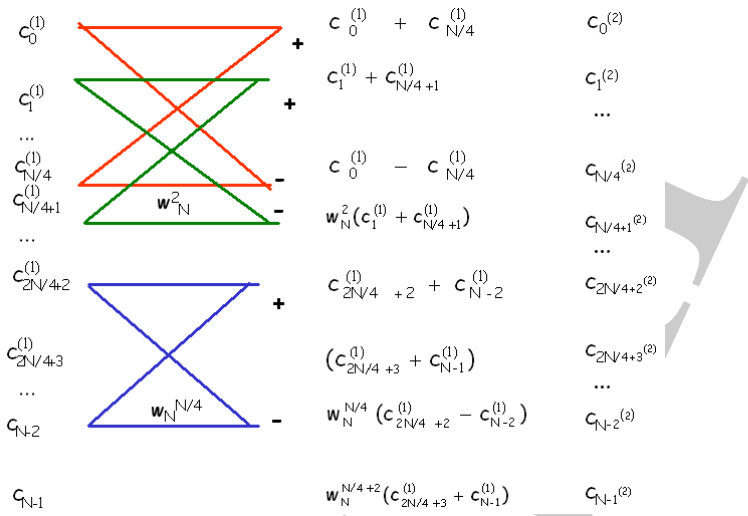

Figura 7.10: Schema del passo 2 di una DFT di lunghezza N

- nel passo k si utilizzano 2^{m-1} schemi butterfly ottenuti combinando combiando le compenenti di f a distanza $p = N/2^k$, come mostrato nella Figura 7.11.

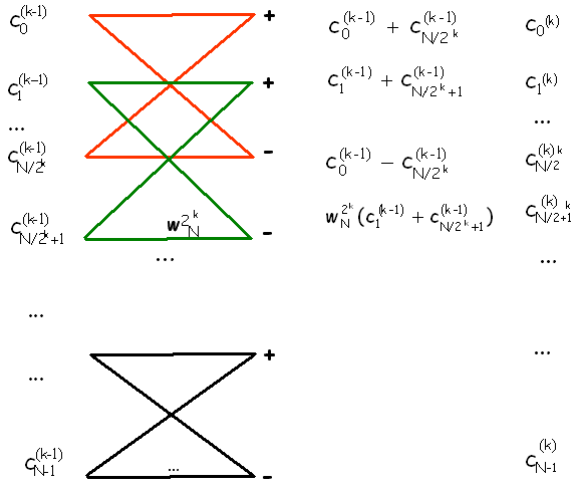

Figura 7.11: Schema del passo k di una DFT di lunghezza N

- ...
- al passo m si utilizzano 2^{m-1} schemi butterfly ottenuti combinando gli elementi di f a distanza $p = N/2^{m-1}$, come mostrato nella Figura 7.12.

In conclusione per il calcolo di una DFT di lunghezza N si sono effettuati m passi, in cui è richiesto il calcolo di 2^{m-1} DFT di lunghezza 2. In definitiva il numero di addizioni

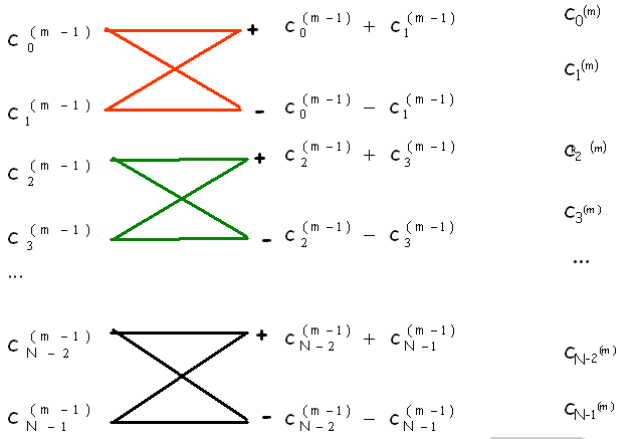

Figura 7.12: Schema del passo m di una DFT di lunghezza N

complesse coinvolte è:

$$T(N) = m(2^{m-1} \cdot DFT_2) = m * 2^m$$

Poiché $m = log_2(N)$ la complessità di tempo di questo algoritmo è:

$$T(N) = Nlog_2(N)$$

Osserviamo, infine, che il vettore che si ottiene all'ultimo passo è la DFT del vettore F nell'ordine *scrambled* (o anche del *bit inverso*). Al fine di chiarire questo tipo di relazione d'ordine facciamo un esempio.

♣ **Esempio 7.10.** Consideriamo 4 interi ordinati secondo la relazione di ordine naturale:

$$0 < 1 < 2 < 3$$

rappresentiamoli nel sistema binario:

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

invertendo la rappresentazione binaria di ciascun numero:

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

la relazione di ordine del bit inverso $<_{bitinv}$ sarà:

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

♣

7.2.3 Formulazione matriciale dell'algoritmo FFT radix-2

♣ **Esempio 7.11.** Sia $N = 2^1$; come nell'**esempio 7.8**, la DFT del vettore:

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

è la combinazione lineare delle sue componenti secondo lo schema:

$$\begin{aligned} F_0 &= f_0 + f_1 \\ F_1 &= f_0 - f_1 \end{aligned} \tag{7.25}$$

Se si considera la matrice di Fourier di dimensione 2:

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

si ha:

$$W \cdot f = F$$

La matrice A_1 è una matrice a blocchi del tipo:

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

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

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

♣

♣ **Esempio 7.12.** Sia $N = 2^2 = 4$; come nell'**esempio 7.9**, la DFT del vettore:

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

è la combinazione lineare delle componenti del vettore f secondo lo schema:

$$\begin{aligned} F_0 &= f_0 + f_1 + f_2 + f_3 \\ F_1 &= f_0 - if_1 - f_2 + if_3 \\ F_2 &= f_0 - f_1 + f_2 - f_3 \\ F_3 &= f_0 + if_1 - f_2 - if_3 \end{aligned} \tag{7.26}$$

Se si definisce la matrice di Fourier di dimensione quattro:

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

allora:

$$W \cdot f = F$$

Nell'**esempio 7.9** si è visto anche che, per calcolare le somme (7.26) è necessario effettuare il calcolo, in due passi, di opportune quantità. In particolare:

- al **passo 1** vengono calcolate le componenti del vettore c:

$$\begin{array}{ll} c_0 = (f_0 + f_2) & c_2 = (f_0 - f_2) \\ c_1 = (f_1 + f_3) & c_3 = -i(f_1 - f_3) \end{array} \tag{7.27}$$

  Introdotta la matrice:

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

  le operazioni (7.27) corrispondono al prodotto matrice per vettore:

$$A_1 \cdot f = c$$

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

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

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

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

  Se si pone:

$$B_1 = A_1$$

  risulta:

$$A_1 = I_1 \otimes B_1$$

  dove il simbolo $\otimes$ denota il prodotto tensoriale di due matrici[7] e I_1 la matrice identica di ordine 1.

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

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

- Al **passo 2** vengono calcolate le componenti del vettore d:

$$\begin{array}{ll} d_0 = (c_0 + c_1) & d_2 = (c_2 + c_3) \\ d_1 = (c_0 - c_1) & d_3 = (c_2 - c_3) \end{array} \tag{7.28}$$

Introdotta la matrice:

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

le operazioni (7.28) corrispondono al prodotto matrice per vettore:

$$A_2 \cdot c = d$$

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

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

essendo I_1 una matrice identica di ordine 1 e

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

una matrice diagonale di ordine 1. Se si pone:

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

risulta che:

$$A_2 = I_2 \otimes B_2,$$

dove I_2 è la matrice identica di ordine due.

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

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

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

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

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

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

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

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

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

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

da cui:

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

♣

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

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

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

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

  $$A_1 = I_1 \otimes B_1$$

  con:

  $$B_1 = \begin{bmatrix} I_{2^{m-1}} & I_{2^{m-1}} \\ \Omega_2^m & -\Omega_2^m \end{bmatrix}$$

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

  $$\Omega_2^m = (w_{2^m}^0, w_{2^m}^1, ..., w_{2^m}^{2^{m-1}})$$

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

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

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

  $$A_2 = I_2 \otimes B_1$$

  con

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

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

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

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

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

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

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

con

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

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

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

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

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

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

1. $A_1 \cdot f = c^{(1)}$;
2. $A_2 \cdot c^{(1)} = c^{(2)}$;

   ...

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

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

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

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

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

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

$$W = W^* \cdot P = A_m \cdot A_{m-1} \cdots A_1 \cdot P \tag{7.29}$$

da cui:

$$Wf = F \Leftrightarrow A_m \cdot A_{m-1} \cdot A_1 \cdots P \; f = F \tag{7.30}$$

7.2.4 Stabilità dell'algortimo FFT radix-2

Supponiamo di calcolare la FFT di un vettore f in un sistema aritmetico a precisione finita $\Im$. Siamo interessati ad analizzare la stabilità dell'algoritmo FFT radix-2.
Si supponga, come in [Van Loan], che gli esponenziali complessi ω_k siano pre-calcolati e su di essi sia stato commesso un errore $|\epsilon_{jk}|$ tale che:

$$\hat{\omega}_j^k = fl(\omega_j^k) = \omega_j^k + \epsilon_{jk} \quad |\epsilon_{jk}| \leq u \tag{7.31}$$

essendo u la massima accuratezza relativa di $\Im$.
Si enuncia e dimostra il seguente risultato [Higham].

Teorema 7.4. *Sia f un vettore di lunghezza $N = 2^m$ esattamente rappresentabile in $\Im$ ovvero $f = fl(f) = \hat{f}$. Sia $\hat{F} = fl(W \cdot f)$ il risultato ottenuto in $\Im$ utilizzando l'algoritmo FFT radix-2 con W matrice di Fourier di dimensione N. Allora:*

$$\frac{||F - \hat{F}||_2}{||F||_2} \leq \frac{m\eta}{1 - m\eta}, \quad \eta := \mu + \gamma_4(\sqrt{2} + \mu) \tag{7.32}$$

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

Dimostrazione Si osserva che per le matrici in (7.30):

$$||A_k||_2 = \sqrt{2} \tag{7.33}$$

e che:

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

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

$$\hat{F} = fl(\hat{A}_m....\hat{A}_1) = (\hat{A}_m + \Delta\hat{A}_m)...(\hat{A}_1 + \Delta\hat{A}_1) \cdot P_r \cdot f \tag{7.34}$$

dove con P_r si indica una matrice di permutazione di ordine r. Poiché ciascuna matrice A_k ha solo due elementi non nulli per riga ed inoltre ciascun elemento è un numero complesso[8] si ha:

$$|\Delta\hat{A}_k| \leq \gamma_4|\hat{A}_k|$$

per cui:

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

Dall'ipotesi (7.31) segue che:

$$\hat{A}_k = A_k + \Delta A_k \quad ||\Delta A_k|| \leq \sqrt{2}\mu = \mu||A_k||_2 \tag{7.35}$$

[8]Utilizzando un modello standard di aritmetica discreta a precisione finita [Higham] se x ed y sono numeri complessi esattamente reppresentabili si assume che:

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

Utilizzando la (7.33) e la (7.35) si ottiene che:

$$|| \, |\hat{A}_k| \, ||_2 \leq || \, |A_k| \, ||_2 + || \, |\delta A_k| \, ||_2 \leq (\sqrt{2} + \mu)||A_k||_2. \tag{7.36}$$

La (7.34), mediante la (7.35) si può scrivere come:

$$\hat{F} = (A_m + \Delta A_m + \Delta \hat{A}_m)...(A_1 + \Delta A_1 + \Delta \hat{A}_1) \cdot P_r \cdot f \tag{7.37}$$

Posto:

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

si ottiene:

$$\hat{F} = (A_m + E_m)...(A_1 + E_1) \cdot P_r \cdot f \tag{7.38}$$

e dalla (7.35) e (7.36) si ha:

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

Applicando il Lemma 7.1[9] si trova che:

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

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

utilizzando poi il Lemma 7.2[10] si ha la seconda diseguaglianza. In fine essendo A $\sqrt{r}$ volte la norma di una matrice unitaria ($A * A = nI$), si ha che:

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

da cui 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 FEA.

[9]

Lemma 7.1. *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:*

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

[10]

Lemma 7.2. *Sia assegnato un sistema aritmetico a precisione finita $\Im$, con massima accuratezza relativa u, se si considerano numeri $|\delta_i| \leq u$ e $\rho_i = \pm 1$ con $i = 1, ..., n$ con $nu < 1$ allora:*

$$\prod_{i=1}^{n} (1 + \delta_i)^{\rho_j} = 1 + \Theta_j \tag{7.39}$$

con:

$$|\Theta_n| \leq \frac{nu}{1 - nu} := \gamma_n$$

7.2.5 Algoritmi di Cooley e Tukey

In questo paragrafo si descrive una classe di algoritmi del tipo *dividi et impera*, ovvero *dividi e conquista*, per il calcolo di una DFT di lunghezza N introdotti da Cooley e Tukey nel 1965.

♣ **Esempio 7.13.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_8)$ di lunghezza $N = 3 \cdot 3 = 9$:

$$DFT[f]_k := F_k = \sum_{j=0}^{8} f_j w_9^{jk} \tag{7.40}$$

posto:

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

Si ha

$$F(3k_1 + k_0) = \sum_{j_0=0}^{2} \sum_{j_1=0}^{2} f(3j_0 + j_1) w_9^{(3j_0+j_1)(3k_1+k_0)} \tag{7.41}$$

Poiché:

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

e $w_9^{9j_0k_1} = 1$ la (7.41) si può scrivere come:

$$F(3k_1 + k_0) = \underbrace{\sum_{j_1=0}^{2} w_9^{3j_1k_1} \underbrace{\left[\sum_{j_0=0}^{2} f(3j_0 + j_1) w_9^{3j_0k_0} \right]}_{DFT\ di\ lunghezza 3} w_9^{j_1k_0}}_{3\ DFT\ di\ lunghezza\ 3} \tag{7.42}$$

Il calcolo di una DFT di lunghezza 9 si riconduce al calcolo 3 DFT di lunghezza 3, come illustrato nella Figura.

♣

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

♣ **Esempio 7.14.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{11})$ di lunghezza $N = 4 \cdot 3 = 12$:

$$DFT[f]_k := F_k = \sum_{j=0}^{11} f_j w_{12}^{jk} \tag{7.43}$$

posto:

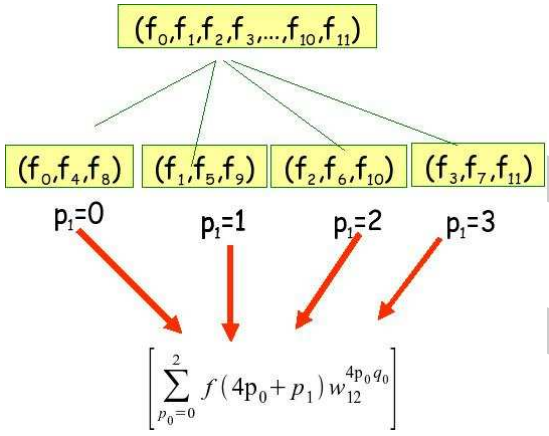

Figura 7.13: 4 DFT di lunghezza 3

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

allora:

$$F(3k_1 + k_0) = \sum_{j_0=0}^{2} \sum_{j_1=0}^{3} f(4j_0 + j_1) w_{12}^{(4j_0+j_1)(3k_1+k_0)} \tag{7.44}$$

Poiché:

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

e $w_{12}^{12j_0k_1} = 1$ la (7.44) si può scrivere come:

$$F(3k_1 + k_0) = \underbrace{\sum_{j_1=0}^{3} w_{12}^{3j_1k_1} \underbrace{\left[\sum_{j_0=0}^{2} f(4j_0 + j_1) w_{12}^{4j_0k_0}\right]}_{DFT\ di\ lunghezza 3} w_{12}^{j_1k_0}}_{4\ DFT\ di\ lunghezza\ 3} \tag{7.45}$$

Il calcolo di una DFT di lunghezza 12 si riconduce al calcolo 4 DFT di lunghezza 3, come illustrato nella Figura 7.13. ♣

La strategia descritta nell'esempio precedente è alla base degli algoritmi FFT-*mixed radix*, applicabile per il calcolo di DFT di vettori f di lunghezza $N = r_1 \times r_2$.

♣ **Esempio 7.15.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{11})$ di lunghezza $N = 3{\cdot}4 = 12$:

$$DFT[f]_k := F_k = \sum_{j=0}^{11} f_j w_{12}^{jk} \tag{7.46}$$

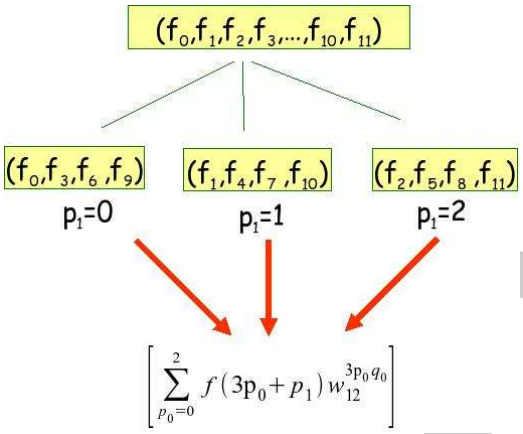

Figura 7.14: 3 DFT di lunghezza 4

posto:

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

è possibile riscrivere la (7.46) come:

$$F(4k_1 + k_0) = \sum_{j_0=0}^{2} \sum_{j_1=0}^{3} f(3j_0 + j_1) w_{12}^{(4j_0+j_1)(3k_1+k_0)} \tag{7.47}$$

Poiché:

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

e $w_{12}^{12j_0k_1} = 1$ allora la (7.47) si può scrivere come:

$$F(4k_1 + k_0) = \underbrace{\sum_{j_1=0}^{2} w_{12}^{4j_1k_1} \underbrace{\left[\sum_{j_0=0}^{3} f(3j_0 + j_1) w_{12}^{3j_0k_0}\right]}_{DFT\ di\ lunghezza 4} w_{12}^{j_1k_0}}_{3\ DFT\ di\ lunghezza\ 4} \tag{7.48}$$

Il calcolo di una DFT di lunghezza 12 si riconduce al calcolo 3 DFT di lunghezza 4, come illustrato nella Figura 7.14.

Se consideriamo la (7.45), per ogni fissato valore di j_1 bisogna calcolare una DFT di lunghezza 4. È possibile continuare ad applicare lo schema dividi e conquista.
Posto $w_{12}^{3j_0k_0} = e^{3\frac{-2\pi j_0 q_0}{12}} = e^{\frac{-2\pi j_0 q_0}{4}} = w_4^{j_0k_0}$, e $p_0 = 3j_0 + j_1$ calcoliamo:

$$F^*(k_0) := \sum_{j_0=0}^{3} f(3j_0 + j_1) w_{12}^{3j_0k_0} = \sum_{j_0=0}^{3} f(p_0) w_4^{j_0k_0} \tag{7.49}$$

Posto:

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

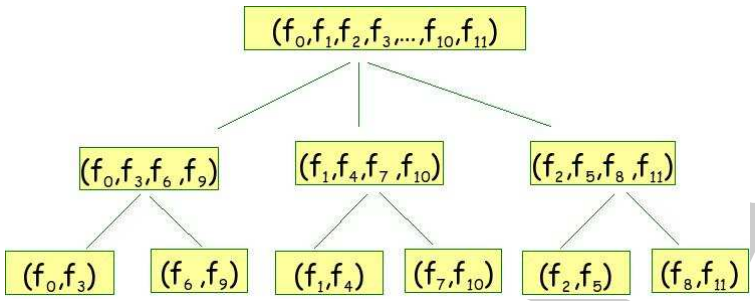

Figura 7.15: 4 DFT di lunghezza 3

allora:

$$F^*(k_0) = \underbrace{\sum_{j_1'=0}^{1} w_4^{2j_1'k_1'} \underbrace{\left[\sum_{j_0'=0}^{1} f(p_0) w_4^{2j_0'k_0'}\right]}_{DFT\ di\ lunghezza 2} w_4^{j_1'k_0'}}_{2\ DFT\ di\ lunghezza\ 2} \tag{7.50}$$

ovvero una DFT di lunghezza quattro è ricondotta a 2 DFT di lunghezza 2.
Una DFT di lunghezza 12 si può quindi esprimere mediante DFT di lunghezza 2, ovvero utilizzando la (7.48) e la (7.50):

$$F(4k_1+k_0) = \underbrace{\sum_{j_1=0}^{2} w_{12}^{4j_1k_1} \underbrace{\sum_{j_1'=0}^{1} w_4^{2j_1'k_1'} \underbrace{\left[\sum_{j_0'=0}^{1} f(p_0) w_4^{2j_0'k_0'}\right]}_{DFT\ di\ lunghezza 2} w_4^{j_1'k_0'}}_{2\ DFT\ di\ lunghezza\ 2} w_{12}^{j_1k_0}}_{6\ DFT\ di\ lunghezza\ 2} \tag{7.51}$$

La Figura 7.15 illustra lo schema dividi e conquista appena descritto.

♣

Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_N)$ di lunghezza $N = r_1 \cdot r_2$:

$$DFT[f]_k := F_k = \sum_{j=0}^{N-1} f_j w_N^{jk} \tag{7.52}$$

posto:

$$\begin{array}{llll} k = & r_1k_1 + k_0 & k_1 = 0,1,...,r_1-1 & k_0 = \ 0,1,...,r_1-1 \\ j = & r_2j_0 + j_1 & j_1 = 0,1,...,r_2-1 & j_0 = \ 0,1,...,r_2-1 \end{array}$$

allora:

$$F(r_1k_1+k_0) = \sum_{j_0=0}^{r_2-1} \sum_{j_1=0}^{r_1-1} f(r_2j_0+j_1) w_N^{(r_2j_0+j_1)(r_1k_1+k_0)} \tag{7.53}$$

Poiché:

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

e $w_N^{N j_0 k_1} = 1$ si ha che la (7.53) si può scrivere come:

$$F(r_1 k_1 + k_0) = \underbrace{\sum_{j_1=0}^{r_1-1} w_N^{r_1 j_1 k_1} \underbrace{\left[\sum_{j_0=0}^{r_2-1} f(r_2 j_0 + j_1) w_N^{r_2 j_0 k_0}\right]}_{DFT\ di\ lunghezza r_2} w_N^{j_1 k_0}}_{r_1\ DFT\ di\ lunghezza\ r_2} \tag{7.54}$$

Il calcolo di una DFT di lunghezza $N = r_1 \cdot r_2$ si riconduce al calcolo di r_1 DFT di lunghezza r_2. La classe degli algoritmi appena descritti prende il nome di algortimi *FFT mixed-radix*.
In generale, quindi, il calcolo di una DFT di lunghezza N è basato sulla decomposizione del problema in sottoproblemi di dimensione inferiore che dipende dalla fattorizzazione di N in particolare:

1. Se $N = r_1 r_2$ gli algoritmi FFT calcolano r_1 DFT di lunghezza r_2 (algoritmi *mixed-radix*).
2. Se $r_2 = r^p$ allora $N = r_1 r^p$ gli algoritmi FFT calcolano $p \cdot r_1$ DFT di lunghezza r .
3. Se $N = r^p$ gli algoritmi FFT calcolano p DFT di lunghezza r (algoritmi *radix-r*).

7.2.6 Complessità computazionale

L'algoritmo di Cooley e Tukey, consente il calcolo di una DFT di lunghezza N con una complessità di tempo che è significativamente inferiore a quella di un prodotto matrice-vettore.
In particolare l'algoritmo radix-2 è costituito da $log_2 N$ passi, ed in ciascuno di tali passi si calcolano $N/2$ DFT di lunghezza 2. La complessità di tempo $T(N)$ è data da:

$$T_{DFT}(N) = log_2 N \cdot \left[\frac{N}{2} \cdot \underbrace{T_{DFT}(2)}_{2}\right] = O(N log_2 N)$$

Nel grafico in Figura 7.16 sono confrontate la complessità computazionale della valutazione diretta della DFT e quella dell'algoritmo FFT radix-2.

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

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

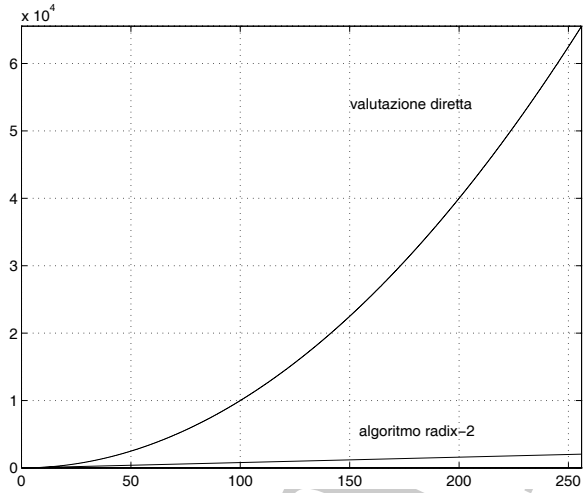

Figura 7.16: confronto tra la complessità computazionale della valutazione diretta della DFT di una sequenza di numeri complessi di lunghezza N e la FFT della medesima sequenza

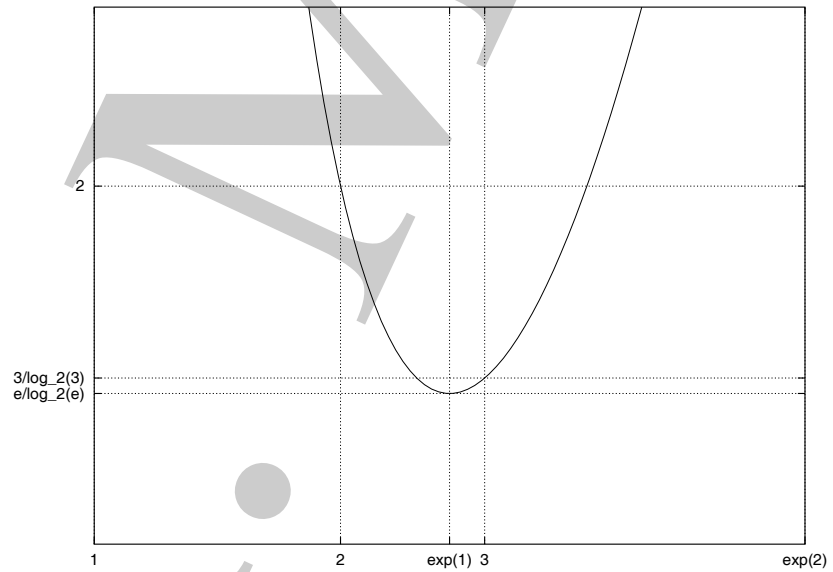

Figura 7.17: rappresentazione del fattore di proporzionalità $\frac{r}{log_2 r}$

Il fattore di proporzionalità $r/log_2 r$, come anche mostrato in Figura 7.17, 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).

7.3 Software matematico per la FFT

Nei paragrafi precedenti sono stati introdotti diversi tipi di algoritmi FFT, tali algoritmi sono implementati in numerosi package di dominio pubblico come, ad esempio, `FFTPACK` [1], `GO` di `GAMS` [1], `VFFTPACK` [1] disponibili sul sito web `http://www.netlib.org`.
Una vera e propria rivoluzione nell'utilizzo di software matematico per il calcolo della FFT si è avuta nel 1998, grazie allo sviluppo del package *Fast Fourier Transform in the West* (`FFTW`)[2].
Il software `FFTW` nel 1999 vince il premio *J. H. Wilkinson Prize for Numerical Software*, come il miglior prodotto sviluppato in termini di efficienza, accuratezza ed affidabiltà.
La Figura 7.18 mostra i tempi di esecuzione della libreria `FFTW` rispetto a quelli degli altri software.

La Figura 7.19 mostra un confronto tra la libreria `FFTW` e gli altri software in termini di accuratezza.

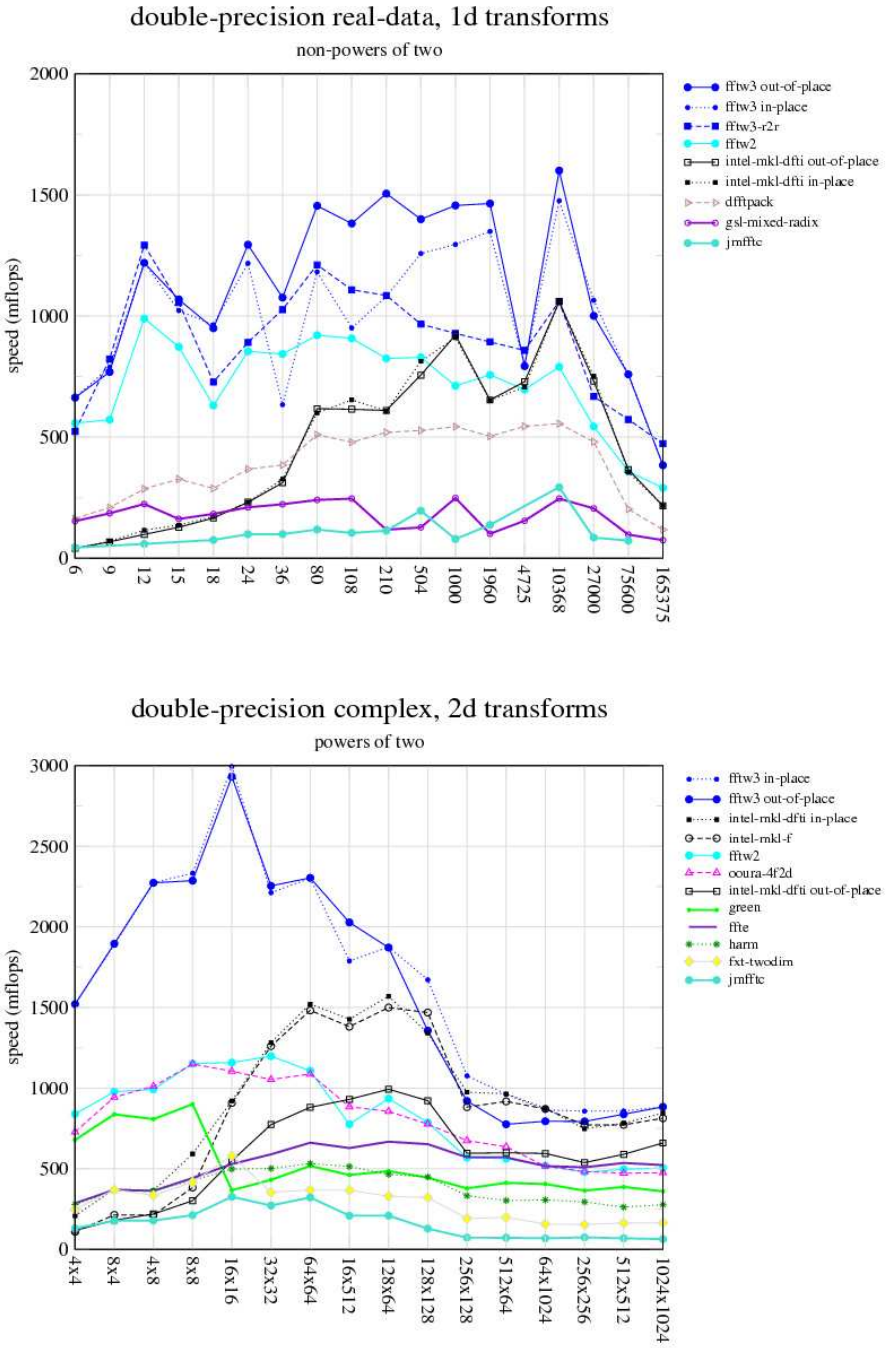

Figura 7.18: Test relativi allo speed-up eseguiti su un Pentium IV (Xeon), 2.2GHz, con compilatore `gcc`. A sinistra - FFT 1d di un vettore di complessi doppia precisione . A destra - FFT 1d di un vettore di reali doppia precisione.

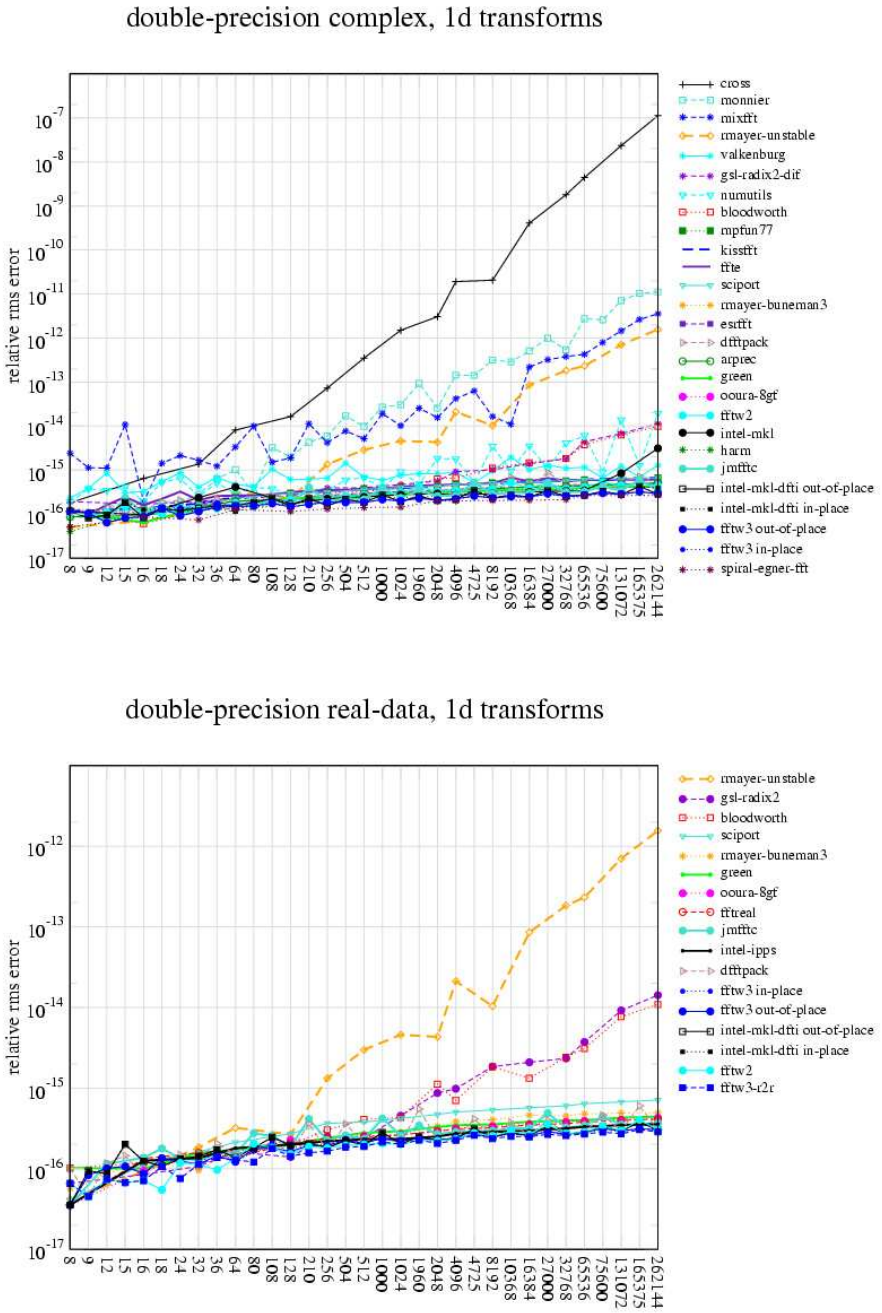

Figura 7.19: Test relativi all'accuratezza su un Pentium IV (Xeon), 2.2GHz, con compilatore gcc. A sinistra - FFT 1d di un vettore di complessi doppia precisione . A destra - FFT 1d di un vettore di reali doppia precisione.

`FFTW` è una collezione di routine scritte in C, per il calcolo di FFT m-dimensionali di vettori reali e complessi. Le sue principali caratteristiche sono:

a. scelta automatica del tipo di algoritmo FFT in base alla dimensione N del vettore di input;

b. generazione automatica di codice ottimizzato secondo le caratteristiche hardware e software dell'ambiente di calcolo;

Il punto a. rappresenta un evidente vantaggio in termini di facilità di utilizzo, per cui un utente non deve scegliere una strategia algoritmica per il calcolo della FFT. Il punto b. caratterizza la *portabilità* e la *performance* del software `FFTW`.
La libreria `FFTW`, costruita sulla base delle metodologie AEOS (Automated Empirical Optimization of Software) rendendo disponibile automaticamente software ottimizzato per la specifica piattaforma di calcolo su cui è implementata la libreria (uso opportuno dei registri, della memoria cache e dell'unità logico-aritmetica). Le sue routine sono anche alla base delle funzioni `MATLAB` della versione 7.0 e del pacchetto software `Octave`.

7.3.1 La libreria FFTW

La libreria `FFTW` è strutturata in routine, dette **codelets**, che sono combinabili fra loro. Ciascun codelets esegue il calcolo di una DFT di lunghezza n fissata.

Nel seguito si descrivono alcune parti di un codice scritto in C, che utilizza codelets di `FFTW`.

```
#include <fftw3.h>
...
{ fftw_complex *in, *out;
fftw_plan p;
in = fftw_malloc(sizeof(fftw_complex)*n);
/* Generazione del plan */
out =fftw_malloc(sizeof(fftw_complex)*n);
p= fftw_plan_dft_1d(n,in,out,FFTW_FORWARD,FFTW_ESTIMATE);
...
/* Esecuzione del plan (executor) */
fftw_execute(p);
...
fftw_destroy_plan(p);
fftw_free(in);
fftw_free(out);}
```

`genfft` è il compilatore che a partire da n, lunghezza del vettore, genera il codice più efficiente per il calcolo della DFT.
Tale codice è detto `plan`, creato dalla chiamata alla routine:

```
fftw_plan_dft_1d(...);
```

In particolare, il `plan` è un insieme di istruzioni, che indica in quale ordine devono essere eseguiti i codelets. Nelle versioni attuali di `FFTW` il plan ha struttura di albero e può essere memorizzato per successivi impieghi.
Le istruzioni contenute nel `plan` vengono eseguite da un opportuno programma detto `executor`, mediante l'istruzione:

```
fftw_execute(...);
```

L'`executor` è la componente di `FFTW` che si occupa del calcolo della DFT seguendo le istruzioni del plan. L'`executor` implementa le diverse strategie algoritmiche per il calcolo della FFT.
Inoltre, l'utente deve selezionare il `plan` opportuno per il calcolo di FTT 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 per una corretta compilazione, utilizzata, tra l'altro, nelle routine per l'allocazione ed il rilascio di memoria:

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

7.4 Alcuni problemi da risolvere con il calcolatore

Esercizio 1 Assegnate due matrici circolanti C_1 e C_2 descrivere un procedimento efficiente per effettuare tramite FFT il prodotto righe per colonne di C_1per C_2. Discutere la riduzione della complessità computazionale.

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

Esercizio 3 Assegnato il vettore di numeri complessi:

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

si calcoli, utilizzando la libreria **fftw3** la fft $u := fft(h)$ 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 $v := fft(g)$ fft del vettore g.
(c) Individuare il legame che sussiste tra u e v ed il calcolo della FFT-mixed radix nel caso $N = 15$.

Esercizio 4 Assegnato il vettore di numeri reali:

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

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

Esercizio 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 $G = (37.0000, -1.0000 + 18.0000i, 5.0000, -1.0000 - 18.0000i)$ 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.

Esercizio 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 determini di quale proprietà gode il vettore r.
(c) Descrivere lo schema ad albero in base al quale si esprime il calcolo della FFT-mixed radix nel caso $N = 20$.

Bibliografia

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

[2] E. Brigham, *The Fast Fourier Transform and its Applications*, Prentice Hall Signal Processing Series.

[3] J. James, *A Student's Guide to Fourier Transforms*, Cambridge University Press.

[4] J. Walker, *Fast Fourier Transforms*, CRC Press.

[5] N. Higham, *Accuracy and Stability of Nuemrical Algorithms*, SIAM 2002.

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