UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
LAUREA IN INGEGNERIA INFORMATICA

Corso di Calcolo Numerico
A.A. 2008/2009
PROF.SSA: *Maria Morandi Cecchi*

DFT E FFT: PROPRIETÀ, STABILITÀ NUMERICA E APPLICAZIONI

STUDENTI: *Marco Bettiol* 586580
Antonio Quercia 588537

Padova, 13 maggio 2009

Indice

Capitolo 1

DFT

1.1 Definizione

Un vettore x di lunghezza N può essere visto come una sequenza $x_0, ..., x_{N-1}$ di N numeri complessi. Si definisce X *trasformata discreta di Fourier (DFT, discrete Fourier transform)* di x la sequenza $X_0, ..., X_{N-1}$ espressa dalla seguente relazione:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\jmath \frac{2\pi}{N} kn} \quad k = 0, ..., N-1 \tag{1.1}$$

Ricordando la *Formula di Eulero*

$$e^{\pm 2\pi\omega x} = \cos(2\pi\omega x) \pm i \sin(2\pi\omega x)$$

è evidente come, in generale, X possa risultare in una sequenza complessa sebbene x fosse reale.
La relazione inversa della **??**, chiamata *trasformata discreta inversa di Fourier (IDFT, inverse discrete Fourier transform)* è data da:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\jmath \frac{2\pi}{N} kn} \quad k = 0, ..., N-1 \tag{1.2}$$

Da **??** si può notare che i numeri complessi X_k rappresentano l'ampiezza e la fase delle diverse componenti sinusoidali del 'segnale' d'ingresso x_n.
Introduciamo quindi la seguente notazione per indicare DFT e IDFT

$$X = \mathcal{F}x$$
$$x = \mathcal{F}^{-1}X$$

e inoltre indichiamo esplicitamente con x_N (pedice maiuscolo) un vettore x N-dimensionale e con X_N la sua trasformata.

1.2 Proprietà

La maggior parte delle proprietà valide per la trasformata di Fourier a tempo continuo hanno un analogo equivalente anche per la DFT.
Il cambiamento più significativo da apportare ai teoremi per adattarli al caso finito-discreto è quello di modificare gli indici per tenere conto della periodicità della DFT, ovvero prendere gli indici modulo **N**.
Date X e Y le DFT di x e y rispettivamente, valgono:

- Completezza:
  La DFT è una trasformazione *lineare e invertibile*:

  $$\mathcal{F} : \mathbb{C}^N \longrightarrow \mathbb{C}^N \tag{1.3}$$

  dove $\mathbb{C}$ è l'insieme complesso.

  $$x \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\rightleftharpoons}} X$$

  DFT e IDFT mappano vettori complessi N-dimensionali in vettori complessi N-dimensionali attraverso una corrispondenza biunivoca.

- Ortogonalità:
  I vettori $e^{\jmath \frac{2\pi}{N} kn}$ formano una *base ortogonale* per l'insieme dei vettori complessi N-dimensionali:

  $$\sum_{n=0}^{N-1} (e^{\jmath \frac{2\pi}{N} kn})(e^{-\jmath \frac{2\pi}{N} k'n}) = N\delta_{kk'} \tag{1.4}$$

  dove $\delta_{kk'}$ è il delta di Kronecker.

  $$\delta_{kk'} = \begin{cases} 1 & \text{if } k = k' \\ 0 & \text{if } k \neq k' \end{cases}$$

  Questa condizione può essere sfruttata per ricavare la formula della IDFT dalla definizione di DFT.

- Il teorema di Plancherel e il teorema di Parseval:
  Il *teorema di Plancherel* dice che:

  $$\sum_{n=0}^{N-1} x_n y_n^* = \frac{1}{N} \sum_{k=0}^{N-1} X_k Y_k^* \tag{1.5}$$

  Il *teorema di Parseval* è un caso particolare del teorema di Plancherel e afferma che:

  $$\sum_{n=0}^{N-1} |x_n|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X_k|^2 \tag{1.6}$$

- Periodicità:
  Se la relazione **??** che definisce la DFT è valutata per tutti i k interi anzichè soltanto $k = 0, ..., N-1$, allora la sequenza infinita risultante è un'*estensione periodica* di periodo N della DFT. Tale periodicità può essere mostrata direttamente dalla definizione:

$$\begin{aligned} X_{k+N} &= \sum_{n=0}^{N-1} x_n e^{-\jmath \frac{2\pi}{N}(k+N)n} && (1.7)\\ &= \sum_{n=0}^{N-1} x_n e^{-\jmath \frac{2\pi}{N}kn} e^{-\jmath 2\pi n} && (1.8)\\ &= \sum_{n=0}^{N-1} x_n e^{-\jmath \frac{2\pi}{N}kn} && (1.9)\\ &= X_k && (1.10) \end{aligned}$$

  dove si è utilizzato il fatto che $e^{-2\pi\jmath} = 1$. Allo stesso modo può essere mostrato che la **??** porta a un'estensione periodica della IDFT.

- Il *Circular shift theorem*:
  Moltiplicare x_N per una fase lineare $e^{-\jmath \frac{2\pi}{N}nm}$, per un qualche m intero, è equivalente ad uno shift circolare di X_N di m posizioni: X_k diventa X_{k-m}, il cui pedice si deve intendere come *modulo N*. Analogamente, uno shift circolare di x_N corrisponde alla moltiplicazione di X_N per una fase lineare.

$$se\ \mathcal{F}((x_n))_k = X_k \tag{1.11}$$

$$allora\ \mathcal{F}((x_n e^{\jmath \frac{2\pi}{N}nm}))_k = X_{k-m} \tag{1.12}$$

$$e\ \mathcal{F}((x_{n-m}))_k = X_k e^{-\jmath \frac{2\pi}{N}km} \tag{1.13}$$

- *Linear Convolution Theorem*
  Siano $x = x_N$ e $y = y_N$ due vettori arbitrari N-dimensionali e sia $z = x * y$, un vettore di $2N-1$ componenti ottenuto dalla convoluzione lineare $*$. Allora,

$$z_i = \sum_{j=\max\{0,i-N+1\}}^{\min\{i,N-1\}} x_i y_{i-j} \qquad 0 \le i \le 2N-1 \tag{1.14}$$

  dove i limiti superiore e inferiore della sommatoria sono scelti in modo tale che i e $i-j$ siano sempre nell'intervalllo $[0\ N-1]$. Possiamo calcolare la convoluzione lineare utilizzando la DFT nel seguente modo:

$$(z|0_1) = \mathcal{F}_{2N}^{-1}(\mathcal{F}_{2N}(x|0_N) \odot \mathcal{F}_{2N}(y|0_N) \tag{1.15}$$

dove $\odot$ indica il prodotto componente per componente e $(x|0_k)$ indica il vettore ottenuto estendendo il vettore x_N con padding di k zero in coda.

- *Cyclic Convolution Theorem*
  Siano $x = x_N$ e $y = y_N$ due vettori arbitrari N-dimensionali, definiamo *convoluzione ciclica* di x e y, indicata con $x \circledast y$, il vettore w di N componenti dato da:

$$w_i = \sum_{j=0}^{N-1} x_i y_{(i-j) \mod N} \qquad 0 \leq i \leq N-1 \tag{1.16}$$

  Possiamo calcolare la convoluzione ciclica utilizzando la DFT nel seguente modo:

$$w = \mathcal{F}_N^{-1}(\mathcal{F}_N x \odot \mathcal{F}_N y) \tag{1.17}$$

  dove $\odot$ indica il prodotto componente per componente.

1.3 Defizione Matriciale

Dalla defizione **??** di DFT emerge immadiatamente una interpretazione matriciale della *trasformata discreta di Fourier*

$$X = \mathcal{F}(x) = F_N x \tag{1.18}$$

con

$$x = \begin{pmatrix} x_0 \\ x_1 \\ ... \\ x_{N-1} \end{pmatrix} \tag{1.19}$$

e

$$F_N = \begin{pmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & ... & \omega_N^{0\cdot(N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & ... & \omega_N^{1\cdot(N-1)} \\ ... & ... & ... & ... \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & ... & \omega_N^{(N-1)\cdot(N-1)} \end{pmatrix} \tag{1.20}$$

Il termine

$$\omega_N = e^{-j\frac{2\pi}{N}} \tag{1.21}$$

è una *radice complessa N-esima dell'unita.*
L'elemento in posizione (i, j) della matrice è ω_N^{ij}, con $i, j = 0, 1, ..., N-1$.
Analogamente è possibile esprimere la IDFT come:

$$x = \mathcal{F}^{-1}(X) = F_N^{-1} X \tag{1.22}$$

Dove F_N^{-1} è la *matrice inversa* di F_N e può essere facilmente ottenuta da F attraverso

$$F_N^{-1} = \frac{1}{N} F_N^* \tag{1.23}$$

dove pertanto in posizione (i, j) di F_N^{-1} è presente il valore $\frac{1}{N}\omega_N^{-ij}$

1.3.1 Analogie tra DFT e IDFT

Questa affinità tra F_N e F_N^{-1} mette in risalto come in realtà la procedura per il calcolo di DFT e IDFT sia esattamente la stessa a meno di minimi adattamenti. In altre parole è possibile calcolare la IDFT attraverso lo stesso algoritmo di *forward computation* della DFT attraverso un semplice preprocessing dell'input e moltiplicazione per un fattore di scala:

$$\mathcal{F}^{-1}(\{x_n\}) = \frac{1}{N}\mathcal{F}(\{x_{N-n}\}) \tag{1.24}$$

dove l'operazione effettuata sull'input risulta essere un *inversione dell'input modulo N*:

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ ... \\ x_{N-2} \\ x_{N-1} \end{pmatrix} \longrightarrow \begin{pmatrix} x_0 \\ x_{N-1} \\ x_{N-2} \\ ... \\ x_2 \\ x_1 \end{pmatrix} \tag{1.25}$$

Analogamente è possibile sfruttare la seguente proprietà per il calcolo della IDFT

$$\mathcal{F}^{-1}(x) = \frac{1}{N}(\mathcal{F}(x^*))^* \tag{1.26}$$

coniugando l'input, applicando la DFT, coniugando l'output e quindi scalando.

1.4 Segnali di durata finita: relazione tra trasformata di Fourier e DFT

Capitolo 2

Calcolo della DFT

La diffusione nell'utilizzo della DFT è principalmente dovuta all'introduzione, avvenuta negli anni '60, di una classe di algoritmi computazionale efficienti per la sua valutazione: la *(Fast Fourier Transform, i.e. trasformata veloce di Fourier).* Essa rende possibile in taluni casi l'elaborazione in tempo reale (sistemi di navigazione, trasmissione e processo di segnali vocali o televisivi) e consente di risolvere alcuni problemi di grandi dimensioni che risulterebbero altrimenti intrattabili.
L'FFT viene normalmente attribuita a Cooley e Tukey. Di fatto questo particolare algoritmo venne scoperto (ed usato) da Gauss nel XIX secolo. Pare che in realtà l'algoritmo sia stato riscoperto ed utilizzato varie volte (anche prima del lavoro di Cooley e Tuckey menzionato sopra). Grazie alla FFT, infatti, è possibile implementare al calcolatore operazioni nel dominio della frequenza, cioè utilizzando le trasformate di Fourier, con complessità di calcolo inferiore a quella ottenibile implementando lo stesso calcolo nel dominio del tempo.

2.1 Attraverso la definizione

La tecnica di calcolo banale passa per la definizione stessa di DFT

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn} \quad k = 0, ..., N-1$$

ed essendo

$$\mathcal{F} : \mathbb{C}^N \longrightarrow \mathbb{C}^N$$

ci sono N termini da calcolare, ognuno dei quali richiede N moltiplicazioni e $N-1$ addizzioni complesse. Anche evidando di calcolare le moltiplicazioni per $\omega_N^0 = 1$ si devono comunque effettuare in totale $2(N-1)^2$ operazioni complesse. Il costo è pertanto $O(N^2)$

Nel dettaglio ogni moltiplicazione complessa si sviluppa in $(a+jb)(c+jd) = ac - bd + j(ad + bc)$ richiedendo 4 moltiplicazioni reali e 2 addizioni e pertanto il numero di operazioni macchina totali risultante è $4(N-1)^2$ moltiplicazioni e $2N(N-1) + 2(N-1)^2 = 2(N-1)(2N-1)$ addizioni.

2.2 Algoritmi per il calcolo di FFT

Sia $T(N)$ il tempo di calcolo associato ad un problema di cardinalità N e supponiamo per semplicità $N = 2^\nu$.
Se partizioniamo gli N dati in due sottoinsiemi di $N/2$ dati:

$$N \longrightarrow \frac{N}{2} + \frac{N}{2}, \tag{2.1}$$

risolviamo i due sottoproblemi e ricombiniamo le due soluzioni, il tempo di calcolo diviene:

$$T(N) = 2 \cdot T\left(\frac{N}{2}\right) + R(N) \tag{2.2}$$

con $R(N)$ pari al costo di combinazione. Allo stesso risultato si arriva se $R(N)$ rappresenta il costo di costruzione e/o ricombinazione dei due sottoproblemi.

Non è difficile verificare che se $R(N) = O(N)$, cioè il costo di ricombinazione è lineare nei dati e si itera il procedimento fino ad N sottoinsiemi di un solo dato, il tempo di calcolo relativo alla procedura diventa:

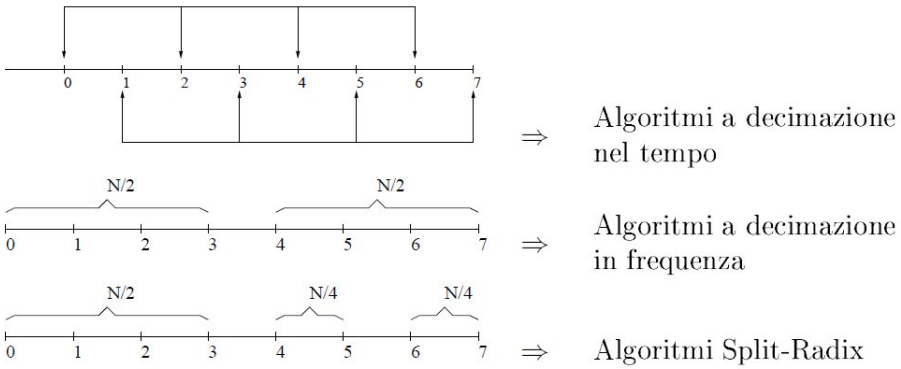
$$T(N) = O(N \cdot \log_2(N)) \tag{2.3}$$

qualsiasi sia la complessita di calcolo originale $T(N)$, purchè di tipo polinomiale, cioè del tipo

$$T(N) = O(N^\alpha), \quad \alpha \in \mathbb{R} \tag{2.4}$$

Ovviamente, minore è la complessità originaria (quindi l'esponente α) e più piccolo sarà il tempo di calcolo al quare si arriva applicando il principio *'divide et impera'*; in ogni caso, comunque, esso sarà dell'ordine di $N \cdot \log_2 N$

Osserviamo inoltre che il principio *'divide et impera'* conduce spontaneamente a strutture di calcolo parallele. Con $N = 2_\nu$, le partizioni dei dati più efficienti sono le più 'spontanee', e sono 'schematizzate' nella figura seguente per $N = 8$



2.2.1 Cooley-Tukey Algorithm - DIT

Gli algoritmi di decimazione nel tempo si costituiscono a partire dalla seguente scomposizione ricorsiva.
Consideriamo un vettore x di lunghezza $N = 2^\nu$. Dividiamo i valori in ingresso in elementi di posizione pari e dispari

$$n = \begin{cases} 2 \cdot r \\ 2 \cdot r + 1 \end{cases}$$

Sapendo che $\omega_N^2 = \omega_{\frac{N}{2}}$, la componente X_k della DFT può essere espressa come

$$X_k = \sum_{n=0}^{N-1} x_n \omega_N^{nk} \qquad \text{con} \quad \omega_N = e^{-\jmath \frac{2\pi}{N}} \tag{2.5}$$

$$= \sum_{r=0}^{N/2-1} x_{2r} \omega_N^{2rk} + \sum_{r=0}^{N/2-1} x_{2r+1} \omega_N^{(2r+1)k} \tag{2.6}$$

$$= \underbrace{\sum_{r=0}^{N/2-1} x_{2r} \omega_{\frac{N}{2}}^{rk}}_{X_{even}} + \omega_N^k \underbrace{\sum_{r=0}^{N/2-1} x_{2r+1} \omega_{\frac{N}{2}}^{rk}}_{X_{odd}} \tag{2.7}$$

$$= X_{even} \quad + \quad \omega_N^k \cdot X_{odd} \qquad k = 0, 1, ..., N-1 \tag{2.8}$$

Entrambe le sommatorie in (**??**) possono quindi essere interpretate come DFT di taglia $N/2$: la prima coinvolge gli elementi in posizione pari $\{x_{2k} | k = 0, 1, ..., N/2 - 1\}$ mentre la seconda gli elementi in posizione dispari $\{x_{2k+1} | k = 0, 1, ..., N/2 - 1\}$. Ci siamo quindi ricondotti a due sottoproblemi analoghi a quello di partenza ma di daglia dimezzata.
Definiendo $y_r = x_{2r}$ e $z_r = x_{2r+1}$ e quindi le rispettive DFT

$$Y_k = \sum_{r=0}^{\frac{N}{2}-1} y_r \omega_{\frac{N}{2}}^{rk}, \qquad k = 0, 1, ..., N/2 - 1 \tag{2.9}$$

e

$$Z_k = \sum_{r=0}^{\frac{N}{2}-1} z_r \omega_{\frac{N}{2}}^{rk}, \qquad k = 0, 1, ..., N/2 - 1 \tag{2.10}$$

dopo che la soluzione di questi due problemi è stata ricorsivamente calcolata è possibile calcolare i primi $N/2$ termini di X attraverso

$$X_k = Y_k + \omega_N^k Z_k, \qquad k = 0, 1, ..., N/2 - 1 \tag{2.11}$$

e utilizzando il fatto che $\omega_N^{\frac{N}{2}+k} = -\omega_N^k$ e $\omega_{\frac{N}{2}}^{\frac{N}{2}} = 1$, i rimanenti termini sono dati

da:

$$X_{k+\frac{N}{2}} = \sum_{r=0}^{\frac{N}{2}-1} y_r \omega_{\frac{N}{2}}^{(r+\frac{N}{2})k} + \omega_N^{(k+\frac{N}{2})} \sum_{r=0}^{\frac{N}{2}-1} z_r \omega_{\frac{N}{2}}^{(r+\frac{N}{2})k} \tag{2.12}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} y_r \omega_{\frac{N}{2}}^{rk} + \omega_N^k \sum_{r=0}^{\frac{N}{2}-1} z_r \omega_{\frac{N}{2}}^{rk} \tag{2.13}$$

$$= Y_k - \omega_N^k Z_k, \qquad k = 0, 1, ..., N/2 - 1 \tag{2.14}$$

2.2.2 Gentleman-Sande Algorithm - DIF

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

Schema di scomposizione

Se dividiamo gli $N = 2^\nu$ elementi di x_N nelle due sotto sequenze $x_{head} = (x_0, x_1, ..., x_{\frac{N}{2}-1})$ e $x_{tail} = (x_{\frac{N}{2}}, x_{\frac{N}{2}+1}, ..., x_{N-1})$, la DFT di x_N si può scrivere nella forma

$$X_k = \sum_{n=0}^{N-1} x_n \omega_N^{nk} \qquad \text{con} \quad \omega_N = e^{-j\frac{2\pi}{N}} \tag{2.15}$$

$$= \sum_{n=0}^{N/2-1} x_n \omega_N^{nk} + \sum_{n=N/2-1}^{N-1} x_n \omega_N^{nk} \qquad \text{per} \quad n' = n - \frac{N}{2} \tag{2.16}$$

$$= \sum_{n=0}^{N/2-1} x_n \omega_N^{nk} + \sum_{n'=0}^{N/2-1} x_{\frac{N}{2}+n'} \omega_N^{(\frac{N}{2}+n')k} \qquad \omega_N^{N\frac{k}{2}} = (-1)^k \tag{2.17}$$

$$= \sum_{n=0}^{N/2-1} \left[x_n + (-1)^k x_{\frac{N}{2}+n} \right] \tag{2.18}$$

Notiamo che la (**??**) non è una DFT su $N/2$ punti in quanto è presente il termine ω_N^{nk} invece di $\omega_{\frac{N}{2}}^{nk}$

Procedendo separatamente con il calcolo dei valori di X_k di posto pari $\left((-1)^K = 1\right)$ e dispari $\left((-1)^K = -1\right)$ si ottiene

$$X_{2r} = \sum_{n=0}^{N/2-1} \left[\underbrace{x_n + x_{\frac{N}{2}+n}}_{g_n} \right] \omega_{N/2}^{rn} \qquad r = 0, 1, ..., N/2-1 \tag{2.19}$$

$$X_{2r+1} = \sum_{n=0}^{N/2-1} \left[\underbrace{(x_n - x_{\frac{N}{2}+n})}_{h_n} \omega_N^n \right] \omega_{N/2}^{rn} \tag{2.20}$$

Il calcolo della DFT su $N = 2^\nu$ punti viene quindi ricondotto a

1. il calcolo di due segnali di lunghezza $N/2$ corrispondenti rispettivamente alla somma di $x_{head} + x_{tail}$ e alla differenza $x_{head} - x_{tail}$

$$g_n = x_n + x_{\frac{N}{2}+n} \qquad n = 0, 1, ..., N/2-1 \tag{2.21}$$

$$h_n = x_n - x_{\frac{N}{2}+n} \tag{2.22}$$

   che corrisponde a $N/2$ DFT su coppie punti, $[x_n \; x_{\frac{N}{2}+n}]$. Infatti

$$F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.23}$$

2. alla moltiplicazione di h_n per ω_N^n
3. al calcolo di due DFT su $N/2$ punti, una su g_n e l'altra su $h_n\omega_N^n$

Con l'algoritmo DIF quindi si precondizionano i dati dei due blocchi di lunghezza $N/2$ in modo che una trasformata su $N/2$ dati dia i valodi di X_k per k pari e l'altra sempre su $N/2$ dati dia i valori di X_k per k dispari.

Complessità

2.2.3 Split-radix Algorithm

2.2.4 Goertzel's Algorithm

2.2.5 Thomas and Prime Factor Algorithm (Good)

Capitolo 3

Applicazioni

3.1 FFT of two Real Functions Simultaneosly

Nell'applicazione della FFT spesso consideriamo solo funzioni reali del tempo mentre le funzioni in frequenza utilizzate nei calcoli sono, in generale, funzioni complesse. Pertanto, un comune programma in grado determinare la DFT e la sua inversa è scritto in modo da ricevere in input una forma d'onda complessa

$$H(n) = \sum_{k=0}^{N-1} [h_r(k) + jh_i(k)]\, e^{-j\frac{2\pi}{N}nk} \tag{3.1}$$

Sfruttando la proprietà (**??**) per il calcolo dell'inversa è possibile riscrivere

$$h(k) = \frac{1}{N}\left[\sum_{n=0}^{N-1} [H_r(n) + jH_i(n)]^* e^{-j\frac{2\pi}{N}nk}\right]^* \tag{3.2}$$

e poichè entrambe contengono $e^{-j\frac{2\pi}{N}nk}$ lo stesso programma può essere usato per calcolare sia la trasformata diretta che la sua inversa.

Se i dati in ingresso sono reali allora la loro parte immaginaria risulterà essere nulla. Tuttavia poichè i programmi effettuano comunque le operazioni anche per i coefficienti posti a zero, otteniamo uno spreco di capacità di calcolo. Varie tecniche possono essere adottate per aumentare l'efficienza di calcolo di sequenze rali. Volendo infatti calcolare le trasformate di $h(k)$ e $g(k)$ è possibile considerare l'ingresso fittizio $y(k) = h(k) + jg(k)$

Per la linearità della DFT

$$\begin{aligned} Y(n) &= H(n) + jG(n) \\ &= [H_r(n) + jH_i(n)] + j[G_r(n) + jG_i(n)] \\ &= [H_r(n) - G_i(n)] + j[H_i(n) + G_r(n)] \\ &= \mathrm{Re}(n) + j\mathrm{Imm}(n) \end{aligned}$$

Bibliografia

[1] Nicholas J. Higham, *Accuracy and stability of numerical algorithms*, Edizione 1, SIAM, 1996, pg. 466-471

[2] W. M. Gentleman, G. Sande, *Fast Fourier Transforms: for fun and profit*, AFIPS Joint Computer Conferences, Proceedings of the November 7-10, ACM, 1966, pg. 563-578

[3] M. Tasche, H. Zeuner, *Worst and Average Case Roundoff Error Analysis for FFT*, BIT Numerical Mathematics, Vol. 41 No.3, Springer Netherlands, 2001, pg. 563-581

[4] http://en.wikipedia.org/wiki/Discrete_Fourier_transform

[5] G. Pucci, *Convolutions and the Discrete Fourier Transfom*, materiale a supporto del corso di Dati e Algoritmi 2

/