UNIVERSITÀ DEGLI STUDI DI BOLOGNA

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
CORSO DI LAUREA IN FISICA

Analisi wavelet multirisoluzione: dalla teoria matematica ad una possibile applicazione nell'ambito dell'esperimento ALICE

Tesi di Laurea
di:
Matteo Masotti

Relatore:
Prof. Enzo Gandolfi

Correlatori:
Ing. Ignazio D'Antone
Dott. Davide Falchieri

Anno Accademico 2000–2001— Sessione I

UNIVERSITÀ DEGLI STUDI DI BOLOGNA

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
CORSO DI LAUREA IN FISICA

Analisi wavelet multirisoluzione: dalla teoria matematica ad una possibile applicazione nell'ambito dell'esperimento ALICE

Tesi di Laurea
di:
Matteo Masotti

Relatore:
Prof. Enzo Gandolfi

Correlatori:
Ing. Ignazio D'Antone
Dott. Davide Falchieri

Parole chiave: Compressione, Decompressione, Wavelets, Multirisoluzione, ALICE

Anno Accademico 2000–2001— Sessione I

Indice

Introduzione

L'analisi wavelet multirisoluzione è una tecnica di analisi relativamente recente, che sta subendo una crescente diffusione, sia nell'ambito della compressione di segnali mono-dimensionali, che nell'ambito della compressione di segnali bi-dimensionali.

Gli obiettivi del presente lavoro di tesi sono, lo sviluppo di un algoritmo di compressione basato sull'analisi wavelet multirisoluzione, finalizzato alla compressione dei dati provenienti dai rivelatori SDD (*Silicon Drift Detectors*), appartenenti all'*Inner Tracking System* dell'esperimento ALICE (*A Large Ion Collider Experiment*), la valutazione delle prestazioni di tale algoritmo, sia in termini di predisposizione alla compressione dei dati, che in termini di errore nella ricostruzione dei dati originali a partire da quelli compressi, e da ultimo, il confronto diretto con le prestazioni ottenute mediante l'algoritmo attualmente in uso, sviluppato presso l'Istituto Nazionale di Fisica Nucleare di Torino.

Nel Capitolo 1 sono introdotti i concetti base relativi alla compressione dati: in particolare, è puntata l'attenzione sulla codifica *Run Length*, Delta e Huffman, tecniche di codifica su cui si basa l'algoritmo di compressione sviluppato presso l'Istituto Nazionale di Fisica Nucleare di Torino.
Nel Capitolo 2 sono trattati approfonditamente i vari aspetti dell'analisi wavelet: in particolare, è discussa la necessità dell'analisi wavelet continua, come alternativa all'analisi di Fourier ed alla *Short Time Fourier Transform*, è approfondita l'analisi wavelet multirisoluzione, con una attenzione specifica rivolta alla formalizzazione matematica formulata da S. G. Mallat, è discussa l'implementazione dell'analisi wavelet multirisoluzione in termini di filtri digitali, e da ultimo, sono discusse due implementazioni esemplificative dell'analisi wavelet multirisoluzione, una mono-dimensionale ed una bi-dimensionale.

Nel Capitolo 3 è descritto in maniera sintetica l'esperimento ALICE, con particolare riferimento alla struttura dell'*Inner Tracking System* del rivelatore principale, alla struttura dei rivelatori SDD ad esso appartenenti, ed alla struttura dell'elettronica di *drift*, che gestisce i dati a partire dai rivelatori fino al sistema di acquisizione dati (*Data AcQuisition*, DAQ); è inoltre trattato l'algoritmo di compressione sviluppato presso l'Istituto Nazionale di Fisica Nucleare di Torino, la sua implementazione attraverso il sottomodulo CARLOS, e le sue prestazioni in termini di valori di compressione raggiunti, ed in termini di errore nella ricostruzione dei *clusters* di carica originali, a partire dai dati compressi. Nel Capitolo 4 è descritto lo sviluppo dell'algoritmo di compressione basato sull'analisi multirisoluzione; in particolare, è descritta la prima fase della ricerca, tesa alla individuazione delle modalità di base con cui implementare l'algoritmo nell'ambito della compressione dei dati provenienti dai rivelatori SDD, è descritta la seconda fase della ricerca, sviluppata attraverso il Wavelet Toolbox di Matlab, tesa all'individuazione della configurazione dell'algoritmo multirisolutivo maggiormente prestante, sia in termini di predisposizione alla compressione dei dati, che in termini di errore nella ricostruzione dei dati originali a partire da quelli compressi; è descritta, inoltre, la terza fase della ricerca, sviluppata attraverso Simulink ed il Fixed-Point Bockset, tesa all'individuazione di una possibile architettura attraverso cui implementare l'algoritmo multirisolutivo, e da ultimo, è descritta la subroutine FORTRAN, richiamante alcune funzioni della libreria HBOOK di PAW, sviluppata al fine di consentire un confronto diretto tra le prestazioni dell'algoritmo di compressione multirisolutivo e quelle dell'algoritmo sviluppato presso l'Istituto Nazionale di Fisica Nucleare di Torino.

Capitolo 1

Introduzione alla compressione dati

In questo capitolo verranno introdotti i concetti base riguardanti la compressione dati [1].
In particolare, si focalizzerà l'attenzione sulla codifica *Run Length*, Delta e Huffman, su cui si basa l'algoritmo di compressione, implementato nel sottomodulo CARLOS dell'esperimento ALICE, e descritto nel Paragrafo 3.3.

1.1 Compressione dati

La compressione dati è la disciplina che si occupa della rappresentazione dell'informazione in maniera compatta, tramite l'identificazione e lo sfruttamento delle particolarità della struttura dei dati.

Ad esempio, il codice Morse è un forma di compressione dati applicata alla comunicazione tramite telegrafo, e finalizzata alla riduzione del tempo medio richiesto per l'invio di un messaggio; partendo dalla considerazione che certe lettere dell'alfabeto vengono usate più frequentemente di altre, tale codice assegna ad ogni lettera una sequenza identificativa costituita da punti ($\cdot$) e tratti ($-$), tanto più lunga, quanto più bassa è la probabilità di occorrenza della lettera, e tanto più corta, quanto più alta è la probabilità di tale occorrenza: ad esempio, i codici identificativi di lettere molto frequenti come la a o la e sono, rispettivamente, $\cdot -$ e $\cdot$, mentre, il codice di una lettera poco frequente, come la q, è $- - \cdot -$.

1.1.1 Struttura delle sequenze di dati

La compressione ottenuta attraverso il codice Morse, sfrutta la struttura statistica della sequenza da comprimere, ovvero, una struttura che mostra una probabilità di occorrenza più elevata per certe lettere e più bassa per altre; tuttavia, la struttura statistica, non è certamente l'unica esistente, infatti, vari sono i tipi di strutture che si possono trovare in natura, e che possono essere sfruttati in termini di compressione.

Si pensi, ad esempio, alla voce: una configurazione fisica particolare degli organi fonatori, quali l'apparato respiratorio, laringeo e risonatore, determina l'emissione di un ben determinato tipo di suono; in altri termini, cioè, la struttura specifica del segnale sonoro emesso, è determinata dalla meccanica degli organi fonatori.
In questo caso, sfruttare la struttura del segnale ai fini della compressione, significa rappresentare la sequenza sonora emessa, attraverso i parametri che ne contraddistinguono la fonazione, piuttosto che attraverso il numeroso insieme di campioni che identifica la sequenza sonora vera e propria.

La compressione dati può anche trarre profitto da una eventuale struttura ridondante del segnale, una struttura, cioè, che contiene più informazioni di quelle realmente indispensabili.
Si pensi, ad esempio, alla trasmissione di una sequenza sonora caratterizzata da frequenze comprese entro un intervallo molto ampio, e che debba essere udita da un essere umano: poiché l'intervallo di frequenze udibili dall'orecchio umano va da circa 20 Hz, per le frequenze più basse, fino a circa 20 KHz, per le frequenze più alte, tutte le frequenze non incluse in questo intervallo possono essere decurtate, consentendo, da una parte, la compressione della sequenza trasmessa, ed evitando, dall'altra, di pregiudicare la percezione del segnale sonoro, da parte dell'orecchio umano in ascolto.

1.1.2 Modellizzazione e codifica

Come detto in precedenza, la compressione dati si basa sulla identificazione e sullo sfruttamento delle particolarità della struttura della sequenza.

La fase di identificazione viene anche detta fase di modellizzazione, ed è responsabile della ricerca di eventuali peculiarità, relative alla struttura della sequenza da comprimere; la fase di sfruttamento, o fase di codifica, invece, si occupa di come rappresentare il carico informativo associato ad ogni dato della sequenza, onde ottenere un livello di compressione adeguato.

Nel caso della compressione effettuata dal codice Morse, ad esempio, la modellizzazione mette in luce la natura statistica della struttura della sequenza, ovvero, evidenzia la diversa frequenza con cui ciascuna lettera occorre; la codifica, invece, imposta la rappresentazione di ciascuna lettera, in maniera tale che, le lettere più frequenti abbiano una rappresentazione di lunghezza minore, mentre le lettere meno frequenti, una rappresentazione di lunghezza maggiore. Appare chiaro che, sia la fase di modellizzazione, che la fase di codifica, necessitano l'una dell'altra, e che perdono ogni efficacia, in termini di compressione, se separate; fatta questa precisazione, occorre, tuttavia, notare, che è la fase di codifica ad essere più propriamente responsabile della compressione sulla sequenza, andando a disegnare, la rappresentazione del carico informativo associato a ciascun dato della sequenza, su misura alla struttura emersa in fase di modellizzazione.

Approfondendo questi due aspetti per il caso della compressione effettuata dal codice Morse, il rapporto che intercorre tra, la struttura evidenziata dalla fase di modellizzazione, e la rappresentazione del carico informativo associato a ciascun dato della sequenza, è particolarmente evidente.
In prima analisi, occorre notare una sorta di identificazione tra, il carico informativo associato a ciascuna lettera, e la lunghezza della rappresentazione con cui tale lettera viene rappresentata in fase di codifica; assunto questo, si nota l'esistenza di una relazione di inversa proporzionalità tra, la probabilità di occorrenza di ciascuna lettera, evidenziata in fase di modellizzazione, e la lunghezza della rappresentazione associata a ciascuna lettera in fase di codifica.

In realtà, questa relazione di inversa proporzionalità, non è esclusiva della sola compressione effettuata dal codice Morse, qui presa ad esempio, bensì, è caratteristica di tutte quelle forme di compressione dati che sfruttano una struttura statistica della sequenza da comprimere; questo può essere mostrato formalizzando il concetto di carico informativo, associato ad un dato di una sequenza, attraverso la definizione di *self information*, sviluppata nell'ambito della teoria dell'informazione (C. E. Shannon e W. Weaver, 1949).
Data una sequenza S, in cui, ciascun dato generico s ha probabilità di occorrenza $p(s)$, si definisce *self information* $i(s)$ di ciascun dato s, la seguente:

$$i(s) = log_x \left(\frac{1}{p(s)} \right) \tag{1.1}$$

dove, a seconda che la base x del logaritmo sia uguale a 2, e o 10, l'unità di misura utilizzata per esprimere $i(s)$ è, rispettivamente, bit, nat, o hartley.

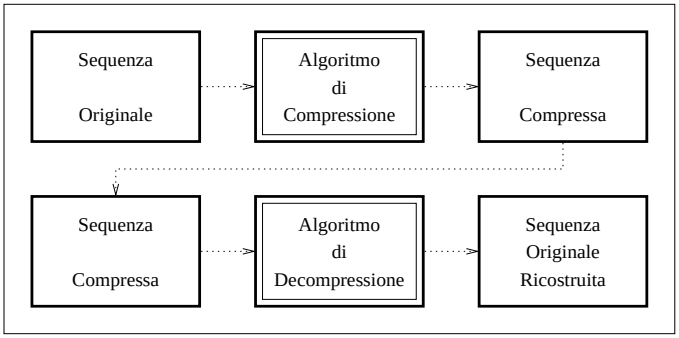


Figura 1.1: Compressione e decompressione di una sequenza.

In prima analisi, la 1.1 mostra l'identificazione, o quantomeno la stretta relazione, tra il contenuto informativo $i(s)$ di ciascun dato s, e la lunghezza, ad esempio in termini di bit, della rappresentazione con cui tale dato viene rappresentato in fase di codifica; in seconda analisi, mostra come, per tutte quella forme di compressione che si basano sulla struttra statistica dei dati—per le quali la fase di modellizzazione consiste nell'individuare la probabilità di occorrenza $p(s)$ di ciascun dato s—la fase di codifica consista nel calcolo, per ciascun dato s, della quantità, inversamente proporzionale a $p(s)$, $log_x\left(\frac{1}{p(s)}\right)$.

Riassumendo, quindi, la fase di modellizzazione e di codifica sono caratteristiche di ogni forma di compressione: la relazione che intercorre tra le due, può apparire più o meno sfumata, a seconda della struttura caratterizzante la sequenza; nel caso particolare di strutture statistiche, questa relazione si manifesta in una inversa proporzionalità tra, la probabilità di occorrenza di ogni singolo dato della sequenza, e la lunghezza con cui questo viene rappresentato in fase di codifica.

1.1.3 Compressione e decompressione

Lo sviluppo di algoritmi di compressione dati, va di pari passo con lo sviluppo di algoritmi di decompressione dati; in particolare, se un algoritmo di compressione trasforma una sequenza originale in ingresso, nella relativa sequenza compressa, l'algoritmo di decompressione corrispondente, è quello che ricostruisce la sequenza originale, a partire da quella compressa, come mostrato in Fig. 1.1.

Applicare l'algoritmo di compressione ad una sequenza e, in successione, applicare l'algoritmo di decompressione alla sequenza compressa, è una tecnica estremamente funzionale.
Infatti, questo consente da una parte di effettuare operazioni quali la trasmissione o la memorizzazione del carico informativo associato alla sequenza originale trasmettendo o memorizzando la sequenza compressa, equivalente dal punto di vista informativo, ma più "snella"; dall'altra, consente di ricostruire la sequenza originale a partire dalla sequenza compressa, memorizzata o trasmessa che sia.

Tali tecniche di compressione-decompressione si dividono, in realtà, in due sostanziali famiglie: le tecniche senza perdita di informazione (*lossless*) e quelle con perdita di informazione (*lossy*).
La prima famiglia è costituita da tutte le tecniche di compressione-decompressione, che consentono una ricostruzione della sequenza originale—a partire dalla sequenza compressa—esatta, viceversa, la seconda, è costituita da tutte le tecniche compressione-decompressione, che determinano una ricostruzione della sequenza, parzialmente diversa da quella originale.
Le tecniche di compressione-decompressione senza perdita di informazione, pur garantendo una ricostruzione esatta della sequenza originale, hanno, tuttavia, prestazioni limitate in termini di compressione; questa loro natura le rende particolarmente indicate in tutti quei casi in cui un errore in ricostruzione è da evitare accuratamente, quali la compressione di testi o la compressione di immagini radiografiche.
Le tecniche di compressione-decompressione con perdita di informazione, invece, possono raggiungere prestazioni estremamente elevate in termini di compressione, prestazioni che, tuttavia, determinano l'imperfetta ricostruzione della sequenza originale; per questa ragione sono particolarmente indicate in tutti quei casi in cui non è necessaria una fedeltà totale nella ricostruzione dei dati, quali la trasmissione di sequenze video o di sequenze audio a bassa qualità.

Le prestazioni complessive di una tecnica di compressione-decompressione *lossless*, possono essere valutate in termini di vari parametri: generalità e complessità dell'algoritmo di compressione e dell'algoritmo di decompressione, tempo impiegato per completare la compressione della sequenza originale e tempo impiegato per completare la sua successiva decompressione, quantità di memoria richiesta in fase di esecuzione dai due algoritmi e, più significativo tra gli altri, compressione effettuata.

Nello specifico, le prestazioni in termini di compressione vengono generalmente valutate in due maniere: si parla di *compression ratio* come del rapporto tra il numero totale di bit richiesti per rappresentare l'intera sequenza prima della compressione, ed il numero totale di bit richiesti per rappresentare l'intera sequenza compressa; si parla invece di *rate* come del numero medio di bit necessari per rappresentare un singolo dato della sequenza compressa.
Ad esempio, si supponga di applicare un generico algoritmo di compressione ad una sequenza costituita da 65536 dati, ed avente dimensioni pari a 65536 byte, e di ottenere, così, una sequenza compressa costituita sempre da 65536 dati, ma avente dimensioni pari a 16384 byte; in questo caso, il *compression ratio* è pari a $\frac{65536\times8}{16384\times8} = \frac{4}{1}$, mentre il *rate* è pari a $\frac{16384\times8}{65536} = 2$.

Per una valutazione complessiva delle prestazioni di una tecnica *lossy*, vale quanto detto per le tecniche *lossless*, tuttavia, in aggiunta, occorre quantificare la differenza tra la sequenza ricostruita da quella compressa, e la sequenza originale; tale differenza viene comunemente indicata come errore di ricostruzione o distorsione, e può essere calcolata attraverso una semplice differenza matematica o percentuale tra i dati della sequenza originale, ed i corrispondenti dati della sequenza ricostruita, come verrà approfondito nel Paragrafo 4.2.2.

1.2 Algoritmi di compressione

Tra gli algoritmi di compressione più rappresentativi ed interessanti, vi sono quelli basati, rispettivamente, sulla codifica *Run Length*, sulla codifica Delta, e sulla codifica Huffman.
Ciascuno di questi algoritmi sfrutta una struttura della sequenza da comprimere—messa in luce in fase di modellizzazione—particolare; nella fattispecie, l'algoritmo basato sulla codifica *Run Length* sfrutta la presenza, all'interno della sequenza da comprimere, di lunghe sottosequenze ininterrotte di uno stesso dato, l'algoritmo basato sulla codifica Delta sfrutta la somiglianza numerica tra i dati della sequenza, l'algoritmo basato sulla codifica Huffman, sfrutta, in analogia all'algoritmo basato sulla codifica Morse, la struttura statistica della sequenza.

Oltre ad essere particolarmente esemplificativi, gli algoritmi basati, rispettivamente, sulla codifica *Run Length*, Delta, e Huffman, sono alla base dell'algoritmo implementato nel sottomodulo CARLOS dell'esperimento ALICE; per questa ragione, l'attenzione verrà focalizzata su tali algoritmi.

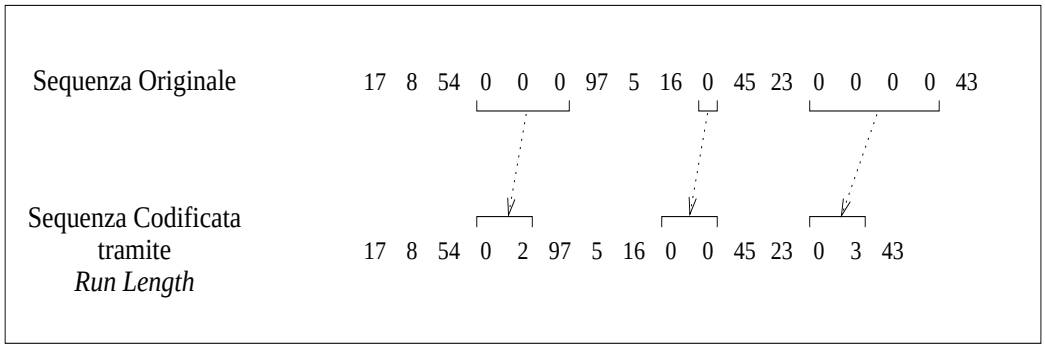


Figura 1.2: La compressione sviluppata dalla codifica *Run Length* dello 0.

1.2.1 Codifica *Run Length*

In molte circostanze, accade che una sequenza sia costituita da lunghe sottosequenze ininterrotte di uno stesso dato: questo accade nei segnali digitali, e sta ad indicare che la grandezza digitalizzata mantiene lo stesso valore su un certo intervallo, accade nei file di testo, dove un carattere può essere ripetuto più volte consecutivamente, e accade nelle immagini digitali indicizzate, dove, spazi di una stessa tonalità, sono digitalizzati tramite pixels aventi stesso valore; l'algoritmo di compressione, sviluppato tramite la codifica *Run Length* [2], è particolarmente indicato per una tale struttura della sequenza da comprimere.

Come si nota dall'esempio mostrato in Fig. 1.2—dove a titolo esemplificativo si è assunto lo zero, come dato ripetuto nelle sottosequenze—ogni sottosequenza di zeri nella sequenza originale, viene rappresentata, o come si usa dire, codificata, tramite una coppia di informazioni: la prima indica il dato trasmesso, in questo caso lo zero, la seconda indica quante volte tale dato dovrà essere ulteriormente ripetuto—da cui il nome *Run Length*—al fine di eguagliare il numero di occorrenze nella sequenza originale.

Le prestazioni di tale algoritmo migliorano, in termini di compressione, all'aumentare della lunghezza delle sottosequenze e, parallelamente, al diminuire di sottosequenze singole, quali la seconda codifica, 0→00, in Fig. 1.2.

Da ultimo, questo algoritmo di compressione può essere implementato in diverse varianti: su un solo elemento della sequenza originale di dati, come in questo caso esemplificativo lo zero, su più elementi della sequenza, in maniera tale che per ciascuno di questi dati venga implementata la codifica *Run Length*, od eventualmente su tutti i dati della sequenza.

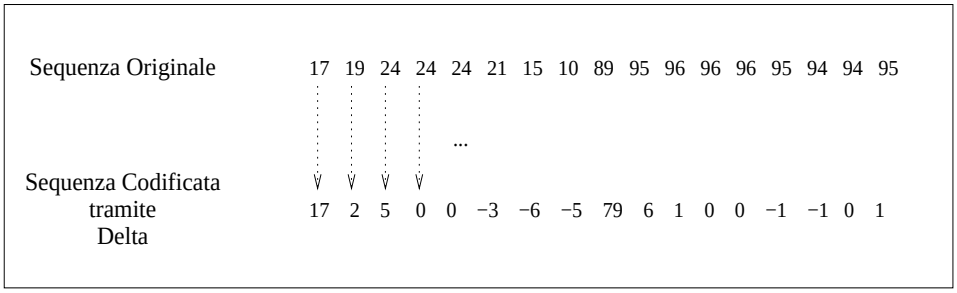


Figura 1.3: La compressione sviluppata dalla codifica Delta.

1.2.2 Codifica Delta

La codifica Delta [2] prende nome dal significato che, in ambito matematico, viene comunemente associato alla lettera dell'alfabeto greco Δ, ovvero, il significato di differenza; infatti, data una sequenza di dati da comprimere, l'algoritmo di compressione basato sulla codifica Delta, genera la sequenza codificata, effettuando la differenza fra ciascun dato della sequenza originale ed il suo precedente, escluso il primo, che viene lasciato inalterato nella sequenza codificata, come mostrato in Fig. 1.3.
Si noti che, ciascun dato della sequenza originale può essere ricostruito, sommando, al corrispondente dato della sequenza codificata, tutti i dati precedenti, ad esempio, $89 = 79+17+2+5+0+0+(-3)+(-6)+(-5)$; in questo senso, lasciare inalterato il primo dato nella sequenza codificata, è di fondamentale importanza, poiché da questo parte tutta la ricostruzione.

L'algoritmo di compressione così sviluppato, è particolarmente indicato per tutte quelle sequenze di dati che presentano lievi cambiamenti, in valore, tra dati adiacenti: per strutture di questo genere, infatti, la codifica Delta diminuisce in maniera decisa la dinamica della sequenza codificata, ovvero, rende la differenza tra, il dato di valore massimo della sequenza codificata, ed il suo dato di valore minimo, minore rispetto alla stessa differenza calcolata per la sequenza originale; in questa maniera, consente la rappresentazione della sequenza codificata, con un numero di bit inferiore rispetto a quella originale. Inoltre, applicare l'algoritmo di compressione basato sulla codifica Delta, a questo genere di sequenze, aumenta la probabilità che, nella sequenza codificata, compaiano lunghe sottosequenze ininterrotte di dati nulli; queste possono essere ulteriormente compresse, tramite il posizionamento dell'algoritmo basato sulla codifica *Run Length*, in cascata a quello Delta.

1.2.3 Codifica Huffman

Data una sequenza avente una struttura dei dati statistica, l'algoritmo di compressione basato sulla codifica Huffman [2] (D. A. Huffman, 1950), verte sulla rappresentazione di ogni dato della sequenza, con un numero di bit inversamente proporzionale alla probabilità della sua occorrenza; questo significa che, dati che hanno una probabilità di occorrenza bassa, sono rappresentati con un numero di bit elevato, mentre dati che hanno una probabilità di occorrenza alta, sono rappresentati con un numero di bit ridotto.

La scelta delle rappresentazioni da associare ad ogni dato, ovvero la costruzione della cosiddetta tavola di Huffman, avviene secondo criteri ben precisi. Si supponga di avere cinque dati, a_1, a_2, a_3, a_4 e a_5, caratterizzati da una probabilità di occorrenza, $P(a_1) = 0.2$, $P(a_2) = 0.4$, $P(a_3) = 0.2$, $P(a_4) = 0.1$, $P(a_5) = 0.1$; come primo passo, al fine di determinare la rappresentazione, o codifica, $c(a_i)$ di ciascun dato a_i, occorre riordinare tali dati secondo un ordine decrescente delle rispettive probabilità, come mostrato in Tab. 1.1.

Dato	Probabilità	Codifica
a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a_4	0.1	$c(a_4)$
a_5	0.1	$c(a_5)$

Tabella 1.1: I cinque dati iniziali.

I due dati aventi probabilità minore, sono a_4 e a_5; a questi, possono essere associate le codifiche:

$$c(a_4) = \alpha_1 * 0 \tag{1.2}$$
$$c(a_5) = \alpha_1 * 1 \tag{1.3}$$

dove α_i è una stringa binaria generica, e $*$ rappresenta la concatenazione tra due stringhe.
Sia a'_4 un dato per il quale vale che $P(a'_4) = P(a_4) + P(a_5) = 0.2$, allora, i dati in Tab. 1.1, possono essere riordinati secondo un ordine decrescente delle rispettive probabilità, alla luce di questo nuovo dato, come mostrato in Tab. 1.2.

Dato	Probabilità	Codifica
a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a'_4	0.2	α_1

Tabella 1.2: Introduzione del dato a'_4.

In questo caso, i due dati aventi probabilità minore, sono a_3 e a'_4; a questi, possono essere associate le codifiche:

$$c(a_3) = \alpha_2 * 0 \quad (1.4)$$
$$c(a'_4) = \alpha_2 * 1 \quad (1.5)$$

Tuttavia, essendo $c(a'_4) = \alpha_1$, si veda Tab. 1.2, allora dalla 1.5 si ha che $\alpha_1 = \alpha_2 * 1$, e quindi, la 1.2 e la 1.3 diventano:

$$c(a_4) = \alpha_2 * 10 \quad (1.6)$$
$$c(a_5) = \alpha_2 * 11 \quad (1.7)$$

Definendo a'_3 come il dato per il quale vale che $P(a'_3) = P(a_3) + P(a'_4) = 0.4$, si possono riordinare i dati in Tab. 1.2 secondo un ordine decrescente delle rispettive probabilità, alla luce di questo nuovo dato, si veda Tab. 1.3.

Dato	Probabilità	Codifica
a_2	0.4	$c(a_2)$
a'_3	0.4	α_2
a_1	0.2	$c(a_1)$

Tabella 1.3: Introduzione del dato a'_3.

I due dati aventi probabilità minore, in Tab. 1.3, sono a'_3 e a_1; a questi, possono essere associate le codifiche:

$$c(a'_3) = \alpha_3 * 0 \quad (1.8)$$
$$c(a1) = \alpha_3 * 1 \quad (1.9)$$

Essendo $c(a'_3) = \alpha_2$, si veda Tab. 1.3, allora dalla 1.8 si ha che $\alpha_2 = \alpha_3 * 0$, e quindi la 1.4, la 1.6 e la 1.7, diventano:

$$c(a_3) = \alpha_3 * 00 \tag{1.10}$$
$$c(a_4) = \alpha_3 * 010 \tag{1.11}$$
$$c(a_5) = \alpha_3 * 011 \tag{1.12}$$

Da ultimo, definendo a''_3 come il dato per il quale vale che $P(a''_3) = P(a'_3) + P(a_1) = 0.6$, si possono riordinare i dati in Tab. 1.3 secondo un ordine decrescente delle rispettive probabilità, alla luce di questo nuovo dato, si veda Tab. 1.4.

Dato	Probabilità	Codifica
a''_3	0.6	α_3
a_2	0.4	$c(a_2)$

Tabella 1.4: Introduzione del dato a''_3.

Essendo rimasti solo due dati, l'assegnamento della codifica è immediato:

$$c(a''_3) = 0 \tag{1.13}$$
$$c(a_2) = 1 \tag{1.14}$$

Inoltre, essendo $c(a''_3) = \alpha_3$, si veda Tab. 1.4, allora dalla 1.13 si ha che $\alpha_3 = 0$, ovvero, la 1.9, la 1.10, la 1.11 e la 1.12, diventano:

$$c(a_1) = 01 \tag{1.15}$$
$$c(a_3) = 000 \tag{1.16}$$
$$c(a_4) = 0010 \tag{1.17}$$
$$c(a_5) = 0011 \tag{1.18}$$

Si veda Tab. 1.5 per un riassunto della tavola Huffman generata.

Dato	Probabilità	Codifica
a_2	0.4	1
a_1	0.2	01
a_3	0.2	000
a_4	0.1	0010
a_5	0.1	0011

Tabella 1.5: Tavola Huffman.

La tecnica con cui è stata costruita la tavola—o codice—Huffman per questo caso specifico, ha comunque una validità del tutto generale, nel senso che può essere applicata, in maniera del tutto analoga, a qualsiasi sequenza avente una struttura dei dati statistica.

Le codifiche $c(a_i)$ di un codice Huffman, così generato, sono caratterizzate, in prima analisi, da una univoca decodificabilità (*unique decodability*); questo significa che, da una sequenza ininterrotta di codifiche $c(a_i)$, può essere ricostruita una, ed una sola, sequenza di dati a_i.
In secondo luogo, come si vede nell'esempio in Tab. 1.5, nessuna della codifiche $c(a_i)$ risulta prefisso per le altre codifiche; i codici che godono di questa proprietà vengono chiamati codici prefissi (*prefix codes*), ed in particolare, un codice che goda di questa proprietà, gode, automaticamente, dell'univoca decodificabilità, mentre non è sempre verificato il contrario.
Da ultimo, un codice Huffman viene definito un codice ottimale (*optimum code*), poiché, data una particolare struttura statistica della sequenza da comprimere, è tra tutti i codici prefissi, quello che minimizza la lunghezza media delle codifiche.

Capitolo 2

Wavelets: teoria ed implementazioni

In questo capitolo verrà trattato l'aspetto teorico delle wavelets e la sua naturale estensione a problemi di carattere numerico.
La difficoltà maggiore che si incontra nell'approfondire questo argomento, risiede nel recente sviluppo di tale tecnica di analisi e, congiuntamente, nella sua forte diffusione sperimentale in tempi antecedenti alla sua formalizzazione teorica.
Inoltre, i campi di studio in cui questa disciplina si è andata sviluppando indipendentemente, sono tra i più vari, dalla analisi matematica pura, alla analisi dei segnali digitali, passando per la compressione di immagini e per la trattazione di modelli della visione umana; tutto ciò ha contribuito a darne diverse formulazioni empiriche, tanto varie e sfaccettate, quanto apparentemente inconciliabili, in un primo momento, con una teoria unificatrice.
L'approccio che, a mio parere, si rivela più efficace nella trattazione della teoria wavelet, è quello, per così dire, storico: nel senso che, seguendo l'evoluzione dell'analisi wavelet nel tempo, si comprendono ed apprezzano gli sforzi mirati ad una unificazione teorica dei risultati empirici precedenti; questo approccio porta a trattare, in primo luogo, la trasformata wavelet continua e la sua estensione al caso discreto, ed in secondo luogo, l'analisi multirisoluzione, così importante per le applicazioni numeriche.

2.1 Cenni storici

Lo strumento che, per eccellenza, viene posto alla base dell'analisi dei segnali, è l'analisi di Fourier; questa tecnica, sviluppata da J. Fourier nel primo ventennio dell'800, consente di studiare un segnale lineare nel dominio della frequenza, diverso, dunque, dal suo dominio naturale, il tempo [3].
Tuttavia, la trasformata di Fourier, oltre a trattare efficientemente solo problemi lineari, nasconde l'informazione relativa al tempo, così come, un segnale nel dominio del tempo, nasconde l'informazione in frequenza che porta con sé.

Gli sforzi mirati a superare tali limiti, hanno portato D. Gabor, nel 1946, alla definizione della *Short Time Fourier Transform*; questa tecnica consente, attraverso la moltiplicazione del segnale con una funzione "finestra" dal supporto compatto, di considerarlo lineare a tratti—permettendo così anche l'analisi di segnali non lineari—e di ottenere una rappresentazione del segnale, contemporaneamente, in tempo e frequenza.
Anche questo tipo di analisi, tuttavia, soffre di un limite: una volta scelta la "finestra", si è fissata, per tutto il corso dell'analisi, sia la risoluzione in tempo che quella in frequenza, precludendosi, così, ogni possibilità di modificarle in corsa; inoltre, a causa del Principio di Indeterminazione, una migliore risoluzione temporale diminuisce quella in frequenza, e viceversa.

L'analisi wavelet nasce in risposta ai limiti caratterizzanti i due tipi di analisi fin qui discussi.
Dagli inizi del 1900 fino al 1970 si hanno diversi approcci a questo argomento: da una parte, la comunità matematica cerca di superare i limiti analitici della *Short Time Fourier Transform*, iniziando a concepire la trasformata wavelet continua; da un punto di vista sperimentale, invece, in diversi laboratori si riproducono algoritmi per l'analisi dei segnali o delle immagini—attraverso, ad esempio, filtri in cascata—che si riveleranno essere strettamente connessi con l'analisi wavelet.
Questi approcci, più o meno consapevoli, all'analisi wavelet, partono nel 1909 con il lavoro del matematico tedesco A. Haar, che scopre l'omonimo sistema di basi ortonormali, e proseguono con numerosi e diversi contributi, a partire da quello del fisico K. Wilson (1971) e dei ricercatori francesi, nel campo dei segnali digitali, A. Croisier, D. Esteban, C. Galand (1976), proseguendo con quello di D. Marr (1980) nella trattazione del sistema di visione umana, fino al 1975, data che può essere presa come indicativa della nascita dell'analisi wavelet; J. Morlet, a differenza di quanto fatto nella *Short Time Fourier Trans-*

form—dove viene tenuta fissa l'ampiezza della funzione "finestra" di Gabor, colmandola con oscillazioni di diversa frequenza—blocca il numero di oscillazioni nella funzione, cambiandone invece l'ampiezza attraverso compressioni o allungamenti.
A portare avanti il lavoro di Morlet sono Y. Meyer, fisico di Marsiglia e A. Grossman, professore della École Polytechnique; ad essi si associa S. Mallat a partire dal 1986, dando il contributo, a mio avviso, più importante nella teoria dell'analisi wavelet: Mallat formalizza la teoria wavelet, delineando, in particolar modo, la naturale estensione di questa analisi al caso discreto.
In questo ambito enuncia la "Teoria della analisi multirisoluzione", capace di riunire la teoria della trasformata discreta wavelet, a tutti quegli algoritmi sperimentali—come il *Pyramid Algorithm* nella analisi di immagini o la tecnica dei *Quadrature Mirror Filters* nella analisi dei segnali—fino ad allora utilizzati con successo, ma mai formalizzati nel contesto dell'analisi wavelet.
Partendo dal lavoro di Mallat, I. Daubechies, intorno al 1987, costruisce l'insieme omonimo di basi di wavelet ortonormali, divenuto pietra miliare per la maggior parte delle applicazioni wavelet.

L'algoritmo wavelet è considerato, ad oggi, di grande interesse.
Le sue molteplici caratteristiche lo rendono estremamente duttile e funzionale in diverse discipline: la capacità che ha, ad esempio, di ben approssimare segnali non lineari, lo rende uno strumento determinante negli studi dell'Institut du Globe in Paris, sull'influenza che la corrente oceanica El Niño, ha sulla velocità rotazionale terrestre.
La tolleranza che la trasformata wavelet dimostra nei confronti di eventuali errori nei suoi coefficienti, la rende strumento adatto al matematico D. Healy, del Dartmouth Math Department, per lo studio di risonanze magnetiche e al Rochester Medical Center per elettrocardiogrammi ad alta risoluzione.
In campo astronomico, la trasformata wavelet viene apprezzata per la sua capacità di identificare strutture a risoluzioni diverse: questo consente di studiare con profitto le distribuzioni a larga-scala nell'Universo.
Anche nelle tecniche di compressione e di riduzione del "rumore" in un segnale, le wavelets trovano largo impiego: infatti, la loro tolleranza a coefficienti errati, consente l'azzeramento di parte dei coefficienti wavelet—quelli minori, in modulo, di una certa soglia—consentendo così la rappresentazione dello stesso segnale con un numero inferiore di informazioni; allo stesso tempo, questo ne azzera la componente ad alta frequenza determinata dal "rumore".

2.2 Dall'analisi di Fourier alla trasformata wavelet

L'analisi di Fourier e la *Short Time Fourier Transform*, limitano la loro efficacia a problemi aventi una natura ben definita; si segua dunque il cammino che ha portato l'analisi dei segnali, a considerare l'algoritmo wavelet uno strumento alternativo [4].

2.2.1 L'analisi di Fourier

Dato un segnale $f(t) \in L^2(\mathbb{R})$ nel dominio del tempo, la Trasformata di Fourier (FT) ne dà una rappresentazione nel dominio della frequenza ν, si veda Fig. 2.1.

Figura 2.1: La trasformata di Fourier.

L'espressione matematica che identifica la trasformata di Fourier è la seguente:

$$FT_f(\nu) \equiv \hat{f}(\nu) = \int_{-\infty}^{+\infty} f(t)e^{-j2\pi\nu t}dt \tag{2.1}$$

Dalla (2.1) si nota come la $\hat{f}(\nu)$ perda, dal punto di vista analitico, la dipendenza funzionale dal tempo t.
Sostanzialmente, cioè, un segnale $f(t)$ fornisce una informazione con risoluzione infinita—ovvero indeterminazione nulla—sull'asse dei tempi, occultando, però,

l'informazione relativa alla frequenza, sul cui asse ha risoluzione nulla; la trasformata di Fourier $\hat{f}(\nu)$ di un segnale $f(t)$, invece, fornisce un'informazione con risoluzione infinita sull'asse della frequenza, ma con risoluzione nulla sull'asse dei tempi.

Il seguente esempio chiarirà come in certi casi, invece, possa essere necessaria una duplice, e contemporanea, rappresentazione del segnale, sia nel dominio del tempo, che in quello della frequenza.
Si supponga di avere due segnali: sia $f_1(t) = cos(2\pi\nu_1 t) + cos(2\pi\nu_2 t) + cos(2\pi\nu_3 t) + cos(2\pi\nu_4 t)$ e sia $f_2(t)$ tale che, dall'istante $t = 0$ all'istante $t = t_1$ abbia frequenza ν_1, dall'istante $t = t_1$ a quello $t = t_2$ abbia frequenza ν_2, e così via, si veda la Fig. 2.2.

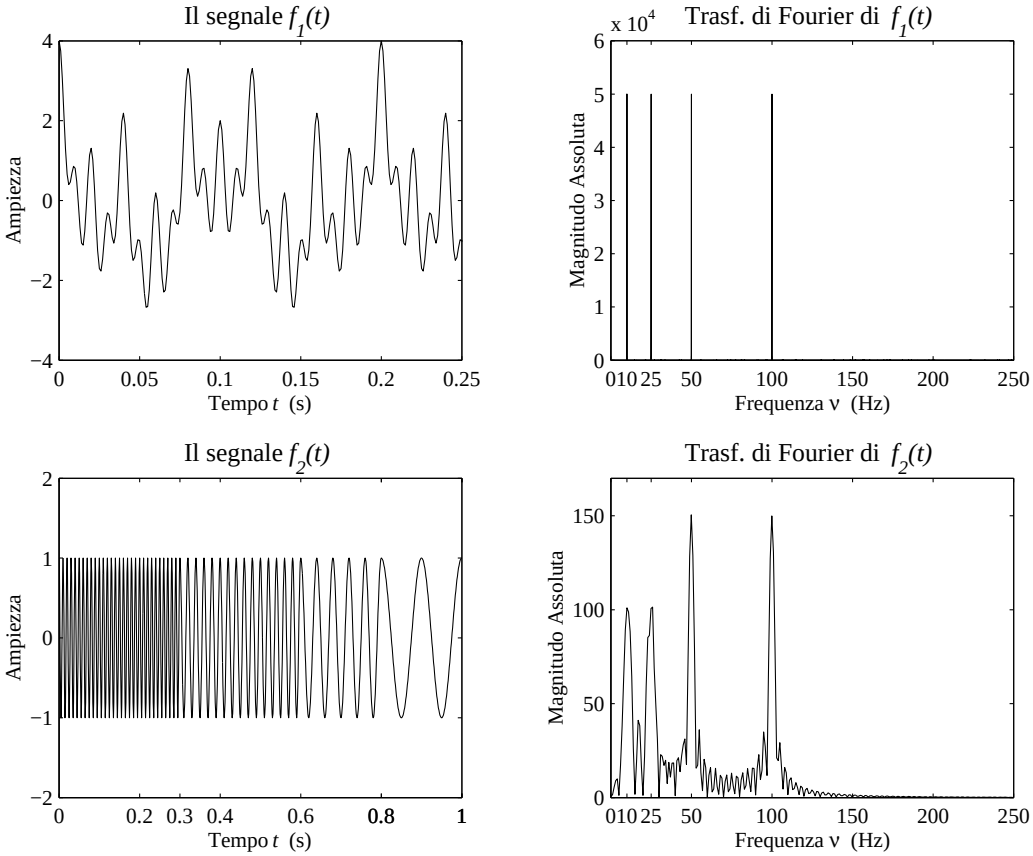


Figura 2.2: I segnali $f_1(t)$ e $f_2(t)$ con $\nu_1 = 10Hz$, $\nu_2 = 25Hz$, $\nu_3 = 50Hz$, $\nu_4 = 100Hz$, e le rispettive trasformate di Fourier.

Si dice che $f_1(t)$ è un segnale stazionario, nel senso che, le sue quattro componenti in frequenza, esistono su tutti i tempi; viceversa, $f_2(t)$ è non stazionario, poiché, pur avendo le stesse quattro componenti in frequenza di $f_1(t)$, le sue esistono in intervalli di tempo ben precisi.
In sintesi, avendo i due segnali le stesse componenti in frequenza, e poiché la FT nasconde l'informazione legata al tempo, lo spettro di questi due—pur così diversi—segnali, risulta pressoché medesimo.
Il paradosso rappresentato da questo esempio è facilmente comprensibile, osservando che, l'integrale in (2.1), è calcolato per ogni valore reale di t; in altri termini, la FT fornisce il contenuto in frequenza del segnale $f(t)$ su tutti i tempi, da $t = -\infty$ a $t = +\infty$, ignorando la precisa locazione temporale di ciascuna frequenza.

Questo è il motivo che rende l'analisi di Fourier inadatta alla rappresentazione di segnali non stazionari.

2.2.2 La *Short Time Fourier Transform*

Un modo per superare le difficoltà che l'analisi di Fourier trova nel rappresentare segnali non stazionari, consiste nell'analizzare porzioni successive di segnale, approssimabili a stazionarie: questa è l'idea base della *Short Time Fourier Transform*, od anche detta STFT.

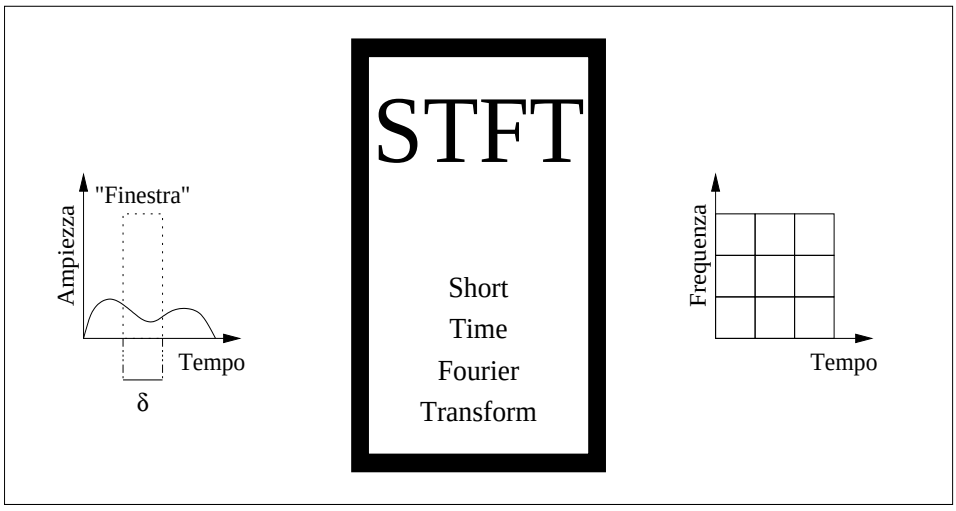


Figura 2.3: La *Short Time Fourier Transform*.

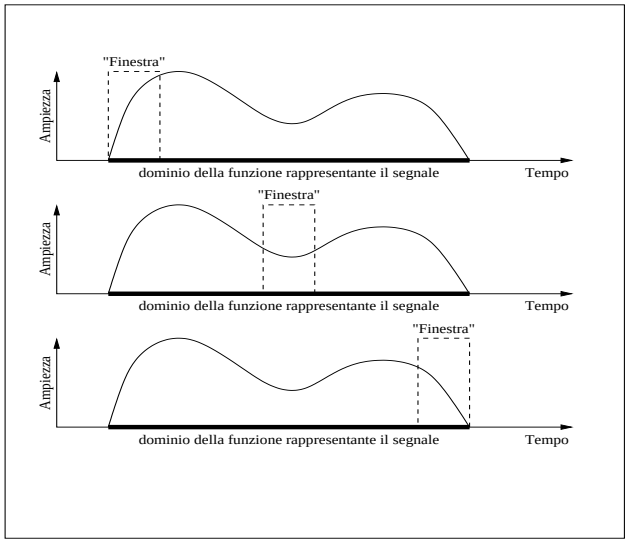


Figura 2.4: L'algoritmo di analisi tramite STFT.

La STFT prevede l'utilizzo di una funzione "finestra" $g(t)$, avente supporto δ compatto, si veda Fig. 2.3:

$$STFT_f(\tau, \nu) = \int_{-\infty}^{+\infty} f(t) g^*(t-\tau) e^{-j2\pi\nu t} dt \tag{2.2}$$

dove $*$ rappresenta l'operazione di complesso coniugato, e τ rappresenta il parametro di traslazione della funzione "finestra" sull'asse dei tempi.
Moltiplicare il segnale $f(t)$ con $g^*(t-\tau)$ equivale, di fatto, a considerare una porzione limitata di $f(t)$, nella fattispecie, la porzione che ha, come dominio, l'intervallo di ampiezza δ, traslato di τ sull'asse dei tempi; in questo modo, scegliendo accuratamente la $g(t)$, si può ottenere, come prodotto delle due funzioni, una funzione stazionaria.
Operare la trasformata di Fourier di tale prodotto di funzioni, poi, equivale a dare una rappresentazione di $f(t)$ in frequenza, nell'intervallo di tempo individuato dal supporto della funzione "finestra".

L'analisi tramite STFT prevede una traslazione progressiva della funzione "finestra" su tutto il dominio della $f(t)$, consentendo un'analisi di tutto il segnale; graficamente, l'algoritmo alla base di questa analisi, può essere schematizzato come in Fig. 2.4.

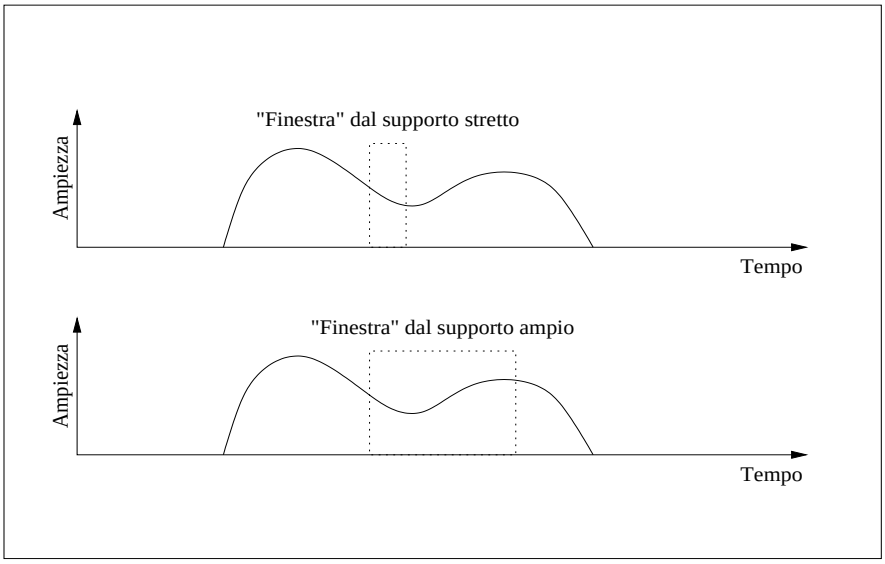


Figura 2.5: Funzioni "finestra" con supporto differente.

Il limite di tale analisi, come si spiega oltre, è legato al suo essere fortemente condizionato dal Principio di Indeterminazione, il quale si manifesta sotto forma di una inversa proporzionalità tra l'incertezza Δt sull'asse dei tempi e quella $\Delta\nu$ sull'asse delle frequenze.
L'analisi tramite STFT è una tecnica a risoluzione fissa, infatti, la scelta della funzione $g(t)$, e quindi del suo supporto compatto, fissa, per tutta la durata dell'analisi, la distanza minima Δt, che deve esistere tra due punti sull'asse dei tempi, affinché siano discriminabili: in altre parole, fissa l'incertezza con cui si indaga sull'asse dei tempi, si veda Fig. 2.5.
A causa del Principio di Indeterminazione, il fissare l'incertezza sull'asse dei tempi, implica il fissare l'incertezza $\Delta\nu$ sull'asse delle frequenze per tutta la durata dell'analisi; poiché il rapporto tra tali incertezze è di inversa proporzionalità, durante il corso dell'analisi non si può avere, contemporaneamente, una buona localizzazione sull'asse dei tempi e sull'asse delle frequenze.
Questo è visualizzabile attraverso le cosiddette "scatole" di Heisenberg, rappresentate in Fig. 2.6: si noti che al diminuire dell'incertezza lungo un asse, corrisponde l'aumento dell'incertezza lungo l'altro; questo rende costante l'area sottostante ad ogni quadrato, manifestazione grafica del Principio di Indeterminazione.

In sintesi, il limite da affrontare nell'analisi tramite STFT, è dato dal fatto che, una volta scelta la $g(t)$ e fatta partire l'analisi, sono fissate, in maniera irreversibile, l'incertezza Δt sull'asse dei tempi e, a causa del Principio di Indeterminazione, l'incertezza $\Delta\nu$ sull'asse delle frequenze; questo risulta limitativo per tutti quei casi in cui potrebbe essere utile un'analisi a diverse risoluzioni.

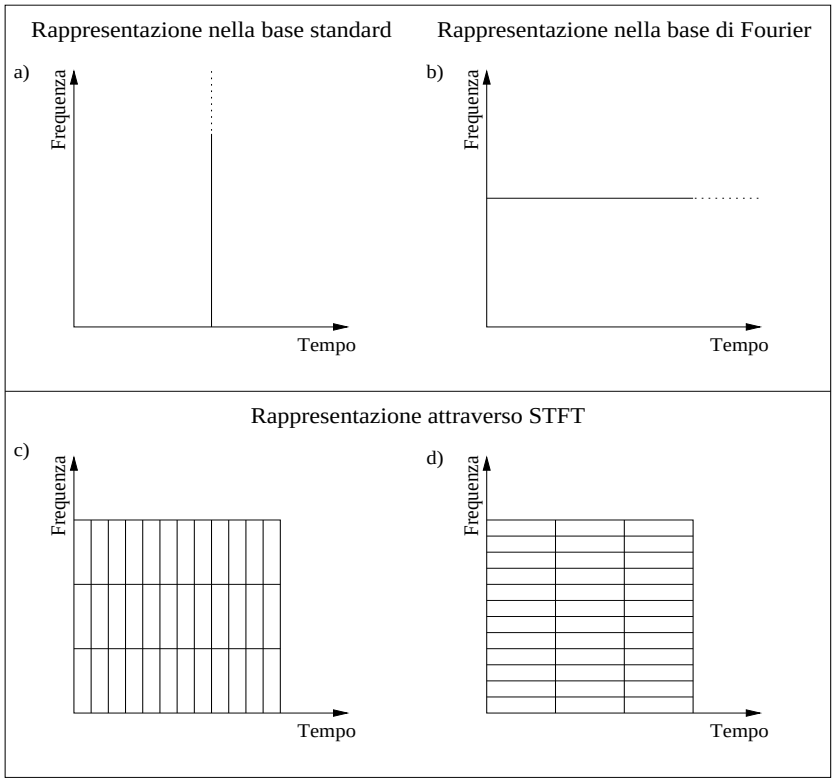


Figura 2.6: Le "scatole" di Heisenberg: a) la rappresentazione classica di un segnale in funzione del tempo garantisce una incertezza nulla sull'asse dei tempi ed una incertezza infinita sull'asse delle frequenze; b) la rappresentazione di un segnale attraverso l'analisi di Fourier garantisce una incertezza nulla sull'asse delle frequenze ed una incertezza infinita sull'asse dei tempi; c) e d) la STFT consente una rappresentazione del segnale sia in tempo che in frequenza tale che a basse incertezze sull'asse dei tempi corrispondano alte incertezze sull'asse delle frequenze, e viceversa.

2.2.3 La Trasformata Wavelet Continua

L'analisi wavelet, a differenza della STFT, opera con una funzione "finestra", avente l'ampiezza del supporto, variabile nel corso dell'analisi; inoltre, non viene calcolata la trasformata di Fourier del prodotto tra il segnale e la funzione, come fatto, invece, nel caso della STFT.
La Trasformata Wavelet Continua (CWT), è così definita:

$$CWT_f(a,b) \equiv C(a,b) = \int_{-\infty}^{+\infty} f(t)\psi_b^a(t)dt \tag{2.3}$$

$$\text{con:} \quad \psi_b^a(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \quad \text{wavelet madre} \tag{2.4}$$

dove $a \in \mathbb{R}^+ - \{0\}$ è il parametro di scala e $b \in \mathbb{R}$ il parametro di traslazione; in particolare, la $\psi_b^a(t)$ deriva il suo nome "wavelet madre", dal fatto che è una funzione oscillatoria supportata compattamente ("wavelet") e dalla quale, al variare di a e b, sono derivate le funzioni—o wavelets—con diverse regioni di supporto ("madre").
Per il parametro di scala a, valgono le seguenti considerazioni:

- valori del parametro di scala alti equivalgono ad allungare la wavelet, e quindi il suo supporto, lungo l'asse dei tempi, ed a diminuire, conseguentemente, la frequenza della wavelet stessa.

- valori del parametro di scala bassi equivalgono a comprimere la wavelet, e quindi il suo supporto, lungo l'asse dei tempi, e ad aumentare, conseguentemente, la frequenza della wavelet stessa.

L'analisi tramite CWT opera per successive traslazioni e scalamenti della wavelet: nella fattispecie, per ogni valore del parametro di scala $a \in \mathbb{R}^+ - \{0\}$, il parametro di traslazione b viene fatto variare su tutto $\mathbb{R}$, consentendo, così, un'analisi dell'intero segnale a risoluzioni diverse.
In particolare, come mostrato in Fig. 2.7, l'analisi tramite CWT utilizza in maniera specifica:

- un parametro di scala alto—corrispondente ad una frequenza bassa della wavelet—ove si desiderino informazioni con un margine di precisione più elevato sull'asse delle frequenze rispetto al margine di precisione sull'asse dei tempi.

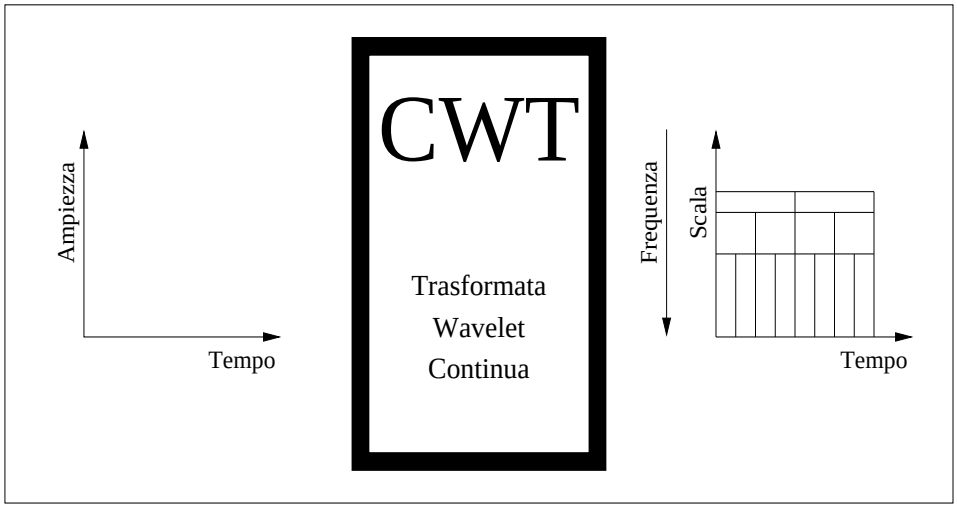


Figura 2.7: Analisi a risoluzioni variabili tramite CWT.

- un parametro di scala basso—corrispondente ad una frequenza alta della wavelet—ove si desiderino informazioni con un margine di precisione più elevato sull'asse dei tempi rispetto al margine di precisione sull'asse delle frequenze.

Questo è di notevole importanza, poiché, in natura, i segnali contengono informazioni approssimative—cioè individuanti la struttura generale del segnale—a basse frequenze, ed informazioni di dettaglio—cioè individuanti le minuzie del segnale—ad alte frequenze; con l'analisi wavelet, quindi, si dà una rappresentazione del segnale avente informazioni precise sulle basse frequenze, che costituiscono la struttura portante del segnale, ed informazioni meno precise su quella parte del segnale che è di contorno, ovvero la parte ad alta frequenza.

L'utilità di una trasformata è chiaramente nella sua invertibilità, ed in questo la CWT non manca; se l'analisi del segnale consiste nell'operare la trasformata wavelet sul segnale originale $f(t)$, ottenendo i coefficienti wavelet $C(a,b)$, allora la sintesi consiste nel ricostruire il segnale a partire da tali coefficienti; per segnali di energia finita, ovvero $f \in L^2(\mathbb{R})$, vale la seguente:

$$f(t) = \frac{1}{K_\psi} \int_{\mathbb{R}^+} \int_{\mathbb{R}} C(a,b) \frac{1}{\sqrt{a}} \psi \left(\frac{t-b}{a} \right) \frac{dadb}{a^2} \tag{2.5}$$

dove K_ψ è una costante che dipende dalla wavelet ψ scelta.

2.3 Dall'analisi wavelet continua a quella discreta

La definizione di CWT data in (2.3), formalizza in modo esaustivo l'algoritmo wavelet, tuttavia, ha una utilità limitata: l'unico utilizzo della CWT, così definita, è nella valutazione analitica della trasformata wavelet continua di un segnale; infatti, un approccio numerico alla trasformata wavelet continua, come definita in (2.3), appare impossibile per valori del parametro a spazianti su tutto $\mathbb{R}^+ - \{0\}$ e valori del parametro b spazianti su tutto $\mathbb{R}$.

Partendo dal presupposto che, nella realtà fisica, si lavora con segnali discreti, un metodo che consente di implementare numericamente l'analisi tramite CWT su un segnale discreto [5], deve, in prima battuta effettuare l'analisi assegnando ai parametri a e b un insieme finito di valori, si veda Fig. 2.8: in particolare, a può operare su un insieme finito di valori di scala, a partire dalla risoluzione con cui è stato discretizzato il segnale, fino ad una scala massima, determinata dalle esigenze dell'analisi—si veda Fig. 2.9—mentre b può essere incrementato, un numero finito di volte, in maniera tale da traslare la wavelet su tutto il dominio del segnale; in seconda battuta, per ogni coppia di valori a e b, tale metodo deve approssimare numericamente l'integrale del prodotto tra il segnale e la wavelet $\psi_b^a(t)$.
In questa maniera, l'implementazione numerica dell'analisi wavelet continua su un segnale discreto, produce un numero di coefficienti pari al prodotto tra il numero di valori differenti assunti da a ed il numero di valori differenti assunti da b.

Tuttavia, calcolare i coefficienti wavelet su un insieme di valori di scala e traslazione potenzialmente molto ampio, sebbene finito, risulta un discreto sovraccarico per il sistema computazionale, e genera una vasta quantità di valori; nella fattispecie, si dice che la CWT dà una rappresentazione del segnale completa—ovvero che consente la ricostruzione del segnale originale—ma ridondante, nel senso che, il numero di campioni prodotti nell'analisi—cioè il numero di coefficienti—è maggiore del numero dei campioni del segnale analizzato.

Mallat, nel 1986, formalizza un algoritmo per l'analisi wavelet, che consente una analisi più efficiente ma, allo stesso tempo, ugualmente accurata; egli, infatti, suggerisce una particolare discretizzazione degli insiemi finiti di valori assunti dai parametri a e b, tale da determinare una rappresentazione del segnale completa ma non ridondante.

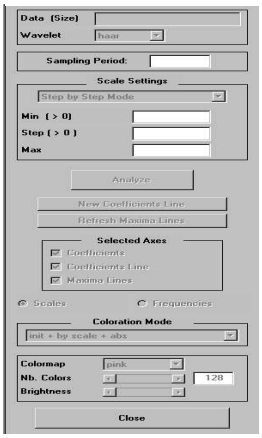


Figura 2.8: Finestra di dialogo del Wavelet Toolbox di Matlab in cui si possono impostare gli intervalli di lavoro per i parametri a e b: Scale Settings consente di impostare il valore (Min) di scala minimo, il valore (Max) di scala massimo ed il valore (Step) con cui viene incrementato il parametro di scala; Sampling Period consente di impostare il valore con cui viene incrementato il parametro di traslazione su tutto il dominio del segnale da analizzare.

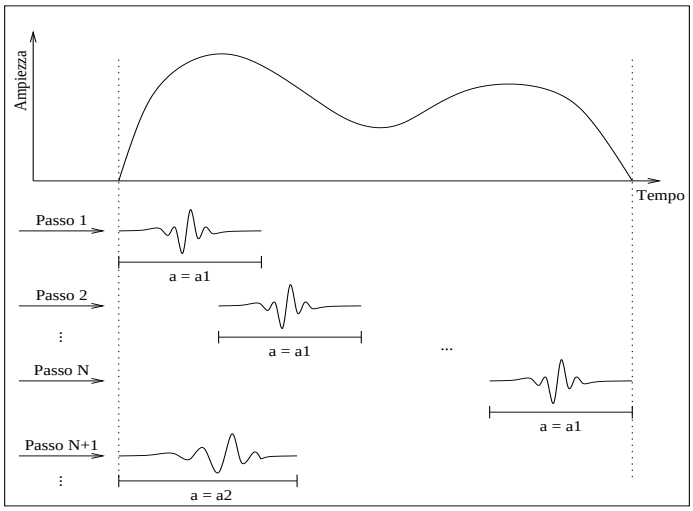


Figura 2.9: Una possibile implementazione numerica dell'algoritmo di analisi tramite CWT: a e b assumono un insieme finito di valori.

Alla base della scelta di tale discretizzazione, vi sono le seguenti considerazioni riguardanti l'analisi wavelet [6]:

- al variare del parametro di scala a, l'incertezza relativa $\Delta\nu/\nu$ è costante, si veda Fig. 2.7, pertanto si può pensare ad una discretizzazione logaritmica di tale parametro.

- quando il parametro a è piccolo, le wavelets hanno supporto temporale stretto, pertanto il parametro di traslazione b deve essere piccolo, onde garantire una copertura sufficiente dell'asse-tempi, da parte della serie di wavelets.

- viceversa, quando a è grande, le wavelets hanno supporto temporale ampio, pertanto il parametro di traslazione b può essere maggiore.

Da quanto detto si intuisce che la discretizzazione più efficace è:

$$a = a_0^{-j} \qquad \text{con} \qquad j \in \mathbb{Z}, a_0 > 1 \tag{2.6}$$

$$b = na \qquad \text{con} \qquad n \in \mathbb{Z} \tag{2.7}$$

Il caso diadico, $a_0 = 2$, è quello proposto da Mallat e definisce, per $j \in \mathbb{Z}$ e $n \in \mathbb{Z}$, la famiglia di wavelets:

$$\psi_n^j(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)\Bigg|_{a=2^{-j},b=2^{-j}n} = 2^{j/2}\psi(2^j t - n) \tag{2.8}$$

Chiaramente, i valori interi assunti da j e n nel corso dell'analisi sono finiti: in particolare, definendo a_{min} e a_{max}, rispettivamente, il valore di scala minimo ed il valore di scala massimo con cui effettuare l'analisi, allora j assume tutti i valori interi compresi tra j_{max} e j_{min} dove $a_{min} = 2^{-j_{max}}$ e $a_{max} = 2^{-j_{min}}$; inoltre, per ogni $a = 2^{-j}$ fissato, n assume tutti i valori interi tali da traslare la wavelet su tutto il dominio del segnale da analizzare.
L'analisi wavelet, come definita in (2.3), ma sviluppata tramite la wavelet madre definita in (2.8), viene comunemente indicata con il nome di analisi tramite trasformata wavelet discreta o analisi tramite DWT, in virtù della discretizzazione diadica dei parametri a e b.

2.4 Analisi multirisoluzione

Come detto nel Paragrafo 2.2.3, l'analisi wavelet consente una rappresentazione del segnale più precisa alle basse frequenze e meno alle alte, in virtù del suo essere un'analisi a scala variabile.

L'idea alla base dell'analisi multirisoluzione [7] è quella di separare, utilizzando l'analisi tramite DWT, il contenuto a bassa frequenza di un segnale, rappresentante la struttura approssimativa del segnale, da quello ad alta frequenza, rappresentante le struttura di dettaglio del segnale.

Per ottenere tale separazione, Mallat suggerisce di implementare l'algoritmo wavelet tramite DWT, utilizzando due famiglie di funzioni—si veda, ad esempio, Fig. 2.10 e Fig. 2.11—leggermente diverse: la famiglia di funzioni $(\phi_n^j(t))_{(n,j)\in\mathbb{Z}^2}$, detta di scala, definita [8] ad arte al fine di estrarre il contenuto di approssimazione del segnale, e la famiglia di funzioni wavelet, discussa in (2.8), $(\psi_n^j(t))_{(n,j)\in\mathbb{Z}^2}$, derivata dalla precedente, ed estraente il contenuto di dettaglio del segnale.

In particolare, implementare l'analisi tramite DWT con la famiglia di funzioni $(\psi_n^j(t))_{(n,j)\in\mathbb{Z}^2}$ equivale a calcolare—al variare di j ed n sull'insieme finito dei loro valori—i coefficienti di dettaglio:

$$\int_{-\infty}^{+\infty} f(t)2^{j/2}\psi(2^j t - n)dt \tag{2.9}$$

mentre implementarla con la famiglia di funzioni $(\phi_n^j(t))_{(n,j)\in\mathbb{Z}^2}$, equivale a calcolare—al variare di j ed n sull'insieme finito dei loro valori—i coefficienti di approssimazione:

$$\int_{-\infty}^{+\infty} f(t)2^{j/2}\phi(2^j t - n)dt \tag{2.10}$$

All'aumentare della risoluzione verso $+\infty$, cioè al diminuire della scala, l'insieme dei coefficienti di approssimazione, converge progressivamente al segnale originale, mentre l'insieme dei coefficienti di dettaglio dà una rappresentazione sempre più minuziosa dei dettagli; al diminuire della risoluzione verso 0, invece, cioè all'aumentare della scala, sia i coefficienti di approssimazione, che quelli di dettaglio, contengono sempre meno carico informativo, convergendo a zero.

Un approfondimento degli aspetti matematici dell'analisi wavelet multirisoluzione, così come trattati da Mallat, è di grande interesse; le dimostrazioni dei teoremi enunciati qui di seguito, possono essere trovate in [7].

2.4.1 L'operatore di approssimazione

Poiché il metodo multirisolutivo di Mallat si basa su di una progressione diadica dell'analisi, è lecito chiedersi quale sia la differenza di informazione tra due approssimazioni successive, una alla risoluzione 2^j ed una alla risoluzione 2^{j+1}, con $j \in \mathbb{Z}$; a tal fine, occorre formalizzare matematicamente l'operazione di approssimazione di un segnale alla risoluzione 2^j, introducendo l'operatore di approssimazione A_{2^j}.

L'azione dell'operatore A_{2^j} su un segnale $f(x) \in L^2(\mathbb{R})$, dove alla variabile tempo t è stata sostituita una più generale x, può essere così definita:

$$A_{2^j} f(x) = \sum_{n=-\infty}^{+\infty} \left\langle f(u), 2^{j/2}\phi(2^j u - n) \right\rangle 2^{j/2}\phi(2^j x - n) \tag{2.11}$$

La (2.11) mostra come, fissata la risoluzione a 2^j, l'operatore A_{2^j} approssimi il segnale $f(x)$ attraverso la somma delle traslazioni $2^{j/2}\phi(2^j x - n)$ della funzione di scala—si noti la sommatoria su n—pesate opportunamente dai coefficienti $\left\langle f(u), 2^{j/2}\phi(2^j u - n) \right\rangle$; essendo $\left\langle f(u), 2^{j/2}\phi(2^j u - n) \right\rangle = \int_{-\infty}^{+\infty} f(u) 2^{j/2}\phi(2^j u - n)du$, i coefficienti con cui viene pesata ciascuna traslazione $2^{j/2}\phi(2^j x - n)$, non sono niente altro che i coefficienti prodotti dall'analisi wavelet tramite DWT utilizzando, con j fissato, la famiglia di funzioni di scala $(\phi_n^j(x))_{(n,j)\in\mathbb{Z}^2}$ al posto della famiglia di funzioni wavelet $(\psi_n^j(x))_{(n,j)\in\mathbb{Z}^2}$, come visto in (2.10).
Ad esempio, fissata la risoluzione a 2^j, e scelta la funzione di scala in cui ciascuna delle traslazioni $2^{j/2}\phi(2^j x - n)$ vale $2^{j/2}$ nell'intervallo $]n2^{-j}, (n+1)2^{-j}[$ con $n \in \mathbb{Z}$, e 0 altrove, la (2.11) equivale ad approssimare il segnale $f(x)$ con una funzione costante a tratti, sugli intervalli $]n2^{-j}, (n+1)2^{-j}[$, con $n \in \mathbb{Z}$.

In particolare, scegliendo la funzione di scala in maniera tale che le sue traslazioni siano tra loro ortonormali, come nel caso appena citato, si può verificare che l'operatore di approssimazione A_{2^j} è caratterizzato da alcune proprietà:

- l'approssimazione $A_{2^j} f(x)$ non è modificata, se ulteriormente approssimata alla risoluzione 2^j.
- $A_{2^j} f(x)$ è, tra tutte le funzioni approssimate alla risoluzione 2^j, la più simile ad $f(x)$.
- l'approssimazione alla risoluzione 2^{j+1} contiene tutte le informazioni necessarie al calcolo dell'approssimazione alla risoluzione 2^j.

- le approssimazioni a varie risoluzioni sono derivabili, l'una dall'altra, scalando il segnale approssimato, di un fattore pari al rapporto tra le diverse risoluzioni.
- l'approssimazione $A_{2^j} f(x)$ può essere caratterizzata da 2^j campioni—i coefficienti in (2.11)—per unità di lunghezza; traslando $f(x)$, $A_{2^j} f(x)$ è parimenti traslata e può essere caratterizzata dagli stessi campioni traslati.
- aumentando la risoluzione, l'approssimazione deve convergere al segnale, diminuendola, invece, deve convergere a zero.

Ciascuna di queste proprietà intuitive, viene corrispondentemente formalizzata nella seguente definizione di operatore di approssimazione:

Def. 2.4.1 *Sia $f(x) \in L^2(\mathbb{R})$ e A_{2^j} un operatore lineare tale che:*

i) A_{2^j} *è un operatore di proiezione su di un particolare spazio vettoriale* $V_{2^j} \subset L^2(\mathbb{R})$*, interpretabile come lo spazio vettoriale costituito da tutte le possibili approssimazioni alla risoluzione* 2^j *delle funzioni in* $L^2(\mathbb{R})$*.*

ii) $\forall g(x) \in V_{2^j}$*,* $\big\|g(x)-f(x)\big\| \geq \big\|A_{2^j} f(x)-f(x)\big\|$*, cioè* A_{2^j} *è una proiezione ortogonale su* V_{2^j}*.*

iii) $\forall j \in \mathbb{Z}$ *si ha che* $V_{2^j} \subset V_{2^{j+1}}$*.*

iv) $\forall j \in \mathbb{Z}$ *si ha che* $f(x) \in V_{2^j} \Leftrightarrow f(2x) \in V_{2^{j+1}}$*.*

v) posto $I^2(\mathbb{Z}) = \big\{(\alpha_i)_{i\in\mathbb{Z}} : \sum_{-\infty}^{+\infty} |\alpha_i|^2 < \infty\big\}$*:*

- *esiste un isomorfismo* I *da* V_1 *a* $I^2(\mathbb{Z})$*.*
- $\forall k \in \mathbb{Z}$*,* $A_1 f_k(x) = A_1 f(x-k)$*, dove* $f_k(x) = f(x-k)$*.*
- $I(A_1 f(x)) = (\alpha_i)_{i\in\mathbb{Z}} \Leftrightarrow I(A_1 f_k(x)) = (\alpha_{i-k})_{i\in\mathbb{Z}}$*.*

vi) $\lim_{j\to+\infty} V_{2^j} = \bigcup_{-\infty}^{+\infty} V_{2^j}$ *è denso in* $L^2(\mathbb{R})$*.*
$\lim_{j\to-\infty} V_{2^j} = \bigcap_{-\infty}^{+\infty} V_{2^j} = \{0\}$*.*

Si dice che una sequenza di spazi vettoriali $(V_{2^j})_{j\in\mathbb{Z}}$*, come definita in i), che soddisfa i punti dal iii) al vi), è una approssimazione multirisoluzione di* $L^2(\mathbb{R})$*; si dice inoltre che l'operatore* A_{2^j}*, come definito in i), che soddisfa i punti dal ii) al v), approssima ciascuna funzione di* $L^2(\mathbb{R})$ *ad una risoluzione* 2^j*.*

2.4.2 La funzione di scala ϕ

La definizione teorica data in 2.4.1 equivale alla definizione operativa data in (2.11) sotto l'ipotesi che le traslazioni $2^{j/2}\phi(2^j x - n)$ della funzione di scala siano tra loro ortonormali; il seguente teorema mostra la validità di tale ipotesi, ammettendo l'esistenza e l'unicità di una base ortonormale dello spazio vettoriale V_{2^j}.

Teo. 2.4.2 *Sia $(V_{2^j})_{j\in\mathbb{Z}}$ una approssimazione multirisoluzione di $L^2(\mathbb{R})$. Allora esiste un'unica funzione $\phi(x) \in L^2(\mathbb{R})$, chiamata funzione di scala, tale che, posto $\phi_{2^j}(x) = 2^j\phi(2^j x)$, per $j \in \mathbb{Z}$, si ha che: $(2^{-j/2}\phi_{2^j}(x - 2^{-j}n))_{n\in\mathbb{Z}}$, è una base ortonormale di V_{2^j}.*

Questo teorema, in sostanza, mostra che è possibile costruire una base ortonormale di ciascun V_{2^j}, scalando la funzione di scala con un coefficiente di scala a pari a 2^{-j}, e traslando la funzione risultante su di una griglia il cui intervallo è proporzionale a 2^{-j}.

Decomponendo $A_{2^j}f(x)$ nella base ortonormale di V_{2^j}, ora che la sua esistenza ed unicità è assicurata dal Teorema 2.4.2, si ottiene nuovamente la (2.11):

$\forall f(x) \in L^2(\mathbb{R})$

$$A_{2^j}f(x) = 2^{-j}\sum_{n=-\infty}^{+\infty}\left\langle f(u), \phi_{2^j}(u - 2^{-j}n)\right\rangle \phi_{2^j}(x - 2^{-j}n) \tag{2.12}$$

Si definisca $A^d_{2^j}f$, come la "approssimazione discreta" di $f(x)$ alla risoluzione 2^j.

$$\begin{aligned} A^d_{2^j}f &\equiv \left(\left\langle f(u), \phi_{2^j}(u - 2^{-j}n)\right\rangle\right)_{n\in\mathbb{Z}} = && (2.13)\\ &= \left(\int_{-\infty}^{+\infty} f(u)\phi_{2^j}(u - 2^{-j}n)du\right)_{n\in\mathbb{Z}} = && (2.14)\\ &= \left((f(u) * \phi_{2^j}(-u))(2^{-j}n)\right)_{n\in\mathbb{Z}} && (2.15)\end{aligned}$$

Si noti come, in virtù della (2.13) e della (2.14), il calcolo della approssimazione discreta $A^d_{2^j}f$ di un segnale, equivalga al calcolo dei coefficienti dell'analisi wavelet tramite DWT, utilizzando, con j fissato, la famiglia di funzioni di scala $(\phi^j_n(x))_{(n,j)\in\mathbb{Z}^2}$.

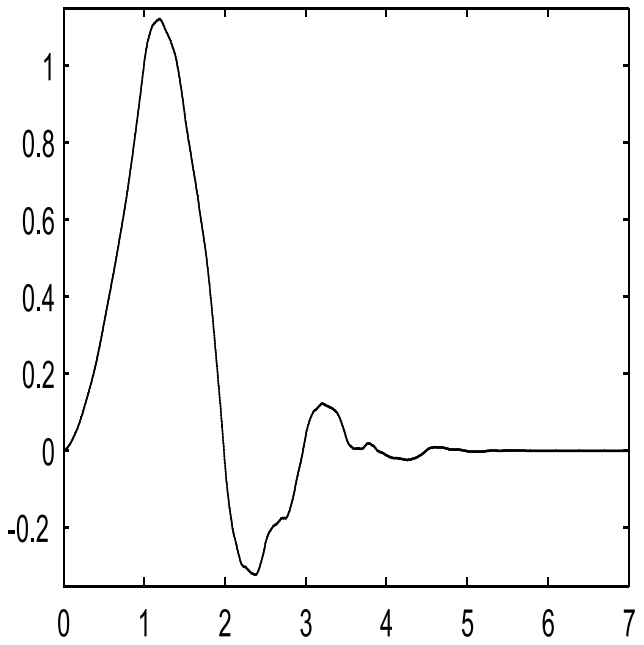


Figura 2.10: La natura passa basso della funzione $\phi(x)$; in questo esempio la funzione di scala daub4 della famiglia Daubachies.

Contemporaneamente, si noti come, in virtù della (2.15), il calcolo della approssimazione discreta equivalga ad una operazione di convoluzione tra il segnale $f(x)$ e la funzione $\phi_{2^j}(x)$ e ad una successiva operazione di campionamento ogni 2^{-j} campioni, detta operazione di sottocampionamento.
Da ultimo, essendo le operazioni di convoluzione riconducibili ad operazioni di filtro, si veda a questo proposito (A.2), la (2.15) può essere vista come un filtrare il segnale $f(x)$ attraverso il filtro passa basso $\phi_{2^j}(x)$, si veda Fig. 2.10, ed un successivo sottocampionamento ogni 2^{-j} campioni.

Date queste premesse, ovvero la Definizione 2.4.1 ed il Teorema 2.4.2, è immediato trovare un algoritmo semplice che consenta di calcolare iterativamente le approssimazioni discrete $A^d_{2^j} f$ del segnale, al variare di j.
Considerato che, nella realtà, ogni segnale può essere misurato solo con una risoluzione finita, si definisca, per comodità di normalizzazione, $A^d_1 f$ come l'approssimazione avente risoluzione pari a quella del segnale originale; ogni approssimazione $A^d_{2^j}$, con $j < 0$, tenderà a dare una rappresentazione sempre più grossolana del segnale.

Sia $V_{2^{j+1}}$ lo spazio vettoriale delle approssimazioni alla risoluzione 2^{j+1}; una sua base è data da $(\sqrt{2^{-j-1}}\phi_{2^{j+1}}(x-2^{-j-1}k))_{k\in\mathbb{Z}}$.
In virtù del fatto che $V_{2^j} \subset V_{2^{j+1}}$, è possibile rappresentare la funzione di V_{2^j}, $\phi_{2^j}(x-2^{-j}n)$, tramite la base ortonormale di $V_{2^{j+1}}$:

$$\phi_{2^j}(x-2^{-j}n) = 2^{-j-1}\sum_{k=-\infty}^{+\infty}\left\langle \phi_{2^j}(u-2^{-j}n), \phi_{2^{j+1}}(u-2^{-j-1}k)\right\rangle \cdot \cdot\phi_{2^{j+1}}(x-2^{-j-1}k) = \tag{2.16}$$

$$= \sum_{k=-\infty}^{+\infty}\langle \phi_{2^{-1}}(u), \phi(u-(k-2n))\rangle \cdot \cdot\phi_{2^{j+1}}(x-2^{-j-1}k) \tag{2.17}$$

Utilizzando la (2.17) nel calcolo di ciascuno dei coefficienti della approssimazione discreta, si veda (2.13), si ha che:

$$\left\langle f(u), \phi_{2^j}(u-2^{-j}n)\right\rangle = \sum_{k=-\infty}^{+\infty}\langle \phi_{2^{-1}}(u), \phi(u-(k-2n))\rangle \cdot \cdot\left\langle f(u), \phi_{2^{j+1}}(u-2^{-j-1}k)\right\rangle \tag{2.18}$$

Sia H un filtro discreto la cui risposta all'impulso—si veda Appendice A per la definizione di risposta all'impulso—è data da:

$$\forall n \in \mathbb{Z} \quad h(n) = \langle \phi_{2^{-1}}(u), \phi(u-n)\rangle \tag{2.19}$$

e sia $\widetilde{H}$ il filtro specchio di H, avente risposta all'impulso $\tilde{h}(n) = h(-n)$. Esprimendo (2.18) in funzione di (2.19), si ottiene:

$$\left\langle f(u), \phi_{2^j}(u-2^{-j}n)\right\rangle = \sum_{k=-\infty}^{+\infty}\tilde{h}(2n-k)\left\langle f(u), \phi_{2^{j+1}}(u-2^{-j-1}k)\right\rangle \tag{2.20}$$

La (2.20), in virtù della sua somiglianza alla (A.1), mostra come ogni approssimazione discreta $A^d_{2^j}$, alla risoluzione 2^j, possa essere calcolata a partire dalla approssimazione discreta precedente, meno grossolana, $A^d_{2^{j+1}}$, filtrata attraverso il filtro $\widetilde{H}$, e sottocampionando il segnale risultante ogni 2 campioni.

2.4.3 La funzione wavelet ψ

Si definisca lo spazio vettoriale O_{2^j}, complemento ortogonale di V_{2^j} in $V_{2^{j+1}}$, ovvero $O_{2^j} \oplus V_{2^j} = V_{2^{j+1}}$.
Le approssimazioni di un segnale alle risoluzioni 2^j e 2^{j+1}, sono le sue proiezioni ortogonali, rispettivamente, su V_{2^j} e $V_{2^{j+1}}$, pertanto, la differenza d'informazione tra due approssimazioni successive, ovvero il segnale di dettaglio, risiede nel complemento ortogonale di V_{2^j} in $V_{2^{j+1}}$, ovvero O_{2^j}.
Può essere dimostrato [7] che un base ortonormale di O_{2^j} è costituita dalla famiglia di wavelet $(2^{-j/2}\psi_{2^j}(x - 2^{-j}n))_{n\in\mathbb{Z}}$, dove $\psi_{2^j}(x) = 2^j\psi(2^j x)$ e dove la scelta della famiglia di wavelets dipende dalla famiglia di funzioni di scala $(2^{-j/2}\phi_{2^j}(x-2^{-j}n))_{n\in\mathbb{Z}}$ con cui si sta lavorando; tale dipendenza verrà palesata oltre.

Con un procedimento analogo a quello usato in (2.13), per il calcolo delle approssimazioni di un segnale, si mette in luce che anche i segnali di dettaglio sono caratterizzati dal calcolo di prodotti interni, chiamati "dettagli discreti", che contengono la differenza di informazione tra $A^d_{2^{j+1}}f$ e $A^d_{2^j}f$:

$$D_{2^j}f = ((f(u) * \psi_{2^j}(-u))(2^{-j}n))_{n\in\mathbb{Z}} \tag{2.21}$$

Si noti come la (2.21) equivalga al calcolo dei coefficienti (2.9) prodotti dall'analisi wavelet tramite DWT, utilizzando, con j fissato, la famiglia di funzioni wavelet $(\psi^j_n(x))_{(n,j)\in\mathbb{Z}^2}$.
Come per le approssimazioni discrete, la (2.21) mostra come il calcolo dei dettagli discreti si riduca al filtrare il segnale $f(x)$ attraverso il filtro passa alto $\psi_{2^j}(x)$—si veda Fig. 2.11—e ad un successivo sottocampionamento ogni 2^{-j} campioni.

L'altro punto che, in analogia alle approssimazioni discrete, si dimostra essere verificato anche per i dettagli discreti, è l'esistenza di un algoritmo ricorsivo per il calcolo di questi: infatti, dato un filtro discreto G avente risposta all'impulso:

$$\forall n \in \mathbb{Z} \quad g(n) = \langle \psi_{2^{-1}}(u), \phi(u - n)\rangle \tag{2.22}$$

e dato il suo filtro specchio $\widetilde{G}$, tale che $\tilde{g}(n) = g(-n)$, vale la seguente:

$$\left\langle f(u), \psi_{2^j}(u - 2^{-j}n)\right\rangle = \sum_{k=-\infty}^{+\infty} \tilde{g}(2n - k)\left\langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}k)\right\rangle \tag{2.23}$$

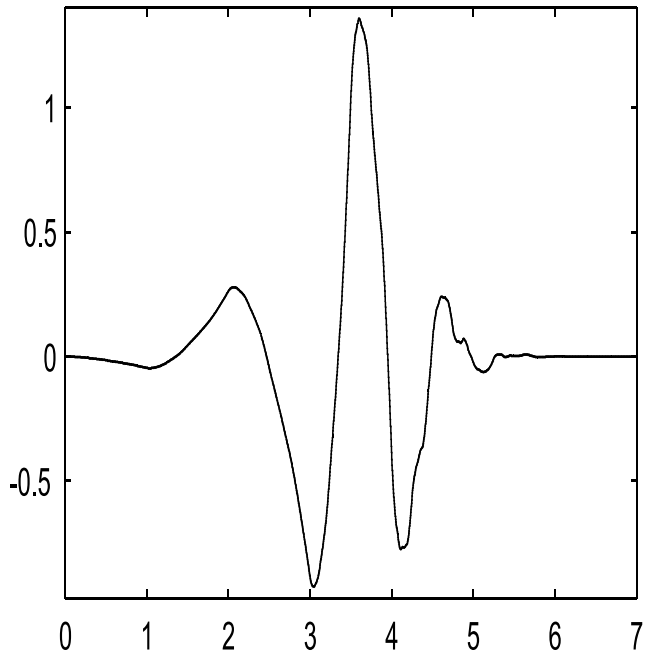


Figura 2.11: La natura passa alto della funzione $\psi(x)$; in questo esempio la funzione wavelet daub4, della famiglia Daubechies.

La (2.23) dimostra che il dettaglio discreto D_{2^j}, alla risoluzione 2^j, è ottenibile a partire dalla approssimazione discreta $A^d_{2^{j+1}} f$, filtrata attraverso il filtro $\widetilde{G}$, e sottocampionando il segnale risultante ogni 2 campioni.

La dipendenza, cui si accennava prima, tra la famiglia di funzioni di scala $(2^{-j/2}\phi_{2^j}(x - 2^{-j}n))_{n\in\mathbb{Z}}$ e la famiglia di wavelets $(2^{-j/2}\psi_{2^j}(x - 2^{-j}n))_{n\in\mathbb{Z}}$, si manifesta sotto forma di una relazione tra i filtri H e G, alla base della analisi mutirisoluzione di un segnale.
In particolare, tale dipendenza può essere espressa come una relazione tra le ripettive risposte all'impulso $g(n)$ e $h(n)$, come definite in (2.19) e (2.22):

$$\forall n \in \mathbb{Z} \quad g(n) = (-1)^{(1-n)} h(1-n) \tag{2.24}$$

La (2.24), inoltre, manifesta analiticamente ciò che, nell'ambito dell'analisi dei segnali, corrisponde a costruire un sistema di filtri particolare, detto sistema di *Quadrature Mirror Filters*.

2.4.4 Analisi e sintesi

La (2.20) e la (2.23) mostrano come sia possibile calcolare la approssimazione discreta $A^d_{2^j} f$ ed il dettaglio discreto $D_{2^j} f$, ad una risoluzione 2^j, a partire dalla approssimazione discreta $A^d_{2^{j+1}} f$ alla risoluzione 2^{j+1}; dal punto di vista operativo, tale algoritmo può essere implementato filtrando la approssimazione discreta $A^d_{2^{j+1}} f$ attraverso il filtro $\widetilde{H}$, definito in (2.19), onde ottenere, a seguito di un sottocampionamento ogni 2 campioni, la approssimazione discreta $A^d_{2^j} f$, e attraverso il filtro $\widetilde{G}$, definito in (2.22), ottenendo, a seguito di un sottocampionamento ogni 2 campioni, il dettaglio discreto $D_{2^j} f$.

Effettuando $J > 0$ livelli di approssimazioni successive, a partire dalla approssimazione discreta A^d_1, si ottiene una rappresentazione alternativa del segnale stesso, detta analisi—o decomposizione—wavelet ortogonale:

$$(A^d_{2^{-J}} f, (D_{2^j} f)_{-J \leq j \leq -1}) \tag{2.25}$$

Si noti che, se il segnale è costituito da N campioni, i segnali discreti $A^d_{2^j} f$ e $D_{2^j} f$ sono costituiti da $2^j N$ campioni ciascuno, cioè la decomposizione wavelet ortogonale in (2.25) è costituita dallo stesso numero di campioni N del segnale originale, ovvero è completa; questo è dovuto alla ortogonalità della trasformata.

In analogia al caso della decomposizione, esiste un algoritmo per la ricostruzione—o sintesi—del segnale originale, a partire dalla sua decomposizione wavelet ortogonale (2.25).
Infatti, essendo $O_{2^j} \oplus V_{2^j} = V_{2^{j+1}}$, $(2^{-j/2}\phi_{2^j}(x - 2^{-j}n), 2^{-j/2}\psi_{2^j}(x - 2^{-j}n))_{n \in \mathbb{Z}}$ è una base di $V_{2^{j+1}}$; operando in maniera analoga a quanto fatto in (2.16) e (2.17), si decompone $\phi_{2^{j+1}}(x - 2^{-j-1}n)$ nelle componenti della base precedentemente menzionata, ed usando i filtri H e G, definiti in (2.19) e (2.22), si ottiene che:

$$\begin{aligned} &\left\langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}n) \right\rangle = \\ &2 \sum_{k=-\infty}^{+\infty} h(n-2k) \left\langle f(u), \phi_{2^j}(u - 2^{-j}k) \right\rangle + \\ + \quad &2 \sum_{k=-\infty}^{+\infty} g(n-2k) \left\langle f(u), \psi_{2^j}(u - 2^{-j}k) \right\rangle \end{aligned} \tag{2.26}$$

La (2.26) equivale ad affermare che $A^d_{2^{j+1}}$ può essere ricostruita a partire da $A^d_{2^j} f$ e $D_{2^j} f$, inserendo uno zero tra ciascun campione e filtrando il risultato di tale sovracampionamento, rispettivamente, attraverso i filtri H e G.

Sia l'analisi del segnale, formalizzata dalla (2.20) e dalla (2.23), che la sintesi, formalizzata dalla (2.26), si basano su una struttura piramidale: in particolare, questa struttura ricorda molto da vicino il *Pyramid Algorithm*, sviluppato da P. J. Burt nell'ambito dell'analisi delle immagini.

2.4.5 Generalizzazione al caso bi-dimensionale

Sia $f(x,y) \in L^2(\mathbb{R}^2)$ un segnale bi-dimensionale, quale potrebbe essere una immagine: si definisce approssimazione multirisoluzione $(V_{2^j})_{j\in\mathbb{Z}}$ di $L^2(\mathbb{R}^2)$, una sequenza di sottospazi di $L^2(\mathbb{R}^2)$ che soddisfano le condizioni $iii) - vi)$ della Definizione 2.4.1, generalizzate al caso bi-dimensionale.

Il Teorema 2.4.2 continua a valere anche per il caso bi-dimensionale, cosicché, la famiglia di funzioni $(2^{-j}\Phi_{2^j}(x-2^{-j}n, y-2^{-j}m))_{(n,m)\in\mathbb{Z}^2}$, con $\Phi_{2^j}(x,y) = 2^{2^j}\Phi(2^j x, 2^j y)$, è una base ortonormale di V_{2^j}.
Si dice inoltre che V_{2^j} è separabile se $V_{2^j} = V^1_{2^j} \otimes V^1_{2^j}$, ovvero se è decomponibile nel prodotto tensoriale di due sottospazi identici di $L^2(\mathbb{R})$.
Se questa condizione è verificata, vale che $\Phi(x,y) = \phi(x)\phi(y)$, con $\phi(x)$ funzione di scala mono-dimensionale di $(V^1_{2^j})_{j\in\mathbb{Z}}$, ovvero una base ortonormale di V_{2^j} è data da:

$$(2^{-j}\phi_{2^j}(x - 2^{-j}n)\phi_{2^j}(y - 2^{-j}m))_{(n,m)\in\mathbb{Z}^2} \tag{2.27}$$

Il seguente teorema prova che si può costruire una base di O_{2^j} scalando e traslando tre funzioni wavelets, $\Psi^1(x,y)$, $\Psi^2(x,y)$, $\Psi^3(x,y)$.

Teo. 2.4.3 *Sia $(V_{2^j})_{j\in\mathbb{Z}}$ una approssimazione multirisoluzione separabile di $L^2(\mathbb{R}^2)$, sia $\Phi(x,y) = \phi(x)\phi(y)$ la funzione di scala associata, e sia $\psi(x)$ la wavelet mono-dimensionale associata alla funzione di scala $\phi(x)$.*
Allora esistono tre wavelets:

$$\Psi^1(x,y) = \phi(x)\psi(y) \tag{2.28}$$
$$\Psi^2(x,y) = \psi(x)\phi(y) \tag{2.29}$$
$$\Psi^3(x,y) = \psi(x)\psi(y) \tag{2.30}$$

tali che, una base ortonormale di O_{2^j} è data da:

$$\begin{aligned}&(2^{-j}\Psi^1_{2^j}(x - 2^{-j}n, y - 2^{-j}m),\\ &2^{-j}\Psi^2_{2^j}(x - 2^{-j}n, y - 2^{-j}m),\\ &2^{-j}\Psi^3_{2^j}(x - 2^{-j}n, y - 2^{-j}m))_{(n,m)\in\mathbb{Z}^2}\end{aligned} \tag{2.31}$$

Come per il caso mono-dimensionale, la differenza tra $A^d_{2^j} f$ e $A^d_{2^{j+1}} f$, è data dalla proiezione ortogonale del segnale su O_{2^j}, ed è caratterizzata dai prodotti interni del segnale con la base ortonormale di O_{2^j}.

$$D^1_{2^j} f = (\langle f(x,y), \Psi^1_{2^j}(x - 2^{-j}n, y - 2^{-j}m)\rangle)_{(n,m)\in\mathbb{Z}^2} \tag{2.32}$$
$$D^2_{2^j} f = (\langle f(x,y), \Psi^2_{2^j}(x - 2^{-j}n, y - 2^{-j}m)\rangle)_{(n,m)\in\mathbb{Z}^2} \tag{2.33}$$
$$D^3_{2^j} f = (\langle f(x,y), \Psi^3_{2^j}(x - 2^{-j}n, y - 2^{-j}m)\rangle)_{(n,m)\in\mathbb{Z}^2} \tag{2.34}$$

In particolare, $D^1_{2^j} f$ rappresenta il dettaglio discreto verticale alla risoluzione 2^j, dato che è calcolato attraverso il prodotto interno del segnale $f(x,y)$ e la wavelet $\Psi^1(x,y) = \phi(x)\psi(y)$; similmente, $D^2_{2^j} f$ rappresenta il dettaglio discreto orizzontale e $D^3_{2^j} f$ rappresenta il dettaglio discreto diagonale.

In analogia a quanto fatto in (2.15) ed in (2.21) per il caso mono-dimensionale, si può dimostrare che il calcolo delle approssimazioni discrete e dei dettagli discreti è riconducibile ad operazioni di filtro e sottocampionamento:

$$A^d_{2^j} f = ((f(x,y) * \phi_{2^j}(-x)\phi_{2^j}(-y))(2^{-j}n, 2^{-j}m))_{(n,m)\in\mathbb{Z}^2} \tag{2.35}$$
$$D^1_{2^j} f = ((f(x,y) * \phi_{2^j}(-x)\psi_{2^j}(-y))(2^{-j}n, 2^{-j}m))_{(n,m)\in\mathbb{Z}^2} \tag{2.36}$$
$$D^2_{2^j} f = ((f(x,y) * \psi_{2^j}(-x)\phi_{2^j}(-y))(2^{-j}n, 2^{-j}m))_{(n,m)\in\mathbb{Z}^2} \tag{2.37}$$
$$D^3_{2^j} f = ((f(x,y) * \psi_{2^j}(-x)\psi_{2^j}(-y))(2^{-j}n, 2^{-j}m))_{(n,m)\in\mathbb{Z}^2} \tag{2.38}$$

Effettuando $J > 0$ approssimazioni successive su di una approssimazione discreta bi-dimensionale $A^d_1 f$, questo è completamente rappresentato da $3J + 1$ segnali bi-dimensionali—o immagini—detti, una "rappresentazione wavelet ortogonale in due dimensioni":

$$(A^d_{2^{-J}} f, (D^1_{2^j})_{-J\le j\le -1}, (D^2_{2^j})_{-J\le j\le -1}, (D^3_{2^j})_{-J\le j\le -1}) \tag{2.39}$$

L'immagine $A^d_{2^{-J}} f$ è la approssimazione discreta più grossolana alla risoluzione 2^{-J} e le immagini $D^k_{2^j}$, con $-J \le j \le -1$, sono i dettagli discreti per diverse orientazioni e risoluzioni, si veda la Fig. 2.12.

Data l'ortogonalità della decomposizione, se l'immagine originale ha N campioni, ogni sotto immagine a risoluzione 2^j, avrà $2^j N$ campioni; pertanto la decomposizione è completa.

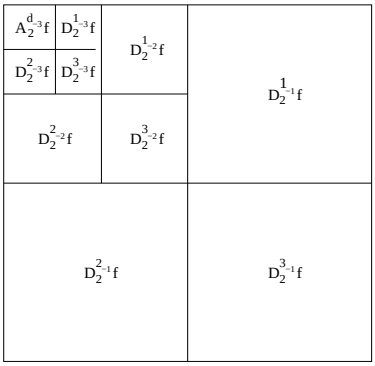


Figura 2.12: Decomposizione wavelet di una immagine bi-dimensionale.

2.5 L'analisi multirisoluzione come implementazione di filtri

La (2.20) e la (2.23) mostrano come sia possibile calcolare l'approssimazione discreta $A^d_{2^j} f$ ed il dettaglio discreto $D_{2^j} f$, ad una risoluzione 2^j, filtrando l'approssimazione discreta $A^d_{2^{j+1}} f$ attraverso, rispettivamente, il filtro passa basso $\widetilde{H}$ ed il filtro passa alto $\widetilde{G}$ e facendo seguire, a queste operazioni di filtro, un sottocampionamento dei segnali risultanti, ogni 2 campioni; si veda a tal proposito Fig. 2.13.

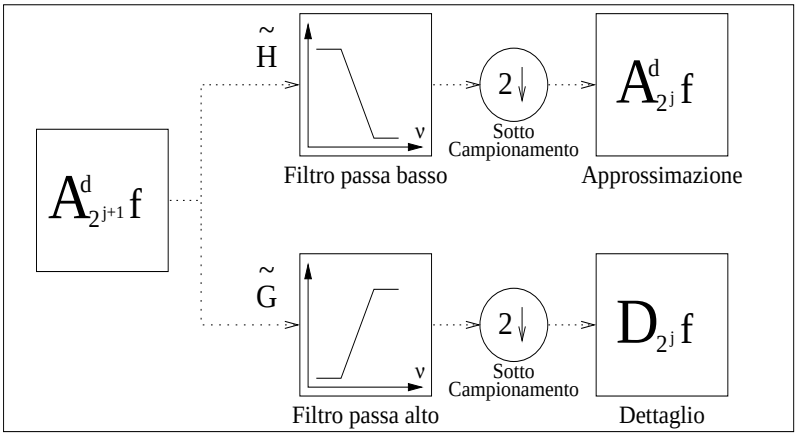


Figura 2.13: Decomposizione su un livello, ovvero calcolo di $A^d_{2^j} f$ e $D_{2^j} f$ attraverso operazioni di filtro e sottocampionamento su $A^d_{2^{j+1}} f$.

La fase di sottocampionamento è estremamente importante ai fini della non ridondanza della decomposizione.
Infatti, si supponga di avere un segnale costituito da 1000 campioni: poiché l'operazione di filtro, di per sé, non altera il numero di campioni in uscita[1], in mancanza di sottocampionamento, si avrebbe la situazione presentata in Fig. 2.14, cioè un segnale risultante costituito da 2000 campioni.

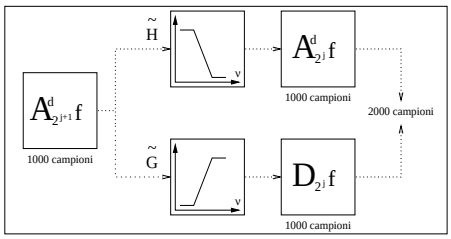


Figura 2.14: Operazioni di filtro su un segnale di 1000 campioni.

Al fine di evitare tale ridondanza nella decomposizione, occorre operare un sottocampionamento all'uscita dei filtri, onde dimezzare il numero di campioni uscenti, ed ottenere un segnale costituito da un numero di campioni, pari al numero di campioni del segnale entrante, si veda Fig. 2.15.

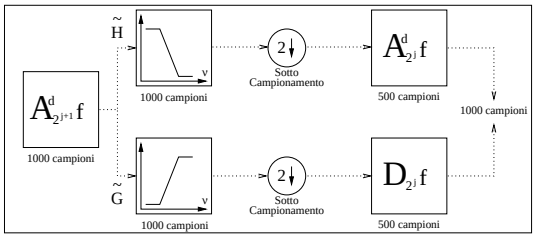


Figura 2.15: Operazioni di filtro e sottocampionamento su un segnale di 1000 campioni.

[1]In realtà, come visto in (A.1), un segnale di N campioni, filtrato attraverso un filtro di M campioni, risulta in un segnale di $N + M - 1$ campioni; tuttavia, in questo contesto, fare l'ipotesi che il numero di campioni in uscita da un filtro, sia uguale al numero di campioni entrante, consente di trattare il problema in maniera più lineare, ma ugualmente corretta: infatti, l'algoritmo reale, ponendo al segnale particolari condizioni al contorno, riesce ad eliminare gli $M - 1$ campioni provenienti dal filtraggio degli estremi del segnale, riconducendosi, così, al caso degli N campioni in uscita dal filtro.

L'operazione di sottocampionamento ha delle implicazioni fisiche molto profonde.
Definendo ν_N la frequenza di campionamento di un segnale, il Teorema di Nyquist assicura che la componente di frequenza più elevata in tal segnale, possa essere al massimo $\nu_N/2$.
Scegliendo i filtri $\widetilde{H}$ e $\widetilde{G}$ in maniera tale che filtrino, dallo spettro del segnale, rispettivamente, le frequenze maggiori di $\nu_N/4$ e le frequenze minori di $\nu_N/4$, la frequenza massima possibile del segnale all'uscita del filtro passa basso, è $\nu_N/4$.
Il Teorema di Nyquist, allora, assicura che un tale segnale può essere campionato a due volte il valore della frequenza massima, ovvero, $2 \cdot \nu_N/4 = \nu_N/2$; per portare la frequenza di campionamento del segnale, da ν_N—il filtraggio ha infatti lasciato inalterato il numero di campioni—a $\nu_N/2$, occorre ignorare un campione sì ed uno no, ovvero sottocampionare.
In sintesi, il filtraggio dimezza la risoluzione con cui si effettua l'analisi, lasciando invariato il numero di campioni del segnale, cioè la scala; il sottocampionamento, invece, dimezza la scala, lasciando invariata la risoluzione.

Per quanto concerne la ricostruzione del segnale, invece, si ha che, in virtù di (2.26), le operazioni di filtro sui segnali discreti $A^d_{2^j} f$ e $D_{2^j} f$, sono precedute da un sovracampionamento che, aggiungendo un campione nullo tra ogni coppia di campioni, raddoppia la scala del segnale, lasciandone invariata la risoluzione, si veda Fig. 2.16.

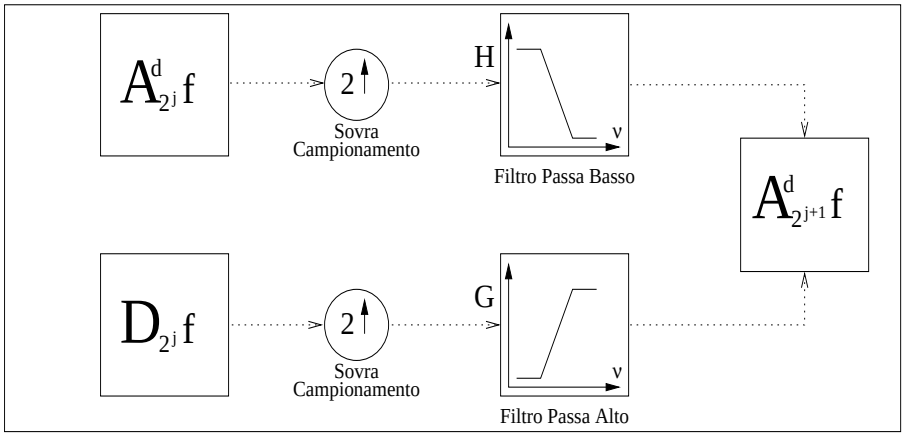


Figura 2.16: Ricostruzione su un livello, ovvero calcolo di $A^d_{2^{j+1}} f$ attraverso operazioni di filtro e sovracampionamento su $A^d_{2^j} f$ e $D_{2^j} f$.

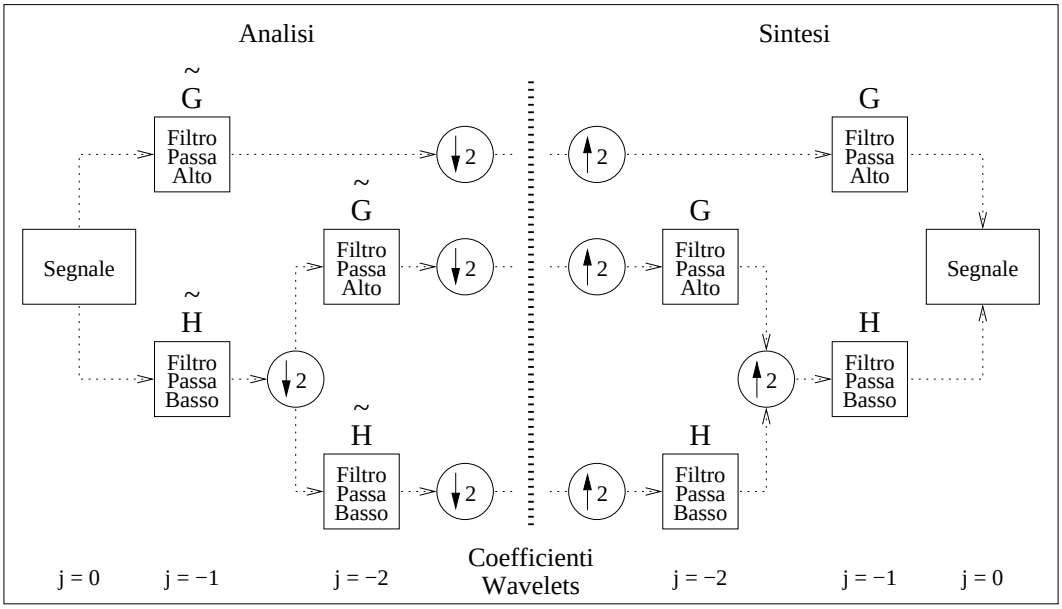


Figura 2.17: Decomposizione e ricostruzione su due livelli.

Sia l'algoritmo di decomposizione, individuato dalla (2.20) e dalla (2.23), che l'algoritmo di ricostruzione, individuato dalla (2.26), determinano la possibilità di impostare l'analisi sotto forma di una struttura piramidale: infatti, in fase di analisi, la decomposizione in approssimazione e dettaglio discreti, ad una determinata risoluzione 2^j, è completamente ricavabile dalla approssimazione discreta alla risoluzione immediatamente superiore 2^{j+1}, mentre, in fase di sintesi, la approssimazione discreta ad una determinata risoluzione 2^{j+1} è completamente ricostruibile dalla decomposizione in approssimazione e dettaglio discreti alla risoluzione immediatamente inferiore 2^j.
Questo consente di impostare la decomposizione e la ricostruzione su più livelli, come mostrato in Fig. 2.17.
In particolare, ammettendo che il segnale da analizzare sia costituito da N campioni, con $N = 2^J$, J intero positivo, il primo livello di decomposizione, ovvero alla risoluzione 2^{-1}, a causa del sottocampionamento, produce $N/2$ campioni, rappresentanti l'approssimazione discreta di primo livello, ed $N/2$ campioni, rappresentanti il dettaglio discreto di primo livello, si veda Tab. 2.1.

Livello	Risoluzione	N. campioni app. disc.	N. campioni det. disc.
1	2^{-1}	$N/2$	$N/2$
2	2^{-2}	$N/4$	$N/4$
...	...	...	...
j	2^{-j}	$N/2^j$	$N/2^j$
...	...	...	...
$J-1$	$2/N$	2	2
J	$1/N$	1	1

Tabella 2.1: Decomposizione su J livelli.

Il secondo livello di decomposizione, ovvero alla risoluzione 2^{-2}, decomponendo gli $N/2$ campioni dell'approssimazione discreta di primo livello, produce $N/4$ campioni, rappresentanti l'approssimazione discreta di secondo livello, ed $N/4$ campioni, rappresentanti il dettaglio discreto di secondo livello.
Il livello di decomposizione $log_2 N = log_2 2^J = J$ è l'ultimo, e decompone l'approssimazione discreta del livello precedente, in due soli campioni: uno rappresentante l'approssimazione discreta di livello J, ed uno rappresentante il dettaglio discreto di livello J.
Pertanto, si veda Fig. 2.18, a seguito di J livelli di decomposizione, il segnale originale, come formalizzato in (2.25), è decomposto in un campione, rappresentante l'approssimazione discreta di livello J, ed in $N-1$ campioni, rappresentanti i dettagli discreti ai vari livelli: per la precisione, di questi $N-1$ campioni, $N/2$ rappresentano i dettagli discreti di primo livello, $N/4$ rappresentano i dettagli discreti di secondo livello, $N/8$ rappresentano i dettagli discreti di terzo livello, e così di seguito fino all'ultimo campione, rappresentante il dettaglio discreto di livello J.

Il discorso è speculare per quel che riguarda la ricostruzione di un segnale a partire dalla sua decomposizione wavelet ortogonale.
Infatti, partendo dal livello di decomposizione J, il sovracampionamento aggiunge un campione nullo, al singolo campione rappresentante l'approssimazione discreta di livello J, ed un altro campione nullo al singolo campione rappresentante il dettaglio discreto di livello J; sommando poi l'approssimazione discreta ed il dettaglio discreto sovracampionati e filtrati, si ottiene l'approssimazione discreta di livello $J-1$.
Il procedimento continua analogamente, fino a che, dopo J livelli di ricostruzione, è ottenuta l'approssimazione di livello 0, ovvero il segnale originale.

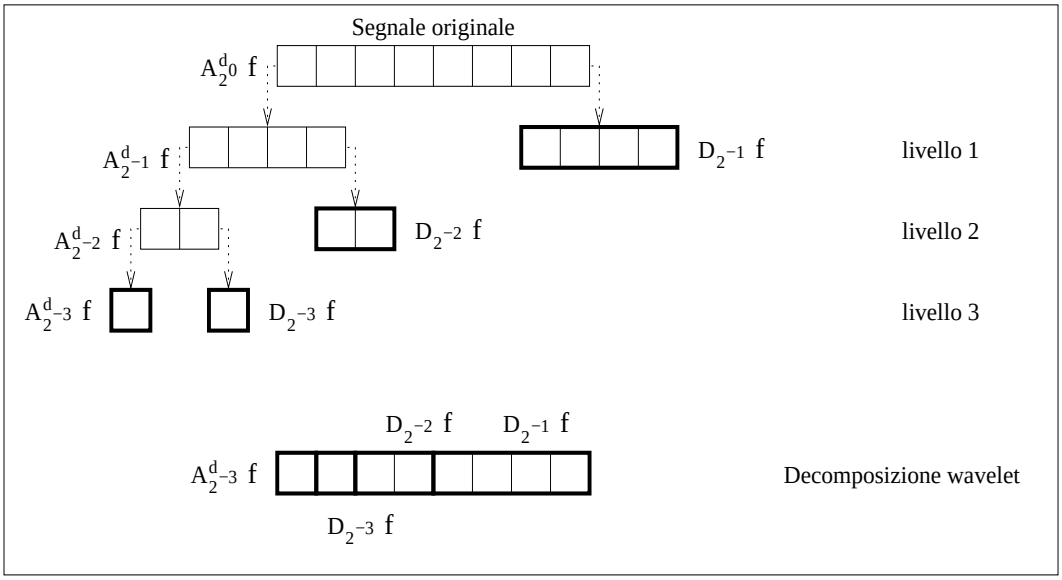


Figura 2.18: Decomposizione di un segnale di 8 campioni su 3 livelli.

2.6 Implementazioni numeriche dell'analisi multirisoluzione

L'analisi multirisoluzione pone le basi per una efficace implementazione numerica dell'analisi wavelet: in particolare, essendo la fase di decomposizione, e quella di ricostruzione, riconducibili ad operazioni di filtro del segnale, tutta l'analisi, in virtù di (A.2), è riconducibile ad operazioni di convoluzione tra il segnale ed i filtri; questo consente una implementazione estremamente lineare, dal punto di vista numerico, dell'analisi multirisoluzione.

2.6.1 Analisi multirisoluzione mono-dimensionale

A titolo di esempio [9], sia $f(x)$ un segnale mono-dimensionale discretizzato:

$$f(x) = [8, 4, 1, 3] \tag{2.40}$$

In precedenza si è assunto che, per comodità di normalizzazione, la risoluzione finita del segnale originale fosse da considerarsi $2^0 = 1$, cosicché il segnale $f(x)$ in (2.40) può essere raffigurato come in Fig. 2.19.

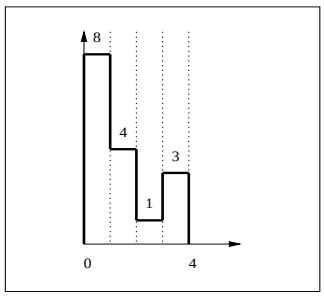


Figura 2.19: Il segnale [8,4,1,3].

Al fine di calcolare la decomposizione wavelet ortogonale del segnale $f(x)$, occorre effettuare successive proiezioni sugli spazi V_{2^j} e O_{2^j}, ovvero riesprimerlo in termini delle rispettive basi ortogonali:

$$(\phi_n^j)_{n\in\mathbb{Z}} = \big(\phi(2^j x - n)\big)_{n\in\mathbb{Z}} \tag{2.41}$$

$$(\psi_n^j)_{n\in\mathbb{Z}} = \big(\psi(2^j x - n)\big)_{n\in\mathbb{Z}} \tag{2.42}$$

dove si è tralasciato momentaneamente il fattore di normalizzazione $2^{j/2}$.
In particolare, una scelta mirata alla semplicità, potrebbe essere:

$$\phi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & altrove \end{cases} \qquad \psi(x) = \begin{cases} 1 & 0 \le x < 1/2 \\ -1 & 1/2 \le x < 1 \\ 0 & altrove \end{cases} \tag{2.43}$$

il che equivale a scegliere, come base ortogonale, quella di Haar.
Con una tale scelta, le basi ortogonali degli spazi V_{2^j} e O_{2^j} si riducono a funzioni lineari a tratti, quindi relativamente semplici da gestire.
Il segnale (2.40), decomposto nella base di V_1, si veda Fig. 2.20, risulta:

$$f(x) = 8 \cdot \phi_0^0(x) + 4 \cdot \phi_1^0(x) + 1 \cdot \phi_2^0(x) + 3 \cdot \phi_3^0(x) \tag{2.44}$$

Calcolando i coefficienti di approssimazione e dettaglio alla risoluzione successiva 2^{-1}, si veda (2.13), ovvero, calcolando $(\langle f(u), \phi_{2^{-1}}(u - 2^1 n)\rangle)_{n\in\mathbb{Z}}$ e $(\langle f(u), \psi_{2^{-1}}(u - 2^1 n)\rangle)_{n\in\mathbb{Z}}$, attraverso le basi di $V_{2^{-1}}$ e $O_{2^{-1}}$ mostrate in Fig. 2.21, si ha:

$$f(x) = 6 \cdot \phi_0^{-1}(x) + 2 \cdot \phi_1^{-1}(x) + 2 \cdot \psi_0^{-1}(x) - 1 \cdot \psi_1^{-1}(x) \tag{2.45}$$

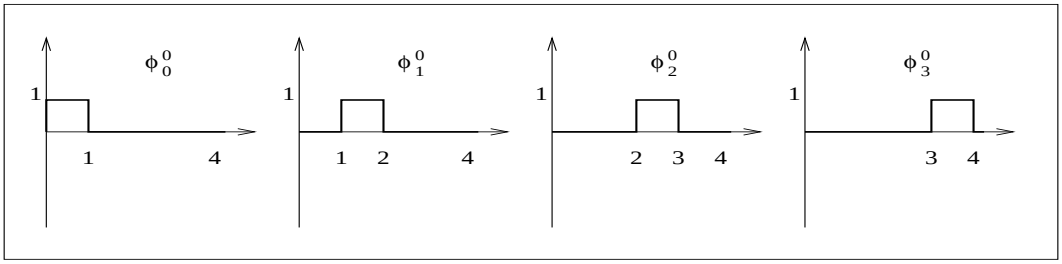


Figura 2.20: Alcuni elementi della base ortogonale di V_1.

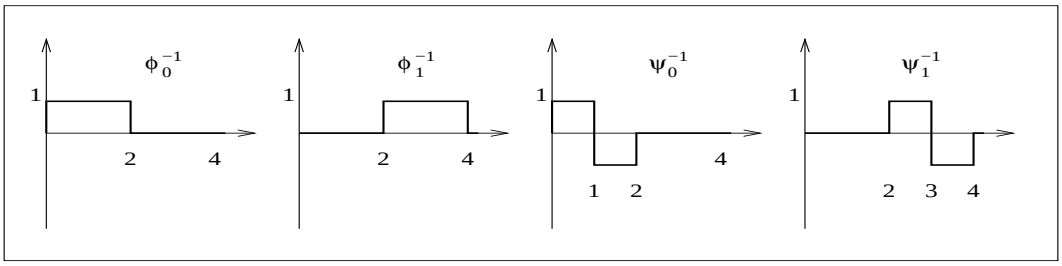


Figura 2.21: Alcuni elementi della base ortogonale di $V_{2^{-1}}$ e della base ortogonale di $O_{2^{-1}}$.

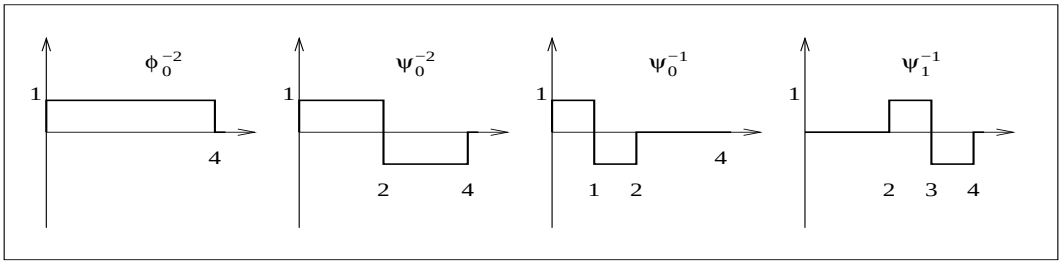


Figura 2.22: Alcuni elementi della base ortogonale di $V_{2^{-2}}$ e della base ortogonale di $O_{2^{-2}}$.

Da ultimo, calcolando, in maniera analoga, i coefficienti alla risoluzione 2^{-2}, ovvero decomponendo $f(x)$ sulle basi di $V_{2^{-2}}$ e $O_{2^{-2}}$ mostrate in Fig. 2.22, si ottiene:

$$f(x) = 4 \cdot \phi_0^{-2}(x) + 2 \cdot \psi_0^{-2}(x) + 2 \cdot \psi_0^{-1}(x) - 1 \cdot \psi_1^{-1}(x) \tag{2.46}$$

In sostanza, il segnale $f(x) = [8, 4, 1, 3]$, essendo costituito da $4 = 2^2$ campioni, viene completamente decomposto in 2 livelli, come si vede dalla Tab. 2.2.

Livello di Decomposizione	Approssimazioni	Dettagli
0	[8 , 4 , 1 , 3]	
1	[6 , 2]	[2 , -1]
2	[4]	[2]

Tabella 2.2: Decomposizione del vettore $[8, 4, 1, 3]$

In particolare, i coefficienti della sua decomposizione wavelet sono:

$$[4, 2, 2, -1] \tag{2.47}$$

Si noti una particolarità molto interessante del lavorare con la base di Haar non normalizzata: approssimare il segnale ad un determinato livello, equivale a fare la media, a due a due, dei campioni dell'approssimazione discreta di livello immediatamente precedente, mentre, calcolare i dettagli, equivale a farne, a due a due, la semidifferenza.
La semplicità dell'algoritmo, dovuta alla scelta della base di Haar non normalizzata fatta in (2.43), è esclusivamente un caso particolare della (2.20) e della (2.23), che prevedono il filtraggio del segnale attraverso un filtro passa basso ed uno passa alto, seguita da una operazione di sottocampionamento ogni 2 campioni.
Questo è comprensibile, considerando che le operazioni di semisomma e semidifferenza possono essere facilmente ricondotte ad operazioni di filtro.
Infatti, dato il segnale $f(x) = [8, 4, 1, 3]$ ed il filtro $\widetilde{H}$, avente risposta all'impulso $[1/2, 1/2]$, l'operazione di filtro tra f e $\widetilde{H}$—si veda la definizione di operazione di filtro in (A.1)—dà, come segnale in uscita, ignorando i due campioni estremi:

$$[6, 2.5, 2] \tag{2.48}$$

D'altro canto, l'operazione di filtro tra f ed il fitro $\widetilde{G}$, avente risposta all'impulso $[-1/2, 1/2]$, dà come segnale in uscita, ignorando i due campioni estremi:

$$[2, 1.5, -1] \tag{2.49}$$

Operando un sottocampionamento ogni 2 campioni dei segnali risultanti dall'operazione di filtro, si ottengono i segnali:

$$[6, 2] \quad e \quad [2, -1] \tag{2.50}$$

ovvero l'approsimazione discreta ed il dettaglio discreto di livello 1, visti in Tab. 2.2.

Anche l'algoritmo di ricostruzione può essere implementato sotto forma di semplici operazioni: in particolare, ciascuna approssimazione discreta di livello j, è calcolata sommando, prima, e sottraendo, dopo, ai campioni dell'approssimazione discreta di livello $j+1$, i corrispondenti campioni del dettaglio discreto di livello $j+1$; in maniera analoga a quanto detto per la decomposizione, questo semplice algoritmo può essere ricondotto ad operazioni di filtro e sovracampionamento.

Si noti che, per basi diverse dalla Haar non normalizzata—già a partire dalla Haar normalizzata le risposte all'impulso per i filtri di decomposizione sono $\widetilde{H} = \left[1/\sqrt{2}, 1/\sqrt{2}\right]$ e $\widetilde{G} = \left[-1/\sqrt{2}, 1/\sqrt{2}\right]$—operazioni semplici come semisomme e semidifferenze, scompaiono in favore di più generali operazioni di moltiplicazione e addizione tra i campioni del segnale ed i coefficienti della risposta all'impulso del filtro.

2.6.2 Analisi multirisoluzione bi-dimensionale

L'algoritmo appena descritto per il caso mono-dimensionale, può essere generalizzato facilmente al caso bi-dimensionale [10].

Sia $\boldsymbol{M}_o$ una matrice di interi 8×8, come mostrato in (2.51).

$$\boldsymbol{M}_o = \begin{pmatrix} 64 & 2 & 3 & 61 & 60 & 6 & 7 & 57 \\ 9 & 55 & 54 & 12 & 13 & 51 & 50 & 16 \\ 17 & 47 & 46 & 20 & 21 & 43 & 42 & 24 \\ 40 & 26 & 27 & 37 & 36 & 30 & 31 & 33 \\ 32 & 34 & 35 & 29 & 28 & 38 & 39 & 25 \\ 41 & 23 & 22 & 44 & 45 & 19 & 18 & 48 \\ 49 & 15 & 14 & 52 & 53 & 11 & 10 & 56 \\ 8 & 58 & 59 & 5 & 4 & 62 & 63 & 1 \end{pmatrix} \tag{2.51}$$

Scelta una base, in questo caso la Haar non normalizzata di (2.43), ciascuna

riga della matrice originale $\boldsymbol{M}_o$ viene decomposta completamente, analizzando su 3 livelli successivi il segnale mono-dimensionale rappresentato dalla riga, ed ottenendo, così, la matrice semi-trasformata $\boldsymbol{M}_{st}$, in (2.52).

$$\boldsymbol{M}_{st} = \begin{pmatrix} 32.5 & 0 & 0.5 & 0.5 & 31 & -29 & 27 & -25 \\ 32.5 & 0 & -0.5 & -0.5 & -23 & 21 & -19 & 17 \\ 32.5 & 0 & -0.5 & -0.5 & -15 & 13 & -11 & 9 \\ 32.5 & 0 & 0.5 & 0.5 & 7 & -5 & 3 & -1 \\ 32.5 & 0 & 0.5 & 0.5 & -1 & 3 & -5 & 7 \\ 32.5 & 0 & -0.5 & -0.5 & 9 & -11 & 13 & -15 \\ 32.5 & 0 & -0.5 & -0.5 & 17 & -19 & 21 & -23 \\ 32.5 & 0 & 0.5 & 0.5 & -25 & 27 & -29 & 31 \end{pmatrix} \tag{2.52}$$

Su ciascuna colonna di $\boldsymbol{M}_{st}$ si applica, in seguito, l'algoritmo mono-dimensionale, decomponendo ciascuna colonna su 3 livelli ed ottenendo la matrice trasformata $\boldsymbol{M}_t$, ovvero la trasformata wavelet bi-dimensionale della matrice originale $\boldsymbol{M}_o$, si veda (2.53).

$$\boldsymbol{M}_t = \begin{pmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & 0.5 & 0.5 & 27 & -25 & 23 & -21 \\ 0 & 0 & -0.5 & -0.5 & 11 & 9 & -7 & 5 \\ 0 & 0 & -0.5 & -0.5 & -5 & 7 & -9 & 11 \\ 0 & 0 & 0.5 & 0.5 & 21 & -23 & 25 & -27 \end{pmatrix} \tag{2.53}$$

Avendo decomposto sia le colonne che le righe in maniera completa, la matrice $\boldsymbol{M}_t$ risulta costituita da una approssimazione discreta del segnale, individuata dal coefficiente 32.5 in alto a sinistra, e dai 63 restanti dettagli discreti.

Da quanto detto nel Paragrafo 2.5, l'approssimazione discreta di un segnale è ottenuta filtrando il segnale attraverso un filtro passa basso, mentre il dettaglio discreto è ottenuto filtrandolo attraverso un filtro passa alto: in sostanza, l'approssimazione discreta contiene una informazione legata alle basse frequenze del segnale, mentre il dettaglio discreto alle alte frequenze del segnale.
Dal momento che, in natura, la componente a bassa frequenza di un segnale ne dà la struttura portante, mentre la componente ad alta frequenza ne dà informazioni di contorno, può essere ragionevole ignorare parte dei coefficienti di dettaglio di $\boldsymbol{M}_t$, nella ricostruzione di $\boldsymbol{M}_o$.

Pertanto, scegliendo una soglia da porre ai coefficienti di dettaglio, si determina il grado di informazione che si è disposti a perdere; tanto la soglia è maggiore, tanto il numero di coefficienti di dettaglio minori, in modulo, di questa, è alto, tanto l'informazione perduta è imponente.
Ad esempio, azzerando i coefficienti di dettaglio minori, in modulo, di una soglia posta a 5, la matrice $\boldsymbol{M}_t$ diventa la matrice $\boldsymbol{M}_{tsoglia}$, mostrata in (2.54).

$$\boldsymbol{M}_{tsoglia} = \begin{pmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 27 & -25 & 23 & -21 \\ 0 & 0 & 0 & 0 & 11 & 9 & -7 & 5 \\ 0 & 0 & 0 & 0 & -5 & 7 & -9 & 11 \\ 0 & 0 & 0 & 0 & 21 & -23 & 25 & -27 \end{pmatrix} \tag{2.54}$$

Una tale matrice può essere trasmessa o memorizzata, in maniera compressa, tramite, ad esempio, una codifica *Run Length*; tuttavia, la ricostruzione di $\boldsymbol{M}_o$ a partire dalla matrice in (2.54), non darà esattamente la matrice $\boldsymbol{M}_o$, a causa dell'informazione perduta dopo l'applicazione della soglia: in questo senso, occorre valutare le prestazioni in termini di compressione che si vogliono raggiungere e l'errore in ricostruzione che si è disposti a tollerare.

Capitolo 3

L'esperimento ALICE

In questo capitolo si descriverà, in maniera sintetica, l'esperimento ALICE (*A Large Ion Collider Experiment*); in particolare, si focalizzerà l'attenzione sull'algoritmo studiato dall'Istituto Nazionale di Fisica Nucleare (INFN) di Torino, per la compressione dei dati provenienti dai *Silicon Drift Detectors* dell'*Inner Tracking System* del rivelatore di ALICE, e sulle sue prestazioni.

3.1 ALICE al CERN

ALICE [11] è un esperimento progettato dal CERN (*European Organization for Nuclear Research*) e finalizzato allo studio di collisioni tra ioni pesanti presso l'acceleratore LHC (*Large Hadron Collider*), un anello superconduttore facente parte del complesso di acceleratori del CERN, si veda Fig. 3.1.

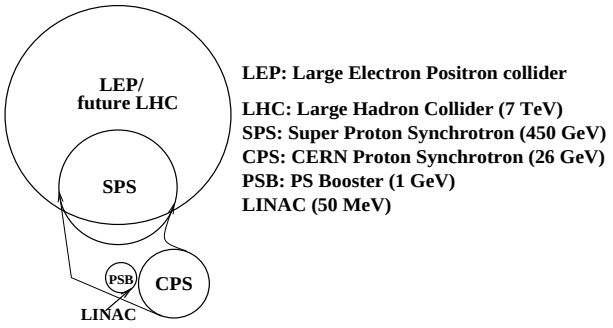


Figura 3.1: Il complesso di acceleratori del CERN.

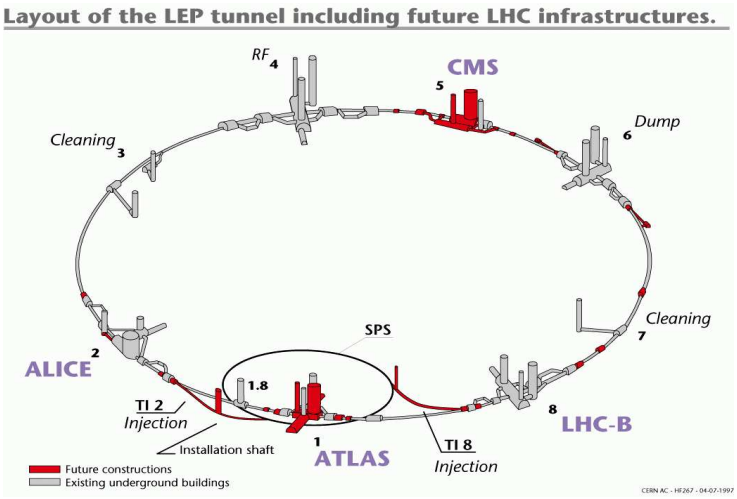


Figura 3.2: Il tunnel circolare di LEP.

Il progetto per la realizzazione di LHC, prevede l'installazione di tale anello, all'interno dei 27 Km del già esistente tunnel circolare di LEP (*Large Electron Positron collider*), come si vede in Fig. 3.2, e prevede l'inizio del suo funzionamento a partire dall'anno 2005.
Oltre ad ALICE, anche altri esperimenti sono previsti all'interno dell'acceleratore LHC: ATLAS (*A Toroidal Lhc ApparatuS*), CMS (*Compact Muon Solenoid*) e LHCb (*B-physics at LHC*).
Il fine ultimo di tutti questi esperimenti è l'approfondimento della fisica ad energie oltre la soglia del TeV.

Il sistema di iniezione dei protoni nell'anello di LHC prevede l'azione combinata di tre acceleratori, facenti parte del complesso del CERN, e già utilizzati per pre-accelerare gli elettroni ed i positroni di LEP: l'acceleratore lineare LINAC (*LINear ACcelerator*), l'acceleratore circolare CPS (*Cern Proton Synchrotron*) e l'acceleratore circolare SPS (*Super Proton Synchrotron*), capaci di accelerare i protoni fino a, rispettivamente, 50 MeV, 26 GeV e 450 GeV.
Tale pre-accelerazione, e l'azione combinata di cavità risonanti a radiofrequenza e magneti superconduttori raffeddati ad elio liquido, consentono, presso LHC, collisioni frontali di fasci di protoni con una energia di 7 TeV ciascuno, corrispondenti ad una energia del centro di massa di 14 TeV—ovvero circa 10 volte l'energia prodotta presso LEP—e collisioni tra ioni pesanti aventi energia del centro di massa di 5.5 TeV.

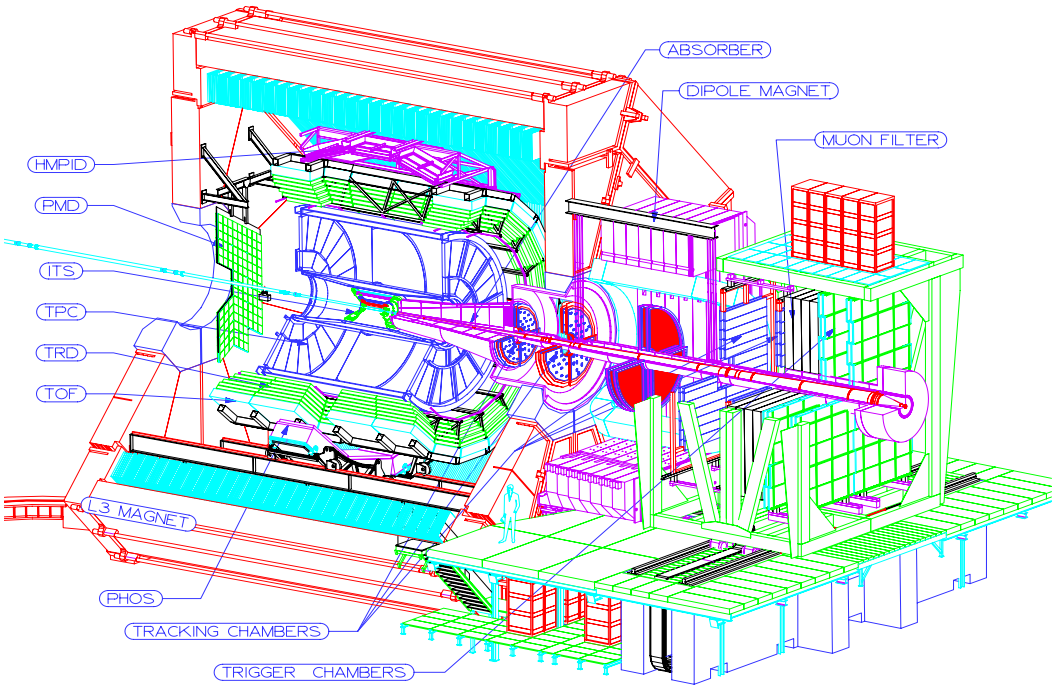


Figura 3.3: Sezione longitudinale del rivelatore di ALICE.

Il fine ultimo dello studio della fisica degli ioni pesanti a così alte energie, si riconduce, di fatto, allo studio della materia ad alte densità e temperature. La teoria prevede che, per densità sufficientemente alte, abbia inizio una transizione dalla materia adronica ad un plasma deconfinato di quarks e gluoni (*Quark Gluon Plasma*, QGP), transizione che avvenne, nel senso inverso, agli albori della nascita dell'universo, circa 10^{-5}s dopo il *Big Bang*, e che ancora oggi, con tutta probabilità, gioca un ruolo fondamentale nel nucleo delle stelle di neutroni in fase di collasso.
L'esperimento ALICE, in particolare, studia la formazione del plasma di quarks e gluoni QGP in seguito a collisioni di ioni pesanti (Pb o Ca), aventi energia del centro di massa di circa 5.5 TeV.
Tale studio richiede che la tecnologia di rivelazione dell'esperimento ALICE sia sensibile alla maggior parte degli osservabili conosciuti, come adroni, elettroni, muoni, fotoni, e che sia in grado di distinguere tracce diverse, anche alle elevate densità caratteristiche degli esperimenti ad alta energia, nel caso specifico di ALICE, fino a 90 tracce/cm^2.

Il rivelatore [12, 13] di ALICE, in Fig. 3.3, è costituito da due componenti principali: una parte centrale, composta da rivelatori adibiti allo studio dei segnali adronici, ed una parte frontale, costituita da uno spettrometro muonico adibito allo studio del comportamento dei quarkonia nella materia densa.
La parte centrale copre $\pm 45°$ sull'asse azimutale ed è racchiusa entro un magnete (*L3 MAGNET*) che produce un campo di 0.2 T.
Questa parte del rivelatore è costruita su più strati, ed è costituita, nell'ordine, dai sei strati cilindrici di rivelatori al silicio di ITS (*Inner Tracking System*), dal rivelatore cilindrico TPC (*Time Projection Chamber*), dai sei strati del rivelatore TRD (*Transition Radiation Detector*), dai contatori del TOF (*Time Of Flight*), dallo spettrometro fotonico PHOS (*PHOton Spectrometer*) e da un insieme di contatori RICH (*Ring Imaging CHerenkov counters*) ottimizzati per l'identificazione di particelle ad alto momento (*High Momentum Particle IDentification*, HMPID).
La parte frontale, invece, è costituita da un rivelatore muonico, di cui fanno parte, un elemento assorbente (*ABSORBER*) prossimo al vertice di collisione, uno spettrometro con un dipolo magnetico (*DIPOLE MAGNET*) e un filtro muonico (*MUON FILTER*), costituito da una parete di ferro che seleziona i muoni.
Si noti, da ultimo, che la geometria aperta del rivelatore di ALICE ne consente eventuali modificazioni o miglioramenti durante il corso degli esperimenti.

3.2 *Inner Tracking System* di ALICE

Il tracciamento delle particelle cariche, prodotte dalle collisioni centrali tra ioni pesanti, è ottenuto principalmente da TPC, un rivelatore a gas che si trova ad una distanza minima dall'asse centrale—raggio minimo—di circa 90 cm, ad una distanza massima—raggio massimo—di circa 250 cm, ed avente, come massima densità delle tracce discriminabili, 0.1 cm^{-2}.
Il gas contenuto in tale rivelatore è sottoposto ad un campo elettrico: in questa maniera, al passaggio di particelle cariche, alcuni elettroni appartenenti agli atomi del gas vengono strappati e successivamente condotti dal campo elettrico alle estremità della camera, ove viene ricostruito il cammino effettuato dalla particella.
Per distanze, dall'asse centrale, inferiori a 90 cm, e quindi a densità delle tracce maggiori, il tracciamento è ottenuto da ITS [14].

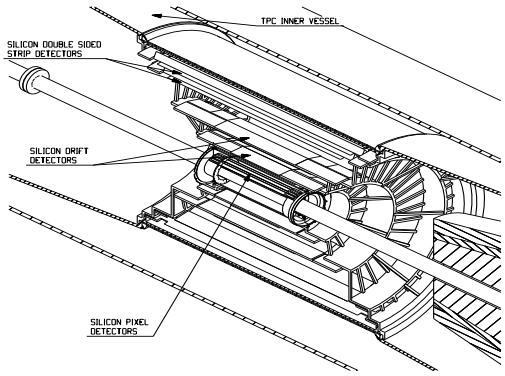


Figura 3.4: I diversi strati di ITS.

3.2.1 I sei strati di rivelatori al silicio

ITS è costituito da sei strati cilidrici (*layers*) di rivelatori al silicio, si veda Fig. 3.4: *layer* 1 di raggio 4 cm, *layer* 2 di 7 cm, *layer* 3 di 14.9 cm, *layer* 4 di 23.8 cm, *layer* 5 di 39.1 cm e *layer* 6 di raggio 43.6 cm.
Poiché la densità delle particelle aumenta al diminuire del raggio—ad esempio, per valori del raggio minori di 24 cm la densità delle particelle può arrivare fino a 90 cm^{-2}, mentre per valori del raggio intorno ai 45 cm la densità delle particelle può arrivare fino a 1 cm^{-2}—ciascuna coppia di strati è costituita da rivelatori al silicio aventi specifiche di risoluzione e precisione, adatte alla densità caratteristica della zona in cui operano: nella fattispecie, *layer* 1 e *layer* 2 sono costituiti da *Silicon Pixel Detectors* (SPD), *layer* 3 e *layer* 4 da *Silicon Drift Detectors* (SDD), mentre *layer* 5 e *layer* 6 da *Silicon Strip Detectors* (SSD); si veda Tab. 3.1 per alcune specifiche di tali rivelatori.

Parametro		*Pixel*	*Drift*	*Strip*
Precisione spaziale rϕ	μm	12	38	20
Precisione spaziale z	μm	70	28	830
Risoluzione di due tracce rϕ	μm	100	200	300
Risoluzione di due tracce z	μm	600	600	2400
Area attiva	mm^2	13.8×82	72.5×75.3	73×40

Tabella 3.1: Alcune specifiche dei rivelatori al silicio di ITS.

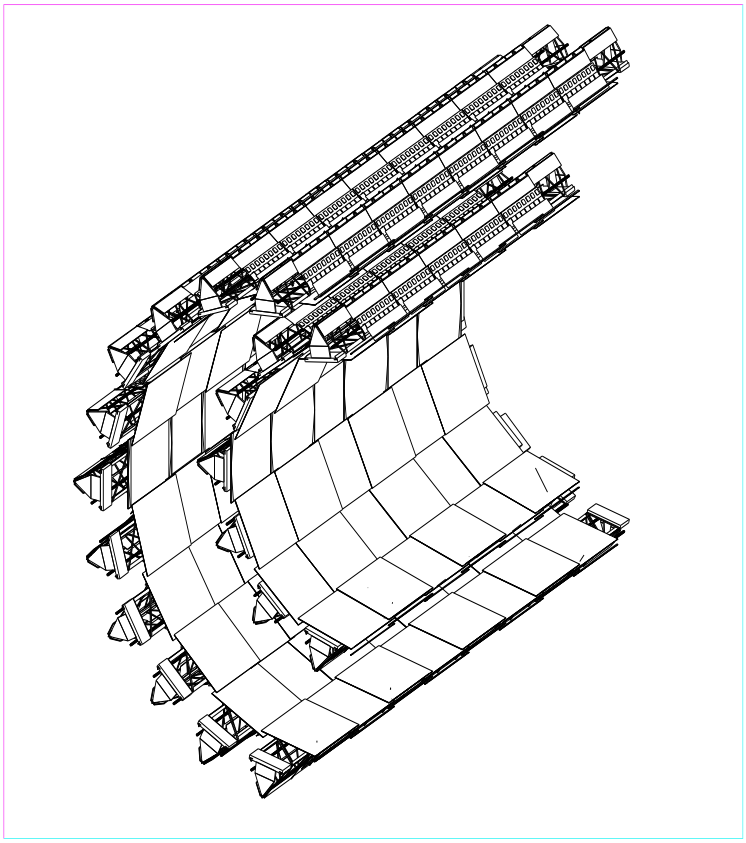


Figura 3.5: Sezione longitudinale di *layer* 3 e di *layer* 4 dell'ITS.

3.2.2 Gli strati intermedi: *layer* 3 e *layer* 4

Entrando nel dettaglio di *layer* 3 e di *layer* 4, i rivelatori SDD stazionano su strutture portanti a pioli, dette *ladders*; nella fattispecie, *layer* 3 è costituito da 14 *ladders*, ciascuna reggente 6 rivelatori SDD, mentre *layer* 4 è costituito da 22 *ladders*, ciascuna reggente 8 rivelatori SDD, si veda Fig. 3.5.
Su ogni *ladder*, in aggiunta ai rivelatori SDD, è posto anche un numero doppio di moduli elettronici, detti *front-end modules*, ciascuno dei quali è interfacciato ad un semi-rivelatore, si veda Fig. 3.10; inoltre, su entrambe le estremità di ciascuna *ladder* è posto un modulo, detto *end ladder module*, adibito alla compressione dei dati provenienti dal modulo *front-end module* ed all'invio dei dati compressi ad un apposito sistema di acquisizione dati (*Data AcQuisition system*, DAQ).

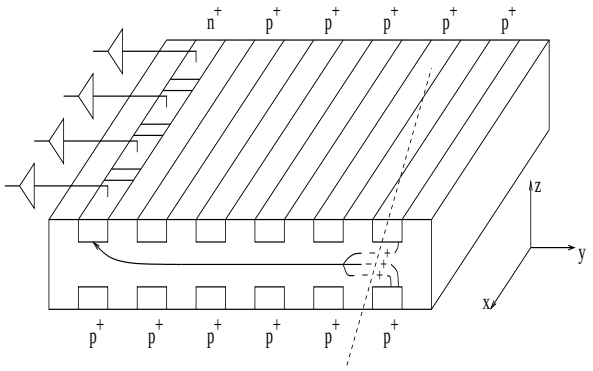


Figura 3.6: Principo di funzionamento di un rivelatore SDD.

3.2.3 *Silicon Drift Detectors*

Un rivelatore SDD è costituito, fisicamente, da un substrato di silicio drogato n, una serie di impianti paralleli drogati p^+, posizionati su entrambe le superfici del substrato, ed un array di impianti drogati n^+, posizionato ad una estremità del substrato, come mostrato in Fig. 3.6.
La presenza degli impianti paralleli drogati p^+, determina la formazione di due regioni di svuotamento, che si estendono, in lunghezza, parallelamente al piano x-y del substrato, ed in profondità, lungo l'asse z, in particolare, dalle due superfici impiantate, verso l'interno del substrato, fino a toccarsi.
Tramite l'utilizzo di *guard cathodes* e *voltage dividers*, viene applicato, a ciascun impianto drogato p^+, una determinata tensione negativa; la tensione applicata è tanto più negativa, quanto più l'impianto drogato p^+ è distante dall'array di impianti drogati n^+.
In questa maniera si viene a creare, all'interno del rivelatore, un potenziale parabolico, con il minimo sul piano centrale del substrato, come mostrato in Fig. 3.7, e quindi, un campo elettrostatico, parallelo al piano centrale del substrato, determinante una regione di deriva (*drift*) per gli elettroni.
In particolare, gli elettroni originati da un evento ionizzante nella regione di svuotamento, si veda Fig. 3.6, sono portati a raggiungere il minimo di tale potenziale, in questo caso il piano centrale del substrato, dove, a causa della componente longitudinale del campo elettrico, iniziano a muoversi verso l'array di impianti drogati n^+; d'altra parte, le lacune che si vengono a creare in seguito allo stesso evento ionizzante, tendono ad essere raccolte dal più vicino impianto drogato p^+.

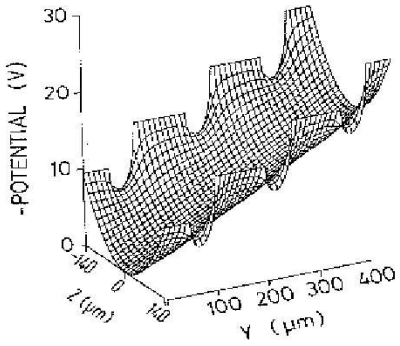


Figura 3.7: Potenziale elettrico sul piano y-z del rivelatore: le superfici del rivelatore sono sui piani $z = 140$ μm e $z = -140$ μm.

Data la funzione che hanno all'interno del rivelatore SDD, gli impianti drogati p^+ vengono anche detti catodi del rivelatore, mentre, gli impianti drogati n^+, vengono anche detti anodi del rivelatore.
Il tempo impiegato dalla distribuzione di carica, creata in seguito ad un evento ionizzante, per raggiungere un determinato anodo (tempo di *drift*), consente di risalire alla posizione, sull'asse y (o anche detto asse dei tempi), in cui è avvenuto l'impatto della particella; inoltre, la posizione di tale anodo all'interno dell'array di impianti drogati n^+, consente la determinazione della posizione, sull'asse x (o anche detto asse degli anodi), dell'impatto; tale metodo, permette di determinare le coordinate dell'impatto della particella ionizzante sul rivelatore, con una precisione di circa 30 μm, e permette una risoluzione di due tracce di circa 300 μm.

Vari progetti di rivelatori SDD sono stati portati avanti nell'ambito dell'esperimento ALICE; sviluppato nel 1988, il progetto ALICE-D1 è tra i più recenti e funzionali, e prevede, per il substrato del rivelatore, l'utilizzo di silicio NTD (*Neutron Transmutation Doped*), in virtù della estrema omogeneità del suo drogaggio.
Il prototipo ALICE-D1 prevede una forma del rivelatore esagonale, si veda Fig.3.8, onde minimizzare la sovrapposizione di rivelatori adiacenti su una stessa *ladder*; tale struttura, infatti, separa, del tutto, l'area attiva, responsabile della effettiva rivelazione delle particelle ionizzanti, dalla cosiddetta *guard area*, contenente *guard cathodes* e *voltage dividers*, responsabili del movimento di deriva degli elettroni.

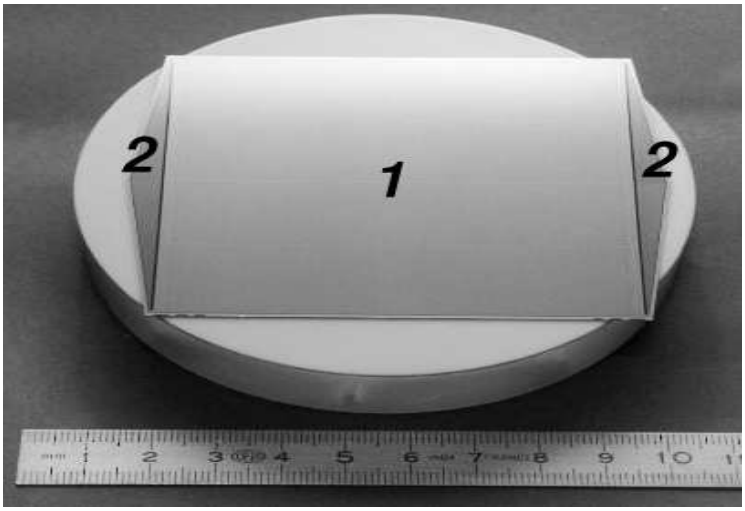


Figura 3.8: Il prototipo ALICE-D1 per il rivelatore SDD: 1) area attiva, 2) *guard area.*

In tale struttura esagonale, avente resistività pari a 3 kΩcm e spessore pari a 300 μm, un array di 256 anodi, lungo 75.3 mm, è posizionato su uno dei lati maggiori del rettangolo costituente l'area attiva del rivelatore, ed un ulteriore array di 256 anodi, è posizionato sul lato opposto, ad una distanza di 72.5 mm; all'interno dell'area attiva, sono posizionati 512 catodi paralleli ai due anodi, di cui, 256 nella prima metà del rivelatore, ed i restanti 256 nella seconda metà. Il rivelatore SDD così progettato ha una struttura bi-direzionale, infatti, il movimento di deriva della distribuzione di carica creata da un evento ionizzante, avviene dal centro del rivelatore, verso uno dei due array di anodi.

3.2.4 Elettronica di *drift*

La distribuzione di carica generata su ciascun SDD, in seguito ad un evento, è un segnale elettrico avente un andamento gaussiano.
Tale segnale può essere raccolto da uno o più anodi, a seconda, ovviamente, della quantità di carica rilasciata dalla particella ionizzante e, a seconda, della distanza che intercorre tra, la posizione sull'asse dei tempi dell'impatto della particella, e l'array di anodi; questa ultima osservazione [15] trova spiegazione nella forza di repulsione reciproca che gli elettroni esercitano l'uno sull'altro durante il tragitto di deriva, si veda Fig.3.9.

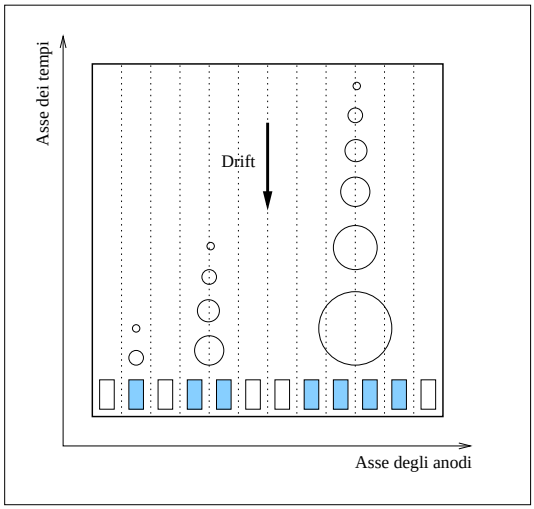


Figura 3.9: Schema di evoluzione della distribuzione elettronica.

La quantità di dati prodotta da ciascun SDD è estremamente elevata: ciascun semi-rivelatore, infatti, ha 256 anodi, e, per ciascun anodo, occore rilevare 256 campioni, in maniera tale da avere una informazione completa su tutto il tragitto di deriva.
L'elettronica di *drift* ha, sostanzialmente, il compito di condurre tali dati, dal rivelatore SDD, ad una unità di acquisizione dati, DAQ; nello specifico, ha il compito di amplificare il segnale proveniente dal rivelatore SDD, convertire tale segnale in una rappresentazione digitale, effettuarne la compressione, ed inviarlo a DAQ, digitalizzato e compresso.
Il vantaggio di tale architettura per l'elettronica di *drift* è duplice: da una parte, convertire il segnale analogico in campioni digitali, immediatamente dopo la preamplificazione, consente di evitare una degradazione del segnale durante la trasmissione, dall'altra, inviare i dati direttamente a DAQ, senza un'analisi in tempo reale dei dati, consente l'utilizzo di strumenti di analisi più flessibili e potenti sui dati acquisiti definitivamente.
Le funzionalità dell'elettronica di *drift* sono suddivise in due blocchi elettronici ben distini, entrambi, come accennato in Paragrafo 3.2.2, integrati sulla struttura della *ladder*, si veda Fig. 3.10: il modulo elettronico *front-end module*, che si occupa dell'amplificazione e della conversione da analogico a digitale, ed il modulo elettronico *end ladder module*, che si occupa della compressione e della trasmissione a DAQ.

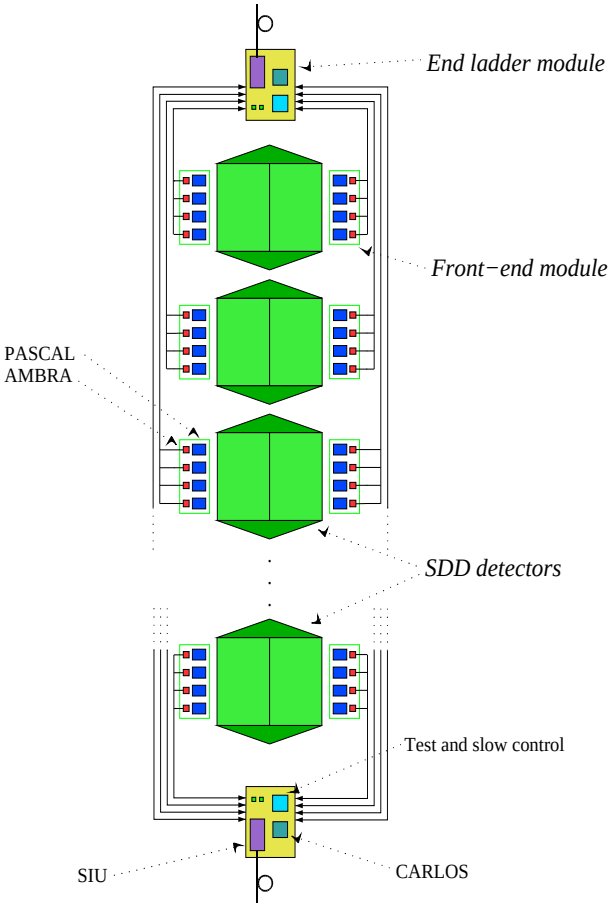


Figura 3.10: Elettronica di *drift* su una *ladder* di SDD.

Il modulo elettronico *front-end module* è il primo ad interfacciarsi con i segnali provenienti da ciascuno dei 256 anodi, si veda Fig. 3.10; questo è composto da quattro sottomoduli PASCAL (*Preamplifier, Analog Storage and Conversion from Analog to digitaL*), che effettuano l'amplificazione, la memorizzazione analogica e la conversione da analogico a digitale, e da quattro sottomoduli AMBRA (*A Multievent Buffer Readout Architecture*), che effettuano la memorizzazione digitale di più eventi, in maniera tale da diminuire la frequenza di trasmissione dei dati all'esterno del modulo *front-end module*.

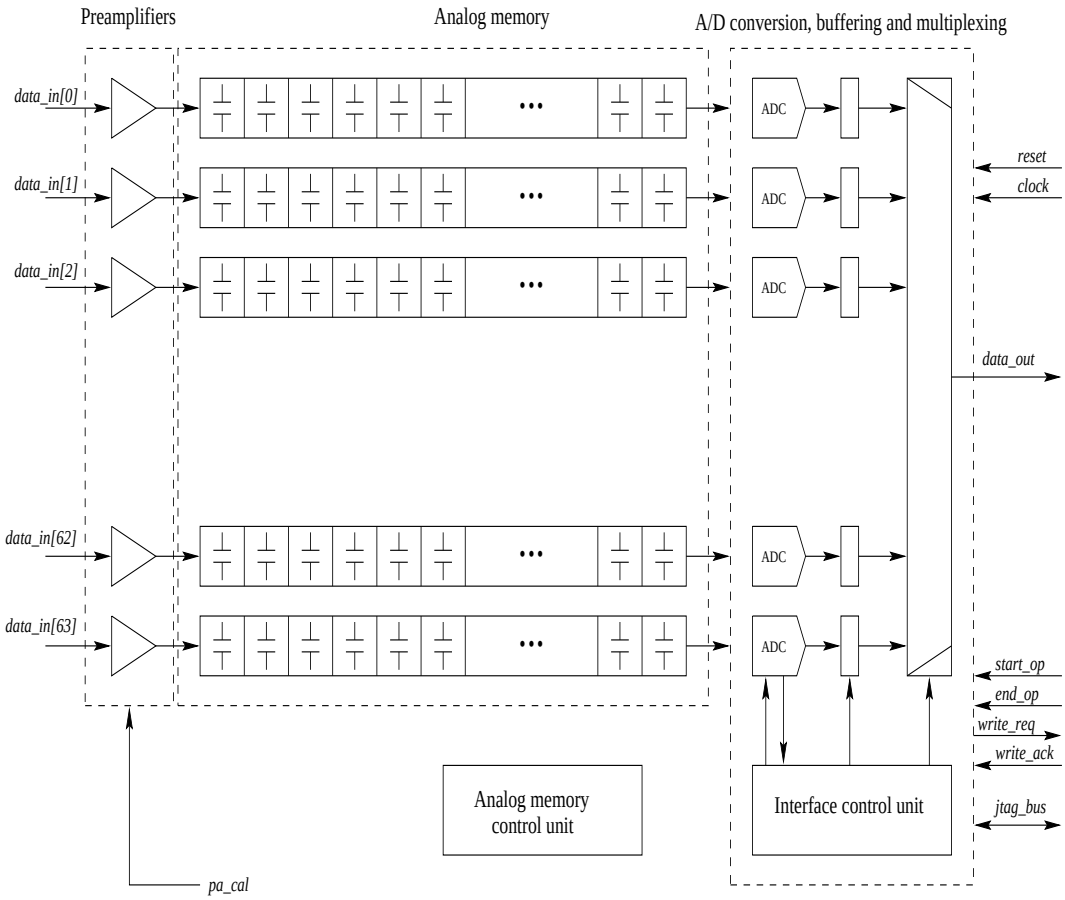


Figura 3.11: Architettura del sottomodulo PASCAL.

Ciascun sottomodulo PASCAL, si veda Fig. 3.11, amplifica i segnali di 64 anodi, riscrivendoli a 40 MHz all'interno di una memoria analogica costituita da 64 $\times$ 256 celle; successivamente, il convertitore ADC converte tali segnali analogici in segnali digitali ad 8 bit, trasmettendoli al sottomodulo AMBRA. In realtà, essendo il tempo medio di *drift* della distribuzione elettronica, inferiore a 6 μs, campionare i segnali provenienti dagli anodi a 40 MHz, ovvero ogni 25 ns, significa produrre al massimo 240 campioni, contro i 256 previsti dalla memoria analogica; tuttavia, le 256 celle della memoria analogica, consentono di far fronte ad eventuali cambiamenti nei parametri dei rivelatori e relativi aumenti del tempo medio di *drift*.

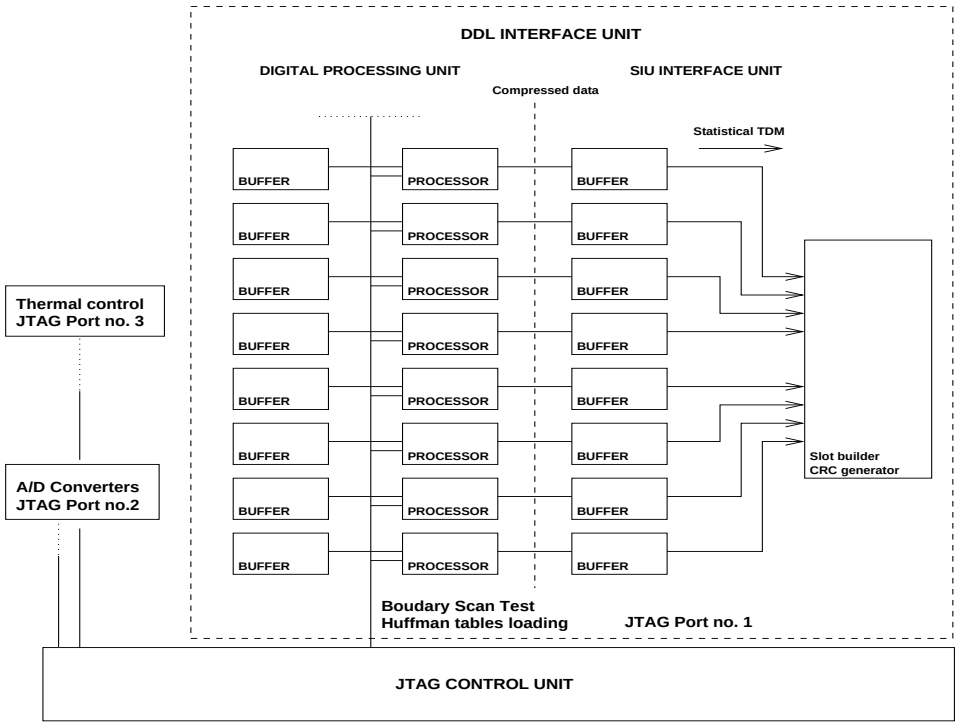


Figura 3.12: Architettura del sottomodulo CARLOS.

Dal sottomodulo AMBRA, i dati vengono successivamente trasmessi al modulo *end ladder module*; essendo presenti due moduli *end ladder module* per *ladder*, ciascuno di tali moduli si occupa della metà dei rivelatori SDD presenti sulla *ladder* in cui si trova, si veda Fig. 3.10.
All'interno di ciascun modulo *end ladder module* è ospitato il sottomodulo CARLOS (*Compression And Run Length encOding Subsystem*), che gestisce la compressione dei dati provenienti dai sottomoduli AMBRA.
Come mostrato in Fig. 3.12, il sottomodulo CARLOS è costituito da 8 *macro channels*, ciascuno dei quali è una sequenza di un *buffer* in ingresso, un processore dedicato, che fornisce l'implementazione fisica degli algoritmi di compressione che saranno discussi in Paragrafo 3.3, ed un *buffer* in uscita.
Al fine di consentire una lettura ed una elaborazione in tempo reale, dei dati prodotti per evento, ciascun sottomodulo CARLOS lavora in parallelo su tutti gli 8 canali, ognuno dei quali processa i dati provenienti da uno solo dei semirivelatori presenti sulla *ladder* in cui si trova il sottomodulo.

3.3 Compressione dati tramite CARLOS

La necessità della compressione dati ottenuta tramite il sottomodulo CARLOS, nasce dalla elevata quantità di dati prodotti per evento e dal finito spazio di memorizzazione previsto per ciascuno di tali eventi.
Ogni semi-rivelatore, infatti, ha 256 anodi ed il segnale di ognuno di questi viene campionato a 40 MHz, ottenendo 256 campioni per ciascun anodo; questo significa che, usando 8 bit per campione, ciascun semi-rivelatore produce, per ogni evento, circa 64 Kbyte di dati.
In particolare, per il *layer* 3, costituito da 14 *ladders* aventi 6 rivelatori ciascuna, la quantità totale di dati, per ogni evento, ammonta a circa 10.5 Mbyte, mentre per il *layer* 4, costituito da 22 *ladders* aventi 8 rivelatori ciascuna, ammonta a circa 22 Mbyte, portando l'ammontare totale dei dati prodotti da questi due strati per ciascun evento, a circa 32.5 Mbyte.
Essendo disponibili 1.5 Mbyte, per la memorizzazione dei dati prodotti in ciascun evento dal *layer* 3 e dal *layer* 4, occorre sviluppare una tecnica di compressione capace di ridurre i 32.5 Mbyte di dati, prodotti dal *layer* 3 e dal *layer* 4 per ciascun evento, ai 1.5 Mbyte disponibili in fase di memorizzazione; in altre parole, tale tecnica di compressione deve essere in grado di ridurre il totale dei dati prodotti dal *layer* 3 e dal *layer* 4 in ciascun evento, di un fattore pari ad almeno 22.

Un algoritmo di compressione dati mono-dimensionale ed uno bi-dimensionale, sono in fase di sviluppo e valutazione presso l'Istituto Nazionale di Fisica Nucleare di Torino [16].

3.3.1 L'algoritmo di compressione mono-dimensionale

Essendo sviluppato ad hoc, sul modello della distribuzione statistica dei dati campione prodotti in collisioni di test, tale algoritmo è estremamente funzionale e mirato al problema.
Tuttavia, la specificità che lo contraddistingue, rende necessaria la presenza di parametri modificabili al suo interno, onde poterlo ottimizzare a fronte di dati reali, aventi strutture statistiche diverse da quelle supposte.

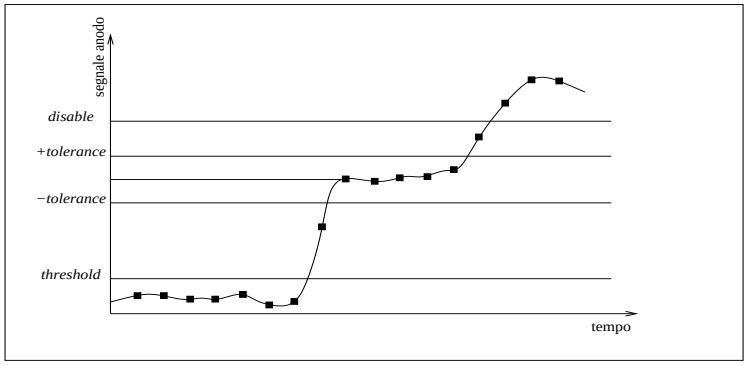


Figura 3.13: Parametri di *threshold*, *tolerance* e *disable*.

I parametri che, all'interno dell'algoritmo, possono essere modificati, sono, il parametro *threshold*, il parametro *tolerance* ed il parametro *disable*, si veda Fig 3.13; variando i valori di tali parametri, si può ottimizzare l'algoritmo in funzione della distribuzione statistica dei dati con cui ci si trova a lavorare.
Tali parametri entrano, nell'algoritmo, nella seguente maniera:

- *threshold*: un campione in ingresso minore di questo valore viene posto a zero.
- *tolerance*: una differenza tra campioni successivi minore in modulo di questo valore viene posta a zero.
- *disable*: un campione in ingresso maggiore di questo valore non è soggetto al meccanismo *tolerance*.

In sostanza, il meccanismo di *threshold*, annullando i campioni aventi valore inferiore al parametro *threshold*, elimina, di fatto, quella parte del segnale che rappresenta il "rumore" di fondo del rivelatore SDD.
Il meccanismo di *tolerance*, ponendo il valore di un campione uguale a quello del suo precedente, se la differenza tra i due è inferiore al parametro *tolerance*, consente di assorbire piccole variazioni, fisicamente non significative, tra campioni adiacenti.
Da ultimo, il meccanismo di *disable*, imponendo la non applicazione del meccanismo di *tolerance*, per tutti i campioni aventi valore superiore al parametro *disable*, consente una analisi molto più precisa dei campioni aventi valore elevato, ovvero, quei campioni che rappresentano parte della carica rilasciata sul rivelatore dall'evento ionizzante.

L'algoritmo mono-dimensionale viene applicato parallelamente a ciascuno degli 8 segnali digitali in ingresso a CARLOS; in particolare, i parametri *threshold*, *tolerance* e *disable*, possono essere impostati diversamente a seconda del canale, in maniera tale da ottimizzare l'algoritmo su ciascun semi-rivelatore.
In una prima fase di test, si sfrutta la disattivabilità dell'algoritmo, onde creare la tavola di Huffman relativa alle differenze tra i valori di campioni adiacenti non compressi; tale tavola di Huffman viene caricata, su ciascun sottomodulo CARLOS, solo durante la fase operativa.
L'algoritmo, vero e proprio, parte analizzando sequenzialmente i campioni in ingresso: i campioni minori del parametro *threshold* vengono posti a zero, mentre quelli maggiori vengono lasciati inalterati.
Di ciascun campione passato attraverso il meccanismo di *threshold*, l'algoritmo ne calcola la differenza con il precedente; se il campione è minore del parametro *disable* ed il valore assoluto della differenza precedentemente calcolata, è minore del parametro *tolerance*, allora il valore di tale differenza viene posto a zero, altrimenti, il valore di tale differenza viene lasciato inalterato.
Le differenze ottenute dall'analisi tramite i meccanismi di *threshold*, *disable* e *tolerance*, sono poi codificate, attraverso la tavola di Huffman precedentemente calcolata.
In ultima analisi, i valori ottenuti tramite la codifica di Huffman, sono ulteriormente codificati tramite la codifica *Run Length*.
L'alta probabilità di trovare lunghe sequenze di zeri o lunghi piedistalli nei dati provenienti dal modulo elettronico *front-end module*, rende estremamente efficace l'implementazione di una codifica *Run Length*, in cascata alla codifica di Huffman: in questa maniera, infatti, l'azione combinata dei meccanismi di *threshold*, *tolerance* e *disable*, trasforma le lunghe sequenze costanti (di zeri o non), provenienti dal modulo *front-end module*, in lunghe sequenze di zeri; tali sequenze di zeri, passando attraverso la codifica di Huffman e la codifica *Run Length*, sono successivamente espresse tramite un solo bit posto a zero, rappresentante la codifica di Huffman per lo zero, ed un valore ad otto bit, rappresentante il valore di *Run Length*.

3.3.2 Progetto di algoritmo bi-dimensionale

Ciascun semi-rivelatore SDD, è caratterizzato, in uscita, da 256 segnali mono-dimensionali, ognuno dei quali è, a sua volta, caratterizzato da 256 campioni.

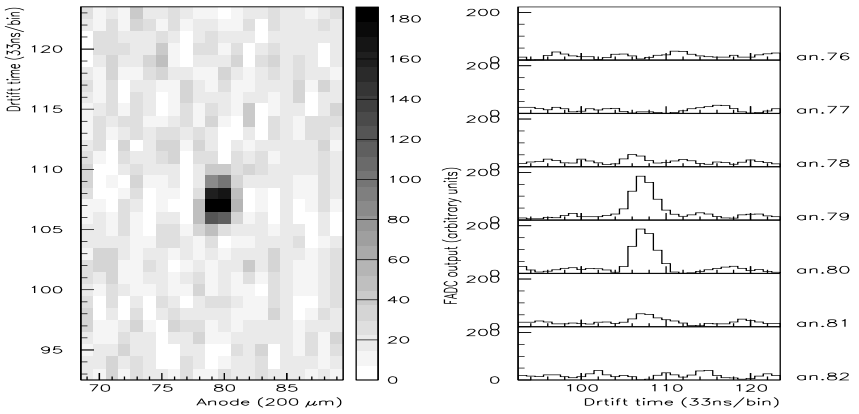


Figura 3.14: Operazione di *cluster finding*.

L'idea alla base della compressione bi-dimensionale, è di non considerare i singoli segnali mono-dimensionali indipendentemente l'uno dall'altro, ma di considerare il segnale fornito da ciascun semirivelatore SDD, nella sua bi-dimensionalità, come fosse una immagine.
Tale algoritmo è, tuttavia, in una fase di sviluppo ancora molto arretrata, rispetto all'algoritmo mono-dimensionale.

3.3.3 Prestazioni

Gli aspetti qualificanti un algoritmo di compressione sono, il valore di compressione raggiunto e l'errore nella ricostruzione dei dati originali a partire da quelli compressi.

Nell'ambito della compressione dei dati provenienti dai *Silicon Drift Detectors* dell'*Inner Tracking System* di ALICE, il valore di compressione raggiunto può essere caratterizzato dal coefficiente $c = \frac{\text{n° bit in uscita}}{\text{n° bit in ingresso}}$, ovvero l'inverso del valore di *compression ratio* definito nel Paragrafo 1.1.3.

La quantificazione dell'errore di ricostruzione è invece meno immediata: in questo caso specifico, infatti, non interessa un errore generale, calcolato cioè su tutti i campioni relativi ad ogni semi-rivelatore, bensì un errore limitato ai campioni che, data la loro elevata ordinalità, sono rappresentativi del passaggio di una particella; la quantificazione di questo errore, pertanto, deve passare dal riconoscimento di tali agglomerati di campioni, come mostrato in Fig. 3.14, operazione conosciuta col nome di *cluster finding*.

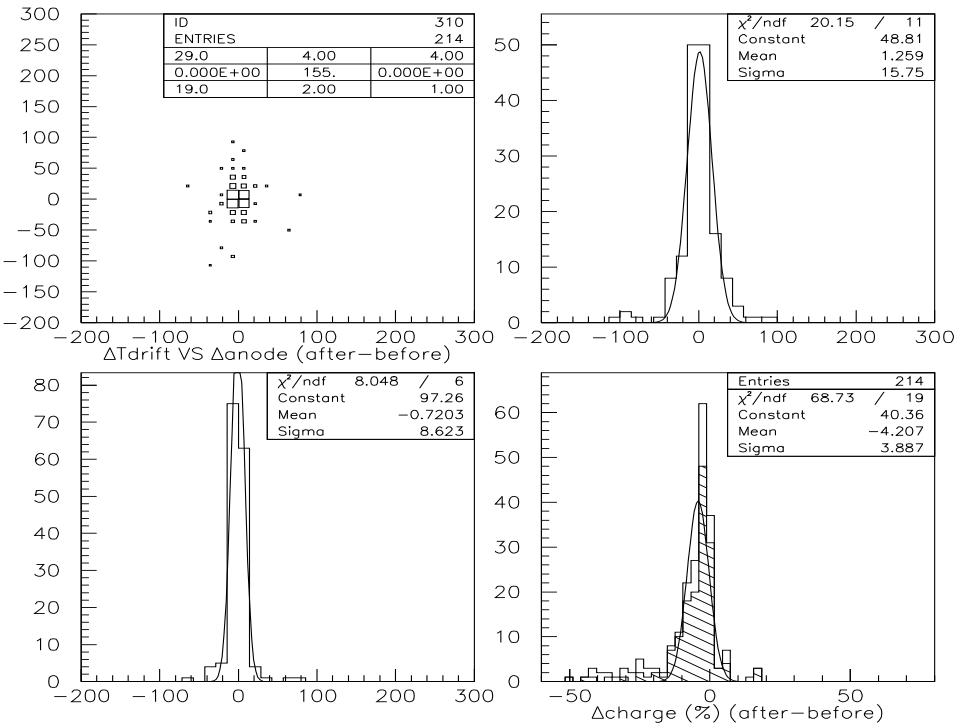


Figura 3.15: Effetti della compressione sulla misura delle coordinate del centroide del *cluster* e della sua carica (in basso a destra).

La ricerca dei *cluster* di carica viene effettuata analizzando il segnale proveniente da ciascun anodo: in particolare, quando un campione sull'asse dei tempi, ed il suo successivo, sono superiori ad una certa soglia, il campione si intende facente parte di un *cluster* di carica, mentre, quando un campione sull'asse dei tempi, ed il suo precedente, sono inferiori alla soglia, il campione si intende non facente più parte del *cluster* di carica; due o più *cluster* di carica, trovati su anodi adiacenti, formano un *cluster* bi-dimensionale, manifestazione del passaggio di una particella in quella zona del SDD.
In particolare, il calcolo dell'errore si riconduce a quantificare le inesattezze sistematiche generate dalla compressione sui *cluster* ricostruiti, quali, una traslazione sistematica delle coordinate del centroide del *cluster* ricostruito, un peggioramento della risoluzione, sull'asse dei tempi e degli anodi, con cui è determinata la posizione del centroide del *cluster* ricostruito, una stima errata della carica sottesa dal *cluster* ricostruito.

Applicando l'algoritmo di compressione mono-dimensionale, descritto in Paragrafo 3.3.1, ai dati generati dai fasci di prova del Settembre '98, si ottiene un coefficiente di compressione $c = 30 \times 10^{-3}$, con il parametro di *threshold* posto a 40 ed il parametro di *tolerance* posto a 0; questo risultato soddisfa i requisiti minimi di compressione per l'asperimento ALICE, che prevedono la riduzione di 32.5 Mbyte a 1.5 Mbyte, ovvero $c \leq 46 \times 10^{-3}$.
L'analisi dell'errore sui dati ricostruiti, non palesa traslazioni sistematiche del centroide; nel riquadro in alto a sinistra di Fig. 3.15, infatti, si nota come, comprimendo e ricostruendo 214 eventi, l'istogramma delle differenze tra la coordinata del centroide sull'asse del tempo, prima e dopo la compressione, e delle differenze tra la coordinata del centroide sull'asse degli anodi, prima e dopo la compressione, manifesti una popolazione tendenzialmente concentrata sull'origine (0,0), ad indicare che l'algoritmo di compressione preserva la posizione del centroide del *cluster* di carica ricostruito.
La compressione, tuttavia, determina un peggioramento della risoluzione con cui sono identificate le coordinate del *cluster* ricostruito, in particolare, determina un'incertezza sull'asse dei tempi pari a 16 μm, ed un'incertezza sull'asse degli anodi pari a 8 μm; nei riquadri in alto a destra ed in basso a sinistra di Fig. 3.15, infatti, si nota come la proiezione sull'asse dei tempi, dell'istogramma bi-dimensionale precedentemente discusso, porti la distribuzione ad essere fittata con una gaussiana avente $\sigma = 15.75$, mentre, la proiezione sull'asse degli anodi, porti la distribuzione ad essere fittata con una gaussiana avente $\sigma = 8.623$.
In ultima analisi, il riquadro in basso a destra, di Fig. 3.15, mostra come l'andamento, sui 214 eventi, della differenza percentuale tra la carica del *cluster* prima e dopo la compressione, possa essere fittato con una gaussiana centrata in -4.207, ad indicare che la compressione determina una sottostima della carica del *cluster* ricostruito, di circa il 4%.

Capitolo 4

Analisi multirisoluzione: compressione dei dati di ALICE

In questo capitolo verrà illustrato il lavoro svolto al fine di sviluppare un algoritmo di compressione basato sull'analisi multirisoluzione, e finalizzato alla compressione dei dati prodotti dai semi-rivelatori SDD appartenenti al *layer* 3 e al *layer* 4 dell'*Inner Tracking System* di ALICE.
Lo sviluppo di tale algoritmo di compressione si è articolato in quattro fasi distinte, alcune delle quali caratterizzate dall'utilizzo di uno strumento software mirato:

- una prima fase, tesa alla individuazione delle modalità di base con cui implementare l'algoritmo, nell'ambito della compressione dei dati provenienti dai semi-rivelatori SDD.

- una seconda fase, sviluppata attraverso il Wavelet Toolbox di Matlab, finalizzata all'ottimizzazione dell'algoritmo di compressione in base alla natura dei dati.

- una terza fase, sviluppata mediante Simulink, mirata all'individuazione di una possibile architettura attraverso cui implementare l'algoritmo di compressione ottimizzato.

- una quarta fase, sviluppata attraverso una subroutine FORTRAN, atta alla quantificazione delle prestazioni dell'algoritmo sviluppato, in termini di compressione effettuata ed errore in ricostruzione.

4.1 Compressione dei dati prodotti dai rivelatori SDD

Come visto all'inizio del Paragrafo 3.3, entrambi, il *layer* 3 ed il *layer* 4, sono costituiti da rivelatori SDD; in particolare, poiché ciascun semi-rivelatore SDD produce, per ogni evento, una quantità di dati pari a 65536 campioni, ognuno dei quali digitalizzato ad 8 bit, l'ammontare complessivo di dati prodotti dall'insieme di tutti i semi-rivelatori presenti sul *layer* 3 e sul *layer* 4, è pari ad un totale di 32.5 Mbyte, a seguito di ciascun evento.
L'esigenza di comprimere tali dati, nasce dal limitato spazio di memorizzazione disponibile per i dati prodotti in ciascun evento: in particolare, essendo tale spazio pari a 1.5 Mbyte, la compressione effettuata sul totale dei dati prodotti in ciascun evento dal *layer* 3 e dal *layer* 4, e quindi sui dati di ciascun semi-rivelatore SDD, deve essere tale da ridurli di un fattore pari ad almeno 22 o, equivalentemente, deve garantire un valore del parametro $c = \frac{\text{n}^\text{o} \text{ bit in uscita}}{\text{n}^\text{o} \text{ bit in ingresso}}$, minore od uguale rispetto 46×10^{-3}.

4.1.1 Compressione tramite CARLOS

Il progetto attuale, circa la compressione di tali dati, prevede l'utilizzo dell'algoritmo discusso nel Paragrafo 3.3.1, ovvero l'algoritmo di compressione sviluppato dall'Istituto Nazionale di Fisica Nucleare di Torino.
Tale algoritmo comprime, sequenzialmente, ciascuno dei 256 segnali corrispondenti agli anodi del semi-rivelatore; in particolare, la compressione viene sviluppata attraverso il posizionamento in cascata di più tecniche di codifica, quali la codifica Delta, la codifica Huffman, e la codifica *Run Length*.
Inoltre, al fine di rendere l'algoritmo più duttile, in particolar modo a fronte di dati aventi una struttura statistica diversa, da quella supposta nella progettazione dell'algoritmo, sono introdotti tre parametri modificabili, rispettivamente, il parametro di *threshold*, di *tolerance* e di *disable*.

Le prestazioni dell'algoritmo in termini compressione, determinano un valore di c pari a circa 30×10^{-3}, minore del valore di riferimento 46×10^{-3}; inoltre, come discusso nel Paragrafo 3.3.3, l'errore di ricostruzione dovuto alla compressione, determina una sottostima della carica dei *cluster* ricostruiti di circa il 4%, ed una incertezza sulla posizione dei rispettivi centroidi, di circa 16 μm sull'asse dei tempi, e 8 μm sull'asse degli anodi, pur non palesando una loro traslazione sistematica rispetto alle posizioni originali.

4.1.2 Compressione basata sull'analisi multirisoluzione

L'idea di poter comprimere in maniera alternativa i dati prodotti da ciascun semi-rivelatore SDD, attraverso un algoritmo di compressione basato sull'analisi multirisoluzione, nasce sostanzialmente dalla crescente diffusione di tale tecnica, sia nell'ambito della compressione di segnali mono-dimensionali, che nell'ambito della compressione di segnali bi-dimensionali.
Due sono gli aspetti su cui verte tale algoritmo: da una parte, la possibilità di dare, attraverso l'analisi multirisoluzione, una rappresentazione equivalente del segnale analizzato, in termini di coefficienti di approssimazione e dettaglio; dall'altra, la propensione di tali coefficienti ad essere compressi attraverso semplici tecniche di codifica quali, ad esempio, la codifica *Run Length*.
L'idea caratterizzante tale algoritmo, infatti, è quella di effettuare l'analisi multirisoluzione dei segnali provenienti da ciascun semi-rivelatore SDD, in maniera tale da darne una rappresentazione equivalente in termini di coefficienti di approssimazione e dettaglio, comprimere tale rappresentazione equivalente, e memorizzarla, così compressa, al posto del segnale stesso.

Nell'implementazione dell'analisi multirisoluzione sui dati provenienti dai rivelatori SDD, il segnale $\boldsymbol{S}$ prodotto da ciascun semi-rivelatore, non è più visto come 256 segnali anodici separati, ciascuno dei quali costituito da 256 campioni digitalizzati ad 8 bit, bensì, è visto, o come un unico segnale di 65536 campioni, risultante dalla concatenazione dei 256 segnali anodici del semi-rivelatore, oppure come un segnale bi-dimensionale, costituito da 256 righe, ognuna delle quali caratterizzata dai 256 campioni corrispondenti ad un particolare segnale anodico del semi-rivelatore.
Nel primo caso, l'insieme dei dati prodotti per evento da ciascun semi-rivelatore, viene trattato come un unico array di 65536 campioni, come il seguente:

$$\boldsymbol{S} = \left(\underbrace{s_1, s_2, \ldots, s_{256}}_{1^{\text{o}}\text{ anodo}}, \underbrace{s_{257}, s_{258}, \ldots, s_{512}}_{2^{\text{o}}\text{ anodo}}, \ldots, \underbrace{s_{65281}, s_{65282}, \ldots, s_{65536}}_{256^{\text{o}}\text{ anodo}} \right) \tag{4.1}$$

Nel secondo caso, invece, l'insieme dei dati prodotti per evento, viene trattato come una matrice 256×256, come la seguente:

$$\boldsymbol{S} = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,256} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,256} \\ \vdots & \vdots & \ddots & \vdots \\ s_{256,1} & s_{256,2} & \cdots & s_{256,256} \end{pmatrix} \begin{matrix} 1^{\text{o}}\text{ anodo} \\ 2^{\text{o}}\text{ anodo} \\ \vdots \\ 256^{\text{o}}\text{ anodo} \end{matrix} \tag{4.2}$$

In particolare, per l'analisi multirisoluzione del segnale mono-dimensionale (4.1), una volta scelta la coppia di filtri di decomposizione $\widetilde{H}$ e $\widetilde{G}$, viene applicato lo schema di decomposizione mostrato in Fig. 2.18 su un numero di livelli da scegliersi tra 1 e 16.
Tale schema determina una decomposizione wavelet ortogonale $\boldsymbol{C}$ costituita da 65536 coefficienti; nella fattispecie, il numero dei coefficienti di approssimazione a_i ed il numero di quelli di dettaglio d_i, rispetto alla totalità dei coefficienti della decomposizione wavelet ortogonale, dipende dal numero di livelli di decomposizione con cui viene effettuata l'analisi multirisoluzione, come si vede dalla seguente:

$$\boldsymbol{S} = \left(s_1, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, s_{65536} \right) \quad \text{0 livelli di decomposizione}$$

$$\boldsymbol{C} = \left(\underbrace{a_1, \ldots\ldots, a_{32768}}_{\text{coeffs. app.}}, \underbrace{d_{32769}, \ldots\ldots, d_{65536}}_{\text{coeffs. dett.}} \right) \quad \text{1 livello di decomposizione}$$

$$\boldsymbol{C} = \left(\underbrace{a_1, \ldots\ldots, a_{16384}}_{\text{coeffs. app.}}, \underbrace{d_{16385}, \ldots\ldots\ldots\ldots, d_{65536}}_{\text{coeffs. dett.}} \right) \quad \text{2 livelli di decomposizione}$$

$$\boldsymbol{C} = \left(\underbrace{a_1, \ldots, a_{8192}}_{\text{coeffs. app.}}, \underbrace{d_{8193}, \ldots\ldots\ldots\ldots\ldots, d_{65536}}_{\text{coeffs. dett.}} \right) \quad \text{3 livelli di decomposizione}$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$\boldsymbol{C} = \left(\underbrace{a_1, a_2, a_3, a_4}_{\text{coeffs. app.}}, \underbrace{d_5, \ldots\ldots\ldots\ldots\ldots, d_{65536}}_{\text{coeffs. dett.}} \right) \quad \text{14 livelli di decomposizione}$$

$$\boldsymbol{C} = \left(\underbrace{a_1, a_2}_{\text{coeffs. app.}}, \underbrace{d_3, \ldots\ldots\ldots\ldots\ldots\ldots, d_{65536}}_{\text{coeffs. dett.}} \right) \quad \text{15 livelli di decomposizione}$$

$$\boldsymbol{C} = \left(\underbrace{a_1}_{\text{coeff. app.}}, \underbrace{d_2, \ldots\ldots\ldots\ldots\ldots\ldots, d_{65536}}_{\text{coeffs. dett.}} \right) \quad \text{16 livelli di decomposizione}$$

Nel caso dell'analisi multirisoluzione del segnale bi-dimensionale (4.2), invece, una volta scelta la coppia di filtri di decomposizione $\widetilde{H}$ e $\widetilde{G}$, viene applicato lo schema di decomposizione bi-dimensionale, di cui si è già parlato nel Paragrafo 2.6.2, su un numero di livelli da scegliersi tra 1 e 8.

Tale schema di decomposizione consiste, in prima battuta, nel decomporre ciascuna riga del segnale bi-dimensionale, attraverso l'analisi multirisoluzione, sul numero di livelli scelto; in seconda battuta, consiste nel decomporre ciascuna colonna, della matrice risultante dalla precedente decomposizione sulle righe, attraverso l'analisi multirisoluzione sviluppata su uno stesso numero di livelli. In questa maniera il segnale bi-dimensionale (4.2) viene decomposto nella corrispondente decomposizione wavelet ortogonale bi-dimensionale $\boldsymbol{C}$, caratterizzata da 65536 coefficienti; anche in questo caso, il numero dei coefficienti di approssimazione a_i, nelle sottomatrici in alto a sinistra, ed il numero dei rimanenti coefficienti di dettaglio d_i, dipende dal numero di livelli di decomposizione con cui viene effettuata l'analisi multirisoluzione:

$$\boldsymbol{S} = \begin{pmatrix} s_{1,1} & \cdots & s_{1,256} \\ \vdots & & \vdots \\ s_{256,1} & \cdots & s_{256,256} \end{pmatrix} \quad \text{0 livelli di decomposizione}$$

$$\boldsymbol{C} = \left(\begin{array}{ccc|ccc} a_{1,1} & \cdots & a_{1,128} & d_{1,129} & \cdots & d_{1,256} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{128,1} & \cdots & a_{128,128} & d_{128,129} & \cdots & d_{128,256} \\ \hline d_{129,1} & \cdots & d_{129,128} & d_{129,129} & \cdots & d_{129,256} \\ \vdots & & \vdots & \vdots & & \vdots \\ d_{256,1} & \cdots & d_{256,128} & d_{256,129} & \cdots & d_{256,256} \end{array}\right) \quad \text{1 livello di decomposizione}$$

$$\vdots \qquad\qquad \vdots$$

$$\boldsymbol{C} = \left(\begin{array}{cc|ccc} a_{1,1} & a_{1,2} & d_{1,3} & \cdots & d_{1,256} \\ a_{2,1} & a_{2,2} & d_{2,3} & \cdots & d_{2,256} \\ \hline d_{3,1} & d_{3,2} & d_{3,3} & \cdots & d_{3,256} \\ \vdots & \vdots & \vdots & & \vdots \\ d_{256,1} & d_{256,2} & d_{256,3} & \cdots & d_{256,256} \end{array}\right) \quad \text{7 livelli di decomposizione}$$

$$\boldsymbol{C} = \left(\begin{array}{c|ccc} a_{1,1} & d_{1,2} & \cdots & d_{1,256} \\ \hline d_{2,1} & d_{2,2} & \cdots & d_{2,256} \\ \vdots & \vdots & & \vdots \\ d_{256,1} & d_{256,2} & \cdots & d_{256,256} \end{array}\right) \quad \text{8 livelli di decomposizione}$$

L'utilità dell'effettuare l'analisi multirisoluzione dei segnali provenienti da ciascun semi-rivelatore SDD, si manifesta sotto forma di due proprietà, del tutto generali, caratterizzanti i coefficienti appartenenti alla decomposizione wavelet ortogonale di un segnale.
In primo luogo, i coefficienti di approssimazione sono caratterizzati da valori relativamente alti, poiché rappresentativi dell'approssimazione del segnale, mentre i coefficienti di dettaglio, essendo rappresentativi dei dettagli del segnale, sono caratterizzati da valori prossimi a zero.
In secondo luogo, i coefficienti di approssimazione e di dettaglio che costituiscono la decomposizione wavelet ortogonale di un segnale, intervengono in maniera diversa in fase di ricostruzione: in particolare, i coefficienti di approssimazione, essendo rappresentativi del solo contenuto a bassa frequenza del segnale, sono responsabili della ricostruzione della struttura portante del segnale, viceversa, i coefficienti di dettaglio, essendo rappresentativi del solo contenuto ad alta frequenza del segnale, sono responsabili della ricostruzione dei dettagli del segnale.

La tecnica più semplice ed efficace, che consente di sfruttare tali proprietà ai fini della compressione, consiste nell'azzerare tutti i coefficienti a_i e d_i minori, in modulo, di un determinato valore di soglia fissato; proprio questa tecnica è stata adottata nel caso specifico dell'algoritmo finalizzato alla compressione dei segnali provenienti dai semi-rivelatori SDD.
In particolare, la prima delle due proprietà appena citate, fa sì che, a seguito dell'imposizione di una soglia relativamente bassa, i coefficienti di approssimazione a_i rimangano pressoché inalterati, mentre, si vengano a creare lunghe sequenze di coefficienti di dettaglio d_i nulle; tali sequenze, come già visto nel Capitolo 1, si prestano ad essere compresse attraverso varie tecniche di codifica, una tra tutte la codifica *Run Length*.
Si noti che, se da una parte l'imposizione della soglia ai coefficienti, e la successiva codifica delle sequenze nulle, determina una compressione efficace della decomposizione wavelet ortogonale del segnale, dall'altra, proprio l'azzeramento dei coefficienti a_i e d_i minori, in modulo, della soglia—e quindi la modifica dei loro valori nei confronti dei rispettivi valori originali—fa sì che la ricostruzione del segnale, a partire dalla sua decomposizione wavelet ortogonale, non possa più essere *lossless*; in particolare, quanto più il valore della soglia applicato è alto, tanto più il numero di coefficienti azzerati è elevato, tanto più la ricostruzione del segnale è *lossy*.

In realtà, l'utilità di questa tecnica, nasce proprio dal fatto che, alla luce della seconda, delle due proprietà appena citate, l'errore introdotto dall'imposizione di una soglia relativamente bassa ai coefficienti a_i e d_i, si manifesta principalmente nella ricostruzione di quella parte di segnale che compete ai coefficienti d_i, ovvero, la parte di dettaglio del segnale; in questa maniera, cioè, si riesce a circoscrivere l'errore di ricostruzione a quella parte di segnale che generalmente è di minore interesse, ricostruendo, invece, in maniera abbastanza fedele, la struttura portante del segnale.

In sintesi, l'algoritmo di compressione basato sull'analisi multirisoluzione, finalizzato alla compressione dei dati prodotti dai rivelatori SDD, appartenenti al *layer* 3 e al *layer* 4 dell'*Inner Tracking System* di ALICE, è così realizzato:

- il segnale proveniente dal rivelatore SDD viene decomposto mediante l'analisi multirisoluzione mono-dimensionale o bi-dimensionale.

- ai coefficienti a_i e d_i della decomposizione, viene applicata una soglia, in maniera tale che, i coefficienti minori, in modulo, di tale soglia, vengano posti a zero.

- le lunghe sequenze di coefficienti nulli che si vengono a creare all'interno della decomposizione, in seguito all'imposizione della soglia, vengono compresse tramite una variante della codifica *Run Length*, si veda Paragrafo 4.4.

- la decomposizione, così compressa, viene memorizzata al posto del segnale originale, con un determinato risparmio in termini di spazio di memorizzazione.

La ricostruzione *lossy* del segnale prodotto dal rivelatore SDD, è ottenuta, a partire dalla decomposizione compressa memorizzata, nella seguente maniera:

- le codifiche dello zero, caratterizzanti la decomposizione compressa, vengono decodificate.

- a partire dalla decomposizione decodificata, viene ricostruito il segnale mediante la sintesi multirisoluzione, mono-dimensionale o bi-dimensionale, a seconda dell'analisi effettuata.

4.1.3 Parametri di configurazione dell'algoritmo multirisolutivo

In base a quanto detto nel Paragrafo precedente, risulta chiaro che, al di là della struttura generale dell'algoritmo multirisolutivo, la variazione di alcuni parametri specifici dell'algoritmo, può determinare prestazioni differenti in termini di compressione e di errore in ricostruzione.
Tali parametri sono:

- la coppia di filtri di decomposizione $\widetilde{H}$ e $\widetilde{G}$, scelta al fine di implementare l'analisi wavelet multirisoluzione.
- la dimensionalità con cui si effettua l'analisi multirisoluzione: monodimensionale o bi-dimensionale.
- il numero di livelli di decomposizione su cui si effettua l'analisi multirisoluzione.
- il valore della soglia applicata ai coefficienti a_i e d_i della decomposizione wavelet ortogonale del segnale.

In particolare, gran parte del tempo occorso nello sviluppo di tale algoritmo multirisolutivo, è stato finalizzato alla ricerca di una configurazione ottimale di questi parametri, ovvero, una configurazione determinante le migliori prestazioni possibili.

4.2 Ottimizzazione dell'algoritmo multirisolutivo

L'ottimizzazione dell'algoritmo multirisolutivo è stata effettuata utilizzando il Wavelet Toolbox di Matlab.
In una prima fase si è individuata la coppia di filtri di decomposizione che, fissato uno specifico valore della soglia, determinasse il numero maggiore di coefficienti a_i e d_i prossimi a zero e, contemporaneamente, determinasse l'errore di ricostruzione minore; in una seconda fase, fissata tale coppia di filtri, si è stabilito quale configurazione, dei restanti tre parametri modificabili, determinasse a sua volta il maggior numero di coefficienti a_i e d_i prossimi a zero e, allo stesso tempo, l'errore di ricostruzione minore.

4.2.1 Il Wavelet Toolbox di Matlab

Il Wavelet Toolbox è una collezione di funzioni Matlab che, attraverso i comandi in linea di Matlab ed una apposita interfaccia grafica, consente lo sviluppo di tecniche wavelet mirate ad essere implementate nell'ambito di problemi reali. In particolare, le funzionalità che si sono rivelate maggiormente utili, in questa fase dello sviluppo dell'algoritmo multirisolutivo, sono: la possibilità di effettuare l'analisi multirisoluzione di un segnale, e la corrispondente sintesi, attraverso una ampia scelta di filtri di decomposizione e di filtri di ricostruzione, la possiblità di effettuarle sia in maniera mono-dimensionale che in maniera bi-dimensionale, la possiblità di effettuarle su un numero di livelli regolabile a piacere e la possibilità di applicare diversi valori di soglia ai coefficienti a_i e d_i.

L'ampia scelta di filtri è dovuta alla vasta gamma di famiglie wavelet attraverso cui il Wavelet Toolbox può implementare l'analisi e la sintesi multirisoluzione, come si vede in Tab. 4.1 ed in Fig. 4.1, Fig. 4.2 e Fig. 4.3.

Famiglia	Nome Identificativo
Haar wavelet	'haar'
Daubechies wavelets	'db'
Symlets	'sym'
Coiflets	'coif'
Biorthogonal wavelets	'bior'
Reverse Biorthogonal wavelets	'rbio'

Tabella 4.1: Famiglie wavelet atte all'analisi multirisoluzione.

In particolare, la famiglia Haar è costituita dalla funzione wavelet $\psi(x)$ e dalla corrispondente funzione di scala $\phi(x)$, già discusse in (2.43); ciascuna famiglia Daubechies, Symlets e Coiflets, invece, è costituita da più di una coppia di funzioni corrispondenti $\psi(x)$ e $\phi(x)$, nella fattispecie, le coppie appartenenti alla famiglia Daubechies sono individuate dai nomi db1, ..., db10, le coppie appartenenti alla famiglia Symlets dai nomi sym2, ..., sym8, mentre le coppie appartenenti alla famiglia Coiflets dai nomi coif1, ..., coif5.
A differenza delle altre, la famiglia Biorthogonal (bior1.1, ..., bior6.8) e la famiglia Reverse Biorthogonal (rbio1.1, ..., rbio6.8), sono costituite da quartetti di funzioni $\psi_1(x)$, $\phi_1(x)$, $\psi_2(x)$ e $\phi_2(x)$, dove, la prima coppia viene utilizzata in fase di decomposizione e la seconda coppia in fase di ricostruzione.

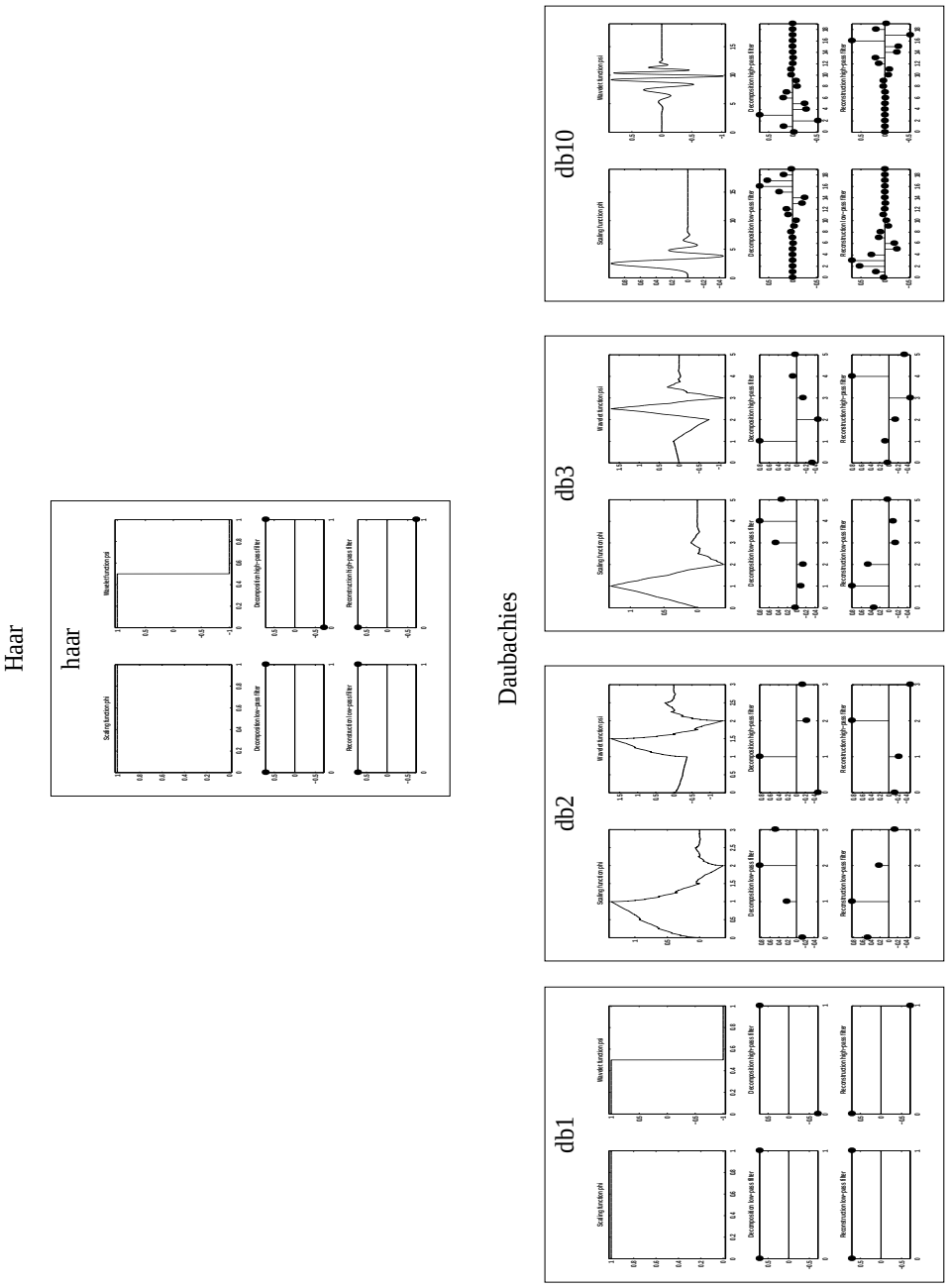


Figura 4.1: Alcune funzioni appartenenti a diverse famiglie wavelet: si noti che db1 è equivalente ad haar.

Figura 4.2: Alcune funzioni appartenenti a diverse famiglie wavelet.

Figura 4.3: Alcune funzioni appartenenti a diverse famiglie wavelet: si noti che bior1.1 ed rbior1.1 sono equivalenti ad haar.

Attraverso una particolare funzione del Toolbox, che richiede come argomento, il nome della coppia di funzioni $\psi(x)$ e $\phi(x)$ con cui si ha intenzione di lavorare—o il nome del quartetto $\psi_1(x)$, $\phi_1(x)$, $\psi_2(x)$ e $\phi_2(x)$, nel caso delle famiglie Biorthogonal e Reverse Biorthogonal—è possibile determinare le risposte all'impulso rappresentanti, rispettivamente, il filtro passa basso $\widetilde{H}$ ed il filtro passa alto $\widetilde{G}$, con cui viene effettuata la decomposizione, ed il filtro passa basso H ed il filtro passa alto G, con cui viene effettuata la ricostruzione, si veda Fig. 4.1, Fig. 4.2 e Fig. 4.3:

```
% determinazione dei filtri per la Haar normalizzata
>>[Htilde,Gtilde,H,G] = wfilters('haar')
Htilde = 0.7071 0.7071
Gtilde = -0.7071 0.7071
H = 0.7071 0.7071
G = 0.7071 -0.7071
```

Determinati i filtri di decomposizione e di ricostruzione, l'analisi e la sintesi multirisoluzione vengono effettuate in maniera del tutto analoga a quanto detto nel Paragrafo 2.4; in particolare, l'analisi viene effettuata mediante operazioni di convoluzione tra il segnale ed i filtri $\widetilde{H}$ e $\widetilde{G}$, seguite da operazioni di sottocampionamento, mentre la sintesi viene effettuata mediante operazioni di sovracampionamento, seguite da operazioni di convoluzione tra il segnale ed i filtri H e G.
Nella fattispecie, l'analisi mono-dimensionale viene effettuata mediante una funzione che richiede come argomenti, nell'ordine, il segnale mono-dimensionale da analizzare, il numero di livelli di decomposizione con cui si vuole effettuare l'analisi, il filtro passa basso di decomposizione $\widetilde{H}$, ed il filtro passa alto di decomposizione $\widetilde{G}$; successivamente, può essere applicata una soglia ai coefficienti della decomposizione, e dai coefficienti così modificati, può essere ricostruito il segnale in maniera *lossy*:

```
% analisi del segnale mono-dimensionale S su N livelli
>>[C,L] = wavedec(S,N,Htilde,Gtilde);
% applicazione della soglia TH ai coefficienti C
>>wthresh(C,'h',TH);
% sintesi lossy del segnale S su N livelli:
% a partire dai coefficienti modificati C
% la sintesi viene effettuata e memorizzata in R
>>R = waverec(C,L,H,G);
```

In maniera del tutto analoga, è possibile analizzare un segnale bi-dimensionale su un numero di livelli a piacere, applicare una soglia ai coefficienti relativi alla sua decomposizione, e ricostruirlo dai coefficienti così modificati, attraverso l'utilizzo di funzioni simili a quelle utilizzate nel caso mono-dimensionale:

```
% analisi del segnale bi-dimensionale S su N livelli
>>[C,L] = wavedec2(S,N,Htilde,Gtilde);
% applicazione della soglia TH ai coefficienti C
>>wthresh(C,'h',TH);
% sintesi del segnale R su N livelli:
% a partire dai coefficienti modificati C
% la sintesi viene effettuata e memorizzata in R
>>R = waverec2(C,L,H,G);
```

Da ultimo, occorre precisare che, durante la fase di sviluppo dell'algoritmo multirisolutivo, si è preferito richiamare le funzioni suddette, ed altre, mediante l'interfaccia grafica, piuttosto che attraverso i comandi in linea: questo ha consentito una notevole agevolazione del lavoro.

4.2.2 Scelta dei filtri

Al fine di individuare i filtri $\widetilde{H}$, $\widetilde{G}$, H e G, che meglio operano con i segnali provenienti dai semi-rivelatori SDD di ALICE, si è proceduto all'analisi di 10 di tali segnali, attraverso le diverse famiglie wavelet supportate dal Wavelet Toolbox, mostrate in Tab.4.1.
Su ciascun segnale originale $\boldsymbol{S}$, interpretato, sia nell'ottica mono-dimensionale (4.1), come si vede in Fig. 4.4, che nell'ottica bi-dimensionale (4.2), come si vede in Fig. 4.5, lo studio è stato condotto nella seguente maniera:

- si è scelta la coppia di funzioni $\psi(x)$ e $\phi(x)$—o il quartetto $\psi_1(x)$, $\phi_1(x)$, $\psi_2(x)$, $\phi_2(x)$—e si sono determinati i filtri corrispondenti $\widetilde{H}$, $\widetilde{G}$, H e G.
- si è analizzato il segnale $\boldsymbol{S}$ attraverso i filtri $\widetilde{H}$ e $\widetilde{G}$ e si sono ottenuti i coefficienti di decomposizione $\boldsymbol{C}$.
- si è applicata una soglia th ai coefficienti $\boldsymbol{C}$, e si sono ottenuti i coefficienti modificati $\boldsymbol{C}_{th}$.
- si è sintetizzato il segnale $\boldsymbol{R}$, attraverso i filtri H e G, a partire dai coefficienti $\boldsymbol{C}_{th}$.

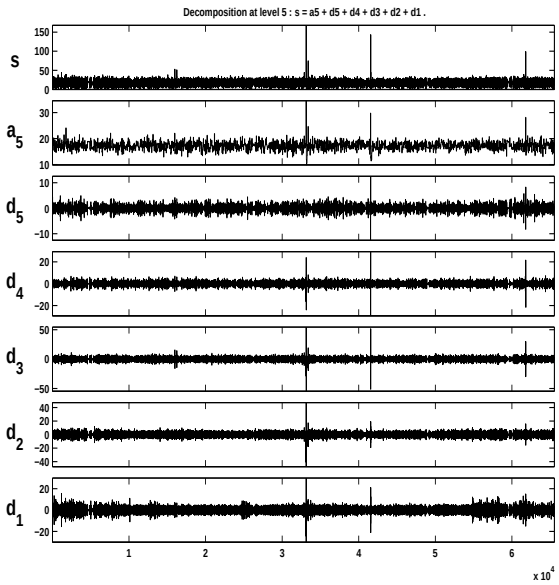


Figura 4.4: Analisi mono-dimensionale su 5 livelli del segnale $\boldsymbol{S}$.

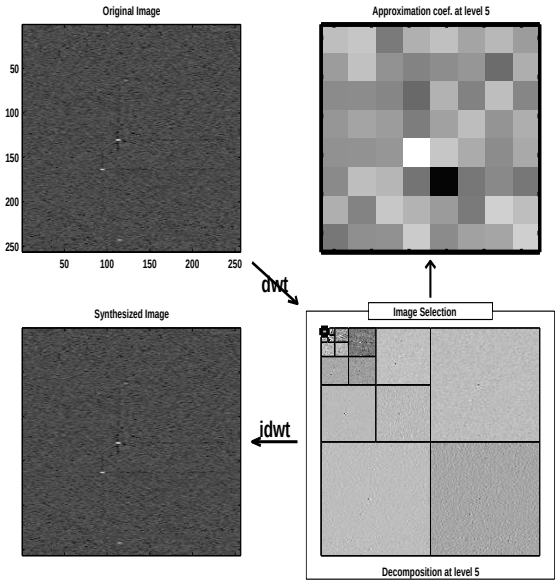


Figura 4.5: Analisi bi-dimensionale su 5 livelli del segnale $\boldsymbol{S}$.

Sia nel caso mono-dimensionale, che in quello bi-dimensionale, le prestazioni relative alla compressione sono state quantificate attraverso la percentuale P del numero di coefficienti nulli in $\boldsymbol{C}_{th}$, mentre, le prestazioni relative all'errore di ricostruzione, sono state quantificate attraverso l'errore quadratico medio E tra il segnale originale $\boldsymbol{S}$ ed il segnale $\boldsymbol{R}$, sintetizzato a partire da $\boldsymbol{C}_{th}$.
In particolare, il numero totale di elementi in $\boldsymbol{C}_{th}$ è pari a 65536, pertanto, nel caso mono-dimensionale, il paramentro P può essere espresso nella seguente maniera:

$$P = \frac{100 \cdot (\text{numero di coefficienti nulli in } \boldsymbol{C}_{th})}{65536} \tag{4.3}$$

Anche il numero totale di elementi in $\boldsymbol{S}$ ed in $\boldsymbol{R}$ è pari a 65536, pertanto, detti s_i e r_i, rispettivamente, l'i-esimo elemento del segnale mono-dimensionale $\boldsymbol{S}$ e l'i-esimo elemento del segnale mono-dimensionale $\boldsymbol{R}$, il parametro E può essere espresso, sempre nel caso mono-dimensionale, nella seguente maniera:

$$E = \sqrt{\frac{1}{65536} \sum_{i=1}^{65536} (s_i - r_i)^2} \tag{4.4}$$

Nel caso bi-dimensionale il calcolo del parametro P rimane tale, mentre, detti $s_{i,j}$ e $r_{i,j}$, rispettivamente, l'i,j-esimo elemento del segnale bi-dimensionale $\boldsymbol{S}$ e l'i,j-esimo elemento del segnale bi-dimensionale $\boldsymbol{R}$, il parametro E può essere espresso nella seguente maniera:

$$E = \sqrt{\frac{1}{65536} \sum_{i=1}^{256} \sum_{j=1}^{256} (s_{i,j} - r_{i,j})^2} \tag{4.5}$$

Si noti che, sia il parametro P, che il parametro E, pur non fornendo dei valori relativi alla compressione ed all'errore, direttamente confrontabili con i risultati ottenuti dall'attuale algoritmo di compressione sviluppato tramite CARLOS, forniscono comunque delle indicazioni fondamentali, circa le prestazioni di ciascun set di filtri utilizzato per l'analisi.
In particolare, il parametro P può essere considerato come indicante la predisposizione dei coefficienti di $\boldsymbol{C}_{th}$ ad essere compressi tramite una codifica *Run Length*; il parametro E, invece, rappresentando l'errore quadratico medio relativo a tutto il segnale $\boldsymbol{R}$, può essere considerato come l'errore commesso nella stima del valore associato a ciascun campione del semi-rivelatore SDD.

Effettuando uno studio sistematico su 10 segnali relativi ad un semi-rivelatore SDD, si sono ottenuti i risultati mostrati nelle tabelle Tab. 4.2-Tab. 4.7; in particolare, in Tab. 4.2 sono mostrati tutti i valori relativi al parametro di compressione P ed al parametro di errore E, ottenuti a seguito di un'analisi su 5 livelli, effettuata tramite i filtri di Haar, sia in maniera mono-dimensionale (1D), che in maniera bi-dimensionale (2D), e con valori della soglia th variabili tra 0,1,2,...,25; nelle restanti tabelle, invece, sono stati evidenziati i soli valori di P ed E relativi ad una soglia th pari a 25, applicata, anche in questo caso, a seguito di una analisi su 5 livelli, sia mono-dimensionale, che bi-dimensionale, ed effettuata, rispettivamente, tramite i filtri appartenenti alla famiglia Daubechies (Tab. 4.3), Symlets (Tab. 4.4), Coiflets (Tab. 4.5), Biorthogonal (Tab. 4.6) e Reverse Biorthogonal (Tab. 4.7); si noti che, dato il carattere orientativo, e non statistico, di questa prima fase dell'analisi, le incertezze ΔP e ΔE sono state indicate esclusivamente in termini dei rispettivi ordini di grandezza.

Una caratterisitica estremamente interessante relativa a tale studio, ed evidenziata in Tab. 4.2, è il progressivo aumento dei valori di P ed E, all'aumentare dei valori di soglia th applicati ai coefficienti di decomposizione $\boldsymbol{C}$.
L'andamento di P è comprensibile, considerando che, applicare la soglia th ai coefficienti di decomposizione $\boldsymbol{C}$, significa azzerare tutti i coefficienti minori, in modulo, di th, pertanto, aumentare il valore di th, significa aumentare l'intervallo di valori $[-th, th]$, entro cui i coefficienti di decomposizione vengono azzerati; in particolare, tanto più l'intervallo $[-th, th]$ è ampio, tanto più è probabile trovare coefficienti aventi valori in esso compresi, conseguentemente, tanto più è probabile avere valori di P elevati.
Per quel che riguarda l'andamento di E, invece, occorre considerare che, azzerare i coefficienti minori, in modulo, di th, significa trasformare l'array dei coefficienti di decomposizione originali $\boldsymbol{C}$, nell'array $\boldsymbol{C}_{th}$ dei coefficienti modificati dall'applicazione della soglia th; in particolare, tanto più la soglia è elevata, tanto più $\boldsymbol{C}_{th}$ si discosta dall'originale $\boldsymbol{C}$, conseguentemente, tanto più la sintesi, a partire da $\boldsymbol{C}_{th}$, è caratterizzata da un errore consistente.
Si noti che, per un valore di th pari a 0, il parametro P vale 9.12, mentre il parametro E vale 1.26 e-14, ovvero, la percentuale di coefficienti nulli in $\boldsymbol{C}_{th}$, e l'errore in fase di sintesi, sono minimi; questo è comprensibile, nel caso del parametro P, considerando che, in assenza dell'applicazione della soglia, gli unici coefficienti nulli sono quelli che la sola analisi multirisoluzione ha reso tali, e quindi, in percentuale, relativamente pochi.

Nel caso del parametro E, invece, l'assenza dell'applicazione della soglia, e quindi la non alterazione dei coefficienti di $\boldsymbol{C}$, garantisce una sintesi del segnale quasi *lossless*, a meno di un errore infinitesimo, dovuto alla finita precisione con cui vengono effettuati i calcoli relativi all'analisi e alla sintesi.

	Haar			
	1D		2D	
Valore della soglia th	P	E	P	E
0	9.12	1.26 e-14	3.68	2.50 e-14
1	24.68	0.27	22.21	0.28
2	40.01	0.63	42.63	0.75
3	58.60	1.64	56.34	1.19
4	67.08	1.71	67.76	1.67
5	75.56	2.09	75.50	2.09
6	79.87	2.38	80.77	2.44
7	83.56	2.68	84.96	2.77
8	86.71	2.99	88.21	3.08
9	88.82	3.23	90.75	3.36
10	90.70	3.48	92.88	3.63
11	92.21	3.72	94.49	3.87
12	93.20	3.89	95.80	4.08
13	94.16	4.07	96.78	4.26
14	94.81	4.21	97.56	4.42
15	95.33	4.34	98.20	4.57
16	95.72	4.44	98.73	4.71
17	96.03	4.54	99.05	4.80
18	96.20	4.60	99.25	4.86
19	96.41	4.67	99.44	4.93
20	96.54	4.72	99.55	4.97
21	96.62	4.76	99.64	5.01
22	96.69	4.79	99.69	5.03
23	96.73	4.81	99.74	5.05
24	96.76	4.83	99.77	5.07
25	96.79	4.85	99.80	5.09

Tabella 4.2: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi effettuata su 5 livelli, utilizzando il set di filtri haar relativo alla famiglia Haar.

	Daubechies			
	1D		2D	
Filtri	P	E	P	E
db1	96.79	4.85	99.80	5.09
db2	96.75	4.82	99.63	5.08
db3	96.73	4.81	99.54	5.07
db4	96.73	4.81	99.48	5.07
db5	96.72	4.81	99.33	5.07
db6	96.71	4.81	99.27	5.07
db7	96.72	4.82	99.20	5.07
db8	96.70	4.81	99.08	5.08
db9	96.69	4.81	98.98	5.09
db10	96.68	4.80	98.98	5.09

Tabella 4.3: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi effettuata su 5 livelli, utilizzando i filtri appartenenti alla famiglia Daubechies ed un valore della soglia th pari a 25; i valori ottenuti con db1 sono equivalenti ai valori ottenuti con haar, in virtù dell'equivalenza dei rispettivi filtri.

	Symlets			
	1D		2D	
Filtri	P	E	P	E
sym2	96.75	4.82	99.63	5.08
sym3	96.73	4.81	99.54	5.07
sym4	96.74	4.82	99.43	5.07
sym5	96.72	4.81	99.38	5.06
sym6	96.73	4.81	99.33	5.07
sym7	96.70	4.80	99.17	5.06
sym8	96.71	4.80	99.11	5.08

Tabella 4.4: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi effettuata su 5 livelli, utilizzando i filtri appartenenti alla famiglia Symlets, ed un valore della soglia th pari a 25.

	Coiflets			
	1D		2D	
Filtri	P	E	P	E
coif1	96.74	4.82	99.51	5.07
coif2	96.72	4.80	98.32	4.75
coif3	96.72	4.81	99.60	5.06
coif4	96.69	4.80	98.62	5.06
coif5	96.68	4.80	98.29	5.05

Tabella 4.5: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi effettuata su 5 livelli, utilizzando i filtri appartenenti alla famiglia Coiflets, ed un valore della soglia th pari a 25.

	Biorthogonal			
	1D		2D	
Filtri	P	E	P	E
bior1.1	96.79	4.85	99.80	5.09
bior1.3	96.68	4.81	99.48	5.07
bior1.5	96.64	4.82	99.25	5.05
bior2.2	96.28	4.71	98.70	4.94
bior2.4	96.28	4.65	98.56	4.92
bior2.6	96.23	4.62	98.27	4.91
bior2.8	96.21	4.63	97.81	4.91
bior3.1	93.41	5.68	94.15	5.58
bior3.3	94.37	4.84	95.43	5.01
bior3.5	94.70	4.65	96.60	5.10
bior3.7	94.81	4.59	95.13	4.85
bior3.9	94.88	4.56	94.13	4.85
bior4.4	96.75	4.82	99.39	5.07
bior5.5	96.78	4.88	99.46	5.10
bior6.8	96.68	4.79	98.95	5.04

Tabella 4.6: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi su 5 livelli, utilizzando i filtri appartenenti alla famiglia Biorthogonal ed un valore della soglia th pari a 25; i valori ottenuti con bior1.1 sono equivalenti ai valori ottenuti con haar, in virtù dell'equivalenza dei rispettivi filtri.

	Reverse Biorthogonal			
	1D		2D	
Filtri	P	E	P	E
rbio1.1	96.79	4.85	99.80	5.09
rbio1.3	96.77	4.85	99.57	5.08
rbio1.5	96.75	4.86	99.39	5.06
rbio2.2	96.78	4.92	96.89	4.58
rbio2.4	96.79	4.88	99.47	5.12
rbio2.6	96.77	4.87	99.32	5.11
rbio2.8	96.78	4.88	99.18	5.12
rbio3.1	96.38	8.67	98.76	11.29
rbio3.3	96.72	5.14	99.29	5.39
rbio3.5	96.76	4.95	99.28	5.18
rbio3.7	96.76	4.92	99.09	5.18
rbio3.9	96.74	4.91	98.97	5.20
rbio4.4	96.68	4.80	99.29	5.06
rbio5.5	93.32	4.63	98.56	4.92
rbio6.8	96.71	4.81	99.10	5.08

Tabella 4.7: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi su 5 livelli, utilizzando i filtri appartenenti alla famiglia Rev. Biorthogonal ed un valore della soglia th pari a 25; i valori ottenuti con rbio1.1 sono equivalenti ai valori ottenuti con haar, in virtù dell'equivalenza dei rispettivi filtri.

Anche se in Tab. 4.3, Tab. 4.4, Tab. 4.5, Tab. 4.6 e Tab. 4.7 non è stata messa in evidenza la dipendenza tra il parametro P, il parametro E, e la variazione del valore di soglia th, l'aumentare dei valori di P ed E, conseguente all'aumentare del valore di th, è una caratteristica comune a tutte le famiglie. Tuttavia, al di là di questo andamento comune, alcune famiglie si prestano meglio ad essere utilizzate in fase di compressione, rispetto ad altre, ed in particolare, alcuni set di filtri, appartenenti ad una specifica famiglia, sono più efficaci di altri; confrontando, ad esempio, i valori ottenuti per $th = 25$, nei vari casi, appare evidente che il set di filtri associato alla famiglia Haar, è quello che meglio si adatta alla compressione di tali segnali: in particolare, con $P = 96.79$ ed $E = 4.85$, nel il caso mono-dimensionale, e con $P = 99.80$ ed $E = 5.09$, nel caso bi-dimensionale, è il set di filtri determinante la percentuale di coefficienti nulli maggiore, a fronte di un errore relativamente contenuto.

Famiglia	Nome del set di filtri	Lunghezza filtri
Haar	haar	2
Daubechies	dbN	2N
Symlets	symN	2N
Coiflets	coifN	6N
Biorthogonal	bior1.1 biorN1.N2, N1≠1,N2≠1	2 max(2N1,2N2)+2
Reverse Biorthogonal	rbio1.1 rbioN1.N2, N1≠1,N2≠1	2 max(2N1,2N2)+2

Tabella 4.8: Lunghezza dei filtri appartenenti a ciascuna famiglia.

A supportare la scelta dei filtri di Haar, oltre i risultati sperimentali appena discussi, contribuisce anche una considerazione relativa alla lunghezza dei suoi filtri $\widetilde{H}$, $\widetilde{G}$, H e G, ovvero, una considerazione relativa al numero di coefficienti caratterizzanti le rispettive risposte all'impulso.
In particolare, come mostrato in Tab. 4.8, i filtri appartenenti alla famiglia Haar, con 2 soli coefficienti ciascuno, sono quelli aventi la lunghezza minore, insieme, chiaramente, agli equivalenti set di filtri db1, bior1.1 e rbio1.1; poiché l'analisi e la sintesi multirisoluzione vertono su successive convoluzioni tra il segnale da analizzare, o sintetizzare, ed i rispettivi filtri, questo basso numero di coefficienti, si traduce in una maggiore velocità dell'analisi e della sintesi, e quindi, in generale, in una maggiore velocità dell'algoritmo di compressione multirisolutivo.

4.2.3 Scelta della dimensionalità, del livello e della soglia

Individuato in quello di Haar, il set di filtri determinante le prestazioni complessivamente migliori, si è proceduto alla valutazione dell'influenza che hanno, sui parametri P ed E, la dimensionalità (1D o 2D) attraverso cui si effettua l'analisi e la sintesi del segnale, il numero di livelli di decomposizione (1,2,...,16 nel caso mono-dimensionale o 1,2,...,8 nel caso bi-dimensionale), ed il valore della soglia th applicata ai coefficienti di decomposizione.

A tale scopo, come mostrato in Tab. 4.9 ed in Tab. 4.10, si è proceduto allo studio dei 10 segnali precedenti, sia attraverso un'analisi ed una sintesi mono-dimensionale, che attraverso un'analisi ed una sintesi bi-dimensionale; in particolare, ciascuna di queste è stata effettuata su 1,3 e 5 livelli di decomposizione, e per ciascuna di queste si è calcolato il parametro P ed il parametro E, al variare del valore della soglia th tra 0,1,2,...,25.
Quanto emerge dai risultati mostrati in Tab. 4.9 ed in Tab. 4.10, è che, a parità di livelli di decomposizione, l'analisi bi-dimensionale determina una percentuale P di coefficienti di decomposizione nulli, maggiore, rispetto al caso mono-dimensionale; inevitabilmente, però, determina anche valori del parametro di errore E maggiori.
Ad esempio, confrontando i valori di P ed E per valori della soglia th pari a 25, si nota che, l'analisi mono-dimensionale effettuata su 1 livello, determina $P = 50.01$ ed $E = 1.85$, contro i valori $P = 74.96$ ed $E = 3.96$, ottenuti nel corrispondente caso bi-dimensionale; analogamente, l'analisi mono-dimensionale effettuata su 3 livelli, determina $P = 87.45$ ed $E = 4.18$, contro i valori $P = 98.35$ ed $E = 5.00$, ottenuti nel corrispondente caso bi-dimensionale, mentre, l'analisi mono-dimensionale effettuata su 5 livelli, determina $P = 96.79$ ed $E = 4.85$, contro i valori $P = 99.80$ ed $E = 5.09$, del caso bi-dimensionale.
Allo stesso tempo, fissata la dimensionalità dell'analisi e della sintesi al caso mono-dimensionale, o al caso bi-dimensionale, appare chiaro che, un aumento del numero di livelli di decomposizione, determina un aumento generalizzato dei valori del parametro P e dei valori del parametro E.
Ad esempio, confrontando i valori in Tab. 4.9 ottenuti per valori della soglia th pari a 25, si nota che, l'analisi mono-dimensionale effettuata su 1 livello, determina $P = 50.01$ ed $E = 1.85$, l'analisi effettuata su 2 livelli, determina $P = 87.45$ ed $E = 4.18$, mentre l'analisi effettuata su 3 livelli, determina $P = 96.79$ ed $E = 4.85$; analogamente, confrontando i valori in Tab. 4.10 ottenuti per valori della soglia th pari a 25, si nota che, l'analisi bi-dimensionale effettuata su 1 livello, determina $P = 74.96$ ed $E = 3.96$, l'analisi effettuata su 2 livelli, determina $P = 98.35$ ed $E = 5.00$, mentre l'analisi effettuata su 3 livelli, determina $P = 99.80$ ed $E = 5.09$.

In sintesi, a parità di soglia applicata, l'analisi multirisoluzione che determina la percentuale P di coefficienti di decomposizione nulli, maggiore, a fronte di un errore E relativamente contenuto, è l'analisi multirisoluzione bi-dimensionale, effettuata tramite i filtri appartenenti alla famiglia Haar, e sviluppata sul numero massimo di livelli di decomposizione.

	Haar					
	1D					
	1 livello		3 livelli		5 livelli	
Valore della soglia th	P	E	P	E	P	E
0	7.78	3.02 e-15	9.05	7.11 e-15	9.12	1.26 e-14
1	17.51	0.22	23.67	0.26	24.68	0.27
2	31.23	0.65	38.11	0.62	40.01	0.63
3	40.09	1.01	55.81	1.21	58.60	1.64
4	44.28	1.25	63.48	1.56	67.08	1.71
5	47.84	1.52	71.20	2.00	75.56	2.09
6	48.78	1.61	74.80	2.26	79.87	2.38
7	49.31	1.68	77.81	2.52	83.56	2.68
8	49.71	1.74	80.38	2.79	86.71	2.99
9	49.78	1.76	82.02	2.99	88.82	3.23
10	49.87	1.78	83.41	3.19	90.70	3.48
11	49.91	1.79	84.50	3.38	92.21	3.72
12	49.94	1.80	85.17	3.50	93.20	3.89
13	49.97	1.81	85.81	3.64	94.16	4.07
14	49.98	1.82	86.25	3.75	94.81	4.21
15	49.98	1.83	86.60	3.84	95.33	4.34
16	49.99	1.83	86.85	3.92	95.72	4.44
17	50.00	1.84	87.02	3.98	96.03	4.54
18	50.00	1.84	87.12	4.02	96.20	4.60
19	50.00	1.84	87.24	4.07	96.41	4.67
20	50.00	1.84	87.32	4.10	96.54	4.72
21	50.01	1.84	87.36	4.12	96.62	4.76
22	50.01	1.84	87.40	4.14	96.69	4.79
23	50.01	1.84	87.42	4.16	96.73	4.81
24	50.01	1.85	87.43	4.17	96.76	4.83
25	50.01	1.85	87.45	4.18	96.79	4.85

Tabella 4.9: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi mono-dimensionale effettuata su un numero di livelli pari a 1,3,5, utilizzando il set di filtri haar relativo alla famiglia Haar.

	Haar					
	2D					
	1 livello		3 livelli		5 livelli	
Valore della soglia *th*	*P*	*E*	*P*	*E*	*P*	*E*
0	3.54	5.32 e-15	3.67	1.5 e-14	3.68	2.50 e-14
1	18.90	0.26	22.06	0.28	22.21	0.28
2	36.05	0.69	42.33	0.74	42.63	0.75
3	46.42	1.07	55.90	1.19	56.34	1.19
4	55.25	1.47	67.15	1.66	67.76	1.67
5	60.69	1.80	74.78	2.07	75.50	2.09
6	64.01	2.06	79.95	2.42	80.77	2.44
7	66.46	2.30	84.03	2.75	84.96	2.77
8	68.30	2.51	87.18	3.05	88.21	3.08
9	69.73	2.70	89.64	3.33	90.75	3.36
10	70.95	2.90	91.72	3.59	92.88	3.63
11	71.87	3.06	93.25	3.82	94.49	3.87
12	72.63	3.22	94.51	4.03	95.80	4.08
13	73.20	3.35	95.46	4.21	96.78	4.26
14	73.65	3.47	96.21	4.36	97.56	4.42
15	74.06	3.59	96.84	4.51	98.20	4.57
16	74.38	3.69	97.34	4.64	98.73	4.71
17	74.53	3.75	97.63	4.72	99.05	4.80
18	74.65	3.80	97.82	4.79	99.25	4.86
19	74.76	3.85	98.01	4.85	99.44	4.93
20	74.82	3.87	98.11	4.89	99.55	4.97
21	74.87	3.90	98.20	4.93	99.64	5.01
22	74.91	3.92	98.25	4.95	99.69	5.03
23	74.93	3.94	98.29	4.97	99.74	5.05
24	74.94	3.95	98.32	4.99	99.77	5.07
25	74.96	3.96	98.35	5.00	99.80	5.09

Tabella 4.10: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$): analisi bi-dimensionale effettuata su un numero di livelli pari a 1,3,5, utilizzando il set di filtri haar relativo alla famiglia Haar.

Per quanto riguarda la soglia th, invece, come già notato nel Paragrafo 4.2.2, sia ha che, ad un aumento del suo valore, corrisponde un aumento di entrambi i valori di P ed E, viceversa, ad una sua diminuzione, corrisponde una diminuzione dei due; in particolare, è ragionevole pensare che, al fine di ottenere livelli di compressione estremamente elevati, dell'ordine di $c \leq 46 \times 10^{-3}$, occorra applicare un valore di soglia abbastanza elevato, tale da azzerare il numero maggiore possibile di coefficienti di decomposizione.

4.3 Individuazione di una possibile architettura

Una influenza consistente sulla percentuale P di coefficienti nulli, e sull'errore quadratico medio E di ricostruzione, è sicuramente esercitata dalla specifica architettura della *word* con cui vengono sviluppati i calcoli relativi all'analisi ed alla sintesi del segnale, e nella fattispecie, dalla precisione associata a tale architettura: si pensi, ad esempio, ai casi discussi in precedenza, in cui, nonostante un valore nullo della soglia th, e quindi la possibilità di una sintesi *lossless* del segnale, la precisione finita, benché elevata, con cui vengono condotti i calcoli su un processore PentiumII, determina un valore dell'errore E, sebbene infinitesimo, comunque diverso da 0.

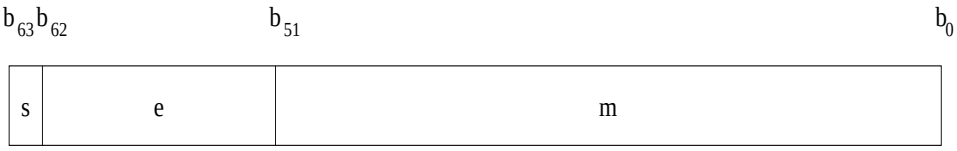
Al fine di quantificare tale influenza, ed al fine di avere una indicazione precisa sull'architettura attraverso cui sviluppare l'algoritmo multirisolutivo, si è implementata l'analisi e la sintesi dei 10 segnali precedenti, mediante Simulink, un pacchetto software di Matlab atto allo sviluppo di sistemi complessi, e mediante una sua specifica libreria, il Fixed-Point Blockset, che consente di simulare il comportamento dei sistemi sviluppati attraverso Simulink, al variare di diverse architetture, sia in virgola mobile che in virgola fissa.

4.3.1 Simulink ed il Fixed-Point Blockset

Simulink [17] è un pacchetto software che consente di sviluppare, simulare ed analizzare sistemi anche complessi, in maniera estremamente intuitiva.
La fase di sviluppo è agevolata da una interfaccia grafica, che consente di costruire i vari sistemi sotto forma di diagrammi a blocchi, semplicemente utilizzando il mouse; in particolare, Simulink è fornito di numerose librerie costituite da blocchi di base predefiniti, quali, la libreria *Sources*, che contiene

blocchi generanti vari tipi di segnale, la libreria *Sinks*, che contiene blocchi visualizzanti vari tipi di segnali e risultati numerici, la libreria *Math*, che contiene blocchi rappresentanti le principali funzioni matematiche, ed altre.
I sistemi vengono sviluppati gerarchicamente, in maniera tale che, i blocchi di base, appartenenti alle varie librerie, possano essere inglobati in blocchi di più alto livello costruiti ad arte dall'utente, ed a loro volta, questi, possano essere inglobati in blocchi di livello superiore; questo approccio consente, sia una visione d'insieme del sistema, individuata dai blocchi di più alto livello, che una visione particolareggiata del sistema, individuata dai singoli blocchi di base costituenti il sistema.

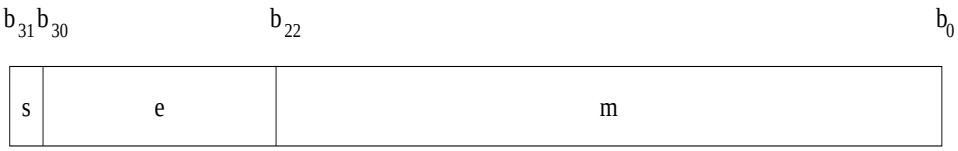
Il Fixed-Point Blockset [18] è una libreria esterna di Simulink che può essere utilizzata in aggiunta alle librerie predefinte, ed è costituita da blocchi che effettuano varie operazioni tra segnali, quali la somma, la moltiplicazione, la convoluzione, ed altre, simulando vari tipi di architetture, sia in virgola mobile, che in virgola fissa; in particolare, la grande utilità di questa libreria, è che consente di quantificare gli effetti di una specifica architettura, sull'algoritmo sviluppato, a livello software, senza dover ricorrere necessariamente ad una implementazione hardware dell'algorimo.
Tra le varie architetture in virgola mobile e fissa, gestite dal Fixed-Point Blockset, quelle studiate in relazione all'analisi ed alla sintesi multirisoluzione sono: l'architettura standard IEEE 754 in virgola mobile a doppia precisione, l'architettura standard IEEE 754 in virgola mobile a singola precisione, e l'architettura in virgola fissa frazionale.
Lo standard IEEE 754 è uno degli standard più diffusi, ed è utilizzato nella maggior parte dei processori a virgola mobile e dei co-processori aritmetici.
Nell'architettura in virgola mobile in doppia precisione, tale standard prevede una *word* di 64 bit, di cui, 1 bit indicante il segno s, 11 bit indicanti l'esponente e, e 52 bit indicanti la mantissa m:



La relazione che intercorre tra la rappresentazione binaria e quella decimale è la seguente:

$$\text{valore decimale} = (-1)^s \cdot \left(2^{e-1023}\right)(1.m)\,, \quad 0 < e < 2047 \tag{4.6}$$

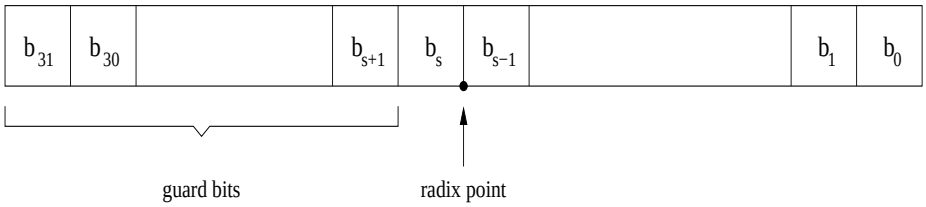
Nell'architettura in virgola mobile in singola precisione, tale standard prevede una *word* di 32 bit, di cui, 1 bit indicante il segno s, 8 bit indicanti l'esponente e, e 23 bit indicanti la mantissa m:



In questo caso, la relazione che intercorre tra la rappresentazione binaria e quella decimale è la seguente:

$$\text{valore decimale} = (-1)^s \cdot \left(2^{e-127}\right)(1.m)\,, \quad 0 < e < 255 \tag{4.7}$$

Per quanto riguarda, invece, l'architettura in virgola fissa frazionale, fissato il punto decimale (*radix point*) tra due dei 32 bit della *word*, questo determina un certo numero di bit ($b_0 - b_{s-1}$) alla sua destra, indicanti la parte frazionaria del numero, un bit (b_s) alla sua sinistra, indicante il segno del numero, ed un certo numero di *guard bits* ($b_{s+1} - b_{31}$), alla sinistra del bit di segno, indicanti la parte intera del numero:



Si noti che, l'architettura standard IEEE 754 in virgola mobile a doppia precisione, ha una precisione dell'ordine di $2^{-52} \approx 10^{-16}$, l'architettura standard IEEE 754 in virgola mobile a singola precisione, ha una precisione dell'ordine di $2^{-23} \approx 10^{-7}$, mentre l'architettura in virgola fissa frazionale, ha una precisione pari a 2^{-s}, ovvero una precisione che dipende dal numero di bit utilizzati per indicare la parte frazionaria del numero, e quindi, che dipende dalla posizione del *radix point* all'interno della *word*; in particolare, alla luce di questa ultima considerazione, lo studio relativo all'influenza dell'architettura in virgola fissa frazionale, è stato effettuato al variare del *radix point*, in differenti posizioni all'interno della *word* a 32 bit.

4.3.2 Scelta dell'architettura

Al fine di stimare l'influenza di ciascuna architettura sui calcoli, si sono implementate, attraverso Simulink, un'analisi ed una sintesi mono-dimensionali, sviluppate attraverso i filtri appartenenti alla famiglia Haar, ed effettuate sul numero massimo di livelli di decomposizione possibili, ovvero 16.
La necessità di effettuare tale studio su una configurazione differente, da quella adottata precedentemente, ovvero un'analisi ed una sintesi bi-dimensionali, sviluppate attraverso i filtri di Haar, ed effettuate su 8 livelli di decomposizione, è resa tale dalla estrema difficoltà che caratterizza l'implementazione dell'analisi e della sintesi bi-dimensionali attraverso Simulink.
In questo senso, lo studio effettuato sull'influenza delle tre architetture, relativamente ai valori di P ed E, non può essere preso come totalmente indicativo per la configurazione bi-dimensionale adottata precedentemente, tuttavia, data la medesima natura delle operazioni effettuate, sia nel caso mono-dimensionale, che in quello bi-dimensionale, tale studio può essere considerato, comunque, come una stima abbastanza attendibile.

L'implementazione, attraverso Simulink, dell'analisi e della sintesi multirisoluzione, verte, sostanzialmente, sui due blocchi esterni mostrati in Fig.4.6: il blocco di sinistra effettua l'analisi del segnale $\boldsymbol{S}$, in maniera mono-dimensionale, su 16 livelli, utilizzando i filtri di Haar, mentre, il blocco di destra, applica una soglia ai coefficienti di decomposizione, ed effettua, a partire dai coefficienti così modificati, la sintesi del segnale $\boldsymbol{R}$ in maniera mono-dimensionale, su 16 livelli, ed utilizzando i filtri di Haar.
Il blocco di analisi, in particolare, è stato implementato sotto forma di una cascata sviluppata su 16 livelli, si veda Fig. 4.7, costituita da operatori di filtro passa alto (Hi_Dec Filter), operatori di filtro passa basso (Low_Dec Filter), ed operatori di sottocampionamento (Downsample); nella fattispecie, gli operatori Hi_Dec Filter effettuano la convoluzione tra il segnale in essi entrante, ed il filtro di decomposizione passa alto di Haar, gli operatori Low_Dec Filter effettuano la convoluzione tra il segnale in essi entrante ed il filtro di decomposizione passa basso di Haar, mentre gli operatori Downsample effettuano, semplicemente, il sottocampionamento del segnale in essi entrante.
Il blocco di applicazione della soglia e di sintesi, invece, è stato implementato sotto forma di tre sotto-blocchi, si veda Fig. 4.8: il sotto-blocco di sinistra, unicamente preposto all'applicazione della soglia, il sotto-blocco di destra, unicamente preposto alla sintesi del segnale, ed il sotto-blocco centrale, denominato

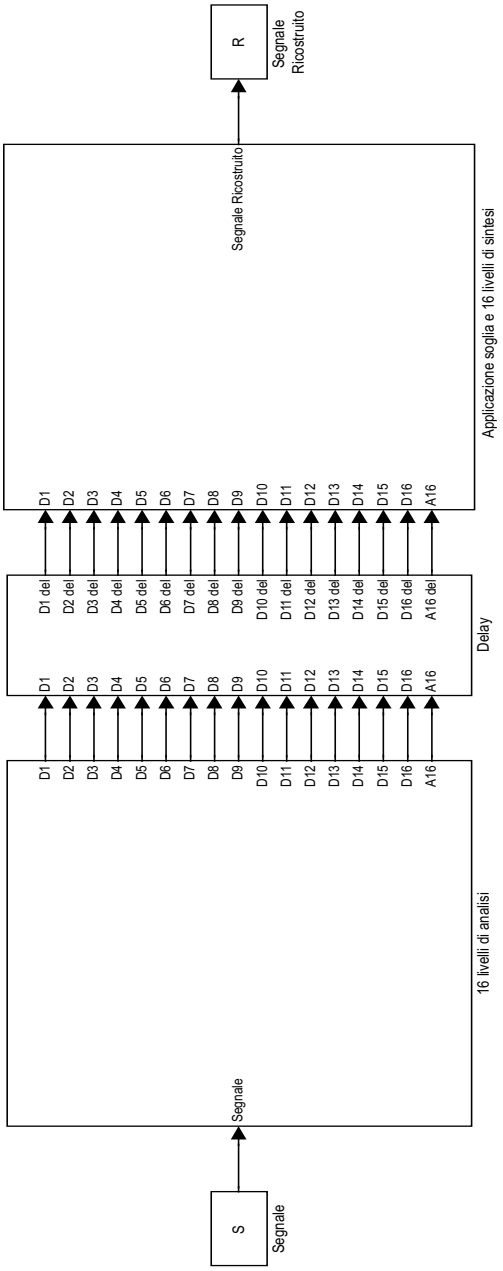


Figura 4.6: Blocchi Simulink sviluppati: a sinistra il blocco di analisi, al centro il blocco di ritardo e destra il blocco di applicazione della soglia e di sintesi.

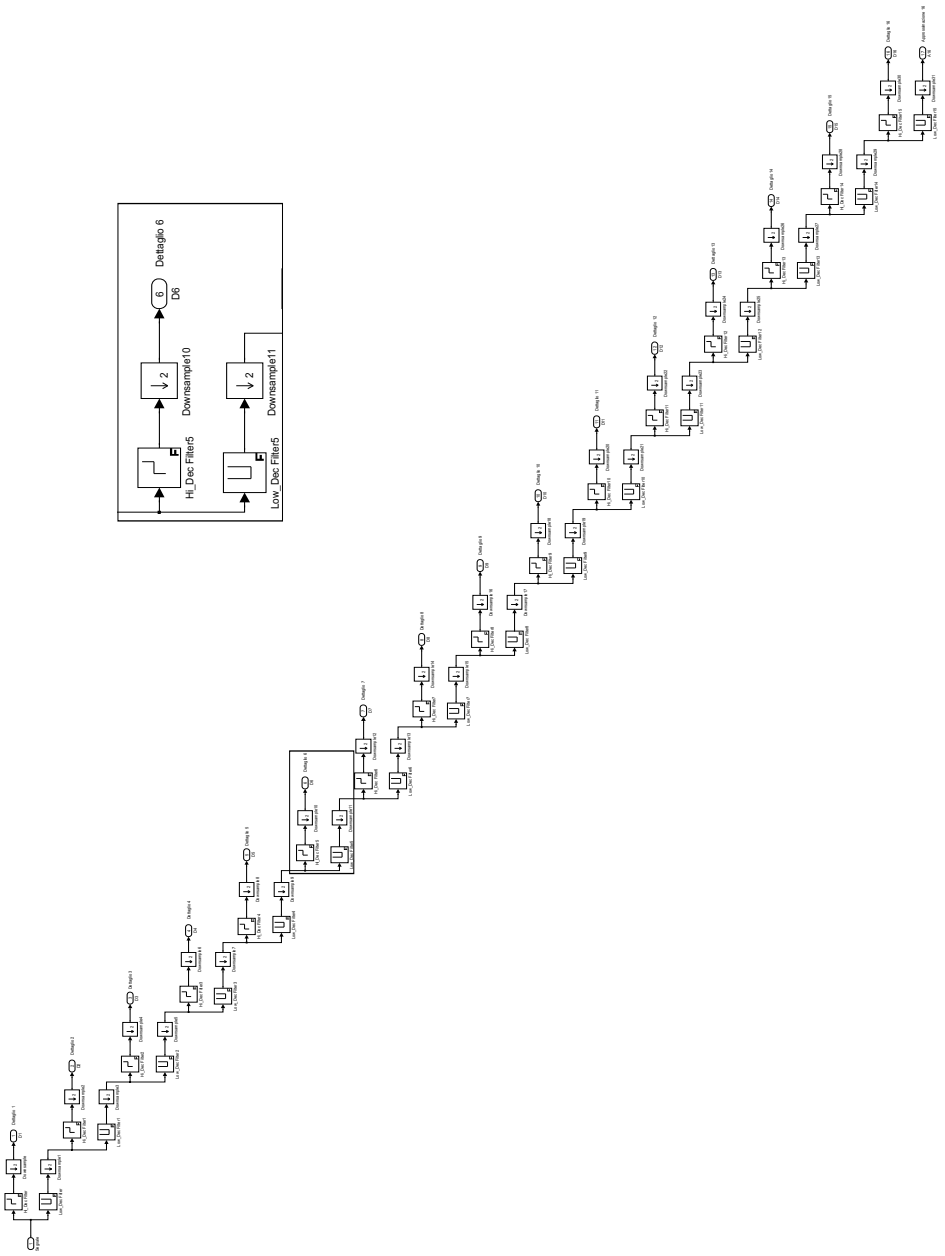


Figura 4.7: Particolare del blocco di analisi sviluppato.

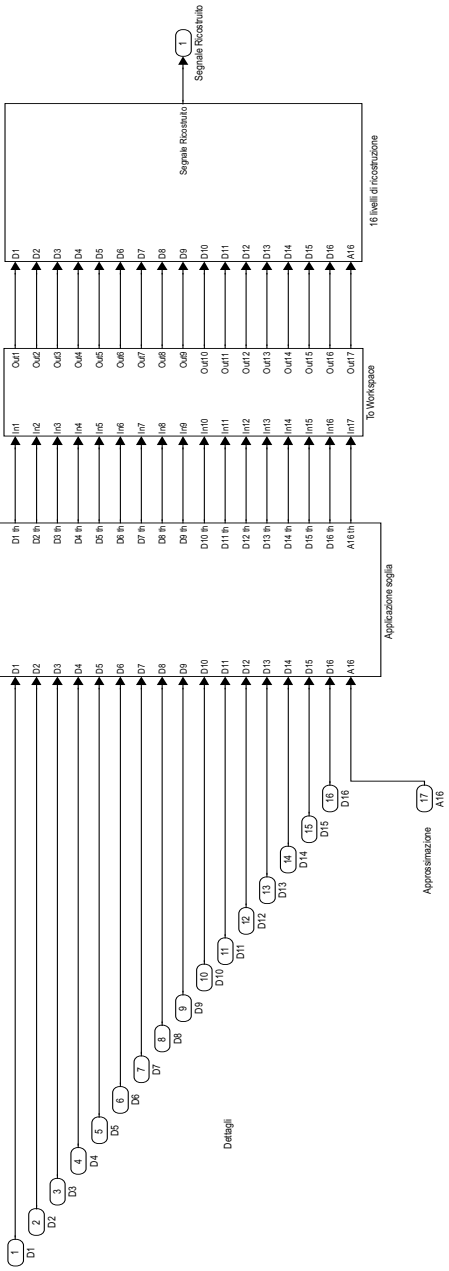


Figura 4.8: Particolare del blocco di applicazione della soglia e di sintesi sviluppato.

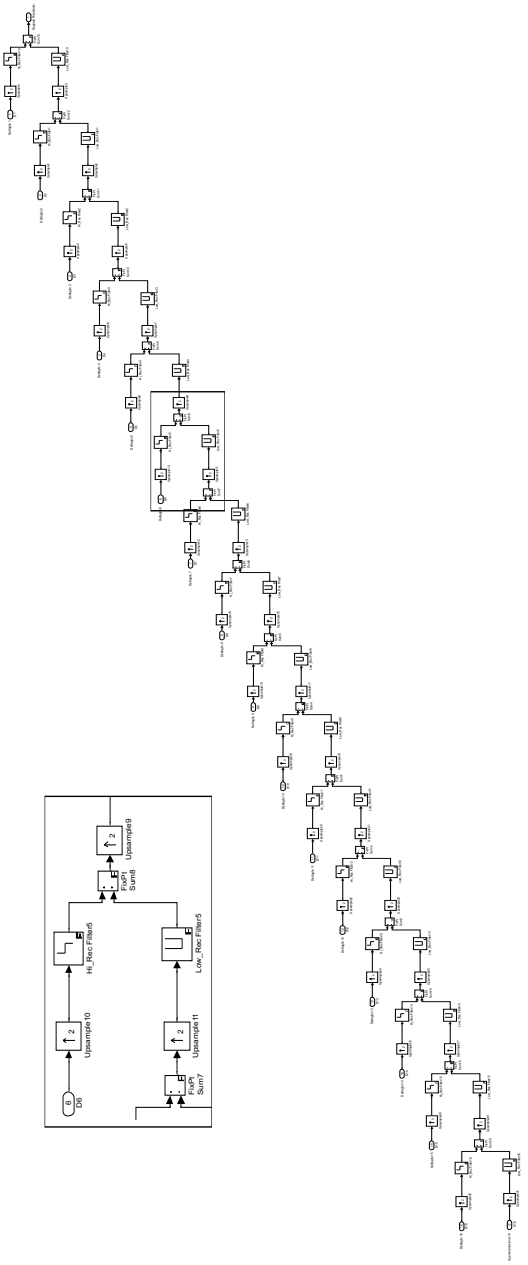


Figura 4.9: Particolare del blocco di sintesi sviluppato.

To Workspace, preposto alla memorizzazione dei coefficienti di decomposizione, così come modificati dall'applicazione della soglia, al fine di un successivo calcolo della percentuale P dei coefficienti nulli.
In particolare, il sotto-blocco di applicazione della soglia, applica una soglia th, da definirsi, a ciascun coefficiente di decomposizione proveniente dal blocco di analisi, in maniera tale che, tutti i coefficienti minori, in modulo, del valore th vengano posti a 0; il sotto-blocco di sintesi, invece, è stato implementato, in analogia con il blocco di analisi, sotto forma di una cascata sviluppata su 16 livelli, si veda Fig. 4.9, costituita da operatori Hi_Rec Filter, che effettuano la convoluzione tra il segnale in essi entrante, ed il filtro di ricostruzione passa alto di Haar, operatori Low_Rec Filter, che effettuano la convoluzione tra il segnale in essi entrante, ed il filtro di ricostruzione passa basso di Haar, operatori FixPt Sum, che effettuano la somma tra i segnali filtrati, ed operatori Upsample, che effettuano il sovracampionamento dei segnali in essi entranti.
Da ultimo, il blocco Delay mostrato in Fig. 4.6, è il blocco che garantisce che la sintesi del segnale parta solo quando tutti i coefficienti di dettaglio, ed il coefficiente di approssimazione, sono stati definitivamente calcolati dal blocco di analisi; in particolare, l'ingresso dei coefficienti di ciascun livello nel blocco di applicazione della soglia e di sintesi, è ritardato di un tempo che è pari al tempo intercorso tra il calcolo dei coefficienti di quel particolare livello, ed il calcolo dei coefficienti del livello 16.
Si noti che, mentre il blocco di analisi, il blocco Delay, il blocco di sintesi, ed i suoi sotto-blocchi, sono stati appositamente sviluppati alla luce del sistema da simulare, i blocchi più piccoli, come i blocchi di filtro, i blocchi di somma, i blocchi di sottocampionamento ed i blocchi di sovracampionamento, sono blocchi base appartenenti alla libreria Fixed-Point Blockset.

Effettuando l'analisi e la sintesi dei 10 eventi precedenti con un valore della soglia th fissato a 25, ed in particolare, simulando per ciascuno di questi, le tre architetture discusse nel Paragrafo 4.3.1, si sono ottenuti i valori mostrati in Tab. 4.11; si noti che, per comodità di notazione, si è indicata l'architettura standard IEEE 754 in virgola mobile a doppia precisione, come *ieee754doub*, l'architettura standard IEEE 754 in virgola mobile a singola precisione, come *ieee754sing*, e l'architettura in virgola fissa frazionale, come *fixed(s)*, dove s è il numero di bit rappresentanti la parte frazionale del numero.
Ciò che emerge dalle simulazioni effettuate attraverso Simulink, è una prevedibile dipendenza dei valori di P ed E, rispetto alla precisione con cui vengono condotti i calcoli relativi all'analisi ed alla sintesi del segnale; in particolare,

Architettura	Precisione	P	E
ieee754doub	2^{-52}	99.88	5.07
ieee754sing	2^{-23}	99.88	5.11
fixed(18)	2^{-18}	99.88	5.11
fixed(15)	2^{-15}	99.88	5.11
fixed(12)	2^{-12}	99.88	5.11
fixed(9)	2^{-9}	99.88	5.11
fixed(7)	2^{-7}	99.87	6.04
fixed(5)	2^{-5}	99.81	12.75
fixed(3)	2^{-3}	99.52	89.09

Tabella 4.11: Valori medi di P ed E su 10 segnali ($\Delta P \approx \Delta E \approx 0.01$), ottenuti in seguito alle simulazioni tramite Simulink.

prendendo a riferimento i valori di P ed E meno influenzati dalla finita precisione con cui vengono condotti i calcoli, e cioè i valori relativi all'architettura *ieee754doub*, si nota, nei casi *ieee754sing*, *fixed(18)*, *fixed(15)*, *fixed(12)* e *fixed(9)*, un lieve aumento dell'errore E, a fronte di un medesimo valore di P, mentre, nei casi *fixed(7)*, *fixed(5)* e *fixed(3)*, la discrepanza rispetto ai valori ottenuti nel caso *ieee754doub*, va aumentando fortemente.
I risultati ottenuti, quindi, sembrano indicare, come soluzione architetturale più idonea all'analisi e la sintesi multirisoluzione, una soluzione che garantisca un certo grado di precisione, come i casi *ieee754sing*, *fixed(18)*, *fixed(15)*, *fixed(12)* e *fixed(9)*, senza richiedere, tuttavia, in maniera imprescindibile, gli elevati livelli di precisione forniti dal caso *ieee754doub*.

4.4 Prestazioni dell'algoritmo multirisolutivo

Alla luce delle considerazioni emerse nel Paragrafo 4.2, e nel Paragrafo 4.3, si è sviluppata una subroutine FORTRAN implementante un'analisi ed una sintesi bi-dimensionali, sviluppate attraverso i filtri di Haar ed effettuate su 8 livelli di decomposizione, e la si è mandata in esecuzione su un processore SPARC5 basato su un'architettura IEEE 754 in virgola mobile a 32 bit; in particolare, richiamando alcune funzioni grafiche della libreria HBOOK di PAW, si è tentato un confronto diretto tra le prestazioni dell'algoritmo di compressione multirisolutivo, sviluppato in questo lavoro di tesi, e quelle dell'algoritmo di compressione sviluppato presso l'INFN di Torino.

La subroutine FORTRAN sviluppata, è sostanzialmente divisa in due parti, una tesa alla valutazione delle prestazioni dell'algoritmo in termini di compressione, l'altra tesa alla valutazione dell'errore specifico commesso sui *cluster* di carica appartenenti ai segnali sintetizzati.

La prima parte della subroutine effettua l'analisi, l'applicazione della soglia th, e la sintesi, su segnali caratterizzati, ognuno, da più *cluster* di carica.
A seguito dell'analisi e dell'applicazione della soglia th, per ciascuno di questi segnali viene calcolato il parametro di compressione $c = \frac{\text{n}^\text{o} \text{ bit in uscita}}{\text{n}^\text{o} \text{ bit in ingresso}}$, assumendo che ogni coefficiente di decomposizione non nullo venga codificato attraverso due *word* a 32 bit, una rappresentante il numero di coefficienti nulli che intercorrono tra questo, ed il primo coefficiente di decomposizione non nullo ad esso antecedente, l'altra rappresentante il valore del coefficiente stesso; in questa maniera, il numero di bit in ingresso all'algoritmo di compressione, è dato dal numero di campioni per ciascun segnale analizzato, moltiplicato per il numero di bit con cui ciascun campione è digitalizzato, ovvero $65536 \times 8 = 524288$ bit, mentre, il numero di bit in uscita dall'algoritmo di compressione, è dato dal numero di coefficienti di decomposizione non nulli, moltiplicato per i $32 + 32 = 64$ bit con cui ciascun coefficiente è digitalizzato.
Al termine dell'analisi di tutti i segnali a disposizione, vengono graficati, sotto forma di istogrammi PAW, i risultati relativi alla compressione c ottenuti su tutti i segnali analizzati, ed i risultati relativi alla compressione c ottenuti sui soli segnali che determinano valori di c inferiori od uguali a 30×10^{-3} che, come discusso nel Paragrafo 3.3.3, è il valore di c ottenuto mediante l'algoritmo sviluppato presso l'INFN di Torino.

La seconda parte della subroutine, invece, effettua l'analisi, l'applicazione della soglia, e la sintesi, su segnali caratterizzati, ognuno, da un solo *cluster*.
A seguito dell'analisi, dell'applicazione della soglia th, e della sintesi, per ciascuno di questi segnali viene calcolata la differenza tra le coordinate del centroide del *cluster* originale e quelle del *cluster* ricostruito, e viene calcolata la differenza percentuale tra la carica del *cluster* originale e la carica del *cluster* ricostruito.
Al termine dell'analisi di tutti i segnali a disposizione, vengono graficati, sotto forma di istogrammi PAW simili a quelli mostrati in Fig. 3.15, i risultati relativi alla differenza tra le coordinate del centroide del *cluster* originale e quelle del *cluster* ricostruito, risultati ottenuti su tutti i segnali analizzati, ed i risultati relativi alla differenza percentuale tra la carica del *cluster* originale e la carica del *cluster* ricostruito, anch'essi ottenuti su tutti i segnali analizzati.

In Fig. 4.10, Fig. 4.11, Fig. 4.12, Fig. 4.13, Fig. 4.14 e Fig. 4.15, sono mostrati i valori del parametro di compressione c al variare della soglia th; in riferimento a ciascuna figura, l'istogramma superiore rappresenta i valori di c relativi a 500 segnali analizzati, mentre, l'istogramma inferiore rappresenta i valori di c relativi ai soli segnali che determinano valori di c inferiori od uguali a 30×10^{-3}.
Come si nota dai risultati ottenuti al variare della soglia th tra 20,21,...,25, i valori di c sono mediamente inferiori rispetto alla richiesta minima di compressione discussa nel Paragrafo 3.3, ovvero $c = 46 \times 10^{-3}$, ed in particolare, per valori della soglia th maggiori od uguali rispetto 21, i valori di c sono mediamente inferiori rispetto al valore di riferimento $c = 30 \times 10^{-3}$; in questo senso, pertanto, l'algoritmo multirisolutivo è in grado di raggiungere, ed eventualmente superare, i valori di compressione ottenuti dall'algoritmo sviluppato presso l'INFN di Torino.

Per quanto riguarda, invece, l'analisi delle differenze tra le coordinate del centroide del *cluster* originale e quelle del *cluster* ricostruito, e l'analisi della differenza percentuale tra la carica del *cluster* originale e la carica del *cluster* ricostruito, non si è potuto fare affidamento su una statistica altrettanto numerosa, come nel caso dell'analisi del parametro c, in cui l'analisi è stata effettuata su 500 eventi caratterizzati da più *cluster* di carica ciascuno.
In particolare, avendo a disposizione solo 20 segnali caratterizzati da un solo *cluster* di carica ciascuno, gli istogrammi relativi alle differenze tra le coordinate del centroide del *cluster* originale e quelle del *cluster* ricostruito, e l'istogramma relativo alle differenze percentuali tra la carica del *cluster* originale e la carica del *cluster* ricostruito, hanno una rilevanza statistica estremamente limitata.
Ciò che può essere comunque affermato è che, a seguito dell'analisi dei suddetti 20 segnali, e per valori di th pari a 21, i valori relativi alle differenze tra le coordinate del centroide del *cluster* originale e quelle del *cluster* ricostruito, ed i valori relativi alle differenze percentuali tra la carica del *cluster* originale e la carica del *cluster* ricostruito, sono confrontabili, come ordine di grandezza, con quelli ottenuti attraverso l'algoritmo di compressione sviluppato presso l'INFN di Torino; in particolare, le differenze tra le coordinate dei centroidi sono dell'ordine del μm, mentre, le differenze tra le cariche dei *cluster*, rivelano una sottostima della carica pari a qualche punto percentuale.

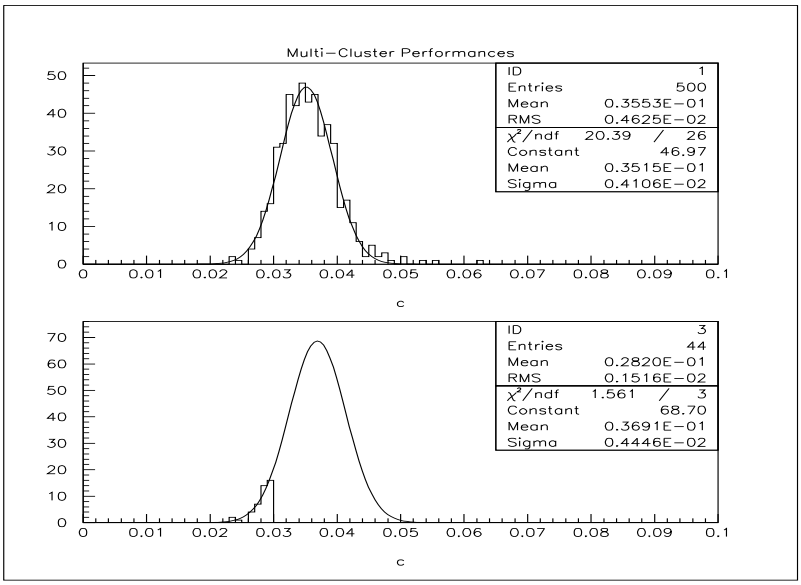


Figura 4.10: Valori di c per th=20.

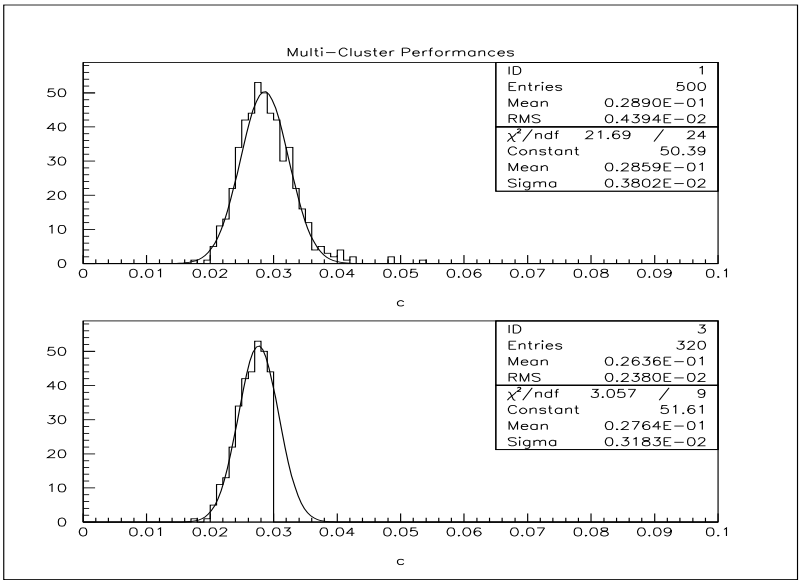


Figura 4.11: Valori di c per th=21.

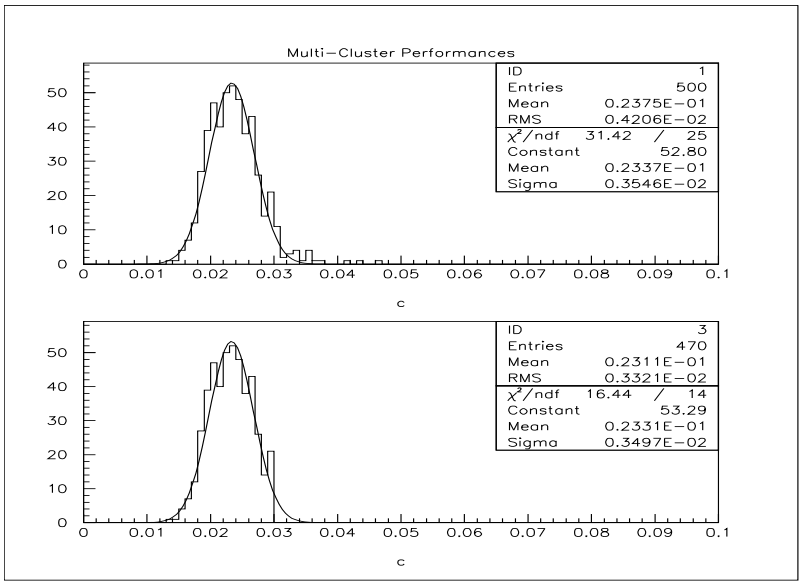


Figura 4.12: Valori di c per th=22.

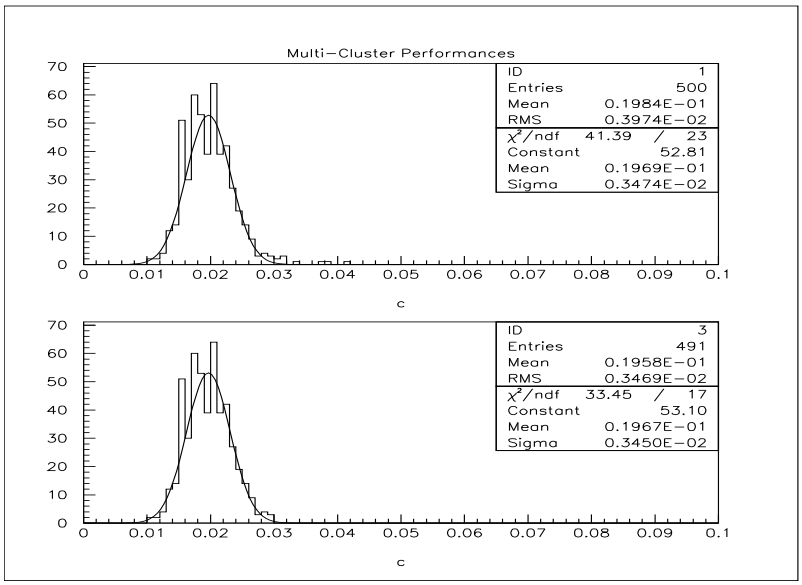


Figura 4.13: Valori di c per th=23.

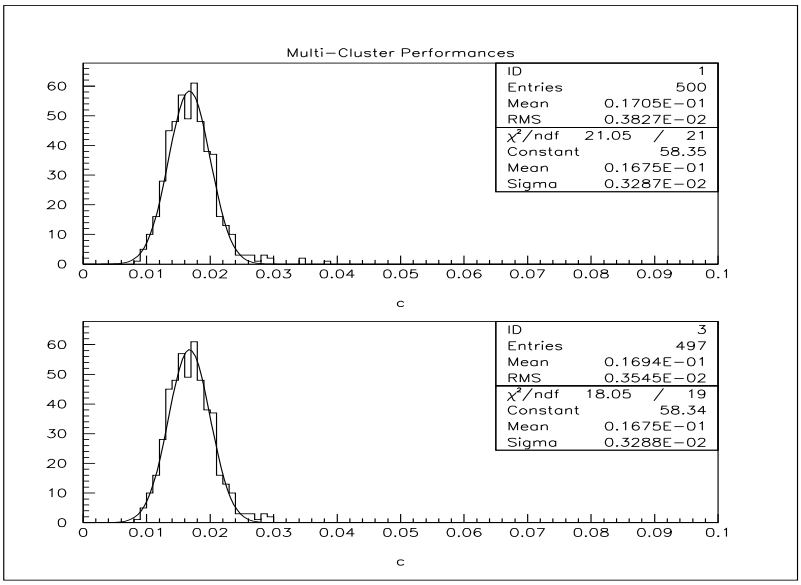


Figura 4.14: Valori di c per th=24.

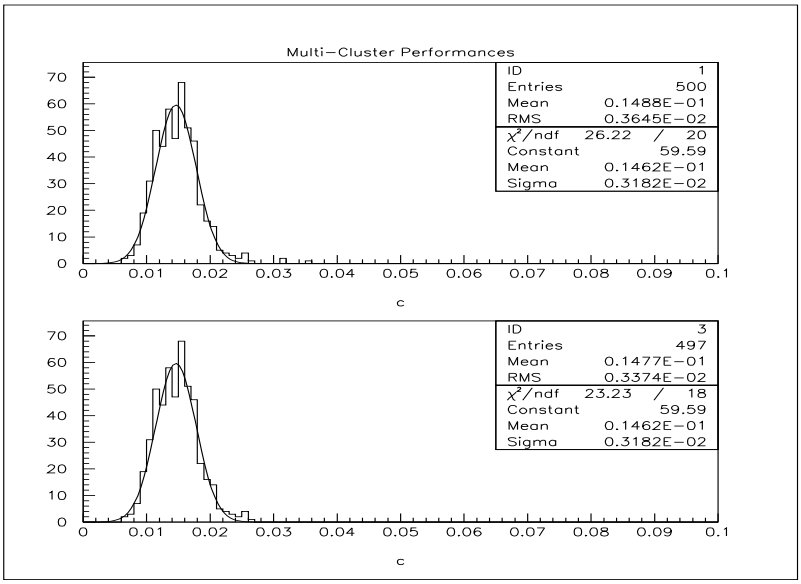


Figura 4.15: Valori di c per th=25.

Capitolo 5

Conclusioni

Alla luce degli studi condotti attraverso il Wavelet Toolbox di Matlab, si può affermare che, l'analisi multirisoluzione che meglio si adatta alla natura dei segnali provenienti dai rivelatori SDD dell'*Inner Tracking System* di ALICE, è quella bi-dimensionale, sviluppata sul numero massimo di livelli di decomposizione, ovvero 8 livelli, ed utilizzante i filtri di Haar.
Gli studi effettuati attraverso Simulink ed il Fixed-Point Blockset, mostrano come l'errore di ricostruzione, ed in parte minore il valore di compressione raggiunto, dipendano dall'architettura della *word* con cui sono condotti i calcoli relativi all'analisi wavelet multirisoluzione ed alla corrispondente sintesi; in particolare, una architettura IEEE 754 a 32 bit in virgola mobile, sembra la più indicata, sia per il livello di precisione che garantisce, sia per la semplicità che contraddistingue l'implementazione dell'analisi e della sintesi wavelet multirisoluzione, in processori digitali quali i DSP (*Digital Signal Processors*), molti dei quali proprio basati su questo tipo di architettura.
La subroutine FORTRAN sviluppata, implementa l'analisi e la sintesi wavelet multirisoluzione nella loro configurazione più prestante, ovvero, quella emersa dagli studi compiuti attraverso il Wavelet Toolbox, e prevede, come codifica dei coefficienti di decomposizione nulli, quella discussa nel Paragrafo 4.4; mandandola in esecuzione su un processore SPARC5, basato su una architettura IEEE 754 a 32 bit in virgola mobile, è possibile un confronto diretto tra i valori di compressione e di errore nella ricostruzione dei *cluster* di carica, ottenuti dall'algoritmo multirisolutivo sviluppato in questo lavoro di tesi, ed i corrispondenti valori ottenuti dall'algoritmo di compressione sviluppato presso l'Istituto Nazionale di Fisica Nucleare di Torino.

In particolare, calcolando il coefficiente di compressione $c = \frac{\text{n}^\text{o} \text{ bit in uscita}}{\text{n}^\text{o} \text{ bit in ingresso}}$, attraverso la subroutine sviluppata, su 500 segnali caratterizzati da più *cluster* ciascuno, si è notato che, per valori della soglia applicata ai coefficienti di decomposizione pari a 21, si ottengono valori di c mediamente inferiori rispetto a quelli ottenuti dall'algoritmo sviluppato presso l'Istituto Nazionale di Fisica Nucleare di Torino, ovvero $c = 30 \times 10^{-3}$; per quanto riguarda i valori relativi all'errore commesso nella ricostruzione dei *cluster* di carica, invece, l'unica affermazione che si può fare è che, per valori della soglia th pari a 21, i valori relativi all'indeterminazione delle coordinate del centroide del *cluster*, ed i valori relativi alla sottostima della carica del *cluster*, sono confrontabili, come ordine di grandezza, con quelli ottenuti attraverso l'algoritmo di compressione sviluppato presso l'Istituto Nazionale di Fisica Nucleare di Torino: infatti, i soli 20 eventi in nostro possesso, caratterizzati da un solo *cluster* ciascuno, non hanno consentito una analisi dell'errore statisticamente rilevante.

Il presente lavoro di tesi lascia aperti due fronti di indagine, relativi all'algoritmo multirisolutivo: da una parte, previo un arricchimento della statistica degli eventi caratterizzati da un solo *cluster*, occorre puntualizzare, attraverso la subroutine FORTRAN sviluppata, l'analisi sull'errore commesso nella ricostruzione di ogni singolo *cluster*, dall'altra, alla luce dei risultati ottenuti attraverso Simulink ed il Fixed-Point Blockset, occorre sviluppare una implementazione hardware dell'algoritmo multirisolutivo.
In particolare, con riferimento al secondo punto, è stato acquistato un DSP della Analog Devices, ADSP-21160, basato su una architettura IEEE 754 a 32 bit in virgola mobile, sul quale sono stati iniziati studi relativi al numero di cicli di clock con cui viene eseguita la compressione dei dati provenienti dai rivelatori SDD.

Appendice A

I filtri digitali

Sia $\{x_n\}_{n\in\mathbb{Z}}$ un segnale campionato e digitalizzato: un filtro digitale effettua operazioni numeriche sui suoi campioni, ottenendo, in uscita, il segnale $\{y_n\}_{n\in\mathbb{Z}}$ [2].

Si definisce *unit impulse*, il segnale $\{\delta_n\}_{n\in\mathbb{Z}}$ composto da campioni tutti nulli eccetto il campione δ_0 avente valore unitario, si veda Fig. A.1.

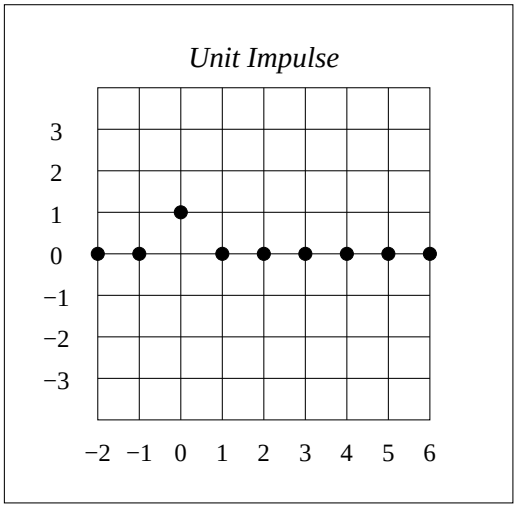


Figura A.1: Lo *unit impulse*.

Si definisce *impulse response*, il segnale $\{h_n\}_{n\in\mathbb{Z}}$ in uscita da un filtro, a seguito di uno *unit impulse* in ingresso.

Se il filtro è lineare, ogni scalamento e traslazione dello *unit impulse* si manifesta sotto forma di un medesimo scalamento e di una medesima traslazione della *impulse response* in uscita; poiché ciascun campione di un segnale in ingresso può essere interpretato come uno *unit impulse* opportunamente scalato e traslato, cui corrisponde in uscita dal filtro una *impulse response* scalata e traslata in maniera analoga, definire la *impulse response* di un filtro digitale, equivale a stabilirne, in maniera univoca, la natura intrinseca.
In particolare, conoscere la $\{h_n\}_{n\in\mathbb{Z}}$ di un filtro, consente la previsione del segnale in uscita, a partire da qualsiasi segnale entrante: infatti, il segnale uscente è espresso dalla somma dei segnali uscenti a seguito di ciascun campione in ingresso, inteso come uno *unit impulse* opportunamente scalato e traslato.
In sostanza, se $\{x_n\}_{n\in\mathbb{Z}}$ è un segnale in ingresso costituito da N campioni e $\{h_n\}_{n\in\mathbb{Z}}$ una *impulse response* costituita da M campioni, il segnale $\{y_n\}_{n\in\mathbb{Z}}$, uscente dal filtro, sarà costituito da $N + M - 1$ campioni, ciascuno calcolabile nella seguente maniera:

$$y_i = \sum_{j=0}^{M-1} h_j x_{i-j} \tag{A.1}$$

In ultima analisi, la (A.1) equivale ad affermare che il segnale $\{y_n\}_{n\in\mathbb{Z}}$, in uscita dal filtro, si ottiene dalla convoluzione del segnale in ingresso $\{x_n\}_{n\in\mathbb{Z}}$ con la *impulse response* $\{h_n\}_{n\in\mathbb{Z}}$ del filtro, si veda (A.2).

$$\boldsymbol{y} = \boldsymbol{x} * \boldsymbol{h} \tag{A.2}$$

Bibliografia

[1] K. Sayood, "Introduction to Data Compression", Morgan Kaufmann, S. Francisco, 1996.

[2] S. W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing", California Technical Publishing, S. Diego, 1999.

[3] B. Burke Hubbard, "The World According to Wavelets: the story of a mathematical technique in the making", A K Peters, Ltd., Wellesley, 1998.

[4] R. Polikar, "The Engineer's ultimate guide to wavelet analysis", http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html, 2001.

[5] M. Misiti, Y. Misiti, G. Oppenheim e J. M. Poggi, "Wavelet Toolbox User's Guide", The MathWorks, Inc., Natick, 2000.

[6] A. Pizzato, "Sviluppo di un'architettura VLSI dedicata alla valutazione della trasformata Wavelet di immagini", tesi di laurea presso il Dipartimento di Elettronica del Politecnico di Torino, Marzo 1999.

[7] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE Transactions on pattern analysis and machine intelligence, Vol. II, NO. 7, pagg. 674-693, July 1989.

[8] P. G. Lemarié e Y.Meyer, "Ondelettes et bases hilbertiennes", Revista Matematica Iberoamericana, Vol. 2, pagg. 1-18, 1986.

[9] E. J. Stollnitz, T. D. DeRose e D. H. Salesin, "Wavelets for computer graphics: a primer", IEEE Computer Graphics and Applications, Vol. 3, NO. 15, pagg.76-84, May 1995 (part 1) e Vol. 4, NO. 15, pagg. 75-85, July 1995 (part 2). Vol. 3, NO. 15, pagg. 76-84, May 1995.

[10] P. Morton, "Image Compression Using the Haar Wavelet Transform", http://online.redwoods.cc.ca.us/instruct/darnold/maw/haar.htm, 1998.

[11] The LHC study group, "The Large Hadron Collider Conceptual Design", October 1995, CERN/AC/95-05(LHC).

[12] ALICE Collaboration, "Technical Proposal for A Large Ion Collider Experiment at the CERN LHC", December 1995, CERN/LHCC/95-71.

[13] P. Giubellino e E. Crescio, "The ALICE experiment at LHC: physics prospects and detector design", January 2001, ALICE-PUB-2000-35.

[14] CERN/LHCC 99-12 ALICE TDR 4, 18 June 1999.

[15] E. Crescio, D. Nouais e P. Cerello, "A detailed study of charge diffusion and its effect on spatial resolution in Silicon Drift Detectors", September 2001, ALICE-INT-2001-09.

[16] D. Cavagnino, P. De Remigis, P. Giubellino, G. Mazza, e A. E. Werbrouck, "Data Compression for the ALICE Silicon Drift Detector", 1998, ALICE-INT-1998-41.

[17] "Simulink User's Guide: Dynamic System Simulation for Matlab", The MathWorks, Inc., Natick, 2000.

[18] "Fixed-Point Blockset User's Guide: for Use with Simulink", The MathWorks, Inc., Natick, 2000.