Analisi Spettrale discreta (DFT e FFT)

Mario Ambrosino

29 Giugno 2018

Capitolo 1

Esercitazione

L'indirizzo GitHub associato a questa esercitazione è dato da:

`https://github.com/mario-ambrosino/ft`

I sorgenti rilevanti ai fini dell'esercitazione sono

- `dft.c`
  `https://github.com/mario-ambrosino/ft/blob/master/sources/dft.c`
- `fft.c`
  `https://github.com/mario-ambrosino/ft/blob/master/sources/fft.c`
- `bit_reverse.c`
  `https://github.com/mario-ambrosino/ft/blob/master/sources/bit_reverse.c`
- `cfr.sh`
  `https://github.com/mario-ambrosino/ft/blob/master/cfr.sh`

Per l'analisi dello spettro del pendolo caotico invece ho utilizzato i dati dell'esercitazione precedente assieme al seguente script

- `cfr_pendulum.sh`
  `https://github.com/mario-ambrosino/ft/blob/master/cfr_pendulum.sh`

Ho generato un'animazione dell'andamento dello spettro di potenza al variare della frequenza della forzante presso l'indirizzo `https://github.com/mario-ambrosino/ft/blob/master/dataset/extracted_pendulum/animation.gif`

Gli altri sono solo utilità collaterali, già viste nelle altre esercitazioni.

Gli eseguibili associati ai primi due sorgenti sono tali da accettare un file in input separati da punto e virgola e con valori reali e immaginari dei punti da analizzare tramite DFT(FFT) come prima e seconda colonna. FFT controlla se l'input ha la cardinalità adatta (potenza di 2), altrimenti rifiuta l'esecuzione. In output forniscono un file `*.dat` con stessa struttura dati dell'input. DFT presenta anche la possibilità di valutare l'inversa tramite la posizione (obbligatoria) di un intero come argomento acquisito tramite shell che definisce il funzionamento: 0 sta per *trasformata di Fourier* e 1 per *antitrasformata*.

Non è effettuato alcun controllo sulla variabile indipendente, anzi essa è da considerarsi assente ai fini dell'esercitazione. L'eseguibile `bit_reverse.exe` è un prototipo dell'inversione di bit necessaria per l'esecuzione dell'algoritmo FFT scelto. Da notare che per FFT e DFT ho utilizzato la libreria standard complessa di C, facendo a meno dell'esecuzione di funzioni trigonometriche, e ho usato l'algoritmo di bit reversal spiegato in aula.

Ho usato poi lo script `cfr.sh` per generare gli istogrammi per tutte le funzioni prototipo utilizzate. In particolare riporto qui l'esecuzione per la funzione suggerita in aula

$$f = x\,(1 - x) \quad x \in [0, 1]$$

ma sono disponibili per l'analisi dell'analisi la funzione rettangolare, la $\theta\,(x)$ di Heaviside, la funzione seno e del semplice segnale aleatorio uniformemente distribuito. Riporto in basso alcune immagini rilevanti, non ritenendo significativo estendere troppo l'analisi.

Qualitativamente i grafici delle inverse (tramite DFT) delle trasformate secondo i metodi DFT e FFT coincidono con i dati originali ma non ho valutato numericamente la discrepanza.

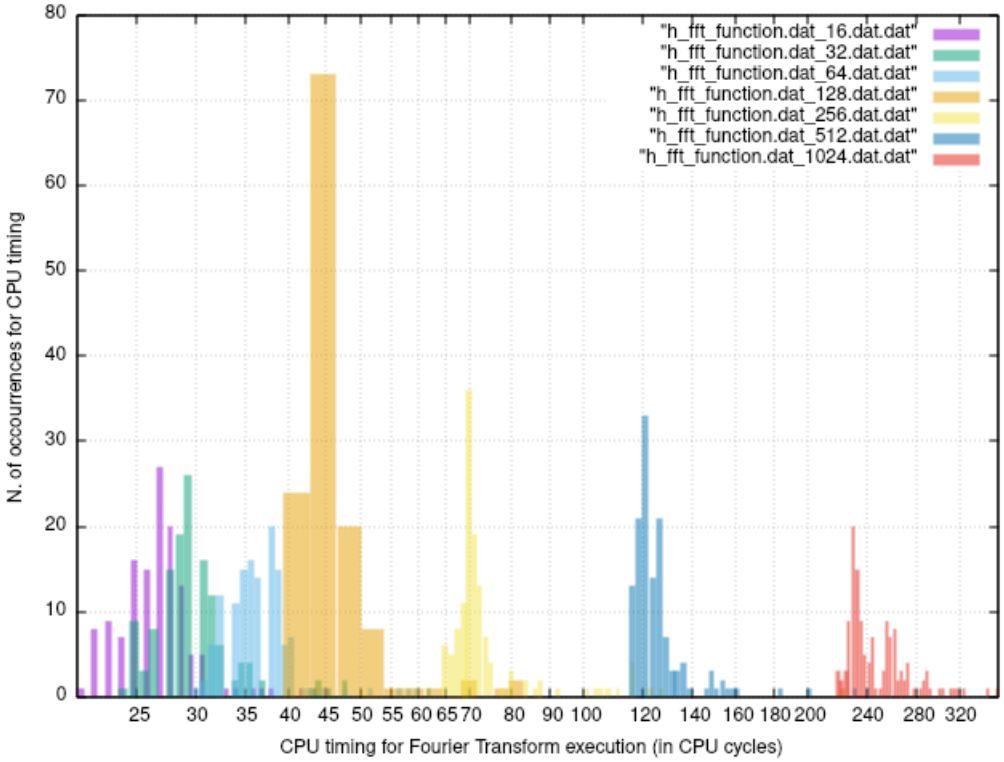

Figura 1.1: Confronto per l'algoritmo FFT dei tempi di esecuzione al variare della cardinalità del dataset (che scala come potenza di 2)

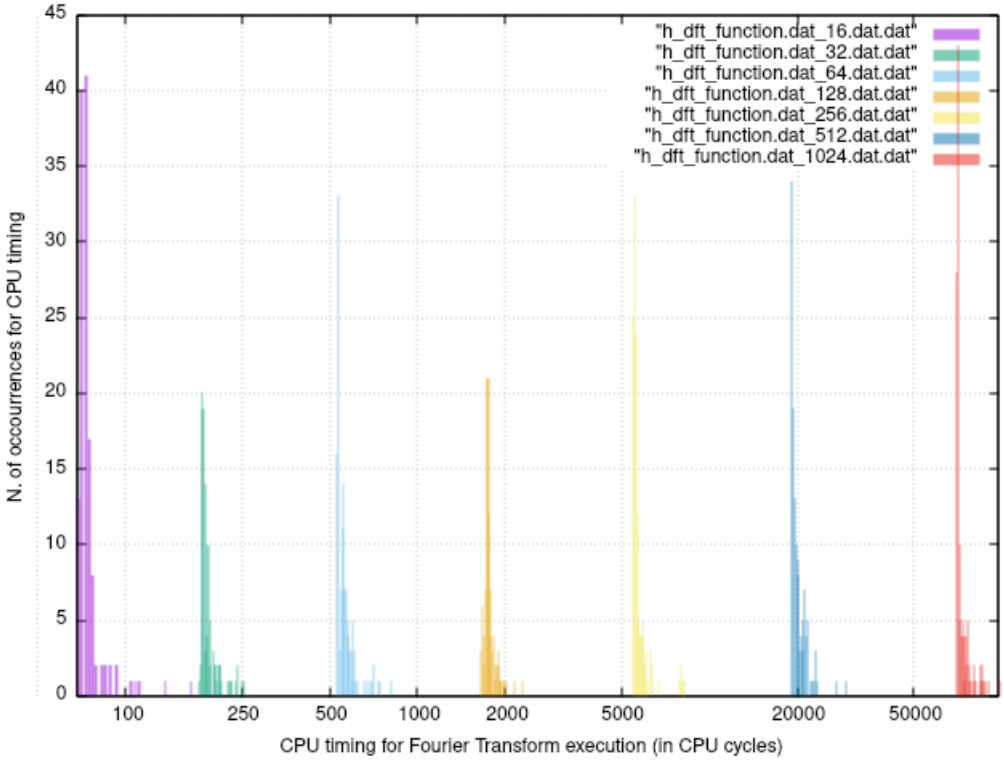

Figura 1.2: Confronto per l'algoritmo DFT dei tempi di esecuzione al variare della cardinalità del dataset (che scala come potenza di 2 per un confronto diretto con FFT)