Università degli studi di Napoli “Federico II”

Facoltà di Scienze MM.FF.NN. - Informatica

Corso di Calcolo Scientifico

Elaborato 3

Sviluppo di un software per il calcolo della DFT di un vettore di reali x *di* n *elementi, con* $n=2^k$ *e* $k=1,\ldots,m$ *(* $m \geq 12$ *), mediante l'utilizzo delle librerie FFTW3 e DFFTPACK.*

Ferrara Francesco Saverio

566/811

prof.ssa L. D'Amore

A.A. 2004/2005

Scopo

Lo scopo di questo programma è effettuare un confronto, in termini di velicità, tra le librerie DFFTPACK e FFTW3. Per fare ciò si calcola la DFT di m vettori (con $m \geqslant 12$) con entrambe le librerie e alla fine si stampano i tempi di esecuzione.

Analisi del problema

Scegliamo di effettuare il calcolo di DFT radix 2 di vettori reali. Per questo i vettori che diamo in pasto alle routine di DFFTPACK e FFTW3 sono di n elementi, con $n=2^k$ e $k=1,\ldots,m$.

Descrizione della strategia utilizzata:
Un vettore di dimensione $n=2^{12}$ ha ben 4096 elementi, quindi è impensabile costringere l'utente ad inserire da tastiera un vettore così lungo: per questo motivo è stato scelto di generare i vettori casualmente.
Ognuno degli m vettori generati casualmente, viene passato prima alle routine di FFTW3 e poi a DFFTPACK, in modo da confrontare le due librerie facendole lavorare esattamente sullo stesso vettore.

Descrizione dell'algoritmo:
In una prima fase l'algoritmo esamina gli argomenti passati per riga di comando, in quanto se viene inserita l'opzione '-h` oppure '--help` viene stampata a video una mini guida che aiuta l'utente a inserire un input corretto.
Con l'opzione '-v` oppure '--verbose` si chiede al programma di stampare a video messaggi aggiuntivi per avere maggiori informazioni.
Se invece il programma viene chiamato senza opzioni, allora chiede all'utente di inserire il numero di vettori usati per effettuare le prove delle librerie.
Dopo aver fatto questo vengono eseguite le seguenti istruzioni:

```
01 // Calcola la dimensione del vettore piu' lungo
02 n = pow(2, m);
03
04 // Allocazione della memoria per i vettori
05 x = dveal(n);
06 if (x == NULL) { // Se c'e' stato un errore durante l'allocazione della memoria
07         fprintf(stderr, "Errore durante l'allocazione della memoria\n");
08         return EXIT_FAILURE;
09 }
10
11 wsave = dveal(2*n+15);
12 if (wsave == NULL) { // Se c'e' stato un errore durante l'allocazione della memoria
13         fprintf(stderr, "Errore durante l'allocazione della memoria\n");
14         return EXIT_FAILURE;
15 }
```

Viene calcolata la massima dimensione del vettore (riga 2) e viene allocata la memoria per i vettori (righe 5 e 11).

A questo punto dell'algoritmo troviamo un ciclo:

```
01 for  (k=1 ; k <= m ; k++) {
02         n = pow(2, k); // Calcolo la dimensione del k-esimo vettore
03         if (verbose) // Se e' stata immessa l'opzione "-v"
04                 fprintf(stdout, "---> Calcolo la DFT del vettore %d di lunghezza %d.\n", k, n);
05
06         for (i=0 ; i<n ; i++) // Generazione di elementi casuali
07                 x[i] = rand() % MAX_CASUALE;
08
09         if (verbose) { // Se e' stata immessa l'opzione "-v"
10                 fprintf(stdout, "Genero il vettore con numeri casuali -> ");
11                 dvest(x, n);
12         }
13
14         // Calcolo della DFT del vettore con fftw3
15         if ( gettimeofday(&start, &tz) != 0 ) {
16                 perror("getimeoftime()");
17                 return EXIT_FAILURE;
18         }
19
20         p = fftw_plan_r2r_1d(n, x, wsave, FFTW_R2HC, FFTW_ESTIMATE);
21         fftw_execute(p);
22         fftw_destroy_plan(p);
23
24         if ( gettimeofday(&stop, &tz) != 0 ) {
25                 perror("gettimofday()");
26                 return EXIT_FAILURE;
27         }
28
29         fprintf(stdout, "Tempo:    FFTW3 vettore %2d -> ",k);
30         stampa_tempo(&start, &stop);
31         fprintf(stdout, "\n");
32                 if (verbose) { // Se e' stata immessa l'opzione "-v"
33                 fprintf(stdout, "DFT con FFTW3 -> ");
34                 dvest(wsave, n);
35         }
36
37         // Calcolo della DFT del vettore con dfftpack
38         if ( gettimeofday(&start, &tz) != 0 ) {
39                 perror("getimeoftime()");
40                 return EXIT_FAILURE;
41         }
42
43         dffti_(&n, wsave);
44         dfftf_(&n, x, wsave);
45
46         if ( gettimeofday(&stop, &tz) != 0 ) {
47                 perror("gettimofday()");
48                 return EXIT_FAILURE;
49         }
50
51         fprintf(stdout, "Tempo: DFFTPACK vettore %2d -> ",k);
52         stampa_tempo(&start, &stop);
53         fprintf(stdout, "\n");
54
55         if (verbose) { // Se e' stata immessa l'opzione "-v"
56                 fprintf(stdout, "DFT con DFFTPACK -> ");
57                 dvest(x, n);
58         }
59
60         fprintf(stdout, "\nPremi INVIO per continuare\n");
61         getchar();
62 }
```

Ad ogni iterazione del cilo for di riga 1 viene:
- calcolata la dimensione del vettore (riga 2),
- generati gli elementi del vettore in modo casuale (righe 6-7),
- fatto partire il timer per FFTW3 (riga 15),
- calcolata la DFT con FFTW3 (righe 20-22),
- fermato il timer per FFTW3 (riga 24),
- fatto partire il timer per DFFTPACK (riga 38),
- calcolata la DFT con DFFTPACK (righe 43-44),
- fermato il timer per DFFTPACK (riga 46).

Uscito dal ciclo for vengono effettuate le seguenti operazioni di chiusura:

```
01 // Libero la memoria usata per i vettori
02 free(x);
03 free(wsave);
04
05 fprintf(stdout, "Programma terminato... arrivederci!\n");
```

Notiamo che sia per FFTW3 che per DFFTPACK sono state usare funzioni che calcolano la DFT di un vettore reale e restituiscono il risultato sotto forma di un vettore reale.

Indicatori di errore

Il programma ritorna un indicatore di errore in caso di passaggio di parametri non riconosciuti oppure di errore nell'allocazione della memoria.

Routine ausiliarie

Per il corretto funzionamento, il programma si avvale di diverse routine classificabili in routine interne, esterne, e appartenenti alle librerie FFTW3 e DFFTPACK.

Routine interne:

```
void help();
```

Questa procedura stampa a video una semplice mini-guida per aiutare l'utente a inserire i dati in input.

```
void stampa_tempo(struct timeval *start, struct timeval *stop);
```

Dati due tempi (start e stop), stampa a video la differenza tra i due.

Routine esterne:

Sono tutte definite nel file “dvetlib.h”.

```
double *dveal(int n);
```

Alloca lo spazio per un vettore double di n elementi e restituisce il puntatore all'area di memoria allocata.

```
void dvest(double *x, int n);
```

Stampa sullo standard output il vettore x di n elementi.

Routine esterne (FFTW3):

Sono alcune delle routine messe a disposizione dalla libreria FFTW3.

La routine:

```
fftw_plan_r2r_1d();
```

è usata per creare un plan di una DFT di un vettore (1d) reale mettendo il risultato in un altro vettore reale (r2r).

La routine:

```
fftw_execute();
```

è usata per eseguire il plan creato con fftw_plan_r2r_1d().

La routine:

```
fftw_destroy_plan();
```

è usata per cancellare un plan.

Routine esterne (DFFTPACK):

Sono alcune delle routine messe a disposizione dalla libreria DFFTPACK.

La routine:

```
void dffti_(int *n, double *wsave);
```

è usata per inizializzare i vettori “n” e “wsave”.

La routine:

```
void dfftf_(int *n, double *x, double *wsave);
```

è usata per calcolare la DFT del vettore “x”.

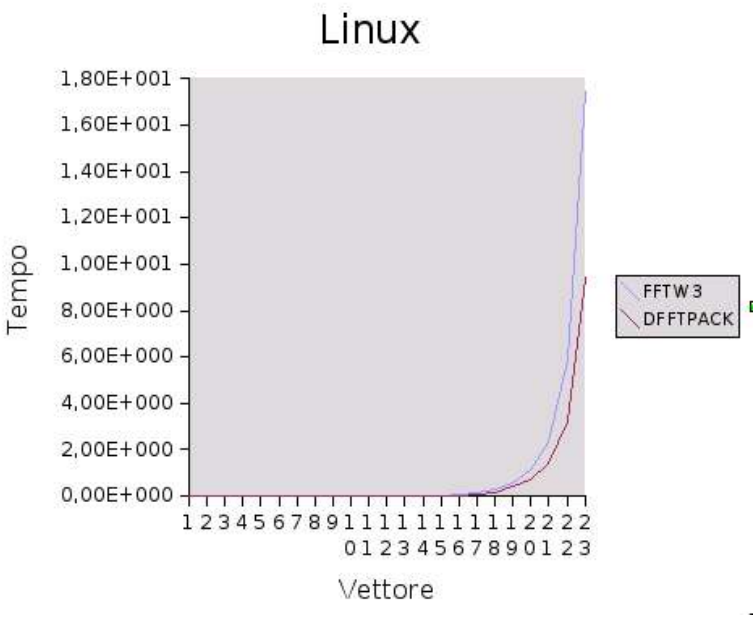
Complessità

Siccome abbiamo utilizzato sostanzialmente librerie di terze parti, non abbiamo modo di agire sull'algoritmo per diminuire la sua complessità perchè essa è strettamente legata a quella i DFFTPACK e DFFTW3.

Complessità di tempo:
La complessità di tempo dipende molto da quella delle librerie DFFTPACK e FFTW3.
Dopo aver fatto due prove su:

1) Linux 2.4.27 su un X86 Pentium Celeron 1,8 GHz (256MB):

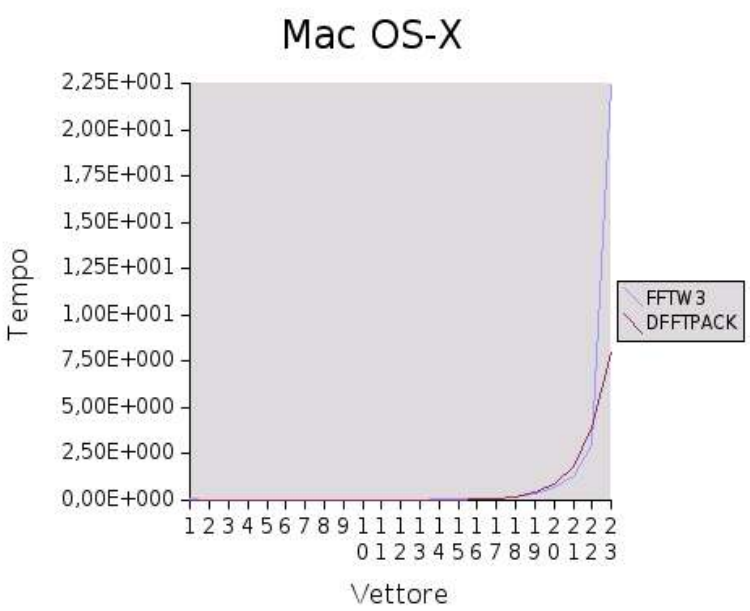
Linux	FFTW3	DFFTPACK
1	6,50E-004	6,25E-003
2	5,00E-005	6,00E-006
3	4,20E-005	2,60E-005
4	4,70E-005	4,00E-006
5	5,73E-004	7,00E-006
6	1,28E-003	1,00E-005
7	2,16E-003	2,20E-005
8	3,75E-003	5,20E-005
9	5,75E-003	9,50E-005
10	7,74E-003	1,98E-004
11	1,01E-002	3,95E-004
12	1,26E-002	8,17E-004
13	1,59E-002	1,77E-003
14	2,09E-002	4,77E-003
15	3,48E-002	1,37E-002
16	7,07E-002	3,40E-002
17	1,50E-001	6,99E-002
18	2,81E-001	1,79E-001
19	5,69E-001	4,18E-001
20	1,11E+000	6,84E-001
21	2,31E+000	1,41E+000
22	5,85E+000	3,21E+000
23	1,75E+001	9,40E+000



Su questa architettura risulta essere sepre più veloce DFFTPACK.

2) Mac OS-X 10.3.4 su un G4 Power PC 1,25 Ghz (512MB):

Mac	FFTW3	DFFTPACK
1	6,41E-002	1,40E-005
2	6,00E-005	2,00E-006
3	4,70E-005	1,10E-005
4	5,10E-005	4,00E-006
5	7,39E-004	5,00E-006
6	1,65E-003	8,00E-006
7	3,07E-003	1,80E-005
8	5,13E-003	5,40E-005
9	7,69E-003	9,20E-005
10	1,05E-002	1,80E-004
11	1,37E-002	3,53E-004
12	1,72E-002	7,17E-004
13	2,16E-002	1,65E-003
14	2,78E-002	3,60E-003
15	3,96E-002	1,13E-002
16	5,65E-002	3,50E-002
17	9,90E-002	9,44E-002
18	1,82E-001	1,93E-001
19	3,41E-001	4,07E-001
20	6,58E-001	8,35E-001
21	1,34E+000	1,86E+000
22	2,96E+000	3,93E+000
23	2,24E+001	7,99E+000



Su questa architettura invece, per vettori piccoli DFFTPACK è più veloce, ma per vettori più grandi FFTW3 si dimostra migliore.

Ricordiamo che nel tempo di FFTW3 è incluso anche il tempo di creazione del "plan", e che se dobbiamo calcolare le DFT di molti vettori dello stasso tipo e della stessa lunghezza allora basta creare una solo plan ed utilizzarlo più volte.

Complessità di spazio:

L'algoritmo alloca spazio per:

- vettore x di $n=2^m$ elementi,
- vettore wsave di $n=2^m$ elementi,
- 11 variabili.

Quindi: $S(n)=2\cdot n+11$; asintoticamente si ha:

$$S(n)=O(2\cdot n)$$

Accuratezza fornita

L'algoritmo utilizza numeri reali double (doppia precisione).

Raccomandazioni di utilizzo

Pe far funzionare il programma, non bisogna passargli argomenti dalla linea di comando.
Se si vuole che venga stampata a video una mini-guida all'uso del programma, basta passargli l'opzione '-h` o '--help`.
Se invece si desiderano informazioni aggiuntive mentre si esegue il programma, basta passargli l'opzione '-v` o '--verbose`.

Riferimenti bibliografici

“A. Murli”
Lucidi del corso di Calcolo Scientifico
http://www.dma.unina.it/~murli/didattica/mat_didattico_cs0405.html
2004-2005

“Paul N. Swartztrauber”
Documentazione di DFFTPACK
http://www.netlib.org/
1985 – National center for atmosperic research

“Matteo Frigo” “Steven G. Johnson”
FFTW3 Manual
http://www.fftw.org/
2004

Esempio d'uso

Facciamo un esempio su un computer x86 (Pentium Celeron 1.8 Mhz) con il sistema operativo Linux 2.4.27:

```
Inserisci il numero di prove:
23

Tempo:    FFTW3 vettore  1 -> 0.000650 secondi
Tempo: DFFTPACK vettore  1 -> 0.000009 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  2 -> 0.000050 secondi
Tempo: DFFTPACK vettore  2 -> 0.000006 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  3 -> 0.000042 secondi
Tempo: DFFTPACK vettore  3 -> 0.000026 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  4 -> 0.000047 secondi
Tempo: DFFTPACK vettore  4 -> 0.000004 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  5 -> 0.000573 secondi
Tempo: DFFTPACK vettore  5 -> 0.000007 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  6 -> 0.001283 secondi
Tempo: DFFTPACK vettore  6 -> 0.000010 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  7 -> 0.002160 secondi
Tempo: DFFTPACK vettore  7 -> 0.000022 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  8 -> 0.003754 secondi
Tempo: DFFTPACK vettore  8 -> 0.000052 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  9 -> 0.005749 secondi
Tempo: DFFTPACK vettore  9 -> 0.000095 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 10 -> 0.007736 secondi
Tempo: DFFTPACK vettore 10 -> 0.000198 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 11 -> 0.010122 secondi
Tempo: DFFTPACK vettore 11 -> 0.000395 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 12 -> 0.012613 secondi
Tempo: DFFTPACK vettore 12 -> 0.000817 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 13 -> 0.015875 secondi
Tempo: DFFTPACK vettore 13 -> 0.001774 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 14 -> 0.020945 secondi
Tempo: DFFTPACK vettore 14 -> 0.004768 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 15 -> 0.034800 secondi
Tempo: DFFTPACK vettore 15 -> 0.013742 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 16 -> 0.070698 secondi
Tempo: DFFTPACK vettore 16 -> 0.034006 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 17 -> 0.150275 secondi
Tempo: DFFTPACK vettore 17 -> 0.069943 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 18 -> 0.280896 secondi
Tempo: DFFTPACK vettore 18 -> 0.178726 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 19 -> 0.569449 secondi
Tempo: DFFTPACK vettore 19 -> 0.417750 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 20 -> 1.106566 secondi
Tempo: DFFTPACK vettore 20 -> 0.684390 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 21 -> 2.310942 secondi
Tempo: DFFTPACK vettore 21 -> 1.405625 secondi

Premi INVIO per continuare
```

```
Tempo:    FFTW3 vettore 22 -> 5.851658 secondi
Tempo: DFFTPACK vettore 22 -> 3.207357 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 23 -> 17.482657 secondi
Tempo: DFFTPACK vettore 23 -> 9.403271 secondi

Premi INVIO per continuare
Programma terminato... arrivederci!
```

Un ulteriore esempio su un computer G4 (Power PC 1.25 Mhz) con il sistema operativo Mac OS-X 10.3.4:

```
Inserisci il numero di prove:
23

Tempo:    FFTW3 vettore  1 -> 0.064099 secondi
Tempo: DFFTPACK vettore  1 -> 0.000014 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  2 -> 0.000060 secondi
Tempo: DFFTPACK vettore  2 -> 0.000002 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  3 -> 0.000047 secondi
Tempo: DFFTPACK vettore  3 -> 0.000011 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  4 -> 0.000051 secondi
Tempo: DFFTPACK vettore  4 -> 0.000004 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  5 -> 0.000739 secondi
Tempo: DFFTPACK vettore  5 -> 0.000005 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  6 -> 0.001654 secondi
Tempo: DFFTPACK vettore  6 -> 0.000008 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  7 -> 0.003065 secondi
Tempo: DFFTPACK vettore  7 -> 0.000018 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  8 -> 0.005128 secondi
Tempo: DFFTPACK vettore  8 -> 0.000054 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore  9 -> 0.007692 secondi
Tempo: DFFTPACK vettore  9 -> 0.000092 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 10 -> 0.010468 secondi
Tempo: DFFTPACK vettore 10 -> 0.000180 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 11 -> 0.013655 secondi
Tempo: DFFTPACK vettore 11 -> 0.000353 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 12 -> 0.017191 secondi
Tempo: DFFTPACK vettore 12 -> 0.000717 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 13 -> 0.021587 secondi
Tempo: DFFTPACK vettore 13 -> 0.001647 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 14 -> 0.027787 secondi
Tempo: DFFTPACK vettore 14 -> 0.003602 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 15 -> 0.039587 secondi
Tempo: DFFTPACK vettore 15 -> 0.011282 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 16 -> 0.056488 secondi
Tempo: DFFTPACK vettore 16 -> 0.034967 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 17 -> 0.099041 secondi
Tempo: DFFTPACK vettore 17 -> 0.094356 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 18 -> 0.182136 secondi
Tempo: DFFTPACK vettore 18 -> 0.192690 secondi
```

```
Premi INVIO per continuare
Tempo:    FFTW3 vettore 19 -> 0.341279 secondi
Tempo: DFFTPACK vettore 19 -> 0.406508 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 20 -> 0.658197 secondi
Tempo: DFFTPACK vettore 20 -> 0.835162 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 21 -> 1.335838 secondi
Tempo: DFFTPACK vettore 21 -> 1.857525 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 22 -> 2.964865 secondi
Tempo: DFFTPACK vettore 22 -> 3.925346 secondi

Premi INVIO per continuare
Tempo:    FFTW3 vettore 23 -> 22.447257 secondi
Tempo: DFFTPACK vettore 23 -> 7.991035 secondi

Premi INVIO per continuare
Programma terminato... arrivederci!
```

Codice sorgente

dveal.c :

```
#include "dvetlib.h"


double *dveal(int n) {
        double *x;

        x = (double *) calloc(n, sizeof(double)); // Alloca lo spazio di memoria
        if (x==NULL) // Se c'e' stato un errore
                fprintf(stderr, "Errore nell'allocazione della memoria\n");
        return x;
}
```

dvest.c :

```
#include "dvetlib.h"


void dvest(double *x, int n) {
        int i;

        fprintf(stdout, "Stampa del vettore:\n");
        for (i=0 ; i<n ; i++)
                fprintf(stdout, " [%2d]-> %lf ", i, x[i]);
        fprintf(stdout, "\n");
}
```

dvetlib.h :

```
#ifndef DVETLIB
#define DVETLIB



#include <stdio.h>


double *dveal(int n);
        /*
        Alloca un vettore di n elementi double e restituisce il puntatore
        all'area di memoria allocata.
        Ritorna NULL in caso di errore.
        */

void dvest(double *x, int n);
        /*
        Stampa il vettore "x" di "n" elementi sullo standard output.
        */


#endif
```

main.c :

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <errno.h>
#include <sys/time.h>
#include "dvetlib.h"

#include <fftw3.h> // Utilizzato da fftw3

#define MIN_PROVE 12
#define MAX_CASUALE 100


void dffti_(int *n, double *wsave);
void dfftf_(int *n, double *x, double *wsave);



void stampa_tempo(struct timeval *start, struct timeval *stop) {


        double secondi, microsecondi;


        secondi = stop->tv_sec - start->tv_sec; // Differenza tra i secondi
        microsecondi = abs(stop->tv_usec - start->tv_usec); // Differenza tra i microsecondi

        // Aggiunge i microsecondi ai secondi
        secondi += ( microsecondi / 1000000. );

        fprintf(stdout, "%lf secondi", secondi);
}


void help() {
        fprintf(stdout,
"###############################################################################\n");
        fprintf(stdout, "#                      LABORATORIO DI CALCOLO SCIENTIFICO
#\n");
        fprintf(stdout, "#                               A.A. 2004/2005
#\n");
        fprintf(stdout, "#
#\n");
        fprintf(stdout, "#                      Ferrara Francesco Saverio - 566/811
#\n");
        fprintf(stdout, "#                          fsterrar@studenti.unina.it
#\n");
        fprintf(stdout,
"###############################################################################\n\n");
        fprintf(stdout, "Sia M il numero di prove da effettuare, questo programma calcola la
DFT del vettore\n");
        fprintf(stdout, "x di dimensione n con n=2^k (per k=1,...,M).\n");
        fprintf(stdout, "Per calcolare la DFT del vettore generato con numeri casuali, viene
impiegata prima\n");
        fprintf(stdout, "la libreria FFTW3 e poi la libreria DFFTPACK. Alla fine viene
stampato il tempo\n");
        fprintf(stdout, "impiegato da ognuna per confrontarle.\n\n");
        fprintf(stdout, "Opzioni accettate dal programma:\n");
        fprintf(stdout, "\t -h o --help\t:\tVisualizza questa guida\n");
        fprintf(stdout, "\t -v o --verbose\t:\tVisualizza le DFT dei vettori\n\n");
}


int main(int argc, char *argv[]) {

        int n; // Dimensione del vettore
        int m; // Numero di prove
        int k; // Potenze di due
        int i;
        int errore = 1; // Utile al passaggio dei parametri
        int verbose = 0; // Se settata ad 1 il programma stampera' tutte le informazioni sulle
DFT
        struct timeval start, stop; // Questa struttura ha due campi: secondi e microsecondi
        struct timezone tz; // Utilizzato da gettimeofday()
        double tmp; // Variabile temporanea usata per memorizzare gli stessi elementi casuali

        double *x, *wsave; // Utilizzati con dfftpack

        fftw_plan p; // Utilizzato con fftw3



        if (argc > 1) { //Se sono stati passati argomenti
                if (argc == 2) {
                        if ((strncmp(argv[1],"--help",7)==0) || (strncmp(argv[1],"-h",3)==0))
{
                                help(); //Stampo a video una mini-guida all'uso del programma
                                return EXIT_SUCCESS; //Esco con un valore di successo
                        }
                        if ((strncmp(argv[1],"--verbose",10)==0) || (strncmp(argv[1],"-v",3)
```

```
==0)) {
                                errore = 0; // Evito l'uscita dal programma
                                verbose = 1; // Setto la variabile in modo da stampare a
video le informazioni sulle DFT
                        }
                }
                if (errore) {
                        fprintf(stderr, "Errore nel passaggio dei parametri\nUsa: %s --
help\nper avere maggiori informazioni\n", argv[0]);
                        return EXIT_FAILURE; //Ritorno alla shell un indicatore di errore
                }
        }


        // Leggo da input il numero di prove che bisogna effettuare
        do {
                fprintf(stdout, "Inserisci il numero di prove: ");
                scanf("%d", &m);
                getchar();
        } while(m < MIN_PROVE);


        // Calcola la dimensione del vettore piu' lungo
        n = pow(2, m);


        // Allocazione della memoria per i vettori
        x = dveal(n);
        if (x == NULL) { // Se c'e' stato un errore durante l'allocazione della memoria
                fprintf(stderr, "Errore durante l'allocazione della memoria\n");
                return EXIT_FAILURE;
        }

        wsave = dveal(2*n+15);
        if (wsave == NULL) { // Se c'e' stato un errore durante l'allocazione della memoria
                fprintf(stderr, "Errore durante l'allocazione della memoria\n");
                return EXIT_FAILURE;
        }



        for  (k=1 ; k <= m ; k++) {

                n = pow(2, k); // Calcolo la dimensione del k-esimo vettore
                if (verbose) // Se e' stata immessa l'opzione "-v"
                        fprintf(stdout, "---> Calcolo la DFT del vettore %d di lunghezza %
d.\n", k, n);

                for (i=0 ; i<n ; i++) // Generazione di elementi casuali
                        x[i] = rand() % MAX_CASUALE;

                if (verbose) { // Se e' stata immessa l'opzione "-v"
                        fprintf(stdout, "Genero il vettore con numeri casuali -> ");
                        dvest(x, n);
                }



                // Calcolo della DFT del vettore con fftw3
                if ( gettimeofday(&start, &tz) != 0 ) {
                        perror("getimeoftime()");
                        return EXIT_FAILURE;
                }

                p = fftw_plan_r2r_1d(n, x, wsave, FFTW_R2HC, FFTW_ESTIMATE);
                fftw_execute(p);
                fftw_destroy_plan(p);

                if ( gettimeofday(&stop, &tz) != 0 ) {
                        perror("gettimofday()");
                        return EXIT_FAILURE;
                }

                fprintf(stdout, "Tempo:    FFTW3 vettore %2d -> ",k);
                stampa_tempo(&start, &stop);
                fprintf(stdout, "\n");

                if (verbose) { // Se e' stata immessa l'opzione "-v"
                        fprintf(stdout, "DFT con FFTW3 -> ");
                        dvest(wsave, n);
                }



                // Calcolo della DFT del vettore con dfftpack
                if ( gettimeofday(&start, &tz) != 0 ) {
                        perror("getimeoftime()");
                        return EXIT_FAILURE;
                }

                dffti_(&n, wsave);
                dfftf_(&n, x, wsave);

                if ( gettimeofday(&stop, &tz) != 0 ) {
```

```
                        perror("gettimofday()");
                        return EXIT_FAILURE;
                }

                fprintf(stdout, "Tempo: DFFTPACK vettore %2d -> ",k);
                stampa_tempo(&start, &stop);
                fprintf(stdout, "\n");

                if (verbose) { // Se e' stata immessa l'opzione "-v"
                        fprintf(stdout, "DFT con DFFTPACK -> ");
                        dvest(x, n);
                }


                fprintf(stdout, "\nPremi INVIO per continuare\n");
                getchar();
        }


        // Libero la memoria usata per i vettori
        free(x);
        free(wsave);


        fprintf(stdout, "Programma terminato... arrivederci!\n");


        return EXIT_SUCCESS;
}
```

Makefile :

```
# Path delle librerie
PATH_LIB = -L/usr/lib/
LIB = -lg2c -ldfftpack -lfftw3


# Decommentare la prossima riga per scegliere un altro compilatore
# CC = gcc


# Nome del file eseguibile da creare
MAIN = eseguibile


# Compilare con 'make DEBUG=yes` se si vogliono aggiungere informazioni per il
# debug all'eseguibile
ifeq ($(DEBUG), yes)
        CFLAG := $(CFLAG) -g
endif


# File sorgenti (.c) e i rispettivi file oggetto (.o)
SOURCES = $(wildcard *.c)
OBJECTS = $(SOURCES:.c=.o)


# Comando per cancellare i file usato durante il target "clean"
DEL = rm -rf


# File momentaneo usato per ricavarsi le dipendenze dei file
DIPENDENZE = .depend


# Comando usato per fare il debug del programma
GDB = gdb

# Target "all" • il target principale.
all : $(DIPENDENZE) $(MAIN)


# Calcolo delle dipendenze per tutti i file dorgenti. Per fare questo si usa
# l'opzione -M del compilatore che produce un output direttamente in formato
# Makefile
$(DIPENDENZE) :
        $(CC) $(CFLAG) -M $(SOURCES) >> $(DIPENDENZE)

# Questo "if" serve a far includere il file con le dipendenze solo quando
# esso • stato effettivamente creato
ifeq ($wildcard $(DIPENDENZE), $(DIPENDENZE))
        include $(DIPENDENZE)
endif

# Linkaggio per avere il file eseguibile
$(MAIN) : $(OBJECTS)
        $(CC) $(CFLAG) -o $(MAIN) *.o $(PATH_LIB) $(LIB)


# Definisco i target PHONY (target che non corrispondono a file)
.PHONY: clean clean_exe clean_dip clean_obj clear run debug


# Pulisce i file dopo la compilazione
clean: clean_exe clean_dip clean_obj


clean_exe:
        $(DEL) $(MAIN)

clean_dip:
        $(DEL) $(DIPENDENZE)

clean_obj:
        $(DEL) *.o


# Cancella i fastidiosi file temporanei che terminano con il carattere '~«
# creati da alcuni editor di testo.
clear:
        $(DEL) *~


# Manda in esecucione il programma
run:
        ./$(MAIN)


# Avvia il debug del programma (se si sceglie questa modalita' bisogna aver
# compilato in precedenza il programma con informazioni per il debug)
debug:
        $(GDB) $(MAIN)
```