ISSN/0393-3016

S. GIUSEPPONI, M. CELINO
Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Casaccia, Roma

G. BRACCO
Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Frascati, Roma

S. MIGLIORI
Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Sede Legale, Roma

PORTING E BENCHMARK IN AMBIENTE MULTIPIATTAFORMA ENEAGRID DI CODICI DI CALCOLO PER APPLICAZIONI SCIENTIFICHE, CON PARTICOLARE RIGUARDO PER QUELLI DEL SETTORE CHIMICA E FISICA DEI MATERIALI

(PARTE I)

RT/2017/12/ENEA

AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE,
L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE

S. GIUSEPPONI, M. CELINO

Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Casaccia, Roma

G. BRACCO

Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Frascati, Roma

S. MIGLIORI

Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Sede Legale, Roma

PORTING E BENCHMARK IN AMBIENTE MULTIPIATTAFORMA ENEAGRID DI CODICI DI CALCOLO PER APPLICAZIONI SCIENTIFICHE, CON PARTICOLARE RIGUARDO PER QUELLI DEL SETTORE CHIMICA E FISICA DEI MATERIALI

(PARTE I)

RT/2017/12/ENEA

AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE,
L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE

PORTING E BENCHMARK IN AMBIENTE ENEAGRID (2006-2007) DI CODICI DI CALCOLO PER APPLICAZIONI SCIENTIFICHE, CON PARTICOLARE RIGUARDO PER QUELLI DEL SETTORE CHIMICA E FISICA DEI MATERIALI

(PARTE I)

S. Giusepponi, M. Celino, G. Bracco, S. Migliori

Riassunto

In questo report descriviamo le principali tematiche affrontate nell'installazione in ambiente multipiattaforma ENEAGRID, di codici di calcolo per applicazioni nel settore della Scienza dei Materiali. Attualmente, l'architettura principalmente utilizzata è Linux x86_64, tuttavia, la descrizione di questo tipo di approccio multipiattaforma, mantiene una sua importanza perché illustra le potenzialità dell'architettura ENEAGRID che permettono flessibilità verso possibili evoluzioni future delle tecnologie per il calcolo scientifico.
Inizialmente, verrà presentata l'installazione delle librerie matematiche BLAS e LAPACK. In seguito, presenteremo le fasi per la compilazione e installazione del programma di dinamica molecolare CPMD. Tale codice verrà utilizzato per lo studio di una interfaccia magnesio-idruro di magnesio, nel contesto dello stoccaggio di idrogeno mediante composti solidi. Infine, verrà testata l'efficienza della parallelizzazione del codice CPMD.

Parole chiave: Codici Scientifici, Calcolo ad Alte Prestazioni, Scienza dei Materiali Computazionale, Stoccaggio di Idrogeno.

Abstract

In this report, we describe the main issues addressed in the installation of computer codes for applications in the field of Materials Science in ENEAGRID. Currently, the main architecture used is Linux x86_64, however, the description of this type of multiplatform approach, remains important because it illustrates the potentialities of the ENEAGRID architecture that allow flexibility in future evolutions of scientific computing technologies.
Initially, the installation steps of the BLAS and LAPACK mathematical libraries are presented. Later, the details for compiling and installing the molecular dynamics program CPMD are given. This code is used for the study of an interface between magnesium hydride and magnesium, in the context of the hydrogen storage in solid compounds. Finally, the parallelization efficiency of code CPMD is tested.

Key words: *Scientific Software, High Performance Computing, Computational Material Science, Hydrogen Storage.*

INDICE

Introduzione

In questo report descriviamo brevemente le principali tematiche affrontate relativamente all'attività di studio avente per oggetto: Porting e benchmark in ambiente ENEAGRID di codici di calcolo per applicazioni scientifiche, con particolare riguardo per quelli del settore Chimica e Fisica dei Materiali. I primi due capitoli saranno dedicati alla presentazione delle librerie matematiche BLAS e LAPACK, e alla loro compilazione su alcune delle macchine presenti nel centro di calcolo ENEA di Frascati. Tali librerie risultano necessarie per la compilazione e l'installazione del programma di dinamica molecolare CPMD, che verrà utilizzato per lo studio di una interfaccia magnesio-idruro di magnesio così come esposto nella Sezione 3 del rapporto. Infine, nell'ultima Sezione, verrà testata la scalabilità della parallelizzazione di questo codice, vedendo come variano le performance di calcolo al variare del numero dei processori utilizzati.

1. Librerie matematiche BLAS

Le librerie BLAS, acronimo del termine inglese *Basic Linear Algebra Subprograms*, sono routine che forniscono i blocchi fondamentali per effettuare le operazioni di base tra vettori e matrici. Grazie all'efficienza, portabilità e facile disponibilità, le BLAS sono comunemente utilizzate nello sviluppo di software di algebra lineare ad alto livello, come ad esempio le librerie LAPACK (cfr. Sezione 2).

1.1. Le librerie BLAS: *Basic Linear Algebra Subprograms*

Le librerie BLAS con la relativa documentazione sono disponibili presso il sito web: http://www.netlib.org/blas/.

Queste routine sono suddivise in tre sottoinsiemi: le BLAS di livello 1 effettuano operazioni che coinvolgono i vettori, come la somma tra vettori o i prodotti vettoriali; le BLAS di livello 2 gestiscono le operazioni tra vettori e matrici, così come nel caso del prodotto tra una matrice e un vettore; infine, le BLAS di livello 3 fanno le operazioni tra matrici, come ad esempio il prodotto matriciale.

È possibile compilare le librerie BLAS in quattro differenti maniere, e cioè, in modo tale che le operazioni vengano fatte in singola o in doppia precisione, sia nel campo dei numeri reali sia in quello dei numeri complessi. Per questi quattro differenti tipi di compilazione, avremo librerie con i suffissi *s* (single) e *d* (double) per numeri reali in singola e doppia precisione; avremo invece i suffissi *c* (complex) e *z* (double complex) per numeri complessi in singola e doppia precisione[1].

Va evidenziato che prima di scaricare e installare le librerie BLAS, è estremamente utile consultare la sezione relativa alle FAQ[2], e in particolare, leggere la risposta alla domanda 5: *"Are optimized BLAS libraries available? Where can I find optimized BLAS libraries?"* dove si suggerisce di utilizzare, quando possibile, le librerie matematiche specifiche per l'architettura sulla quale si intende lavorare, poiché, queste librerie sono

[1] Per l'elenco completo delle BLAS si consulti il file blasqr.pdf disponibile presso la pagina web: http://www.netlib.org/blas/ nella sezione DOCUMENTATION AND TEST SUITES.

[2] http://www.netlib.org/blas/faq.html.

ottimizzate per il particolare tipo di processore in modo da migliorarne le performance di calcolo[3]. Così, ad esempio, su computer IBM è consigliato utilizzare le librerie ESSL (*Engineering and Scientific Subroutine Library*), per i processori AMD sono consigliate le ACML (*AMD Core Math Library*), mentre le librerie MKL (*Math Kernel Library*) sono specifiche per i processori INTEL. Queste librerie ottimizzate, oltre alle routine BLAS, includono anche altre librerie matematiche.

Alternativamente, si possono generare automaticamente le routine BLAS ottimizzate per la propria architettura, scaricando e installando le librerie ATLAS (*Automatically Tuned Linear Algebra Software*) presso il sito web: http://math-atlas.sourceforge.net/, dove sono anche disponibili delle versioni ottimizzate delle BLAS già precompilate per alcune architetture specifiche (http://www.netlib.org/atlas/archives/).

1.2. Scaricare le librerie BLAS

Il software per l'installazione delle BLAS è distribuito mediante un file di archivio compresso del tipo blas.tgz, che contiene i file sorgente, scritti in Fortran77, per la compilazione delle librerie. Il file compresso può essere scaricato mediante il comando:

```
wget http://www.netlib.org/blas/blas.tgz
```

che utilizza wget come software di download.

Consultando la pagina web http://www.netlib.org/blas/, si possono ricavare tutte le informazioni necessarie riguardanti le BLAS; tale pagina risulta suddivisa secondo le seguenti sezioni:

i. *Documentation and Test Suites*: dove sono disponibili le varie documentazioni e le subroutine per testare le librerie BLAS di livello 1, livello 2 e livello 3, in singola e doppia precisione nel campo dei numeri reali e in quello dei numeri complessi;
ii. *Miscellaneous and Auxiliary Routines*: dove sono presenti le subroutine attraverso le quali risalire alle costanti di macchina del processore e altre subroutine ausiliarie;
iii. *Level 1 BLAS routines*: in questa sezione è possibile scaricare, sia singolarmente sia in blocco, le routine BLAS di livello 1 nelle quattro differenti precisioni;
iv. *Level 2 BLAS routines*: sezione analoga alla precedente relativa alle BLAS di livello 2;
v. *Level 3 BLAS routines*: sezione analoga alla precedente relativa alle BLAS di livello 3;
vi. *Extended precision Level 2 BLAS routines*: routine BLAS di livello 2 in extended precision accumulation.

1.3. Installare le librerie BLAS

[3] Per una lista completa delle librerie specifiche per i vari processori, si consulti la pagina web: http://www.netlib.org/blas/faq.html#5, dove ci sono anche i link ai siti dei produttori di processori.

Una volta spostato il file compresso `blas.tgz` nelle propria directory di lavoro, ad esempio `/home/utente/tempwork/`, questo si può scompattare con il comando

```
tar -xvzf blas.tgz
```

che produrrà la cartella `/home/utente/tempwork/BLAS`. In questa cartella sono presenti i file sorgente `*.f` che andranno compilati per creare le librerie. La compilazione avviene mediante l'esecuzione del `Makefile` in base alle impostazioni definite nel file di configurazione `make.inc`, entrambi presenti nella directory `BLAS`.

1.3.1. Modifica del file `make.inc`

Visualizzando con un editor di testo il file `make.inc`, osserviamo che sono presenti delle variabili attraverso le quali definire i parametri della compilazione. Mediante la definizione di `PLAT` è possibile identificare le librerie `BLAS` nel caso in cui si vogliano compilare su differenti piattaforme, infatti il valore assegnato a `PLAT` sarà posto come suffisso al nome della libreria creata. Ponendo ad esempio `PLAT=_Linux-ABC`, verrà creata una libreria con suffisso `_Linux-ABC.a`. Attraverso la definizione di `FORTRAN` e `OPTS` si sceglie il compilatore con le sue opzioni di ottimizzazione, analogamente con `LOADER` e `LOADOPT` si definisce il loader con le sue opzioni, così come con `ARCH` e `ARCHFLAGS` si sceglie il programma di archiviazione delle librerie con le sue flag e con `RANLIB` si definisce il programma di creazione dell'indice di archivio delle librerie. Infine, mediante `BLASLIB` si sceglie il nome da dare alla libreria. Così, avendo posto `PLAT=_Linux-ABC`, se poniamo `BLASLIB=blas$(PLAT).a` verrà creata la libreria `blas_Linux-ABC.a`, mentre per `BLASLIB=libblas$(PLAT).a`, la libreria si chiamerà `libblas_Linux-ABC.a`.

Una volta modificato il file di configurazione `make.inc`, basterà eseguire nella directory `BLAS` il comando

```
make
```

per compilare i vari file sorgente `*.f` nei rispettivi moduli `*.o` che verranno archiviati nella libreria indicata dal file con estensione `.a`. Dopo aver creato la nostra libreria, possiamo cancellare i moduli `*.o` digitando il comando

```
make clean
```

potendo così procedere all'istallazione delle `BLAS` su un'altra piattaforma.

A questo punto procediamo all'istallazione delle librerie `BLAS` sulle differenti strutture di calcolo disponibili nel centro ENEA di Frascati.

1.3.2. Installazione su macchine IBM

Nel Centro di Calcolo dell'ENEA di Frascati sono disponibili due tipi di macchine IBM. Sono presenti 4 nodi SP4 ciascuno costituito da 32 processori Power4 con sistema operativo `AIX 5.2.4` e 13 nodi SP5 con sistema operativo `AIX 5.3.3` per un totale di 256 processori Power5, ripartiti in 12 nodi di 16 processori ciascuno più un nodo di 64 processori. Per un elenco completo delle risorse di calcolo disponibili presso i centri ENEA si consulti la pagina web:

`http://www.afs.enea.it/project/eneagrid/Resources/CompRes.`

Per entrambi questi tipi di macchine abbiamo scelto lo stesso compilatore e le stesse opzioni di ottimizzazione. Il compilatore scelto è quello realizzato dall'IBM, e cioè `xlf`[4]. Analoga è stata anche la scelta del loader e del programma di archiviazione con le relative opzioni.
Si è quindi posto:

```
FORTRAN = xlf
OPTS = -O3 -qstrict -q64 -qmaxmem=-1 -qtune=pwr5 -qarch=pwr5
DRVOPTS = $(OPTS)
NOOPT = -q64
LOADER = xlf
LOADOPTS = -q64
ARCH = ar
ARCHFLAGS= -cr -X64
RANLIB = ranlib
```

nei file di configurazione `make.inc` che riportiamo per comodità nell'Appendice A. Comunque, va fatta una precisazione riguardo alla ulteriore scelta di poter compilare le librerie a 32 bit o a 64 bit, questo si traduce nel mettere o meno l'opzione `-q64` e `-X64` nella lista precedente.
Sulle macchine IBM sono disponibili le librerie proprietarie `ESSL`, specifiche per questi tipi di macchine, che includono anche le `BLAS`. È preferibile fare ricorso a queste librerie ottimizzate in quanto hanno prestazioni notevolmente superiori.

1.3.3. Installazione su macchine LINUX

Tra le risorse di calcolo presenti al centro ENEA di Frascati, sono presenti due infrastrutture di calcolo LINUX con sistema operativo `Scientific Linux`: il cluster bw305 costituito da 16 nodi con processori Intel Pentium IV e la macchina lin4p che utilizza 4 processori Intel Xeon. I compilatori disponibili su queste infrastrutture sono quello `GNU`, `g77`[5] e quello `PGI`, `pgf77`[6].
Nei file di configurazione `make.inc` si è definito in un caso:

```
FORTRAN = g77
OPTS = -funroll-all-loops -O3
DRVOPTS = $(OPTS)
NOOPT =
```

[4] `XL Fortran Enterprise Edition V10.1.`

[5] `GNU project Fortran 77 compiler.`

[6] `The Portland Group Compiler Technology Fortran 77 compiler.`

```
LOADER = g77
LOADOPTS =
ARCH = ar
ARCHFLAGS= cr
RANLIB = ranlib
```

mentre nell'altro:

```
FORTRAN = pgf77
OPTS = -O3
DRVOPTS = $(OPTS)
NOOPT =
LOADER = pgf77
LOADOPTS =
ARCH = ar
ARCHFLAGS= cr
RANLIB = ranlib
```

Va osservato che queste compilazioni sono state eseguite a 32 bit in quanto le macchine bw305 e lin4p hanno architettura `i386`. Poiché, su queste macchine non sono presenti le librerie `MKL` proprietarie INTEL, e quindi ottimizzate per i processori INTEL, non possiamo utilizzare le loro elevate prestazioni al posto delle `BLAS`.

1.3.4. Installazione delle librerie `BLAS` nella cella `AFS`: `/afs/enea.it/`

Le librerie `BLAS` sono state rese disponibili ai vari utenti dei sistemi di calcolo dei centri di ricerca ENEA, collocandole nella cella `AFS` di `enea.it`[7]. Per far ciò è stata creata la directory apposita: `/afs/enea.it/software/libblas/`.

Questa cartella è stata suddivisa nelle seguenti sotto-directory:

- `src`: in questa directory è presente il file sorgete, `blas.tgz`, per le librerie `BLAS`;
- `i386_linux24_glibc23_g77`: in questa cartella è stato scompattato il file `blas.tgz`, e conseguentemente creata la sottodirectory `BLAS` all'interno della quale è stata creata la libreria `blas_bw305_LINUX-g77.a` utilizzando il compilatore `g77` da utilizzare sulla macchina bw305 (lin4p) con sistema operativo `LINUX`. Un procedimento analogo è stato adottato anche nei casi

[7] `AFS` (*Andrew File System*) è un filesystem distribuito che permette di accedere un gran numero di file e directory su macchine diverse in modo uniforme. E' un prodotto proprietario realizzato e distribuito dalla Transarc Corp. (Pittsburg,USA). Per maggiori informazioni si consulti la pagina web: `http://www.afs.enea.it/project/eneagrid/afs/`.

seguenti, dove però, cambia il tipo di architettura e il compilatore utilizzato così come specificato nel nome della directory;

- `i386_linux24_glibc23_pgf77`: in questa cartella abbiamo la libreria `blas_bw305_LINUX-pgf77.a` compilata sulla stessa macchina del caso precedente ma con il compilatore `pgf77`;
- `rs_aix52_pwr4_32_xlf`: in questo caso abbiamo compilato le librerie `BLAS` sulle macchine IBM-SP4 con sistema operativo `AIX`, compilatore `xlf` e architettura a 32 bit. Il file di archivio è `blas_SP4-32xlf.a`;
- `rs_aix52_pwr4_64_xlf`: situazione analoga alla precedente ma con architettura a 64 bit. La libreria è stata chiamata `blas_SP4-64xlf.a`;
- `rs_aix53_pwr5_32_xlf`: in questa cartella abbiamo le librerie `BLAS`, indicate con `blas_SP5-32xlf.a`, compilate a 32 bit su processori IBM-SP5 e sistema operativo `AIX`, utilizzando il compilatore `xlf`;
- `rs_aix53_pwr5_64_xlf`: situazione analoga alla precedente ma con architettura a 64 bit. La libreria è stata indicata con il nome `blas_SP5-64xlf.a`;
- `lib`: in questa cartella sono presenti i link che puntano alle libreria `BLAS` compilate per varie risorse di calcolo;
- `html`: cartella riservata a documenti e risorse da rendere disponibili mediante internet.

2. Librerie matematiche LAPACK

Le librerie LAPACK, acronimo del termine inglese *Linear Algebra PACKage*, sono un insieme di subroutine scritte in Fortran77 realizzate per risolvere numericamente i problemi di algebra lineare più comuni, come ad esempio l'analisi e la soluzione di sistemi di equazioni algebriche lineari e il problema agli autovalori per una matrice.

2.1. Le librerie LAPACK: *Linear Algebra PACKage*

Le librerie LAPACK con la relativa documentazione sono disponibili presso il sito web:
http://www.netlib.org/lapack/.
Tali librerie sono state realizzate in modo tale da essere il più possibile performanti sui computer ad alte prestazioni attualmente disponibili. Il metodo per raggiungere la massima efficienza è basato sull'utilizzo delle librerie BLAS descritte nella sezione precedente, che costituiscono appunto un insieme di subroutine standard per la soluzione di problemi basilari di algebra lineare. Così come per le BLAS, anche le librerie LAPACK possono essere compilate in quattro differenti maniere, ovvero, in singola e doppia precisione, sia nel campo dei numeri reali, sia in quello dei numeri complessi (single, double, single complex e double complex).
Prima di scaricare e installare le LAPACK, è utile consultare la sezione relativa alle FAQ[8]. In particolare segnaliamo la risposta alla domanda 1.1): *"What and where is LAPACK"*, dalla quale è possibile avere informazioni in merito alle versioni commerciali delle librerie matematiche specifiche per i diversi tipi di processori, così come sulle distribuzioni Linux che forniscono le librerie LAPACK mediante pacchetto precompilato. Ripetendo l'osservazione fatta per le BLAS, queste librerie matematiche, essendo specifiche per ciascun processore, ne ottimizzano le performance di calcolo. Perciò, quando è possibile, è preferibile utilizzarle al posto delle LAPACK. Tuttavia, ci sono situazioni nelle quali queste librerie specifiche non sostituiscono completamente le LAPACK, così come nel caso delle librerie ESSL dell'IBM, nelle quali mancano alcune delle routine che costituiscono le LAPACK[9]. In caso contrario si procederà alla compilazione come specificato in seguito.

2.2. Scaricare le librerie LAPACK

Il software per l'istallazione delle librerie LAPACK è distribuito mediante un file di archivio compresso del tipo lapack-number_version.tgz, che contiene i file sorgente sia per le librerie LAPACK che per le librerie BLAS. Inoltre, in questo file vengono forniti i programmi per testare le librerie e, fino alla versione 3.0, misurarne i tempi di esecuzione. Il file compresso è reperibile presso il sito web:
http://www.netlib.org/lapack/

[8] http://www.netlib.org/lapack/faq.html.

[9] Per la lista completa delle routine si consulti la risposta alla domanda 1.15): *"How do I find a particular routine?"*.

così, la versione più recente attualmente disponibile, che è la 3.1.1 rilasciata il 26 febbraio 2007[10], può essere scaricata utilizzando wget

```
wget http://www.netlib.org/lapack/lapack-3.1.1.tgz.
```

Nel sito web www.netlib.org/lapack/ sono reperibili tutte le documentazioni riguardanti l'istallazione e l'utilizzo delle librerie LAPACK. Tale sito risulta suddiviso secondo le seguenti sezioni:

- Available Software: sono disponibili i pacchetti contenenti i file sorgente per la compilazione delle librerie LAPACK;
- Individual Routines: sono disponibili singolarmente tutte le routine che costituiscono le librerie LAPACK. Ovvero le routine matematiche (single, double, single complex e double complex), quelle di utilità, quelle per effettuare i test e quelle per misurare i tempi di esecuzione;
- Documentation: è disponibile la documentazione riguardante l'istallazione e l'utilizzo delle librerie LAPACK[11];
- LAPACK Working Notes in ps/pdf: vengono fornite le *Working Notes*;
- Related Projects: vendono elencati i progetti collegati alle librerie LAPACK.

2.3. Installare le librerie LAPACK

Dopo aver scaricato il file compresso nella propria home o directory di lavoro, lo possiamo scompattare mediante il comando

```
tar -xvzf lapack-number_version.tgz
```

che genererà una directory del tipo lapack-number_version che risulta suddivisa nelle cartelle BLAS, html, INSTALL, manpages, SRC e TESTING. Nella directory lapack-number_version è presente il file Makefile che effettua l'intera procedura di installazione sulla base del file di configurazione make.inc, del quale viene fornita la versione di esempio make.inc.example. Nella directory lapack-number_version/INSTALL sono presenti alcuni file di configurazione (make.inc.xxx) specifici per alcune piattaforme[12].

---

[10] La prima versione risale al febbraio del 1992. Per avere maggiori informazioni sull'ultima versione rilasciata, si consulti la pagina http://www.netlib.org/lapack/revisions.info.

[11] Risultano particolarmente utili le seguenti referenze:

LAPACK Working Note 81. Quick Installation Guide for LAPACK on Unix Systems;

LAPACK Working Note 41. Installation Guide for LAPACK;

LAPACK Users' Guide.

[12] Si faccia riferimento all'appendice *"Caveats"* presente come app. A in LAPACK Working Note 81 ovvero come app. D in LAPACK Working Note 41 e anche al release_notes file (*Machine-specific Installation Hints*) in netlib all'indirizzo:

http://www.netlib.org/lapack/release_notes.html.

L'installazione delle Librerie LAPACK avverrà in due fasi, nella prima modificando il file `make.inc.example` sulla base del tipo di macchina e architettura che si ha a disposizione e salvandolo come file `make.inc`, mentre nella seconda fase si modificherà il `Makefile` stesso.

2.3.1. Modifica del file `make.inc`

Prima di compilare e testare le librerie bisogna definire i parametri di installazione per il tipo di macchina sulla quale installare le `LAPACK`. Come detto, tali parametri sono specificabili nel file di configurazione `LAPACK/make.inc`. Tramite un qualsiasi editor di testo possiamo aprire il file `make.inc.example`. In questo file è presente la variabile `PLAT` mediante la quale è possibile identificare le librerie `LAPACK` nel caso in cui si vogliano compilare su differenti piattaforme. Il valore assegnato a `PLAT` sarà posto come suffisso al nome della libreria creata, così ad esempio, ponendo `PLAT=_Linux-ABC`, verrà creata una libreria `lapack_Linux-ABC.a`. Attraverso la definizione delle variabili `FORTRAN` e `OPTS` viene scelto il compilatore da utilizzare con le relative opzioni di compilazione; con `DRVOPTS` si definiscono le opzioni di compilazione per i programmi utilizzati per testare e misurare i tempi di esecuzione, mentre con `NOOPT` si scelgono le opzioni per le compilazioni senza ottimizzazioni. Con `LOADER` e `LOADOPTS` si definisce il loader con le sue opzioni, così come con `ARCH` e `ARCHFLAGS` si sceglie il programma di archiviazione delle librerie con le sue *flag* e con `RANLIB` si definisce il programma di creazione dell'indice di archivio delle librerie. Infine nell'ultima parte del file sono indicate la collocazione delle librerie alle quali fare riferimento. È importante sottolineare che, ogni qualvolta è possibile, dovrebbero essere considerate le librerie `BLAS` ottimizzate per lo specifico tipo di macchina utilizzata.

2.3.2. Modifica del file `Makefile`

Analogamente al caso delle librerie `BLAS`, l'installazione delle librerie `LAPACK` avverrà mediante il comando `make`. In questo caso però, essendo presenti differenti target nel `Makefile`, possiamo anche procedere ad una installazione a passi successivi. Riportando di seguito `Makefile`,

```
#
# Top Level Makefile for LAPACK
# Version 3.1.1
# February 2007
#
include make.inc

all: lapack_install lib lapack_testing blas_testing

lib: lapacklib tmglib
#lib: blaslib lapacklib tmglib
```

```
clean: cleanlib cleantesting cleanblas_testing

lapack_install:
      ( cd INSTALL; $(MAKE); ./testlsame; ./testslamch; \
       ./testdlamch; ./testsecond; ./testdsecnd; ./testversion )

blaslib:
      ( cd BLAS/SRC; $(MAKE) )

lapacklib: lapack_install
      ( cd SRC; $(MAKE) )

tmglib:
      ( cd TESTING/MATGEN; $(MAKE) )

lapack_testing: lib
      ( cd TESTING ; $(MAKE) )
blas_testing:
      ( cd BLAS/TESTING; $(MAKE) -f Makeblat1 )
      ( cd BLAS; ./xblat1s > sblat1.out ; \
                 ./xblat1d > dblat1.out ; \
                 ./xblat1c > cblat1.out ; \
                 ./xblat1z > zblat1.out )
      ( cd BLAS/TESTING; $(MAKE) -f Makeblat2 )
      ( cd BLAS; ./xblat2s < sblat2.in ; \
                 ./xblat2d < dblat2.in ; \
                 ./xblat2c < cblat2.in ; \
                 ./xblat2z < zblat2.in )
      ( cd BLAS/TESTING; $(MAKE) -f Makeblat3 )
      ( cd BLAS; ./xblat3s < sblat3.in ; \
                 ./xblat3d < dblat3.in ; \
                 ./xblat3c < cblat3.in ; \
                 ./xblat3z < zblat3.in )

cleanlib:
      ( cd INSTALL; $(MAKE) clean )
      ( cd BLAS/SRC; $(MAKE) clean )
      ( cd SRC; $(MAKE) clean )
```

```
	( cd TESTING/MATGEN; $(MAKE) clean )

cleanblas_testing:
	( cd BLAS/TESTING; $(MAKE) -f Makeblat1 clean )
	( cd BLAS/TESTING; $(MAKE) -f Makeblat2 clean )
	( cd BLAS/TESTING; $(MAKE) -f Makeblat3 clean )
	( cd BLAS; rm -f xblat* )

cleantesting:
	( cd TESTING/LIN; $(MAKE) clean )
	( cd TESTING/EIG; $(MAKE) clean )
	( cd TESTING; rm -f xlin* xeig* )
cleanall: cleanlib cleanblas_testing cleantesting
	rm -f *.a TESTING/*.out INSTALL/test* BLAS/*.out
```

si vede come l'installazione completa avvenga così come elencato nel target `all: lapack_install lib lapack_testing blas_testing`. Quindi, inizialmente (`lapack_install`) verranno eseguiti dei test e le installazioni delle routine dipendenti dalle specifiche del processore; in seguito (`lib`) verrano compilate le librerie `LAPACK` (`lapacklib`) e le librerie `Test Matrix Generator Library` (`tmglib`) ed infine verranno eseguiti i test per le `LAPACK` (`lapack_testing`) e le `BLAS` (`blas_testing`). Va osservato che nel nostro caso, avendo compilato le `BLAS` in precedenza, abbiamo tolto dal target `lib` l'argomento `blaslib`. Inoltre, quando verrà effettuato il test delle librerie `BLAS`, queste verranno cercate sulla base della variabile `BLASLIB` definita nel file `make.inc`. Per procedere all'installazione delle librerie per un altro tipo di configurazione, si dovranno cancellare i moduli compilati mediante il comando `make clean` e ripetere la procedura precedente con le opportune modifiche.

2.3.3. Compilazione delle librerie LAPACK su macchine IBM e LINUX

La scelta dei parametri di installazione delle librerie `LAPACK` sulle macchine con processori IBM e su quelle con processori INTEL sarà del tutto analoga a quella fatta in precedenza per le `BLAS`. Avremo così ad esempio stessi compilatori con stesse opzioni di ottimizzazione. Va precisato però, che in questo caso, nel file `make.inc` bisogna specificare nella variabile `BLASLIB`, la versione e locazione della libreria `BLAS` appropriata. Ad esempio, compilando con `xlf` a 64 bit per processori SP5, porremo `BLASLIB=-L/afs/enea.it/software/OSafs/lib -lblas64p5xlf`. Mentre nel caso di compilazione con `g77` per il cluster LINUX bw305, definiremo nel file `make.inc`: `BLASLIB=-L/afs/enea.it/software/OSafs/lib/ -lblaslinuxg77` in modo da fornire il link della libreria appropriata. In ogni caso, nell'Appendice B sono specificati per tutte le varie configurazioni i file `make.inc`.

Analogamente al caso delle BLAS, anche le librerie LAPACK sono state rese disponibili ai vari utenti dei sistemi di calcolo dei centri di ricerca ENEA, collocandole nella cella AFS di enea.it. In questo caso abbiamo creato l'apposita directory: /afs/enea.it/software/liblapack che è stata suddivisa in modo del tutto uguale al caso precedente nelle sotto-directory: src, lib, html, i386_linux24_glibc23_g77, i386_linux24_glibc23_pgf77, rs_aix52_pwr4_32_xlf, rs_aix52_pwr4_64_xlf, rs_aix53_pwr5_32_xlf, rs_aix53_pwr5_64_xlf.

La necessità di dover installare queste librerie matematiche è dovuta al fatto che sono necessarie per poter utilizzare il codice di simulazione numerica CPMD per lo studio di una interfaccia magnesio - idruro di magnesio così come esposto nella sezione seguente.

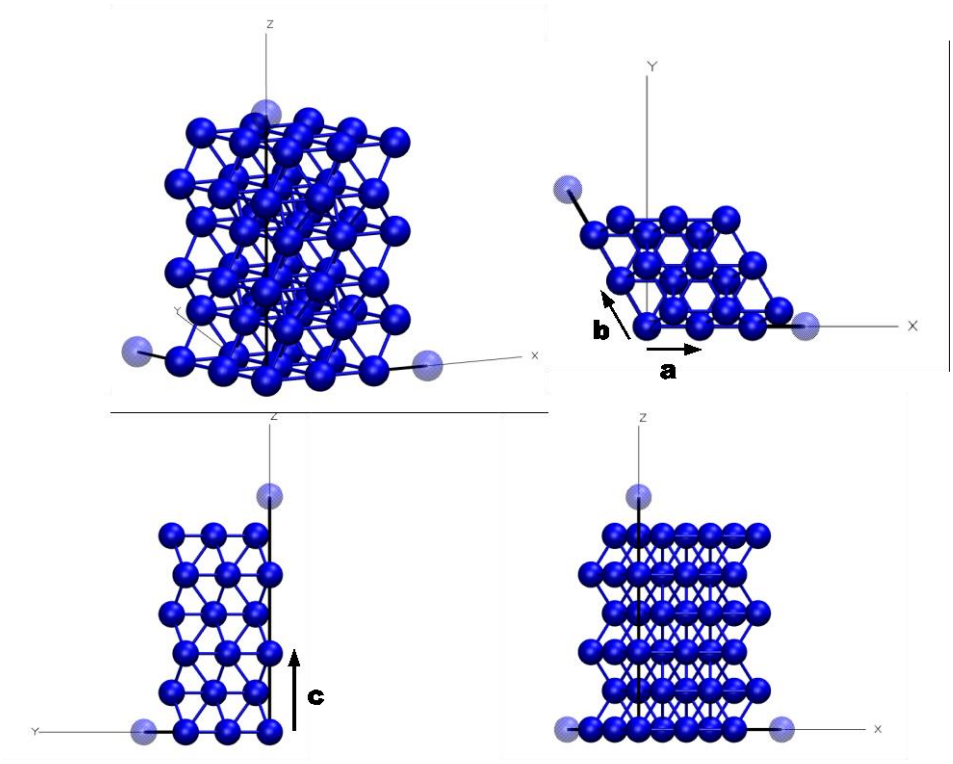

Figura1: Struttura esagonale compatta per il magnesio con proiezioni sui piani xy, yz e xz, dove sono indicati i vettori **a**, **b** e **c**.

3. Studio di una interfaccia Magnesio - Idruro di Magnesio

Attualmente, uno degli argomenti di maggiore interesse nell'ambito della ricerca di nuove tecnologie mirate alla riduzione dell'impiego di combustibili di origine fossile, responsabili in larga misura della produzione di gas inquinanti, consiste nell'utilizzo dell'idrogeno per la produzione di energia. Tra i principali ostacoli per la diffusione di questo tipo di combustibile, ci sono le difficoltà legate al modo in cui poterlo stoccare e distribuire in modo sicuro e conveniente. Una soluzione è rappresentata dalla nota capacità del magnesio di assorbire e desorbire idrogeno. Sperimentalmente, si stanno svolgendo numerosi lavori che hanno come oggetto miscele di polveri di idruro di magnesio con magnesio puro. Fornendo una migliore comprensione teorica di tali lavori sperimentali, si avrà come diretta conseguenza applicativa, una diminuzione dei costi nelle realizzazioni tecnologiche di questa tecnica di stoccaggio dell'idrogeno.

In questa ottica abbiamo iniziato uno studio nel quale ci prefiggiamo di analizzare il comportamento di una interfaccia tra una superficie di magnesio e una di idruro di magnesio, al fine di comprendere da un punto di vista teorico-computazionale, i meccanismi attraverso i quali questi due componenti interagiscono, ed in particolare ricercare eventuali processi di diffusione dell'idrogeno mediante l'interfaccia. Per la parte computazionale è stato utilizzato il codice di simulazione numerica CPMD (*Car-Parrinnello Molecular Dynamics*)[13].

[13] Per approfondimenti si consulti il sito web: `http://www.cpmd.org`.

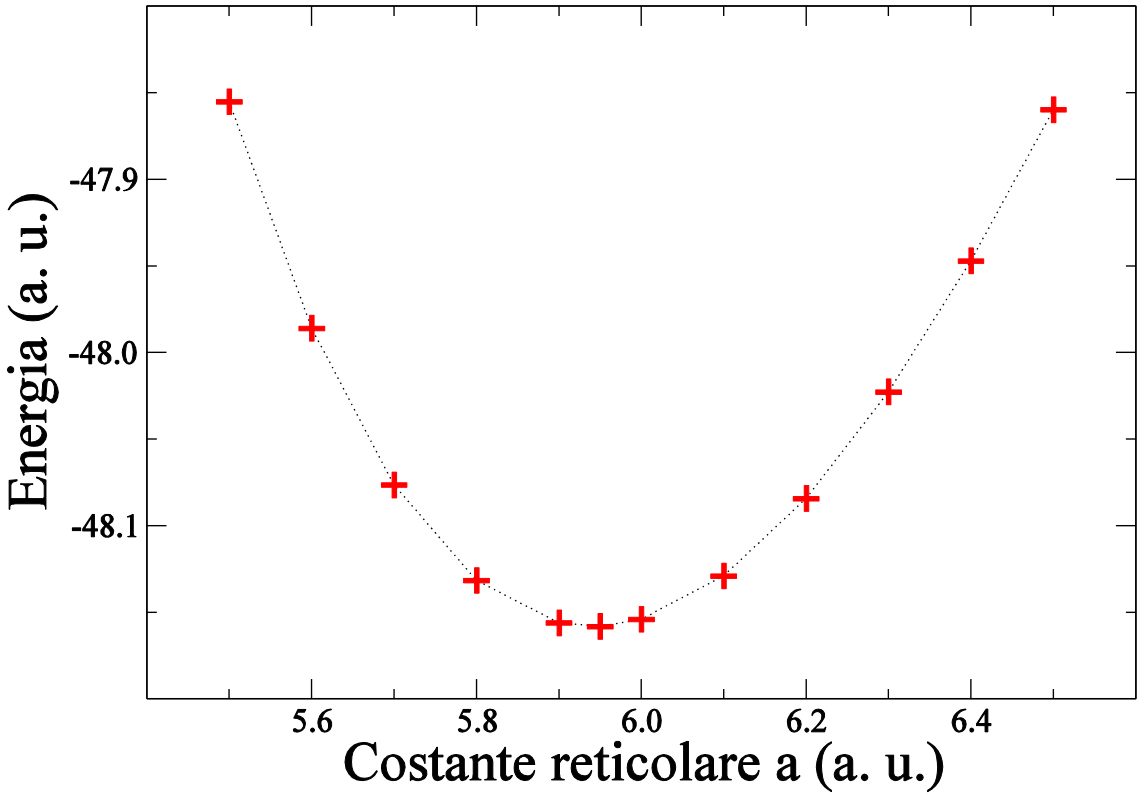

Figura2: Andamento dell'energia totale della struttura cristallina del magnesio in funzione del valore del modulo del vettore **a**.

Inizialmente sono stati studiati separatamente i due componenti; in particolare si sono cercati i parametri reticolari che meglio si adattano al particolare tipo di pseudopotenziale utilizzato per studiare tali elementi. In seguito si è proceduto all'analisi diretta dell'interfaccia.

3.1. Reticolo di Magnesio

La struttura cristallina del magnesio (Mg) è quella esagonale compatta (hpc). In questa struttura, se consideriamo gli strati di atomi di magnesio, questi si dispongono in modo tale da coincidere con i vertici e il centro di un esagono, ed inoltre, considerando strati successivi di atomi, ciascun atomo è circondato da sei atomi posizionati al centro di un prisma retto con base triangolare. Si veda la Figura1 per una rappresentazione grafica di tale struttura cristallina. Il reticolo cristallino del magnesio è costruito considerando i tre vettori **a**, **b** e **c**, per i quali sono soddisfatte le seguenti condizioni: **a** e **b** sono vettori che hanno lo stesso modulo e formano tra loro un angolo di 120° e **c** è un vettore perpendicolare al piano individuato dagli altri due vettori. Perciò per definire il reticolo sarà sufficiente specificare il modulo a del vettore **a** (ovvero **b**) e quello del vettore **c** (oppure del rapporto c/a tra i moduli di **c** e **a**). La struttura cristallina si realizza aggiungendo ad ogni punto del reticolo la base, che nel caso del magnesio è costituita da due atomi di magnesio. Per individuare i parametri reticolari che meglio si adattano al particolare tipo di pseudopotenziale utilizzato per studiare il magnesio, abbiamo considerato la cella di simulazione mostrata nella Figura 1 nella quale sono contenuti 54 atomi di magnesio. Avendo fissato il rapporto c/a = 1,63 abbiamo fatto variare il valore numerico assegnato ad a. In Figura2, è mostrato come varia l'energia totale della struttura cristallina del magnesio, in funzione del valore

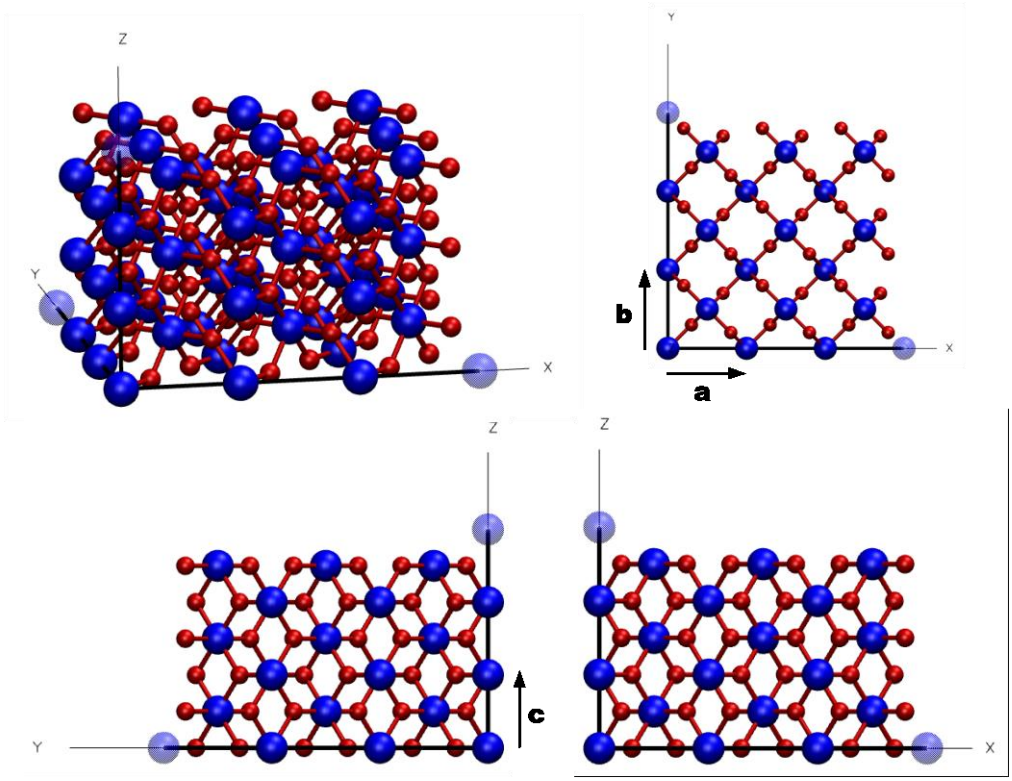

Figura3: Struttura tetragonale per l'idruro di magnesio, con proiezioni sui piani xy, yz e xz, dove sono indicati i vettori **a**, **b** e **c**. Gli atomi di magnesio sono indicati dalle sfere di colore blu, mentre quelli di idrogeno sono rappresentati dalle sfere di colore rosso.

assunto dalla lunghezza del vettore **a**. Da questa curva si ricava che il valore di a per il quale si ha il minimo di energia è pari a: a = 5,95 u.a.

3.2. Reticolo di Idruro di Magnesio

In questa sezione faremo uno studio analogo a quello già fatto nella sezione precedente, solamente che ora, considereremo l'idruro di magnesio (MgH_2) al posto del magnesio. Per questo tipo di idruro dovremo considerare una struttura cristallina tetragonale, nella quale i vettori **a**, **b** e **c** sono ortogonali tra loro e con l'ulteriore vincolo di avere i moduli di **a** e **b** uguali. Tale struttura è mostrata nella Figura3 assieme alle sue proiezioni sui piani cartesiani. Anche in questo caso la struttura cristallina sarà realizzata aggiungendo in ogni punto del reticolo la base che, nel caso dell'idruro di magnesio, è costituita da due atomi di magnesio e quattro di idrogeno.

In questo caso, per individuare i parametri reticolari che meglio descrivono la struttura cristallina dell'MgH_2, dobbiamo scegliere i valori numerici da assegnare al modulo di **a** e a quello di **c**. Ciò è stato fatto cercando il minimo del valore dell'energia al variare di a e c, considerando la cella di simulazione mostrata nella Figura3 nella quale sono contenuti 54 atomi di magnesio e 108 di idrogeno. Analizzando la Figura4, dove sono mostrati i valori dell'energia totale della struttura cristallina, ottenuti dalle diverse simulazioni al variare dei parametri reticolari, si deduce che i valori numerici da assegnare al modulo del vettore **a** e al modulo del vettore **c** sono rispettivamente: a = 8,3 u.a. e c = 5,59 u.a.

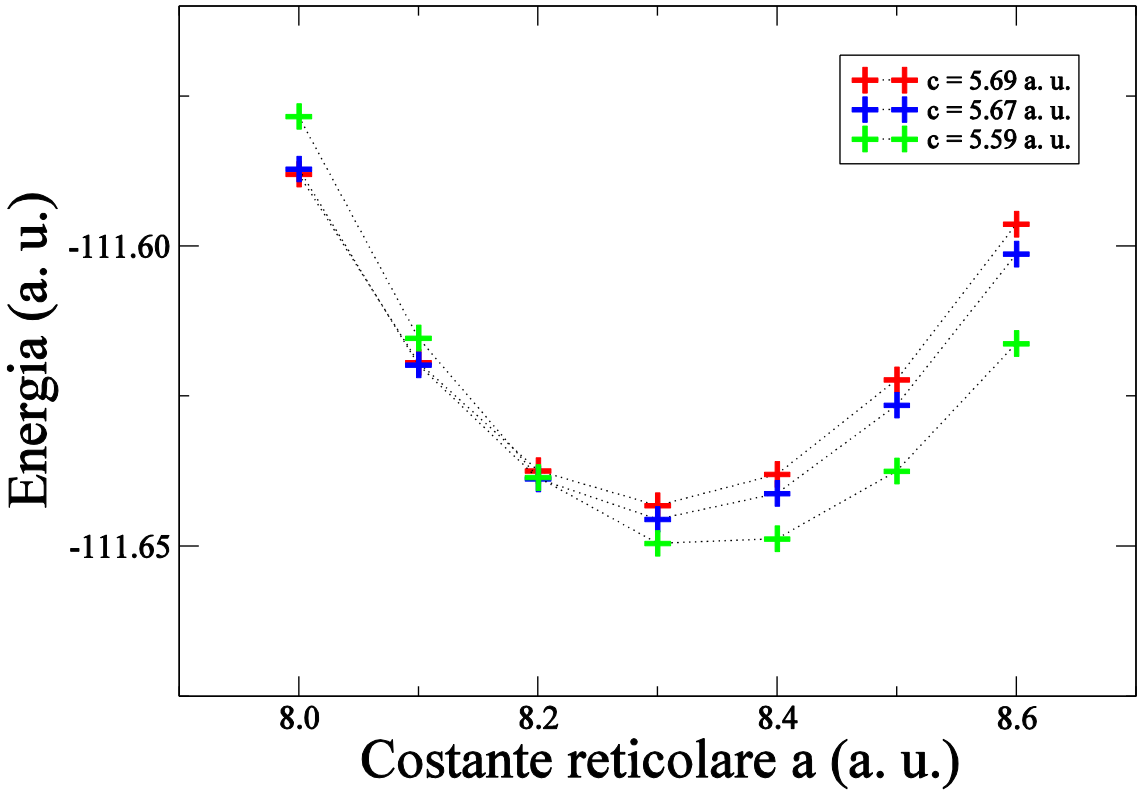

Figura4: Andamento dell'energia totale del cristallo di idruro di magnesio in funzione del valore del modulo del vettore **a** per tre differenti valori del modulo del vettore **c**.

3.3. Interfaccia Magnesio - Idruro di Magnesio

Dopo aver ricercato i valori dei parametri reticolari che meglio descrivono le strutture cristalline del magnesio e dell'idruro di magnesio, abbiamo ricercato i piani lungo i quali tagliare questi cristalli. Ciò è stato fatto in modo da ottenere una superficie di magnesio e una di idruro dalle dimensioni commensurabili, così da poterle inserirle in un'unica cella di simulazione ortorombica rispettando le condizioni periodiche al contorno nelle direzioni parallele alle superfici. Si è così costruito un'interfaccia tra questi due cristalli.

Tagliamo il reticolo di magnesio secondo il piano (010), che risulta parallelo al piano cartesiano xz (si veda Figura1), e consideriamo il rettangolo di lati:

$$A_{Mg} = 2a_{Mg} = 2 \times 5{,}95 \text{ a.u.} = 11{,}9 \text{ a.u.}$$

$$B_{Mg} = 3c_{Mg} = 3 \times 1{,}63 \times 5{,}95 \text{ a.u.} = 29{,}0955 \text{ a.u.}$$

Per il reticolo di idruro di magnesio, consideriamo il piano (110) (si veda Figura3) in modo da poter definire il rettangolo di lati:

$$A_{MgH2} = a_{MgH2} \times \sqrt{2} = 8{,}3 \times \sqrt{2} \text{ a.u.} = 11{,}738 \text{ a.u.}$$

$$B_{MgH2} = 5 \times c_{MgH2} = 5 \times 5{,}59 \text{ a.u.} = 27{,}95 \text{ a.u.}$$

Confrontando le dimensioni di questi due rettangoli, risulta che lungo la prima direzione c'è un accordo nell'ordine del 1%, mentre lungo l'altra direzione, la differenza tra le lunghezze dei lati dei rettangoli è minore del 4%. Riduciamo le dimensioni del reticolo del magnesio in modo tale che i lati dei due rettangoli coincidano così da poter utilizzare una supercella di simulazione ortorombica.

Possiamo perciò costruirci un'interfaccia nella quale una superficie di magnesio si affaccia ad una superficie di idruro di magnesio. Affinché il sistema possa essere ripetuto nello spazio, lo racchiudiamo in una cella di

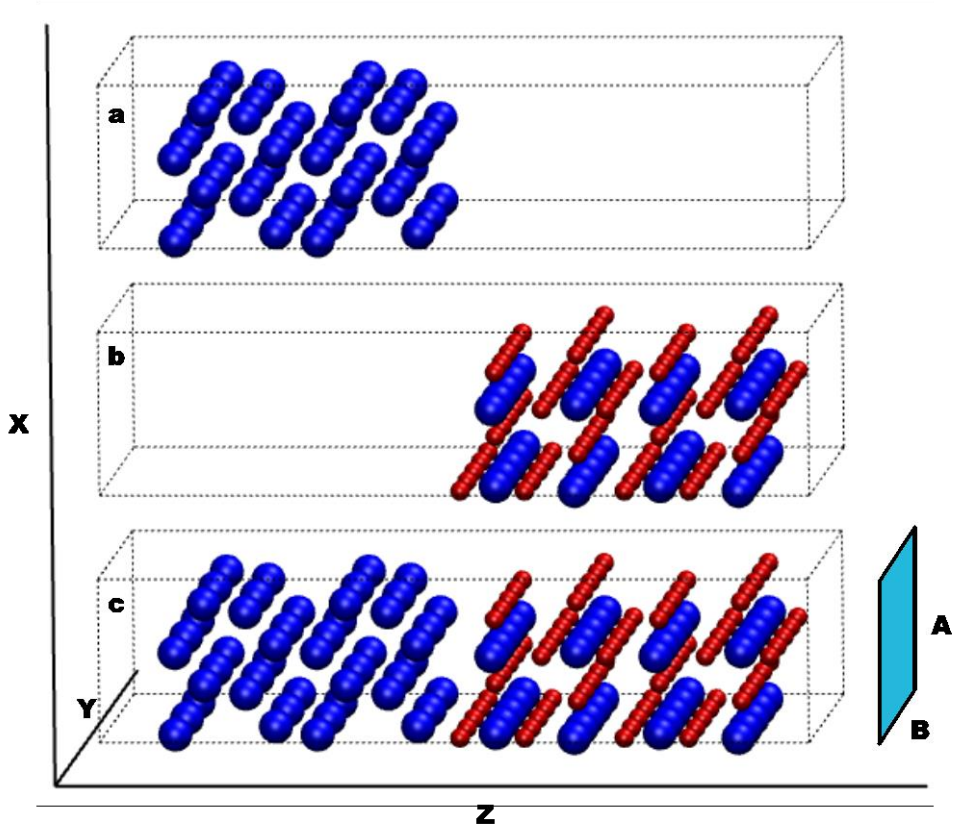

Figura5: Celle di simulazione nelle quali sono contenute: a) il magnesio con le superfici libere; b) l'idruro di magnesio con le superfici libere; c) l'interfaccia magnesio-idruro di magnesio.

simulazione con base il rettangolo di lati A = 11,738 a.u. e B = 27,95 a.u. Si veda la Figura5 per la rappresentazione grafica.

Consideriamo inizialmente la parte dell'interfaccia costituita dal magnesio. Abbiamo preso il reticolo di magnesio e lo abbiamo tagliato lungo due piani paralleli al piano (010), in modo tale da ottenere uno strato infinitamente esteso lungo le direzioni x e y, con due superfici libere ortogonali alla direzione z. La cella di simulazione utilizzata è indicata dalla linea tratteggiata di Figura5a; ha per base il rettangolo di lati A e B e contiene 48 atomi di magnesio. La simulazione numerica di questo sistema, mediante il codice CPMD, ci ha fornito i seguenti valori per le energie: i alla terza direzione (si veda la Figura5b). Nella cella di simulazione utilizzata per calcolare le varie energie di questo sistema, sono contenuti 40 atomi di magnesio ed 80 di idrogeno. La simulazione numerica ci ha fornito i seguenti valori per le energie:

```
*****************************************************************
(F+E2+X-V) TOTAL ENERGY = -42.64955102 A.U.
(F) ELECTRONIC FREE ENERGY = -12.34195912 A.U.
(E2=I-H-S+R) ELECTROSTATIC ENERGY = -36.49321938 A.U.
(S) ESELF = 63.83076486 A.U.
(R) ESR = .00016926 A.U.
(X) EXCHANGE-CORRELATION ENERGY = -20.54858216 A.U.
```

```
(V) EXCHANGE-CORRELATION POTEN. = -26.73420965 A.U.
************************************************************
```

In modo del tutto analogo, procediamo per la parte riguardante l'idruro di magnesio. Consideriamo cioè il reticolo di MgH_2 tagliato secondo due piani paralleli al piano (110). Otteniamo perciò uno strato infinitamente esteso nelle direzioni x e y, con due superfici libere ortogonali alla terza direzione
(si veda la Figura5b). Nella cella di simulazione utilizzata per calcolare le varie energie di questo sistema, sono contenuti 40 atomi di magnesio ed 80 di idrogeno. La simulazione numerica ci ha fornito i seguenti valori per le energie.

```
************************************************************
(K+E1+L+N+X) TOTAL ENERGY = -82.49765116 A.U.
(K) KINETIC ENERGY = 60.43437717 A.U.
(E1=A-S+R) ELECTROSTATIC ENERGY = -60.16472034 A.U.
(S) ESELF = 79.78845608 A.U.
(R) ESR = .31510704 A.U.
(L) LOCAL PSEUDOPOTENTIAL ENERGY = -43.32600543 A.U.
(N) N-L PSEUDOPOTENTIAL ENERGY = 6.14824602 A.U.
(X) EXCHANGE-CORRELATION ENERGY = -45.58954857 A.U.
************************************************************
```

Infine, prendiamo in considerazione l'interfaccia magnesio-idruro di magnesio. Tale interfaccia, rappresentata in Figura5c, è costruita avvicinando gli strati di magnesio e idruro di magnesio precedentemente descritti. Al variare della distanza tra i due strati, si ottengono differenti valori per l'energia totale del sistema, come mostrato in Figura6. Questa assume il suo minimo quando la distanza risulta essere pari a A/2. Di seguito, assieme al valore dell'energia totale, riportiamo gli altri valori delle energie ricavate dalla simulazione numerica per questo particolare valore della distanza tra le superfici.

```
************************************************************
(K+E1+L+N+X) TOTAL ENERGY = -125.24018375 A.U.
(K) KINETIC ENERGY = 78.61300302 A.U.
(E1=A-S+R) ELECTROSTATIC ENERGY = -103.93810553 A.U.
(S) ESELF = 143.61922094 A.U.
(R) ESR = .31766490 A.U.
(L) LOCAL PSEUDOPOTENTIAL ENERGY = -48.22533427 A.U.
(N) N-L PSEUDOPOTENTIAL ENERGY = 14.67922793 A.U.
(X) EXCHANGE-CORRELATION ENERGY = -66.36897491 A.U.
************************************************************
```

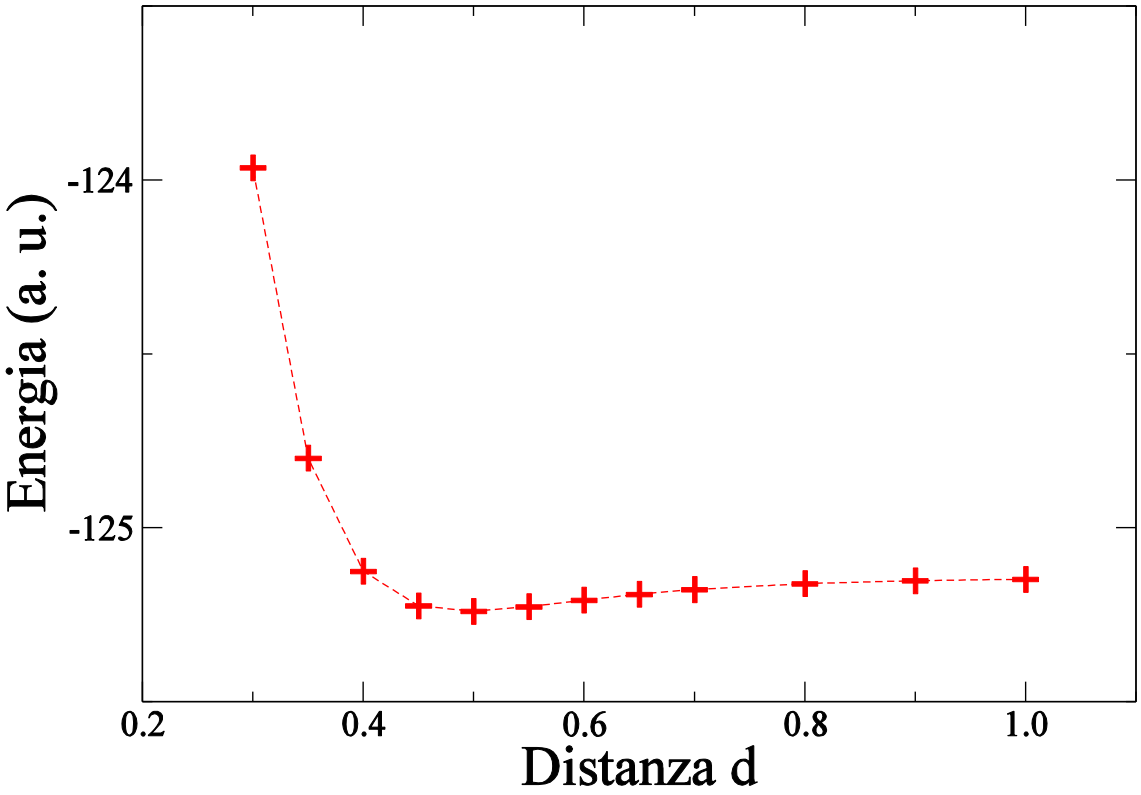

Figura6: Valore dell'energia totale al variare della distanza tra la superficie di magnesio e quella di idruro di magnesio. La distanza è espressa in unità di A.

La raccolta di questi dati ci permette di calcolare il valore dell'energia di adesione dell'interfaccia γ, definita secondo la formula:

$$\gamma = \frac{E_{int} - E_{sup}^{Mg} - E_{sup}^{MgH2}}{A \times B}$$

dove, viene calcolata la differenza di energia tra l'interfaccia e le configurazioni con i singoli costituenti, per poi essere divisa per il valore dell'area della superficie di contatto. Sostituendo i valori numerici otteniamo:

$$\gamma = \frac{-125{,}24018375 \; + \; 42{,}64955102 \; + \; 82{,}49765116}{11{,}738 \times 27{,}95} \frac{Ha}{Bohr^2} = -0{,}000283414 \frac{Ha}{Bohr^2}$$

che convertendo in unità di misura piú convenienti ci forniscono i seguenti risultati:

$$\gamma = -0{,}02754 \frac{eV}{A^2} = -441{,}4753 \frac{mJ}{m^2}$$

In conclusione la configurazione nella quale si realizza l'interfaccia è energeticamente più favorevole rispetto alla configurazione nella quale sono presenti il solo magnesio e il solo idruro di magnesio separati a distanza infinita e con le con le superfici libere.

4. Analisi scalabilità del codice CPMD

Nella sezione precedente abbiamo studiato l'interfaccia magnesio - idruro di magnesio calcolandone il lavoro di adesione. Per avere una stima migliore di tale energia, e procedere ad uno studio dell'interfaccia a differenti temperature, per analizzare la mobilità dell'idrogeno ed evidenziarne eventuali processi diffusivi, sarebbe più indicato considerare un sistema più grande. Incrementiamo perciò il numero di strati di magnesio da un lato e di idruro di magnesio dall'altro. Consideriamo un sistema pari ad una volta e mezzo quello rappresentato nella Figura5c, aumentando cioè, di due strati ciascun lato dell'interfaccia, in modo da avere complessivamente 72 atomi di magnesio da un lato e 60 "molecole" di idruro di magnesio dall'altro.

Dovendo fare uno studio intensivo di questo sistema, vogliamo vedere quali siano le migliori condizioni di lavoro, ovvero, come sfruttare al meglio le risorse di calcolo disponibili. Per fare ciò, abbiamo analizzato al variare del numero dei processori, i tempi di esecuzione, le funzioni utilizzate e il peso dei processi di comunicazione, sia nel caso dell'ottimizzazione delle funzioni d'onda sia nel caso di simulazioni di dinamica molecolare. Di seguito riportiamo i risultati ottenuti utilizzando i nodi IBM da 16 processori SP5.

4.1. Ottimizzazione funzioni d'onda

Per vedere la scalabilità del nostro sistema al crescere del numero n di processori usati per eseguire in parallelo il codice CPMD, abbiamo considerato i tempi di esecuzione necessari per calcolare l'ottimizzazione del set di onde piane utilizzate per descrivere il sistema fisico, impiegando rispettivamente 4, 8, 16, 32, 48 e 64 processori IBM-SP5[14].

Nel grafico in alto di Figura7 sono riportati i tempi di esecuzione totale per l'ottimizzazione della funzione d'onda (comprensivi del tempo di inizializzazione, di quello di calcolo e di quello di scrittura). Da tale grafico si vede solamente che all'aumentare di n diminuisce il tempo di esecuzione, ma non si hanno informazioni in merito all'efficacia della parallelizzazione. Questa informazione può essere ricavata considerando lo *speed up*, ovvero, il rapporto tra il tempo di esecuzione del codice parallelo e quello del codice seriale (*Ts/Tn*), ciò è mostrato nel secondo grafico di Figura7. La linea nera tratteggiata indica lo *speed up* ideale, che rappresenta la condizione nella quale utilizzando n processori, si ha una riduzione del tempo di esecuzione pari ad un fattore n rispetto al tempo del codice seriale ($Tn = Ts/n$). Dal grafico si vede che siamo vicini a questa condizione ideale fino a quando n = 32, quando cioè si impiegano due nodi interi, infatti per questa scelta siamo intorno all'80% dello *speed up* ideale. Invece, aumentando il numero di processori pur diminuendo i tempi di esecuzione, le potenzialità della parallelizzazione vengono sfruttate sempre di meno.

Per comprendere le ragioni per le quali si ha un progressivo scostamento dalla condizione ideale, all'aumentare del numero di processori usati, bisogna analizzare i grafici di Figura8. In questa figura sono prese in considerazione le funzioni maggiormente utilizzate durante l'esecuzione dell'ottimizzazione delle funzioni d'onda mediante CPMD. Queste funzioni sono quelle legate al calcolo diretto (`FWFFT`) e inverso (`INVFFT`) delle trasformate di Fourier, e sono quelle legate alla comunicazione (`FFTCOM`) tra i processori di tali quantità,

[14] Il benchmark è stato fatto utilizzando sempre il minor numero di nodi, e quindi, un unico nodo nel caso di 4, 8 e 16 processori, mentre ne sono stati utilizzati rispettivamente due, tre e quattro negli altri casi.

che deve essere fatta "impacchettandole" opportunamente (`FFT G/S`). Nel primo grafico vengono mostrati i tempi assoluti impiegati in ciascuna funzione al variare di n, mentre nel secondo, per valutare il peso relativo di ciascuna funzione, viene mostrato il rapporto tra questi tempi e il tempo totale di esecuzione *Tn*. Da questi grafici si vede come siano presenti due situazioni concorrenti. Da un lato si osserva che il tempo impiegato per calcolare le trasformate di Fourier dirette ed inverse, e quello necessario per il loro "impacchettamento", diminuisce al crescere di n, rimanendo sempre intorno al 10 - 20% del tempo totale. Dall'altro, si evidenzia che, per quanto riguarda i tempi necessari allo scambio tra i vari processori dei valori delle trasformate, dopo una fase decrescente, dove si arriva a tempi di comunicazione trascurabili, quando si utilizza un solo nodo (n = 16), segue un incremento dei tempi utilizzati per lo scambio dei dati, che diviene predominante nel caso in cui a comunicare siano i processori di tre e quattro nodi. Dal secondo grafico si vede che in questa ultima situazione, il peso della funzione `FFTCOM` è intorno al 40 - 45%; quasi la metà del tempo totale e impiegata per far comunicare i processori tra di loro.

Riassumendo possiamo dire che, finché viene utilizzato un solo nodo (n uguale a 4, 8 e 16), ai vari processori viene destinato un carico di calcolo abbastanza consistente, tale da ridurre al minimo gli scambi di dati tra di loro, e inoltre, gli scambi di informazioni avvengono in modo molto veloce in quanto all'interno dello stesso nodo. Viceversa, nel caso in cui si impieghino molti processori, il carico computazionale di ciascun processore diviene meno rilevante; ciascun processore deve calcolare un minor numero di trasformate di Fourier che vengono anche impacchettato più velocemente, però, per poter calcolare tali trasformate ha bisogno di scambiare informazioni con gli altri processori. Inoltre, all'aumentare di n diminuisce la grandezza del pacchetto di dati da condividere, ma aumenta il numero di pacchetti dei quali ciascun processore ha bisogno e quindi, la maggior parte del tempo è in attesa di dati. Risulta perciò estremamente rilevante il tipo di rete di comunicazione tra i diversi nodi con i relativi tempi di latenza e velocità di comunicazione. Quando si hanno pacchetti grandi è più importante la velocità di comunicazione, viceversa, per piccoli pacchetti di dati diviene determinante il tempo necessario ad instaurare la comunicazione e quindi i tempi di latenza.

La condizione ideale si raggiuge quando si ha un compromesso tra il carico computazionale di ciascun processore e i tempi di comunicazione, infatti, quando si utilizzano interamente due nodi (n = 32), il peso delle principali funzioni è sostanzialmente lo stesso (tra il 10% e il 20%) evidenziando un bilanciamento tra il calcolo e la comunicazione.

4.2. Dinamica molecolare

Dovendo studiare la mobilità dell'idrogeno nel sistema in oggetto a differenti temperature, risulta necessario eseguire delle simulazioni di dinamica molecolare. Come fatto in precedenza, prima di effettuare questo tipo di analisi, verifichiamo per il nostro particolare sistema fisico, in quali condizioni vengano sfruttate al meglio le risorse computazionali disponibili. Procedendo in maniera del tutto simile al caso dell'ottimizzazione delle funzioni d'onda, analizzeremo i tempi impiegati per eseguire la stessa simulazione di dinamica molecolare e i tempi delle funzioni maggiormente impiegate in tale simulazione, al variare del numero n dei processori utilizzati per eseguire il codice parallelo CPMD. I risultati di questa analisi sono mostrati nei grafici della Figura9 e Figura10.

In Figura9 sono mostrati i tempi di esecuzione totali per calcolare 300 passi di dinamica molecolare con relativo *speed up*. Anche in questo caso si ha una parallelizzazione soddisfacente fino a n = 32, dove si raggiunge ancora circa l'80% dello *speed up* ideale. Il motivo di tale risultato è lo stesso del caso precedente. Infatti, dalla Figura10, dove vengono riportati i tempi di esecuzione delle funzioni maggiormente usate (`FWFFT`, `INVFFT`, `FFT G/S`, `FFTCOM`) e il loro peso relativo al tempo di esecuzione totale, si vede come la condizione ottimale sia ancora quella in cui si ha un equilibrio tra il carico di lavoro computazionale e il tempo necessario per lo scambio di dati tra i vari processori.

Possiamo concludere questa sezione, osservando come sia importante per la scalabilità di un codice parallelo, non solo la potenza di calcolo dei processori e il loro numero, ma risulta determinante anche la rete di comunicazione tra le varie componenti della macchina di calcolo utilizzata.

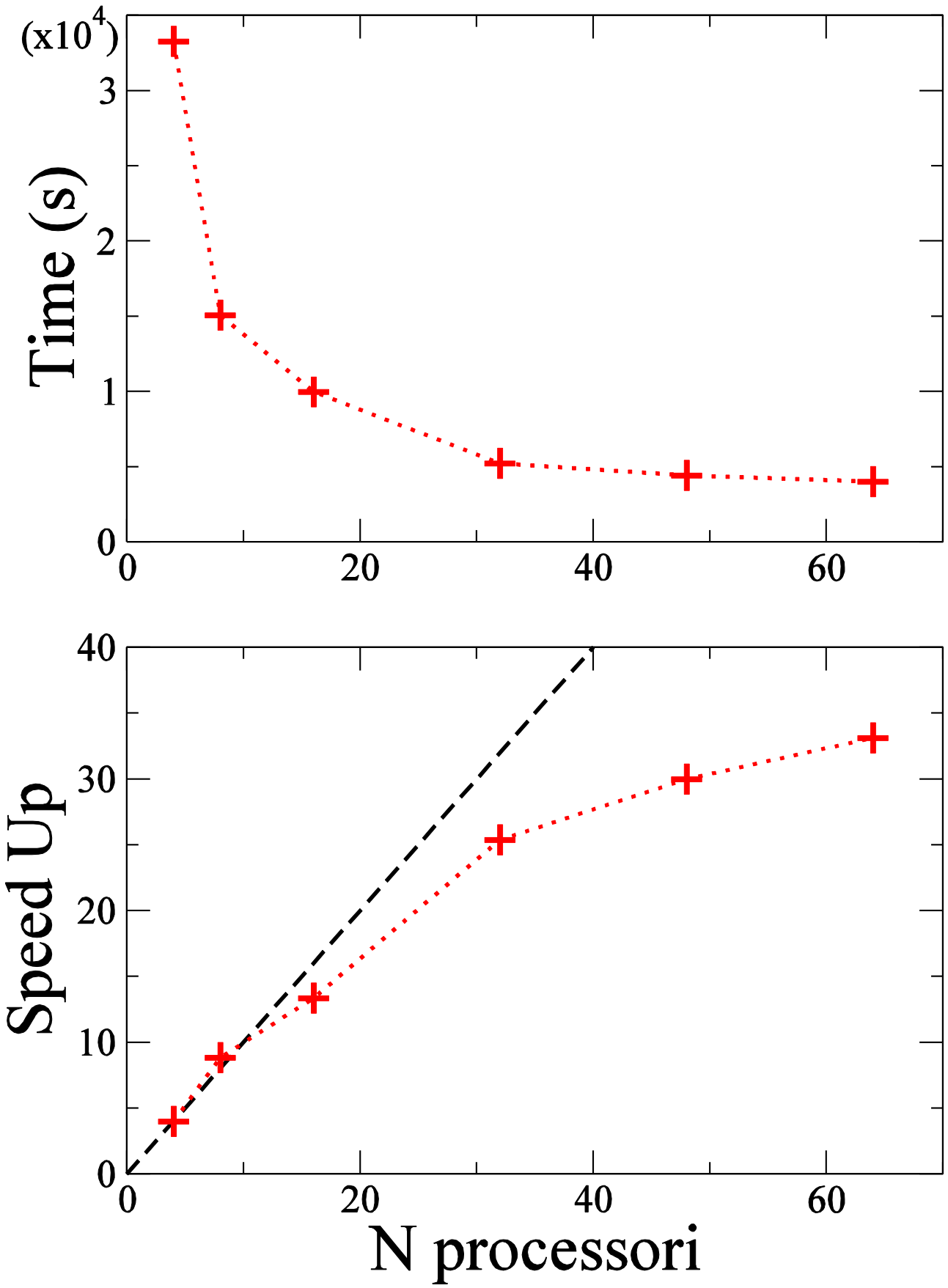

Figura7: Tempi di esecuzione del codice CPMD per l'ottimizzazione delle funzioni d'onda e relativo *speed up*, al variare del numero di processori utilizzati.

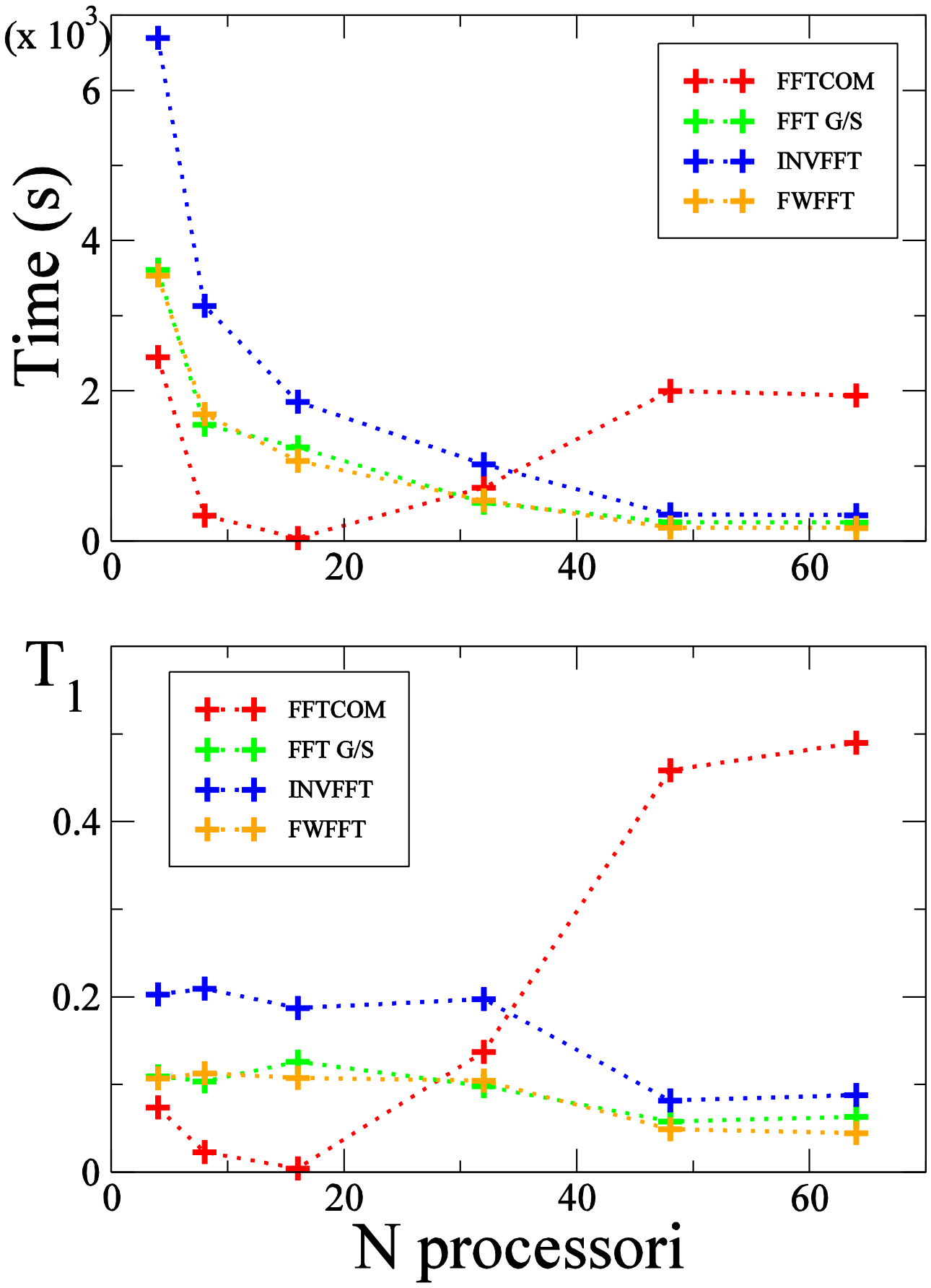

Figura8: Tempi di esecuzione delle principali funzioni utilizzate (`FWFFT`, `INVFFT`, `FFT G/S`, `FFTCOM`) e loro frazione rispetto al tempo di esecuzione totale, al variare del numero di processori.

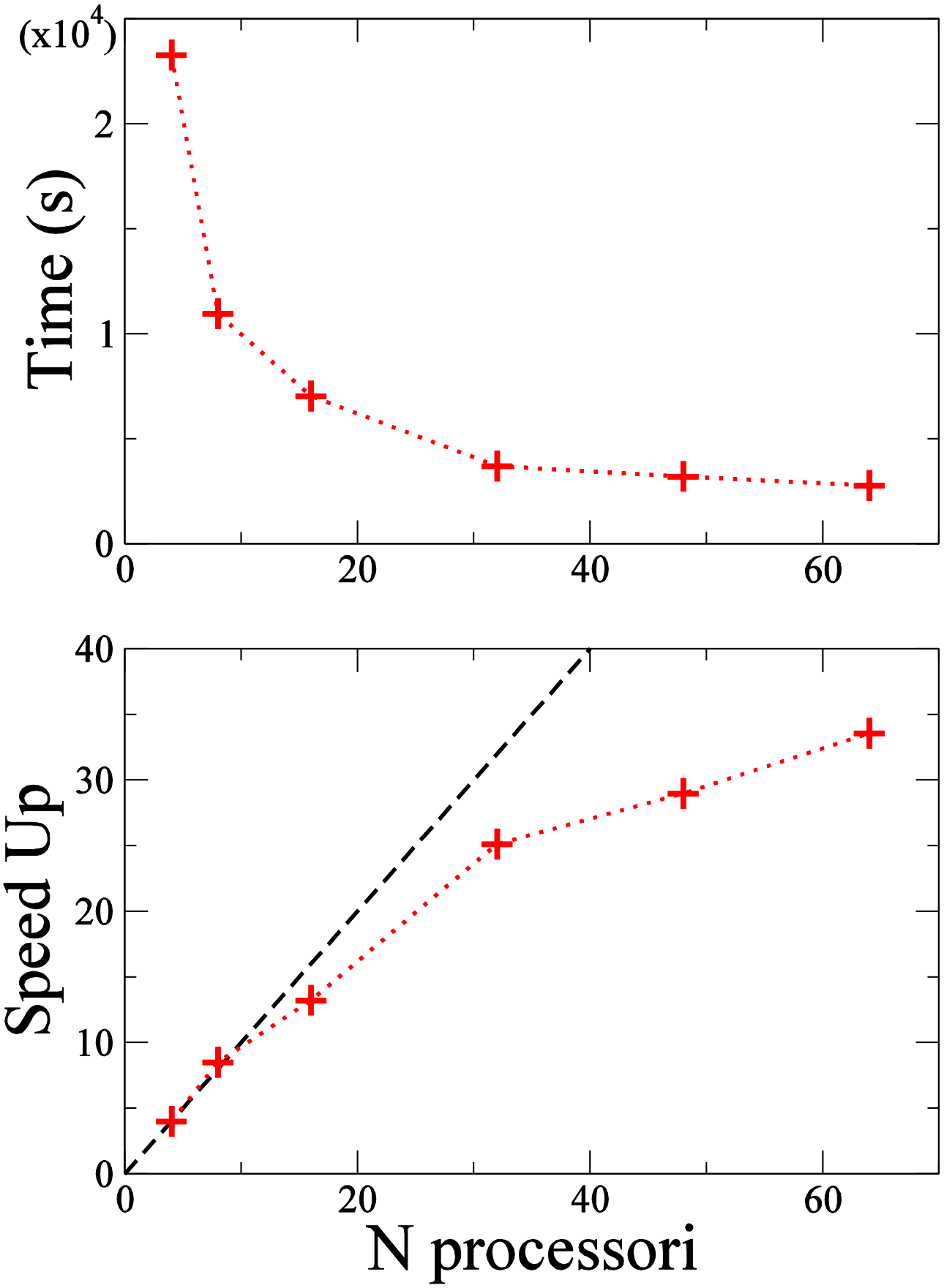

Figura9: Tempi di esecuzione del codice CPMD per la dinamica molecolare e relativo *speed up*, al variare del numero di processori utilizzati.

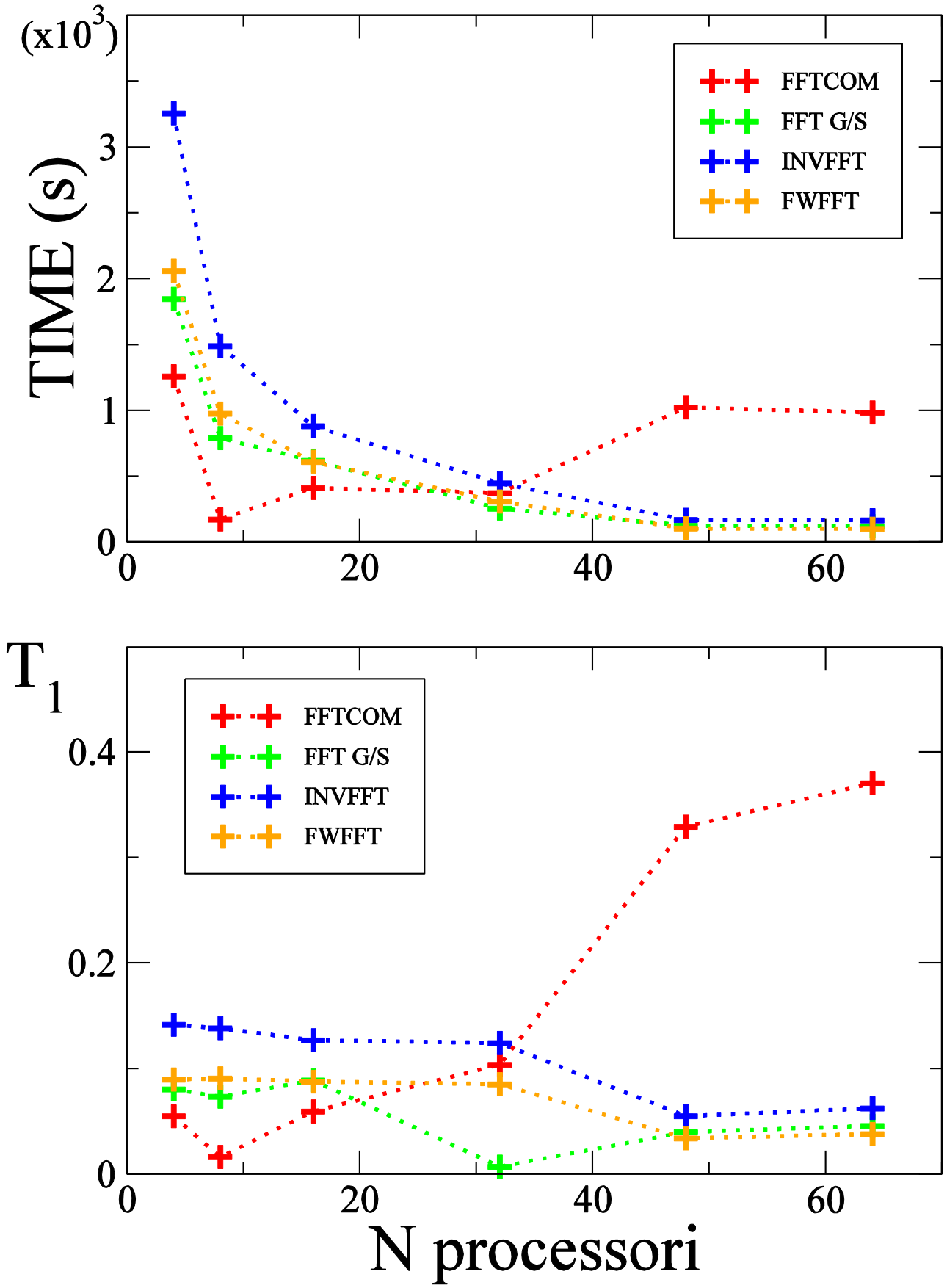

Figura10: Tempi di esecuzione delle principali funzioni utilizzate (`FWFFT`, `INVFFT`, `FFT G/S`, `FFTCOM`) e loro frazione rispetto al tempo di esecuzione totale, al variare del numero di processori.

Appendice A

`BLAS`: File di configurazione

`make.inc`

A.1 Macchine IBM SP4 e SP5

Di seguito riportiamo i file di configurazione utilizzati per la compilazione delle librerie `BLAS` su macchine IBM.

SP4 a 64 bit.

```
####################################################################
# BLAS make include file.                                          #
# March 2007                                                       #
####################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP4-64xlf
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = xlf
OPTS = -O3 -qstrict -q64 -qmaxmem=-1 -qtune=pwr4 -qarch=pwr4
DRVOPTS = $(OPTS)
NOOPT = -q64
LOADER = xlf
LOADOPTS = -q64
#
# The archiver and the flag(s) to use when building archive
#(library)
# If you system has no ranlib, set RANLIB = echo.
#
```

```
ARCH = ar
ARCHFLAGS= -cr -X64
RANLIB = ranlib
#
# The location and name of the Reference BLAS library.
#
BLASLIB = blas$(PLAT).a
```

SP4 a 32 bit.

```
######################################################################
# BLAS make include file.                                            #
# March 2007                                                         #
######################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP4-32xlf
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = xlf
OPTS = -O3 -qstrict -qmaxmem=-1 -qtune=pwr4 -qarch=pwr4
DRVOPTS = $(OPTS)
NOOPT =
LOADER = xlf
LOADOPTS =
#
# The archiver and the flag(s) to use when building archive
# (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
```

```
ARCHFLAGS= -cr
RANLIB = ranlib
#
# The location and name of the Reference BLAS library.
#
BLASLIB = blas$(PLAT).a
```

SP5 a 64 bit.

```
######################################################################
# BLAS make include file.                                            #
# March 2007                                                         #
######################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP5-64xlf
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = xlf
OPTS = -O3 -qstrict -q64 -qmaxmem=-1 -qtune=pwr5 -qarch=pwr5
DRVOPTS = $(OPTS)
NOOPT = -q64
LOADER = xlf
LOADOPTS = -q64
#
# The archiver and the flag(s) to use when building archive
#(library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= -cr -X64
```

```
RANLIB = ranlib
#
# The location and name of the Reference BLAS library.
#
BLASLIB = blas$(PLAT).a
```

SP5 a 32 bit.

```
######################################################################
# BLAS make include file.                                            #
# March 2007                                                         #
######################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP5-32xlf
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = xlf
OPTS = -O3 -qstrict -qmaxmem=-1 -qtune=pwr5 -qarch=pwr5
DRVOPTS = $(OPTS)
NOOPT =
LOADER = xlf
LOADOPTS =
#
# The archiver and the flag(s) to use when building archive (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= -cr
RANLIB = ranlib
#
```

```
# The location and name of the Reference BLAS library.
#
BLASLIB = blas$(PLAT).a
```

IBM(R) XL Fortran Enterprise Edition V10.1 for AIX(R) opzioni di compilazione:

```
-O[<level>]
Specifies whether to optimize code during compilation and, if so, at
which level.
   3
      Performs additional optimizations that are memory intensive,
      compile-time intensive, and may change the semantics of the program
      slightly, unless -qstrict is specified. We recommend these
      optimizations when the desire for run-time speed improvements
      outweighs the concern for limiting compile-time resources. This
      level of optimization also affects the setting of the -qfloat
      option, turning on the 'fltint' and 'rsqrt' suboptions by default,
      and setting -qmaxmem=-1. This option implies -qhot=level=0.

-q32
      Selects 32-bit mode compilation mode. The -q32 and –qarch options
      determine the target machines that the 32-bit executable will run
      on. The default is -q32.

-q64[=<suboption>]
      Selects 64-bit compilation mode. The -q64 option indicates that the
      object module will be created in 64-bit object format and that the
      64-bit instruction set will be generated. Note that you may compile
      in a 32-bit environment to create 64-bit objects, but you must link
      them in a 64-bit environment with the -q64 option. Use -q32 and -
      q64 options, along with the -qarch and -qtune compiler options, to
      optimize the output of the compiler to the architecture on which
      that output will be used.
-qarch=<suboption>
      Specifies which instructions the compiler can generate.
      pwr4
            Produces an object that contains instructions that run on the
            POWER4, POWER5, POWER5+ or PowerPC 970 hardware platforms.
```

pwr5

Produces an object that contains instructions that run on the POWER5 or POWER5+ hardware platforms.

-qmaxmem=<kbytes>

Limits the amount of memory that the compiler allocates while performing specific, memory-intensive optimizations. If <kbytes> is -1, the compiler will take as much memory as it needs without checking for limits. The default is
-qmaxmem=2048 when using optimization level -O2, and
-qmaxmem=-1 when using -O3 or greater.

-qstrict | -qnostrict

Ensures that optimizations done by the -O3 and greater (and optionally -O2) options do not alter the semantics of a program. This option is ignored for -qnoopt.
For -O3 and greater, the default is -qnostrict. Otherwise the default is -qstrict

-qtune=<suboption>

Tunes instruction selection, scheduling, and other implementation-dependent performance enhancements for a specific implementation of a hardware architecture. -qtune will not alter the ability of a program to run on a processor but will produce optimal code sequences tuned to a particular processor.

pwr4

Produces an object optimized for the POWER4 hardware platforms.

pwr5

Produces an object optimized for the POWER5 hardware platforms.

A.2 Macchine LINUX bw305 e lin4p

Di seguito riportiamo i file di configurazione utilizzati per la compilazione delle librerie BLAS su macchine LINUX.

bw305 (lin4p) con compilatore g77.

```
####################################################################
# BLAS make include file.                                          #
# March 2007                                                       #
####################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _bw305_LINUX-g77
#
# Modify the FORTRAN and OPTS definitions to refer to the compiler and
# desired compiler options for your machine. NOOPT refers to the
# compiler options desired when NO OPTIMIZATION is selected. Define
# LOADER and LOADOPTS to refer to the loader and desired load options
# for your machine.
FORTRAN = g77
OPTS = -funroll-all-loops -O3
DRVOPTS = $(OPTS)
NOOPT =
LOADER = g77
LOADOPTS =
#
# The archiver and the flag(s) to use when building archive (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= cr
RANLIB = ranlib
#
# The location and name of the Reference BLAS library.
#
BLASLIB = blas$(PLAT).a
```

bw305 (lin4p) con compilatore pgf77.

```
####################################################################
# BLAS make include file.                                          #
# March 2007                                                       #
####################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _bw305_LINUX-pgf77
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = pgf77
OPTS = -O3
DRVOPTS = $(OPTS)
NOOPT =
LOADER = pgf77
LOADOPTS =
#
# The archiver and the flag(s) to use when building archive (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= cr
RANLIB = ranlib
#
# The location and name of the Reference BLAS library.
#
BLASLIB = blas$(PLAT).a
```

Appendice B

LAPACK: File di configurazione

make.inc

B.1 Macchine IBM SP4 e SP5

Di seguito riportiamo i file di configurazione utilizzati per la compilazione delle librerie LAPACK su macchine IBM.

SP4 a 64 bit.

```
###################################################################
# LAPACK make include file.                                       #
# LAPACK, Version 3.1.1                                           #
# February 2007                                                   #
###################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP4-32xlfblas
#
# Modify the FORTRAN and OPTS definitions to refer to the compiler and
# desired compiler options for your machine. NOOPT refers to the
# compiler options desired when NO OPTIMIZATION is selected. Define
# LOADER and LOADOPTS to refer to the loader and desired load options
# for your machine.
FORTRAN = xlf
OPTS = -O3 -qstrict -qmaxmem=-1 -qtune=pwr4 -qarch=pwr4
DRVOPTS = $(OPTS)
NOOPT =
LOADER = xlf
LOADOPTS =
#
# Timer for the SECOND and DSECND routines
#
# Default : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME
```

```
TIMER = NONE
# For RS6K : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME_
# TIMER = EXT_ETIME_
# For gfortran compiler: SECOND and DSECND will use a call to the
# INTERNAL FUNCTION ETIME
# TIMER = INT_ETIME
# If your Fortran compiler does not provide etime (like Nag Fortran
# Compiler, etc...) SECOND and DSECND will use a call to the INTERNAL
# FUNCTION CPU_TIME
# TIMER = INT_CPU_TIME
# If neither of this works..you can use the NONE value..In that case,
# SECOND and DSECND will always return 0
# TIMER = NONE
#
# The archiver and the flag(s) to use when building archive (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= -cr
RANLIB = ranlib
#
# The location of the libraries to which you will link. (The machine
# specific, optimized BLAS library should be used whenever possible.)
#
#BLASLIB = ../../blas$(PLAT).a
BLASLIB = -L/afs/enea.it/software/OSafs/lib -lblas32p4xlf
LAPACKLIB = lapack$(PLAT).a
TMGLIB = tmglib$(PLAT).a
EIGSRCLIB = eigsrc$(PLAT).a
LINSRCLIB = linsrc$(PLAT).a
```

SP4 a 64 bit.

```
####################################################################
# LAPACK make include file.                                        #
# LAPACK, Version 3.1.1                                            #
# February 2007                                                    #
####################################################################
```

```
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP4-64xlfblas
#
# Modify the FORTRAN and OPTS definitions to refer to the compiler and
# desired compiler options for your machine. NOOPT refers to the
# compiler options desired when NO OPTIMIZATION is selected. Define
# LOADER and LOADOPTS to refer to the loader and desired load options
# for your machine.
FORTRAN = xlf
OPTS = -O3 -qstrict -q64 -qmaxmem=-1 -qtune=pwr4 -qarch=pwr4
DRVOPTS = $(OPTS)
NOOPT = -q64
LOADER = xlf
LOADOPTS = -q64
#
# Timer for the SECOND and DSECND routines
#
# Default : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME
TIMER = NONE
# For RS6K : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME_
# TIMER = EXT_ETIME_
# For gfortran compiler: SECOND and DSECND will use a call to the
# INTERNAL FUNCTION ETIME
# TIMER = INT_ETIME
# If your Fortran compiler does not provide etime (like Nag
# Fortran Compiler, etc...) SECOND and DSECND will use a call to
# the INTERNAL FUNCTION CPU_TIME
# TIMER = INT_CPU_TIME
# If neither of this works..you can use the NONE value.. In that case,
# SECOND and DSECND will always return 0
# TIMER = NONE
#
```

```
# The archiver and the flag(s) to use when building archive (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= -cr -X64
RANLIB = ranlib
#
# The location of the libraries to which you will link. (The machine
# specific, optimized BLAS library should be used whenever possible.)
#
#BLASLIB = ../../blas$(PLAT).a
BLASLIB = -L/afs/enea.it/software/OSafs/lib -lblas64p4xlf
LAPACKLIB = lapack$(PLAT).a
TMGLIB = tmglib$(PLAT).a
EIGSRCLIB = eigsrc$(PLAT).a
LINSRCLIB = linsrc$(PLAT).a
```

SP5 a 32 bit.

```
######################################################################
# LAPACK make include file.                                          #
# LAPACK, Version 3.1.1                                              #
# February 2007                                                      #
######################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP5-32xlfblas
#
# Modify the FORTRAN and OPTS definitions to refer to the compiler and
# desired compiler options for your machine. NOOPT refers to the
# compiler options desired when NO OPTIMIZATION is selected. Define
# LOADER and LOADOPTS to refer to the loader and desired load options
# for your machine.
FORTRAN = xlf
OPTS = -O3 -qstrict -qmaxmem=-1 -qtune=pwr5 -qarch=pwr5
DRVOPTS = $(OPTS)
```

```
NOOPT =
LOADER = xlf
LOADOPTS =
#
# Timer for the SECOND and DSECND routines
#
# Default : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME
TIMER = NONE
# For RS6K : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME_
# TIMER = EXT_ETIME_
# For gfortran compiler: SECOND and DSECND will use a call to the
# INTERNAL FUNCTION ETIME
# TIMER = INT_ETIME
# If your Fortran compiler does not provide etime (like Nag
# Fortran Compiler, etc...) SECOND and DSECND will use a call to
# the INTERNAL FUNCTION CPU_TIME
# TIMER = INT_CPU_TIME
# If neither of this works..you can use the NONE value.. In that case,
# SECOND and DSECND will always return 0
# TIMER = NONE
#
# The archiver and the flag(s) to use when building archive (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= -cr
RANLIB = ranlib
#
# The location of the libraries to which you will link. (The machine
# specific, optimized BLAS library should be used whenever possible.)
#
BLASLIB = -L/afs/enea.it/software/OSafs/lib -lblas32p5xlf
LAPACKLIB = lapack$(PLAT).a
TMGLIB = tmglib$(PLAT).a
EIGSRCLIB = eigsrc$(PLAT).a
LINSRCLIB = linsrc$(PLAT).a
```

SP5 a 64 bit.

```
############################################################################
# LAPACK make include file.                                                #
# LAPACK, Version 3.1.1                                                    #
# February 2007                                                            #
############################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _SP5-64xlfblas
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = xlf
OPTS = -O3 -qstrict -q64 -qmaxmem=-1 -qtune=pwr5 -qarch=pwr5
DRVOPTS = $(OPTS)
NOOPT = -q64
LOADER = xlf
LOADOPTS = -q64
#
# Timer for the SECOND and DSECND routines
#
# Default : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME
TIMER = NONE
# For RS6K : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME_
# TIMER = EXT_ETIME_
# For gfortran compiler: SECOND and DSECND will use a call to the
# INTERNAL FUNCTION ETIME
# TIMER = INT_ETIME
# If your Fortran compiler does not provide etime (like Nag
```

```
# Fortran Compiler, etc...) SECOND and DSECND will use a call to
# the INTERNAL FUNCTION CPU_TIME
# TIMER = INT_CPU_TIME
# If neither of this works...you can use the NONE value... In that
# case,
# SECOND and DSECND will always return 0
# TIMER = NONE
#
# The archiver and the flag(s) to use when building archive
# (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= -cr -X64
RANLIB = ranlib
#
# The location of the libraries to which you will link. (The
# machine-specific, optimized BLAS library should be used whenever
# possible.)
#
#BLASLIB = ../../blas$(PLAT).a
BLASLIB = -L/afs/enea.it/software/OSafs/lib -lblas64p5x1f
LAPACKLIB = lapack$(PLAT).a
TMGLIB = tmglib$(PLAT).a
EIGSRCLIB = eigsrc$(PLAT).a
LINSRCLIB = linsrc$(PLAT).a
```

B.2 Macchine LINUX bw305 e lin4p

Di seguito riportiamo i file di configurazione utilizzati per la compilazione delle librerie LAPACK su macchine LINUX.

bw305 (lin4p) con compilatore g77.

```
###################################################################
# LAPACK make include file.                                       #
# LAPACK, Version 3.1.1                                           #
# February 2007                                                   #
###################################################################
#
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _bw305_LINUX-g77
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = g77
OPTS = -funroll-all-loops -O3
DRVOPTS = $(OPTS)
NOOPT =
LOADER = g77
LOADOPTS =
#
# Timer for the SECOND and DSECND routines
#
# Default : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME
TIMER = NONE
# For RS6K : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME_
# TIMER = EXT_ETIME_
```

```
# For gfortran compiler: SECOND and DSECND will use a call to the
# INTERNAL FUNCTION ETIME
# TIMER = INT_ETIME
# If your Fortran compiler does not provide etime (like Nag
# Fortran Compiler, etc...) SECOND and DSECND will use a call to
# the INTERNAL FUNCTION CPU_TIME
# TIMER = INT_CPU_TIME
# If neither of this works...you can use the NONE value... In that
# case,
# SECOND and DSECND will always return 0
# TIMER = NONE
#
# The archiver and the flag(s) to use when building archive
# (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= cr
RANLIB = ranlib
#
# The location of the libraries to which you will link. (The
# machine-specific, optimized BLAS library should be used whenever
# possible.)
#
#BLASLIB = ../../blas$(PLAT).a
BLASLIB = -L/afs/enea.it/software/OSafs/lib/ -lblaslinuxg77
LAPACKLIB = lapack$(PLAT).a
TMGLIB = tmglib$(PLAT).a
EIGSRCLIB = eigsrc$(PLAT).a
LINSRCLIB = linsrc$(PLAT).a
```

---

bw305 (lin4p) con compilatore pgf77.

```
####################################################################
# LAPACK make include file.                                        #
# LAPACK, Version 3.1.1                                            #
# February 2007                                                    #
####################################################################
#
```

```
SHELL = /bin/sh
#
# The machine (platform) identifier to append to the library names
#
PLAT = _bw305_LINUX-pgf77
#
# Modify the FORTRAN and OPTS definitions to refer to the
# compiler and desired compiler options for your machine. NOOPT
# refers to the compiler options desired when NO OPTIMIZATION is
# selected. Define LOADER and LOADOPTS to refer to the loader and
# desired load options for your machine.
#
FORTRAN = pgf77
OPTS = -O3
DRVOPTS = $(OPTS)
NOOPT =
LOADER = pgf77
LOADOPTS =
#
# Timer for the SECOND and DSECND routines
#
# Default : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME
TIMER = NONE
# For RS6K : SECOND and DSECND will use a call to the EXTERNAL
# FUNCTION ETIME_
# TIMER = EXT_ETIME_
# For gfortran compiler: SECOND and DSECND will use a call to the
# INTERNAL FUNCTION ETIME
# TIMER = INT_ETIME
# If your Fortran compiler does not provide etime (like Nag
# Fortran Compiler, etc...) SECOND and DSECND will use a call to
# the INTERNAL FUNCTION CPU_TIME
# TIMER = INT_CPU_TIME
# If neither of this works...you can use the NONE value... In that
# case,
# SECOND and DSECND will always return 0
# TIMER = NONE
```

```
#
# The archiver and the flag(s) to use when building archive
# (library)
# If you system has no ranlib, set RANLIB = echo.
#
ARCH = ar
ARCHFLAGS= cr
RANLIB = ranlib
#
# The location of the libraries to which you will link. (The
# machine-specific, optimized BLAS library should be used whenever
# possible.)
#
#BLASLIB = ../../blas$(PLAT).a
BLASLIB = -L/afs/enea.it/software/OSafs/lib -lblaslinuxpgf77
LAPACKLIB = lapack$(PLAT).a
TMGLIB = tmglib$(PLAT).a
EIGSRCLIB = eigsrc$(PLAT).a
LINSRCLIB = linsrc$(PLAT).a
```

ENEA
Servizio Promozione e Comunicazione
www.enea.it

Stampa: Laboratorio Tecnografico ENEA - C.R. Frascati
maggio 2017