DIPARTIMENTO DI SCIENZE CHIMICHE
CORSO DI LAUREA MAGISTRALE IN CHIMICA

TESI DI LAUREA MAGISTRALE

Approcci teorico/computazionali per l'interpretazione di dinamiche molecolari 50-500 ns in proteine.

Relatore: Prof. Antonino Polimeno

Controrelatore: Dott.ssa Chiara Maccato

LAUREANDO: Nicola Fortunati

ANNO ACCADEMICO 2011-2012

INDICE

Approcci teorico/computazionali per l'interpretazione di dinamiche molecolari 50-500 ns in proteine.

di

Fortunati Nicola

Tesi svolta presso il Dipartimento di Scienze Chimiche per il conseguimento della Laurea Magistrale in Chimica il 29 Marzo 2012.

Sommario

In questo lavoro di tesi è presentata una serie di metodologie teoriche e computazionali per il calcolo di proprietà diffusive di proteine (globulari e non), in particolare del tensore di diffusione rotazionale.

I metodi impiegati sono basati sul formalismo delle equazioni stocastiche a molti corpi, che negli ultimi quarant'anni è stato applicato con successo all'interpretazione di numerose osservabili in fase liquida, quali: spettroscopie, processi di trasferimento di carica, studi di reattività chimica. L'approccio seguito in questa Tesi prevede l'impiego di simulazioni di dinamica molecolare (MD) per parametrizzare i modelli stocastici allo scopo di ottenere informazioni a tempi lunghi del sistema studiato (ad es. per riprodurre osservabili di tipo spettroscopico).

Dopo una presentazione del formalismo fisico-matematico che certifica le simulazioni MD come uno strumento valido in questo campo d'indagine, il lavoro si concentra sullo studio del tensore di diffusione rotazionale di proteine globulari esplorando tre approcci di natura differente: due basati sull'analisi di funzioni di autocorrelazione (del momento angolare globale della proteina e di appropriate funzioni di Wigner) calcolate dalla traiettoria MD e uno basato su un approccio di tipo idrodinamico.

Infine, il lavoro si concentra sullo studio della dinamica rotazionale della Calmodulina, proteina non globulare costituita da due domini rigidi collegati da un *linker* flessibile. In questo caso s'impiega un modello stocastico a due corpi, parametrizzato da calcoli idrodinamici (tensori di diffusione rotazionali) e simulazioni MD (potenziale di campo medio), per il calcolo *ab-initio* tempi di correlazione rotazionale dei due domini.

1 Introduzione

L'obiettivo di questa Tesi è la messa a punto dei protocolli teorico / computazionali per la valutazione di proprietà dissipative di proteine, globulari e non. Si è interessati in particolar modo al calcolo del tensore di diffusione rotazionale, una proprietà molecolare molto importante nell'interpretazione di osservabili fisiche macroscopiche, quali le misure di natura spettroscopica. Il piano di lavoro prevede l'impiego di simulazioni di dinamica molecolare (MD) per ottenere informazioni sull'evoluzione temporale del sistema, a livello atomico-molecolare, a tempi relativamente brevi (ordine di 10^2 *ns*) da interpretate con appropriati modelli stocastici, costruiti su un'attenta scelta delle coordinate rilevanti. Questi ultimi sono poi utilizzati per valutare il comportamento a tempi lunghi del sistema, non ancora accessibile con simulazioni MD a livello atomico.

Questo lavoro di Tesi è organizzato nel modo seguente. In primo luogo sarà discusso l'impiego della dinamica molecolare come utile strumento per lo studio dei moti molecolari ed il modo di correlare l'osservazione del moto di una singola molecola ad osservabili macroscopiche misurate su un campione esteso (dell'ordine del numero di Avogadro) di molecole. Il Capitolo 2 sarà dedicato all'esplorazioni di tre differenti metodologie per il calcolo del tensore di diffusione rotazionale di proteine globulari basate (i) sull'impiego di appropriate funzioni di correlazione di funzioni di coordinate o momenti della molecola, (ii) su un approccio di tipo idrodinamico. Nel Capitolo 3 saranno combinate informazioni di natura atomistica (simulazioni MD) e *coarse-grained* (metodi idrodinamici) per la parametrizzazione completa di un modello stocastico a due corpi per il calcolo *ab-initio* di tempi di correlazione rotazionali dei due domini della Calmodulina come esempio di applicazione di questi protocolli computazionali per l'interpretazione di osservabili fisiche di sistemi complessi.

1.1 Esperimenti Computazionali

L'esperimento computazionale occupa un ruolo importante nella Scienza odierna [1, 3]. Fin dal passato, le Scienze Fisiche e Chimiche sono state caratterizzate dall'interazione tra *esperimenti* e *metodi teorici*. In un esperimento, un sistema è soggetto a misure e i risultati sono espressi in forma numerica. Con i metodi teorici, un modello è costruito in forma di equazioni matematiche per descrivere il sistema; il

modello è quindi convalidato dalla sua abilità nel descrivere il comportamento del sistema in determinati casi, abbastanza semplici.

Nel passato, i modelli teorici potevano essere testati solamente in pochi casi particolari: per esempio, nella fisica della materia condensata, in specifici materiali, un modello di forze intermolecolari poteva essere verificato solamente in una molecola biatomica o in un cristallo ideale infinito. Sfortunatamente, moltissimi problemi chimico-fisici odierni di grande interesse non rientrano nel campo di questi casi particolari: si può menzionare la Fisica e la Chimica delle superfici, dei cluster di atomi, delle molecole organiche e biorganiche [5], sistemi che coinvolgono molti gradi di libertà e sistemi disordinati in generale.

L'avvento di strumenti per il calcolo ad alte prestazioni ha permesso di introdurre un nuovo tipo di esperimento nella ricerca in scienze Chimiche: l'esperimento computazionale. Quest'ultimo parte dalla formulazione di una modellistica teorica di base e la sua implementazione in un algoritmo con cui simulare il comportamento di un sistema. In base alla complessità del modello e all'efficienza dell'algoritmo è possibile investigare sistemi più realistici e via via più estesi.

Lo sviluppo degli esperimenti computazionali ha sostanzialmente alterato il rapporto tra metodi teorici ed esperimenti di laboratorio: da un lato le simulazioni aumentano la richiesta di modelli più precisi, dall'altro esse possono raggiungere condizioni molto vicine a quelle sperimentali, nella misura in cui i risultati computazionali a volte possono essere confrontati direttamente con i risultati sperimentali. Quando ciò accade, le simulazioni diventano uno strumento molto potente non solo per capire e interpretare gli esperimenti a livello microscopico ma anche per studiare fenomeni non accessibili sperimentalmente o che implicano esperimenti costosi. Esistono scuole di pensiero differenti sulla classificazione di un esperimento computazionale come semplice teoria o come vero e proprio strumento di indagine molecolare al pari di tecniche di laboratorio. Alcuni autori suggeriscono interpretazioni più o meno complesse, come quella riportata in **Fig.1.1.1** in cui vengono rappresentate le connessioni tra esperimento, teoria ed esperimento computazionale, nello studio di un liquido [1].

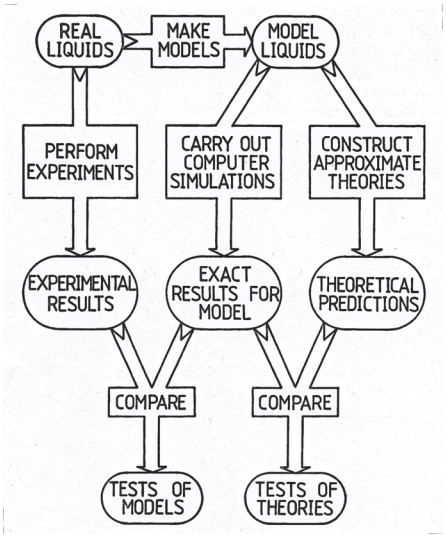


Fig.1.1.1: connessioni tra esperimento, teoria e esperimento computazionale [1] (*Fonte: Computer Simulation of Liquids, pag.4, Fig.1.1).*

L'impiego dell'esperimento computazionale, rinforzato dal continuo sviluppo da una parte di metodi teorici *ad hoc* e dall'altra di strumenti hardware e software dedicati ad alte prestazioni, è possibile aumentare la soglia della complessità dei sistemi che si possono simulare. Un esempio è rappresentato dai campi di forze interatomici nella dinamica molecolare: in passato le interazioni tra atomi erano descritte con semplici modelli di potenziale a due corpi (es: Morse o Lennard Jones); oggi, i potenziali più accurati contengono termini multi corpo e sono *determinati numericamente*. È interessante ricordare il pacchetto software CHARMM [7, 10], che implementa un campo di forze largamente impiegato nella simulazione di proprietà strutturali e dinamiche di macromolecole biologiche, tra cui proteine e lipidi. Lo sviluppo di nuovi campi di forze via via più efficaci è possibile grazie alle simulazioni stesse: gli esperimenti computazionali, perciò, non sono solo un collegamento tra esperimento e teoria ma anche un potente strumento che spinge la conoscenza scientifica in nuove direzioni.

1.2 Dinamica Molecolare

1.2.1 Aspetti teorici

La Dinamica Molecolare (MD) è una metodologia computazionale [2] molto utilizzata per calcolare proprietà termodinamiche (equilibrio, trasporto, strutturali, ecc.) di *sistemi classici a molti corpi*. Il termine *classico* significa che il moto

nucleare delle particelle, costituenti il sistema, obbedisce rigorosamente alle leggi della *Meccanica Classica*: questa è un'eccellente approssimazione per un gran numero di materiali.

In MD l'evoluzione temporale di un set di N particelle interagenti è seguita dall'integrazione delle loro equazioni del moto [3]; per ciascuna particella i-esima, che compone il sistema studiato, è possibile scrivere l'equazione di Newton:

$$\boldsymbol{F}_i = -\nabla_{\boldsymbol{r}_i} V\left(\boldsymbol{r}_1,...,\boldsymbol{r}_i,...,\boldsymbol{r}_N\right) \qquad (1.2.1)$$

Qui $\boldsymbol{F}_i$ è la forza che agisce sull'atomo i-esimo, ricavata come gradiente del potenziale d'interazione $V\left(\boldsymbol{r}_1,...,\boldsymbol{r}_i,...,\boldsymbol{r}_N\right)$ rispetto agli spostamenti atomici. In termini pittorici, le particelle si "muovono" all'interno del computer, urtano tra loro vagando e oscillando, in un modo piuttosto simile a quello che gli atomi e molecole fanno in una sostanza reale [4]. Le proprietà statiche del sistema sono definite se è noto il potenziale che contribuisce all'Hamiltoniana totale, H, del sistema:

$$H\left(\boldsymbol{r}_1,...,\boldsymbol{r}_i,...,\boldsymbol{r}_N;\boldsymbol{p}_1,...,\boldsymbol{p}_i,...,\boldsymbol{p}_N\right) = K+V = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + V\left(\boldsymbol{r}_1,...,\boldsymbol{r}_i,...,\boldsymbol{r}_N\right) \qquad (1.2.2)$$

Qui K è l'energia cinetica totale del sistema, $\boldsymbol{r}_i$ e $\boldsymbol{p}_i$ sono rispettivamente la posizione ed il momento coniugato dell'i-esima particella, m_i è la sua massa. La presenza di eventuali campi esterni (elettrici o magnetici) permette di simulare sistemi fluidi in condizioni di non-equilibrio.

MD genera informazioni a livello microscopico (coordinate e velocità atomiche e/o molecolari) e la conversione di queste in termini termodinamici-macroscopici avviene mediante i teoremi della Meccanica Statistica (MS).

Lo stato termodinamico di un sistema è completamente definito da un piccolo insieme di parametri (ad es. numero di particelle N, temperatura T e pressione P). Le altre proprietà termodinamiche (come densità ρ, potenziale chimico μ) possono essere calcolate mediante le *equazioni di stato* e mediante le *equazioni fondamentali* della Termodinamica.

La dinamica molecolare è una *tecnica deterministica* per lo studio di sistemi molecolari [1, 2]: per un insieme di N atomi viene calcolata la traiettoria deterministica nello spazio delle fasi $6N$ dimensionale ($3N$ posizioni e $3N$ momenti coniugati). Se con $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}\left(\boldsymbol{r}_1,...,\boldsymbol{r}_i,...,\boldsymbol{r}_N;\boldsymbol{p}_1,...,\boldsymbol{p}_i,...,\boldsymbol{p}_N\right)$ si definisce un generico punto nello spazio delle fasi, il valore istantaneo di una generica proprietà fisica A

(ad es. l'energia totale) sarà una funzione $A(\boldsymbol{\Gamma})$, che mappa la traiettoria di $\boldsymbol{\Gamma}$ sull'evoluzione temporale di $A(\Gamma)$. Nell'ipotesi di *ergodicità* l'*osservabile macroscopica* A_{obs} sia la media temporale di $A(\Gamma)$ campionata in un lungo intervallo temporale t_{obs}:

$$A_{obs} = \langle A \rangle_{time} = \langle A(\Gamma(t)) \rangle_{time} = \lim_{t_{obs}\to\infty} \frac{1}{t_{obs}} \int_0^{t_{obs}} A(\Gamma(t)) dt \qquad (1.2.3)$$

Il tempo t_{obs} deve essere tale da permettere alla molecola in esame di esplorare tutto lo spazio delle fasi campionando la distribuzione statistica sottostante (ad es. quella di Boltzmann per una simulazione di un sistema canonico – vedi sotto) caratteristica di un insieme esteso di repliche della stessa molecola. In questo modo l'intera traiettoria può essere interpretata come un'istantanea del sistema composto da un numero elevato di repliche del sistema ed una media temporale corrisponde alla media sul campione "macroscopico". Per motivi pratici non è possibile estendere l'integrazione dell'eq. (1.2.3) a un tempo infinito, al massimo per un tempo t_{obs} grande finito che dipenderà dal sistema considerato, ma è necessario sincerarsi che sia sufficiente a garantire che il sistema descriva sia all'equilibrio termodinamico. In una simulazione MD. Le equazioni del moto sono risolte per un numero elevato τ_{obs} di intervalli temporali discreti $\delta t = t_{obs}/\tau_{obs}$. In questo caso l'equazione (1.2.3) diventa:

$$A_{obs} = \langle A \rangle_{time} = \frac{1}{\tau_{obs}} \sum_{\tau=1}^{\tau_{obs}} A(\boldsymbol{\Gamma}(\tau)) \qquad (1.2.4)$$

dove τ è un indice che rappresenta gli intervalli temporali.

L'equazione 1.2.4 rimane valida anche in quei casi (come nel metodo Monte Carlo) in cui τ non corrisponde al trascorrere del tempo in senso fisico [1].

Il calcolo di proprietà termodinamiche da medie temporali, eq. (1.2.3), non è implicito nella MD. A causa della complessità dell'evoluzione temporale di $A(\Gamma)$, per un numero elevato di molecole (ordine del numero di Avogadro), Gibbs ha suggerito di rimpiazzare la media temporale con quella d'insieme (*ensemble average*) [1]. Un *ensemble* è considerato come un insieme di punti $\boldsymbol{\Gamma}$ nello spazio delle fasi distribuiti in accordo a una funzione densità di probabilità $\rho_{ens}(\boldsymbol{\Gamma}, t)$. La

sua forma funzionale è determinata dalla scelta dell'*ensemble* che si sta simulando (ad es.. canonico *NVT*, microcanonico *NVE*, ecc.) e secondo il . teorema di Liouville l'evoluzione temporale di $\rho_{ens}\left(\boldsymbol{\Gamma},t\right)$ è regolata dalla seguente equazione:

$$\frac{\partial\rho_{ens}\left(\boldsymbol{\Gamma},t\right)}{\partial t}=-i\hat{L}\rho_{ens}\left(\boldsymbol{\Gamma},t\right)=-\sum_{i}^{N}\left(\dot{\boldsymbol{r}}_i\nabla_{\boldsymbol{r}_i}+\dot{\boldsymbol{p}}_i\nabla_{\boldsymbol{p}_i}\right)\rho_{ens}\left(\boldsymbol{\Gamma},t\right) \tag{1.2.5}$$

Quest'equazione afferma che l'evoluzione temporale di $\rho_{ens}\left(\boldsymbol{\Gamma},t\right)$ in un punto generico dello spazio delle fasi è legata al flusso in/da quel punto. Qui $i\hat{L}$ è l'operatore di Liouville, $\dot{\boldsymbol{r}}_i$ e $\dot{\boldsymbol{p}}_i$ sono le derivate temporali totali della posizione e momento per la particella *i*-esima, $\nabla_{\boldsymbol{r}_i}$ e $\nabla_{\boldsymbol{p}_i}$ sono le derivate rispetto le posizioni e i momenti delle particelle.

La soluzione formale dell'eq. (1.2.5) è:

$$\rho_{ens}\left(\boldsymbol{\Gamma},t\right)=e^{-i\hat{L}t}\rho_{ens}\left(\boldsymbol{\Gamma},0\right) \tag{1.2.6}$$

e ammette una e sola soluzione stazionaria, $\rho_{eq}\left(\boldsymbol{\Gamma}\right)$, a tempo infinito. Quest'ultima rappresenta la distribuzione di probabilità del sistema all'equilibrio termodinamico.

In maniera analoga, l'equazione del moto di una funzione $A\left(\Gamma\right)$, che non dipende esplicitamente dal tempo assume la seguente forma [1]:

$$\frac{d}{dt}A\left(\boldsymbol{\Gamma}\left(t\right)\right)=i\hat{L}A\left(\boldsymbol{\Gamma}\left(t\right)\right) \tag{1.2.7}$$

$$A\left(\boldsymbol{\Gamma}\left(t\right)\right)=e^{i\hat{L}t}A\left(\boldsymbol{\Gamma}\left(0\right)\right) \tag{1.2.8}$$

Nelle equazioni (1.2.5) e (1.2.6) è stata considerata la dipendenza temporale di $\rho_{ens}\left(\boldsymbol{\Gamma},t\right)$ in un punto fisso Γ dello spazio delle fasi; nelle equazioni (1.2.7) e (1.2.8) $A\left(\Gamma\right)$ ha dipendenza temporale perché si sta seguendo l'evoluzione $\Gamma\left(t\right)$ lungo una traiettoria.

L'evoluzione del sistema diventa "speciale": se lascia un particolare stato $\Gamma\left(\tau\right)$ e si sposta nel successivo $\Gamma\left(\tau+1\right)$ un altro sistema arriva dallo stato $\Gamma\left(\tau-1\right)$ per rimpiazzarlo. Il moto assomiglia a una fila di persone che si muove in mezzo a una folla (**Fig.1.2.1**). Ci potrebbero essere diverse processioni di questi sistemi, ciascuna passante attraverso diverse regioni dello spazio delle fasi. Se esiste almeno una traiettoria che attraversa tutti i punti dello spazio delle fasi tale per cui $\rho_{ens}\left(\boldsymbol{\Gamma},t\right)$ è

non nulla (ossia la processione forma un lungo circuito chiuso) allora questo sistema attraverserà tutti gli stati e viene definito *ergodico*. Il tempo necessario a completare tale circuito aumenta al crescere delle dimensioni dello spazio delle fasi.

Secondo la teoria ergodica, se il numero dei sistemi dell'*ensemble* in esame è infinitamente grande, la media temporale di una proprietà del sistema è equivalente alla media istantanea della medesima proprietà: siccome lo stato del sistema è rappresentato con un punto nello spazio delle fasi e vincolato da considerazioni energetiche su una particolare superficie immersa in esso, l'ipotesi ergodica assicura che il punto finirà con l'attraversare tutti i punti della superficie.

Se una simulazione MD iniziasse in una regione ciclica ombreggiata, in **Fig.1.2.1**, sarebbe disastroso: il sistema rimarrebbe "intrappolato" e non riuscirebbe a campionare tutto lo spazio delle fasi a disposizione. In modo simile, regioni che agiscono da barriere, causando strettoie attraverso cui solo poche traiettorie passano, possono derivare da un campionamento "povero" dello spazio delle fasi (simulazioni troppo corte).

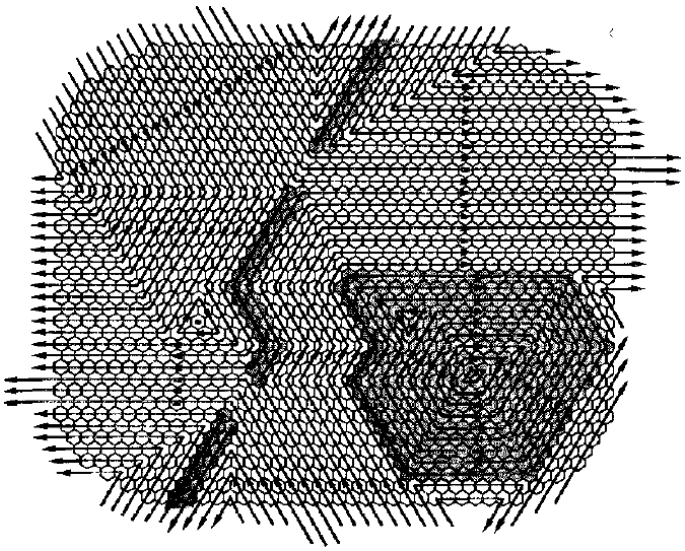


Fig.1.2.1: rappresentazione dello spazio delle fasi. Le celle esagonali rappresentano gli stati Γ. In un sistema ergodico tutte le traiettorie qui sono sezioni diverse di una singola e lunga traiettoria. Le regioni ombreggiate indicano traiettorie cicliche e barriere/strettoie (regioni non ergodiche). [1] (*Fonte: Computer Simulation of Liquids, pag.36, Fig.2.1).*

Spesso è più conveniente usare, al posto di $\rho_{ens}\left(\Gamma\right)$, una funzione non normalizzata $w_{ens}\left(\Gamma\right)$ tale per cui valgano le seguenti espressioni:

$$\rho_{ens}\left(\Gamma\right) = Q_{ens}^{-1} w_{ens}\left(\Gamma\right) \tag{1.2.9}$$

$$Q_{ens} = \sum_{\Gamma} w_{ens}\left(\Gamma\right) \tag{1.2.10}$$

$$\langle A \rangle_{ens} = \sum_{\Gamma} w_{ens}(\Gamma) A(\Gamma) / \sum_{\Gamma} w_{ens}(\Gamma) \tag{1.2.11}$$

Dove il fattore di normalizzazione, Q_{ens}, è detto *funzione di partizione*.

Entrambe $w_{ens}(\Gamma)$ e Q_{ens} possono contenere una costante moltiplicativa arbitraria la cui scelta corrisponde alla definizione dello "Zero di Entropia". È una funzione delle proprietà macroscopiche dell'*ensemble* e la connessione con la termodinamica classica avviene definendo il potenziale termodinamico Ψ_{ens} [1]:

$$\Psi_{ens} = -lnQ_{ens} \tag{1.2.12}$$

Si consideri, per esempio, *l'insieme microcanonico*; la densità di probabilità ρ_{NVE} è proporzionale a:

$$\rho_{NVE} \propto \delta(H(\Gamma) - E) \tag{1.2.13}$$

Qui $H(\Gamma)$ è la funzione Hamiltoniana, Γ è il set delle posizioni e momenti (che rappresenta un punto nello spazio delle fasi), la funzione δ seleziona gli stati di un sistema di N particelle in un volume V che possiedono energia costante E.
La funzione di partizione dell'insieme microcanonico è:

$$Q_{NVE} = \sum_{\Gamma} \delta(H(\Gamma) - E) \tag{1.2.14}$$

Qui la sommatoria prende atto dell'indistinguibilità delle particelle.
Per un sistema atomico, *l'espressione quasi-classica* di Q_{NVE} è:

$$Q_{NVE} = \frac{1}{N!h^{3N}} \int d\mathbf{r}\, d\mathbf{p}\, \delta(H(\mathbf{r},\mathbf{p}) - E) \tag{1.2.15}$$

Qui l'indistinguibilità degli atomi è gestita dal fattore $1/N!$, $\int d\mathbf{r}\, d\mathbf{p}$ è l'integrazione sulle $6N$ coordinate dello spazio delle fasi, h è la costante di Planck. Il potenziale termodinamico relativo all'insieme microcanonico è l'entropia S:

$$S = k_B lnQ_{NVE} \tag{1.2.16}$$

Qui k_B è la costante di Boltzmann.

Un altro esempio è l'*insieme canonico*; la densità di probabilità ρ_{NVT} è proporzionale a:

$$\rho_{NVT} \propto e^{-H(\Gamma)/k_B T} \tag{1.2.17}$$

Qui sono selezionati gli stati di un sistema di N particelle contenute in un volume V costante accoppiato con un termostato a temperatura T costante.
La funzione di partizione è:

$$Q_{NVT} = \sum_{\boldsymbol{\Gamma}} e^{-H(\boldsymbol{\Gamma})/k_B T} \tag{1.2.18}$$

L'espressione quasi-classica di Q_{NVT} è:

$$Q_{NVT} = \frac{1}{N!h^{3N}} \int d\boldsymbol{r}\, d\boldsymbol{p} e^{-H(\boldsymbol{\Gamma})/k_B T} \tag{1.2.19}$$

Il potenziale termodinamico relativo all'insieme canonico è l'energia libera di Helmotz A:

$$A = -k_B T ln Q_{NVT} \tag{1.2.20}$$

In un ensemble canonico tutti i valori di energia sono permessi e le fluttuazioni sono non nulle. Dato che è sempre possibile esprimere l'energia di un sistema come la somma di contributi cinetici e potenziali, eq. (1.2.2), la funzione di partizione può essere fattorizzata in una parte cinetica (gas ideale) e in una parte potenziale (eccesso):

$$Q_{NVT} = \frac{1}{N!h^{3N}} \int d\boldsymbol{p} e^{-K(\boldsymbol{p})/k_B T} \int d\boldsymbol{r} e^{-V(\boldsymbol{r})/k_B T} = Q_{NVT}^{id} Q_{NVT}^{ex} \tag{1.2.21}$$

Se il sistema atomico è ideale, allora la funzione di partizione è:

$$Q_{NVT}^{id} = \frac{h^{3N}}{N!\boldsymbol{\Lambda}_T^{3N}} \tag{1.2.22}$$

Qui $\boldsymbol{\Lambda}_T$ è la lunghezza d'onda termica di De Broglie", definita nel seguente modo:

$$\boldsymbol{\Lambda}_T = \frac{h^2}{\sqrt{2\pi m k_B T}} \tag{1.2.23}$$

MD permette di valutare facilmente le proprietà termodinamiche del gas ideale; la valutazione dei sistemi reali sarà migliore se sarà più accurata la funzione di partizione di configurazione Q_{NVT}^{ex} che dipende dal potenziale d'interazione $V(\boldsymbol{r})$.

Un altro esempio importante, il più studiato, è l'*insieme isotermico-isobaro*; la densità di probabilità ρ_{NPT} è proporzionale a:

$$\rho_{NPT} \propto e^{-(H+PV)/k_B T} \tag{1.2.24}$$

Qui P è la pressione, V è il volume.

La quantità che appare nell'esponente, quando mediata, fornisce l'entalpia termodinamica $\mathrm{H} = H + P\langle V \rangle$. La funzione di partizione corrispondente è:

$$Q_{NPT} = \sum_{\boldsymbol{\Gamma}} \sum_{V} e^{-\left(H+PV\right)/k_B T} = Q_{NVT} \sum_{V} e^{-PV/k_B T} \tag{1.2.25}$$

L'espressione quasi-classica di Q_{NPT} è:

$$Q_{NPT} = \frac{1}{N! h^{3N} V_0} \int dV \int d\boldsymbol{r}\, d\boldsymbol{p} e^{-\left(H+PV\right)/k_B T} \tag{1.2.26}$$

Qui sia $1/N!$ che $1/V_0$ sono coefficienti di normalizzazione.

Il potenziale termodinamico relativo all'insieme isotermico-isobaro è l'energia libera di Gibbs *G*:

$$G = -k_B T \ln Q_{NPT} \tag{1.2.27}$$

Per generare stati in un *ensemble NPT* bisogna, chiaramente, variare sia l'energia che il volume.

Un possibile approccio al calcolo di proprietà termodinamiche consiste nella valutazione di Q_{ens} con l'eq. (1.2.10) [1,2]; questa somma, su tutti gli stati, non è possibile per sistemi multi-corpo: ci sono troppi stati e la maggior parte ha un bassissimo peso sulla statistica dell'*ensemble*. Si potrebbe calcolare l'eq. (1.2.10) escludendo gli stati irrilevanti, però non è possibile una stima di questo tipo. L'idea di generare un set di stati, nello spazio delle fasi, campionato completamente dalla densità di probabilità ρ_{ens} è lo scopo principale della tecnica Monte Carlo.

Analogamente si procede con MD, nel senso che la media sull'ensemble (1.2.11) è rimpiazzata dalla media sulla traiettoria (1.2.4): le equazioni di Newton generano una successione di stati in accordo con la funzione di distribuzione ρ_{ens} per un generico *ensemble*; gli esperimenti in laboratorio si eseguono in condizioni di pressione e temperatura costante (insieme isotermico-isobaro, ρ_{NPT}).

Per essere utile, questa prescrizione deve soddisfare alcune condizioni:

a) La densità di probabilità $\rho_{ens}\left(\boldsymbol{\Gamma}\right)$ per l'*ensemble* d'interesse (soluzione stazionaria) non deve cambiare durante l'evoluzione del sistema.
b) Qualunque distribuzione di partenza $\rho\left(\boldsymbol{\Gamma}\right)$, durante la simulazione, tende alla soluzione stazionaria.

c) Dovrebbe essere possibile sostenere che la teoria ergodica regge, anche se non è possibile sperare di provarlo per sistemi realistici.

Se queste condizioni sono valide, è possibile generare una successione di stati che per tempi lunghi sono campionati in accordo con la desiderata densità di probabilità ρ_{ens}; in tali circostanze, la media sull'ensemble è equivalente alla media sui tempi:

$$A_{obs} \Leftrightarrow \langle A \rangle_{ens} = \frac{1}{\tau_{obs}} \sum_{\tau=1}^{\tau_{obs}} A\left(\boldsymbol{\Gamma}(\tau)\right) \tag{1.2.28}$$

Qui τ è l'indice degli stati campionati, τ_{obs} è il numero degli stati; in una simulazione è un numero molto grande.

1.2.2 Aspetti computazionali

Il protocollo che viene usualmente impiegato per eseguire simulazioni MD di sistemi all'equilibrio prevede i seguenti cinque passaggi [2, 3, 4, 95, 96]:

1. *Definizione delle condizioni del sistema*: il primo importante punto è la scelta dell'*ensemble* termodinamico da simulare, che stabilisce quali informazioni, e in che modo, si possono ricavare dalla funzione di partizione, come descritto nella sezione precedente.
2. *Definizione dei gradi di libertà e delle interazioni tra le particelle del sistema*: nelle simulazioni MD il problema principale è la scelta del potenziale d'interazione tra gli atomi (che genera il campo di forze), cioè della funzione $V = V\left(\boldsymbol{r}_1,\ldots,\boldsymbol{r}_N\right)$ dipendente dalle posizioni dei nuclei. Questa funzione deve essere invariante rispetto alla traslazione e alla rotazione del sistema.

   Che cosa significa veramente potenziale d'interazione? Partendo da una descrizione quantistica del sistema, bisognerebbe considerare tutti i nuclei e gli elettroni e scrivere l'operatore Hamiltoniano, $\hat{H}_{atoms}$

$$\begin{aligned} \hat{H}_{atoms} = & \sum_i \frac{P_i^2}{2M_i} + \sum_{m,n} \frac{p_k^2}{2m_e} + \sum_{i>j} \frac{Z_i Z_j e^2}{\left|\boldsymbol{R}_i - \boldsymbol{R}_j\right|} + \\ & + \sum_{n>m} \frac{e^2}{\left|\boldsymbol{r}_n - \boldsymbol{r}_m\right|} - \sum_{i,n} \frac{Z_i e^2}{\left|\boldsymbol{R}_i - \boldsymbol{r}_n\right|} \end{aligned} \tag{1.2.29}$$

Qui gli indici i, j corrono sui nuclei mentre n, m sugli elettroni; $\boldsymbol{R}_i$, $\boldsymbol{P}_i$ e $\boldsymbol{r}_n$, $\boldsymbol{p}_n$ sono le coordinate e momenti, rispettivamente, dell'i-esimo nucleo e dell'n-esimo elettrone; Z_i è la massa atomica dell'i-esimo nucleo ed M_i la sua massa, m_e è la massa dell'elettrone ed e la sua carica. Dato l'Hamiltoniano definito in eq. (1.2.29) si dovrebbe risolvere l'equazione di Schröedinger dipendente dal tempo e, dalla funzione d'onda, calcolare per ogni istante il valore d'aspettazione delle coordinate nucleari. Questo tipo di calcolo, però, è impraticabile per sistemi con più di una decina di atomi. Secondo l'*approssimazione di Born-Oppenheimer*, che si basa sulla separabilità delle scale dei tempi del moto dei nuclei, più pesanti e quindi più lenti, e quello degli elettroni, è possibile fattorizzare la funzione d'onda del sistema in due termini, nucleare ed elettronico. Nella funzione d'onda nucleare viene persa l'informazione dettagliata sugli elettroni (posizione e velocità), ma rimane solamente un termine di campo medio, parametrico nelle posizioni dei nuclei. Quest'ultimo è il valore di aspettazione del solo Hamiltoniano elettronico (contenente l'energia cinetica degli elettroni e i termini elettrostatici elettrone-elettrone ed elettrone-nucleo). Sebbene l'approssimazione di Born-Oppenheimer permetta di ridurre la dimensionalità del sistema, ottenere il potenziale di campo medio generato dagli elettroni è ancora impraticabile. In generale, è possibile definire l'energia potenziale dalla somma delle interazioni a un corpo, a due corpi, a tre corpi, ecc.:

$$V=V\left(\boldsymbol{r}_1,...,\boldsymbol{r}_N\right)=\sum_i^N V_1\left(\boldsymbol{r}_i\right)+\sum_i^{N-1}\sum_{j>i}^N V_2\left(\boldsymbol{r}_i,\boldsymbol{r}_j\right)+ \\ +\sum_i^{N-2}\sum_{j>i}^{N-1}\sum_{k>j>i}^N V_2\left(\boldsymbol{r}_i,\boldsymbol{r}_j,\boldsymbol{r}_k\right)+... \tag{1.2.30}$$

Qui $\boldsymbol{r}_i$ è la coordinata dell'i-esimo atomo. Il primo termine dell'equazione è la somma delle energie d'interazione di singole particelle con campi esterni, se questi esistono. Il secondo e terzo sono i potenziali dovuti all'interazione tra due e tre particelle. Una volta stabilita la forma per il potenziale elettronico è possibile risolvere numericamente l'equazione di Schrödinger nucleare (derivante dall'approssimazione di Born-Oppenheimer):

$$\left[\sum_i \frac{P_i^2}{2M_i}+V\left(\boldsymbol{R}_i\right)\right]\boldsymbol{\Xi}\left(\boldsymbol{R}_i\right)=\mathrm{E}\boldsymbol{\Xi}\left(\boldsymbol{R}_i\right) \tag{1.2.31}$$

Qui $\Xi(\boldsymbol{R})$ è la funzione d'onda nucleare e gli autovalori, E, dipendono parametricamente dalle coordinate dei nuclei e compongono l'energia potenziale totale del sistema MD. Anche la soluzione numerica di questa equazione richiede un massiccio dispendio computazionale e pone limiti nella dimensione del sistema. Il miglior compromesso è scegliere forme funzionali che mimano il comportamento "vero". La costruzione di un potenziale internucleare implica due stadi:

- Scelta della forma analitica del potenziale; nel passato era una somma di termini a coppie, eq. (1.2.30), oggi nuove forme multi corpo sono testate nel tentativo di catturare il più possibile la fisica e la chimica del legame. Una tipica forma analitica è costituita da un numero di funzioni che dipendono da quantità geometriche come le distanze di legame, angoli torsionali e termini di non legame come i parametri di Lennard Jones e interazioni elettrostatiche [7].
- Trovare una parametrizzazione effettiva per le funzioni che costituiscono la forma analitica scelta; questo passaggio è tecnicamente elaborato e si basa su calcoli quantomeccanici. In tutti i casi, però, è importante avere in mente cosa simulare. A causa delle grandi differenze nelle strutture elettroniche, sarebbe troppo ambizioso voler modellare un metallo e una molecola biatomica con lo stesso potenziale: l'ambiente è totalmente diverso.

Considerazioni particolari vanno fatte per le interazioni elettrostatiche e di non legame perché il loro calcolo determina la durata del tempo necessario a completare la simulazione. Per l'energia di non legame è consuetudine introdurre un raggio di *cut-off* che determina una sfera attorno all'atomo al di fuori della quale le interazioni sono ignorate: questa è un'approssimazione ragionevole per interazioni di Van Der Waals, che decadono rapidamente a grande distanza, ma non sufficiente per le interazioni elettrostatiche che decadono come $1/\boldsymbol{r}$. I metodi di calcolo, ora impiegati per le interazioni elettrostatiche, sono ***E**wald **S**ummation* (ES) e ***P**article-**M**esh-**E**wald* (PME). ES è un caso speciale della sommatoria di Poisson in cui l'energia d'interazione è calcolata nello spazio di Fourier e non in quello reale [2]. PME è una miglioria e consiste nell'applicare ES in posizioni fisse di una

griglia piuttosto che nelle posizioni degli atomi. Entrambi i metodi sono molto veloci, infatti, per *N* particelle ES scala come $N^{3/2}$ e PME come *Nln(N)*. Nello studio di sistemi formati da molecole poliatomiche, la definizione dei gradi di libertà è complicata dalla presenza di coordinate rotazionali, ed eventualmente, per molecole non rigide, di coordinate interne [1]. Da un punto di vista energetico, i sistemi molecolari possono essere descritti da siti localizzati (gruppi di atomi) interagenti mediante potenziali sito-sito, in cui si specifica l'orientazione della molecola (ad es. angoli di Eulero, quaternioni), oppure termini di energia interna vibrazionale, torsionale, interazioni dipolari, quadrupolari, ecc.

3. *Inizializzazione del sistema, equilibrazione e condizioni periodiche al contorno*: per avviare una simulazione si assegnano le posizioni di tutte le particelle, compatibili con la struttura del sistema che si vuole simulare; in ogni caso, le particelle non devono sovrapporsi tra di loro: questo è possibile ponendo le particelle in un reticolo cristallino cubico. Le velocità iniziali, di tutti gli atomi, sono scelte in modo che i termini cartesiani siano distribuiti uniformemente all'interno della distribuzione di Maxwell-Boltzmann e che la quantità di moto totale del sistema si conservi.

   Se la simulazione è avviata da una configurazione geometrica a reticolo, o da una configurazione disordinata con diversa densità e temperatura, è necessario eseguire un'equilibrazione in modo che il sistema perda memoria della configurazione iniziale e inizi a generare configurazioni che rispettano l'*ensemble* statistico scelto; per capire quando il sistema è equilibrato, si analizza l'andamento della pressione ed energia potenziale: se il sistema era in configurazione di reticolo cristallino, a seguito dell'equilibrazione l'energia potenziale cresce, da valori molto negativi, fino a quelli tipici di un liquido denso. In **Fig.1.2.2**, è rappresentato il comportamento istantaneo dell'energia potenziale e pressione di un liquido denso.

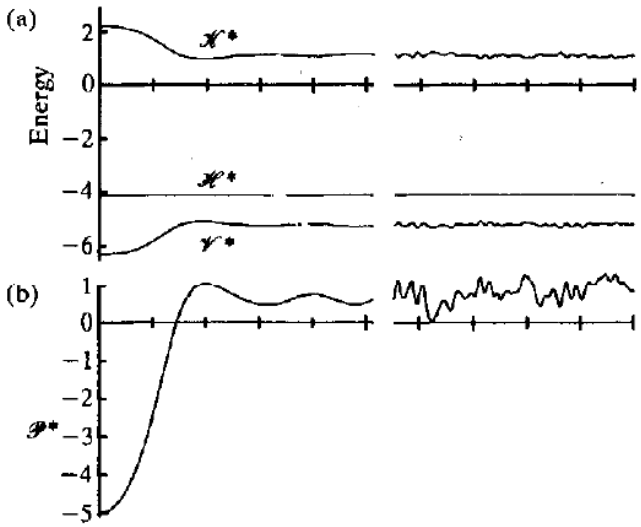


Fig.1.2.2: [1] fase di equilibrazione in una simulazione MD. In dettaglio i primi 50 steps. Il sistema consiste di 108 atomi interagenti via potenziale Lennard-Jones. La configurazione iniziale è un reticolo FCC e distribuzione delle velocità Maxwell Boltzmann. a) Energia totale, cinetica e potenziale; b) Pressione istantanea.
(Fonte: Computer Simulation of Liquids, pag. 172, Fig.5.1.1.)

Il periodo d'equilibrazione deve essere esteso almeno fino a quando queste quantità oscillano attorno a un valore medio.

In simulazioni MD *NPT*, che rappresentano le condizioni canoniche alle quali gli esperimenti avvengono, pressione e temperatura sono controllate accoppiando il sistema al Termostato di Andersen (TA) e al Barostato di Nosé-Hoover (BNH). TA simula un bagno termico che impone al sistema simulato una temperatura fissa. BNH impone una pressione fissa mediante un tampone stocastico.

Le condizioni al contorno sono solitamente di tipo periodico per eliminare, o ridurre il più possibile, gli effetti di superficie: le particelle sono contenute in una scatola ed è possibile immaginare che questa sia replicata in tutte le direzioni cartesiane, producendo un infinito sistema periodico; ciascuna particella interna alla scatola risentirà dell'interazione delle repliche nei volumi circostanti [3]. In altre parole, se una particella è situata in posizione $\boldsymbol{r}$ della scatola, è come se esistesse un numero infinito di particelle situate in:

$$\boldsymbol{r} + l\boldsymbol{a} + m\boldsymbol{b} + n\boldsymbol{c}, \left(l,m,n=-\infty,\infty\right) \qquad (1.2.32)$$

Qui l,m,n sono numeri reali e $\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}$ sono i vettori corrispondenti ai bordi della scatola. In **Fig.1.2.3**, sono raffigurate le condizioni periodiche in un sistema tridimensionale.

Attraverso le condizioni periodiche è possibile evitare l'*effetto superficie* (il sistema, vicino ai bordi, è meno denso rispetto all'interno della scatola); considerazioni particolari sono le simulazioni di clusters di atomi, in cui tali condizioni sono studiate caso per caso.

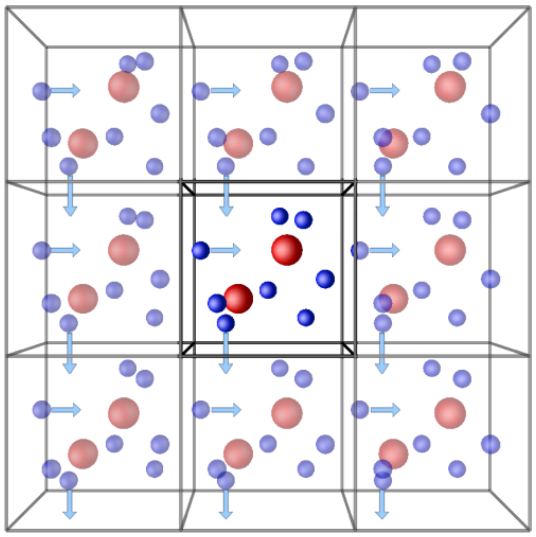


Fig.1.2.3: condizioni periodiche al contorno; una particella, che esce dalla scatola di simulazione da un lato, è reintrodotta dal lato opposto [68].

4. *Scelta del metodo di risoluzione numerica delle equazioni del moto.*

   Il motore di un programma MD è l'algoritmo d'integrazione del tempo richiesto per risolvere le equazioni del moto delle N particelle. Gli algoritmi d'integrazione temporale sono basati sui metodi delle differenze finite, dove il tempo è discretizzato in una griglia e δt è la distanza consecutiva tra due punti. Conoscendo le posizioni e velocità al tempo t, lo schema d'integrazione restituisce le stesse quantità al tempo $t+\delta t$. Iterando la procedura, è possibile seguire l'evoluzione temporale per lungo tempo. È necessario porre particolare attenzione alla scelta del propagatore per evitare errori di troncamento troppo elevati, originati dall'approssimazione alle differenze finite. L'errore dipende in particolare dall'implementazione dell'algoritmo e dal passo temporale, δt , scelto nella simulazione.

   Fra i vari metodi d'integrazione, *l'algoritmo di Verlet* è sicuramente il più importante perché relativamente semplice ed efficiente. L'idea di base è quella di scrivere due serie di Taylor del terzo ordine per le posizioni $\boldsymbol{r}(t)$, una in avanzamento del tempo e una in retrocessione; chiamando $\boldsymbol{v}(t)$ la

velocità, $\boldsymbol{a}(t)$ l'accelerazione e $\boldsymbol{b}(t)$ la derivata terza di $\boldsymbol{r}(t)$ rispetto al tempo, si ottiene:

$$\begin{aligned}\boldsymbol{r}(t+\delta t) &= \boldsymbol{r}(t) + \boldsymbol{v}(t)\delta t + \frac{1}{2}\boldsymbol{a}(t)\delta t^2 + \frac{1}{6}\boldsymbol{b}(t)\delta t^3 \\ \boldsymbol{r}(t-\delta t) &= \boldsymbol{r}(t) - \boldsymbol{v}(t)\delta t + \frac{1}{2}\boldsymbol{a}(t)\delta t^2 - \frac{1}{6}\boldsymbol{b}(t)\delta t^3\end{aligned} \tag{1.2.33}$$

Sommando le due espressioni, si ottiene:

$$\boldsymbol{r}(t+\delta t) = 2\boldsymbol{r}(t) - \boldsymbol{r}(t-\delta t) + \boldsymbol{a}(t)\delta t^2 \tag{1.2.34}$$

Siccome al tempo *t* sono noti $\boldsymbol{a}(t)$, $\boldsymbol{r}(t)$, $\boldsymbol{r}(t-\delta t)$, al tempo $t+\delta t$ si ottiene $\boldsymbol{r}(t+\delta t)$, mentre la velocità al tempo *t* è:

$$\boldsymbol{v}(t) = \frac{\boldsymbol{r}(t+\delta t) - \boldsymbol{r}(t-\delta t)}{2\delta t} \tag{1.2.35}$$

L'algoritmo di Verlet è molto stabile, permette di usare δt dell'ordine del fs, è reversibile e conserva sia l'energia sia la quantità di moto. Altro metodo d'integrazione, molto usato, è l'algoritmo Predictor-Corrector: la fase di predizione calcola una stima approssimativa della quantità desiderata, mentre la fase di correzione affina la prima approssimazione utilizzando altri mezzi [1,2].

5. *Calcolo delle osservabili fisiche, analisi dei dati elaborati e confronto con i dati sperimentali.*

   Il calcolo di osservabili fisiche, in MD, significa eseguire medie sul tempo e funzioni di autocorrelazione temporale delle suddette sulla traiettoria del sistema: le proprietà fisiche sono funzione delle coordinate e velocità degli atomi [24, 25, 26].

   In **Fig.1.2.4**, il protocollo di lavoro in una generica simulazione MD.

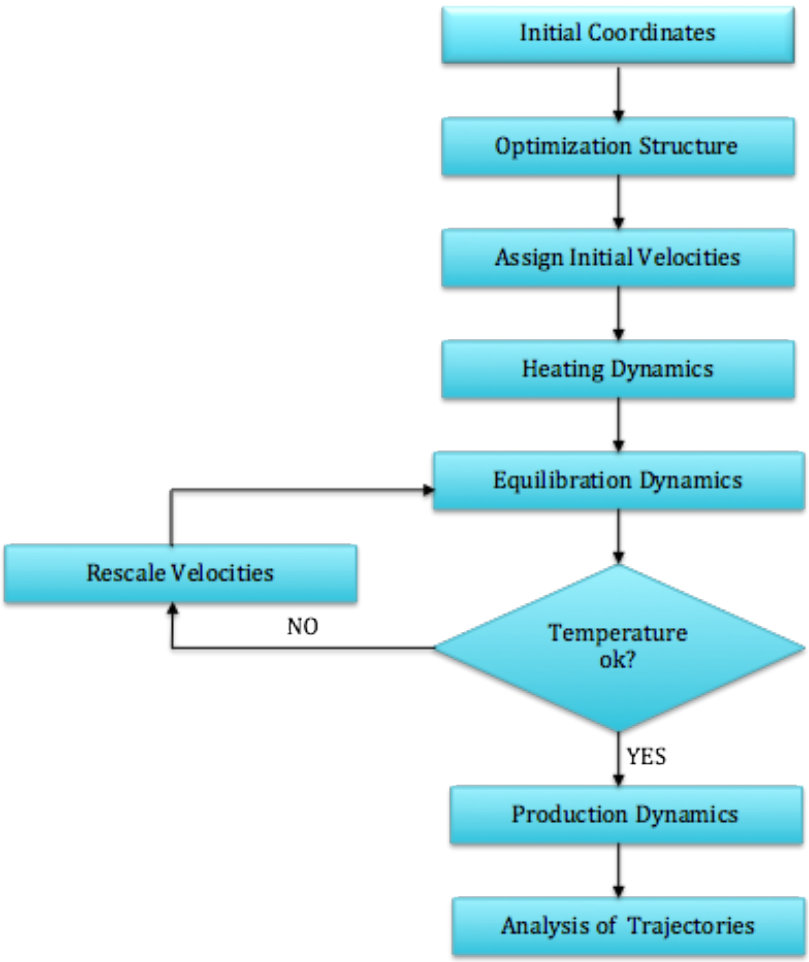


Fig.1.2.4: protocollo di lavoro in una generica simulazione MD.

1.2.3 Limiti delle simulazioni MD

Pur essendo la dinamica molecolare una tecnica molto potente, ha delle limitazioni.

1. Com'è possibile usare le leggi della Meccanica Classica per studiare l'interazione delle particelle, quando si sa che i sistemi in scala atomica obbediscono alle leggi della Meccanica Quantistica?
   [2] Se si considera la lunghezza d'onda termica di De Broglie, eq. (1.2.23), gli effetti quantistici subentrano nelle dinamiche roto-traslazionali di atomi e molecole leggere (es: H_2, D_2, He) o dinamiche vibrazionali con frequenza ν, tale che $h\nu = k_B T$, oppure in sistemi dove la temperatura è sufficientemente bassa (vicino 0 K).
2. Descrizione realistica delle forze: una simulazione MD è realistica solamente quando le forze d'interazione fra particelle sono simili a

quelle reali; spesso sono ottenute come gradiente della funzione d'energia potenziale (quindi dipendono dalla posizione delle particelle): il realismo della simulazione dipenderà dall'abilità della funzione potenziale scelta nel riprodurre il comportamento del sistema sotto le condizioni chimico-fisiche al quale la simulazione sta procedendo.

3. Spesso le simulazioni MD sono fatte in condizioni in cui i limiti spazio-temporali diventano importanti. Una simulazione è "sicura", dal punto di vista temporale, se la sua durata è molto maggiore rispetto ai tempi di rilassamento della quantità che si è interessati a studiare; dal punto di vista spaziale, è utile comparare la dimensione della cella MD con quelle del sistema studiato, utilizzando le funzioni di correlazione.

1.3 Simulazioni MD di biomolecole

Le simulazioni MD furono concepite durante gli anni 50 all'interno della comunità dei fisici teorici [5, 10, 11]. Nel 1957, Alder e Wainwright eseguirono le prime simulazioni MD usando il "modello a sfere rigide": qui, le particelle erano impenetrabili e potevano mimare la repulsione degli atomi solo a piccole distanze. Durante gli anni '70 furono sviluppate metodologie di calcolo per sistemi complessi che culminarono nel 1976 con la prima dinamica molecolare di una proteina. La simulazione interessava l'inibitore pancreatico della tripsina bovina (BPTI): nonostante questo calcolo fu eseguito nel vuoto, con un potenziale poco accurato, per un tempo appena di 9.2 *ps*, i risultati ottenuti permisero di sostituire la visione collettiva delle proteine come strutture relativamente rigide a favore di sistemi dinamici, il cui moto interno gioca un ruolo fondamentale nella dinamica globale. Durante i successivi dieci anni, una vasta gamma di fenomeni dinamici di proteine e amminoacidi fu investigata mediante MD; la maggior parte di questi studi si focalizzava sugli aspetti fisici dei moti interni e l'interpretazione degli esperimenti: ruolo delle dinamiche in parametri spettroscopici misurati (ad es. NMR, *scattering* anelastico neutronico,…), l'effetto del solvente e della temperatura.

Rispetto al primo calcolo MD della BPTI [12], le simulazioni moderne coprono un periodo di produzione molto lungo, dalle decine di *ns* fino al μs, e con un numero di

atomi variabile di 10^4-10^6. I vari processi dinamici che possono essere caratterizzati per le proteine hanno scale temporali che vanno da *fs* a *s*; essi coprono anche un ampio *range* di ampiezze ed energie. La maggior parte di questi moti ha un ruolo critico nelle funzioni biochimiche: moti veloci e locali giocano un ruolo nelle reazioni enzimatiche; moti lenti, che avvengono nella scala temporale dell'intera proteina, includono accoppiamento allosterico e transizioni di *folding*. L'associazione di subunità avviene su distanze ancora più lunghe. In **Tab.1.3.1**, le scale spaziali e temporali caratteristiche della dinamica delle proteine.

event	spatial extent (nm)	amplitude (nm)	time (s)	appropriate simulations
bond-length vibration	0.2−0.5	0.001−0.01	10^{-14}−10^{-13}	QM methods
elastic vibration of globular domain	1.0−2.0	0.005−0.05	10^{-12}−10^{-11}	conventional MD
rotation of solvent-exposed side chains	0.5−1.0	0.5−1.0	10^{-11}−10^{-10}	conventional MD
torsional libration of buried groups	0.5−1.0	0.05	10^{-11}−10^{-9}	conventional MD
hinge bending (relative motion of globular domains)	1.0−2.0	0.1−0.5	10^{-11}−10^{-7}	Langevin dynamics, enhanced sampling MD methods?
rotation of buried side chains	0.5	0.5	10^{-4}−1	enhanced sampling MD methods?
allosteric transitions	0.5−4.0	0.1−0.5	10^{-5}−1	enhanced sampling MD methods?
local denaturation	0.5−1.0	0.5−1.0	10^{-5}−10^{1}	enhanced sampling MD methods?
loop motions	1.0−5.0	1.0−5.0	10^{-9}−10^{-5}	Brownian dynamics?
rigid-body (helix) motions		1.0−5.0	10^{-9}−10^{-6}	enhanced sampling MD methods?
helix−coil transitions		>5.0	10^{-7}−10^{4}	enhanced sampling MD methods?
protein association	≫1.0			Brownian dynamics

Tab.1.3.1: scale temporali caratteristiche dei movimenti delle proteine [11].

Le simulazioni MD odierne possono essere applicate di routine all'investigazione di un ampio *range* di proprietà dinamiche di proteine da ricercatori in numerosi campi, tra i quali biochimica strutturale, biofisica, enzimologia, biologia molecolare, farmaceutica e biotecnologie [5]. Specifici aspetti della biochimica strutturale, cinetica e termodinamica sono la stabilità macromolecolare, proprietà conformazionali e allosteriche, riconoscimento molecolare e proprietà di complessi, trasporto di ioni/molecole, associazione-*folding* di proteine.

Generalmente, è importante ricreare le condizioni sperimentali all'interno della simulazione; nel caso delle proteine, oltre al protocollo discusso nel capitolo precedente, è importante considerare l'ambiente in cui si trovano realmente: a livello odierno di conoscenza e di risorse computazionali, non è possibile caratterizzare totalmente l'ambiente fisiologico di una proteina. I casi non fisiologici che coinvolgono le proteine nel vuoto, o in disposizione d'impacchettamento cristallino, sono relativamente banali; ciò nonostante, per la maggior parte delle simulazioni, è scelto un solvente acquoso.

La maggior parte delle proteine esiste, anche parzialmente, all'interno di un ambiente acquoso; giustificato da questo fatto, è comune assumere che una proteina è totalmente solvatata in acqua pura, o contenente ioni, durante una simulazione. Esistono due modi per simulare la solvatazione di una proteina:

- **Solvente implicito**: è un metodo che consente di rappresentare il solvente come un mezzo continuo, piuttosto che molecole esplicite; è giustificato nei liquidi, dove il potenziale di campo medio può essere applicato per approssimare il comportamento medio delle molecole di solvente (in numero sorprendentemente grande rispetto la proteina): questo comportamento non è necessariamente uniforme e le loro proprietà possono essere descritte da diverse funzioni analitiche. Esistono due tipi di solventi impliciti: modelli basati su aree superficiali accessibili (ASA), storicamente i primi, e modelli elettrostatici continui, ad es. modello generalizzato di Born (GB). ASA opera direttamente con l'energia libera di solvatazione e con l'area superficiale del soluto a contatto del solvente, mentre i metodi elettrostatici solo i termini d'entalpia. GB consente di modellizzare la proteina come set di sfere con costante dielettrica che differisce da quelle a contatto con il solvente.

  Questo metodo è applicabile nella stima dell'energia libera delle interazioni soluto-solvente, nel folding, nel trasporto trans-membrana, ecc.

  La rappresentazione continua del solvente migliora, efficacemente, la velocità di calcolo MD e riduce gli errori statistici che derivano dal campionamento incompleto delle conformazioni del solvente; nonostante ciò, si tratta di un metodo approssimato con alcune limitazioni e problemi riguardanti la parametrizzazione e trattamento degli effetti di ionizzazione.
- **Solvente esplicito**: esiste un'ampia gamma di modelli a disposizione; ciascuno dipende dal numero di punti utilizzati per definire il modello (gli atomi + siti fittizi), se la struttura è rigida o flessibile, se il modello include gli effetti di polarizzazione. Comunemente, i parametri sono aggiustati in modo da riprodurre l'entalpia di vaporizzazione e la densità dell'acqua nelle simulazioni. I modelli più semplici di acqua trattano la molecola come rigida e interagente solo con interazioni di non legame. L'interazione elettrostatica è modellata utilizzando la legge di Coulomb e le forze di dispersione e repulsione con il potenziale di Lennard-Jones. I più popolari includono modelli a tre siti (TIP3P, TIPS, SPC, SPC/E), quattro siti (TIP4P), cinque siti (TIP5P), ecc.

  I modelli a tre siti, i più usati a causa della loro semplicità ed efficienza computazionale, hanno tre punti d'interazione corrispondenti ai tre atomi di una molecola d'acqua. A ognuno è assegnata una carica puntiforme e l'atomo

di ossigeno contiene anche i parametri di Lennard-Jones. La maggior parte di questi utilizza una geometria rigida. Un'eccezione è il modello SPC, che assume una forma ideale tetraedrica (angolo HOH di 109.47°) al posto dell'angolo osservato di 104.5°.

Le caratteristiche principali del modello TIP3P, usato in questa tesi, sono riassunte in **Tab.1.3.2.**

r_{OH}/Å	θ_{HOH}/deg	A·10^{-3}/Kcal Å^{12} mol^{-1}	C/Kcal Å^{6} mol^{-1}	q(O)/e	q(H)/e
0.9572	109.5	629.4	625.5	-0.82	0.41

Tab.1.3.2.: parametri del modello TIP3P [13].

I termini A, C e $q(H)$ sono scelti in modo da restituire dati strutturali ed energetici noti dell'acqua, mentre il termine q(O) è definito $q(O) = -2q(H)$.

I termini A e C sono costanti che rientrano nell'espressione del potenziale d'interazione elettrostatico intermolecolare del solvente, definita [13]:

$$\varepsilon_{mn} = \sum_{i}^{m}\sum_{j}^{n}\frac{q_i q_j e^2}{r_{ij}} + \frac{A}{r_{OO}^{12}} - \frac{C}{r_{OO}^{6}} \qquad (1.3.1)$$

Qui il primo termine è riferito alle interazioni elettrostatiche a lungo raggio, il secondo e il terzo termine corrispondono al potenziale di Lennard-Jones per le forze di dispersione (tra due atomi di ossigeno di molecole di solvente differenti), i pedici *i* e *j* sono riferiti agli *n* e *m* siti di carica elettrostatica.

Il costo computazionale di una simulazione di un sistema di molecole d'acqua aumenta con il numero di siti d'interazione nel modello; il tempo di calcolo per CPU è approssimativamente proporzionale al numero di distanze interatomiche che devono essere calcolate: per il modello TIP3P (tre siti d'interazione) sono necessarie nove distanze per ogni coppia di molecole d'acqua (ogni atomo di una molecola contro ogni atomo di altre molecole).

Al fine di ricreare le condizioni sperimentali in una simulazione MD di proteine, l'accuratezza del potenziale d'interazione è cruciale [5, 15, 14, 7], [6]: solitamente, l'espressione matematica racchiude in sé entrambi i termini di legame e non legame. La specifica decomposizione dei termini dipende dal potenziale, ma una forma generale dell'energia totale in un campo di forze additivo è la seguente:

$$E_{tot} = E_{bond} + E_{angle} + E_{diedral} + E_{electrostatic} + E_{VdW} \qquad (1.3.2)$$

Qui i primi tre termini descrivono le interazioni di *stretching*, *bending* e angoli torsionali:

$$E_{bond} = \sum_{bonds,i} k_i^{bond}\left(r_i - r_i^0\right)^2$$

$$E_{angle} = \sum_{angles,i} k_i^{angle}\left(\theta_i - \theta_i^0\right)^2 \qquad (1.3.3)$$

$$E_{dihedral} = \sum_{dihedral,i} \begin{cases} k_i^{dihedral}\left[1 + cos\left(n_i\phi_i - \varphi_i\right)\right], n_i \neq 0 \\ k_i^{dihedral}\left(0_i - \varphi_i\right)^2, n_i = 0 \end{cases}$$

Qui *bonds* conta ciascun legame covalente nel sistema, *angles* sono gli angoli tra ciascuna coppia di legami covalenti, *dihedral* descrive coppie di atomi separati da tre legami covalenti con il legame centrale soggetto a torsione (ad es. permettono di far rispettare la planarità degli anelli aromatici e altri sistemi coniugati). Un angolo diedro improprio governa la geometria di quattro atomi legati in modo covalente.
In **Fig.1.3.5.** coordinate interne per le interazioni di legame.

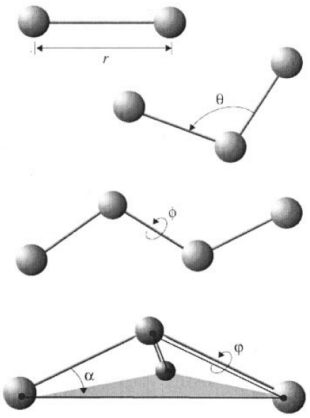


Fig.1.3.5: coordinate interne per interazioni di legame. *r* governa lo stretching dei legami; θ governa gli angoli di legame; ϕ governa gli angoli diedri, α, l'angolo fuori dal piano, è governato dall'angolo improprio φ [7].

I termini di non legame, sono espressi come in eq. (1.3.1). Per ciascun atomo, costituente la proteina, sono definiti i parametri del potenziale (solitamente presenti in files predisposti nei vari software disponibili, come NAMD, CHARMM, AMBER, GROMACS): ad esempio, un campo di forze include parametri distinti per un atomo di ossigeno in un gruppo funzionale carbonilico rispetto uno ossidrilico. Il set di parametri tipico include i valori di massa atomica, raggio di VdW, la carica parziale per singoli atomi, i valori di equilibrio delle lunghezze dei legami, angoli di legame e diedri per coppie, terzine e quartine di atomi legati e i valori corrispondenti della costante di Morse. In aggiunta del potenziale intrinseco descritto dal campo di forza,

[6] è possibile aggiungere forze esterne al sistema (ad es. per guidare la proteina in una configurazione d'interesse particolare).

L'applicazione pratica del calcolo MD, oggigiorno, è alimentata dalla grande disponibilità di software [5-7]: se da un lato molti programmi puntano ad avere un'ampia gamma di funzionalità, dall'altro cercano di differenziarsi su specifiche caratteristiche. Pochi gruppi di ricerca limitano le proprie simulazioni a un singolo pacchetto software. La maggior parte dei pacchetti MD può utilizzare vari campi di forza, strutture e formati di file traiettoria originariamente introdotti in altri pacchetti: questo consente, in qualche modo, di validare e riprodurre facilmente i risultati pubblicati, anche senza il software originale.

Considerando i lavori scientifici presenti nei vari database elettronici (ad es. ISI 'Web of Science'), la maggior parte di essi è stata resa possibile usando uno dei seguenti software: GROMACS, AMBER, CHARMM, NAMD.

- GROMACS [8]: è molto utile nel calcolare interazioni di non legame in lipidi e proteine. I vantaggi principali sono la semplicità d'utilizzo e le notevoli performance nei personal desktop computer. E' distribuito liberamente sotto licenza GNU-Linux.
- AMBER (**A**ssisted **M**odel **B**uilding with **E**nergy **R**efinement) [9]: è una famiglia di campi di forza per calcoli MD di biomolecole, originariamente sviluppata dal gruppo di Peter Kollman, all'Università di San della California, San Francisco. AMBER è anche il nome del pacchetto di softwre MD che consiste in una suite di programmi separati, ciascuno dei quali esegue uno specifico compito. E' scritto con il linguaggio di programmazione C e Fortran 90 e supporta la maggior parte dei sistemi Unix-like. Consente il calcolo parallelo. E' molto usato per simulare DNA, RNA, peptidi, proteine, carboidrati, piccole molecole organiche.
- CHARMM (**C**hemistry at **HAR**vard **M**olecular **M**echanics) [7]: è il nome di un diffuso insieme di campi di forza MD, nonché il nome dei pacchetti di software ad essi associati. Il progetto CHARMM coinvolge una rete di sviluppatori in tutto il mondo che collabora con Martin Karplus e il suo gruppo di Harvard. Le licenze per questo gruppo sono disponibili, a pagamento, per le persone e gruppi di lavoro nel mondo accademico. Questo software consente simulazioni di macromolecole (ad es. proteine, peptidi,

carboidrati, lipidi), tools di analisi traiettorie (ad es. RMSD, funzioni di correlazione). Consente il calcolo parallelo.

- NAMD (**NA**noscale **M**olecular **D**ynamics) [6]: molto utile nel simulare sistemi di grandi dimensioni (> $2 \cdot 10^5$ atomi), esibisce notevoli performance nelle piattaforme di calcolo parallelo ad alto livello con un gran numero di processori, è integrato con il software VMD [44] per la visualizzazione grafica, è compatibile con i campi di forza di CHARMM e AMBER, è implementato con il linguaggio di programmazione C++ *objected-oriented* ed è distribuito gratuitamente.

Il grande passo avanti fatto nel campo MD è senza dubbio dovuto al fenomenale sviluppo dell'hardware dei computer [5]. Il calcolo di sistemi proteici esige strutture di calcolo all'avanguardia: la lunghezza e l'accuratezza delle simulazioni sono legate alla possibilità di usare processori potenti e memorie di grandi dimensioni.

I super computer più potenti, oggi a disposizione, consistono in una schiera di processori che comunicano via connessioni veloci [12], [15]. Il potenziale di questi sistemi, però, può essere utilizzato solo se gli algoritmi calcolati sono partizionati in un numero separato di processi, eseguiti ciascuno su singolo processore; in aggiunta, se i processi non devono essere sincronizzati (ad eccezione del calcolo d'interazioni di non legame) la potenza di elaborazione non sarà sprecata.

In questi ultimi dieci anni, la comunità chimico-fisica computazionale, cercando d'utilizzare calcolatori all'avanguardia, è stata fra i primi a utilizzare il "calcolo GPU" (***G**raphical **P**rocessing **U**nit*). La GPU è una tipologia di coprocessore (ovvero supporta il processore principale CPU) che si contraddistingue per essere specializzata nel *rendering* d'immagini grafiche da un computer. Le GPU moderne, sebbene operino a frequenze più basse delle CPU, sono molto più veloci di esse nell'eseguire i compiti in cui sono specializzate. In questo momento, entrano pesantemente in funzione solo nell'accelerazione grafica 3D: la CPU si occupa solo del calcolo delle coordinate geometriche dei vertici dei poligoni che compongono gli oggetti della scena e lascia alla GPU il compito di riempire le "facce" formate da questi vertici (chiamate in gergo *Mesh*) e del calcolo delle ombre e degli effetti grafici da applicare ai poligoni, sgravandosi da pesanti operazioni di calcolo.

I primi tentativi di mettere a punto algoritmi MD che sfruttino pienamente le potenzialità delle GPU hanno avuto molto successo [15], in alcuni casi fornendo prestazioni due ordini di grandezza più veloci rispetto quelli già raggiunti con le CPU.

Con il rilascio di CUDA (***C**ompute **U**nified Device **A**rchitecture*, architettura di elaborazione realizzata da NVIDIA per il calcolo parallelo sulle GPU), negli ultimi cinque anni sono stati fatti numerosi progressi nello sviluppo e implementazione di algoritmi fondamentali, librerie di calcolo algebrico, protocolli di comunicazione, aritmetica in virgola mobile, fornendo le basi per un'ampia adozione di metodi per sviluppare software scientifici basati sul calcolo GPU.

Una delle più avvincenti applicazioni del calcolo GPU è stata l'accelerazione delle simulazioni MD. Con la disponibilità di CUDA, NAMD [6] è stato il primo pacchetto software a incorporare l'accelerazione GPU: NAMD *benchmarks*, per un sistema virus di 10^6 atomi calcolato al NCSA (***N**ational **C**enter for **S**upercomputing **A**pplications*) dell'Università dell'Illinois (Urbana Champaign), hanno dimostrato che la velocità di simulazione su un massimo di 60 GPU è equivalente a 330 CPU cores (microprocessori) dello stesso cluster [16]. E' stato calcolato il potenziale elettrostatico ES, mediato nel tempo, di una porzione di $6 \cdot 10^5$ atomi di un ribosoma batterico (*T. Thermophilus*), mediante simulazione MD: utilizzando traiettorie di 1 *ns* è scegliendo le strutture da calcolare ogni 1 *ps*, il tempo di calcolo impiegato per 1 CPU è 5.24 CPU/ora mentre per 1 GPU è 0.147 GPU/ora. In **Fig.1.3.6.** i risultati del calcolo

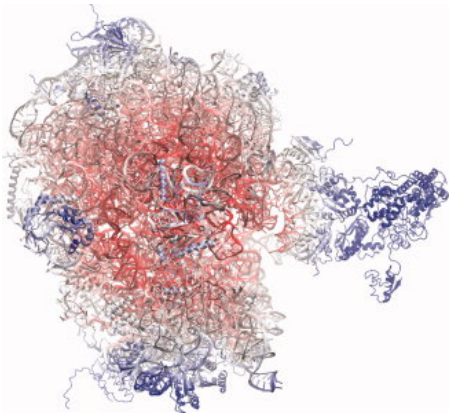


Fig.1.3.6: potenziale elettrostatico medio calcolato via MD per un tempo di 1 ns; il potenziale è stato campionato ogni 1 ps usando tre GPU. Nella scala di colore, il rosso indica i valori del minori di -2500 k_BT/e e il blu i valori maggiori di -1800 k_BT/e, con T= 298.15 K [16].

Le GPU hanno migliorato il rapporto prezzo/prestazione delle simulazioni, lo spazio fisico occupato, la potenza e i sistemi di raffreddamento richiesti per gestire un sistema computer cluster.

In futuro sarà possibile eseguire simulazioni MD in *modalità interattiva* su desktop computer o laptop piuttosto che in *modalità batc*h su clusters (elaborazione senza intervento manuale, *via scripting*).

2 Proprietà diffusive di proteine globulari

2.1 Piano di lavoro

Questo capitolo è organizzato nel seguente modo: nella sezione 2.2 lo stato dell'arte e i problemi riscontrati nella valutazione della diffusione rotazionale di proteine; nella sezione 2.3 una breve descrizione delle proteine analizzate; nella sezione 2.4 i dettagli tecnici delle simulazioni MD; nella sezione 2.5 l'interpretazione teorica della funzione di autocorrelazione del momento angolare e del comportamento bi-esponenziale; nella sezione 2.6 un breve accenno al modello idrodinamico su cui si basa il software DITE e la derivazione delle funzioni di autocorrelazione rotazionale; nella sezione 2.7 presentazione dei risultati ottenuti per le tre proteine. In sezione 2.8, alcune considerazioni conclusive.

2.2 Introduzione

Le proprietà dinamiche delle macromolecole, in particolare proteine, sono rilevanti nella modellazione, interpretazione e comprensione delle caratteristiche chimico-fisiche in soluzione [17-18]. Le molecole che costituiscono qualsiasi materiale si spostano e ruotano nello spazio come tali e possiedono anche moti interni. Gli atomi non sono congelati a posizioni fisse, ma oscillano attorno a tali posizioni. In passato, sono state introdotte metodologie sperimentali e teoriche per ottenere informazioni di tali proprietà. Importanti tecniche sperimentali sono la polarizzazione di fluorescenza [19], dicroismo [20], *dynamic light scattering* [21] e NMR [22], [23]. Dal lato teorico/computazionale, sono stati implementati diversi approcci idrodinamici [24-28] che hanno mostrato di predire bene i dati sperimentali, al costo però di avere un solo grado di libertà (vale a dire il raggio di sfere che identificano atomi o gruppi di atomi). Le proprietà dissipative e di trasporto possono essere ottenute anche da traiettorie di dinamica molecolare.

In questo lavoro ci siamo occupati di calcolare il tensore di diffusione rotazionale di proteine da simulazioni MD. A nostra conoscenza, lo standard impiegato a tale scopo è basato sull'analisi delle funzioni di autocorrelazione rotazionale [17-18], [29-32].

Le macromolecole in soluzione acquosa subiscono una considerevole varietà di movimenti termici: la loro rotazione complessiva è regolata da frequenti collisioni con le molecole di acqua (più leggere). Per una molecola quasi rigida, questo

modello fisico porta a un regime diffusivo di rotazione, in cui l'orientazione di un vettore unitario attaccato alla molecola subisce un cammino casuale sulla superficie di una sfera. Se $P\left(\boldsymbol{\Omega},t\right)$ è la densità di probabilità condizionale che descrive l'orientazione di un sistema di riferimento solidale a una molecola sferica, sarà valida la semplice equazione diffusiva [18], [29]:

$$\frac{\partial P\left(\Omega,t\right)}{\partial t} = D_{rot}\hat{J}^2 P\left(\Omega,t\right) \qquad (2.2.1)$$

Una molecola non sferica ruoterà più velocemente in certe direzioni rispetto altre, in tal modo il coefficiente di diffusione rotazionale D_{rot} diventerà un tensore e l'equazione precedente diventa:

$$\frac{\partial P\left(\Omega,t\right)}{\partial t} = \hat{\boldsymbol{J}}^{\boldsymbol{tr}}\left(\Omega\right)\boldsymbol{D}_{\boldsymbol{rot}}\hat{\boldsymbol{J}}\left(\Omega\right)P\left(\Omega,t\right) \qquad (2.2.2)$$

Questa è l'equazione di Smoluchowski (**Appendice B**). Qui $\boldsymbol{\Omega}$ è il set di angoli di Eulero che specifica l'orientazione della macromolecola rispetto un generico sistema di riferimento, $\hat{\boldsymbol{J}}\left(\boldsymbol{\Omega}\right)$ è l'operatore momento angolare che agisce sull'orientazione $\boldsymbol{\Omega}$.

Nel sistema di riferimento in cui $\boldsymbol{D}_{rot}$ è diagonale, le funzioni di autocorrelazione delle matrici di Wigner hanno espressioni analitiche molto semplici: sono decadimenti multi esponenziale e le costanti di decadimento sono funzioni dei termini diagonali del tensore di diffusione.

Uno dei maggiori problemi nell'applicazione di quest'approccio alle proteine è legato ai lunghi tempi di rilassamento dell'ordine di 10 *ns*, associati al lento tasso di diffusione dell'ordine di 10^6-10^7Hz. Per avere funzioni di autocorrelazioni convergenti, in un tempo di correlazione sufficiente, è necessario calcolare lunghe traiettorie MD, dell'ordine di 100 *ns*. Allo stato attuale, queste traiettorie lunghe sono abbordabili, ma richiedono molto tempo di calcolo (da settimane a mesi di simulazione in sistemi computer clusters). Un altro problema è legato al tipo di modello di acqua usato nella simulazione. Si è visto che i tensori diffusivi calcolati dipendono fortemente dal modello impiegato per descrivere le molecole di acqua: l'accordo con i dati sperimentali scala circa come il rapporto tra la viscosità ottenuta dalla simulazione MD e quella misurata. L'accordo peggiore è stato trovato con il modello d'acqua maggiormente impiegato, il modello TIP3P [13]: a 298 K, tale dà una viscosità bulk 2.5 volte maggiore rispetto la viscosità misurata [33-35] e i tassi di

diffusione calcolati sono sostanzialmente nello stesso rapporto con le misure sperimentali.

La qualità di conformità delle proprietà dissipative, con i valori sperimentali, dipende dalle interazioni acqua-proteina, ad es. il potenziale: l'effetto di un campo di forza non ben definito può crescere con la lunghezza delle traiettorie MD [17], portando a un errato campionamento dello spazio delle fasi per processi lunghi (come le proprietà in regime diffusivo) e conseguentemente a statistiche errate. Questo lavoro non entra nella discussione dei campi di forza dell'acqua: ciò che si è cercato di fare è introdurre un protocollo per ottenere il tensore diffusivo rotazionale di proteine da osservabili che rilassano in tempi molto veloci, in modo che i problemi derivanti dal potenziale d'interazione, per tempi lunghi, non influenzino (o in misura inferiore) la statistica. Il momento angolare è una buona osservabile fisica che, per le proteine, rilassa nella scala dei sub-picosecondi: questo dà la possibilità di calcolare la diffusione rotazionale da traiettorie MD molto corte di 2-3 *ns* e l'intera analisi può essere completata in tempi molto brevi (1 o 2 giorni di lavoro).

Come sarà dimostrato nella sezione 2.4, il protocollo è basato sull'assunzione che il moto rotatorio delle proteine è sempre in regime di alto attrito. Se la proteina può essere descritta approssimativamente come un corpo rigido, le funzioni di autocorrelazione dei termini cartesiani del momento angolare $L_{i=x,y,z}$, espresse nel frame in cui il tensore d'attrito è diagonale, sono dei decadimenti mono-esponenziali e i tempi di autocorrelazione sono $\tau_i = I_{ii} / \xi_{ii}$, dove I_{ii} indica il termine diagonale del tensore d'inerzia e ξ_{ii} indica il termine diagonale del tensore d'attrito; qui si sta assumendo che $\boldsymbol{I}$ e ξ sono collineari.

Per le proteine i tempi di autocorrelazione sono dell'ordine 10^{-1} *ps*, mentre le funzioni di autocorrelazione rilassano entro 1÷10 *ps*: di conseguenza una traiettoria generata da simulazioni MD di 2-3 ns è sufficiente per ottenere buone funzioni di autocorrelazione. Un'altra qualità di quest'approccio è che il tensore d'inerzia scala come R^2 e l'attrito rotazionale scala come R^3, dove R rappresenta la dimensione della proteina: maggiori sono le dimensioni della proteina, minore è il tempo di correlazione e la traiettoria MD necessaria. Questo è in contrasto con il metodo standard che richiede traiettorie lunghe per proteine grandi.

Comunque, esiste un problema interpretativo. Come si vedrà nelle sezioni successive, la funzione di correlazione del momento angolare non è un decadimento

mono-esponenziale: è meglio descritta da una funzione bi-esponenziale. Questo comportamento è stato incontrato nel passato anche per altre osservabili [Kurnikova] e può essere interpretato in termini d'accoppiamento vettoriale del momento angolare con alcune coordinate del sistema, proteina e/o solvente (moti di librazione), che rilassano in tempi vicini; ciò perché la proteina non è un "oggetto rigido". Il punto chiave, per estrarre il tensore diffusivo di una proteina, è postulare che nella funzione bi-esponenziale la scala temporale più veloce è sempre associata al tempo di rilassamento del momento angolare. Fatta quest'assunzione a priori, sarà mostrato come un "trattamento perturbativo" conduce alla forma bi-esponenziale per la funzione di autocorrelazione del momento angolare.

Al fine di confermare questo protocollo, è stata analizzata la diffusione rotazionale di tre proteine globulari perché in letteratura [17-18] sono già state studiate ampiamente: l'inibitore pancreatico della tripsina bovina (BPTI, [36]), il terzo frammento IgG-binding della proteina G (GB3, [37]), il lisozima presente nell'uovo di gallina (LYS, [40]).

Conferme del protocollo sono state ottenute confrontando i dati ottenuti con i coefficienti D_{rot} calcolati tramite il software DITE [26] (basato sull'implementazione di un modello idrodinamico per l'analisi di sistemi molecolari con gradi di libertà interni) e mediante il fitting mono-esponenziale di funzioni di autocorrelazione rotazionale, derivanti dal rilassamento dell'interazione dipolo-dipolo degli spin nucleari ^{15}N e ^{1}H dei legami peptidici degli amminoacidi rigidi e semirigidi che costituiscono la proteina. Per fare ciò, è stato necessario realizzare simulazioni lunghe di 200 *ns* per ciascuna proteina.

2.3 Proteine analizzate

2.3.1 BPTI

L'inibitore pancreatico della tripsina bovina (BPTI) è una delle proteine globulari meglio caratterizzate [36, 69]. La grande stabilità contro la denaturazione (termica, acida o basica) così come le sue dimensioni ridotte, la rendono un sistema modello particolarmente trattabile per studi sperimentali/teorici di proteine globulari. Tali studi includono simulazioni MD (è stata la prima proteina, nel 1977) [11], analisi NMR, ripiegamento delle proteine; la BPTI è stata oggetto d'indagine sulle

interazioni proteina-proteina e riconoscimento molecolare perché forma complessi con diversi enzimi che inibisce. La funzione principale della BPTI è inibire la tripsina e i relativi enzimi proteolitici, tra i quali anche chimotripsina e plasmina.
Questa proteina è costituita da una singola catena polipeptidica che si ripiega in una struttura terziaria compatta, stabile; consiste di cinquantotto amminoacidi (16 tipi diversi). [41] La struttura terziaria è definita da un'ansa β intrecciata e da un'α-elica C terminale. [42] La sequenza degli amminoacidi è: RPDFCLEPPY TGPCKARIIRYFYNAKAGLCQTFVYGGCRAKRNNFKSAEDCMRTCGGA.
L'elevata stabilità della molecola è dovuta alla presenza di tre ponti disolfuro che legano i membri della catena ^{5}C-^{55}C,^{14}C-^{38}C,^{30}C-^{51}C. L'amminoacido ^{15}K è responsabile dell'inibizione proteolitica legandosi nella tasca specifica del sito attivo della tripsina.

2.3.2 GB3

La proteina G è una proteina capace di legarsi alle immunoglobuline e si trova sulla superficie cellulare dei batteri *Streptococchi* [37-9]. Queste proteine hanno attirato particolarmente l'interesse a causa del loro possibile ruolo nel migliorare la virulenza microbica, per studiare l'interazione proteine-proteine e per applicazioni commerciali, come l'etichettatura e purificazione di anticorpi. La proteina G è costituita da tre sequenze omologhe di circa sessanta amminoacidi; le strutture tridimensionali di queste sequenze sono state studiate mediante NMR e metodi cristallografici e s'è visto che formano dei domini. Tutti i domini hanno lo stesso arrangiamento di struttura secondaria: consistono in un'α-elica centrale contro quattro filamenti, antiparallelo-parallelo-antiparallelo, β-sheet.
La topologia dell'avvolgimento del polipeptide è insolita e la struttura più simile nota è quella dell'ubiquitina; questa differisce per la presenza di un loop e di un breve β-sheet tra i filamenti tre e quattro. La preponderanza della struttura secondaria nel dominio, combinata con un core idrofobico ben definito, spiega l'estrema stabilità termica della molecola.
In questo lavoro, è stato analizzato il terzo dominio della proteina G.
La sequenza degli amminoacidi (56) è: MQYKLVINGKTLKGETTTKAVDAET AEKAFKQYANDNGVDGVWTYDDATKTFTVTE.

2.3.3 LYS

Il lisozima è un enzima presente in tessuti animali, dotato di attività battericida, appartenente alle glicosidasi [40]: lisa la parete batterica di alcuni batteri (Gram-positivo), catalizzando l'idrolisi del legame β-1,4 tra l'acido N-acetilmuramico (NAM) e la N-acetilglucosamina (NAG) che sono i componenti principali del peptidoglicano [Yoshimura, Peters].

È abbondantemente presente in numerose secrezioni animali e umane come le lacrime (fanno eccezione quelle dei bovini) e nella saliva. Si trova in concentrazioni elevate anche nell'albume d'uovo. Il LYS, legandosi alla superficie batterica del Gram+, ne riduce la carica elettrica negativa superficiale, rendendo più facile la fagocitosi del batterio, prima che intervengano le opsonine del sistema immunitario.

Il LYS fu descritto la prima volta nel 1922 da Alexander Fleming, scopritore della penicillina. La struttura della molecola è stata descritta, in seguito, da David Chilton Phillips nel 1965, che riuscì a ottenerne un'immagine con una risoluzione di 2 Å [43]. La struttura proteica del LYS è stata la seconda a essere risolta via diffrazione a raggi X e il primo enzima, contenente tutti i venti amminoacidi comuni, a essere interamente sequenziato. Il suo lavoro portò a spiegare come l'attività catalitica sia correlata con la struttura. Il LYS è costituito da 129 amminoacidi. Ha una struttura secondaria con prevalenza di β-sheet, rispetto alle α-eliche. La configurazione tridimensionale della molecola è mantenuta dalla presenza di quattro ponti disolfuro e nella sua parte interna si hanno pochi residui polari. Non ha gruppi prostetici. La sequenza degli amminoacidi è:

KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDG
STDYGILQINSRWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDG
NGMNAWVAWRNRCKGTDVQAWIRGCRL.

Gli amminoacidi ^{35}E e ^{52}D sono i responsabili dell'attività catalitica.

2.4 Protocollo simulazione MD

Nel seguito è presentato un rapporto dettagliato delle simulazioni MD *NPT* effettuate su BPTI, GB3 e LYS. Tutte le traiettorie sono state sviluppate e analizzate con il pacchetto 2.7b2 NAMD [6], VMD1.8.7 [44], con il campo di forza CHARMM [7] *par_all22_prot_cmap*. Le simulazioni sono state eseguite presso il "**L**aboratorio **I**nterdipartimentale di **C**himica **C**omputazionale" (LICC) [75] del Dipartimento di

Scienze Chimiche dell'Università degli studi di Padova, utilizzando il computer cluster AVOGADRO: questo sistema è costituito da 80 nodi interconnessi con *rete di servizio ethernet* (1Gbit) per la gestione dei jobs e *rete InFiniband* (20 Gbit) per il calcolo parallelo MPI; ciascun nodo è costituito da 4 processori Intel Xenon Socket Dual Core da 266 GHz. Sono presenti 8 Gbyte RAM.

Lo scopo di questa sezione è di tipo tecnico operativo, volto a fornire sufficienti informazioni per una completa riproduzione degli esperimenti numerici.

Le strutture di riferimento delle proteine sono state prese dal sito-database *www.rcsb.org*. Il *fileformat pdb*[1] contiene in prima approssimazione l'elenco degli atomi presenti, le loro posizioni, etc. Questi dati, ottenuti grazie alla cristallografia a raggi X e/o a spettroscopia NMR, depositati da biologi e chimici di tutto il mondo, sono di pubblico dominio e accessibili gratuitamente. È stato impiegato il seguente protocollo standard per tutte le simulazioni:

i. Generazione del file *psf*.
ii. Addizione del solvente.
iii. Addizione dei controioni.
iv. Ottimizzazione geometria proteina.
v. Riscaldamento.
vi. Equilibrazione del solvente.
vii. Dinamica Molecolare.

Le varie operazioni sono state eseguite secondo le seguenti regole.

i. Generazione del file *psf*

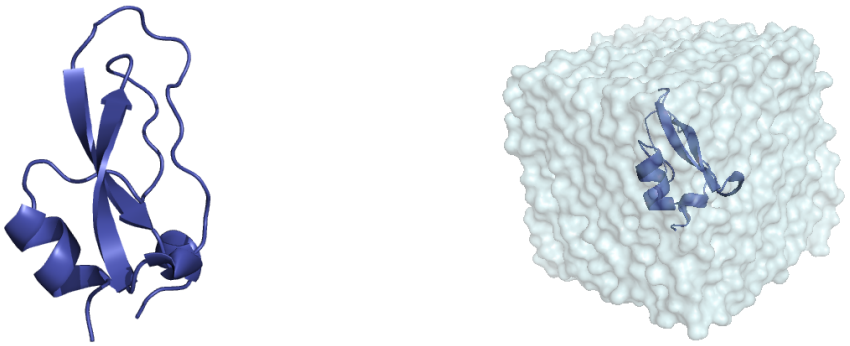
Il formato *psf* contiene al suo interno le informazioni riguardanti la connettività tra gli atomi quali distanze e angoli di legame, le loro masse, il potenziale d'interazione, le strutture degli amminoacidi etc., ed è necessario affinché il programma riesca a interpretare correttamente i dati contenuti nel file principale *pdb*. La generazione dei file *psf* è effettuata tramite VMD utilizzando il comando *psfgen* con uno script opportuno. La sintassi dello script utilizzato per la BPTI è riportata in **Appendice C1**.

ii. Addizione del solvente

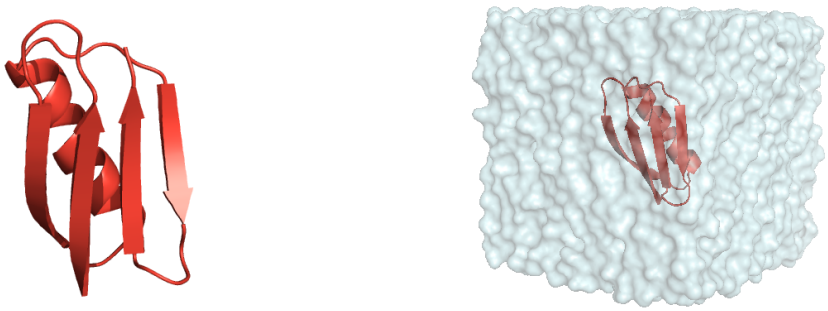
La proteina è inserita in una scatola, di volume scelto, contenente H_2O. Le informazioni riguardo al modello del solvente sono contenute all'interno del file *top_all22_prot_cmap.inp*, che contiene anche le informazioni sul campo

[1] Informazioni dettagliate sul *fileformat pdb* si possono reperire all'indirizzo *http://www.wwpdb.org/docs.html*.

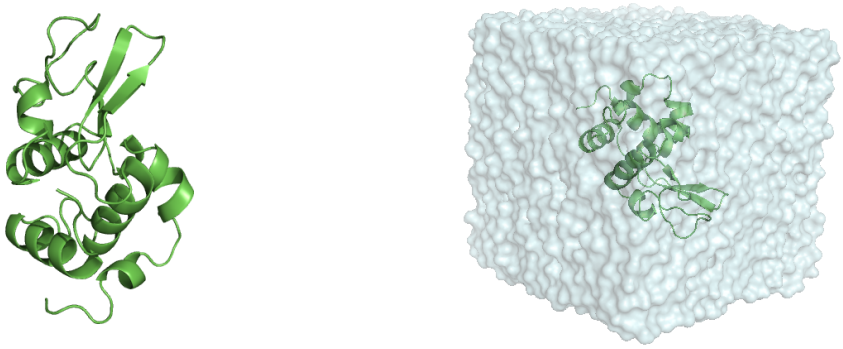
di forza che descrive le interazioni interatomiche. Nel caso in esame, il solvente è esplicito di tipo TIP3P. Di seguito, in **Fig.2.4.1,** sono raffigurate le proteine prima e dopo il processo di solvatazione; la sintassi dello script utilizzato è riportata in **Appendice C2**.



BPTI



GB3



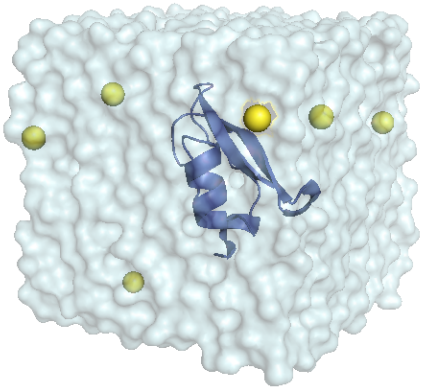
LYS

Fig.2.4.1: proteine nel vuoto e solvatate.

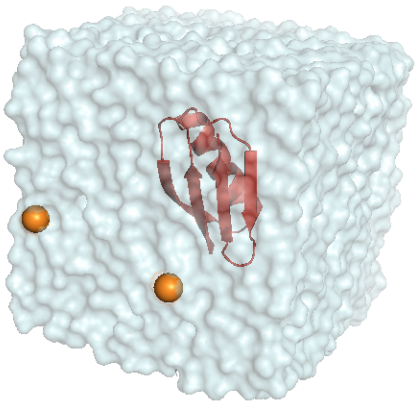
iii. Addizione dei controioni

Per assicurare che il sistema sia elettricamente neutro, è necessario aggiungere dei controioni per bilanciare le cariche dei residui amminoacidici ionizzabili. VMD permette quest'operazione, aggiungendo il numero necessario di ioni Na^+ e Cl^-, tramite il comando "autoionize".

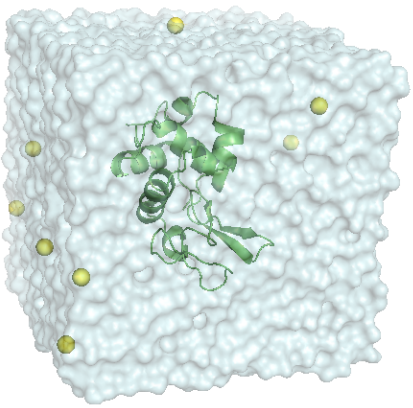
Di seguito, in **Fig.2.4.2** le proteine, dopo il processo di neutralizzazione della carica; la sintassi dello script utilizzato è riportata in **Appendice C3**:



BPTI



GB3



LYS

Fig.2.4.2: proteine solvatate con carica neutralizzata. Na^+ arancione, Cl^- giallo.

iv. Ottimizzazione geometria sistema (proteina + solvente +controioni)

Per definire le condizioni iniziali "naturali", è stato fatto un calcolo di meccanica molecolare per minimizzare le forze di cui risentono gli atomi; siccome le forze sono conservative, esse sono generate dal potenziale, perciò

minimizzare le forze è equivalente a ricercare la disposizione strutturale di minima energia del sistema. Il file d'input è riportato in **Appendice C4**.

v. Riscaldamento

La simulazione MD è stata eseguita alla temperatura di 300 K; è stato necessario portare il sistema a tale temperatura così da ottenere una distribuzione iniziale delle velocità basata sull'equazione di Maxwell Boltzmann:

$$f\left(v\right)=\left(\frac{1}{2\pi mk_BT}\right)^{3/2}e^{-mv^2/k_BT} \qquad (2.4.1)$$

Nei casi esaminati è valida l'assunzione di sistema isotropo, perciò l'energia cinetica media è:

$$\frac{3}{2}m\left\langle v^2\right\rangle=\int_{-\infty}^{\infty}f\left(v\right)dv=\frac{3}{2}k_BT \qquad (2.4.2)$$

Qui $\left\langle v^2\right\rangle$ è la velocità quadratica media degli atomi del sistema.

Il file d'input è riportato in **Appendice C4**.

vi. Equilibrazione del solvente

In fase di equilibrazione la proteina è stata bloccata nella posizione originale mentre è stata eseguita una dinamica sul solvente per consentirne la disposizione ed eliminare eventuali tensioni dovute, per esempio, ad atomi troppo vicini. Nel nostro caso il calcolo è durato 6.1 ns. Il file d'input è riportato in **Appendice C4**.

vii. Dinamica Molecolare

A questo punto è stato possibile lanciare le simulazioni nel computer cluster Avogadro, utilizzando 32 processori in parallelo.

Per le traiettorie corte, a causa dei tempi veloci di rilassamento delle proteine, sono state raccolte le conformazioni molecolari ogni 25 steps MD (50 *fs*). Per le traiettorie lunghe, invece, le conformazioni molecolari sono state raccolte ogni 250 steps MD (500 *fs*). Per ragioni tecniche (dimensioni dei file di output), è stato modificato il codice di NAMD per far stampare solo le traiettorie delle proteine; ciascun calcolo (traiettoria proteina) è stato diviso in blocchi da 5 *ns* e i dati prodotti occupano circa 5 Gbyte. Il file d'input è riportato in **Appendice C4**.

I parametri delle simulazioni sono riportati in **Tab.2.4.1**, di seguito.

	BPTI	LYS	GB3
Protein Data Bank File	6PTI	2LZT	1P7E
Carica Proteina	+6	+8	-2
Numero molecole di acqua	2642	6845	3921
Modello molecola di acqua	TIP3P		
Dimensione scatola periodica	44.0 Å	60.0 Å	50.0 Å
Ensemble	N (8802 atomi)	N (22487 atomi)	N (12621 atomi)
	P (1 atm)	P (1 atm)	P (1 atm)
	T (300 K)	T (300 K)	T (300 K)
Termostato	Temperature coupling		
Barostato	Nosé-Hoover Langevin piston (piston period 200 fs, piston decay 100 fs, piston temperature 300 K)		
Cut-Off interazioni di non legame	12 Å, smoothing switch at 10 Å		
Pair list distance	13.5 Å		
Elettrostatica	PME		
Time step d'integrazione	2 fs		
Frequenza di salvataggio coordinate e velocità	25 MD steps (corta) 250 MD steps (lunga)		
Periodo di equilibrazione	6.1 ns	6.1 ns	6.1 ns
Periodo di produzione	3 ns (corta) 200 ns (lunga)	3 ns (corta) 200 ns (lunga)	3 ns (corta) 200 ns (lunga)
Velocità di calcolo NAMD con 32 CPU in parallelo	0.07 giorni/ns (lunga)	0.08 giorni/ns (lunga)	0.14 giorni/ns (lunga)

Tab.2.4.1: Parametri simulazione MD per le tre proteine.

2.5 Interpretazione teorica

Se il momento angolare globale è disaccoppiato dal resto del sistema di coordinate, la sua funzione di autocorrelazione è:

$$\left\langle \boldsymbol{L}(0)\boldsymbol{L}(t)\right\rangle = \sum_{i=x,y,z}\left\langle L_i \left| e^{-\hat{\Gamma}_L t}\right| L_i P_{eq}(\boldsymbol{L})\right\rangle \qquad (2.5.1)$$

Qui L_i è l'i-esimo termine del momento angolare, $P_{eq}(\boldsymbol{L})$ è la distribuzione d'equilibrio, $\hat{\Gamma}_L$ è l'operatore di Kramers-Klein definito:

$$\hat{\boldsymbol{\Gamma}}_L = -k_B T \frac{\partial}{\partial \boldsymbol{L}}^{tr} \boldsymbol{\xi} P_{eq}(\boldsymbol{L}) \frac{\partial}{\partial \boldsymbol{L}} P_{eq}^{-1}(\boldsymbol{L}) \qquad (2.5.2)$$

La distribuzione d'equilibrio può essere fattorizzata nel sistema di riferimento che rende diagonale il tensore d'inerzia $\boldsymbol{I}$ e diventa:

$$P_{eq}\left(\boldsymbol{L}\right)=P_{eq}\left(L_x\right)P_{eq}\left(L_y\right)P_{eq}\left(L_z\right)=\prod_i e^{-L_i^2/2I_{ii}k_BT} \tag{2.5.3}$$

Se è possibile assumere che il tensore d'attrito è diagonale nello stesso sistema di riferimento che rende diagonale il tensore d'inerzia, allora l'operatore in eq. (2.5.2) diventa:

$$\hat{\Gamma}_L=-k_BT\sum_i \xi_{ii}\frac{\partial}{\partial L_i}^{tr} P_{eq}\left(L_i\right)\frac{\partial}{\partial L_i}P_{eq}^{-1}\left(L_i\right) \tag{2.5.4}$$

È possibile dimostrare che l'autofunzione di quest'operatore è $\left|L_iP_{eq}\left(\boldsymbol{L}\right)\right\rangle$:

$$\hat{\Gamma}_L\left|L_iP_{eq}\left(\boldsymbol{L}\right)\right\rangle=\frac{\xi_{ii}}{I_{ii}}\left|L_iP_{eq}\left(\boldsymbol{L}\right)\right\rangle=\frac{1}{\boldsymbol{\tau}_i}\left|L_iP_{eq}\left(\boldsymbol{L}\right)\right\rangle \tag{2.5.5}$$

Cosi, ogni addendo nella somma sul lato destro di eq. (2.5.1) è:

$$C_{i=x,y,z}\left(t\right)=\left\langle L_i\left|e^{-\hat{\Gamma}_Lt}\right|L_iP_{eq}\left(\boldsymbol{L}\right)\right\rangle=\left\langle L_i\left(0\right)^2\right\rangle e^{-t/\tau_i} \tag{2.5.6}$$

La funzione di autocorrelazione del momento angolare può essere calcolata come funzione bi-esponenziale:

$$C(t)=A^2e^{-t/\tau_1}+\left(1-A^2\right)e^{-t/\tau_2} \tag{2.5.7}$$

Qui $\tau_1\left(<\tau_2\right)=I/\xi$: τ_1 è il tempo di correlazione riferito alla dinamica più veloce (momento angolare globale) mentre τ_2, più grande, è riferito al moto più lento.

2.6 Metodi comparativi

2.6.1 Modello idrodinamico

La valutazione di proprietà tensoriali di molecole di grandi dimensioni (ad es. proteine globulari) è fatta con simulazioni MD o metodi mesoscopici. Dal lavoro pionieristico di Bloomfield [45], diversi autori hanno descritto principi e applicazioni di approcci idrodinamici. Fondamentalmente, la molecola è suddivisa in un insieme di sfere scelte in modo da descrivere la forma globale nel miglior modo possibile: ciascuna sfera può rappresentare un singolo atomo o un gruppo di atomi (ad es. gruppo metilico). Il tensore di diffusione rotazionale complessivo è dato da argomenti classici [27]. In **Fig.2.6.1** questo modello è applicato alla proteina LYS.

Una notevole implementazione di questi principi è prevista nel codice HYDRONMR: più efficacemente, le sfere possono essere scelte per descrivere solo la superficie esterna delle molecole (ad es. la parte esposta al solvente). Un percorso alternativo è la definizione di un'equivalente descrizione della forma molecolare, basata su un ellissoide triassiale.

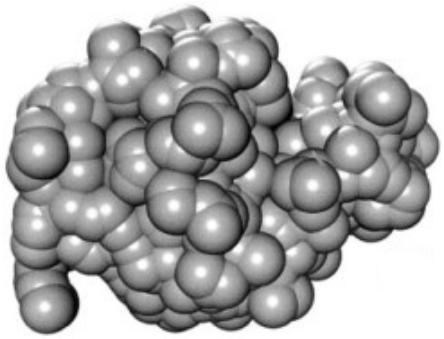


Fig.2.6.1: Modello a sfere della LYS; ciascuna rappresenta un atomo [27].

DITE [26] è l'implementazione di un modello idrodinamico per la valutazione diretta di proprietà tensoriali di diffusione e attrito, applicabile a molecole flessibili i cui gradi di libertà interni sono rappresentati come angoli torsionali [24-25]; nel nostro caso, le coordinate interne delle proteine sono state "congelate" al fine di rendere tali oggetti rigidi e anisotropi.

Il modello, semplicemente, assume gli atomi che costituiscono la proteina come sfere, immerse in un fluido con basso numero di Reynolds, interagenti con forze idrodinamiche: questo è chiamato approccio di Rotne-Prager [46]. Le sfere hanno un volume definito dal raggio di Van Der Waals (è stato utilizzato lo standard UA0 [70-71]).

Gli elementi di $\boldsymbol{D}_{rot}$ dipendono da una componente tensoriale geometrica $\bar{\boldsymbol{D}}$ e dal coefficiente di diffusione per una sfera isolata D_0, definito dall'equazione di Stokes-Einsten:

$$\boldsymbol{D}_{rot} = \left(\boldsymbol{B}^{tr}\boldsymbol{d}^{-1}\boldsymbol{B}\right)^{-1} = \frac{k_B T}{C R_e \eta \pi}\bar{\boldsymbol{D}} = \frac{k_B T}{\Xi_0}\bar{\boldsymbol{D}} = D_0\bar{\boldsymbol{D}} \qquad (2.6.1)$$

Qui $\boldsymbol{B}$ è la matrice geometrica (la cui forma dipende dalla topologia della molecola e dallo schema numerico scelto per definire i gruppi di atomi), $\boldsymbol{d}$ è il tensore di diffusione libero (sistema senza vincoli), η è il coefficiente di viscosità della proteina, R_e il raggio effettivo medio delle sfere (raggio di Van der Waals), T è la temperatura del sistema, C una costante idrodinamica che dipende dalle condizioni al

contorno. Solitamente C=6 (condizione *stick boundary* [47-48]: la velocità tangenziale del fluido sulla superficie della sfera è uguale alla velocità locale della superficie della sfera).

Gli elementi del tensore libero $\boldsymbol{d}$ sono:

$$\begin{cases} d_{ii} = \dfrac{\mathrm{k_B T}}{\Xi_0}\boldsymbol{I}_3 \\ d_{ij} = \dfrac{k_B T}{\Xi_0}\dfrac{3R_e}{4r_{ij}^3}\left[\left(r_{ij}^2 + \dfrac{2}{3}R_e^2\right)\boldsymbol{I}_3 + \left(1 - 2\dfrac{R_e^2}{r_{ij}^2}\right)\boldsymbol{r}_{ij} \otimes r_{ij}\right], r_{ij} > 2R_e \\ d_{ij} = \dfrac{k_B T}{\Xi_0}\left[\left(1 - \dfrac{9}{32}\dfrac{r_{ij}}{R_e}\right)\boldsymbol{I}_3 + \dfrac{3}{32}\dfrac{\boldsymbol{r}_{ij} \otimes \boldsymbol{r}_{ij}}{r_{ij}^2}\right], r_{ij} < 2R_e \end{cases} \qquad (2.6.2)$$

Qui i e j indicano due atomi generici, $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$, il simbolo $\otimes$ indica il prodotto diadico. I termini del tensore rotazionale sono stati ottenuti da conformazioni molecolari estratte dalle simulazioni lunghe.

2.6.2 Funzioni di autocorrelazione rotazionale

Il coefficiente di diffusione rotazionale D_{rot} di proteine può essere calcolato dalla corrispondente funzione di autocorrelazione rotazionale ottenuta con tecniche NMR di rilassamento [49-53]. Dato che i parametri di rilassamento sono molto sensibili al moto complessivo di rotolamento e all'anisotropia della proteina, D_{rot} ottenuto usando questa tecnica è molto più affidabile rispetto a quello ottenuto dalla relazione di Debye-Stokes-Einstein, già vista nell'esperienza con DITE.

Mediante esperimenti NMR multidimensionali [49], è possibile stimare il tempo di rilassamento longitudinale (o spin-reticolo) T_1, il tempo di rilassamento trasversale (o spin-spin) T_2 e l'Effetto Nucleare Overhauser (NOE), per spin nucleari ^{15}N, legati a ^{1}H, che interagiscono tramite interazioni dipolari; questi parametri sono espressi dalle seguenti equazioni:

$$\begin{cases} \mathrm{T}_1 = \dfrac{d}{4}\left[J\left(\omega_\mathrm{H}-\omega_\mathrm{N}\right)+3J\left(\omega_\mathrm{N}\right)+6J\left(\omega_\mathrm{H}+\omega_\mathrm{N}\right)\right]+cJ\left(\omega_\mathrm{N}\right) \\ \mathrm{T}_2 = \dfrac{d}{8}\left[4J\left(0\right)+J\left(\omega_\mathrm{H}-\omega_\mathrm{N}\right)+3J\left(\omega_\mathrm{N}\right)+6J\left(\omega_\mathrm{H}\right)+6J\left(\omega_\mathrm{H}+\omega_\mathrm{N}\right)\right]+ \\ +\dfrac{c}{6}\left[4J\left(0\right)+3J\left(\omega_\mathrm{N}\right)\right] \\ \mathrm{NOE} = 1+\dfrac{d}{R_1}\dfrac{\gamma_\mathrm{H}}{\gamma_\mathrm{N}}\left[6J\left(\omega_\mathrm{H}+\omega_\mathrm{N}\right)-J\left(\omega_\mathrm{H}-\omega_\mathrm{N}\right)\right] \end{cases} \tag{2.6.3}$$

Qui *d* e *c* sono coefficienti, definiti:

$$\begin{cases} d = \dfrac{\mu_0^2}{4\pi^2}\dfrac{\hbar^2\gamma_\mathrm{N}^2\gamma_\mathrm{H}^2}{r_\mathrm{NH}^6} \\ c = \dfrac{\omega_\mathrm{N}^2\Delta\sigma_\mathrm{N}^2}{3} \end{cases} \tag{2.6.4}$$

Qui μ_0 è la permeabilità magnetica nel vuoto, $\hbar$ è la costante di Planck, γ_X è il rapporto giromagnetico del nucleo X, r_NH è la distanza internucleare tra i nuclei ^{15}N e ^{1}H, $\Delta\sigma_\mathrm{N}$ è l'anisotropia chemical shift del nucleo ^{15}N. La densità spettrale $J\left(\omega\right)$, è definita come la trasformata di Fourier della funzione di correlazione $C\left(t\right)$:

$$J\left(\omega\right) = \int_0^\infty C\left(t\right)cos\left(\omega t\right)dt \tag{2.6.5}$$

É evidente che i parametri di rilassamento, nelle equazioni (2.6.3), dipendono da $J\left(\omega\right)$ e dalle distanze internucleari r_NH . $J\left(\omega\right)$ caratterizza la modulazione dell'accoppiamento dipolare tra i nuclei ^{15}N e ^{1}H nel tempo, causata dalle fluttuazioni delle distanze nucleari r_NH . NOE e i tempi di rilassamento T_1 e T_2, dovuti all'interazione dipolo-dipolo tra due nuclei, possono essere calcolati mediante la seguente funzione di autocorrelazione rotazionale temporale:

$$C\left(t\right) = \left\langle P_l\left[\hat{\mu}^\mathrm{LF}\left(0\right)\hat{\mu}^\mathrm{LF}\left(t\right)\right]\right\rangle \tag{2.6.6}$$

Qui $\hat{\mu}^\mathrm{LF}\left(t\right)$ è un versore dipendente dal tempo, definito nel sistema di riferimento inerziale di laboratorio, che si trova lungo il vettore del legame peptidico ^{15}N-^{1}H, $P_l\left(x\right)$ è il polinomio di Legendre di ordine *l*. Le parentesi indicano che la media temporale è su tutti i versori del sistema. Il tasso di decadimento di $C\left(t\right)$, calcolato con i polinomi di Legendre di ordine $2\left(P_2\left(x\right)\right)$, è inversamente proporzionale al

coefficiente di diffusione rotazionale D_{rot} del versore considerato. Il polinomio $P_2(x)$ è definito:

$$P_2(x) = \frac{1}{2}\left(3x^2 - 1\right) \tag{2.6.7}$$

Le funzioni di correlazione sono i termini chiave per ottenere tempi di rilassamento di spin, sia per le tecniche sperimentali sia per i metodi teorici.

Assumendo le proteine come oggetti rigidi che ruotano in un ambiente isotropo (solvente acquoso), la funzione di correlazione $C(t)$ può essere espressa dal seguente fitting mono-esponenziale:

$$C(t) = e^{-t/\tau_{rot}} = e^{-l(l+1)D_{rot}t} \tag{2.6.8}$$

Quest'equazione è valida quando un oggetto è considerato come un corpo rigido [50]. Qui *l* è l'ordine del polinomio $P_l(x)$ e τ_{rot} è il tempo di correlazione, definito:

$$\tau_{rot} = \int_0^\infty C(t)\,dt \tag{2.6.9}$$

2.7 Risultati ottenuti

In letteratura sono disponibili i valori sperimentali isotropi del tensore di diffusione rotazionale: in particolare per BPTI, LYS e GB3 i valori di D_{rot}^{exp} sono, rispettivamente, $4.3 \cdot 10^7$ Hz, $2.4 \cdot 10^7$ Hz e $5.5 \cdot 10^7$ Hz [18], [55].

L'analisi delle traiettorie è stata fatta con algoritmi sviluppati dal gruppo di Chimica Teorica.

2.7.1 Protocollo momento angolare globale

Sono state analizzate le traiettorie corte (3 *ns*) MD delle tre proteine.

Per calcolare le funzioni di autocorrelazione del momento angolare e i termini D_{ii} del tensore di diffusione rotazionale, è stato seguito il seguente protocollo:

1. È stato calcolato il tensore d'inerzia, delle proteine, in ogni struttura. CHARMM restituisce i termini principali del tensore e gli autovettori ogni 50 *fs*. La sintassi dello script utilizzato è riportata in **Appendice D1**.

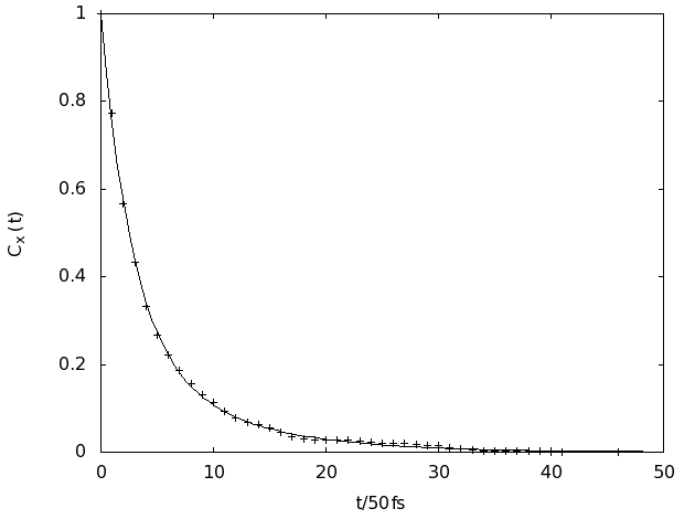
2. È stato rimosso il centro di massa di tutte le coordinate e velocità degli atomi che compongono le proteine. La sintassi dello script utilizzato è riportata in **Appendice D1**.
3. È stato calcolato il momento angolare globale nel frame centrato sul centro di massa. Usando gli autovettori ottenuti nello step 1, è stato possibile ottenere l'espressione del momento angolare nel frame che rende diagonale il tensore d'inerzia. La sintassi dello script utilizzato è riportata in **Appendice D1**.
4. Sono state calcolate le funzioni di autocorrelazione per le tre traiettorie. La sintassi dello script utilizzato è riportata in **Appendice D2**.
5. È stato fatto il fitting delle funzioni di autocorrelazione con la seguente funzione bi-esponenziale:

$$C(t) = A^2 e^{-t/\tau_1} + \left(1 - A^2\right) e^{-t/\tau_2} \qquad (2.7.1)$$

Qui $\tau_1 \left(< \tau_2\right) = I / \xi$.

Per calcolare i coefficienti d'attrito, dai tempi di autocorrelazione, sono stati considerati i coefficienti d'inerzia come media sulle traiettorie.

Le funzioni di autocorrelazione delle proteine e il fitting bi-esponenziale, per i termini spaziali $L_{i=x,y,z}$, sono visualizzate di seguito.



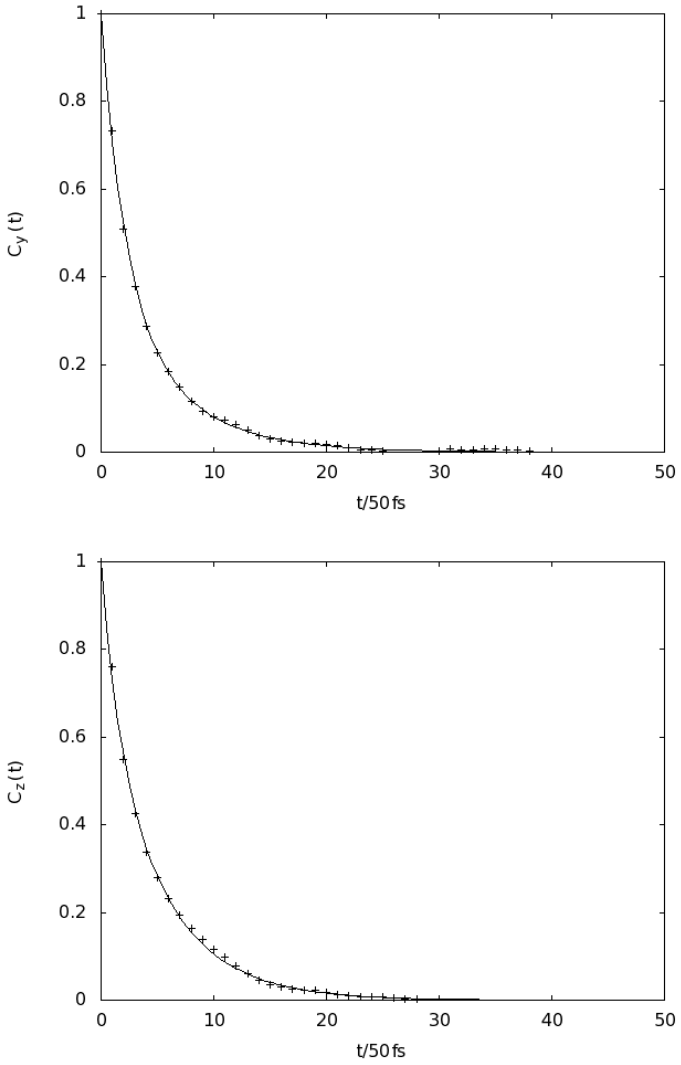
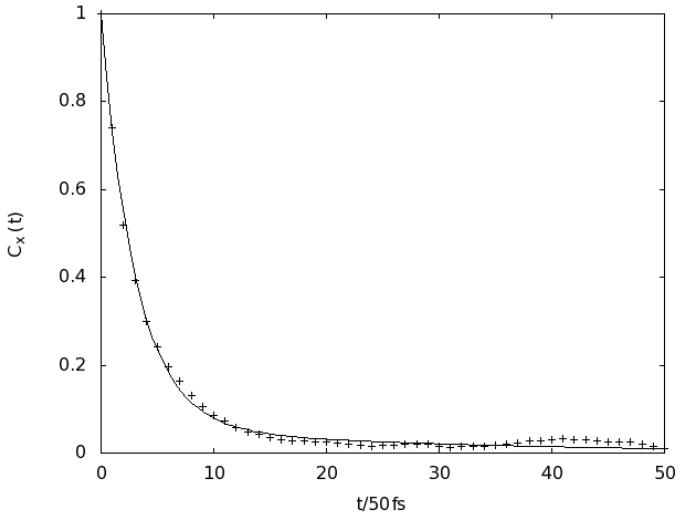


Fig.2.7.1: Funzioni di autocorrelazione e fitting bi-esponenziale, per i termini L_i ($i = x, y, z$) della BPTI.



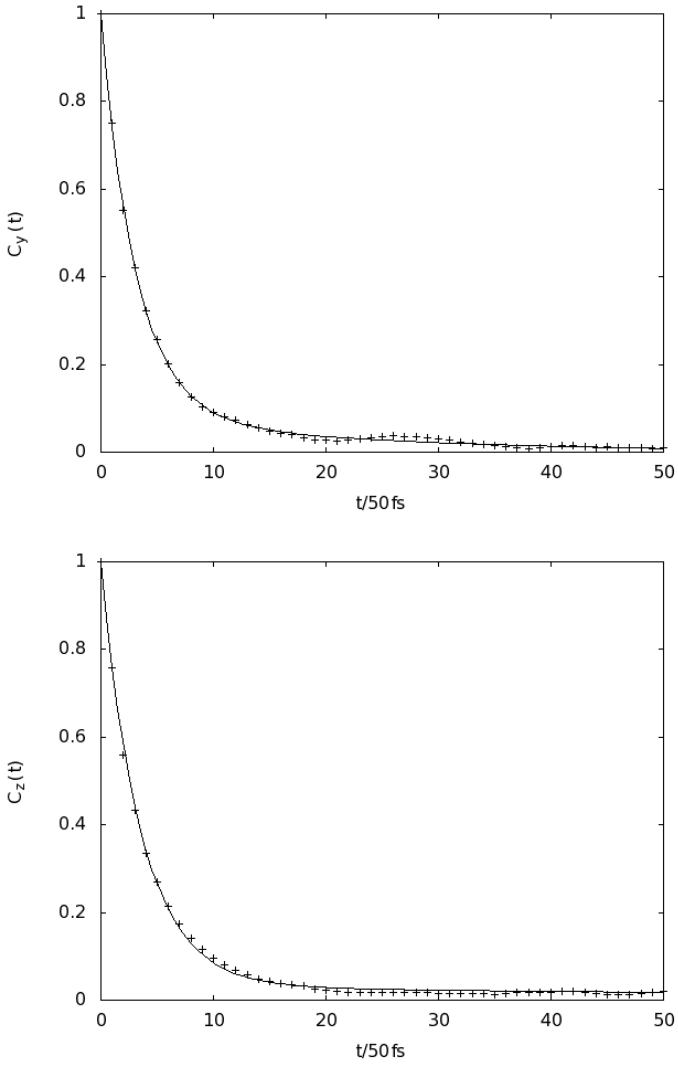
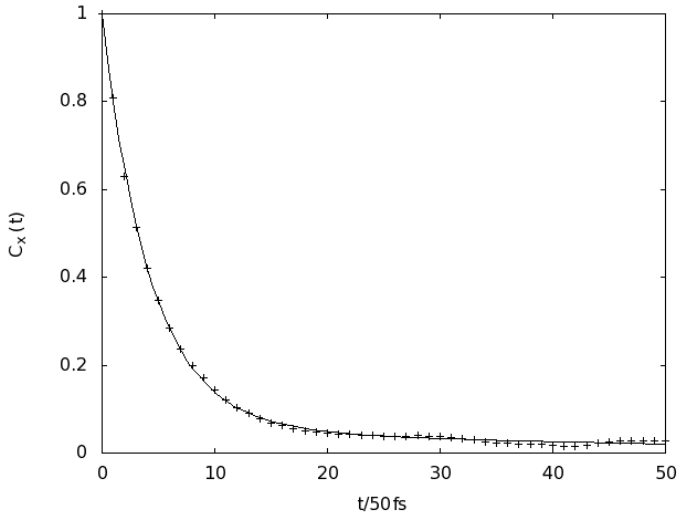


Fig.2.7.2: Funzioni di autocorrelazione e fitting bi-esponenziale, per i termini L_i ($i = x, y, z$) della GB3.



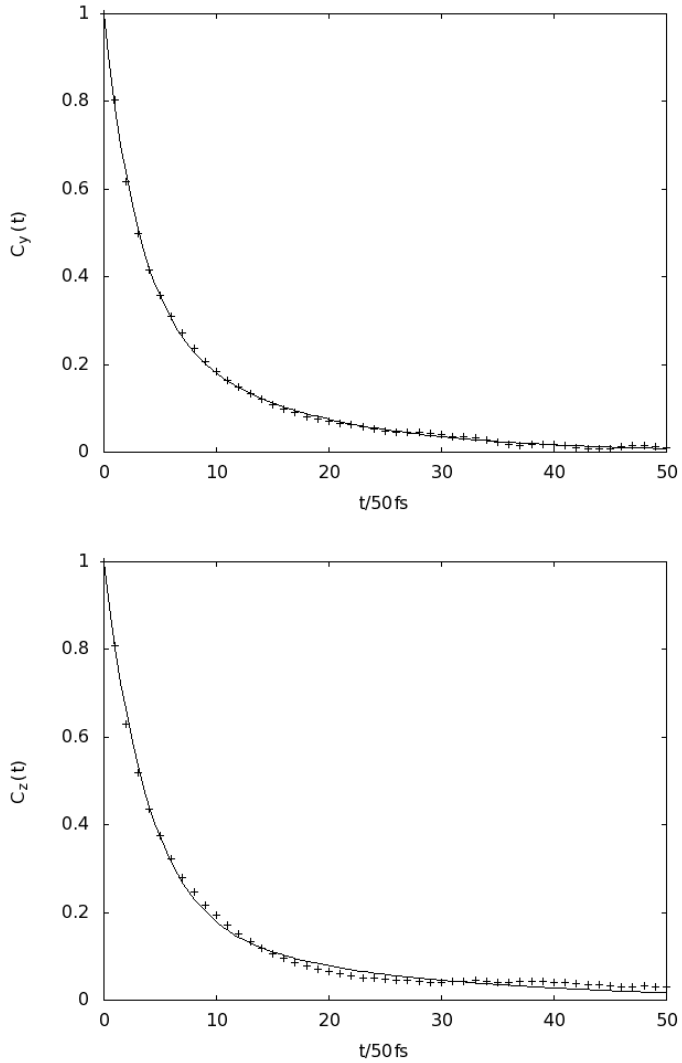


Fig.2.7.3: Funzioni di autocorrelazione e fitting bi-esponenziale, per i termini L_i ($i = x, y, z$) della LYS.

I risultati dei fitting sono riportati in **Tab.2.7.1**.

BPTI				
A^2	**τ_1/fs**	**τ_2/ps**	**$I/10^{-41}$Kg m^2**	**$\xi/10^{-41}$Kg m^{2}s^{-1}**
0.34	82.3	0.274	1.30(0.04)	1.58
0.62	106.9	0.307	1.11(0.05)	1.03
0.73	138.2	0.450	0.60(0.03)	0.432
LYS				
0.67	151.1	0.673	3.75(0.10)	2.48
0.81	191.8	1.03	4.18(0.14)	2.18
0.94	212.3	2.30	2.77(0.09)	1.30
GB3				
0.97	177.5	4.95	1.06(0.05)	0.599
0.91	155.1	1.09	0.83(0.03)	0.535
0.94	152.4	1.34	0.69(0.03)	0.456

Tab.2.71: Risultati del fitting delle funzioni di autocorrelazione del momento angolare; i numeri fra parentesi indicano la deviazione standard.

I termini diagonali D_{ii} delle tre proteine sono riportati in **Tab.2.7.2.**

Componente	BPTI	GB3	LYS
$D_{xx}/10^7$ Hz	2.6	1.7	6.9
$D_{yy}/10^7$ Hz	4.0	1.9	7.7
$D_{zz}/10^7$ Hz	9.6	3.2	9.1

Tab.2.7.2: termini diagonali D_{ii} ottenuti via fitting bi-esponenziale.

2.7.2 Protocollo idrodinamico: calcolo con DITE.

Sono stati calcolati i termini diagonali medi del tensore di diffusione rotazionale con DITE [26]:

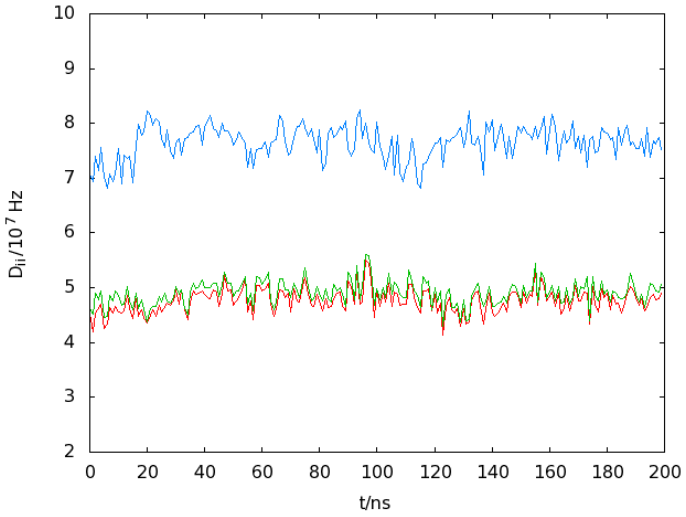
Componente	BPTI	LYS	GB3
$D_{xx}/10^7$ Hz	4.7	2.4	5.7
$D_{yy}/10^7$ Hz	4.9	2.5	6.1
$D_{zz}/10^7$ Hz	7.6	2.8	8.1

Tab.2.7.3: termini diagonali di D_{ii} calcolati con DITE.

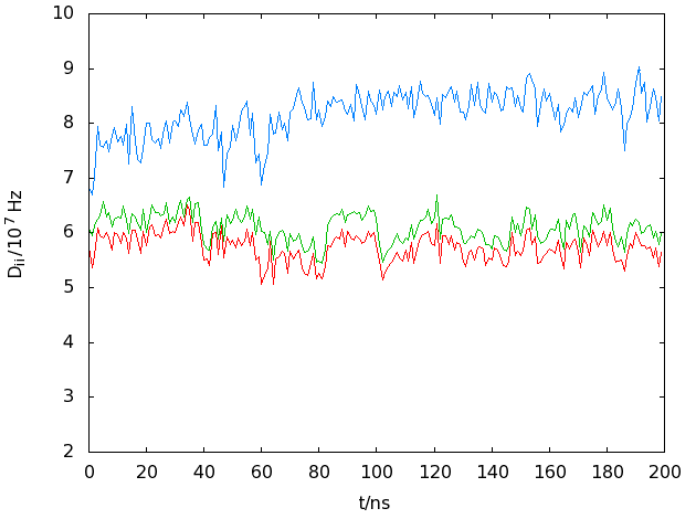
I termini D_{ii} sono stati ottenuti da conformazioni molecolari estratte dalle simulazioni lunghe (200 *ns*): in pratica, a ogni 1 *ns* è stata estratta una conformazione delle proteine, è stato calcolato il tensore di diffusione rotazionale e i valori ottenuti sono stati mediati lungo l'intera simulazione.

La serie temporale dei termini D_{ii}, per le tre proteine, è riportata in **Fig.2.7.4**.

BPTI



GB3



LYS

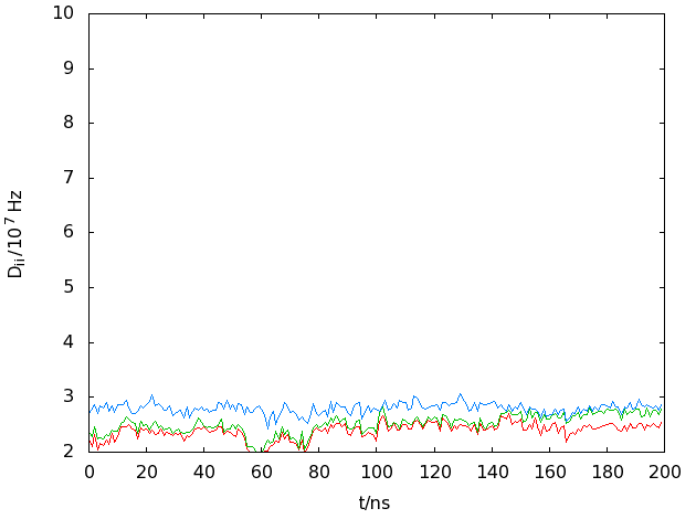


Fig.2.7.4: serie temporale dei termini D_{ii} per **BPTI**, **GB3** e **LYS**. In rosso i termini D_{xx}, in verde D_{yy} e in blu D_{zz}.

I parametri macroscopici implementati in DITE sono riportati in **Tab.2.7.4**.

Parametri DITE	**BPTI**	**LYS**	**GB3**
$\eta/10^{-4}$**Pa·s**	8.90	8.90	8.90
R_e**/Å**	2.01	1.87	2.00
***T*/K**	300	300	300
C	6	6	6

Tab.2.7.4: parametri idrodinamici; i raggi effettivi sono stati scelti come una media dei raggi di VdW di tutti gli atomi.

2.7.3 Protocollo funzioni di autocorrelazione rotazionale.

Per validare i protocolli momento angolare e DITE, sono stati confrontati i coefficienti di diffusione rotazionale D_{rot} delle proteine analizzate con quelli

ottenuti dal metodo standard, cioè dal fitting mono-esponenziale delle funzioni di autocorrelazione rotazionale; in questo caso è stata considerata l'orientazione, nel tempo, dei dipoli N-H dei legami ammidici, dei frammenti rigidi e semirigidi che costituiscono le proteine[50] [18], [17]. Il motivo è semplice: considerando solo i frammenti rigidi e semirigidi appartenenti a motivi di struttura secondaria, si può pensare che vi sia una maggiore separazione tra le scale dei tempi della dinamica interna e globale, con lo scopo di cogliere, approssimativamente, solo il rilassamento della dinamica rotazionale globale.

Quest'approssimazione è stata possibile perché i frammenti rigidi/semirigidi sono distribuiti in modo omogeneo nella catena peptidica delle proteine e vanno a costituire le strutture secondarie α-elica e β-sheet. Ovviamente, la PRO non è considerata perché non possiede il dipolo N-H. Per ciascuno di questi residui è stata calcolata la funzione di correlazione di eq. (2.6.6); poi è stata mediata su tutti i residui rigidi/semirigidi e infine normalizzata. In **Fig.2.7.5**, sono evidenziati gli amminoacidi rigidi e semirigidi; in giallo le strutture α-elica e in azzurro le strutture β-sheet. In **Tab.2.7.5**, i residui rigidi e semirigidi che costituiscono le proteine.

	α-elica	**β-sheet**
BPTI	^{3}D,^{4}F,^{5}C,^{6}L; ^{25}A,^{26}K,^{27}A; ^{48}A,^{49}E,^{50}D,^{51}C,^{52}M,^{53}R,^{54}T, ^{55}C;	^{18}I,^{19}I,^{20}R,^{21}Y,^{22}F,^{23}Y,^{24}N; ^{28}G,^{29}L,^{30}C,^{31}Q,^{32}T,^{33}F,^{34}V,^{35}Y;
GB3	^{23}A,^{24}E,^{25}T,^{26}A,^{27}E,^{28}K,^{29}A,^{30}F, ^{31}K,^{32}Q,^{33}Y,^{34}A,^{35}N,^{36}ASP,^{37}N;	^{2}Q,^{3}Y,^{4}K,^{5}L,^{6}V,^{7}I,^{8}N; ^{13}K,^{14}G,^{15}GLU,^{16}T,^{17}T,^{18}T,^{19}K,^{20}A; ^{42}V,^{43}T,^{44}T,^{45}Y,^{46}D; ^{50}K,^{51}T,^{52}F,^{53}T,^{54}V,^{55}T;
LYS	^{5}R,^{6}C,^{7}E,^{8}L,^{9}A,^{10}A,^{11}A,^{12}M,^{13}K,^{14}r,^{15}H; ^{25}L,^{26}G,^{27}N,^{28}W,^{29}V,^{30}C,^{31}A,^{32}A, ^{33}K,^{34}F,^{35}E,^{36}S; ^{80}C,^{81}S,^{82}A,^{83}L,^{84}L; ^{88}I,^{89}T,^{90}A,^{91}S,^{92}V,^{93}N,^{94}C,^{95}A,^{96}K,^{97}K,^{98}I,^{99}V,^{100}S; ^{105}M,^{106}N,^{107}A; ^{109}V,^{110}A,^{111}W,^{112}R,^{113}N,^{114}R,^{115}C; ^{120}V,^{121}Q,^{122}A,^{123}W,^{124}I;	^{43}T,^{44}N,^{45}R,^{46}N; ^{49}G,^{50}S,^{51}T,^{52}D,^{53}Y; ^{57}Q,^{58}I,^{59}N;

Tab.2.7.5: residui rigidi e semirigidi. L'apice in alto a sinistra sta a indicare il numero sequenziale del residuo lungo la catena peptidica.

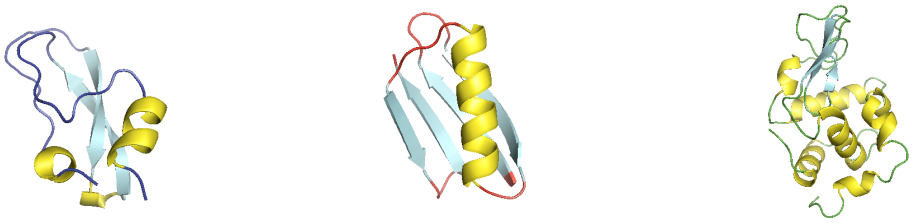


Fig.2.7.5: in evidenza i residui rigidi/semirigidi nelle proteine. BPTI (sinistra), GB3 (centro), LYS (destra). In giallo le strutture α-elica e in azzurro le strutture β-sheet.

I coefficienti D_{rot} sono stati ottenuti dal fitting mono-esponenziale, eq. (2.7.8), della funzione di autocorrelazione media (media su tutti i residui rigidi e semirigidi) normalizzata. Il protocollo di calcolo è il seguente:

1. Sono state calcolate le funzioni di autocorrelazione, non normalizzate, per ciascun residuo rigido/semirigido e mediate. La sintassi dello script utilizzato è riportata in **Appendice E1**.
2. Le funzioni di autocorrelazione medie sono state normalizzate. La sintassi dello script utilizzato è riportata in **Appendice E2**.
3. I coefficienti di diffusione sono stati ottenuti con il fitting di eq. (2.6.8). La sintassi dello script utilizzato è riportata in **Appendice E3**.

I risultati ottenuti, a confronto, sono in **Tab.2.7.6.**

	BPTI	**GB3**	**LYS**
$D_{rot}/10^7$ Hz (*misura sperimentale*	4.1	5.5	2.4
$D_{rot}/10^7$ Hz (protocollo funzioni di autocorrelazione rotazionale)	4.7	5.9	2.1
$D_{rot}/10^7$ Hz (protocollo momento angolare)	5.4	7.9	2.3
$D_{rot}/10^7$ Hz (*protocollo* DITE)	5.7	6.6	2.5

Tab.2.7.6: coefficienti D_{rot} per le tre proteine.

All'aumentare delle dimensioni delle proteine, i protocolli momento angolare e DITE riportano dei coefficienti D_{rot} comparabili con i valori sperimentali.

Le funzioni di autocorrelazione medie e normalizzate, per ciascuna proteina, sono in **Fig.2.7.6** di seguito**.**

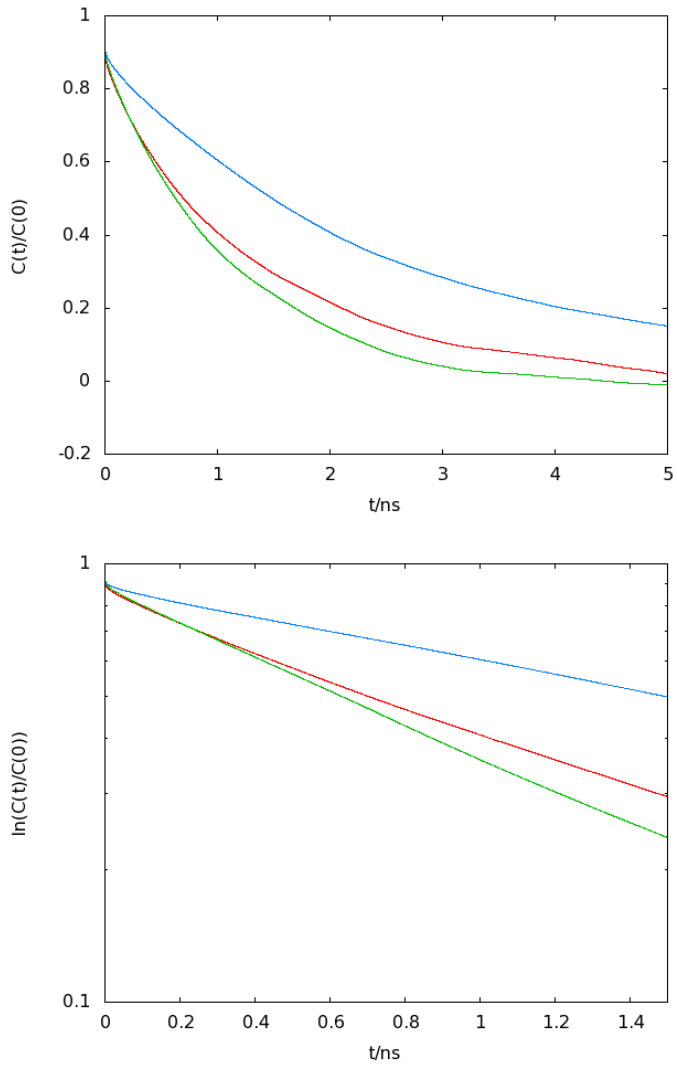


Fig.2.7.6: funzioni di autocorrelazione medie (e ln) delle proteine studiate. In rosso la BPTI, in verde la GB3 e in blu la LYS.

Dall'andamento lineare del logaritmo naturale delle funzioni di correlazione medie normalizzate non si riscontrano dinamiche interne veloci: ciò significa che l'approssimazione *Model Free* è valida per questi sistemi.

2.8 Conclusioni

L'analisi proposta delle traiettorie MD consente un calcolo veloce del tensore $\boldsymbol{D}_{rot}$ di proteine. Per quanto riguarda approcci basati sull'idrodinamica, non ci sono parametri liberi, a condizione che sia applicato un buon campo di forze. Alcune assunzioni sono necessarie per rendere il metodo applicabile.

La prima è che la proteina possiede una conformazione tale che il suo tensore non è molto influenzato da fluttuazioni della geometria molecolare. Il protocollo può essere esteso a proteine che sono rappresentate da un numero di strutture, ciascuna con

diverso $\boldsymbol{D}_{rot}$, tali che le dinamiche tra loro possono essere descritte da un modello di stati Markoviani. Se il tensore $\boldsymbol{D}_{rot}$ dipende molto dalla conformazione, il nostro protocollo (come pure quello standard [18]) non può essere applicato per il motivo che "l'approssimazione quasi-rigida" non è più valida. Ipotesi minori sono le considerazioni sui frames che diagonalizzano i tensori d'inerzia e d'attrito (collineari tra loro) e le considerazioni che, in una proteina, il tempo di correlazione più veloce è legato al rilassamento del momento angolare globale. Rispetto al protocollo standard e a quelli comparativi, i vantaggi sono:

i. Sono richieste traiettorie MD brevi.
ii. La stima del tensore è molto più vicina ai valori sperimentali.
iii. Può essere impiegato il modello di solvente esplicito TIP3P.

3 Studio MD della Calmodulina

3.1 Piano di lavoro

L'obiettivo di questo lavoro è stimare i tensori di diffusione rotazionale dei singoli domini, per la Calmodulina libera (CaM) e la Calmodulina con gli ioni Ca^{2+} (CaM-Ca^{2+}), la loro dinamica orientazionale e il potenziale medio orientante. Per questi calcoli s'è fatto uso dei software DITE e COPPS.

Questo capitolo è organizzato nel seguente modo: nella sezione 3.2 una breve descrizione della Calmodulina; nella sezione 3.3 i dettagli tecnici delle simulazioni MD; nella sezione 3.4 una breve descrizione del *modello a due corpi*; nella sezione 3.5 la presentazione dei risultati ottenuti per la CaM. In sezione 3.6, alcune considerazioni conclusive.

3.2 Introduzione

La Calmodulina (CaM) è una piccola proteina ubiquitaria negli eucarioti, costituita da 148 residui, situata nello spazio intracellulare, recettrice dello ione Ca^{2+} [57], [58]. Nello stato legato CaM-Ca^{2+} può legarsi a oltre 100 proteine bersaglio: prende parte in processi di fosforilazione/defosforilazione interagendo con gli enzimi chinasi CaM-dipendente I, II e IV, fosforilasi chinasi e calcioneurina fosfatasi, regola numerose proteine di segnali cellulari con gli enzimi ossido nitrico sintasi, PDE (*cyclic nucleotide phosphodiesterase*), interagisce con le proteine del citoscheletro, interviene nei processi di movimento e crescita cellulare e nel fissaggio della memoria a lungo termine.

CaM è formata da due sub-unità globulari molto simili, *domini*, collegate da un *linker* centrale; ogni dominio può legarsi a due ioni Ca^{2+}. In **Tab.3.2.1** la sequenza di amminoacidi per la CaM.

dominio1	ADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTI
linker	DFPEFLTMMARKMKDTDSEEEIREAFRVF
dominio2	DKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREADIDGDGQVNYEEFVQMMTAK

Tab.3.2.1: sequenza amminoacidi della CaM.

I domini presentano due siti di legame per Ca^{2+}, ognuno formato da una sequenza elica-loop-elica che prende il nome di *E-F hand*. Il *linker* centrale è dotato di una flessibilità variabile in presenza/assenza degli ioni Ca^{2+}. Gli amminoacidi responsabili di trattenere i quattro ioni Ca^{2+} sono:

- *dominio1*:^{20}D,^{22}D,^{24}D,^{26}T,^{31}E e ^{56}D,^{58}D,^{60}N,^{62}T,^{67}E;
- *dominio2*:^{93}D,^{95}D,^{97}D,^{99}Y,^{104}E;^{129}D,^{131}D,^{133}D,^{135}N,^{140}E.

Nella proteina allo stato cristallino di Ca^{2+}-legata, il linker centrale si presenta come una α-elica di otto giri; tale linker, formato da 29 residui, ha la capacità di passare da α-elica a random coil nel range di 10 *ns*. Le due subunità sono in orientazione trans tra di loro. Tale disposizione prende il nome di struttura a *dumbell* [59].

In soluzione i due lobi dell'apo-CaM e CaM-Ca^{2+} presentano una mobilità diversa: in particolare l'estremità N-terminale (*dominio1*) va incontro a variazioni conformazionali minori rispetto l'estremità C-terminale (*dominio2*); questa differenza si ripercuote sulla precisione con cui si può determinare la struttura mediante misure NMR [58].

In **Fig.3.2.1** le strutture di entrambe le proteine.

Fig.3.2.1: Proteine simulate: CaM-Ca^{2+} e CaM. In evidenza il *dominio1* (rosso), e il *dominio2* (blu). Gli ioni Ca^{2+} sono le sfere gialle.

3.3 Protocollo simulazione MD

Tutte le traiettorie MD sono state sviluppate e analizzate con il pacchetto 2.7b2 NAMD [6], VMD1.8.7 [44] e con il campo di forza CHARMM [7] par_all22_prot_cmap.

Le simulazioni sono state eseguite presso il "**L**aboratorio **I**nterdipartimentale di **C**himica **C**omputazionale" (LICC) del Dipartimento di Scienze Chimiche dell'Università degli studi di Padova, utilizzando il computer cluster AVOGADRO e il computer cluster GPU CURIE: quest'ultimo, a differenza di AVOGADRO, è un sistema costituito da cinque nodi, due socket e due GPU; ciascun socket è costituito da otto CPU AMD Opteron (Modello 6128), mentre le GPU sono NVIDIA (Modello TESLA M2050). I nodi sono interconnessi con *rete di servizio ethernet* (1Gbit) per la

gestione dei jobs e *rete InFiniband* (10 Gbit) per il calcolo parallelo MPI. Sono presenti 16 Gbyte RAM.

Per le simulazioni MD, è stato seguito lo stesso protocollo standard del capitolo 2.

In **Fig.3.3.1** le proteine dopo il processo di solvatazione.

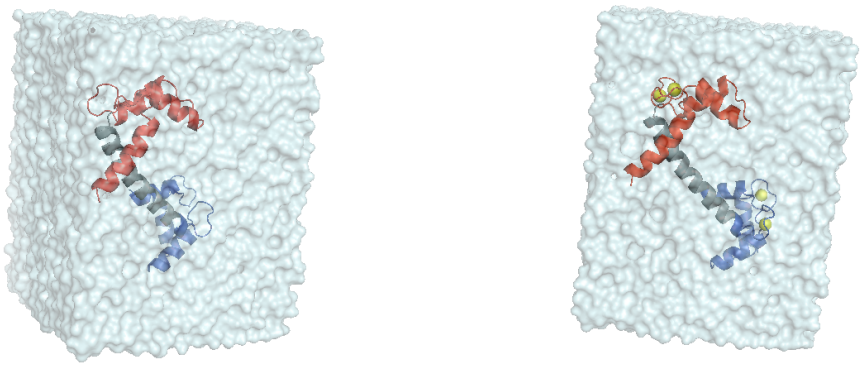


Fig.3.3.1: Proteine solvatate: CaM (sinistra) e CaM-Ca^{2+} (destra). In evidenza il *dominio1* (rosso), e il *dominio2* (blu). Gli ioni Ca^{2+} (sfere gialle).

In **Fig.3.3.2** le proteine dopo il processo di neutralizzazione della carica.

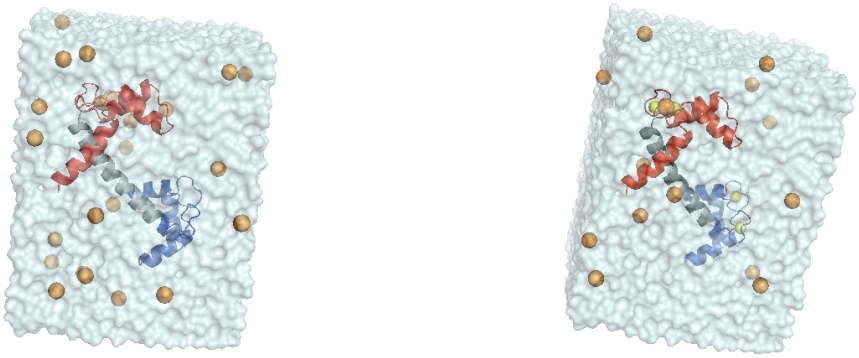


Fig.3.3.2: Proteine solvatate e neutralizzate: CaM (sinistra) e CaM-Ca^{2+} (destra). In evidenza il *dominio1* (rosso), e il *dominio2* (blu). Gli ioni Ca^{2+} (sfere gialle), gli ioni Na^{+} sono le sfere arancioni.

Le simulazioni della proteina CaM-Ca^{2+} sono state fatte con il computer cluster AVOGADRO, utilizzando 32 processori in parallelo. Le simulazioni della CaM sono state fatte con il computer cluster GPU CURIE, utilizzando 12 processori in parallelo: in questo caso, nonostante il sistema simulato contenesse almeno 3000 atomi più, la simulazione è stata più veloce.

Per evitare files troppo grandi, difficilmente gestibili, è stato modificato il codice di NAMD per far stampare solo le traiettorie delle proteine: queste sono state prodotte dopo l'ottimizzazione delle geometrie, il riscaldamento a 300 K e l'equilibrazione del sistema (1 ns); infine, le simulazioni di 400 *ns*. I controioni sono stati aggiunti al

fine di neutralizzare i sistemi. Le conformazioni molecolari sono state raccolte ogni 0.5 *ps*. I parametri delle simulazioni sono riportati in **Tab.3.1.1**.

	CaM_Ca^{2+}	**CaM**
Protein Data Bank File	1CLL	1CLL
Carica Proteina	-23	-24
Carica ioni Ca^{2+}	+8	0
Carica controioni	+15	+24
Numero molecole di acqua	7980	9025
Modello molecola di acqua	TIP3P	
Dimensione scatola periodica	61.67x78.77x58.53 Å	
Ensemble	N (26159 atomi)	N (29300 atomi)
	P (1 atm)	P (1 atm)
	T (300 K)	T (300 K)
Termostato	Temperature coupling	
Barostato	Nosé-Hoover Langevin piston (piston period 200 fs, piston decay 100 fs,piston temperature 300 K)	
Cut-Off interazioni di non legame	12 Å, smoothing switch at 10 Å	
Pair list distance	13.5 Å	
Electrostatica	PME	
Time step d'integrazione	2 fs	
Frequenza di salvataggio coordinate e velocità	250 MD steps	
Periodo di equilibrazione	1 ns	1 ns
Periodo di produzione	400 ns	400 ns
Velocità di calcolo NAMD nel cluster	0.19 giorni/ns (cluster AVOGADRO)	0.15 giorni/ns (cluster CURIE)

Tab.3.3.1: Parametri simulazione MD usati per le due proteine.

3.4 Modello a due corpi

Con il *modello a due corpi* [76] si vuole descrivere il sistema Calmodulina come formato da due corpi rigidi (i due domini) la cui diffusione nel mezzo è di tipo rotazionale e che sono accoppiati tra loro da un potenziale d'interazione. I due corpi sono visti come due oggetti che per qualche motivo sono costretti a rimanere vicini, possono riorientarsi in qualsiasi modo e il loro moto è accoppiato.

Dalla traiettoria dell'orientazione relativa dei due domini si può ricavare la superficie di potenziale di campo medio (PCM) una volta definita una sua forma funzionale. Data la natura rotazionale del problema, viene naturale espandere il PCM sulla base delle matrici di Wigner.

Per il modello a due corpi si usa una forma fenomenologica, che deriva dall'applicazione del modello a sonde molecolari in ambienti ordinati (cristalli liquidi, vetri), che prevede solo termini di rango 2 [60]:

$$\mathrm{U}\left(\Omega\right)=c_0^2 D_{0,0}^2\left(\Omega\right)+\sum_{K>0}^{2} c_{\pm K}^2\left[D_{0,K}^2\left(\Omega\right)+D_{0,-K}^2\left(\Omega\right)\right] \tag{3.4.1}$$

Qui Ω è il set di angoli di Eulero che trasformano dal sistema di riferimento che diagonalizza il tensore di diffusione rotazionale del primo dominio (M1F) a quello che diagonalizza il tensore di diffusione rotazionale del secondo dominio (M2F).

È più conveniente, in termini d'implementazione ed interpretazione, applicare una rotazione (indipendente dal tempo) a M2F in maniera tale che nella sommatoria che appare nell'eq. (3.4.1) risulti, in generale, non nullo e reale solo il coefficiente con $K=2$. A tale scopo, si costruisce il tensore d'ordine (Cartesiano) di Saupe dalla media, calcolata dalla traiettoria, delle matrici di Wigner $D_{0,K}^2\left(\Omega\right)$ come:

$$\boldsymbol{S}_{\mathrm{M1}\to\mathrm{M2}}=\begin{bmatrix} \sqrt{\frac{3}{2}}\Re\mathrm{e}\left\{\left\langle \mathrm{D}_{0,2}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\}-\frac{1}{2}\left\langle \mathrm{D}_{0,0}^2\left(\boldsymbol{\Omega}\right)\right\rangle & -\sqrt{\frac{3}{2}}\Im\mathrm{m}\left\{\left\langle \mathrm{D}_{0,2}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\} & \sqrt{\frac{3}{2}}\Re\mathrm{e}\left\{\left\langle \mathrm{D}_{0,1}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\} \\ -\sqrt{\frac{3}{2}}\Im\mathrm{m}\left\{\left\langle \mathrm{D}_{0,2}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\} & -\sqrt{\frac{3}{2}}\Re\mathrm{e}\left\{\left\langle \mathrm{D}_{0,2}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\}-\frac{1}{2}\left\langle \mathrm{D}_{0,0}^2\left(\boldsymbol{\Omega}\right)\right\rangle & -\sqrt{\frac{3}{2}}\Im\mathrm{m}\left\{\left\langle \mathrm{D}_{0,1}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\} \\ \sqrt{\frac{3}{2}}\Re\mathrm{e}\left\{\left\langle \mathrm{D}_{0,1}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\} & -\sqrt{\frac{3}{2}}\Im\mathrm{m}\left\{\left\langle \mathrm{D}_{0,1}^2\left(\boldsymbol{\Omega}\right)\right\rangle\right\} & \left\langle \mathrm{D}_{0,0}^2\left(\boldsymbol{\Omega}\right)\right\rangle \end{bmatrix} \tag{3.4.2}$$

Il tensore definito in eq. (3.4.2) esprime l'ordine di M2F rispetto a M1F. Dalla diagonalizzazione di tale tensore si ottiene

$$\boldsymbol{S}_{\mathrm{M1}\to\mathrm{M2}}=\boldsymbol{R}_{\mathrm{O}\to\mathrm{M2}}\boldsymbol{S}_{\mathrm{M1}\to\mathrm{O}}\boldsymbol{R}_{\mathrm{O}\to\mathrm{M2}}^{t} \tag{3.4.3}$$

dove $\boldsymbol{S}_{\mathrm{M1}\to\mathrm{O}}$ è il tensore (diagonale) che esprime l'ordine di un sistema di riferimento, OF, solidale al secondo dominio rispetto M1F, mentre la matrice di rotazione che porta da M2F a OF è la trasposta della matrice degli autovettori, ossia $\boldsymbol{R}_{\mathrm{M2}\to\mathrm{O}}=\boldsymbol{R}_{\mathrm{O}\to\mathrm{M2}}^{t}$, da cui si possono ottenere gli angoli di Eulero per la trasformazione da M2F a OF. Essendo $\boldsymbol{S}_{\mathrm{M1}\to\mathrm{O}}$ diagonale (e a traccia nulla) solo le due componenti sferiche

$$\begin{cases} S_0^2=S_{ZZ} \\ S_2^2=\sqrt{\frac{2}{3}}\left(S_{XX}-S_{YY}\right) \end{cases} \tag{3.4.4}$$

che implica che sono non nulli e reali soli i coefficienti del potenziale c_0^2 e c_2^2. Questi ultimi sono calcolati dai valori dei parametri d'ordine ,risolvendo il seguente sistema di equazioni non lineari nei due coefficienti:

$$\begin{cases} S_0^2=\int d\Omega_{M1\to O} D_{0,0}^2\left(\Omega_{M1\to O}\right) e^{c_0^2 D_{0,0}^2(\Omega_{M1\to O})+c_2^2\left[D_{0,2}^2(\Omega_{M1\to O})+D_{0,-2}^2(\Omega_{M1\to O})\right]} \\ S_2^2=\int d\Omega_{M1\to O}\left[D_{0,2}^2\left(\Omega_{M1\to O}\right)+D_{0,-2}^2\left(\Omega_{M1\to O}\right)\right] e^{c_0^2 D_{0,0}^2(\Omega_{M1\to O})+c_2^2\left[D_{0,2}^2(\Omega_{M1\to O})+D_{0,-2}^2(\Omega_{M1\to O})\right]} \end{cases} \tag{3.4.5}$$

che viene risolto numericamente con il metodo di Powell [61]. Dall'analisi delle linee di livello delle superfici $S_0^2\left(c_0^2,c_2^2\right)$ e $S_0^2\left(c_2^2,c_2^2\right)$ (**Fig.3.4.1**) si può dedurre che la soluzione del sistema in eq. (3.4.5) è unica.

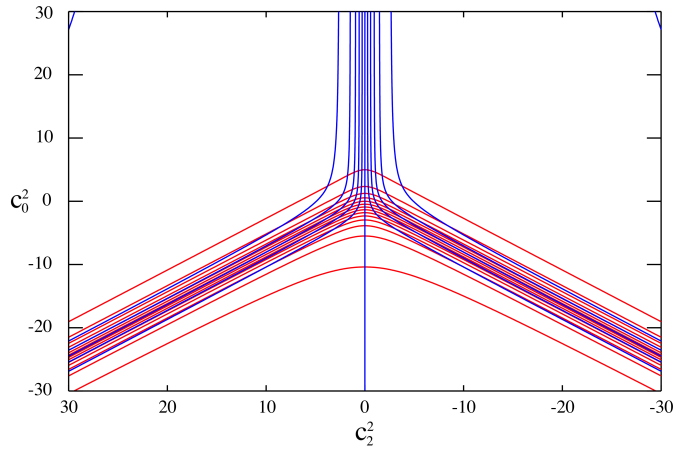


Fig.3.4.1. Isolinee delle superfici di S_0^2 (linee rosse) e S_2^2 (linee blu) in funzione dei coefficienti del potenziale.

3.5 Risultati ottenuti

Il protocollo di lavoro per i due domini della CaM è il seguente:

1. Dalle traiettorie MD delle proteine, ogni 0.5 *ps* sono state estratte le coordinate dei domini e salvate in files diversi. La sintassi dello script utilizzato è in **Appendice F1**.
2. È stato calcolato, per ciascun dominio, il tensore $\boldsymbol{D_{rot}}$ e la matrice di rotazione che definisce l'orientazione dei domini rispetto il sistema di riferimento di laboratorio con il programma DITE [26].
3. È stata calcolata $\left(E\left(\Omega_{1\to 2}\right)\right)$ la matrice di rotazione di Eulero che definisce l'orientazione relativa dei domini lungo tutta la traiettoria MD e le matrici di Wigner $D_{0,0}^2\left(E\left(\Omega_{1\to 2}\right)\right)$ e $\left[D_{0,2}^2\left(E\left(\Omega_{1\to 2}\right)\right)+D_{0,-2}^2\left(E\left(\Omega_{1\to 2}\right)\right)\right]$.

   La sintassi dello script è riportata in **Appendice F2**.
4. Sono state calcolate, con le matrici di rotazione $E\left(\Omega_{1\to 2}\right)$, le funzioni di autocorrelazione delle matrici di Wigner di rango 2 per gli angoli interni e i parametri d'ordine S_0^2, S_2^2.

5. Sono stati calcolati, mediante COPPS, i valori dei coefficienti c_0^2, c_2^2 del potenziale orientante, le funzioni di autocorrelazione degli angoli interni e poi messe a confronto con quelle calcolate nel punto 4.

II termini diagonali D_{ii} sono stati calcolati da conformazioni molecolari estratte dalla simulazione: in pratica, ogni 0.5 *ps*, sono stati calcolati con DITE, dalle strutture dei domini per entrambe le proteine e i valori ottenuti sono stati mediati lungo l'intera simulazione di 345*ns*. I valori sono riportati in **Tab.3.5.1**.

dominio1	$\boldsymbol{D_{xx}/10^7}$**Hz**	4.5±0.2
	$\boldsymbol{D_{yy}/10^7}$**Hz**	4.8±0.2
	$\boldsymbol{D_{zz}/10^7}$**Hz**	5.8±0.3
	$\boldsymbol{D_{iso}/10^7}$**Hz**	5.1±0.2
dominio2	$\boldsymbol{D_{xx}/10^7}$**Hz**	5.3±0.2
	$\boldsymbol{D_{yy}/10^7}$**Hz**	5.9±0.2
	$\boldsymbol{D_{zz}/10^7}$**Hz**	6.5±0.2
	$\boldsymbol{D_{iso}/10^7}$**Hz**	5.9±0.2

Tab.3.5.1: valori medi dei termini D_{ii} e valori isotropi.

L'andamento temporale dei termini D_{ii} per i due domini è riportato in **Fig.3.5.1**; inoltre è riportata la distribuzione gaussiana dei valori numerici assunti dai termini lungo l'intera simulazione.

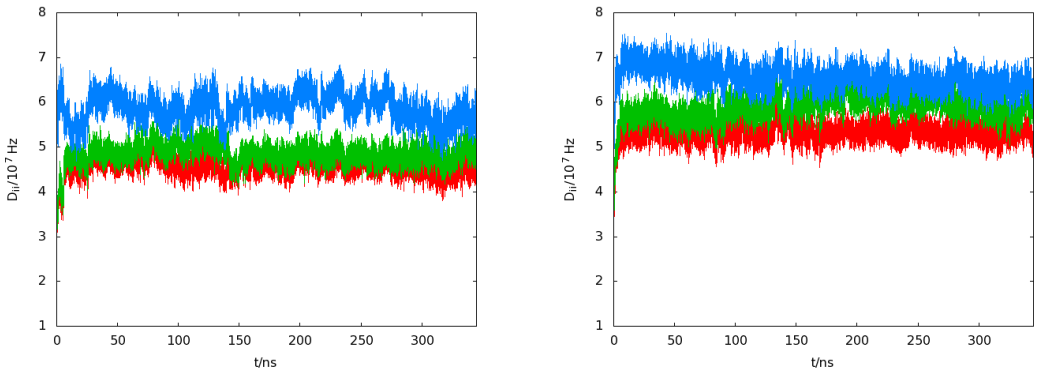


Fig.3.51a: Serie temporale dei termini D_{ii} del *dominio1* (a sinistra) e *dominio2* (a destra) per CaM. In rosso i termini D_{xx}, in verde D_{yy} e in blu D_{zz}.

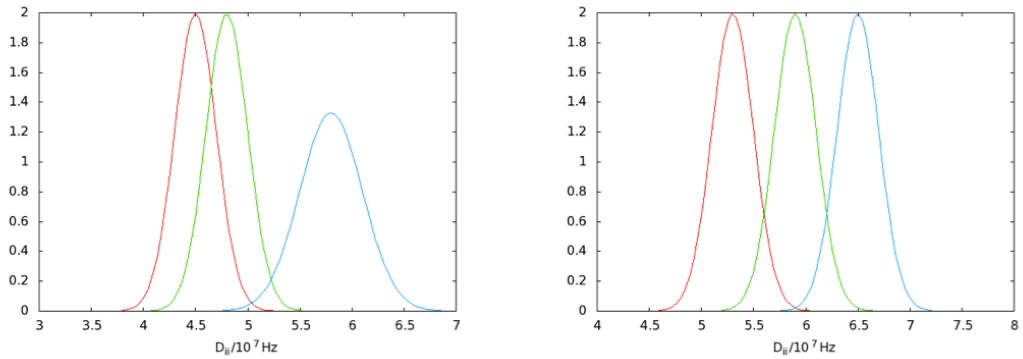


Fig.3.5.1b: Distribuzioni gaussiane dei termini D_{ii} del *dominio1* (a sinistra) e *dominio2* (a destra) per CaM. In rosso i termini D_{xx}, in verde D_{yy} e in blu D_{zz}.

I risultati del potenziale sono riportati qui in seguito.

Molecola	**Parametri d'ordine**	**Coefficienti del PCM**	$\boldsymbol{\Omega}_{\mathrm{M2}\rightarrow\mathrm{O}}$ **/ deg**
CaM(345ns)	$\begin{cases} S_0^2 = -0.11 \\ S_2^2 = -0.46 \end{cases}$	$\begin{cases} c_0^2 = -0.26 \\ c_2^2 = -1.17 \end{cases}$	$\begin{cases} \alpha = 30.2 \\ \beta = 96.4 \\ \gamma = 92.4 \end{cases}$

Tabella 3.5.2. Parametri d'ordine, coefficienti del potenziale e angoli di tilt tra M2F e OF ottenuti dall'analisi delle traiettorie della calmodulina libera. I coefficienti del potenziale sono in unità k_BT con $T = 300$ K.

Dalla forma del potenziale di campo medio si vede gli angoli β e γ del set di angoli di Eulero che danno l'orientazione relativa tra i due corpi sono confinati in un minimo a $\pm k\frac{\pi}{2}, \left(k = 0, \pm 1, \pm 2, ...\right)$ con barriere dell'ordine di 1 – 2 k_BT. La superficie di potenziale è riportata in **Fig.3.5.2**.

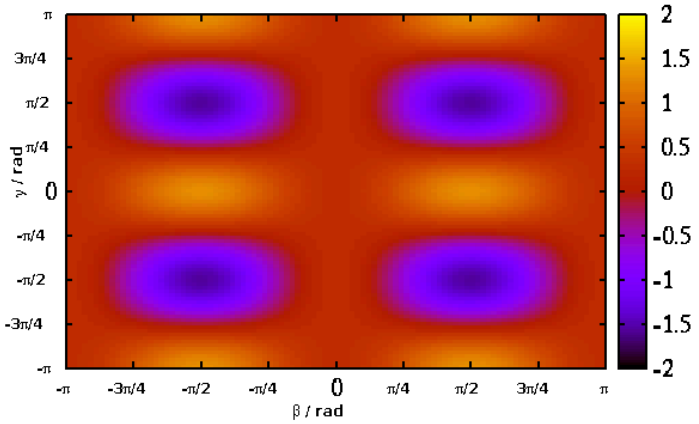


Fig.3.5.2. Superficie PCM per la CaM. Il potenziale è espresso in unità k_BT, con $T = 300$ K.

La funzione di autocorrelazione normalizzata degli angoli interni $\boldsymbol{\Omega}_{1\to 2}$, derivante dalla traiettoria MD, calcolata con le matrici di Wigner di rango 2, è riportata in **Fig.3.5.3** e confrontata con quella calcolata con COPPS [75].

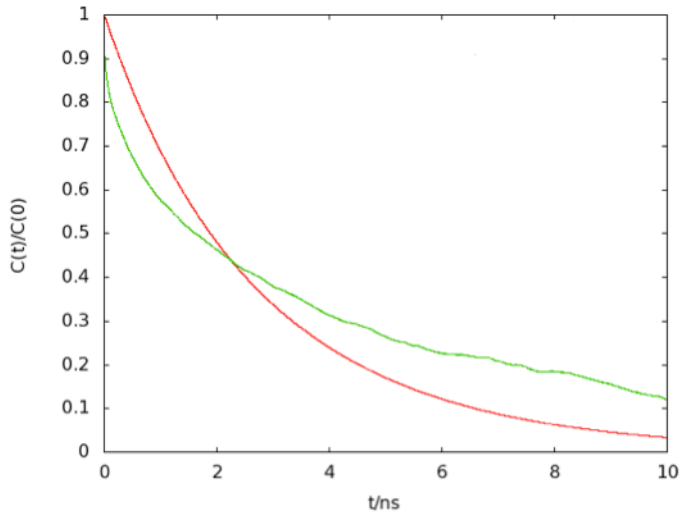


Fig.3.5.3: funzioni di autocorrelazione delle matrici di Wigner di rango 2 degli angoli interni della CaM, calcolate via MD (in verde) e via COPPS (in rosso).

La funzione di correlazione calcolata via MD è più rumorosa perché la statistica non è eccellente (il tempo di correlazione è troppo piccolo); sarebbe stato necessario calcolare la funzione con una traiettoria più lunga. Inoltre, risente della dinamica interna dei domini.

Con COPPS sono stati calcolati i tempi di correlazione delle funzioni di correlazione delle matrici di Wigner $D^2_{0,0}\left(\Omega_{L\to i}\right)$, con $\Omega_{L\to i}$ il set di angoli di Eulero che trasformano dal laboratorio al sistema di riferimento solidale ad uno dei due domini (i = 1, 2). Sono stati messi a confronto con valori sperimentali [62] ottenuti da esperimenti NOE-NMR. I risultati sono riportati in **Tab. 3.5.2**.

τ_{rot} / ns		CaM
τ_{rot} / ns **SPERIMENTALE**	*Dominio1*	2.8 ± 0.2
	Dominio2	2.2 ± 0.2
τ_{rot} / ns **COPPS (moti disaccoppiati)**	*Dominio1*	3.3
	Dominio2	2.7
τ_{rot} / ns **COPPS (moti accoppiati)**	*Dominio1*	3.5
	Dominio2	2.9

Tab. 3.5.2: confronto tempi di correlazione rotazionali dei domini.

3.6 Conclusioni

In questo lavoro è stato utilizzato un metodo stocastico per interpretare una simulazione MD *full-atom* e ottenere informazioni a tempi lunghi (come i tempi di correlazione di osservabili fisiche) che permettono di accedere a osservabili sperimentali (ad es. spettroscopiche). Il modello stocastico descrive un sistema a dimensionalità ridotta rispetto al sistema molecolare: essenzialmente si associa al sistema in esame, la Calmodulina, un sistema modello formato da due corpi rigidi (i due domini) la cui diffusione nel mezzo è di tipo rotazionale, accoppiati tra loro da un potenziale d'interazione.

I dati ottenuti, tempi di correlazione rotazionali, sono stati confrontati con valori sperimentali a loro volta calcolati indirettamente da misure NMR a 296 K. Questi assumono l'indipendenza statistica del moto dei due corpi (*Model Free*).

L'effetto della temperatura è stato trascurato, in 4 K di differenza, tra esperimento (296 K) e simulazione MD (300 K) assumendo che gli effetti contrapposti sul tempo di correlazione della temperatura si elidano: abbassando la temperatura il potenziale aumenta (τ_{rot} diminuisce) e la viscosità aumenta (τ_{rot} aumenta).

τ_{rot} sperimentale è più alto rispetto a quello calcolato con COPPS perché la dinamica dei domini è disaccoppiata. Il tempo di correlazione τ_{rot} calcolato con COPPS in assenza di potenziale di accoppiamento si avvicina a quello sperimentale.

Infine, s'è cercato di ripetere questo lavoro per la Calmodulina con gli ioni Ca^{2+} legati ma non ci si è riusciti perché, dopo 25-30 *ns* di simulazione, questi ioni si sono staccati.

In **Fig.3.6.1** la proteina e gli ioni Ca^{2+} liberi.

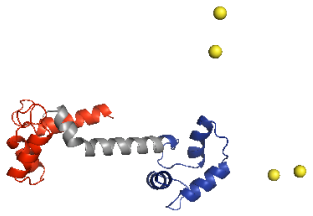


Fig.3.6.1: proteina e ioni Ca^{2+} liberi.

L'effetto si vede anche dal confronto delle serie temporali degli angoli β: nell'intervallo 30-60 *ns*, gli angoli tendono a assumere gli stessi valori perché la conformazione dei domini non è più determinata dalla presenza degli ioni Ca2+; in pratica, si sta osservando ancora una volta la dinamica della CaM. In **Fig.3.6.2**, le serie temporali degli angoli β.

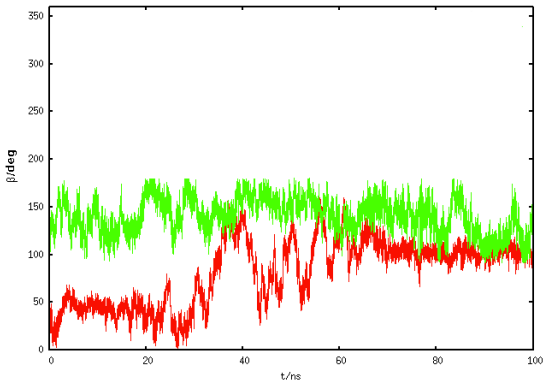


Fig.3.6.2: le serie temporali degli angoli β per le due proteine; in verde (CaM) e in rosso (CaM-Ca^{2+})

Possibili cause a questo problema sono: campo di forze non efficace a descrivere l'interazione elettrostatica proteina-Ca^{2+} e/o le condizioni ambientali non ideali (ad es. pH, forza ionica che agiscono sulla protonazione dei residui)

Appendice A.

A1 - Matrici di rotazione e angoli di Eulero.

L'orientazione relativa tra due sistemi di riferimento destrogiri $\{x,y,z\}$ e $\{X,Y,Z\}$ è descritta con una matrice di rotazione $\boldsymbol{R}(\boldsymbol{\Omega})$, parametrizzata dagli angoli di Eulero $\boldsymbol{\Omega} = (\alpha,\beta,\gamma)$.

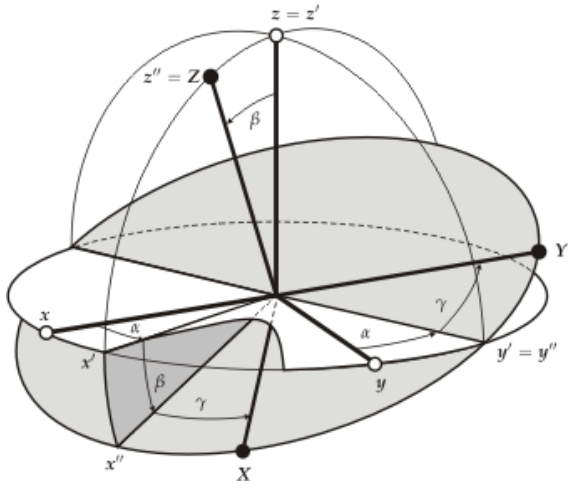


Fig.A1.1: schema orientazione relativa dei sistemi di riferimento $\{x,y,z\}$ e $\{X,Y,Z\}$.
(*Fonte: www.easyspin.org/documentation/eulerangles.html).*

La trasformazione che porta un sistema sull'altro, $\{x,y,z\} \xrightarrow{\boldsymbol{\Omega}} \{X,Y,Z\}$, può essere scomposta in tre rotazioni successive:

$$\{x,y,z\} \xrightarrow{\alpha(z)} \{x',y',z'\} \xrightarrow{\beta(y')} \{x'',y'',z''\} \xrightarrow{\gamma(z'')} \{X,Y,Z\} \qquad \text{(A1.1)}$$

Questa è la convenzione zyz.

Qui $\alpha(z)$ indica la rotazione attorno all'asse z di un angolo $0 \le \alpha \le 2\pi$, $\beta(y')$ indica la rotazione attorno all'asse y' di un angolo $0 \le \beta \le \pi$, $\gamma(z'')$ indica la rotazione attorno all'asse z'' di un angolo $0 \le \gamma \le 2\pi$.

La matrice di Eulero è il prodotto delle tre matrici di rotazione:

$$\boldsymbol{R}(\boldsymbol{\Omega}) = \boldsymbol{R}_{z''}(\gamma)\boldsymbol{R}_{y'}(\beta)\boldsymbol{R}_{z}(\alpha) =$$
$$= \begin{pmatrix} c\gamma & s\gamma & 0 \\ -s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c\beta & 0 & -s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{pmatrix} \begin{pmatrix} c\alpha & s\alpha & 0 \\ -s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} = \tag{A1.2}$$
$$= \begin{pmatrix} c\alpha c\beta c\gamma - s\alpha s\gamma & s\alpha & -s\beta c\gamma \\ -c\alpha c\beta s\gamma - s\alpha s\gamma & c\alpha & s\beta s\gamma \\ c\alpha s\beta & s\alpha s\beta & c\beta \end{pmatrix}$$

Qui *c* e *s* sono *seno* e *coseno*. [64] Le matrici di rotazione sono reali, simmetriche e ortogonali. Le righe e le colonne hanno un semplice significato geometrico:

- Le righe di $\boldsymbol{R}$ sono la rappresentazione di $\{X,Y,Z\}$ nel sistema di riferimento $\{x,y,z\}$. Ad es., la prima riga di $\boldsymbol{R}$ è la rappresentazione vettoriale di X nel sistema di riferimento $\{x,y,z\}$.
- Le colonne di $\boldsymbol{R}$ contengono la rappresentazione di $\{x,y,z\}$ nel sistema di riferimento $\{X,Y,Z\}$. Ad es., la terza colonna di $\boldsymbol{R}$ fornisce il vettore z rappresentato dai suoi tre *coseni* direttori nel sistema di riferimento $\{X,Y,Z\}$.

E' possibile ottenere la stessa matrice $\boldsymbol{R}$, se le rotazioni sono effettuate in ordine inverso rispetto altri assi: primo di un angolo γ attorno l'asse z, poi di β attorno l'asse *originale* y e infine di α ancora attorno all'*originale* z.

$$\boldsymbol{R}(\Omega) = \boldsymbol{R}_{z}(\alpha)\boldsymbol{R}_{y}(\beta)\boldsymbol{R}_{z}(\gamma) \Leftrightarrow \boldsymbol{R}_{z''}(\gamma)\boldsymbol{R}_{y'}(\beta)\boldsymbol{R}_{z}(\alpha) \tag{A1.3}$$

Per ottenere gli angoli di Eulero della trasformazione $\boldsymbol{R}_2 : \{X,Y,Z\} \xrightarrow{\boldsymbol{\Omega}} \{x,y,z\}$, conoscendo gli angoli della rotazione descritti sopra, è valida la seguente relazione:

$$(\alpha_2, \beta_2, \gamma_2) = (-\gamma, -\beta, -\alpha) \tag{A1.4}$$

A2 - Matrici di Wigner.

L'orientazione relativa di due sistemi di riferimento può essere indicata dagli angoli di Eulero $\boldsymbol{\Omega} = (\alpha, \beta, \gamma)$. La proprietà principale di tali angoli è che la rotazione complessiva dal sistema di riferimento iniziale (**laboratorio**, SL) a quello finale (**molecolare**, SM), può essere scritta come il prodotto di tre rotazioni consecutive successive (definite in SL). [72-74] La rotazione è un esempio di trasformazione finita, convenientemente espressa dalla relativa trasformazione infinitesima. La generica rotazione $\hat{R}_{\hat{n}}(\alpha)$, di un sistema, è identificata da un versore $\hat{n}$ e l'angolo α; se si considera la rotazione come passiva, cioè rimane fermo il sistema e cambia il sistema di riferimento (SDR), è possibile dimostrare che $\hat{R}_n(\alpha)$ è:

$$\hat{R}_n(\alpha) = exp(-i\alpha \hat{\boldsymbol{J}} \hat{n}) = exp\left(-i\alpha \frac{\partial}{\partial \alpha}\right) \tag{A2.1}$$

Qui $\hat{\boldsymbol{J}}$ è l'operatore momento angolare del sistema, definito in SDR fisso iniziale.
Si può dimostrare che la matrice (A1.2) può essere espressa in questo modo:

$$\hat{\boldsymbol{R}}(\boldsymbol{\Omega}) = exp(-i\alpha \hat{J}_z) exp(-i\beta \hat{J}_y) exp(-i\gamma \hat{J}_z) \tag{A2.2}$$

NB: qui la matrice è considerata come un operatore.

Si consideri il set di autofunzioni $|J, M\rangle$ degli operatori $\hat{\boldsymbol{J}}^2$ e $\hat{J}_z$, con $J \geq 0$ e M=*-J,-J+1,...,0,..,J-1,J* (numero quantico di proiezione).

La rotazione da SL a SM determina la trasformazione di $\hat{J}_z$ in $\hat{J}'_z$, mentre $\hat{\boldsymbol{J}}^2$ è invariante sotto rotazione (commuta).

Le autofunzioni $|J, M'\rangle$ di $\hat{\boldsymbol{J}}^2$ e $\hat{J}_z$, ovvero le $|J, M\rangle$ trasformate sotto rotazione, sono le seguenti:

$$|J, M'\rangle = \hat{\boldsymbol{R}}(\boldsymbol{\Omega})|J, M\rangle = \sum_{M'} D^J_{M',M}(\boldsymbol{\Omega})|J, M\rangle \tag{A2.3}$$

Qui $D^J_{M',M}(\boldsymbol{\Omega})$ sono le funzioni di Wigner di rango J.

L'insieme delle $(2J+1)^2$ funzioni con lo stesso rango (numero massimo di righe/colonne linearmente indipendenti), ordinato rispetto gli indici M',M (riga/colonna), costituisce la MATRICE DI WIGNER (MW) di rango J : $\boldsymbol{D}^J(\boldsymbol{\Omega})$.

MW è la rappresentazione matriciale dell'operatore $\hat{\boldsymbol{R}}(\boldsymbol{\Omega})$ sulla base di $\hat{\boldsymbol{J}}^2$ e $\hat{J}_z$ (riferiti a SL) ed è unitaria. Poiché le funzioni $|J,M\rangle$ sono ortonormali, è valida la seguente relazione:

$$D^J_{M',M}(\boldsymbol{\Omega}) = \langle J,M' | \hat{\boldsymbol{R}}(\boldsymbol{\Omega}) | J,M \rangle = exp(-i\alpha M') d^J_{M',M}(\beta) exp(-i\gamma M) \qquad \text{(A2.4)}$$

Qui $d^J_{M',M}(\beta) = \langle J,M' | exp(-i\beta \hat{J}_y) | J,M \rangle$ è la funzione di Wigner ridotta e assume valori reali. In seguito sono elencate alcune proprietà delle funzioni di Wigner.

- *Proprietà di simmetria*.

$$\begin{aligned}
&d^J_{M',M}(\boldsymbol{\beta}) = d^J_{M,M'}(-\boldsymbol{\beta}) \\
&d^J_{M',M}(\boldsymbol{\beta}) = (-1)^{M'-M} d^J_{M,M'}(\boldsymbol{\beta}) \\
&d^J_{M',M}(\boldsymbol{\beta}) = d^J_{-M,-M'}(\boldsymbol{\beta}) \qquad \text{(A2.5)} \\
&d^J_{M',M}(\boldsymbol{\beta}) = (-1)^{M'-M} d^J_{-M',M}(\boldsymbol{\beta}) \\
&D^J_{M',M}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})^* = D^J_{M,M'}(-\boldsymbol{\gamma},-\boldsymbol{\beta},-\boldsymbol{\alpha}) = (-1)^{M'-M} D^J_{-M',-M}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})
\end{aligned}$$

- *Base ortonormale nello spazio degli angoli di Eulero*.

  Dall'integrazione nello spazio degli angoli di Eulero definito:

$$\int d\boldsymbol{\Omega}(...) \equiv \int_0^{2\pi} d\alpha \int_0^{\pi} d\beta \,\mathrm{sen}\beta \int_0^{2\pi} d\gamma (...) \qquad \text{(A2.6)}$$

  Si ottiene il prodotto interno:

$$\int d\boldsymbol{\Omega} D^{J_1}_{M_1,K_1}(\boldsymbol{\Omega})^* D^{J_2}_{M_2,K_2}(\boldsymbol{\Omega}) \equiv \frac{8\pi^2}{2J_1+1} \delta_{J_1,J_2} \delta_{M_1,M_2} \delta_{K_1,K_2} \qquad \text{(A2.7)}$$

  Dalla precedente si rileva facilmente che l'insieme di funzioni:

$$|J,M,M'\rangle \equiv \sqrt{\frac{2J+1}{8\pi^2}} D^J_{M,M'}(\Omega) \qquad \text{(A2.8)}$$

  Costituisce una base ortonormale completa rispetto al prodotto scalare nello spazio degli angoli di Eulero definito in eq. (A2.6).

- *Operatori momento angolare.*

  Una matrice di Wigner, dipendente da un set di angoli $\boldsymbol{\Omega} = (\alpha, \beta, \gamma)$ che mettono in relazione due sistemi di riferimento SL e SM è autofunzione degli operatori $\hat{\boldsymbol{J}}^2$ e $\hat{J}_z$, definiti nei due sistemi di riferimento.

  Per il sistema di riferimento SL:

$$\begin{cases} {}^{\mathrm{SL}}\hat{\boldsymbol{J}}^2 D^J_{M,M'}(\Omega) = J(J+1) D^J_{M,M'}(\Omega) \\ {}^{\mathrm{SL}}\hat{J}_z D^J_{M,M'}(\Omega) = -M D^J_{M,M'}(\Omega) \\ {}^{\mathrm{SL}}\hat{J}_{\pm} D^J_{M,M'}(\Omega) = -c^{\mp}_{J,M} D^J_{M\mp 1,M'}(\Omega) \end{cases} \tag{A2.9}$$

  Qui $c^{\mp}_{J,M} = \sqrt{J(J+1) - M(M \mp 1)}$.

  Per il sistema di riferimento SM:

$$\begin{cases} {}^{\mathrm{SM}}\hat{\boldsymbol{J}}^2 D^J_{M,M'}(\Omega) = J(J+1) D^J_{M,M'}(\Omega) \\ {}^{\mathrm{SM}}\hat{J}_z D^J_{M,M'}(\Omega) = -M' D^J_{M,M'}(\Omega) \\ {}^{\mathrm{SM}}\hat{J}_{\pm} D^J_{M,M'}(\Omega) = -c^{\pm}_{J,M'} D^J_{M,M'\pm 1}(\Omega) \end{cases} \tag{A2.10}$$

  Qui $c^{\pm}_{J,K} = \sqrt{J(J+1) - M'(M' \pm 1)}$.

Appendice B - Equazione di Smoluchowski.

B1 – Moto browniano.

[65], [66] Sia data una particella sferica, di massa *m*, immersa in un fluido: su di essa agisce una forza d'attrito, da parte del fluido, e la più semplice espressione per questa forza è data dalla legge di Stokes. Considerando il caso monodimensionale:

$$F_c(t) = -\alpha v(t) \tag{B.1.1}$$

Qui α è il coefficiente d'attrito. L'equazione del moto della particella è:

$$\begin{cases} \dot{v}(t) + \gamma v(t) = 0 \\ \gamma = \dfrac{\alpha}{m} = \dfrac{1}{\tau} \end{cases} \tag{B.1.2}$$

Qui $\dot{v}$ è la derivata temporale della velocità, γ è il coefficiente d'attrito per unità di massa, τ è il tempo di rilassamento. Integrando l'equazione, si ricava che:

$$v(t) = v(0)e^{-\gamma t} = v(0)e^{-\frac{t}{\tau}} \tag{B.1.3}$$

L'attrito deriva dalla collisione del fluido con la particella: questa cede la propria "quantità di moto" alle particelle di fluido, perciò la velocità della particella decade a zero. L'eq. (B.1.2) è *un'equazione deterministica*: $v(t)$ è completamente determinata dalle condizioni iniziali del sistema.

L'eq. (B.1.2) è valida se *m* è grande abbastanza che la velocità derivante dalle fluttuazioni termiche è trascurabile. Ricordando il principio di equipartizione dell'energia:

$$\frac{3}{2}m\langle v^2 \rangle = \frac{3}{2}k_B T \tag{B.1.4}$$

Se $m << 1$, la *velocità termica* definita:

$$v_{th} = \sqrt{\langle v^2 \rangle} = \sqrt{\frac{k_B T}{m}} \tag{B.1.5}$$

deve essere osservabile, perciò la velocità di particelle molto piccole non può essere descritta dalle equazioni (B.1.2-3): devono essere modificate cosi da ottenere la corretta energia termica. La modifica consiste nell'aggiungere una *forza fluttuante*

(stocastica), in modo che la forza totale agente sulla particella può essere decomposta in un continuo smorzante F_c e in un termine stocastico F_f :

$$F_{tot}(t) = F_c(t) + F_f(t) = -\alpha v(t) + F_f(t) \qquad \text{(B.1.6)}$$

$F_f(t)$ è necessario perché non è possibile risolvere analiticamente le equazioni accoppiate del moto di tutte le molecole del sistema considerato (particella+fluido). Per risolvere questa situazione, si usa la Meccanica Statistica: si considera un insieme (*ensemble*) di tali sistemi (*Gibbs ensemble*); la forza varia da sistema a sistema e ciò che andrà fatto è considerare le medie di queste forze per l'ensemble.
Conseguentemente a questo, l'equazione del moto della particella diventa:

$$\dot{v}(t) + \gamma v(t) = \frac{F_f(t)}{m} = \Gamma(t) \qquad \text{(B.1.7)}$$

$\Gamma(t)$ è la *Forza di Langevin,* mentre l'eq. (B.1.7) è *un'equazione stocastica differenziale*. La proprietà più importante di $\Gamma(t)$ è il "Teorema di Fluttuazione Dissipazione":

$$\left\langle \Gamma(t)\Gamma(t') \right\rangle = \frac{2\gamma k_B T}{m}\delta(t\text{-}t') \qquad \text{(B.1.8)}$$

Questo teorema quantifica la relazione tra le fluttuazioni del sistema all'equilibrio termico e la sua risposta a perturbazioni subite. Si assume che $\boldsymbol{\Gamma}(t)$ è una distribuzione gaussiana con correlazione δ.
Integrando l'eq. (B.1.7) e cosiderando il Teorema di Fluttuazione Dissipazione, è possibile calcolare il coefficiente di diffusione della particella.
Poiché $\Gamma(t)$ è una quantità stocastica, anche $v(t)$ sarà stocastica; pertanto si parlerà di: "probabilità di trovare la velocità della particella in un intervallo infinitesimo $(v, v+dv)$, oppure, il numero di stati dell'ensemble le cui velocità sono nell'intervallo $(v, v+dv)$ diviso il numero di sistemi dell'ensemble.

Siccome $v(t)$ è una variabile continua, bisogna parlare di funzione densità di probabilità $P(v,t)$ (distribuzione di probabilità): questa funzione dipende dal tempo t e dalla distribuzione iniziale. L'equazione del moto per la distribuzione $P(v,t)$ è descritta dall'equazione di Fokker Planck:

$$\frac{\partial P(v,t)}{\partial t} = \gamma \frac{\partial \left(vP(v,t) \right)}{\partial v} + \gamma \frac{k_B T}{m} \frac{\partial^2 P(v,t)}{\partial v^2} \tag{B.1.9}$$

Risolvendo quest'equazione, date le condizioni iniziali e al contorno, è possibile calcolare $P(v,t)$ per tutti i tempi successivi. Conoscendo $P(v,t)$, qualunque valore medio di $v(t)$ può essere calcolato attraverso il seguente integrale:

$$\left\langle \mathrm{h}\left(v(t) \right) \right\rangle = \int_{-\infty}^{+\infty} \mathrm{h}(v) P(v,t)\, dv \tag{B.1.10}$$

Qui $\mathrm{h}(v)$ è una funzione arbitraria di *v*.

B2 – Equazione di Kramers e Smoluchowski.

[67] Le equazioni di Kramers-Klein e Smoluchowski sono forme speciali di equazione Fokker-Planck.

L'equazione di Kramers è riferita alla funzione di distribuzione nello spazio delle fasi (posizione, velocità), $P(x,v,t)$, che descrive il moto browniano di particelle in presenza di un potenziale esterno:

$$\frac{\partial P(x,v,t)}{\partial t} = \left[-\frac{\partial}{\partial x} v + \frac{\partial}{\partial v} \left(\gamma v + \frac{1}{m} \frac{\partial F(x)}{\partial x} \right) + \gamma \frac{k_B T}{m} \frac{\partial^2}{\partial v^2} \right] P(x,v,t) \tag{B.2.1}$$

Qui $F(x)$ è la forza che deriva dal potenziale esterno. In condizioni stazionarie, la soluzione dell'equazione di Kramers è la distribuzione di Boltzmann:

$$\begin{cases} P_{st}(x,v) = N e^{-E/k_B T} \\ E = \frac{1}{2} m v^2 + m f(x) \end{cases} \tag{B.2.2}$$

Qui *N* è la costante di normalizzazione.

L'equazione stocastica differenziale, riferita all'eq. (B.2.1), che descrive la dinamica della particella nello spazio delle fasi, è:

$$\begin{cases} \dot{x}(t) = v(t) \\ \dot{v}(t) + \gamma v(t) = \dfrac{F(x)}{m} + \Gamma(t) \\ \left\langle \Gamma(t)\Gamma(t') \right\rangle = \dfrac{2\gamma k_B T}{m}\delta(t-t') \end{cases} \tag{B.2.3}$$

In condizioni di grande attrito, l'equazione precedente diventa:

$$\gamma v(t) = \frac{F(x)}{m} + \Gamma(t) \tag{B.2.4}$$

La corrispondente equazione di Fokker-Planck sarà applicata solamente alla distribuzione densità di probabilità delle coordinate: $P(x,t)$.

$$\frac{\partial P(x,t)}{\partial t} = \frac{1}{m\gamma}\left[-\frac{\partial}{\partial x}F(x) + k_B T \frac{\partial^2}{\partial x^2}\right]P(x,t) \tag{B.2.5}$$

In assenza di forze esterne, la precedente diventa:

$$\frac{\partial P(x,t)}{\partial t} = \frac{k_B T}{m\gamma}\frac{\partial^2}{\partial x^2}P(x,t) \tag{B.2.6}$$

Questa è l'equazione di Smoluchowski. Qui $D=\frac{k_B T}{m\gamma}$ è il coefficiente di diffusione della particella nel fluido isotropo.

Appendice C – Input Files simulazioni.

Gli scripts riportati in seguito sono riferiti alla proteina BPTI; considerazioni analoghe per la GB3, LYS, CaM e CaM-Ca^{2+}.

C1 - Generazione del file *psf*.

Da bash linux eseguo:

```
$ grep -v HETATM 6PTI.pdb > bpti.pdb
```

Creo lo script makeProtein con il seguente codice:

```
topology ../common/par/top_all22_prot_cmap.inp
alias residue HIS HSE
segment GB3 {pdb ../common/pdb/1P7E.pdb}
coordpdb ../common/pdb/1P7E.pdb GB3
writepsf ../common/psf/gb3.psf
guesscoord
writepdb ../common/pdb/gb3.pdb
```

Da bash linux, eseguo lo script con il comando "psfgen":

```
$ ./psfgen < makeProtein
```

C2 - Addizione del solvente.

Per creare la scatola di solvente e inserire la proteina, si esegue la seguente procedura:

Si usa il comando "Add Solvation Box" del software VMD per costruire la scatola di solvente: si creano i files *solvate.pdb* e *solvate.psf*.

Con il seguente codice, da TKconsole, si pone la proteina (*bpti.pdb*) nel centro del sistema di riferimento (in generale, è il sistema di riferimento di laboratorio LF):

```
set prot [atomselect top all]
set cm [measure center $prot weight mass]
set ac [$prot get {x y z}]
set nc {}
foreach r $ac {
lappend nc [vecsub $r $cm]
}
$prot set {x y z} $nc
```

Per inserire la proteina (*bpti.pdb*) nella scatola di solvente (*solvate.pdb*) si crea lo script *makeSystem* con il seguente codice:

```
topology top_all22_prot_cmap.inp
```

```
segment PTI {
pdb bptiCM.pdb
}
coordpdb bptiCM.pdb PTI
segment WAT {
auto none
pdb solvate.pdb
}
coordpdb solvate.pdb WAT
writepsf bptiCM_wat.psf
guesscoord
writepdb bptiCM_wat.pdb
```

Da bash linux, si esegue lo script con il comando “psfgen”:

```
$ ./psfgen < makeSystem
```

C3 - Addizione dei controioni.

Per aggiungere i controioni alla proteina nel solvente, si usa il comando “AddIons” di VMD: in particolare, per neutralizzare la carica, si usa l’opzione “only neutralize system with NaCl”.

C4 - Dinamica Molecolare.

```
set mdN 0
####################
# Useful variables #
####################
set restartFlag  1
set minimizeFlag 0
set heatFlag     0
set eqFlag       0
set mdFlag       1
####################
set nameRoot         bpti_wat_ion
set energyOut        _ene
set heatingOut       _heat
set equilibrationOut  _eq
set oldMdOut         _md_01
####################
set mdOut           _md_01
set energyPath ../energy
set heatPath   ../heat
set eqPath     ../equilibrate
set mdPath     ../md
set mdPathLong ../md_long
```

```
###############
# Input files #
###############
structure   ../common/psf/$nameRoot.psf
coordinates ../common/pdb/$nameRoot.pdb
####################
if {$restartFlag} {
        if {!$mdFlag} {
                set restartpath $heatPath
                set restartname $nameRoot$heatingOut
                binCoordinates $restartpath/$restartname.coor
                binVelocities  $restartpath/$restartname.vel
                extendedSystem $restartpath/$restartname.xsc
        }
}
##############
# Parameters #
##############
paraTypeCharmm on
parameters     ../common/par/par_all22_prot_cmap.inp
if {!$minimizeFlag} {
set restart     250
restartfreq     $restart
dcdfreq         $restart
xstFreq         $restart
outputEnergies  $restart
outputPressure  $restart
 if {$mdFlag} {
   subsystemdcd  888  ;# output protein only
 }
}
##################
# Output control #
##################
if {$minimizeFlag} {
set outputname $nameRoot$energyOut
set outputpath $energyPath
outputName $outputpath/$outputname
}
if {$heatFlag} {
set outputname $nameRoot$heatingOut
set outputpath $heatPath
outputName $outputpath/$outputname
}
if {$seqFlag} {
set outputname $nameRoot$equilibrationOut
set outputpath $eqPath
outputName $outputpath/$outputname
}
if {$mdFlag} {
set outputname $nameRoot$mdOut
set outputpath $mdPath
```

```
}
#####################
# System parameters #
#####################
# Integrator Parameters
timestep           2.0  ;# 2fs/step
rigidBonds         all  ;# needed for 2fs steps
nonbondedFreq      1
fullElectFrequency  1
stepspercycle      10
zeroMomentum       yes
# Force field
exclude            scaled1-4
1-4scaling         1.0
cutoff             12.0
switching          on
switchdist         10.0        ;# should be less than cutoff
pairlistdist       13.5        ;# should be greater than cutoff
# System temperature
set T        300.0
if {!$heatFlag} {
# Constant Temperature Control
tCouple      on
tCoupleTemp   $T
}
# Periodic Boundary Conditions
cellBasisVector1  44.0    0.0    0.0
cellBasisVector2   0.0   44.0    0.0
cellBasisVector3   0.0    0.0   44.0
cellOrigin         0.0    0.0    0.0
wrapAll            on
# PME (for full-system periodic electrostatics)
PME                yes
PMEInterpOrder     6
PMETolerance       0.00001
PMEGridSizeX       45
PMEGridSizeY       45
PMEGridSizeZ       45
# Constant Pressure Control (variable volume)
useGroupPressure      yes ;# needed for rigidBonds
useFlexibleCell       no
useConstantArea       no
langevinPiston        on
langevinPistonTarget  1.01325 ;#  in bar -> 1 atm
langevinPistonPeriod  200.
langevinPistonDecay   100.
langevinPistonTemp    $T
###########
# Run job #
###########
# Minimization
if {$minimizeFlag} {
```

```
temperature   $T
set minimize_steps  15000
minimize $minimize_steps
}
# Heating
if {$heatFlag} {
for {set i 1} {$i <= $T} {incr i} {
langevinTemp    $i
reinitvels      $i
run 300
}
reinitvels $T
}
# Equilibration
if {$seqFlag} {
run 500000 ;# 1000 ps
}

# Molecular Dynamics
if {$mdFlag} {
        if {!$mdN} {
                set restartpath $mdPath
                set restartname bpti_wat_ion_md_03
        }
        if {$mdN} {
                set restartpath ../md_long
                set str _md_$mdN
                set restartname $nameRoot$str
        }
#####################
        binCoordinates $restartpath/$restartname.restart.coor
        binVelocities  $restartpath/$restartname.restart.vel
        extendedSystem $restartpath/$restartname.restart.xsc
        set j [expr $mdN+1]
        set str _md_$j
        set outputname $nameRoot$str
        set outputpath $mdPathLong
        outputName  $outputpath/$outputname
#####################
        set timechunk 2500000
        set fts [expr $mdN*$timechunk]
        firsttimestep $fts
        run $timechunk ;# 5 ns
}
```

Appendice D - Momento angolare

Questi script sono stati scritti in codice CHARMM [7] per calcolare il momento angolare globale delle proteine BPTI, Gb3, LYS e le relative funzioni di autocorrelazione del momento angolare.

D1 – Calcolo del momento angolare della proteina.

```
* This script:
* 1) calculates and prints in log file the inertia tensor of bpti
* 2) calculates the global angular momenutm time series expressed in the LABORATORY FRAME and print in the j_cm.dat file of bpti
*
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Some useful definitions !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
set INPATH  ../
set COMPATH ../
set OUTPATH ../
set TRJNAME bpti_wb_md_01
set PSFFILE bpti_charmm.psf
set NFRAMES 60000
set OFFSET  0
set PROTEIN U
set NATOMS  888
set NCHUNKS 200
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Read topology and parameter !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
read  rtf card name @COMPATH/top_all27_prot_lipid.inp
read para card name @COMPATH/par_all27_prot_lipid.inp
!!!!!!!!!!!!
! Read PSF !
!!!!!!!!!!!!
read psf card name @COMPATH/@PSFFILE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Define the whole protein selection !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
define PROT sele segi @PROTEIN end
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Calculate inertia tensor !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open read unit 31 file name @INPATH/@TRJNAME.dcd
traj query unit 31
set  BEGIN ?START
set  SKIP  ?SKIP
calc ST    @SKIP*@NFRAMES + @OFFSET
```

```
traj iread 31 nread 1 begin @BEGIN skip @SKIP stop @ST
set N 1
label INERTIA_LOOP
      traj read
      ! Print inertia tensor
      coor iner
      incr N by 1
if N .le. @NFRAMES goto INERTIA_LOOP
close unit 31
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Generate MASS WEIGHTED trj of coordinates on CM frame !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open  read unit 31 file name @INPATH/@TRJNAME.dcd
open write unit 33 file name @INPATH/@TRJNAME_mass.dcd
traj iread 31 nread 1 skip @SKIP begin @BEGIN stop @ST iwrite 33 nwrite 1 skip @SKIP begin @BEGIN
set I 1
label R_MASS_WEIGHT_LOOP
            traj read
            COOR ORIE NORO
            COOR MASS_weighting
            traj write
            incr I by 1
if I .le. @NFRAMES goto R_MASS_WEIGHT_LOOP
close unit 31
close unit 33
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Remove CM velocity from veldcd !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open write unit 31 file name @INPATH/@TRJNAME.veldcd
open write unit 33 file name @INPATH/@TRJNAME_mass.veldcd
traj iread 31 nread 1 skip @SKIP begin @BEGIN stop @ST iwrite 33 nwrite 1 skip @SKIP begin @BEGIN
set I 1
label V_CENTER_MASS_LOOP
            traj read
            COOR ORIE NORO
            traj write
            incr I by 1
if I .le. @NFRAMES goto V_CENTER_MASS_LOOP
close unit 31
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Open the mass weighted coor and the vel trajectories depurated from CM !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open read unit 31 file name @INPATH/@TRJNAME_mass.dcd
traj query unit 31
set BEGINR ?START
set SKIPR  ?SKIP
open read unit 32 file name @INPATH/@TRJNAME_mass.veldcd
traj query unit 32
set BEGINV ?START
set SKIPV  ?SKIP
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Divide the calculation in chunks of atoms !
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
calc CHUNKSIZE INT( @NATOMS / (@NCHUNKS - 1) )
set  IC 1
label CHUNKS_LOOP
        calc AI ( (@IC - 1) * @CHUNKSIZE + 1 )
        calc AF ( @IC * @CHUNKSIZE )
        if AF .gt. @NATOMS then set AF @NATOMS
        calc N ( (@AF - @AI) + 1 )
        calc MAXSER ( 3 + 6 * @N )  ! J, R(1,N), V(1,N)
        set  MAXATS 4000
        set  MAXTMS @NFRAMES
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        ! Load partial J time series for IC > 1 !
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        if IC .gt. 1 then open unit 51 read name @OUTPATH/j_cm.dat form
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        ! Start the correl environment !
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        correl maxseries @MAXSER maxatoms @MAXATS maxtimesteps @MAXTMS
                ! Angular momentum time series
                enter J  zero
                enter JY zero
                enter JZ zero
                edit J veccod 3
                ! Mass weighted coordinates
                set I 1
                label AT_R_LOOP
                        calc NAT ( @AI + @I - 1 )
                        enter R@I atom xyz sele PROT .subset. @NAT end
                        incr I by 1
                if I .le. @N goto AT_R_LOOP
                trajectry firstu 31 nunit 1 begin @BEGINR skip @SKIPR
                ! Atoms Velocities
                set I 1
                label AT_V_LOOP
                        calc NAT ( @AI + @I - 1 )
                        enter V@I atom xyz sele PROT .subset. @NAT end
                        incr I by 1
                if I .le. @N goto AT_V_LOOP
                trajectory firstu 32 nunit 1 begin @BEGINV skip @SKIPV
                ! Update angular momentum
                if IC .gt. 1 then
                        read J unit 51 dumb
                        close unit 51
                endif
                set I 1
                label J_LOOP
                        ! J = J + M(I) * [R(I)-RCM] x [V(I)-VCM]
                        mantime R@I cros V@I
                        mantime J add R@I
                        incr I by 1
                if I .le. @N goto J_LOOP
```

```
                ! Output partial angular momentum info
                open unit 52 write name @OUTPATH/j_cm.dat form
                write J unit 52 dumb
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        ! Quit the correl environment !
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        end
        incr IC by 1
if IC .le. @NCHUNKS goto CHUNKS_LOOP
close unit 31
close unit 32
stop
```

D2 – Calcolo funzioni di autocorrelazione

```
* This script calculated the angular momentum autocorrelation function
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Some useful definitions !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
set INPATH   ../
set OUTPATH  ../
set FILENAME j_cm.dat
set NFRAMES  60000
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Load angular velocity and momentum time series!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open unit 51 read name @OUTPATH/@FILENAME form
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Start the correl environment !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
correl maxseries 6 maxtimesteps @NFRAMES
        ! Angular momentum time series
        enter LX zero
        enter LY zero
        enter LZ zero
        edit LX veccod 3
        read LX unit 51 dumb
    edit LX veccod 1
        edit LY veccod 1
        edit LZ veccod 1

        !Correlation functions
        enter CFX zero
        enter CFY zero
        enter CFZ zero
        ! Calculate autocorrelation functions
        ! [1] < LX(t) LX(0) >
        corfun LX LX PROD FFT NLTC ! NONORM
        mantime CFX copy CORR
```

```
            ! [2] < LY(t) LY(0) >
            corfun LY LY PROD FFT NLTC ! NONORM
            mantime CFY copy CORR
            ! [3] < LZ(t) LZ(0) >
            corfun LZ LZ PROD FFT NLTC ! NONORM
            mantime CFZ copy CORR
            ! Output
            edit CFX veccod 3
!           open unit 52 write name @OUTPATH/jacf_cm_mf.dat form
            open unit 52 write name @OUTPATH/jacf_test.dat form
            write CFX unit 52 dumb
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Quit the correl environment !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
end
stop
```

Appendice E - funzioni rotazionali.

Questi script sono stati scritti in codice CHARMM [7] e in linguaggio C++ e permettono di calcolare le funzioni di autocorrelazione del rilassamento dei dipoli N-H dei legami peptidici dei residui rigidi e semirigidi delle proteine BPTI, GB3 e LYS.

E1 – Calcolo funzione di autocorrelazione.

```
* This script outputs in the log file the backbone N-H bond vector
*
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Some useful definitions !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
set INPATH  ./
set OUTPATH ./
set PROTEINNAME PTI
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Read topology and parameter !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
read  rtf card name  top_all22_prot_cmap.inp
read para card name par_all22_prot_cmap.inp
!!!!!!!!!!!!
! Read PSF !
!!!!!!!!!!!!
read psf card name bpti_charmm.psf
!!!!!!!!!!!!
! Read PDB !
!!!!!!!!!!!!
read coor pdb resid name bpti.pdb
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Define the whole protein selection !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
define PROT sele segi @PROTEINNAME end
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Open file unit of trajectory input !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open read unit 20 file name @INPATH/trj_charmm_bpti_wat_ion_md.dcd
!Number of residues of protein analized
set nres 57
set res 1
set resi 0
label p2enter
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!Invoke CORREL mode!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!!!!!P2 correlation function of NH vectors!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !The MAXTimesteps value is the largest number of steps any          !
```

```
            !time series will contain. The MAXSeries keyword is the largest number !
            !of timeseries that will be contained at any time within CORREL.          !
            !A vector time series will counts as 3 time series in allocating space. !
            !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
            correl maxtime 400000 maxseries 6
                        define nvec sele resi @res .and. type N end
                        if ?nsel ne 1 goto skp2
                        define hvec sele resi @res .and. type HN end
                        if ?nsel ne 1 goto skp2
                    enter nv atom xyz sele nvec end
                    enter hv atom xyz sele hvec end
                        !Specify the trajectory to process
                        traj firstu 20 nunit 1 begin 250 stop 100000000 skip 250
                        !Q(t) = real * Q(t)
            mantime hv mult -1.0
                    !Q(t) = Q(t) + Q2(t)
            mantime nv add hv
                    ! Q(t) = Q(t) / |Q(t)|
            mantime nv normal
                    !This is to obtain the correlation function of second order Legendre
                    !Polynomial, (3 <[Qa(0).Qb(t)]**2> - 1)/2.Qa and Qb will be unit vectors
            corfun nv nv fft p2 nonorm
                    !Write the time series of every residue to a file
                    open write unit 21 card name @OUTPATH/acf_NH_P2_@res.dat
                    write corr unit 21 dumb
                    close unit 21
                    label skp2
        end
        incr res
if @res le @nres goto p2enter
!close unit 20
!END script
stop
```

E2 – Calcolo media funzioni di autocorrelazione.

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

int main (int argc, char* argv[])
{
 if (argc < 2)
            {
            cout << "Inserisci npunti, cfwignerDJMK.dat" << endl;
              return 1;
        }

 int npunti, i;
 sscanf(argv[1],"%d",&npunti);
 fstream file;
 double * cfwignerDJMK = new double [npunti];
```

```
 file.open(argv[2],ios::in);
 for ( i = 0; i < npunti; i++)
        {
          file >> cfwignerDJMK[i];
//        cout << cfwignerDJMK[i] <<endl;
        }
 file.close();

 double comodo = cfwignerDJMK[0];

 for ( i= 0; i < npunti; i++)
        {
          cfwignerDJMK[i] = cfwignerDJMK[i]/comodo;
          cout << cfwignerDJMK[i] <<endl;
        }

 delete [] cfwignerDJMK;
 return (0);

}
```

E4 – Calcolo tempo di autocorrelazione.

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

int main (int argc, char* argv[])
{
 if (argc < 2)
        {
        cout << "Inserisci npunti, cfwignerDJMK.dat" << endl;
          return 1;
      }

 int npunti, i;
 sscanf(argv[1],"%d",&npunti);
 fstream file;
 double * cfwignerDJMK = new double [npunti];

 file.open(argv[2],ios::in);
 for ( i = 0; i < npunti; i++)
        {
          file >> cfwignerDJMK[i];
//        cout << cfwignerDJMK[i] <<endl;
        }
 file.close();
 double s = 0.0 , x = 0.0;
 double dx = 1.0;
for ( i= 0; i < npunti; i++)
        {
          if (cfwignerDJMK[i] >= 0.0)
                {
                        s = s + dx * cfwignerDJMK[i];
                        x = x + dx * 500.0;
                }
          else
                break;/*
                {
                        cfwignerDJMK[i] = 0.0;
                        i = npunti;
                }*/
        }

 cout << "L'integrale della funzione di autocorrelazione C(t) è :" << s << endl;
 cout << "L'intervallo d'integrazione è:" << x << "fs" << endl;
 cout << "Lintervallo d'integrazione è costituito da " << x/500.0 << " punti" << endl;

 delete [] cfwignerDJMK;
```

```
 return (0);
}
```

Appendice F - Calcolo MD Calmodulina.

Questi script sono scritti in linguaggio C++.

F1 – Estrazione delle coordinate dei domini.

Questo script consente di estrarre le coordinate dei domini della Calmodulina per poi calcolare i termini diagonali del tensore di diffusione rotazionale con DITE [26].

```
//--///GOOD VERSION!!!04-01-12
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

std::string int_to_string(int J);

using namespace std;

int main (int argc, char* argv[])
{
        if (argc < 4)
        {
          cout << "**Import nfile and nframe and frame_nframe.pdb of your protein.**" << endl;
          return 1;
        }
//--///DECLARATION OF VARIABLES
        int i, j, k, l, nframe, nfile;
        std::string s;
//--///natoms of protein and single domains is like number of records of frame.pdb
        int natoms_calmodulin = 2205;
//--///domain1: 890 atoms + END tag
        int natoms_domain_1 = 891;
//--///domain2: 825 atoms + END tag
        int natoms_domain_2 = 826;

//--///Allocation of dynamic arrays in memory
        string * calmodulin, * domain_1, * domain_2;
        calmodulin  = new string [natoms_calmodulin];
        domain_1  = new string [natoms_domain_1];
        domain_2  = new string [natoms_domain_2];

//--///Import nfile
        sscanf(argv[1],"%d",&nfile);
//--//      cout << argv[1] << endl;

//--///Import nframe of pdb file
        sscanf(argv[2],"%d",&nframe);
//--//      cout << argv[2] << endl;

//--///Open frame.pdb and import data
        fstream file;
        file.open(argv[3],ios::in);

//--///Index of calmodulin
        i = 0;
//--///Index of domain_1
        j = 0;
//--///Index of domain_2
        k = 0;

        while ( i < natoms_calmodulin)
          {
                getline (file, calmodulin[i]);
                if (i < 890)//0-889(890)
                        {
```

```
                                        domain_1[j] = calmodulin[i];
                                        j++;
                                }
                        if (i >= 1375 and i < 2200)//1375,2199(824)
                                {
                                        domain_2[k] = calmodulin[i];
                                        k++;
                                }
                        //ions Ca2+ on domain1
        //              if (i >= 2200 and i < 2202)//2200,2201
        //                      {
        //                              domain_1[j] = calmodulin[i];
    //                      j++;
        //                      }
                        //ions Ca2+ on domain2
        //              if (i >= 2202 and i < 2204)//2202,2203
        //                      {
        //                              domain_2[k] = calmodulin[i];
    //                      k++;
        //                      }
                        //END tag
                        if (i == 2204)
                                {
                                        domain_1[j] = calmodulin[i];
                                        domain_2[k] = calmodulin[i];
                                }
                        i++;
                }

        file.close();

//--//Create new frame.pdb with the coordinates of calmodulin in the Laboratory Frame (LF);
/*      s = "log_" + int_to_string(nframe) + ".pdb";
        file.open(s.c_str(),ios::out);
        i = 0;
        while (i < natoms_calmodulin)
                {
                        file << calmodulin[i] << endl;
                        i++;
                }
        file.close();*/

//--//Create domain1.pdb with the coordinates in Laboratory Frame (LF);
//      s = "domain_1_" + int_to_string(nframe) + ".pdb";
        s = "../../frames_domains/domain1/" + int_to_string(nfile) + "ns/domain_1_" + int_to_string(nframe) + ".pdb";
        file.open(s.c_str(),ios::out);
        j = 0;
        while (j < natoms_domain_1)
                {
                        file << domain_1[j] << endl;
                        j++;
                }
        file.close();

//--//Create domain2.pdb with the coordinates in Laboratory Frame (LF);
        s = "../../frames_domains/domain2/" + int_to_string(nfile) + "ns/domain_2_" + int_to_string(nframe) + ".pdb";
        file.open(s.c_str(),ios::out);
        k = 0;
        while (k < natoms_domain_2)
                {
                        file << domain_2[k] << endl;
                        k++;
                }
        file.close();

 //--//Deallocation of dynamic arrays in memory
        delete [] calmodulin;
        delete [] domain_1;
        delete [] domain_2;


        return (0);
}//--//end main

std::string int_to_string(int J)
{
      std::stringstream ss;
```

```
    ss << J;
    return (ss.str());
}
```

F2 – Calcolo della matrice di rotazione.

Questo script consente di calcolare gli elementi della matrice di rotazione che descrive l'orientazione dei domini rispetto il sistema di riferimento di laboratorio (SL).

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <math.h>
#include <string>
#include <algorithm>

#define PI 3.14159265

//--Prototypes of functions

using namespace std;

int main (int argc, char* argv[])
{
 if (argc < 3)
        {
        cout << "Inserisci nline_file, domain_dif_mat" << endl;
          return 1;
    }

//--///Declaration variables: dynamic allocation
 int nline_file, ncolumn_file = 15;
 sscanf(argv[1],"%d",&nline_file);
 int i, j, k, l, m;
 double *alpha = new double [nline_file];
 double *beta = new double [nline_file];
 double *gamma = new double [nline_file];
 double **Data_File, **e1, **e2, **e3;
// double *test = new double [nline_file];
 double test1, test2, test3;
 Data_File = new double *[nline_file];
 e1 = new double *[nline_file];
 e2 = new double *[nline_file];
 e3 = new double *[nline_file];
 for (i=0; i < nline_file; i++)
        {
          Data_File[i] = new double [ncolumn_file];
          e1[i] = new double [ncolumn_file];
          e2[i] = new double [ncolumn_file];
          e3[i] = new double [ncolumn_file];
        }

//--///Import data from file
 fstream file;
 file.open(argv[2],ios::in);
 for (i = 0; i < nline_file; i++)
        {
        k = 0;
        l = 0;
        m = 0;
        for (j = 0; j < ncolumn_file; j++)
          {
                file >> Data_File[i][j];
                if (j > 2 and j < 6)
                        {
                                e1[i][k] = Data_File[i][j];
                                k++;
                        }
```

```
                        else if (j > 5 and j < 9)
                                {
                                        e2[i][l] = Data_File[i][j];
                                        l++;
                                }
                        else if (j > 8 and j < 12)
                                {
                                        e3[i][m] = Data_File[i][j];
                                        m++;
                                }
                }
        }
 file.close();
//--DOMAINS--//
 for (i = 0; i < nline_file; i++)
        {
        if ((i+1) < nline_file)
        {
        //--versore e1--//
        if (e1[i][0]*e1[i+1][0] < 0)
           e1[i+1][0] = -1.0*e1[i+1][0];
        if (e1[i][1]*e1[i+1][1] < 0)
           e1[i+1][1] = -1.0*e1[i+1][1];
        if (e1[i][2]*e1[i+1][2] < 0)
           e1[i+1][2] = -1.0*e1[i+1][2];
        //--versore e2--//
        if (e2[i][0]*e2[i+1][0] < 0)
           e2[i+1][0] = -1.0*e2[i+1][0];
        if (e2[i][1]*e2[i+1][1] < 0)
           e2[i+1][1] = -1.0*e2[i+1][1];
        if (e2[i][2]*e2[i+1][2] < 0)
           e2[i+1][2] = -1.0*e2[i+1][2];
        //--versore e3--//
        if (e3[i][0]*e3[i+1][0] < 0)
           e3[i+1][0] = -1.0*e3[i+1][0];
        if (e3[i][1]*e3[i+1][1] < 0)
           e3[i+1][1] = -1.0*e3[i+1][1];
        if (e3[i][2]*e3[i+1][2] < 0)
           e3[i+1][2] = -1.0*e3[i+1][2];
        }
        else if ((i+1) == nline_file)
        {
        //--versore e2--//
        if (e1[i][0]*e1[i-1][0] < 0)
           e1[i][0] = -1.0*e1[i][0];
        if (e1[i][1]*e1[i-1][1] < 0)
           e1[i][1] = -1.0*e1[i][1];
        if (e1[i][2]*e1[i-1][2] < 0)
           e1[i][2] = -1.0*e1[i][2];
        //--versore e2--//
        if (e2[i][0]*e2[i-1][0] < 0)
           e2[i][0] = -1.0*e2[i][0];
        if (e2[i][1]*e2[i-1][1] < 0)
           e2[i][1] = -1.0*e2[i][1];
        if (e2[i][2]*e2[i-1][2] < 0)
           e2[i][2] = -1.0*e2[i][2];
        //--versore e3--//
        if (e3[i][0]*e3[i-1][0] < 0)
           e3[i][0] = -1.0*e3[i][0];
        if (e3[i][1]*e3[i-1][1] < 0)
           e3[i][1] = -1.0*e3[i][1];
        if (e3[i][2]*e3[i-1][2] < 0)
           e3[i][2] = -1.0*e3[i][2];
        }
}

// //--Print e1, e2, e3--//
 for (i = 0; i < nline_file; i++)
        {
        for (j = 0; j < 3; j++)
           {
                        cout << e1[i][j] << "\t";
           }
        for (j = 0; j < 3; j++)
           {
                        cout << e2[i][j] << "\t";
           }
```

```
        for (j = 0; j < 3; j++)
           {
                   cout << e3[i][j] << "\t";
           }
        cout << endl;
}


//--///Extraction Euler Angles
 for (i = 0; i < nline_file-1; i++)
        {
           alpha[i] = (atan2(e3[i][1],e3[i][0]) * 180 / PI);
           beta[i] = (acos(e3[i][2]) * 180.0 / PI);
           gamma[i] = (atan2(e2[i+1][2],-1.0*e1[i][2]) * 180.0 / PI);
        }

//--///Print data to file
 file.open("domain_2_euler",ios::out);
 i = 0;
 while (i < nline_file)
         {
                   file << alpha[i] << "\t" << beta[i] << "\t " << gamma[i] <<"\n";
                   i++;
         }
 file.close();

//--///Delete dynamic allocation
 delete [] alpha;
 delete [] beta;
 delete [] gamma;

 for (i=0; i < nline_file; i++)
        {
           delete [] Data_File[i];
           delete [] e1[i];
           delete [] e2[i];
           delete [] e3[i];
        }

 return (0);
}
```

F3 – Calcolo funzioni di autocorrelazione rotazionale.

Questo script consente di calcolare gli elementi della matrice di rotazione che descrive l'orientazione relativa dei domini e le serie temporali delle matrici di Wigner.

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <math.h>
#include <string>
#include <algorithm>
#define PI 3.14159265

//--Prototypes of functions
double **transposition_E1(int nline_file, double **E1, double **E1_transp);
double **compute_E12(int nline_file, double **E1_transp, double **E2, double **E12);
void **extract_euler_angles(int nline_file, double **E12, double *alpha, double *beta, double *gamma);
void **compute_D200(int nline_file, double **E12);
void **compute_sum_D20K(int nline_file, double **E12);

using namespace std;

int main (int argc, char* argv[])
{
 if (argc < 3)
         {
           cout << "Inserisci nline_file, domain_1_element_matrix, domain_2_element_matrix" << endl;
```

```
            return 1;
        }

//--///Declaration variables: dynamic allocation
 int nline_file;
 sscanf(argv[1],"%d",&nline_file);

 int i, j, k;

 double *alpha = new double [nline_file];
 double *beta = new double [nline_file];
 double *gamma = new double [nline_file];

 double **E1, **E2, **E12, **E1_transp;
 E1 = new double *[nline_file];
 E1_transp = new double *[nline_file];
 E2 = new double *[nline_file];
 E12 = new double *[nline_file];

 for (i=0; i < nline_file; i++)
                {
                        E1[i] = new double [9];
                        E1_transp[i] = new double[9];
                        E2[i] = new double [9];
                        E12[i] = new double [9];
                }

//--///Import E1 from domain_1_rot_mat_exchange
 fstream file;
 file.open(argv[2],ios::in);
 for (k = 0; k < nline_file; k++)
                {
                for (i = 0; i < 3; i++)
                        {
                        for (j = 0; j < 3; j++)
                {
                                                file >> E1[k][i*3+j];
                                }
                        }
                }
 file.close();

//--///Import E2 from domain_2_rot_mat_exchange
 file.open(argv[3],ios::in);
 for (k = 0; k < nline_file; k++)
                {
                for (i = 0; i < 3; i++)
                        {
                        for (j = 0; j < 3; j++)
                {
                                                file >> E2[k][i*3+j];
                                }
                        }
                }
 file.close();

//--///Transposition of E1 matrix
 transposition_E1(nline_file, E1, E1_transp);

//--///Computation of E12 matrix
 compute_E12(nline_file, E1_transp, E2, E12);

//--///Extraction Euler Angles
 extract_euler_angles(nline_file, E12, alpha, beta, gamma);

// compute_D200(nline_file, E12);

// compute_sum_D20K(nline_file, E12);

//--///Delete dynamic allocation
 delete [] alpha;
 delete [] beta;
 delete [] gamma;
 for (i=0; i < nline_file; i++)
                {
                        delete [] E1[i];
                        delete [] E1_transp[i];
```

```
                delete [] E2[i];
                delete [] E12[i];
        }

 return (0);
}//end main

double **transposition_E1(int nline_file, double **E1, double **E1_transp)
{
 int i, j, k;
 for (k = 0; k < nline_file; k++)
        {
        for (i = 0; i < 3; i++)
                {
                for (j = 0; j < 3; j++)
        {
                                        E1_transp[k][i*3+j] = E1[k][j*3+i];
                        }
                }
        }
 return (E1_transp);
}

double **compute_E12(int nline_file, double **E1_transp, double **E2, double **E12)
{
 int i, j, k, l;
 for (k = 0; k < nline_file; k++)
        {
        for (i = 0; i < 3; i++)
                {
                for (j = 0; j < 3; j++)
        {
                        E12[k][i*3+j] = 0.0;
                        for (l = 0; l < 3; l++)
                        {
                                        E12[k][i*3+j] = E12[k][i*3+j] + E2[k][i*3+l] * E1_transp[k][l*3+j];
                        }
                        }
                }
        }
 return (E12);
}

void **extract_euler_angles(int nline_file, double **E12, double *alpha, double *beta, double *gamma)
{
 int i, j, k, f;
 double ac[11], at, diff, check, big, periodo;

 periodo = 360.0;
 big = 1.0e16;

 for (k = 0; k < nline_file; k++)
        {
                        alpha[k] = (atan2(E12[k][7],E12[k][6]) * 180 / PI);
                        beta[k] = (acos(E12[k][8]) * 180.0 / PI);
                        gamma[k] = (atan2(E12[k][5],-1.0*E12[k][2]) * 180.0 / PI);
        }//end for

 //Print data to file
 fstream file;
 file.open("CaM_free_E12_euler_matrix",ios::out);
 for (k = 0; k < nline_file; k+
   for (i = 0; i < 3; i+
                {
                for (j = 0; j < 3; j++)
        {
                                        file << E12[k][i*3+j] << "\t";
                        }
                }
        file << "\n";
        }
 file.close();

 file.open("CaM_E12_euler_angles",ios::out);
 k = 0;
 while (k < nline_file)
        {
```

```
                                        file << alpha[k] << "\t" << beta[k] << "\t " << gamma[k] <<"\n";
                                        k++;
                }
 file.close();
}

void **compute_D200(int nline_file, double **E12)
{
 int k, i;
 double *Re_D200 = new double [nline_file];
 double *Im_D200 = new double [nline_file];
//--///Compute D200(t)
 for (k = 0; k < nline_file; k++)
                {
                        Re_D200[k] = (3.0*(E12[k][8] * E12[k][8])-1.0)/2.0;
                        Im_D200[k] = 0.0;
                }
//--///Print data to file
 fstream file;
 file.open("./CaM_free_D200",ios::out);
 k = 0;
 while (k < nline_file)
                {
                                        file << Re_D200[k] <<"\t"<< Im_D200[k] << endl;
                                        k++;
                }
 file.close();
//--///Delete dynamic allocation
 delete [] Re_D200;
 delete [] Im_D200;
}

void **compute_sum_D20K(int nline_file, double **E12)
{
//--///Declaration variables
 int i, j, k;
 double *Re_D20K = new double [nline_file];
 double *Im_D20K = new double [nline_file];

//--///Compute D202(t)
 for (k = 0; k < nline_file; k++)
                {
                        Re_D20K[k] = 1.2248*E12[k][6]*E12[k][6]*E12[k][4];
                        Im_D20K[k] = 0.0;
                }

//--///Print data to file
 fstream file;
 file.open("./CaM_free_sum_D202_D20_2",ios::out);
 k = 0;
 while (k < nline_file)
                {
                                        file << Re_D20K[k] <<"\t"<< Im_D20K[k] << endl;
                                        k++;
                }
 file.close();

//--///Delete dynamic allocation

 delete [] Re_D20K;
 delete [] Im_D20K;
}
```

Bibliografia

[1] Allen, M. P.; Tildesley, D. J. Computer Simulation of Liquids, **1989**, Oxford University Press, USA.

[2] Frenkel, D.; Smit, B. Understanding Molecular Simulation, 2nd edition: From Algorithms to Applications, **2001**, Academic Press.

[3] Ercolessi, F. A Molecular Dynamics Primer, www.fisica.uniud.it/~ercolessi/md/ (accesso **10/01/12**).

[4] Polimeno, A. Dispensa Chimica Teorica, **2009-2010**, Padova.

[5] Adcock, S. A.; McCammon, J. A. *Chem. Rev.* **2006**, 106, 1589.

[6] Phillips, J.C.; Schulten, K.; et al. *J. Comput. Chem.* **2005**, 26, 1781.

[7] Brooks, B.R.; Karplus, M.; et al. *J. Comput. Chem.* **2009**, 30, 1545.

[8] Hess, B.; Lindah, L.; et al. *J. Chem. Theory Comput.* **2008**, 4, 435.

[9] Case, D. A.; et al. AMBER 11, **2010**, University of California, San Francisco.

[10] Karplus, M.; Petsko, G. A. *NATURE* **1990**, 347, 631.

[11] Karplus, M.; McCammon, J. A. *Nat. Struct. Biol.* **2002**, 9, 646.

[12] Klepeis, J. L.; Shaw, D. E.; et al. *Curr. Opin. Struct. Biol.* **2009**, 19, 120.

[13] Jorgensen, W. L.; Chandrasekhar, J.; et al. *J. Chem. Phys.* **1983,** 79, 926.

[14] Ponder, J. W.; Case, D. A. *Adv. Protein. Chem.* **2003**, 66, 27.

[15] Stone, A. J.; Schulten, K.; et al. *J. Mol. Graphics and Modelling* **2010**, 29, 116.

[16] Stone, J. E.; Schulten, K.; et al. *J. Comput. Chem.* **2007**, 28, 2618.

[17] Smith, P. E.; Van Gunsteren, W. F. *J. Mol. Biol.* **1994**, 236, 629.

[18] Wong, V.; Case, D. A. *J. Phys. Chem. B* **2008,** 112, 6013.

[19] Belford, G. G.; Weber, G.; et al. *Proc. Nat. Acad. Sci. USA* **1972**, 69, 1392.

[20] Wegener, W. A.; Koester, V. J.; et al. *J. Chem. Phys.* **1979,** 70, 622.

[21] Schmitz, K. S.; Schurr, J. M. *Biopolymers* **2004**, 12, 1543.

[22] Ghose, R. H.; Cowburn, D.; et al. *J. Magn. Res.* **2001**, 149, 204-217.

[23] Markwick, P. R. L.; Nilges, M. *Chem. Phys.* **2011**.

[24] Moro, G. *Chem. Phys.* **1987**, 118, 167.

[25] Moro, G. *Chem. Phys.* **1987**, 118, 181.

[26] Barone, V.; Zerbetto, M.; Polimeno, A. *J. Comput. Chem.* **2009**, 30, 2.

[27] De La Torre, J. G.; Carrasco, B.; et al. *Biophysical Journal*, **2000**, 78, 719.

[28] Ryabov, Y. E.; Fushman, D.; et al. *J. Am. Chem. Soc.* **2006**, 128, 15432.

[29] Favro, L. D. *Phys. Rev.* **1960**, 119, 53.

[30] Kalmykov, Y. P. *J. Chem. Phys.* **2009,** 130, 134105.

[31] Leicknam, J. C.; Bratos, S.; et al. *Phys. Rev. A* **1980**, 21, 1005.

[32] Hinze, G.; Basché, T.; et al. *Phys. Rev. Lett.* **2004**, 93, 20, 203001.

[33] Pekka, M.; Nilsson, L. *J. Comput. Chem.* **2002**, 23, 1211.

[34] Takemura, K.; Kitao, A. *J. Phys. Chem. B* **2007,** 41, 11870.

[35] González, M. A.; Abascal, J. L. F. *J. Chem. Phys.* **2010,** 132, 096101.

[36] Wlodawer, A.; Nachman, J.; Gilliland, G. L.; Gallagher, W.; Woodward, C. *J. Mol. Biol.* **1987**, 198, 469.

[37] Ulmer, T. S.; Bax, A.; et al. *J. Am. Chem. Soc.* **2003**, 125, 9179.

[38] Derrick, J. P.; Wigley, D. B. *J. Mol. Biol.* **1994**, 243, 906.

[39] Sjöbring, U.; Kastern, W.; et al. *J. Biol. Chem.* **1991**, 266, 399.

[40] Ramanadham, M.; Jensen, L. H.; et al. *Acta Cryst.* **1990**, B46, 63.

[41] Richardson, J.S.; *Adv. Prot. Chem.* **1981**, 34, 167.

[42] Kassel, B.; Laskowski, M; et al. *Biochem. Biophis. Res. Commun.* **1965**, 18, 255.

[43] Johnson, L. N.; Phillips, D. C. *Nature* **1965**, 206, 761.

[44] Humphrey, W.; Dalke, A.; Schulten, K. *J. Molec. Graphics* **1996**, 14, 33.

[45] Bloomfield, V. A. *Science* **1968**, 161, 1212.

[46] Rotne, J.; Prager, S. *J. Chem. Phys.* **1969,** 50, 4831.

[47] Schmidt, J. R.; Skinner, J. L. *J. Chem. Phys.* **2003,** 119, 8062.

[48] Padding, J. T.; Louis, A. A.; et al. *J. Phys.: Condens. Matter* **2005**, 17, S3393.

[49] Bora, R. P.; Prabhakar, R. *J. Chem. Phys.* **2009,** 131, 155103.

[50] Lipari, G.; Szabo, A. *J. Am. Chem. Soc.* **1982**, 104, 4546.

[51] Daragan, V. A.; Mayo, K. H. . *J. Phys. Chem. B* **1999,** 103, 6829.

[52] Wittebort, R. J.; Szabo, A. *J. Chem. Phys.* **1978,** 69, 1722.

[53] Wallach, W. *J. Chem. Phys.* **1963,** 47, 5258.

[54] Steele, W. A. *J. Chem. Phys.* **1963,** 38, 2404.

[55] Steele, W. A. *J. Chem. Phys.* **1963,** 38, 2411.

[56] Zuo, C. S.; Wiest, O.; Wu, Y. D. *J. Chem. Phys. A* **2009,** 113, 12028.

[57] Baber, J. B.; Tjandra, N.; et al. *J. Am. Chem. Soc.* **2001**, 123, 3953.

[58] Kuboniwa, H.; Bax, A.; et al. *Nat. Struct. Biol.* **1995**, 2, 768.

[59] Sheperd, C. M.; Vogel, H. J. *Biophysical Journal*, **2004**, 87, 780.

[60] Polnaszek, C. F.; Freed, J. H. *J. Phys. Chem.* **1975**, *21*, 2283.

[61] Powell, M. J. D. *Comp. J.* **1964**, *7*, 155.

[62] Tjandra, N.; Bax, A. *Eur. J. Biochem.* **1995**, 1014.

[63] Chattopadhyaya, R.; Quiocho, F.; et al. A. *J. Mol. Biol.* **1992**, 228, 1177.

[64] Goldstein, H. Classical Mechanics, 3rd edition, **2001**, Addison Wesley.

[65] Risken, H. The Fokker-Planck Equation: Methods of Solutions and Applications, 2nd edition **1989**, Springer Series in Synergetics, Springer.

[66] Gardiner, C. W. Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences, 3rd edition **2004**, Springer Series in Synergetics, Springer.

[67] Kramers, H. A. *Physica VII* **1940**, 4, 284.

[68] en.wikipedia.org.

[69] Ascenzi, P.; Menegatti, E.; et al. *Curr. Protein. Peptide Sci.* **2003**, 4, 231.

[70] Cossi, M.; Barone, V.; et al. *J. Chem. Phys.* **2002,** 117, 43.

[71] Barone, V.; Cossi, M.; Tomasi, J. *J. Chem. Phys.* **1997,** 107, 3210.

[72] Zare, R. N. Angular Momentum: Understanding Spatial Aspects in Chemistry and Physics, 1st edition **1988**, Wiley-Interscience.

[73] Varshalovich, D. A.; Moskalev, A. N.; Khersonskii, V. K. Quantum Theory of Angular Momentum: Irreducible Tensors, Spherical Harmonics, Vector Coupling Coefficients, 3*nj* Symbols, **1988**, World Scientific.

[74] Rose, M. E. Elementary theory of angular momentum, **1957**, N. Y. John Wiley & Sons, Inc.

[75] www.chimica.unipd.it/licc

[76] Zerbetto, M.; Meirovitch, E.; et al. *J. Phys. Chem.* **2009**, *113*, 13613.

Ringraziamenti

Desidero ringraziare tutto il gruppo di Chimica Teorica, in particolare il Professor Antonino Polimeno per avermi dato la possibilità, attraverso questo lavoro, di approfondire alcuni aspetti della Chimica Computazionale e il Dott. Mirco Zerbetto per avermi aiutato nella risoluzione dei persistenti problemi informatici e non.

Desidero ringraziare di cuore la mia Famiglia, in particolare i miei genitori e mia nonna, per avermi sempre sostenuto in questi anni universitari.

Un saluto e un abbraccio ai compagni della Sala Calcolo con i quali ho passato ore interminabili davanti al computer: Elisa, Matteo, Marco, Mauro e Cristina.

Un saluto agli amici delle Cucine Popolari di Padova e agli amici della compagnia.