Corso di Laurea Magistrale in Scienze Chimiche, Curriculum "Struttura, dinamica e reattività chimica" "Struttura e Dinamica Molecolare di Sistemi Biologici" 2011/2012

Appunti per le esercitazioni di Laboratorio

Dr. Giorgio F. Signorini
Università di Firenze
Dipartimento di Chimica
giorgio.signorini@unifi.it
http://www.chim.unifi.it/~signo

(20 dicembre 2011)

Indice

Parte I

Introduzione

Capitolo 1

Preliminari

(laboratorio n.1 - 7/11/2011)

1.1 assegnazione account

- Assegnazione nome utente e password

1.2 ripasso UNIX[1] e configurazione utente

1. Aprire una sessione facendo login con il nome del vostro utente e password.
2. Aprire un terminale usando il menù principale del Desktop.
3. Ripassare la "sintassi generale di un comando" [tab. 4.1]
4. Ripassare il "percorso assoluto e relativo" per indicare file e directory [fig. 4.1]
5. Ripassare i comandi per la "gestione essenziale di file e directory" [tab. 4.2]
6. Copiare il file `~signorini/etc/bashrc.global` su `~/.bashrc`
7. Copiare il file `~signorini/etc/init.el` su `~/.emacs`
8. uscire dal terminale e rientrare

1.3 Editor di testo: emacs[2]

1. differenza tra editor di testo e word processor [3]
2. Lanciare emacs, col comando
   `emacs`
3. provare a uscire
4. rientrare lanciando il comando in background
   `emacs &`
5. osservare le varie aree dello schermo
   (a) barra menu
   (b) area di lavoro: il buffer
   (c) riga di stato (Mode Line)
   (d) minibuffer

[1] fare riferimento alle dispense sul "sistema operativo" in `http://www.chim.unifi.it/u/signo/did/unix`
[2] sez. 5.1 `http://www.chim.unifi.it/~signo/did/unix/lez-split/node96.html`
[3] sez. 4.5.1 `http://www.chim.unifi.it/~signo/did/unix/lez-split/node54.html`

6. menu, scorciatoie e comandi “complessi”
7. help: Apropos, Key, Whereis (locate); Tutorial
8. differenza tra buffer e file
9. editare il file `~/.plan`
    (a) inserire n.o matricola e codice corso di laurea
10. “editare” una directory: dired

Capitolo 2

Piano di lavoro

Nelle nostre esercitazioni studieremo le proprietà di una serie di piccoli peptidi che hanno una certa propensione ad assumere la strutture di "forcina β" (β-*hairpin*). Essi sono derivati da un frammento del dominio B1 della proteina G (PDB code `2gb1`); il modello è costituito da un 10-peptide denominato GPM12, la cui struttura secondaria in soluzione però non è stata risolta

GPM12	Gly1-Tyr2-Asp3-**Asp4-Ala5-Thr6-Lys7**-Thr8-Phe9-Gly10	`GYDDATKTFG`

Possiamo analizzare i seguenti mutanti di GPM12:

D4P	Gly1-Tyr2-Asp3-**Pro4-Ala5-Thr6-Lys7**-Thr8-Phe9-Gly10	`GYDPATKTFG`
K7G	Gly1-Tyr2-Asp3-**Asp4-Ala5-Thr6-Gly7**-Thr8-Phe9-Gly10	`GYDDATGTFG`
D4P/K7G	Gly1-Tyr2-Asp3-**Pro4-Ala5-Thr6-Gly7**-Thr8-Phe9-Gly10	`GYDPATGTFG`

Di quest'ultimo doppio-mutante è nota la struttura in soluzione (PDB code `2e4e`).

Costruiamo le proteine in una conformazione (il più possibile) lineare, in modo da verificare se evolvono spontaneamente verso la struttura a forcina, e quali sono i fattori che condizionano questa evoluzione (sequenza primaria, presenza del solvente, condizioni termodinamiche (temperatura), ...).

Il piano di lavoro è il seguente:

1. costruzione della molecola
2. minimizzazione dell'energia
3. prima simulazione con controllo della conservazione di E
4. simulazione nel vuoto
5. simulazione nel solvente
6. analisi dei risultati della simulazione classica

Capitolo 3

programma VMD

Una versione funzionante del programma VMD dovrebbe essere già attiva (con la configurazione creata al punto precedente, il comando `'vmd'` dovrebbe lanciare `~signorini/bin/vmd`)

3.1 Costruzione di proteine: `molefacture`

Usare la versione 1.8.7 di VMD perché il `molefacture` della 1.8.6 non funzione (prolina sbagliata).

Menu `Extensions / Modeling / Molefacture` : partire da molecola vuota

Menu `Build / Protein Builder` : aggiungere i residui configurando gli angoli di Ramachandran a 180° e −180° (straight).

Alla fine modificare i residui terminali, aggiungendo un `H` all'`N` terminale ("*Add hydrogen to selected atom*") e un ossidrile al C terminale (Menu `Build / Replace hydrogen with fragment`)

Salvare come PDB e ricaricare la molecola in VMD. È possibile che alcuni atomi di catene laterali siano venuti troppo vicini e che ce li presenti come legati. Invece di cercare di muovere pezzi della molecola con Molefacture, proviamo a minimizzare l'energia potenziale in ORAC. In ogni caso si possono cancellare gli atomi di `H` perché `ORAC` li riassegna automaticamente.

Capitolo 4

programma ORAC

Una versione funzionante del programma ORAC dovrebbe essere già attiva (il comando `'orac'` dovrebbe lanciare `~signorini/bin/orac`). Altrimenti, eseguire installazione e test come di seguito:

4.1 installazione

Scaricare la versione aggiornata del programma ORAC dal sito ufficiale `http://www.chim.unifi.it/orac`, spacchettarla e compilarla sul proprio PC seguendo le istruzioni.

4.2 test

Eseguire i test fondamentali (directory `tests/basic_tests`)
Analisi del file di input

4.3 rendere disponibile il programma

[fatto nel laboratorio n.2]
Per utilizzare il programma appena compilato, è comodo poterlo richiamare in modo semplice da qualunque directory. Lo si fa copiando il file eseguibile (`orac_Linux`) su una directory che si trovi sul `PATH`, ad esempio `~/bin`. Meglio ancora è creare un link simbolico (in questo modo anche se si ricompila lo si ritrova sempre):
`ln -s orac.5_x_x.ryyyy/src/default/orac_Linux ~/bin/orac`

4.4 documentazione

presentazione del programma e dei file di input e output:
`http://www.chim.unifi.it/orac/orac-advanced-present.pdf`

Parte II

Simulazioni classiche

Capitolo 5

Minimizzazione energia della struttura di partenza con ORAC

5.1 preparazione ambiente di lavoro

1. Creare una directory di lavoro (ad esempio `~/orac`) con sottodirectory `lib` e `pdb` (Si può copiare tutte queste sottodirectory da `~signorini/orac`)

5.2 preparazione input

5.2.1 input principale

Come già visto, definisce tre file di dati

1. coordinate (PDB)
2. topologia (`.tpg`)[
3. parametri di potenziale (`.prm`)

Questi file devono usare gli stessi nomi per gli oggetti che definiscono. In particolare, le coerenze devono esistere tra:

nomi di:	esempi	input ORAC	PDB	`.tpg`	`.prm`
residui	`ile, gly-h`	X		X	
atomi (label)	`ca, hg21`		X	X	
tipi atomici	`hc`			X	X

Nell'istruzione `JOIN SOLUTE` del file di input va data la sequenza dei residui, che si può ricavare dal file PDB. Ricordare che i due residui terminali sono definiti a parte nel file delle topologie.

5.2.2 struttura

[fatto nel laboratorio n.2]
La struttura creata con VMD usa in certi casi dei nomi di atomi diversi da quelli di AMBER. Questi vanno "tradotti" nei corrispondenti nomi di AMBER; lo si fa confrontando il file PDB e il file di topologia (nei casi dubbi, usando VMD per identificare gli atomi). Es:
`HN-> H`

5.2.3 topologia

Notare che le due glicine terminali sono, rispettivamente, N-terminale e C-terminale, ovvero la sequenza, in termini di residui definiti nella topologia AMBER, è

```
gly-h ...  gly-o
```

5.2.4 parametri di potenziale

5.3 minimizzazione energia

Si può eseguire la minimizzazione con due metodi: Steepest Descent e Conjugate Gradient

5.3.1 preparazione input ed esecuzione

1. Portarsi sulla directory di lavoro
2. Creare un input (es `minim.in`) che esegua una minimizzazione dell'energia della molecola:

`minim.in`

```
&SETUP
  CRYSTAL 200.0 200.0 200.0
  READ_PDB ../pdb/D4PK7G_all.pdb
&END

&PARAMETERS
  READ_TPG_ASCII ../lib/amber03.tpg
  READ_PRM_ASCII ../lib/amber03.prm
  JOIN SOLUTE
    gly-h tyr asp pro ala thr gly thr phe gly-o
  END
&END

&POTENTIAL
  EWALD OFF
  CUTOFF 100.
  STRETCHING
&END

&SIMULATION
  MINIMIZE
    CG 0.00001
    # oppure SD 0.00001
  END
&END

&INOUT
  ASCII 100.0 OPEN minim.pdb
&END

&RUN
  TIME    3000.0
  PRINT    100.0
  PROPERTY 100.0
&END
```

3. lanciare il programma con

   orac < minim.in > minim.out

(laboratorio n.1 - 7/11/2011 - fine)

5.3.2 correggere il file della struttura (vedi sopra)

5.3.3 analisi risultati

1. Visualizzare i risultati con vmd
2. Misurare con vmd le variazioni rispetto alla struttura di partenza di:
   (a) RMSD
   (b) angoli di Ramachandran
   (c) distanza *"head-to-tail"*
   (d) distanza tra residui 4 e 7
   (e) Analizzare i diversi contributi all'energia.
   Ad es:
   ```
   orac-post-out P="TotPot Bonded NonBond" min.out > ene
   plot -1,2-4 ene
   ```
   (f) Osservare
      i. se vi è correlazione tra variazione nei RMSD e nell'energia
      ii. quale modifica conformazionale determina la massima variazione di energia
      iii. quale parte del potenziale guida la transizione verso il minimo.
      N.B. NonBond=Coulomb+Ener14+LennardJones (non stampata!)
      iv. nel caso che ci sia una struttura sperimentale, fare il confronto con quella.

Capitolo 6

simulazione della proteina a 300 K nel vuoto (E,V=cost).

6.1 primo run: controllo della conservazione di E

1. portarsi in `~/orac/data`; creare un input di ORAC (es. `NVE-1.in`) per una simulazione:

   (a) nel potenziale si omette il cutoff (i cutoff per le varie shell sono definiti nel blocco `&INTEGRATOR`), e si levano gli stretching C-H, N-H, O-H

   NVE.in (1/3)

   ```
   ...
   &POTENTIAL
     [...]
     #CUTOFF 100.
     STRETCHING HEAVY
   &END
   ...
   ```

   (b) inserire il blocco `&INTEGRATOR` e modificare il blocco `&SIMULATION`:

   NVE.in (2/3)

   ```
   ...
   &SIMULATION
     MDSIM
     TEMPERATURE 300.0 40.0
   &END

   &INTEGRATOR
     TIMESTEP 2.0
     MTS_RESPA
       step intra 2
       step intra 2
       step nonbond 1 100.
       test_times OPEN energie
     END
   &END
   ...
   ```

   Note:
   - nel caso di una molecola nel vuoto si suddivide solo il potenziale "legato"; quello "non legato" ha un timestep unico.
   - il timestep del potenziale non legato deve essere più basso che nel solvente, anche a medio-lungo raggio (tanto l'aggravio di calcolo è trascurabile); questo perché i moti sono presumibilmente più veloci
   - il cutoff deve essere alto (cariche), ma meno del lato della pseudo-cella

   (c) modificare il blocco `&RUN` :

NVE.in (3/3)

```
...
&RUN
  CONTROL       0
  REJECT      0.0
  TIME    10000.0
  PRINT     100.0
  PROPERTY 1000.0
&END
```

2. lanciare il programma con output sullo schermo

   orac < protein.in

3. lanciare il programma con output su file

   orac < protein.in > protein.out &

4. verificare la conservazione dell'Hamiltoniana totale (media stabile):

   ```
   plot -2,3 energie
   ```

5. se l'energia non si conserva, diminuire i timestep

6. notare che alla fine la T è bassa: siamo partiti dal minimo dell'energia potenziale, una parte dell'energia cinetica che abbiamo dato al sistema viene spesa per alzare l'energia potenziale media.

(laboratorio n.2 - 14/11/2011- fine)

6.2 simulazione con equilibratura iniziale

1. modificare il blocco `&RUN` inserendo un ciclo di equilibratura appropriato:

NVE-eq.in

```
...
&RUN
  CONTROL 0
  REJECT  50000.0
  TIME    30000.0
  PRINT     100.0
  PROPERTY 1000.0
&END
```

6.3 Analisi delle proprietà

Analizzare

- Energie (del sistema)

  ```
  plot -2,7-9 energie
  ```

- RMSD rispetto a struttura di partenza e a struttura sperimentale del D4P/K7G:
  `/home1/signorini/biomol/orac/GPM12-mutants/pdb/2e4e-mod1.pdb`
- Angoli di Ramachandran
- distanza "*head-to-tail*" e $C_\alpha^4 - C_\alpha^7$

6.3.1 ruolo della minimizzazione iniziale

Si può fare un run di confronto, con lo stesso periodo di equilibratura, ma partendo da una struttura non minimizzata (o minimizzata pochi step giusto per sistemare le sovrapposizioni di atomi)

- Se partendo dal minimo dell'energia durante l'equilibratura viene immessa energia cinetica, partendo lontano dal minimo è probabile che sia necessario rimuoverla.
- in entrambi i casi la struttura si evolve durante l'equilibratura; con tutta probabilità la configurazione alla fine dell'equilibratura sarà diversa

6.3.2 uso di strumenti di ORAC per calcolare proprietà

(non fatto)

1. Calcolare gli RMSD con `rmsd.sh`.

   Ad es:

   rmsd.sh -r pdb/D4PK7G-backb.pdb pdb/D4PK7G_min.pdb NVE-eq.pdb > NVE-eq.rmsd

   plot -0,4 NVE-eq.rmsd

   (a) per confrontare solo il backbone occorre prima costruire un file con i soli atomi di backbone; lo si può fare con alcune semplici righe di `awk`, riunite nello shell script *mk-pdb-backbone.sh*. Il comando da dare è ad es:

   mk-pdb-backbone.sh D4PK7G_min.pdb > D4PK7G-backb.pdb

2. calcolare con strumento `analysis` di ORAC:

   (a) angoli di Ramachandran

   (b) distanza "*head-to-tail*"

   (c) distanza tra residui 4 e 7

   Ad esempio, il seguente input produce l'evoluzione di tre parametri nella traiettoria `loop_tmp.pdb`: la distanza testa-coda, la distanza res4- res7, gli angoli di Ramachandran del residuo 7:

   analysis.in

```
  # name of the PDB file to analize:
NVE-eq.pdb
  # head-to-tail (D4P):
& atom_distance 5 139
  # CA 4-7 distance (D4P):
& atom_distance 44 83
  # Ramachandran angles of residue 7
& ramachandran 7
```

   Il programma si lancia con

   ```
   $ analysis < analysis.in
   ```

   e crea un file per ogni variabile. Questi si possono rinominare opportunamente con comandi tipo

   ```
   $ cp atom_distance.1.out NVE-eq.h2t
   $ cp atom_distance.2.out NVE-eq.4t7
   $ cp ramachandran.3.out  NVE-eq.ram7
   ```

Capitolo 7

simulazione (N,V,T) nel vuoto

1. preparazione dell'input; si deve solo modificare il blocco `&SIMULATION`

NVT.in

```
...
&SIMULATION
...
  THERMOS
    cofm 30.0
    solute 30.0
    solvent 30.0
    temp_limit 1000.
  END
&END
...
```

2. verifica delle proprietà del sistema

   (a) energie

      i. L'energia del sistema, EREAL, non è costante; è costante l'Hamiltoniano di Nosé-Hoover $ETOT = EREAL + KINH + HPOT$

$$H_{NH} = \sum \frac{p_i^2}{2m_i} + U(\mathbf{r}) + \frac{p_\eta^2}{2Q} + \frac{L}{\beta}\eta \tag{7.1}$$

$$EREAL = \sum \frac{p_i^2}{2m_i} + U(\mathbf{r}) \tag{7.2}$$

$$KINH = \frac{p_\eta^2}{2Q} \tag{7.3}$$

$$HPOT = \frac{L}{\beta}\eta \tag{7.4}$$

      ii. confrontare l'andamento e soprattutto le fluttuazioni dell'energia totale nell'insieme NVT con quella dell'insieme NVE con medesimo input

      iii. vedere la dipendenza del risultato dal valore delle "masse" del termostato;

      iv. verificare che se $Q \to \infty$ si ricade nell'insieme N, V, E

      v. verificare che le fluttuazioni dell'energia cinetica sono dell'ordine atteso ($\sim \frac{1}{\sqrt{N}}$)

   (b) RMSD

   confrontare l'andamento e il valore finale di RMSD nell'insieme NVT con quelle dell'insieme NVE con medesimo input

Capitolo 8

Simulazioni con solvente.

8.1 Preparazione input

1. Volume cella e densità solvente

   (a) lato cella: è opportuno che sia almeno il doppio della lunghezza della proteina (stirata)

   (b) densità dell'acqua:

$$\begin{aligned} \rho/amu \cdot \mathring{A}^{-3} &= \rho/g \cdot cm^{-3} \cdot \frac{g}{amu} \cdot \left(\frac{\mathring{A}}{cm}\right)^3 \\ &= \rho/g \cdot cm^{-3} \cdot N_A \cdot \left(10^{-8}\right)^3 \\ &= \rho/g \cdot cm^{-3} \cdot 0.6023 \\ &\simeq 0.6 \end{aligned}$$

   Per m^3 molecole di acqua in un box di lato L Angstrom:

$$\rho/amu \cdot \mathring{A}^{-3} = 0.6 = \frac{18m^3}{L^3}$$

   da cui

$$L = 3.107m$$

   Ad esempio

$$\begin{aligned} m &= 10 \\ L &= 31.07 \end{aligned}$$

   (c) Quando si inserisce il soluto, questo si sovrappone al solvente; si cancellano molecole di solvente entro un certo raggio dagli atomi del soluto, raggio definito dal parametro `INSERT` del blocco `&SOLVENT` (due molecole si considerano sovrapposte se la somma delle loro distanze è minore del loro raggio di Lennard-Jones moltiplicato per `INSERT`). Notare che in questo modo la densità (e pressione) sperimentale non è più rispettata: la soluzione ideale sarà fare simulazione a pressione costante, almeno all'inizio, e lasciare che il volume si aggiusti (cfr. avanti).

2. creare sulla propria directory di lavoro il file di input, usando come modello ad es:
   `~signorini/biomol/orac/GPM12-mutants/D4P_K7G/slv/slv-nvt.in`

3. copiare nel proprio ambiente di lavoro il file di coordinate della molecola di H_2O:
   `~signorini/biomol/orac/pdb/water.pdb`

4. Notare le differenze del file di input rispetto al file utilizzato per la simulazione nel vuoto (magari usando il comando ediff di Emacs):

   (a) aggiungere solvente

slv-nvt.in (1/4)

```
&SETUP
  CRYSTAL 31.07 31.07 31.07
&END

&SOLUTE
  COORDINATES ../pdb/D4PK7G_min.pdb
#  charge neutrality not strictly necessary if Ewald
#  SCALE_CHARGES 1 1
&END

&SOLVENT
  CELL SC
  INSERT 0.8
  COORDINATES ../pdb/water.pdb
  GENERATE RANDOMIZE 10 10 10
&END

&PARAMETERS
  [...]
  JOIN SOLVENT
    spce
  END
&END
...
&SIMULATION
...
  THERMOS
    cofm       30.0
    solute     30.0
    solvent    30.0
    temp_limit 1000.
  END
&END
...
```

(b) aggiornare i time-step

slv-nvt.in (2/4)

```
&INTEGRATOR
  TIMESTEP 10.0
  MTS_RESPA
    step intra 2
    step intra 2
    step nonbond 2 4.7
    step nonbond 3 7.5 reciprocal
    step nonbond 1 9.7
    test_times OPEN energie
  END
&END
```

(c) Ewald

slv-nvt.in (3/4)

```
&POTENTIAL
  EWALD PME 0.43 30 30 30 4
  [...]
&END
```

(d) salvare un restart

slv-nvt.in (4/4)

```
&INOUT
  RESTART
    write 500.0 OPEN D4PK7G.rst
```

```
  END
  ASCII 200.0 OPEN D4PK7G.pdb
  [...]
&END
```

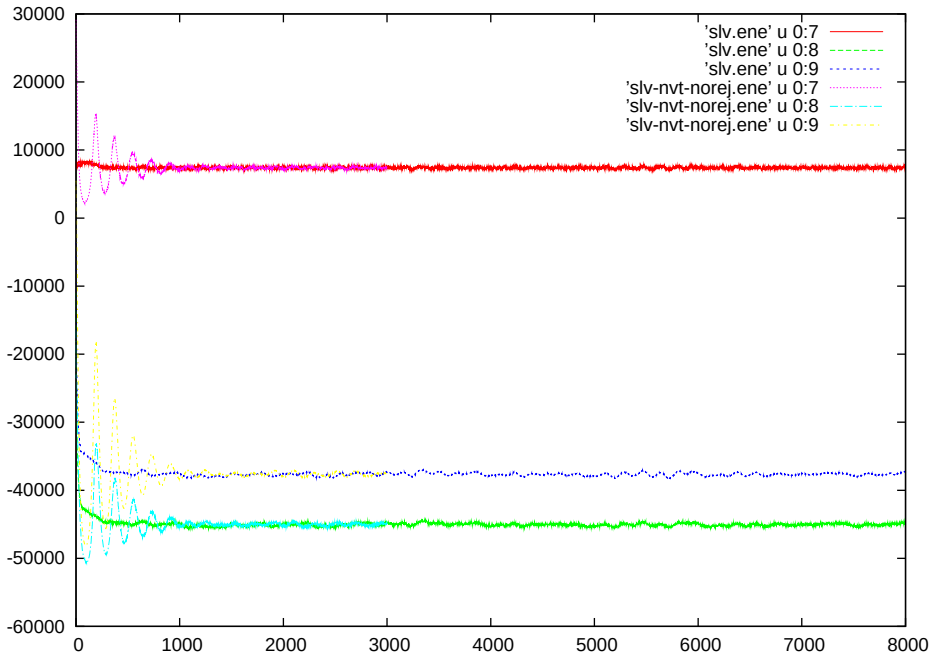
8.1.1 Verifica conservazione E

È bene verificare la scelta dei timestep e dei cutoff associati. La quantità che si deve conservare è `ETOT` nel file `energie`. Notare che se il solvente è stato appena creato l'eccesso di energia è grande e può darsi che il termostato non ce la faccia a assorbirlo: bisogna prima fare un'equilibratura di qualche picosecondo.

8.1.2 Equilibrazione

1. Anche in simulazioni a $T = cost$ conviene fare una prima parte di equilibrazione, per evitare che il termostato si scaldi troppo. Eventualmente si riparte dal restart.

Figura 8.1: confronto di energia cinetica, totale e potenziale con e senza ciclo di equilibratura



2. notare quanto è più lenta la simulazione aggiungendo il solvente (circa 10 volte)
3. monitorare le energie, in particolare:
   (a) Energia totale EREAL [scende , poi cost]
   (b) EPTOT
   (c) EKIN [oscilla intorno a 300K]
   (d) ESLV [scende]
   (e) ESLV-SLT [scende?]
   (f) ESLT [potrebbe salire]

   Tutte queste grandezze sono listate nel file `energie.`

   Si controllano usando gnuplot
4. Visualizzare il sistema in VMD
   (a) vedere se la RMSD rispetto alla conformazione sperimentale è minore che nel vuoto

8.1.3 usare pressione costante

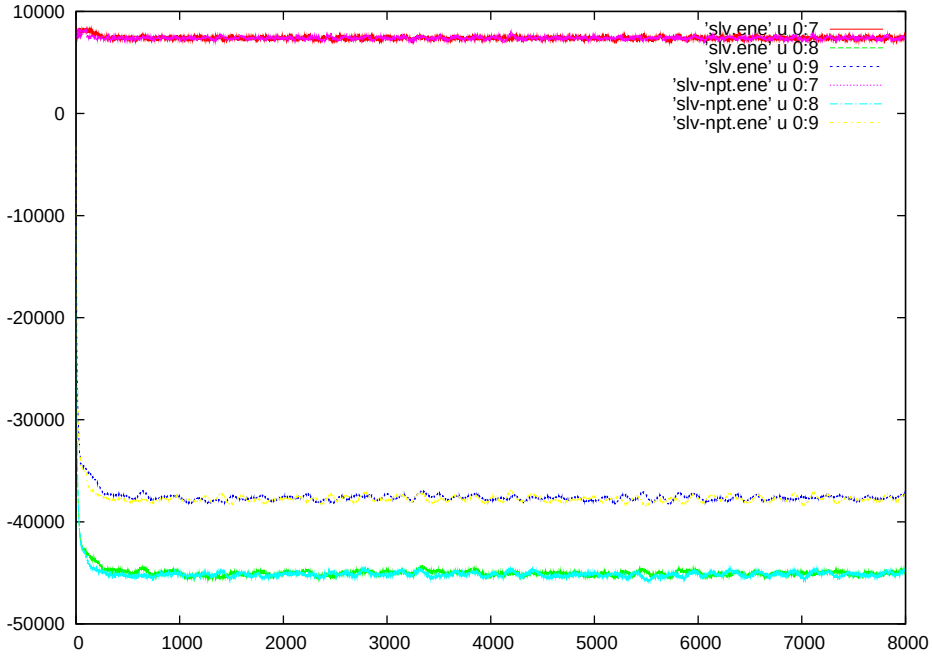
È opportuno fare una simulazione N, p, T per avere la densità corretta. Si parte da un alto valore del raggio di sovrapposizione e si permette alla cella di comprimersi sotto l'effetto della pressione esterna ($0.1MPa = 1atm$)

slv-npt.in

```
&SOLVENT
  CELL SC
  INSERT 1.4   # <- NOTE THIS
  COORDINATES ../pdb/water.pdb
  GENERATE RANDOMIZE 10 10 10
&END

&SIMULATION
...
THERMOS
...
END
ISOSTRESS PRESS-EXT 0.1 BARO-MASS 10. # <- NOTE THIS
&END
```

Figura 8.2: confronto di energia cinetica, totale e potenziale in NVT e NpT



In questa simulazione monitorare le energie delle variabili estese:

- `PV`
- `HPOT`, `KINH`

8.1.4 Simulazioni di solvente puro

(non fatto)
Per verificare la correttezza del potenziale e di tutto l'input, si possono fare delle simulazioni del solo solvente, verificando la struttura del liquido con la funzione di calcolo della $g(r)$ di VMD

(laboratorio n.3 - 21/11/2011- fine)

Parte III

Simulazioni REM

Capitolo 9

Simulazioni H-REM in ORAC

9.1 Descrizione

Si applica il metodo di scambio di repliche (REM) ad una simulazione MD. Il metodo adottato è lo "Hamiltonian Replica Exchange": le varie repliche, invece di avere temperatura crescente, hanno potenziale scalato di un fattore decrescente. Il risultato è lo stesso, ma così

- si riduce il flusso di dati che vengono scambiati tra i processi
- si lavora sempre a temperatura ambiente: le molecole non sono calde e non c'è bisogno di ridurre il time step, o simili

Nella terminologia del programma ORAC, una "**replica**" $(1, 2, ..., n)$ individua un determinato campo di forze, cioè un valore del fattore di scala del potenziale; ciascuna **traiettoria**, che viene registrata su una directory separata (`PAR0000 ...`), è soggetta via via a scambi di potenziale, cioè di replica. Per ricostruire la statistica della replica 1 (a potenziale pieno) si devono raccogliere tratti di traiettoria della replica nelle varie directory.

I dati aggiuntivi necessari per una simulazione MD-REM sono:

1. numero di repliche
2. fattore di potenziale di ciascuna replica, eventualmente distinto per classi di potenziale
3. l'intervallo di tempo per i tentativi di scambio

9.2 Esecuzione

9.2.1 Schema delle simulazioni

- NVT (con NpT nella replica più "calda" l'acqua bolle)
- 32 repliche
- tempo totale $\sim 10ns$, in tratti di 1 nanosecondo l'uno
- si parte da struttura elongata

9.2.2 lavorare in ambiente parallelo

Si deve usare la versione parallela di `ORAC` (ad es. `orac-p`) e gestire l'esecuzione parallela con `MPI`

1. compilare ORAC nella versione parallela o rendere disponibile una versione già compilata facendone il link
2. fare login su un nodo (`cl11,cl15,cl21`) del cluster a `sm.chim.unifi.it`
3. preparare un file `mpd.hosts` contenente la lista dei nodi da usare, che deve comprendere il nodo su cui si è fatto login . Ad es.:

mpd.hosts

```
nd21
nd22
nd23
nd24
```

4. far partire MPI con il numero di nodi richiesto (N.B. ogni nodo ha più processori; il numero delle repliche viene specificato quando si lancia il programma):

   ```
   #> mpdboot -n 4
   ```

5. preparare l'input (v. sezioni seguenti)

6. lanciare il job parallelo

   Per lanciare il programma con 32 repliche:

   ```
   #> mpiexec -n 32 orac-p < rem-cold.in
   ```

9.2.3 input per partenza fredda

In generale, per eseguire un calcolo REM con ORAC bisogna solo aggiungere all'input il blocco `&REM`:

rem-cold-simple.in

```
&REM
  SETUP 0.75 1
  STEP 100.
  PRINT 100000.
&END
...
```

Nella riga `SETUP` si specifica soltanto il valore minimo del fattore di scala del potenziale (0.75); i fattori sono scalati in modo ottimizzato nell'intevallo $1 - 0.75$

Con valori dell'ultimo parametro $\neq 1$ i fattori sono letti dal file `REM.set` ($= 2$), oppure ricavati da un restart ($= 0$).

Si possono anche specificare diversi fattori di scala per diverse porzioni del potenziale:

- *stretching, bending,* torsioni improprie
- torsioni proprie (diedri) e interazioni 1-4
- potenziale *non bonded*

rem-cold.in (1/3)

```
&REM
  SETUP 1.0 0.1 0.5 1
  STEP 100.
  PRINT 1000.
&END
...
```

In questo esempio si sono applicati fattori di scala minimi di 1.0 (niente scalatura), 0.1, e 0.5, rispettivamente, alle tre porzioni di potenziale. Una scalatura sulle torsioni molto maggiore di quella sul potenziale *non bonded* è efficiente in sistemi di cui si studi la stabilità conformazionale in soluzione.

Il parametro `STEP` indica l'intervallo con cui sono tentati gli scambi (in fs).

Con `PRINT` si sceglie ogni quanto stampare la statistica sui fattori di accettazione.

La configurazione per la partenza fredda si prende dalla traiettoria NpT canonica:

- coordinate (proteina e solvente)

  Si prende ad es. l'ultima istantanea della simulazione dal file PDB della traiettoria e la si pone in un file a sé, ad esempio `XXX-REM-start.pdb`

- dimensioni del box

  Si prendono dall'output della simulazione, facendo attenzione che si riferiscano allo stesso step dell'istantanea

- numero di molecole di solvente

rem-cold.in (2/3)

```
&REM
  ...
&END

&SETUP
   CRYSTAL    29.76 29.76 29.76
   READ_PDB   ../XXX-REM.start.pdb
&END

&SOLUTE
   SCALE_CHARGES 1 1
&END

&SOLVENT
   ADD_UNITS 843
&END
...
```

La simulazione REM con partenza fredda si fa girare per pochi step, quindi si salva un file di *restart* per la successiva partenza calda, `old.rst`. Dato che un run di restart utilizza i file di topologia e di parametri in formato binario, questi vanno salvati adesso in formato binario:

rem-cold.in (3/3)

```
...
&PARAMETERS
   READ_TPG_ASCII ../../lib/amber03.tpg
   READ_PRM_ASCII ../../lib/amber03.prm
   WRITE_TPGPRM_BIN amber03.tpgprm
   JOIN SOLUTE
     gly-h tyr asp asp ala thr lys thr phe gly-o
   END
   JOIN SOLVENT
     spce
   END
&END

&POTENTIAL
...
&END

&INTEGRATOR
...
&END

&SIMULATION
...
#  ISOSTRESS PRESS-EXT 0.1 BARO-MASS 10. # <- NO BAROSTAT
&END

&RUN
   CONTROL                   0
   REJECT                  0.0
   TIME                 1000.0
   PRINT                 100.0
   PROPERTY             1000.0
&END

&INOUT
  ASCII   1000.0  OPEN  XXX-REM.pdb
  RESTART
    write 1000.0 OPEN old.rst
```

```
  END
&END
```

Alla fine del breve run si verificano i fattori di accettazione e gli scambi tra repliche (vedi 9.4.1)

9.2.4 input per partenza calda (restart)

Per proseguire un run precedente, oltre al normale input per un restart (nei blocchi `&RUN` e `&INOUT`) va modificato anche il blocco `&REM` specificando che si tratta di un restart. Ricordare che in una partenza calda il tempo totale deve tenere conto di quello già trascorso fino al restart. Ricordare che nel blocco `&PARAMETERS` va data l'istruzione per leggere parametri e topologia dal file binario creato nella partenza fredda.

rem-warm.in (1/2)

```
 &REM
   SETUP 1.0 0.1 0.5 0
   ...
 &END
 ...

 &PARAMETERS
   READ_TPGPRM_BIN amber03.tpgprm
 &END

&RUN
   CONTROL  1
   REJECT                   0.0
   TIME                  2000.0
   PRINT                  100.0
   PROPERTY              1000.0
 &END

 &INOUT
   RESTART
     read old.rst
     write 10000.0 OPEN new.rst
   END
 &END
```

Conviene comunque salvare un nuovo file di restart (`new.rst`) per potere proseguire da dove si è finito, specie in caso di interruzione accidentale.

Ricordare che nel restart non si deve più né definire il box, né costruire soluto e solvente. Nel blocco `&PARAMETERS` si omettono le istruzioni `JOIN`, mentre i blocchi `&SETUP` e `&SOLVENT` si possono omettere del tutto (il blocco &SOLUTE resta solo se contiene istruzioni speciali tipo `SCALE_CHARGES`)

rem-warm.in (2/2)

```
 ...

 # &SETUP
 #    CRYSTAL    29.76 29.76 29.76
 #    READ_PDB   ../XXX-REM.start.pdb
 # &END
 ...
 # &SOLVENT
 #   ADD_UNITS 843
 # &END
 ...
```

9.3 Come trattare run MD-REM lunghi in ORAC

I run più lunghi di $5ns$ vanno fatti in tratti successivi per evitare un accumulo di errori che in certi casi può addirittura portare ad un'interruzione della simulazione. Ogni run crea un file di *restart* da cui riparte quello successivo. All'inizio si fa un run brevissimo, solo per creare il primo *restart*.

9.3.1 Run di preparazione

Si deve fare un run a partenza fredda (vedi sopra), salvando un file di *restart* con il nome fisso `loop-old.rst`:

loop-start.in

```
...

&RUN
   CONTROL                   0
   REJECT                  0.0
   TIME                 1000.0
   PRINT                 100.0
   PROPERTY             1000.0
&END

&INOUT
  RESTART
    write 1000.0 OPEN loop-old.rst
  END
&END
```

9.3.2 Run di produzione

9.3.2.1 modello di input per un segmento di 1 ns

Nel run di produzione si fa una partenza calda (vedi sopra) dal restart appena creato salvando un nuovo restart (`loop-new.rst`) e un file di traiettoria (`loop-new.xyz`). Il resto dell'input rimane uguale al run di preparazione.

Notare che per risparmiare spazio si salva la traiettoria in formato `xyz` (solo coordinate) invece che `PDB`, e escludendo il solvente (cioè definendo un frammento che comprende solo gli atomi del soluto).

N.B. Attenzione a inserire i corretti indici di atomo del soluto in DEF_FRAGMENT

loop-0.in

```
&REM
   SETUP 1. 0.1 0.5 0
   STEP 100.
   PRINT 100000.
&END

&SOLUTE
   SCALE_CHARGES 1 1
&END

&PARAMETERS
   READ_TPGPRM_BIN amber03.tpgprm
&END

&POTENTIAL
...
&END

&INTEGRATOR
...
&END

&SIMULATION
...
&END

&RUN
   CONTROL                   1
   REJECT                  0.0
   TIME                      0
   PRINT                1000.0
   PROPERTY            10000.0
```

```
&END

&INOUT
  print_xyz_only
  PLOT  FRAGMENT 1000.0   OPEN  loop-new.xyz
  RESTART
    read loop-old.rst
    write 10000.0 OPEN loop-new.rst
  END
&END

&PROPERTIES
  DEF_FRAGMENT 1 142
&END
```

9.3.2.2 ciclo su segmenti di traiettoria

L'input `loop-0.in` così come è stato scritto gira per 0 step. In realtà esso è solo un modello per una semplice procedura (`orac-loop.sh`) che per ogni segmento di traiettoria

- salva i file del segmento precedente, rinominandoli con un numero d'ordine
- rinomina il restart da `new` a `old`
- crea un nuovo input aggiungendo una quantità fissa al `TIME` (tempo di durata del run), specificata con il parametro `-a`

La "ricetta" delle cose da fare per lanciare il ciclo è la seguente:

1. Per prima cosa si deve copiare il file di partenza del loop:

   ```
   > cp loop-0.in loop-old.in
   ```

2. Per eseguire un ciclo del programma che si lancerebbe con `‘‘mpiexec -n 32 orac-p’’`, il comando che si dà è del tipo:

   ```
   > ~signo/bin/orac-loop.sh -f 0 -t 9 -a 1000000 -p "mpiexec -n 32 orac-p"
   ```

   che lancia il run per 10 volte, con indici che vanno da 0 a 9, aggiungendo ogni volta 1 nanosecondo. In questo modo in ogni sottodirectory `PARxxxx` si ottengono una serie di file `REM_DIAGNOSTIC.0000 ...` e `0000.out ...` `0000.pdb ...`

   Per lanciare il ciclo in *background* evitando che alla chiusura della sessione di terminale si interrompa il programma, si usa `nohup`:

   ```
   > rm nohup.log; nohup ~signo/bin/orac-loop.sh -f 0 -t 9 -a 1000000 -p "mpiexec -n
   32 orac-p" &
   ```

9.3.2.3 dettagli sulle procedure di loop

Il testo della procedura shell è il seguente:

loop.sh

```
#! /bin/bash

FROM=0;
TO=1;
PROG=orac;

while getopts "a:f:p:t:" Option
do
  case $Option in
      a     )     ADD_TIME=$OPTARG ;;
      f     )     FROM=$OPTARG ;;
      p     )     PROG="$OPTARG" ;;
      t     )     TO=$OPTARG ;;
```

```
  esac
done

echo "Running program $PROG´, loop will run from $FROM to $TO, in steps of $ADD_TIME"


for ((i=$FROM; i<=$TO; i++)); do
    printf -v l "%04d" $i
    echo $l;
# --   loop-pre-run
      awk -v ADD=$ADD_TIME '/   TIME   /  time=$2+ADD; print "   TIME   ",time;next;print' < loop-old.
# ----


    $PROG < loop-new.in > loop-new.out


# --    loop-post-run $i $l;
    for d in PAR00*;do
        cd $d;
        for f in loop-new.*; do
            cp $f $f/loop-new/$l
            cp $f $f/loop-new/loop-old
        done
        cp REM_DIAGNOSTIC REM_DIAGNOSTIC.$l
        cp out* $l.out
        cd ..;
    done
    for f in loop-new.*; do
        cp $f $f/loop-new/$l
        cp $f $f/loop-new/loop-old
    done
done
```

In sostanza, ad ogni ciclo:

- si prepara un nuovo input
- si lancia orac
- si manipolano i file creati

9.4 Verifica dell'efficienza del REM

Il REM è efficiente se

1. ci sono abbastanza scambi tra le repliche
2. la replica più calda esplora bene lo spazio configurazionale

9.4.1 flusso tra le repliche

È essenziale che ci sia un flusso efficiente tra la replica "più calda" e la replica "più fredda" (a potenziale pieno), che è quella di interesse. Perché questo avvenga si devono verificare due condizioni:

1. Ci deve essere un sufficiente fattore di accettazione di scambi tra repliche contigue (10-20%).
2. Il fattore di accettazione deve essere omogeneo per *tutte* le coppie di repliche; infatti, se anche una sola coppia scambia meno delle altre coppie, si crea un "collo di bottiglia" che riduce il flusso tra la cima e il fondo della serie delle repliche.

Come si verifica:

1. il fattore di accettazione medio per gli scambi tra repliche contigue è stampato nell'output di ORAC per ogni coppia di repliche con la frequenza determinata dal parametro `PRINT` del blocco `&REM`.

2. Verificando il grado di esplorazione delle varie repliche da parte di una traiettoria. Nell'arco della simulazione ogni traiettoria dovrebbe esplorare in misura uguale (e comunque, almeno una volta) tutte le repliche.

   La diffusione delle traiettorie si può vedere facilmente con il comando

   ```
   $ plot PAR00*/REM_DIAGNOSTIC -1,2
   ```

   Per fare una distribuzione delle repliche esplorate da ogni traiettoria:

   ```
   $ for i in {00..31} ; do
   $   awk '{print $2}' PAR00$i/REM_DIAGNOSTIC.000?  |histog +i > 00$i.r.h;
   $ done
   ```

   che si può poi diagrammare con:

   ```
   $ plot 00??.r.h
   ```

9.4.2 campionamento dello spazio configurazionale

In ogni caso l'efficienza di una simulazione REM è data dal grado di esplorazione dello spazio configurazionale del sistema, che si verifica esaminando la distribuzione statistica di alcuni parametri configurazionali significativi (RMSD, distanze interatomiche, angoli di Ramachandran).

La replica "più calda" dovrebbe esplorare in modo tendenzialmente uniforme tutto lo spazio configurazionale accessibile alle condizioni date. Se non è così, bisogna scal(d)are di più -e probabilmente aumentare il numero di repliche.

La replica "più fredda" dovrebbe accedere a tutto lo spazio configurazionale, campionando le varie regioni in modo proporzionale al(logaritmo del)l'energia libera.

Capitolo 10

Analisi dei risultati

10.1 Ricreare il file PDB e ordinare le traiettorie

1. Da un file in formato `.xyz` usando un riferimento si può ricreare la traiettoria in formato `.PDB` con i comandi ‘`xyz2pdb`‘ o ‘`x2p`‘.

   Ad es con il comando:

   ```
   $ x2p -R 1 K7G-template.pdb PAR00*/0000.xyz > r0001_0000-unordered.pdb
   ```

   si convertono tutte le configurazioni relative alla replica 1 nei file `0000.xyz` di tutte le traiettorie, usando il modello `K7G-template.pdb`, e si manda il risultato su `STDOUT`. Notare che le configurazioni che si ottengono non sono in ordine di tempo, ma seguono la sequenza con cui appaiono nelle varie traiettorie `PAR0000`, `PAR0001`, ...

2. Le traiettorie di una replica si possono riordinare nel tempo con i comandi ‘`order.sh`‘ o ‘`order0.pl`‘

   Ad es con il comando:

   ```
   $ order0.pl R=1 r0001_0000-unordered.pdb > r0001_0000.pdb
   ```

   si riordinano nel tempo convertono tutte le configurazioni relative alla replica 1 nei file `0000.xyz` di tutte le traiettorie, usando il modello `K7G-template.pdb`, e si manda il risultato su `STDOUT`

3. I due comandi precedenti si possono dare in successione evitando di creare il file intermedio, non ordinato.

   ```
   $ x2p -R 1 K7G-template.pdb PAR00*/0000.xyz| order0.pl R=1 > r0001_0000.pdb
   ```

4. Si può ricostruire l'andamento delle energie con semplici comandi; ad esempio, per selezionare la replica 1:

   ```
   $ awk '$2==1' PAR00??/REM_DIAGNOSTIC.0000 | sort -nk1 > r0001_0000.ene
   ```

10.2 Energie delle repliche

La mappa degli scambi tra le varie repliche si può diagrammare con

```
$ plot PAR00??/REM_DIAGNOSTIC -1,2
```

L'energia potenziale delle varie *traiettorie* si può diagrammare con

```
$ plot PAR00??/REM_DIAGNOSTIC -1,5
```

L'energia potenziale della *replica 1* si può diagrammare con

```
$ plot r0001_0000.ene -1,5
```

10.3 Parametri configurazionali

10.3.1 Calcolo di parametri configurazionali dalle traiettorie

Per creare file con l'evoluzione temporale di una o più variabili configurazionali, si usa il programma (estendibile) `analysis`. Ad esempio, il seguente input produce l'evoluzione di tre parametri nella traiettoria `r0001_0000.pdb`: la distanza testa-coda, la distanza res4- res7, gli angoli di Ramachandran del residuo 7:

analysis.in

```
  # name of the PDB file to analize:
r0001_0000.pdb
  # head-to-tail K7G:
& atom_distance 5 122
  # K7G, CA 4-7 distance:
& atom_distance 45 81
  # Ramachandran angles of residue 7
& ramachandran 7
```

Il programma si lancia con

```
$ analysis < analysis.in
```

e crea un file per ogni variabile. Questi si possono rinominare opportunamente con comandi tipo

```
$ cp atom_distance.01.out r0001_0000.h2t
$ cp atom_distance.02.out r0001_0000.4t7
$ cp ramachandran.03.out  r0001_0000.ram7
```

Per creare istogrammi di probabilità di una variabile, si usa il comando `histog`. Ad esempio, per creare un istogramma ad area normalizzata con larghezza di canale 0.2:

```
$ histog +.2i r0001_0000.h2t > r0001_0000.h2t.h
```

10.3.2 calcolo di RMSD

Per calcolare gli RMSD rispetto alla struttura sperimentale 1 della chignolina, considerando soltanto i C^{α}_{2-9}:

```
$ rmsd.sh -1 chigno-PDB-CA29.pdb -2 K7G-CA29.pdb chigno-01.pdb r0001_0000.pdb > r0001_0000.rmsd
```