Manuales

Lista de cosas para hacer

- ☐ Arreglar script envío de MD a Slar
- ☐ Añadir pasos para enviar el cálculo a Slar (`sbatch script.sh`. `squeue -u user`...)

Índice general

Capítulo 1

Docking

1.1. Preparación de los ficheros

Para realizar un cálculo *docking*, que permitirá estimar la colocación de un sustrato en una proteína, se deben tener dos ficheros: la proteína y el sustrato.

Para realizar la preparación de la proteína se utilizará el software `UCSF Chimera`. De su entorno gráfico se utilizará principalmente la línia de comandos, que se puede activar en `Favorites > Command Line`, así como también un conjunto de herramientas preinstaladas en el propio `UCSF Chimera`.

1.1.1. La proteína

Para preparar la proteína para un *docking*, es necesario primero limpiarla de solvente e iones, eliminar el substrato con que se ha cristalizado la proteína i, finalmente protonarla, que se puede conseguir mediante dos procedimientos distintos, según se desee que la protonación dependa del pH o no.

1.1.1.1. Limpieza de la proteína

El primer paso para preparar la proteína es limpiarla de solvente, iones y sustrato cristalográfico. Para ello, deben seguirse estos pasos:

1. **Eliminar el solvente:** `sel solvent`; `del sel`. *Advertencia: en caso de tener una agua coordinada a un metal situado en el centro activo, se debe deseleccionar para no eliminarla. Con el comando* `show sel` *se pueden mostrar los elementos seleccionados y ver cual es el que se quiere deseleccionar.*

2. **Eliminar los iones:** `sel ions`; `del sel`. *Advertencia: en caso de tener un ión en el centro activo, se debe deseleccionar para no eliminarlo.*

3. **Eliminar el ligando cristalográfico:**`sel ligand`; `del sel`.

4. **Desprotonar:** En caso que la proteína contenga H y se desee protonar en función del pH, se deben eliminar. `del element.H`

1.1.2. Protonación de la proteína sin depender del pH

Si se desea protonar la proteína sin tener en cuenta los protones, se debe usar la herramienta `Add H`, que se encuentra en `Tools > Sutructure Editing`.

1.1.3. El ligando

El Ligando debe estar libre de solvente y iones. En caso de no poder obtenerlo de una estructura cristalográfica de una proteína, el mismo debe ser obtenido a partir de su estructura canónica y ser optimizado. Para ello, el Sitio web Pub Chem ofrece la estructura canónica (canonical smiles), la cual debe ser copiada y optimizada en cimera. Para esto,en Chimera ir a Tools, Structure Editing, Build Structure. Seleccionar SMILES string y pegar la estructura canónica en SMILES string. Nombrar el residuo y luego Apply, y Close. Posteriormente, ingresar nuevamente a Tools, Structure Editing, Minimize Structure. Seleccionar Minimize y tildar Gasteiger. Al final se muestra la estructura minimizada. Guardar el file con extensión pdb.

1.2. AutoDock Vina [1]

Una vez que se tienen las estructuras del sustrato y del ligando, se puede realizar el autodock.Para tal fin, se deben seguir los siguientes pasos:

1. **Abrir las estructuras de ligando y proteína en extensión pdb.**

2. **Programar el autodock vina.** Para esto ir a Tools, Surface/Bindign Analysis, Autodock vina. En dicho casillero uno debe nombrer el uotput file y seleccionar la estructura del receptor y el ligando. Posteriormente se debe tildar en Resize search volume using button, que posteriormente al tildar sobre la pantalla de las estructuras una caja donde se debe ajustar aprentando con el scroll del mouse, sobre cada cara para ajustar a la posición donde uno quiere evaluar el sitio de interacción receptor-ligando. Luego se deben tildar en Receptor options y Ligando optios, las opciones requeridas. En Receptor options las únicas en opción true son Merge charges and remove non-polar hydrogens; Merge charges ando remove ione pairs y Ignore chians of non-standard residues. En Ligand options ambas opciones deben estar en true. En Advanced options indicar: Number of binding modes:10; Exhaustiveness of search: 8; Maximun energy difference: 3.

3. **Efectuar el autodocking** Para esto indicar Apply y esperar que los resultados. Para evaluar el tiempo de procesamiento, ir a Tools, General Controls, Task Panel. Cuando termina el cálculo, solo en chimera se muestran los resultados.

4. **Resultados del autodocking** Los resultados son presentados en la ventana View Dock. Muestra los scores y los RMSD correspondientes. Los 10 clusters se muestran en el archivo con extensión pdbqt.

[1] DOI: 10.1002/jcc.21334

Capítulo 2

Dinámica Molecular (MD) con el *software AMBER* 16[1]

[1] AMBER Software

2.1. Preparación de los ficheros

Para poder realizar una dinámica se deben preparar, al menos, la proteína a estudiar. Esta proteína debe contener el ligando si lo que se quiere estudiar es como se coloca ésta en el centro activo. Para ello debemos conocer primero cómo se coloca realizando, por ejemplo, cálculos de *docking* (sección **??**).

Una vez se conocen la proteína, el ligando y como éste se coloca dentro de la proteína, ya se puede realizar el cálculo de dinámica. Para ello deberemos parametrizar el sistema y, a su vez, conocer las coordenadas. Lo haremos mediante la construcción y preparación de los archivos necesarios gracias a los paquetes *pdb4amber* y *leap (tleap)*, que forman parte de *AmberTools16*.

Para trabajar con *AmberTools16* podemos o bien trabajar en local o bien en *Slar*. Si trabajamos en local bastará con llamar al programa desde el terminal, mientras que si trabajamos en *Slar* deberemos cargar el módulo previamente mediante el comando `module load interactive/cuda/8.0/amber/16`.

2.1.1. Preparación del *pdb* de la proteína

Para que un *pdb* sea convenientemente entendido por *leap* y, por extensión, por *AMBER*, se debe formatar corrrectamente. Para simplificar este formatado usaremos el software *pdb4amber*.

2.1.1.1. Preparación previa del *pdb*

En caso que no se haya realizado un cálculo de *docking* y, por tanto, se haya eliminado todo lo que no es proteína en sí, se tiene que hacer como paso previo al formatado. Para ello se pueden seguir los pasos explicados en la sección **??**. Esta misma preparación se puede llevar a cabo en algunos casos con el propio *pdb4amber*, pero no en sistemas con aguas activas.

En el caso del estudio del mecanismo de la pig12LOX, la proteína se protonó mediante *PropKa* con el *force field AMBER*, así como se conservó únicamente el agua coordinada al Fe^{2+}, de modo que no se requería de ninguna preparación del *pdb*.

2.1.1.2. Estado de protonación de la proteína

Hay algunos residuos que, por su naturaleza química, pueden ser protonados o desprotonados según el pH del medio en el que se encuentren. Una estimación de su estado se puede calcular utilizando un *software* como *PropKa*. Una vez se conocen los estados de protonación, y para que *leap* y *AMBER* los interprete correctamente, se deben renombrar aquellos residuos que tienen un estado protonado distinto al estándard según el convenio especificado en el manual del *software* (Tabla **??**).

2.1.1.3. Formatado del *pdb*

Una vez tenemos la proteína limpia de solvente e iones no activos, ya podemos formatar el *pdb*. Para ello se ha de ejecutar el programa *pdb4amber*, introduciendo los archivos de entrada y salida, así como también las opciones necesarias:

```
pdb4amber -i nombre-archivo-in.pdb -o nombre-archivo-out.pdb [opciones]
```

Cuadro 2.1: Convenio de nomenclatura de los residuos protonables en AMBER

Residuo	**Nombre estándard**	**Nombre según el convenio**
Histidina-δ	HIS	HID
Histidina-ϵ	HIS	HIE o HIS
Histidina-δ, ϵ	HIS	HIP
Asparagina protonada	ASP	ASH
Glutamina protonada	GLU	GLH
Lisina desprotonada	LYS	LYN

Algunas de las opciones son:

- `-d`; `--dry`: Elimina todo el solvente si no se ha eliminado antes. *Advertencia: si hay aguas en el centro activo que son cruciales para la reactividad, éstas también serán eliminadas.*
- `-p`; `--prot`: eliminará todo aquello que no es proteína, como iones o solvente, en el fichero de salida. Por defecto, aunque no esté activada la opción, guarda un archivo llamado *nonprot* en el que se incluye todo lo que no es proteína. *Es recomendable activar esta opción si en la posterior parametrización se tiene que parametrizar externamente zonas del centro activo que no son proteína, como iones o aguas*[2].
- `-y`; `--nohyd`: elimina todos los hidrógenos presentes en el *pdb* para añadirlos posteriormente mediante *leap*. Gracias a esta opción se evitan problemas con los tipos de H y sus nomenclaturas en *tleap*.
- `-h`; `--help`: ayuda dentro del programa, informará de las distintas opciones que se pueden usar.

En el caso del estudio de la pig12LOX, la única opción activada es `-p`, ya que no se requería eliminar los hidrógenos introducidos por *PropKa*, gracias a que el *force field* utilizado para ello fue *AMBER*, y el solvente no necesario.

2.1.2. Preparación del centro activo

En el caso en el que el centro activo de la proteína contenga átomos que no son comunes en las proteínas y que, por tanto, no están parametrizados en el *force field* de *AMBER*, se tiene que generar los parámetros de esa zona de la proteína con el *software MCPB*. Tanto para obtener nuevos parámetros como para obtener posteriormente el fichero de topología y parámetros, el `.prmtop`, se debe generar un `.mol2` para cada residuo del centro activo.

2.1.2.1. Creación de los ficheros para cada residuo

Para generar los ficheros de cada residuo del centro activo, que deben ser `.mol2`, ya que se requieren las cargas, se puede usar un software como `UCSF Chimera`, con el cual se elimina todo lo que no sea residuo, para luego guardarlo. Esto debe hacerse para cada uno de los residuos del centro activo.

[2]En el caso de las LOX o COX, dado que incluyen Fe^{2+} y OH^- en el centro activo es conveniente activarla ya que el *force field* no incluye parámetros para estos grupos químicos y, en consecuencia, el centro activo se tiene que parametrizar externamente

En caso de calcular los parámetros con *MCPB*, se obtendrán los `.mol2` de cada residuo con la carga correspondiente.

En el caso del estudio de la pig12LOX, dado que los parámetros del centro activo ya estaban creados anteriormente, se guardaron los `.mol2` de cada residuo para luego copiar las cargas correspondientes en la última columna del fichero. No se pudieron aprovechar los `.mol2` creados para crear los parámetros dado que éstos estaban referenciados a un centro de coordenadas distinto.

Una vez obtenidos los `.mol2` finales, se tiene que cambiar la penúltima columna, donde aparece el nombre del residuo, por el nombre que tendrá dentro del *script* para generar el `.prmtop`, como veremos más adelante. La última sección del archivo debe contener el tipo de residuo que es y no el nombre que se le dará en el *script* según el convenio de *AMBER* (tabla **??**).

2.1.2.2. Parametrización del centro activo

ñqhbksdgbdasvaòsdgvbas

2.1.3. Creación del fichero de parámetros y topología y de las coordenadas de entrada y solvatación y neutralización del sistema

Una vez creadas las estructuras de cada residuo de forma independiente y con las cargas y nombres correspondientes, así como los parámetros para cada uno de ellos (fichero `.frcmod`), se tiene que modificar el `.pdb` de la proteína completa (con el centro activo y el ligando) y crear el *script*.

Para crear el fichero de parámetros y topología, así como el de las coordenadas iniciales, se usará el *software LEaP* en su versión *prompt* (*tLEaP*).

Además, *LEaP* permite solvatar y neutralizar el sistema en el mismo *script*, de modo que se pueden crear los parámetros y la topologia del sistema tanto para la proteína aislada como para la proteína solvatada.

2.1.3.1. Modificación del `.pdb`

Para que *LEaP* sepa donde se encuentra cada residuo del centro activo en la secuencia peptídica de la proteína, se tiene que tener un `.pdb` que incluya la proteína, el metal si existe, el agua reactiva si existe y el ligando con sus correspondientes números de residuo y de átomo. Una vez añadidos, se tiene que cambiar los nombres de los residuos por los nombres que éstos tendrán en el *script*, es decir, por los mismos que se han cambiado en los `.mol2` del centro activo. De éste modo, *LEaP* será capaz de reconocer qué residuo de la proteína corresponde con qué `.mol2` y, por tanto, sabrá en que átomos deberá añadir los parámetros calculados con *MCPB*.

2.1.3.2. Creación del *script* de *tLEaP*

El *script* necesario para crear las coordenadas iniciales y los paramétros y topología es una simple lista de comandos que se le tienen que dar a *tLEaP*, de modo que puede no crearse y, en su lugar, introducirlos manualmente. Este archivo es un archivo de texto plano con la extensión deseada, aunque es común usar la extensión `.in`.

Un ejemplo de *script*, el utilitzado para el estudio de la pig12LOX con DHA, es el fichero siguiente (archivo):

```
source leaprc.protein.ff14SB
source leaprc.gaff
```

```
addAtomTypes {
{ "M1" "Fe" "sp3" }
{ "Y1" "N" "sp3" }
{ "Y2" "N" "sp3" }
{ "Y3" "N" "sp3" }
{ "Y4" "N" "sp3" }
{ "Y5" "O" "sp3" }
{ "Y6" "O" "sp3" }
}
addAtomTypes{
{ "o" "O" "sp3" }
{ "c2" "C" "sp2" }
{ "c3" "C" "sp3" }
{ "ha" "H" "sp3" }
{ "hb" "H" "sp3" }
}
HD1 = loadmol2 HID250.mol2
HD2 = loadmol2 HID255.mol2
HD3 = loadmol2 HID430.mol2
HE1 = loadmol2 HIE434.mol2
IE1 = loadmol2 ILE552.mol2
OH1 = loadmol2 OH.mol2
FE2 = loadmol2 FE2.mol2
DHA = loadmol2 DHA.mol2
loadAmberParams OH.frcmod
loadAmberParams frcmod.ionsjc_tip3p
loadAmberParams centro_activo.frcmod
loadAmberParams ligando.frcmod
prot = loadPdb proteina_completa.pdb
bond prot.250.NE2 prot.554.FE
bond prot.255.NE2 prot.554.FE
bond prot.430.NE2 prot.554.FE
bond prot.434.ND1 prot.554.FE
bond prot.552.OXT prot.554.FE
bond prot.553.O1 prot.554.FE
bond prot.249.C prot.250.N
bond prot.250.C prot.251.N
bond prot.254.C prot.255.N
bond prot.255.C prot.256.N
bond prot.429.C prot.430.N
bond prot.430.C prot.431.N
bond prot.433.C prot.434.N
```

```
bond prot.434.C prot.435.N
bond prot.551.C prot.552.N
savePdb prot proteina_seca.pdb
saveAmberParm prot proteina_seca.prmtop proteina_seca.inpcrd
source leaprc.water.tip3p
solvateBox prot TIP3PBOX 10.0
addIons prot Na+ 0
addIons prot Cl- 0
savePdb prot proteina_solvatada.pdb
saveAmberParm prot proteina_solvatada.prmtop proteina_solvatada.inpcrd
quit
```

Como se puede observar en el *script*, aparecen varios tipos de comandos, que se describen a continuación:

- `source` : cargan un *force field* ya integrado en *AmberTools* como el de la proteína o el de las aguas.
- `addAtomTypes` : se utiliza para definir nuevos tipos de átomo y su hibridación, aunque no siempre se obecede. Se requiere definirlos ya que son los tipos de átomos definidos en los `.frcmod` de cada residuo del centro activo. La sintaxis de las definiciones és:

  ```
  addAtomTypes{
      { " nuevo tipo " " elemento " " hibridación " }
  }
  ```

- `loadMol2` : cargan un fichero `.mol2` presente en la carpeta en la que se trabaja. Estan asociados a una variable, que corresponde al nombre que se le ha dado al residuo del centro activo en el correspondiente `.mol2` y en el `.pdb`.
- `loadPdb` : cargan un fichero `.pdb` presente en la carpeta y le asocia una variable ya que se modificará en próximos comandos.
- `loadAmberParams` : carga los ficheros `.frcmod` generados con *MCPB*.
- `bond` : define un enlace entre un átomo de un residuo de la proteína (del `.pdb` cargado) con el indicado, de modo que, dado que los residuos del centro activo estan asociados al `.mol2` del residuo, genera los enlaces entre los residuos presentes en el `.pdb` y los parametrizados externamente (los que provienen del `.mol2`). Su sintaxis es:

  ```
  bond variable_pdb_cargado.numero_primer_atomo.nombre_primer_atomo
    variable_pdb_cargado.numero_segundo_atomo.nombre_segundo_atomo
  ```

- `savePdb` : guarda un `.pdb` con las modificaciones hechas hasta el momento asociadas al `.pdb` cargado a través de la variable.
- `saveAmberParm` : guarda las coordenadas iniciales (`.inpcrd`) y la topología y parámetros (`.prmtop`) con las modificaciones hechas hasta el momento asociadas al `.pdb` cargado asociado a la proteina

- `solvateBox` : añade una caja de aguas a la proteína. Su sintaxis es:

  `solvateBox variable_proteina tipo_aguas distancia_desde_último_átomo`

- `addIons` : añade iones según se especifique. Si se indica un 0, se añadirán iones hasta compensar la carga. Si se indica otro número, en cambio, se añadirán tantos iones como se indique. Su sintaxis es:

  `addIons variable_proteina tipo_ion 0_o_número_específico`

- `quit` : salir del programa.

Una vez corrido el *script* se van a generar tres archivos para la proteína seca y otros tres para la proteína solvatada: el `.pdb`, las coordenadas de entrada (el `.inpcrd`) y la topología y los parámetros (el `.prmtop`).

2.2. Cálculo de la dinámica

Generalmente, una dinámica consta de 5 etapas: la minimización, el calentamiento (o *heating*), la equilibración de la densidad o equilibración NPT, la equilibración NVT y la producción. Cada una de estas etapas, dado que se realizan cálculos distintos excepto en el caso de la equilibración NVT y la producción, requieren de una configuración distinta y, por tanto, de un *script* distinto.

2.2.1. Como enviar el cálculo

El cálculo se puede llevar a cabo de dos formas distintas: se puede correr en local si el ordenador tiene una tarjeta gráfica (GPU) o bien se puede enviar a *Slar*.

2.2.1.1. Los programas de cálculo: `pmemd.cuda` y `sander`

Dentro del paquete AMBER existen dos programas de cálculo: `pmemd.cuda`, que está preparado para cálculos paralelos en GPUs y que es el que se usará en este manual, y `sander`, que está preparado para llevar a cabo los cálculos en CPUs.

El uso de ambos programas es prácticamente idéntico. Se les llama desde el Terminal mediante un comando como el siguiente:

```
pmemd.cuda -O -i input.in -o output.out -p topology.prmtop -c coordinates.inpcrd
   -ref coordinates.inpcrd -r restart.rst -inf information.mdinfo -x dynamics.nc
```

Como se puede observar en el comando, se activan 8 opciones, cada una de ellas correspondiente a un fichero de entrada o de salida:

- `-i input.in`: corresponde al fichero de configuración del cálculo.
- `-o output.out`: corresponde al fichero resumen de salida, donde aparecerán datos macroscópicos del cálculo como la energía cinética, la potencial, la total, la temperatura...
- `-p topology.prmtop`: corresponde al fichero de topología y parámetros generado en la sección **??**.
- `-c coordinates.inpcrd`: corresponde al fichero de coordenadas iniciales generado en la sección **??**. En el caso de no tratarse de un cálculo inicial, es decir, de un cálculo que es la continuación de uno previo, en lugar del `.inpcrd` se tiene que usar el `.rst`, que se genera como resultado de un cálculo.

- `-ref coordinates.inpcrd`: corresponde a un fichero con las coordenadas iniciales del cálculo, tanto si son iniciales absolutas como si provienen de un cálculo anterior, que servirá como referencia a lo largo del cálculo. No es necesario incluir esta opción pero si no se incluye puede dar error por falta de referncias.

- `-r restart.rst`: corresponde a las coordenadas del último *step* de la dinámica. Servirá para poder lanzar dinámicas que se basen en esta o para enlazar fragmentos de la dinámica.

- `-inf information.mdinfo`: corresponde a un fichero resumen con la información sobre el tiempo total de cálculo y otros datos macroscópicos de la dinámica.

- `-x dynamics.nc`: corresponde a los datos de posición y velocidad para cada *frame* de la dinámica. Por defecto se guarda en formato binario para reducir el tamaño del archivo.

Si se tiene que calcular una dinámica relativamente larga es conveniente fraccionar la dinámica en dinámicas de, como máximo, 5 ns. Para ello se genera un único fichero de configuración que calcule 5 ns, por ejemplo, y en el *script* de envío del cálculo se añaden tantos comandos que activen *pmed.cuda* como sean necesarios hasta completar el tiempo de dinámica requerido con las mismas opciones a excepción de las coordenadas y la referencia, que deben cambiarse por el *restart* del cálculo anterior y los ficheros de salida (`.out`, `.rst`, `.mdinfo` y `.nc`), a los cuales se les debe cambiar el nombre para no sobreescribirlos. Partir los cálculos en pequeñas dinámicas puede ser muy útil en el caso en el que uno de ellos de resultados incoherentes o erróneos, ya que se podría aprovechar los cálculos anteriores y reiniciar la dinámica en un punto medio y, dado que es un cálculo estocástico, el cálculo puede seguir evitando estos resultados incoherentes. También se pueden utilizar archivos de configuración (los `.in`) distintos dentro de un mismo tipo de cálculo si se desean especificaciones distintas como pueden ser restricciones de movimiento de partes del sistema.

2.2.1.2. Enviar el cálculo a *Slar*

Si enviamos el cálculo a *Slar* deberemos crear un *script batch* para enviarlo. A continuación se muestra un ejemplo general (archivo):

```
#!/bin/bash -l
#SBATCH --job-name=AMBER
#SBATCH --partition=gorn
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --output=AMBER.o%j
#SBATCH --error=AMBER.e%j
#SBATCH --gres=gpu:1
#SBATCH --mail-type=ALL
#SBATCH --mail-user=usuario@klingon.uab.cat

### MODULES ###
module load gcc/4.8.5/cuda/8.0/amber/16
```

```
### ENVIRONMENT ###
. /QFcomm/environment.bash

### EXECUTION ###
pmemd.cuda -O -i input.in -o output.out -p topology.prmtop -c coordinates.inpcrd
   -ref coordinates.inpcrd -r restart.rst -inf information.mdinfo -x dynamics.nc

### RESULTS ###
cp -rp $SWAP_DIR $SLURM_SUBMIT_DIR/$SLURM_JOB_ID
```

En la primera sección del *script* es donde aparece la información sobre el cálculo: el nombre del cálculo, de los ficheros de error y *output* (no los resultados), el numero de nodos CPU (que será siempre 1 ya que no se calcula en CPU sino que se calcula en GPU) y la dirección de *e-mail* donde se desea recibir los correos con información sobre el cálculo.

En la segunda sección (`### MODULES ###`) es donde se indica qué módulos cargar, es decir, que programas se van a utilizar. En este caso se tiene que cargar *AMBER16* sobre el *software* de paralelización en GPU *CUDA*. Se puede consultar la lista de módulos disponibles en *Slar* mediante el comando `md av` o `module avail`.

En la tercera sección (`### ENVIRONMENT ###`) es donde se indica que la carpeta de trabajo es desde la que se envia el cálculo.

En la cuarta sección (`### EXECUTION ###`) es donde se inicia el programa de cálculo de dinámica. En la última sección (`### RESULTS ###`) se copian los archivos presentes en la carpeta de trabajo a una carpeta llamada como el número de identificación del cálculo dentro de la carpeta desde la que se ha enviado el cálculo.

2.2.1.3. Como enviar el cálculo en local

Si se tiene instalado Amber en local, se puede correr el cálculo de dos formas: o bien se envía cálculo a cálculo o bien se construye un *script* en `BASH` que envíe automáticamente cada uno de los cálculos.

El modo más cómodo si se tienen los cálculos de cada etapa fraccionados es con un *script* sencillo como éste, con en el que se puede enviar una dinámica completa, desde la minimización hasta la producción con todos los pasos, además de informar cuando acaba cada paso y del tiempo que ha necesitado.

Si lo que se desea es simplemente enviar paso por paso se debe utilizar, simplemente, el comando en el terminal

2.2.2. Creación de los archivos de configuración

Los archivos de configuración de los cálculos de dinámica son archivos de texto plano con la extensión `.in`. En ellos pueden aparecer varias secciones según el tipo de cálculo que se lleve a cabo.

2.2.2.1. Cálculo de minimización MM

La primera etapa de un cálculo de dinámica acostumbra a ser una minimización de la energía del sistema. Es usual ya que la estructura a estudiar proviene, normalmente, de una estructura cristalográfica que, además,

se ha solvatado sin tener en cuenta la energía, de modo que el sistema se prevé que esté lejos de un mínimo energético.

Para el cálculo de la dinámica de la pig12LOX con el DHA como ligando se ha llevado a cabo una minimización en tres etapas: en la primera se retenido la proteína (residuos 1 a 555) dejando las aguas libres (archivo), en la segunda se ha retenido el *backbone* y se ha dejado las aguas y las cadenas laterales de los residuos libre (archivo) y, finalmente, en la tercera se ha liberado todo el sistema (archivo). Todas las restricciones se han llevado a cabo con una fuerza de 5 $\frac{kcal}{mol \cdot \bar{A}}$. Para todas las minimizaciones se han realizado 15000 ciclos primero con el método *steepest decent* y luego con el método *conjugate gradient* para obtener resultados más precisos.

2.2.2.2. Cálculo de *heating*

La segunda etapa de un cálculo de dinámica es el *heating*. La estructura de la cual se parte es una estructura minimizada energéticamente teniendo en cuenta únicamente el potencial, de modo que no tiene en cuenta el efecto de la temperatura. Además, el *heating* nos permitirá obtener unas primeras velocidades para cada átomo del sistema, a partir de las cuales se podrá correr las siguientes etapas.

Para el cálculo de la dinámica de la pig12LOX con el DHA como ligando se ha llevado a cabo un *heating* en una sola etapa de 200 *ps* (`.in` *heating*). Se ha calentado hasta 300 *K* en 10 pasos de 30 *K* cada uno, aplicando una restricción al *backbone* de la proteína con una fuerza de 5 $\frac{kcal}{mol \cdot \bar{A}}$.

2.2.2.3. Cálculo de equilibración de la densidad o equilibración NPT

En esta tercera etapa se somete el sistema (cerrado) bajo el control de la presión y la temperatura, de modo que se logre una equilibración de la densidad del solvente (la proteína no se tiene en cuenta) y, por tanto, una representación fiel a la realidad. Además, la equilibración NPT corrige defectos en la caja de aguas como burbujas de vacío o esquinas cortadas.

Para el cálculo de la dinámica de la pig12LOX con el DHA como ligando se han llevado a cabo 5 pasos, siendo los cuatro primeros de 50 *ps* cada uno (`.in` *eq. NPT, pasos 1-4*),y el último de 800 ps (`.in` *eq. NPT, paso 5*), obteniendo un total de 1 *ns* para la etapa NPT. Se ha mantenido la restricción sobre el *backbone*. En este estudio, el *heating* provocó la desaparición de cuatro aristas de la caja que fue solventada en los primeros pasos de esta etapa.

2.2.2.4. Cálculo de la equilibración

La cuarta etapa del cálculo de dinámica es una equilibración en condiciones NVT, es decir, del sistema cerrado a volumen y temperatura constantes. La finalidad de ésta etapa es dejar evolucionar el sistema durante un breve tiempo (un 10 % del tiempo total de la producción, la siguiente etapa) de modo que se logre una estabilización tanto de la proteína como del solvente.

La temperatura se mantiene constante durante el transcurso de la dinámica aplicando un termostato basado en las ecuaciones de la dinámica de Langevin, cuya principal ecuación es la siguiente:

$$m_i a(t) = \frac{dv_i(t)}{dt} = F_i(x,t) - m_i \cdot \gamma(t) \cdot v_i(t) + R_i(t)$$

Donde $\gamma(t)$ es la frecuencia de colisión de las partículas y $R_i(t)$ es la fuerza estocástica. Debido a que la ecuación de Langevin se usa como únicamente termostato, $\gamma(t)$ debe ser pequeño (de 5 ps^{-1} como máximo),

ya que si éste fuese demasiado grande se consideraría el efecto del solvente sobre la proteína por duplicado. El término $R_i(T)$ podrá no incluirse.

Gracias a la ecuación de Langevin se consigue regular la temperatura del sistema añadiendo una fuerza que regula las colisiones entre átomos en el sistema, de modo que la velocidad se mantenga bien repartida entre todo el sistema y, por lo tanto, lo hagan la energía cinética y la temperatura.

Por otro lado, N y V vienen definidos por el número de partículas (átomos o moléculas) y por el volumen de la caja de aguas, respectivamente.

Para el cálculo de la dinámica de la pig12LOX con el DHA como ligando se ha llevado a cabo una equilibración en cinco etapas de 2 *ns* cada una (`.in` *eq. NVT, pasos 1-5*), obteniendo un total de 10 *ns* para la equilibración NVT. Para esta etapa no se ha aplicado ninguna restricción al sistema. Para este cálculo se ha considerado una frecuencia de colisión de 3 ps^{-1}.

2.2.2.5. Cálculo de la producción

La quinta y última etapa de la dinámica es la producción. Físicamente es idéntica a la etapa de equilibración.

Para el cálculo de la dinámica de la pig12LOX con el DHA como ligando se ha llevado a cabo una producción de 100 *ns* dividida en 20 etapas de 5 *ns* cada una (`.in` *producción, pasos 1-20*). Para este cálculo tampoco se ha impuesto ninguna restricción al sistema y se ha considerado una frecuencia de colisión de 3 ps^{-1}.

2.3. Análisis de los resultados

Una vez calculada la dinámica molecular y, por lo tanto, el o los archivos `.nc`, se deben analizar los resultados. Para ello se usará `cpptraj` y `mdout_analysis.pl`, que están incluido en *AmberTools16* y un script en *python*.

2.3.1. `Cpptraj`

`cpptraj` es un programa incluido en el paquete *AmberTools* que permite modificar y trabajar con archivos de topologías, coordenadas y trayectorias, entre otros. Gracias a `cpptraj` podremos, por ejemplo, unificar los distintos ficheros parciales que hemos generado para la etapa de la producción, de modo que podamos trabajar con un único fichero que contenga toda la producción. Además, permite calcular el RMSD a lo largo de la trayectoria, entre otras muchísimas funciones que se explican en el manual de *Amber*.

2.3.1.1. Unificación de las trayectorias parciales

Para unificar todas la trayectorias parciales generadas por `pmemd.cuda` siguiendo la sección **??**, se puede utilitzar un pequeño script para `cpptraj`. Para correr el *script*, se usa el comando siguiente:

```
cpptraj -p topology.prmtop -i unificacion.in
```

Gracias a este *script* se generan dos ficheros: el `5_prod_total.nc` y el `5_prod_total_autoimage.nc`. El primer fichero es simplemente la unión de los 20 pasos calculados en un solo fichero, mientras que el segundo fichero además centra la proteína y redefine la caja de aguas para el nuevo centro. Principalmente trabajaremos con el `5_prod_total_autoimage.nc`, de forma que lo podamos visualizar asegurándonos que la proteína está en el centro de la caja.

2.3.1.2. Cálculo del RMSD