UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE QUÍMICA BIOLÓGICA

Implementación de un Modelo de pH Constante en OpenAWSEM, un Campo de Fuerza de Grano Grueso para Dinámicas Moleculares

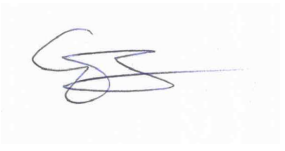
Tesis de Licenciatura en Ciencias Biológicas

Jose Armando Daniel Cuellar Estrada

Director: Dr. Ernesto A. Roman
Co Director: Dr. Esteban Lanzarotti
Lugar de Trabajo: Laboratorio de Ingeniería Enzimática y Nanobiotecnología (QB-81)

Buenos Aires, Marzo de 2025

Jose Cuellar Ernesto Andres Roman Esteban Lanzarroti



Índice general

Resumen

Las proteínas son biopolímeros que pueden plegarse a una estructura tridimensional con función biológica. Como la función está estrechamente ligada a la estructura, se desarrollaron métodos que nos permiten obtener la estructura de una proteína. Sin embargo, las proteínas no son estructuras rígidas y cuando se sintetizan no se encuentran desde un principio perfectamente plegadas. El estudio de la dinámica de una proteína y su plegamiento es tan importante como el de la estructura. Entre los efectos de los que depende la dinámica proteica está el pH, ya que existen aminoácidos básicos y ácidos que se protonan en diferentes rangos de pH. Esta protonación depende del entorno del residuo y de sus propias características fisicoquímicas. Las dinámicas moleculares nos permiten entender mejor cómo una proteína se pliega o cómo es que lleva a cabo su función. A grandes rasgos, existen dos tipos de dinámicas moleculares: las, en la jerga, llamadas “clásicas” que consideran todos los átomos, y las de “grano grueso” que hacen algún tipo de simplificación al campo de fuerza o representación de la estructura. Las dinámicas clásicas, son muy precisas pero debido a la gran cantidad de átomos que tiene una proteína, son computacionalmente costosas. Las dinámicas de grano grueso buscan resolver ese problema, y así permitirnos simular sistemas más grandes y llegar a escalas de tiempo mayores como microsegundos, milisegundos o segundos. En este trabajo incorporaremos un modelo de pH constante dentro del programa OpenAWSEM, un campo de fuerza de grano grueso. El programa está construido bajo la plataforma de OpenMM lo cual permite poder correr las dinámicas optimizadas en GPU y hacer modificaciones fácilmente a los campos de fuerza. Analizaremos curvas de plegado de la proteína CRO a diferentes pHs y realizaremos un muestreo sesgado usando “umbrella sampling” para ver el paisaje de energía en función del pH. Esperamos que el cambio del pH se vea reflejado en las simulaciones. También esperamos que sean acordes a los datos experimentales que tenemos sobre la proteína CRO.

Implementation of a Constant pH Model in OpenAWSEM, a Coarse-Grained Force Field for Molecular Dynamics

Abstract

Proteins are biopolymers that can fold into a three-dimensional structure with biological function. As function has a close relationship with structure, methods have been developed to determine a protein’s structure. However, proteins are not rigid structures and are not perfectly folded from the beginning of synthesis. Studying the dynamics of a protein and its folding is as important as studying its structure. One of the factors affecting protein dynamics is pH, as there are basic and acidic amino acids that become protonated at different pH ranges. This protonation depends on the residue's environment and its own physicochemical properties. Molecular dynamics simulations help us better understand how a protein folds or carries out its function. Broadly speaking, there are two types of molecular dynamics: the so-called "classical" simulations, which consider all atoms, and "coarse-grained" simulations, which introduce simplifications to the force field or structural representation. Classical molecular dynamics are highly accurate, but due to the large number of atoms in a protein, they are computationally expensive. Coarse-grained dynamics resolve this problem, enabling the simulation of larger systems and reaching longer time scales, such as microseconds or milliseconds. In this work, we will incorporate a constant pH model into

OpenAWSEM, a coarse-grained force field. OpenAWSEM is built on the OpenMM platform, enabling optimized GPU-based simulations and easy modifications to force fields. We will analyze folding curves of the CRO protein at different pH values and perform biased sampling using "umbrella sampling" to explore the energy landscape as a function of pH. We expect the pH change to be reflected in the simulations. We also expect them to be consistent with the experimental data we have on the CRO protein.

Introducción

Proteínas

La palabra “proteína” proviene del griego antiguo *proteios* (πρωτεῖος), que significa “primario”, “fundamental” o “preeminente”. También puede estar relacionada con el dios griego *Proteo*, que podía adoptar diversas formas.

Las proteínas son componentes esenciales de todos los organismos y tejidos vivos, están presentes de manera tan universal que es imposible consumir alimentos naturales que no las contengan (Sahyun 1947). Las funciones que cumplen son diversas: estructurales, de transporte, hormonales, enzimáticas, entre otras. Podemos entenderlas como polímeros de aminoácidos que, bajo ciertas condiciones, adoptan una estructura tridimensional que les permite realizar su función biológica. La predicción de su estructura es de suma importancia en el campo de la biología molecular y la bioinformática ya que está estrechamente vinculada a su función. Al predecir la estructura se puede comprender mejor cómo la proteína lleva a cabo sus actividades biológicas, lo que es crucial para entender los procesos celulares y el mecanismo de las enfermedades (Ogunjobi et al. 2024).

Las proteínas no son estructuras rígidas sino dinámicas. El estudio de la dinámica de una proteína es fundamental para comprender cómo lleva a cabo su función y, además, es esencial para entender cómo se pliega correctamente. La dinámica de plegado puede estudiarse como una serie de factores físicos que guían a la proteína hacia su estructura nativa. Estos factores incluyen enlaces de puente de hidrógeno, interacciones de Van der Waals, ángulos de torsión en la cadena principal, interacciones hidrofóbicas e hidrofílicas, interacciones electrostáticas, entre otras (Dill & MacCallum 2012). Esta diversidad de fuerzas físicas son solo un reflejo de la variedad de aminoácidos que tiene una proteína y de las características químicas de cada uno.

Aún con todos los conocimientos físicos y químicos que se conocen de las proteínas, falta un factor clave para comprender estas macromoléculas. Las proteínas son un producto de la evolución. La información de la proteína se encuentra codificada en el ADN, y por esa razón es susceptible a la selección natural. Sin embargo se cree que las proteínas superan incluso la antigüedad del ADN o el ARN. Se ha demostrado que, al replicar un caldo primitivo, las primeras macromoléculas en formarse son los péptidos y proteínas (Milner-White 2019). La evolución de las proteínas es pura termodinámica, y puede ser más antigua qué la selección natural.

Con el pasar de los años han surgido nuevas técnicas para predecir la estructura proteica y también nuevas perspectivas sobre el plegado. La mirada clásica era concebirlo como una serie de pasos discretos con intermediarios hacia la estructura nativa. Actualmente La Teoría del Paisaje Energético ve al plegado como una organización progresiva de un conjunto de estructuras parcialmente plegadas que llevan a la estructura nativa. Según esta teoría, la proteína se desplaza por un paisaje energético con forma de embudo, guiada hacia

conformaciones de mínima energía (Onuchic & Wolynes 2004). Este enfoque resalta cómo la evolución ha moldeado las proteínas para que su paisaje energético favorezca la conformación nativa.

Aminoácidos

Antes de continuar, es importante profundizar en la clasificación de las moléculas que componen a las proteínas. Los aminoácidos, como su nombre lo indica, tienen un grupo amino ($-NH_2$) y un grupo ácido carboxílico ($-COOH$), ambos unidos a un carbono central denominado carbono α. La Figura 1.A muestra la estructura química general de un aminoácido. Las diferencias químicas entre los aminoácidos son debidas a la cadena lateral, representada por la letra R. En la figura 1 se muestran todas las cadenas laterales, donde el atomo marcado de negro es el carbono β, el primer carbono de la cadena lateral.

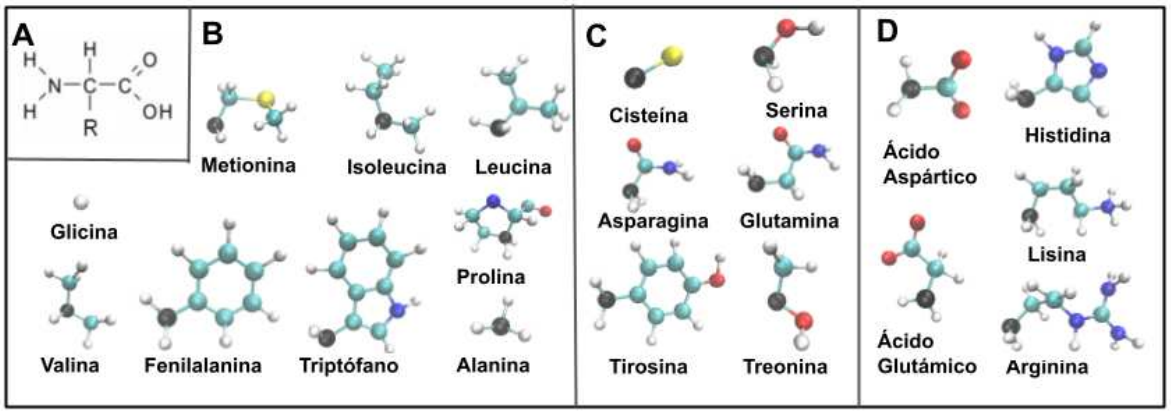


Figura 1: Clasificación de aminoácidos. A. Esquema general de un aminoácido. B. Cadenas laterales de aminoácidos no polares. C. Cadenas laterales de aminoácidos polares. D Cadenas laterales de aminoácidos ácidos y básicos. Los colores son negro para carbono β, *celeste para los demas carbonos, rojo para oxigenos, azul para nitrógeno, blanco para hidrógeno y amarillo para el azufre. Imágenes creadas con el programa VMD*

En general los aminoácidos se clasifican por sus propiedades químicas. En la Figura 1B se pueden ver las cadenas laterales hidrofóbicas o no polares, que tienen baja solubilidad en agua. En la figura 1C vemos su contraparte, las hidrofílicas o polares. Por último, existen cadenas laterales que pueden adquirir una carga neta al protonarse o desprotonarse, y estas se muestran en la Figura 1D. Estas cadenas laterales se dividen en dos categorías: ácidos (glutámico y aspártico, que contienen un grupo carboxilo) y básicos (lisina grupo amino, arginina grupo guanidino e histidina con un grupo imidazol). La cisteína y la tirosina, aunque se clasifican como polares, también pueden adquirir carga neta y comportarse como ácidos en determinadas condiciones.

Organización de la estructura proteica

La información del orden secuencial de los aminoácidos de una proteína es lo que conocemos como **estructura primaria**. Podemos verla como una cadena continua de aminoácidos unidos por enlaces peptídicos, figura 2.A. Si a la cadena de aminoácidos o cadena polipeptídica se le quitan todas las cadenas laterales, nos quedaría lo que se denomina cadena principal. La disposición local de los átomos de la cadena principal es la **estructura secundaria** (Nelson & Cox 2021). Las más comunes son las hélices, hojas beta y los giros. Cuando varias hojas beta están juntas una a lado de la otra pueden formar láminas Beta. Las láminas Beta pueden ser paralelas o antiparalelas como las que se ilustra en la figura 2.B. Los giros suelen estar constituidos por glicinas o prolinas (Nelson & Cox 2021) y debido a su tamaño permiten que la cadena principal sea más flexible en esa zona. Si bien son más comunes en las láminas beta antiparalelas pueden encontrarse también separando hojas beta y hélices, figura 2.B.

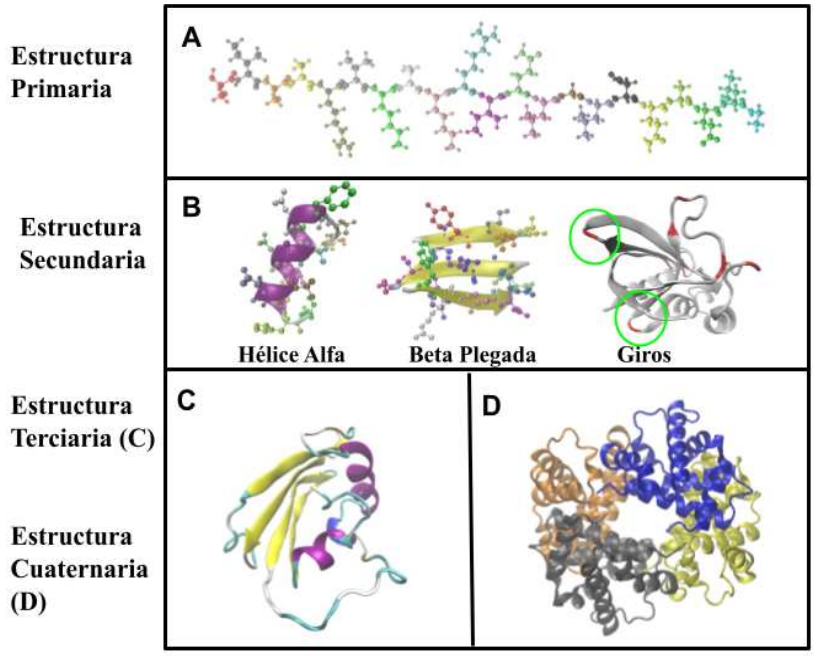


Figura 2: Nivel de organización estructural de proteínas. A) Cadena aminoacídica solo con estructura primaria, cada aminoácido está coloreado por un color único. B) Las tres estructuras secundarias más comunes. Para los Giros se colorearon de rojo las glicinas de una proteína y se marcaron con círculos verdes 2 giros. C) Una proteína completa de una sola subunidad coloreada por estructura secundaria. D) Proteína Globular de 4 subunidades, cada subunidad se representa con un color. Imágenes creadas con el programa VMD.

La **estructura terciaria** es la conformación tridimensional completa de una cadena polipeptídica, figura 2.C. Cuando una proteína tiene más de una subunidad, es decir está formada por más de una cadena polipeptídica, se forma la **estructura cuaternaria** que está unida por interacciones no covalentes y en ocasiones covalentemente por puentes disulfuro, figura 2.D.

El pH y la función proteica

Las proteínas suelen encontrarse en solución, aunque también hay proteínas que se encuentran en la membrana celular. Las propiedades del solvente son fundamentales para la estructura y la función de la proteína. Hay varios parámetros que afectan la estabilidad de una proteína, por ejemplo la temperatura, la presión, la concentración de iones, la polaridad o, en la que nos enfocaremos en este trabajo, el pH. El pH es un factor importante para la estabilidad y dinámica estructural de la proteína y por consiguiente en su función biológica (Talley & Alexov 2010). Se define como:

$$pH = - \log[H^+] \quad (1)$$

Es una escala que va de 1 a 14 y refleja la concentración de protones en unidades de molaridad. Un pH = 1 es una concentración de 10^{-1}M mientras que una de pH=14 es 10^{-14} M. Los cambios en el pH pueden cambiar los estados de carga de los aminoácidos básicos y ácidos, y esos cambios pueden llevar a interacciones entre cargas desfavorables que inestabilizan la estructura o interacciones favorables que la estabilizan.

Las proteínas suelen tener un pH óptimo. Por ejemplo, las proteínas enzimáticas al alejarse del pH óptimo pueden perder gradualmente su actividad o perderla completamente. Podemos mencionar el caso de la pepsina, una proteína digestiva del estómago, donde se ha visto que su pH óptimo es cercano a 2, un pH muy ácido. También se vio que esta pierde actividad a pH mayores a 6. Sin embargo, es posible recobrar su función si se vuelve a bajar el pH (Johnston et al. 2007), es decir, el cambio es reversible. Por otro lado, también se sabe que la actividad catalítica de los sitios activos, está relacionada con los residuos ácidos y básicos. Debido a su capacidad de adquirir o perder un protón tienen el potencial de ser electrófilos (Harris & Turner 2002).

Constante ácida

El estado de protonación o la carga de un aminoácido depende exclusivamente de una propiedad intrínseca, el pKa. Este valor se puede calcular para los grupos amino, carboxilo y la cadena lateral. Cuando un aminoácido se encuentra en una cadena polipeptídica tiene los grupos amino y carboxilo unidos. En estos casos el pKa que se calcula es solo de la cadena lateral. Siendo estrictos se calcula como:

$$pKa = - \log(Ka) = -\log[\frac{H+}{HA}] \quad (2)$$

La Ka, es la constante de disociación entre el estado protonado y no protonado. Simplificando, el pKa es el pH en donde los estados de protonación de una molécula están en igual proporción o equilibrio. En general es posible obtener este valor realizando una titulación. En la figura 3.A se puede ver un ejemplo de una titulación para la obtención del pKa de un aminoácido ácido o básico.

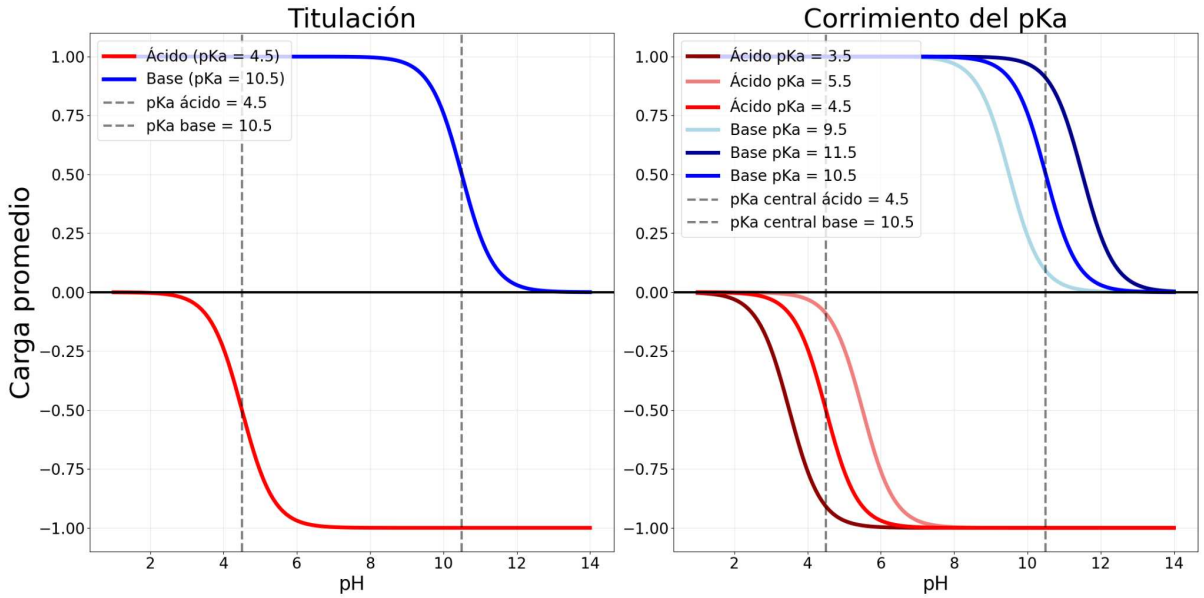


*Figura 3: Frecuencia de carga de un aminoácido ácido o básico a diferentes pHs **A**, Titulación de un aminoácido ácido y básico a diferentes pH, las rectas del eje x corresponden al valor de pKa de esas curvas. **B**, Lo mismo que A pero se agregan dos sigmoideas extras para el ácido y la base, que muestran los dos tipos de corrimiento que pueden ocurrir.*

Se suele usar al pKa como un indicador de acidez para una molécula, cuanto menor es el pKa mayor es la tendencia de ceder un protón al medio. Cada aminoácido ionizable tiene determinado experimentalmente su propio pKa libre en solución, tabla 1. Sin embargo dentro de un entorno proteico este puede variar. El pKa en la proteína está afectado por el entorno en el que ese residuo se encuentra, ya que el entorno define la polaridad, accesibilidad al solvente, interacción con otras cargas, entre otros factores. En la figura 3.B se muestra el corrimiento que ocurre en los residuos ácidos y básicos. Los colores más claros indican que el estado cargado está menos presente y los oscuros lo opuesto.

Aminoácido	pKa libre en solución
ASP	4.0
GLU	4.5
LYS	10.6
ARG	12.0
HIS	6.4
TYR	11
CYS	8.3

N-ter	7.5
C-ter	3.5

Tabla 1: pKa en libre en solución de los diferentes aminoácidos.

Particularmente los extremos de la cadena principal de la proteína también pueden poseer carga neta. Esto se debe a que el grupo amino (N-terminal) del primer aminoácido y el grupo carboxilo (C-terminal) del último no están involucrados en enlaces peptídicos y, por tanto, pueden ionizarse y exhibir comportamiento ácido-base (tabla 1).

Metodos de prediccion para constante ácidas

La predicción del pKa de los aminoácidos en una proteína es útil para entender varios problemas bioquímicos. Los grupos ionizables que presentan grandes corrimientos tienden a estar presentes en los sitios activos. Debido a eso, el cálculo correcto de estos pKa puede ayudar a entender mejor el mecanismo de su actividad. Un método muy utilizado son los métodos empíricos. En estos métodos se dispone de datos experimentales de pKa proteicos y se busca interpretar la razón del corrimiento. El programa PROPKA es un ejemplo de método empírico (Li et al. 2005). Este programa, busca calcular el corrimiento de los pKa a partir de las interacciones electrostáticas, puentes de hidrógeno y un efecto de solvatación. Este modelo tiene en cuenta los puentes de hidrógeno dado que también afectan a la fuerza de interacción de las cargas. Además, entiende al efecto de la solvatación como una forma de contemplar el "enterramiento" de los aminoácidos. El enterramiento es cuán escondido se encuentra un aminoácido del solvente. Es posible medirlo por ejemplo con la Superficie Solvente Accesible o ASA (*Accessible Surface Area* en inglés). Bajos valores de ASA indican enterramiento y valores altos que está expuesto.

El programa PROPKA no solo es muy útil para calcular pKa sino que nos permite explicar en términos químicos como afecta el entorno proteico. Aunque, con la llegada de la inteligencia artificial, han surgido nuevos métodos más precisos para calcular pKa, el programa pKAI es una inteligencia artificial que también tiene en cuenta la identidad química de los átomos (Reis et al. 2021). En este caso, el modelo de inteligencia artificial es entrenado con la información de pKas calculados con otro programa (PypKa). Si bien es difícil comprender cómo "piensa" el programa, se ha demostrado que es mucho más rápido y predice pKas levemente mejor que PROPKA. Sin embargo, ambos programas mencionados dependen de la información de la estructura. El programa pKALM no solo es más rápido que los otros programas, sino que puede calcular el pKa solo con la secuencia proteica (Xu & Onoda 2024). El modelo que desarrolló nuestro laboratorio es equivalente al método PROPKA. Nuestro modelo contempla el enterramiento de los aminoácidos como un término de polaridad del entorno. Además, al igual que PROPKA, modela también las interacciones de cargas. Nuestro modelo si bien no predice los pKa al mismo nivel que los modelos basados en IA, si nos permite usarlo en una dinámica molecular, especialmente para el estudio del plegado proteico.

Dinámicas Moleculares

Por muchos años uno de los problemas principales de la bioinformática estructural fue la obtención de la estructura proteica a partir de solamente la secuencia aminoacídica (Rokde & Kshirsagar 2013, July). Gracias a la explosión de la inteligencia artificial, como AlphaFold (Jumper et al. 2021), contamos con más de 1.000.000 de estructuras predichas a la fecha. Además experimentalmente se ha avanzado bastante también, gracias a las mejoras en las técnicas de cristalografía de rayos x o crio-microscopía electrónica (cryo-EM) (Earl et al. 2017). Hoy en día, la base de Datos Protein Data Bank cuenta con más de 200.000 estructuras proteicas resueltas experimentalmente.

Ahora nos enfrentamos a otro problema y es el estudio de la dinámica de las proteínas. Las proteínas viven como un conjunto dinámico de conformaciones distribuidas en un paisaje de energía libre de acuerdo con su probabilidad de ocurrencia ponderada por Boltzmann (Allison 2020). Es decir, su probabilidad de estar en una conformación con cierta energía depende de la temperatura. En el paisaje de energía libre las estructuras más probables, son las que tienen menor energía, y es donde suele encontrarse la estructura nativa. Para explorar más ampliamente el paisaje de energía nos valemos del estudio por dinámicas moleculares.

Las dinámicas moleculares son simulaciones de un modelo molecular. En el mejor de los casos el modelo es totalmente predictivo lo cual corresponde a un modelo que prediga correctamente el comportamiento físico de un sistema, como por ejemplo el mecanismo de plegado, la estabilidad de una proteína en función de la temperatura, o el pH. Sin embargo, no existen modelos perfectos, por lo que se desarrollan variantes con diferentes fortalezas y áreas de aplicación. El uso exitoso de las simulaciones de dinámica molecular depende de la solución de dos problemas distintos, aunque relacionados: el "problema de muestreo" y el "problema del campo de fuerza" (Bottaro & Lindorff-Larsen 2018).

Existen dos grandes categorías de estrategias de dinámicas moleculares bastante utilizadas: las que consideran todos los átomos, y las simplificadas o coloquialmente llamadas de *Grano Grueso*. Cuando consideramos todos los átomos existen problemas con el muestreo, estas dinámicas son de gran precisión pero con un gran costo de computo. Por otro lado, las dinámicas de Grano Grueso realizan algún tipo de simplificación: desde la cantidad de átomos a los términos tenidos en cuenta en los campos de fuerza para calcular las energías. En este caso podemos aumentar el muestreo llegando a escalas de tiempo mayores como microsegundo, milisegundo o segundos donde ocurren los procesos de plegado. Sin embargo, acá nos encontramos con el problema del campo de fuerza. Es decir la búsqueda de las ecuaciones y los parámetros que mejor representen la dinámica conformacional.

Modelos basados en estructura (Modelos de Go)

En este trabajo nos enfocaremos en simulaciones de plegado y el efecto del pH en las mismas. Usaremos un modelo basado en la estructura. Esto quiere decir que los términos de energía calculados durante la simulación tienen de referencia los valores de la estructura nativa. De esta forma, la energía será mínima en esa conformación y será mayor en cualquier

otra. Se ha demostrado que este tipo de simulación es capaz de reproducir mecanismos de plegado observados experimentalmente. A este tipo de modelos se los suele denominar informalmente como “potencial Go”.

Dentro de la Teoría del Paisaje de Energía, el plegamiento de una proteína tiene un paisaje energético en forma de embudo hacia la estructura nativa. El potencial Go es capaz de simular este escenario a través de favorecer la formación de contactos nativos (Eastwood & Wolynes 2001). Los contactos nativos son la topología de la estructura de la proteína plegada. Si se tiene la estructura resuelta por ejemplo en un archivo PDB, entonces podemos asignar distancias entre los residuos de la proteína. Durante una dinámica estas distancias varían pero el potencial favorecerá al sistema a mantenerse cercano a la estructura de referencia. Si partimos de un estado desplegado, la formación de contactos nativos se verá favorecida y nos dará un paisaje energético en forma de embudo. La ecuación del potencial que determina la energía involucrada en los contactos es la ecuación 3.

$$Potencial_{\mathrm{go}} = -\frac{1}{2}\sum_{i} |E_i|^p \qquad (3.1)$$

Donde

$$E_i = \sum_{j} \epsilon_{ij}(r_{ij}) \qquad (3.2)$$

y

$$\epsilon_{ij}(r_{ij}) = \left|\frac{\epsilon}{a}\right|^{\frac{1}{p}} \Theta(r_c - r_{ij}^N)\gamma_{ij} \exp\left(-\frac{(r_{ij} - r_{ij}^N)^2}{2\sigma_{ij}^2}\right) \qquad (3.3)$$

Ei es la energía asociada a los contactos de cada residuo i de la proteína. El parámetro p es lo que nos permite modular la forma en la que crece la energía con la formación de los contactos. Este parámetro toma valores positivos y cuando vale 1 la energía crece linealmente. A este caso lo llamamos *aditivo*. Cuando p es mayor a 1, Ei crece de forma exponencial y a este caso lo denominamos no-aditivo. Ei se calcula como la sumatoria de todos los contactos nativos locales en función de la distancia, ecuación (3.2). El índice j, al igual que el i, son los residuos de la proteína. La distancia entre ambos residuos es rij y se puede calcular a partir del carbono alfa de cada residuo. Sin embargo también es posible incluir los carbono beta para el cálculo. El potencial que usamos tiene en cuenta los dos carbonos, alfa y beta, de cada residuo. La energía eij de cada interacción se calcula con la ecuación 3.3. Las unidades de energía las da el epsilon y se basa en la estructura nativa (Eastwood & Wolynes 2001), “a” es un parámetro de normalización para que los contactos locales de un residuo tengan valores de 0 a 1. La función escalón depende de ‘rc’ un umbral de corte, cuando rijN es mayor o igual que rc entonces todo se multiplica por cero, de esta

forma se cuentan solo los contactos más cercanos. gamma ij es el valor del contacto y puede tener dos valores, como muestra la ecuación 3.4. el sigma ij representa la anchura del embudo energético, y se calcula con la ecuación 3.5

$$\gamma_{ij} = \begin{cases} 0.125 & \text{si } |i-j| < 5, \\ 0.5 & \text{de lo contrario.} \end{cases} \quad (3.4)$$

$$\sigma_{ij} = |i-j| \cdot 0.15\,\text{Å} \quad (3.5)$$

Ambos parámetros dependen de i-j que es la distancia en secuencia. En el caso de gamma la distancia en secuencia se usa para darle más peso a los contactos más alejados en secuencia.

Por último la ecuación 3.6 es la del parámetro de normalización. La suma de todos los contactos se divide por 8N, donde N es la cantidad de residuos.

$$a = \frac{1}{8N} \sum_i \left| \sum_j \gamma_{ij} \Theta(r_c - r_{ij}^N) \right|^p \quad (3.6)$$

Este modelo basado en estructura es útil para el análisis del mecanismo de plegado pero al requerir de una estructura resuelta, limita su uso. Por eso también simularemos con un campo de fuerza transferible, es decir un campo de fuerza que nos permita no solo estudiar el mecanismo de plegado sino también la predicción de estructura, AWSEM.

OpenAWSEM

El campo de fuerza AWSEM surgió de la familia de potenciales del Hamiltoniano de memoria asociativa (AMH), desarrollada durante muchos años por el profesor Peter Wolynes y su grupo. El potencial AWSEM se basa en una representación de tres átomos por aminoácido. Es decir, es un modelo de grano grueso donde cada aminoácido es representado por su Carbono alfa, Carbono beta y un oxígeno. Además la intensidad de las interacciones de corto y largo alcance están representadas por coeficientes determinados teniendo en cuenta la teoría del Paisaje de Energía y el principio de mínima frustración. Brevemente, a partir de estructuras plegadas y sus correspondientes estados desplegados, se obtienen los valores de los coeficientes que maximizan la diferencia entre el estado nativo y los no nativos, minimizando la rugosidad del embudo de plegado, siendo esta una representación de las trampas cinéticas en la reacción de plegado. AWSEM tiene muchos términos que representan la estereoquímica de la cadena principal, enlaces de puente hidrógeno independientes y cooperativos, interacciones terciarias mediadas por agua y preferencias estructurales locales sesgadas basadas en memorias de fragmentos cortos. El Dr. Aram Davtyan, diseñó la arquitectura del código de AWSEM e implementó sus características principales en C++, para su próxima incorporación en el programa de dinámica molecular LAMMPS (Davtyan et al. 2012). Años después, en el 2020, se implementa el potencial AWSEM dentro de la

plataforma de dinámicas moleculares OpenMM naciendo así OpenAWSEM (Lu et al. 2021). Una de las principales virtudes de este programa, era la posibilidad de correr las simulaciones en placas de video aumentando así la velocidad de las simulaciones hasta 30 veces, comparada con dinámicas de Lammps. En la figura 4 se puede ver una comparación de las velocidades entre OpenMM y Lamps para distintos largos de proteínas.

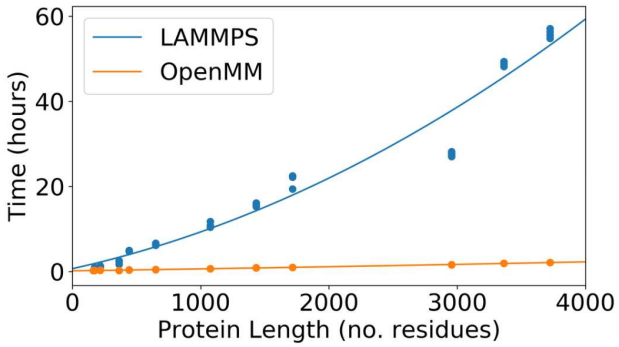


Figura 4: Gráfico del trabajo OpenAWSEM, compara los tiempos de simulación entre Lammps y OpenMM para diferentes largos de proteínas,

Además de este aumento en la velocidad, OpenAWSEM presentaba una plataforma muy interesante y sencilla de simulaciones. La mayoría de los potenciales fueron incorporados mediante las plantillas de fuerzas que proporcionaba OpenMM, permitiendo entonces escribir todo el programa en Python (Eastman 2017).

Objetivo

El objetivo general de este trabajo es aprovechar esta nueva herramienta llamada OpenAWSEM para el estudio de dinámicas de plegado de proteínas y acoplarla al pH. En particular le agregaremos un modelo de pH constante desarrollado en nuestro laboratorio (Tavella et al 2018), para cambiar las cargas durante las simulaciones. Una forma de simular el pH en las dinámicas es cambiar periódicamente la carga de los aminoácidos (Meng & Roitberg 2010).

Objetivos específicos:

- Programar un módulo de python que contenga el modelo de pH constante e implementarlo a OpenAWSEM.
- Simular el plegado de la proteína CRO a diferentes pHs.
- Probar el funcionamiento del potencial Go y el potencial AWSEM

Esperamos que el cambio del pH se vea reflejado en las simulaciones de plegado. También esperamos que sean acordes a los datos experimentales que tenemos sobre la proteína CRO.

Metodología

Modelo de pH constante

El modelo calcula la diferencia de energía ante un cambio en el estado de carga/protonación de un aminoácido ionizable. La diferencia de energía dependerá principalmente del pH elegido. Hay dos casos concretos que podemos imaginar, cuando el residuo está a un pH cercano al pKa y cuando está a un pH lejano al pKa. Cuando estamos en el primer caso el residuo estará en un equilibrio donde ambos estados, protonado y no protonado, tienen igual probabilidad de aparición. En el segundo caso, uno de los estados tendrá una prevalencia frente al otro. La fórmula del modelo de pH constante es la ecuación (4).

$$\Delta E = \Delta q\ (pH - pKa^{ref})\ kb\ T\ ln(10) + \Delta E_{elec} + \Delta E_{pol} \quad (4)$$

La ecuación 4 me permite calcular la diferencia de energía libre entre dos estados, cargado y no cargado. Donde el pKa_{ref} es el pKa en solución del aminoácido (tabla 1) y Δq es la dirección del cambio de estado y se calcula con la ecuación (5).

$$\Delta q = q_{final} - q_{incial} \quad (5)$$

La carga puede ser -1 si es ácido, +1 si es básico o 0 si no está cargado. K_b T ln(10), le da las unidades de energía al modelo. Kb es la constante de Boltzmann y T es la Temperatura en Kelvin. Por último, los términos ΔE_{elec} y ΔE_{pol} son las penalidades por interacciones electrostáticas y entorno polar. Al incluir estos términos de energía podemos representar los corrimientos que se encuentran experimentalmente en los pKa proteicos (Reis et al. 2022).

Penalización por interacción electrostática

Uno de los factores que influyen en el corrimiento del pKa proteico son las interacciones electrostáticas. Un aminoácido cargado puede interaccionar de forma repulsiva o atractiva con los demás aminoácidos cargados. Las interacciones atractivas, cargas opuestas, generan una mayor estabilidad en la proteína. En cambio las repulsivas, cargas iguales, provocan una mayor desestabilidad. Dentro del modelo, se tienen en cuenta todas las interacciones que forma un aminoácido cargado. Sin embargo, la fuerza de cada interacción no será la misma debido a que varían con la distancia. El modelo tiene en cuenta todo esto y agrega un término de energía que contempla las interacciones electrostáticas de a pares, ecuación (6) :

$$\Delta E_{elec} = \Delta q\ K_{elec}\ U_{elec} \quad (6)$$

Como se ve en la ecuación (6), este término tiene también un Δq, un parámetro Kelec para representar la fuerza de las interacciones electrostáticas y un Uelec que es la suma de las interacciones de a pares con todos los otros residuos cargados, ecuación (7).

$$U_{elec} = \Sigma_i \frac{qi}{r} e^{-r/I} \quad (7)$$

$$I = \sqrt{\frac{\varepsilon r\, \varepsilon o\, Kb\, T}{2e\, Na\, I}} \quad (8)$$

Cada interacción se calcula de acuerdo al qi, la carga del vecino ionizable, y r, su distancia en Angstroms (Å). $e^{-r/I}$ es el factor de apantallamiento que depende de la constante l , que en este trabajo se usa como 10 Å, ecuación (8). Este modelo está inspirado en las interacciones electrostáticas de Debye Hückel (Ullner et al. 1996).

Penalización por polaridad

Un residuo en un entorno proteico no solo está en contacto con otros residuos cargados sino que también está interaccionando con residuos polares y no polares. Generalmente los residuos no polares se encuentran más distribuidos en el núcleo de la proteína y los polares más en el lado externo, con contacto con el solvente. Lo mismo suele ocurrir con los residuos ionizables, suelen estar más en el lado externo. Sin embargo, su presencia en el interior de la proteína no es nulo. En el núcleo de la proteína, la mayor presencia de aminoácidos no polares perjudica la presencia de cargas. Esto repercute en un corrimiento en el pKa que desfavorece los estados cargados en el interior de la proteína. Para contemplar este fenómeno se agregó un término de energía que depende de la polaridad del entorno.

$$\Delta E_{pol} = q_{ab}\, \Delta q\, [B_{np}\, U_{np} - B_p\, U_p] \quad (9)$$

Este término de energía tiene nuevamente el Δq pero también tiene un factor q_{ab} , que indica la identidad del residuo ionizable, es decir, si es ácido o básico. Es semejante a la carga pero su valor solo puede ser +1 si es ácido o -1 si es básico. El último factor tiene en cuenta dos contribuciones, los vecinos no polares y polares. Ambas contribuciones se calculan de la misma forma. B es un parámetro que se usa para medir la intensidad de cada contribución. U define el entorno, va de 0 a 1 y se calcula con la ecuación (10).

$$U = exp\,[-\alpha(N - N_{max})^2] \qquad si\ N \leq N_{max} \quad (10.1)$$

$$U = 1 \qquad si\ N > N_{max} \quad (10.2)$$

Cuando N, el número de vecinos polares/no polares es mayor a Nmax (vecinos máximos) el entorno tiene su máxima contribución polar o no polar. N se cuenta en base a un Rmax donde si la distancia entre un residuo es mayor a Rmax ya cuenta cómo un vecino sino su valor empieza a reducirse de acuerdo a la ecuación 11 y la clasificación de polar o no polar se define en la tabla 2. K se usa para contar a todos los aminoácidos de la proteína.

$$N = \Sigma_k C_k \tag{11.1}$$

$$C = 1 \qquad si\ r \leq R_{max} \tag{11.2}$$
$$C = exp\ [-\tau\ (r - R_{max})^2] \qquad si\ r > R_{max} \tag{11.3}$$

Los cambios de carga dentro de la proteína pueden desembocar en cambios estructurales importantes, que son necesarios para la función biológica de la proteína (Isom et al. 2008). Como el modelo tiene en cuenta la distancia de los aminoácidos polares y no polares nos permite modelar esos cambios, dentro y fuera de la proteína.

Tipo	Aminoácido
No polar	ALA - PHE - ILE - LEU - MET - PRO - VAL - TRP
Polar	CYS - ASN - GLN - SER - THR - TYR + Base + Ácido
Base	LYS - ARG - HIS
Ácido	GLU - ASP - CYS -TYR
No contribuye	GLY

Tabla 2 : Clasificación de residuos polares, no polares, bases, ácidos y no contribuyentes.

Parametrización

En el laboratorio de Emil Alexov se publicó una base de datos con más de 600 pKa experimentales de 45 proteínas (Pahari et al. 2019). Cada proteína disponía de su archivo PDB con la estructura resuelta. En el caso de estas proteínas es posible definir una ecuación para el cambio de energía debido al cambio de carga de un aminoácido, en la ecuación (12):

$$\Delta E = \Delta q\ (pH - pKa^{prot})\ kb\ T\ ln(10) \tag{12}$$

Esta ecuación a diferencia de la ecuación (4) no contempla ninguna penalidad y el pKaprot corresponde al pKa experimental de la Base de Datos. Como tanto la ecuación (12) y (4) calculan el mismo ΔE, podemos igualarlas.

$$\Delta q\ (pH - pKa^{prot})\ kb\ T\ ln(10) = \Delta q\ (pH - pKa^{ref})\ kb\ T\ ln(10) + \Delta E_{elec} + \Delta E_{pol} \qquad (13)$$

las 2 ecuaciones dependen tanto de Δq como del pH por lo que podemos independizarnos de estas variables y obtener la ecuación (14)

$$pKa\ prot = pKa\ ref - \frac{Kelec}{kb\ T\ ln(10)}\ Uelec + \frac{qab\ Bnp}{kb\ T\ ln(10)} Unp - \frac{qab\ Bp}{kb\ T\ ln(10)} Up \qquad (14)$$

La ecuación (14) nos permite calcular el pKa prot a partir del pKa ref y las penalidades electrostáticas y de polaridad. Esta es la ecuación que usaremos para la parametrización del modelo. Mediante cuadrados mínimos se buscará los mejores parámetros que me permitan reducir el error entre el valor predicho de la ecuación (14) y el valor experimental.
Para poder realizar esta parametrización fue necesario tomar ciertas aproximaciones. Como la estructura del PDB es estática la posición de todos los residuos es la misma por lo que será posible calcular la Unp y la Up. Respecto de la Uelec será necesario considerar que todos los residuos están cargados. Esta aproximación es correcta para pHs cercanos a 7. La Figura 5 muestra el RMSE que se obtuvo luego del ajuste de los parámetros. El RMSE (Root Mean Squared Error) o Raíz del Error Cuadrático Medio es una métrica de evaluación utilizada en modelos para medir la diferencia entre los valores predichos y los valores experimentales. Se calcula con la ecuación (15) :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad (15)$$

n es la cantidad de observaciones, yi son los pka experimentales e ŷi son los valores predichos. Un menor RMSE implica que el modelo predice mejor que los otros modelos con mayor RMSE

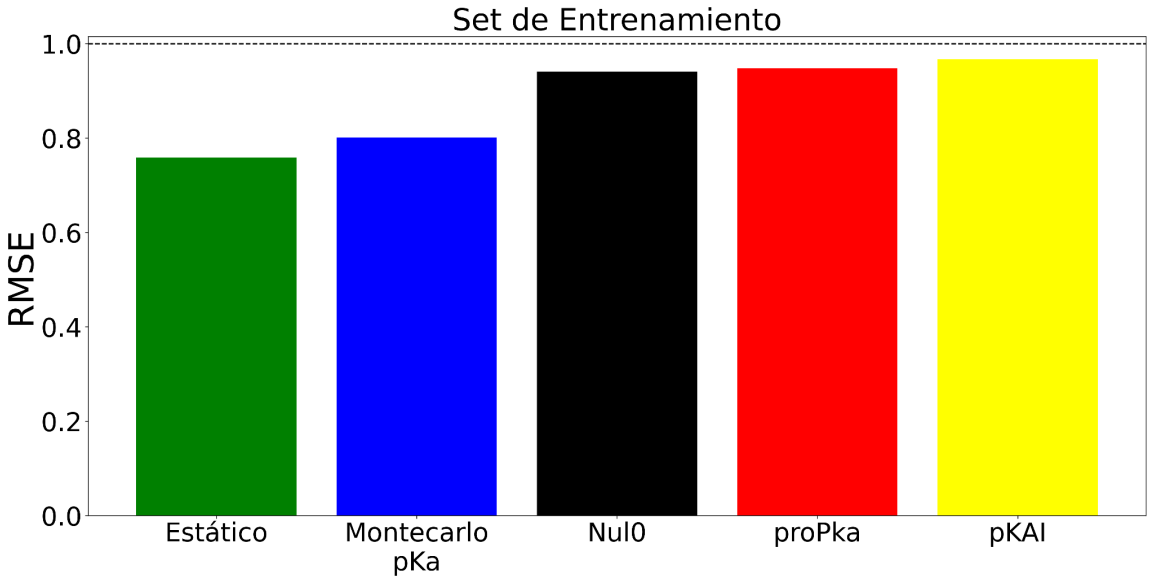


Figura 5 : Cálculo de RMSE de entrenamiento de diferentes modelos con los parámetros optimizados por la base de datos de pKas

No se usaron todos los datos para la parametrización porque entonces nos quedamos sin datos para validar el modelo. Por eso solo se usó el 75% de los datos de los pKa y con el resto se midió nuevamente el RMSE para ver la performance del modelo al predecir el 25% de pKa experimentales, figura 6.

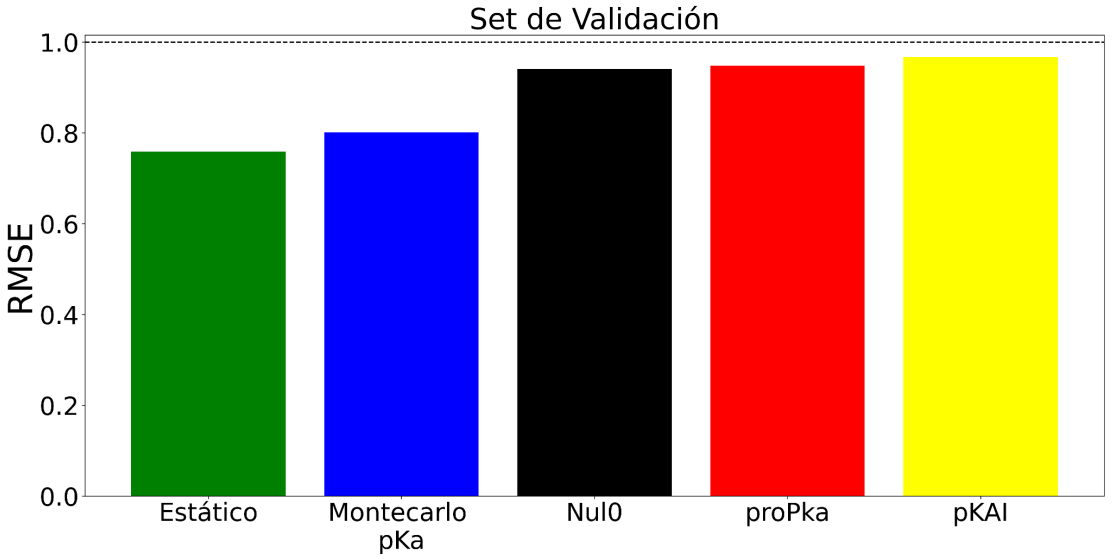


Figura 6: Cálculo de RMSE para la predicción de pka experimentales de diferentes modelos con los parámetros optimizados.

El método Estático y el Montecarlo pKa comparten los mismos parámetros pero en el segundo se usó el algoritmo de Montecarlo para cambiar la carga de los aminoácidos en una simulación y con la frecuencia de carga (*tiempo cargado / tiempo total* ,Tiempo = pasos de simulación) calcular el pKa. El RMSE es mayor para el Montecarlo, seguramente porque el algoritmo es capaz de aceptar cambios no favorables con cierta probabilidad. Los otros

modelos PROPKA y PKAI se explicaron en la introducción y se usaron para comparar sus resultados contra nuestro modelo. Finalmente, el modelo Nulo no contempla ninguna penalidad. Los tres casos tienen un RMSE mayor a nuestro modelo. La parametrización del modelo fue realizada por el Dr. Rodolfo M. Gonzalez-Lebrero. Los parámetros finales, se encuentran en la tabla 1 del apéndice.

Esquema de Trabajo

1) Incorporación del Modelo de pH constante en el Programa OpenAWSEM

Se desarrollará un módulo de python que tendrá incorporado el modelo de pH constante. Se usará el módulo para calcular el pKa de los aminoácidos ionizables de la Frataxina, (PDB ID 4HS5). Una vez realizado se trabajará con la plataforma de OpenMM para la implementación dentro OpenAWSEM.

2) Simulaciones de plegado usando el Potencial Go

Las simulaciones de plegado se harán en dos instancias, una en condiciones aditivas (P = 1 en la ecuación 3) y otra en condiciones no aditivas (P =2 en la ecuación 3).
Todas las simulaciones partirán de un estado desplegado único, que tiene un Q = 0.07 (este valor se calcula con la ecuación 18.1). La curva de enfriamiento irá desde temperaturas altas a bajas durante 5000000 de pasos. Inicialmente se calculará una temperatura de plegado en una simulación sin cargas y con el modelo aditivo. Esto para estimar un rango de temperatura. Luego se relativiza todas las Tm al valor del pH 6, y así comparar con los datos experimentales. La razón es que en las dinámicas moleculares la temperatura varía de acuerdo a cómo se calcula la energía, por ese motivo su valor no corresponde siempre con lo empírico.
Se simularán de pHs 2 a 7 y el módulo MC se llamará cada 1000 pasos para el cambio de cargas. La constante dieléctrica relativa, para la fuerza de la interacción de cargas, se usará en 11. Este valor se eligió para aumentar el valor del potencial de cargas y poder ver efectos más grandes. Para todos los casos se simularán 5 réplicas.
En cuanto al cálculo de la Tm, se ajustará a una ecuación sigmoidea los cambios de Q, que reporta el estado de la proteína, ecuación (16):

$$Q\ global = L / (1 + e^{(k * (T - Tm))}) + b \qquad (16)$$

Se verá cómo varía el Q global respecto de la temperatura T de la simulación. Los parámetros que se ajustan son k, Tm, b y L. La Tm es el punto de inflexión de la sigmoidea, y biológicamente representa la Temperatura a la cual la probabilidad de encontrar la proteína plegada o desplegada es la misma. L es el valor superior al que tiende la sigmoidea, que representa el estado plegado. b es exactamente lo opuesto a L. Finalmente k modula la pendiente de la sigmoidea, es decir que tan abrupto es el cambio.

3) Simulaciones de muestreo sesgado, usando "*Umbrella Sampling*"
Para estudiar el paisaje de energía de la proteína CRO se forzara a la estructura de la proteína a llegar a diferentes valores de Q, desde 0 (desplegada) a 1(plegada), con intervalos de 0.05 .
Se usarán los mismos pH que la sección anterior y la temperatura será la Tm que se calculó

en las curvas de enfriamiento. La constante para el potencial umbrella será de 1000 kcal/mol. Cada simulación tendrá 500000 pasos y cada 1000 se usará el módulo MC.

Una explicación más detallada de este método de muestreo se deja a continuación:

Muestreo sesgado *"Umbrella Sampling"*

El cálculo de diferencias de energía libre es uno de los principales desafíos en biología computacional y bioquímica. El *"Umbrella sampling"*, es una técnica de dinámica molecular sesgada, que permite obtener la energía libre a lo largo de una coordenada de reacción (Kästner 2011). La coordenada de reacción es el cambio de un estado termodinámico a otro (por ejemplo, proteína desplegada a proteína plegada). Para cubrir toda la coordenada de reacción, se suelen usar potenciales sesgados que pueden tener cualquier forma funcional. A menudo, los potenciales armónicos se usan por su simplicidad, ecuación (17).

$$Potencial_{sesgado} = \frac{K}{2}(Q - Q_{ref}) \text{ ** } 2 \qquad (17)$$

Q es la fracción de contactos nativos que tiene la proteína. Q_{ref} es la fracción a la cual queremos muestrear. Y la K es el parámetro de la fuerza potencial. Lo que se consigue al agregar este potencial a la simulación no es anclar el Q al Qref, sino hacerlo oscilar alrededor del Qref. La elección de *K* es la decisión crítica. En general, *K* debe ser lo suficientemente grande como para conducir el sistema sobre la barrera energía, entre ambos estados. Sin embargo un *K* ,demasiado grande, genera distribuciones muy estrechas. Sí la distribución es muy estrecha entonces no podemos muestrear por todo los valores Q y la exploración del espacio conformacional de la proteína no será completa.

A partir de la distribución muestreada del sistema a lo largo de la coordenada de reacción, se puede calcular el cambio en la energía libre. Uno de los métodos que más se usa es el WHAM (*Weighted Histogram Analysis Method*). La figura 7 muestra un esquema de cómo podría estudiarse una coordenada de reacción con la elección de diferentes Qref y con el uso de WHAM.

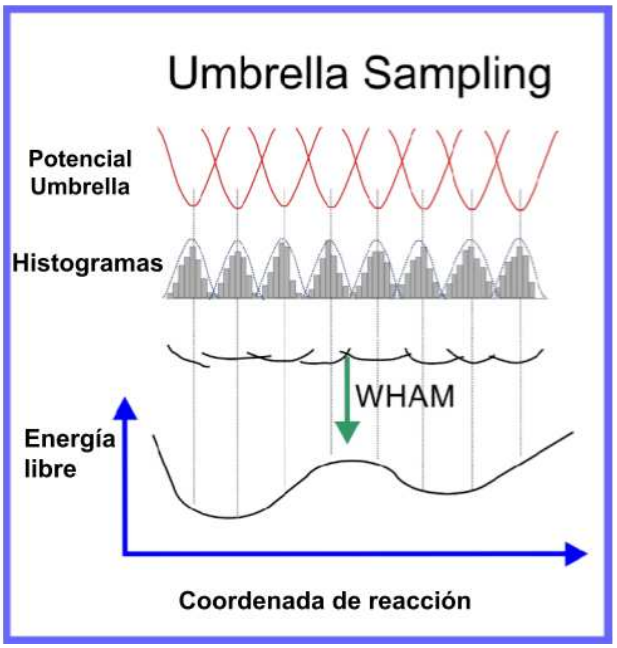


Figura 7 : Esquema de cálculo del Paisaje de energía con "Umbrella Sampling" . Cada histograma se realiza con una simulación con un Qref definido para el potencial sesgado del Umbrella sampling. En la figura no se aprecia pero los histogramas deben estar superpuestos de tal forma que cada valor de Q esté igualmente representado.

4) Análisis de mecanismos de plegado

Para analizar el mecanismo de plegado se calculará los contactos locales de cada residuo de la proteína. La ecuación que se usará para el cálculo de Contactos es la misma que se usa en el programa OpenAWSEM. Ecuación 18 que solo considera los CA:

$$Q\ global = \frac{1}{(N-2)(N-3)} \Sigma\ \ exp\ (\frac{((r-rijN)**2)}{((2*sigma)**2}) \tag{18.1}$$

$$sigma = abs(i-j)^{0.15} \tag{18.2}$$

La ecuación (18) se usa para calcular los contactos globales de toda la proteína. Es posible usar la misma ecuación para calcular los contactos para un solo aminoácidos. Simplemente se debe mantener constante el valor de i deseado. Esto lo denominamos q local, y me da una cuantificación de cuan plegado está un solo aminoácido. Ecuación (19)

$$Q\ local = \frac{1}{(N-2)(N-3)} \Sigma\ \ exp\ (\frac{((r-rijN)**2)}{((2*sigma)**2}) \tag{19}$$
$$i = constante$$

5) Simulaciones de plegado usando el Potencial AWSEM

El procedimiento será el mismo que de las simulaciones de plegado con el potencial Go. Pero quitando el caso sin cargas, porque el potencial AWSEM las considera dentro de su campo de fuerza. Modularemos la fuerza de las interacciones de cargas, usando como punto de partida la misma constante dieléctrica que se usó para el potencial Go, 10

Resultados

Implementación del Modelo de pH constante en el programa OpenAWSEM

1- Módulo de python para la selección del estado de protonación

En programación, un módulo es una sección de código que resuelve un problema específico dentro de un programa. Se desarrolló un módulo que usa el modelo de la ecuación (4) para decidir el cambio del estado de carga de un aminoácido. Para que el módulo sea capaz de decidir se utilizó también el Algoritmo de búsqueda Montecarlo acoplado al criterio de Metropolis-Hastings (Robert & Casella 2004). Básicamente lo que realiza el módulo es, elegir al azar un residuo protonable, cambiar el estado de carga, calcular la diferencia de energía entre el estado previo al cambio y el actual, y aceptar o rechazar el cambio de acuerdo a una probabilidad que depende del ΔE según:

$$p(\Delta E) = e^{-\Delta E/kB*T} \qquad (20)$$

Donde ΔE es la diferencia de energía calculada, k_B la constante de Boltzmann y T la temperatura. Si al obtener un número al azar entre 0 y 1 este es menor al valor calculado con la ecuación (20) entonces se acepta el cambio.
Este proceso se repite, de forma de que el muestreo se encuentre en equilibrio y el estado de carga final sea representativo de la carga a ese pH. En la figura 8, se muestra un esquema simplificado del módulo

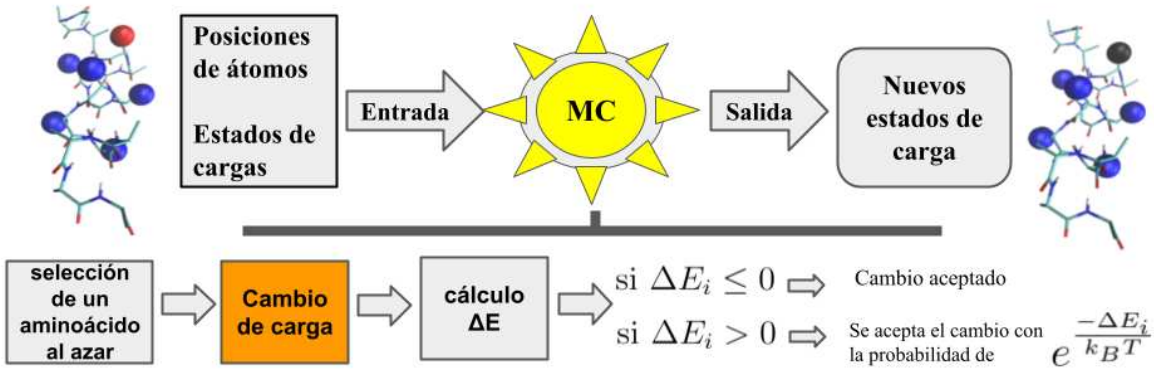


Figura 8: Esquema del Programa MC, las moleculas en el esquema tienen marcados los aminoacidos básicos cargados de azul y los acidos de rojo, cuando pasan a negro simbolizan la perdida de la carga.

Para verificar que el módulo funciona correctamente, comparamos los datos de pKas que calcula este módulo con los obtenidos en la parametrización. En la figura 9 se muestran los pKa de la frataxina calculados con la ecuación (4) y los calculados por el módulo MC. Ambos fueron calculados con la estructura de la frataxina, (PDB ID 4HS5).

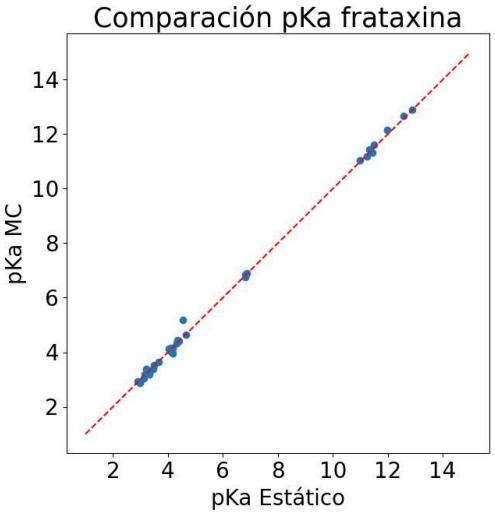


Figura 9: pKa calculado con la versión inicial y pKa calculado por el Módulo MC. La recta roja tiene pendiente 1 y ordenada 0.

Los pKa calculados por ambos programas fueron similares. Se usó la frataxina por ser una proteína con una cantidad de 39 residuos ionizables y por que ya había sido estudiada en nuestro laboratorio. La conclusión de esta sección es que nuestro módulo MC, está listo para ser implementado en OpenAWSEM.

2- Implementación del Módulo MC en OpenAWSEM

Para realizar la implementación dentro de OpenAWSEM fue necesario entender cómo funciona el programa y la plataforma OpenMM. La figura 10 tiene un diagrama de las clases de OpenMM.

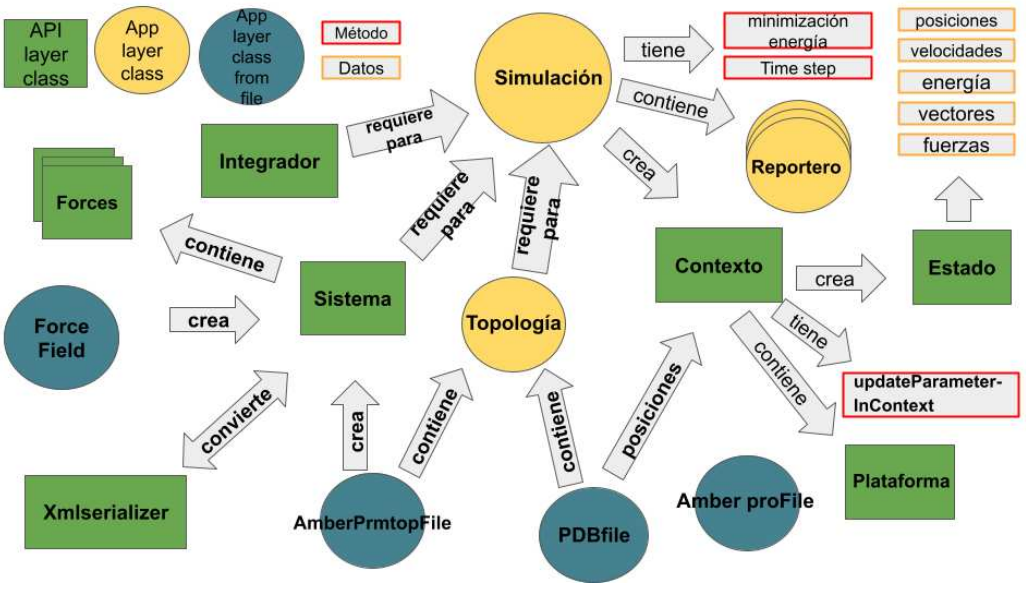


Figura 10: Diagrama de clases de OpenMM.

Una clase en programación se usa para encapsular diferentes módulos que comparten atributos y métodos. Los atributos son variables y los métodos son funciones. El objetivo de una clase es optimizar la creación de objetos. OpenMM me permite crear una simulación, con solo definir la topología, el sistema y el integrador de la simulación. La topología se define con el archivo PDB. El sistema se define mediante el campo de fuerza, que para OpenAWSEM es AWSEM. Y el integrador, para avanzar la simulación, se puede elegir de OpenMM.

La simulación contiene un "contexto", el cual tiene toda la información de cada paso de la simulación. Dentro del contexto podemos acceder a los "estados", donde se puede obtener la información específica de un paso de la simulación. Podemos pensar en el "contexto" como un álbum de fotos, donde cada foto es un estado y el álbum completo el contexto.

La idea general fue usar el módulo MC dentro de la simulación y con la información del estado cambiar el valor de las cargas. Para realizar esto usamos un método que se encontraba dentro del contexto, updateParametersInContext(). Este módulo fue desarrollado específicamente para modificar los parámetros de un potencial durante la simulación sin modificar ni reiniciar la simulación. Actualmente es la forma más eficiente para este propósito.

El módulo MC se usará de forma periódica, por lo que se tendría que interrumpir la simulación y pedirle los datos necesarios, hasta el final de la simulación. En la figura 11 se ve una representación de los pasos de la simulación y como son interrumpidas de forma periódica para usar el módulo de MC

Figura 11: Esquema de los pasos de la simulación interrumpidos por el módulo MC, la frecuencia es cada 3 pasos.

Para verificar que la implementación estaba bien se corrió una simulación, a 0 K, de 1 millón de pasos por pH de 1 a 14 con intervalos de 0.5 en OpenAWSEM y se calculó el pKa de cada aminoácido de la frataxina para compararlo con la estimación del módulo por fuera de OpenAWSEM (figura 12). Se realizó la simulación a 0 K para que la estructura permanezca estática.

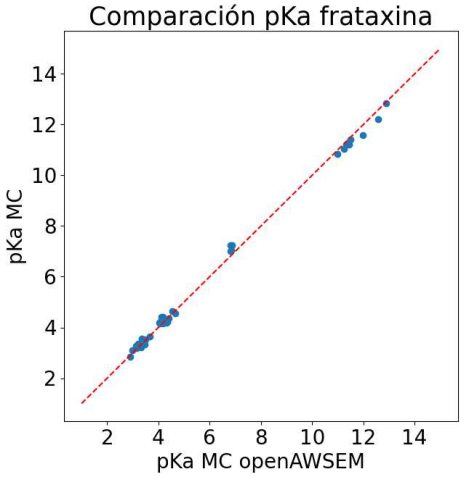


Figura 12: pKa calculado con el Módulo MC dentro de OpenAWSEM y pKa calculado por el Módulo MC. La recta roja tiene pendiente 1 y ordenada 0.

No se vieron grandes diferencias entre los pKas calculados con el módulo MC dentro o fuera del programa OpenAWSEM. Sin embargo se aclara que las distancias entre los átomos en OpenAWSEM están en nanómetros y en el PDB en Angstroms. Esta diferencia implica que los parámetros de distancia del módulo MC tuvieron que ser pasados a nanómetros para ser usados dentro de OpenAWSEM. Teniendo solo eso en cuenta, el módulo MC fue implementado con éxito dentro de OpenAWSEM.

Puesta a punto de las simulaciones de plegado

Como se mencionó en la introducción, si bien OpenAWSEM usa el potencial AWSEM decidimos como una primera instancia correr las simulaciones con el potencial Go. El potencial Go se encuentra correctamente implementado en el programa Lammps. En cambio en OpenAWSEM encontramos dos implementaciones, una donde la formación de contactos nativos minimiza la energía linealmente (Go aditivo) y otra donde la disminución de la energía es exponencial con un parámetro modificable “p” (Go no aditivo). En la figura 13 se graficó la energía de ambos potenciales (p=1 aditivo y p= 2 caso no aditivo) durante unas simulaciones cortas para ver el comportamiento de la energía.

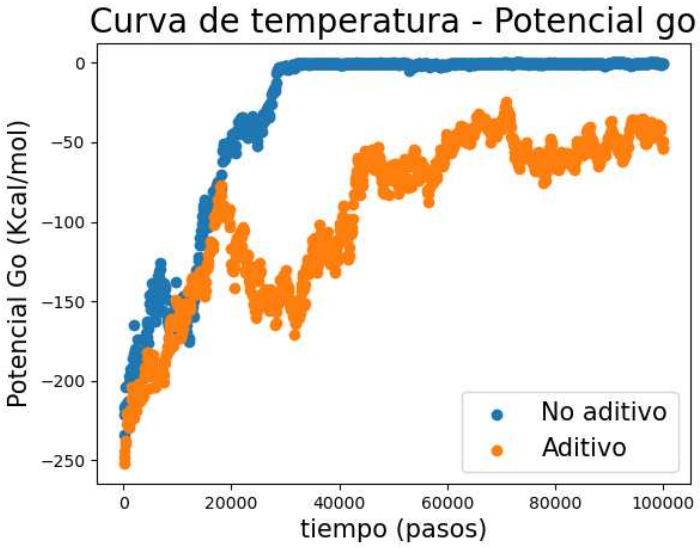


Figura 13: Energía total del potencial Go en Lammps y OpenAWSEM. Simulaciones de aumento de temperatura a partir de la estructura nativa.

Las simulaciones no se hicieron a una temperatura constante, se decidió ir de temperaturas bajas a altas. Al avanzar en la simulación la temperatura aumenta. La energía del potencial Go está basado en la formación de contactos, cuanto más baja, mayor es la cantidad de contactos formados. Si la energía llega a cero no hay o se perdieron los contactos. Vemos en la figura 13 como a medida que vamos avanzando en la simulación la pérdida de contactos es más rápida para el caso no aditivo. Esto pasa porque la energía aumenta de forma exponencial en el modelo no aditivo (p=2) y de forma lineal con el aditivo (p=1).. Esto hace que la pérdida de contactos sea más fácil pero al mismo tiempo que la formación requiera más energía. El modelo de Go no-aditivo permite aumentar la barrera de transición que separa los estados para favorecer la cooperatividad del sistema.

Simulaciones de plegado - Potencial aditivo

Para estudiar los efectos del pH en las dinámicas moleculares, realizamos curvas de plegado usando el potencial Go y el módulo MC, figura 14.A. Elegimos la proteína CRO del fago 434 como caso de estudio porque disponemos de datos experimentales (Padmanabhan et al. 1999). Además CRO es una proteína pequeña de 71 aminoácidos (PDB ID 1zug), por lo que las simulaciones serán rápidas, lo cual permite probar muchas condiciones. Esta proteína modifica su estabilidad siendo menor a pHs bajos y aumentando a pHs neutros. Todas las simulaciones partieron de una misma estructura desplegada con una fracción de contactos nativos de 0.07, figura 15. Se observa que las diferentes curvas tienen un estado desplegado con un Q promedio menor a 0.2. A temperaturas bajas se empiezan a plegar llegando todas las curvas a valores cercanos a 0.8 fracción de contactos nativos.

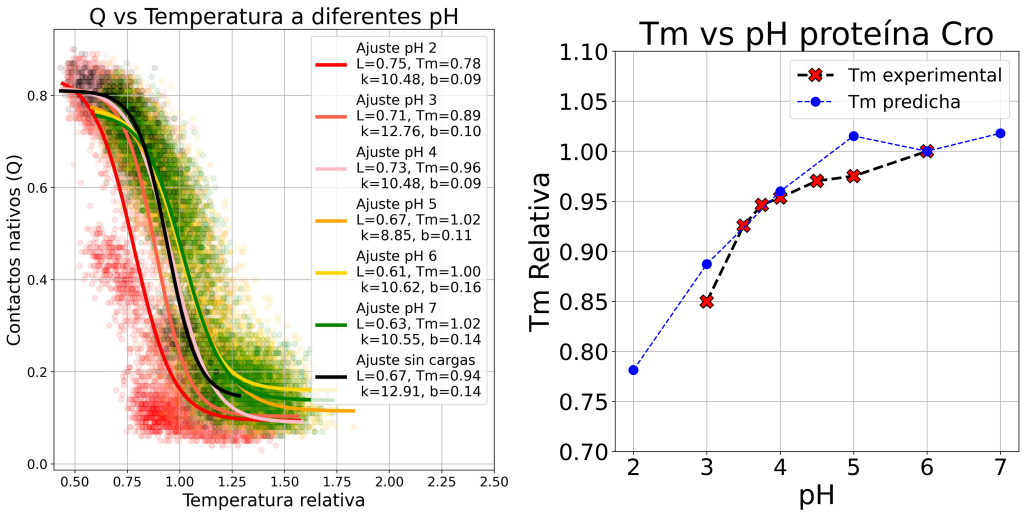


Figura 14: ***A.*** *Curvas de enfriamiento de la Proteína CRO a diferentes pH, 2-7 y sin el potencial de cargas Debye Hückel. Los parámetros de cada sigmoidea se encuentran en las etiquetas* ***B.*** *Temperaturas de plegado predichas vs experimentales para la proteína CRO. Los pKa están relativizados a pH 6*

Calculamos que con un Q ≈ 0.8 la proteína se encuentra plegada, ya que el RMSD promedio de los últimos 500000 pasos de la dinámica era menor 2 Å. Esto también se confirmó viendo las dinámicas con VMD, donde las simulaciones llegaban a formar la estructura secundaria, figura 15. La figura 14.B es la comparación de la temperatura de plegado calculada y los datos experimentales de la proteína CRO del fago 434 [25]. Vemos que las diferencias de temperatura comparten el mismo comportamiento que las experimentales. Se ve que a pH más bajos la temperatura de plegado disminuye. Finalmente la curva negra de la figura 14.A son simulaciones sin cargas que las usamos como control. Su comportamiento es más similar a la curva de pH 4 pero puede ser porque el potencial de cargas estabiliza más la proteína a pHs 5, 6 y 7.

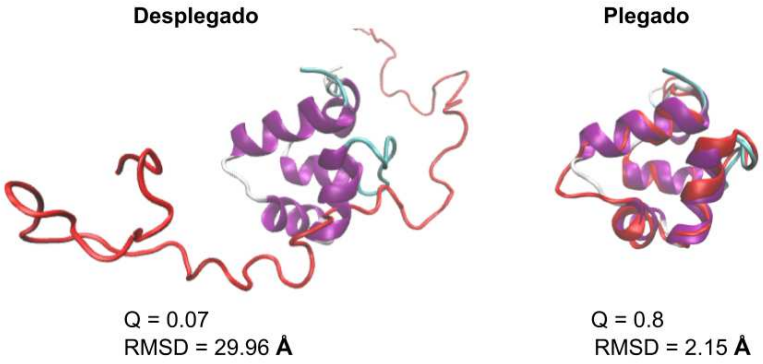


Figura 15: Proteína CRO estado desplegado inicial y un ejemplo de estado plegado. En violeta y celeste se representa la estructura nativa (PDB ID 1ZUG). En rojo se representa el estado desplegado del que parten todas las simulaciones y un estado plegado con 0.8 contactos nativos. Las estructuras se encuentran alineadas por los carbonos alfa y el RMSD es calculado con el programa VMD

Estudio del paisajes de Energía Libre de plegado - potencial aditivo

Para estudiar el Paisaje de Energía libre en relación a la temperatura de plegado, realizamos muestreo sesgado a lo largo de coordenada de reacción . Exploramos todos los valores de Q usando el potencial *sesgado,* ecuación 17.

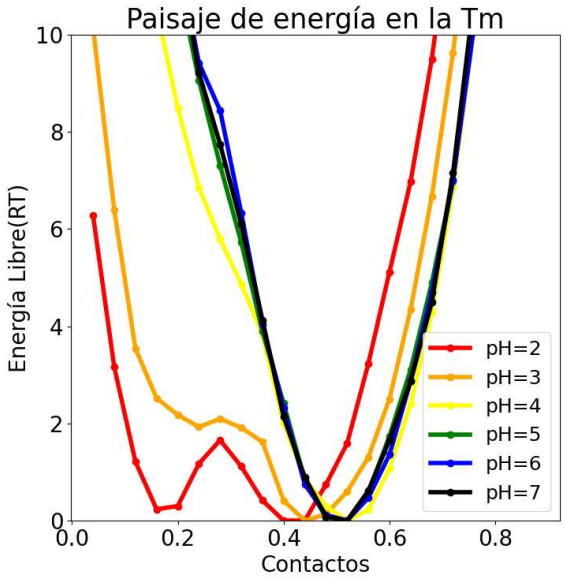


Figura 16 : Paisaje de energía de la proteína CRO a la temperatura de plegado del pH 6, a diferentes pHs. El eje y es energía y está en unidades de kbT y el x son los contactos nativos. Se usó el potencial Go con un P=1

Cuando estamos cerca de la temperatura de plegado de una proteína, esperamos ver la presencia de dos estados de mínima energía. El de menor Q correspondería al estado desplegado y el de mayor Q al estado plegado o nativo. Al usar el potencial Go con un p = 1, contemplamos un caso aditivo para la formación de contactos. Este método nos permite que a bajas temperaturas la proteína pueda plegarse. Sin embargo, como se ve en la figura 16, carece de una barrera de energía. Para los pHs 5 ,6, 7 no observamos la presencia de un estado nativo o desplegado. Lo que sí vemos es que el mínimo de energía se encuentra a valores de Q = 0.5. Este valor no corresponde a un estado de transición, dado que no está entre dos estados termodinámicos como plegado y desplegado. Lo que representa es que la proteína tiene la mitad de sus contactos formados.

Lo que sí podemos observar es la influencia del pH. Si bien para los pH 5, 6, 7 el mínimo de energía es el mismo, los pHs 2 y 3 presentan un estado desplegado. Es más evidente para el pH 2, el efecto del pH muestra una clara desestabilización del potencial Go aditivo. También ahora podemos observar una barrera de energía donde se encontraría un estado de transición a $Q \approx 0.3$.

Esto trae ciertas dificultades ya que al no haber barrera, o ser muy baja, es difícil definir claramente las vías de plegado. Ya que la proteína termina difundiendo por una diversidad de caminos indefinidos. Si bien esto podría observarse experimentalmente, sabemos que para el caso de CRO existe un estado de transición definido ya que el plegado es cooperativo [25].

Por esta razón, en la sección siguiente describiremos los resultados del mismo potencial, pero modulando el parámetro P de no aditividad descrito anteriormente.

Simulaciones de plegado - Potencial no aditivo

Los resultados que obtuvimos con el módulo aditivo fueron alentadores, repetimos las simulaciones de plegado en función de la temperatura pero con un potencial no aditivo observando los resultados de la figura 17.

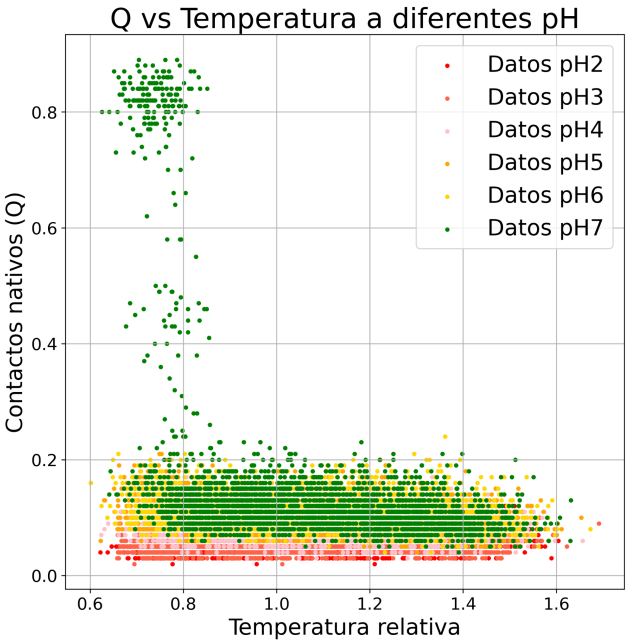


Figura 17: Curvas de enfriamiento de la proteína CRO a diferentes pH, el potencial Go se usó con un P=2 para simular contactos no aditivos.

Cuando usamos el caso no aditivo vemos que en la mayoría de los pHs estudiados la proteína no se pliega. Los valores de Q global, se mantienen bajos para casi todos los pHs, menores a 0.15, figura 17. Cabe recordar que el módulo no aditivo hace que el comportamiento sea no lineal, es decir más cooperativo. De esta forma, si la barrera energética (estado de transición) de plegado resulta artificialmente alta, si partimos de un estado desplegado y enfriamos rápidamente, corremos el riesgo de quedarnos atrapados en ese estado. Sin embargo, podemos ver que a pH 7 la proteína logra plegarse. Esto puede ser un indicio de que la barrera que separa los estados nativos y desplegados varía con el pH siendo más baja a pH 7.

Estudio del Paisajes de Energía Libre - Potencial no aditivo

Aunque no se pudo calcular una temperatura de plegado estudiamos la reacción a la misma temperatura observada en el caso aditivo ya que no esperemos que el cambio de potencial afecte este parámetro. El paisaje de energía para el potencial Go no aditivo, a la temperatura de plegado (Tm) del pH 6, de la figura 14, se ve en la figura 18.

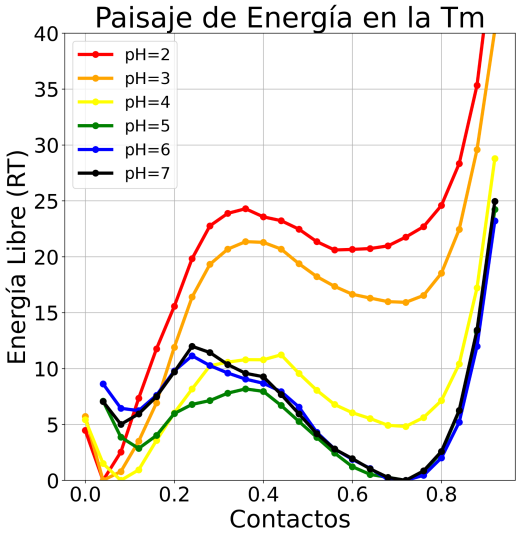


Figura 18 : Paisaje de energía de la proteína CRO a la misma temperatura de la figura 16 y a diferentes pHs. El eje y es energía y está en unidades de kbT y el x es la coordenada de reacción, medida como contactos nativos. Se usó el potencial Go con un p=2.

En el caso del potencial Go no aditivo, logramos ver una barrera de energía que separa dos estados para todos los pHs. El estado plegado se encuentra alrededor de 0.7 contactos nativos y el desplegado cercano a 0.1. Esto permite concluir que el módulo no aditivo representó mejor la barrera de transición y los estados nativos y desplegados. Los valores de energía a Q =0.7 son más altos a pHs bajos, en particular se ve que una mayor predominancia del estado desplegado para los pH 2, 3, 4. Esto se analiza viendo que los mínimos de energía para el desplegado son menores que el estado nativo. Por último los pH 5, 6, 7 varían en el estado desplegado, 0,1. La energía a 0.1 para los pHs 6, 7 es mayor que la de pH 5 y la de estas tres es mayor a la de los pH 4, 3 y 2. Como ningún paisaje de energía tiene precisamente los dos estados de mínima energía a igual energía, concluimos que estamos cerca de la Tm pero no exactamente en ella.

Modulación de la Constante Dieléctrica

Se simuló la constante dieléctrica para cambiar la fuerza de la interacción entre cargas dentro de la simulación. El objetivo era ver como varían los paisajes de energía. La constante dieléctrica es inversamente proporcional a la fuerza electrostática. Valores más grandes de constante dieléctrica disminuye la fuerza electrostática y valores bajos la aumentan. Para ver

mejor los cambios graficamos la diferencia de energía entre los estados plegado y desplegado de la proteína, 0.72 y 0.12 respectivamente, figura 19.

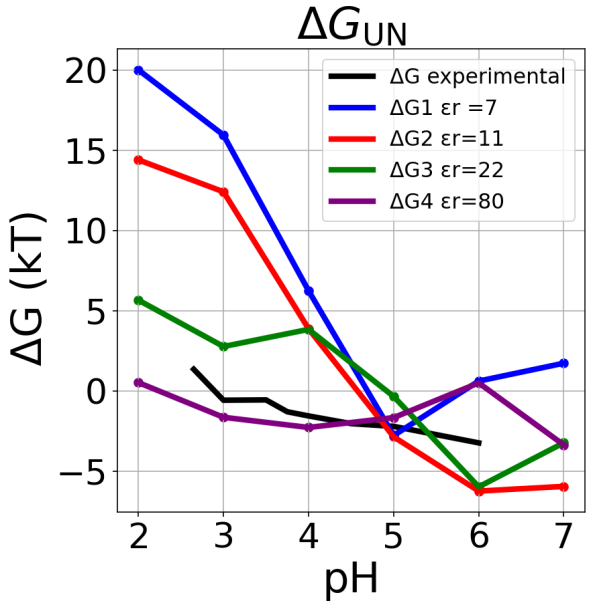


Figura 19: Cálculo ΔG de plegamiento de la proteína CRO a diferentes valores de pH. Los colores representan las diferentes constantes dieléctricas. La diferencia se calcula como la energía libre del estado plegado menos la energía libre del estado desplegado.

El *ΔG* se calculó restando a la energía libre del estado plegado la energía del estado desplegado. Entonces valores más grandes de *ΔG* implican que la proteína se encuentra mayormente representada por el estado desplegado. En la figura 19 vemos que a pHs ácidos, 2-3, el *ΔG* de plegamiento es mayor que a pH neutros 7, para todas las curvas. Con este patrón podemos decir que en el modelo no aditivo también vemos una dependencia con el pH. A pH ácidos la proteína pierde estabilidad. Y la modulación de la constante dieléctrica, nos muestra que sí aumentamos la fuerza de electrostática, la desestabilidad aumenta. Respecto a los datos experimentales, estos se encuentran entre la curva violeta y la curva verde. La curva violeta es la que corresponde a la constante dieléctrica del agua. En principio la constante dieléctrica en un entorno proteico debería ser menor a la constante del agua, principalmente porque suele ser un entorno menos polar. Los datos experimentales nos muestran efectivamente eso, la constante es menor pero no tanto como las que se muestrearon en este trabajo.

Análisis del Mecanismo de Plegamiento

Para un análisis más exhaustivo del mecanismo de plegado es posible ver el plegado local de cada residuo. Usamos los datos de la figura 18 y la ecuación (17) para calcular los Q locales, los resultados se ven en la figura 20.

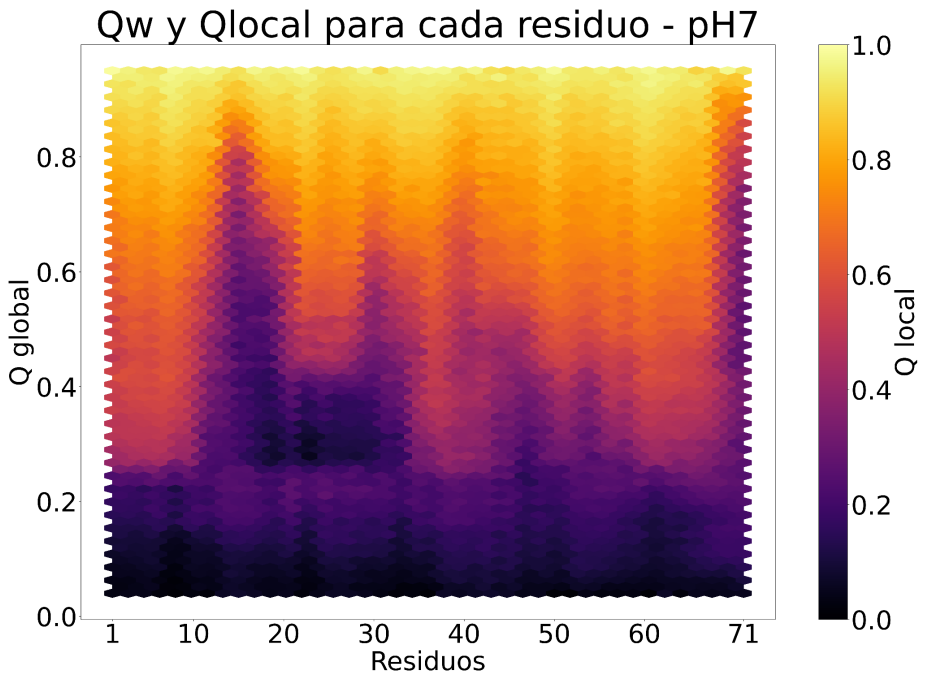


Figura 20: Mapa de calor de Q locales, eje y Q global, eje x residuos. Para un mejor análisis, estos son los aa ionizables. E (6), R (7), K (9), K (10), R (11), R (12), K (16), E (21), K (25), K (29), E (37), K (42), R (43), R (45), E (49), D (57),Y(63) K (66), R (67), K (69).

El estado desplegado está cercano a 0.1, el plegado a 0.7 y el de transición a 0.4 (figura 18). A valores de Q global 0.1 en general se ve que hay pocos o ningún residuo plegado. En el estado de transición, a 0.4 , los residuos de 1 a 10 están más plegados que el resto, pero de 10 a 30 se ve que el plegamiento es menor, valores de q local ≈ 0.1. Esta zona en la proteína tiene una cantidad importante de residuos básicos, que pueden estar interaccionando repulsivamente complicando el plegado. Cuando estamos a 0,7 vemos que las zonas que les falta plegarse son los residuos de 10 a 20 y también los últimos residuos. La cola carboxilo terminal corresponde a los últimos aminoácidos de la proteína, esta también tiene algunos aminoácidos básicos que podrían estar complicando el plegado. Sin embargo los extremos de las proteínas suelen ser zonas laxas, es decir con mucha movilidad.

También se hizo un gráfico de Q global para pH 2 y se lo comparó con el pH 7. La idea era encontrar diferencias en el plegamiento local de residuos, dado que globalmente se veían bastante parecidos, figura 21.

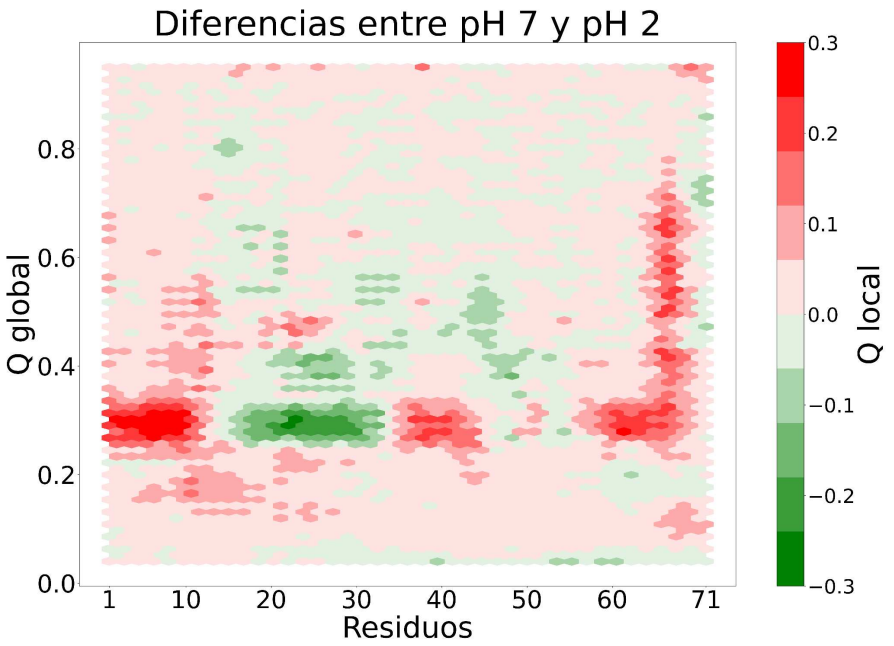


Figura 21: Mapa de calor de los Q locales locales para pH 7 menos Q locales para pH2. El eje "y" es el Q global y el eje "x" los residuos. *Para un mejor análisis, estos son los aa ionizables.* E (6), R (7), K (9), K (10), R (11), R (12), K (16), E (21), K (25), K (29), E (37), K (42), R (43), R (45), E (49), D (57),Y(63) K (66), R (67), K (69).

Los colores más rojizos indican que los Q locales a pH 7 son mayores que a pH 2 y los colores verdes indican lo opuesto, que el pH 2 tiene Q locales mayores. En términos generales, casi a todos los valores de Q global, el pH 7 empieza a plegarse a Q globales más bajos. Esto se ve más claro a Q global 0.1, donde es mayoritariamente rojo claro.
En particular los últimos residuos de la proteína en el estado transitorio, Q global 0.4, se pliegan mucho más rápido a pH 7. Esto también lo vemos en los primeros 10 residuos. Pero entre los residuos 20 y 30 el comportamiento es el opuesto y se pliega antes a pH 2. Analizaremos estas dos zonas, 1 a 10 y de 20 a 30, directamente en la estructura, figura 22 y 23.

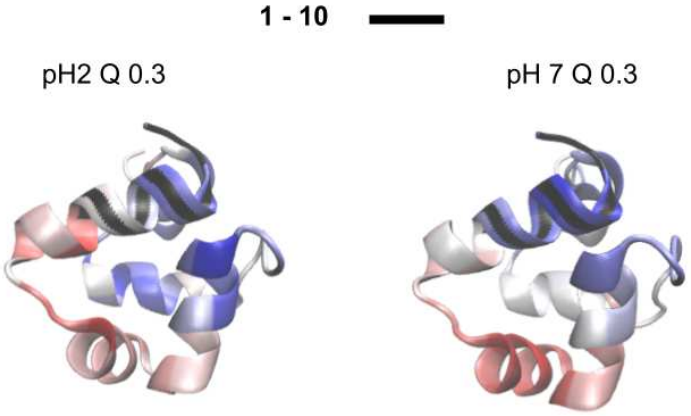


Figura 22: Proteína CRO coloreada por Q local, rojo Q cercano a 0, azul Q cercano a 1. Los valores de Q local son los correspondientes a Q global 0.3. Tambien se encuentra marcada la region de los primeros 10 residuos de la proteina con negro

En ambos pHs los primeros 10 residuos están bastante plegados, pero se ve que el pH 7 tiene un tono azul más fuerte, se encuentra más plegado. También se ve que los residuos que le siguen al residuo 10 están más desplegados a pH 2. Los residuos ionizables que hay en esta zona son E (6), R (7), K (9), K (10). A pH 2 deberíamos tener al glutámico sin carga y a pH 7 cargado. Es probable que este residuo esté disminuyendo las fuerzas repulsivas que generan los otros residuos básicos. La siguiente zona que se analizó fue la de 20 a 30, figura 23.

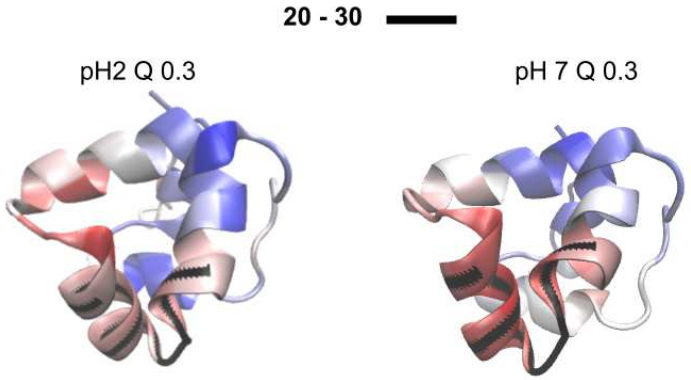


Figura 23: Proteína CRO coloreada por Q local, rojo cercano a 0, azul cercano a 1. Los valores de Q local son los correspondientes a Q global 0.3. Tambien se encuentra marcada la region de los residuos 20 a 30 de la proteina con negro

Esta zona de la proteína a diferencia de los primeros 10 residuos está más desplegada en ambos pHs. A pH 7 tiene un tono rojizo más intenso. La zona también parece tener un loop y tiene la presencia de dos alfa hélices. Los residuos ionizables acá son E (21), K (25), K (29). En este caso la pérdida de carga del glutámico no parece estar ayudando al plegado. El mecanismo debe ser más complejo e involucrar más residuos.

Podemos hacer un análisis similar con los demás Q globales de la proteína, figura 24.

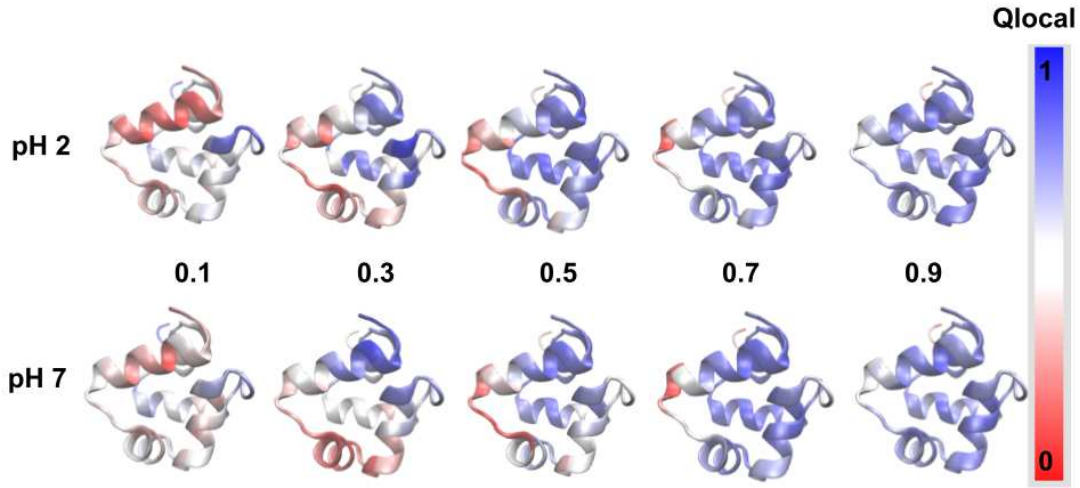


Figura 24: Estructura de la proteína CRO coloreada por el Q local, se usan 5 valores de Q global para pH 7 y pH 2.

En la figura 24 se ve como el plegamiento de algunas estructuras de la proteína, se encuentran más plegadas a pH 7, este comportamiento suele ser igual en todos los casos. Salvo por ciertas partes, por ejemplo la zona central de la proteína se pliega antes a pH 2 que pH 7. Las diferencias se observan más a Q globales bajos, 0.1 y 0.3. En los valores más plegados como 0.7 y 0.9 no observamos diferencias grandes.

Potencial AWSEM

Todas las simulaciones que se corrieron con el potencial Go, se realizaron usando la estructura resuelta de la proteína CRO. El potencial AWSEM, en principio no requiere, aunque puede usarla, la estructura resuelta en un archivo PDB. Este potencial es transferible y puede usarse para la predicción de estructura de una proteína. En la figura 25 mostramos las Tms que se calcularon con este potencial a diferentes pHs.

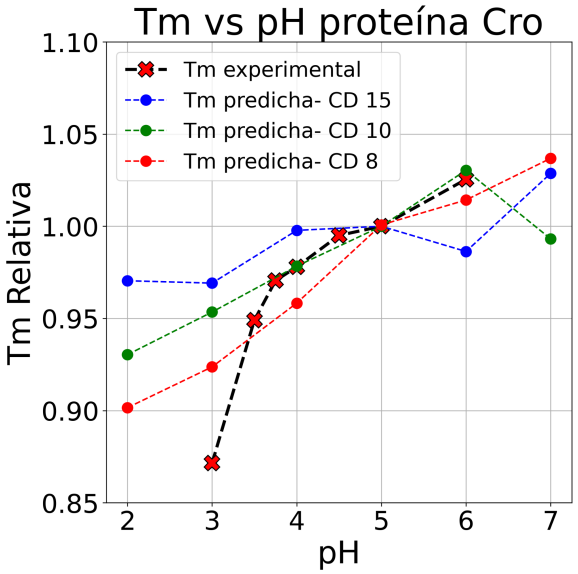


Figura 25 : TMs que se calcularon con el potencial AWSEM diferentes pHs. CD corresponde a la constante dieléctrica usada.

Las tres constantes dieléctricas usadas, muestran una disminución de la Tm al bajar el pH. La disminución de la Tm se ve es mayor al usar la constante dieléctrica de 8. Particularmente a pH 2 se llega a 0.9 cómo Tm relativa mínima. La constante dieléctrica de 10 muestra mejores resultados reproduciendo los datos experimentales dentro del rango de pH 4 a 6. Cuando se usa la constante dieléctrica de 15, la fuerza electrostática es menor respecto de las otras. Parece ser necesario que el pH sea muy ácido, menor a 4, para que el efecto del pH desestabilice el plegado de la proteína, mostrando Tms menores a 1.
Las simulaciones no llegaban a 0.8 contactos nativos. Su promedio rondaba los 0.7 contactos nativos. El potencial AWSEM, es predictivo, por lo que era esperable que el estado plegado tuviera menos contactos formados que al usar el potencial Go. Analizamos el RMSD a lo largo de la dinámica para ver si se predecían solo un estado plegado, o se formaban otros, figura 26.

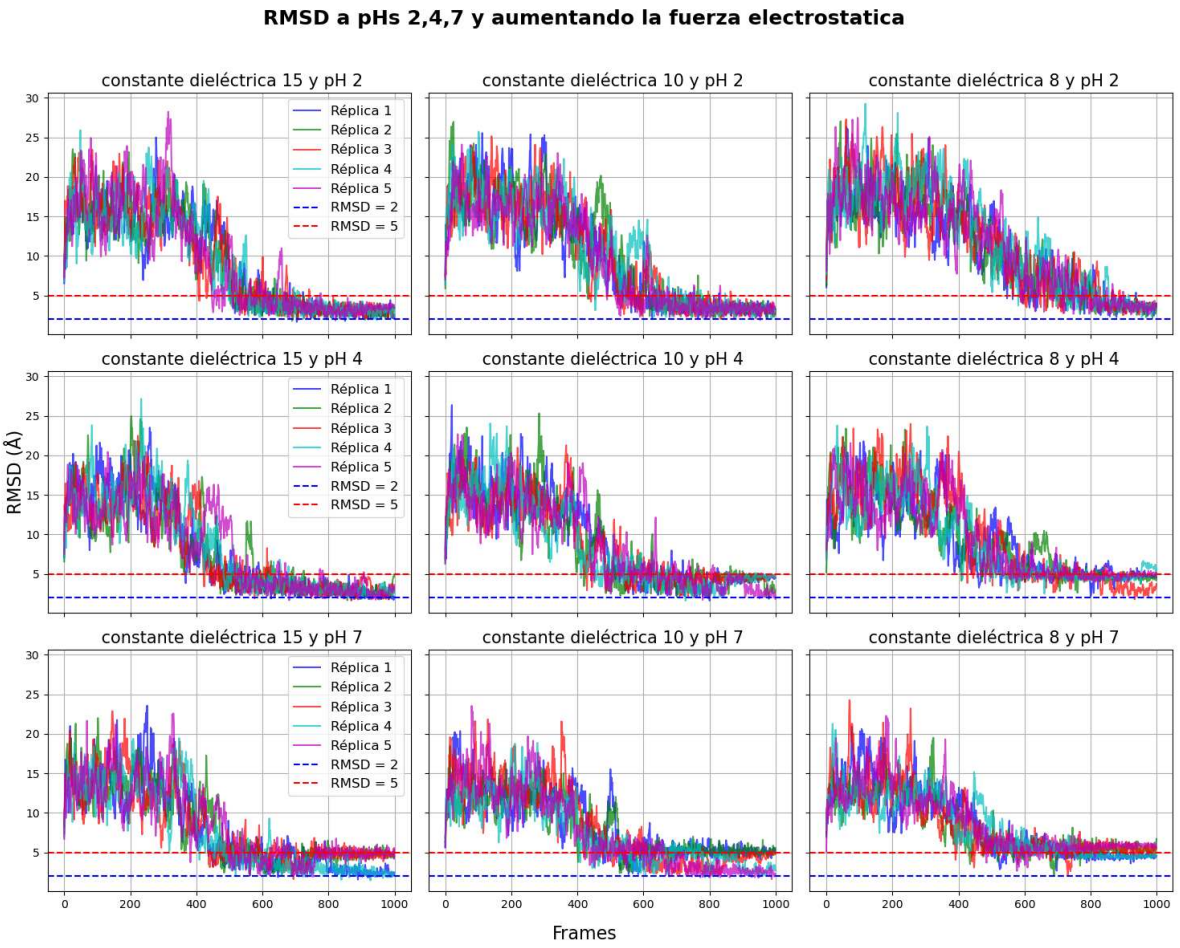


Figura 26: RMSD de las dinámicas a diferentes pH y con diferentes constantes dieléctricas. El eje "y" es el RMSD, se calcula a partir de la estructura del archivo PDB id 1zug. El eje "x" son los frames de las simulaciones. Como la frecuencia del reportero era de 5000 y la simulaciones tenían 5000000 de pasos, tenemos 1000 frames totales por simulación.

Recordamos que las constantes dieléctricas bajas aumentan las fuerzas de las interacciones electrostáticas en la simulación. También que a pH bajos la carga de los aminoácidos ácidos se pierden. Entonces solo me quedan cargas positivas que interaccionan entre ellas de forma repulsiva. En la figura 26 vemos como a pH 2 el RMSD al que llegan las simulaciones, es levemente superior a 2 Å. Un RMSD menor a 2 Å implica que la estructura a la que se llega tiene pocas diferencias con la estructura de referencia (nativa). La disminución en la constante dieléctrica muestra que cuanto mayor es la fuerza electrostática, más tiempo le toma a las simulaciones llegar a valores cercanos a 2 Å. Para el pH 2 y constante 8 se ve cómo se estabiliza recién en los últimos 100 frames.

Al aumentar el pH a 4, el RMSD al que llegan las simulaciones sigue siendo levemente superior a 2 Å. Sin embargo al aumentar la fuerza de las interacciones electrostáticas se empieza a ver algunas réplicas se estabilizan a un RMSD alrededor de 5 Å. Esto nos indica la presencia de otra estructura. Si seguimos aumentando el pH, a 7, se ve como la presencia de otra estructura termoestable, distinta a la nativa, aumenta. Y se ve efectivamente que AWSEM predice otra estructura, con un RMSD 5 Å, si aumentamos la fuerza de las interacciones electrostáticas (constante dieléctrica 8 y pH 7).

En el trabajo Padmanabhan et al. 1999, se menciona la presencia de un puente salino importante para el plegado de la proteína CRO. Pensamos que puede estar relacionado con esta nueva estructura que se predice con AWSEM. La Figura 27 muestra la estructura que encontramos.

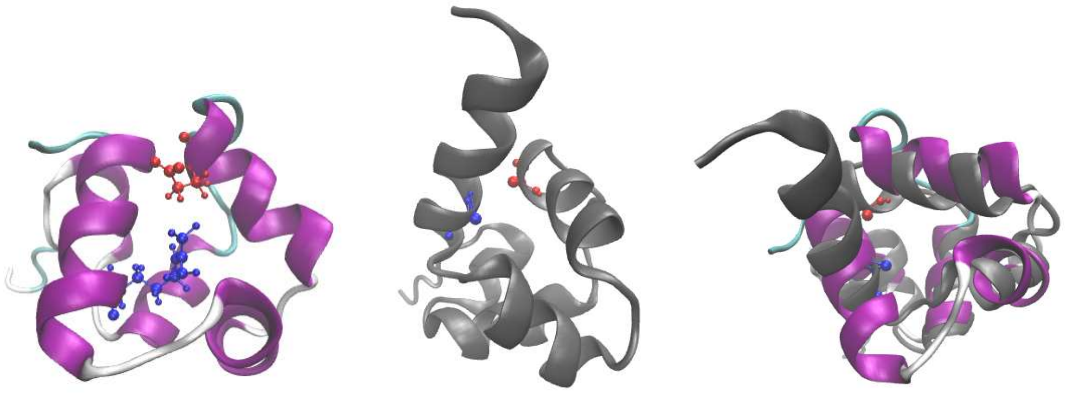


Figura 27: Superposición de la estructura predicha con AWSEM y los residuos 37 (ácido Glutámico) y 12 (Arginina). En violeta se muestra la proteína CRO resuelta experimentalmente y en gris, la predicción de estructura de AWSEM con un constante dieléctrica de 8 y a pH 7. En azul se marca el residuo 12 y con rojo el 37. La superposición de ambas estructuras solo tiene marcada los residuos para la predicción de AWSEM

La figura 27 muestra como el residuo 37 y 12 forman un puente salino. Las diferencias en las estructuras se ven por el hecho de que AWSEM tiene menos átomos para representar a todos los residuos. El aumento en la fuerza electrostática permite que estos dos aminoácidos se acerquen y formen un puente salino afectando a la estructura.

Discusión

Módulo MC

Desarrollamos un módulo en python que cumple con las especificaciones que se describen en la figura 8. Es capaz de tomar la decisión del cambio de carga de un aminoácido ionizable a partir de la estructura. Los pKas que calcula para la frataxina son similares a los calculados con la ecuación 14 de la parametrización. Las diferencias pueden explicarse principalmente al uso del algoritmo de búsqueda estocástico como Montecarlo. Al tener la capacidad de aceptar cambios de cargas desfavorables, puede afectar al cálculo del pKa. Sin embargo esta característica lo hace interesante. El poder aceptar cambios de cargas no favorables nos permite, en la dinámica, explorar diferentes caminos de plegado.
Otro factor importante es que para la parametrización se usaron también los terminales de las proteínas. El módulo MC desarrollado inicialmente no contemplaba estos aminoácidos. Si bien no vimos cambios grandes en los cálculos de los pKas, creemos que es importante tenerlos en cuenta. Hay casos donde la carga de los aminoácidos terminales son cruciales para el estudio del plegado (Pal & Chakrabarti 2000). La implementación se hará posteriormente.

Implementación en OpenAWSEM

La implementación del módulo MC al programa OpenAWSEM fue dentro del módulo de cargas de OpenAWSEM. Ese módulo permitía darle un valor discreto a las cargas de un átomo y cambiar la fuerza de las interacciones electrostáticas. Gracias a eso pudimos acoplar el pH a las dinámicas. Sin embargo, creemos que es posible mejorar la implementación. El cambio de cargas en la simulación requiere del uso de un método dentro del "Contexto". Si bien ese método está optimizado para el cambio de parámetros, en la documentación de OpenMM aclaran que, un uso en exceso podría enlentecer drásticamente las dinámicas. Una mejor implementación del módulo MC sería usarlo dentro del Integrador de la dinámica. De esta forma, podremos usar el módulo MC con mayor frecuencia. En nuestras dinámicas lo usamos con una frecuencia de cada 1000 pasos. Esa frecuencia podría ser suficiente para una proteína pequeña como CRO (71 aa) pero insuficiente para otras proteínas de mayor tamaño. Se trabajará en esta implementación más adelante. No obstante, nuestra implementación inicial nos permitió obtener resultados alentadores para seguir mejorando el código del módulo MC y su implementación.
Finalmente nos gustaría hablar sobre la velocidad de las simulaciones. Gracias a que OpenAWSEM se podía usar en GPU, cada simulación se pudo correr en tiempos menores a una hora. No reportamos en los resultados nada sobre la velocidad de las simulaciones y tampoco cuanto enlentece nuestro módulo MC. Buscamos seguir mejorando el código y una vez incorporado en el Integrador sería interesante ver la diferencia de velocidades.

Todas las simulaciones fueron realizadas en el cluster CCAD (Centro de Cómputo de Alto Desempeño), gracias a ellos este trabajo pudo realizarse en los tiempos previstos del plan de tesis.

Puesta a punto Modelo de Go

La implementación en OpenAWSEM del modelo Go no Aditivo requería de ciertas modificaciones para funcionar correctamente. El factor de normalización de la ecuación 3 no estaba contemplado. Sin la normalización el potencial Go tenía una energía muy negativa y provocaba que la proteína fuera muy estable. Al agregar la normalización el comportamiento mejoró y como se ve en la figura 13, en un modelo no aditivo la energía sube a cero más rápidamente. Este comportamiento, a diferencia del aditivo, muestra como la pérdida de contactos implica la perdida mas rapida de otros contactos. Con estos resultados esperábamos que sea posible replicar una barrera de energía. Lo cual efectivamente vimos en las simulaciones de plegado con el potencial no aditivo. Logramos simular la aditividad y no aditividad de los contactos en las simulaciones.

Simulaciones de plegado, potencial Go modelo aditivo

El modelo aditivo del potencial Go, nos permitió calcular temperaturas de plegado para la simulación sin cargas y para los diferentes pHs. La figura 14 muestra que todas las curvas llegaban alrededor de 0.8 contactos nativos. A pesar de que el potencial Go tiende a la proteína a buscar la estructura definida por el archivo PDB, no esperábamos que llegará a 1. En una dinámica las posiciones van cambiando constantemente no es posible que se mantengan en la posición del PDB original, cuando esté a temperaturas bajas lo que va a ocurrir es que la proteína va oscilar en conformaciones similares a la estructura del PDB pero como no será exactamente la misma el cálculo de los contactos dará menor a 1. Aparte muchas veces las cristalográficas de rayos x, requieren de un posterior paso minimización de energía en las simulaciones. Generalmente las condiciones de cristalografía ponen a la proteína con conformación algo tensas (Ramachandran et al. 2011)
La figura 14 también muestra que en el caso sin cargas la curva se parece más al pH 4 que a pH cercanos a 7. Pensamos que esto ocurre porque el agregado de un potencial extra para las cargas hace más estable la proteína. Por lo que la curva de enfriamiento se desplaza a valores más altos de temperatura. La constante dieléctrica usada para estas simulaciones era menor que la del agua, 11. Se probaron las mismas constantes dieléctricas que la figura 19 y 11 fue la que mejores resultados dio al comparar con los datos experimentales, Figura 14.B.
El paisaje de energía de la figura 16 no era lo esperado. Como vimos que la curva de enfriamiento mostraba que se pasaba de un estado desplegado a uno plegado esperábamos ver, al realizar el paisaje de energía, la presencia de una barrera energética que separa los dos estados, algo parecido a lo que se observa para pH 2. El pH 2 tenía un par de réplicas que se quedaban cercanas a 0.4. En general estos resultados indican que el modelo aditivo puede no ser el que mejor representa al plegado de la proteína CRO.

Por último, nos gustaría hablar sobre el método que usamos para calcular las Tms. Durante todo el trabajo usamos una función sigmoidea para ajustar los datos a esa función. Existen mejores modelos para la estimación de la Tm, modelos termodinámicos que se usan para simular el estado plegado, el estado desplegado y la transición. Además también pueden contemplar la presencia de intermediarios. Estrictamente deberíamos haber usado esos modelos. Optamos por usar la función sigmoidea, porque el comportamiento de los datos era similar y más sencillo de usar, sin embargo es un punto a mejorar en este trabajo.

Simulaciones de plegado, modelo Go no aditivo

El modelo Go no aditivo implica el cambio del parámetro P del potencial Go de 1 a 2. Este modelo no nos permitió calcular la temperatura de plegado. Cuando pasamos a un modelo no aditivo la formación de contactos es más difícil, debido a que se forma una barrera energética. Si la barrera energética que formamos es muy alta, la probabilidad de llegar a un estado de transición será menor. Entonces requerirá de un mayor tiempo de muestreo. Al definir en OpenAWSEM la temperatura inicial y final de la simulación, esta se divide en intervalos y cada intervalo se simula a una temperatura constante y el integrador aumenta o baja la temperatura. La cantidad de pasos que se va a pasar en cada temperatura ya está definida. Si queremos aumentar el tiempo en la temperatura de plegado, es necesario aumentar la cantidad de pasos de simulación o incluso el tiempo de integración por paso.
El potencial no aditivo hace más fuertes los contactos nativos y romperlos implica más energía, lo mismo que formarlos. Por eso el uso de un potencial segado es perfecto para ayudarnos a caminar por el paisaje energético. La figura 18 muestra efectivamente la presencia de una barrera de energía. El hecho de que las barreras de pHs bajos tengan esa forma un poco más plana, puede ser indicativo de un intermediario relativamente estable. El comportamiento del aumento de la energía del estado plegado con pHs más bajo muestra como hay pérdida de la estabilidad.
Ambos modelos de Go, aditivo como no aditivo tienen sus ventajas y desventajas. El modelo de go aditivo es más rápido porque realiza menos cálculos pero puede no representar bien el plegado en algunas proteínas. El modelo no aditivo no nos permitió realizar las curvas de enfriamiento por lo que estimar la temperatura de plegado no fue posible, pero conseguimos explorar el paisaje de energía, al realizar muchas simulaciones a diferentes Q. Es muy probable que usando un valor de P entre 1 y 2 podamos tener un modelo intermedio, que nos de lo mejor de ambos modelos.

Diferentes constantes dieléctricas

Cuando estamos simulando las cargas en una dinámica, es importante elegir una buena constante dieléctrica. La figura 19 muestra como varía el ΔG del estado plegado y desplegado con diferente constantes dieléctricas. OpenAWSEM tiene por defecto la constante dieléctrica a 80, que es la constante dieléctrica del agua. Se sabe que la constante dieléctrica en ambientes proteicos es menor a la que hay en agua (Nelson & Cox 2021). La razón es que el agua disminuye la fuerza electrostática pero en un ambiente hidrofóbico como el de una

proteína la cantidad de agua es menor y por lo tanto la fuerza electrostática aumenta. La constante dieléctrica parece estar entre la de 80 y 22. Esperábamos que fuera menor a la del agua, debido a los resultados del potencial Go aditivo. Considerando que el potencial Go no aditivo, en principio es más fiel a la realidad, creemos que la constante dieléctrica debería encontrarse entre estos valores, 80-22. Posiblemente como la proteína CRO es pequeña y la mayoría de sus residuos cargados están en contacto con el solvente, eso explicaría porque no es tan diferente a la constante dieléctrica del agua.
Vemos como hay un rol importante sobre qué constante dieléctrica usar. Dependiendo del modelo, la fuerza electrostática afecta de forma diferente. Una crítica que se puede hacer, es que la constante dieléctrica en las simulaciones es siempre la misma. Sería muy interesante poder cambiar esta constante dieléctrica durante la simulación. Cuando la proteína se encuentra desplegada las cargas están más expuestas al solvente y en esos casos una constante de 80 sería más apropiada. Realizar este trabajo dentro de OpenAWSEM no debería representar mayor dificultad que todas las otras modificaciones que ya realizamos al programa.

Análisis del Mecanismo de Plegamiento a Diferentes pHs

El análisis del plegamiento por residuo de la proteína, mostró sitios interesantes con aminoácidos cargados. La figura 20 son los q locales a pH 7. Lo que vemos es que los valores cercanos al intermediario 0.4 tienen más plegado los primeros 10 residuos de la proteína y la zona de 10 a 20 tiene muchas dificultades para plegarse. Esta zona tiene varios residuos cargados, K (10), R (11), R (12), K (16). Todos los residuos son básicos. La repulsión de todas estas cargas positivas a pH 7 puede explicar la dificultad del plegamiento. Yendo ahora al estado plegado, Q mayores a 0.7, se ve también tiene dificultades para plegar entre los residuos 10 a 20 pero también con los últimos residuos de la proteína. Cuando vimos las simulación, esta zona era la que más libertad de movimiento tenía. Es más, si abrimos el archivo PDB (1zug) y lo coloreamos por ocupancía veremos que la zona de la cola c terminal, últimos residuos, es una zona que tiene muchas conformaciones. Tener muchas conformaciones implica una zona muy móvil. También se encuentra llena de aminoácidos básicos, Y(63), K (66), R (67), K (69), que podrían impedir el plegado por repulsión de cargas.
Se realizaron varios mapas de calor, para los diferentes pHs, en general el comportamiento era el mismo. Se comparó el pH 2 con el pH 7 por ser el otro caso más extremo. La figura 21 muestra las diferencia de Q locales entre el pH 7 y el pH 2. Esto nos mostró mejor las diferencias que no se veían a simple vista entre los dos pHs. El estado de transición, Q 0.4, se ven bastante parecidos para ambos pHs. Las principales diferencias se ven a 0.3. En particular las zonas más diferentes son los primeros 10 residuos y los residuos de 20 a 30. La figura 22 muestra mejor los primeros 10 residuos en la proteína. Son los siguientes: E (6), R (7), K (9), K (10). En ambos pHs esta zona tiene los residuos bastante plegados pero para pH 2 los últimos residuos están más desplegados. La diferencia principal radica en la ausencia de la carga del glutámico a pH 2. Da indicios que este residuo disminuye la repulsión entre los demás residuos básicos cuando está cargado a pH 7. La siguiente zona es la de 20 a 30, los residuos ionizables son : E (21), K (25), K (29). Aca el comportamiento es el opuesto esta

zona está más desplegada para pH 7 que para pH 2. No encuentro un sentido a que la presencia de la carga del glutámico haga más inestable esta zona, se requiere de un análisis más exhaustivo.
Finalmente el gráfico 24, se usó para comparar las estructuras. Lo que podemos agregar de este gráfico es que la zona del medio la proteína está más plegada a pH 2 y podría deberse a que la presencia de glutámico 49 que hay en esa zona, desestabiliza la estructura cuando está cargado a pH 7 y no puede hacerlo a pH 2 por no estar cargado. Quiero agregar que no encontramos diferencias relacionadas con el glutámico 37, en bibliografía (Padmanabhan et al. 1999) vimos que este residuo era importante por formar un puente saliendo con la Arginina 12. A pH 4 se pierde la carga del glutámico y eso provoca que se pierda gran parte de la estabilidad de la proteína.

Simulaciones con Campo de Fuerza AWSEM

Los resultados con el potencial AWSEM mostraron una dependencia en la estructura que no habíamos encontrado con el potencial Go. Cuando se analizó la diferencia entre los mecanismos de plegado del pH 2 y 7 con el potencial Go no aditivo, no encontramos nada relevante con el glutámico 37. Ahora vimos cómo se forma un puente salino, entre el glutámico 37 y la arginina 12. Este puente es importante para la estructura debido a que se encuentra en un ambiente muy hidrofóbico de la proteína, por lo cual la fuerza electrostática es mayor justo para este par. Es interesante ver que la que mejor constante dieléctrica es la 8, al menos para simular el desplazamiento de la Tm a pH bajos. Probamos constantes dieléctricas más bajas, para aumentar la fuerza pero las dinámicas empezaron a dar diferentes estructuras, lo cual provocó que el cálculo de la Tm no guardará una relación con el pH. Si aumentamos en exceso la fuerza electrostática las dinámicas empiezan a comportarse regularmente. Una crítica que tenemos con estas simulaciones fue que el rango de la curva de temperatura fue mayor al que hicimos con el potencial Go. Por lo tanto el tiempo que se pasó en cada temperatura fue menor, y esto podría haber afectado al plegado.

Conclusión

El mundo de las dinámicas moleculares en proteínas es muy complejo y existen muchos modelos que nos ayudan a comprender mejor cómo son las interacciones intramoleculares. En este trabajo nos enfocamos solamente en el plegado proteico de una sola proteína pequeña, como la CRO. Aun siendo solo un modelo pequeño logramos usar diversas herramientas para estudiar su plegado, calcular una constante termodinámica como la temperatura de plegado (Tm), ver el mecanismo de plegado y la resistencia al pH.

Usamos tres modelos en total, cada uno con sus ventajas y desventajas. El modelo de Go aditivo, nos permitió replicar un comportamiento bastante similar de la Tm a diferentes pHs. Sin embargo resultó no ser apropiado para replicar la barrera energética que hay entre dos estados termodinámicos, como el plegado y desplegado de la proteína CRO. Al cambiar al modelo de Go no aditivo. Logramos replicar la barrera de energía entre los estados plegado y desplegado y combinado con el método de "Umbrella sampling" pudimos estudiar el paisaje de energía. Donde también vimos un efecto del pH acorde a los datos experimentales que teníamos sobre los ΔG de plegado a diferentes pH.

El campo de fuerza AWSEM, también mostraba una desestabilidad de la proteína CRO con la disminución del pH. Al mismo tiempo nos mostró la diferencia entre un modelo predictivo y los modelos de Go. Es capaz de predecir más estructuras estables ya que no está guiado hacia una estructura de referencia. Aunque la estructura que nos dió finalmente no es exactamente la nativa, vemos que es capaz de dar más información sobre el plegado de una proteína. El hecho de que replicara el único puente salino de la proteína, muestra su capacidad para el estudio del plegado proteico.

Como conclusión general, la implementación del modelo de pH constante, desarrollado por nuestro laboratorio, fue exitosa. Incorporar este modelo dentro del programa OpenAWSEM permitirá a muchas personas poder estudiar dinámicas moleculares a diferentes pHs. La velocidad que nos da el uso de la GPU y un modelo de grano grueso, permite que las dinámicas se hagan en tiempos muy cortos (horas). Sin embargo vimos que el uso de una constante dieléctrica no es arbitrario y dependiendo del modelo/potencial que se use este variara. Un trabajo que resulta interesante es buscar la mejor constante dieléctrica que se pueda usar para cualquier proteína. Ese trabajo requeriría de usar más casos de estudio para generalizar.

Tenemos perspectivas futuras sobre nuestro trabajo, esperamos poder combinar el uso del Potencial Go y el campo de fuerza AWSEM. Vimos que cada uno tiene sus ventajas y la combinación de ambos potenciales es plausible. Si bien no estudiamos el paisaje de energía con el potencial AWSEM, por lo extensa que se volvería la tesis, sabemos que para la proteína CRO no se logra simular una barrera de energía. Esperamos que al combinarlos ambos potenciales puedan verse favorecidos.

Este trabajo fue largo y exigente pero aprendí todo lo que quería saber y me abrí a un mundo nuevo que no conocía, espero sea solo el comienzo.

Bibliografía

Allison, J. R. Computational methods for exploring protein conformations (2020) Biochemical Society Transactions, 48(4), 1707-1724.

Bottaro, S., & Lindorff-Larsen, K. Biophysical experiments and biomolecular simulations: A perfect match? (2018). Science, 361(6400), 355-360.

Davtyan, A., Schafer, N. P., Zheng, W., Clementi, C., Wolynes, P. G., & Papoian, G. A. AWSEM-MD: protein structure prediction using coarse-grained physical potentials and bioinformatically based local structure biasing (2012). The Journal of Physical Chemistry B, 116(29), 8494-8503.

Dill, K. A., & MacCallum, J. L. The protein-folding problem, 50 years on (2012). science, 338(6110), 1042-1046.

Earl, L. A., Falconieri, V., Milne, J. L., & Subramaniam, S. Cryo-EM: beyond the microscope (2017). . Current opinion in structural biology, 46, 71-78.

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., ... & Pande, V. S. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics (2017). PLoS computational biology, 13(7), e1005659.

Eastwood, M. P., & Wolynes, P. G. Role of explicitly cooperative interactions in protein folding funnels: a simulation study (2001). The Journal of Chemical Physics, 114(10), 4702-4716.

Harris, T. K., & Turner, G. J. Structural basis of perturbed pKa values of catalytic groups in enzyme active sites (2002). IUBMB life, 53(2), 85-98.

Isom, D. G., Cannon, B. R., Castañeda, C. A., Robinson, A., & García-Moreno E, B. High tolerance for ionizable residues in the hydrophobic interior of proteins. (2008). Proceedings of the National Academy of Sciences, 105(46), 17784-17788.

Johnston, N., Dettmar, P. W., Bishwokarma, B., Lively, M. O., & Koufman, J. A. Activity/stability of human pepsin: implications for reflux attributed laryngeal disease. (2007). The Laryngoscope, 117(6), 1036-1039.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... & Hassabis, D. Highly accurate protein structure prediction with AlphaFold. (2021). nature, 596(7873), 583-589.

Kästner, J. Umbrella sampling. Wiley Interdisciplinary Reviews: Computational Molecular. (2011). Science, 1(6), 932-942.

Li, H., Robertson, A. D., & Jensen, J. H. Very fast empirical prediction and rationalization of protein pKa values. (2005). Proteins: Structure, Function, and Bioinformatics, 61(4), 704-721.

Lu, W., Bueno, C., Schafer, N. P., Moller, J., Jin, S., Chen, X., ... & Wolynes, P. G. OpenAWSEM with Open3SPN2: A fast, flexible, and accessible framework for large-scale coarse-grained biomolecular simulations. (2021). PLoS computational biology, 17(2), e1008308.

Meng, Y., & Roitberg, A. E. Constant pH replica exchange molecular dynamics in biomolecules using a discrete protonation model. (2010). Journal of chemical theory and computation, 6(4), 1401-1412.

Milner-White, E. J. Protein three-dimensional structures at the origin of life. (2019). Interface Focus, 9(6), 20190057.

Nelson, D. L., & Cox, M. M. Lehninger principles of biochemistry (8th ed.). (2021). W. H. Freeman.

Ogunjobi, T. T., Okorie, I. C., Gigam-Ozuzu, C. D., Olorunleke, J. V., Ogunleye, F. I., Irimoren, E. O., ... & Ojo, E. O. Bioinformatics tools in protein analysis: Structure prediction, interaction modelling, and function relationship. (2024). European Journal of Sustainable Development Research, 8(1).

Onuchic, J. N., & Wolynes, P. G. Theory of protein folding. (2004). Current opinion in structural biology, 14(1), 70-75.

Pahari, S., Sun, L., & Alexov, E. PKAD: a database of experimentally measured pKa values of ionizable groups in proteins. (2019). Database, 2019, baz024.

Pal, D., & Chakrabarti, P. Terminal residues in protein chains: residue preference, conformation, and interaction. (2000). Biopolymers: Original Research on Biomolecules, 53(6), 467-475.

Padmanabhan, S., Laurents, D. V., Fernandez, A. M., Elias-Arnanz, M., Ruiz-Sanz, J., Mateo, P. L., ... & Filimonov, V. V. Thermodynamic analysis of the structural stability of phage 434 Cro protein. (1999). Biochemistry, 38(47), 15536-15547.

Reis, P. B., Clevert, D. A., & Machuqueiro, M. pKPDB: a protein data bank extension database of pKa and pI theoretical values. (2022). Bioinformatics, 38(1), 297-298.

Reis, P., Bertolini, M., Montanari, F., Rocchia, W., Machuqueiro, M., & Clevert, D. A. pKAI: a fast and interpretable deep learning approach for accurate electrostatics-driven pKa predictions. (2021). Research Square.

Robert, C. P., & Casella, G. The metropolis—hastings algorithm. (2004). Monte Carlo statistical methods, 267-320.

Rokde, C. N., & Kshirsagar, M. (2013, July). Bioinformatics: Protein structure prediction. In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.

Ramachandran, S., Kota, P., Ding, F., & Dokholyan, N. V. Automated minimization of steric clashes in protein structures. (2011). Proteins: Structure, Function, and Bioinformatics, 79(1), 261-270.

Sahyun, M. (Ed.). Introducción al estudio de los aminoácidos y proteínas. (1947). Buenos Aires: Médico-Quirúrgica.

Talley, K., & Alexov, E. On the pH‐optimum of activity and stability of proteins. (2010). Proteins: Structure, Function, and Bioinformatics, 78(12), 2699-2706.

Tavella F, Wolynes PG, Roman EA. Constant pH MD simulations using a transferable coarse grained potential. 2018. Presentado en: 3rd Protein Biophysics at the End of the World, Pontificia Universidad Católica de Chile;

Ullner, M., Woodward, C. E., & Jönsson, B. A Debye–Hückel theory for electrostatic interactions in proteins. (1996). The Journal of chemical physics, 105(5), 2056-2065.

Xu, S., & Onoda, A. Accurate and Rapid Prediction of Protein pKa: Protein Language Models Reveal the Sequence-pKa Relationship. (2024). bioRxiv, 2024-09.

Apéndice

Parámetros	Valor
R máximo polar	5
R máximo no polar	7
Tau (polar y no polar)	0.1
Número máximo de vecinos polares	3.9
Número máximo de vecinos no polares	17.78
alfa polar	0.416683
alfa no polar	0.049
Bp (D, E, K. R, H, C, Y, X, Z)	1,21 - 0,64 - 0,0185 - 0,0185 -1,01 -0,037 - 0 - 0,0185 - 1,21
Bnp (D, E, K. R, H, C, Y, X, Z)	9,33 - 15,84 - 9,48 - 9,48 - 15,69 -85,91- 0 - 9,48 - 9,33
K elec	2.43232
L (apantallamiento)	10

Tabla 1 apéndice. Parámetros del modelo de pH constante calculados por cuadrados mínimos.

Los parámetros de distancia como el apantallamiento y los Rmax están en Å. Los demás como Bnp, Bp o K elec, son proporcionales a KbT

Módulo MC

```
import numpy as np
import pandas as pd
import random
import math

def convert_sequence_to_three_letter(seq):
        # Diccionario de mapeo de una letra a tres letras
        one_to_three = {
        'A': 'ALA', 'R': 'ARG', 'N': 'ASN', 'D': 'ASP',
        'C': 'CYS', 'E': 'GLU', 'Q': 'GLN', 'G': 'GLY',
        'H': 'HIS', 'I': 'ILE', 'L': 'LEU', 'K': 'LYS',
        'M': 'MET', 'F': 'PHE', 'P': 'PRO', 'S': 'SER',
        'T': 'THR', 'W': 'TRP', 'Y': 'TYR', 'V': 'VAL',
        'X': 'NTR', 'Z': 'CTR',
        }
        # Convertir la secuencia de una letra a tres letras
        seq_list = [one_to_three[aa] for aa in seq]
        return seq_list

def calcular_distancia(punto1, punto2):
        return np.sqrt((punto1[0] - punto2[0])**2 + (punto1[1] - punto2[1])**2 + (punto1[2] -
punto2[2])**2)

def charge_flip(aob, old_charge):
        new_charge = 0.0
        if aob == -1.0:  # Ácido
        if old_charge == -1.0:
        new_charge = 0.0
        elif old_charge == 0.0:
        new_charge = -1.0
        elif aob == 1:  # Base
        if old_charge == 1.0:
        new_charge = 0.0
        elif old_charge == 0.0:
        new_charge = 1.0
        return new_charge

class Montecarlo_change_charge:

        def __init__(self, matriz_posiciones, charged_residues, seq, enlaces_matrix, pH):
        self.matriz_posiciones = matriz_posiciones
        self.charged_residues = charged_residues
        self.seq_list = convert_sequence_to_three_letter(seq)
        self.type_dict = {
        'ALA': 'N', 'ARG': 'B', 'ASN': 'P', 'ASP': 'A', 'CYS': 'A', 'GLU': 'A', 'GLN': 'P', 'GLY':
'G', 'HIS': 'B',
        'ILE': 'N', 'LEU': 'N', 'LYS': 'B', 'MET': 'N', 'PHE': 'N', 'PRO': 'N', 'SER': 'P', 'THR':
'P', 'TRP': 'N',
        'TYR': 'A', 'VAL': 'N', 'NTR': 'B', 'CTR': 'A',
        }
        self.pH = pH
        self.enlaces_matrix = enlaces_matrix

        def choose_residue_mc(self):
        self.residue_mc, self.charge = random.choice(self.charged_residues)

        def acid_basic_residue_mc(self):
        self.resname_residue_mc = self.seq_list[self.residue_mc]
        self.acid_basic_residue_mc = 1 if self.type_dict[self.resname_residue_mc] == 'B' else -1
```

```
        def distances_residue_mc_neighbors(self):
        # Obtener la posición del residuo de interés
        self.position_residue_mc = self.matriz_posiciones[self.matriz_posiciones['Residue_Number']
== self.residue_mc][['X', 'Y', 'Z']].values[0]
        # Calcular las distancias a todos los demás residuos
        self.neighbors = []
        self.distances = []
        for index, row in self.matriz_posiciones.iterrows():
        if row['Residue_Number'] != self.residue_mc:
                self.position_neighbor = [row['X'], row['Y'], row['Z']]
                self.distance = calcular_distancia(self.position_residue_mc,
self.position_neighbor)
                self.neighbors.append(row['Residue_Number'])  # Cambio de nombre a 'vecinos'
                self.distances.append(self.distance)
        return self.distances, self.neighbors

        def get_resnames_neighbors(self):
        self.distances_residue_mc_neighbors()

        self.neighbors = [int(neighbor) for neighbor in self.neighbors]
        # Obtener los nombres de los residuos correspondientes a los índices en 'vecinos'
        self.resnames_neighbors = [self.seq_list[residue] for residue in self.neighbors]
        #return self.resnames_neighbors

        def get_charge_of_neighbors(self):
        self.distances_residue_mc_neighbors()

        self.charge_of_neighbors = []
        for residue in self.neighbors:
        charge_neighbor = next((charge for r, charge in self.charged_residues if r == residue), 0)
        self.charge_of_neighbors.append(charge_neighbor)
        return self.charge_of_neighbors

        def neighbors_acid_basic(self):
        self.acids = [-1 if self.type_dict[res] == 'A' else 0 for res in self.resnames_neighbors]
        self.basics = [1 if self.type_dict[res] == 'B' else 0 for res in self.resnames_neighbors]
        self.glicinas = [1 if res == 'GLY' else 0 for res in self.resnames_neighbors]
        return self.acids, self.basics, self.glicinas

        def count_polars_no_polars(self):
        self.get_resnames_neighbors()

        # Parametros Polares y No_polares, radio de la esfera para medir
        R_max = 5/10  # Umbral para distancia de polares
        R_max_no_polares = 7/10  # Umbral para distancia de no polares
        tau = 0.1*100  # Valor para tau en la fórmula

        self.polares = []
        self.no_polares = []
        for res, distancia in zip(self.resnames_neighbors, self.distances):
        if self.type_dict[res] in ['P', 'B', 'A']:
                valor_polar = 1 if distancia <= R_max else math.exp(-tau * (distancia - R_max)**2)
                self.polares.append(valor_polar)
        else:
                self.polares.append(0)

        if self.type_dict[res] == 'N':
                valor_no_polar = 1 if distancia <= R_max_no_polares else math.exp(-tau * (distancia
- R_max_no_polares)**2)
                self.no_polares.append(valor_no_polar)
        else:
                self.no_polares.append(0)
```

```
        return self.polares, self.no_polares

        def sum_polares_no_polares(self):
        self.count_polars_no_polars()
        self.environment_polar = sum(self.polares)  # Sumar vecinos polares
        self.environment_no_polar = sum(self.no_polares)# Sumar vecinos no polares
        return self.environment_no_polar, self.environment_polar

        def calculate_term_polar(self): # todos los Bp Bnp estan multiplicados por Kb * T =
0.001987 * 300, por un tema de unidades
        self.sum_polares_no_polares()
        self.polar_no_polar_parameter
={'ASP':(0.72426,5.57075),'CTR':(0.72426,5.57075),'GLU':(0.380763,9.44846),'LYS':(0.0110311,5.65882)
,
'ARG':(0.0110311,5.65882),'NTR':(0.0110311,5.65882),'HIS':(0.602896,9.36507),'CYS':(0.037 * 0.001987
* 300,85.91 * 0.001987 * 300),
                                        'TYR':(0.0, 0.0)}
        Bp, Bnp = self.polar_no_polar_parameter[self.resname_residue_mc]

        Npmax = 3.9
        Nnpmax = 17.78
        alfaP = 0.416683
        alfanp = 0.049

        # Calculate Up and Unp
        self.Up = math.exp(-alfaP * (self.environment_polar - Npmax)**2) if self.environment_polar
<= Npmax else 1
        self.Unp = math.exp(-alfanp * (self.environment_no_polar - Nnpmax)**2) if
self.environment_no_polar <= Nnpmax else 1

        # Calculate the term
        self.term_polar = self.acid_basic_residue_mc  * (Bnp * self.Unp - Bp * self.Up)
        return self.term_polar


        def calculate_term_elec(self):
        self.get_charge_of_neighbors()
        self.distances_residue_mc_neighbors()

        K_elec = 2.43232/10
        L = 10.0/10.0
        self.E_elect = []

        for charge_neighbor, distance in zip(self.charge_of_neighbors, self.distances):
        if charge_neighbor == -1:  # Vecino ácido
                energy_electrostatic = self.charge * ((-1 / distance) * math.exp(-distance / L))
                self.E_elect.append(energy_electrostatic)
        elif charge_neighbor == 1:  # Vecino básico
                energy_electrostatic = self.charge * ((1 / distance) * math.exp(-distance / L))
                self.E_elect.append(energy_electrostatic)
        elif charge_neighbor == 0:  # Vecino sin carga
                self.E_elect.append(0)
        # Sumar la energia Electrostatica
        self.term_elec = sum(self.E_elect) * K_elec
        return self.term_elec

        def calculate_new_charge(self):
        self.choose_residue_mc()
        self.new_charge = charge_flip(self.acid_basic_residue_mc, self.charge)
        self.delta_carga = self.new_charge - self.charge
        return self.new_charge, self.delta_carga
```

```
        def calculate_delta_term_elec(self):
        self.calculate_new_charge()
        self.get_charge_of_neighbors()
        self.distances_residue_mc_neighbors()
        self.calculate_term_elec()

        K_elec = 2.43232/10
        L = 10.0/10.0
        self.new_E_elect = []

        for charge_neighbor, distance in zip(self.charge_of_neighbors, self.distances):
        if charge_neighbor == -1:  # Vecino ácido
                energy_electrostatic = self.new_charge * ((-1 / distance) * math.exp(-distance /
L))
                self.new_E_elect.append(energy_electrostatic)
        elif charge_neighbor == 1:  # Vecino básico
                energy_electrostatic = self.new_charge * ((1 / distance) * math.exp(-distance / L))
                self.new_E_elect.append(energy_electrostatic)
        elif charge_neighbor == 0:  # Vecino sin carga
                self.new_E_elect.append(0)
        self.new_term_elec = sum(self.new_E_elect) * K_elec
        self.delta_term_elec = self.new_term_elec - self.term_elec
        return self.delta_term_elec

        def calculate_delta_term_polar(self):
        self.calculate_new_charge()
        self.calculate_term_polar()

        self.delta_term_polar = self.delta_carga * self.term_polar
        return self.delta_term_polar

        def asignation_pKa_ref(self):
        self.acid_basic_residue_mc()
        # pKas libres en solución
        pKa_ref_dict = {
        'ASP': 4.0, 'GLU': 4.5, 'LYS': 10.6, 'ARG': 12.0,
        'HIS': 6.4, 'CYS': 8.3, 'TYR': 11.0, 'NTR': 7.5, 'CTR': 3.5
        }
        self.pKa_ref = pKa_ref_dict.get(self.resname_residue_mc)
        return self.pKa_ref

        def calculate_delta_term_pH(self):
        self.asignation_pKa_ref()
        kb = 0.001987
        T = 300
        self.delta_term_pH = self.delta_carga * (int(self.pH) - int(self.pKa_ref)) * kb * T *
np.log(10)

        def accept_or_reject(self):
        self.calculate_delta_term_elec()
        self.calculate_delta_term_pH()
        self.calculate_delta_term_polar()

        kb = 0.001987
        T = 300
        self.lista_cambios = []
        self.change_energy_mc = self.delta_term_pH + self.delta_term_elec + self.delta_term_polar

        if self.change_energy_mc < 0:
        # Si la energía disminuye, aceptar la nueva carga
        for index, (residuo, _) in enumerate(self.charged_residues):
                if residuo == self.residue_mc:
                self.charged_residues[index] = (residuo, self.new_charge)
```

```
                self.lista_cambios.extend(self.charged_residues)  # Agregar elementos directamente
        else:
        # Si la energía aumenta, aceptar con una probabilidad determinada por el criterio de
Metropolis
        random_prob = random.uniform(0, 1)
        if random_prob <= np.exp(-self.change_energy_mc/(kb*T)):
                for index, (residuo, _) in enumerate(self.charged_residues):
                if residuo == self.residue_mc:
                        self.charged_residues[index] = (residuo, self.new_charge)
                        self.lista_cambios.extend(self.charged_residues)  # Agregar elementos
directamente
        else:
                # Si no se acepta el cambio, agregar la configuración actual a la lista de cambios
                self.lista_cambios.extend(self.charged_residues)  # Agregar elementos directamente

        bond_matrix = []
        for _, row in self.enlaces_matrix.iterrows():
        if row['seq_i'] == self.residue_mc or row['seq_j'] == self.residue_mc:
                bond_index = int(row['bond_index'])  # Asegurar que bond_index es un entero
                # Obtener las cargas correspondientes de charged_residues
                carga_i = next(carga for residuo, carga in self.lista_cambios if residuo ==
row['seq_i'])
                carga_j = next(carga for residuo, carga in self.lista_cambios if residuo ==
row['seq_j'])
                bond_matrix.append([bond_index, carga_i, carga_j])

        # Convertir bond_matrix a DataFrame
        self.bond_matrix_df = pd.DataFrame(bond_matrix, columns=['bond_index', 'carga_i',
'carga_j'])
        return self.lista_cambios, self.bond_matrix_df
```