UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Estudio comparativo de diferentes metodologías de muestreo acelerado en dinámica molecular

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

FÍSICO

P R E S E N T A :

Jorge Alejandro Amador Herrera

DIRECTOR DE TESIS:
Dr. Marcelino Arciniega Castro
Ciudad Universitaria, CD. MX., 2018

1. Datos del alumno
Amador
Herrera
Jorge Alejandro
56672964
Universidad Nacional Autónoma de México
Facultad de Ciencias
Física
310036069

2. Datos del tutor
Dr.
Marcelino
Arciniega
Castro

3. Datos del sinodal 1
Dr.
David
Philip
Sanders

4. Datos del sinodal 2
Dr.
Isaac
Pérez
Castillo

5. Datos del sinodal 3
Dr.
Ricardo Atahualpa
Solórzano
Kraemer

6. Datos del sinodal 4
Dra.
Laura
Dominguez
Dueñas

7. Datos del trabajo escrito.
Estudio comparativo de diferentes metodologías
de muestreo acelerado en dinámica molecular
82 p
2018

A mis padres

Agradecimientos

Primeramente a mi familia, este trabajo es por y para ellos. Mis padres, Guillermo Amador y Rosario Herrera, que han dado todo por mí, gracias a su inmenso apoyo y esfuerzo he llegado hasta aquí. A mis hermanos, Guillermo y Carlos Amador, por todo lo que hemos vivido y por motivarme a seguir adelante. A mi novia, Rebeca Hernández por enriquecer mi vida a cada momento y acompañarme durante todo este viaje.

A la gente que me ha brindado su valiosa amistad. Misael Zepeda, Luis De la Rosa, Fabio Caballero y Pedro Roldan, mis amigos de toda la vida, las experiencias que hemos vivido juntos son tesoros invaluables. Martín Chávez, Francisco Rodríguez y Eduardo Bahena, amigos inolvidables con quienes forjé mi sentido del humor y cuyas historias siempre llevaré conmigo. A Fernanda Esquinca y Claudia Álvarez por la reciente pero gran amistad que hemos construido.

Al Doctor Marcelino Arciniega Castro por guiarme e impulsarme a mejorar mi trabajo día con día. Agradezco también al Dr. David Sanders, al Dr. Isaac Pérez, al Dr. Ricardo Atahualpa y a la Dra. Laura Domínguez por los valiosos comentarios que enriquecieron el contenido de esta tesis.

A la Universidad Nacional Autónoma de México, por el apoyo infinito que me ha proporcionado desde mis años en el bachillerato.

Finalmente, agradezco el apoyo del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) y a la Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC) mediante los proyectos IA202917 y LANCAD-UNAM-DGTIC-320, respectivamente.

Índice general

Introducción III

1. Modelado de dinámicas moleculares **1**
1.1. Mecánica estadística 1
1.2. Dinámicas moleculares y de Langevin 4

2. Métodos de muestreo acelerado **11**
2.1. Método de inundación 11
2.2. GaMD 13
2.3. Metadinámica 16
2.4. Método Variacional 19

3. Modelos de Markov **24**
3.1. Formalismo de cadenas de Markov 24
3.2. Metaestabilidad mediante cadenas de Markov 27
3.3. Probabilidades de compromiso 29

4. Resultados y discusión **33**
4.1. Una dimensión 33
4.1.1. Potencial con sector difusivo 33
4.1.2. Potencial con perturbaciones 38
4.2. Dos dimensiones 42
4.2.1. Potencial con sectores difusivos 42
4.2.2. Potencial con distintos caminos y sectores alejados 47
4.3. Discusión 53
4.3.1. Método de inundación 53
4.3.2. GaMD 54
4.3.3. Metadinámica 56
4.3.4. Método variacional 58

4.3.5. Comparación 60

5. Conclusiones **62**

A. Ponderación energética con expansión al segundo orden **63**

B. Código principal utilizado en este trabajo **64**

Introducción

Las proteínas son biomoléculas cuyas funciones incluyen catalizar reacciones metabólicas, replicación de ADN, transporte de moléculas, entre otras [1]. Las funciones de una proteína en particular dependen de la composición atómica de la misma así como de la conformación o arreglo geométrico en el que se encuentre.

Al ser moléculas flexibles, las proteínas experimentan diversos cambios estructurales a lo largo de su evolución dinámica. Las distintas conformaciones de una proteína o *estados conformacionales* corresponden a los conjuntos metaestables del espacio de configuraciones asociado a las mismas [2]. Así, una proteína puede visitar distintas conformaciones al evolucionar en el tiempo, dependiendo de las condiciones de su entorno. Por ejemplo, la proteína priónica (PrP) se encuentra normalmente en el cerebro de varios animales y en el del ser humano. Al encontrarse en su conformación *natural*, esta proteína está relacionada con procesos de creación y formación de sinapsis, memoria a largo plazo y regulación de muerte celular [3]. Sin embargo, en otra conformación, conocida como PrP^{SC}, esta proteína se vuelve resistente a proteasas (enzimas encargadas de romper enlaces en proteínas) y crea aglomerados (llamados priones) con otras PrP^{SC}. Los priones están relacionados con enfermedades neurodegenerativas como encefalopatías en bovinos y la enfermedad de Creutzfeldt-Jakob en humanos [4].

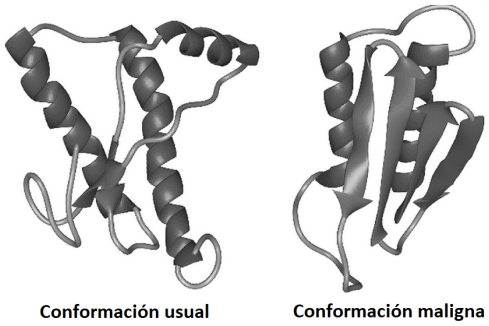

Figura 1: La proteína priónica en su conformación natural PrP y en la conformación maligna PrP^{SC}. Imagen cortesía de [5]

Debido a que varias características funcionales de las proteínas dependen de sus distintas conformaciones [6], es importante estudiar estas estructuras, así como la manera en que la proteína cambia de una conformación a otra.

Con la implementación de simulaciones atomistas de dinámica molecular, se han podido analizar diversas proteínas mediante la descripción de sus espacios de configuraciones. En estas simulaciones se consideran los distintos potenciales que actúan en una proteína ya sea que se encuentre solventada en alguna sustancia, interactuando con otras moléculas o en el vacío. Luego, al integrar numéricamente las ecuaciones que determinan su evolución bajo estos potenciales, se obtiene una distribución de probabilidad $p(\boldsymbol{r})$ de las posibles configuraciones $\boldsymbol{r}$ del sistema.

Asumiendo que el sistema es cerrado a temperatura constante, la superficie de energía potencial (PES por las siglas en inglés de *potential energy surface*) será proporcional a $-1/\beta \ln[p(\boldsymbol{r})]$. Para estudiar las conformaciones de una proteína, es común utilizar *coordenadas colectivas*, que son un conjunto de parámetros que caracterizan los cambios físicos significativos de una conformación a otra. En términos de un vector $\boldsymbol{s}$ de coordenadas colectivas, la PES, también llamada FES en [7] por las siglas en inglés de *free energy surface*, será proporcional a $-1/\beta \ln[p_c(\boldsymbol{x})]$, donde $p_c(\boldsymbol{x})$ es la distribución probabilística de $\boldsymbol{x}$. Por ejemplo, en el dipéptido de alanina, la FES es una función de dos ángulos (figura 2).

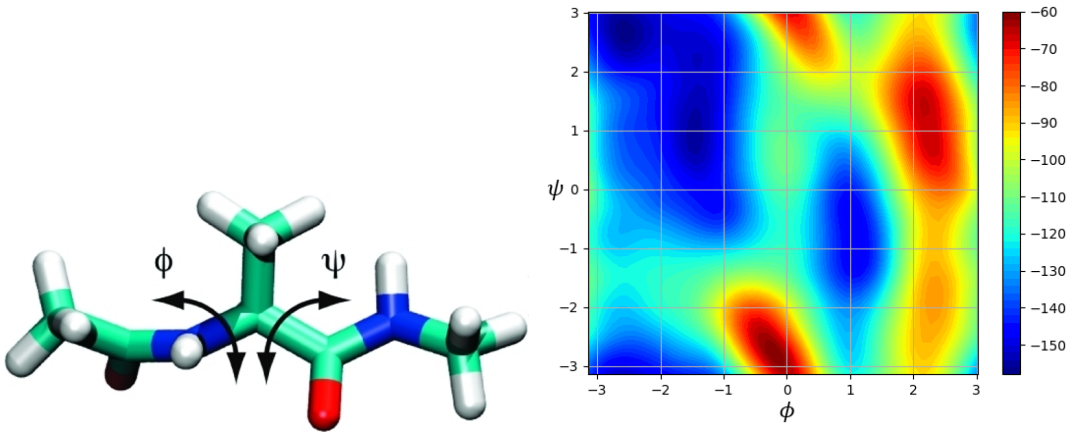

Figura 2: Dipéptido de alanina con su FES asociada a los ángulos ϕ y ψ. Imagen cortesía de [8]

La obtención de la FES asociada a una proteína permite conocer sus distintas conformaciones así como los cambios físicos característicos entre ellas [9].

Sin embargo, la transición entre conformaciones usualmente ocurre en escalas de milisegundos o incluso mayores. Esto representa una de las mayores limitantes en simulaciones moleculares, pues usualmente éstas se ven restringidas a reproducir procesos del orden de nanosegundos y microsegundos [10]. En la actualidad, se han ideado diversas técnicas computacionales de muestreo

acelerado para resolver estas dificultades. Entre ellas se encuentran el método de inundación [11], dinámicas moleculares gausianas aceleradas (GaMD, por las siglas en inglés de *Gaussian accelerated Molecular Dynamics*) [12], metadinámica [13] y el método variacional [14]. Los cuatro algoritmos se basan en la modificación del potencial de la proteína (de manera que se facilita la salida de una conformación), pero cada uno tiene distintas características en su metodología.

En este trabajo se realizaron comparaciones cualitativas y cuantitativas de estos cuatro métodos de muestreo acelerado con el objetivo de determinar las ventajas y desventajas de cada uno. Con este fin, se implementaron dichas técnicas en potenciales previamente construidos (en una y dos dimensiones), y se analizaron las reconstrucciones obtenidas con cada método. Dicho análisis consistió en determinar el número de pasos necesarios para salir de la conformación inicial y para muestrear el espacio de configuraciones, errores cuadráticos respecto al potencial real y la reconstrucción de los caminos de transición entre conformaciones.

Organización de la tesis

A continuación se presenta un breve resumen del contenido de los capítulos del presente trabajo con la finalidad de guiar al lector.

§1 Modelado de dinámicas moleculares

La construcción del espacio fase que caracteriza a una proteína se hace con base en el formalismo de la mecánica estadística. En esta sección se presenta una breve introducción del mismo, además de que se describen la parametrización de los potenciales involucrados en la evolución de una proteína, así como las distintas ecuaciones que describen esta dinámica.

§2 Métodos de muestreo acelerado

Uno de los mayores problemas en la simulación de dinámicas moleculares es la falta de muestreo del espacio fase debido a limitaciones computacionales, por lo que se han ideado distintas técnicas que aceleran dichos muestreos. Esta sección está dedicada a la descripción de los cuatro métodos que se analizaron en este trabajo: metadinámica, GaMD, inundación y método variacional.

§3 Modelos de Markov

Con la finalidad de tener más herramientas para comparar las metodologías presentadas previamente, se introduce en esta sección el modelado del espacio fase de una proteína con cadenas de

Markov. Después, se establece un método para delimitar los conjuntos metaestables en este espacio a partir del análisis espectral de la matriz de transición del sistema. Finalmente, se presentan las probabilidades de compromiso, con las que se obtiene un *mapa probabilístico* de las transiciones entre un par de conformaciones de la proteína.

§4 Resultados y discusión

En esta sección se presentan los cuatro potenciales (2 en 1D y 2 en 2D) sobre los cuales se implementaron cada uno de los métodos de muestreo acelerado. En cada caso se hace un análisis de los conjuntos metaestables del potencial, se presentan las reconstrucciones y probabilidades de compromiso de cada método de aceleración y tablas con los datos de pasos en la dinámica así como errores cuadráticos.

Posteriormente, se discuten las ventajas y desventajas de cada método con base en los resultados obtenidos, para después realizar una comparación entre ellos.

§5 Conclusiones y apéndices

Aquí se presentan las conclusiones del trabajo, así como las posibilidades para futuro trabajo de investigación. En el primer apéndice se desarrolla el método de ponderación energética con expansión al segundo orden, que es utilizado por los algoritmos de inundación y GaMD. En el segundo apéndice se muestra el código de las funciones principales utilizadas en 1D; el resto del código, junto con las funciones análogas para 2D, se pueden encontrar en `https://github.com/marciniega/accelesamp`.

Capítulo 1

Modelado de dinámicas moleculares

Las proteínas son biomoléculas flexibles que tienen distintas estructuras geométricas o *conformaciones* asociadas a ellas. Además, las características funcionales de una proteína dependen de la conformación en la que se encuentre. [15]
A la transición de una estructura a otra se le conoce como *transición conformacional*. Usualmente, las proteínas se mantienen en cada conformación durante largos periodos de tiempo (comparados con movimientos moleculares típicos, que suceden en el rango de 10^{-15} a 10^{-9} segundos) antes de sufrir una transición. Es decir, los diferentes estados conformacionales de una proteína son conjuntos metaestables en el espacio de configuraciones que caracteriza a la misma [9].

1.1. Mecánica estadística

El objetivo de esta rama de la física es establecer relaciones entre las propiedades macroscópicas de un sistema a partir del estudio estadístico de sus componentes microscópicos. Así, cantidades termodinámicas como presión, temperatura, etc., se identifican con el comportamiento estadístico de los átomos que componen al sistema. Para cuantificar dicho comportamiento, primero se asigna un espacio muestral S al sistema, así como un espacio de probabilidad (S, F, P).

Tomando un sistema cerrado con N átomos, una forma de construir S es asignando un estado a cada configuración de posición y momento:

$$x \in S \Leftrightarrow x = (r_1, ..., r_N, p_1, ..., p_N), \tag{1.1}$$

donde r_i y p_i son la posición y momento del átomo i, respectivamente, medidos desde algún marco de referencia inercial. Como la posición y la velocidad son cantidades que cubren un continuo de valores, resulta conveniente considerar variables aleatorias (v.a.) definidas por cada r_i y p_i, pues entonces se pueden considerar probabilidades de encontrar al átomo i en $[r_i, r_i + dr_i]$ con momento

en $[p_i, p_i + dp_i]$. Para N átomos en tres dimensiones se tendrán $2N$ vectores de variables aleatorias con una densidad de probabilidad

$$f(\boldsymbol{r}, \boldsymbol{p}) = f(r_1, ..., r_N, p_1, ..., p_N). \tag{1.2}$$

En mecánica estadística, a cada elemento de S definido por (1.1) se le conoce como *microestado* del sistema, y a S como su espacio de configuraciones.

LA densidad de probabilidad f se construye a partir del principio de conservación de energía E. Para un sistema cerrado y aislado, las únicas interacciones posibles son entre las partículas que lo constituyen, y, al no poder recibir ni ceder energía a otros sistemas, E tiene que ser una constante en el tiempo. Esto implica que no hay manera de distinguir un microestado de otro, y se hace la suposición de que todos los microestados que puede tomar el sistema con energía E son igual de probables.

Una colección de sistemas aislados e idénticos se llama *ensamble microcanónico*. Como todos tienen la misma energía constante, los microestados con esta energía son igual de probables en cada uno. Cuando el sistema se encuentra aislado, tendrá una energía constante. Sin embargo, puede que se encuentre en contacto térmico con otros sistemas o con un *depósito de calor*. La colección de sistemas idénticos que se encuentran en equilibrio térmico con un depósito de calor es un *ensamble canónico* (figura 1.1).

Figura 1.1: Un ensamble canónico es un conjunto de sistemas idénticos que se encuentran en equilibrio térmico con un depósito de calor.

Como ahora el sistema puede intercambiar energía con el depósito de calor, se tiene que conocer $f(\boldsymbol{r}, \boldsymbol{p})$ para calcular las probabilidades de que se encuentre en un microestado en particular. Se puede demostrar que esta distribución es de la forma [16]

$$f(\boldsymbol{r}, \boldsymbol{p}) = C^{-1} e^{-\beta E(\boldsymbol{r}, \boldsymbol{p})}, \tag{1.3}$$

en donde β es una constante asociada al depósito de calor. Tanto para energías continuas como discretas, la distribución de energía está dada por (1.3). La constante C^{-1} se obtiene imponiendo

la condición de normalización en cada posibilidad, y recibe el nombre de función de partición Z:

$$\text{energía discreta} \qquad Z = \sum_k e^{-\beta E_k} \tag{1.4}$$

$$\text{energía continua} \qquad Z = \int_{\mathbb{R}^N}\int_{\mathbb{R}^N} d^3r_1 \dots d^3r_N d^3p_1 \dots d^3p_N \, e^{-\beta E(\boldsymbol{r},\boldsymbol{p})}. \tag{1.5}$$

Cabe destacar que la probabilidad $P(E_0)$ de que el sistema tenga una energía E_0 será la suma de contribuciones de todos los microestados con esa energía, es decir,

$$P(E_0) = \int\limits_{E(\boldsymbol{r},\boldsymbol{p})=E_0} d^3r_1 \dots d^3r_N d^3p_1 \dots d^3p_N \, f(\boldsymbol{r},\boldsymbol{p}). \tag{1.6}$$

Con la medida (1.3) sobre la sigma álgebra F de Borel de S se construye el espacio de probabilidad (S, F, P). Al identificar operaciones en este espacio con variables termodinámicas se pueden calcular distintas cantidades físicas.

Al tener un ensamble estadístico de sistemas idénticos que tienen la misma distribución probabilística, las variables termodinámicas macroscópicas del sistema serán el promedio en el ensamble [17]. Por ello, la energía interna U del sistema se identifica con el valor esperado de E; en el caso de energías discretas, esto se expresa mediante

$$U = \mathbb{E}\,[E] = \sum_k E_k P(E_k) = Z^{-1} \sum_k E_k e^{-\beta E_k}, \tag{1.7}$$

donde la función de partición Z, dada por (1.4), depende de β. Al derivarla se obtiene

$$\frac{\partial Z}{\partial \beta} = \sum_k \frac{\partial}{\partial \beta} e^{-\beta E_k} = -\sum_k E_k e^{-\beta E_k},$$

dividiendo esta derivada entre Z se obtiene el lado derecho de (1.7) con un signo negativo, al igualar ambas ecuaciones se llega a

$$Z^{-1}\frac{\partial Z}{\partial \beta} = -U.$$

Como $Z^{-1}\partial Z/\partial\beta = \partial(\ln Z)/\partial\beta$, esta ecuación es equivalente a

$$U = -\frac{\partial(\ln Z)}{\partial \beta}. \tag{1.8}$$

Con energías continuas se obtiene la misma ecuación, pues sólo se sustituyen sumas por integrales. De aquí se sigue que U se puede calcular una vez que se conoce la función de partición del sistema.

Luego, como lo único que cambia de un depósito a otro es su temperatura, se debe de tener $\beta = \beta(T)$. En [16] se demuestra que

$$\beta = \frac{1}{kT}, \tag{1.9}$$

donde k es la constante de Boltzmann. Análogamente, se pueden calcular otras variables termodinámicas. Por ejemplo, de las dos primeras leyes de la termodinámica se sabe que

$$dU = T\,dS_e - p\,dV, \tag{1.10}$$

donde S_e, p son la entropía y presión en el sistema, respectivamente. Tomando la derivada total de (1.7) e igualando cada término diferencial, se concluye que

$$\begin{aligned} dS_e &= kd(\ln Z + \beta U) \\ p &= kT\left(\frac{\partial \ln Z}{\partial V}\right). \end{aligned} \tag{1.11}$$

Para estudiar las diferentes conformaciones de una proteína, así como los cambios geométricos y dinámicos en las transiciones de una a otra, se hacen simulaciones por computadora para obtener el espacio fase (S, F, P).

1.2. Dinámicas moleculares y de Langevin

Los aminoácidos son moléculas orgánicas que consisten en un átomo de carbono central (alfa) unido a un grupo un amino (NH_2), un grupo carboxilo (COOH), un hidrógeno y una cadena lateral (R) (figura 1.2). Dos o más aminoácidos pueden formar una cadena mediante la formación de un enlace covalente entre sus grupos amino y carboxilo; a este arreglo se le da el nombre de proteína, y cumple diversas funciones en un organismo, que incluyen acelerar reacciones químicas, replicar ADN y transporte de moléculas, entre otras [18].

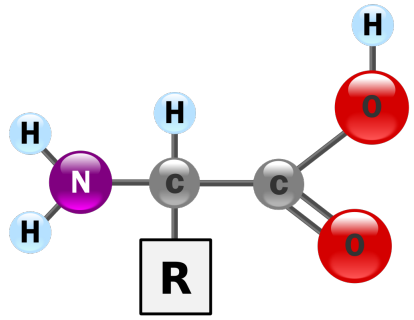

Figura 1.2: Un aminoácido consiste en un carbono alfa unido a un grupo amino, un grupo carboxilo, un hidrógeno y una cadena lateral. Imagen cortesía de [19].

Experimentalmente, es posible medir la temperatura de un sistema, así como conocer la estructura geométrica, composición atómica y otras propiedades importantes de las proteínas. Sin embargo, las transiciones conformacionales son difíciles de examinar y entender sólo mediante resultados experimentales. Por ello, para conocer la evolución dinámica en estos sistemas de escala molecular, se han diseñado simulaciones por computadora que permiten hacer una conexión cuantitativa con la parte experimental.

En dichas simulaciones se utilizan aproximaciones que toman en cuenta diversas interacciones y evalúan la energía potencial del sistema como función de las coordenadas de cada átomo. En la mayoría de los casos, estas aproximaciones clásicas son efectivas, aunque no reproducen efectos cuánticos como la formación o rompimiento de enlaces [20].

Los campos clásicos se dividen en dos clases. Las *interacciones de enlace* incluyen estiramientos en enlaces covalentes, vibraciones en ángulos y potenciales de torsión (figura 1.3). Por otro lado, las *interacciones de largo alcance* consisten en repulsiones y dispersiones de Lennard-Jones, así como en interacciones electrostáticas de Coulomb.

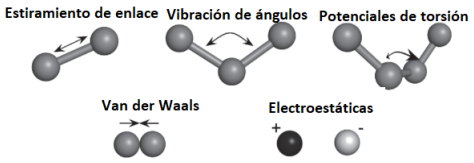

Figura 1.3: Representación gráfica de los distintos tipos de interacciones que se toman en cuenta para calcular el potencial clásico utilizado en simulaciones moleculares. Imagen cortesía de [21]

Formalmente, en una proteína con N átomos la energía potencial del sistema es $U = U_{\text{enlace}} + U_{\text{la}}$, que corresponden a las interacciones de enlace y las de largo alcance, respectivamente. Para el

estiramiento de enlaces covalentes se utiliza una aproximación de oscilador armónico con respecto a la separación $r_{ij} = |r_i - r_j|$ de dos átomos que tienen un enlace entre ellos. Para las otras interacciones de enlace, se toman en cuenta ángulos ϕ formados por cada tercia de átomos, además de ángulos diedros w formados por cuatro átomos (figura 1.4). Mientras que los potenciales de los ángulos ϕ se aproximan como osciladores armónicos, el modelo para los diedros es un potencial de la forma $K_\omega[1 + \cos(n\omega - \delta)]$, donde K_ω, n y δ son parámetros libres [22]. Combinando todas estas contribuciones, el potencial que corresponde a las interacciones de enlace es

$$\begin{aligned} U_{\text{enlace}} = &\sum_{\phi} K_\phi(\phi - \phi_0)^2 \\ &+ \sum_{\omega} K_\omega \left[1 + \cos(n\omega - \delta)\right] \\ &+ \sum_{\text{enlaces}} K_b(r_{ij} - r_0)^2, \end{aligned}$$

donde los valores de las constantes K_ω, n, δ, K_ϕ, K_b y ϕ_0, r_0 depende de la proteína con la que se esté trabajando [23].

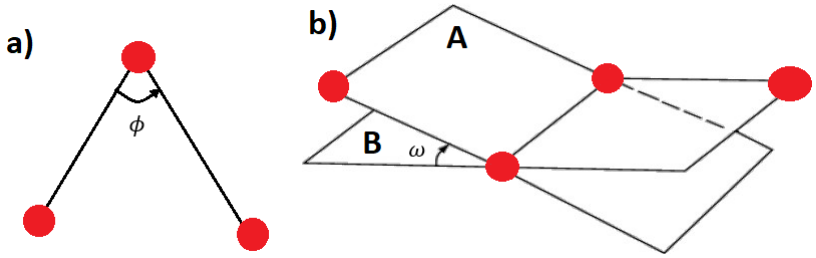

Figura 1.4: Distintos ángulos que se toman en cuenta en las interacciones de enlace de una proteína: a) Una tercia de puntos define un ángulo ϕ. b) Cuatro puntos definen dos planos (A y B), el ángulo entre dichos planos es un diedro.

La primera interacción de *largo alcance* corresponde a repulsiones y dispersiones entre átomos (interacciones de Van der Waals), y se modelan mediante el potencial de Lennard-Jones.

$$U_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]. \tag{1.12}$$

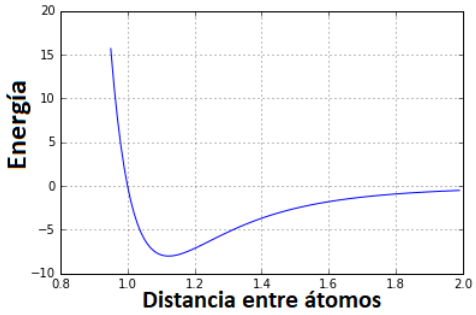

Figura 1.5: Gráfica típica del potencial de Lennar-Jones. En este caso $\sigma = 1$ y $\epsilon = 7.5$

De esta manera, al incluir también el término correspondiente a interacciones electrostáticas, el potencial de interacciones de largo alcance es

$$U_{la} = \sum_{j>i} 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] + \sum_{j>i} \frac{Q_i Q_j}{4\pi\epsilon_0 r_{ij}}, \tag{1.13}$$

donde las sumas son sobre $i = 1 \dots N$, ϵ_0 es la permitividad del vacío y Q_i es la carga eléctrica del átomo i.

Dados el potencial total y la fuerza $(-\nabla U)$ para los N átomos, se hacen integraciones numéricas para obtener una simulación dinámica de la proteína. Mientras que en el método de dinámicas moleculares se utilizan las ecuaciones de movimiento de Newton, en las dinámicas de Langevin se añaden un término estocástico, así como una fuerza de fricción:

$$m_i \ddot{r}_i = F_i - \gamma \dot{r}_i + \sqrt{2\beta^{-1}} dW, \tag{1.14}$$

en donde γ es la fricción y $W(t)$ una variable aleatoria con distribución normal $N(0, 1)$ para cada t, llamada proceso de Wiener [24]. La diferencia entre estos métodos es que una dinámica de Langevin reproduce el comportamiento de un sistema en contacto con un depósito de calor a temperatura constante, mientras que en una dinámica molecular se mantiene una energía constante en el sistema. Por otro lado, en el límite $t \to \infty$ la ecuación de Langevin se reduce a [25]

$$0 = F_i - \gamma \dot{r}_i + \sqrt{2\beta^{-1}} dW, \tag{1.15}$$

que es la ecuación de Langevin con sobreamortiguamiento. En este caso límite (llamado también dinámica Browniana) se asume que la distribución de las velocidades es de Maxwell, y que ya se hizo el promedio sobre las mismas. En condiciones de equilibrio térmico, las dinámicas de Langevin y Browniana convergen al mismo equilibrio [25]. Por ello, al resolver numéricamente las ecuaciones de

Langevin con sobreamortiguamiento, se obtiene la distribución probabilística del espacio canónico, a partir de la cual se obtiene la FES de la proteína.

Con notación vectorial, las ecuaciones de Langevin en forma adimensional ($\gamma = 1$) se pueden expresar como

$$d\boldsymbol{r} = -\nabla U(\boldsymbol{r})dt + \sqrt{2\beta^{-1}}dW, \tag{1.16}$$

Bajo ciertas condiciones de diferenciabilidad y continuidad de U, se puede demostrar que (1.16) va a reproducir una medida de probabilidad cuya densidad es la misma que la de un ensamble canónico [26]. Sin embargo, como las dinámicas de Langevin están descritas mediante ecuaciones diferenciales estocásticas, no se pueden integrar numéricamente de manera usual, sino que se tiene que tomar en cuenta la naturaleza aleatoria del proceso de Wiener. En la actualidad, existen diferentes algoritmos de integración que introducen reglas de selección y números aleatorios para reproducir la densidad de probabilidad límite de una ecuación diferencial estocástica de manera asintótica [27]. Uno de ellos es el método de Euler–Maruyama. Para implementar dicho algoritmo, primero se discretiza el tiempo de simulación T como $T = nh$, donde n es el número de pasos de tiempo de magnitud h. Después, se hace la aproximación discreta de (1.16) como

$$r_{k+1} = r_k - h\nabla U(r_k) + \sqrt{2\beta^{-1}}W_{k+1}, \tag{1.17}$$

con la notación $r_k = r(t_k)$ y $W_{k+1} = W(t_{k+1})$. A partir de (1.17) se obtiene una distribución que describe la probabilidad de que un estado x evolucione a y en un paso de tiempo h. Dicha distribución está dada por [28]

$$q_h(x, y) = (4\pi\beta^{-1}h)^{-n/2} \exp\left(-\frac{|y - x + h\nabla U(x)|^2}{4\beta^{-1}h}\right). \tag{1.18}$$

La estabilidad de Euler-Maruyama depende del potencial en cuestión, mientras en algunos casos se obtiene una distribución q_h que converge a la del espacio canónico, en otros no se va a converger a ninguna distribución estacionaria [29].

Una variación de este algoritmo que asegura la convergencia de q_h es el método conocido como Euler-Maruyama Metropolizado (MALA, por sus siglas en inglés), el cual consiste en calcular el paso siguiente dado por (1.17) pero aceptarlo o rechazarlo basado en q_h [30]. Esto es, en MALA se toma una variable aleatoria η_k con distribución uniforme $U(0, 1)$ para cada $k = 0, ..., n$. Después se propone el paso r^*_{k+1} de acuerdo al método de Euler-Maruyama, y se decide si se avanza o no con base en la probabilidad

$$\alpha_h(x, y) = \text{mín}\left(1, \frac{q_h(y, x)e^{-\beta U(y)}}{q_h(x, y)e^{-\beta U(x)}}\right). \tag{1.19}$$

Es decir, el siguiente paso se define como

$$r_{k+1} = \begin{cases} r^*_{k+1} & \text{si } \eta_k < \alpha_h(r_k, r^*_{k+1}) \\ r_k & \text{en otro caso .} \end{cases} \tag{1.20}$$

La convergencia de la distribución mediante este método se demuestra en [31]. Aunque MALA puede no converger para funciones que no son Lipschitz globales, los potenciales típicos que se utilizan en física son lo suficientemente *suaves* para cumplir los requerimientos de diferenciabilidad y continuidad.

Así, mediante la integración numérica de Langevin se obtiene la PES/FES de la proteína. Es común proyectar esta FES en un espacio coordenadas colectivas $x = (x_1, \ldots, x_n)$ que describen los cambios conformacionales significativos que permiten distinguir un estado de otro. Por ejemplo, en el caso del dipéptido de alanina se suele hacer la proyección sobre dos ángulos.

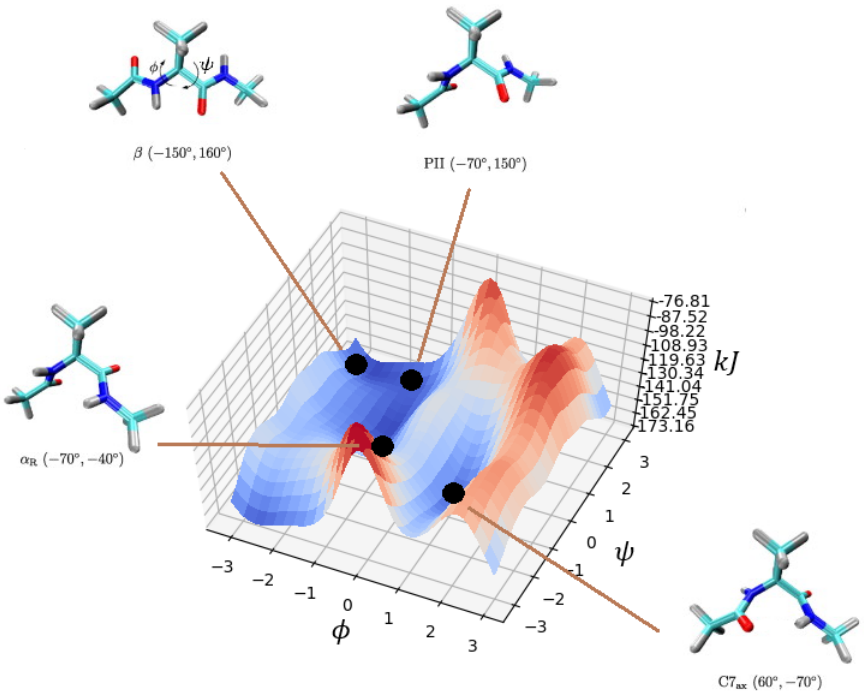

Figura 1.6: PES del dipéptido de alanina proyectada en los ángulos ϕ y ψ. Se observan cuatro conformaciones distintas de esta molécula.

Aunque en teoría se puede obtener la superficie energética de una proteína mediante una simulación lo suficientemente larga, en la práctica una de las dificultades más grandes al momento de

muestrear la FES es que las dinámicas de Langevin (dinámica molecular, brownianas) no alcanzan a simular escalas de tiempo relevantes (figura 1.7). Esto se debe a que los valores de longitudes y fuerzas en cada simulación obligan a que h sea del orden de 10^{-15} segundos, por lo cual, simular procesos bioquímicos de interés puede tardar cientos de años en una computadora de escritorio [32].

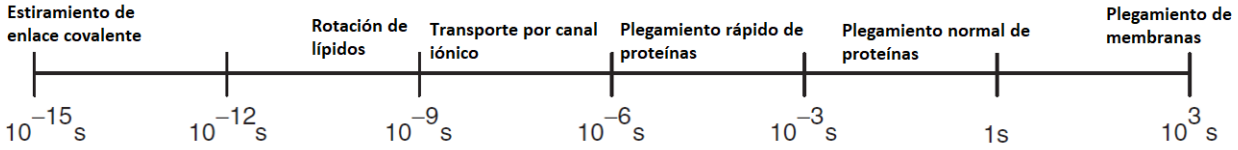

Figura 1.7: Escala de tiempo de diversos procesos bioquímicos.

Lo que sucede usualmente es que la dinámica se estanca en algún mínimo local de la FES durante un largo tiempo antes de hacer una transición a otros sectores (figura 1.8).

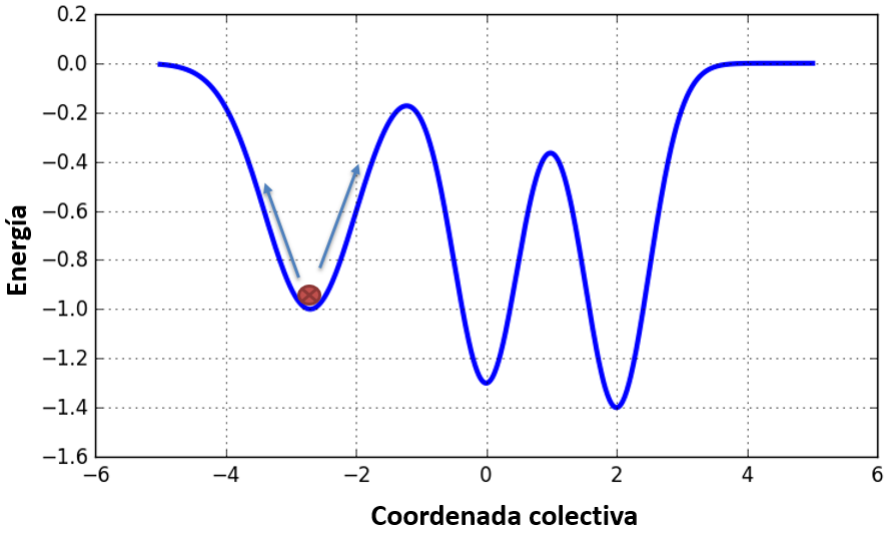

Figura 1.8: En una dinámica usual, la trayectoria queda estancada en un mínimo local durante un gran lapso de tiempo antes de continuar a otros sectores de la FES.

En la actualidad se han desarrollado diversas técnicas computacionales de muestreo acelerado para resolver estas dificultades. Entre ellas se encuentran el método de inundación, dinámicas moleculares gausianas aceleradas (GaMD, por las siglas en inglés de *gaussian accelerated molecular dynamics*), metadinámica y el método variacional.

Capítulo 2

Métodos de muestreo acelerado

El muestreo efectivo de la superficie de energía libre se ha convertido en uno de los problemas principales en la simulación de dinámicas moleculares [15]. Aunque en la actualidad existe hardware construido especialmente para alcanzar escalas temporales del orden de milisegundos de simulación molecular [33], estas herramientas, en general, no se encuentran disponibles. En consecuencia, en las últimas décadas se han desarrollado diversos algoritmos de muestreo acelerado.

En este trabajo se analizarán cuatro de métodos: potencial de inundación, dinámicas gaussianas aceleradas, metadinámica y el método variacional.

2.1. Método de inundación [34]

En este algoritmo sólo se supone el conocimiento previo de un mínimo de energía en el cual la dinámica se queda estancada durante un largo tiempo. La idea básica de inundación es desestabilizar este mínimo para lograr visitar otros estados de la FES. Como se muestra en la figura 2.1, esto se consigue por medio de un potencial de inundación ΔU que afecta a la FES de manera local sin perturbar los caminos de transición originales. Por tanto, disminuyen las barreras de energía entre lo que era el mínimo local y el resto del espacio de energías, con lo que incrementa la probabilidad de que la dinámica explore otros estados.

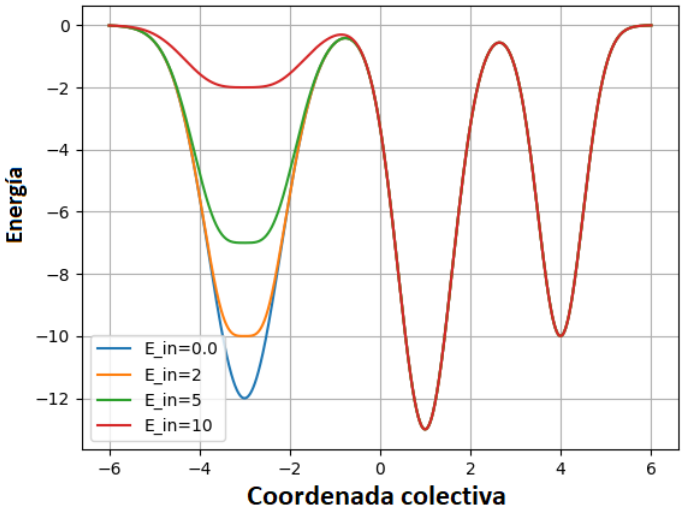

Figura 2.1: En el método de inundación se añade localmente un potencial ΔU que incrementa la probabilidad de transición de la dinámica de Langevin. La magnitud de ΔU depende de la intensidad de inundación E_{in}

Una ventaja de este método es que no se necesita información *a priori* sobre los caminos de transición, ni de otros conjuntos metaestables aparte del asociado al ducto. Esto implica que la distribución del canónico considerando la FES modificada con ΔU será igual al canónico original a excepción de un cambio local en el mínimo de energía donde se empieza.

El algoritmo de inundación tiene dos pasos principales. En primer lugar, se aproxima el ducto conocido con un potencial cuasi-armónico. Después, se construye el potencial de inundación a partir de esta aproximación, con lo cual, se eleva el mínimo local sin afectar otras regiones de mayor energía, en particular, las barreras que rodean al pozo, y que determinan las transiciones en S.

En términos de coordenadas colectivas, la aproximación cuasi-armónica de la FES se define como

$$F(x_1, ...x_n) = \frac{1}{2} kT \sum_{j=1}^{m} \lambda_j (x_j - x_j^0)^2, \tag{2.1}$$

donde $x^0 = (x_1^0, ..., x_n^0)$ son las coordenadas del mínimo del ducto, k es la constante de Boltzmann, T la temperatura del medio, y las variables λ_i determinan el ancho de la aproximación en cada eje coordenado. Para estimar los valores de λ_i, se aproxima la anchura del ducto mediante un promedio de amplitudes:

$$\lambda_j^{-1} = \alpha^2 \mathbb{E}((x_j - x_j^0)^2), \tag{2.2}$$

que se puede calcular, por ejemplo, mediante una dinámica molecular corta. Aquí, el parámetro α se incluye para ajustar la aproximación cuando el muestreo no es tan grande como para obtener un cálculo de fluctuaciones que se ajuste al promedio real.

Luego, el potencial de inundación se define como

$$\Delta U(x_1, ..., x_n) = E_{\text{in}} \exp\left[-\frac{kT}{2E_{\text{in}}} \sum_{j=1}^{m} \lambda_j (x_j - x_j^0)^2\right], \tag{2.3}$$

que se construye de tal manera que tiene forma gausiana con desviación estandar proporcional a las fluctuaciones térmicas en F. La intensidad de inundación E_{in} determina el ancho y la altura del potencial que se va a agregar a la FES.

Ya que se ha calculado el potencial de inundación, se realizan nuevas dinámicas sobre $U(x) + \Delta U(x)$, que es el potencial original más la inundación. Aunque la distribución obtenida de esta manera tendrá cambios respecto a la original, se puede hacer una ponderación para regresar a la distribución del canónico. Aquí se utilizó el método de ponderación energética con expansión acumulada a segundo orden, mismo que se detalla en el apéndice.

2.2. GaMD [12]

El método de dinámicas moleculares aceleradas (aMD, por las siglas en inglés de *accelerated molecular dynamics*) es una técnica de muestreo que funciona mediante la adición de un potencial de impulso ΔU a toda la FES con el objetivo de disminuir las barreras de energía entre cada par de conjuntos metaestables y, por lo tanto, aumentar la probabilidad de que hayan transiciones entre todos ellos. Como en el caso de inundación, se presupone que el uso de un potencial de impulso no cambiará las transiciones originales. Además de que cualquier cambio introducido en la distribución podrá ser retirado mediante técnicas de ponderación.

Sin embargo, cuando las proteínas son demasiado grandes, el error (o ruido energético) al momento de hacer la ponderación crece tanto que ya no es posible recuperar la forma original de la FES. Por otro lado, estudios recientes muestran que si el potencial de impulso tiene una distribución normal, el método de expansión acumulada a segundo orden consigue disminuir el error en las ponderaciones de manera significativa [35]. Por lo tanto, el algoritmo de dinámicas gausianas aceleradas (GaMD) es una modificación de aMD que disminuye el ruido energético mediante la construcción de un potencial de impulso que se aproxima, a primer orden, a una gausiana.

Como se va a aplicar el impulso en toda la FES, primero se escoge un umbral energético E de tal manera que, cuando el potencial $U(x)$ es menor que este valor, se añadirá un factor

$$\Delta U(x) = \frac{1}{2} k(E - U(x))^2, \quad U(x) < E, \tag{2.4}$$

donde k es la constante de fuerza armónica. El potencial modificado $U^*(x) = U(x) + \Delta U(x)$ es

$$U^*(x) = U(x) + \frac{1}{2}k(E - U(x))^2. \tag{2.5}$$

Cuando el potencial está por encima del umbral ($U(x) \geq E$), el potencial de impulso se hace igual a cero, es decir, $U^*(x) = U(x)$.

Así, sólo se tienen que determinar los valores de k y E que permitan una mejor exploración de la FES, sin introducir una gran cantidad de ruido energético al momento de hacer la ponderación. La estimación de estos parámetros se hace con base en ciertas condiciones que acotan a ΔU, así como en datos que se obtienen de una dinámica corta sobre $U(x)$.

Los datos que se utilizarán de dicha dinámica corta son los valores extremos $U_{\min}$ y $U_{\max}$ que corresponden al mínimo y máximo del muestreo obtenido, respectivamente, el valor promedio U_{prom} y su desviación estándar σ_U.

Luego, se busca reducir la diferencia de energía entre diferentes estados de la proteína, sin alterar la forma de la superficie $U(x)$. Por ello, ΔU debe de satisfacer algunas propiedades. Primero, para cualesquiera dos puntos x_1 y x_2, la monotonía de $U^*(x)$ debe conservar la del potencial original. En particular, si $U(x_1) < U(x_2)$, entonces $U^*(x_1) < U^*(x_2)$. Sustituyendo esta relación en (2.5) y despejando E se obtiene

$$E < \frac{1}{2}\left[U(x_1) + U(x_2)\right] + \frac{1}{q}. \tag{2.6}$$

En segundo lugar, si $U(x_1) < U(x_2)$, la diferencia de potencial en la FES modificada debería ser menor que en la original, es decir, $U^*(x_2) - U^*(x_1) < U(x_2) - U(x_1)$. Reemplazando el valor de $U^*(x)$, se tiene otra relación para E:

$$E > \frac{1}{2}\left[U(x_1) + U(x_2)\right]. \tag{2.7}$$

Además, como $U_{\min} \leq U(x_1) < U(x_2) \leq U_{\max}$, la energía umbral E tiene que estar en un rango de

$$U_{\max} \leq E \leq U_{\min} + \frac{1}{k}, \tag{2.8}$$

que se calcula a partir de las dos relaciones obtenidas anteriormente. Utilizando transitividad en las desigualdades de esta ecuación se sigue que

$$U_{\max} \leq U_{\min} + \frac{1}{k}, \tag{2.9}$$

por lo que k cumple

$$k \leq \frac{1}{U_{\max} - U_{\min}}. \tag{2.10}$$

Definiendo $k = k_o(1/(U_{\max} - U_{\min}))$, se tendrá $0 < k_o \leq 1$. Este parámetro determina la magnitud del potencial de impulso (figura 2.2). Mientras k_o aumenta, se van añadiendo impulsos mayores a la FES original.

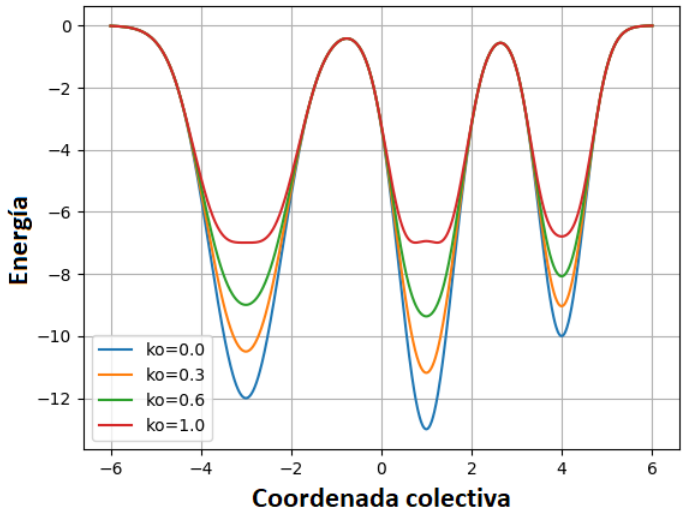

Figura 2.2: En el método de GaMD, el parámetro normalizado k_o determina la magnitud del impulso añadido a $U(x)$ en todo su dominio.

Por último, se requiere que la desviación estándar de ΔU sea lo suficientemente pequeña como para asegurar una ponderación con poco ruido energético; en [36] se calcula

$$\sigma_{\Delta U} = \sqrt{\left(\frac{\partial \Delta U}{\partial U}\bigg|_{U=U_{prom}}\right)^2 \sigma_U^2} = k(E - U_{\text{prom}})\sigma_U \leq \sigma_o, \tag{2.11}$$

donde σ_o es una cota superior que depende del caso particular que se esté estudiando[1]. Despejando k de esta desigualdad se tiene

$$k \leq \frac{\sigma_o}{\sigma_U}\frac{1}{E - U_{\text{prom}}}. \tag{2.12}$$

Luego, suponiendo que E toma su valor mínimo, que es $U_{\max}$, y expresando a k en términos de k_o, se llega a

$$k_o \leq \frac{\sigma_o}{\sigma_U}\frac{U_{\max} - U_{\min}}{U_{\max} - U_{\text{prom}}}. \tag{2.13}$$

[1]Es común fijar este parámetro en $10kT$.

Se define el lado derecho de esta ecuación como $k_o^{'} = (\sigma_o/\sigma_U)((U_{\max} - U_{\min})/(U_{\max} - U_{\text{prom}}))$. De esta manera, para tener la mayor aceleración posible sin introducir tanto ruido energético, se utiliza el máximo k_o posible, que es

$$k_o = \min(1.0, k_o^{'}) = \min\left(1.0, \frac{\sigma_o}{\sigma_U}\frac{U_{\max} - U_{\min}}{U_{\max} - U_{\text{prom}}}\right). \tag{2.14}$$

Análogamente, cuando E se fija en la cota superior, dada por $U_{\min} + \frac{1}{k}$, se calcula

$$k_o \geq \left(1 - \frac{\sigma_o}{\sigma_U}\right)\frac{U_{\max} - U_{\min}}{U_{\text{prom}} - U_{\min}}. \tag{2.15}$$

Se define $k_o^{''} = (1 - (\sigma_o/\sigma_U))((U_{\max} - U_{\min})/(U_{\text{prom}} - U_{\min}))$. Cuando $\sigma_U \leq \sigma_o$, se tiene $k_o^{''} \leq 0$, por lo que k_o puede tomar cualquier valor entre 0 y 1, sin que exista un error significativo al momento de ponderar; siguiendo a [36], en la implementación de este algoritmo se tomó $k_o = 1$.

En su cota superior, E varía de manera inversamente proporcional al valor de k. Por esto, cuando $0 < k_o^{''} \leq 1$, k_o se puede fijar en $k_o^{''}$ para conseguir el mayor umbral de energía E o en la cota 1 para tener la mayor constante armónica k. En este caso, siguiendo al mismo texto, se eligió $k_o = k_o^{''}$. Finalmente, si $k_o^{''} > 1$, se escoge por defecto $k_o = 1$.

Una vez que se han estimado E y k_o, se calcula el potencial de impulso mediante

$$\Delta U(x) = \frac{1}{2}k_o\frac{1}{U_{\max} - U_{\min}}(E - U(x))^2, \quad U(x) < E. \tag{2.16}$$

Como con el algoritmo de inundación, las dinámicas moleculares se harán sobre el potencial modificado, en este caso $U^*(x)$, y después se aplica una ponderación para reconstruir la FES original.

2.3. Metadinámica [13]

En los métodos presentados hasta ahora, se necesita una primera dinámica pequeña para estimar el potencial artificial que se añade a $U(x)$. Después, se realiza una dinámica sobre el potencial perturbado y, al final, se pondera para reconstruir la FES original a partir de la distribución obtenida. Existen otros métodos que, en cambio, van reconstruyendo la superficie energética en cada iteración que realizan, además de que no añaden una perturbación fija al potencial sino que la van construyendo a lo largo de la dinámica. Uno de estos métodos es el de metadinámica.

Metadinámica funciona como una dinámica de Langevin que va eliminando gradualmente la metaestabilidad de cada ducto en el que se estancaría normalmente, además de que hace a la par una reconstrucción de la FES. Para lograr esto, se modifica $U(x)$ con un término dependiente del tiempo que consiste en la suma de gausianas con centro en cada punto de la trayectoria en S explorada por la simulación. Así, la perturbación del potencial está dada por

$$\Delta U(x,t) = w \sum_{t' \leq t} \exp\left[-\frac{(x - x(t'))^2}{2\delta x^2}\right], \tag{2.17}$$

donde w y δx denotan la altura y la anchura de las gausianas, respectivamente, y $x(t')$ denota la posición de la trayectoria al tiempo t'.

La aceleración en metadinámica se basa en que (2.17) va a converger a $-U(x)$, módulo una constante aditiva, en el límite $w/t \to 0$. La explicación cualitativa de esta convergencia es que si la dinámica original se estanca en un mínimo local, la acción del potencial dependiente del tiempo es llenar con gausianas dicho mínimo hasta que la dinámica pueda salir a otros sectores de la FES. Esto se repetirá para cada conjunto metaestable de S, hasta que el potencial modificado $U(x) + \Delta U(x,t)$ se haya *aplanado* (figura 2.3), es decir, hasta que

$$\Delta U(x,t) + U(x) = m, \quad m \in \mathbb{R}. \tag{2.18}$$

Una vez que se cumpla esta igualdad de manera aproximada, las únicas perturbaciones que quedarán en el potencial modificado son aquellas del orden del tamaño de las gausianas añadidas. Así, el parámetro δx determina la resolución de la aproximación de la FES. Por otro lado, en el límite (2.18), la distribución del canónico asociada al potencial total será

$$P(E) = Z^{-1} e^{-\beta m}, \tag{2.19}$$

que es simplemente la distribución uniforme definida en todo S. De aquí se sigue que si metadinámica sigue haciendo iteraciones aún después de haber alcanzado (2.18), lo único que pasará es que se añadirán gausianas de manera uniforme en todo el dominio de la FES, por lo que el valor de m cambiará, pero se seguirá teniendo la igualdad deseada módulo una constante aditiva. La demostración rigurosa de esta convergencia se detalla en [37].

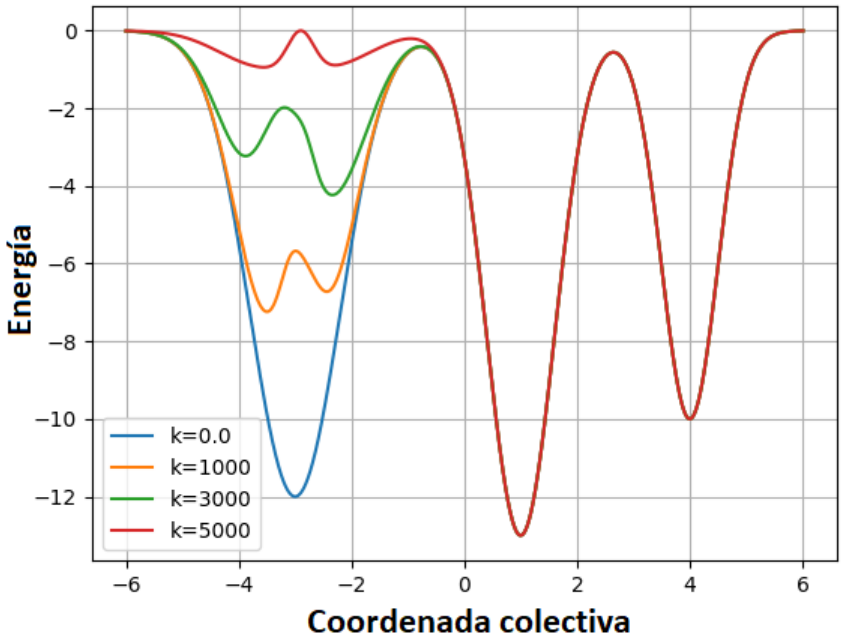

Figura 2.3: La aceleración en metadinámica ocurre porque se van llenando los mínimos locales de energía de la FES conforme pasa el tiempo de simulación. En la imagen, k es el número de pasos transcurridos.

Otra característica importante de metadinámica es que, a diferencia del método de potencial de inundación y GaMD, no necesita hacer una ponderación para recuperar la FES original. Simplemente se toma el negativo de (2.17) cuando terminan las iteraciones. Es por esto que en este algoritmo se aceleran las dinámicas, a la vez que se va estimando la superficie energética asociada a S.

La eficiencia de metadinámica se puede aproximar por el número de gausianas necesarias para aplanar la FES; dicho número es proporcional a $(1/\delta x)^n$, donde n es la dimensionalidad de S. Por lo tanto, δx tiene que ser lo más grande posible para asegurar la convergencia de (2.18) en un tiempo razonable. Por otro lado, este parámetro determina también la resolución de la FES reconstruida, por lo que darle un valor muy grande ocasiona que el sistema salte irregularmente entre mínimos locales, con lo que se pierde información de los caminos reales de transición. Así, se escoge δx de tal manera que se llegue a un equilibrio entre tiempo de cómputo y resolución de la FES.

El otro parámetro libre es la altura de las gausianas w, y se puede calcular de manera aproximada. Como se requiere que al final de las iteraciones la distribución sea la uniforme definida en S, el término de fuerza (en la dinámica de Langevin) que proviene de $\Delta U(x)$ y el original que viene

de $U(x)$ tienen que equilibrarse conforme pasa el tiempo. Así, imponiendo que el valor máximo del potencial artificial de gausianas sea más pequeño que el valor promedio de las fuerzas sobre la FES, se llega a [38]

$$\frac{\sqrt{e}w}{\delta x} = \alpha\sqrt{\mathbb{E}(F^2)}, \tag{2.20}$$

donde $\mathbb{E}(F^2)$ denota el valor esperado del cuadrado de la fuerza en la FES, y α es un parámetro entre 0 y 1.

2.4. Método variacional [14]

Como en el caso de metadinámica, el método variacional construye un potencial artificial de manera adaptativa que acelera el muestreo de S, a la vez que reconstruye la FES. En este caso, se comienza con el funcional

$$\Omega[\Delta U] = \frac{1}{\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\Delta U(x)]}}{\int dx\, e^{-\beta U(x)}} + \int dx\, p(x)\Delta U(x), \tag{2.21}$$

donde las integrales se hacen sobre todo S, p es una distribución arbitraria normalizada y ΔU es una función definida en todo el espacio canónico. De esta manera, el segundo término de esta ecuación es el valor esperado de ΔU bajo la distribución p.

El funcional Ω tiene tres propiedades importantes. En primer lugar, es invariante ante la adición de una función constante $T(x) = c \in \mathbb{R}$. Esto se sigue de

$$\begin{aligned}
\Omega[\Delta U + c] &= \frac{1}{\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\Delta U(x)+c]}}{\int dx\, e^{-\beta U(x)}} + \int dx\, p(x)[\Delta U(x) + c] && (2.22)\\
&= \frac{1}{\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\Delta U(x)]}}{\int dx\, e^{-\beta U(x)}} + \frac{1}{\beta}\log e^{-\beta c}\int dx\, p(x)\Delta U(x) + c \\
&= \Omega[\Delta U] + c - c && (2.23)\\
&= \Omega[\Delta U].
\end{aligned}$$

Es decir, el funcional no cambia cuando se añade una constante arbitraria a $\Delta U(x)$. Luego, utilizando la desigualdad de Cauchy-Schwarz en el espacio de funciones donde se define Ω, se obtiene

$$\begin{aligned}
\left(\int dx\, e^{-\beta[U(x)+\frac{\Delta U_1(x)+\Delta U_2(x)}{2}]}\right)^2 &= \left(\int dx\, e^{-\frac{\beta}{2}[U(x)+\Delta U_1(x)]}e^{-\frac{\beta}{2}[U(x)+\Delta U_2(x)]}\right)^2 && (2.24)\\
&\leq \int dx\, e^{-\beta[U(x)+\Delta U_1(x)]}\int dx\, e^{-\beta[U(x)+\Delta U_2(x)]},
\end{aligned}$$

donde ΔU_1 y ΔU_2 son dos funciones distintas. Dividiendo esta ecuación por $\left(\int dx\, e^{-\beta U(x)}\right)^2$ y tomando el logaritmo en ambos lados de la igualdad se llega a

$$\frac{1}{\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\frac{\Delta U_1(x)+\Delta U_2(x)}{2}]}}{\int dx\, e^{-\beta U(x)}} \leq \frac{1}{2\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\Delta U_1(x)]}}{\int dx\, e^{-\beta U(x)}} + \frac{1}{2\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\Delta U_2(x)]}}{\int dx\, e^{-\beta U(x)}} \tag{2.25}$$

Además, por linealidad de la integral, el segundo termino en $\Omega[V]$ cumple

$$\int dx\, p(x)\left[\frac{\Delta U_1(x)+\Delta U_2(x)}{2}\right] = \frac{1}{2}\int dx\, p(x)\Delta U_1(x) + \frac{1}{2}\int dx\, p(x)\Delta U_2(x), \tag{2.26}$$

Sumando (2.25) con (2.26) resulta

$$\begin{aligned}
&\frac{1}{\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\frac{\Delta U_1(x)+\Delta U_2(x)}{2}]}}{\int dx\, e^{-\beta U(x)}} + \int dx\, p(x)\left[\frac{\Delta U_1(x)+\Delta U_2(x)}{2}\right] \\
&\leq \frac{1}{2}\left[\frac{1}{\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\Delta U_1(x)]}}{\int dx\, e^{-\beta U(x)}} + \int dx\, p(x)\Delta U_1(x)\right] \\
&+ \frac{1}{2}\left[\frac{1}{\beta}\log\frac{\int dx\, e^{-\beta[U(x)+\Delta U_2(x)]}}{\int dx\, e^{-\beta U(x)}} + \int dx\, p(x)\Delta U_2(x)\right],
\end{aligned} \tag{2.27}$$

lo cual es equivalente a

$$\Omega\left[\frac{\Delta U_1+\Delta U_2}{2}\right] \leq \frac{1}{2}\Omega[\Delta U_1] + \frac{1}{2}\Omega[\Delta U_2]. \tag{2.28}$$

Un funcional que cumple (2.28) es convexo [39]; por lo tanto, sólo tiene un punto crítico en todo su dominio, y es un mínimo global (figura 2.4). Para ver la importancia de la convexidad de Ω, se calcula la función que lo minimiza.

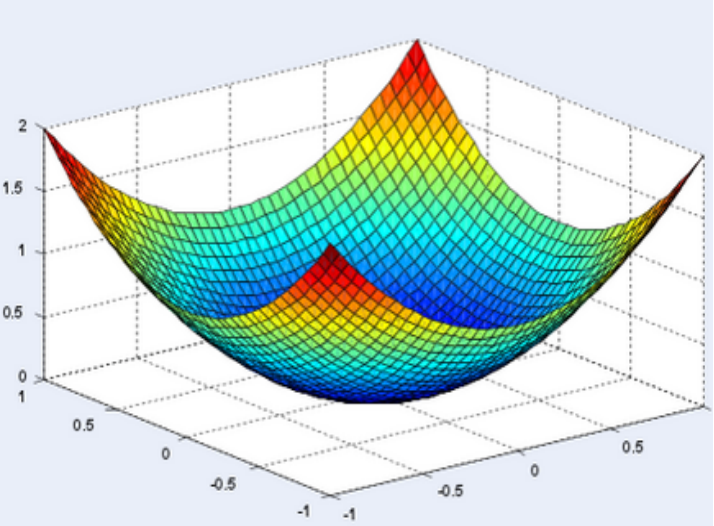

Figura 2.4: Un funcional convexo tiene sólo un punto crítico, que es además su mínimo global.

Tomando la derivada variacional de Ω respecto de ΔU e igualando a cero se llega a

$$0 = \frac{\delta\Omega[\Delta U(x)]}{\delta\Delta U} = \frac{e^{-\beta[U(x)+\Delta U(x)]}}{\int dj\, e^{-\beta[U(j)+\Delta U(j)]}} + p(x); \tag{2.29}$$

despejando ΔU de esta ecuación se calcula el punto (función) crítico, que es

$$\Delta U(x) = -U(x) - \frac{1}{\beta}\log p(x) - \frac{1}{\beta}\log \int dj\, e^{-\beta[U(j)+\Delta U(j)]}. \tag{2.30}$$

De esta manera, escogiendo cualquier distribución $p(x)$, se puede obtener una parametrización de la FES mediante la minimización de Ω. Además, cuando se añade el potencial artificial (2.30) al sistema, la distribución obtenida en la dinámica será $p(x)$. Esto permite escoger sectores particulares de S que se quieran muestrear mediante una elección particular de $p(x)$. En el caso de que no se quiera hacer la reconstrucción de una zona particular de la FES, se puede escoger la distribución uniforme $p = 1/\mathrm{vol}(S)$, donde $\mathrm{vol}(S)$ denota el volumen de S. En ambos casos, se cambia el problema de reconstruir la FES por el de minimizar Ω.

Para poder minimizar Ω numéricamente, primero se escribe el potencial artificial como una función $\Delta U(x, \boldsymbol{\alpha})$ de x y de un vector de parámetros variacionales $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$. Así, se minimizará la función $\Omega(\boldsymbol{\alpha}) = \Omega[\Delta U(x, \boldsymbol{\alpha})]$ con respecto a $\boldsymbol{\alpha}$ utilizando una variación estocástica del método de gradiente descendiente. Una vez que se converga al mínimo global, se puede obtener $U(x)$ módulo una constante directamente de (2.30) y del cálculo numérico del logaritmo de $p(x)$.

En el método de optimización que se utilizará, es necesario conocer las dos primeras derivadas de Ω respecto a $\boldsymbol{\alpha}$; el gradiente se aproxima como [40]

$$\frac{\partial\Omega(\boldsymbol{\alpha})}{\partial\alpha_i} = -\mathbb{E}\left[\frac{\partial\Delta U(x,\boldsymbol{\alpha})}{\partial\alpha_i}\right]_{\Delta U(\boldsymbol{\alpha})} + \mathbb{E}\left[\frac{\partial\Delta U(x,\boldsymbol{\alpha})}{\partial\alpha_i}\right]_p . \tag{2.31}$$

Para la hessiana se tiene

$$\frac{\partial^2\Omega(\boldsymbol{\alpha})}{\partial\alpha_j\partial\alpha_i} = \beta\,\mathrm{Cov}\left[\frac{\partial\Delta U(x,\boldsymbol{\alpha})}{\partial\alpha_j}, \frac{\partial\Delta U(x,\boldsymbol{\alpha})}{\partial\alpha_i}\right]_{\Delta U(\boldsymbol{\alpha})} - \mathbb{E}\left[\frac{\partial^2\Delta U(x,\boldsymbol{\alpha})}{\partial\alpha_j\alpha_i}\right]_{\Delta U(\boldsymbol{\alpha})} + \mathbb{E}\left[\frac{\partial^2\Delta U(x,\boldsymbol{\alpha})}{\partial\alpha_j\alpha_i}\right]_p . \tag{2.32}$$

En estas aproximaciones, el operador $\mathbb{E}\left[\ldots\right]_p$ es el valor esperado bajo la distribución $p(x)$, y los operadores $\mathbb{E}\left[\ldots\right]_{\Delta U(\boldsymbol{\alpha})}$ y $\mathrm{Cov}\left[\ldots\right]_{\Delta U(\boldsymbol{\alpha})}$ denotan al valor esperado y covarianza, respectivamente, obtenidos en un sistema con potencial $U(x)+\Delta U(x,\boldsymbol{\alpha})$. Es decir, para todo par de funciones $f(x)$, $g(x)$ se define

$$\mathbb{E}\left[f(x)\right]_{\Delta U(\boldsymbol{\alpha})} = Z^{-1}\int dx\, e^{-\beta[U(x)+\Delta U(x,\boldsymbol{\alpha})]} f(x) \tag{2.33}$$

$$\mathrm{Cov}\left[f(x),g(x)\right]_{\Delta U(\boldsymbol{\alpha})} = \mathbb{E}\left[f(x)g(x)\right]_{\Delta U(\boldsymbol{\alpha})} - \mathbb{E}\left[f(x)\right]_{\Delta U(\boldsymbol{\alpha})}\mathbb{E}\left[g(x)\right]_{\Delta U(\boldsymbol{\alpha})} . \tag{2.34}$$

Por otro lado, la expresión de $\Delta U(x,\boldsymbol{\alpha})$ se obtiene expandiendo $\Delta U(x)$ en una serie de funciones, donde cada coeficiente es uno de los parámetros variacionales,

$$\Delta U(x,\boldsymbol{\alpha}) = \sum_k \alpha_k G_k(x). \tag{2.35}$$

Como, en general, la FES es una superficie *suave* en S, se puede obtener una buena aproximación de la misma utilizando pocos términos de la expansión de $\Delta U(x,\boldsymbol{\alpha})$. Tomando (2.35) en cuenta, se simplifican las ecuaciones para el gradiente y la hessiana,

$$\frac{\partial\Omega(\boldsymbol{\alpha})}{\partial\alpha_i} = -\mathbb{E}\left[G_i(x)\right]_{\Delta U(\boldsymbol{\alpha})} + \mathbb{E}\left[G_i(x)\right]_p \tag{2.36}$$

$$\frac{\partial^2\Omega(\boldsymbol{\alpha})}{\partial\alpha_j\alpha_i} = \beta\mathrm{Cov}\left[G_j(x),G_i(x)\right]_{\Delta U(\boldsymbol{\alpha})} . \tag{2.37}$$

Las derivadas de Ω están dadas en términos de valores esperados, por lo que se pueden calcular mediante dinámicas cortas sobre el potencial $U(x)+\Delta U(x,\boldsymbol{\alpha})$. Sin embargo, estos cálculos de naturaleza estadística introducen ruidos numéricos cuando se minimiza Ω con métodos deterministas. Como se mencionó anteriormente, en este caso se utiliza un algoritmo de optimización estocástico que es una variación del método del gradiente descendiente [41]. En dicho algoritmo,

se consideran, para la iteración n, la iteración instantánea $\boldsymbol{\alpha}^{(n)}$ y el promedio de las iteraciones $\bar{\boldsymbol{\alpha}}^{(n)} = (n-1)^{-1} \sum_{k<n} \boldsymbol{\alpha}^{(\boldsymbol{k})}$. La nueva solución se actualiza mediante la ecuación de recursión

$$\boldsymbol{\alpha}^{(n+1)} = \boldsymbol{\alpha}^{(n)} - \mu \left[\Omega'(\bar{\boldsymbol{\alpha}}) + \Omega''(\bar{\boldsymbol{\alpha}}^{(n)}) \left[\boldsymbol{\alpha}^{(n)} - \bar{\boldsymbol{\alpha}}^{(n)} \right] \right], \tag{2.38}$$

donde μ es un parámetro libre, y el gradiente y la hessiana se calculan en el promedio de las iteraciones $\bar{\boldsymbol{\alpha}}^{(n)}$, lo cual equivale a tomar una expansión de Taylor a primer orden del gradiente $\Omega'(\boldsymbol{\alpha}^n)$ alrededor de $\bar{\boldsymbol{\alpha}}^{(n)}$. En la práctica, se vuelve muy costoso calcular la matriz hessiana a medida que aumenta el número de funciones en la expansión de $\Delta U(x)$, por lo que sólo se considera su diagonal; aunque no existen cálculos analíticos que determinen el error que se introduce al hacer esto, empíricamente se ha visto que no afecta a la reconstrucción final de la FES [42].

Para conocer qué tan cerca está el algoritmo del mínimo global de Ω, es necesario poder estimar el funcional en cada iteración. Esto se logra escribiendo (2.21) como

$$\begin{aligned} \Omega[\Delta U] &= \frac{1}{\beta} \log \frac{\int dx\, e^{-\beta[U(x)+\Delta U(x)]}}{\int dx\, e^{-\beta[U(x)+\Delta U(x)]} e^{\beta \Delta U(x)}} + \int dx\, p(x) \Delta U(x) \\ &= \frac{1}{\beta} \log \frac{1}{\mathbb{E}\left[e^{\beta \Delta U(x)}\right]_{\Delta U(\boldsymbol{\alpha})}} + \mathbb{E}\left[\Delta U(x)\right]_p \\ &= -\frac{1}{\beta} \log \mathbb{E}\left[e^{\beta \Delta U(x)}\right]_{\Delta U(\boldsymbol{\alpha})} + \mathbb{E}\left[\Delta U(x)\right]_p . \end{aligned} \tag{2.39}$$

Además, sustituyendo el punto crítico (2.30) en la definición de Ω se obtiene el valor mínimo global,

$$\begin{aligned} \Omega[\Delta U] &= -\frac{1}{\beta} \log \int dx\, e^{-\beta U(x)} - \int dx\, p(x) \left[U(x) + \frac{1}{\beta} \log p(x) \right] \\ &= -\frac{1}{\beta} \log Z - \mathbb{E}\left[U(x)\right]_p - \frac{1}{\beta} \mathbb{E}\left[\log p(x)\right]_p . \end{aligned} \tag{2.40}$$

Capítulo 3

Modelos de Markov

Un modelo o cadena de Markov es una red de estados en donde la evolución probabilista hacia un estado particular depende únicamente del estado actual, no de todos los estados que ocurrieron anteriormente.
Al tener la FES de una proteína, se puede utilizar este formalismo para obtener información relevante de la misma. Específicamente, es posible delimitar las conformaciones de una proteína en la FES utilizando conjuntos metaestables, y obtener información sobre la evolución de una conformación a otra a partir de *probabilidades de compromiso*.

3.1. Formalismo de cadenas de Markov

Un proceso estocástico con estados finitos se puede describir mediante una sucesión $X_0, X_1, \ldots,$ de variables aleatorias. Como esta sucesión representa una misma variable m de un sistema que evoluciona en el tiempo, cada una de las v.a. tiene que tomar un valor dentro de un conjunto de valores $\{0, 1, \ldots, M\}$. Se dice que X_n es el estado del sistema al tiempo n y, que al tiempo n, el sistema se encuentra en estado i si $X_n = i$. Esta sucesión de variables aleatorias recibe el nombre de *cadena de Markov* si cada vez que el sistema se encuentre en el estado i, existe una probabilidad $P_{ij}(t)$ de que al siguiente paso de tiempo se encuentre en el estado j. Esto es, el sistema no tiene memoria de las posiciones anteriores, lo cual se puede escribir como

$$P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, ..., X_1 = i_1, X_0 = i_0) = P(X_{n+1} = j | X_n = i) \equiv P_{ij}. \tag{3.1}$$

Los valores P_{ij}, con $0 \leq i \leq M, 0 \leq j \leq M$, reciben el nombre de probabilidades de transición de la cadena de Markov. Por definición, los P_{ij} son el valor de una medida de probabilidad, por lo que cumplen

$$P_{ij} \geq 0 \quad \forall i, j. \tag{3.2}$$

Además, si a un determinado tiempo n el sistema se encuentra en el estado i, al siguiente paso temporal, X_{n+1} forzosamente debe de tomar uno de los posibles estados, con lo cual, para cada i se cumple

$$\sum_{j=0}^{M} P_{ij} = 1. \tag{3.3}$$

Usualmente, se acomodan las probabilidades de transición en una matriz P, definida como

$$P = \begin{pmatrix} P_{00} & \dots & P_{0M} \\ P_{10} & \dots & P_{1M} \\ \vdots & \ddots & \vdots \\ P_{M0} & \dots & P_{MM} \end{pmatrix}, \tag{3.4}$$

llamada matriz de transición. Con esta notación, cada renglón de P está normalizado, debido a (3.3).

Al conocer la distribución de X_0 y la matriz de transición P, se pueden calcular todas las probabilidades de interés. Por ejemplo, la función de densidad conjunta de $X_0, ..., X_n$ está dada por

$$\begin{aligned} &P(X_n = i_n, X_{n-1} = i_{n-1}, ..., X_1 = i_1, X_0 = i_0) \\ &\quad = P(X_n = i_n | X_{n-1} = i_{n-1}, ..., X_0 = i_0) P(X_{n-1} = i_{n-1}, ..., X_0 = i_0) \\ &\quad = P_{i_{n-1}, i_n} P(X_{n-1} = i_{n-1}, ..., X_0 = i_0), \end{aligned}$$

y, continuando de esta manera

$$P(X_n = i_n, ..., X_0 = i_0) = P_{i_{n-1}, i_n} P_{i_{n-2}, i_{n-1}} \dots P_{i_1, i_2} P_{i_0, i_1} P(X_0 = i_0). \tag{3.5}$$

La entrada ij de P representa la probabilidad de que el sistema en estado i cambie al estado j en el siguiente paso de tiempo. De manera análoga, se puede definir la probabilidad de transición en dos niveles, $P_{ij}^{(2)}$, de que el sistema en estado i esté en el estado j después de 2 transiciones, como

$$P_{ij}^{(2)} = P(X_{m+2} = j | X_m = i). \tag{3.6}$$

Esta transición en dos niveles también se puede calcular en términos de las entradas P_{ij} mediante

$$\begin{aligned} P_{ij}^{(2)} &= P(X_2 = j | X_0 = i) \\ &= \sum_{k=0}^{M} P(X_2 = j, X_1 = k | X_0 = i) \\ &= \sum_{k=0}^{M} P(X_2 = j | X_1 = k, X_0 = i) P(X_1 = k | X_0 = i) \\ &= \sum_{k=0}^{M} P_{kj} P_{ik} \\ &= \sum_{k=0}^{M} P_{ik} P_{kj}. \end{aligned}$$

Intuitivamente, se están sumando las contribuciones de todos los posibles caminos en el espacio de estados que existen entre i y j en dos pasos de tiempo. Por otro lado, la última igualdad es simplemente un producto de matrices, y se puede escribir como

$$P_{ij}^{(2)} = \left[P^2\right]_{ij}, \tag{3.7}$$

donde P^2 es la matriz de transición elevada al cuadrado.

En general, se define la probabilidad de transición en n niveles mediante

$$P_{ij}^{(n)} = P(X_{n+m} = j | X_m = i), \tag{3.8}$$

y (3.7) se generaliza mediante inducción a

$$P_{ij}^{(n)} = P_{ij}^{n}, \tag{3.9}$$

que, por conveniencia, suele escribirse de la forma

$$P_{ij}^{(n)} = \sum_{k=0}^{M} P_{ik}^{(r)} P_{kj}^{(n-r)} \quad \forall \quad 0 < r < n. \tag{3.10}$$

Las probabilidades de transición en n niveles, conocidas como ecuaciones de Chapman–Kolmogorov, se utilizan para conocer el comportamiento del sistema en el límite $n \to \infty$ a partir de (3.10). Para ello, se calcula primero

$$\begin{aligned} P(X_n = j) &= \sum_{i} P(X_n = j | X_0 = i) P(X_0 = i) \\ &= \sum_{i} P_{ij}^{(n)} P(X_0 = i), \end{aligned} \tag{3.11}$$

donde el único factor que depende del tiempo es $P_{ij}^{(n)}$. Existe una gran variedad de cadenas de Markov donde este término converge, cuando $n \to \infty$, a un valor Π_j que depende sólo de j. Es decir, cuando ha pasado un determinado tiempo, la probabilidad de estar en un estado j es, aproximadamente, Π_j, sin importar cual fue el estado inicial del sistema. Se puede demostrar [43] que una condición suficiente para que una cadena de Markov tenga esta propiedad es que, para alguna n, se cumpla

$$P_{ij}^{(n)} > 0 \quad \forall \quad i, j = 0, ..., M. \tag{3.12}$$

Las cadenas de Markov que satisfacen (3.12) se llaman cadenas *ergódicas*. En este caso, como las ecuaciones de Chapman-Kolmogorov implican

$$P_{ij}^{(n+1)} = \sum_{k=0}^{M} P_{ik}^{(n)} P_{kj}, \tag{3.13}$$

se sigue que, al tomar límite al infinito sobre n, se cumple

$$\Pi_j = \sum_{k=0}^{M} \Pi_k P_{kj}. \tag{3.14}$$

Además, como $\sum_{j=0}^{M} P_{ij}^{(n)} = 1 \ \forall n$, se obtiene en el límite $n \to \infty$ la normalización de Π:

$$\sum_{j=0}^{M} \Pi_j = 1, \tag{3.15}$$

y se puede demostrar [44] que $\Pi = (\Pi_0, \ldots, \Pi_M)$ es la única solución no negativa de (3.14) y (3.15). Para sistemas ergódicos, a la distribución Π se le conoce como distribución de equilibrio o distribución límite.

3.2. Metaestabilidad mediante cadenas de Markov

Intuitivamente, un conjunto $A \subset S$ es metaestable si el sistema tiene la propiedad de que se mantiene en ese subconjunto un largo periodo de tiempo antes de hacer una transición al resto del espacio de estados. Una manera de formalizar este concepto es con el enfoque de tiempos de salida (figura 3.1). De acuerdo con este enfoque, se dice que A es metaestable si una dinámica que comienza dentro del conjunto no saldrá del mismo aún después de que haya pasado un largo tiempo ω.

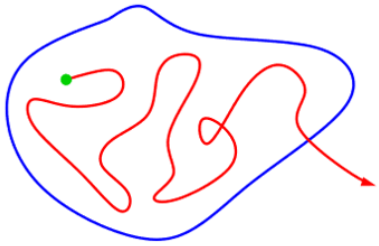

Figura 3.1: En el enfoque de tiempos de salida se considera una dinámica con posición inicial en A durante largo tiempo y se verifica si salió o no del conjunto.

Se busca identificar una familia de conjuntos metaestables $D = \{D_1, \ldots, D_m\}$ que correspondan a todos los arreglos conformacionales de la proteína que se esté estudiando. Esta familia será un subconjunto de la potencia de S, por lo que tiene que cumplir

$$\begin{aligned} &i) \quad \mu(D_k) > 0 \quad \forall k, \\ &ii) \quad \mu(D_k \cap D_l) = 0 \quad \forall k \neq l, \\ &iii) \quad \bigcup_{k=1}^{m} D_k \subset S, \end{aligned} \tag{3.16}$$

donde μ es la medida de probabilidad del canónico. Esto significa que (i) la proteína puede encontrarse en cualquiera de las conformaciones, (ii) que un estado del canónico no puede corresponder a dos arreglos conformacionales diferentes, y (iii) que cualquier arreglo conformacional está contenido en el conjunto de estados posibles. De esta manera, al identificar una descomposición D, se pueden caracterizar por completo todas las conformaciones de una proteína.

Para encontrar esta descomposición es necesario introducir un operador P_t de evolución temporal. Dicho operador actúa sobre cada $x \in S$ y regresa la probabilidad de evolución a cualquiera de los otros estados. Mientras que la definición formal de P_t es como un operador integral que actúa sobre el espacio $L(\mu)$ de variables aleatorias asociadas a μ, es posible reducirlo a [45]

$$P_t = P^t, \tag{3.17}$$

que es simplemente una potencia de la matriz de transición (3.4), misma que puede ser obtenida directamente de la FES. Una vez que se tiene la energía asociada a cada estado discreto i de x, se calcula P con

$$P_{ij} = \frac{\min[1, \exp\left(-\beta(E_j - E_i)\right)]}{\sum_k \min[1, \exp\left(-\beta(E_k - E_i)\right)]}, \tag{3.18}$$

donde E_i, E_j son las energías de los estados i y j, respectivamente. Se utiliza específicamente esta definición ya que, llevada al infinito, reproduce la distribución invariante del canónico [46].

Con base en el enfoque de tiempos de salida, si A es metaestable, la aplicación del operador de evolución a un estado contenido en A dará como resultado otro estado contenido en el mismo conjunto, incluso si se evalúa P_t en un lapso de tiempo $\omega \gg 0$. De esta manera, si 1_A es el vector identidad de A, definido como

$$1_A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{en otro caso} , \end{cases} \tag{3.19}$$

entonces, A es metaestable sí y sólo sí se satisface la ecuación de eigenvalores

$$P_t 1_A = 1_A \qquad \forall \quad 0 < t < \omega. \tag{3.20}$$

Para calcular el número de estados metaestables en los que se descompone S, se analiza el espectro de eigenvalores del propagador P_t. Específicamente, se detectan dichos conjuntos vía los eigenvalores cercanos a 1, contando multiplicidad. Por otro lado, las eigenfunciones asociadas a estos valores, llamadas eigenfunciones dominantes, permiten identificar y delimitar a cada D_j mediante un análisis de aglomeración de estados.

En [45] se muestra que, en las regiones de S correspondientes a conjuntos metaestables, todas las eigenfunciones toman valores aproximadamente constantes. La variación sobre este valor constante fue el criterio utilizado en este trabajo para la aglomeración de estados.

Primero se obtienen las m eigenfunciones dominantes del propagador, que se escriben en términos de eigenvectores $v_1,\ldots,v_m$. Después, se etiquetan los estados d_i que corresponden a mínimos locales en S. En D_i se toma como referencia el valor de cada eigenvector en d_i, y se van buscando estados vecinos donde todos los eigenvectores tengan aproximadamente el mismo valor (salvo un porcentaje de error) que tienen en d_i. Cuando se encuentran vecinos donde el valor de algún eigenvector difiera de la referencia más allá del porcentaje permitido, se termina el algoritmo y todos los estados visitados previamente se toman dentro de la aglomeración.

3.3. Probabilidades de compromiso

Habiendo encontrado al menos dos subconjuntos metaestables A y B en el espacio de estados (los cuales pueden corresponder, por ejemplo, a la proteína plegada en A y desplegada en B), se busca conocer la probabilidad de que, empezando en un estado particular, la proteína evolucione a un subconjunto o al otro. Para ello, se introduce la probabilidad de compromiso, que es una función q_i que va de S a $\mathbb{R}$, y que calcula, para cada $i \in S$, la probabilidad de que el estado i evolucione a B antes que a A.

Por definición, se debe tener $q_i = 0$ para $i \in A$ y $q_i = 1$ si $i \in B$. Mientras tanto, para los estados i que no se encuentren en ninguno de los dos conjuntos, se utiliza la relación que existe entre estados vecinos en S para el cálculo de probabilidades. Esto es, la probabilidad de compromiso para un estado $i \notin A \cup B$ está dada por la suma de productos de las probabilidades de alcanzar un estado vecino j por la probabilidad de compromiso evaluada en j. Por otro lado, cuando $i \in A \cup B$, se fijan condiciones de frontera, de modo que

$$q_i = \begin{cases} 0 & \text{si } i \in A, \\ 1 & \text{si } i \in B, \\ \sum\limits_j P_{ij} q_j & \text{en otro caso.} \end{cases} \tag{3.21}$$

Además, la probabilidad de compromiso inversa q_i^- se define de manera análoga como la probabilidad de que, comenzando en el estado i, se alcance A antes que B. con lo cual, para procesos ergódicos, se cumple

$$q_i + q_i^- = 1. \tag{3.22}$$

Las relaciones de compromiso (3.21) son un sistema lineal de ecuaciones que se pueden resolver numéricamente con cualquier paquete de álgebra lineal. Sin embargo, para sistemas con un gran número de estados, el costo y el error computacional pueden aumentar considerablemente, por lo que se vuelve una manera poco eficiente de calcular las probabilidades de compromiso.

Una alternativa consiste en formular el problema de los compromisos en términos de eigenvalores dominantes [47]. Para ello, primero se construye una matriz de transición con estados absorbentes A y B

$$T_{ij} = \begin{cases} P_{ij} & \text{si } i \notin A \cup B, \quad j \in \mathbb{N}, \\ 1 & \text{si } i \in A \cup B, \quad j = i, \\ 0 & \text{si } i \in A \cup B, \quad j \neq i. \end{cases} \tag{3.23}$$

La dinámica dada por T es casi igual a la de P, con la única diferencia de que cuando un estado llega a A o a B se queda ahí para siempre, pues la probabilidad de que evolucione a cualquier otro $j \in S$ es cero. Asumiendo una dinámica ergódica, P^n converge a la distribución de equilibrio del canónico, por lo que también existe el límite

$$T^\infty = \lim_{n \to \infty} T^n, \tag{3.24}$$

el cual lleva cualquier distribución inicial a A o a B. De esta manera, el compromiso q_i será la suma de probabilidades después de haber propagado una distribución local que se encuentra sólo en i, y que está dada por un vector canónico $e_r^i = \delta_{ir}$ propagado al infinito mediante T^∞:

$$q_i = \sum_{k \in B} (e^i T^\infty)_k = \sum_{k \in B} T^\infty_{ik}. \tag{3.25}$$

En pocas palabras, se inicia en un estado i que después es propagado hasta infinito y se suman las probabilidades de que haya sido absorbido por cualquiera de los estados que conforman a B.

Para simplificar los cálculos que siguen, se toma $A = \{a\}$ y $B = \{b\}$; cuando los conjuntos son más grandes, simplemente se aglomeran en uno sólo. Ahora, diagonalizando T se obtiene

$$T^{\infty} = R \cdot \lim_{n\to\infty} \text{diag}(\lambda_1^n, ..., \lambda_N^n) \cdot R^{-1}, \tag{3.26}$$

donde $R = (r_1, ...r_N)$ es la matriz de eigenvalores derechos de T, y λ_i son los eigenvalores correspondientes, ordenados de mayor a menor módulo complejo. Se tiene por el teorema de Perron-Frobenius que existen exactamente dos eigenvectores izquierdos con eigenvalores 1: e^a y e^b, mientras que el módulo de los otros eigenvalores es estrictamente menor que uno. De esto se sigue que

$$\lim_{n\to\infty} |\lambda_i^n| = 0 \quad \forall\, |\lambda_i| < 1; \tag{3.27}$$

sustituyendo en (3.26) se llega a

$$T^{\infty} = R \cdot \text{diag}(1, 1, 0, ..., 0) \cdot R^{-1}. \tag{3.28}$$

Definiendo $L \equiv R^{-1}$ como la inversa de la matriz de eigenvectores derechos, se tiene que L son eigenvectores izquierdos pues, sustituyendo por L en (3.26):

$$T^{\infty} = L^{-1} \cdot \lim_{n\to\infty} \text{diag}(\lambda_1^n, ..., \lambda_N^n) \cdot L; \tag{3.29}$$

multiplicando L por la derecha en ambos lados de la igualdad esto se convierte en

$$L \cdot T^{\infty} = L \cdot \text{diag}(\lambda_1^n, ..., \lambda_N^n), \tag{3.30}$$

que es la definición de eigenvalores izquierdos.

Con esto, se puede evitar calcular la inversa de R una vez que se obtienen los eigenvalores izquierdos. A pesar de que, en general, el costo computacional de ambos procedimientos es similar, en el caso de T resulta mucho menos costoso calcular una base para los eigenvalores izquierdos. En primer lugar, expresando a L de la forma $(l_1, ..., l_N)'$ (donde l' denota la transpuesta de l), se deduce

$$\begin{aligned} T^{\infty} &= (r_1, ..., r_N) \cdot \text{diag}(1, 1, 0, ..., 0) \cdot (l_1, .., l_N)' \\ &= (r_1, r_2) \cdot (l_1, l_2)'. \end{aligned} \tag{3.31}$$

Por otro lado, como e^a y e^b son los únicos eigenvectores izquierdos con eigenvalor 1, l_1 y l_2 deben ser una combinación lineal de los mismos:

$$(l_1, l_2)' = S_l \cdot (e^a, e^b)' = \begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix} (e^a, e^b)'. \tag{3.32}$$

Además, debido a que T^∞ sigue siendo una matriz estocástica, todas sus entradas son positivas, por lo que, de acuerdo al teorema de Perron–Frobenius, uno de sus eigenvectores derechos tiene todas sus entradas iguales. Se puede escoger, sin pérdida de generalidad, $r_1 = (1, ..., 1) = \mathbf{1}$. Sustituyendo este resultado y (3.32) en (3.31) se desprende

$$T^\infty = (\mathbf{1}, r_2) \cdot S_l \cdot (e^a, e^b)'. \tag{3.33}$$

Por otro lado, (3.25) implica

$$[q^A, q^B] = T^\infty \cdot [e^a, e^b], \tag{3.34}$$

sustituyendo la descomposición de T^∞ en esta ecuación se llega a

$$[q^A, q^B] = [\mathbf{1}, r_2] \cdot S_l. \tag{3.35}$$

Así, el compromiso se calcula en términos de eigenvectores dominantes de T^∞. Para conocer las entradas de S_l, se utilizan las condiciones de frontera del compromiso: $q_a^A = 1$, $q_b^A = 0$, $q_a^B = 0$, $q_b^B = 1$. Considerando ambos casos, se tienen matrices del tipo

$$\begin{pmatrix} q_k^A \\ q_k^B \end{pmatrix} = \begin{pmatrix} \delta_{ak} \\ \delta_{bk} \end{pmatrix} = (\mathbf{1}_k, (r_2)_k) \cdot S_l \quad k \in \{a, b\}. \tag{3.36}$$

Resolviendo el sistema matricial en términos de S_l, queda

$$S_l = \begin{pmatrix} 1 & (r_2)_a \\ 1 & (r_2)_b \end{pmatrix}^{-1}. \tag{3.37}$$

Por último, multiplicando todas las matrices en (3.33) y sustituyendo en (3.25), el compromiso de que un estado i visite B antes que A será

$$q_i = (e_i T^\infty)_B = \frac{(r_2)_i - (r_2)_a}{(r_2)_b - (r_2)_a}. \tag{3.38}$$

De esta manera, el problema de las probabilidades de compromiso se reduce al de encontrar el eigenvector dominante no trivial de T. El método de eigenvectores dominantes, a diferencia del sistema de ecuaciones, funciona aún cuando el sistema es muy grande ($|S| \approx 10^6$) [47].

Las conformaciones de una proteína pueden ser descritas a través de la identificación y delimitación de los conjuntos metaestables en la FES. Por otro lado, la información de transiciones entre estados se obtiene con las probabilidades de compromiso.

Capítulo 4

Resultados y discusión

Se analizaron los cuatro métodos de muestreo acelerado descritos en los capítulos anteriores. Para ello, se aplicaron dichos algoritmos en 4 potenciales (cada método se utilizó 10 veces en cada potencial para calcular promedios de las cantidades deseadas) y se midió el error cuadrático entre la FES original y la reconstruida por cada método, además de que se compararon las probabilidades de compromiso de forma cualitativa, pasos necesarios para escapar de mínimos locales y para hacer la reconstrucción de cada potencial. En todos los casos se trabajó de manera adimensional.

4.1. Una dimensión

Se construyeron dos potenciales: uno con un sector difusivo y otro con perturbaciones sinusoidales en todo su dominio. En ambos casos se realizó primero una dinámica de Langevin de 20 mil pasos para la estimación de parámetros en cada método de aceleración. En todas las dinámicas realizadas se fijó la temperatura con $\beta = 0.8$. Además, en el método variacional se utilizó una expansión en serie de senos y cosenos con 1 término constante, 14 términos del tipo $\alpha_i \sin(nx)$, $i = 1 \dots 14$, y 14 del tipo $\alpha_j \cos(nx)$, $j = 1 \dots 14$.

4.1.1. Potencial con sector difusivo

El primer potencial fue

$$U_1(x) = -12 \exp\left[-\frac{(x+4)^2}{1.28}\right] - 12.5 \exp\left[-\frac{(x-2)^2}{0.32}\right] - 13 \exp\left[-\frac{(x-4)^2}{0.5}\right]. \tag{4.1}$$

En este caso se tienen tres mínimos locales A, B, y C, así como un sector difusivo que separa a A de B. Al graficar la distribución del canónico dada por $P(x) = Z^{-1} \exp\left[-\beta f(x)\right]$ se observa

que, mientras más profundo sea el pozo, es más probable que el sistema se encuentre dentro del mismo.

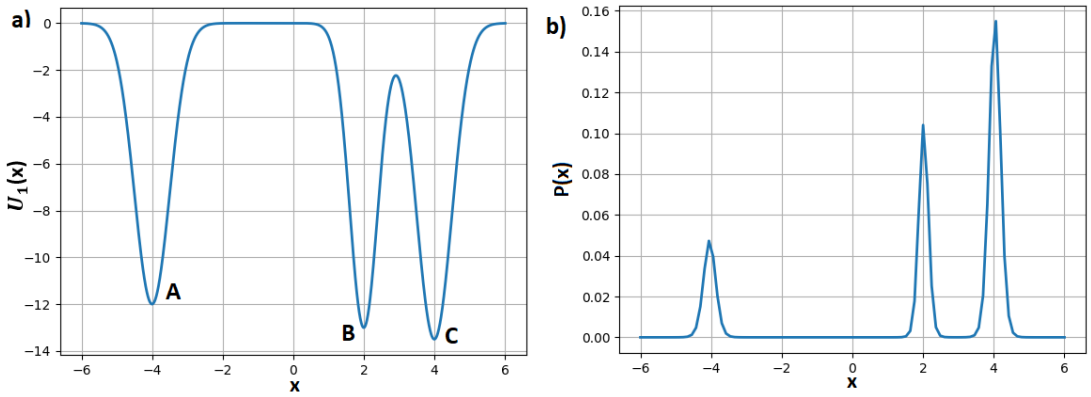

Figura 4.1: FES con sector difusivo: a) Potencial utilizado. b) Distribución asociada a la FES.

Construyendo la matriz estocástica mediante (3.18) y haciendo el análisis espectral descrito en el capítulo anterior, se encontraron tres conformaciones o conjuntos metaestables (correspondientes a eigenvalores $\lambda_1 = 0.992$, $\lambda_2 = 0.989$, $\lambda_3 = 0.984$, $\lambda_4 = 0.351$, $\lambda_{n>4} < \lambda_4$), con dominios en $[-6, -3.19]$, $[1.09, 2.67]$ y $[3.25, 6.01]$, respectivamente. Esto coincide con las ubicaciones de los mínimos A,B,C. Para la conformación A se calcularon los valores $e_1(x_A), e_2(x_A), e_3(x_A)$ de cada eigenvector en el punto $x_A = -4$. Después, partiendo de este punto se tomaron como parte de A todos los x tales que

$$\left| \frac{e_i(x) - e_i(x_A)}{e_i(x_A)} \right| \leq 0.1 \quad i = 1, 2, 3. \tag{4.2}$$

Este procedimiento se utilizó también para delimitar B y C. Es decir, para cada conformación se tomaron como parte del conjunto los puntos donde los tres eigenvectores tuvieran valores que difirieran porcentualmente de su valor en el mínimo local a lo más en 10 %.

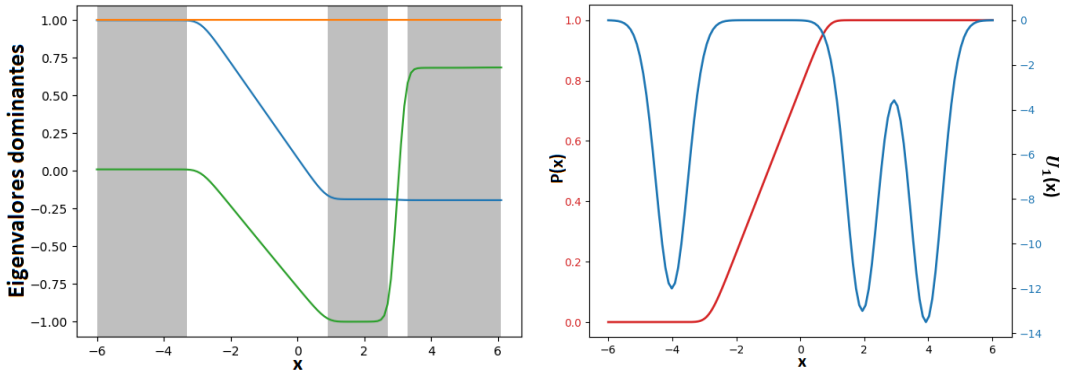

Figura 4.2: a) Análisis de conjuntos metaestables para el primer potencial. b) Probabilidades de compromiso de la primera conformación a la penúltima, de izquierda a derecha.

Una vez delimitados los conjuntos metaestables, se calcularon las probabilidades de compromiso de A a C. Comparando la gráfica del potencial con la del compromiso se puede ver que si el sistema se encuentra en B, va a evolucionar a C antes que a A. Esto tiene sentido por el sector difusivo que separa a A de B,C. Por otro lado, si el sistema empieza en x_o en el sector difusivo, la probabilidad de que visite C antes que A va creciendo mientras x_o está más cerca de B

Después de este análisis inicial, se hicieron las dinámicas de Langevin sobreamortiguado utilizando cada método de aceleración, así como una sin aceleración. Se fijó en todos los casos la posición inicial $x_0 = 2$. A continuación se muestran las FES reconstruidas por cada método.

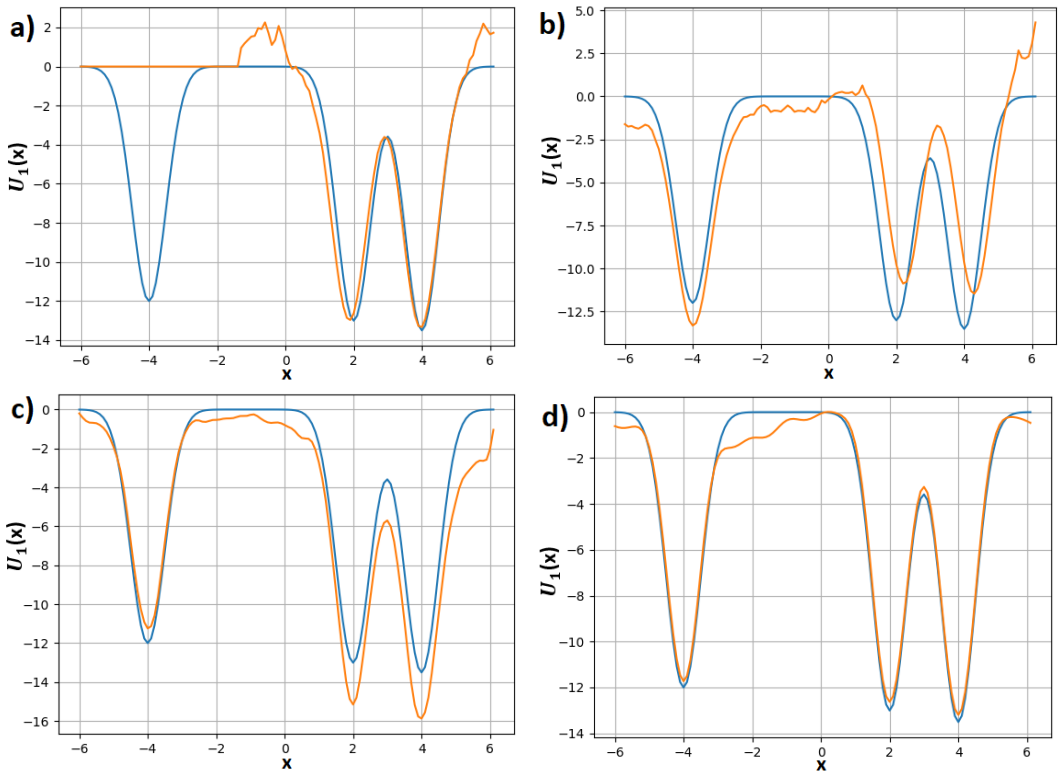

Figura 4.3: FES original (azul) superpuesta con las reconstrucciones de método de aceleración: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

El único método que no consiguió muestrear todas las conformaciones fue el de inundación, que tan sólo estuvo en B y C. Con los otros algoritmos se muestreo el potencial completo; sin embargo, en GaMD el primer mínimo local es menor que los otros dos, lo cual no corresponde a $f(x)$. Esto ocurre por falta de muestreo, pues aunque se explora todo el potencial, no se estiman correctamente las probabilidades de cada conformación.

Las probabilidades de compromiso calculadas con las FES reconstruidas se presentan en la figura 4.4. En este caso cada metodología logró reconstruir la forma del compromiso en A,B y C. Pero en la región difusiva el crecimiento de la probabilidad no tiene el comportamiento del compromiso real en ningún caso salvo en metadinámica. Esto coincide con que la FES de metadinámica fue la que reconstruyó mejor la región difusiva, en los otros casos hubieron curvas significativas que no existen en $U_1(x)$.

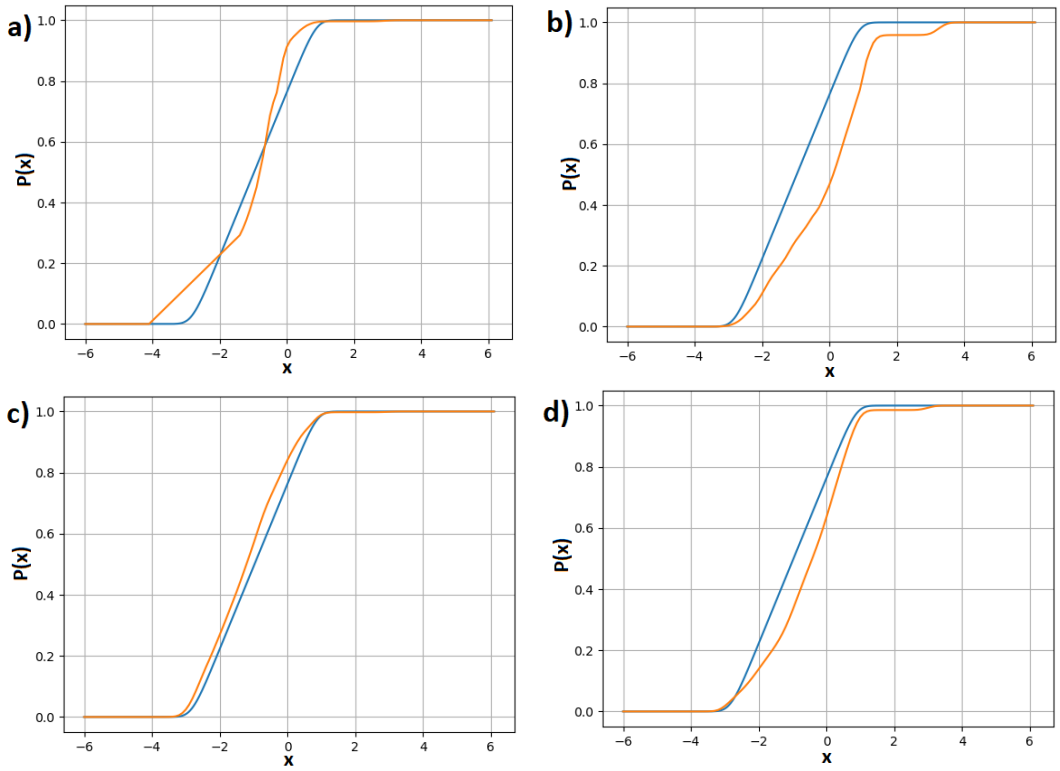

Figura 4.4: Probabilidad de compromiso original (azul) superpuesta con la reconstruida por cada método: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

Finalmente, se calcularon los errores, tiempos de salida (de la conformación inicial) y pasos necesarios para la reconstrucción.

Método	Pasos para salida	Error cuadrático	Pasos totales
No acelerado	170,161±1178	-	> 160M
Inundación	9,265±1428	3.055±0.114	30M
GaMD	11,661±1153	2.219±0.054	15M
Metadinámica	7,368±1343	1.073±0.017	300,000
Variacional	25,664±2107	0.852±0.012	14M

Tabla 4.1: Resultados cuantitativos para $U_1(x)$.

Tanto en términos de los pasos necesarios para salir de la conformación inicial así como para muestrear la FES por completo, los cuatro métodos de aceleración requieren una cantidad menor

de pasos que la dinámica original en al menos un orden de magnitud. La dinámica no acelerada más larga que se realizó tuvo 160 M de pasos y no logró muestrear la FES entera. Por otro lado, los métodos con un menor error cuadrático fueron metadinámica y variacional.

4.1.2. Potencial con perturbaciones

Después, se utilizó el potencial

$$U_2(x) = -13\exp\left[-\frac{(x+5)^2}{0.32}\right] - 7.4\exp\left[-\frac{x^2}{0.32}\right] - 10\exp\left[-\frac{(x-4)^2}{0.5}\right] + 3\sin(3x)\cos(3x). \quad (4.3)$$

Aquí hay también tres mínimos locales principales A,B y C, además de perturbaciones sinusoidales en todo el dominio. La distribución muestra que, a temperatura con $\beta = 0.8$, los mínimos locales resultado de las perturbaciones sinusoidales tienen probabilidades muy bajas en comparación a los mínimos principales.

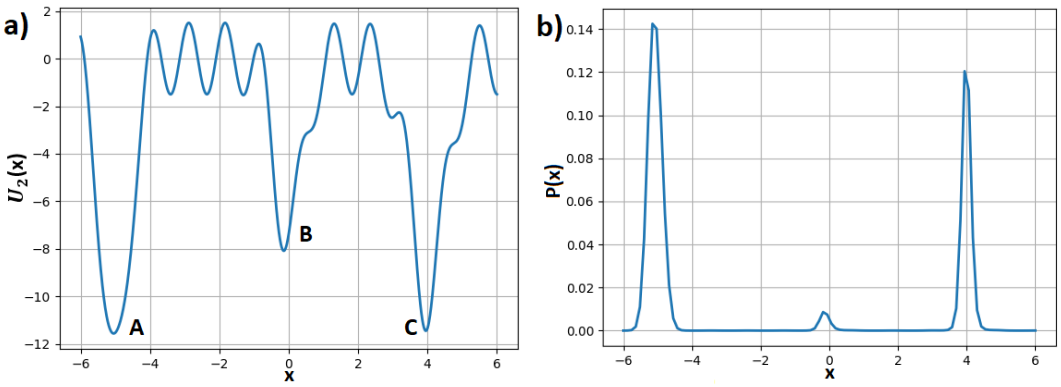

Figura 4.5: FES con perturbaciones sinusoidales: a) Potencial utilizado. b) Distribución asociada a la FES.

Al realizar el análisis espectral en este potencial, se encontraron tres conformaciones con dominios en $[-6, -4.25]$, $[-0.81, 1.11]$ y $[3.32, 6.01]$, respectivamente. Se puede ver que los tres eigenvectores tienen también perturbaciones sinusoidales en las regiones en que no son aproximadamente constantes. Cabe resaltar que mediante este método se confirma la existencia de sólo tres conformaciones, por el número de eigenvectores dominantes obtenidos. Utilizando el mismo procedimiento que en el potencial anterior se delimitaron los conjuntos metaestables.

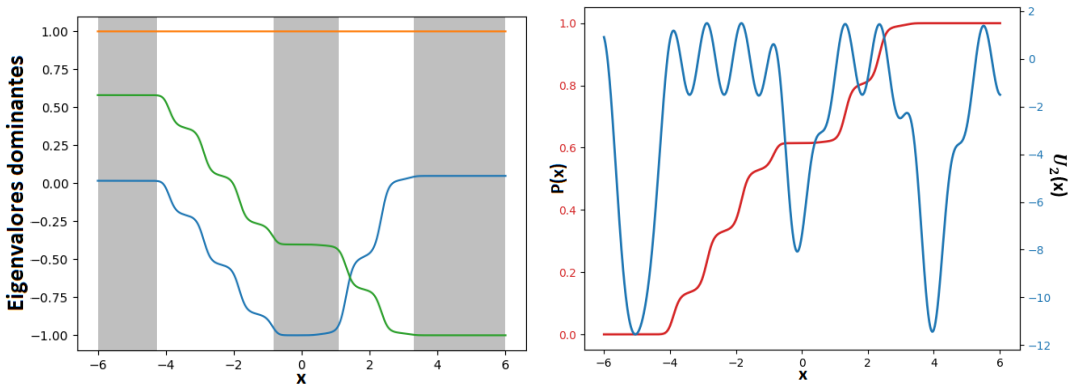

Figura 4.6: a) Análisis de conjuntos metaestables para el tercer potencial. b) Probabilidades de compromiso de la primera conformación a la última, de izquierda a derecha.

Los compromisos fueron calculados de A a C. Como ahora no hay una gran región difusiva entre las conformaciones, cuando el sistema empieza en B tiene la posibilidad de evolucionar tanto a C como a A. Sin embargo, como C está más cerca y existen menos barreras entre B y C, se esperaría que sea más probable que B evolucione a C que a A, esto se confirma en la gráfica del compromiso, con un valor aproximado de 0.6 en B. Además, se aprecia que el compromiso también tiene perturbaciones en las regiones de transición entre mínimos locales.

Las dinámicas en este caso tuvieron como posición inicial $x_0 = 4$. A continuación se muestran las FES reconstruidas por cada uno de los métodos de aceleración.

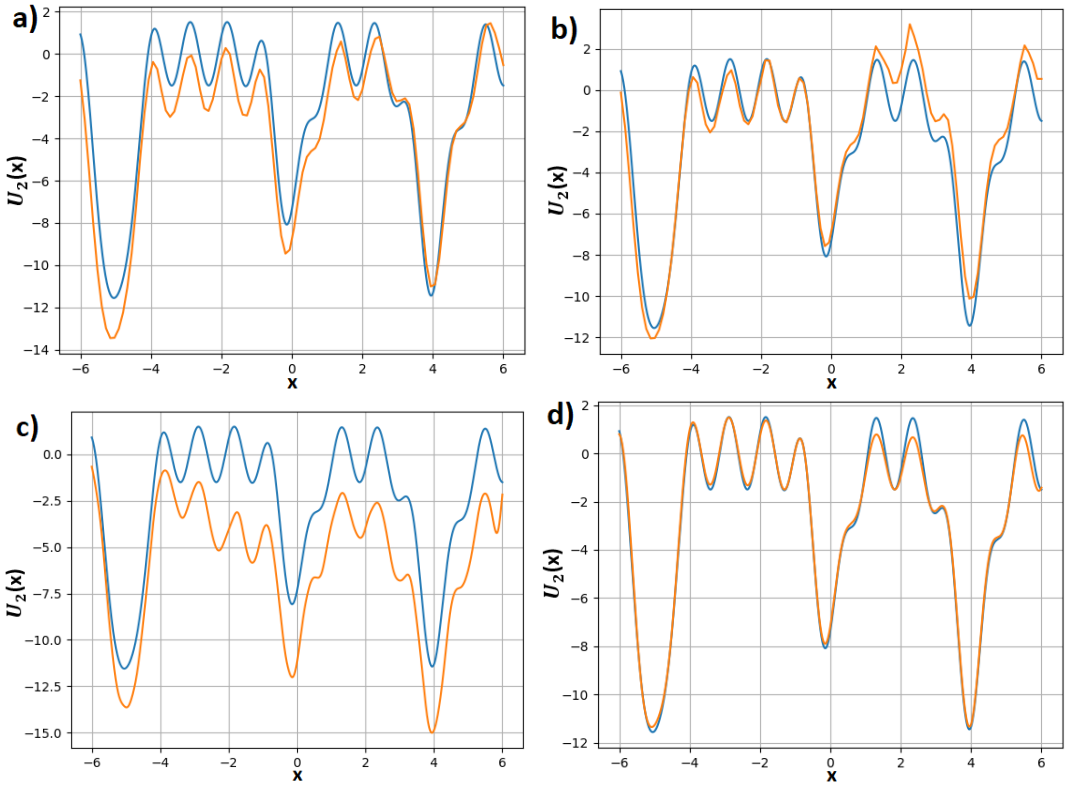

Figura 4.7: FES obtenidas por cada método de aceleración: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

Los cuatro métodos fueron capaces de muestrear la FES en su totalidad. A pesar de esto, el problema de falta de muestreo se presentó de nuevo en GaMD, así como en inundación; en ambos métodos $|A - C|$ fue mayor que en el potencial real. Esto también ocurrió en metadinámica, donde C aparece con un valor menor que A, pero en este algoritmo el error no se debe a una falta de muestreo, sino al sobremuestreo de C, pues un valor menor indica que se depositaron más gausianas en C que en A. Respecto a las regiones de transición, el mejor método fue el variacional, lo que era de esperarse por la forma sinusoidal de la expansión que se utilizó en dicho algoritmo.

A partir de las FES reconstruidas se calcularon los compromisos en cada caso. A pesar de que todos los métodos muestrearon la FES completa, los errores que tuvieron en las regiones de trancisión se ven reflejados en los compromisos. La probabilidad de compromiso en B calculada con inundación y GaMD es menor que en el compromiso real, mientras que en metadinámica y variacional es mayor.

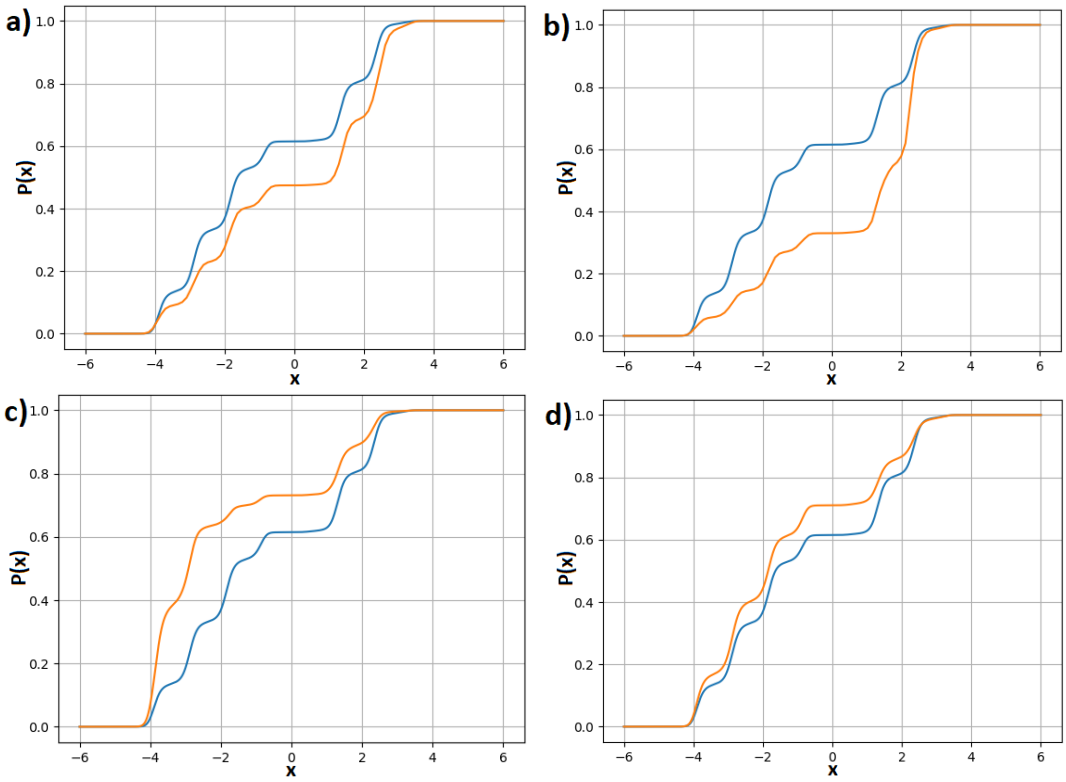

Figura 4.8: Probabilidades de compromiso sobre la FES reconstruida por cada método: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

Por ultimo, se calcularon los errores, tiempos de salida y pasos totales de cada metodología.

Método	Pasos para salida	Error cuadrático	Pasos totales
No acelerado	422,541±4523	-	> 160M
Inundación	76,411±3856	1.071±0.052	24M
GaMD	58,149±3941	0.867±0.034	12M
Metadinámica	27,822±5734	0.976±0.081	100,000
Variacional	82,672±4832	0.269±0.011	12M

Tabla 4.2: Resultados cuantitativos para este potencial.

Una vez más los cuatro métodos mostraron una ganancia de al menos un orden de magnitud en el número de pasos necesarios para reconstruir toda la FES. Tanto en este caso como en el anterior, metadinámica fue el método que necesitó menos pasos para terminar. Además, en términos del error cuadrático respecto a $f(x)$, metadinámica y variacional fueron los mejores algoritmos.

4.2. Dos dimensiones

Se parametrizaron dos FES con distintas características: una con sectores difusivos y otra con un sector alejado, además de dos caminos distintos entre conformaciones. En ambos potenciales se realizó primero una dinámica sin aceleración con 320 mil pasos, para la estimación de parámetros en cada método de aceleración. La temperatura se fijó con $\beta = 0.5$. Además, en el método variacional se utilizó una expansión de senos y cosenos con 4 términos del tipo $\sin(nx)\sin(my)$, 4 términos $\cos(nx)\cos(my)$, 4 términos $\sin(nx)\cos(my)$ y 4 $\cos(nx)\sin(my)$.

4.2.1. Potencial con sectores difusivos

El primer potencial fue

$$U_3(x,y) = \begin{cases} 11\sin(3x - 1.5) & \text{si } y \geq 0 \\ 11\left[1 + \frac{y}{2}\right]\sin(3x - 1.5) & \text{si } -2 \leq y < 0 \\ 0 & \text{en otro caso.} \end{cases} \tag{4.4}$$

En la parte del dominio con $y \geq 0$ el potencial es sinusoidal, con tres mínimos locales A,B y C. Al disminuir el valor de y por abajo de 0, la magnitud del potencial va disminuyendo hasta que se llega a una región difusiva rectangular. Al graficar la distribución del canónico dada por $P(x,y) = Z^{-1}\exp\left[-\beta f(x,y)\right]$ se obtienen tres regiones equiprobables, que corresponden a los tres mínimos de $f(x,y)$.

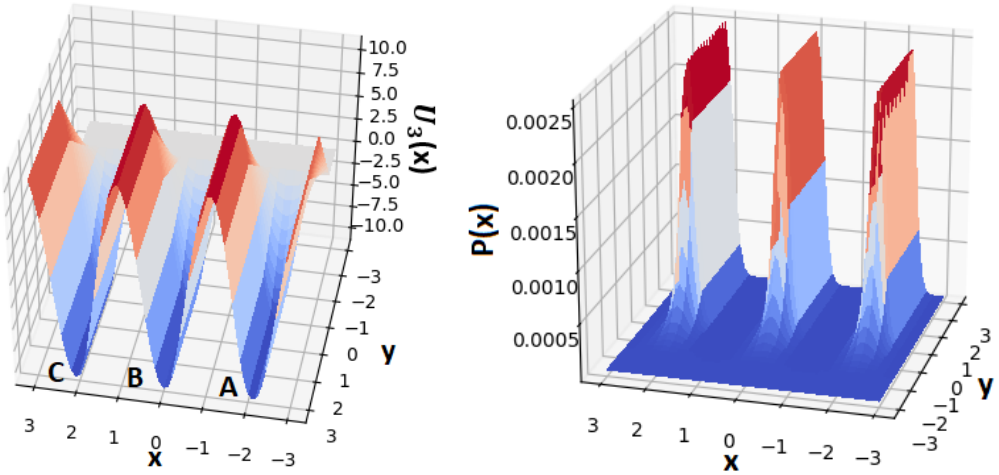

Figura 4.9: FES con sector difusivo: a) Potencial utilizado. b) Distribución asociada a la FES.

Con el análisis de eigenvectores extendido a dos dimensiones se encontraron tres conformaciones, correspondientes a los mínimos locales A, B y C. Análogamente al caso en una dimensión, para la conformación B se calcularon los valores $e_i(\boldsymbol{x_B})$ de cada eigenvector ($i = 1, 2, 3$) en $\boldsymbol{x_B} = (0, 1)$. Después, partiendo de ese punto se tomaron como parte de B todos los $\boldsymbol{x}$ adyacentes tales que

$$\left| \frac{e_i(\boldsymbol{x}) - e_i(\boldsymbol{x_B})}{e_i(\boldsymbol{x_B})} \right| \leq 0.1 \quad i = 1, 2, 3 \tag{4.5}$$

Análogamente, se delimitaron A y C. La diferencia porcentual máxima se fijo en todos los casos de 10 %.

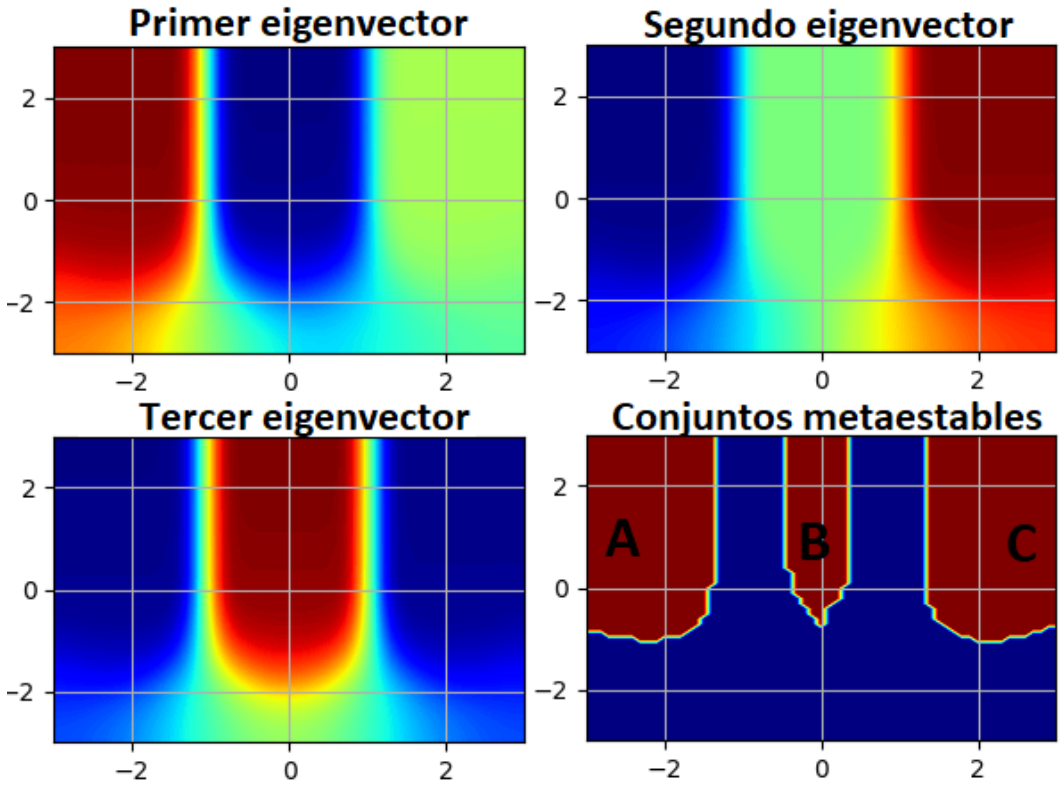

Figura 4.10: Análisis de conjuntos metaestables para el segundo potencial.

Los compromisos se calcularon de A a B. Aunque al comenzar en C lo más probable es que el sistema evolucione a B, esta probabilidad no es igual a 1 puesto que existe la posibilidad de que se recorra la región difusiva hasta llegar a A. Además, existe una pequeña franja en el compromiso donde la probabilidad es igual a 0.5, y corresponde al máximo local del sector sinusoidal de $f(x, y)$

entre A y B, pues se trata de un equilibrio inestable, y el sistema tiene la misma posibilidad de caer tanto en A como en B.

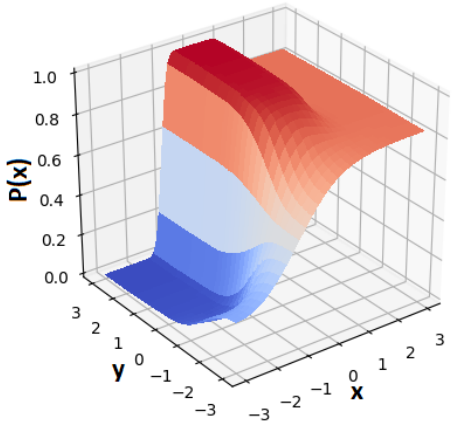

Figura 4.11: Probabilidades de compromiso de A a B.

Al finalizar este análisis inicial, se hicieron dinámicas utilizando cada método de aceleración así como una sin aceleración. Todas las simulaciones comenzaron en $\boldsymbol{x_0} = (0, 2)$. A continuación se muestra la distribución obtenida con una dinámica sin acelerar de 320 mil pasos.

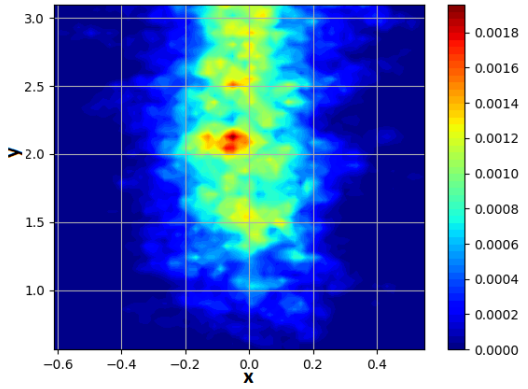

Figura 4.12: Distribución obtenida con una dinámica pequeña.

Como se puede ver, en esta dinámica sin acelerar sólo se muestrea una parte de la conformación que corresponde a B. De aquí se estiman los parámetros de cada método de aceleración. Las FES reconstruidas por cada método de muestran a continuación.

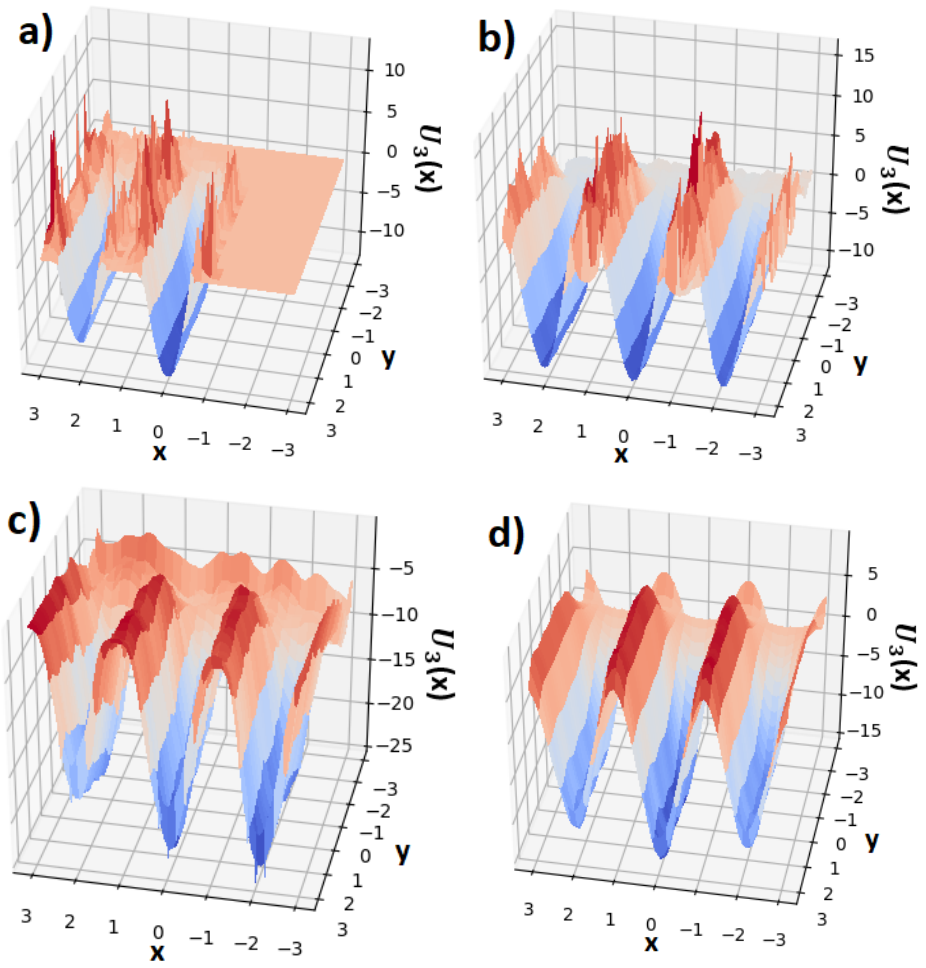

Figura 4.13: FES obtenidas por cada método de aceleración: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

Inundación no logró muestrear la FES completa, tan sólo las conformaciones By C. Además, en las regiones de transición existen muchos puntos que no fueron muestreados, por lo que existen *huecos* en la FES reconstruida con este método. El mismo fenómeno ocurre en GaMD, que a pesar de haber muestreado las 3 conformaciones, tiene fallas en los máximos locales. Mientras que metadinámica reconstruyó la FES completa, por un problema de sobremuestro distorsiona el sector difusivo con la adición de gausianas extra en algunos puntos del mismo. Es el problema análogo al del caso 1D. Por último, el método variacional tuvo una buena reconstrucción en parte por la similitud entre la expansión utilizada y $f(x, y)$, al menos en la región sinusoidal. En la frontera $y = -3$ se puede ver que, por la imposición de periodicidad en el potencial variacional, existen

máximos y mínimos locales que no corresponden a la FES real.

Las probabilidades de compromiso calculadas con las FES reconstruidas se presentan en la siguiente figura. Como el método de inundación no exploró A, su compromiso es el que respeta menos la forma del original. En los otros tres métodos esta forma se conserva cerca de A y B, pero en la franja con probabilidad 0.5 se presentan diferencias, especialmente en GaMD y metadinámica. En el caso de GaMD esto se debe a la falta de muestreo en el máximo local que une A con B, y en metadinámica al sobremuestreo en la región difusiva.

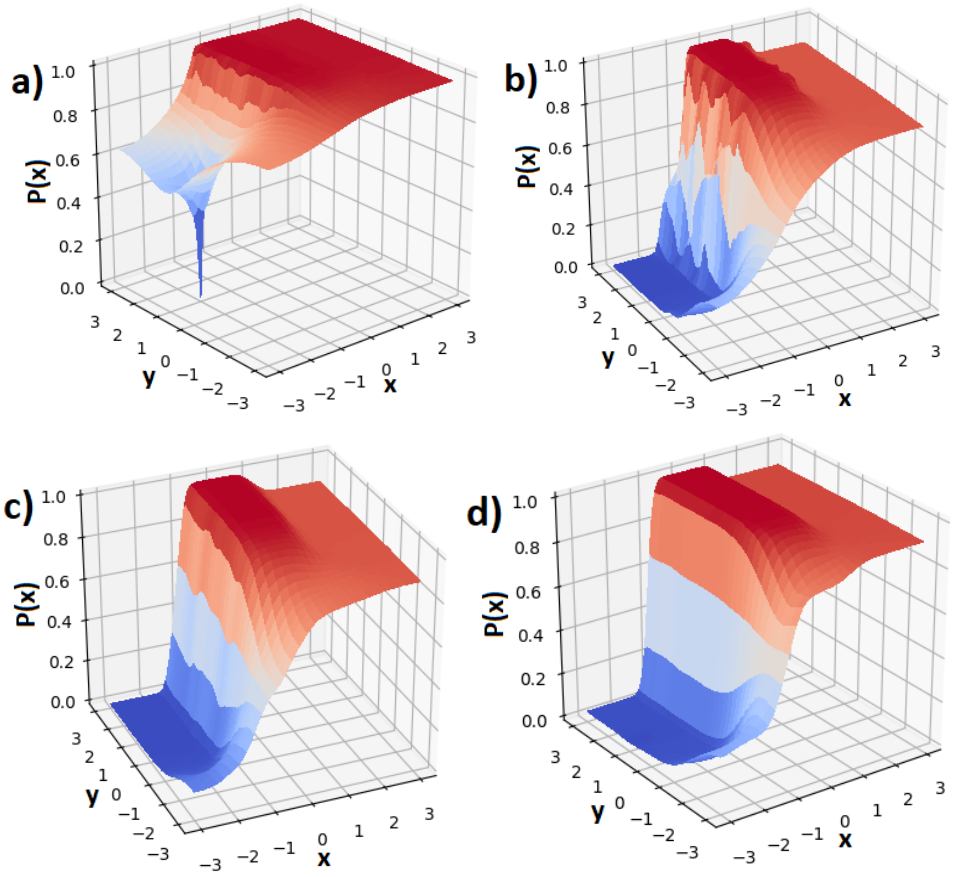

Figura 4.14: Probabilidades de compromiso sobre la FES reconstruida por cada método: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

Finalmente, se obtuvieron los resultados cuantitativos, que se muestran en la siguiente tabla.

Método	Pasos para salida	Error cuadrático	Pasos totales
No acelerado	2,724,182±106272	-	> 400M
Inundación	1,724,182±246728	4.523±0.0297	64M
GaMD	1,457,251±185249	2.037±0.0814	32M
Metadinámica	1,023,227±102465	1.805±0.0176	6M
Variacional	1,102,321±152815	1.509±0.0162	20M

Tabla 4.3: Resultados cuantitativos para este potencial.

A pesar de que el número de pasos necesarios para salir de la conformación inicial tiene el mismo orden de magnitud tanto en la dinámica sin acelerar como en los cuatro métodos de aceleración, sigue habiendo una diferencia de al menos un orden de magnitud en la reconstrucción de la FES. En la sección de pasos totales, el símbolo $> 400M$ indica que se hizo una dinámica con 400 millones de pasos y no logró muestrear toda la FES. En términos del error cuadrático, metadinámica y variacional son los mejores. Además, metadinámica fue el método que requirió un menor número de pasos para la reconstrucción.

4.2.2. Potencial con distintos caminos y sectores alejados

Para el segundo potencial se utilizó la función

$$\begin{aligned} f_1(x,y) = &-14\exp\left[-\frac{x^2}{0.5}-\frac{(y-2)^2}{0.5}\right]-13\exp\left[-\frac{(x-1)^2}{0.72}-\frac{y^2}{0.72}\right] \\ &+12\exp\left[-\frac{(x+1.5)^2}{0.5}-\frac{(y-1.5)^2}{0.5}\right]-3\exp\left[-\frac{(x+2.5)^2}{0.08}-\frac{(y-1.8)^2}{3.38}\right] \\ &-2.5\exp\left[-\frac{(x+2)^2}{2}-\frac{(y-2.5)^2}{0.08}\right]-13\exp\left[-\frac{(x-2)^2}{0.5}-\frac{(y+2)^2}{0.32}\right] \\ &+12\exp\left[-\frac{x-^2}{0.08}-\frac{y^2}{2}\right]+11\exp\left[-\frac{(x-1.5)^2}{2.88}-\frac{y^2}{0.08}\right] \\ &+14\exp\left[-\frac{x^2}{0.08}-\frac{(y+2)^2}{0.72}\right]-11\exp\left[-\frac{(x-1)^2}{0.08}-\frac{(y+1)^2}{0.08}\right], \end{aligned} \tag{4.6}$$

y después se añadieron perturbaciones mediante

$$f_2(x,y) = \cos(3x)+\sin(3y)+\sin(3xy)+\cos(3xy). \tag{4.7}$$

Así, el potencial total fue

$$U_4(x, y) = f_1(x, y) + 0.4f_2(x, y). \tag{4.8}$$

Aquí hay cuatro mínimos locales principales A,B,C,D, y dos barreras energéticas que separan A y B de C y D. Además, hay perturbaciones sinusoidales en todo el dominio, y algunas barreras extra entre A y B. La gráfica de la distribución muestra que, a esta temperatura con $\beta = 0.5$, las perturbaciones añadidas no afectan el espacio canónico.

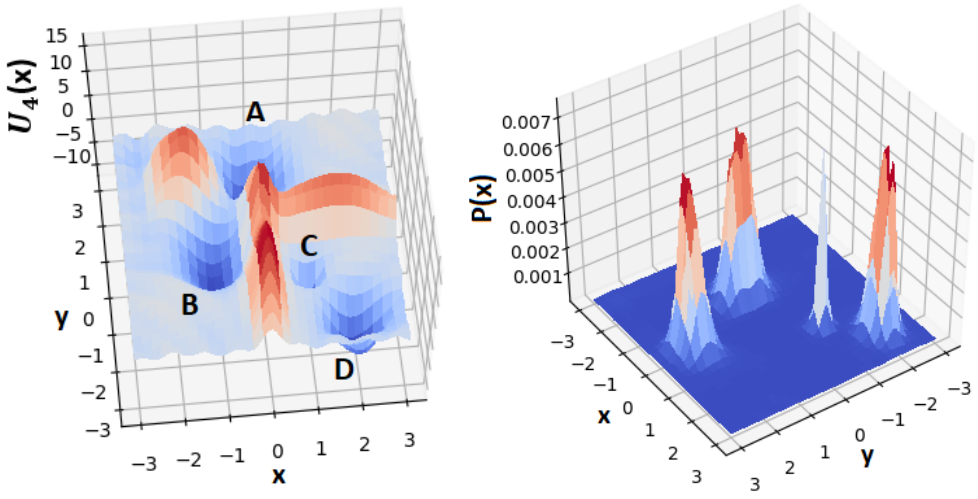

Figura 4.15: FES con sectores alejados: a) Potencial utilizado. b) Distribución asociada a la FES.

Al realizar el análisis espectral en este potencial, se encontraron cuatro conformaciones, correspondientes a A, B, C y D. El procedimiento utilizado para delimitar el dominio de cada una fue el mismo que en el ejemplo anterior. La existencia de sólo cuatro eigenvectores dominantes (con eigenvalores $\lambda_1 = 0.997$, $\lambda_2 = 0.995$, $\lambda_3 = 0.981$, $\lambda_4 = 0.983$, $\lambda_5 = 0.621$, $\lambda_{n>5} < \lambda_5$) confirma que las perturbaciones sinusoidales no constituyen un conjunto metaestable. Por otro lado, dichas perturbaciones no se ven reflejadas en los eigenvectores, como ocurrió en el caso en 1D.

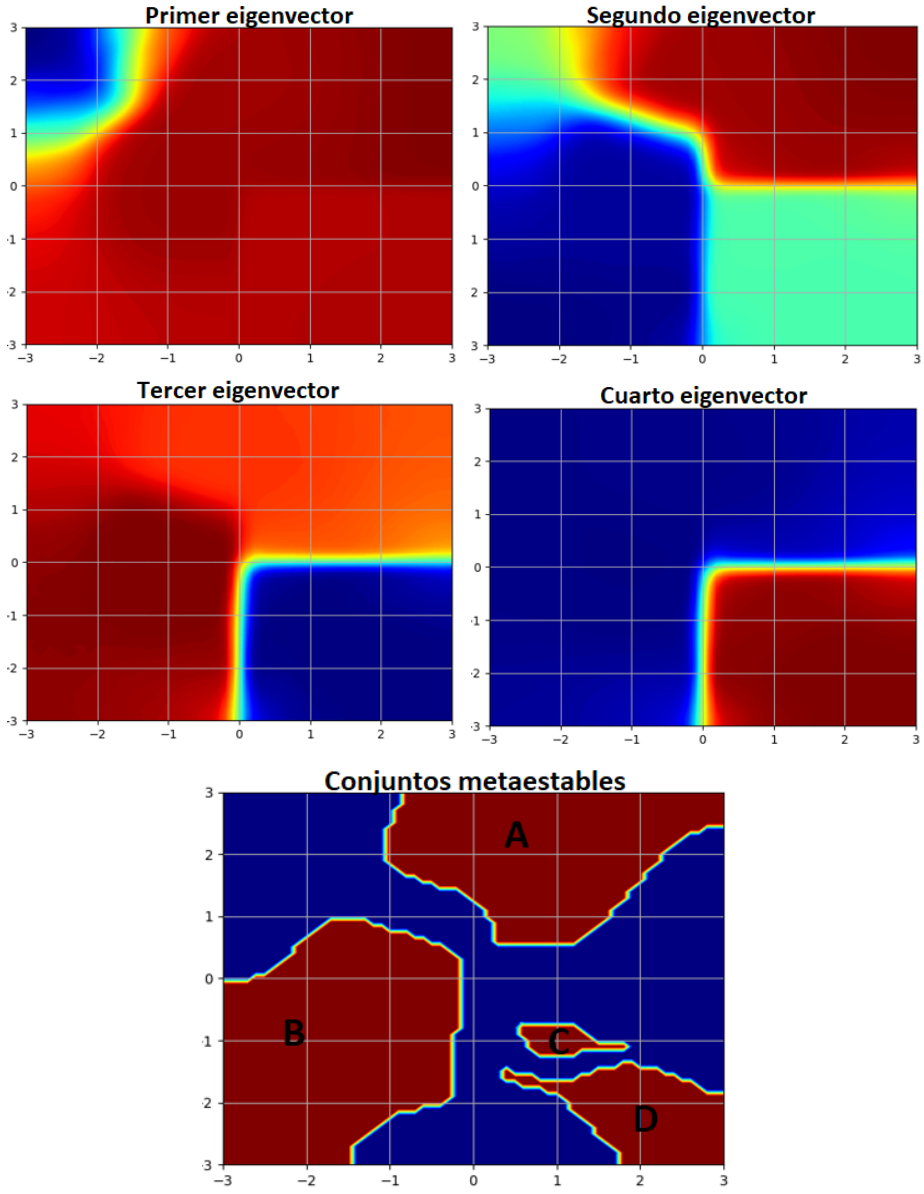

Figura 4.16: Análisis de conjuntos metaestables para este potencial.

Los compromisos se calcularon de A a D. Debido a la presciencia de las barreras energéticas, el compromiso se divide en dos sectores delimitados por dichas barreras. Por un lado, una región rectangular que encierra a C y D, y por otro, una región que corresponde a A y B.

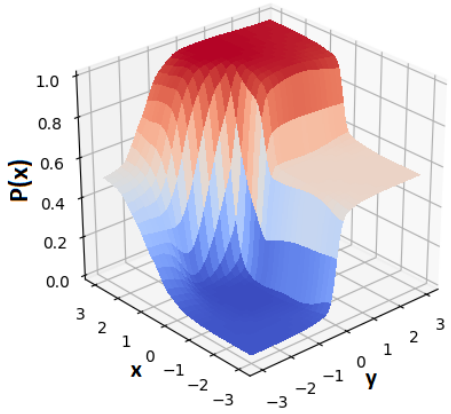

Figura 4.17: Probabilidades de compromiso de A a C.

Las dinámicas en este caso tuvieron como posición inicial $\boldsymbol{x_o} = (0, 2)$. A continuación se muestran las FES reconstruidas por cada método de aceleración.

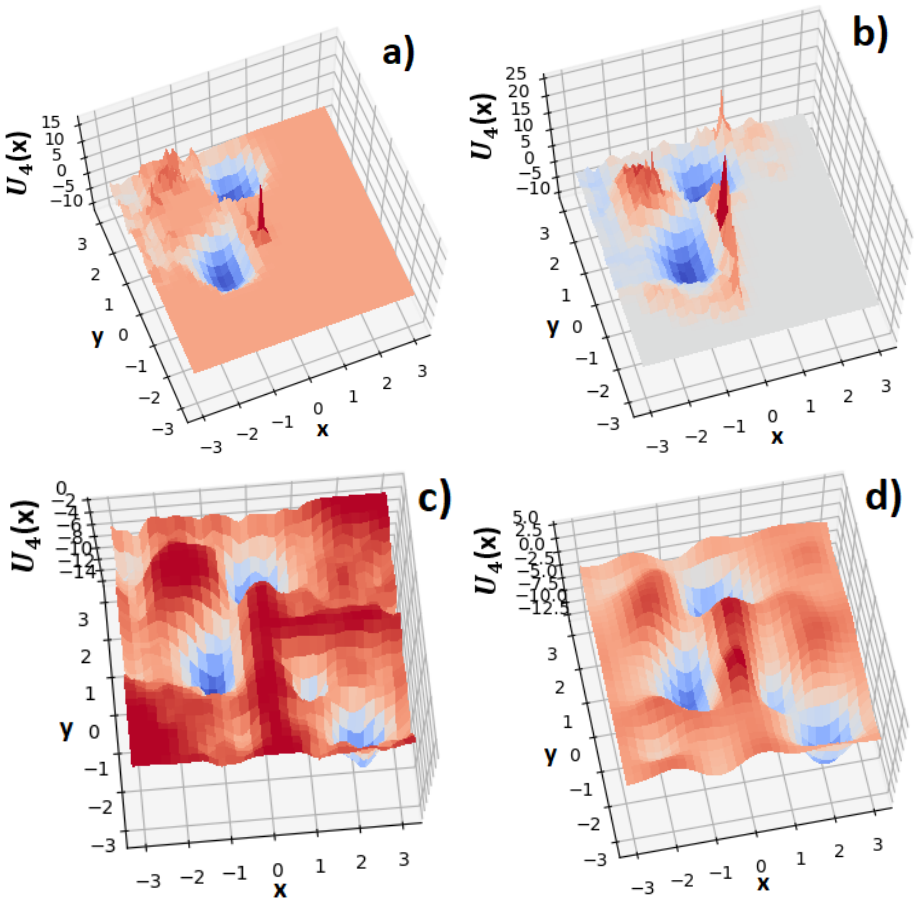

Figura 4.18: FES obtenidas por cada método de aceleración: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

Los método de inundación y GaMD exploraron las conformaciones A y B, así como parte del espacio de transición entre ellas. En particular, GaMD tuvo acceso al otro camino de transición, el que une A y B rodeando la barrera entre ellos. En las gráficas de ambos algoritmos hay *huecos* debidos a la falta de muestreo, y no existe ningún punto asociado al sector de C y D.

Por otro lado, tanto metadinámica como el método variacional muestrearon todas las conformaciones, aunque las barreras energéticas se aproximan más al potencial real en la reconstrucción de metadinámica. Esto se debe principalmente a la cantidad limitada de funciones sinusoidales que se utilizaron en la expansión del potencial variacional.

Con estas FES se calcularon los compromisos presentados a continuación. Como GaMD e inundación no muestrearon ningún punto de C y D, sus compromisos presentan grandes diferencias respecto al real. Específicamente, no se respetan las formas poligonales de los sectores en que se

divide el compromiso. En los otros dos métodos se aprecian ambos sectores, pero las barreras en sus fronteras presentan algunas distorsiones. Esto era de esperarse, pues en el método variacional las barreras energéticas no se reconstruyeron adecuadamente, y en metadinámica la deposición de gausianas extra altera los compromisos, como ya se había visto en el potencial anterior.

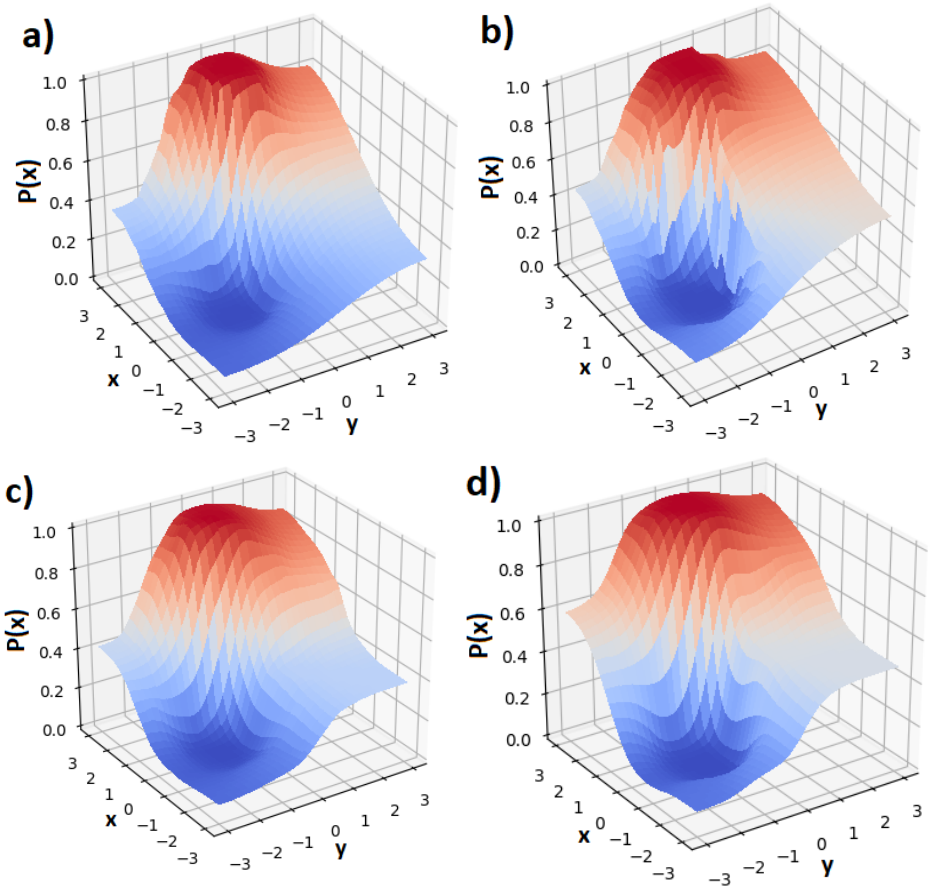

Figura 4.19: Probabilidades de compromiso sobre la FES reconstruida por cada método: a) Inundación. b) GaMD. c) Metadinámica. d) Variacional.

Finalmente, se obtuvieron los errores, tiempos de salida y pasos totales.

Método	Pasos para salida	Error cuadrático	Pasos totales
No acelerado	10,264,821±228711	-	> 800M
Inundación	4,841,627±149623	3.627±0.029	200M
GaMD	3,642,824±164721	3.309±0.041	120M
Metadinámica	3,524,812±175623	2.659±0.023	40M
Variacional	3,585,137±96275	2.571±0.018	80M

Tabla 4.4: Resultados cuantitativos para este potencial.

Este potencial fue el que requirió más pasos en general para su reconstrucción. La dinámica sin acelerar se hizo hasta los 800 millones de pasos y aún así no logró muestrear toda la FES. En términos de errores cuadráticos y pasos necesarios para la reconstrucción, metadinámica y variacional fueron de nuevo los mejores métodos.

4.3. Discusión

A continuación se analizan las ventajas y debilidades de cada método de aceleración, para al final compararlos entre ellos de acuerdo a varios criterios.

4.3.1. Método de inundación

La mayor ventaja de este método es que en su implementación no se necesitan realizar muchos cálculos adicionales a los de una dinámica usual de Langevin. Pues además de la facilidad con la que se estima el parámetro E_{in} (a partir de la profundidad del mínimo local), este algoritmo es simplemente una dinámica de Langevin sobre el potencial original sumado a una función constante en el tiempo.

Por otro lado, la mayor desventaja del método de inundación es que acelera la dinámica sólo de manera local: todos los otros sectores en la FES modificada son exactamente iguales a la original. Además, esta aceleración local está limitada, puesto que al utilizar una intensidad de inundación muy grande se distorsionan los caminos de transición, incluso después de haber ponderado la distribución.

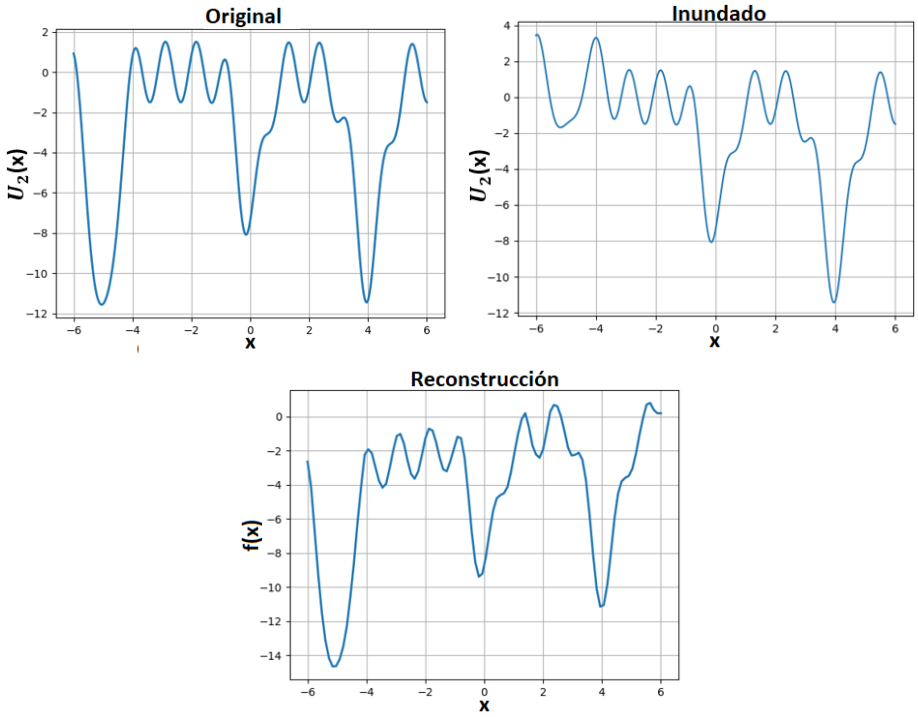

Figura 4.20: Al utilizar intensidades de inundación muy grandes, se distorsionan los caminos de transición aun después de ponderar.

Finalmente, otra debilidad de este algoritmo es que no tiene un criterio de paro. Mientras mayor sea el muestro, mejor sera la ponderación hacia la FES original; sin embargo, no hay una manera de estimar en qué momento se ha alcanzado un muestreo lo suficientemente bueno.

4.3.2. GaMD

Mientras que su funcionamiento es similar al método de inundación, en GaMD se requiere una subrutina previa en donde se elige automáticamente el parámetro k_0 a partir de la dinámica pequeña que se realiza primero. Por esto, el número de cálculos necesarios en GaMD aumenta un poco a comparación del método anterior. La automatización para seleccionar el parámetro más importante de este algoritmo hace que, en general, no se requieran ajustes manuales de las variables de entrada (dejando los valores por defecto se obtienen buenos resultados).

Otra de las mayores ventajas de GaMD es que la aceleración de las dinámicas se hace en toda la FES, además de que se tiene poco ruido energético al momento de ponderar. Respecto a sus desventajas, las más notable es que la eficiencia de este método depende de la posición inicial de

la simulación. Esto se debe a que los parámetros se escogen con base en un muestreo pequeño, por lo que sus valores dependen completamente de la posición inicial.

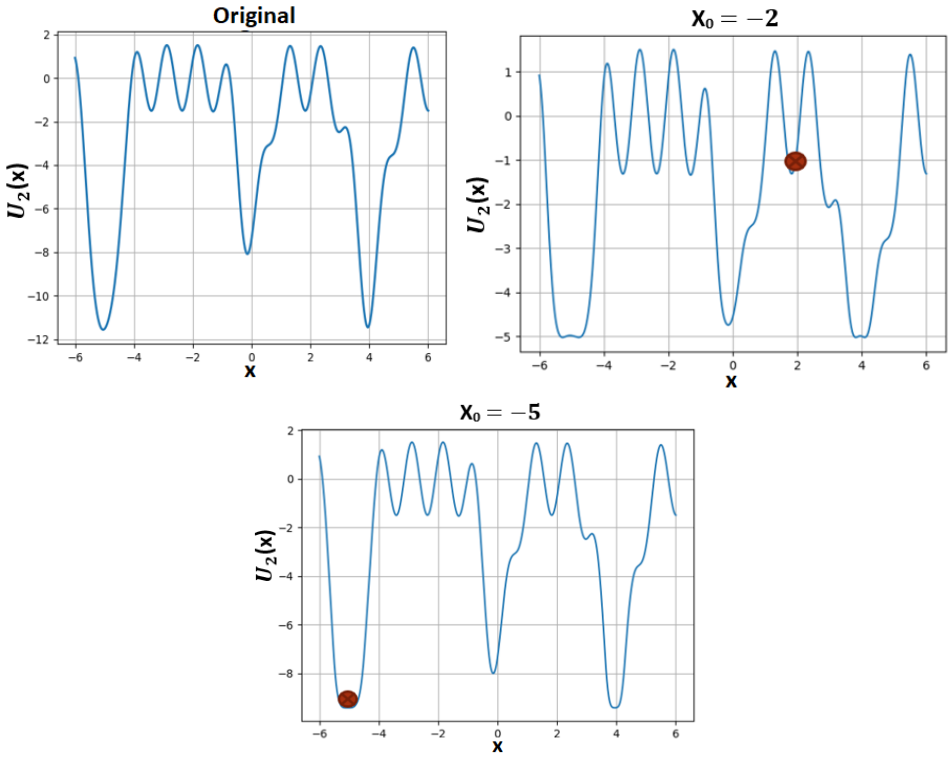

Figura 4.21: La eficiencia de GaMD depende de la posición inicial de la simulación. Se muestra el potencial original así como las FES modificadas empezando en dos coordenadas diferentes. En este ejemplo se tiene una mucho mayor aceleración cuando se empieza en $x_0 = 2$.

Además, se encontró (figura 4.18) que este método es poco efectivo cuando hay barreras energéticas con magnitudes de al menos de un orden mayor a la energía donde comienza la dinámica. Esto se relaciona con el punto anterior, pues la aceleración en GaMD parte de la exploración local de un pozo, por lo que no toma en cuenta la posibilidad de que otras zonas de la FES tengan este tipo de subidas energéticas tan pronunciadas.

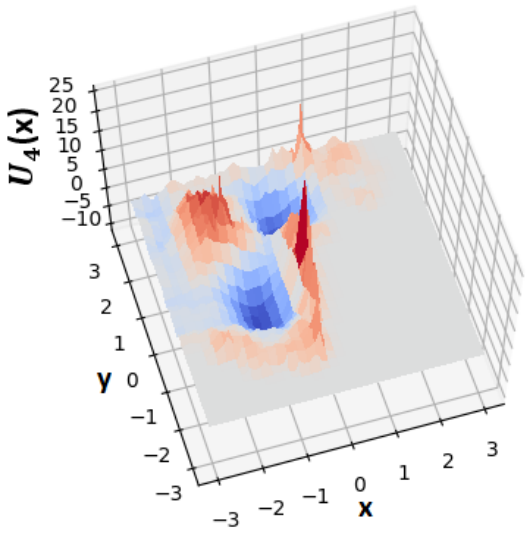

Figura 4.22: GaMD se vuelve poco eficiente cuando la FES tiene grandes barreras energéticas, en este caso hubo todo un sector que no fue muestreado.

Adicionalmente, a pesar de que GaMD ofrece la posibilidad de fijar E en su cota superior o inferior, se hicieron pruebas con ambas cotas en las FES y en todos los casos se obtuvieron resultados prácticamente idénticos sin importar la cota utilizada, por lo que, al final, se decidió utilizar solo la cota superior para los cuatro potenciales.

Por ultimo, en este método tampoco se establece un criterio de paro, por lo que no hay manera de saber en que momento se ha obtenido un muestreo lo suficientemente bueno.

4.3.3. Metadinámica

En términos de número de pasos de la dinámica, este método fue siempre el primero en escapar de la conformación inicial, así como en reconstruir por completo la FES. Además, por la forma en que hace dicha reconstrucción, no necesita un gran muestreo, ya que no utiliza ponderación: reconstruye a la vez que explora.

Como se basa en la adición de gausianas a lo largo de su recorrido, no falla en la presencia de barreras energéticas pronunciadas, ya que eventualmente alcanza su magnitud con la suma de gausianas.

Sin embargo, mientras que δx y δy son parámetros que se pueden estimar fácilmente, en general la entrada α es una variable muy sensible (cambios del 2 % alteran por completo la reconstrucción de la FES), y no existe una manera sistemática para encontrar el valor que resultará en la reconstrucción mas eficiente y de buena calidad.

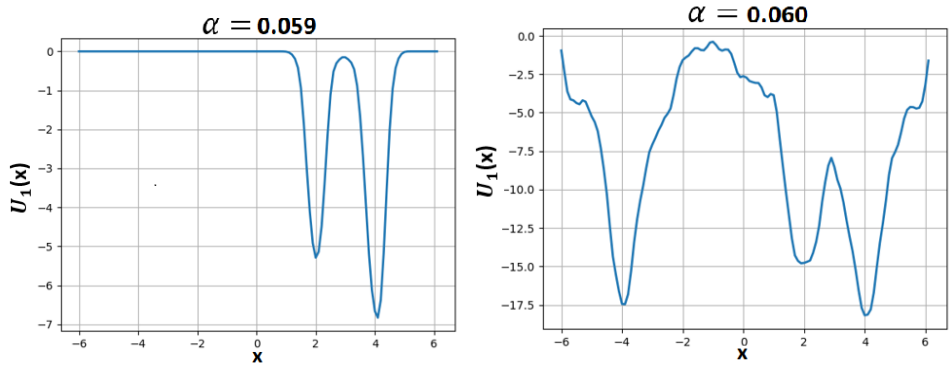

Figura 4.23: En metadinámica, el parámetro α determina en gran medida la calidad de la reconstrucción, además de que no hay un método sistemático para encontrar el mejor valor posible del mismo.

Como en los métodos anteriores, en este caso tampoco existe un criterio de paro; pero, a diferencia de GaMD e inundación, en metadinámica se altera la reconstrucción cuando se tiene un sobremuestreo. Aunque en teoría la distribución de la dinámica se va a ir aproximando a una uniforme de acuerdo a la ecuación (2.18), en la practica sucede que, dependiendo de en qué momento se detenga la dinámica, puede haber tenido un muestro mayor en ciertas zonas de la FES, lo cual afecta la reconstrucción de la misma.

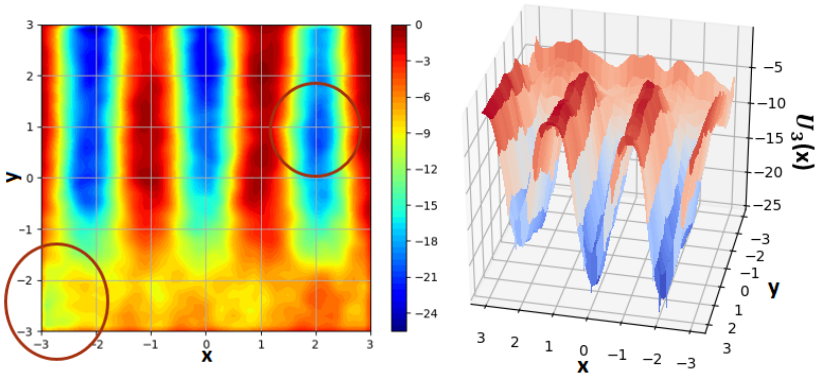

Figura 4.24: Aunque en teoría metadinámica va a converger a una distribución uniforme, en la práctica pueden haber zonas con sobremuestreo, debido a que se *corta* la dinámica antes de que reproduzca la distribución uniforme. En este ejemplo se encuentran dos sectores donde hubo una mayor acumulación de gausianas.

4.3.4. Método variacional

La calidad de la reconstrucción del método variacional (medida en términos del error cuadrático con la FES original y observando los cambios en las probabilidades de compromiso) fue en todos los casos la mejor.

Por la manera en que funciona este método, no es difícil estimar los valores de los parámetros libres. En particular, el numero de funciones a utilizar en la expansión depende simplemente de la capacidad de la computadora en la que se esté trabajando. Pues mientras más funciones se utilicen mejor será la reconstrucción de la FES.

Otra gran ventaja del método variacional es que se puede formular un criterio de paro con base en la convergencia del funcional Ω. Ya que, aunque no se conozca el valor critico del mismo, al graficar los valores que se van obteniendo de Ω se puede estimar en que momento se ha alcanzado un punto cercano al crítico.

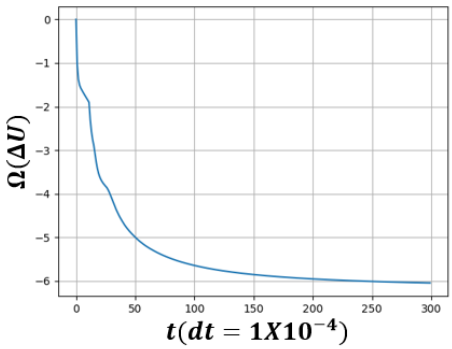

Figura 4.25: En el método variacional se puede formular un criterio de paro basado en la convergencia del funcional Ω.

Aunque en el método variacional no se necesita usar ponderación, el numero de pasos necesarios para la reconstrucción es muy grande, debido a que por cada avance en el espacio de funciones que contiene a Ω, se hace una dinámica pequeña para estimar las derivadas funcionales y poder seguir con la minimización. Por esta razón, en cada iteración de este método se necesitan realizar muchos más cálculos que en los otros tres métodos, lo cual es su mayor desventaja.
Por otro lado, dependiendo de la forma de la FES, puede ser necesario introducir varias funciones adicionales en la expansión para reproducir ciertas características del potencial original. Por ejemplo, en el potencial con perturbaciones en una dimensión, primero se habían utilizado 21 términos en la expansión de U, y se encontró que las perturbaciones en la FES no lograban reproducirse aun cuando el método convergía; en cambio, al incorporar 8 términos extra, el potencial final fue más similar a la FES original.

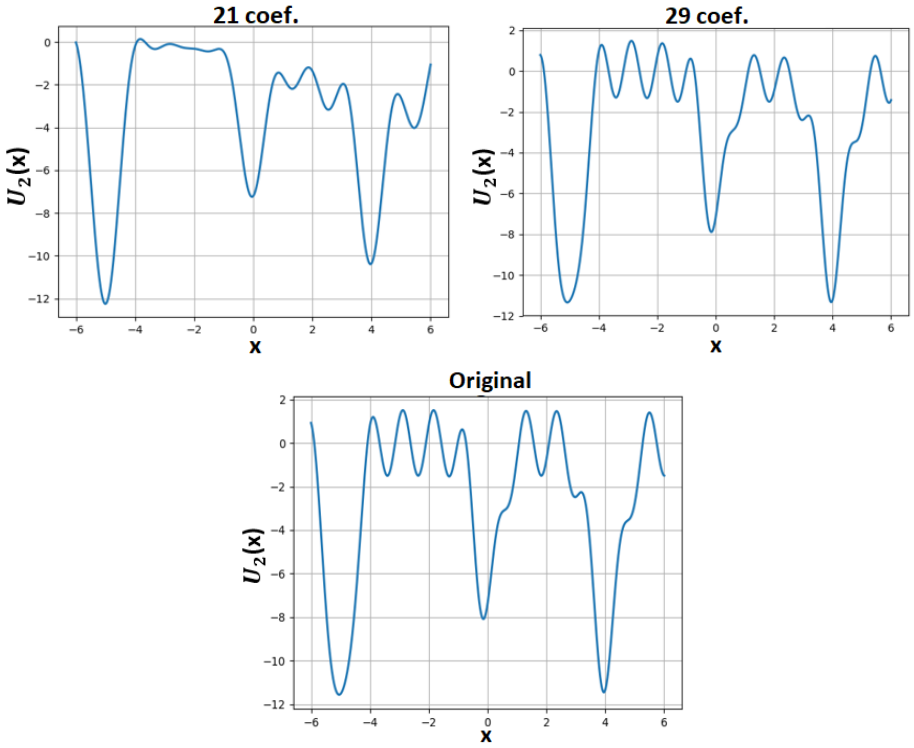

Figura 4.26: Dependiendo de la complejidad de la FES original, se tienen que incorporar más términos en la expansión de U para reproducir el comportamiento del potencial real.

Cuando el potencial no tiene condiciones de frontera periódicas, se tiene que incrementar el dominio de la FES para evitar distorsiones en la frontera al momento de hacer la reconstrucción con este método. El problema con este incremento del dominio es que aumenta también el tiempo de computo, pues es necesario hacer un muestreo aun mayor dependiendo de que tanto se alargaron los intervalos. Adicionalmente, si el dominio no se extiende demasiado, son más notables los errores en la frontera del potencial.

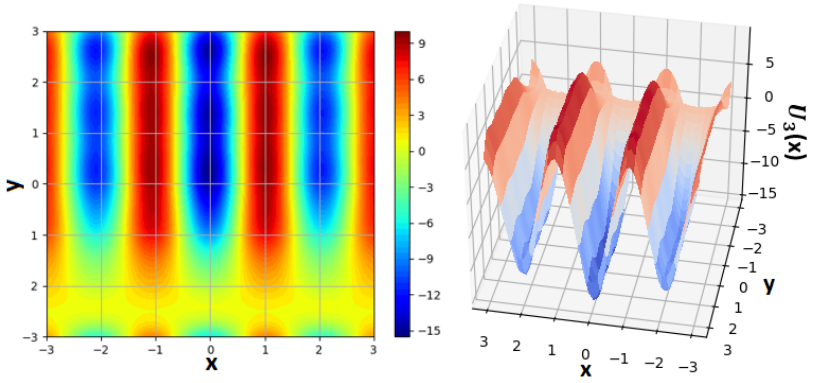

Figura 4.27: Extender el dominio de una FES no periódica aumenta el tiempo de computo pero evita los problemas de frontera. Mientras más pequeña sea dicha extensión, las distorsiones en la frontera incrementan. En este caso, hay distorsiones en la frontera definida por la recta $y = -3$.

4.3.5. Comparación

Con base en los rubros que se han analizado, se puede hacer una comparación entre los cuatro métodos de aceleración, como se muestra en la siguiente tabla:

Rubro	Mejor método	Peor método
Pasos para escapar de un mínimo local	Metadinámica	Variacional
Pasos para reconstrucción de la FES	Metadinámica	Inundación
Calidad de la reconstrucción (Error cuadrático)	Variacional	Inundación
Calidad de reconstrucción del compromiso (topología)	Variacional	Inundación
Estimación de parámetros libres	GaMD/Inundación	Metadinámica
Criterio de paro	Variacional	Metadinámica
Cantidad de cálculos necesarios	Inundación	Variacional

Tabla 4.5: Comparación de los cuatro métodos de aceleración.

También se pueden considerar, para cada metodología, sus mayores ventajas y desventajas:

Método	Mayor ventaja	Mayor desventaja
Inundación	Facilidad para implementar.	Sólo funciona de manera local.
GaMD	Facilidad para estimar parámetros libres.	Su eficiencia depende de la posición inicial.
Metadinámica	Método que escapa de la conformación inicial y reconstruye la FES más rápidamente (numero de pasos).	Parámetro libre es muy sensible.
Variacional	Hace las mejores reconstrucciones (error cuadrático, topología del compromiso).	Gran cantidad de cálculos por iteración.

Tabla 4.6: Comparación de las ventajas y desventajas de cada metodología.

Tanto GaMD como el método de inundación son los métodos que requieren un menor número de cálculos en cada iteración, además de que sus parámetros libres se estiman fácilmente. En los rubros de calidad y número de pasos necesarios para la reconstrucción, metadinámica y el método variacional fueron los mejores. Por otro lado, el método variacional es el único en el que se puede formular un criterio de paro.

Capítulo 5

Conclusiones

En este trabajo se compararon de manera cualitativa y cuantitativa cuatro métodos de muestreo acelerado (inundación, GaMD, metadinámica y variacional) utilizados en simulaciones de dinámicas moleculares, con el objetivo de determinar las cualidades y debilidades de cada uno. Para ello, se implementaron estas metodologías en distintos potenciales (2 en 1D, 2 en 2D) y se estudiaron las reconstrucciones obtenidas con cada una en términos de errores cuadráticos, pasos necesarios para la reconstrucción, probabilidades de compromiso y pasos para salir del conjunto metaestable inicial.

De los resultados obtenidos se pudo comprobar, en primer lugar, que todos los métodos son menos robustos de lo que se presenta en la literatura, especialmente cuando se utilizan en potenciales con barreras entrópicas o perturbaciones en todo su dominio. Por otro lado, de acuerdo con las métricas utilizadas en este estudio, metadinámica y el método variacional son los métodos más robustos.

Particularmente, el método variacional obtuvo las mejores reconstrucciones de todos los potenciales. Sin embargo, se encontró que una de sus mayores desventajas es la dificultad para implementarlo por la gran cantidad de cálculos que requiere al momento de minimizar Ω. Esta dificultad es inherente al algoritmo que utiliza, por lo que no es un problema que se pueda superar fácilmente.

Por otro lado, en metadinámica la mayor debilidad es la sensibilidad de α, lo que puede provocar sectores con sobremuestreo. No obstante, existe la posibilidad de resolver este problema mediante la introducción de parámetros adaptativos.

De aquí se puede ver que el estudio sistemático de estas metodologías permite idear modificaciones a las mismas e incluso generar nuevos algoritmos, lo que es de vital importancia debido a la aplicación de las mismas en el estudio de procesos biológicos así como en el diseño de fármacos, entre otros.

Apéndice A

Ponderación energética con expansión al segundo orden

Para simulaciones de dinámica molecular con un potencial U, la distribución de probabilidad sobre una variable colectiva x es una función $f(x)$. Dado un potencial de impulso $\Delta U(x)$ para todo el dominio, la distribución $f^*(x)$ asociada a $U(x) + \Delta U(x)$ se puede ponderar para recuperar la distribución canónica escribiendo

$$f(x_j) = f^*(x_j) \frac{\mathbb{E}[e^{\beta \Delta U(x)}]_j}{\sum_{j=1}^{M} \mathbb{E}[e^{\beta \Delta U(x)}]_j}, \; j = 1, ..., M, \tag{A.1}$$

donde M es el número de bines, $\beta = 1/kT$, y el factor de ponderación $\mathbb{E}[e^{\beta \Delta U(x)}]_j$ es el promedio sobre los pasos de la simulación que se encontraron en el j-ésimo bin.

Para reducir el ruido energético, el factor de ponderación se aproxima utilizando una serie de potencias[48]:

$$\mathbb{E}[e^{\beta \Delta U(x)}] = \exp \sum_{n=1}^{\infty} \frac{\beta^n}{n!} C_n, \tag{A.2}$$

donde los primeros dos términos están dados por

$$\begin{aligned} C_1 &= \mathbb{E}[\Delta U], \\ C_2 &= \mathbb{E}[\Delta U^2] - (\mathbb{E}[\Delta U])^2 = \sigma^2_{\Delta U}. \end{aligned} \tag{A.3}$$

Finalmente, la FES ponderada se calcula mediante $U(x) = -(1/\beta) \ln f(x)$

Apéndice B

Código principal utilizado en este trabajo

```
from tempfile import TemporaryFile
import matplotlib.pyplot as plt
import os
import multiprocessing as mp
import time
import math
import numpy as np
from scipy import integrate
import scipy.linalg as nps
import scipy.misc as sci

#Dinamica de Langevin con MALA
def emone(a, b, fes, beta, binos, dt, steps, mu, std, xo, name, data=None, seed=None):

    local_st = np.random.RandomState(seed)
    if data==None:
        try:
            size_space = len(fes)
            dynamic = np.zeros(steps+1)
            ds = float(b-a)/(size_space-1)
            sigma = np.sqrt(2.0/beta)
            noise = sigma*np.sqrt(dt)
            xo_ar = point_location(a, ds, xo)
            pot = fes[xo_ar]
            pot_dx = devone_array(ds, fes, xo_ar)

            for m in range(steps):
                dynamic[m] = xo
                r = local_st.normal(mu, std)
                boundary = xo - (dt*pot_dx)+(noise*r)

                if a<=boundary<=b:
                     xn = boundary
```

```
            else:
                 xn = xo

            xn_ar = point_location(a, ds, xn)
            potn = fes[xn_ar]
            potn_dx = devone_array(ds, fes, xn_ar)
            transition_one = abs(xn-xo+dt*pot_dx)**2-abs(xo-xn+dt*potn_dx)**2
            transition_two = (4.0/beta)*dt
            transition_term = np.exp(transition_one/transition_two)
            density = np.exp(beta*(pot-potn))
            val = transition_term*density

            if val > 1:
                pot, pot_dx = potn, potn_dx
                xo = xn
            else:
                if val > local_st.uniform():
                    pot, pot_dx = potn, potn_dx
                    xo = xn
        dynamic[-1] = binos
        np.save(name, dynamic)


    except TypeError:
        dynamic = np.zeros(steps+1)
        sigma = np.sqrt(2.0/beta)
        noise = sigma*np.sqrt(dt)
        pot = fes(xo)
        pot_dx = derivative_point(fes, xo)

        for m in range(steps):
            dynamic[m] = xo
            r = local_st.normal(mu, std)
            boundary = xo - (dt*pot_dx)+(noise*r)

            if a<=boundary<=b:
                xn = boundary
            else:
                xn = xo

            potn = fes(xn)
            potn_dx = derivative_point(fes, xn)
            transition_one = abs(xn-xo+dt*pot_dx)**2-abs(xo-xn+dt*potn_dx)**2
            transition_two = (4.0/beta)*dt
            transition_term = np.exp(transition_one/transition_two)
            density = np.exp(beta*(pot-potn))
            val = transition_term*density

            if val > 1:
                pot, pot_dx = potn, potn_dx
                xo = xn
            else:
                if val > local_st.uniform():
                    pot, pot_dx = potn, potn_dx
                    xo = xn
```

```
        dynamic[-1] = binos
        np.save(name, dynamic)

else:
    try:
        size_space = len(fes)
        dynamic = np.zeros(steps)
        ds = float(b-a)/(size_space-1)
        sigma = np.sqrt(2.0/beta)
        noise = sigma*np.sqrt(dt)
        xo_ar = point_location(a, ds, xo)
        pot = fes[xo_ar]
        pot_dx = devone_array(ds, fes, xo_ar)

        for m in range(steps):
            dynamic[m] = xo
            r = local_st.normal(mu, std)
            boundary = xo - (dt*pot_dx)+(noise*r)

            if a<=boundary<=b:
                 xn = boundary
            else:
                 xn = xo

            xn_ar = point_location(a, ds, xn)
            potn = fes[xn_ar]
            potn_dx = devone_array(ds, fes, xn_ar)
            transition_one = abs(xn-xo+dt*pot_dx)**2-abs(xo-xn+dt*potn_dx)**2
            transition_two = (4.0/beta)*dt
            transition_term = np.exp(transition_one/transition_two)
            density = np.exp(beta*(pot-potn))
            val = transition_term*density

            if val > 1:
                pot, pot_dx = potn, potn_dx
                xo = xn
            else:
                if val > local_st.uniform():
                    pot, pot_dx = potn, potn_dx
                    xo = xn
        return dynamic

    except TypeError:
        dynamic = np.zeros(steps)
        sigma = np.sqrt(2.0/beta)
        noise = sigma*np.sqrt(dt)
        pot = fes(xo)
        pot_dx = derivative_point(fes, xo)

        for m in range(steps):
            dynamic[m] = xo
            r = local_st.normal(mu, std)
            boundary = xo - (dt*pot_dx)+(noise*r)
```

```
                if a<=boundary<=b:
                    xn = boundary
                else:
                    xn = xo

                potn = fes(xn)
                potn_dx = derivative_point(fes, xn)
                transition_one = abs(xn-xo+dt*pot_dx)**2-abs(xo-xn+dt*potn_dx)**2
                transition_two = (4.0/beta)*dt
                transition_term = np.exp(transition_one/transition_two)
                density = np.exp(beta*(pot-potn))
                val = transition_term*density

                if val > 1:
                    pot, pot_dx = potn, potn_dx
                    xo = xn
                else:
                    if val > local_st.uniform():
                        pot, pot_dx = potn, potn_dx
                        xo = xn
            return dynamic

#METADINAMICA

#Construye grid con contribuciones de gaussianas.
def contribution_point(xgrid, fes_sample, point, gaussian_height, delta_sigma):

    x_res = xgrid[1]-xgrid[0]
    x_in = xgrid[0]
    three_sigma_array = int(round(((6.0*delta_sigma)/(x_res))))
    w = gaussian_height
    point_array = point_location(x_in, x_res, point)
    center_point = xgrid[point_array]
    exp_center = np.exp(-float((center_point-point)**2)/(2*(delta_sigma**2)))
    fes_sample[point_array] = fes_sample[point_array] + w*exp_center

    for i in range(1, three_sigma_array):
        if point_array-i>=0:
            left_point = xgrid[point_array-i]
            exp_left = np.exp(-float(abs(left_point-point)**2)/(2*(delta_sigma**2)))
            contribution_left = w*exp_left
            fes_sample[point_array-i] = fes_sample[point_array-i] + contribution_left
            continue
        else:
            continue

    for l in range(1, three_sigma_array):
        try:
            right_point = xgrid[point_array+l]
            exp_right = np.exp(-float(abs(right_point-point)**2)/(2*(delta_sigma**2)))
            contribution_right = w*exp_right
            fes_sample[point_array+l] = fes_sample[point_array+l] + contribution_right
            continue
        except IndexError:
            continue
```

```
    return fes_sample

#Metodo de metadinamica.
def mdyone(x_grid, potential, steps, initial_position, delta_sigma, beta, dt,
gaussian_height,alpha=None, data=None):

    start = time.time()

    if alpha==None:
        alfa = 0.5
    else:
        alfa = alpha

    size_space = np.size(x_grid)
    a = x_grid[0]
    b = x_grid[-1]
    ds = x_grid[2]-x_grid[1]
    sigma = np.sqrt(2.0/beta)
    noise = sigma*np.sqrt(dt)
    xo = initial_position
    fes_sample = np.zeros(size_space)
    xo_ar = point_location(a, ds, xo)
    pot_dx = derivative_point(potential, xo) + devone_array(ds, fes_sample, xo_ar)
    pot = potential(xo)+fes_sample[xo_ar]
    dinamica = []
    cont = 0

    for i in xrange(1, steps+1):
        cont = 0
        dinamica.append(xo)
        r = np.random.normal(0,1)
        boundary = xo-(pot_dx*dt)+(noise*r)

        if a<=boundary<=b:
            xn = boundary
        else:
            xn = xo
            cont = 1

        xn_ar = point_location(a, ds, xn)
        potn = potential(xn) + fes_sample[xn_ar]
        potn_dx = derivative_point(potential, xn) + devone_array(ds, fes_sample, xn_ar)
        transition_one = abs(xn-xo+dt*pot_dx)**2-abs(xo-xn+dt*potn_dx)**2
        transition_two = (4.0/beta)*dt
        transition_term = np.exp(transition_one/transition_two)
        density = np.exp(beta*(pot-potn))
        val = transition_term*density

        if val > 1:
            pot, pot_dx = potn, potn_dx
            xo = xn
        else:
            if val > np.random.uniform():
                pot, pot_dx = potn, potn_dx
```

```
                xo = xn

        if cont==0:
            fes_sample = contribution_point(x_grid, fes_sample,
            xo, gaussian_height, delta_sigma)
        else:
            continue

    end = time.time()
    duration = (1.0*(end-start))/60


    fes_samplet = np.zeros(size_space+3)
    for j in range(size_space):
        fes_samplet[j] = -fes_sample[j]
    fes_samplet[-3] = a
    fes_samplet[-2] = b
    fes_samplet[-1] = duration
    if data==None:
        np.save("mdyone_results", fes_samplet)
        np.save("mdyone_dynamic", dinamica)
    else:
        return fes_sample, duration


#INUNDACION


#Metodo de ponderacion
def reone(xgrid, fes, delta_fes, beta, binos, ndynamics, dt, steps, xo, namae):

    mu = 0
    std = 1
    a = xgrid[0]
    b = xgrid[-1]
    dx = xgrid[1]-xgrid[0]
    size_space = len(xgrid)
    def total_pot(x):
        value = delta_fes(x)+fes(x)
        return value
    trajecto = emone_para(a, b, total_pot, beta, binos, ndynamics+1, dt, steps, mu, std,
    xo, 1, 1)
    rew_simulation = trajecto[0]
    salvado = trajecto[1]
    np.save(namae,salvado)
    ndist, x = np.histogram(rew_simulation, bins=int(binos))
    ndist_normal = (1.0/np.sum(ndist))*ndist
    dsize = np.size(x)-1
    dx = x[1]-x[0]
    xeo = x[0]
    expos = np.zeros(dsize)
    contador = np.zeros(dsize)
    expos2 = np.zeros(dsize)
    expos3 = np.zeros(dsize)
    for i in range(steps):
```

```
        xnow = rew_simulation[i]
        xwhere = pointloc1(xeo, dx, xnow, dsize-1)
        exp = delta_fes(xnow)
        exp2 = (delta_fes(xnow))**2
        expos[xwhere] = expos[xwhere]+exp
        expos2[xwhere] = expos2[xwhere]+exp2
        contador[xwhere] = contador[xwhere]+1
    for k in range(dsize):
        if contador[k]==0:
            continue
        else:
            expos[k] = expos[k]/contador[k]
            expos2[k] = expos2[k]/contador[k]
    for l in range(dsize):
        expos3[l] = np.exp(beta*expos[l]+((beta**2)/2.0)*(expos2[l]-(expos[l])**2))
    expos3 = (1.0/np.sum(expos3))*expos3
    z = (ndist_normal*expos3)
    f = np.zeros(dsize)
    for t in range(dsize):
        if z[t]!=0:
            f[t] = -(1.0/beta)*np.log(z[t])
        else:
            if t==0:
                f[t] = f[t+1]
            else:
                try:
                    f[t] = (f[t-1]+f[t+1])/2.0
                except IndexError:
                    f[t] = f[t-1]

    return f

#Metodo de inundacion
def fldone(xgrid, fes, minima, flood_strengths, floods_coefficients, beta, binos,
ndynamics, dt, steps, xo, data=None):

    start = time.time()
    a = xgrid[0]
    b = xgrid[-1]
    dx = xgrid[1]-xgrid[0]
    numero_ducts = np.size(flood_strengths)
    size_space = np.size(xgrid)
    fes_sample = np.zeros(size_space)

    def flooding(x):
        flood_potential = 0
        for i in range(0, numero_ducts):
            centro = floods_coefficients[i,0]
            lambdo = floods_coefficients[i,1]
            flood_potential = flood_potential +
            (flood_strengths[i])*np.exp(-(float(lambdo)/flood_strengths[i])*((x-centro)**2))
        return flood_potential

    fes_reweight = reone(xgrid, fes, flooding, beta,
    binos, ndynamics, dt, steps, xo, "fldone_dynamic")
```

```
    end = time.time()
    duration = float(end-start)/60

    tamano = np.size(fes_reweight)
    fesgrido = np.zeros(tamano+3)
    fesgrido[-3] = a
    fesgrido[-2] = b
    fesgrido[-1] = duration
    for k in range(tamano):
        fesgrido[k] = fes_reweight[k]

    if data==None:
        np.save("fldone_results", fesgrido)
    else:
        return fes_sample, duration


#VARIACIONAL

#Valor esperado con potencial biased.
def biased_expected(function, dinamica):

    steps = np.size(dinamica)
    expected = 0
    for i in range(steps):
        xnow = dinamica[i]
        value = function(xnow)
        expected = expected+value
    expected = expected/steps
    return expected

#Covarianza en potencial con bias
def biased_covariance(f_one, f_two, dinamica):

    def product_f(x):
        return f_one(x)*f_two(x)

    biased_p = biased_expected(product_f,dinamica)
    biased_one_two = biased_expected(f_one,dinamica)*biased_expected(f_two,dinamica)
    covariance = biased_p - biased_one_two

    return covariance

#Metodo variacional
def varone(x_grid, fes, number_coeficients, iterations, mu, beta,
epsilon, binos, ndynamics, dt, steps, xo, distribution=None, cores=None, data=None):

    start = time.time()
    a = x_grid[0]
    b = x_grid[-1]
    at = x_grid[0]-epsilon
    bt = x_grid[-1]+epsilon
    avg_alpha_notnormal = np.zeros(number_coeficients)
    alpha_new = 0
    last_pos = xo
```

```
toda_tray = np.array([])


if distribution==None:
    def target_distribution(x):
        return 1.0/(bt-at)
else:
    target_distribution=distribution

meta = omega_min1(fes, target_distribution, at, bt, beta)
bajado = np.zeros(iterations)
print meta
for i in range(0, iterations):
    avg_alpha_notnormal = avg_alpha_notnormal + alpha_new
    avg_alpha = (1.0/(i+1))*avg_alpha_notnormal

    def biased(x):
        value = fes(x)+biased_potential(avg_alpha, x, at, bt)
        return value
    trayectos = emone_para(at, bt, biased, beta, binos, ndynamics,
    dt, steps, 0, 1, last_pos, 1, 1)
    mini_trayec = trayectos[0][0:-1]
    seguir = trayectos[1][0:-1]
    toda_tray = np.concatenate((toda_tray, seguir))

    a_local = min(mini_trayec)
    b_local = max(mini_trayec)
    last_pos = mini_trayec[-1]

    derivatives = der_omega_para(beta, fes, avg_alpha, target_distribution, a_local,
    b_local, at, bt, mini_trayec,cores)
    gradient, hessian = derivatives[0], derivatives[1]
    alpha_new = alpha_new - mu*(gradient+hessian*(alpha_new-avg_alpha))
    p = compro(x_grid, at, bt, fes, avg_alpha, target_distribution, beta)
    bajado[i] = p
    print (a_local, b_local, p, i)

def zv(x):
    value = np.exp(-beta*(fes(x)+biased_potential(avg_alpha, x, at, bt)))
    return value

z_v = integrate.quad(zv, at, bt)[0]
constant = (1.0/beta)*np.log(z_v)
fes_sample = [-biased_potential(avg_alpha, i, at, bt)
-(1.0/beta)*np.log(target_distribution(i))-constant for i in x_grid]
end = time.time()
duration = (1.0*(end-start))/60


np.save('varibajada',bajado)
tamano = np.size(fes_sample)
fesgrido = np.zeros(tamano+3)
fesgrido[-3] = a
fesgrido[-2] = b
fesgrido[-1] = duration
```

```
    for k in range(tamano):
        fesgrido[k] = fes_sample[k]

    if data==None:
        np.save("varone_results", fesgrido)
        np.save("varone_dynamic", toda_tray)
    else:
        return fes_sample, duration

#GAMD

#Metodo de GAMD
def gsone(x_grid, fes, sigma_o, option, beta, binos, ndynamics, dt, steps, xo, data=None):

    start = time.time()
    a = x_grid[0]
    b = x_grid[-1]
    dx = x_grid[1]-x_grid[0]
    size_space = np.size(x_grid)
    fes_almost = np.load('./emone_results.npy')
    xgrid_sample = fes_almost[0:-1]
    fes_sample = [fes(x) for x in xgrid_sample]
    avg_pot = np.mean(fes_sample)
    sigma_pot = np.std(fes_sample)
    max_pot = np.amax(fes_sample)
    min_pot = np.amin(fes_sample)
    fes_reweight = np.zeros(size_space)

    if option==1:
        ko_prime = (float(sigma_o)/sigma_pot)*(float(max_pot-min_pot)/(max_pot-avg_pot))
        ko = min(1.0, ko_prime)
        energy = max_pot
    else:
        ko_biprime = (1-float(sigma_o)/sigma_pot)*(float(max_pot-min_pot)/(avg_pot-min_pot))
        if 0<ko_biprime<=1:
            ko = ko_biprime
        else:
            ko = 1.0
        energy = min_pot+float(max_pot-min_pot)/ko

    k = ko*(1.0/(max_pot-min_pot))
    def delta_v(x):
        if fes(x)>=energy:
            return 0
        else:
            increment = float(k*((energy-fes(x))**2))/2

            return increment

    fes_reweight = reone(x_grid, fes, delta_v, beta, binos, ndynamics, dt,
    steps, xo, "gsone_dynamic")
    end = time.time()
    duration = float(end-start)/60

    tamano = np.size(fes_reweight)
```

```
    fesgrido = np.zeros(tamano+3)
    fesgrido[-3] = a
    fesgrido[-2] = b
    fesgrido[-1] = duration
    for k in range(tamano):
        fesgrido[k] = fes_reweight[k]

    if data==None:
        np.save("gsone_results", fesgrido)
    else:
        return fes_sample, duration

#COMPROMISOS

#Compromiso de a a b, dada matriz estocastica, metodo de eigenvectores.
def commitor_eig(t, a, b):

    abs_m = np.zeros(np.shape(t))
    size_space = np.size(t[0,:])
    comm_a_b = np.zeros(size_space)

    for i in xrange(size_space):
        for j in xrange(size_space):
            if i==(a-1) or i==(b-1):
                if j==i:
                    abs_m[i][j] = 1
                else:
                    pass
            else:
                abs_m[i][j] = t[i][j]

    eigenvec = power_method(abs_m, 10000)

    for k in xrange(0, size_space):
        comm_a_b[k] = float(eigenvec[k]-eigenvec[a-1])/(eigenvec[b-1]-eigenvec[a-1])

    return comm_a_b

#Da los compromisos dados dos estados a y b.
def commone(xgrid, fes, beta, a, b, method=None, data=None):

    start = time.time()

    try:
        temp = len(fes)
        delta = xgrid[1]-xgrid[0]
        astate = point_array(xgrid[0], delta, a)
        bstate = point_array(xgrid[0], delta, b)
        fes_trans = trans_matrix_1D(fes, beta)

        if method==None:
            fes_comm = commitor_alg(fes_trans, astate, bstate)
        else:
            fes_comm = commitor_eig(fes_trans, astate, bstate)
```

```
    except TypeError:
        delta = xgrid[1]-xgrid[0]
        astate = point_array(xgrid[0], delta, a)
        bstate = point_array(xgrid[0], delta, b)
        fes_grid = [fes(i) for i in xgrid]
        fes_trans = trans_matrix_1D(fes_grid, beta)

        if method==None:
            fes_comm = commitor_alg(fes_trans, astate, bstate)
        else:
            fes_comm = commitor_eig(fes_trans, astate, bstate)

    end = time.time()
    duration = float(end-start)/60

    tamano = np.size(fes_comm)
    fesgrido = np.zeros(tamano+3)
    fesgrido[-3] = xgrid[0]
    fesgrido[-2] = xgrid[-1]
    fesgrido[-1] = duration
    for k in range(tamano):
        fesgrido[k] = fes_comm[k]

    if data==None:
        np.save("commone_results", fesgrido)
    else:
        return fesgrido[0:-3], duration


#METAESTABLES

#Calcula eigenvectores dominantes.
def dominance(xgrid, fes_grid, beta):
        operator = trans_matrix_1D(fes_grid, beta)
        dominant_eigen = []
        for i in range(10):#20
            operator = np.dot(operator,operator)
        eigen = np.linalg.eig(operator)
        eigenvalues = eigen[0]
        eigenvectors = eigen[1]
        print(eigenvalues[0:5])
        for j in range(100):
            check = eigenvalues[j]
            if check > 0.9:
                dominant_eigen.append(np.real(eigenvectors[:,j]))
            else:
                break

        tomono = len(dominant_eigen)
        for t in range(tomono):
            maxo = np.max(abs(dominant_eigen[t]))
            dominant_eigen[t] = dominant_eigen[t]/maxo
            if dominant_eigen[t][0] < 0:
                dominant_eigen[t] = -dominant_eigen[t]
            else:
```

```
                continue
        return dominant_eigen

#Dados punto inicial y porcentaje de error, regresa limites de conjuntos metaestables.
def change_finder(array, start, percentage):
    actual = array[start]
    tamano = len(array)-1
    top = abs(percentage*actual)
    a = int
    b = int
    check_left = False
    check_right = False
    count_left = start
    count_right = start
    while check_left==False:
        maybe_left = array[count_left]
        if count_left==0 or abs(maybe_left-actual)>top:
            check_left=True
        else:
            count_left = count_left-1
            continue
    while check_right==False:
        maybe_right = array[count_right]
        if count_right==tamano or abs(maybe_right-actual)>top:
            check_right=True
        else:
            count_right = count_right + 1
            continue
    a = count_left
    b = count_right

    return [a,b]

#Regresa conjuntos metaestables.
def metaestable(xgrid, fes, beta, porcentaje, minima, data=None):

    try:
        tamanot = np.size(fes)
        fes_grid = np.zeros(tamanot)
        for k in range(tamanot):
            fes_grid[k] = fes[k]
    except TypeError:
        fes_grid = [fes(u) for u in xgrid]

    if data==None:
        tamano = np.size(xgrid)
        ds = xgrid[1]-xgrid[0]
        a = xgrid[0]
        dominant_eig = dominance(xgrid, fes_grid, beta)
        eignum = int(len(dominant_eig))
        min_loc = [point_location(a, ds, point) for point in minima]
        min_num = len(min_loc)
        loc_all = np.zeros((eignum,min_num,2))
        for j in range(eignum):
            for t in range(min_num):
```

```
            loc = change_finder(dominant_eig[j], min_loc[t], porcentaje)
            loc_a = loc[0]
            loc_b = loc[1]
            loc_all[j,t,0] = loc_a
            loc_all[j,t,1] = loc_b
    locations = np.zeros((2,min_num))
    for r in range(min_num):
        locations[0,r] = xgrid[int(np.max(loc_all[:,r,0]))]
        locations[1,r] = xgrid[int(np.min(loc_all[:,r,1]))]
    print(locations)
    plt.figure()
    for u in range(min_num):
        plt.axvline(x=locations[0,u], linewidth=1.5, color = 'k', linestyle=':')
        plt.axvline(x=locations[1,u], linewidth=1.5, color = 'k', linestyle=':')
    for t in range(eignum):
        plt.plot(xgrid, dominant_eig[t], linewidth=1.5)
    plt.grid()
    plt.show()
elif data=='help':
    tamano = np.size(xgrid)
    ds = xgrid[1]-xgrid[0]
    a = xgrid[0]
    dominant_eig = dominance(xgrid, fes_grid, beta)
    eignum = int(len(dominant_eig))
    min_loc = [point_location(a, ds, point) for point in minima]
    min_num = len(min_loc)
    loc_all = np.zeros((eignum,min_num,2))
    for j in range(eignum):
        for t in range(min_num):
            loc = change_finder(dominant_eig[j], min_loc[t], porcentaje)
            loc_a = loc[0]
            loc_b = loc[1]
            loc_all[j,t,0] = loc_a
            loc_all[j,t,1] = loc_b
    locations = np.zeros((2,min_num))
    for r in range(min_num):
        locations[0,r] = xgrid[int(np.max(loc_all[:,r,0]))]
        locations[1,r] = xgrid[int(np.min(loc_all[:,r,1]))]
    return locations
else:
    tamano = np.size(xgrid)
    ds = xgrid[1]-xgrid[0]
    a = xgrid[0]
    dominant_eig = dominance(xgrid, fes_grid, beta)
    eignum = int(len(dominant_eig))
    min_loc = [point_location(a, ds, point) for point in minima]
    min_num = len(min_loc)
    loc_all = np.zeros((eignum,min_num,2))
    for j in range(eignum):
        for t in range(min_num):
            loc = change_finder(dominant_eig[j], min_loc[t], porcentaje)
            loc_a = loc[0]
            loc_b = loc[1]
            loc_all[j,t,0] = loc_a
            loc_all[j,t,1] = loc_b
```

```
locations = np.zeros((2,min_num))
for r in range(min_num):
    locations[0,r] = xgrid[int(np.max(loc_all[:,r,0]))]
    locations[1,r] = xgrid[int(np.min(loc_all[:,r,1]))]
print(locations)
plt.figure()
for u in range(min_num):
    plt.axvline(x=locations[0,u], linewidth=1.5, color = 'k', linestyle=':')
    plt.axvline(x=locations[1,u], linewidth=1.5, color = 'k', linestyle=':')
for t in range(eignum):
    plt.plot(xgrid, dominant_eig[t], linewidth=1.5)
plt.grid()
plt.xlabel("Coordenada Colectiva")
plt.ylabel("Eigenvalores Dominantes")
plt.title("Conjuntos metaestables")
plt.savefig('graph.png')
```

Bibliografía

[1] James E Darnell. Molecular cell biology. Technical report, 1986.

[2] Katherine Henzler-Wildman and Dorothee Kern. Dynamic personalities of proteins. *Nature*, 450(7172):964, 2007.

[3] Laura Westergard, Heather M Christensen, and David A Harris. The cellular prion protein (prpc): its physiological function and role in disease. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1772(6):629–644, 2007.

[4] Stanley B Prusiner. Prions. *Proceedings of the National Academy of Sciences*, 95(23):13363–13383, 1998.

[5] Mad-Cow. `http://www.mad-cow.org/`.

[6] Alexander Rich and Norman Davidson. Structural chemistry and molecular biology. 1968.

[7] Alessandro Laio, Antonio Rodriguez-Fortea, Francesco Luigi Gervasio, Matteo Ceccarelli, and Michele Parrinello. Assessing the accuracy of metadynamics. *The journal of physical chemistry B*, 109(14):6714–6721, 2005.

[8] CP2K. `https://www.cp2k.org/exercises:2014_ethz_mmm:alanine_dipeptide`.

[9] Martin Karplus and J Andrew McCammon. Molecular dynamics simulations of biomolecules. *Nature Structural and Molecular Biology*, 9(9):646, 2002.

[10] David E Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, Michael P Eastwood, Joseph A Bank, John M Jumper, John K Salmon, Yibing Shan, et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, 2010.

[11] Oliver F Lange, Lars V Schäfer, and Helmut Grubmüller. Flooding in gromacs: Accelerated barrier crossings in molecular dynamics. *Journal of computational chemistry*, 27(14):1693–1702, 2006.

[12] Donald Hamelberg, John Mongan, and J Andrew McCammon. Accelerated molecular dynamics: a promising and efficient simulation method for biomolecules. *The Journal of chemical physics*, 120(24):11919–11929, 2004.

[13] Alessandro Laio and Francesco L Gervasio. Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science. *Reports on Progress in Physics*, 71(12):126601, 2008.

[14] Michele Parrinello. A variational approach to enhanced sampling and free energy calculations. In *APS Meeting Abstracts*, 2015.

[15] James B Clarage, Tod Romo, B Kim Andrews, B Montgomery Pettitt, and George N Phillips. A sampling problem in molecular dynamics simulations of macromolecules. *Proceedings of the National Academy of Sciences*, 92(8):3288–3292, 1995.

[16] Edwin Atlee Jackson. *Equilibrium statistical mechanics*. Courier Corporation, 2000.

[17] Ryogo Kubo. *Thermodynamics: an advanced course with problems and solutions*, volume 1. North-Holland, 1968.

[18] Raymond Chang and Jason Overby. *General chemistry: the essential concepts*. Mc Graw Hill, 2000.

[19] Wikipedia. `https://en.wikipedia.org/wiki/Amino_acid`.

[20] Erik R Lindahl. Molecular dynamics simulations. In *Molecular modeling of proteins*, pages 3–23. Springer, 2008.

[21] James M Haile. *Molecular dynamics simulation: elementary methods*, volume 1. Wiley New York, 1992.

[22] James M Haile. *Molecular dynamics simulation: elementary methods*, volume 1. Wiley New York, 1992.

[23] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier, 2001.

[24] Jesus A Izaguirre, Daniel P Catarello, Justin M Wozniak, and Robert D Skeel. Langevin stabilization of molecular dynamics. *The Journal of chemical physics*, 114(5):2090–2098, 2001.

[25] Tamar Schlick. *Molecular modeling and simulation: an interdisciplinary guide*, volume 21. Springer Science & Business Media, 2010.

[26] Gareth O Roberts, Richard L Tweedie, et al. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[27] Hans C Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *The Journal of chemical physics*, 72(4):2384–2393, 1980.

[28] Nawaf Bou-Rabee and Eric Vanden-Eijnden. Pathwise accuracy and ergodicity of metropolized integrators for sdes. *Communications on Pure and Applied Mathematics*, 63(5):655–696, 2010.

[29] Denis Talay and Luciano Tubaro. Expansion of the global error for numerical schemes solving stochastic differential equations. *Stochastic analysis and applications*, 8(4):483–509, 1990.

[30] Martin Hutzenthaler, Arnulf Jentzen, Peter E Kloeden, et al. Strong convergence of an explicit numerical method for sdes with nonglobally lipschitz continuous coefficients. *The Annals of Applied Probability*, 22(4):1611–1641, 2012.

[31] Jonathan C Mattingly, Andrew M Stuart, and Desmond J Higham. Ergodicity for sdes and approximations: locally lipschitz vector fields and degenerate noise. *Stochastic processes and their applications*, 101(2):185–232, 2002.

[32] Manuel Luitz, Rainer Bomblies, Katja Ostermeir, and Martin Zacharias. Exploring biomolecular dynamics and interactions using advanced sampling methods. *Journal of Physics: Condensed Matter*, 27(32):323101, 2015.

[33] Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, and David E Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

[34] Helmut Grubmüller. Predicting slow structural transitions in macromolecular systems: Conformational flooding. *Physical Review E*, 52(3):2893, 1995.

[35] Yinglong Miao, William Sinko, Levi Pierce, Denis Bucher, Ross C Walker, and J Andrew McCammon. Improved reweighting of accelerated molecular dynamics simulations for free energy calculation. *Journal of chemical theory and computation*, 10(7):2677–2689, 2014.

[36] Yinglong Miao, Victoria A Feher, and J Andrew McCammon. Gaussian accelerated molecular dynamics: Unconstrained enhanced sampling and free energy calculation. *Journal of chemical theory and computation*, 11(8):3584–3595, 2015.

[37] Alessandro Laio, Antonio Rodriguez-Fortea, Francesco Luigi Gervasio, Matteo Ceccarelli, and Michele Parrinello. Assessing the accuracy of metadynamics. *The journal of physical chemistry B*, 109(14):6714–6721, 2005.

[38] Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the National Academy of Sciences*, 99(20):12562–12566, 2002.

[39] Walter Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 1987.

[40] Omar Valsson and Michele Parrinello. Variational approach to enhanced sampling and free energy calculations. *Physical review letters*, 113(9):090601, 2014.

[41] Philip N Sabes and Michael I Jordan. Advances in neural information processing systems. In *In G. Tesauro & D. Touretzky & T. Leed (Eds.), Advances in Neural Information Processing Systems*. Citeseer, 1995.

[42] Patrick Shaffer, Omar Valsson, and Michele Parrinello. Enhanced, targeted sampling of high-dimensional free-energy landscapes using variationally enhanced sampling, with an application to chignolin. *Proceedings of the National Academy of Sciences*, 113(5):1150–1155, 2016.

[43] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[44] James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.

[45] Christof Schütte and Wilhelm Huisinga. Biomolecular conformations can be identified as metastable sets of molecular dynamics. *Handbook of numerical analysis*, 10:699–744, 2003.

[46] Y Ueda, H Taketomi, and N Go. Studies on protein folding, unfolding and fluctuations by computer simulation. i. the effects of specific amino acid sequence represented by specific inter-unit interactions. *Int. J. Peptide Res*, 7:445–459, 1975.

[47] Jan-Hendrik Prinz, Martin Held, Jeremy C Smith, and Frank Noé. Efficient computation, sensitivity, and error analysis of committor probabilities for complex dynamical processes. *Multiscale Modeling & Simulation*, 9(2):545–567, 2011.

[48] Gerhard Hummer. Fast-growth thermodynamic integration: Error and efficiency analysis. *The Journal of Chemical Physics*, 114(17):7330–7337, 2001.