Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico-Matemáticas

Cálculo del potencial energético de redes de regulación celular por métodos Monte Carlo y su representación de red neuronal artificial

Tesis presentada al

Posgrado en Física Aplicada

como requisito parcial para la obtención del grado de

Maestra en Ciencias

(Física Aplicada)

por

Natalia López Paleta

asesorada por

Dr. Jorge Velázquez Castro, Dr. Eduardo Moreno Barbosa

Puebla Pue.
2020

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico-Matemáticas

Cálculo del potencial energético de redes de regulación celular por métodos Monte Carlo y su representación de red neuronal artificial

Tesis presentada al

Posgrado en Física Aplicada

como requisito parcial para la obtención del grado de

Maestra en Ciencias

(Física Aplicada)

por

Natalia López Paleta

asesorada por

Dr. Jorge Velázquez Castro, Dr. Eduardo Moreno Barbosa

Puebla Pue.
2020

Título: Cálculo del potencial energético de redes de regulación celular por métodos Monte Carlo y su representación de red neuronal artificial

Estudiante: Natalia López Paleta

COMITÉ

Dr. Beatriz Bonilla Capilla
Presidente

Dr. Benito De Celis Alonso
Secretario

Dr. Andrés Anzo Hernández
Vocal

Dr. Jorge Velázquez Castro, Dr. Eduardo Moreno Barbosa
Asesor

Índice general

Índice de figuras

Resumen

Las enfermedades crónico degenerativas son responsables del 63 % de las muertes en todo el mundo y representan altos costos en la economía de un país, especialmente en aquellos países en vías de desarrollo. Hasta hace algunos años, se creía que bastaba conocer las unidades básicas que determinan el funcionamiento de las células vivas, llamadas genes, para prevenir y tratar este tipo de enfermedades. Sin embargo, ahora se sabe que los factores ambientales, fisiológicos y conductuales también influyen de manera importante en la aparición de las ENT(enfermedades no transmisibles).
Es por este motivo que se ha vuelto indispensable el entender la relación entre los factores de riesgo, los rasgos genéticos de un individuo y la aparición y progresión de una enfermedad para la generación de mejores tratamientos.

La Biología Sistémica estudia esas relaciones desde el punto de vista de los Sistemas Dinámicos, es decir, modelando a la célula, el tejido o, incluso, el individuo como un sistema complejo que evoluciona en el tiempo. Al analizar la evolución temporal de este tipo de sistemas, se podría predecir el conjunto de factores que pueden llevar al sistema a un estado de enfermedad o de salud. El paisaje epigenético es un concepto sumamente útil para el análisis de este tipo de sistemas, y tiene las propiedades de un potencial energético.

En este trabajo se diseñará y entrenará una red neuronal artificial que represente el paisaje epigenético de un sistema mínimo biestable. Para ello, se resolverá la ecuación de Fokker- Planck asociada al sistema por medio de un método Monte Carlo. Posteriormente, se entrenará la red neuronal con los datos generados de la integración de la ecuación de Fokker-Planck.

Introducción

La vida, y todos los mecanismos que la hacen posible, han sido uno de los principales intereses de estudio del ser humano desde que éste comenzó a sistematizar y a registrar el conocimiento. Por otro lado, uno de los principales objetivos de la ciencia ha consistido en implementar los conocimientos que de ella surgen para mejorar la calidad de vida del ser humano.

La física, puesto que estudia todo lo relativo a la naturaleza, también ha tenido como objeto de estudio a los sistemas vivos y a los fenómenos que ocurren dentro de los organismos y ha desarrollado herramientas y conocimientos aplicables a la salud humana. En particular, para modelar a algunos sistemas biológicos, una de esas herramientas, de la que se hará uso en este trabajo de tesis, es la de los Sistemas Dinámicos.

La *Biología de Sistemas* es el enfoque teórico que estudia a los sistemas vivos y los factores, tanto internos como externos, que conducen al sistema en su desarrollo o modifican su dinámica. Desde esta perspectiva, es decir, viendo a un organismo como un sistema dinámico, que puede ser descrito por medio de ecuaciones matemáticas, una de las preguntas básicas que se han planteado es ¿cómo se relaciona la genotipo con el fenotipo? o, lo que es lo mismo, ¿qué expresiones matemáticas nos dan la relación que existe entre la expresión genética y las características observables de ese sistema?

Para responder a estas pregunta, distintos investigadores [2],[5], [16], [17], [19], [35] han optado por estudiar a la célula y los fenómenos biológicos básicos que tienen lugar dentro y alrededor de ella.

El concepto de gen surge a raíz de los experimentos de Mendel. Él plantea una unidad básica de información (que llama factores) que transmite las características de los padres a los hijos y que, al unirse con otros genes ordenadamente, da lugar a los cromosomas. Posteriormente se planteó al gen como un segmento de cadena del ADN encargado de dirigir la síntesis de una proteína.

Estas ideas son las que dan origen a la *Era genómica* en la que los investigadores sumaron esfuerzos para secuenciar el genoma humano con la idea de que, al conocer las partes más elementales que contienen toda la información de un organismo, podrían "reparar" aquellas que dieran origen a enfermedades. Sin embargo, ahora se sabe que el surgimiento y la progresión de una enfermedad no depende únicamente de la información genética del individuo sino también,

y en gran medida, de los factores ambientales en los que se desarrolla. Es por esto que se vuelve importante el estudio de esos factores y de cómo modifican el desarrollo de un organismo y qué variables pueden llevar al organismo de un estado de salud a uno de enfermedad o viceversa.

Así una red de regulación genética es un conjunto de genes dentro de una célula que, al interactuar entre sí y con otras sustancias a su alrededor, potencian o inhiben la producción de ciertas proteínas y, por lo tanto, la expresión de esos u otros genes. Estas redes han sido objeto de estudio desde hace aproximadamente medio siglo pues, del correcto funcionamiento de ellas, depende, en gran medida, el estado de salud general de un individuo.

Se han empleado diversos enfoques para hacer una descripción de ellas y así lograr predicciones más acertadas de ese proceso,[5],[16],[17]. Un enfoque consiste en establecer una serie de ecuaciones diferenciales que describen las concentraciones de las distintas sustancias involucradas en la regulación. En 1969, Kauffman[17] estudia este tipo de sistemas desde el enfoque de la teoría de redes y encuentra ciertas conclusiones sobre estabilidad del sistema en función de las propiedades topológicas de la red de regulación genética. Por otro lado, puesto que aspectos importantes al relacionar una red de regulación genética con las características de los organismos dependen de la topología y la dinámica de dicha red, ésta se vuelve muy compleja.

Para reflejar las restricciones morfogenéticas derivadas de las interacciones entre los genes de la red, Waddington [33] propuso en 1957, el concepto de paisaje epigenético el cual es clave para abordar aspectos globales importantes como la robustez de una red celular[19]. De esta manera, si se conoce el paisaje epigenético de una red de regulación celular se pueden explorar las propiedades globales del sistema y la estabilidad de las cuencas de atracción.

En particular, en Kim and Wang [19] se propone un método para construir el paisaje epigenético probabilistico empleando la aproximación de Hartree en la ecuación maestra. Sin embargo, este método se vuelve poco práctico para muchas dimensiones. Cuando hay un gran número de dimensiones, se vuelve necesario utilizar métodos numéricos para encontrar una solución que nos permita conocer, a partir de ella, el paisaje epigenético de una red de muchos nodos.

El método Monte Carlo para las solución de ecuaciones diferenciales parciales parabólicas, es adecuado para sistemas de muchas dimensiones. Es decir, los métodos de Monte Carlo se traducen en el uso deliberado de números aleatorios en un cálculo. Dos áreas principales de aplicación están en la mecánica estadística y en la teoría de transporte cinético lineal como, por ejemplo, el flujo de tráfico, las finanzas, la genética, etc. que se basan en el estudio de muchos procesos estocásticos interesantes imitando los procesos aleatorios directamente en la computadora.

Por otro lado, los algoritmos de inteligencia artificial, a los cuales pertenecen las redes neuronales artificiales, nacen de la necesidad de resolver problemas demasiado complejos como para resolverlos por medio de una programación explicita. Aunque la primera aparición formal de las redes neuronales artificiales data de alrededor de 1940, cuando McCulloch and Pitts [21] proponen un modelo computacional inspirado en el funcionamiento del cerebro humano, no es sino hasta la

década de los 80 del siglo pasado que comienza una ardua investigación de parte de académicos de multiples disciplinas para encontrar métodos de optimización de las RNAs. Y, a lo largo de estos últimos años, estas herramientas computacionales han probado ser de gran utilidad por su flexibilidad para resolver numerosos tipos de problemas [4], [11], [13], [14].

Capítulo 1

La Biología Sistémica

Según la Organización Mundial de la Salud las enfermedades crónico-degenerativas son responsables del 63 % de las muertes en todo el mundo y representan un gasto público elevado. A pesar de esto son prevenibles si se crean medidas que aborden los factores de riesgo que conllevan a ellas. Para crear estas medidas es necesario entender cómo influye cada uno de los factores de riesgo a la aparición y a la progresión de estas enfermedades en un individuo.

Los genes fueron concebidos, primero, por Gregor Mendel. En sus experimentos propone la existencia de ciertas unidades elementales que fueran las responsables de transmitir la herencia de padres a hijos. Posteriormente, cuando en 1953, James Watson, Francis Crick, Maurice Wilkins y Rosalind Franklin descubren la estructura de doble hélice del ADN, se define el gen como una cadena de ADN que dirige la síntesis de una proteína a través del proceso de transcripción con el ARN mensajero, con lo que quedaría establecida la idea de que a un gen corresponde una proteína y viceversa. A ésta idea se le llamó *Dogma central de la biología molecular*.

Así, en la *Era genómica*, que comienza con el descubrimiento de la estructura de doble hélice del ADN en 1953 y concluye a inicios del siglo XXI con la secuenciación del genoma humano, se creía que, para entender el funcionamiento de un organismo vivo, era suficiente con conocer las partes elementales que lo conforman. De esta manera, se podría establecer un vinculo entre los componentes genéticos de un individuo y ciertos padecimientos para poder, entonces, crear tratamientos para curar o prevenir enfermedades.

Sin embargo, actualmente se cuenta con bastante más información sobre la estructura y el funcionamiento de los genes y se sabe que los mecanismos que llevan a cabo son más complejos de lo que se creía en un principio. No es el gen el que, con la información que contiene, determina por completo el desarrollo de una célula, un tejido o un organismo; sino que, cada uno de los componentes genéticos de un organismo, junto con cada uno de los factores externos a él, influyen en su desarrollo. Así pues, no basta con que un individuo cuente con cierto gen para contraer una enfermedad sino que los factores de riesgo ambientales, fisiológicos y conductuales juegan un papel muy importante en el estado de salud de una persona.

El reto, entonces, es establecer enfoques que modelen cómo todos esos factores de riesgo (genéticos, moleculares, ambientales, fisiológicos, etc.) guían o modifican el desarrollo de un sistema vivo y cómo los cambios en éstos pueden llevar al organismo de un estado saludable a uno de enfermedad o viceversa.

Hasta ahora el desarrollo de las enfermedades no transmisibles y la relación entre ellas y los factores ambientales se ha estudiado mediante análisis estadísticos. Sin embargo, la *Biología Sistémica* propone un enfoque novedoso para abordar esos retos y estudia las funciones biológicas complejas desde un modelo basado en redes, generando, así, conocimiento cuantitativo y predecible sobre los procesos celulares complejos (como, por ejemplo, el metabolismo, el ciclo celular, la apoptosis, etc.). Además de que contribuye a establecer enfoques científicos que ayuden a prevenir y a crear tratamientos contra las enfermedades crónicas.

En Alberghina and Westernhoff [1] se define a la Biología Sistémica como "La ciencia que descubre los principios subyacentes al surgimiento de las propiedades funcionales de organismos vivos a partir de las interacciones ente macromoléculas ". Así pues, el enfoque que se propone Alvarez-Buylla et al. [3] es un enfoque que, partiendo de los datos (tanto genéticos como no genéticos), implicados en la salud humana, que se han acumulado en las últimas décadas, modela las interacciones regulatorias complejas que provocan las transiciones de un estado celular a otro y que están relacionados con la aparición y la progresión de enfermedades complejas.

1.1. El paisaje epigenético

En 1957, Conrad H. Waddington publica un libro titulado "The Strategy of genes"[33] en el que plantea la imagen del *paisaje epigenético*. Con esto sentaría las bases para el desarrollo de lo que ahora llamamos Biología Sistémica.

En ese entonces la estructura de doble hélice del ADN acababa de ser descubierta y el mayor interés, en la biología, era el estudio de la genética. Muchos científicos de la época se volcaron a estudiar, desde distintos ángulos, el ADN, los genes y su funcionamiento. Con esto aspiraban a descubrir los mecanismos por los cuales, cada gen, de forma individual, determina el desarrollo de un organismo. Este enfoque del estudio de los sistemas vivos llevó a la biología a una visión reduccionista de su campo de estudio[1]: al conocer las partes elementales de un organismo y su funcionamiento se esperaba entender, inmediatamente, el funcionamiento del organismo entero y, en particular, el origen de la salud y la enfermedad humanas.

Así pues, la importancia de la publicación de Waddington radica en que, en ella, se plantea un cambio de paradigma en la biología. No es la intención de su planteamiento el demeritar los avances y los descubrimientos que la genética estaba logrando, ya que ellos, dice el autor, son importantes y relevantes en sí, pero no toman en cuenta la característica fundamental del objeto de estudio de la biología: la vida. Y es por esto que hacía falta un marco teórico que

[1]Sobre los origenes y las implicaciones de este enfoque de la biología se puede leer en Marcos [20].

estuviera en concordancia con los datos obtenidos experimentalmente pero que tomara en cuenta las características que determinan a los organismos vivos: la *Forma* y el *Fin*[2].

Waddington define a la *Forma* como la característica de todos los organismo de poseer cierta "completitud" o "integración" que los hace autosuficientes. No es una configuración que toma el organismo y permanece estática sino que se mantiene a pesar de que el material, del cual está hecho, fluye continuamente a través de él. El *Fin*, por otro lado, se refiere a la "direccionalidad" que posee el organismo, es decir, a la capacidad de desarrollarse y organizarse con cierto fin y que no se limita a transmitir sus genes a nuevas generaciones sino que tiene un carácter menos general y está relacionado a las operaciones de determinadas partes del organismo.

De esta manera, un modelo que provea de un entendimiento más completo acerca de los sistemas vivos debería responder a las preguntas: "¿cómo, el desarrollo, produce entidades que tienen Forma, en el sentido de integración o completitud? y ¿cómo, la evolución, trae a la existencia organismos que tienen Fin, en el sentido de búsqueda de objetivos o de direccionalidad?"[3].

Y para responder estas preguntas, Waddington plantea estas tres consideraciones:

- En el desarrollo embrionario, y para ciertos procesos, una pequeña variación en las condiciones iniciales puede ocasionar un efecto "exagerado" en los estados estables del sistema en cuestion.
- En el desarrollo embrionario los productos finales terminan siendo esencialmente distintos. No existen(salvo muy pocas excepciones) estados intermedios.
- La trayectoria por el cual el embrión se desarrolla es la "más favorecida". Si algo forzara al sistema a salirse de su camino, éste experimentará un comportamiento regulador para regresar a su trayectoria "normal".

Con éstas ideas en mente Waddington propone modelar el desarrollo de una célula, estableciendo ecuaciones diferenciales que describan la evolución temporal de las concentraciones de las sustancias que lo modifican. Por ejemplo, en un proceso autocatalítico, si P y Q son sustancias que se forman a partir de los materiales "crudos" A y B, y B y C, respectivamente; la evolución en el tiempo de cada una de estas sustancias estará regida por las ecuaciones

$$\frac{dA}{dt} = k(a - A) - k_1 PAB + k_2 P^2 \tag{1.1}$$

$$\frac{dB}{dt} = k(b - B) - k_1 PAB + k_2 P^2 - k_1 QBC + k_2 Q^2 \tag{1.2}$$

$$\frac{dC}{dt} = k(c - C) - k_1 QBC + k_2 Q^2 \tag{1.3}$$

$$\frac{dP}{dt} = k_1 PAB - k_2 P^2 + k_3 P \tag{1.4}$$

$$\frac{dQ}{dt} = k_1 QCB - k_2 Q^2 + k_3 Q \tag{1.5}$$

[2]Waddington también menciona a la *Mente* como una tercer características, sin embargo, él no trata en su libro sobre ella y tampoco hablaremos de ella en este trabajo.

[3]Tomado y traducido de Waddington [33]

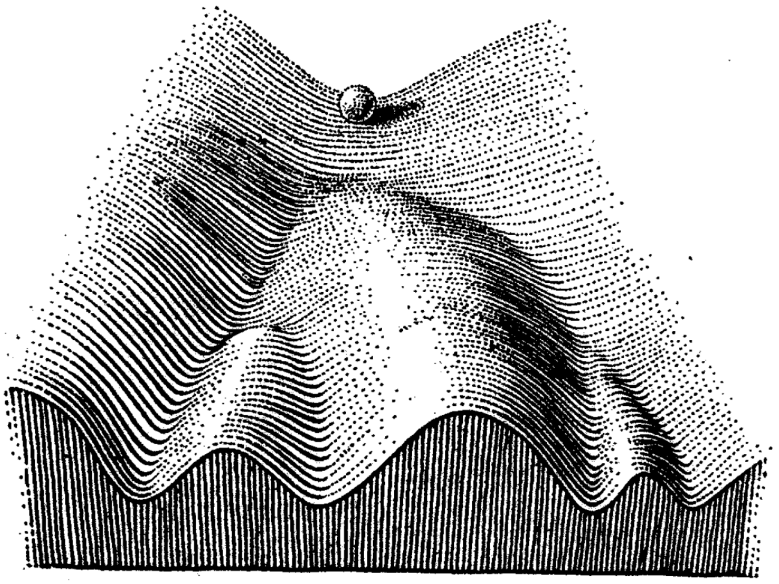


Figura 1.1: *Imagen tomada del libro de Waddington [33] que representa a un paisaje epigenético.*

con a, b y c las concentraciones de los materiales A, B y C fuera del sistema.

En este caso, podemos observar que, para este tipo de procesos, una pequeña variación en las condiciones iniciales puede ocasionar un efecto "exagerado" en los estados estables del sistema en cuestión. De esta misma manera, los genes y su actividad pueden ser entendidos a través de procesos autocatalíticos pues producen sus productos primarios y además llevan a cabo la autoreduplicación.

Hay que tomar en cuenta que, además de el estado final del proceso de desarrollo, son de interés los trayectos por los cuales pasa el organismo antes de llegar a él. Así, para ilustrar todo el proceso, Waddington propone visualizarlo de manera geométrica por medio del *paisaje epigenético*. Éste consiste en una superficie irregular, con montañas y valles sobre el cual se mueve una bola (ver figura 1.1) y el camino seguido por la bola estaría condicionado al estado inicial del sistema, es decir, el punto sobre el que comience, la bola, su trayectoria.

Así, cada valle representaría un estado final del proceso, es decir, una característica fenotípica bien definida del organismo; las montañas, estados aberrantes intermedios entre un estado final y otro; la forma de cada valle nos daría una idea de qué tan probable es que el organismo salga de un estado final y llegue a otro, etc.

Y aunque esta imagen tridimensional es válida para un sistema con un espacio de estados de dos dimensiones, es perfectamente aplicable para dimensionalidades más altas. Además, aunque Waddington propone la imagen del paisaje epigenético como eso, una imagen mental en la cual apoyarse, podemos intuir que cada valle, entre mas profundo y escarpado sea, tiene más probabilidades de recibir a la bola; es decir, representa un estado al que es sistema es más probable

que llegue después de determinado tiempo y, también, un estado del que le será más difícil salir: es una cuenca de atracción de un sistema dinámico.

De esta manera, podemos concluir que el paisaje epigenético es análogo al potencial energético de un sistema dinámico.

1.2. Red de regulación genética

Como ya se mencionó anteriormente los mecanismos por los cuales los genes definen el desarrollo de un organismo, no son sencillos. La relación entre la producción de las proteínas dentro de una célula y la activación o desactivación de sus genes no es lineal como se creía al inicio de la Era genómica.

Sin embargo, es cierto que la activación o desactivación de los genes depende de las concentraciones de distintas sustancias, en particular proteínas. También es cierto que los genes influyen fuertemente en la producción de proteínas, y por lo tanto en la concentración de ellas dentro de la célula, aunque estas relaciones no sean lineales.

Así pues podemos pensar que una célula lleva a cabo, constantemente, un proceso con el que modifica las concentraciones de ciertas sustancias dentro y fuera de ella. Además, puesto que, dependiendo de la concentración de algunas proteínas, se pueden activar o desactivar ciertos genes, este proceso es llamado de *Regulación genética*.

En términos muy simples, en las etapas tempranas del desarrollo celular, si cierta hormona H se produce en los alrededores de la célula, ésta puede entra a ella. Ya dentro, la hormona se une a una proteína receptora P_1 para, juntas, entrar al núcleo de la célula. Esté complejo receptor hormonal $H+P_1$, se une al ADN y, con ello, se activa o desactiva cierto gen G. Finalmente, el ADN transcribe la información del complejo receptor hormonal al ARNm que sale del núcleo celular y promueve la creación de una nueva proteína P_2 que, a su vez, puede promover otro proceso similar. Ver figura 1.2

Por lo tanto, podríamos decir que la concentración de la proteína P_1 promueve la activación o inhibición de cierto gen G_1 que, a su vez, promueve la producción de una nueva proteína P_2 y, posteriormente la activación o inhibición de otro gen G_2. Con esto, la célula guía su desarrollo hacia alguna característica fenotípica en particular y, este proceso, depende tanto de la información contenida en su genoma como en los estimulantes ambientales a su alrededor.

Así pues, Kauffman [17] propone modelar a esta concatenación de procesos, como una red en la que los nodos representarían a los genes con dos posibles estados: activos o inactivos, y que estarían unidos a otros genes de los que dependería ese estado o, bien, a los que modificarían. Es por esto que a este proceso también se le suele llamar *Red de regulación genética*. Ver figura 1.3

Por otro lado, como ya se había mencionado, este proceso puede ser descrito con un conjunto de ecuaciones diferenciales como (1.1)-(1.5) y es posible modelarlo por medio de su paisaje

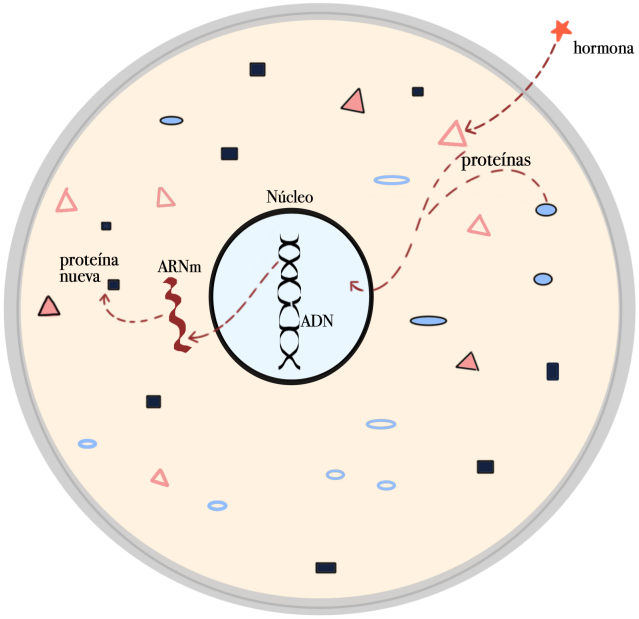


Figura 1.2: *Representación de un proceso de regulación genética en el que las proteínas, triangular y ovalada, modifican la concentración de la proteína cuadrada.*

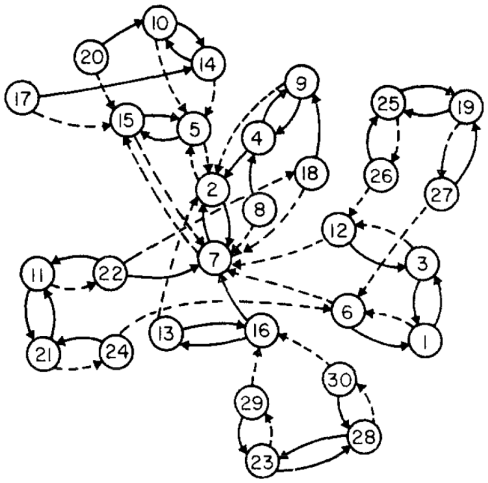


Figura 1.3: *Representación de un proceso de regulación genética vista como una red. En ella, los genes(numerados) se relacionan unos con otros y estas relaciones están representadas con las flechas que los unen.*

epigenético por lo que muchos investigadores, [1], [2], [5], [16], [17], [19], [32], lo han abordado con las herramientas de las que provee la Biología Sistémica y, en particular, con la imagen del paisaje epigenético.

Así, por ser de gran interés, en este trabajo nos proponemos desarrollar un método para para obtener el paisaje epigenético de un sistema mínimo biestable bajo un proceso de desarrollo.

Capítulo 2

Procesos estocásticos

Con las herramientas que nos provee la física clásica podemos estudiar gran parte de fenómenos naturales con suficiente precisión. En particular, para analizar los sistemas macroscópicos, la mecánica clásica, nos proporciona leyes simples pero que describen y predicen su comportamiento de forma suficientemente acertada.

Sin embargo, para hacer uso de ésta herramienta, es necesario analizar nuestro sistema de interés y enfocarse en las características que más influyan en su comportamiento y construir, con ellas, un modelo de pocas variables. Esto, porque no todas las variables que pudieran verse involucradas en un evento físico, influyen de forma significativa en su desarrollo. Es decir que, al analizar los sistemas físicos, se desatienden a las variables involucradas que no influyen significativamente en la dinámica del sistema para garantizar que el modelo sea lo más fácil de tratar posible.

Una vez hecho lo anterior, hay que plantear las ecuaciones diferenciales que describan la evolución temporal de las variables elegidas. Esto es a lo que llamamos un "sistema dinámico". Y un sistema que obedece estas ecuaciones bien definidas y cuyo comportamiento se puede predecir dadas ciertas condiciones iniciales, es, fundamentalmente, determinista.

No obstante, cuando tratamos con sistemas con demasiados grados de libertad, la influencia que tienen las variables menos predominantes sobre la dinámica del sistema puede ser significativa y, por lo tanto, la descripción que obtenemos de ellos, haciendo uso sólo de la mecánica clásica, puede diferir de los resultados experimentales. Así, conforme el sistema evoluciona, el modelo hecho solo con esta herramienta, se va alejando, cada vez más, de la realidad.

En estos casos, por lo tanto, es necesario tomar en cuenta las características del sistema que menos contribuyen en su dinámica y modelar como, en conjunto, lo modifican. Esto se logra añadiendo, en el sistema de ecuaciones diferenciales, una variable aleatoria de ruido que refleje la influencia de todas esas variables.

A los procesos que experimentan estos sistemas, descritos, ya no por ecuaciones diferenciales ordinarias, sino por ecuaciones diferenciales estocásticas, los llamamos, presisamente, *Procesos*

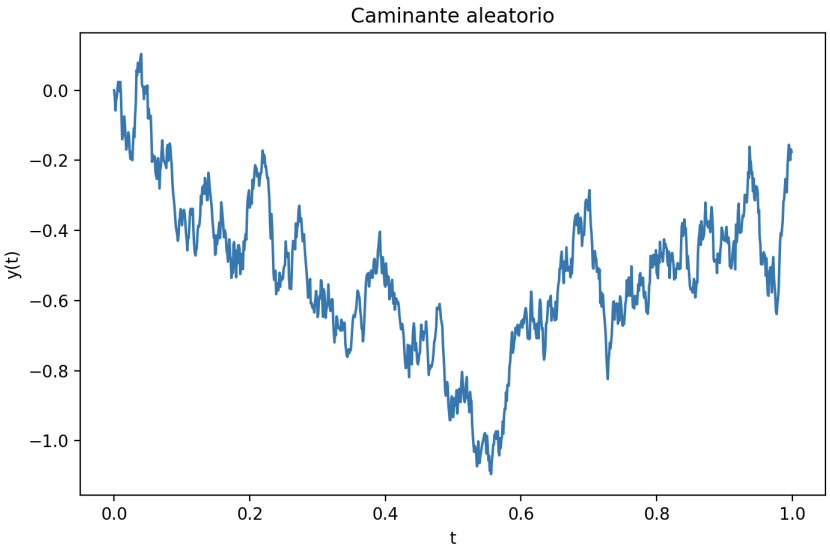


Figura 2.1: *Ejemplo de proceso estocástico*

estocásticos.

El estudio de estos procesos es de suma relevancia pues su uso se extiende a campos del conocimiento muy variados como, por ejemplo, las finanzas, la ecología, los microfluidos o las señales sismicas, biomédicas y de telecomunicacion. En particular, el uso en el que estamos interesados en este trabajo es el del estudio de redes genéticas que determinen el desarrollo celular.

2.1. Proceso de Markov

Se dice que un proceso de Markov es un proceso estocástico que "no tiene memoria", esto quiere decir que su estado futuro depende de su estado actual pero no de los pasados. Un proceso de Markov puede ser de tiempo continuo o discreto pero un proceso de Markov con espacio de estados $S \in \mathbb{Z}$, es decir, con espacio de estados discreto, es llamado una cadena de Markov.

Así pues, para definir un proceso de Markov, podemos codificar toda la información de sus estados pasados en un conjunto de σ-algebras.

Sea $(\Omega, \mathcal{F}, \mu)$ el espacio de probabilidad que modela un experimento aleatorio, con Ω el espacio muestral, $\mathcal{F}$ la colección indexada de todos los sucesos aleatorios y μ la función que asigna, a cada suceso, una probabilidad. Un proceso de Markov está definido[1] de la siguiente manera:

Definición **2.1** *Sea X_t un proceso estocástico definido sobre un espacio de probabilidad $(\Omega, \mathcal{F}, \mu)$ con valores en $\mathbb{R}^d$ y sea $\mathcal{F}_s^x$ la filtración generada por $\{X_t; t \in \mathbb{R}_+\}$. Entonces $\{X_t; t \in \mathbb{R}_+\}$ es un*

[1] En Pavliotis [24] podemos ver esta definición y sus detalles.

proceso de Markov si

$$\mathbb{P}(X_t \in \Gamma | \mathcal{F}_s^x) = \mathbb{P}(X_t \in \Gamma | X_s) \tag{2.1}$$

para todos $t, s \in T$ *con* $t \geq s$ *y* $\Gamma \in \mathcal{B}(\mathbb{R}^d)$.

Aquí $\mathbb{P}(A|B)$ es la probabilidad condicional y está definida como la probabilidad de que ocurra A dado B.

Es decir que, para un proceso de Markov, dado un valor x_{n-1} en t_{n-1}, $\mathbb{P}$ no se ve afectada por los valores en tiempos anteriores, o lo que es lo mismo, que para un estado futuro del sistema, solo contribuirá su estado actual pero no los estados pasados.
Así, la probabilidad condicional $\mathbb{P}$ tendrá la propiedad:

$$\mathbb{P}(x_n, t_n | x_1, \cdots, x_{n-1}; t_1, \cdots, t_{n-1}) = \mathbb{P}(x_n, t_n | x_{n-1}; t_{n-1}) \tag{2.2}$$

Por lo que, por ejemplo, para tres estados distintos x_1, x_2, x_3 en instantes de tiempo distintos $t_1 < t_2 < t_3$, respectivamente,

$$\begin{aligned} \mathbb{P}(x_1, x_2, x_3; t_1, t_2, t_3) &= \mathbb{P}(x_3, t_3 | x_1, x_2; t_1, t_2) \cdot \mathbb{P}(x_2, t_2 | x_1; t_1) \cdot p(x_1; t_1) \\ &= \mathbb{P}(x_3; t_3 | x_2; t_2) \cdot \mathbb{P}(x_2, t_2 | x_1; t_1) \cdot p(x_1; t_1) \end{aligned} \tag{2.3}$$

Así, una cadena de Markov consta de tres partes: un término de arrastre, una parte aleatoria y un "salto" en el espacio de estados. Por lo que un proceso de difusión es un tipo de proceso de Markov pues sus trayectorias son continuas y por lo tanto sin "saltos".
Para todo proceso de Markov de tiempo continuo podemos definir la *función de transición* como

$$P(\Gamma, t | x, s) := \mathbb{P}(X_{s+t} \in \Gamma | X_s = x) \tag{2.4}$$

De esta manera, para determinar un proceso de Markov de forma única, solo son necesarias su función de transición y su distribución inicial X_t.

2.2. Procesos de difusión

Un proceso de difusión es un proceso estocástico de tiempo continuo que satisface la ecuación diferencial estocástica

$$dX_t = b(X_t, t)dt + \xi(X_t, t)dW_t \tag{2.5}$$

pero cuyas funciones b y ξ no dependen explícitamente del tiempo.

Los procesos de difusión, vistos como procesos de Markov, pueden ser definidos dando sus dos primeros momentos y tomando en cuenta la condición de que no debe de haber saltos.
Se definen[24], entonces, a los procesos de difusión en una dimensión como sigue:

Definición **2.2** *Un proceso de Markov* X_t *en* $\mathbb{R}$ *con función de transición* $P(\Gamma, t|x, s)$ *es llamado un proceso de difusión si se satisfacen las siguientes condiciones:*

(i) Continuidad. Para cada x *y cada* $\epsilon > 0$.

$$\int_{|x-y|>\epsilon} P(dy, t|x, s) = o(t - s) \tag{2.6}$$

uniformemente sobre $s < t$.

(II) *Coeficiente de arrastre. Existe una función $b(x,s)$ tal que, para cada x y cada $\epsilon > 0$.*

$$\int_{|x-y|\leq\epsilon} (y-x)P(dy,t|x,s) = b(x,s)(t-s) + o(t-s) \tag{2.7}$$

uniformemente sobre $s < t$.

(III) *Coeficiente de difusión. Existe una función $\xi(x,s)$ tal que, para cada x y cada $\epsilon > 0$.*

$$\int_{|x-y|\leq\epsilon} (y-x)^2P(dy,t|x,s) = b(x,s)(t-s) + o(t-s) \tag{2.8}$$

uniformemente sobre $s < t$.

Cabe aclarar que esta definición se puede generalizar a procesos de más de una dimensión y, asumiendo que los dos primeros momentos existen, se pueden escribir la fórmulas para los coeficientes de arrastre y de difusión como:

$$\lim_{t\to s} \mathbb{E}\left(\frac{X_t - X_s}{t-s}|X_s = x\right) = b(x,s) \tag{2.9}$$

$$\lim_{t\to s} \mathbb{E}\left(\frac{|X_t - X_s|^2}{t-s}|X_s = x\right) = \xi(x,s) \tag{2.10}$$

respectivamente.

Los ejemplos clásicos de procesos de Markov son el movimiento Browniano y el proceso de Ornstein-Uhlenbeck estacionario.

Movimiento Browniano

En 1827 el médico y botánico Robert Brown observó en su microscopio el movimiento de unas partículas de polen suspendidas en agua, a este movimiento aleatorio es a lo que ahora llamamos *Movimiento Browniano*. Aunque Brown no explicó qué mecanismo era el responsable de este comportamiento sí lo documentó y, posteriormente, muchos trabajos se hicieron al respecto sin llegar a una conclusión útil. Fue hasta 1905 que Albert Einstein publicó un artículo[6] explicando el movimiento browniano basandose en dos hipótesis:

1. El movimiento de la partícula se debe a los impactos de de las moléculas de fluido contra las partículas de polen.
2. El efecto de el movimiento del fluido sobre el polen puede ser descrito de forma probabilistica en términos de colisiones estadísticamente independientes muy frecuentes.

Con ellas concluye que la ecuación diferencial de difusión

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} \tag{2.11}$$

tiene por solución

$$f(x,t) = \frac{n}{\sqrt{4\pi D}} \frac{\exp\left[-\frac{x^2}{4Dt}\right]}{\sqrt{t}} \tag{2.12}$$

con D el coeficiente de difusión que puede obtenerse de manera estadística. En particular, para una partícula browniana esférica de radio R, Einstein calculó

$$D = \frac{k_B T}{6\pi\nu R} \tag{2.13}$$

con k_B la constante de Boltzman, T la temperatura absoluta y ν la viscosidad de corte del fluido.

Con este resultado, Einstein establece una clara relación entre el movimiento de cuerpos microscópicos, como lo son las moléculas, y el fenómeno físico de la difusión.

Por otro lado, Langevin deduce la ecuación (2.13) de diferente manera a Einstein y es su trabajo el que inspirará el de Ornstein y Uhlenbeck. Él, considera el movimiento browniano como consecuencia de dos fuerzas, la fuerza de arrastre debida a la viscosidad del fluido f_v y una fuerza aleatoria f_a y, con esto, deduce que la ecuación de movimiento de una partícula browniana esta dada por

$$m\frac{d^2x}{dt^2} = -\gamma\frac{dx}{dt} + f_a(t) \tag{2.14}$$

con $-\gamma\frac{dx}{dt} = f_v$ y $\gamma = 6\pi\nu R$. Multiplicando por x:

$$mx\frac{d^2x}{dt^2} = -\gamma x\frac{dx}{dt} + xf_a(t) \tag{2.15}$$

Para el lado izquierdo de esta ecuación, vemos que

$$\begin{aligned} mx\frac{d^2x}{dt^2} &= mx\frac{d^2x}{dt^2} + mv^2 - mv^2 \\ &= m\left(v^2 + x\frac{d^2x}{dt^2}\right) - mv^2 \\ &= \frac{m}{2}\frac{d}{dt}\left(2x\frac{dx}{dt}\right) - mv^2 \\ &= \frac{m}{2}\frac{d}{dt}\left(\frac{dx^2}{dt}\right) - mv^2 \\ &= \frac{m}{2}\frac{d^2x^2}{dt^2} - mv^2 \end{aligned} \tag{2.16}$$

y, para el lado derecho, fácilmente podemos ver que $\frac{\gamma}{2}\frac{d}{dt}x^2 = \gamma x\frac{dx}{dt}$ por lo que la ecuación (2.14) se puede reescribir como

$$\frac{m}{2}\frac{d^2x^2}{dt^2} - mv^2 = -\frac{\gamma}{2}\frac{dx^2}{dt} + xf_a(t) \tag{2.17}$$

Además, puesto que la f_a es una función aleatoria, también la trayectoria $x(t)$ lo será. De esta manera Langevin "promedia" la ecuación para un número grande de partículas en movimientos brownianos similares y obtiene el resultado

$$\frac{m}{2}\frac{d^2}{dt^2}\langle x^2\rangle - \langle mv^2\rangle = -\frac{\gamma}{2}\frac{d}{dt}\langle x^2\rangle + \langle xf_a(t)\rangle \tag{2.18}$$

de donde Langevin hace $\langle xf_a(t)\rangle = 0$ "debido a su gran irregularidad", mientras que, la energía cinética promedio $\langle mv^2\rangle = k_B T$, con k_B la constante de Boltzmann y T la temperatura absoluta, por el teorema de la equipartición de la energía. Así

$$m\frac{d^2}{dt^2}\langle x^2\rangle + \gamma\frac{d}{dt}\langle x^2\rangle = 2k_BT \tag{2.19}$$

cuyo resultado, al integrar, es

$$\langle x^2\rangle - \langle x_0^2\rangle = 2\frac{k_bT}{\gamma}t = 2Dt \tag{2.20}$$

que está en concordancia con el resultado (2.13) que Einstein calculó.

Proceso de Ornstein-Uhlenbeck

El proceso de Ornstein-Uhlenbeck es un proceso estocástico de difusión que surge de agregar fricción al movimiento browninano.
Es llamado así en honor a Leonard Ornstein y a George Eugene Uhlenbeck, que en 1930 publican un artículo llamado "*On the Theory of the Brownian Motion*"[30] en el que presentan un modelo de la velocidad de la partículas "grandes" suspendidas en un fluido sobre las que actúa una fuerza de fricción. Es por ello que también se le llama *Caminante aleatorio amortiguado.*

En general, un proceso de Ornstein-Uhlenbeck está descrito por una ecuación diferencial estocástica de tipo

$$dY_t = -\alpha(Y_t - \mu)dt + \xi dW_t \tag{2.21}$$

de la cual podemos observar que, sin el término estocástico, la ecuación (2.21) describe la dinámica de un sistema con un punto estacionario en μ y si nos alejamos mucho de éste, el término $-\alpha(Y_t - \mu)dt$ obliga a Y_t a regresar a μ, por lo que a este término se le llama *de regresión a la media.*
Además, éstos coeficientes satisfacen la demostración de Îto [15], por lo que, para cada estado inicial $y_0 \in \mathbb{R}$, existe una única solución Y_t. Así, para el caso en el que $\mu = 0$, proponiendo

$$Y_t = A(t)\left(y_0 + \int_0^t B(s)dW_s\right) \tag{2.22}$$

como una solución y tomando $A(0) = 1$, es posible demostrar que

$$Y_t = \exp\{-\alpha t\}y_0 + \xi\exp\{-\alpha t\}\int_0^t \exp\{-\alpha s\}dW_s \tag{2.23}$$

con $Y_0 = y_0$. De esta ecuación podemos concluir que, puesto que la integral de una función no aleatoria sobre dW_t es una variable aleatoria Gaussiana con media cero y varianza $\int f(s)^2ds$, el proceso de Ornstein-Uhlenbeck tiene una distribución también Gaussiana. Además, si se consideran dos soluciones a (2.21), Y_t y Y_t', para distintas condiciones iniciales y_0 y y_0', la diferencia de éstas será

$$Y_t - Y_t' = \exp\{-\alpha t\}(y_0 - y_0') \tag{2.24}$$

que decrece exponencialmente a velocidad α por lo que a este parámetro se le llama *de relajamiento.*

2.3. Ecuación de Chapman-Kolmogorov

Sydney Chapman y Andréi Kolmogorov dedujeron, de forma independiente, una ecuación que ahora lleva el nombre de ambos y que describe la relación que hay entre las probabilidades de que

dos eventos aleatorios ocurran simultáneamente.
Partiendo de la expresión (2.3) que caracteriza a los procesos de Markov, se puede deducir la expresión llamada la *Ecuación de Chapman-Kolmogorov*

$$P(\Gamma, t|x, s) = \int_{\mathbb{R}^d} P(\Gamma, t|y, u)P(dy, u|x, s) \tag{2.25}$$

para toda $x \in \mathbb{R}^d$ y $s, u, t \in \mathbb{R}_+$. Así pues, esta ecuación nos permite analizar los procesos de Markov.
De esta expresión, y dando por hecho que se dan las condiciones (2.6)-(2.8) se puede obtener[2] una fórmula para el generador de un proceso de difusión que cumple con los siguientes teoremas

Teorema **2.1** *Sea $f(x) \in C_b(\mathbb{R})$ y sea*

$$u(x, s) := \mathbb{E}(f(X_t)|X_s = x) = \int f(y)P(dy, t|x, s) \tag{2.26}$$

con t fija y $P(dy, t|x, s)$ la función de transición. Asuma que las funciones $b(x, s)$ y $\xi(x, s)$ son suaves en x y en s. Entonces, $u(x, s)$ es solución del problema de valor final

$$-\frac{\partial u}{\partial s} = b_b(x, s)\frac{\partial u}{\partial x} + \frac{1}{2}\xi_b(x, s)\frac{\partial^2 u}{\partial x^2} \quad , \quad u(t, x) = f(x) \tag{2.27}$$

para $s \in [0, t]$

Teorema **2.2** *Asuma que las funciones $p(y, t|\cdot, \cdot)$ $b(y, t)$ y $\xi(y, t)$ son suaves en y y en t. Entonces, la densidad de probabilidad trancisional $p(x, s)$ es solución del problema de valor inicial*

$$\frac{\partial p}{\partial t} = -\frac{\partial}{\partial y}(b_f(t, y)p) + \frac{1}{2}\frac{\partial^2}{\partial y^2}(\xi_f(t, y)p) \quad , \quad p(s, y|x, s) = \delta(x - y) \tag{2.28}$$

Las expresiones (2.27) y (2.28) son llamadas *Ecuación "backward" de Kolmogorov* (EBK) y *Ecuación "forward" de Kolmogorov*, respectivamente.

La primera, aunque es la solución de un problema de valor final para una EDP parabólica, podemos reescribirla para un problema de valor inicial si hacemos el cambio de variable $T = t - s$ y tomamos b y ξ que no dependan explícitamente del tiempo. Además, si tomamos el tiempo inicial $s = 0$, tendremos

$$\frac{\partial u}{\partial t} = b_b(x)\frac{\partial u}{\partial x} + \frac{1}{2}\xi_b(x)\frac{\partial^2 u}{\partial x^2} \quad , \quad u(x, 0) = f(x) \tag{2.29}$$

con u el valor de expectación descrito en (2.26).

Notemos que el lado derecho de la igualdad (2.29) corresponde con un operador diferencial y, es éste, el que genera el proceso de difusión X_t.

Por otro lado la ecuación (2.28) es llamada la *Ecuación "Forward" de Kolmogorov*(EFK) y, si $\rho_0(x)$ es la distribución inicial del proceso X_t,

$$\frac{\partial p}{\partial t} = -\frac{\partial}{\partial y}(b_f(t, y)p) + \frac{1}{2}\frac{\partial^2}{\partial y^2}(\xi_f(t, y)p) \quad , \quad p(y, 0) = \rho_0(y) \tag{2.30}$$

[2]Estos teoremas y sus demostraciones se pueden encontrar en Pavliotis [24]

nos da la probabilidad de que el proceso de difusión X_t tome el valor y en un tiempo t. Esta ecuación es también llamada la *Ecuación de Fokker-Planck* en honor a Adriaan Fokker y Max Planck pues, aunque tiene la mima forma que la EFK, la deducción de Fokker y Planck está basada en suposiciones físicas a diferencia de la de Kolmogorov que es una deducción puramente matemática[27].

Ambas ecuaciones, la EBK y la EFK, se pueden generalizar para el caso de más dimensiones. En este caso, el coeficiente de arrastre $b(x,t)$ será un campo vectorial de dimensión d y el coeficiente de difusión $\xi(x,t)$ será una matriz $d \times d$ simétrica y no-negativa. Así, si b y ξ no dependen explicitamente del tiempo, el proceso de difusión estará descrito por la EBK

$$\frac{\partial u}{\partial t} = \sum_{j=1}^{d} b_j(x)\frac{\partial u}{\partial x_j} + \frac{1}{2}\sum_{i,j=1}^{d} \xi_{ij}(x)\frac{\partial^2 u}{\partial x_i \partial x_j}, \quad u(x,0) = f(x) \tag{2.31}$$

o bien, por la EFK

$$\frac{\partial p}{\partial t} = \sum_{j=1}^{d} \frac{\partial}{\partial y_j}\left(-b_j(y)p + \frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial y_i}\xi_{ij}(y)p\right), \quad p(y,0) = \rho_0(y) \tag{2.32}$$

2.4. Fórmula de Feynman-Kac

La llamada fórmula de Feynman-Kac fue primero propuesta, por Richard Feynman en el área de Mecánica Cuántica al darse cuenta de que, al promediar las trayectorias de la dinámica de un sistema cuántico, podía resolver la ecuación de Schrödinger. Al mismo tiempo, Mark Kac, un matemático que trabajaba en la teoría de la probabilidad, se dió cuenta que Feynman y él estaban trabajando una representación similar pero, mientras que Feynman la estaba aplicando a la teoría cuántica, Kac la utilizaba para solucionar la ecuación de calor.

En el caso de los procesos de difusión, esta ecuación establece una relación entre el valor de expectación de algunas funcionales del movimiento browniano y la ecuación de calor con un término disipativo representado con una EDP lineal de segundo orden.

En general, para un proceso de difusión cuyo generador sea $\mathscr{L}$, esta fórmula nos da una solución a la ecuación que lo describe

$$\frac{\partial u}{\partial t} = \mathscr{L}u - Vu \quad , \quad u(0,x) = f(x) \tag{2.33}$$

con V y f acotadas. Esta solución será[3]:

$$u(x,t) = \mathbb{E}\left(e^{-\int_0^t V(x)ds} f(x)\right) \tag{2.34}$$

Esta fórmula resulta muy útil pues puede emplearse para encontrar, de forma probablilistica la solución a EDPs de tipo parabólico como lo es la EBK, y frecuentemente se resuelve con métodos numéricos, en particular con Monte Carlo.

[3]La demostración de esta fórmula se encuentra en Pavliotis [24] y, distintas variantes y aplicaciones pueden consultarse en Gobet [9] y en Mörters and Peres [22]

Relación entre la ecuación de Langevin, la EBK y la fórmula de Feynman-Kac

En el caso en el que el coeficiente del movimiento browniano sea una constante se puede establecer una relación entre la ecuación de Langevin (2.14) y la ecuación de Fokker-Planck (o EFK) si el coeficiente de arrastre γ es lineal, haciendo $y = \frac{dx}{dt}$ y siempre que la fuerza aleatoria f_a satisfaga las propiedades

$$\begin{aligned} \langle f_a(t) \rangle &= 0 \\ \langle f_a(t) f_a(t') \rangle &= \Gamma \delta(t - t') \end{aligned} \tag{2.35}$$

así, la ecuación (2.14) tendrá la forma

$$m\frac{dy}{dt} = -\gamma y + f_a(t) \tag{2.36}$$

haciendo $m = 1$, integrando y haciendo $y(0) = y_0$

$$y(t) = y_0 e^{-\gamma t} + e^{-\gamma t} \int_0^t e^{\gamma s} f_a(s) ds \tag{2.37}$$

y, eleveando esta ecuación al cuadrado

$$y(t)^2 = y_0^2 e^{-2\gamma t} + e^{-2\gamma t} \int_0^t e^{2\gamma s} f_a^2(s) ds + 2y_0 e^{-2\gamma s} \int_0^t e^{\gamma s} f_a(s) ds \tag{2.38}$$

Promediando las ecuaciones (2.37) y (2.38) y aplicando las condiciones (2.35) tenemos

$$\langle y(t) \rangle = y_0 e^{-\gamma t} \tag{2.39}$$

$$\begin{aligned} \langle y(t)^2 \rangle &= y_0^2 e^{-2\gamma t} + e^{-2\gamma t} \int_0^t e^{2\gamma s} f_a^2(s) ds \\ &= y_0^2 e^{-2\gamma t} + \frac{\Gamma}{2\gamma} e^{-2\gamma t} \left(e^{2\gamma t} - 1 \right) \\ &= y_0^2 e^{-2\gamma t} + \frac{\Gamma}{2\gamma} \left(1 - e^{-2\gamma t} \right) \end{aligned} \tag{2.40}$$

Sabemos que, conforme $y \to \infty$, el sistema alcanzará el equilibrio por lo que por el teorema de la equipartición de la energía:

$$\frac{1}{2} m \langle y^2(\infty) \rangle = \frac{1}{2} k_B T \tag{2.41}$$

mientras que, por la expresión (2.40), cuando $y \to \infty$

$$\langle y^2(\infty) \rangle = \frac{2\Gamma}{\gamma} \tag{2.42}$$

así que, comparando estas dos últimas ecuaciones, vemos que

$$\Gamma = -\frac{2\gamma}{m} k_B T. \tag{2.43}$$

Por otro lado, si en (2.39) tomamos como t un Δt, podemos usar la expansión en serie de la función exponencial para ver que

$$\begin{aligned}\langle y(\Delta t)\rangle &= y_0(1-\gamma\Delta t+\frac{(\gamma\Delta t)^2}{2}+\cdots)\\ &= y_0-y_0\gamma\Delta t+\mathscr{O}^2(\Delta t)\end{aligned} \tag{2.44}$$

así, si $\langle\Delta y\rangle=\langle y(t)\rangle-y_0$:

$$\langle y(\Delta t)\rangle=-y_0\gamma\Delta t+\mathscr{O}^2(\Delta t) \tag{2.45}$$

por lo que el coeficiente de arrastre de la ecuación de Fokker-Planck estará dado por[4].

$$a_1(y_0)=\lim_{\Delta t\to 0}\frac{\langle\Delta y\rangle}{\Delta t}=-\gamma y_0 \tag{2.46}$$

Análogamente, partiendo de la ecuación (2.40) podemos obtener el coeficiente $a_2=\Gamma$. Con la cual, en sus dos primeros momentos, la EFK

$$\frac{\partial p}{\partial t}=-\frac{\partial}{\partial y}\left[(\gamma y)p\right]+\frac{1}{2}\frac{\partial^2}{\partial y^2}(\Gamma p) \tag{2.47}$$

será análoga a la ecuación de Langevin.

Si, por otro lado, el coeficiente de arrastre es una función no lineal $A(y)$, la ecuación de Langevin será equivalente a la ecuación:

$$\frac{\partial p}{\partial t}=-\frac{\partial}{\partial y}\left[(A(y)p\right]+\frac{1}{2}\frac{\partial^2}{\partial y^2}(\Gamma p) \tag{2.48}$$

A su vez, está ecuación la podemos relacionar con la fórmula de Feynman-Kac (2.34) pues sabemos que ésta nos da la solución a una ecuación diferencial parcial del tipo (2.33) y, en el caso de una ecuación de Fokker-Planck como (2.48), el generador del proceso de difusión será

$$\mathscr{L}=\frac{\partial}{\partial y}\left(-A(y)+\frac{1}{2}\frac{\partial}{\partial y}\Gamma\right). \tag{2.49}$$

Sin embargo, en el caso en el que queramos resolver una EBK del tipo (2.29), es decir en la que el coeficiente de difusión no depende explícitamente del tiempo y que representa un problema de valor inicial, tenemos, primero, que establecer una relación entre nuestra ecuación y una de Fokker-Plank, para que de esta forma, se pueda resolver por medio de Feynman-Kac.
Para ello, partimos de la ecuación (2.29) y la expresamos de la forma

$$\frac{\partial u}{\partial t}=\mathscr{L}_b u \tag{2.50}$$

con $\mathscr{L}_b=b_b(x)\frac{\partial}{\partial x}+\frac{1}{2}\frac{\partial^2}{\partial x^2}$.
Tomamos, por otro lado, el primer término del lado derecho de la ecuación (2.28) y hacemos

$$-\frac{\partial}{\partial y}(b_f(t,y)p)=-\frac{\partial b_f(t,y)}{\partial y}p-\frac{\partial p}{\partial y}b_f(y) \tag{2.51}$$

[4]Para ver los detalles de cómo se justifica esta aseveración ver Scott [27]

y , si tomamos $\xi_f(t,y)$ como una constante, la ecuación (2.28) quedará de la forma

$$\frac{\partial p}{\partial t} = -\frac{\partial b_f(t,y)}{\partial y}p - \frac{\partial p}{\partial y}b_f(y) + \frac{1}{2}\xi_f\frac{\partial^2}{\partial y^2}p \tag{2.52}$$

Reacomodando:

$$\frac{\partial p}{\partial t} = -b_f(y)\frac{\partial p}{\partial y} + \frac{1}{2}\xi_f\frac{\partial^2 p}{\partial y^2} - p\frac{\partial b_f(t,y)}{\partial y} \tag{2.53}$$

que podemos escribir de la forma (2.33) con

$$\mathscr{L}_f = -b_f(y)\frac{\partial}{\partial y} + \frac{1}{2}\xi_f\frac{\partial^2}{\partial x^2} \tag{2.54}$$

$$V_f = \frac{\partial b_f(t,y)}{\partial y} \tag{2.55}$$

Si además renombramos a las variable de tal forma que $y \to x$, $p \to u$ y $\xi_f = 1 = \xi_b$, podemos ver que si $\mathscr{L}_f = \mathscr{L}_b$ entonces $b_b = -b_f$. De esta forma, tendremos una EFK expresada de tal manera que podemos resolverla con la Fórmula de Feynman-Kac (2.34), tomando $V = -\frac{\partial b_b(t,y)}{\partial x}$ y $b_b = -b_f$.

Este resultado se puede extender al caso multidimensional si al generador del proceso de difusión lo expresamos como

$$\mathscr{L}_b = \sum_{j=1}^{d} b_j(x,t)\frac{\partial}{\partial x_j} + \frac{1}{2}\sum_{i,j=1}^{d}\xi_{i,j}(x,t)\frac{\partial^2}{\partial x_i \partial x_j} \tag{2.56}$$

para la EBK de dimension d, o bien

$$\mathscr{L}_f = \sum_{j=1}^{d}\frac{\partial}{\partial x_j}\left(b_j(x,t) + \frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial x_i}\xi_{i,j}(x,t)\right) \tag{2.57}$$

para la EFK de dimension d.
Por ejemplo, para el caso bidemensional (d=2), siendo ξ la matríz identidad, y partiendo de la ecuación (2.32) un proceso de difusión estará descrito por

$$\begin{aligned}
\frac{\partial p}{\partial t} &= \frac{\partial}{\partial y_1}\left[-b_1(y)p + \frac{1}{2}\sum_{i=1}^{d}\frac{\partial p}{\partial x_i}\right] + \frac{\partial}{\partial y_2}\left[-b_2(y)p + \frac{1}{2}\sum_{i=1}^{d}\frac{\partial p}{\partial x_i}\right] \\
&= -p\frac{\partial}{\partial y_1}b_1(y) - b_1(y)\frac{\partial p}{\partial y_1} - p\frac{\partial}{\partial y_2}b_2(y) - b_2(y)\frac{\partial p}{\partial y_2} + \frac{1}{2}\left[\frac{\partial}{\partial y_1}\sum_{i=1}^{2}\frac{\partial p}{\partial y_i} + \frac{\partial}{\partial y_2}\sum_{i=1}^{2}\frac{\partial p}{\partial y_i}\right] \\
&= -\sum_{j=1}^{2} b_j(y)\frac{\partial p}{\partial y_j} + \frac{1}{2}\sum_{i,j=1}^{2}\frac{\partial^2 p}{\partial y_i \partial y_j} - p\sum_{j=1}^{2}\frac{\partial}{\partial y_j}b_j(y)
\end{aligned} \tag{2.58}$$

de donde podemos ver que los dos primeros términos tienen la forma de la EBK (2.31) con $d = 2$ con un término extra y con $b(y) = -b(x)$. De esta forma nuestra EFK de dos dimensiones toma la forma de (2.33) con $V = \sum_{j=1}^{2}\frac{\partial b_j(y)}{\partial y_j}$ y $\mathscr{L} = \mathscr{L}_b$ de (2.56).

Capítulo 3

Método Monte Carlo

Como hemos dicho, la física clásica nos ofrece herramientas útiles para modelar, comprender y predecir los fenómenos naturales. Sin embargo, hay sistemas que, por su complejidad, no son suficientemente bien descritas con esta herramienta. Al plantear las ecuaciones que describen su evolución en el tiempo, y al ser tantas las variables que pudieran influir en su dinámica, siempre es necesario hacer simplificaciones y, aún así, muchas veces resultan expresiones bastante complejas que no pueden ser resueltas analíticamente.

En particular, en el caso de las ecuaciones que describen a los procesos estocásticos, se han hecho numerosos intentos de encontrar soluciones analíticas a problemas particulares, sin embargo, para conseguir resultados, se ha tenido que recurrir a numerosas aproximaciones y, por lo tanto, las soluciones encontradas son restrictivas. Es por esto que se ha recurrido a la utilización de métodos numéricos para encontrar soluciones a las ecuaciones *backward* y *forward* de Kolmogorov para casos más generales y sistemas más complejos[1].

La intención de este trabajo es, justamente, hacer uso del método Monte Carlo para encontrar una solución numérica a una EFK bidimensional no lineal que describa a un sistema mínimo biestable, por lo que, a continuación, daremos un breve repaso de este método.

3.1. Una pequeña descripción

El método Monte Carlo se desarrolló en la década de los 40 del siglo pasado, recibe su nombre en honor al Casino Montecarlo pues es, básicamente, un generador de datos aleatorios. En Reiter [25] se define como "el uso deliberado de números aleatorios en un cálculo que tiene la estructura de un proceso estocástico".

Justamente por ser estocástico se suele utilizar como complemento a modelos deterministas, como el de Dinámica Molecular, para hacerlos más precisos aunque su uso se extiende a cualquier

[1] En Spencer and Bergman [28] se puede encontrar un resumen de los resultados que se han obtenido al intentar solucionar las ecuaciones de Kolmogorov para distintos sistemas modelo.

sistema que presente cierta aleatoriedad como la Estadística, los Sistemas de Modelos Atómicos, las Finanzas o la Ecología, e incluso, puede usarse para resolver un problema determinista si se plantea en términos de un sistema estocástico inventado pero cuyo valor esperado coincida con la solución del problema determinista.

Aunque hay evidencias del uso de este método o, al menos de las técnicas en las que se basa, en el problema de Buffon que data de 1733 y, posteriormente, en un trabajo de Lord Kelvin en 1901[29] no es hasta la segunda guerra mundial que Neumann, Ulam y Fermi, lo caracterizan y lo dan a conocer[2].

Este método es especialmente útil cuando tratamos con sistemas de muchas dimensiones pues, con otro tipo de métodos, al discretizar el dominio de las funciones que tratamos, se eleva mucho el coste computacional; mientras que con el método Monte Carlo, al tomar valores aleatorios o pseudo-aleatorios el tiempo de cómputo se reduce y los recursos se optimizan. Sin embargo, hay que tomar en cuenta que cada número aleatorio que se toma añade cierto error a los cálculos, con lo que se podría perder precisión. Aún así, puesto que existen varios métodos para minimizar estos errores[3], el método en sí ha demostrado ser muy útil y confiable.

3.2. Comparación con otros métodos

Supongamos que queremos encontrar:

$$I = \int_a^b g(x)dx \tag{3.1}$$

con $g(x)$ conocida. Está claro que para encontrar I necesitamos integrar a $g(x)$, pero supongamos que esa integral no tiene solución analítica conocida. Recurrimos entonces a los métodos numéricos para encontrar una solución a nuestro problema.

La mayoría de los métodos numéricos de integración, parten de disctretizar el segmento del dominio $[a, b]$ y dividirlo en segmentos más pequeños de ancho h para luego encontrar $g'(x)$ en cada uno de los segmentos pequeños e inferir alguna relación entre ellos y el valor real de la función(ver figura3.1). Con éstos métodos entre más pequeño sea h más cerca nos encontraremos del valor real de I.

Aunque los métodos más comunes de integración (en especial con las mejoras que se les han hecho a lo largo de los años) nos dan resultados suficientemente buenos a funciones de dimensión uno o dos, al incrementarse la dimensión del problema, el trabajo de discretizar y hacer los cálculos en cada uno de los segmentos se vuelve muy pesado.

En estos casos, en los que la dimensión del sistema aumenta, es mucho más eficiente computacionalmente integrar con el método Monte Carlo.

[2]En Hammersley and Handscomb [12] hay una breve história y una explicación amplia y completa del método.
[3]Ver [12]

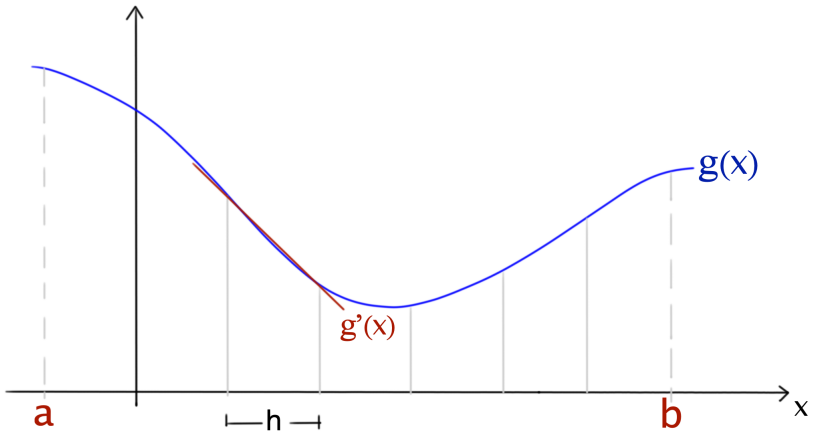


Figura 3.1: *Representación de un método de integración numérico*

En términos muy simples, una integración por Monte Carlo se basa en promediar los resultados de trabajar con n valores aleatorios y así obtener el resultado que nos interesa. En nuestro ejemplo anterior, para obtener I se toman n valores aleatorios de x entre a y b, para cada uno de los valores x_i, $i = 1, 2, ..., n$, se obtiene $g_i = g(x_i)$ y se calcula el área $A_i = g_i * (b - a)$ (ver figura3.2). Con lo que

$$I \approx \frac{1}{n} \sum_{i=1}^{n} A_i \tag{3.2}$$

y resulta claro que, conforme $n \to \infty$ nuestro cálculo de I será más exacto.

Un aspecto importante del método, que hay que tratar con cuidado, es la elección de los números aleatorios[4]. Aunque los llamemos "números aleatorios", la forma de generarlos casi siempre obedece a algún método particular descrito previamente. Sin embargo, no son importantes estos métodos sino cómo estarán distribuidos los números obtenidos. Cuando, de un conjunto de números tomamos elementos que tengan, cada uno, la misma probabilidad de ser elegidos, llamamos a éstos números *aleatorios* mientras que, si de estos elementos, algunos tienen una probabilidad más alta de ser elegidos, se les llamará *pseudoaleatorios*. Estos números son los que más se suelen utilizar cuando se resuelven problemas físicos pues la distribución de probabilidad que rige a estos números es, generalmente, representativa del sistema a resolver.

Como ya dijimos, y como sucede con cualquier otro método númerico, el método Monte Carlo no está libre de errores, no obstante, se han creado variaciones o refinamientos del método para

[4]En Reiter [25] se tratan distintas técnicas de generación de números aleatorios a partir de diferentes densidades de probabilidad.

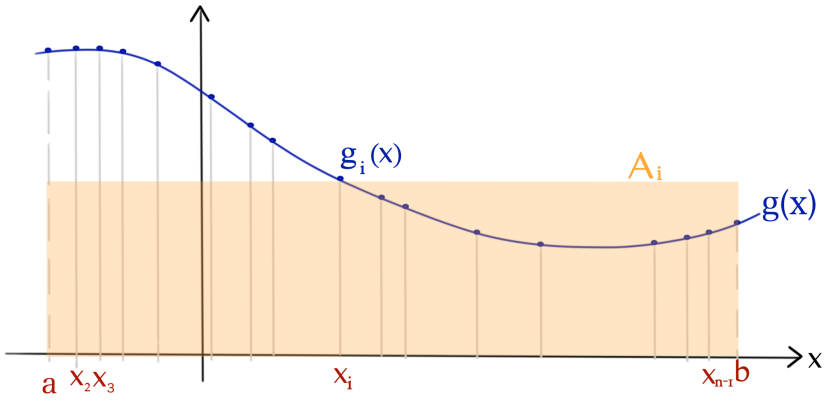


Figura 3.2: *Representación del método de integración Monte Carlo*

minimizar esos errores. Por ejemplo, tomando el ejemplo estándar:

$$I = \int_0^1 f(x)dx \tag{3.3}$$

Podemos tomar $x_1, ..., x_n$ números aleatorios distribuidos uniformemente con lo que cada g_i serán variables aleatorias independientes con valor de expectación I y varianza

$$\frac{1}{2}\int_0^1 (f(x) - I)^2 dx \tag{3.4}$$

y

$$\frac{1}{n}\sum_{i=1}^{n} f_i \tag{3.5}$$

nos dará una estimación de I. A este procedimiento se le llama *crude* Monte Carlo.

Por otro lado, la técnica llamada *muestreo estratificado*, consiste en dividir el rango de integración en varios segmentos y aplicar, en cada uno, el *crude* Monte Carlo. [5].

En términos de un sistema físico, si lo que queremos medir es una propiedad determinada $A = A(x)$, y tenemos la función de distribución de una partícula (distribución de microestado) p(x), el valor promedio de A, estará dado por

$$\langle A \rangle = \int A(x)p(x)dx \tag{3.6}$$

el tomar números aleatorios igualmente probables, equivaldría a tener $p = cte$. mientras que tomar otras distribuciones incluiría un elemento estocástico al cálculo de A. Esto no representa ningún problema pues, aunque un método que discretice al dominio de forma sistemática, converge más rápido a la solución; para problemas con dimensionalidades altas, el cálculo converge más rápido entre más representativa sea la distribución $p(x)$ que tomemos.

[5] En Hammersley and Handscomb [12] se puede consultar éste método de forma más detallada y otros para minimizar errores.

3.3. Monte Carlo aplicado a las cadenas de Markov

Si x_t es el estado en que se encuentra el sistema en el instante t y f es una función del espacio de estados que define a la variable aleatoria A_t

$$A_t = f(x_t) \tag{3.7}$$

el valor que queremos encontrar será

$$\langle A_t \rangle = \frac{1}{n} \sum_{s=1}^{t} A_s \tag{3.8}$$

y representa el promedio de f en todos los instantes de tiempo en los que se desarrolla el sistema.

El método se divide en los siguientes pasos:

- En un instante de tiempo i, el sistema se encuentra en un estado x_i.
- La configuración del sistema se perturba ligeramente de forma aleatoria con lo que se consigue un estado propuesto x_i'
- De acuerdo a la distribución de probabilidad $p(x)$ se evalúa si el estado propuesto x_i' se acepta o se rechaza[6].
- Si el estado x_i' es aceptado $x_{i+1} = x_i'$, por el contrario, si se rechaza, $x_{i+1} = x_i$
- Se repiten todos los pasos anteriores.

Con estos pasos se consigue crear una trayectoria de A en función de los estados aceptados con la que podemos calcular (3.8) siendo n el número total de iteraciones.

En este trabajo aplicaremos este método para hallar la solución a una EFK que represente un problema de valor inicial por medio de una discretización de la ecuación de Langevin equivalente. En particular, aquella que describa la evolución temporal de un sistema mínimo biestable. Posteriormente se utilizarán los resultados obtenidos para entrenar una red neuronal artificial.

[6]Más información al respecto puede encontrarse en Hammersley and Handscomb [12] y en]

Capítulo 4

Sistema dinámico biestable

En el estudio de los sistemas dinámicos un ejemplo común que se suele analizar es el de un sistema biestable, es decir, un sistema con dos estados estacionarios estables separados por un estado estacionario inestable.
Matemáticamente, en el caso unidimensional, para representar estos sistemas, necesitamos construir una ecuación diferencial con tres ceros, que corresponden a los puntos estacionarios del sistema. La ecuación más simple que nos da estos puntos es un polinomio de grado tres y para que los puntos estacionarios sean diferentes entre sí y sean no-negativos, el término cuadrático debe ser positivo y el lineal negativo, el término independiente debe de ser cero y el cúbico es negativo

$$\dot{x} = -ax^3 + bx^2 - cx \quad ; \qquad a, b, c > 0 \tag{4.1}$$

La biestabilidad ha demostrado ser de gran utilidad al modelar sistemas en distintas áreas del conocimiento. En electrónica, un *flip-flop* es un multivibrador que puede permanecer en uno de dos estados posibles; en óptica, la biestabilidad se puede observar en sistemas ópticos cuya intensidad de salida pueda adoptar uno de dos valores posibles y, en la biología, un *genetic toggle switch* es un sistema biológico que puede tener dos valores que representan los estados "encendido" y "apagado" de cierta expresión genética.

Justamente en esta área, la biestabilidad resulta una herramienta poderosa pues con ella se pueden modelar múltiples sistemas y procesos biológicos tales como la apoptosis[7] y la diferenciacion celular[35]; así como algunas enfermedades de priones[18] y diferentes fenómenos en bacterias[31].

En Ferrell [8] se hace un análisis de múltiples trabajos en los que diferentes procesos de señalización celular son modelados haciendo uso de sistemas biestables. El autor concluye que una condición necesaria para la biestabilidad es que los sistemas tengan ciclos de retroalimentación no lineal. Por otro lado, dice el autor, "estos circuitos exhiben cierto grado de histéresis" pero un pueden desatar respuestas irreversibles si la retroalimentación es demasiado grande.

Así, dice Ferrell [8], este tipo de sistemas resultan suficientemente simples para comprenderlos a nivel intuitivo pero son suficientemente sofisticados para producir comportamientos interesantes.

4.1. Sistema mínimo biestable

Tomando en cuenta las consideraciones de Ferrell [8], y considerando que un sistema mínimo debe tener:

- El mínimo número de reactivos
- El mínimo número de reacciones
- El mínimo número de términos en las EDO's

en Wilhem [34] se establece que el Sistema Mínimo Biestable de Reacción Química(SMB) es el descrito por las siguientes reacciones químicas

$$\begin{aligned} \mathrm{S} + \mathrm{Y} &\xrightarrow{k1} 2\,\mathrm{X} \\ 2\,\mathrm{X} &\xrightarrow{k2} \mathrm{X} + \mathrm{Y} \\ \mathrm{X} + \mathrm{Y} &\xrightarrow{k3} \mathrm{Y} + \mathrm{P} \\ \mathrm{X} &\xrightarrow{k4} \mathrm{P} \end{aligned}$$

que es único y en donde S y P denotan substratos y productos constantes. Además, si se asumen condiciones espaciales homogéneas, el sistema puede ser descrito por medio del sistema de EDOs siguiente:

$$\dot{x} = 2k_1 y - k_2 x^2 - k_3 yx - k_4 x \tag{4.2}$$

$$\dot{y} = -k_1 y + k_2 x^2 \tag{4.3}$$

que nos da la evolución en el tiempo de la concentración de los reactivos X y Y.

Podemos tomar $k_1 = 0$ sin perdida de generalidad y $k_2, k_3, k_4 > 0$. Así, igualando las ecuaciones (4.2) y (4.3) a cero podemos encontrar que los estados estacionarios del sistema son:

$$x_1 = y_1 = 0 \tag{4.4}$$

$$x_2 = \frac{k_1 - \sqrt{k_1 D}}{2k_3} \quad , \quad y_2 = \frac{(x_2)^2}{k_1} \tag{4.5}$$

$$x_3 = \frac{k_1 + \sqrt{k_1 D}}{2k_3} \quad , \quad y_3 = \frac{(x_3)^2}{k_1} \tag{4.6}$$

con $D = k_1 - 4k_3k_4$. Por lo que los tres estados estacionarios serán reales si $D > 0$ y existe una bifurcación de nodo de silla de montar en $D = 0$.

Por otro lado, analizando el Jacobiano de este sistema

$$J = \begin{pmatrix} -k_4 - 2x - k_3 y & 2k_1 - k_3 x \\ 2x & -k_1 \end{pmatrix} \tag{4.7}$$

podemos ver que puesto que $tr(J) < 0$ el sistema es disipativo y que se restringe a la parte positiva del espacio fase. Además, para cada uno de los puntos estacionarios podemos ver que los

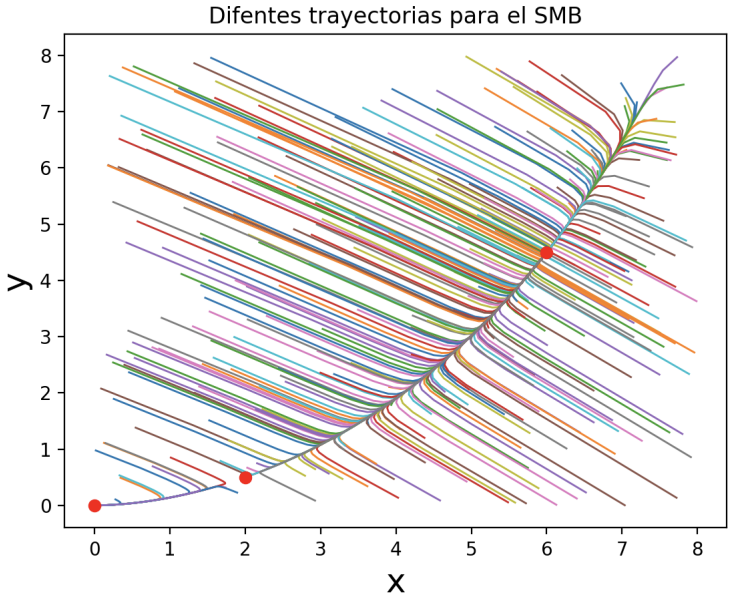


Figura 4.1: *Trayectorias para diferentes puntos iniciales en el espacio de estados del SMB*

determinantes de la matríz Jacobiana son

$$|J|_1 = k_1 k_4 > 0 \tag{4.8}$$

$$|J|_2 = \frac{k_1}{2k_3(D - \sqrt{k_1 D})} < 0 \tag{4.9}$$

$$|J|_3 = \frac{k_1}{2k_3(D + \sqrt{k_1 D})} > 0 \tag{4.10}$$

por lo que el primer y el tercer punto estacionario son localmente estables y el segundo es un punto silla, es decir, es localmente inestable[1]. En la figura 4.1 podemos ver las trayectorias para diferentes putos iniciales en el espacio de estados del SMB. Los puntos marcados en rojo son los puntos estacionarios para $k_1 = 8$, $k_2 = 1$,$k_3 = 1$ y $k_4 = 1.5$.

4.2. Solución a la Ecuación de Fokker-Planck del SMB por Método Monte Carlo

Como ya se ha mencionado, los sistemas biológicos son sistemas complejos cuya dinámica depende de muchas variables, algunas de ellas no controlables. Además, por no poder estar aislados, los alrededores del sistema también influyen en su comportamiento.

Así, aunque la dinámica de nuestro sistema de interés está descrita por las ecuaciones diferenciales (4.2),(4.3), también hay que tomar en cuenta la influencia de las variables no controlables

[1]En Wilhem [34] se dan más detalles sobre el sistema y los cálculos utilizados en su análisis

en el comportamiento del sistema. Esto lo logramos por medio de la ecuación de Langevin

$$dy = A(y,t)dt + \eta(t)dt \tag{4.11}$$

que es equivalente a una Ecuación de Fokker-Planck(EFP) o EFK (2.28) con $A(y) = b_f$ y con η un ruido blanco gaussiano. En nuestro caso, introduciremos la parte determinista de la dinámica del SMB en el coeficiente $A(y) = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}$.

Además sabemos, por la subseccion 2.4, que podemos relacionar una EFP multidimensional con una EBK multidimensional si $\xi_b(y) = \xi_f(x)$ son la matríz identidad y representando la EFP como

$$\frac{\partial u}{\partial t} = \mathscr{L}u - Vu \tag{4.12}$$

con $\mathscr{L} = \mathscr{L}_b$ de (2.56) y $V = \sum_{j=1}^{2} \frac{\partial b_j}{\partial y_j}$, haciendo $\vec{b_b} = -\vec{b}_f$.

Así, para el caso bidimensional, si $\vec{A} = \vec{b}_f = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}$ y $\vec{A'} = \vec{b_b} = \begin{pmatrix} A_1' \\ A_2' \end{pmatrix}$:

$$A_1' = -(2k_1 y - k_2 x^2 - k_3 yx - k_4 x) \tag{4.13}$$

$$A_2' = -(-k_1 y + k_2 x^2) \tag{4.14}$$

$$V = -2k_2 x - k_3 y - k_4 - k_1 \tag{4.15}$$

Queremos resolver por medio de Monte Carlo la EFP asociada a este sistema y, para ello haremos uso de la Fórmula de Feynman-Kac(FFK).
Comenzaremos, entonces, por discretizar la ecuación de Langevin haciendo uso del diagrama de Scott [27] de la figura 4.2 y la resolveremos numéricamente por medio del algoritmo de la sección A.1 tomando $k_1 = 8$, $k_3 = 1$ y $k_4 = 1.5$[2]. Esto nos permite obtener la evolución de ambas variables de estado después de un tiempo t.

Al ejecutar $\mathbf{N}$ veces el algoritmo obtenemos $\mathbf{N}$ valores para $\mathbf{y}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$ y, estos valores podemos introducirlos en la FFK (2.34) para obtener la solución a la EFP en términos del valor esperado

$$u(\mathbf{y},t) = \mathbb{E}\left(e^{-\int_0^t V(\mathbf{y})ds} f(\mathbf{y}) \right) \tag{4.16}$$

con $f(\mathbf{y})$ la función de condición inicial que tomaremos como una distribución normal bidemensional (fig.4.3)

$$f(x,y) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)} \left[\left(\frac{x-\mu_1}{\sigma_1} \right)^2 + \left(\frac{y-\mu_2}{\sigma_2} \right)^2 - 2\rho \left(\frac{x-\mu_1}{\sigma_1} \right) \left(\frac{y-\mu_2}{\sigma_2} \right)^2 \right] \right) \tag{4.17}$$

y

$$-\int_0^t V(\mathbf{y}) = -tV(\mathbf{y}) \tag{4.18}$$

por lo que introduciremos los $\mathbf{N}$ valores de $\mathbf{y}$ obtenidos con la discretización de Langevin en

[2]Que son los valores que propone Wilhem [34].

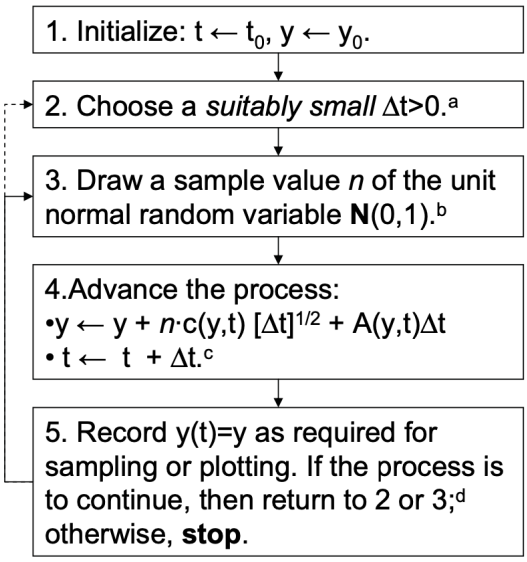


Figura 4.2: *Diagrama del algoritmo para discretizar la ecuación de Langevin*

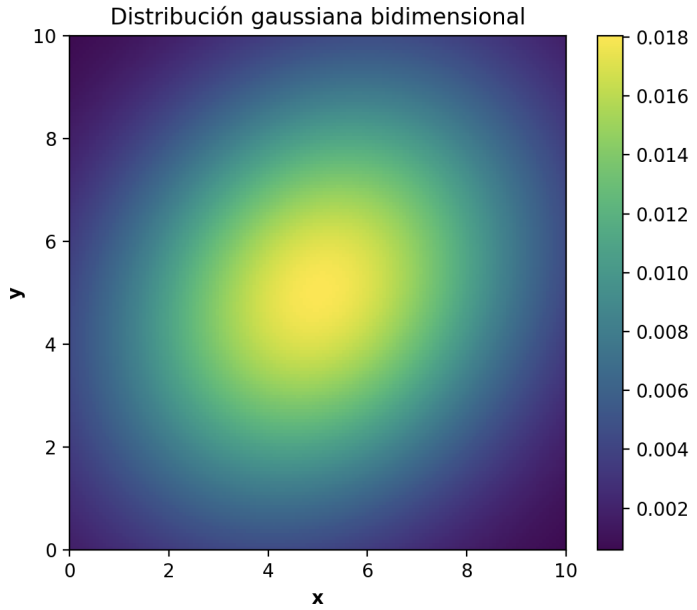


Figura 4.3: *Distribución normal bidimensional como función de condición inicial*

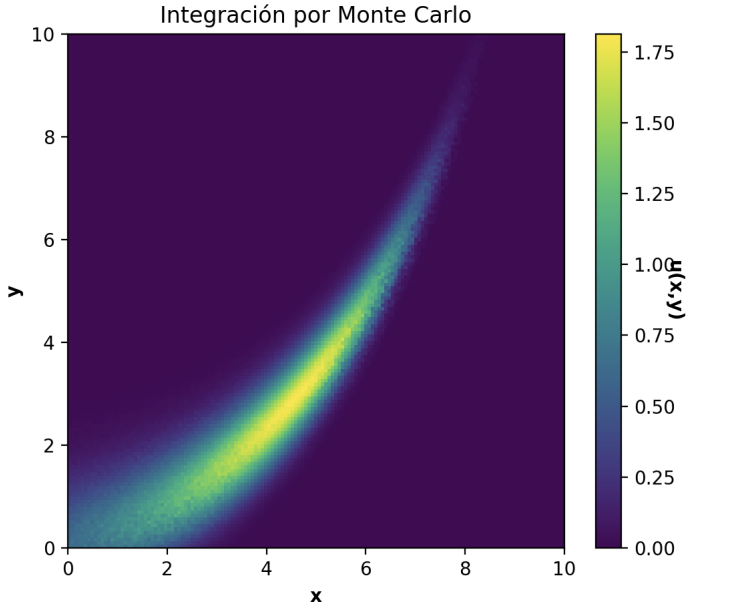


Figura 4.4: *Resultado de integrar por medio de Monte Carlo la EFP para un SMB.*

$$u(\mathbf{y}, t) = \mathbb{E}\left(e^{t(2k_2x+k_3y+k_4+k_1)} f(\mathbf{y})\right) \tag{4.19}$$

con lo que se encontró la solución a la EFP por medio del algoritmo de la sección A.2 para los valores $x, y \in [0, 10]$ que se muestran en la figura4.4.

Esta gráfica nos da la densidad de probabilidad para los estados del SMB a un tiempo t y es de esta solución que tomaremos, de forma aleatoria, los datos que utilizaremos para entrenar una Red Neuronal Artificial que solucione la EFP de nuestro interés.

Capítulo 5

Red neuronal artificial

Una red neuronal artificial (RNA) es un circuito de neuronas artificiales o nodos. Inspiradas en las redes neuronales biológicas que aprenden a través de la experiencia, fueron creadas para "aprender" a resolver tareas al considerar los ejemplos de los que se les provea pues "a pesar de que los modelos propuestos para explicar la estructura y funcionamiento del sistema nervioso de algunos animales son diferentes en muchos aspectos, existe el consenso general de que la esencia del funcionamiento de los arreglos neuronales es la *control a través de la comunicación*"[26].

De esta manera, a la red, se le da un conjunto grande de datos, a cada uno de los cuales, se les asigna un resultado en particular. Después de muchas iteraciones, la red calcula las operaciones necesarias entre los datos de entrada para arrojar un resultado congruente con los ejemplos dados.

Así pues, contrario a otro tipo de programas computacionales, en los que el programador da las instrucciones a la máquina para que, dados unos datos de entrada, opere y nos de un resultado; al entrenar una red, lo que se busca es que, dados los datos de entrada y el resultado buscado, la máquina encuentre las instrucciones que debe seguir para relacionar la entrada con la salida.

Para este trabajo nos proponemos construir y entrenar una RNA que nos arroje, para un puntos del espacio de estados dados, el valor de la EFK correspondiente. Pero, para la construcción de dicha red, primero debemos comprender cómo se construyen y cómo funcionan las RNAs.

5.1. Funcionamiento de las Redes Neuronales

Existen diferentes tipos de neuronas artificiales: los *perceptones*, aunque ahora no son las más comunes, nos permiten entender el comportamiento básico de las distintas neuronas artificiales.

Los perceptrones reciben las entradas, por ejemplo x_1 , x_2 ,$\cdots$ x_n y arrojan un resultado llamado salida (*output*)(fig. 5.1) dado por

$$output = \begin{cases} 0 & \text{si} \quad \sum_j w_j x_j \leq threshold \\ 1 & \text{si} \quad \sum_j w_j x_j > threshold \end{cases} \tag{5.1}$$

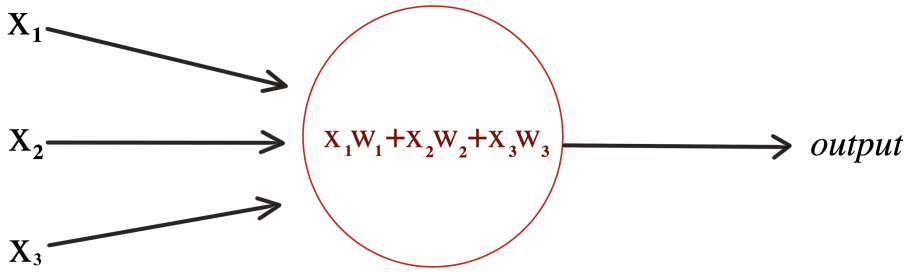


Figura 5.1: *Un perceptrón que obtiene un valor de salida output en función de los valores de entrada x_1, x_2 y x_3.*

con w_j números reales llamados *pesos* y que representan la importancia de las respectivas entradas y el *threshold* un parámetro con valor fijo.

Así pues, este modelo puede pensarse como un mecanismo de toma de desiciones. Con los distintos pesos w_j estaríamos asignándoles distintos valores de importancia a las diferentes variables x_1, x_2,$\cdots x_n$ que influyen en la toma de desiciones y tomaríamos un valor de referencia, *threshold*, para determinar si la decisión es "positiva" ($output = 1$) o "negativa" ($output = 0$). De esta manera, al variar los valores de los pesos y del *threshold*, se pueden obtener distintos modelos para la toma de desiciones.

Este ejemplo, nos permite entender el funcionamiento básico de las neuronas artificiales. Las redes neuronales, sin embargo, están compuestas de muchas neuronas distribuidas en varias capas. La primera de estas capas se le llama la capa de entrada y es en la que se añaden los datos dados por el usuario. Cada neurona en esa capa, produce un valor de salida que recibe cada neurona de la segunda capa como valor de entrada y, a su vez, estas segundas neuronas producen, cada una, un valor de salida que pasan a la siguiente capa, etc. como se puede apreciar en la figura 5.2. Así pues, un arreglo de muchas capas puede llevar a cabo tomas de desiciones más sofisticadas.

Todas las capas que no son la de entrada o la de salida se llaman capas ocultas pero calcular de una manera exacta, con pocas reglas simples, cuántas capas escondidas debe de haber en una red no es posible, en vez de ello, los investigadores han desarrollado varias heurísticas de diseño para las capas ocultas y, así, conseguir que la red se comporte de la manera que queremos.

Para simplificar la ecuación (5.1) podemos escribir la sumatoria como un producto punto de los vectores de pesos w y de entradas x, es decir $w \cdot x \equiv \sum_j w_j x_j$, además podemos reemplazar el valor *threshold* por lo que se conoce como *bias* del perceptrón $b \equiv -threshold$. Así, la salida de cada neurona estará dada por

$$output = \begin{cases} 0 & \text{si} \quad w \cdot x + b \leq 0 \\ 1 & \text{si} \quad w \cdot x + b > 0 \end{cases} \tag{5.2}$$

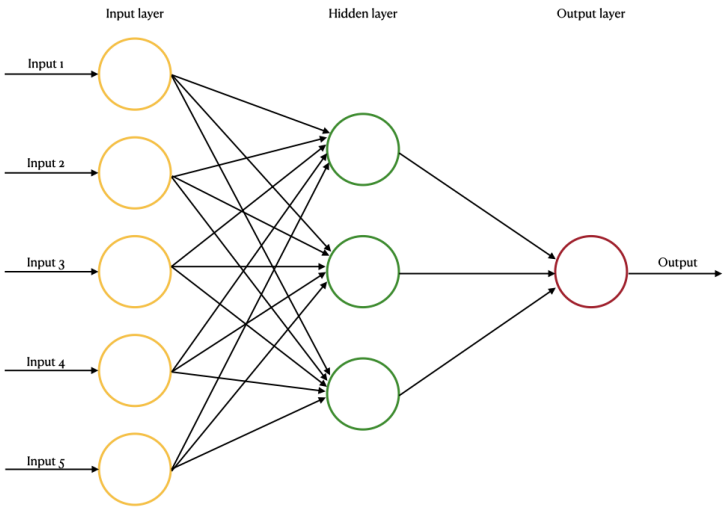


Figura 5.2: *Red neuronal con cinco neuronas en la capa de entrada, tres en la capa escondida y una en la capa de salida.*

Se puede pensar el bias como una medida de que tan fácil es obtener un 1 como salida del perceptrón o qué tan fácil es que el percepción se active.

Por otro lado, otra forma en la que los perceptrones pueden ser usados es como funciones lógicas elementales como AND, OR o NADN, pero esto no representaría ninguna ventaja en comparación con otro tipo de dispositivo computacional, puesto que estas funciones son universales para la computación. Sin embargo, como ya se dijo, la diferencia con otro tipo de algoritmos, las redes neuronales recae en que éstas están diseñadas para "aprender".

Podemos, entonces, desarrollar algoritmos que, de forma automática ajusten los valores de los pesos y los bias sin la intervención directa del programador. Así, las redes neuronales pueden aprender a solucionar problemas en los casos en los que sería muy difícil diseñar directamente un circuito convencional.
Para entender cómo aprenden estos dispositivos, hay que analizar cómo cambia el comportamiento de la red ante pequeños cambios en los pesos o en los biases. Quisiéramos que los pequeños cambios en estos parámetros correspondieran a pequeños cambios en los valores de salida de toda la red para que, de esta manera, se produjeran cada vez mejores resultados y, así, la red "aprenda". Sin embargo, en el caso de los perceptrones, éstos cambios, aún si son pequeños pueden provocar un cambio significativo en la salida. Por ejemplo, al modificar un poco el bias de un perceptron, la salida puede cambiar por completo de 0 a 1 con lo que el comportamiento de la red cambiaría por completo de forma muy complicada.

Este problema se puede evitar haciendo uso de otro tipo de neuronas llamadas *sigmoid neurons* (neuronas sigmoidales). Para estas neuronas, los posibles valores de entrada no son solamente 1 o 0 sino que pueden darse valores entre estos dos números. De esta manera la salida, también podría tener un valor entre 0 y 1 y está definida como $\sigma(w \cdot x + b)$ con σ la función sigmoide dada por

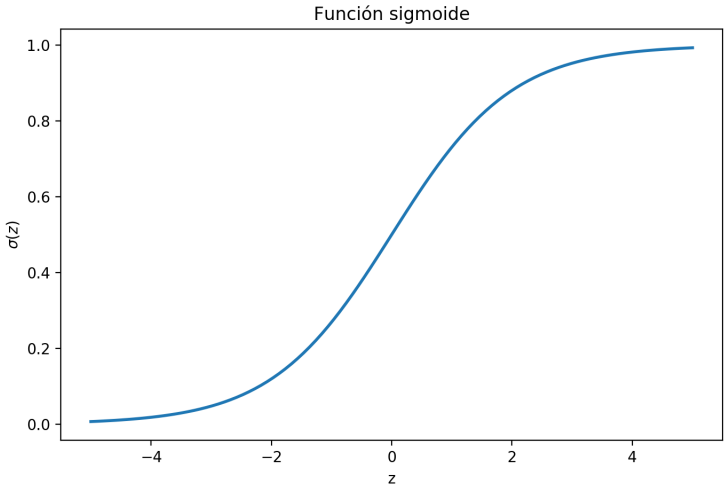


Figura 5.3: *Gráfica de la función sigmoide de z.*

$$\sigma(z) = \frac{1}{1 - e^{-z}} \tag{5.3}$$

De esta manera, como se puede ver en la fugura 5.3, si $z = w \cdot x + b$ es un número positivo grande, e^{-z} será aproximadamente cero y la salida será casi 1. En cambio, si $z = w \cdot x + b$ es muy negativo, e^{-z} tenderá a infinito y la salida de será casi 0. Podemos ver, que esta función es como una función escalón pero suavizada y eso es lo que le da la característica a estas neuronas de que, para cambios pequeños en los pesos Δw_j o del bias Δb corresponde un cambio pequeño en el valor de salida $\Delta output$. Es decir, $\Delta output$ es una función lineal de los cambios Δw_j y $\Delta output$

$$\Delta output \approx \sum_j \frac{\partial output}{\partial w_j} \Delta w_j + \frac{\partial output}{\partial b} \Delta b \tag{5.4}$$

Podríamos, entonces preguntarnos qué sentido tiene un valor de salida del tipo 0.38365... puesto que no representa un estado "activado" ($output = 1$) ni uno "inactivo" ($output = 0$). Lo que se hace, usualmente, en estos casos es interpretar un valor de 0.5 o mayor como 1 y uno menor a 0.5 como 0. Cabe mencionar que, aunque la función sigmoide es comúnmente usada para calcular las salidas de las neuronas en una red, no es la única función que es posible usar, en general podemos usar funciones para la salida $f(w \cdot x + b)$.

Recapitulando, lo que queremos es que la red encuentre los pesos y los biases necesarios para que la red, dado un conjunto de datos de entrada x nos dé, como valor de salida, un valor $y(x)$. Para ello es necesario "entrenar" a la red, este proceso consiste en proporcionarle a la red un conjunto de datos de entrada x junto con sus respectivos datos de salida, con ellos la red tiene que calcular los biases y pesos necesarios para establecer la relación $x \to y(x)$. Por lo tanto, tenemos que construir un algoritmo que permita encontrar los valores óptimos para todos los b y los w_j.

Para cuantificar que tan bien estamos logrando este objetivo definimos a la función de costo

$$C(w,b) \equiv \frac{1}{2n}\sum_x \|y(x)-a\|^2 \tag{5.5}$$

donde b representa todos los biases, w todos los pesos, n es el número total de entradas de entrenamiento, a es el vector de salidas cuando x es la entrada y la suma es sobre todas las entradas. Llamamos a esta función *función cuadrática de costo* o *error medio cuadrado MSE*. Podemos ver que $C(w,b)$ es no negativa y se hace pequeña conforme a se aproxima al valor de $y(x)$ para todo valor de entrada x; si esto ocurre podremos concluir que nuestra red está haciendo un buen trabajo. En cambio, si $C(w,b)$ es grande, significa que a no está cerca de $y(x)$. Es decir, al entrenar una red neuronal, buscamos minimizar $C(w,b)$ como función de los biases y los pesos. Esto lo vamos a lograr usando el método del *descenso del gradiente*.

Así, si por ejemplo, $C(\vec{v})$ es la función que queremos minimizar, con $\vec{v}$ un vector de dimensión m, $\Delta \equiv (\Delta v_1, \cdots, \Delta v_m)^T$ será el vector de cambio en dirección v y el cambio en C estará dado por

$$\Delta C \approx \nabla C \cdot \Delta v \tag{5.6}$$

con $\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \cdots, \frac{\partial C}{\partial v_m}\right)^T$ de donde tenemos que escoger Δv para hacer ΔC negativo.

Supongamos, entonces que escogemos

$$\Delta v = -\eta \nabla C \tag{5.7}$$

con η un parámetro positivo muy pequeño conocido como la *taza de aprendizaje*. Así, tendremos que el cambio en C estará dado por

$$\Delta C \approx -\eta \|\nabla C\|^2 \tag{5.8}$$

Lo cual cumple con la característica que buscábamos de que ΔC sea menor que cero y nos permitirá usar la ecuación (5.6) para calcular el valor de Δv haciendo

$$v \to v' = v - \eta \nabla C \tag{5.9}$$

iterativamente hasta que alcancemos un mínimo global para C. Es decir, para una función $C = C(w,b)$, minimizar C correspondería a aplicar la regla

$$w_k \to w_k' = w_k - \eta \frac{\partial C}{\partial w_k} \tag{5.10}$$

$$b_l \to b_l' = b_l - \eta \frac{\partial C}{\partial b_l} \tag{5.11}$$

Hay que tener cuidado al elegir el valor de η pues un valor muy grande podría hacer $\Delta C > 0$, lo cual no queremos, pero un valor de η muy pequeño podría volver al cálculo muy lento. Además, por cómo está definida la función de costo en la ecuación (5.5), podemos verla como un promedio

$$C = \frac{1}{n}\sum_x C_x \tag{5.12}$$

con $C_x \equiv \frac{\|y(x)-a\|^2}{2}$ para cada valor de entrada x.

Con esta idea y para hacer más rápido el proceso de aprendizaje podemos usar el descenso del gradiente estocástico que consiste en calcular un estimado de ∇C al tomar solo un conjunto de $m < n$ datos de entrada X_j escogidos aleatoriamente en vez de todos los disponibles. Así

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} C_{X_j} \tag{5.13}$$

Con lo que las reglas (5.10) y (5.10), se convertirían en

$$w_k \to w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k} \tag{5.14}$$

$$b_l \to b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l} \tag{5.15}$$

Donde las sumas son sobre todos los ejemplos de entrenamiento X_j en el mini lote escogido. Después, se puede escoger de forma aleatoria otro lote y después otro hasta agotar todos los datos de entrenamiento disponibles. Este procedimiento se denomina una "época de entrenamiento" (*epoch*) y al variar la cantidad de épocas de entrenamiento podemos mejorar el "aprendizaje" de la red.

5.2. Algoritmo de Back Propagation

Como ya se dijo, una vez construida nuestra RNA queremos que esta "aprenda" a resolver problemas y esto lo va a lograr ajustando los valores de los parámetros de pesos y biases. Además una red, entre más capas y neuronas tenga, va a ser capaz de resolver problemas más complejos. Sin embargo, con cada neurona aparecerán nuevos parámetros w y b, por lo que entre más grande sea una red, la función de costo (5.5) se volverá más compleja y más difícil de minimizar.
Así para aplicar el descenso del gradiente a la función de costo, debemos primero calcular el gradiente. No obstante, puesto que C es función de todos los w y los b, para calcular ∇C sería necesario aplicar las reglas (5.14) y (5.15) para cada una de las w_k y las b_l de la red. Para evitar esos cálculos y hacer más eficiente el descenso del gradiente se diseñó el algoritmo de *back propagation*.

Para entrenar una RNA, por medio del "aprendizaje supervisado" tomaremos un conjunto de datos del tipo $(\mathbf{x}, \mathbf{y})$ llamados "datos de entrenamiento". En un principio se asignan valores aleatorios para cada uno de los pesos y los biases y, de los datos de entrenamiento, se introduce un dato de entrada $\mathbf{x}$ a la red. Ésta calcula un dato de salida $\mathbf{a}$ y lo compara con el dato de salida $\mathbf{y}$ asignado a $\mathbf{x}$. Puesto que $\mathbf{a}$ es un valor aleatorio, en general, $\mathbf{a} \neq \mathbf{y}$ con lo que se va a obtener una señal de error.
El algoritmo de *back propagation*, en vez de calcular todos los diferentes $\frac{\partial C}{\partial w_k}$ y $\frac{\partial C}{\partial b_l}$ para calcular ∇C, propaga la señal de error desde la capa de salida hasta la primera capa oculta haciendo pasos hacia atras.

Partimos de ver que la función de costo es, en realidad una composición de funciones

$$C(w,b) = C(f(z(w,b))) \tag{5.16}$$

con f la función de activación y z la suma ponderada de los datos de entrada $\mathbf{x}$ que depende directamente de los w y b. De esta manera, para una red de L capas la variación de la función de coste con respecto a los pesos y los biases de la última capa será

$$\frac{\partial C}{\partial w_L} = \frac{\partial C}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \cdot \frac{\partial z_L}{\partial w_L} \tag{5.17}$$

$$\frac{\partial C}{\partial b_L} = \frac{\partial C}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \cdot \frac{\partial z_L}{\partial b_L} \tag{5.18}$$

así para la j-ésima neurona de la capa L

$$C(f_{j,L}) = \frac{1}{2}\sum_j (y_j - f_{j,L})^2 \implies \frac{\partial C}{\partial f_L} = (f_{j,L} - y_j) \tag{5.19}$$

$$f_L(z_L) = \frac{1}{1+e^{-z_L}} \implies \frac{\partial f_L}{\partial z_L} = f_L(z_L)\cdot(1 - f_L(z_L)) \tag{5.20}$$

$$z_L = \sum_i a_{i,L-1} w_{i_L} + b_L \implies \begin{cases} \frac{\partial z_L}{\partial b_L} = 1 \\ \frac{\partial z_L}{\partial w_L} = f_{i,L-1} \end{cases} \tag{5.21}$$

donde vemos que la variación de la suma ponderada z_L respecto a una variación en los pesos w_L está dada por la salida de cada neurona de la capa anterior. Además podemos interpretar $\frac{\partial C}{\partial z_L}$ como el error imputado a la neurona pues nos dice como varia C si hay un cambio en la suma ponderada z_L de la neurona: si éste es grande un cambio en los valores de la neurona influye mucho en el dato de salida pero, si es pequeño, un cambio en la neurona no afecta tanto a la salida. Detonamos esta parcial como

$$\frac{\partial C}{\partial z_L} = \delta_L \tag{5.22}$$

con lo que, para la ultima capa podemos calcular ∇C con

$$\frac{\partial C}{\partial w_L} = \delta_L f_{i,L-1} \tag{5.23}$$

$$\frac{\partial C}{\partial b_L} = \delta_L \tag{5.24}$$

$$\delta_L = \frac{\partial C}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \tag{5.25}$$

Sin embargo, para calcular los parámetros de la capa $L-1$ tenemos que calcular

$$\frac{\partial C}{\partial w_{L-1}} = \frac{\partial C}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \cdot \frac{\partial z_L}{\partial f_{L-1}} \cdot \frac{\partial f_{L-1}}{\partial z_{L-1}} \cdot \frac{\partial z_{L-1}}{\partial w_{L-1}} \tag{5.26}$$

$$\frac{\partial C}{\partial b_L} = \frac{\partial C}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \cdot \frac{\partial z_L}{\partial f_{L-1}} \cdot \frac{\partial f_{L-1}}{\partial z_{L-1}} \cdot \frac{\partial z_{L-1}}{\partial b_{L-1}} \tag{5.27}$$

lo cual no representa mayor problema pues por (5.21), (5.20) y (5.25) ya tenemos resuelto casi todos los términos en las expresiones anteriores. Solo quedaría

$$\frac{\partial z_L}{\partial f_{L-1}} = W_L \tag{5.28}$$

que es la matriz de parámetros w que conecta ambas capas y representa la variación de la suma ponderada de la capa L cuando hay un cambio en la salida de una neurona en la capa $L-1$[1].
Sin embargo, podemos apreciar que la expresión

$$\frac{\partial C}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \cdot \frac{\partial z_L}{\partial f_{L-1}} \cdot \frac{\partial f_{L-1}}{\partial z_{L-1}} = \delta_{L-1} \tag{5.29}$$

nos da, de forma análoga a (5.25) el error imputado a la neurona de una capa oculta y ésta se puede aplicar recursivamente hacia atrás a cada una de las capas ocultas de la red con lo que se consigue aplicar el algoritmo de *back propagation* siguiendo los siguientes pasos:

- Calcular el error en la última capa

$$\delta_L = \frac{\partial C}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \tag{5.30}$$

- Retropropagar el error a la capa anterior

$$\delta_{l-1} = W_l \delta_l \cdot \frac{\partial f_{l-1}}{\partial z_{l-1}} \tag{5.31}$$

- Calcular las derivadas de la capa oculta

$$\frac{\partial C}{\partial b_{l-1}} = \delta_{l-1} \quad , \quad \frac{\partial C}{\partial w_{l-1}} = \delta_{l-1} f_{l-2} \tag{5.32}$$

Una vez calculado como varia la función de costo con respecto a los parámetros b y w de cada capa podemos modificarlos por medio del descenso del gradiente para minimizar C y lograr que nuestra red "aprenda".

5.3. Diseño y entrenamiento de una RNA

Para diseñar la red que queremos entrenar, partimos del planteamiento de la sección 5.1 y se utilizó la subrutina del Apéndice B que es una modificación del algoritmo propuesto en Nielsen [23]. En él se define una clase de objetos que representarán a cada una de las capas utilizadas en la red. Se inicializa la red con parámetros w y b aleatorios y se entrena dando un conjunto de datos de entrenamiento. Se toma también un conjunto más pequeño y diferente de datos que servirán como datos de prueba.
En cada época de entrenamiento, la red evalúa a cuántos datos de prueba se ajusta bien. Esta es una manera de medir la eficiencia de la red, sin embargo, como ya se dijo, la forma más efectiva con la que podemos evaluar cómo está "aprediendo" la RNA es por medio de la función de costo. Para que nuestra red aprenda C tiene que converger tanto para los datos de entrenamiento como para los de prueba con esto evitamos el *sobreajuste* que sucede cuando la red ajusta sus parámetros para arrojar resultados congruentes con los datos de entrenamiento pero no es capaz de generalizar para nuevos datos. De esta manera, en una gráfica de la función de costo, podríamos apreciar que C converge para los datos de entrenamiento pero no para los de prueba

[1] en Goodfellow et al. [10] se da una explicación más detallada de este cálculo y la aplicación del algoritmo de *back propagation*

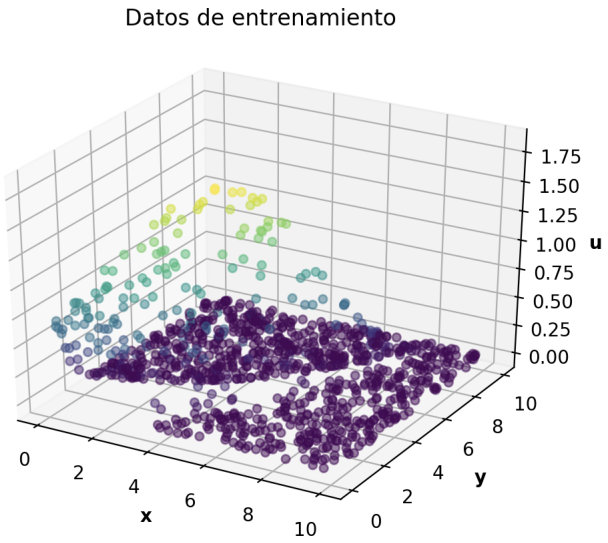


Figura 5.4: *Puntos obtenidos a partir de la solución a la EFP por Monte Carlo que fueron utilizados como datos de entrenamiento para la RNA*

Para nuestro caso particular se utilizaron como datos de entrenamiento una lista de dos entradas $[\mathbf{x}, \mathbf{y}]$ donde la primera entrada es otra lista $\mathbf{x} = [x, y]$ que representa un punto en el espacio de estados de nuestro sistema con $x, y \in [0, 10]$ y la segunda entrada $\mathbf{y} = u$ es el valor de la FFP para el punto correspondiente.

Así, en la consola de python, se creó un conjunto de 15,000 puntos en el plano XY de los cuales 80 % se destinarán para el entrenamiento de la red y el 20 % restante servirán como datos de prueba.

```
IN=10*np.random.random([15000,2])
np.random.shuffle(IN)
x_train=IN[:12000,0]
y_train=IN[:12000,1]
x_test=IN[12000:,0]
y_test=IN[12000:,1]
```

Posteriormente, por medio de la subrutina de A.2, se resolvió por el método Monte Carlo la EFP para cada uno de estos puntos (figura 5.4) y se guardaron los valores obtenidos; tanto para los datos de entrenamiento como para los datos de prueba.

```
training_data= FeyKacMonCar(x_train,y_train)
test_data= FeyKacMonCar(x_test,y_test)

with open("training_data.dat", 'wb') as f:
```

```
    pickle.dump(training_data, f)
with open("test_data.dat", 'wb') as f:
    pickle.dump(test_data, f)
```

Una vez creados los puntos de entrenamiento y de prueba se construyó una primera RNA

```
net=network.Network([2,100,1])
```

La lista $[2, 100, 1]$ representa el número de neuronas en cada capa; siendo el primer número el tamaño de la capa de entrada, el último el de la capa de salida y los intermedios los de las capas ocultas. Puesto que el problema a resolver tiene como datos de entrada un vector bidimensional y como dato de salida un solo valor numérico, la capa de entrada constará de dos neuronas y la capa de salida de una neurona. Como prueba inicial tomamos una capa oculta con 100 neuronas en ella.

Para evaluar la eficiencia de nuestra red se utilizó la función *evaluate*

```
net.evaluate(test_data)
```

que nos da la cantidad de datos de prueba a la que la red de ajustó suficientemente bien. En nuestro caso particular, tomamos como dato "bien ajustado" aquel valor que obtuvo la red que difería del valor de salida de los datos de entrenamiento en dos ordenes de magnitud. Es decir si $(\mathbf{y} - \mathbf{a}) < 0.01$.
Además, para observar cómo converge la función de costo C mientras se entrena la red, se aplicó el método del descenso del gradiente SDG tomando un mini lote de 2,000 puntos y una taza de aprendizaje de 3.0. Se hicieron 300 épocas y en cada una de las épocas de entrenamiento se calculó el valor de la función de costo con un conjunto aleatorio de datos de entrenamiento del tamaño del mini lote

```
C=net.SGD(training_data,300,2000,3.0,test_data=test_data)
```

Con estos parámetros se obtuvo una evaluación correcta de un 66.7 % de los datos de prueba y la gráfica de la figura 5.5 que muestra la convergencia de la función de costo para los conjuntos de datos de entrenamiento y de prueba. Podemos ver que, a partir de las 200 épocas, ya tenemos una buena convergencia de ambos conjuntos de puntos.

Posteriormente se entrenó una RNA más compleja. Esta vez se tomaron dos capas ocultas, la primera de 200 neuronas y la segunda de 100. Se tomó un mini lote de 3,000 datos, la taza de aprendizaje de 3.0 y se hicieron 200 épocas de entrenamiento. Con ello se obtuvo que la red se ajustó bien a un 73.7 % de los datos de prueba. Nuevamente se obtuvo la el valor de la función de costo para cada una de las épocas y para ambos conjuntos de datos y se obtuvo la gráfica de la figura 5.6.

Posteriormente se utilizaron ambas redes ya entrenadas para encontrar la solución a la EFP para los puntos de una malla construida a partir del arreglo

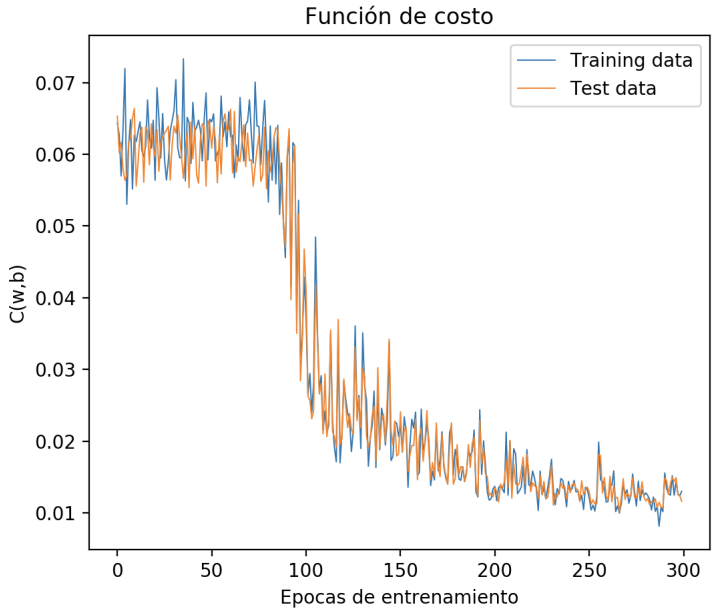


Figura 5.5: *Gráfica de la función de costo de una RNA (2,100,1) para un conjunto de datos de entrenamiento y uno de datos de prueba*

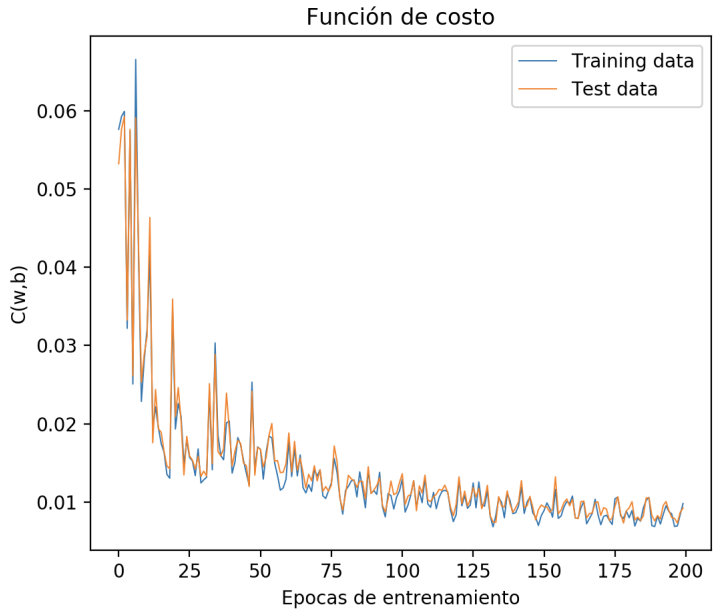
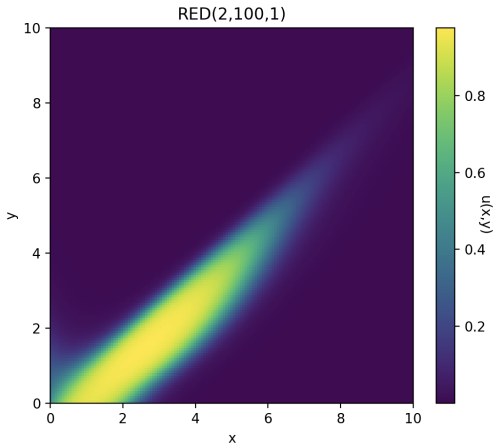


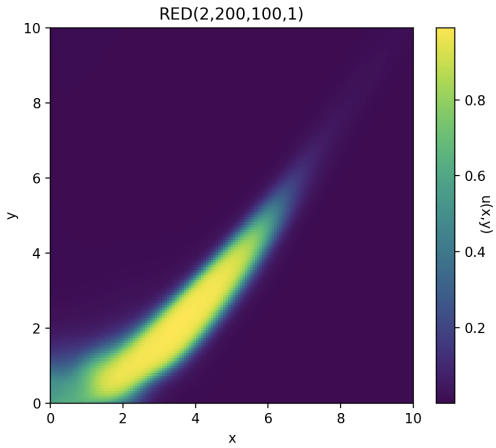
Figura 5.6: *Gráfica de la función de costo de una RNA (2,200,100,1) para un conjunto de datos de entrenamiento y uno de datos de prueba*

```
x=np.linspace(0,10,150)
y=np.linspace(0,10,150)
```

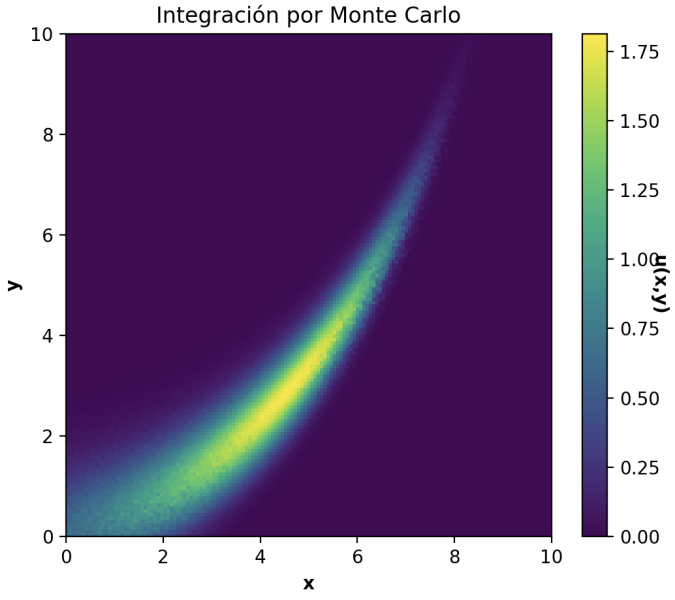
con lo que se obtuvieron las gráficas 5.7a para la red (2,100,1) y 5.7b para la red (2-200-100-1). En la figura 5.7 podemos apreciar la comparación de las soluciones obtenidas por medio de estas redes y por medio de la integración por el método Monte Carlo.



(a) *Solución a la EFP por medio de una RNA con una capa oculta de 100 neuronas*



(b) *Solución a la EFP por medio de una RNA con dos capas ocultas de 200 y 100 neuronas respectivamente*



(c) *Resultado de integrar por medio de Monte Carlo la EFP para un SMB.*

Figura 5.7: *Comparación de las soluciones de la EFP obtenidas por Monte Carlo y por RNAs*

Conclusiones

En este trabajo de investigación nos propusimos resolver la ecuación de Fokker-Planck asociada a un sistema mínimo biestable por medio del método de integración Monte Carlo. Esto debido a que, los sistemas biológicos son, en general, sistemas complejos de muchas dimensiones y este método, al tomar un número finito de puntos aleatorios, nos permite solucionar de forma estadística problemas de muchas dimensiones, por lo que se vuelve más eficiente que otro tipo de métodos numéricos.

Vimos, además, que aunque nuestro sistema de interés es un sistema relativamente simple nos permite modelar y comprender diversos procesos de desarrollo celular, por lo que su estudio es sumamente relevante y la obtención de su paisaje epigenético resulta útil para comprender y predecir la evolución temporal de las variables implicadas en muchos procesos celulares.

Otra de las hipótesis de este trabajo era que, a partir de algunos datos obtenidos por medio de una integración por Monte Carlo podíamos entrenar una red neuronal artificial que nos permitiera obtener una solución completa de la EFP. Como vimos, las RNA nos permiten obtener una buena aproximación cualitativa de este resultado y, por medio de redes más complejas, podemos obtener mejores resultados.

Podemos concluir, entonces, que la metodología propuesta para obtener el paisaje epigenético de procesos de desarrollo podría extenderse a sistemas más complejos y de dimensiones más altas. Es decir, en vez de intentar resolver el sistema de ecuaciones diferenciales que describan su evolución en el tiempo directamente por medio de algún método de integración numérico, podemos encontrar el potencial energético de dicho proceso al encontrar la solución, de forma estadística, para un número suficientemente pequeño de puntos y, con ellos, entrenar una RNA que nos de una imagen completa del paisaje epigenético de nuestro proceso de desarrollo

Apéndice A

Algoritmos para resolver la EFP

A.1. Caminante aleatorio

Discretización de la ecuación de Langevin que nos da la evolución en el tiempo de las variables de estado del SMB después de una unidad de tiempo t.

```
import numpy as np
import random
import numpy.random

def randwalk2D(N,y10,y20):
    t=np.zeros(N)
    y1=np.zeros(N)
    y2=np.zeros(N)
    k1=8
    k2=1
    k3=1
    k4=1.5
    dt=0.001
    y1[0]=y10
    y2[0]=y20

    for i in range(N-1):
        n = random.normalvariate(0, 1)
        y1[i+1] = y1[i]-(2*k1*y2[i] - k2*y1[i]**2-k3*y1[i]*y2[i]-k4
            *y1[i])*dt + n*np.sqrt(dt) #c1(y,t)=1
        y2[i+1] = y2[i]-(k2*y1[i]**2- k1*y2[i])*dt + n*np.sqrt(dt)
            #c2(y,t)=1
        t[i+1] = t[i] + dt

    return y1[-1],y2[-1]
```

A.2. Monte Carlo

Algoritmo para resolver por Método Monte Carlo la Ecuación de Fokker-Planck asociada al SMB. Se obtiene una lista de dos entradas, la primera representa un punto en el espacio de estados bidimensional y la segunda el valor de la EFP para el estado correspondiente.

```
import numpy as np
from randwalk2D import *
import math

sigma1, sigma2=3,3
mu1,mu2=5,5
rho=0.2
R=1-rho**2

def f0(x,y):
    f0=math.exp(-(((x-mu1)/sigma1)**2+((y-mu1)/sigma2)**2-2*rho*((x
       -mu1)/sigma1)*((y-mu1)/sigma2))/(2*R))/(2*math.pi*sigma1*
       sigma2*math.sqrt(R))
    return f0

def FeyKacMonCar(x,y):
    Nit=10
    xr=np.zeros(Nit)
    yr=np.zeros(Nit)
    F=np.zeros(Nit)
    e=np.zeros(Nit)
    u=np.zeros((len(x), len(y)))
    X=np.zeros((len(x)*len(y),2))
    k1=8
    k2=1
    k3=1
    k4=1.5
    T=100
    for i in range(len(x)):
        for k in range(len(xr)-1):
            xr[k],yr[k]= randwalk2D(T,x[i],y[i])
            e[k]=-T*0.001*(2*k2*xr[k]+k3*yr[k]+k4+k1)
            F[k]=f0(xr[k],yr[k])*np.exp(e[k])
            u[i]=sum(F)/Nit


    U=np.zeros(len(u))
    for n in range(len(u)):
```

```
37          U[n]=1000*u[n][0]
38
39      X=np.ravel(x)
40      Y=np.ravel(y)
41
42      outputs=[]
43      for m in range(len(U)):
44          outputs.append(U[m])
45
46      inputs=[]
47      for m in range(len(X)):
48          inputs.append((X[m],Y[m]))
49
50      trainingD=list(zip(inputs,outputs))
51      return trainingD
```

Apéndice B

Algoritmo para la construcción y entrenamiento de una RNA

```
"""
network.py
~~~~~~~~~~

A module to implement the stochastic gradient descent learning
algorithm for a feedforward neural network.  Gradients are calculated
using backpropagation.  Note that I have focused on making the code
simple, easily readable, and easily modifiable.  It is not optimized,
and omits many desirable features.
"""

#### Libraries
# Standard library
import random

# Third-party libraries
import numpy as np

class Network(object):

    def __init__(self, sizes):
        """The list ``sizes`` contains the number of neurons in the
        respective layers of the network.  For example, if the list
        was [2, 3, 1] then it would be a three-layer network, with
           the
```

```
25          first layer containing 2 neurons, the second layer 3
               neurons,
26          and the third layer 1 neuron.  The biases and weights for
               the
27          network are initialized randomly, using a Gaussian
28          distribution with mean 0, and variance 1.  Note that the
               first
29          layer is assumed to be an input layer, and by convention we
30          won't set any biases for those neurons, since biases are
               only
31          ever used in computing the outputs from later layers."""
32          self.num_layers = len(sizes)
33          self.sizes = sizes
34          self.biases = [np.random.randn(y) for y in sizes[1:]]
35          self.weights = [np.random.randn(y, x)
36                          for x, y in zip(sizes[:-1], sizes[1:])]
37
38      def feedforward(self, a):
39          """Return the output of the network if ``a`` is input."""
40          for b, w in zip(self.biases, self.weights):
41              a = sigmoid(np.dot(w, a)+b)
42          return a
43
44      def SGD(self, training_data, epochs, mini_batch_size, eta,
45              test_data=None):
46          """Train the neural network using mini-batch stochastic
47          gradient descent.  The ``training_data`` is a list of
               tuples
48          ``(x, y)`` representing the training inputs and the desired
49          outputs.  The other non-optional parameters are
50          self-explanatory.  If ``test_data`` is provided then the
51          network will be evaluated against the test data after each
52          epoch, and partial progress printed out.  This is useful
               for
53          tracking progress, but slows things down substantially."""
54          C_train=np.zeros(epochs)
55          C_test=np.zeros(epochs)
56          c_train=np.zeros(mini_batch_size)
57          c_test=np.zeros(mini_batch_size)
58          if test_data: n_test = len(test_data)
59          n = len(training_data)
60          for j in range(epochs):
61              random.shuffle(training_data)
```

```
62              mini_batches = [
63                  training_data[k:k+mini_batch_size]
64                  for k in range(0, n, mini_batch_size)]
65              for mini_batch in mini_batches:
66                  self.update_mini_batch(mini_batch, eta)
67
68
69              if test_data:
70                  print ("Epoch {0}: {1} / {2}".format(
71                      j, self.evaluate(test_data), n_test))
72                  mini_test=random.sample(test_data,mini_batch_size)
73                  for k in range(mini_batch_size):
74                      c_test[k]=abs(self.feedforward(mini_test[k][0])
                            -mini_test[k][1])**2
75                  C_test[j]=np.sum(c_test)/(2*mini_batch_size)
76
77              else:
78                  print ("Epoch {0} complete".format(j))
79
80
81              mini_train=random.sample(training_data,mini_batch_size)
82              for k in range(mini_batch_size):
83                  c_train[k]=abs(self.feedforward(mini_train[k][0])-
                        mini_train[k][1])**2
84
85
86              C_train[j]=np.sum(c_train)/(2*mini_batch_size)
87
88          if test_data:
89              return (C_test,C_train)
90          else:
91              return C_train
92
93
94      def update_mini_batch(self, mini_batch, eta):
95          """Update the network's weights and biases by applying
96          gradient descent using backpropagation to a single mini
                batch.
97          The ``mini_batch`` is a list of tuples ``(x, y)``, and ``
                eta``
98          is the learning rate."""
99          nabla_b = [np.zeros(b.shape) for b in self.biases]
100         nabla_w = [np.zeros(w.shape) for w in self.weights]
```

```
101         for x, y in mini_batch:
102             delta_nabla_b, delta_nabla_w = self.backprop(x, y)
103             nabla_b = [nb+dnb for nb, dnb in zip(nabla_b,
                    delta_nabla_b)]
104             nabla_w = [nw+dnw for nw, dnw in zip(nabla_w,
                    delta_nabla_w)]
105         self.weights = [w-(eta/len(mini_batch))*nw
106                         for w, nw in zip(self.weights, nabla_w)]
107         self.biases = [b-(eta/len(mini_batch))*nb
108                        for b, nb in zip(self.biases, nabla_b)]
109
110
111     def backprop(self, x, y):
112         """Return a tuple ``(nabla_b, nabla_w)`` representing the
113         gradient for the cost function C_x.  ``nabla_b`` and
114         ``nabla_w`` are layer-by-layer lists of numpy arrays,
                similar
115         to ``self.biases`` and ``self.weights``."""
116         nabla_b = [np.zeros(b.shape) for b in self.biases]
117         nabla_w = [np.zeros(w.shape) for w in self.weights]
118         # feedforward
119         activation = x
120         activations = [x] # list to store all the activations,
                layer by layer
121         zs = [] # list to store all the z vectors, layer by layer
122         for b, w in zip(self.biases, self.weights):
123             z = np.dot(w, activation) + b
124             zs.append(z)
125             activation = sigmoid(z)
126             activations.append(activation)
127         # backward pass
128         delta = self.cost_derivative(activations[-1], y) * \
129             sigmoid_prime(zs[-1])
130         nabla_b[-1] = delta
131         nabla_w[-1] = np.outer(delta,activations[-2])
132         # Note that the variable l in the loop below is used a
                little
133         # differently to the notation in Chapter 2 of the book.
                Here,
134         # l = 1 means the last layer of neurons, l = 2 is the
135         # second-last layer, and so on.  It's a renumbering of the
136         # scheme in the book, used here to take advantage of the
                fact
```

```
137         # that Python can use negative indices in lists.
138         for l in range(2, self.num_layers):
139             z = zs[-l]
140             sp = sigmoid_prime(z)
141             delta = np.dot(self.weights[-l+1].transpose(), delta) *
                    sp
142             nabla_b[-l] = delta
143             nabla_w[-l] = np.outer(delta, activations[-l-1])
144         return (nabla_b, nabla_w)
145 
146     def evaluate(self, test_data):
147         """Return the number of test inputs for which the neural
148         network outputs the correct result. Note that the neural
149         network's output is assumed to be the index of whichever
150         neuron in the final layer has the highest activation."""
151         test_results=0
152         for (x,y) in test_data:
153             if abs(self.feedforward(x)-y)<=0.01:
154                 test_results=test_results+1
155         return test_results
156 
157     def cost_derivative(self, output_activations, y):
158         """Return the vector of partial derivatives \partial C_x /
159         \partial a for the output activations."""
160         return (output_activations-y)
161 
162     def results(self, x):
163         y=[]
164         y.append(self.feedforward(x))
165         return y
166 
167 #### Miscellaneous functions
168 def sigmoid(z):
169     """The sigmoid function."""
170     return 1.0/(1.0+np.exp(-z))
171 
172 def sigmoid_prime(z):
173     """Derivative of the sigmoid function."""
174     return sigmoid(z)*(1-sigmoid(z))
```

Bibliografía

[1] Alberghina, L. and Westernhoff, H. e. (2008). *System Biology: Definitions and perspectives*. Springer.

[2] Aldana, M., Balleza, E., Kauffman, S., and Resendiz, O. (2007). Robustness and evolvability in genetic regulatory networks. *Journal of Theoretical Biology*, 245:433–448.

[3] Alvarez-Buylla, M., Martinez-Garcia, J., et al. (2018). *Modeling Methods for Medical Systems Biology. Regulatory Dynamics Underlying the Emergence of Disease Processes*. Springer.

[4] Cai, Z. and Liu, J. (2018). Approximating quantum many-body wave functions using artificial neural networks. *Physical Review B*, 97(3).

[5] Davidson, E. (2006). *The regulatory genome: Gene regulatory networks in development and evolution*. Academic Press.

[6] Einstein, A. (1905). Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der Physik*, 17:549–560.

[7] Eissing, T., Conzelmann, H., Gilles, E., Allgoewer, F., Bullinger, E., and Sheurich, P. (2004). Bistability analyses of a caspase activation model for receptor-induced apoptosis. *J Biol Chem*, 279:36892–36897.

[8] Ferrell, J. (2002). Self-perpetuating states in signal transduction: positive feedback, double-negative feedback and bistability. *Current Opinion in Cell Biology*, 14(2):140 – 148.

[9] Gobet, E. (2012). *Introduction to stochastic calculus and to the resolution of PDEs using Monte Carlo simulations - Lectures notes of XV Spanish-French School on Numerical Simulation in Physics and Engineering*. École thématique. XV Spanish-French School on Numerical Simulation in Physics and Engineering.

[10] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[11] Gorin, A. (1994). Introduction to the special issue on neural networks for speech processing. *IEEE Transactions on Speech and Audio Processing*, 2:113–114.

[12] Hammersley, J. and Handscomb, D. C. (1975). *Monte Carlo Methods*. Methuen & Co.

[13] Jain, A. K., Duin, R. P. W., and Jianchang Mao (2000). Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37.

[14] Jenq-Nen Hwang, Sun-Yan Kung, Niranjan, M., and Principe, J. C. (1997). The past, present, and future of neural networks for signal processing. *IEEE Signal Processing Magazine*, 14(6):28–48.

[15] Karatzas, I. and Shreve, S. E. (1991). *Brownian Motion and Stochastic Calculus*. Springer Science.

[16] Karlebach, G. and Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780.

[17] Kauffman, S. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467.

[18] Kellershohn, N. and Laurent, M. (2001). Prion diseases: dynamics of the infection and properties of the bistable transition. *J Biol Chem*, 81.

[19] Kim, K. and Wang, J. (2007). Potential energy landscape and robustness of a gene regulatory network: Toggle switch. *PLoS Computational Biology*, 3(3):0565–0577.

[20] Marcos, A. (2012). Biología sistémica y filosofía de la naturaleza. *Eikasia Revista de Filosofia*, 43:95–109.

[21] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115 – 133.

[22] Mörters, P. and Peres, Y. (2008). *Brownian Motion*. Cambridge University Press.

[23] Nielsen, M. (2015). *Neural Networks and Deep Learning*. Determination Press.

[24] Pavliotis, G. A. (2014). *Stochastic Processes and Applications. Diffusion Processes, the Fokker-Planck and Langevin Equations*. Springer Science.

[25] Reiter, D. (2007). *The Monte Carlo Method, an Introduction*, volume 739, pages 63–78.

[26] Rojas, R. (2013). *Neural Networks. Asystematic Introduction*. Springer.

[27] Scott, M. (2012). *Applied Stochastic Processes in science and engineering*. University of Waterloo.

[28] Spencer, B. F. and Bergman, L. A. (1969). On the numerical solution of the fokker-planck equation for nonlinear stochastic system. *Journal of Theoretical Biology*, 22(3):437–467.

[29] Thompson, W. (1901). Nineteenth century clouds over the dynamical theory of heat and light. *Phil. Mag.*, 2(6):1–40.

[30] Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the theory of the brownian motion. *Phys. Rev.*, 36:823–841.

[31] Veening, J., Smits, W., and Kuipers, O. (2008). Bistability, epigenetics, and bet-hedging in bacteria. *Anunual Review of Microbiology*, 62:193–210.

[32] Villarreal, C., Padilla-Longoria, P., and Álvarez Buylla, M. (2012). General theory of genotype to phenotype mapping: Derivation of epigenetic landscapes from n-node complex gene regulatory networks. *Physical Review Letters*, 109(11):118102.

[33] Waddington, C. (1957). *The Strategy of the Genes. A discussion of some aspects of theorical biology*. George Allen & Unwin.

[34] Wilhem, T. (1993). The smallest chemical reaction system with bistability. *Nonlinear Dynamics*, 4:357–372.

[35] Xiong, W. and Ferrell, J. (2003). A positive-feedback-based bistable 'memory module' that governs a cell fate decision. *Nature*, 426.