Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE MÁSTER

Máster en Matemáticas

ESTIMACIÓN DE ECUACIONES EN DERIVADAS PARCIALES MEDIANTE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO

Autor: Alejandro Gutiérrez Mielgo
Tutores: Víctor Gatón Bustillo, José Belarmino Pulido Junquera
2024

Agosto y vendimia no es cada día, y sí cada año;
unos con ganancia y otros con daño.
Refrán castellano

Quiero agradecer a mis padres el interés y la preocupación que muestran siempre por que me vayan bien las cosas, incluyendo este TFM, que no podía ser una excepción. También quiero dar las gracias a mi hermana por su apoyo y a Carla por estar constantemente animándome y acompañándome, insuflándome fuerzas cada día. Para concluir, me gustaría dar las gracias a mis amigos y, por supuesto, a mis tutores, por guiarme, pero sobre todo, por su esfuerzo y su dedicación en este trabajo.

Tabla de contenidos

Abstract

Abstract in English There are numerous methods for solving partial differential equations, one of which is the use of machine learning techniques. This document introduces the use of neural networks, taking financial option valuation as a case of study. The mentioned problem is approached by solving the differential equations of the underlying economic model. This work covers the necessary concepts to understand the mathematics behind the model and the formulation of the problem. It also presents the classical methods used in its resolution, describes the framework of neural networks from a mathematical point of view, and provides an implementation of the method.

Resumen

Existen multitud de métodos de resolución de ecuaciones diferenciales en derivadas parciales, uno de ellos es el uso de técnicas de aprendizaje automático. En este documento se introduce dicha metodología, utilizando redes neuronales, y tomando como caso de estudio la valoración de derivados financieros. El citado problema de matemática financiera se aborda resolviendo las ecuaciones diferenciales del modelo económico subyacente. Este trabajo recoge los conceptos necesarios para llevar a cabo el desarrollo matemático del modelo y plantear el problema. Además, se exponen los métodos clásicos utilizados en su resolución, se describe matemáticamente el funcionamiento de las redes neuronales y se da una implementación de la metodología.

1. Introducción

La resolución de ecuaciones en derivadas parciales (EDPs) con alta dimensionalidad es un problema de gran relevancia y con una limitación práctica notablemente restrictiva: es extremadamente demandante a nivel computacional. Este tipo de problemas aparece en diversos campos, siendo uno de ellos el de la matemática financiera, donde es de capital importancia, ya que la resolución numérica de este tipo de ecuaciones permite valorar distintos tipos de productos financieros.

El mercado bursátil es, a día de hoy e históricamente, un foco de inversión de gran relevancia para los grandes capitales. Es por ello que el manejo de los datos de que se dispone para optimizar las inversiones suscita un interés abrumador. Consecuentemente, economistas y matemáticos han desarrollado modelos cada vez más precisos a lo largo de los años que permiten describir la evolución de las cotizaciones, es decir, cómo varía el precio de las acciones.

En particular, entre las negociaciones bursátiles, el mercado de derivados financieros es en el que circula más volumen de capital. Un derivado financiero es un producto financiero que depende del valor de otro activo. Esto quiere decir que los derivados bursátiles son productos que fundamentan su valor en cotizaciones de otros activos.

Por otro lado, durante los últimos años se ha asistido a una expansión acelerada del uso de modelos de aprendizaje automático o *machine learning* (ML). La citada expansión ha sido capitaneada por las célebres redes neuronales como máximas representantes de la inteligencia artificial (AI) en el imaginario colectivo, como consecuencia de los avances debidos a las GPUs (unidades de procesamiento gráfico) y otras formas de computación de altas prestaciones (HPC). Nótese el uso diferenciado de los términos *machine learning* e inteligencia artificial. Dado que existe cierta ambigüedad en la literatura en la utilización de estos términos, se esbozará su significado más adelante para no dar lugar a confusiones.

Continuando con el contexto del aprendizaje automático en el entorno de las matemáticas, Leo Breiman ya pugnaba en 2001 por un uso más generalizado en estadística de técnicas que él denominaba algorítmicas. En [6], argumentaba que la asunción de una distribución estadística subyacente a los fenómenos de estudio daba

al traste con la capacidad predictiva de los modelos en buena parte de los casos. Abogaba por un uso incremental de técnicas tales como los árboles de decisión o las redes neuronales, que no presuponen características del conjunto de datos. Estos modelos, clasificables como *machine learning*, copan las publicaciones más recientes en ciencia de datos dado que, pese a sus limitaciones, aportan un enfoque enriquecedor y proveen mejores resultados que los modelos clásicos en muchos casos.

Precisamente el objetivo de este Trabajo Fin de Máster es el estudio y aplicación del *machine learning* en la resolución de EDPs, particularmente en los problemas de valoración de derivados en matemática financiera.

Para poder entrar en materia, cabe definir los conceptos de aprendizaje automático e inteligencia artificial.

- Atendiendo a [37], Herbert Simon define aprendizaje como "cualquier proceso por el cual un sistema mejora su desempeño mediante la experiencia". Posteriormente, Tom Mitchell precisa en [27] que el **Aprendizaje Automático** o ***Machine Learning*** es el "estudio de algoritmos que mejoran su desempeño P en cierta tarea T mediante experiencia E", quedando definido un proceso de aprendizaje por la terna (P, T, E).

- En 1955, McCarthy, explicando cómo impulsar el desarrollo de la **Inteligencia Artificial** propone en [22] "conseguir que las máquinas utilicen el lenguaje, creen abstracciones y conceptos, resuelvan tipos de problema actualmente reservados a los humanos y mejoren por sí mismas". La definición implícita en la cita anterior ha evolucionado a lo largo del tiempo, prueba de ello es que en la primera década de este siglo, Russell y Norvig dan la siguiente definición en uno de los libros de referencia sobre inteligencia artificial [37]: "estudio de agentes que reciben estímulos del medio y actúan en consecuencia, cada uno de estos agentes se corresponde con una función que asocia estímulos a acciones". Finalmente, a nivel profesional en la mayor parte del mundo occidental, actualmente se acepta la definición adoptada por la IEEE Computer Society y la ACM en Computer Science Curricula [10]: "estudio de la resolución de problemas difíciles o poco prácticos de resolver mediante métodos tradicionales".

Atendiendo a las definiciones anteriores, las técnicas de aprendizaje automático son una forma de construir sistemas de inteligencia artificial.

Cabe señalar que el hecho de incluir varias definiciones, así como un esbozo histórico de su evolución, atiende al propósito de ilustrar las diferentes aproximaciones a cada concepto, así como permitir al lector una concepción más general de los mismos.

En relación a la definición de inteligencia artificial, en este documento se denominan métodos clásicos a los algoritmos utilizados tradicionalmente para resolver ciertos problemas. La palabra "tradicionalmente" no tiene ningún tipo de connotación cronológica, sino que se utiliza como contraposición a la proliferación de las técnicas de aprendizaje profundo en las últimas décadas. De hecho, algunos métodos clásicos, como los métodos numéricos sin malla para la resolución de EDPs, datan de la década de 1990 ([29], entre otros). En dicha década, la inteligencia artificial más afamada hasta la aparición de Chat-GPT: Deep Blue, vencía a Gary Kaspárov, convirtiéndose en el primer ordenador capaz de batir en una partida de ajedrez a un vigente campeón del mundo. Más aún, el diseño de redes neuronales similares a las actuales se remonta a 1975 (gracias a resultados publicados en [44]) y los árboles de decisión datan de los años 60, con importantes avances en las siguientes décadas, como el algoritmo ID3 presentado en [32] en 1979 o el algoritmo C4.5 de [33] en 1993.

El desarrollo de las redes neuronales profundas a lo largo de los últimos 20 años ha dado lugar a la proliferación de multitud de diseños y arquitecturas diferentes. Buena parte de estos modelos han constituido un foco de interés en la resolución de problemas de valoración de productos de inversión, dada su capacidad de realizar predicciones rápidamente.

Asociado al objetivo enunciado previamente, la resolución de EDPs para la valoración de derivados financieros mediante técnicas de inteligencia artificial, se han de llevar a cabo una serie de tareas:

1. Desarrollar el modelo matemático que describe el mercado y conduce a las EDPs que permiten determinar el precio de los derivados.

2. Exponer y desarrollar los resultados matemáticos que subyacen a las técnicas de IA aplicadas.

3. Estudiar el estado del arte y razonar sobre las ventajas de implementar técnicas de inteligencia artificial para atacar el problema.

4. Realizar una implementación correcta del modelo de IA elegido y de su ajuste (llamado entrenamiento). Para ello se debe seguir un proceso de selección del modelo en base a los resultados obtenidos, estimar hiperparámetros satisfactorios y extraer conclusiones sobre el método llevado a cabo.

Con el objetivo de exponer con organización y claridad las tareas enumeradas, esta memoria se estructura en cuatro Capítulos. El Capítulo 2 constituye el formalismo sobre el que se fundamenta la teoría financiera, que modela el problema tratado, así como la descripción matemática de los métodos empleados tradicionalmente en su resolución.

En el Capítulo 3 se describen las redes neuronales y el proceso de aprendizaje automático que siguen, se da una descripción de las mismas en términos matemáticos y se exponen los resultados que garantizan su correcto desempeño aproximando funciones. Además, se profundiza en un tipo particular de red, las Physics-Informed Neural Networks (PINN). El Capítulo concluye con un epígrafe dedicado a razonar sobre la aplicabilidad de otros modelos de aprendizaje automático, incluyendo una somera descripción de los mismos.

Tras estos Capítulos de corte más teórico, el Capítulo 4 se centra en la implementación y análisis de un modelo PINN. Tras una introducción en la que se realiza una revisión del estado del arte, se incluye una Sección sobre el porqué del uso de estos modelos. En ella se realiza una comparación con los modelos clásicos descritos en el Capítulo 2, exponiendo los beneficios y desventajas que presenta cada uno. Huelga decir que, el hecho de que la revisión del estado del arte se encuentre en un punto tan avanzado del documento atiende a la necesidad de exponer previamente los conceptos necesarios para su comprensión. El Capítulo continúa con una modesta aplicación de las PINN descrita en una Sección donde se recogen los pormenores de la implementación así como los resultados obtenidos. Para finalizar, se incluye un epígrafe donde se exponen las conclusiones alcanzadas así como posibles líneas de trabajo futuras.

El Capítulo 5 recoge las conclusiones del trabajo, centrándose en la parte de implementación. Asimismo, se mencionan posibles mejoras del modelo y se plantean posibles líneas de trabajo futuro.

Además, dado que este documento trata temas relativos a ámbitos tan dispares como son las finanzas o la inteligencia artificial, se ha considerado conveniente incluir un Índice de Contenidos además de la usual Índice de Figuras , previamente a la Bibliografía.

2. Marco teórico

2.1. Matemática financiera

2.1.1. Planteamiento y motivación

A continuación se propone un ejemplo que permite una visión más concreta del tipo de problema a tratar e introduce el contexto financiero.

Alicia está dispuesta a comprar dentro de dos meses una acción que hoy vale 5€, pero no está dispuesta a pagar más de 7€ por ella en el futuro. En términos económicos se dice que Alicia quiere cubrir su riesgo. Bruno está dispuesto a asumir ese riesgo a cambio de recibir una compensación económica a día de hoy, llegándose a la siguiente situación:

Hoy la acción vale 5€, Alicia paga hoy una cantidad C a Bruno. Dentro de dos meses pueden ocurrir dos escenarios:

- *El precio de la acción es superior a 7€. En este caso, Bruno pagará a Alicia la diferencia entre el precio de la acción y los 7€ estipulados. De esta forma Alicia puede comprar la acción al precio que quería puesto que la cantidad restante la está pagando Bruno.*
- *El precio de la acción es menor o igual a 7€. En este caso Alicia puede comprar la acción al precio que deseaba y Bruno no tiene que incurrir en ningún gasto.*

¿Cuál es la cantidad C que debe pagar Alicia a Bruno a día de hoy para que el trato sea justo?

El contrato descrito es un derivado financiero que recibe el nombre de Call Europea y será tratado en esta Sección. Asimismo, se describirá qué significa que el trato sea justo desarrollando el formalismo matemático del problema.

2.1.2. Conceptos de matemáticas financieras

Las siguientes definiciones y conceptos se han obtenido de [45].

Un mercado es un entorno (físico o abstracto) en el que un conjunto de jugadores, o **agentes económicos**, pueden realizar intercambios. Partiendo de esta base, un mercado financiero es todo mercado en el cual se intercambian activos y productos financieros a cambio de dinero.

Definición 2.1 (Dinero). El dinero es todo activo o bien aceptado por los agentes económicos como medio de pago o como unidad de valor. Además ha de ser

1. Unidad de cuenta: Describe precios de bienes y servicios.
2. Medio de pago: Es aceptado por todos los agentes en los intercambios, es decir, para comprar y vender bienes y servicios.
3. Depósito de valor: Conserva su valor en el tiempo.

En el mercado financiero el concepto de valor no es único y se pueden dar distintas nociones del mismo, aduciendo, por ejemplo, a los resultados de un peritaje o a una cierta función de utilidad. Es más, el valor de un activo o de un derivado puede ser diferente para varios agentes. Afortunadamente existe un concepto, que en ocasiones se confunde con el de valor en la vida cotidiana, pero que es independiente del agente y no depende de interpretaciones: el precio.

Definición 2.2 (Precio). El precio es la cantidad máxima de dinero que algún agente del mercado está dispuesto a pagar por un bien o servicio.

En aras de simplificar la terminología y facilitar la comprensión, en este documento se asume que el valor de los productos se corresponde con su precio.

Los conceptos introducidos a partir de ahora se limitan al mercado financiero donde, como se ha introducido previamente, se opera sobre activos y derivados intercambiándolos por dinero.

Un activo es un bien o servicio que tiene valor o puede generar ingresos. Simplificando, se puede asumir que es cualquier recurso capaz de suministrar dinero a su poseedor.

Definición 2.3 (Activo financiero). Un activo financiero es un título que otorga al inversor el derecho a recibir un ingreso futuro procedente de la entidad emisora.

En lo referente a este TFM se puede asumir que un activo financiero es un producto con un precio en el mercado y que puede generar una rentabilidad económica.

Los mercados financieros se suelen dividir en función de los productos que se negocian, tales como divisas, materias primas, acciones o bonos.

Definición 2.4 (Acción). Una acción representa una parte alícuota del capital social de una empresa. El precio de una acción se conoce como **cotización**.

Matemáticamente, una acción representa una fracción de la titularidad de la empresa. Aunque existen distintos tipos en función de los derechos que otorgan, se asume que todas las acciones son iguales y representan la misma proporción de la empresa.

Definición 2.5 (Retorno). Se denomina retorno de una acción a la variación relativa de su cotización en un periodo de tiempo específico.

Definición 2.6 (Dividendo). Se denomina dividendo a la parte de los beneficios generados por una empresa que se reparte a cada accionista (poseedor de al menos una acción).

Definición 2.7 (Interés). Se denominan intereses a:

- El rendimiento obtenido por la cesión de un capital durante un periodo de tiempo concreto.
- Respectivamente, la contraprestación pagada por la disposición de un capital ajeno durante un plazo determinado.

Entendiéndose que el rendimiento (o contraprestación) son las ganancias (o pérdidas) en términos relativos. Más generalmente se habla de **rentabilidad**, englobando los conceptos de rendimiento y ganancias.

Cabe reseñar que el tipo de interés r se debe expresar utilizando las unidades de tiempo en las cuales se esté trabajando. En este documento, salvo que se especifique lo contrario, los intereses se asumen continuos, es decir, la evolución temporal de un capital inicial A_0 a un tipo de interés r viene descrita por la expresión

$$A(t) = A_0 e^{(1+r)t}. \tag{2.1}$$

Definición 2.8 (Bono). Un bono es un título que representa el derecho a percibir un flujo de pagos periódicos en un futuro a cambio de abonar una cierta cantidad de dinero en el momento de su adquisición.

Los bonos se pueden entender como préstamos que hace el inversor a la entidad emisora. Estos préstamos generarán ingresos futuros a modo de interés. Los bonos más conocidos son las Letras del Tesoro, emitidas por gobiernos.

Definición 2.9 (Descuento de flujos). Se denomina descuento de flujos al proceso de obtención del **valor presente** de un flujo futuro en función de un tipo de interés r.

Financieramente, si $A(t) = A_0 e^{(1+r)t}$, entonces A_0 unidades monetarias hoy son equivalentes a $A(t)$ unidades monetarias en el instante t.

Definición 2.10 (Activo libre de riesgo). Se considera activo libre de riesgo aquel cuya rentabilidad se conoce de antemano. La rentabilidad de un activo libre de riesgo se conoce como **tasa libre de riesgo** o tipo de interés libre de riesgo y se suele representar por r.

El concepto de activo libre de riesgo modela de manera bastante precisa el comportamiento de los bonos cuyo emisor garantiza la seguridad de abonar el importe correspondiente.

En contraposición a los activos libres de riesgo se encuentran los activos con riesgo, que son aquellos de los que se puede conocer su valor en el presente pero se desconoce su evolución futura. Las acciones son un buen ejemplo de este tipo de activo.

Definición 2.11 (Derivado financiero). Un derivado financiero es un producto del mercado financiero cuyo valor depende de la evolución del precio de un activo (o varios) denominado **activo subyacente**.

El derivado financiero en el que se profundiza en este documento es la Call.

Definición 2.12 (Call). Una Call u Opción de Compra es un contrato negociado en el instante t_0 que otorga:

- al comprador el derecho de adquirir el activo S en el instante $T \geq t_0$ al precio K a cambio del pago de una prima $C(t_0)$.
- al vendedor la obligación de vender el activo S en el instante $T \geq t_0$ al precio K a cambio de recibir una prima $C(t_0)$.

En función de cuándo se puede ejercer el derecho del comprador se diferencian dos tipos de Call:

- Call Europea. Se puede ejercer el derecho a compra únicamente en T, el vencimiento del plazo.
- Call Americana. Se puede ejercer el derecho a compra en cualquier instante hasta el vencimiento, es decir, en el intervalo $[t_0, T]$.

En términos prácticos se entiende que el comprador de la Call ha de pagar una cantidad $C(t_0)$ en el instante de la adquisición del derivado y recibirá, si ejerce el derecho, el valor máx$\{S(t) - K, 0\}$ cuando decida (y pueda) ejercerlo.

Nótese que la adquisición de la Call otorga un derecho y no una obligación. Asumiendo racionalidad en los agentes, el derecho no se ejerce si el precio del activo es inferior al **strike** K, ya que es más barato comprar el activo directamente en el mercado. En el caso de que $S(t) > K$ siempre se ejerce ya que el retorno es positivo.

2.1.3. Conceptos y resultados matemáticos

El formalismo matemático necesario para el tratamiento del problema de matemática financiera tratado en este documento se enmarca en la teoría sobre procesos estocásticos. En cuanto a la bibliografía, se toma como referencia [17].

Definición 2.13 (σ-álgebra). Sea Ω un conjunto no vacío. Se dice que $\mathscr{F}$ es una σ-álgebra si satisface:

1. $\Omega \subset \mathscr{F}$.
2. $A \in \mathscr{F} \Rightarrow A^C \in \mathscr{F}$ ($\mathscr{F}$ es cerrado bajo complementarios).
3. $\{A_n\}_{n\in\mathbb{N}} \subset \mathscr{F} \Rightarrow \bigcup_{n\in\mathbb{N}} A_n \in \mathscr{F}$ ($\mathscr{F}$ es cerrado bajo uniones numerables).

Definición 2.14 (Proceso estocástico). Un proceso estocástico es una colección de variables aleatorias $X = \{X_t\}_{t\in\mathbb{T}}$ definidas en un mismo espacio probabilístico $(\Omega, \mathscr{F}, P)$, donde Ω es el espacio muestral y P (normalmente referida como **probabilidad física**) es una medida de probabilidad definida sobre la σ-álgebra $\mathscr{F} \subset \mathcal{P}(\Omega)$. En función del conjunto $\mathbb{T}$ se tiene que

- Si $\mathbb{T} = \{0, 1, \ldots, n\}$ o $\mathbb{T} = \mathbb{N}$, X se denomina proceso en tiempo discreto.
- Si $\mathbb{T} = [0, T]$ o $\mathbb{T} = [0, \infty)$, X se denomina proceso en tiempo continuo.

En la literatura es habitual la notación $X(t)$ para referirse a las variables aleatorias X_t de un proceso, de igual forma que $X(t, \omega)$. El uso de cada una dependerá del contexto y la intención del autor en cada caso. En este TFM se trabaja con modelos en tiempo continuo.

Definición 2.15 (Trayectoria). Dado un proceso estocástico, para ω fijo, la función $t \mapsto X(t, \omega)$ se denomina trayectoria del proceso.

Definición 2.16 (σ-álgebra de Borel). Sea U un espacio topológico. Se denomina σ-álgebra de Borel a la menor σ-álgebra que contiene todos los abiertos del espacio topológico U . Se denota por $\mathscr{B}(U)$.

Definición 2.17 (Proceso medible). Se dice que un proceso estocástico $X : \mathbb{T} \times \Omega \to \mathbb{R}^d$ es $\mathscr{F}$-medible si, para cada A de la σ-álgebra de Borel $\mathscr{B}(\mathbb{R}^d)$, el conjunto $\{(t, \omega) : X_t(\omega) \in A\}$ pertenece a la σ-álgebra producto $\beta = \mathscr{B}(\mathbb{T}) \otimes \mathscr{F}$.

Proposición 2.18. Toda trayectoria de un proceso estocástico medible es una función Borel-medible de $t \in [0, \infty)$.

Demostración. Es consecuencia inmediata de la Definición 2.17 y el Teorema de Fubini. □

La Proposición 2.18 permite interpretar el concepto de proceso de manera intuitiva. Se puede entender como una variable aleatoria con valores en el espacio de funciones de $\mathbb{T}$ en $\mathbb{R}$.

Definición 2.19 (Filtración). Se denomina filtración a toda colección de σ-álgebras $\{\mathscr{F}_t\}_{t\in\mathbb{T}} \subset \mathscr{F}$ tales que $\mathscr{F}_{t_1} \subset \mathscr{F}_{t_2}$, para todo $t_1, t_2 \in \mathbb{T}, t_1 \leq t_2$.

Si se desea dar una interpretación al concepto anterior, una filtración modela la información disponible hasta un instante dado. Es, por tanto, un concepto fundamental tanto a nivel teórico como a nivel de aplicaciones.

Definición 2.20 (Proceso de Wiener). Un proceso estocástico W se denomina P-proceso de Wiener (con respecto a la medida P) o movimiento Browniano estándar si verifica:

1. $W(0) = 0$.
2. Para $s < t$, $W(t) - W(s)$ tiene distribución normal, centrada, con varianza $t - s$.
3. W_t tiene incrementos independientes, es decir, si $s < t$, $W_t - W_s$ es independiente de $\mathscr{F}_s$.
4. W tiene trayectoria continua.

La última condición de la Definición anterior no es imprescindible, pues dado un proceso que satisfaga las tres primeras se puede demostrar la existencia de un proceso modificado que verifica la cuarta.

Definición 2.21 (Proceso adaptado). Se dice que un proceso X está adaptado a la filtración $\{\mathscr{F}_t\}_{t\in\mathbb{T}}$ si, para cada $t \geq 0$, X_t es una variable aleatoria $\mathscr{F}_t$-medible.

Definición 2.22 (Filtración natural). Dado un proceso estocástico X, $\mathscr{F}_t^X \subset \mathscr{F}$ denota $\mathscr{F}_t^X = \sigma\{X(s) : 0 \leq s \leq t\}$ a la filtración natural, es decir la σ-álgebra generada por $\{X(s) : 0 \leq s \leq t\}$.

Por otro lado, si el valor de una variable aleatoria Z puede determinarse completamente con las observaciones de la trayectoria $\{X(s) : 0 \leq s \leq t\}$ se denota por $Z \in \mathscr{F}_t^X$.

Ahora bien, sea X un proceso estocástico. Si Y es otro proceso estocástico tal que $Y(t) \in \mathscr{F}_t^X, \forall t \in \mathbb{T}$, se dice que Y está adaptado a la filtración $\{\mathscr{F}_t^X\}_{t\in\mathbb{T}}$. En particular, todo proceso estocástico X está adaptado a $\{\mathscr{F}_t^X\}$.

Definición 2.23. Se dice que un proceso X pertenece a la clase $\mathscr{L}^2[a, b]$ si se satisface simultáneamente

1. $\int_a^b E\left[X^2(s)\right] ds < \infty$.
2. X está adaptado a la filtración $\{\mathscr{F}_t^W\}_{t\in\mathbb{T}}$, con W un proceso de Wiener.

Asimismo, se dice que X es de clase $\mathscr{L}^2$ si $X \in \mathscr{L}^2[0,t]$ para todo $t > 0$.

La definición anterior constituye el primer paso para desarrollar una integral conocida como integral de Itô, que da sentido a la expresión $\int_a^b X(s)dW(s)$.

Definición 2.24 (Proceso simple). Sea $X \in \mathscr{L}^2[a,b]$ un proceso estocástico. Se dice que X es simple si existen puntos $\{t_k\}_{k=0}^n \in \mathbb{T}$ con $a = t_0 < t_1 < \ldots < t_n = b$ tales que $X(s_1) = X(s_2), \forall s_1, s_2 \in [t_k, t_{k+1})$.

Para procesos simples se puede dar una noción de integral en base a una suma finita

$$\int_a^b X(s)dW(s) = \sum_{k=0}^{n-1} X(t_k)\left[W(t_{k+1}) - W(t_k)\right], \tag{2.2}$$

donde $W(s)$ es un proceso de Wiener.

Así pues, se puede intuir que los procesos simples juegan en la integral de Itô un papel equivalente a las funciones escalonadas en la integral de Lebesgue. A continuación se describe dicho paralelismo, cristalizando en una noción de integral para procesos. Se puede encontrar una formalización rigurosa de los resultados necesarios para desarrollar la integral de Itô en [17] donde, de hecho, se tratan clases más generales de procesos.

De manera análoga a la integral de Lebesgue, se busca aproximar los procesos estocásticos mediante procesos simples. En particular, para todo proceso $X \in \mathscr{L}^2[a,b]$, existe una sucesión de procesos simples $\{X_n\}_{n=0}^\infty$ tales que

$$\int_a^b E\left[\left(X_n(s) - X(s)\right)^2\right] ds \to 0. \tag{2.3}$$

Para todo n, $\int_a^b X_n(s)dW(s)$ es una variable aleatoria Z_n bien definida según la ecuación (2.2). Ahora bien, se puede demostrar que existe una variable aleatoria Z tal que $Z_n \to Z$ cuando $n \to \infty$.

Probadas las anteriores asunciones, se está en condiciones de enunciar la siguiente definición.

Definición 2.25 (Integral de Itô). Dado $X \in \mathscr{L}^2[a,b]$, la integral de Itô de X en $[a,b]$ es el límite

$$\int_a^b X(s)dW(s) = \lim_{n\to\infty} \int_a^b X_n(s)dW(s), \tag{2.4}$$

donde $\{X_n\}_{n=0}^\infty$ es una sucesión de procesos simples que satisface la ecuación (2.3).

Se concluye este epígrafe sobre procesos estocásticos con un teorema y dos definiciones necesarios para formalizar conceptos económicos que se desarrollan más adelante en este documento. La demostración del teorema se puede encontrar en el Capítulo 11 de [3].

Definición 2.26 (Medida equivalente). Sean μ, ν dos medidas en el espacio $(\Omega, \mathscr{F})$. Considérense los conjuntos

$$\mathcal{N}_\mu = \{A \in \mathscr{F} : \mu(A) = 0\}, \quad \mathcal{N}_\nu = \{A \in \mathscr{F} : \nu(A) = 0\}. \tag{2.5}$$

Se dice que μ y ν son medidas equivalentes si y solo si $\mathcal{N}_\mu = \mathcal{N}_\nu$.

Teorema 2.27 (de Girsanov). Sean P una medida de probabilidad, W^P un P-proceso de Wiener y ϕ un proceso adaptado a $\{\mathscr{F}_t^W\}_{t\geq 0}$, tal que $E^P\left[e^{\frac{1}{2}\int_0^T ||\phi(s)||^2 ds}\right] < \infty, \forall T > 0$. La medida de probabilidad Q en $\mathscr{F}_T$ definida para cada $T > 0$ por $dQ = L_T dP$, con

$$L_T = \frac{1}{A_T} e^{\int_0^T \phi(s) dW(s) - \frac{1}{2}\int_0^T ||\phi(s)||^2 ds} \quad \text{y} \quad E^P[L_T] = 1, \tag{2.6}$$

donde A_T es una constante, es una medida equivalente a P y se verifica que

$$dW^Q = \phi dt + dW^P, \tag{2.7}$$

donde W^Q es un Q-proceso de Wiener.

Definición 2.28 (Proceso de Itô). Se dice que un proceso $\{X(t)\}_{t\geq 0}$ es de Itô si admite una representación de la forma

$$X(t) = \int_0^t K(s) ds + \int_0^t H(s) dW(s), \tag{2.8}$$

donde $K, H \in \mathscr{L}^2[0, t], \forall t \geq 0$.

Definición 2.29 (Martingala). Dada una filtración $\{\mathscr{F}_t\}_{t\in\mathbb{T}}$, un proceso estocástico X se denomina una $\{\mathscr{F}_t\}$-martingala (o martingala para la filtración $\{\mathscr{F}_t\}$) si se verifican las siguientes propiedades:

1. X está adaptado a la filtración $\{\mathscr{F}_t\}_{t\in\mathbb{T}}$
2. $E\left[\,|X(t)|\,\right] < \infty$ para todo $t \in \mathbb{T}$.
3. $E\left[X(t)|\mathscr{F}_s\right] = X(s)$, para todos t y s tales que $s \leq t$.

De la primera condición se interpreta que es posible observar el valor $X(t)$ en el momento t, la segunda significa que la esperanza del proceso en cualquier instante de tiempo es finita y la tercera garantiza que asumiendo la información disponible hasta un cierto instante de tiempo, la esperanza del proceso en un futuro es el valor del proceso en dicho instante.

2.1.4. Modelización del mercado de valores

Para poder realizar el desarrollo matemático de un modelo descriptivo del comportamiento de los mercados se asumen ciertas hipótesis sobre los mismos:

1. Existe un activo libre de riesgo (que es único) del cual se pueden comprar, mantener o vender cantidades arbitrarias.
2. El tipo de interés a corto plazo del activo de libre de riesgo r es una constante determinista de dominio público.
3. No se incurre en costes de operación. Es decir, no existen costes por comprar o vender acciones ni derivados.
4. Es posible la venta a corto sin penalizaciones. Esto significa que para alguien que no esté en posesión de un activo existe la posibilidad de aceptar el importe actual de un comprador so compromiso de devolver en una fecha estipulada el valor del activo en dicho momento.
5. La acción no paga dividendos.
6. El mercado es **eficiente**: todos los jugadores disponen de la misma información en cualquier momento.

Siendo rigurosos, no todas las hipótesis son realistas. Por ejemplo, la tercera hipótesis no es cierta en la práctica puesto que existen corretajes por comprar y vender acciones. Aún así, asumir esta hipótesis simplifica enormemente los desarrollos matemáticos más adelante. En cualquier caso, conviene señalar que existen modelos que incluyen los costes de operación (o no asumen alguna de las otras hipótesis), pero su desarrollo es más complejo.

La existencia de la **venta en corto** satisface la necesidad de un inversor que quiera realizar el proceso inverso a la compra-venta, vendiendo hoy el activo que adquirirá en un futuro. Esto puede interesar, por ejemplo, cuando una acción cotiza a la baja, pues el inversor que venda a corto ingresa el precio la acción antes de que disminuya su valor y abona la compra en un momento en el que el precio es menor. En contraposición a la posición corta se define la **posición larga** como la estrategia de compra de un activo para su venta en un futuro.

Los siguientes resultados y definiciones son generalizables a mercados con varios activos, pero por claridad en la exposición se presentan para un mercado formado sólo por dos activos.

Definición 2.30 (Portfolio). Un portfolio h es una **cartera de inversiones** formada por un activo libre de riesgo B, cuya evolución $B(t)$ se conoce de manera

determinista, y una acción S, cuyo valor $S(t)$ tiene un comportamiento estocástico. Es decir $h(t) = (b(t), s(t))$, donde b y s son funciones que varían con el tiempo y representan las cantidades en la cartera de inversión de B y S respectivamente.

Definición 2.31 (Arbitraje). Se dice que un portfolio $h(t) = (b(t), s(t))$ es de arbitraje si su valor $V(t) = b(t)B(t) + s(t)S(t)$ satisface $V(0) = 0$ y, para algún $T > 0$, $V(T) \geq 0$ con probabilidad 1, siendo $P(V(T) > 0) > 0$.

Se dice que un mercado está libre de arbitraje si no se puede construir ningún portfolio de arbitraje.

Definición 2.32 (Derivado replicable). Se dice que un derivado negociado sobre los activos de un mercado es replicable si existe un portfolio formado por los activos de dicho mercado que replica el valor del derivado en todo instante de tiempo.

Definición 2.33 (Mercado completo). Se dice que un mercado es completo si todo derivado negociado sobre activos del mercado es replicable.

Dado un mercado formado por una serie de activos la estrategia, de forma resumida, consiste en ver si existe una medida de probabilidad, equivalente a la medida de probabilidad física del mercado, donde el proceso de los activos con riesgo sean martingalas. A partir de ahí, se analiza si se puede obtener una fórmula para valorar los derivados. El desarrollo completo de los resultados es bastante extenso y técnico. Ya que el TFM se centra más en resolver numéricamente la ecuación en derivadas parciales que rige un derivado, a continuación simplemente se presenta un breve resumen de los resultados más importantes. Para un desarrollo completo de los mismos, consultar [3].

Definición 2.34 (Medida libre de riesgo). Sea (Ω, P, F_t) la probabilidad física. Una medida de probabilidad Q en $\mathscr{F}_T$ se denomina medida libre de riesgo equivalente para el modelo del mercado en $[0, T]$ si satisface

1. Q es equivalente a P en $\mathscr{F}_T$.
2. El proceso de precios $S(t)$ descontado con $r(t)$ es una martingala para la medida Q en el intervalo $[0, T]$.

Teorema 2.35. El modelo del mercado está libre de arbitraje si existe una medida (local) libre de riesgo Q.

Teorema 2.36. Asumiendo ausencia de arbitraje, el modelo del mercado es completo si la medida libre de riesgo Q es única. Entonces, si un derivado es replicable, su valor se puede expresar como el valor actualizado, a día de hoy, del valor esperado futuro en la medida libre de riesgo de su portfolio replicante.

Modelo de Black-Scholes

El modelo de Black-Scholes (BS) fue propuesto por Fischer Black y Myron Scholes en [4], pero su primera aparición en un artículo académico se debe a Robert Merton en [24]. Como consecuencia de estas publicaciones, todos ellos recibieron el Premio Nobel de Economía en 1997.

Definición 2.37 (Black-Scholes). Se dice que el mercado sigue el modelo de Black-Scholes si la dinámica del activo libre de riesgo viene dada por

$$dB(t) = rB(t)dt, \tag{2.9}$$

y la de las acciones por

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t), \tag{2.10}$$

donde r, μ y σ son constantes. Al parámetro μ se le conoce como *drift rate* y σ como volatilidad.

El modelo de BS es uno de los más sencillos a la hora de describir la evolución de las cotizaciones a lo largo del tiempo. Buena parte de los modelos más complejos que refinan los resultados se desarrollan tomando BS como base y, no siendo tan restrictivos con las hipótesis, modificando las ecuaciones en consecuencia.

Teorema 2.38. El modelo de Black-Scholes está libre de arbitraje y la medida libre de riesgo, dada por el Teorema 2.27 con $\phi = \frac{\mu - r}{\sigma}$, es única, es decir, el modelo es completo. Más aún, la dinámica de las acciones viene dada en dicha medida por

$$dS(t) = rS(t)dt + \sigma S(t)dW^Q(t). \tag{2.11}$$

2.1.5. Cálculo de la Call Europea

Dado que el modelo de Black-Scholes es completo, la Call Europea tiene un portfolio replicante y, por el Teorema 2.36, su valor presente se puede expresar como el valor descontado del valor esperado del derivado en la medida libre en el vencimiento..

Teorema 2.39. El precio de la opción europea viene dado por

$$C(S_t) = e^{-r(T-t)} E^Q \left[\text{máx}\{S_T - K, 0\} \;\middle|\; \mathscr{F}_t^{W^Q} \right], \tag{2.12}$$

donde S_t sigue la dinámica dada por la ecuación (2.11).

La gran importancia del modelo de Black-Scholes viene de que la expresión anterior puede calcularse explícitamente.

Lema 2.40. Sea X una variable aleatoria. Si X sigue una normal de media μ y varianza σ^2, entonces se verifica que

$$E\left[\exp(aX) \mid X \geq k\right] = \exp\left(a\mu + \frac{a^2\sigma^2}{2}\right)\Phi(d), \tag{2.13}$$

con $d = (-k+\mu+a\sigma^2)/\sigma$ y donde a es una constante y $\Phi(\cdot)$ la función de distribución de $N(0,1)$.

Demostración. Puesto que $X \sim N(\mu, \sigma^2)$, se tiene que

$$E\left[\exp(aX)|X \geq k\right] = \frac{1}{\sigma\sqrt{2\pi}}\int_k^\infty \exp\left[aX - \frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2\right]dX. \tag{2.14}$$

Completando cuadrados se obtiene la expresión

$$aX - \frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2 = a\mu + \frac{a^2\sigma^2}{2} - \frac{1}{2}\left(\frac{X-(\mu+a\sigma^2)}{\sigma}\right)^2, \tag{2.15}$$

que se puede incorporar a la ecuación (2.14) y se tiene

$$E[\exp(aX)|X \geq k] = \frac{\exp\left(a\mu + \frac{a^2\sigma^2}{2}\right)}{\sigma\sqrt{2\pi}}\int_k^\infty \exp\left(-\frac{1}{2}\left[\frac{X-(\mu+a\sigma^2)}{\sigma}\right]^2\right)dX. \tag{2.16}$$

Considérese el cambio de variable $u = [X-(\mu+a\sigma^2)]/\sigma$ y $H = [k-(\mu+a\sigma^2)]/\sigma$. Entonces

$$\begin{aligned} E[\exp(aX)|X \geq k] &= \exp\left(a\mu + \frac{a^2\sigma^2}{2}\right)\frac{1}{\sqrt{2\pi}}\int_H^\infty \exp\left(-\frac{u^2}{2}\right)du \\ &= \exp\left(a\mu + \frac{a^2\sigma^2}{2}\right)[1 - N(H)] = \exp\left(a\mu + \frac{a^2\sigma^2}{2}\right)N(d), \end{aligned} \tag{2.17}$$

donde $d = -H = (-k+\mu+a\sigma^2)/\sigma$. □

Teorema 2.41. El precio de una call europea para cualquier instante t anterior al vencimiento viene dado por

$$C(S_t) = \Phi(d_1)S_t - \Phi(d_2)Ke^{-r(T-t)}, \tag{2.18}$$

con

$$d_1 = \frac{\ln \frac{S_t}{K} + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \quad ; \quad d_2 = \frac{\ln \frac{S_t}{K} + \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}},$$

donde

- $\Phi(\cdot)$ es la función de distribución de $N(0,1)$,
- T es el instante de tiempo del vencimiento,
- t es el instante de tiempo actual,
- S_t es el precio de la acción en el momento t,
- K es el strike,
- r es el tipo de interés libre de riesgo,
- σ es la volatilidad, que se asume constante, de los retornos de la acción.

Demostración. La opción se ejecuta si $S_T \geq K$, luego, siguiendo un razonamiento que se detalla a partir de la ecuación (2.37) en la Sección 2.2.2, esta condición se reescribe según la ecuación (2.40) como

$$S_T = S_t \exp\left[\left(r - \frac{\sigma^2}{2}\right)(T-t) + \sigma(W(T) - W(t))\right] \geq K, \tag{2.19}$$

equivalentemente,

$$(W(T) - W(t)) \geq \frac{1}{\sigma}\left[\ln\left(\frac{K}{S_t}\right) - \left(r - \frac{\sigma^2}{2}\right)(T-t)\right] =: k. \tag{2.20}$$

En caso de no ejecutarse, la opción vale 0, mientras que si lo hace su valor se corresponde con

$$e^{-r(T-t)} E\left[S_t \exp\left[\left(r - \frac{\sigma^2}{2}\right) + \sigma(W(T) - W(t))\right] - K \,\middle|\, W(T) - W(t) \geq k\right]. \tag{2.21}$$

Basta con observar que $(W(T) - W(t)) \sim N(0, T-t)$ y, por tanto, se puede aplicar el Lema 2.40 para concluir la demostración. □

Dado que se dispone de una expresión analítica que permite determinar el valor de la Call Europea en cualquier instante conociendo el valor de la acción, este modelo es idóneo para estudiar qué tal funcionan diversos métodos numéricos si se emplean para calcular el precio de la opción. Para ello, se verá cómo se puede

resolver numéricamente el problema de valoración de la opción con un método estadístico. Posteriormente, se verá cómo el problema se puede reformular como una EDP, permitiendo también el uso de otros métodos numéricos, entre ellos, el que se desea explorar a través del aprendizaje automático.

2.1.6. Limitaciones teóricas del modelo de Black-Scholes

El modelo de BS conlleva la ventaja fundamental de disponer de una fórmula explícita para el cálculo de opciones Europeas, pero conviene señalar sus carencias.

En primer lugar, aunque se pueda definir un precio para otros derivados a través de la esperanza matemática, no siempre se tiene una fórmula explícita. Por ejemplo, la opción Americana y otros derivados más complejos carecen de ella.

En segundo lugar, las hipótesis asumidas no se corresponden con la realidad. Las observaciones de los mercados indican que la volatilidad de los activos no es constante en el tiempo. Esto obliga a desarrollar modelos que reflejen las propiedades observadas en los mercados, como el desarrollado en [12], donde se asume que la volatilidad es, a su vez, otro proceso estocástico.

En el momento en el que se introducen más procesos (a través de ecuaciones estocásticas) para incorporar las propiedades observadas en la realidad, en general, se deja de tener fórmula explícita incluso para la Call Europea.

También se comentó anteriormente que existen costes de transacción en las operaciones financieras. Si se tienen en cuenta, esto puede llevar a modelos en cuyo desarrollo aparecen desigualdades variacionales, que deben resolverse con métodos numéricos.

A pesar de la gran relevancia que tiene el modelo de Black-Scholes, se han comentado estos inconvenientes para reflejar la necesidad de métodos numéricos que sean capaces de resolver las ecuaciones que aparecen en este campo.

2.2. Métodos clásicos de resolución de EDEs y EDPs

En esta Sección se verán algunos de los métodos habituales para obtener numéricamente los precios de los derivados.

En primer lugar se presentará un resultado conocido como el lema de Itô. Este resultado se empleará para ver cómo obtener el precio a través de simulaciones de Monte-Carlo con la Ecuación Diferencial Estocástica (EDE).

Asimismo, ese resultado puede emplearse para reformular el modelo como una Ecuación en Derivadas Parciales (EDP), habilitando el empleo de todos los métodos numéricos existentes para resolverlas.

2.2.1. Fórmula de Itô

Conviene indicar que la demostración de la fórmula de Itô exige un tratamiento exhaustivo y preciso de conceptos probabilísticos que escapa a los objetivos de este trabajo.

Por lo tanto, se ha optado por realizar una descripción general por las etapas esenciales del desarrollo de la fórmula de Itô. Para ello se ha seguido el esquema de [3] cotejando el contenido con [17], donde se puede encontrar un desarrollo matemático completo.

Para comenzar, se presenta la noción de diferencial estocástica, que emana de la Definición 2.28 de proceso de Itô. Sea X un proceso estocástico y asúmase que existe una constante real ϕ y a, b procesos adaptados (a la filtración $\{\mathscr{F}_t^X\}_{t\in\mathbb{T}}$) tales que se satisface

$$X(t) = \phi + \int_0^t a(X,s)ds + \int_0^t b(X,s)dW(s), \tag{2.22}$$

donde W es un proceso de Wiener. El concepto de **Ecuación Diferencial Estocástica** (**EDE**) surge de reescribir la expresión anterior de la siguiente manera:

$$dX(t) = a(X,t)dt + b(X,t)dW(t), \tag{2.23}$$
$$X(0) = \phi. \tag{2.24}$$

Se dice que X tiene diferencial estocástica dada por la ecuación (2.23) con condición inicial $X(0) = \phi$. Una de las ventajas que aporta la escritura en términos diferenciales es que dota al proceso de una interpretación intuitiva. $a(X,t)$ da cuenta de la deriva o drift que toma $X(t)$ en el instante t y $b(X,t)$ representa la magnitud del ruido Gaussiano que aleatoriza el proceso.

El siguiente paso consiste en demostrar la igualdad formal

$$[dW(t)]^2 = dt. \tag{2.25}$$

La idea intuitiva subyacente a la ecuación (2.25) es que, puesto que $\Delta W(t) = W(t) - W(s) \sim N(0, t-s)$ para $s < t$, se cumple $E\left[(\Delta W)^2\right] = \Delta t = t - s$ y $Var\left[(\Delta W)^2\right] = 2(\Delta t)^2$, luego cuando $\Delta t \to 0$ la varianza de $[\Delta W(t)]^2$ tiende a 0 más rápidamente. Es decir, ese término cuadrático es despreciable frente al otro y, en el paso al límite, $[dW(t)]^2$ se comporta de manera determinista. Se remite al Capítulo 3 de [17] para consultar una prueba rigurosa de la idea anterior.

Teorema 2.42 (Fórmula de Itô). Sea X un proceso con diferencial estocástica

$$dX(t) = a(X,t)dt + b(X,t)dW(t), \tag{2.26}$$

donde a y b son procesos adaptados, y sea $f : \mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$ una función de clase $C^{1,2}$. Entonces, el nuevo proceso $Z(t) = f(t, X(t))$ tiene diferencial estocástica dada por

$$dZ(t) = \left[\frac{\partial f}{\partial t} + a\frac{\partial f}{\partial x} + \frac{1}{2}b^2\frac{\partial^2 f}{\partial x^2}\right] dt + b\frac{\partial f}{\partial x}dW(t), \tag{2.27}$$

donde, por brevedad, se han omitido los argumentos de las funciones.

La ecuación (2.27) se conoce como fórmula de Itô. La idea subyacente a la prueba es demostrar que, para todo t, la siguiente igualdad se satisface con probabilidad uno:

$$f(t, X(t)) - f(0, X(0)) = \int_0^t \left(\frac{\partial f}{\partial t} + a\frac{\partial f}{\partial x} + \frac{1}{2}b^2\frac{\partial^2 f}{\partial x^2}\right) ds + \int_0^t b\frac{\partial f}{\partial x}dW(s). \tag{2.28}$$

La forma de proceder consiste en tomar n subdivisiones equidistantes del intervalo $[0, t]$ para poder escribir

$$f(t, X(t)) - f(0, X(0)) = \sum_{k=0}^{n-1} \left[f(t_{k+1}, X(t_{k+1})) - f(t_k, X(t_k))\right]. \tag{2.29}$$

Ahora bien, se puede aplicar la fórmula de Taylor a cada uno de los sumandos $f(t_{k+1}, X(t_{k+1}))$ en el punto $(t_k, X(t_k))$ para obtener

$$\begin{aligned} &f(t_{k+1}, X(t_{k+1})) - f(t_k, X(t_k)) \\ &\qquad = f_t(t_k, X(t_k))\Delta t + f_x(t_k, X(t_k))\Delta X_k + f_x x(t_k, X(t_k))(\Delta t)^2 + Q_k, \end{aligned} \tag{2.30}$$

donde $\Delta t = t_{k+1} - t_k$, Q_k es el resto y $\Delta X_k = X(t_{k+1}) - X(t_k)$, que se puede reescribir como

$$\begin{aligned} X(t_{k+1}) - X(t_k) &= \int_{t_k}^{t_{k+1}} a(X(s), s)ds + \int_{t_k}^{t_{k+1}} b(X(s), s)dW(s) \\ &= a(X(t_k), t_k)\Delta t + b(X(t_k), t_k)\Delta W_k + S_k, \end{aligned} \tag{2.31}$$

siendo S_k el resto de la aproximación. Además,

$$\begin{aligned} (\Delta X_k)^2 = {} & a^2(X(t_k), t_k)(\Delta t)^2 + b^2(X(t_k), t_k)(\Delta W_k)^2 \\ & + 2a(X(t_k), t_k)b(X(t_k), t_k)(\Delta t)(\Delta W_k) + P_k, \end{aligned} \tag{2.32}$$

donde P_k es el resto correspondiente a los términos que incluyen S_k al calcular $(\Delta X_k)^2$ mediante la ecuación (2.31).

Sustituyendo las expresiones (2.30), (2.31) y (2.32) en la ecuación (2.29) se obtiene

$$f(t,X(t)) - f(0,X(0)) = I_1 + I_2 + I_3 + \frac{1}{2}I_4 + \frac{1}{2}K_1 + K_2 + R, \tag{2.33}$$

donde, con un pequeño abuso de notación, se tiene que

$$\begin{aligned} I_1 &= \sum_k f_t\left(t_k\right)\Delta t, & I_2 &= \sum_k f_x\left(t_k\right)a\left(t_k\right)\Delta t \\ I_3 &= \sum_k f_x\left(t_k\right)b\left(t_k\right)\Delta W_k, & I_4 &= \sum_k f_{xx}\left(t_k\right)b^2\left(t_k\right)\left(\Delta W_k\right)^2 \\ K_1 &= \sum_k f_{xx}\left(t_k\right)a^2\left(t_k\right)\left(\Delta t\right)^2, & K_2 &= 2\sum_k f_{xx}\left(t_k\right)a\left(t_k\right)b\left(t_k\right)\Delta t\Delta W_k, \\ R &= \sum_k \left[Q_k + S_k + P_k\right]. \end{aligned} \tag{2.34}$$

La demostración se concluye probando la convergencia de cada uno de los términos anteriores hacia diversos valores. Valiéndose fundamentalmente de la definición de integral, cuando $n \to \infty$

$$I_1 \to \int_0^t f_t(s,X(s))ds,\ I_2 \to \int_0^t f_x(s,X(s))a(s)ds,\ I_3 \to \int_0^t f_x(s,X(s))b(s)dW(s). \tag{2.35}$$

Para demostrar que $I_4 \to \int_0^t f_{xx}(s,X(s))b^2(X(s),s)ds$ es necesario haber demostrado la ecuación (2.25) y, de nuevo, utilizar la definición de integral de Itô. Por último, no es excesivamente complicado probar que K_1 y K_2 tienden a 0, dejando la demostración de $R \to 0$ como último escollo de la prueba. Precisamente, la demostración de que la suma de los restos converge a 0 es la más laboriosa de todas.

Asumiendo que se han probado correctamente las convergencias de todos los sumandos, se llega a la expresión

$$f(t,X(t)) - f(0,X(0)) = \int_0^t \frac{\partial f}{\partial t}ds + \int_0^t a\frac{\partial f}{\partial x}ds + \int_0^t \frac{1}{2}b^2\frac{\partial^2 f}{\partial x^2}ds + \int_0^t b\frac{\partial f}{\partial x}dW(s), \tag{2.36}$$

tal y como se quería demostrar.

2.2.2. Simulaciones de Montecarlo (EDE)

El método de Montecarlo fue desarrollado en 1946 por John von Neumann y Stanislaw Ulam en el marco del Proyecto Manhattan [9]. Una de sus primeras aplicaciones consistió en determinar valores singulares de la ecuación de Schrödinger

[25]. Desde entonces, el método de Montecarlo ha tenido una relevancia indiscutible más allá de la física, permeando campos tan variopintos como son la generación de imágenes 3D, la matemática financiera o la estimación de parámetros estadísticos.

La idea subyacente a las simulaciones de Montecarlo consiste en generar un conjunto de posibles desarrollos de un evento sujeto a cierta distribución de probabilidad. Posteriormente se utiliza dicho conjunto para realizar inferencias estadísticas. En particular, en el problema de la Call Europea, las simulaciones de Montecarlo se utilizan para estimar el valor de la esperanza que aparece en la fórmula del Teorema 2.39. La fórmula tiene una interpretación sencilla, el valor de la Call es el valor trasladado al instante t del valor esperado de la Opción en el vencimiento.

Es decir, en el caso de las simulaciones de Montecarlo se manejan directamente los posibles valores que puede tomar el activo subyacente. Dado que se dispone del modelo de BS, que describe la evolución de los mismos, basta con realizar simulaciones teniendo en cuenta la aleatoriedad del proceso y comprobar cuál es el valor de la Call en cada caso. La media de todos los resultados se toma como estimador de la esperanza y se aplica la fórmula del Teorema 2.39. Se puede consultar una descripción detallada de las técnicas de simulación en [5] y [41].

Considérese de nuevo la ecuación (2.51):

$$dS(t) = rS(t)dt + \sigma S(t)dW(t).$$

Esta expresión se puede reagrupar de forma que todos los términos dependientes de S se encuentren a la izquierda, obteniéndose

$$\frac{dS(t)}{S(t)} = rdt + \sigma dW(t). \tag{2.37}$$

Ahora bien, tras reparar en que $\frac{dS}{S} = d\log S$, se puede aplicar la fórmula de Itô (Teorema 2.42) tomando $f(t,S) = \log S(t)$ y, consecuentemente

$$\frac{\partial f}{\partial t} = 0, \quad \frac{\partial f}{\partial S} = \frac{1}{S} \quad y \quad \frac{\partial^2 f}{\partial S^2} = -\frac{1}{S^2}. \tag{2.38}$$

Así pues, teniendo en cuenta que $a(S,t) = rS(t)$ y $b(S,t) = \sigma S(t)$, se obtiene

$$\begin{aligned} df &= \left(\frac{\partial f}{\partial t} + \mu S\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2}\right) dt + \sigma S \frac{\partial f}{\partial S} dW \\ &= \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma dW. \end{aligned} \tag{2.39}$$

Integrando la ecuación anterior se alcanza la siguiente expresión:

$$\log S(t) = \log S(0) + \left(\mu - \frac{\sigma^2}{2}\right) t + \sigma\left(W(t) - W(0)\right). \tag{2.40}$$

Cabe introducir un resultado que permite analizar una parte importante de las dinámicas descritas por las ecuaciones (2.51) y (2.39).

Proposición 2.43. Sea g un proceso adaptado a la filtración $\mathscr{F}_t^W$ que satisface

$$\int_a^b E\left[g^2(s)\right] ds < \infty. \tag{2.41}$$

Entonces se tiene que

$$E\left[\int_a^b g(s)dW(s)\right] = 0. \tag{2.42}$$

Demostración. En caso de que g sea un proceso simple, se tiene que

$$\begin{aligned} E\left[\int_a^b g(s)dW(s)\right] &= E\left[\sum_{k=0}^{n-1} g(t_k)(W(t_{k+1}) - W(t_k))\right] \\ &= \sum_{k=0}^{n-1} E\left[g(t_k)(W(t_{k+1}) - W(t_k))\right]. \end{aligned} \tag{2.43}$$

Ahora bien, dado que g está adaptado, el valor de $g(t_k)$ depende del proceso de Wiener únicamente en el intervalo $[0, t_k]$. Por otro lado, los incrementos de W son independientes, es decir, $W(t_{k+1}) - W(t_k)$ es independiente de $\mathscr{F}_{t_k}^W$ y, consecuentemente, $g(t_k)$ y $W(t_{k+1}) - W(t_k)$ son independientes. Por tanto,

$$\begin{aligned} E\left[\int_a^b g(s)dW(s)\right] &= \sum_{k=0}^{n-1} E\left[g(t_k)(W(t_{k+1}) - W(t_k))\right] \\ &= \sum_{k=0}^{n-1} E\left[g(t_k)\right] E\left[(W(t_{k+1}) - W(t_k))\right] = \sum_{k=0}^{n-1} E\left[g(t_k)\right] \cdot 0 = 0. \end{aligned} \tag{2.44}$$

En caso de que g no sea un proceso simple, dado que $\int_a^b E\left[g^2(s)\right] ds < \infty$, existe una sucesión (creciente) de procesos simples $\{g_n\}_{n\in\mathbb{N}}$ tales que

$$\int_a^b E\left[\left[g_n(s) - g(s)\right]^2\right] ds \xrightarrow{n\to\infty} 0. \tag{2.45}$$

y la sucesión $\int_a^b g_n(s)dW(s)$ es creciente. Ahora bien, para cada n, dicha integral $\int_a^b g_n(s)dW(s)$ es una variable aleatoria Z_n bien definida y se puede probar que

existe otra variable aleatoria Z tal que $Z_n \to Z$ en L^2 cuando $n \to \infty$. Por tanto, atendiendo a la definición de integral de Itô, se tiene que

$$E\left[\int_a^b g(s)dW(s)\right] = E\left[\lim_{n\to\infty}\int_a^b g_n(s)dW(s)\right]. \tag{2.46}$$

Aplicando el teorema de la convergencia monótona y teniendo en cuenta que los procesos g_n son simples,

$$E\left[\int_a^b g(s)dW(s)\right] = \lim_{n\to\infty} E\left[\int_a^b g_n(s)dW(s)\right] = 0. \tag{2.47}$$

□

La Proposición anterior formaliza la idea intuitiva de que los últimos términos de las ecuaciones (2.51) y (2.39) únicamente introducen aleatoriedad en la dinámica sin modificar su evolución media.

Ahora bien, en virtud de la Definición 2.20 de proceso de Wiener, se tiene que $W(t) - W(0) \sim N(0,t)$. Es decir, el logaritmo del precio de la acción sigue una distribución normal

$$\log S(t) \sim \mathrm{N}\left(\log S(0) + \left(\mu - \frac{\sigma^2}{2}\right)t, \quad \sigma^2 t\right). \tag{2.48}$$

Finalmente, exponenciando la ecuación (2.40) y tomando ϵ una variable aleatoria con distribución normal estándar, se tiene

$$S(t) = S(0)\exp\left\{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma\sqrt{t}\epsilon\right\} \tag{2.49}$$

Utilizando la fórmula anterior se puede muestrear una colección de variables aleatorias i.i.d. (independientes igualmente distribuidas) que determinen el valor del activo y, realizar inferencia estadística para determinar el de la Opción Europea utilizando el Teorema 2.39:

$$C(S_t) = e^{-r(T-t)}E^Q\left[\text{máx}\{S_T - K, 0\} \;\middle|\; \mathscr{F}_t^{W^Q}\right]. \tag{2.50}$$

2.2.3. Obtención de la EDP y métodos numéricos

Atendiendo a la Definición 2.37, la evolución de las acciones según el modelo de Black-Scholes viene descrita en la medida libre de riesgo por

$$dS(t) = rS(t)dt + \sigma S(t)dW(t). \tag{2.51}$$

Es decir, en el modelo de BS la variable aleatoria $S(t)$ sigue una dinámica tal que los procesos a y b considerados en la fórmula de Itô son de la forma

$$a(S,t) = rS(t) \quad y \quad b(S,t) = \sigma S(t), \tag{2.52}$$

donde $r, \sigma \in \mathbb{R}$.

Así pues, aplicando la fórmula, se obtiene

$$dC(t) = \left[\frac{\partial C}{\partial t} + rS(t)\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S(t)^2 \frac{\partial^2 C}{\partial S^2}\right] dt + \sigma S(t)\frac{\partial C}{\partial S} dW. \tag{2.53}$$

Considérese ahora una cartera de inversiones Π compuesta por la compra de una Opción Europea y la venta de una cantidad δ del activo subyacente. El valor del portfolio en un instante t viene dado por $\Pi(t,S) = C(t,S) - \delta(t)S$ y su dinámica está descrita por

$$d\Pi = dC - \delta dS = \left(\frac{\partial C}{\partial t} + \mu S\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - \mu S\delta\right) dt + \sigma S\left(\frac{\partial C}{\partial S} - \delta\right) dW. \tag{2.54}$$

Un vistazo a la ecuación anterior permite observar que en cada instante de tiempo existe una cantidad particular de acciones vendidas $\Delta = \frac{\partial C}{\partial S}$ que desliga el valor del portfolio de la aleatoriedad del mercado. Esta cartera de inversiones concreta se corresponde con una estrategia de cobertura conocida en finanzas como *Delta Hedging*.

Aplicando dicha estrategia, la ecuación (2.54) se reduce a

$$d\Pi = \left(\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2}\right) dt. \tag{2.55}$$

Teorema 2.44. Si Π es un portfolio completamente determinista en el modelo de BS, entonces Π evoluciona en el tiempo al tipo de interés libre de riesgo:

$$d\Pi = r\Pi dt. \tag{2.56}$$

Demostración. El Teorema 2.38 garantiza la existencia de una medida libre de riesgo en el modelo de BS, lo cual, según el Teorema 2.35, implica que el mercado está libre de arbitraje. La ausencia de arbitraje fuerza al valor del portfolio a evolucionar al tipo de interés libre de riesgo, lo cual se demuestra por reducción al absurdo. Asúmase que el la cartera de inversiones no evoluciona al tipo de interés libre de riesgo, entonces:

- Si el portfolio evoluciona a un ritmo menor que el activo libre de riesgo durante un cierto periodo de tiempo, es decir, si

$$\frac{d\Pi}{dt}(t, S(t)) < \frac{dB}{dt}(t), \ \forall t \in [t_1, t_2], \tag{2.57}$$

se puede establecer un portfolio de arbitraje. Sea $\Gamma = B\Delta - \Pi\Delta$ la cartera de inversiones en la que, en el instante inicial t_1, se compra una cantidad $B(t_1) > 0$ de B y se vende la misma cantidad $\Pi(t_1, S(t_1)) = B(t_1)$ de Π. Γ es un portfolio de arbitraje, porque se obtiene un beneficio estrictamente positivo

$$\frac{d\Gamma}{dt}(t) = \frac{dB}{dt}(t) - \frac{d\Pi}{dt}(t) > 0, \ \forall t \in [t_1, t_2]; \tag{2.58}$$

con riesgo nulo, pues tanto Π como B tienen una evolución determinista, y con un capital inicial nulo, pues $\Gamma(t_1) = B(t_1) - \Pi(t_1, S(t_1)) = 0$.

- Si el portfolio evoluciona a un ritmo mayor que el activo libre de riesgo durante un cierto periodo de tiempo, la cartera de inversión $\Gamma = \Pi\Delta - B\Delta$, que en el instante inicial t_1 satisface $\Pi(t_1, S(t_1)) > 0$, $B(t_1) = \Pi(t_1, S(t_1))$, es un portfolio de arbitraje. Esto se demuestra de manera completamente análoga al punto anterior.

Así pues se llega a una contradicción y se concluye que, necesariamente, el portfolio debe evolucionar al tipo de interés libre de riesgo en todo instante de tiempo $t \in \mathbb{R}^+$. Atendiendo a la ecuación (2.9) se concluye

$$d\Pi = r\Pi dt.$$

□

Retomando la ecuación (2.55), dado que la cartera de inversiones Π tiene una evolución completamente determinista, se ha de cumplir

$$r\Pi dt = \left(\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2}\right) dt. \tag{2.59}$$

Agrupando términos se obtiene que

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - r\Pi = 0, \tag{2.60}$$

y recordando que $\Pi = C - \frac{\partial C}{\partial S} S$, se alcanza la conocida como ecuación en derivadas parciales de Black-Scholes:

$$\frac{\partial C}{\partial t} + rS\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - rC = 0. \tag{2.61}$$

Para finalizar, bastaría imponer las condiciones frontera. Para la Call Europea, el precio en el vencimiento ($t = T$), para un strike K, si la acción toma un valor S viene dado por

$$C(S, T) = \text{máx}\{S - K, 0\}. \tag{2.62}$$

Además, para $S(0) = 0$ el precio de la acción nunca varía luego

$$C(0,t) = 0, \quad \forall t \in [0,T]. \tag{2.63}$$

Por último, utilizando argumentos económicos, para $S >> K$ se puede demostrar que

$$\lim_{S\to\infty} \frac{C(S,t)}{S} = 1. \tag{2.64}$$

La EDP se puede resolver analíticamente, llegando, evidentemente, a la misma solución obtenida en el Teorema 2.41.

Por otro lado, existen multitud de métodos numéricos para la resolución de EDP: diferencias finitas, métodos espectrales, el método de elementos finitos, etc.

Por ejemplo, el **Método de los Elementos Finitos (MEF)** es un método de resolución numérica de EDPs ampliamente utilizado, especialmente en problemas de ingeniería. Constituye una herramienta de gran versatilidad y eficacia cuyo origen se encuentra a mediados del siglo XX en problemas de la teoría de la elasticidad y la resistencia de materiales. Tal y como se describe en [16], el MEF se basa en la discretización del dominio de la EDP en subdominios más pequeños llamados elementos finitos.

Considérese la EDP de Black-Scholes expuesta en la ecuación (2.61), tras el cambio $\tau = T - t$

$$\frac{\partial C}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC. \tag{2.65}$$

Para aplicar el MEF se debe considerar la forma variacional de la ecuación anterior. Sea $\varphi(S)$ una función arbitraria que verifica

$$\varphi(0) = \varphi(S_R) = 0, \tag{2.66}$$

para cierto valor S_R suficientemente grande que sirve como cota superior de S. Multiplicando por $\varphi(S)$ a ambos lados de la ecuación (2.65) e integrando en el intervalo $[0, S_R]$ se obtiene, tras varias manipulaciones,

$$\int_0^{S_R} \frac{\partial C}{\partial \tau}\varphi dS = -\frac{1}{2}\sigma^2 \int_0^{S_R} S^2 \frac{\partial C}{\partial S}\frac{\partial \varphi}{\partial S} dS + (r - \sigma^2)\int_0^{S_R} S\frac{\partial C}{\partial S}\varphi dS - r\int_0^{S_R} C\varphi dS, \tag{2.67}$$

denominada expresión variacional de la ecuación localizada.

A continuación se aplica el método de Galerkin [38] con el objetivo de expresar la solución aproximada como combinación de funciones base $\{\varphi_j(S)\}$, atendiendo a la expresión

$$C_M(S,\tau) = \sum_{j=0}^{M} c_j(\tau)\varphi_j(S). \tag{2.68}$$

Por otro lado, tomando $0 = x_0 < x_1 < \ldots < x_M = S_R$ una partición del dominio, una elección habitual son las funciones $\varphi_j : [0, S_R] \to \mathbb{R}$ definidas por

$$\varphi_j(x) = \begin{cases} \frac{(x-x_{j-1})}{(x_j-x_{j-1})} & x_{j-1} \leq x < x_j \\ \frac{(x_{j+1}-x)}{(x_{j+1}-x_j)} & x_j \leq x < x_{j+1} \\ 0 & x \notin [x_{j-1}, x_{j+1}] \end{cases}, \quad \text{para } j = 0, \ldots, M, \tag{2.69}$$

que se conocen como funciones hat o funciones sombrero por su forma. El uso de estas funciones como funciones base conduce a la reescritura del sistema de manera matricial

$$M\frac{dc}{d\tau} = \left(-\frac{1}{2}\sigma^2 K + (r - \sigma^2)\Gamma - rM\right) v + b, \tag{2.70}$$

donde $c(\tau) = (c_1(\tau), \ldots, c_{M-1}(\tau))^{\mathrm{T}}$, $b = (b_1, \ldots, b_{M-1})^{\mathrm{T}}$ es el vector que recoge las condiciones frontera y las matrices $M = (M_{i,j})_{i,j=1}^{M-1}$, $K = (K_{i,j})_{i,j=1}^{M-1}$ y $\Gamma = (\Gamma_{i,j})_{i,j=1}^{M-1}$, conocidas respectivamente como matriz de masa, matriz de rigidez y matriz de convección, son de la forma

$$M_{ij} = \int_0^{S_R} \varphi_j \varphi_i dS, \quad K_{ij} = \int_0^{S_R} S^2 \frac{\partial \varphi_j}{\partial S}\frac{\partial \varphi_i}{\partial S} dS, \quad \Gamma_{ij} = \int_0^{S_R} S\frac{\partial \varphi_j}{\partial S}\varphi_i dS. \tag{2.71}$$

Imponiendo un integrador temporal, esta formulación matricial permite una resolución directa, conduciendo a la aproximación buscada.

Del método de los elementos finitos destacan su flexibilidad para manejar geometrías complejas y su capacidad para proporcionar soluciones precisas, por ejemplo, mediante la refinación adaptativa de la malla.

Otro posible enfoque para la resolución numérica de ecuaciones diferenciales son los **Métodos de Diferencias Finitas (MDF)**.

Son métodos con una vasta literatura, entre los que se encuentran las diferencias centrales o los métodos multigrid. Entre sus ventajas destacan su precisión y la sencillez de su implementación.

Por último, se considera oportuno mencionar los **métodos espectrales**, técnicas numéricas que se fundamentan en la aproximación de soluciones a partir de funciones conocidas y suficientemente regulares. Si bien tienen una dilatada tradición en dinámica de fluidos, actualmente se aplican como un método general de resolución de EDPs dando buenos resultados en numerosos campos.

La idea detrás de estos métodos es proyectar la solución sobre un espacio de funciones base definidas en todo el dominio, permitiendo una representación de gran precisión con un número relativamente bajo de términos. Este carácter global frente a las discretizaciones locales de los métodos anteriormente tratados en esta Sección aporta ventajas como una convergencia notablemente rápida, pero presenta problemas a la hora de tratar con discontinuidades o geometrías complejas (ver [11]).

2.2.4. Ventajas e inconvenientes de los métodos clásicos

Simulaciones de Montecarlo

Las simulaciones de Montecarlo tienen una ventaja fundamental: su versatilidad. No es necesario hacer grandes cambios en el método para adaptarlo a modelos diferentes u otro tipo de contratos derivados. Basta con actualizar el mecanismo de muestreo, adaptando las simulaciones de forma que reflejen correctamente la dinámica del modelo. Por otro lado, la implementación es muy sencilla en casi cualquier lenguaje y la mayoría de matemáticos, ingenieros e, incluso, economistas han utilizado simulaciones de Monte-Carlo en algún momento.

Si bien este método tiene ventajas claras y sólidas, también tiene un inconveniente principal con una importancia capital: reducir el error es muy costoso computacionalmente. Disponiendo de capacidades de cómputo medias los errores obtenidos son demasiado grandes. Reducir este error conlleva o bien mejoras de hardware, cálculo distribuido y computación paralela, o bien un aumento muy considerable del tamaño de las simulaciones.

Resolución de la EDP mediante métodos numéricos

Frente a la sencillez del método de Monte-Carlo, los métodos numéricos de las EDP, por regla general, requieren un desarrollo más elaborado y una cuidadosa implementación, aunque existen métodos preprogramados en los programas científicos.

La ventaja que ofrecen es que suelen ofrecer una convergencia más rápida y una manera más directa de acotar el error numérico cometido, frente al método de Monte-Carlo que requiere hacerlo con una estimación estadística.

La gran desventaja de estos métodos numéricos, sobre todo aplicados en el ámbito de las finanzas, es la conocida como maldición de la dimensionalidad. En problemas multidimensionales, los requerimientos computacionales para las discretizaciones de los métodos expuestos crecen de forma exponencial, pudiendo ser un serio inconveniente.

Viendo las ventajas e inconvenientes de los métodos más tradicionales empleados, parece interesante realizar un estudio exploratorio de técnicas disponibles en el marco del aprendizaje automático. En el próximo Capítulo se describe matemáticamente el proceso de aprendizaje de las redes neuronales, uno de los modelos de IA más empleados en la literatura.

3. Modelos de aprendizaje automático

3.1. Introducción

En la revisión del estado del arte, se encuentran numerosos modelos de inteligencia artificial aplicados a la regresión de conjuntos de datos, entre los que destacan las redes neuronales, los árboles de decisión y los métodos ensemble, como el bagging y los random forest [20]. En este Trabajo Fin de Máster, el problema abordado consiste en resolver ecuaciones diferenciales para las cuales no se dispone de un conjunto de puntos que permita realizar una regresión directa.

Este Capítulo tiene como objetivo introducir el concepto de red neuronal, proporcionando una explicación detallada de su funcionamiento. Además, se justifica el uso de redes neuronales para la aproximación de funciones, apoyándose en los resultados matemáticos pertinentes. Asimismo se incluye una discusión sobre los aspectos relevantes asociados al uso de este tipo de modelos de inteligencia artificial.

En particular se ha optado por explorar la aplicación de redes neuronales a la resolución de EDPs, puesto que es uno de los modelos de IA con mayor presencia entre las publicaciones, fundamentalmente, valiéndose del concepto de Physics Informed Neural Network (PINN) que se describirá más adelante. Existe una amplia literatura sobre el tema, tanto sobre la aproximación de funciones mediante redes neuronales [8], [26], llegándose a dar cotas de error en ciertos casos [46]; como sobre el uso de PINN para la resolución de ecuaciones diferenciales [23], [34].

Las referencias indicadas constan de una diferencia de más de veinte años entre sus fechas de publicación. Esto permite darse cuenta de que no es un ámbito de estudio nuevo ni un campo poco estudiado, sino que se trata de un campo consolidado, con unas bases definidas y multitud de publicaciones. Tanto es así que, de hecho, la aplicación de las PINN con diversos enfoques a modelos de matemática financiera, como el modelo de Black-Scholes, es notablemente prolífico. Se puede comprobar, por ejemplo en [36]. Esta revisión bibliográfica se centra, precisamente, en la valoración de precios de derivados utilizando redes neuronales y contiene más de cien

referencias. De esta forma, se quiere incidir en la profundidad del tema tratado en este documento, que no pretende sino ser una introducción en el campo.

Por último, se considera oportuno mencionar una aplicación particular de las redes neuronales. En [2] se aplica a las finanzas una heurística conocida como Monte-Carlo tree search, que utiliza aprendizaje reforzado para llevar a cabo simulaciones de Monte-Carlo en procesos de decisión multi-periodo. Esta heurística conduce a resultados que se utilizan como punto de partida para entrenar una NN, obteniéndose mejores soluciones del problema que cuando se aplica cualquiera de los dos métodos por separado.

3.2. Redes neuronales

3.2.1. Estructura

Para dar una descripción matemática de los conceptos básicos sobre redes neuronales se ha optado por combinar la estructura, notación y definiciones expuestos en el Capítulo 20 de [40] con los expuestos en [13].

Desde un punto de vista matemático, el concepto de Red Neuronal Artificial (ANN) o, simplemente, Red Neuronal (NN) se puede entender como un grafo dirigido y ponderado donde cada nodo se corresponde con una función. En particular, en este documento se consideran únicamente redes cuyo grafo asociado no contiene ningún ciclo. Este tipo de redes neuronales se denominan redes neuronales feedforward o hacia adelante, pero, por motivos de claridad en la redacción, se hará referencia a las mismas como redes neuronales.

Las redes que se van a tratar se pueden organizar en capas, así pues, se asume que un grafo $G = (V, E)$ asociado a una NN es dirigido, acíclico y además satisface que existen M subconjuntos disjuntos y no vacíos de V, denominados $V_m, m = 0, 1, \ldots, M$ que cumplen

1. $V = \bigcup_{m=0}^{M} V_m$.
2. Para todo $m < M$ y todo nodo v de V_m existe una arista en E que conecta v con un nodo de V_{m+1}.

Cada uno de los subconjuntos V_m se denomina capa. En particular, la capa V_0 se denomina capa de entrada, la capa V_M se denomina capa de salida y el resto de capas V_m con $n = 1, 2, \ldots, M-1$ se denominan capas ocultas.

Definición 3.1 (Función de activación). Sea $G = (V, E)$ un grafo asociado a una NN. Para cada nodo $v \in V \setminus V_0$ existe una función $\varphi_v : \mathbb{R} \to \mathbb{R}$ que se denomina función de activación de v.

Salvo que se indique lo contrario, todas las redes consideradas tienen asociada la misma función de activación a todos los nodos, es decir $\forall v, v' \in V \setminus V_0, \varphi_v = \varphi_{v'}$. Así pues, se hablará de función de activación de la red y se representará por φ.

Definición 3.2 (Función de pesos). Sea $G = (V, E)$ un grafo dirigido y acíclico. Se denomina función de pesos a la función $w : E \to \mathbb{R}$ que asocia un valor denominado peso a cada arista.

Definición 3.3 (Función de sesgos). Sea $G = (V, E)$ un grafo dirigido y acíclico. Se denomina función de sesgos a la función $b : V \setminus V_0 \to \mathbb{R}$ que asocia un valor denominado sesgo a cada nodo que no pertenece a la capa de entrada.

Asumidas estas definiciones previas, se puede definir de manera matemática una red neuronal:

Definición 3.4 (Red neuronal). Una red neuronal es una quintupla (V, E, φ, w, b) tal que $G = (V, E)$ es un grafo asociado a una NN, φ una función de activación, $w : E \to \mathbb{R}$ una función de pesos y $b : V \setminus V_0 \to \mathbb{R}$ una función de sesgos.

Las funciones w y b se consideran parámetros de la red, es decir, se pueden modificar para satisfacer ciertas necesidades tal y como se verá más adelante. La terna (V, E, φ) compuesta por el resto de elementos se denomina **arquitectura de red**.

El propósito de la estructura matemática introducida en la Definición 3.4 es describir un sistema de IA. Para poder describir completamente el proceso de aprendizaje aún es necesario introducir una serie de conceptos. Así pues, en el siguiente epígrafe se muestra cómo el proceso que sigue la información al atravesar una red neuronal se puede modelizar mediante una función.

3.2.2. Propagación hacia adelante

Considérese una red neuronal (V, E, φ, w, b). De ahora en adelante, para referirse a los n_m nodos de una capa V_m se hará uso de la notación $v_{m,i}$ con $i \in \{1, 2, \ldots, n_m\}$. Se dice que la red está completamente conectada si para todo $v_{m,i}$ con $0 \leq m \leq M, 1 \leq i \leq n_m$ se tiene que $\{(v_{m,i}, v_{m+1,1}), \ldots, (v_{m,i}, v_{m+1,n_{m+1}})\} \subset E$ (a partir de este punto todas las redes se consideran completamente conectadas).

En aras de la claridad, a continuación se introduce notación vectorial junto a una ligera simplificación visual. Para cada $m = 1, \ldots, M$:

- Se define la función

$$\begin{array}{rccc} \phi_m : & \mathbb{R}^{n_m} & \longrightarrow & \mathbb{R}^{n_m} \\ & (x_1, \ldots, x_{n_m})^{\mathrm{T}} & \longmapsto & (\varphi(x_1), \ldots, \varphi(x_{n_m}))^{\mathrm{T}} \end{array} \tag{3.1}$$

- Se adopta la notación $w^m_{i,j} = w(v_{m-1,i}, v_{m,j})$ con $1 \leq i \leq n_{m-1}, 1 \leq j \leq n_m$
- Se define la matriz $W^{[m]}$ de tamaño $n_m \times n_{m-1}$ como

$$\begin{bmatrix} w^m_{1,1} & \cdots & w^m_{n_{m-1},1} \\ \vdots & \ddots & \vdots \\ w^m_{1,n_m} & \cdots & w^m_{n_{m-1},n_m} \end{bmatrix}. \tag{3.2}$$

- Se define el vector $\beta^{[m]} = (b(v_{m,1}), \ldots, b(v_{m,n_m}))^{\mathrm{T}}$.

Se consideran **parámetros de la red** tanto los pesos como los sesgos de la misma, es decir, todos los valores $w^m_{i,j}$ y $b(v_{m,k})$ para $m = 1, 2, \ldots, M$, $1 \leq i \leq n_{m-1}$, $1 \leq j \leq n_m$ y $1 \leq k \leq n_m$. Se denota por θ al vector que recoge todos estos valores.

La capa de entrada de la red se encarga de recibir los datos, de forma que en cada nodo se introduce un número real. El vector que recoge dichos valores se denomina vector de entrada.

Esta información se transmite a los nodos de la siguiente capa multiplicando cada valor por el peso asociado a la arista correspondiente y, posteriormente, se suma el valor del sesgo asociado al nodo de la primera capa. De esta forma, cada nodo de la primera capa recibe un número real de cada nodo de la capa de entrada, suma todos ellos y aplica la función de activación al resultado. El proceso se repite en todas las capas ocultas hasta llegar a la capa de salida, donde se repite por última vez considerándose que los resultados obtenidos en estos nodos conforman el vector de salida.

Este proceso de cálculo se denomina propagación hacia adelante y se describe matemáticamente como sigue:

Definición 3.5. Sea (V, E, φ, w, b) una red neuronal y $x_0 \in \mathbb{R}^{n_0}$ el vector de entrada de la red. Para todo $m \in \{1, \ldots, M\}$ se define

$$x_m = \phi_m \left(W^{[m]} x_{m-1} + \beta^{[m]} \right) \in \mathbb{R}^{n_m}. \tag{3.3}$$

En particular, $x_M \in \mathbb{R}^{n_M}$ es el vector de salida de la red.

Definición 3.6 (Función de red). Sea (V, E, φ, w, b) una red neuronal. Se denomina función de red a la función F que asocia un vector de entrada $x_0 \in \mathbb{R}^{n_0}$ y unos parámetros θ con el vector de salida $x_M \in \mathbb{R}^{n_M}$ correspondiente:

$$F(x_0, \theta) = x_M, \tag{3.4}$$

donde para $m \in 1, ..., M$, $x_m = \phi_m \left(W^{[m]} x_{m-1} + \beta^{[m]} \right)$.

La definición anterior implica que se puede asociar el comportamiento de cualquier NN con una función. Por tanto, se puede definir un conjunto que reúna todas las funciones de red y estudiar sus propiedades. De hecho, bajo ciertas condiciones bastante generales, este conjunto es denso en el conjunto de las funciones continuas definidas en un compacto. Dicho resultado se recoge en el siguiente teorema.

Teorema 3.7 (de Aproximación Universal (clásico)). Sea $\varphi : \mathbb{R} \to \mathbb{R}$ una función continua y no polinómica, y sea $\mathcal{N}_n^\varphi$ la clase de funciones de red cuya red neuronal tiene φ por función de activación, n nodos en su capa de entrada, un nodo en su capa de salida cuya función de activación es la identidad y una única capa oculta con un número arbitrario de nodos. Entonces, para todo $K \subset \mathbb{R}^n$ compacto, $\mathcal{N}_n^\varphi$ es denso en $C(K)$.

Es decir, cualquier función real continua definida en un compacto se puede aproximar utilizando una NN con cualquier función de activación arbitraria que sea continua y no sea polinómica. Debido a esta propiedad de aproximar cualquier función continua, se dice que las redes neuronales son aproximadores universales.

El Teorema 3.7, que está referido a redes con una única capa oculta con un número arbitrario de neuronas, se indica como clásico por ser el primero en desarrollarse. Fue probado para funciones de activación sigmoides en 1989 [8], posteriormente, en [14] se demostró que la propiedad de aproximador universal no es consecuencia de la elección de una u otra función de activación sino de la estructura de la propia red. Finalmente, en 1999, tras las publicaciones [21] y [31], se alcanza la formulación del Teorema de Aproximación Universal expuesta en el Teorema 3.7. Actualmente existen demostraciones que no requieren seguir todo el desarrollo histórico del teorema, pero requieren un desarrollo demasiado dilatado para ser incluidas en este documento.

Existen otras versiones del teorema, siendo una de las más relevantes la que se refiere a redes con una cantidad acotada de nodos por capa pero una cantidad arbitraria de capas. Este resultado fue demostrado en 2020 para redes neuronales con cualquier función de activación continua y diferenciable en al menos un punto [18]. Cabe mencionar que, además, existen trabajos que permiten acotar el error cometido, como [46], donde se dan cotas para redes con funciones de activación ReLU

$$\mathrm{ReLU}(x) = \max\{0, x\}. \tag{3.5}$$

3.2.3. Retropropagación y aprendizaje

Las redes neuronales se han descrito como sistemas capaces de producir unos valores de salida para ciertos valores de entrada. Por otro lado, el proceso de retropropagación se corresponde con el mecanismo de aprendizaje automático que permite que una red neuronal mejore su desempeño, es decir, aprenda.

Aunando la estructura de NN y el procedimiento de aprendizaje, el marco de las redes neuronales permite construir agentes inteligentes con capacidad de aprendizaje, satisfaciendo la definición de IA de Russel y Norving [37].

Para llevar a cabo el proceso de aprendizaje se requiere de un elemento fundamental: la función de pérdidas. La función de pérdidas responde a la necesidad de medir cuán satisfactorio se considera el comportamiento de la red. Esta función debe asociar valores menores a las redes que mejor cumplan con el objetivo perseguido.

Un enfoque muy utilizado consiste en tratar la red de manera similar a un regresor. Dada la función de red F y un conjunto no vacío y finito A de pares $(x, y) \in \mathbb{R}^n \times \mathbb{R}^d$, se considera el error cuadrático medio como función de pérdidas

$$\mathrm{MSE}_A(\theta) = \frac{1}{|A|} \sum_{(x,y)\in A} ||y - F(x, \theta)||^2 . \tag{3.6}$$

Nótese que el error cuadrático medio depende de los parámetros de la red. Esto es un requisito indispensable para cualquier función de pérdidas pues el objetivo del algoritmo de retropropagación es determinar el gradiente de la función de pérdidas tomando derivadas parciales con respecto a los parámetros.

A continuación se muestra una descripción de la retropropagación en notación algorítmica.

Algorithm 1: Retropropagación

Input: (V, E, φ, w, b), $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n_0} \times \mathbb{R}^{n_M}$, función de pérdidas $L = \text{MSE}$

Output: Gradiente con respecto a los parámetros (pesos y sesgos)

Inicialización: Los parámetros considerados son $w_{i,j}^m = w(v_{m-1,i}, v_{m,j})$ para $m = 1, 2, \ldots, M$, $1 \leq i \leq n_{m-1}$ y $1 \leq j \leq n_m$ y $b_{m,j} = b(v_{m,j})$ para $m = 1, 2, \ldots, M$, $1 \leq j \leq n_m$

Propagación hacia adelante:

Establecer $\mathbf{o}_0 = \mathbf{x}$ (siendo $o_{m,j}$ la salida correspondiente al nodo $v_{m,j}$)

for $m = 1, \ldots, M$ **do**

  for $j = 1, \ldots, n_m$ **do**

    Calcular $a_{m,j} = b_{m,j} + \sum_{i=1}^{n_{m-1}} w_{i,j}^m o_{m-1,i}$

    Calcular $o_{m,j} = \varphi\left(a_{m,j}\right)$

Retropropagación:

Establecer $\delta_{M,i} = (o_{M,i} - y_i)\varphi'(a_{M,i})$, para cada i con $1 \leq i \leq n_M$

for $m = M-1, M-2, \ldots, 1$ **do**

  for $i = 1, \ldots, n_m$ **do**

    Calcular $\delta_{m,i} = \sum_{j=1}^{n_{m+1}} w_{i,j}^{m+1} \delta_{m+1,j} \varphi'(a_{m+1,j})$

Salida:

Cada peso $w_{i,j}^m$ se asocia con la derivada parcial $\partial L / \partial w_{i,j}^m = \delta_{m,j} o_{m-1,i}$,

Cada sesgo $b_{m,j}$ se asocia con la derivada parcial $\partial L / \partial b_{m,j} = \delta_{m,j}$.

Si bien el algoritmo de retropropagación permite obtener el gradiente de la función de pérdidas, es necesario otro algoritmo que utilice dicho gradiente para optimizar los parámetros. De esta forma, los pesos y los sesgos se actualizan iterativamente, buscando minimizar la función de pérdidas con cada iteración.

Uno de estos métodos de optimización es el descenso de gradiente, que, utilizando los resultados del algoritmo de retropropagación, consiste en actualizar los parámetros como sigue

$$w_{i,j}^{m,\tau+1} = w_{i,j}^{m,\tau} - \eta \delta_{m,j}^{\tau} o_{m-1,i}^{\tau}, \qquad b_{m,j}^{\tau+1} = b_{m,j}^{\tau} - \eta \delta_{m,j}^{\tau}, \tag{3.7}$$

donde los superíndices τ indican la iteración y η se suele denotar en la literatura como learning rate o tasa de aprendizaje.

Tanto la tasa de aprendizaje como el método de optimización elegidos son **hiperparámetros**, es decir, parámetros de la red que describen parte del proceso de entrenamiento e influyen en el mismo.

En general se tiende a utilizar una variante del descenso del gradiente conocida como descenso de gradiente estocástico (SGD). La descripción de este método de

optimización se puede encontrar en la Sección 20.6 de [13] y se remite a [28] para un tratamiento en mayor profundidad.

Otros ejemplos de métodos de optimización son ADAM [19] o la variante del método de optimización Broyden-Fletcher-Goldfarb-Shanno (BFGS) llamada L-BFGS [30].

3.2.4. Physics-informed neural networks

Existe un tipo particular de red neuronal capaz de aproximar soluciones de EDPs: las physics-informed neural networks (PINN). Estas redes incorporan restricciones adicionales en su entrenamiento que garantizan que la salida de la red cumpla ciertas propiedades deseadas.

La idea básica detrás de la técnica consiste en calcular, mediante los algoritmos de derivación automática, las derivadas numéricas de la propia red con respecto a las distintas variables del problema e incorporarlas a la función de pérdidas del entrenamiento, de forma que se satisfaga la EDP. De esta forma se consigue que la PINN actúe como un aproximador universal capaz de satisfacer las condiciones impuestas por una EDP.

Las PINN pueden trabajar con conjuntos de datos de manera similar a lo expuesto para las NN convencionales (aprendizaje supervisado), pero también pueden hacerlo directamente sobre la ecuación (aprendizaje no supervisado). Esta segunda opción es la que se va a seguir para el problema considerado en este documento.

La metodología que se describe a continuación se corresponde con la utilizada en [39], que se basa en la metodología desarrollada en [34] y refinada en [43].

Considérese una EDP arbitraria como sigue

$$\begin{aligned} &\mathcal{N}_I(v(t,x)) = 0, \quad && x \in \widetilde{\Omega}, t \in [0,T], \\ &\mathcal{N}_B(v(t,x)) = 0, \quad && x \in \partial\widetilde{\Omega}, t \in [0,T], \\ &\mathcal{N}_0(v(t^*,x)) = 0, \quad && x \in \widetilde{\Omega} \text{ y, } t^* = 0 \text{ ó } t^* = T, \end{aligned} \tag{3.8}$$

donde $v(t,x)$ denota la solución de la EDP, $\mathcal{N}_I(\cdot)$ es un operador diferencial dependiente del tiempo, $\mathcal{N}_B(\cdot)$ es un operador de frontera, $\mathcal{N}_0(\cdot)$ es un operador de tiempo inicial o final, $\widetilde{\Omega}$ es un subconjunto de $\mathbb{R}^n$ y $\partial\widetilde{\Omega}$ denota la frontera del dominio $\widetilde{\Omega}$.

El objetivo obtener una función $\hat{v}(t,x)$ minimizando una cierta función $L(v)$, de forma que $\hat{v}(t,x)$ sea solución de la EDP y $L(v)$ esté definida sobre el espacio de las funciones k veces diferenciables (k depende del orden de las derivadas de la EDP). Atendiendo a [34] y [43], la función siguiente en términos de la norma L^p satisface

las condiciones requeridas:

$$\hat{L}(v) = \lambda \int_{\Omega} |\mathcal{N}_I(v(t,x))|^p \, d\Omega + (1-\lambda) \int_{\partial\Omega} (|\mathcal{N}_B(v(t,x))|^p + |\mathcal{N}_0(v(t,x))|^p) \, d\gamma,$$

donde $\Omega = \widetilde{\Omega} \times [0,T]$ y $\partial\Omega$ es la frontera de Ω.

Finalmente, se obtiene la PINN aproximando la función $\hat{v}(t,x)$ mediante la función de red $F(y,\theta)$, donde $y = (t,x)$. Para ello basta con considerar como función de pérdidas la aproximación de $\hat{L}$ mediante técnicas de Monte-Carlo:

$$L(\theta) = \lambda \frac{1}{n_I} \sum_{i=1}^{n_I} \left|\mathcal{N}_I(F(y_i^I,\theta))\right|^p + (1-\lambda) \left(\frac{1}{n_B} \sum_{i=1}^{n_B} \left|\mathcal{N}_B(F(y_i^B,\theta))\right|^p + \frac{1}{n_0} \sum_{i=1}^{n_0} \left|\mathcal{N}_0(F(y_i^0,\theta))\right|^p \right),$$

donde los puntos $\{y_i^I\}_{i=1}^{n_I}$, $\{y_i^B\}_{i=1}^{n_B}$ y $\{y_i^0\}_{i=1}^{n_0}$ están distribuidos, respectivamente, sobre el dominio Ω, la frontera $\partial\Omega$, y el dominio $T \times \widetilde{\Omega}$.

3.3. Otros métodos de aprendizaje automático

Más allá de las redes neuronales, existen multitud de métodos de aprendizaje automático. Dado que existe una enorme variedad de ellos, a la hora de enfrentar un problema conviene aplicar los métodos que mejor se adecúen a las características del mismo. Este es uno de los motivos por el que se ha optado por utilizar redes neuronales en este trabajo. El otro motivo fundamental es que existe una amplia literatura sobre su uso en valoración de derivados.

En cualquier caso, existen otras técnicas que pueden resultar de interés en el ámbito de las finanzas y sería interesante estudiarlas en profundidad en el futuro. Entre ellas se encuentran los árboles de decisión, que se han utilizado extensamente en el ámbito del aprendizaje automático, dando muy buenos resultados como clasificadores y regresores.

Dentro de la categoría de clasificadores, entre otras, se encuentra la metodología Support Vector Machine (SVM), que consiste en construir un hiperplano (o varios) que separe clases dentro del conjunto de datos de un espacio $\mathbb{R}^d$. Para determinar la posición del hiperplano se busca maximizar su distancia con respecto a los puntos más próximos pertenecientes a cualquier clase. La metodología SVM se aplica también en problemas de regresión y admite una gran variedad de modificaciones que permiten adecuarla a cada problema.

Otros ejemplos de clasificador son los One-vs-Rest, que consisten en convertir un problema de clasificación multiclase en varios problemas de clasificación binaria, y los clasificadores Naive Bayes, que se fundamentan en la aplicación del Teorema de Bayes.

Por otro lado, en problemas de regresión, además de los árboles de decisión, se suelen utilizar modelos de regresión isotónica, de survival regression y factorization machines, que también se pueden aplicar en problemas de clasificación binaria.

Actualmente, la literatura cuenta con innumerables ejemplos del uso de métodos que combinan los resultados de varios modelos, asumiendo la hipótesis de que el resultado que puedan proveer de manera conjunta es más correcto que el de cualquiera de ellos por separado. Estos métodos se denominan métodos ensemble que se suelen clasificar en boosting, bagging y stacking (ver [20]).

Existe una gran variedad de modelos en la literatura. Entre los más habituales se encuentran, dentro de la categoría de bagging, los random forest y, en la categoría de boosting, los Gradient-Boosted Tree (GBT) y los Extreme Gradient Boosting Trees (XGBT), todos ellos utilizados en tanto en problemas de clasificación como en problemas de regresión.

En particular, los modelos XGBT suele conducir a resultados especialmente buenos en la literatura cuando se aplican a problemas de regresión. Además son paralelizables y se pueden implementar de manera distribuida. Este modelo fue propuesto y utilizado por primera vez en [42].

4. Aplicación práctica

4.1. Introducción

A lo largo de este documento se han expuesto múltiples métodos de resolución de EDPs, entre ellos se encuentra el que se probará en este Capítulo, el de las PINN.

El Teorema de Aproximación Universal invita al uso de redes neuronales en la aproximación de funciones y las PINN resuelven el problema de aproximar soluciones de ecuaciones diferenciales. De hecho, se ha comentado en el Capítulo 3 que existe una vasta literatura sobre el uso de PINN para la resolución de EDPs, señalando [23] y [34] como ejemplos. Más particularmente, la aplicación de redes neuronales, no sólo a través de las PINN, a modelos de matemática financiera, como el modelo de Black-Scholes, ha demostrado ser un área de investigación prolífica, como se evidencia en [36]. La citada revisión bibliográfica se centra en la valoración de precios de derivados utilizando redes neuronales y cuenta con un compendio de más de cien referencias, usando distintos enfoques. Esto da cuenta de que es un campo de investigación activo y por eso se ha elegido como tema en este TFM, que pretende ser una toma de contacto con este área.

La revisión bibliográfica data de 2019, por tanto, puede estar ligeramente actualizada pero sirve como punto de partida a la hora de tener una visión de conjunto del campo de estudio. En la publicación se llega a varias conclusiones sobre el estado del arte:

- Existen dos formas de utilizar el valor de la acción y el strike como entradas de una red neuronal. O bien se introducen como características separadas o bien se introduce su proporción. En particular, es más común en el caso del modelo de BS y la Call Europea, pues el precio no depende de la magnitud del valor de la acción ni del strike, sino de la proporción entre ambos debido a la linearidad del modelo matemático. Esto además es un tipo de normalización, algo a tener presente en el entrenamiento de redes según se ha visto en bibliografía revisada.

- Trabajando no sólo con modelos teóricos, sino con datos reales de mercado, el valor de la volatilidad y, consiguientemente, el método de estimación de la

misma en un mercado, tiene una gran influencia en los resultados obtenidos.

- También con datos de mercado, en algunos estudios, los datos se dividen en conjuntos de entrenamiento y prueba obviando la estructura temporal subyacente. Esta pérdida de información implícita acarrea varios problemas, muchos relacionados con los errores de generalización.

Una de las primeras publicaciones en mostrar el uso de redes neuronales para estimar el precio de derivados es [15]. Introduce una metodología para evaluar el precio de derivados a lo largo de múltiples periodos que se ha utilizado posteriormente en numerosas ocasiones. Tanto es así, que actualmente ha sido citada en más de 1100 ocasiones. Otra publicación de especial relevancia es [1], que compara el desempeño de las redes neuronales y el modelo de Black-Scholes a la hora de estimar precios de Call utilizando distintos estimadores de la volatilidad del mercado. Para ello utiliza datos recogidos del índice de acciones DAX (índice de las 40 compañías más grandes de Alemania que cotizan en la bolsa de Fráncfort).

Además de los dos casos citados en el párrafo anterior, existen multitud de publicaciones que abordan el problema desde diferentes puntos de vista, valorarando los derivados con un modelo teórico, sólo a través de los precios de mercado o aprovechando la información conjunta de un modelo teórico y datos de mercado. También se puede intentar estimar los parámetros del mercado partiendo de los precios de los derivados negociados en él.

Al ser este un trabajo de toma de contacto con el campo, y además carecer de una base de datos reales de mercado, como experimento numérico se ha decidido implementar una PINN que intente aproximar el valor de una función que verifique la EDP del modelo de Black-Scholes, siguiendo la línea presentada en [39].

Sin entrar en mucha profundidad en todos los tecnicismos inherentes al trabajar con redes neuronales, este experimento sirve para comprobar cómo funcionan las PINN y todos los factores que hay que tener en cuenta en el uso de las redes neuronales.

4.2. Motivación

Por lo visto en las distintas asignaturas del Máster y en este trabajo, parece evidente que no hay un método numérico universalmente bueno. Todos ellos tienen sus ventajas e inconvenientes. Para analizarlas de manera sintética y visual se ha optado por ilustrarlas en la Tabla 4.1.

Como se puede ver en la tabla, todos los métodos tienen pros y contras. De hecho, en muchos problemas, es casi seguro que los dos primeros métodos sean más eficaces que las redes. No obstante, es su fácil implementación y el potencial de

	Simulaciones de Monte-Carlo	**Métodos numéricos**	**Physics-Informed NN**
Convergencia y cotas de error	Estimación estadística y, dependiendo del problema, demandante computacionalmente	Dependiente del método y/o el problema. Abundancia de resultados teóricos.	Requiere elegir bien los hiperparámetros de la misma y resultados téoricos sólo en ciertos casos.
Dimensionabilidad	Es fácilmente paralelizable.	Puede ser muy demandante computacionalmente	Las redes tienen potencial para trabajar con datos multidimensionales
Adaptabilidad	En general es versátil y adaptable	Puede requerir una implementación laboriosa	Fácilmente definibles, pero los hiperparámetros de la red pueden depender del problema

Tabla 4.1: Comparación de los distintos métodos de resolución de EDPs tratados.

que redes neuronales no excesivamente grandes puedan funcionar bien en modelos multidimensionales lo que las convierte en un campo de estudio de interés.

El mayor inconveniente, a efectos de implementación, que presentan las redes neuronales es el gran número de factores que influyen en el entrenamiento y en el nivel de aproximación a la solución buscada de la red.

El entrenamiento de una red neuronal es un problema de optimización en un espacio multidimensional, y a priori no se tiene garantizado alcanzar el mínimo global, pudiendo pararse en un mínimo local. Además, el tiempo empleado en el entrenamiento puede depender del optimizador elegido y los parámetros del mismo.

Por otro lado, la precisión alcanzada por la red (en caso de alcanzarse, el valor del mínimo global del entrenamiento) puede depender de la arquitectura de la red y de las funciones de activación elegidas.

En [7] se puede encontrar una revisión del método de las PINN que ahonda en todos estos detalles. No obstante, pese a los problemas comentados, en esa misma publicación pueden encontrarse numerosas referencias de estructuras de redes neuronales que han funcionado bien para diversas ecuaciones diferenciales, tanto

ordinarias como en derivadas parciales, indicando qué tipo de arquitecturas, funciones de activación, modificaciones de la función de pérdidas, formas de imponer las condiciones frontera en la red, etc. funcionan mejor.

4.3. Experimento numérico

La Call europea se negocia hoy ($t = 0$) con un vencimiento T, luego $t \in [0, T]$. Con respecto a la variable espacial, el problema se localiza en un dominio finito, $S \in [0, S_R]$, donde $S_R > K$. Señalar que la localización del problema genera un error numérico, pero que no se va a tratar en este trabajo. La dinámica de la Call Europea para un strike K viene descrita por la EDP

$$\begin{cases} \mathcal{L}(C) = \frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0\,, & S \in [0, S_R],\ t \in [0, T), \\ C(T, S) = H(S) = \text{máx}\{S - K, 0\}\,, & S \in [0, S_R], \\ C(t, 0) = G_1(t, 0) = 0, & t \in [0, T), \\ C'(t, S_R) = G_2(t, S_R) = 1, & t \in [0, T). \end{cases} \tag{4.1}$$

Consecuentemente, atendiendo a la ecuación (2.61) y manteniendo la notación, se tiene que la función de pérdidas de la PINN viene dada por la expresión

$$\begin{aligned} L(\theta) = \lambda \frac{1}{n_I} \sum_{i=1}^{n_I} \left| \mathcal{L}\left(F\left(\mathbf{y}_i^I, \theta\right)\right)\right|^p + (1-\lambda)\left(\frac{1}{n_0}\sum_{i=1}^{n_0}\left|F\left(\mathbf{y}_i^0, \theta\right) - H\left(S_i^0\right)\right|^p \right. \\ \left. + \frac{1}{n_{B_1}}\sum_{i=1}^{n_{B_1}}\left|F\left(\mathbf{y}_i^{B_1}, \theta\right) - G_1\left(\mathbf{y}_i^{B_1}\right)\right|^p + \frac{1}{n_{B_2}}\sum_{i=1}^{n_{B_2}}\left|F'\left(\mathbf{y}_i^{B_2}, \theta\right) - G_2\left(\mathbf{y}_i^{B_2}\right)\right|^p \right), \end{aligned} \tag{4.2}$$

donde λ es un hiperparámetro, $\mathbf{y} = (t, S)$ y además $\mathbf{y}^I$, $\mathbf{y}^0$, $\mathbf{y}_1^B$ y $\mathbf{y}_2^B$ denotan puntos pertenecientes al interior y a las respectivas fronteras

4.3.1. Implementación

En cuanto a la arquitectura de red, se requiere una capa de entrada con dos nodos y una capa de salida de uno. La red implementada consta de 4 capas ocultas con 40 nodos cada una. La función de activación elegida es la tangente hiperbólica.

Se ha elegido el optimizador ADAM y para calcular las derivadas de la función de red, necesarias para el cómputo de $\mathcal{L}\left(F\left(\mathbf{y}_i^I, \theta\right)\right)$, se ha utilizado la diferenciación automática (ver [35]).

Se han utilizado dos conjuntos de datos disjuntos en el entrenamiento: uno de ellos, generado aleatoriamente con el método de Latin Hypercube en el dominio, se

ha utilizado para entrenar la red (conjunto de entrenamiento), mientras que el otro, calculado como una malla uniforme en el dominio, ha servido para verificar el error de generalización de la misma (conjunto de validación), es decir, el error que comete cuando se introducen datos diferentes a los que ya ha usado en el entrenamiento.

Es importante señalar que esto es crucial en el entrenamiento de las redes neuronales. Se deben elegir dos conjuntos disjuntos, uno para el entrenamiento y el otro para el de validación. Uno de los problemas que puede surgir en el entrenamiento de las redes neuronales es el overfitting, es decir, que la red se sobreajuste a los datos de entrenamiento y deje de tener sentido en el resto del dominio.

Ambos errores se suelen monitorizar conjuntamente. Mientras que, en general, el error de entrenamiento siempre es decreciente, el error de generalización primero decrece y luego puede crecer, señal de que se puede estar entrando en una zona de sobreajuste. Esta consideración es un criterio habitual de parada en el entrenamiento de redes neuronales.

El entrenamiento de la red se ha realizado mediante batches, es decir, se suministra a la red una colección de datos y se actualizan los parámetros de la red tras el procesamiento completo de toda la colección. En particular el tamaño de batch utilizado (el tamaño de la colección) se corresponde con el tamaño del conjunto de entrenamiento.

Tanto el mallado como el Latin Hypercube se han realizado dividiendo los intervalos $[0, T]$ y $[0, S_R]$ en 150 partes. Es decir, tanto el conjunto de entrenamiento como el de validación contienen 22.500 puntos.

Se ha tomado el hiperparámetro $\lambda = 0{,}5$, un tipo de interés libre de riesgo $r = 0{,}03$ y una volatilidad anualizada $\sigma = 0{,}1$. Se han elegido el vencimiento $T = 1$, la cotización máxima $S_R = 2$ y $K = 1$ como valor del strike.

4.3.2. Resultados

Nótese que en las PINN hay dos tipos de errores. Uno es el error del residuo de la EDP, lo que se ha definido como la función de pérdidas. El otro es el error de aproximación del valor de la función, es decir, del precio de la Call Europea.

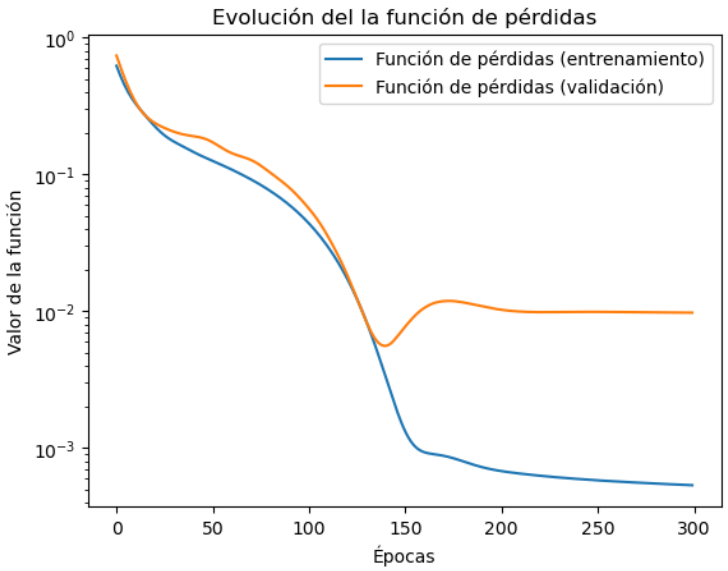


Figura 4.1: Evolución del valor de la función de pérdidas y del error generalizado a lo largo del entrenamiento

En la Figura 4.1 se representa la evolución del valor de la función de pérdidas a lo largo del entrenamiento. El eje de abscisas se corresponde con el número de veces que la red ha procesado completamente el conjunto de entrenamiento, cada uno de los procesados se conoce como época. Conviene observar que la función de pérdidas en el entrenamiento, representada en azul, toma valores menores que en el conjunto de valoración. Esto, es un síntoma de un entrenamiento correcto.

Por otro lado, la Figura 4.2 presenta el error de aproximación. Esta gráfica se prolonga hasta las 2.000 épocas con el objetivo de identificar si se produce overfitting. Precisamente, la gráfica alcanza un mínimo alrededor de las 150 épocas. Esto significa que a partir de ese punto del entrenamiento la red comienza a sobreajustar. De hecho, esto se puede observar también en la Figura 4.1, donde el residuo de la EDP también muestra un mínimo entorno a 150 épocas.

Esta observación sobre el sobreajuste invita a incluir un criterio de parada durante el entrenamiento para evitar invertir recursos computacionales de manera innecesaria.

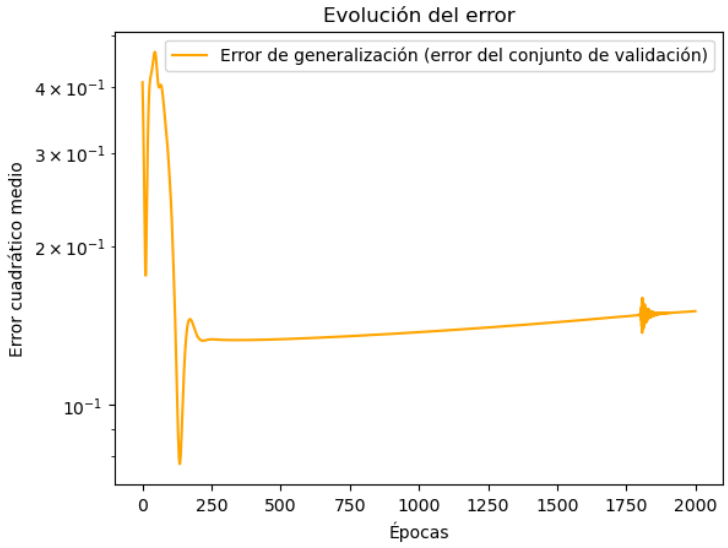
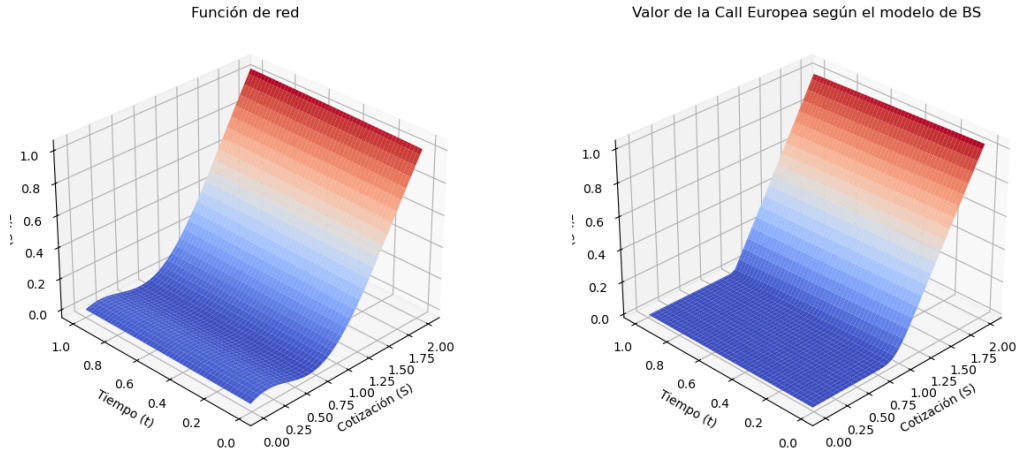


Figura 4.2: Evolución del error de aproximación

La Figura 4.3 muestra, a la izquierda, el precio aproximado de la Call obtenido con la red, y a la derecha el precio exacto. A simple vista se aprecia que, aunque se ha capturado la forma de la función, existe un error numérico.



(a) Función de red tras el de entrenamiento

(b) Valor de la Opción Europea según el modelo de Black-Scholes

Figura 4.3: Comparación de la red con el valor teórico

Para ver con mayor claridad el error cometido por la red, se muestra la Figura 4.4,

que representa

$$\text{Error}(t, S) = F(t, S) - C(t, S), \tag{4.3}$$

donde F es la función de red y C el valor teórico de la Call.

En esta gráfica se observa que el error cometido es inferior a 0.05 (los valores reales de la Call varían de 0 a 1). Como se ha comentado, la precisión alcanzable por la red y el entrenamiento adecuado dependen de muchos factores que no se pretenden analizar en detalle en este trabajo ni el objetivo es encontrar la mejor red para este problema.

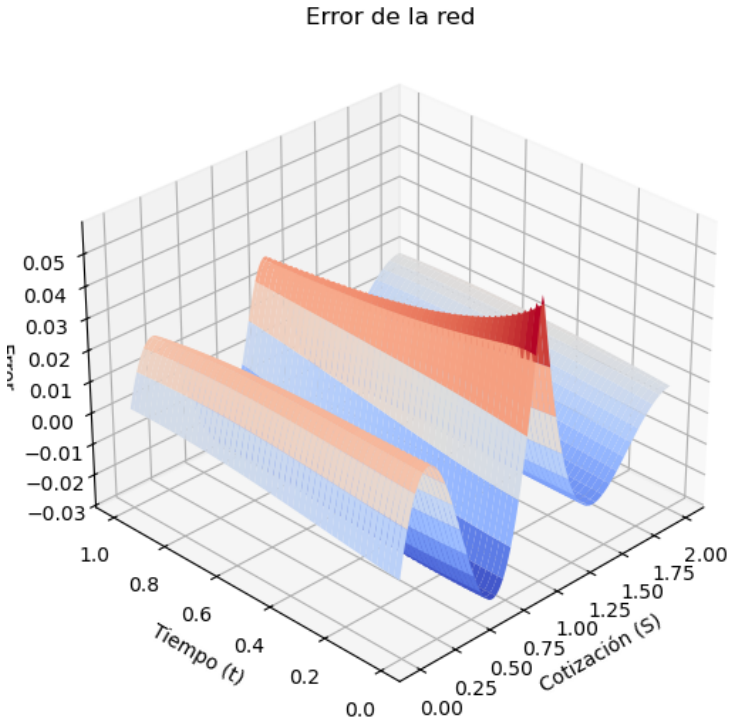


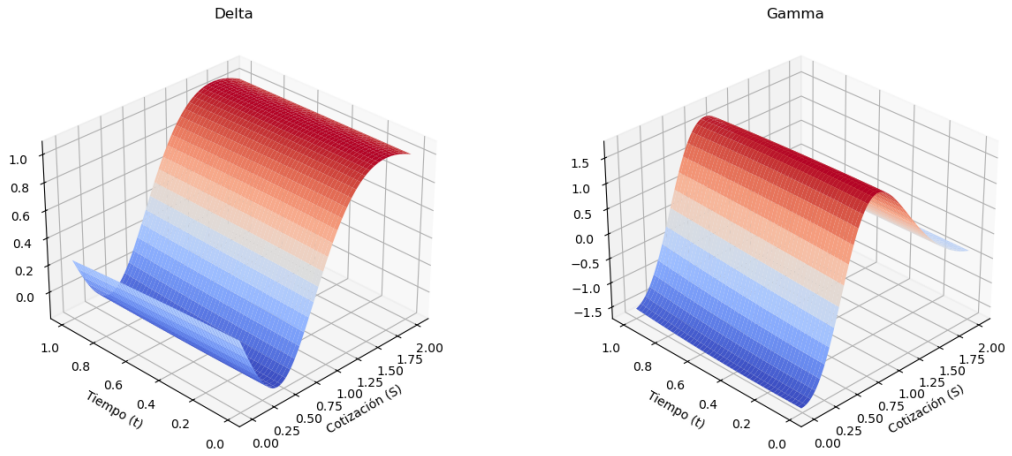
Figura 4.4: Diferencia entre la función de red y al valor teórico

No obstante, hay otro análisis que es conveniente hacer. En la valoración de derivados se utilizan con frecuencia varias funciones de especial importancia para los economistas. Estas funciones se denominan griegas y se definen como

$$\Delta = \frac{\partial C}{\partial S}, \qquad \Gamma = \frac{\partial^2 C}{\partial S^2}, \qquad \Theta = \frac{\partial C}{\partial t}. \tag{4.4}$$

La importancia de la primera griega ya se ha visto, informa de la composición del portfolio de réplica del derivado. Las otras también son de uso habitual en finanzas ya que informan de la variabilidad del precio del derivado

La Figura 4.5 muestra la delta y la gamma calculadas sobre la función de red por diferenciación automática. Aunque existe de nuevo un error numérico, cualitativamente sí se corresponden con las verdaderas funciones delta y gamma. Esto implica que la red, una vez bien afinada, quizás también podría emplearse para calcular estos valores de relevancia para los economistas.



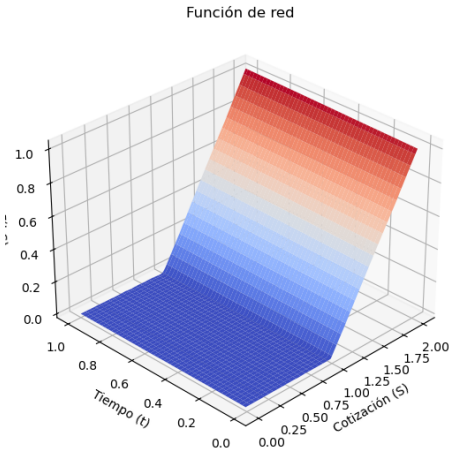
(a) Función delta $\partial C/\partial S$ (b) Función gamma $\partial^2 C/\partial S^2$

Figura 4.5: Griegas de la solución aproximada

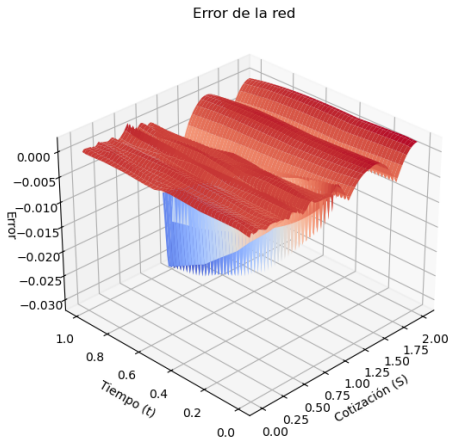
Para finalizar, comentar un resultado que se obtuvo haciendo otro experimento. Por comprobar si las ondulaciones de la Figura 4.4 podían reducirse usando una combinación de funciones de activación se implementó una red con la función tanh en la primera capa y ReLu en las siguientes.

En la Figura 4.6 se puede ver que los resultados inciales fueron prometedores, ya que el error se redujo en gran parte del dominio. No obstante cuando se calcularon las griegas, se observó que la delta había perdido su regularidad y que la Gamma no se corresponde con el comportamiento que debería tener.

Este ejemplo se ha incluido para recalcar la importancia que tiene realizar siempre diversos análisis de los resultados numéricos obtenidos. Aunque se haya obtenido un resultado que a priori parece mejor en la función del precio de la Call, este tipo de red no sería demasiado recomendable, ya que no refleja otras propiedades de la función que se desea aproximar.

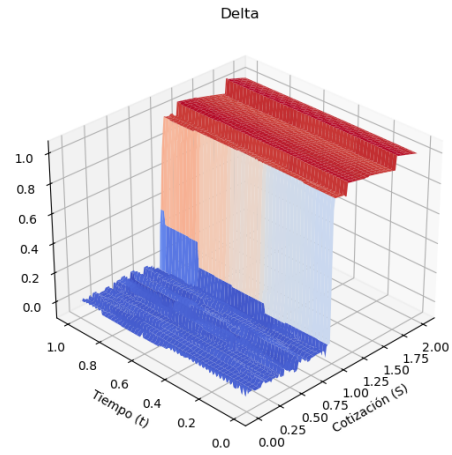


(a) Función de red tras el entrenamiento

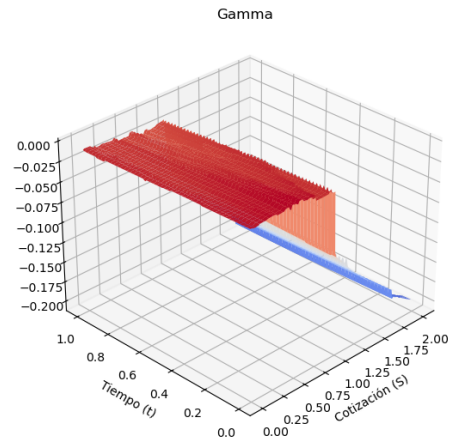


(b) Diferencia entre la función de red y al valor teórico

Figura 4.6: Gráficas de la función de red y del error correspondientes al segundo experimento numérico



(a) Función delta $\partial C/\partial S$



(b) Función gamma $\partial^2 C/\partial S^2$

Figura 4.7: Griegas del segundo experimento numérico

5. Conclusiones y líneas de trabajo futuro

Como se ha comentado, las redes entrenadas han sido un pequeño experimento para comprobar la viabilidad del método. Se considera que la implementación ha sido satisfactoria puesto que se ha seguido la metodología, se han conseguido resultados aceptables y, sobre todo, se ha comprobado la viabilidad de la técnica. De hecho, los experimentos numéricos realizados resultan inspiradores para intentar aprovechar todo el potencial de las PINN.

Por otro lado, obtener redes que ajusten correctamente la función es una tarea laboriosa que requiere estar familiarizado con la metodología. Obtener una buena estimación con errores muy reducidos conlleva realizar un elevado número de entrenamientos en los que se utilicen diferentes combinaciones de hiperparámetros y se prueben diversas arquitecturas. La cantidad de posibles combinaciones es ingente y, además de ciertas directrices generales, el proceso de prueba y error es el único método para determinar en qué dirección avanzar a la hora de introducir cambios en la red que optimicen los resultados al máximo.

Haber aplicado la metodología y conocer de primera mano las dificultades que entraña el entrenamiento de redes neuronales ha sido el principal objetivo de este trabajo.

Por otro lado, entre las ventajas de las redes neuronales, se encuentra que son una herramienta extremadamente versátil. En particular, las PINN permiten adaptar los beneficios de las redes a la resolución de ecuaciones diferenciales. Una vez entrenadas, las redes permiten realizar cálculos rápidamente con grandes cantidades de datos, lo cual tiene especial trascendencia cuando se trata de resolver EDPs con un número elevado de variables en finanzas, donde los precios deben recalcularse velozmente tras cada cambio en la cotización de un activo. Consecuentemente, es una línea de trabajo que despierta gran interés entre los profesionales dedicados a la negociación de productos financieros.

Se ha visto que las NN permiten, en ciertos casos, no solo aproximar la solución de la EDP, sino también sus derivadas. Esto tiene una importancia capital en economía

pues, en el experimento realizado, se han conseguido aproximar las griegas.

Una de las direcciones más claras que se pueden seguir para mejorar el modelo es entrenar más modelos, con mayor eficacia y buscar hiperparámetros que conduzcan a mejores soluciones. Conviene señalar que para ello se requerirían mejores máquinas, pues como se ha observado, se manejan grandes cantidades de datos y redes con un gran número de parámetros. Todo esto acarrea un elevado coste computacional.

Una línea de trabajo futuro interesante sería, siguiendo las indicaciones de [7], estudiar qué nivel de precisión se puede alcanzar en las ecuaciones de finanzas (no sólo Black-Scholes sino también en otros modelos, por ejemplo con costes) y ver su rendimiento en modelos multidimensionales.

Para concluir, se quiere mencionar la posibilidad de aplicar algunos de los métodos mencionados en la Sección 3.3 o la combinación de las simulaciones de Monte-Carlo con redes neuronales descrita en la Sección 3.1. Todas las opciones parecen suficientemente prometedoras e interesantes.

Índice de Contenidos

Índice de figuras

Bibliografía

[1] Ulrich Anders, Olaf Korn, and Christian Schmitt, *Improving the pricing of options: A neural network approach*, Journal of forecasting **17** (1998), no. 5-6,

[2] Afşar Onat Aydınhan, Xiaoyue Li, and John M Mulvey, *Solving multi-period financial planning models: Combining monte carlo tree search and neural networks*,

[3] Tomas Björk, *Arbitrage theory in continuous time*, Oxford university press,

[4] Fischer Black and Myron Scholes, *The pricing of options and corporate liabilities*, Journal of political economy **81** (1973), no. 3,

[5] Paolo Brandimarte, *Handbook in monte carlo simulation: applications in financial engineering, risk management, and economics*, John Wiley & Sons,

[6] Leo Breiman, *Statistical modeling: The two cultures (with comments and a rejoinder by the author)*, Statistical science **16** (2001), no. 3, DOI: 10.1214/ss/1009213726.

[7] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli, *Scientific machine learning through physics–informed neural networks: Where we are and what's next*, Journal of Scientific Computing **92** (2022), no. 3,

[8] George Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems **2** (1989), no. 4,

[9] Roger Eckhardt, *Stan ulam, john von neumann*, Los Alamos Science **100** (1987), no. 15,

[10] Association for Computing Machinery (ACM), IEEE-Computer Society (IEEE-CS), and Association for the Advancement of Artificial Intelligence (AAAI), *Computer science curricula*, SIGCSE 2023: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2,

[11] David Gottlieb and Steven A Orszag, *Numerical analysis of spectral methods: theory and applications*, SIAM,

[12] Steven L Heston, *A closed-form solution for options with stochastic volatility with applications to bond and currency options*, The review of financial studies **6** (1993), no. 2,

[13] Catherine F Higham and Desmond J Higham, *Deep learning: An introduction for applied mathematicians*, Siam review **61** (2019), no. 4,

[14] Kurt Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural networks **4** (1991), no. 2,

[15] James M Hutchinson, Andrew W Lo, and Tomaso Poggio, *A nonparametric approach to pricing and hedging derivative securities via learning networks*, The journal of Finance **49** (1994), no. 3,

[16] Claes Johnson, *Numerical solution of partial differential equations by the finite element method*, Courier Corporation,

[17] Ioannis Karatzas and Steven Shreve, *Brownian motion and stochastic calculus*, vol. 113, Springer,

[18] Patrick Kidger and Terry Lyons, *Universal approximation with deep narrow networks*, Conference on learning theory, PMLR, 2020,

[19] Diederik P Kingma, *Adam: A method for stochastic optimization*,

[20] Ludmila I Kuncheva, *Combining pattern classifiers: methods and algorithms*, John Wiley & Sons,

[21] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural networks **6** (1993), no. 6,

[22] John McCarthy, Marvin L Minsky, Nathan Rochester, and Claude E Shannon, *A proposal for the dartmouth summer research project on artificial intelligence*,

[23] Andrew J Meade Jr and Alvaro A Fernandez, *The numerical solution of linear ordinary differential equations by feedforward neural networks*, Mathematical and Computer Modelling **19** (1994), no. 12,

[24] Robert C Merton, *An intertemporal capital asset pricing model*, Econometrica: Journal of the Econometric Society (1973),

[25] Nicholas Metropolis and Stanislaw Ulam, *The monte carlo method*, Journal of the American statistical association **44** (1949), no. 247,

[26] Hrushikesh Narhar Mhaskar, *Approximation properties of a multilayered feed-forward artificial neural network*, Advances in Computational Mathematics **1** (1993), no. 1,

[27] Tom Mitchell, *Machine learning*, McGraw Hill,

[28] Kevin P Murphy, *Probabilistic machine learning: an introduction*, MIT press,

[29] Bernard Nayroles, Gilbert Touzot, and Pierre Villon, *Generalizing the finite element method: diffuse approximation and diffuse elements*, Computational mechanics **10** (1992), no. 5, DOI: 10.1007/BF00364252.

[30] Jorge Nocedal, *Updating quasi-newton matrices with limited storage*, Mathematics of computation **35** (1980), no. 151,

[31] Allan Pinkus, *Approximation theory of the mlp model in neural networks*, Acta numerica **8** (1999),

[32] J. R. Quinlan, *Discovering rules by induction from large collection of examples*, Expert Systems in the Microelectronic Age (D. Michie, ed.), Edinburgh University Press, 1979,

[33] _____, *C4.5: Programs for machine learning*, Morgan Kaufmann Publishers (1993),

[34] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, *Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations*,

[35] Louis B Rall, *Automatic differentiation: Techniques and applications*, Springer,

[36] Johannes Ruf and Weiguan Wang, *Neural networks for option pricing and hedging: a literature review*,

[37] Stuart Russell and Peter Norvig, *Artificial intelengence. a modern approach*, Prentice Hall,

[38] Sandro Salsa, *Partial differential equations in action: From modelling to theory*, Springer,

[39] Beatriz Salvador, Cornelis W Oosterlee, and Remco van der Meer, *Financial option valuation by unsupervised learning with artificial neural networks*, Mathematics **9** (2020), no. 1,

[40] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge University Press,

[41] Ronald W Shonkwiler, *Finance with monte carlo*, Springer,

[42] Chen Tianqi and Guestrin Carlos XGBoost, *A scalable tree boosting system*, Proceedings of the 22nd ACM SIGKDD **1143** (2016),

[43] Remco van der Meer, Cornelis W Oosterlee, and Anastasia Borovykh, *Optimally weighted loss functions for solving pdes with neural networks*, Journal of Computational and Applied Mathematics **405** (2022),

[44] Paul J Werbos, *Generalization of backpropagation with application to a recurrent gas market model*, Neural networks **1** (1988), no. 4, DOI: 10.1016/0893-6080(88)90007-X.

[45] Paul Wilmott, *Paul wilmott on quantitative finance*, John Wiley & Sons,

[46] Dmitry Yarotsky, *Error bounds for approximations with deep relu networks*, Neural networks **94** (2017),

A. Código Python

```
# Physiscs informed black-scholes

# Cargo librerias

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import time
from scipy.stats import norm


# Fijo la precisión de Keras
tf.keras.backend.set_floatx('float64')

## Para cargar datos de Matlab
import scipy.io as sio

#%% Testeo de si la GPU está activa

if tf.config.list_physical_devices('GPU'):
  print("TensorFlow **IS** using the GPU")
else:
  print("TensorFlow **IS NOT** using the GPU")


#%% Cargo los datos de Matlab.

# # En matlab se han obtenido los conjuntos de test y train por
  ↪  comodidad
# # Formato adecuado para que python lo lea.
```

```
30
31 datos_train=sio.loadmat('datostrainLhc2.mat')
32
33 ## Train
34
35 # Frontera S=0
36 cotfS0=datos_train.get('cotfS0')
37 tiemfS0=datos_train.get('tiemfS0')
38 valcotfS0=datos_train.get('valcotfS0')
39
40 # Frontera S=Smax
41
42 cotfSmax=datos_train.get('cotfSmax')
43 tiemfSmax=datos_train.get('tiemfSmax')
44 valfderSmax=datos_train.get('valfderSmax')
45
46 # Frontera t=0
47
48 cotft0=datos_train.get('cotft0')
49 tiemft0=datos_train.get('tiemft0')
50 valcotft0=datos_train.get('valcotft0')
51
52 # Interior
53 cotint=datos_train.get('cotint')
54 tiemint=datos_train.get('tiemint')
55 valint=datos_train.get('valint')
56
57 # Valores Test (sólo del interior, no coinciden con train)
58 cottest=datos_train.get('cottest')
59 tiemtest=datos_train.get('tiemtest')
60 valtest=datos_train.get('valtest')
61
62 cotintauxtest=datos_train.get('cotintauxtest').flatten()
63 tiemintauxtest=datos_train.get('tiemintauxtest').flatten()
64
65
66 cotfS0, tiemfS0, valcotfS0=map(lambda x:
    ↪  tf.convert_to_tensor(x,dtype=tf.float64),
67                                [cotfS0, tiemfS0, valcotfS0])
```

```
68 cotfSmax, tiemfSmax, valfderSmax=map(lambda x:
   ↪ tf.convert_to_tensor(x,dtype=tf.float64),
69                                     [cotfSmax, tiemfSmax,
                                       ↪ valfderSmax])
70 cotft0, tiemft0, valcotft0=map(lambda x:
   ↪ tf.convert_to_tensor(x,dtype=tf.float64),
71                                 [cotft0, tiemft0, valcotft0])
72 cotint, tiemint, valint=map(lambda x:
   ↪ tf.convert_to_tensor(x,dtype=tf.float64),
73                               [cotint, tiemint, valint])
74 cottest, tiemtest, valtest=map(lambda x:
   ↪ tf.convert_to_tensor(x,dtype=tf.float64),
75                               [cottest, tiemtest, valtest])
76
77
78
79 ## Test
80
81 # Frontera S=0 test
82 tcotfS0=datos_train.get('tcotfS0')
83 ttiemfS0=datos_train.get('ttiemfS0')
84 tvalcotfS0=datos_train.get('tvalcotfS0')
85
86 # Frontera S=Smax test
87 tcotfSmax=datos_train.get('tcotfSmax')
88 ttiemfSmax=datos_train.get('ttiemfSmax')
89 tvalfderSmax=datos_train.get('tvalfderSmax')
90
91 # Frontera t=0 test
92 tcotft0=datos_train.get('tcotft0')
93 ttiemft0=datos_train.get('ttiemft0')
94 tvalcotft0=datos_train.get('tvalcotft0')
95
96 # Interior test
97 tcotint=datos_train.get('tcotint')
98 ttiemint=datos_train.get('ttiemint')
99 tvalint=datos_train.get('tvalint')
100
101
```

```
102 tcotfS0, ttiemfS0, tvalcotfS0=map(lambda x:
    ↪ tf.convert_to_tensor(x,dtype=tf.float64),
103                                [tcotfS0, ttiemfS0, tvalcotfS0])
104 tcotfSmax, ttiemfSmax, tvalfderSmax=map(lambda x:
    ↪ tf.convert_to_tensor(x,dtype=tf.float64),
105                                      [tcotfSmax, ttiemfSmax,
                                         ↪ tvalfderSmax])
106 tcotft0, ttiemft0, tvalcotft0=map(lambda x:
    ↪ tf.convert_to_tensor(x,dtype=tf.float64),
107                                [tcotft0, ttiemft0, tvalcotft0])
108 tcotint, ttiemint, tvalint=map(lambda x:
    ↪ tf.convert_to_tensor(x,dtype=tf.float64),
109                               [tcotint, ttiemint, tvalint])
110
111 #%% DEFINICIÓN DE LA RED NEURONAL A TRAVÉS DE UNA FUNCIÓN
112
113 def RED_constructor(in_capa=2, out_capa=1, num_capas=3,
114                 neuronas_capa=30, act="relu"):
115
116     # Capa de entrada
117     input_layer = tf.keras.layers.Input(shape=(in_capa,))
118     # Capas ocultas
119     hidden = [tf.keras.layers.Dense(neuronas_capa,
120                                     activation=act)(input_layer)]
121     for i in range(num_capas-1):
122             new_layer = tf.keras.layers.Dense(neuronas_capa,
123                                 activation=act,
124                                 activity_regularizer=None)(hidden[-1])
125             hidden.append(new_layer)
126     # Capa de salida
127     output_layer = tf.keras.layers.Dense(1,
        ↪ activation=None)(hidden[-1])
128     # Se compila el modelo:
129     name = f"RED-{num_capas}"
130     model = tf.keras.Model(input_layer, output_layer, name=name)
131     return model
132
133 # Borro otras redes que tenga definidas
134 tf.keras.backend.clear_session()
135
```

```
136 # Se construye la Red
137 model = RED_constructor(2, 1, 4, 40, 'tanh')
138 model.summary()
139 
140 #%% DEFINICIÓN DE FUNCIONES NECESARIAS PARA LA LOSS FUNCTION
141 
142 # Función que recibe las coordenadas correspondientes S y t y
    ↪ evalúa la red en esos puntos
143 
144 def u(s, t):
145     u = model(tf.concat([s, t], axis=1))
146     return u
147 
148 #%% Función que calcula las derivada primera con respecto a la
    ↪ cotizacion
149 # Devuelve la loss correspondiente a la frontera S=Smax
150 
151 def msederSmax(s, t,val_):
152     u0 = u(s, t)
153     u_s = tf.gradients(u0, s)[0]    # Derivada automática con
        ↪ respecto a S
154     return tf.reduce_mean(tf.square(u_s-val_))
155 
156 #%% Función que devuelve la loss correspondiente a las fronteras
    ↪ t=0 y S=0
157 
158 def msedirichlet(valred,val_):
159     return tf.reduce_mean(tf.square(valred-val_))
160 
161 #%% Función que calcula las EDP del modelo de Black-Scholes
162 # Devuelve la loss correspondiente a las derivadas del interior
    ↪ del dominio
163 
164 def lossedpBS(s, t):
165     r=0.03
166     sig=0.1
167     with tf.GradientTape() as t2:
168         t2.watch(s)
169         with tf.GradientTape() as t1:
170             t1.watch(s)
```

```
171             u0 = u(s, t)
172         u_s=t1.gradient(u0,s)
173     u_ss = t2.gradient(u_s,s)
174 
175     with tf.GradientTape() as t3:
176         t3.watch(t)
177         u0 = u(s, t)
178     u_t=t3.gradient(u0,t)       # Derivada primera con respecto a
        ↪ t
179     # u_ss = tf.gradients(u_s, s)[0]      # Derivada segunda con
        ↪ respecto a s (dos veces)
180     F=-u_t+0.5*sig*sig*tf.math.multiply(tf.square(s),u_ss)
        ↪ +r*tf.math.multiply(s,u_s)-r*u0    # EDP
181     return tf.reduce_mean(tf.square(F))
182     #return u_s,u_ss,u_s
183 
184 #%% Funciones para al análisis de las derivadas
185 
186 def calculoderivadas(s, t):
187     with tf.GradientTape() as t2:
188         t2.watch(s)
189         with tf.GradientTape() as t1:
190             t1.watch(s)
191             u0 = u(s, t)
192         u_s=t1.gradient(u0,s)
193     u_ss = t2.gradient(u_s,s)
194 
195     with tf.GradientTape() as t3:
196         t3.watch(t)
197         u0 = u(s, t)
198     u_t=t3.gradient(u0,t)       # Derivada primera con respecto a
        ↪ t
199     # u_ss = tf.gradients(u_s, s)[0]      # Derivada segunda con
        ↪ respecto a s (dos veces)
200     return u_s,u_ss,u_t
201 
202 #%% Hiperparámetros del entrenamiento
203 
204 loss = 0
205 epochs = 2000
```

```
206 opt = tf.keras.optimizers.Adam(learning_rate=5e-4)
207 epoch = 0
208 loss_values = np.array([])
209 genloss_values = np.array([])
210 testloss_values = np.array([])
211 trainloss_values = np.array([])
212
213 u_as_graph = tf.function(u)
214 msedirichlet_as_graph = tf.function(msedirichlet)
215 msederSmax_as_graph = tf.function(msederSmax)
216 lossedpBS_as_graph = tf.function(lossedpBS)
217
218 #%% Entrenamiento
219
220 # Medición del coste computacional (tiempo)
221 start = time.time()
222
223 # Definición del entrenamiento
224 for epoch in range(epochs):
225     with tf.GradientTape() as tape:
226         # Frontera t=0
227         valcotft0_=u_as_graph(cotft0,tiemft0)
228         tvalcotft0_=u_as_graph(tcotft0,ttiemft0)
229         l1=msedirichlet(valcotft0_,valcotft0)
230         genl1=msedirichlet(tvalcotft0_,tvalcotft0)
231         # Frontera S=0
232         valcotfS0_=u_as_graph(cotfS0,tiemfS0)
233         tvalcotfS0_=u_as_graph(tcotfS0,ttiemfS0)
234         l2=msedirichlet(valcotfS0_,valcotfS0)
235         genl2=msedirichlet(tvalcotfS0_,tvalcotfS0)
236         # Frontera S=Smax
237         l3=msederSmax_as_graph(cotfSmax,tiemfSmax,valfderSmax)
238         genl3=msederSmax_as_graph(tcotfSmax,ttiemfSmax,
            ↪  tvalfderSmax)
239         # Error del interior
240         l4=lossedpBS_as_graph(cotint,tiemint)
241         genl4=lossedpBS_as_graph(tcotint,ttiemint)
242         loss =0.5*(l1+l2+l3)+0.5*l4
            ↪  # Loss total
243         genloss =0.5*(genl1+genl2+genl3)+0.5*l4
```

```
244     g = tape.gradient(loss, model.trainable_weights)
245     opt.apply_gradients(zip(g, model.trainable_weights))
246     loss_values = np.append(loss_values, loss)
247     genloss_values = np.append(genloss_values, genloss)
248     test_pred=model(tf.concat([cottest,tiemtest], axis=1))
249     test_err=np.sqrt(np.mean(((test_pred-valtest)**2)))
250     train_pred=model(tf.concat([cotint,tiemint], axis=1))
251     train_err=np.sqrt(np.mean(((train_pred-valint)**2)))
252     testloss_values = np.append(testloss_values, test_err)
253     trainloss_values = np.append(trainloss_values, train_err)
254     print(epoch,epochs)
255     if epoch % 20 == 0 or epoch == epochs-1:
256         print(f"{epoch:5}, {loss.numpy():.8f}")
257         print(f"{epoch:5}, {test_err:.8f}")
258 
259 # Miro lo que ha tardado y la evolución del loss
260 
261 end = time.time()
262 tiempo = {}
263 tiempo["NN"] = end - start
264 print(f"\ntiempo de entrenamiento: {end-start:.3f}\n")
265 
266 # Grafico el loss
267 plt.semilogy(loss_values[:300], label='Función de pérdidas
    ↪ (entrenamiento)')
268 plt.semilogy(genloss_values[:300], label='Función de pérdidas
    ↪ (validación)')
269 plt.legend()
270 plt.xlabel("Épocas")
271 plt.ylabel("Valor de la función")
272 plt.title("Evolución del la función de pérdidas")
273 plt.show()
274 
275 # Grafico el MSE
276 plt.semilogy(testloss_values, label='Error de generalización
    ↪ (error del conjunto de validación)', color='orange')
277 plt.legend()
278 plt.xlabel("Épocas")
279 plt.ylabel("Error cuadrático medio")
280 plt.title("Evolución del error")
```

```
281 plt.show()
282
283
284 #%% Preparación de datos para el visionado numerico de
    ↪  resultados
285
286 predictions=model(tf.concat([cottest,tiemtest], axis=1))
287 error=predictions-valtest
288
289 #%% Mesh Predicciones
290
291 # Creo el Plot General
292 figure_3d=plt.figure(figsize=(24,6))
293 ax = figure_3d.add_subplot(111, projection='3d')
294
295 X, Y = np.meshgrid(cotintauxtest, tiemintauxtest)
296
297 #Defino la función del eje z
298 fz = predictions.numpy().flatten().reshape(tiemintauxtest.size,
    ↪  cotintauxtest.size)
299
300 #Representamos
301 ax.plot_surface(X, Y, fz, linewidth=0, cmap="coolwarm")
302
303 #Perspectiva
304 ax.view_init(azim=225)
305
306 #Ejes
307 ax.set_xlabel('Cotización (S)')
308 ax.set_ylabel("Tiempo (t)")
309 ax.set_zlabel("F(t,S)")
310
311 #Titulo
312 ax.set_title("Función de red")
313
314 plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
315
316 plt.show()
317
318
```

```
319 #%% Valores de la Call según el modelo de BS (cálculo teórico)
320
321 K = 1  # Strike
322 T = 1   # Vencimiento
323 r = 0.03  # Tasa libre de riesgo
324 sigma = 0.1  # Volatilidad
325
326 X, Y = np.meshgrid(cotintauxtest, tiemintauxtest)
327
328 # Función de Black-Scholes para la opción call
329 def black_scholes_call(S, t, K, T, r, sigma):
330     d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * (T - t)) / (sigma
        ↪  * np.sqrt(T - t))
331     d2 = d1 - sigma * np.sqrt(T - t)
332     call_price = S * norm.cdf(d1) - K * np.exp(-r * (T - t)) *
        ↪  norm.cdf(d2)
333     return call_price
334
335 # Prevengo divisiones por cero en T-t
336 epsilon = 1e-5
337 fz = black_scholes_call(X, Y + epsilon, K, T, r, sigma)
338
339 # Creo el Plot General
340 figure_3d=plt.figure(figsize=(24,6))
341 ax = figure_3d.add_subplot(111, projection='3d')
342
343 #Representamos
344 ax.plot_surface(X, Y, fz, linewidth=0, cmap="coolwarm")
345
346 #Perspectiva
347 ax.view_init(azim=225)
348
349 #Ejes
350 ax.set_xlabel('Cotización (S)')
351 ax.set_ylabel("Tiempo (t)")
352 ax.set_zlabel("F(t,S)")
353
354 #Titulo
355 ax.set_title("Valor de la Call Europea según el modelo de BS")
356
```

```
357 plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
358
359 plt.show()
360
361
362 #%% Mesh Errores
363
364 # Creo el Plot General
365 figure_3d=plt.figure(figsize=(24,6))
366 ax = figure_3d.add_subplot(111, projection='3d')
367
368 X,Y=np.meshgrid(cotintauxtest,tiemintauxtest)
369
370 #Defino la función del eje z
371 fz=error.numpy().flatten().reshape(tiemintauxtest.size,
    ↪  cotintauxtest.size)
372
373
374 #Representamos
375 ax.plot_surface(X, Y, fz, linewidth=0, cmap="coolwarm")
376
377 #Perspectiva
378 ax.view_init(azim=225)
379
380 #Ejes
381 ax.set_xlabel('Cotización (S)')
382 ax.set_ylabel("Tiempo (t)")
383 ax.set_zlabel("Error")
384
385 #Titulo
386 ax.set_title("Error de la red")
387
388 plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
389
390 plt.show()
391
392
393 #%% Griegas
394
395 derS1,derS2,dert1=calculoderivadas(cottest,tiemtest)
```

```
396
397 ## Delta
398 # Creo el Plot General
399 figure_3d=plt.figure(figsize=(24,6))
400 ax = figure_3d.add_subplot(111, projection='3d')
401
402 X,Y=np.meshgrid(cotintauxtest,tiemintauxtest)
403
404 #Defino la función del eje z
405 fz=derS1.numpy().flatten().reshape(tiemintauxtest.size,
    ↪  cotintauxtest.size)
406
407 #Representamos
408 ax.plot_surface(X, Y, fz, linewidth=0, cmap="coolwarm")
409
410 #Perspectiva
411 ax.view_init(azim=225)
412
413 #Ejes
414 ax.set_xlabel('Cotización (S)')
415 ax.set_ylabel("Tiempo (t)")
416
417 #Titulo
418 ax.set_title("Delta")
419
420 plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
421
422 plt.show()
423
424 ## Gamma
425 # Creo el Plot General
426 figure_3d=plt.figure(figsize=(24,6))
427 ax = figure_3d.add_subplot(111, projection='3d')
428
429 X,Y=np.meshgrid(cotintauxtest,tiemintauxtest)
430
431 #Defino la función del eje z
432 fz=derS2.numpy().flatten().reshape(tiemintauxtest.size,
    ↪  cotintauxtest.size)
433
```

```
434 #Representamos
435 ax.plot_surface(X, Y, fz, linewidth=0, cmap="coolwarm")
436
437 #Perspectiva
438 ax.view_init(azim=225)
439
440 #Ejes
441 ax.set_xlabel('Cotización (S)')
442 ax.set_ylabel("Tiempo (t)")
443
444 #Titulo
445 ax.set_title("Gamma")
446
447 plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
448
449 plt.show()
```