UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE MATEMÁTICAS

DEPARTAMENTO DE ANÁLISIS MATEMÁTICO Y MATEMÁTICA APLICADA

TRABAJO DE FIN DE GRADO

Resolución de ecuaciones diferenciales no lineales utilizando técnicas de Deep Learning

Supervisor: Gerardo Enrique Oleaga Apadula

Ricardo Buitrago Ruiz

Doble grado en Matemáticas y Física

Junio de 2021

Abstract:

In Finance, developing accurate models that incorporate possible threats of the markets is essential in avoiding exposure to risk, which is detrimental to both financial entities and the society. However, working with such models involve adding non-linear terms to the equations that govern financial instruments' prices, like the Black-Scholes equation. Such non-linearities make it very hard to solve these equations with traditional methods, like Monte Carlo methods. Additionally, in Finance we are interested in the gradient of the solution, which is something that many traditional methods can not calculate. In this thesis, we use Deep Learning to deal with such non-linearities and find numerical approximations to the solutions of partial differential equations and their gradients. In particular, we will deal with the default risk (the risk of a company not paying their financial obligations), and we will develope an innovative model to incorporate this risk into the Black-Scholes equation. Furthermore, we will also deal with exotic instruments that multiply the instruments' value by a number δ on the ocurrence of a certain event, introducing a new variant of the Black-Scholes equation which we call Black-Scholes equation with multipliers. These innovative models allow a more precise representation of the real markets, thus making it easier to avoid risk and bringing benefits to financial institutions and society.

Contents

1 Introducción

Las ecuaciones parabólicas semilineales son una familia de ecuaciones diferenciales que modelizan muchos problemas en física y finanzas [1]. Estas ecuaciones involucran términos no lineales que dificultan su resolución con los métodos numéricos más utilizados, como el método de Monte Carlo. Además, dichos métodos no permiten calcular el gradiente de la solución, que es una información muy útil en ciertas disciplinas, como en Finanzas [7]. En este TFG, analizaremos un método para resolver este tipo de ecuaciones conocido como método Deep BSDE (Deep Backward Stochastic Differential Equation), que está basado en el artículo de Jiequn Han, Arnulf Jentzen, y Weinan E [1]. La clave del método es aplicar el Machine Learning a la resolución de ecuaciones diferenciales, basándose en la relación de éstas con procesos estocásticos. Se utilizarán Redes Neuronales para aproximar el gradiente de las soluciones, y se usarán algoritmos de optimización de Machine Learning para obtener estimadores de la solución de la ecuación. Así, obtendremos valores precisos para la solución y su gradiente.

Nos centraremos en aplicar este método a ecuaciones financieras novedosas que modelizaremos en este TFG, que son variaciones de la ecuación de Black-Scholes [7] y modelizan instrumentos financieros exóticos. Deduciremos dichas ecuaciones en este TFG y las resolveremos con el método Deep BSDE. También discutiremos el sentido financiero de las soluciones y la posible utilidad que pueden tener para la sociedad.

2 Objetivos y Plan de Trabajo

Los objetivos de este trabajo son los siguientes:

- Explicar la implementación del método Deep BSDE [1] en Keras [5] y Tensorflow [6].
- Analizar los estimadores de la solución de la ecuación parabólica semilineal del método Deep BSDE en casos donde la solución exacta es conocida (ecuación de difusión y ecuación de Black-Scholes estándar).
- Deducir los términos adicionales que se deben introducir en la ecuación de Black-Scholes para modelizar diferentes factores, como la quiebra del vendedor de una opción (default risk), o la posibilidad de que el valor de un instrumento se multiplique debido a la ocurrencia de algún evento en el mercado (ecuación de Black-Scholes con multiplicadores).
- Resolver dichas ecuaciones, que son semilineales, y comparar sus soluciones al caso de Black-Scholes estándar.

Para conseguir dichos objetivos, primero nos familiarizaremos con el método Deep BSDE aplicándolo a un caso muy sencillo: la ecuación de difusión. Indicaremos la implementación del método paso por paso, y también haremos referencia a los códigos utilizados, disponibles en un respositorio de GitHub creado para este TFG (link). Después, utilizaremos argumentos financieros para trabajar con la ecuación de Black-Scholes y deducir las ecuaciones con los términos adicionales. Para ello, haremos referencia a teoremas y resultados de Matemáticas Financieras. Finalmente, haremos un análisis de los instrumentos financieros exóticos utilizados en la actualidad, y cómo estos nuevos modelos pueden ayudar a valorar estos instrumentos

3 Modelo de Prueba: La ecuación de Difusión

En esta sección, explicaremos la intuición detrás el método Deep BSDE con un ejemplo lineal muy sencillo: la ecuación de difusión. Conociendo la solución analítica exacta, testearemos la eficiencia del método y lo compararemos con otros métodos. En próximas secciones trabajaremos en el caso verdaderamente interesante: las ecuaciones diferenciales parabólicas semilineales.

3.1 Intuición del método Deep BSDE

3.1.1 Método de Monte Carlo en el caso de la ecuación de difusión homogénea

Consideremos el problema de valores iniciales[1]:

$$\begin{cases} \partial_t u + \partial_{xx} u = 0 & (x,t) \in \mathbb{R} \times [t_0, T] \\ u(x,T) = u_T(x) & x \in \mathbb{R} \end{cases} \tag{1}$$

Este problema es equivalente a la ecuación de difusión $\partial_t u - \partial_{xx} u = 0$ con condiciones iniciales $u(x,t_0) = u_T(x)$ si hacemos el cambio $t \longrightarrow t_0 + (T-t)$. Si $u_T(x) : \mathbb{R} \longrightarrow \mathbb{R}$ es una función que cumple:

1. $u_T(x)$ tiene un número finito de discontinuidades de salto.
2. Existen constantes positivas a y c tales que: $|u_T(x)| \leq ce^{ax^2}$.

Entonces, el problema tiene una solución y es única (ver Corolario 2.3 de [2]). Supongamos que queremos conocer el valor de u en un punto ξ en el momento inicial t_0. Es decir, queremos conocer:

$$u_0 := u(\xi, t_0) \tag{2}$$

Consideremos el proceso estocástico $\{X_t\}$ que satisface la siguiente ecuación diferencia estocástica[2] [8] en $t \in [t_0, T]$:

$$\begin{aligned} X_{t_0} &= \xi \\ dX_t &= \sqrt{2} dW_t \end{aligned} \tag{3}$$

Donde formalmente W_t es un proceso de Wiener [8], que podemos entender como $dW_t \equiv \sqrt{dt} Z_t$ y $Z_t \sim \mathcal{N}(0,1)$, donde las Z_t son independientes para intervalos temporales disjuntos. Si evaluamos la solución u de 1 en el la trayectoria del proceso (X_t, t), generamos un nuevo proceso en $[t_0, T]$:

$$U_t : t \to u\left(X_t, t\right) \tag{4}$$

Donde $U_{t_0} = u_0$ Aplicando el Lema de Itô a este proceso, tenemos:

$$dU_t = \left(\partial_t u\left(X_t, t\right) + \partial_{xx} u\left(X_t, t\right)\right) dt + \sqrt{2} \partial_x u\left(X_t, t\right) dW_t \tag{5}$$

[1] Empleamos la notación $\partial_t = \frac{\partial}{\partial t}$ y $\partial_{xx} = \frac{\partial^2}{\partial x^2}$.

[2] La raíz de 2 que aparece en la fórmula 3 se introduce para poder cancelar términos más adelante en la expresión de dU_t al aplicar la fórmula del Lema de Itô [8]. Para el análisis posterior que haremos de nuestros algoritmos, dicha $\sqrt{2}$ la entenderemos como una desviación típica σ de los caminos.

Y aplicando que u es solución de 1:

$$dU_t = \sqrt{2}\partial_x u\left(X_t, t\right) dW_t \tag{6}$$

Discretizemos ahora el tiempo entre t_0 y T con N intervalos temporales $\Delta t = \frac{T-t_0}{N}$ y definamos la partición temporal:

$$t_i = t_{i-1} + \Delta t \quad i = 1, ..., N \tag{7}$$

Usaremos la notación de subíndice i para denotar la evaluación de estos caminos en el tiempo t_i. Es decir:

$$\begin{aligned} W_i &\overset{\text{def}}{=} W_{t_i} \\ X_i &\overset{\text{def}}{=} X_{t_i} \\ U_i &\overset{\text{def}}{=} U_{t_i} \end{aligned} \tag{8}$$

El proceso de Wiener lo podemos generar en esta discretización temporal de forma exacta:

$$\Delta W_i = \sqrt{\Delta t} Z_i, \quad Z_i \sim \mathcal{N}(0,1) \text{ independientes} \quad i = 0, ..., N-1 \tag{9}$$

Si N es grande manteniendo $N\Delta t = T - t_0$ ($\Delta t \longrightarrow 0$), utilizando las ecuaciones 3, 9 y 6 podemos generar los procesos discretos:

$$\begin{aligned} X_0 &= \xi \\ X_{i+1} - X_i := \sqrt{2}\Delta W_i &= \sqrt{2}\sqrt{\Delta t} Z_i, \quad Z_i \sim \mathcal{N}(0,1) \end{aligned} \tag{10}$$

$$\begin{aligned} U_0 &= u_0 \\ U_{i+1} - U_i &\approx \sqrt{2}\sqrt{\Delta t}\partial_x u\left(X_i, t_i\right) Z_i \end{aligned} \tag{11}$$

Donde $i = 0, ..., N-1$. Hasta aquí, estamos empleado las letras X y U para describir los procesos exactos en tiempo continuo definidos según 3 y 4. A partir de ahora, definiremos unas variables $\tilde{X}$ y $\tilde{U}$, donde la tilde indica que estamos en la aproximación[3] de tiempo discreto dada por las ecuaciones 10 y 11:

$$\begin{aligned} \tilde{X}_0 &= \xi \\ \tilde{X}_{i+1} &= \tilde{X}_i + \sqrt{2}\sqrt{\Delta t} Z_i \quad i = 0, ..., N-1 \end{aligned} \tag{12}$$

Para para simplificar la notación, utilizaremos $\{\tilde{X}_i\}$ para denotar $\{\tilde{X}_i\}_{i=0}^N$. Similarmente, definimos:

$$\begin{aligned} \tilde{U}_0 &= u_0 \\ \tilde{U}_{i+1} &= \tilde{U}_i + \sqrt{2}\sqrt{\Delta t}\partial_x u\left(\tilde{X}_i, t_i\right) Z_i \quad i = 0, ..., N-1 \end{aligned} \tag{13}$$

Utilizando la ecuación 11, tratamos a las U con tilde como una aproximación de las U exactas, que será una igualdad cuando $\Delta t \longrightarrow 0$:

$$\tilde{U}_i \approx U_i \quad i = 0, ..., N \tag{14}$$

[3]En este caso, para la X, debido a la forma tan sencilla de 3, el proceso continuo y la aproximación en tiempo discreto coinciden $\tilde{X}_i = X_i$. Pero más adelante en el TFG generaremos procesos estocásticos más complejos donde dichas expresiones no coincidan. Por ello, introducimos desde ahora la notación con tildes en las X (aunque parezca redundante) para mantener la coherencia con el resto de la tesis. Nótese que esto sólo es válido para la X, ya que el proceso continuo U no coincide con el discreto $\tilde{U}$ que definiremos a continuación.

En particular:

$$\tilde{U}_N \approx U_N = u(X_N, T) = u_T(X_N) \approx u_T(\tilde{X}_N) \tag{15}$$

Por tanto, dado un proceso $\{\tilde{X}_i\}$ con condición inicial ξ, y su proceso asociado $\{\tilde{U}_i\}$, podemos aproximar el valor del camino $\tilde{U}_N$ a partir de la fórmula 15. A partir de esta observación, describiremos un método para hallar u_0, normalmente conocido como método de Monte Carlo. Para ello, observemos que, hallando la esperanza en ambos lados de la ecuación 13:

$$E[\tilde{U}_{i+1} - \tilde{U}_i] = \sqrt{2}\sqrt{\Delta t} E[\partial_x u\left(\tilde{X}_i, t_i\right) Z_i] \tag{16}$$

Donde dicha esperanza se toma respecto a las variables aleatorias $Z_0, ..., Z_{N-1}$ que generan el proceso de Wiener. Para hallar la esperanza del lado derecho, hacemos uso de la identidad válida para cualesquiera variables aleatorias X, Y independientes: $E[XY] = E[X]E[Y]$. Entonces:

$$E\left[\partial_x u\left(\tilde{X}_i, t_i\right) Z_i\right] \quad = E\left[\partial_x u\left(\tilde{X}_i, t_i\right)\right] \underbrace{E[Z_i]}_{0} = 0 \tag{17}$$

Donde hemos usado que $Z_i \sim \mathcal{N}(0,1)$ es independiente de $\tilde{X}_i$ (dado $\tilde{X}_i = x$, el valor de Z_i no está determinado y sigue siendo una normal $\mathcal{N}(0,1)$), y por tanto $\partial_x u\left(\tilde{X}_i, t_i\right)$, que es una función de X_i, es independiente de Z_i). A partir de este cálculo, la ecuación 16 pasa a ser:

$$E[\tilde{U}_{i+1} - \tilde{U}_i] = 0 \tag{18}$$

De nuevo a partir de la ecuación 13, llegamos a que:

$$\tilde{U}_N - \tilde{U}_0 = \sum_{i=0}^{N-1} \tilde{U}_{i+1} - \tilde{U}_i \tag{19}$$

Donde recordamos que $\tilde{U}_0 = u_0$ es un número, que no depende de ninguna variable aleatoria. Tomando la esperanza en ambos lados y usando 18:

$$\tilde{U}_0 = E[\tilde{U}_N] \tag{20}$$

Y por la ecuación 15, se llega a que:

$$u_0 = E\left[\tilde{U}_N\right] \approx E\left[u_T(\tilde{X}_N)\right] \tag{21}$$

Pero el lado de la derecha se puede calcular, ya que u_T es parte de la información de nuestro problema y la esperanza se puede aproximar mediante una muestra generada de $\{\tilde{X}_i\}$. Esto se conoce como método de **método de Monte Carlo**, que podemos resumir en los siguientes puntos:

1. Tomamos M grande y generamos M caminos con la estructura de 12: $\{\tilde{X}_i\}^{(m)},\ m = 0, ..., M-1$.

2. Aproximamos:

$$E\left[u_T(\tilde{X}_N)\right] \approx \frac{1}{M}\sum_{m=0}^{M-1} u_T\left(\tilde{X}_N^{(m)}\right) \tag{22}$$

3. Aplicamos la ecuación 21 y:

$$u_0 \approx \frac{1}{M}\sum_{m=0}^{M-1} u_T\left(\tilde{X}_N^{(m)}\right) \tag{23}$$

Con este sencillo método, se puede calcular el valor inicial u_0 de forma bastante precisa. Pero veamos ahora por qué este método de Monte Carlo no se puede aplicar de forma tan sencilla en el caso inhomogéneo.

3.1.2 Método de Monte Carlo en el caso de la ecuación de difusión no homogénea

Ahora consideremos el siguiente problema:

$$\begin{cases} \partial_t u + \partial_{xx} u = f(u,x,t) & (x,t) \in \mathbb{R}\times[t_0,T] \\ u(x,T) = u_T(x) & x \in \mathbb{R} \end{cases} \tag{24}$$

Aplicando ahora el corolario 2.3 de [2], el problema tiene solución única (suponiendo que la f tiene un comportamiento suficientemente bueno). De nuevo, queremos hallar $u_0 = u(\xi, t_0)$.

Podemos crear un proceso estocástico $\{X_t\}$ con la misma estructura que en caso homogéneo (ecuación 3). Sin embargo, si ahora definimos:

$$U_t : t \to u\left(X_t, t\right) \tag{25}$$

Con u solución de 24, aplicando el Lema de Itô obtenemos:

$$\begin{aligned} dU_t &= \underbrace{\left(\partial_t u\left(X_t,t\right) + \partial_{xx} u\left(X_t,t\right)\right)}_{f(U_t,X_t,t)} dt + \partial_x u\left(X_t,t\right) dW_t \\ &= f(U_t, X_t, t)dt + \partial_x u\left(X_t,t\right) dW_t \end{aligned} \tag{26}$$

Si discretizamos el tiempo como en la sección anterior, podemos definir de nuevo variables:

$$\Delta\tilde{W}_i = \sqrt{\Delta t} Z_i, \quad Z_i \sim \mathcal{N}(0,1) \text{ independientes} \quad i = 0, ..., N-1 \tag{27}$$

$$\begin{aligned} \tilde{X}_0 &= \xi \\ \tilde{X}_{i+1} - \tilde{X}_i &= \sqrt{2}\sqrt{\Delta t} Z_i \quad i = 0, ..., N-1 \end{aligned} \tag{28}$$

$$\begin{aligned} \tilde{U}_0 &= u_0 \\ \tilde{U}_{i+1} - \tilde{U}_i &= f(\tilde{U}_i, \tilde{X}_i, t_i)\Delta t + \sqrt{2}\sqrt{\Delta t}\partial_x u\left(\tilde{X}_i, t_i\right) Z_i \quad i = 0, ..., N-1 \end{aligned} \tag{29}$$

Y de nuevo, tomando N grande y $\tilde{U}_0 = u_0$ se tiene:

$$\tilde{X}_i = X_i \quad i = 0, ..., N \tag{30}$$

$$\tilde{U}_i \approx U_i \quad i = 0, ..., N \tag{31}$$

El problema es que ahora no podemos aplicar el método de Monte Carlo anterior, ya que:

$$\begin{aligned} E\left[\tilde{U}_{i+1} - \tilde{U}_i\right] &= E\left[f(\tilde{U}_i, \tilde{X}_i, t_i)\right]\Delta t + \sqrt{2}\sqrt{\Delta t}\underbrace{E\left[\partial_x u\left(\tilde{X}_i, t_i\right) Z_i\right]}_{0} \\ &= E\left[f(\tilde{U}_i, \tilde{X}_i, t_i)\right]\Delta t \neq 0 \end{aligned} \tag{32}$$

En los casos en los que la f dependa de u, el método ya no funciona, debido a que:

$$u_0 = E\left[u_T(\tilde{X}_N)\right] + \sum_{i=0}^{N-1} E\left[f(\tilde{U}_i, \tilde{X}_i, t_i\right] \tag{33}$$

Pero como no conocemos los valores $\tilde{U}_i$ (dependen de u_0 y de $\partial_x u$ según la ecuación 29), no podemos aplicar esta fórmula. El método que discutiremos en este TFG es una alternativa al método de Monte Carlo que funciona para cualquier tipo de ecuación parabólica semilineal. Se basa en utilizar la fórmula, obtenida a partir de 29:

$$\tilde{U}_N = \tilde{U}_0 + \sum_{i=0}^{N-1} f(\tilde{U}_i, \tilde{X}_i, t_i)\Delta t + \sqrt{2}\sqrt{\Delta t}\partial_x u\left(\tilde{X}_i, t_i\right) Z_i \tag{34}$$

Con esta fórmula, si conociésemos los valores exactos del gradiente $\partial_x u(\tilde{X}_i, t_i)$, podríamos definir funcionales $U^\star$ que dependiesen de un parámetro θ_{u_0} y del proceso $\{\tilde{X}_i\}$ de la siguiente forma:

$$\begin{aligned} U_0^\star[\{\tilde{X}_i\}|\theta_{u_0}] &= \theta_{u_0} \\ U_i^\star[\{\tilde{X}_i\}|\theta_{u_0}] &= U_{i-1}^\star[\{\tilde{X}_i\}|\theta_{u_0}] + f(U_i^\star[\{\tilde{X}_i\}|\theta_{u_0}], \tilde{X}_i, t_i)\Delta t + \sqrt{2}\sqrt{\Delta t}\partial_x u\left(\tilde{X}_i, t_i\right) Z_i \quad i = 1, ..., N \end{aligned} \tag{35}$$

Donde usamos la estrella para diferenciar estos funcionales de los procesos discretos. Estos funcionales son iguales al proceso discreto $\tilde{U}_i$ cuando $\theta_{u_0} = u_0$, es decir $U_i^\star[\cdot|u_0] = \tilde{U}_i$, y por tanto, $U_N^\star[\{\tilde{X}_i\}|u_0] \approx u_T(\tilde{X}_N)$. Sin embargo, para $\theta \neq \tilde{U}_0$, se genera un proceso que difiere significativamente del exacto, debido a que empieza en un punto que no es el correcto. Si se define una función de coste:

$$J(\theta_{u_0}) = E\left[\left(U_N^\star[\{\tilde{X}_i\}|\theta_{u_0}] - u_T(\tilde{X}_N)\right)^2\right] \tag{36}$$

Entonces el mínimo de la función se obtendrá en u_0, por lo que nuestro problema estaría resuelto haciendo $u_0 = \text{argmin}_{\theta_{u_0}}(J(\theta_{u_0})$. Sin embargo, en general no conocemos el gradiente $\partial_x u$. La clave del método Deep BSDE consiste en aproximar este gradiente mediante redes neuronales, y generar funciones $U_i^\star$ que dependen de θ_{u_0} y también de los pesos de dichas redes neuronales. De nuevo, los valores que minimicen la función 36 serán la solución a nuestro problema. Esto es lo que hacemos a continuación.

3.1.3 El método Deep BSDE aplicado a la ecuación de difusión homogénea

Aunque el método de Monte Carlo no funciona para la ecuación de difusión no homogénea, nos ofrece una idea para formular otro método para resolver el problema 24. Con la notación anterior, se tiene que:

$$\tilde{U}_N = \tilde{U}_0 + \sum_{i=0}^{N-1} \left[f(\tilde{U}_i, \tilde{X}_i, t_i)\Delta t + \sqrt{2}\sqrt{\Delta t}\partial_x u\left(\tilde{X}_i, t_i\right) Z_i \right] \tag{37}$$

Y aplicando la aproximación 31:

$$\tilde{U}_N \approx U_N = u_T(X_N) \approx u_T(\tilde{X}_N) \tag{38}$$

El problema ahora reside en hallar los valores de u_0 y $\partial_x u(x, t_i)$ $i = 0, ..., N-1$. Para el caso $i = 0$, sólo es necesario conocer $\partial_x u(\xi, t_0)$, ya que $\tilde{X}_0 = \xi$ para cualquier camino. Al valor $\partial_x u(\xi, t_0)$ lo denotamos como $\partial_x u_0$, y a los valores $\partial_x u(x, t_i)$ $i = 1, ..., N-1$ los denotamos como $\partial_x u_i(x)$.

La clave del método Deep BSDE se basa en utilizar un funcional que tenga un mínimo en los valores u_0, $\partial_x u(\xi, t_0)$ y $\partial_x u(x, t_i)$, $i = 1, ..., N-1$. Para ello, asignamos variables a cada parámetro que queremos determinar: las variables reales θ_{u_0} y $\theta_{\partial_x u_0}$ para u_0 y $\partial_x u_0$:

$$\begin{array}{rcl} u_0 & \longrightarrow & \theta_{u_0} \\ \partial_x u_0 & \longrightarrow & \theta_{\partial_x u_0} \end{array} \tag{39}$$

Y las funciones que actuarán como variables del funcional $h_1(x), ..., h_{N-1}(x)$:

$$\partial_x u_i(x) \longrightarrow h_i(x) \tag{40}$$

Con ellas, podemos definir los funcionales, que dependen del camino $\{\tilde{X}_i\}$:

$$\begin{array}{rl} U_0^\star \left[\{\tilde{X}_i\} | \theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1} \right] & = \theta_{u_0} \\ U_1^\star \left[\{\tilde{X}_i\} | \theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1} \right] & = U_0^\star + f(U_0^\star, \tilde{X}_0, t_0)\Delta t + \sqrt{2}\sqrt{\Delta t} \cdot \theta_{\partial_x u_0} \cdot Z_0 \\ U_{i+1}^\star \left[\{\tilde{X}_i\} | \theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1} \right] & = U_i^\star + f(U_i^\star, \tilde{X}_i, t_i)\Delta t + \sqrt{2}\sqrt{\Delta t} \cdot h_i\left(\tilde{X}_i\right) \cdot Z_i \quad i = 1, ..., N-1 \end{array} \tag{41}$$

Hemos introducido la notación $U^\star$ para distinguirla de $\tilde{U}$ (caminos generados con los valores iniciales exactos y los gradientes exactos pero utilizando la aproximación discreta) y de U (caminos exactos). En el lado derecho de las igualdades segunda y tercera anteriores, cuando escribimos $U_i^\star$ nos referimos a $U_i^\star \left[\{\tilde{X}_i\} | \theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1} \right]$. Por las ecuaciones 37 y 38 se tiene que:

$$U_N^\star \left[\{\tilde{X}_i\} | u_0, \partial_x u_0, \partial_x u_1(x), ..., \partial_x u_N(x) \right] = \tilde{U}_N \approx U_N \approx u_T(\tilde{X}_N) \tag{42}$$

Donde las aproximaciones se vuelven igualdades cuando $\Delta t \longrightarrow 0$ ($N \longrightarrow \infty$). Entonces, tiene sentido definir la siguiente función de coste:

$$J\left[\theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1}\right] = E\left[\left(U^\star \left[\{\tilde{X}_i\} | \theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1} \right] - u_T(\tilde{X}_N) \right)^2 \right] \tag{43}$$

Donde la esperanza se halla con respecto a las variables aleatorias Z_i que generan los procesos discretos $\{\tilde{X}_i\}$. Por tanto, el método Deep BSDE consiste en:

1. Tomar M grande y generar caminos $\{\tilde{X}_i\}^{(m)}$ con $m = 0, ..., M-1$ según 28 .
2. Aproximar la esperanza de la ecuación 43 como:

$$\begin{aligned} E\left[\left(U^\star\left[\{\tilde{X}_i\}|\theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1}\right] - u_T(\tilde{X}_N^{(m)})\right)^2\right] \approx \\ \approx \frac{1}{M}\sum_{m=0}^{M-1}\left(U^\star\left[\{\tilde{X}_i\}^{(m)}|\theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1}\right] - u_T(\tilde{X}_N^{(m)})\right)^2 \end{aligned} \tag{44}$$

3. Tomar como estimadores de nuestros parámetros de interés:

$$(\hat{u}_0, \partial_x\hat{u}_0, \partial_x\hat{u}_1, ..., \partial_x\hat{u}_{N-1}) = \text{argmin} J\left[\theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1}\right] \tag{45}$$

Todavía queda otro paso esencial para el método Deep BSDE, que es cómo optimizar el funcional J. Aquí es donde entra el Machine Learning en el método. La clave es intentar aproximar en cada t_i la función gradiente $\partial_x u(x, t_i)$ como una red neuronal "feedforward" [4]. Es decir, tomamos las funciones h_i como redes neuronales con input x. Su output $h_i(x|\theta_i)$ lo trataremos como una estimación de $\partial_x u(x, t_i)$, donde denotamos θ_i a los pesos de cada una de estas redes neuronales. Por tanto, estamos aproximando:

$$\partial_x u(x, t_i) \approx h_i(x|\theta_i) \tag{46}$$

Así, nuestra funcional coste se convierte en una función de variables:

$$J\left[\theta_{u_0}, \theta_{\partial_x u_0}, h_1, ..., h_{N-1}\right] = J\left[\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}\right] \tag{47}$$

Que se puede aproximar mediante un método iterativo como el "Adam" (Adaptive Moment Estimation) [3], que es típico en redes neuronales. En ese caso, nuestros estimadores para los parámetros de la ecuación 45 serán iguales a los pesos en la última iteración del algoritmo de optimización.

3.2 Implementación del método Deep BSDE para la ecuación de difusión homogénea

Una vez que hemos explicado la teoría matemática detrás del método, vamos a explicar cómo implementarlo en el siguiente problema de prueba[4]:

$$\begin{cases} \partial_t u + \partial_{xx} u = 0 & (x, t) \in \mathbb{R} \times [t_0, T] \\ u(x, T) = \sin(x)e^T & x \in \mathbb{R} \end{cases} \tag{48}$$

De nuevo por el corolario 2.3 de [2], este problema tiene una solución única, que es:

$$u(x, t) = \sin(x)e^t \tag{49}$$

$$\partial_x u(x, t) = \cos(x)e^t \tag{50}$$

Utilizaremos esta solución exacta para evaluar el error de los estimadores del método Deep BSDE en este caso sencillo. Para hallar dichos estimadores, resumimos lo que explicamos el la sección anterior en los siguientes pasos:

[4]Este problema involucra a una ecuación homogénea. Más adelante en el TFG trataremos ecuaciones no homogéneas con términos no lineales, pero comenzamos con este ejemplo sencillo para ganar intuición sobre el método.

1. Tomamos N grande y discretizamos el tiempo:

$$\begin{array}{c} \Delta t = \frac{T-t_0}{N} \\ t_{i+1} = t_i + \Delta t \quad i = 0, ..., N-1 \end{array} \tag{51}$$

2. Generamos dos parámetros entrenables en una red neuronal $\theta_{u_0} \sim u(\xi, t_0)$, $\theta_{\partial_x u_0} \sim \partial_x u(\xi, t_0)$.

3. Tomamos M grande y generamos las variables aleatorias $\{Z_i\}^{(m)}$ para $m = 0, ..., M-1$, con $Z_i \sim \mathcal{N}(0,1)$ independientes. Este será nuestro primer input de la red neuronal:

$$\text{input}_1 = \{Z_i\}^{(m)} \quad m = 0, ..., M-1 \tag{52}$$

   Y con estos valores, generamos los caminos $\{\tilde{X}_i\}^{(m)}$ según 28 para $m = 0, ..., M-1$, que será nuestro segundo input:

$$\text{input}_2 = \{\tilde{X}_i\}^{(m)} \quad m = 0, ..., M-1 \tag{53}$$

4. Para cada t_i con $i = 1, ..., N-1$, generamos una red neuronal que aproxime $\partial_x u(x, t_i)$. A cada una de esas redes la llamaremos $h_i(\cdot|\theta_i)$, donde, como ya comentamos, θ_i son los pesos de la red neuronal. Es decir:

$$h_i(x|\theta_i) \approx \partial_x u(x, t_i) \tag{54}$$

   Esta ecuación en principio es válida para cualquier x, pero como veremos sólo será buena la aproximación para x cerca del punto inicial ξ. Para cada camino $\{\tilde{X}_j\}^{(m)}_{j=0,...,N}$, la red neuronal $h_i(\cdot|\theta_i)$ toma como input $\tilde{X}_i^{(m)}$ y su output será una estimación del gradiente en $\tilde{X}_i^{(m)}$: $h_i(\tilde{X}_i^{(m)}|\theta_i) \approx \partial_x u(\tilde{X}_i^{(m)}, t_i)$

5. Para cada i, podemos estimar el valor de $u(\tilde{X}_N^{(m)}, t_i)$ aplicando la fórmula 41 a $\{\tilde{X}_i\}^{(m)}$ con nuestras funciones $h_i(\cdot|\theta_i)$. Con ese valor, basándonos en la fórmula 43, generamos una función de coste para cada m:

$$J^{(m)}\left[\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}\right] = \left(U^\star\left[\{\tilde{X}_i^{(m)}\}|\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}\right] - u_T(\tilde{X}_N^{(m)})\right)^2 \tag{55}$$

   Y la función de coste final la obtenemos sumando a todos los caminos:

$$J\left[\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}\right] = \frac{1}{M}\sum_{m=0}^{M-1}\left(U^\star\left[\{\tilde{X}_i^{(m)}\}|\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}\right] - u_T(\tilde{X}_N^{(m)})\right)^2 \tag{56}$$

6. Minimizamos dicha función de coste con un algoritmo iterativo como el Adam, y así obtenemos los estimadores $\hat{u}_0, \partial_x\hat{u}_0, \partial_x\hat{u}_1, ..., \partial_x\hat{u}_{N-1}$.

3.3 Resultados de la implementación del método Deep BSDE para la ecuación de difusión homogénea

Ahora analizaremos los resultados del método deep BSDE aplicado al problema 48 de la sección anterior, con $\xi = 1$, $t_0 = 0$, $T = 1$ y $N = 10$. El resto de detalles de la implementación y los hiperparámetros escogidos están explicados en el apéndice 8.2.

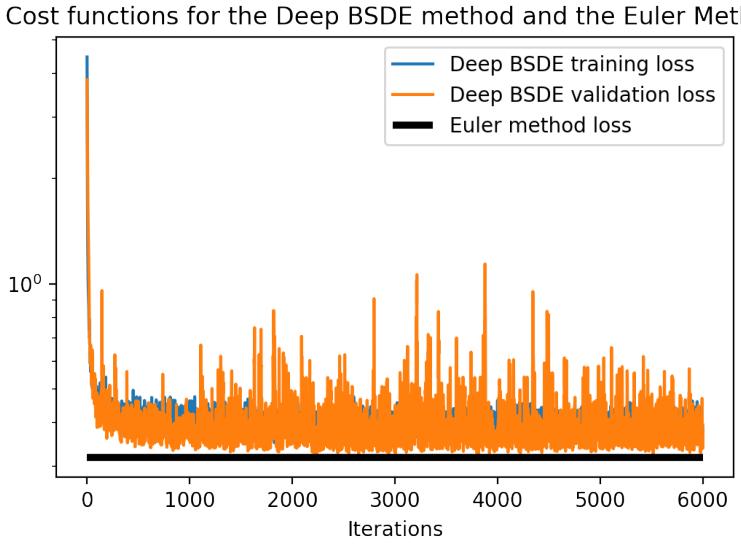

Figure 1: Funciones de coste para el método de Euler y para el método Deep BSDE. Los datos se han dividido en un set de entrenamiento (con el que se entrena la red y en cada iteración se evalúa la función de coste) y un set de validación (con el que sólo se evalúa la función de coste) con porcentajes 50%-50%. Se ha usado escala logarítmica en Y. Para un análisis de la función de coste, ver la sección 8.2.1.

En primer lugar analizaremos la función de coste 56. Para ello, nos aprovecharemos de que conocemos la solución exacta al problema, y compararemos la función de coste del método Deep BSDE con la función de coste de un método auxiliar que nos servirá como punto de referencia. Dicho método será el método de Euler, que consiste en suponer que conocemos exactamente las funciones gradientes en los puntos t_i . Es decir, $h_i(x)$ deja de ser una aproximación en la ecuación 41 y pasa a ser un valor exacto conocido, dado por la ecuación 50. Explícitamente, para calcular la estimación de $u(\tilde{X}_N, T)$ según el método de Euler, definimos:

$$\begin{aligned} \tilde{u}_E(\tilde{X}_0, t_0) &= u_0 \\ \tilde{u}_E(\tilde{X}_{i+1}, t_{i+1}) &= \tilde{u}_E(\tilde{X}_i, t_i) + u_x(\tilde{X}_i, t_i)\sqrt{\Delta t}\sqrt{2}Z_i \quad i = 0, ..., N-1 \end{aligned} \tag{57}$$

Donde estamos asumiendo que conocemos el valor inicial u_0 y los valores de los gradientes $\partial_x u(\cdot, t_i)$, en este caso dados por la ecuación 50 ($\partial_x u(\cdot, t_i) = \cos(\cdot)e^{t_i}$). Entonces nuestra estimación del valor final será $\hat{u}_E(\tilde{X}_N, t_N)$, y la función de coste del método de Euler será:

$$J_E = \frac{1}{M}\sum_{m=0}^{M-1}\left(\tilde{u}_E(\tilde{X}_N^{(m)}, T) - u_T(\tilde{X}_T^{(m)})\right)^2 \tag{58}$$

Naturalmente, la eficiencia de nuestro método está limitada por la eficiencia del método de Euler, ya que con Euler se conocen los valores exactos del gradiente $\partial_x u(\tilde{X}_i, t_i)$, que son necesarios para calcular 41. Es de esperar que nuestro método aproxime bien las funciones gradientes $h_i(\cdot|\theta_i)$, por lo que deberíamos obtener una función de coste cercana al método de Euler.

En la figura 1 se puede observar que nuestro método Deep BSDE efectivamente alcanza un coste cercano al coste del método de Euler. Dicho coste no depende de la iteración (el método de Euler no es iterativo, sólo se muestra la línea vertical para comparar con el método Deep BSDE) y es igual a $J_{\text{Euler}} = 0.319$. Por otro lado, para el método Deep BSDE, la media de la función de coste sobre el set de validación en las últimas 20 iteraciones es $J = 0.375$, que es una diferencia relativa de un 17.6%.

Dicha diferencia es considerable, pero hay que tener en cuenta que el método de Euler requiere muchísima más información (el punto inicial u_0 y el gradiente en todos los puntos) que el método Deep BSDE (sólo utiliza la ecuación diferencial y la condición final). Además, con el método Deep BSDE, nuestro objetivo no es minimizar la función de coste. Es decir, no queremos un método para aproximar el valor final de u sobre un camino, ya que este dato viene dado por la condición final. Lo que queremos con esta función de coste es entrenar nuestra red, para conseguir los estimadores del valor inicial, el gradiente inicial, y los gradientes en cada punto temporal.

Otra característica curiosa de la figura 1 es la forma de la función de coste sobre el set de validación, que explicamos también en el apéndice 8.2.1.

Ahora nos centramos en analizar el error de nuestros estimadores. Utilizaremos la solución exacta conocida dada por las fórmulas 49 y 50 para comparar con los valores reales. Comencemos con $\hat{u}_0$ y con $\partial_x \hat{u}_0$. Según lo comentado en la sección anterior, son iguales a los pesos θ_0 y $\theta_{\partial_x u_0}$ en la última iteración del algoritmo de optimización de la red neuronal. En la figura 2 se puede ver cómo va actualizando nuestra red neuronal estos pesos, partiendo de dos valores aleatorios en la iteración inicial. Claramente se ve que en muy pocas iteraciones, la red neuronal determina qué valores de θ_0 y $\theta_{\partial_x u_0}$ minimizan mejor la función, y luego apenas los modifica. Es decir, la red primero determina estos valores de θ_0 y $\theta_{\partial_x u_0}$, y luego genera pequeñas modificaciones del resto de pesos para ir aproximando $\partial_x u(\cdot, t_i) \approx \partial_x \hat{u}_i(\cdot)$ en cada punto. Como la replicación del valor de u en todos los caminos parte de los valores θ_{u_0} y $\theta_{\partial_x u_0}$, pequeñas modificaciones en los mismos generan grandes cambios en la función de coste J, por lo que determinarlos con precisión es una tarea fundamental de nuestra red, y es lo que primero hace.

Efectivamente, método Deep BSDE predice muy bien el valor de u_0 y de $\partial_x u_0$:

- El valor real es $u_0 \simeq 0.841$ según 49. El Deep BSDE predice un valor $\tilde{u}_0 = 0.845$, con una diferencia relativa de 0.4%.
- Para el gradiente, $u_x(\xi, t_0) \simeq 0.540$ según 50. Y el Deep BSDE predice un valor $\partial_x \hat{u}_0 = 0.553$, con una diferencia relativa del 2.3%.

Por lo tanto, para la ecuación de difusión, el método Deep BSDE predice muy bien los valores iniciales. Si se quisiera aún más precisión, se podría reducir el intervalo Δt aumentando N. Esto incrementaría el coste computacional, pero se conseguirían mejores estimadores.

A continuación, veremos cómo el método Deep BSDE genera los valores $u(\tilde{X}_i, t_i)$ para varios caminos de nuestro input (ecuación 53). La figura 3 muestra los valores de $u(\tilde{X}_i^{(m)}, t_i)$ exactos (según la solución exacta al problema de cauchy 49), los estimados por con el método de Euler según 57, y los estimados por nuestro método Deep BSDE. Se muestran varias gráficas que corresponden a distintos caminos $\{\tilde{X}_i\}^{(m)}$ con m elegido aleatoriamente entre los caminos que pertenecen al set de validación. A la derecha, se muestran los valores del gradiente en cada tiempo discretizado

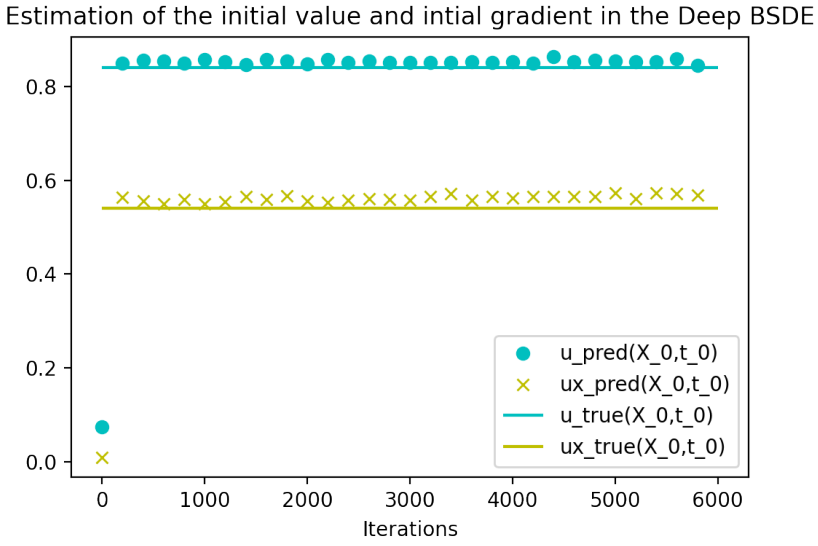

Figure 2: Estimación de los valores iniciales y finales de u según el método Deep BSDE (denotados como u_{pred} y ux_{pred}). Con las líneas sólidas se muestran los valores reales, que se hayan a partir de la solución conocida del problema 24.

t_i. También se muestran los valores exactos $\partial_x u_i(\tilde{X}_i^{(m)})$ obtenidos con la fórmula 50, y los valores aproximados con nuestro método $\partial_x \hat{u}_i(\tilde{X}_i^{(m)})$. Se puede apreciar que el método Deep BSDE replica de forma razonable el valor de u en dichos caminos, a pesar de no haber sido entrenado en ellos. En la figura 3 se muestran los tres casos significativos que pueden ocurrir:

1. Para el primer camino, los valores del gradiente se aproximan bastante bien con $\partial_x \hat{u}_i$, por lo que el camino real se replica con mucha exactitud.

2. Para el segundo camino, existen valores del gradiente que no se aproximan demasiado bien con $\partial_x \hat{u}_i$, por lo que $u(\tilde{X}_i, t_i)$ no se llega a replicar del todo bien por las limitaciones de nuestro método (estamos intentado aproximar el gradiente solamente minimizando una función de coste). En contra, el método de Euler sí que ofrece una mejor replicación, ya que se conocen exactamente los valores de $\partial_x u_i$.

3. Para el tercer camino, $\partial_x \hat{u}_i$ aproxima muy bien los valores del gradiente. Sin embargo, el camino real $u(\tilde{X}_i, t_i)$ no se replica bien, pero no por un fallo a la hora de aproximar los gradientes, sino porque la aproximación de tiempo discreto del método de Euler no es buena en este caso. Es decir, el método Deep BSDE tiene otra limitación adicional, y es que si no funciona bien el método de Euler, por muy bien que se aproxime $\partial_x u_i$, el camino no llega a replicarse del todo bien.

Estudiaremos ahora más a fondo cómo de buenos son los estimadores $\partial_x \hat{u}_i$. Como vimos, estos estimadores son las funciones $h_i(\cdot|\theta_i)$ tomando θ_i como el peso en la última iteración después de haber minimizado la función de coste de la red neuronal. Cabe destacar que estos estimadores predicen mucha más información que los estimadores $\hat{u}_0$ y $\partial_x \hat{u}_0$, puesto que no sólo estiman el valor de u o de $\partial_x u$ en un punto ξ, sino que son capaces de predecir el valor de $\partial_x u$ en un intervalo de

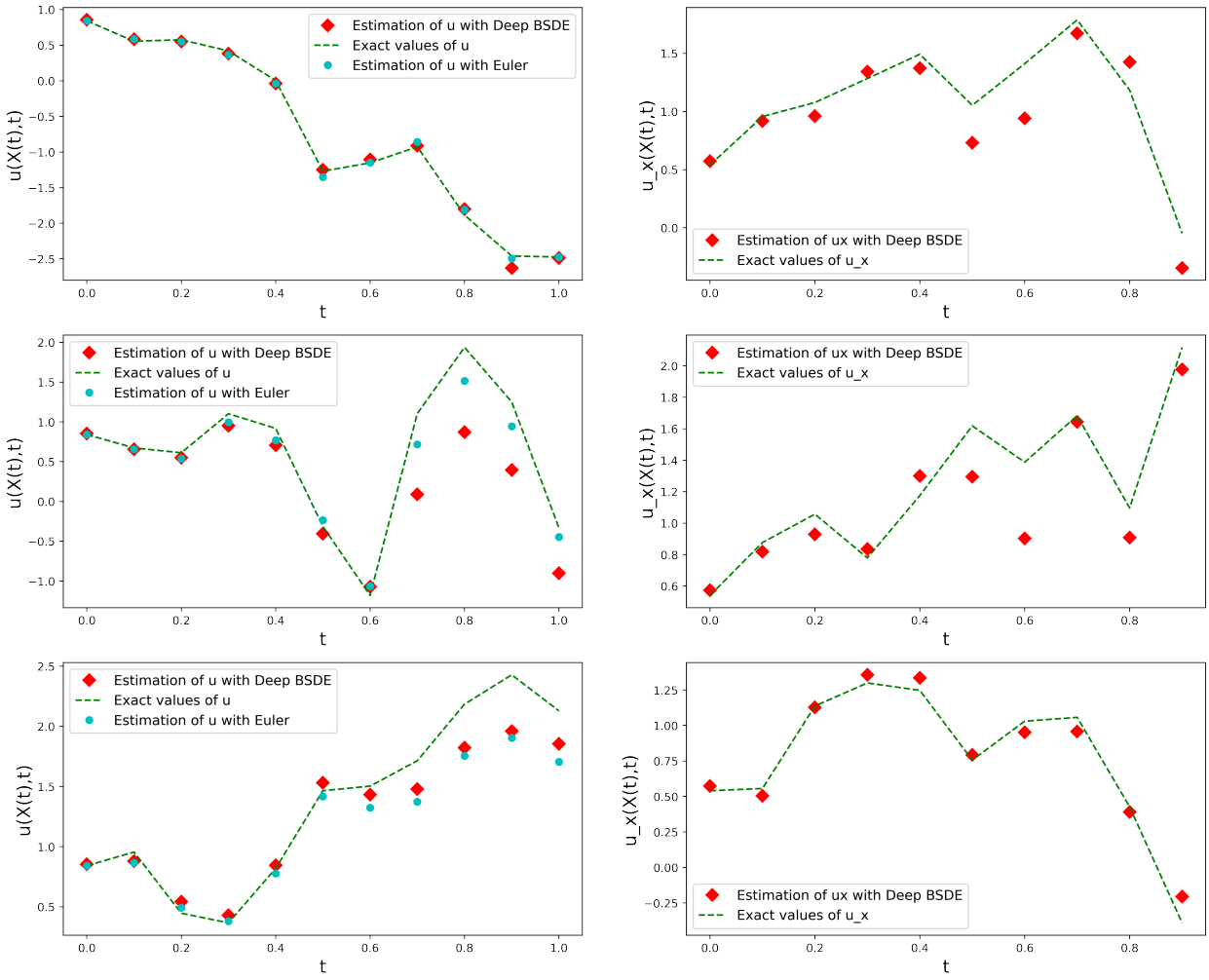

Figure 3: Valores de $u(X_i^{(m)}, t_i)$ (izquierda) y $u_x(X_i^{(m)}, t_i)$ (derecha) exactos y estimados con los métodos de Euler y con el método Deep BSDE, para tres caminos $\{\tilde{X}_i\}^{(m)}$ escogidos del set de validación. Tanto ux como u_x hacen referencia a la misma notación para $\partial_x u$.

puntos x en torno a ξ. $\partial_x \hat{u}_i$ aprende a aproximar $\partial_x u_i$ a partir de los caminos $\{\tilde{X}_i\}$ que han sido empleados para entrenar la red neuronal, por tanto, debemos tener en cuenta cómo es este input para saber los valores de x para los cuales $\partial_x \hat{u}_i$ es una buena estimación. En otras palabras, no podemos pretender que se aproxime bien el gradiente en valores de x que no aparecen en nuestro caminos. Esto se debe a que, como en el entrenamiento no aparecen estos x, los valores de $\partial_x \hat{u}_i$ que produce la red no sirven para entrenarla, por lo que producirá valores cualesquiera que no tendrán relación con $\partial_x u_i$.

A la izquierda de la figura 4 se pueden ver los valores de x en cada punto t_i para los M caminos generados que sirven como input para entrenar nuestra red. Se puede apreciar que, para tiempos iniciales, hay bastantes x que caen dentro del intervalo $(\xi - \sigma, \xi + \sigma)$, pero no dentro del intervalo $(\xi - 2\sigma, \xi + 2\sigma)$. Para valores finales del tiempo, sí que tenemos suficientes x en $(\xi - 2\sigma, \xi + 2\sigma)$.

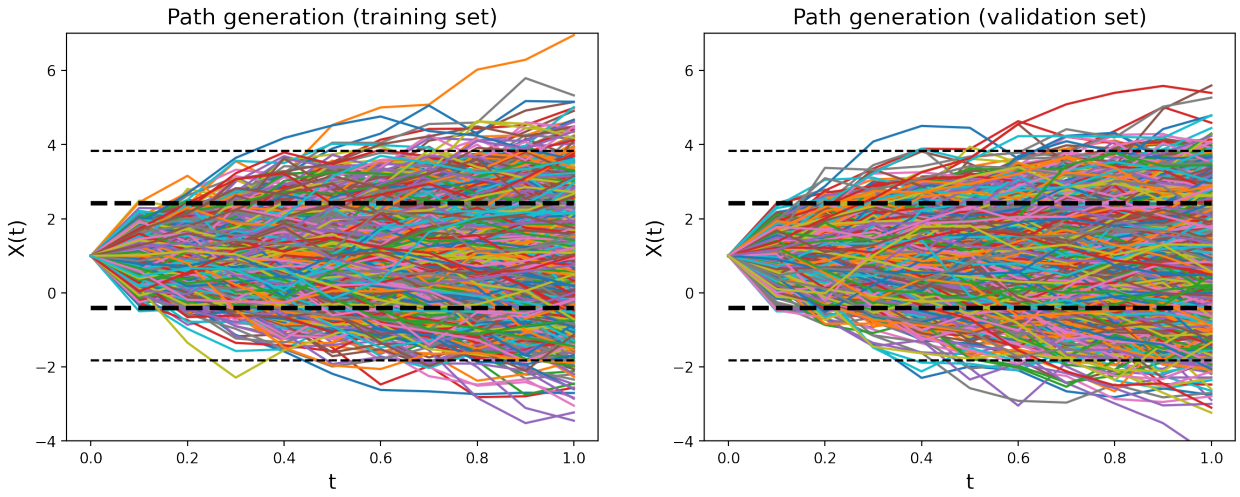

Figure 4: Caminos sobre los que se entrena la red (izquierda) y caminos del set de validación (derecha). Las líneas horizontales discontinuas gruesas indican una distancia de una desviación típica $\sigma = \sqrt{2}$ respecto a $\xi = 1$ ($\xi - \sigma$, $\xi + \sigma$), y las líneas horizontales discontinuas finas indican una distancia de dos desviaciones típicas ($\xi - 2\sigma$, $\xi + 2\sigma$).

Como hemos comentado, esto será crucial para entender nuestros estimadores $\partial_x \hat{u}_i$.

En la figura 5 se pueden visualizar las aproximaciones del gradiente $\partial_x \hat{u}_i$. Podemos ver cómo, a partir de nuestras redes neuronales en cada punto t_i, se aproxima una función continua del tipo $cos(x)e^{t_i}$, es decir, es un caso de aproximación de una función continua a partir de una composición de funciones bastante simples (es una Red Neuronal con capas tipo Denso y función de activación ReLU [4]). Para los primeros tiempos t_i, la aproximación sólo es buena en entornos cercanos a ξ. Sin embargo, según i aumenta, la aproximación es mejor para valores de x más alejados de ξ. Esto se puede entender por lo que hemos explicado de que la red sólo se entrena en los x que aparecen en los caminos input (izquierda de figura 4), y la región que abarcan estos x aumenta con t_i mayores, ya que ha habido más tiempo para que el camino se separe de ξ.

Para analizar cuantitativamente cómo de buenos son estos estimadores, definiremos su error a partir de la distancia en norma 2 a la solución exacta:

$$\text{Error}_{[a,b]}(\partial_x \hat{u}_i) = \sqrt{\frac{1}{b-a}\int_a^b |\partial_x \hat{u}_i(x) - \partial_x u_i(x)|^2 \, dx} \tag{59}$$

Para calcular dicho error, tomamos una partición equiespaciada $x_0 = a, ..., x_K = b$, con $x_{k+1} - x_k = \frac{b-a}{K}$ y K grande, y aproximamos:

$$\text{Error}_{[a,b]}(\partial_x \hat{u}_i) \approx \sqrt{\frac{1}{K}\sum_{k=0}^{K} |\partial_x \hat{u}_i(x_k) - \partial_x u_i(x_k)|^2} \tag{60}$$

Para analizar estos valores, los compararemos con el estimador más sencillo posible para $\partial_x u_i$, que

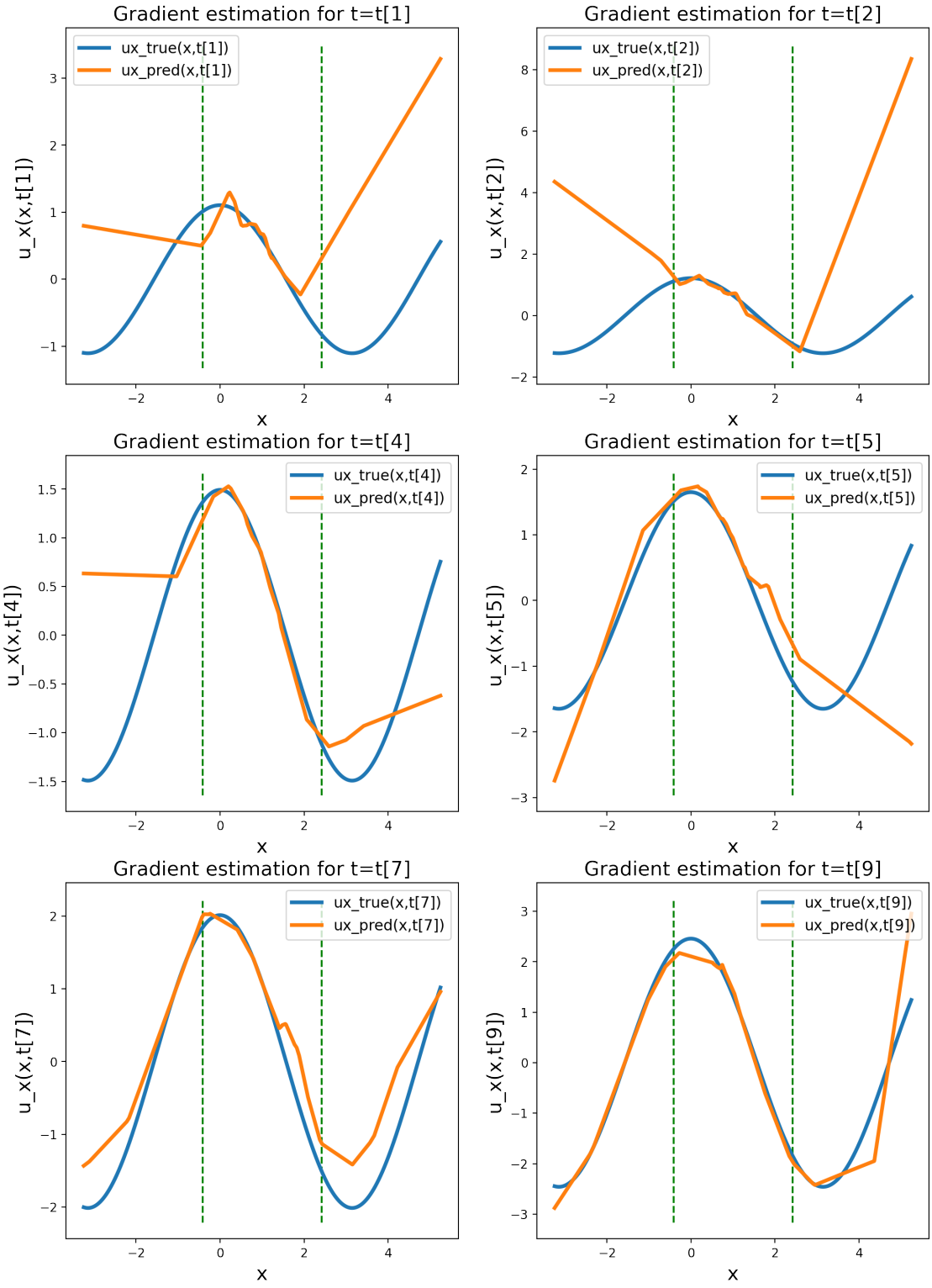

Figure 5: Estimación del gradiente que genera nuestra red neuronal ($\partial_x \hat{u}_i$) frente al valor exacto del gradiente ($\partial_x u_i$) para varios t_i.

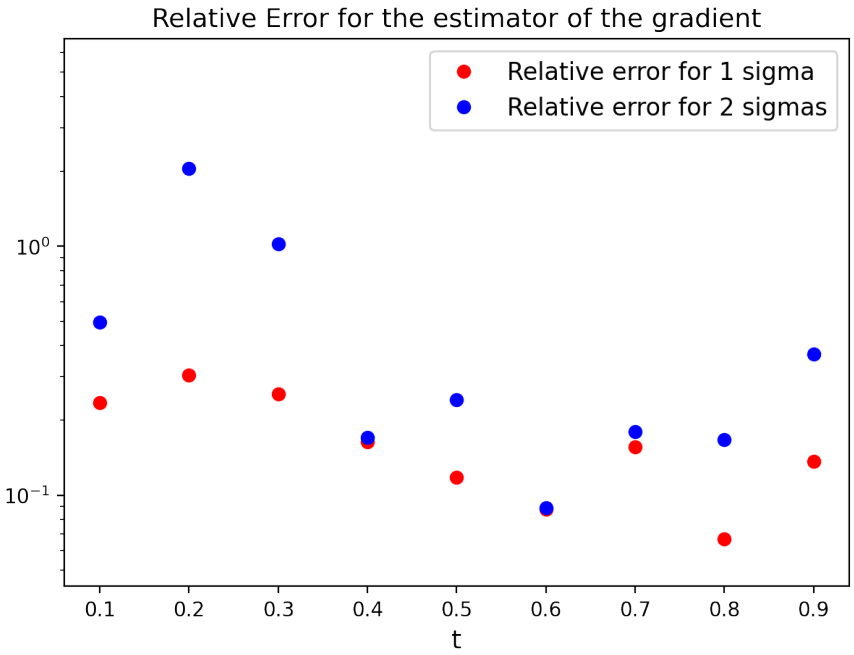

Figure 6: Error cuadrático medio de los estimadores $\partial_x \hat{u}_i$ en cada t_i, hallado a partir de 62 con $[a,b] = [\xi - \sigma, \xi + \sigma]$ (rojo) y $[a,b] = [\xi - 2\sigma, \xi + 2\sigma]$(azul). Nótese la escala logarítmica.

sería un estimador constante igual a la esperanza de $\partial_x u_i(x)$ en $x \in [a,b]$:

$$E_{[a,b]}[\partial_x u_i] = \frac{1}{b-a}\int_a^b \partial_x u_i(x)dx \approx \frac{1}{K}\sum_{k=0}^{K} \partial_x u_i(x_k) \tag{61}$$

Y hallamos el error relativo:

$$\text{Error relativo}_{[a,b]}(\partial_x \hat{u}_i) = \frac{\text{Error}_{[a,b]}(\partial_x \hat{u}_i)}{\text{Error}_{[a,b]}\left(E_{[a,b]}[\partial_x u_i]\right)} \approx \frac{\sqrt{\frac{1}{K}\sum_{k=0}^{K} |\partial_x \hat{u}_i(x_k) - \partial_x u_i(x_k)|^2}}{\sqrt{\frac{1}{K}\sum_{k=0}^{K} \left|E_{[a,b]}[\partial_x u_i] - \partial_x u_i(x_k)\right|^2}} \tag{62}$$

Los valores para cada t_i de dicho error relativo se muestran en la figura 6, tomando dos intervalos $[a,b]$, uno con una amplitud de 2σ en torno a ξ y otro con una amplitud de 4σ en torno a ξ. Esperamos que nuestra aproximación del gradiente sea mejor que simplemente una esperanza constante, por lo que buscamos valores del error relativo mucho menores que 1. Para tiempos bajos, nuestro estimador no es mucho mejor que la esperanza, ya que como hemos comentado no tenemos $\tilde{X}_1$ suficientes que cubran todo el intervalo $[\xi - 2\sigma, \xi + 2\sigma]$ (ver figura 4). Sin embargo, para t_i con $i > 1$, las aproximaciones mejoran sustancialmente, obteniendo errores relativos muy bajos, sobre todo para el intervalo de $[\xi - \sigma, \xi + \sigma]$.

4 El método Deep BSDE

En esta sección explicaremos el método Deep BSDE de forma general.

4.1 Deducción del método

Supongamos que tenemos el siguiente problema que involucra a una ecuación parabólica semilineal:

$$\begin{aligned} \partial_t u + \frac{1}{2}\sigma(x,t)^2 \partial_{xx} u + \mu(x,t)\partial_x u + f(u,x,t) = 0 \quad (x,t) \in \mathbb{R} \times (t_0, T) \\ u(x,T) = u_T(x) \quad x \in \mathbb{R} \end{aligned} \tag{63}$$

Dada u solución de dicho problema (que suponemos única, ver Corolario 2.3 de [2]) , queremos hallar:

$$u_0 \stackrel{\text{def}}{=} u(\xi, t_0); \quad \partial_x u_0 \stackrel{\text{def}}{=} \partial_x u(\xi, t_0); \quad \partial_x u_i(\cdot) \stackrel{\text{def}}{=} \partial_x u(\cdot, t_i) \tag{64}$$

Para ξ fijo. Sea $\{W_t\}_{t\in[0,T]}$ un proceso de Wiener, y a partir de él definimos el proceso $\{X_t\}$ que satisface la siguiente ecuación diferencial estocástica:

$$\begin{aligned} X_{t_0} &= \xi \\ dX_t &= \mu(X_t, t)dt + \sigma(X_t, t)dW_t \end{aligned} \tag{65}$$

A continuación, definimos el proceso:

$$U_t = u(X_t, t) \tag{66}$$

Aplicando el Lema de Itô, se tiene:

$$\begin{aligned} dU_t &= \left(\partial_t u(X_t,t) + \mu_t(X_t,t)\partial_x u(X_t,t) + \frac{1}{2}\sigma(X_t,t)^2 \partial_{xx} u(X_t,t) \right) dt + \sigma(X_t,t)\partial_x u(X_t,t) dW_t \\ &= -f\left(u(X_t,t), X_t, t\right) dt + \sigma(X_t,t)\partial_x u(X_t,t) dW_t \end{aligned} \tag{67}$$

Donde en la segunda igualdad hemos usado que u es solución de 63. Para trabajar con estas expresiones, tomamos N y $\Delta t = \frac{T-t_0}{N}$ y discretizamos el tiempo igual que antes:

$$t_i = t_{i-1} + \Delta t \quad i = 1, ..., N \tag{68}$$

Empleamos la notación:

$$X_i \stackrel{\text{def}}{=} X_{t_i}; \quad U_i \stackrel{\text{def}}{=} U_{t_i} \tag{69}$$

A partir de esta discretización, podemos generar procesos discretos:

$$\begin{aligned} \tilde{X}_0 &= \xi \\ \tilde{X}_{i+1} &= \tilde{X}_i + \mu(\tilde{X}_i, t_i)\Delta t + \sigma(\tilde{X}_i, t_i)\sqrt{\Delta t} Z_i \qquad i = 0, ..., N-1 \end{aligned} \tag{70}$$

Donde Z_0,...,Z_{N-1} son variables aleatorias independientes distribuidas con una normal $\mathcal{N}(0,1)$, que surgen debido a que en tiempo discreto el proceso de Wiener es $\Delta W_i = \sqrt{\Delta t} Z_i$. Estamos utilizando

la tilde para indicar que estamos en la aproximación discreta. Es decir, a raíz de la ecuación 65, se tiene que:

$$\tilde{X}_i \approx X_i \tag{71}$$

Y la igualdad se tiene si $\Delta t \longrightarrow 0$. Similarmente, definimos el proceso:

$$\begin{aligned} \tilde{U}_0 &= u_0 \\ \tilde{U}_{i+1} &= \tilde{U}_i - f(\tilde{U}_i, \tilde{X}_i, t_i)\Delta t + \sigma(\tilde{X}_i, t_i)\partial_x u(\tilde{X}_i, t_i)\sqrt{\Delta t} Z_i \qquad i = 0, ..., N-1 \end{aligned} \tag{72}$$

De nuevo, en virtud de la ecuación 67, se tiene que:

$$\tilde{U}_i \approx U_i \tag{73}$$

Y en consecuencia $\tilde{U}_N \approx U_N \approx u_T(\tilde{X}_N)$. Como lo que nos interesa es calcular u_0, $\partial_x u_0$ y $\partial_x u_i$, creamos variables θ_{u_0},$\theta_{\partial_x u_0}$, $h_i(x|\theta_i)$ que usaremos para sustituir los valores desconocidos u_0, $\partial_x u_0$ y $\partial_x u_i$ en un funcional:

$$\begin{aligned} u_0 &\longrightarrow \theta_{u_0} \\ \partial_x u(\xi, t_0) &\longrightarrow \theta_{\partial_x u_0} \\ \partial_x u(x, t_i) &\longrightarrow h_i(x|\theta_i) \quad i = 1, ..., N-1 \end{aligned} \tag{74}$$

Donde cada $h_i(x|\theta_i)$ es una red neuronal feedforward [4] compuesta por H capas de tipo denso, a cuyos pesos les llamamos θ_i. Dichas redes toman por input x y devuelven $h(x|\theta_i)$. A partir de estas variables, podemos definir un funcional basado en 72, usando las variables de 74:

$$\begin{aligned} U_0^\star[\{\tilde{X}_i\}|\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}] &= \theta_{u_0} \\ U_1^\star[\{\tilde{X}_i\}|\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}] &= U_0^\star - f(U_0^\star, \tilde{X}_0, t_0)\Delta t + \sigma(\tilde{X}_0, t_0)\theta_{\partial_x u_0}\sqrt{\Delta t} Z_0 \\ U_{i+1}^\star[\{\tilde{X}_i\}|\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}] &= U_i^\star - f(U_i^\star, \tilde{X}_i, t_i)\Delta t + \sigma(\tilde{X}_i, t_i)h(\tilde{X}_i|\theta_i)\sqrt{\Delta t} Z_i \quad i = 1, ..., N-1 \end{aligned} \tag{75}$$

Donde en el lado derecho escribimos $U_i^\star$ para referirnos a $U_i^\star[\{\tilde{X}_i\}|\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}]$. A partir de la aproximación 73, se tiene que:

$$U_N^\star[\{\tilde{X}_i\}|u_0, \partial_x u_0, \partial_x u_1, ..., \partial_x u_{N-1}] \approx u_T(\tilde{X}_N) \tag{76}$$

Por tanto, definimos la siguiente función de coste:

$$J(\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}) = E\left[\left(U_N^\star[\{\tilde{X}_i\}|\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}] - u_T(\tilde{X}_N)\right)^2\right] \tag{77}$$

Y tomamos como estimadores de los parámetros de interés los valores de $\theta_{u_0}, \theta_{\partial_x u_0}, \theta_1, ..., \theta_{N-1}$ que minimizen esta función de coste. Para minimizarla, utilizamos un algoritmo de optimización (como el "stochastic gradient descent" o el "Adam" [3]), y tomamos como estimadores los últimos pesos

entrenados de la red, que denotamos con un superíndice (f). Así, la solución que proponemos al al problema 63 es:

$$\begin{aligned} \hat{u}_0 &= \theta_{u_0}^{(f)} \\ \partial_x \hat{u}_0 &= \theta_{\partial_x u_0}^{(f)} \\ \partial_x \hat{u}_i(\cdot) &= h(\cdot|\theta_i^{(f)}) \end{aligned} \tag{78}$$

La parte esencial de este método es la función $f(u, x, t)$, a la cual no estamos imponiendo ninguna restricción. Esto nos permite resolver ecuaciones diferenciales parabólicas semilineales, como mostraremos ahora con la ecuación de Black-Scholes con riesgo de default o con multiplicadores

4.2 Ampliación del método Deep BSDE a varios valores iniciales

En finanzas, normalmente sólo necesitamos conocer el valor inicial de la solución de una ecuación diferencial en un punto ξ del activo subyacente, que sería el precio de un instrumento financiero. Esto es precisamente lo que nos proporciona el método Deep BSDE. Sin embargo, también es posible hallar el valor inicial de la solución para otros puntos distinos de ξ. Para ello, generamos caminos que partan de otros puntos iniciales ξ' y volvemos a entrenar la red neuronal con esos caminos, obteniendo estimadores para $u(\xi', t_0)$ y $\partial_x u(\xi', t_0)$. Es decir, aunque el método Deep BSDE sólo predice los valores iniciales en un punto ξ, a diferencia de otros métodos numéricos que predicen los valores iniciales para un conjunto de puntos x, el método Deep BSDE se puede repetir para todos los puntos iniciales ξ que queramos, consiguiendo así el perfil de la solución en el tiempo inicial. Además, si no reinicializamos los pesos de la red neuronal cuando pasamos de entrenar los caminos que parten ξ a entrenar los que parten de ξ', podemos obtener buenas estimaciones del gradiente para $t > t_0$, tanto para puntos cercanos a ξ como para puntos cercanos a ξ'.

5 El método Deep BSDE aplicado a la ecuación de Black-Scholes con riesgo de default

En esta sección aplicaremos el método Deep BSDE a una función parabólica muy importante en el mundo de las finanzas: la ecuación de Black-Scholes.

La ecuación de Black-Scholes estándar es lineal. Sin embargo, esta linealidad se basa en hipótesis de mercado que a veces no se cumplen. Por ejemplo, el modelo de Black-Scholes asume que los agentes del mercado no incurren en impagos y que siempre cumplen con sus obligaciones financieras. Pero esto no siempre es el caso, a veces ciertas instituciones financieras pueden entrar en quiebra y dejar de pagar sus obligaciones. Esto es lo que se conoce como *default* o *impago*. Al incluir este riesgo en la ecuación de Black-Scholes, se vuelve no lineal[5], y utilizaremos el método Deep BSDE para resolverla.

5.1 La ecuación de Black-Scholes estándar

Comenzaremos derivando la fórmula de Black-Scholes estándar, sin el riesgo de default. Supongamos que $u(x,t)$ es un instrumento financiero, cuyo valor depende de un activo subyacente x (por ejemplo, una acción) y del tiempo t. Para el activo subyacente, suponemos que sigue un proceso estocástico de la siguiente forma:

$$\frac{dX_t}{X_t} = \mu dt + \sigma_{BS} d\tilde{W}_t \tag{79}$$

Donde el término μdt corresponde al crecimiento determinista del activo subyacente, y el término $\sigma_{BS} d\tilde{W}_t$ corresponde a un comportamiento aleatorio debido a efectos externos (ver [7], sección 2.2), siendo σ_{BS} la volatilidad de la acción y $d\tilde{W}_t$ un proceso de Wiener. Bajo un cambio de medida, utilizando el teorema de Girsanov y Teoría de Valoración financiera (ver [8], sección 4.2.2), es posible reescribir este proceso como:

$$\frac{dX_t}{X_t} = r dt + \sigma_{BS} dW_t \tag{80}$$

Donde r es el tipo de interés del mercado y dW_t es otro proceso de Wiener. A esta nueva medida se le conoce *medida neutral al riesgo*, que cumple $d(X_t e^{-rt}) = \sigma_{BS}(X_t e^{-rt}) dW_t$. Es decir, la variación del valor del activo descontado a la tasa de interés es puramente aleatorio. Este proceso estocástico es un caso particular de 65 con $\mu(x,t) = rx$ y $\sigma(x,t) = x\sigma_{BS}$.

Ahora aplicamos el lema de Itô a la función $u(X_t, t)$ para hallar $u(x + dX_t, t + dt)$, obteniendo:

$$\begin{aligned} &u(x + dX_t, t + dt) = u\left(x\left(1 + r dt + \sigma_{BS} dW_t\right), t + dt\right) = \\ &= u(x,t) + \left(\partial_t u(x,t) + x r \partial_x u(x,t) + x^2 \tfrac{\sigma_{BS}^2}{2} \partial_{xx} u(x,t)\right) dt + x \sigma_{BS} \partial_x u(x,t)\, dW_t \end{aligned} \tag{81}$$

Ahora calculamos la esperanza según la medida neutral al riesgo estándar (que denotamos con un sombrero) condicionando al valor $X_t = x$ en tiempo t:

$$\hat{E}\left[u(x + dX_t, t + dt)\right] = u(x,t) + \left(\partial_t u + x r \partial_x u + x^2 \frac{\sigma_{BS}^2}{2} \partial_{xx} u\right) dt \tag{82}$$

[5] La no-linealidad viene de la asunción de que el riesgo de impago depende del valor que se debe pagar.

A continuación, aplicamos un argumento financiero. En finanzas, se denomina *arbitraje* a la posibilidad de ganar un beneficio en el mercado sin correr ningún riesgo. En los mercados reales, se asume que no existe el arbitraje. A partir de esta hipótesis, se puede llegar a ver que la esperanza de un instrumento financiero valorado con la medida neutral al riesgo (ecuación 80) en un tiempo $t + dt$ ($\hat{E}\left[u\left(x + dx, t + dt\right)\right]$) debe ser igual al dinero que obtendríamos con la tasa de interés sin riesgo r entre los tiempos t y $t + dt$ si introdujésemos $u(x, t)$ en el banco: $u(x, t) + ru(x, t)dt$ (ver [10]). Por tanto, se tiene la igualdad:

$$u(x,t) = \frac{\hat{E}\left[u\left(x + dX_t, t + dt\right)\right]}{1 + rdt} \tag{83}$$

Si el lado izquierdo de esta fórmula fuera estrictamente mayor que el derecho, un actor en el mercado podría construir una cartera financiera[6] vendiendo el instrumento $u(x, t)$ e invirtiendo dicho dinero en el banco, por lo que obtendría un beneficio neto sin riesgo con la tasa de interés r. Pero si el lado derecho fuera estrictamente mayor, un actor en el mercado podría pedir un préstamo al banco con interés r y crear una cartera financiera comprando el instrumento $u(x, t)$[7], ganando más con la cartera que lo que le cuesta tomar prestado el dinero, y de nuevo generaría un beneficio neto sin riesgo. Por ello, ambos lados de la ecuación han de ser iguales. Ahora, sustituyendo en esta ecuación el valor para $\hat{E}\left[u\left(x + dx, t + dt\right)\right]$ de 82 y cancelando los dt, se llega a la conocida ecuación de Black-Scholes:

$$\partial_t u + \frac{1}{2}x^2\sigma_{BS}^2\partial_{xx}u + xr\partial_x u - ru = 0 \tag{84}$$

5.2 La ecuación de Black-Scholes con riesgo de default

En finanzas, los instrumentos u que estamos valorando son comprados y vendidos por agentes del mercado. En este TFG trabajaremos con instrumentos que tienen fecha de vencimiento T distinta a la actual t_0. Con la firma de un contrato, el vendedor asume la obligación de pagar al comprador la cantidad $u(x, T)$ en el tiempo T. En ocasiones, el vendedor puede entrar en quiebra en un tiempo anterior a T, y por tanto no cumplir con su obligación financiera. Esto es a lo que nos referimos con riesgo de default o de impago. A partir de ahora, denotaremos con la letra u al instrumento en el caso en el que no se considera el riesgo de default, y v al instrumento cuando consideramos el riesgo de default. En este TFG trataremos un caso de default que modelizamos de la siguiente manera:

1. La posibilidad de que se produzca la quiebra del vendedor viene dada por el primer salto de una distribución de Poisson. Esto significa que, para un intervalo de tiempo diferencial dt, la probabilidad de que se produzca el default depende de una intensidad Q (característica del proceso de Poisson) multiplicada por dt:

$$P\left\{\text{default en [t,t+dt)}\right\} \approx Q\,dt \tag{85}$$

   Esta Q puede tomar cualquier valor siempre que sea positivo. Como veremos, el método Deep BSDE nos permite introducir cualquier Q, que puede en particular depender de v, que

[6] La cartera financiera construida tiene, además del instrumento u, una cantidad de activo subyacente x. Los detalles de cómo construir esta cartera exactamente se omiten en este TFG y se pueden consultar en [7].

[7] En este caso, la cartera financiera también poseería una cantidad de activo subyacente x.

es lo que añadirá la no-linealidad a la ecuación. Tiene sentido pensar que Q es una función creciente con v, ya que cuanto mayor sea la obligación financiera del vendedor del instrumento v, mayor será la probabilidad de que entre en quiebra.

Podemos definir un proceso de Poisson[8] Y_t de tasa o intensidad Q que valga:

$$\begin{array}{ll} Y_t = 0 & \text{si no se ha producido default entre } t_0 \text{ y } t. \\ Y_t \geq 1 & \text{si se ha producido el default entre } t_0 \text{ y } t. \end{array} \tag{86}$$

Por las propiedades del proceso de Poisson, se tiene que $P(Y_{t+dt} - Y_t = 1) = Qdt$.

2. En el caso de que ocurra el default, el vendedor sólo paga una fracción $\delta \in [0, 1)$ del valor del instrumento financiero u al comprador en el tiempo T. Es decir, si ocurre default, se paga una fracción δ de lo que se pagaría si no hubiera default.

   Podemos definir entonces la función $v(x, t, y)$, donde y puede tomar el valor 0 o el valor 1, de la siguiente forma:

$$\begin{array}{lcl} v(x,t,0) & = & \text{Valor de } v \text{ en } (x,t) \text{ cuando no ha ocurrido default en } [t_0, t) \\ v(x,t,1) & = & \text{Valor de } v \text{ en } (x,t) \text{ cuando ha ocurrido default en } [t_0, t) \end{array} \tag{87}$$

   Según nuestro modelo, se tiene:

$$v(x, t, 1) = \delta u(x, t) \tag{88}$$

   Es importante notar que $v(x, t, 0) \neq u(x, t)$, ya que con el instrumento $u(x, t)$, tenemos certeza de que no va a ocurrir default entre t_0 y T, mientras que con $v(x, t, 0)$ sólo sabemos que no ha ocurrido default entre t_0 y t, pero sigue existiendo la probabilidad de que ocurra default entre t y T. Según este razonamiento, $v(x, t, 0) < u(x, t)\ \forall\ t < T$[9], y $v(x, T, 0) = u(x, t)$.

   En general, conocemos los valores de $u(x, t)$ (vendrán dados por la resolución de la ecuación de Black-Scholes estándar) y estamos interesados en calcular los valores de $v(x, t, 0)$. Los valores de $v(x, t, 1)$ son triviales a partir de la fórmula 88. Pero introducir esta variable y será esencial a la hora de deducir la ecuación de Black-Scholes con riesgo de default.

3. Entendemos que el movimiento del activo subyacente x es independiente del suceso de default o no default. Por tanto, nuestra fórmula para el proceso que sigue X_t sigue siendo 80. Es decir, los movimientos en el mercado son independientes de que un vendedor particular presente quiebra o no.

Con este modelo, podemos generar una nueva ecuación que modifique a la de Black-Scholes estándar y que tenga en cuenta el riesgo de default, que es lo que hacemos a continuación.

Basándonos en el Primer Teorema Fundamental de Valoración de Activos [9], asumimos que existe una medida de valoración que permite escribir el precio actual del instrumento dado por $v\,(x, t, 0)$ en función de sus posibles valores $v\,(x + dX_t, t + dt, dY_t)$ en el instante $t + dt$ como:

$$v\,(x, t, 0) = \frac{\tilde{E}\left[v\,(x + dX_t, t + dt, dY_t)\,|X_t = x, Y_t = 0\right]}{1 + rdt} \tag{89}$$

[8] En el caso de que Q dependa de t, sería un proceso no homogéneo.

[9] Asumiendo probabilidad de default no nula.

Donde $\tilde{E}$ denota el valor esperado según esta medida de valoración de riesgo neutro, que tiene en cuenta todos los escenarios posibles, haya o no default. Las variables aleatorias con respecto a las cuales se calcula la esperanza son dX_t, (la variación estocástica del precio del subyacente) y dY_t, que es la que indica si hay un evento de default en $[t, t+dt)$. r es el tipo de interés, que aparece porque estamos usando una medida de riesgo neutro[10].

Recordamos que $v(x,t,0)$ es el precio del instrumento financiero dado que no ha ocurrido default hasta t, y para calcular este precio correctamente debemos tener en cuenta la posibilidad de que ocurra default en tiempos posteriores.

Ahora calculamos:

$$\begin{aligned}\tilde{E}\left[v\left(x+dX_t, t+dt, dY_t\right)|X_t=x, Y_t=0\right] = &\underbrace{P(dY_t=1)}_{Qdt}\tilde{E}\left[v\left(x+dX_t, t+dt, dY_t\right)|X_t=x, dY_t=1\right] \\ &+\underbrace{P(dY_t=0)}_{1-Qdt}\tilde{E}\left[v\left(x+dX_t, t+dt, dY_t\right)|X_t=x, dY_t=0\right]\end{aligned} \tag{90}$$

Donde asumimos que el intervalo dt es lo suficientemente pequeño como para que las probabilidades de $dY_t > 1$ sean despreciables. En el caso de que no haya default ($dY_t = 0$), nos encontraremos en $t+dt$ con los valores $v\left(x+dX_t, t+dt, 0\right)$, pues en ese caso no habría ocurrido default hasta $t+dt$ y por lo tanto los valores vienen dados por la función $v\left(\cdot,\cdot,0\right)$. En el caso de que haya default en el intervalo $[t, t+dt)$, el pago sería la fracción δ del valor $u(x,t)$, según la fórmula 88. Es decir, si en el intervalo dt se produce el impago, ya conocemos exactamente la el valor de v durante el resto de tiempos t hasta T, que es igual a $\delta u(x,t)$. Esto nos lleva a:

$$\begin{aligned}\tilde{E}\left[v\left(x+dX_t, t+dt, dY_t\right)|X_t=x, Y_t=0\right] = &\, Qdt\hat{E}\left[\delta u\left(x+dX_t, t+dt\right)\right] \\ &+\left(1-Qdt\right)\hat{E}\left[v\left(x+dX_t, t+dt, 0\right)\right]\end{aligned} \tag{91}$$

Donde $\hat{E}$ se refiere a los escenarios posibles una vez que está decidido el default (se utiliza la misma medida de probabilidad que en la ecuación 83), y dichas esperanzas se calculan condicionadas a $X_t = x$, aunque no lo escribamos explícitamente. Esto es válido siempre que la evolución del activo subyacente sea independiente de la posibilidad de default, que es la tercera hipótesis de nuestro modelo. Reescribiendo:

$$\begin{aligned}\underbrace{\frac{\tilde{E}\left[v\left(x+dX_t, t+dt, dY_t\right)|X_t=x, Y_t=0\right]}{1+rdt}}_{v(x,t,0)} = &\, Q\delta dt\,\underbrace{\frac{\hat{E}\left[u(x+dX_t, t+dt)\right]}{1+rdt}}_{u(x,t)} \\ &+\left(1-Qdt\right)\frac{\hat{E}\left[v\left(x+dX_t, t+dt, 0\right)\right]}{1+rdt}\end{aligned} \tag{92}$$

Donde las sustituciones que hacemos con las llaves inferiores se deben a las ecuaciones 89 y 83 respectivamente. Multiplicando por $1+rdt$ y sustituyendo $\hat{E}\left[v\left(x+dX_t, t+dt, 0\right)\right]$ por el valor

[10] El interés aparece en la fórmula 89 porque tener una expresión de este tipo es equivalente a no tener arbitraje en el mercado (el valor esperado que ofrece el instrumento financiero v ha de ser igual a la ganancia que generaría nuestro capital con una tasa de interés r sin riesgo).

deducido en 82, se tiene:

$$v(x,t,0)(1+rdt) = Q\delta dt(1+rdt)u(x,t) + (1-Qdt)\left[v(x,t,0) + \left(\partial_t v(x,t,0) + xr\partial_x v(x,t,0) + x^2\frac{\sigma_{BS}^2}{2}\partial_{xx}v(x,t,0)\right)dt\right] \tag{93}$$

Desarrollando e ignorando los diferenciales dt de orden 2, se tiene:

$$rv(x,t,0)dt = Q\delta u(x,t)dt - Qv(x,t,0)dt + \left(\partial_t v(x,t,0) + xr\partial_x v(x,t,0) + x^2\frac{\sigma_{BS}^2}{2}\partial_{xx}v(x,t,0)\right)dt \tag{94}$$

Cancelando los dt, reordenando y usando la notación $v(x,t) \overset{\text{def}}{=} v(x,t,0)$, se llega a la ecuación de Black-Scholes con default risk:

$$\partial_t v + xr\partial_x v + x^2\frac{\sigma_{BS}^2}{2}\partial_{xx}v - (r+Q)v = -Q\delta u(x,t) \tag{95}$$

Donde v hace referencia a $v(x,t)$. Se trata de una ecuación parabólica que será no lineal si Q depende de v. Realizaremos tres observaciones sobre esta ecuación:

- Si $Q \longrightarrow \infty$, la ecuación queda:
$$v(x,t) = \delta u(x,t) \tag{96}$$
Es decir, que si el evento de default ocurre seguro, el valor de nuestro instrumento será igual al valor sin default multiplicado por δ, como es lógico a partir de las hipótesis de nuestro modelo.

- Si $\delta = 1$, la ecuación queda:
$$\partial_t v + xr\partial_x v + x^2\frac{\sigma_{BS}^2}{2}\partial_{xx}v - rv - Q\left(v - u(x,t)\right) = 0 \tag{97}$$
Y por unicidad de la solución, $v(x,t) = u(x,t)$. Es decir, cuando $\delta = 1$, el default no debe de modificar en nada el precio de nuestra opción, por lo que, independientemente de la intensidad de default Q, obtenemos el valor $u(x,t)$.

- En la ecuación para v, debemos tener en cuenta que podemos perder dinero si el vendedor del instrumento financiero entra en quiebra. Por ello, esperamos que el precio para v sea menor que el precio para u ($v_0 < u_0$), debido a que con u estamos ignorando este riesgo de default. Esto lo comprobaremos numéricamente en las siguientes secciones.

Sólo queda modelizar la forma de Q. Como dijimos, tiene sentido asumir que Q es una función creciente con $v(x,t)$. En este TFG trabajaremos con una función de la forma:

$$Q(v) = \gamma^{(l)} + (\gamma^{(h)} - \gamma^{(l)})\frac{1}{1+e^{-\beta(v-z)}} \tag{98}$$

Con $\gamma^{(l)} > 0$, $\gamma^{(h)} > \gamma^{(l)}$, $\beta > 0$ y $z > 0$. Se trata de una función sigmoidea, que toma un valor $\gamma^{(l)}$ cuando $v \longrightarrow -\infty$ y un valor $\gamma^{(h)}$ cuando $v \longrightarrow +\infty$. El crecimiento de $\gamma^{(l)}$ a $\gamma^{(h)}$ se produce principalmente todo en un intervalo en torno a z, que será mayor cuanto más pequeño sea β.

5.3 El método Deep BSDE aplicado a la ecuación de Black-Scholes estándar

Consideramos el problema:

$$\begin{aligned} \partial_t u + \frac{1}{2}x^2\sigma_{BS}^2\partial_{xx}u + xr\partial_x u - ru = 0 \quad (x,t) \in \mathbb{R}\times[t_0,T] \\ u(x,T) = \max(x-E,0) \quad x \in \mathbb{R} \end{aligned} \tag{99}$$

Que es la ecuación de Black Scholes estándar 99 para una opción europea de tipo "Call" [7]. Este problema es un caso particular de 63, tomando:

$$\begin{aligned} \sigma(x,t) &= \sigma_{BS}x \\ \mu(x,t) &= rx \\ f(u,x,t) &= -ru \end{aligned} \tag{100}$$

La solución exacta de este problema es conocida y es [7]:

$$u(x,t) = xN(d_1) - Ee^{-r(T-t)}N(d_2) \tag{101}$$

$$d_1 = \frac{\log x/E + (r+\frac{1}{2}\sigma_{BS}^2)(T-t)}{\sigma_{BS}\sqrt{T-t}} \tag{102}$$

$$d_2 = \frac{\log x/E + (r-\frac{1}{2}\sigma_{BS}^2)(T-t)}{\sigma_{BS}\sqrt{T-t}} \tag{103}$$

Donde $N(\cdot)$ es la función de distribución de una normal de media 0 y desviación típica 1:

$$N(d) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d} e^{-\frac{y^2}{2}}dy \tag{104}$$

Para hallar la derivada con respecto de x, calculamos:

$$\partial_x[N(d)] = (\partial_d N)(d)\cdot\partial_x[d] = \frac{e^{-\frac{d^2}{2}}}{\sqrt{2\pi}}\cdot\frac{1}{x\sigma_{BS}\sqrt{T-t}} \tag{105}$$

Y por tanto:

$$\begin{aligned} \partial_x u(x,t) &= N(d_1) + x\cdot\frac{e^{-\frac{d_1^2}{2}}}{\sqrt{2\pi}}\cdot\frac{1}{x\sigma_{BS}\sqrt{T-t}} - Ee^{-r(T-t)}\cdot\frac{e^{-\frac{d_2^2}{2}}}{\sqrt{2\pi}}\cdot\frac{1}{x\sigma_{BS}\sqrt{T-t}} \\ &= N(d_1) + \frac{1}{\sqrt{2\pi}\sigma_{BS}\sqrt{T-t}}\left(e^{-\frac{d_1^2}{2}} - \frac{E}{x}e^{-r(T-t)}e^{-\frac{d_2^2}{2}}\right) \end{aligned} \tag{106}$$

Utilizaremos estas soluciones para testear la eficiencia del método Deep BSDE aplicado al problema 99. Para ello, aplicamos el método descrito en 4.1 con los valores de σ, μ y f dados por 100. Los detalles de la implementación en este caso son parecidos a los utilizados en la sección 3.3 y los especificamos en 8.3. A continuación, exponemos los resultados.

La estimación para los valores iniciales de u_0 (premium de la opción) y $\partial_x u_0$ es muy buena, como puede verse en la figura 7.

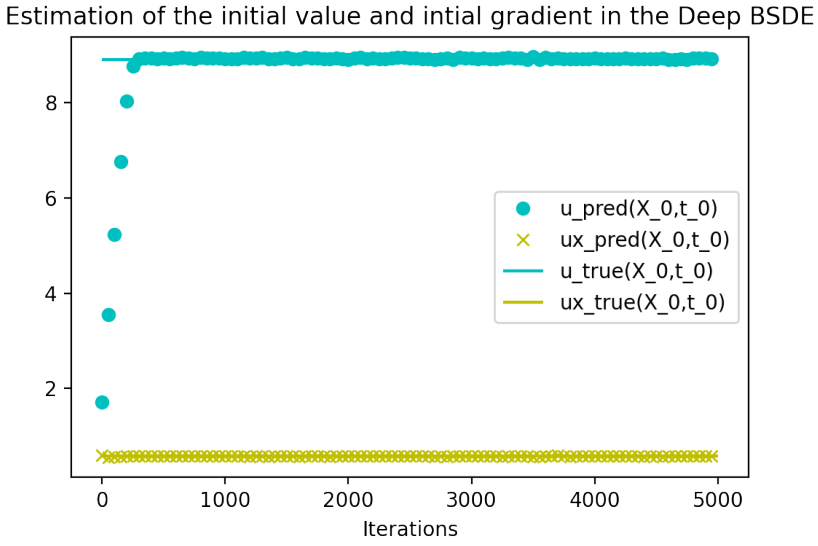

Figure 7: Estimación de u_0 y de $\partial_x u_0$ para la ecuación de Black-Scholes estándar 99. Los puntos representan las estimaciones del gradiente en función de la iteración de nuestra red neuronal (los valores de θ_{u_0} y $\theta_{\partial_x u_0}$ en cada iteración del entrenamiento de la red), y las líneas horizontales son los valores reales.

- Para u_0, el valor exacto según la fórmula 101 es $u_0 = 8.9160$, y nuestro estimador es $\hat{u}_0 = 8.9282$, con una diferencia relativa de 0.1%.
- Para $\partial_x u_0$, el valor exacto según la fórmula 106 es $\partial_x u_0 = 0.5793$, y nuestro estimador es $\partial_x \hat{u}_0 = 0.5723$, con una diferencia relativa de 1.2%.

Como vemos, los estimadores son extraordinariamente buenos, a pesar de tener sólo $N = 10$ pasos temporales en un intervalo $T - t_0 = 1$año.

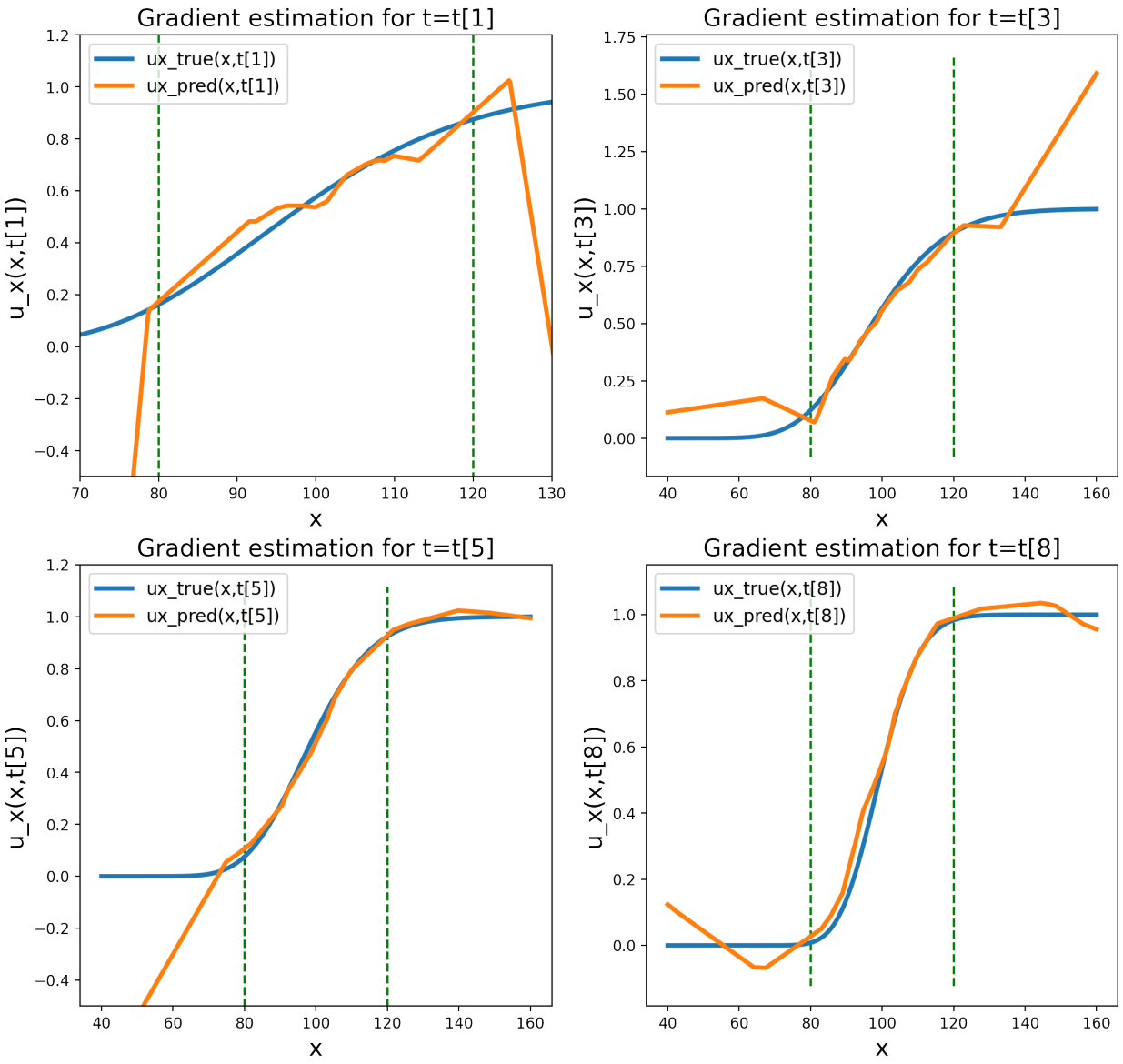

Figure 8: Estimación para el gradiente de u, solución de 99. Se muestra el gradiente exacto, dado por 106, y las aproximaciones a dicho gradiente en nuestra red neuronal $h(\cdot\theta_i)$ para varios t_i. Se muestra el intervalo de x de $[\xi - \xi\sigma_{BS}, \xi + \xi\sigma_{BS}]$ en barras verticales discontinuas ($\xi = 100$). En color azul se muestra el gradiente real, y en color naranja el gradiente aproximado.

Ahora veamos los estimadores para el gradiente. En la figura 8 se muestran los gráficos de nuestros estimadores para el gradiente $\partial_x \hat{u}_i$, comparados con el valor real según 106. En dicha figura se muestran los aproximadores en un intervalo de $[\xi - 2\sigma_{BS}\xi, \xi + 2\sigma_{BS}\xi]$. Tomamos el valor $\sigma_{BS}\xi$ como un valor de referencia para la desviación típica de los caminos $\{\tilde{X}_i\}$. En este caso se trata de caminos geométricos, por lo que la desviación típica depende del valor de $\tilde{X}$ (ecuación 80), pero podemos usar el valor $\sigma(\xi, t_0) = \sigma_{BS}\xi$ como una desviación típica aproximada.

Se puede apreciar que, como ocurría con la ecuación de difusión, nuestros estimadores para el gradiente son buenos en esta región en torno a ξ. No llegan a ser tan buenos como los estimadores de u_0 y $\partial_x u_0$, pero debemos tener en cuenta que los estimadores $\partial_x \hat{u}_i$ tratan de aproximar un continuo de puntos x, no sólo un único punto ξ, por lo que es esperable que sean un tanto peores.

Cuantitativamente, podemos hallar errores para estos estimadores, y compararlos con el estimador de referencia. De nuevo, utilizamos el error cuadrático medio entre $\partial_x \hat{u}_i$ y $\partial_x u_i$ según la fórmula 59, y lo comparamos con el estimador esperanza constante 61. Los errores relativos vienen dados por la fórmula 62, y se muestran en la figura 9. Podemos confirmar con estos datos cuantitativos que nuestros estimadores son buenos, sobre todo para el intervalo $[\xi - \sigma_{BS}\xi, \xi + \sigma_{BS}\xi]$, ya que disminuyen el error hasta 10 o más veces, en comparación con el estimador media en ese intervalo, que ya de por sí tiene un error cuadrático medio bastante bajo.

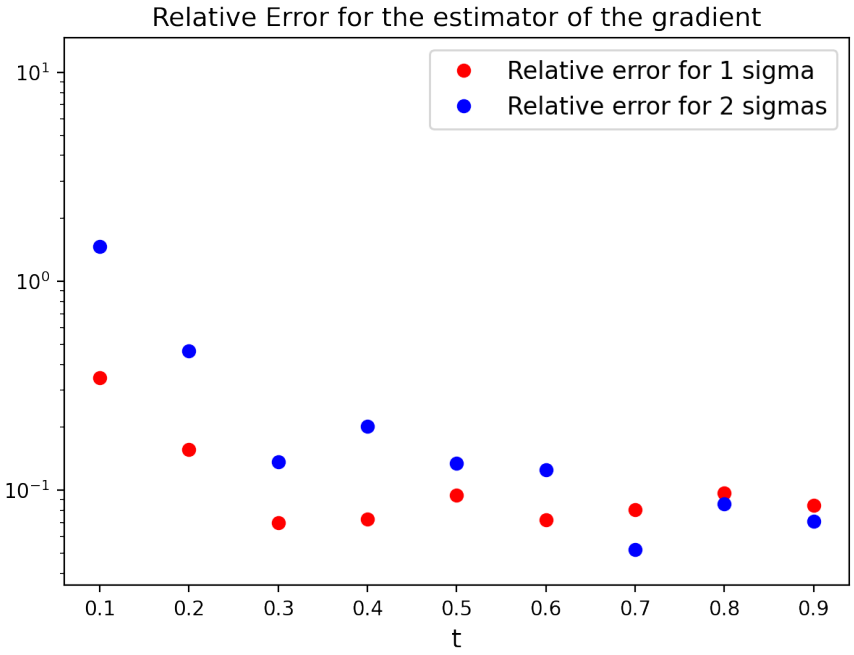

Figure 9: Error cuadrático medio de los estimadores $\partial_x \hat{u}_i$ en cada t_i, hallado a partir de 62 con $[a, b] = [\xi - \xi\sigma_{BS}, \xi + \xi\sigma_{BS}]$ (rojo) y $[a, b] = [\xi - 2\xi\sigma_{BS}, \xi + 2\xi\sigma_{BS}]$(azul). Nótese la escala logarítmica.

Además, se puede apreciar que, sobre todo a tiempos grandes (como es el caso de t_5 y t_8), el gradiente se aproxima mejor para valores $x \gg \xi$ que para valores $x \ll \xi$. Esto se debe a los x que se utilizan para entrenar la red. Como los procesos $\{\tilde{X}_i\}$ son geométricos (80), ocurren con más frecuencia valores $x \gg \xi$ que valores $x \ll \xi$, como se aprecia en la figura 10.

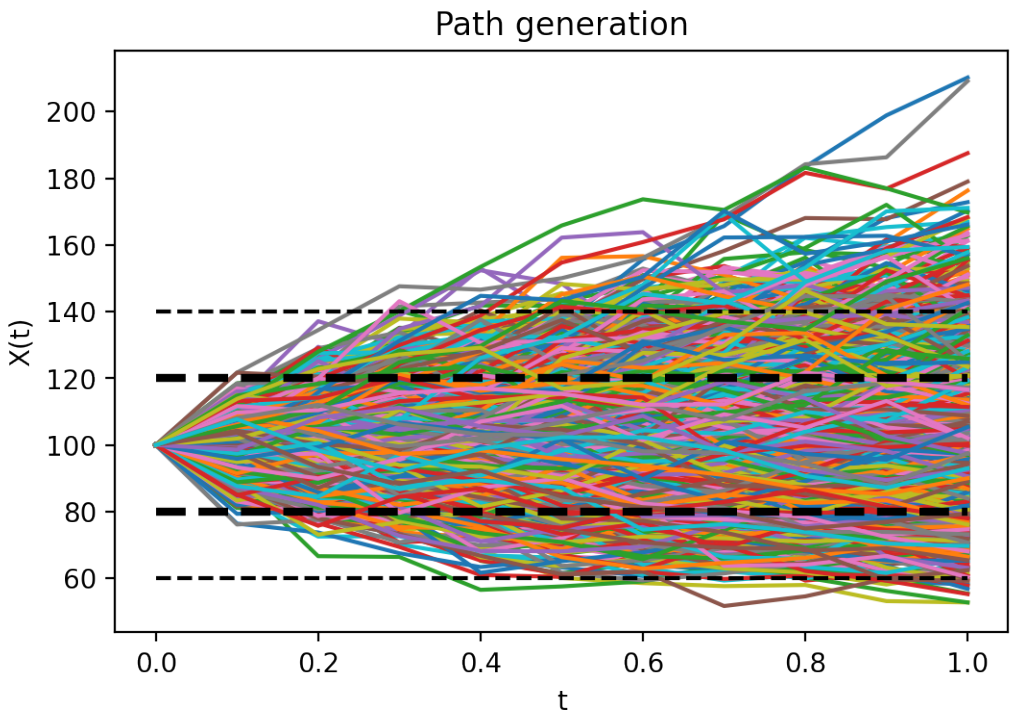

Figure 10: Caminos con los que se ha entrenado la red para el método Deep BSDE aplicado al problema 99.

5.4 El método Deep BSDE aplicado a la ecuación de Black-Scholes con riesgo de default

Ahora, resolveremos la ecuación de Black-Scholes con el riesgo de default (95). Sea el problema:

$$\begin{aligned} \partial_t v + xr\partial_x v + x^2 \frac{\sigma_{BS}^2}{2} \partial_{xx} v - [r + Q(v)]\, v = -Q(v)\, \delta u(x,t) \quad (x,t) \in \mathbb{R} \times [t_0, T] \\ v(x,T) = \max(x - E, 0) \quad x \in \mathbb{R} \end{aligned} \tag{107}$$

Donde $u(x,t)$ es la solución al problema sin riesgo de default (solución al problema 99 dada por 101) y Q viene dado por la fórmula 98. Este problema modeliza una opción "Call" con riesgo de default. El hecho de que la condición terminal siga siendo $\max(x-E,0)$, como en el caso sin riesgo de default, se debe a que es una ecuación para $v(x,t,0)$, que es valor del instrumento financiero suponiendo que no ha ocurrido default hasta t, por lo que en $t = T$ el valor de $v(x,T,0)$ coincide con el de $u(x,T)$ (si en T no ha ocurrido default, ya no hay tiempo para que ocurra).

Para resolver este problema, aplicamos el método Deep BSDE (4.1) con:

$$\begin{aligned} \mu(x,t) &= rx \\ \sigma(x,t) &= \sigma_{BS} x \\ f(v,x,t) &= -(r + Q(v))v + \delta Q(v) u(x,t) \end{aligned} \tag{108}$$

Las especificaciones de dicho método están en 8.4. A continuación, exponemos los resultados. Los compararemos con los valores que se obtienen a partir de la fórmula de Black-Scholes sin riesgo de default u y su gradiente (ecuaciones 101 y 106). Claramente v no debe ser igual a u, ya que en este caso v se calcula teniendo en cuenta la posibilidad de default. Pero usaremos el valor de u como referencia para entender el efecto de este riesgo de default.

Para el valor inicial v_0, obtenemos $v_0 = 7.266$ mientras que $u_0 = 8.916$, con una diferencia relativa del 18.5%. Es decir, hay una diferencia muy significativa entre el valor que deberíamos pagar por la opción dependiendo de si tenemos en cuenta el riesgo de default o no. A pesar de que nuestro método Deep BSDE es aproximativo, este valor es fiable, puesto que la precisión con la que el método halló el valor de u_0 fue del 0.1% (sección 5.3), por lo que el error al estimar u_0 en este caso es mucho menor que la diferencia entre v_0 y u_0. Para el valor inicial de la derivada, se tiene que $\partial_x v_0 = 0.480$ mientras que $\partial_x u_0 = 0.579$, con una diferencia relativa de 17.2%.

Estas diferencias tan altas entre los valores iniciales se deben a que hemos tomado parámetros para los cuales la probabilidad de default del vendedor es muy significativa. En la sección 8.4 se explica que se toma $\gamma^{(l)} = 0.6$ y $\gamma^{(h)} = 0.9$ para la fórmula de $Q(v)$ (98). Como Q no es constante, el proceso de Poisson Y_t que indica si ha ocurrido default o no (definido en 86) es no homogéneo. Pero para hacernos una idea de la probabilidad de que ocurra default entre t_0 y T, podemos tomar $Q \approx 0.75$. Entonces:

$$P(\{\text{Ocurra default en } [t_0, T]\}) = P(Y_T - Y_{t_0} \geq 1) \approx 1 - e^{-Q(T-t_0)} \simeq 0.53 \tag{109}$$

Es decir, hay una probabilidad de más del 50% de que ocurra el default, y que por tanto sólo recibamos una fracción δu de lo que nos debieran pagar. Por ello, el precio inicial del instrumento v es más bajo del que obtendríamos si obviáramos esta posibilidad de default.

También comentaremos la estimación del gradiente. En la figura 11 se puede observar cómo el gradiente obtenido es similar al gradiente de la solución de Black-Scholes estándar (fórmula 106), pero siempre suele quedar por debajo del mismo. Para los primeros tiempos, el gradiente está apreciablemente por debajo de la solución del caso estándar. Sin embargo, para valores más altos del tiempo, el gradiente parece aproximarse mejor. Esto se debe a que $v(x,t) \longrightarrow u(x,t)$ cuando $t \longrightarrow T$. Es decir, cuanto más cerca estamos de T, menos probable es que ocurra default, y por lo tanto más parecidos son los valores de v y u.

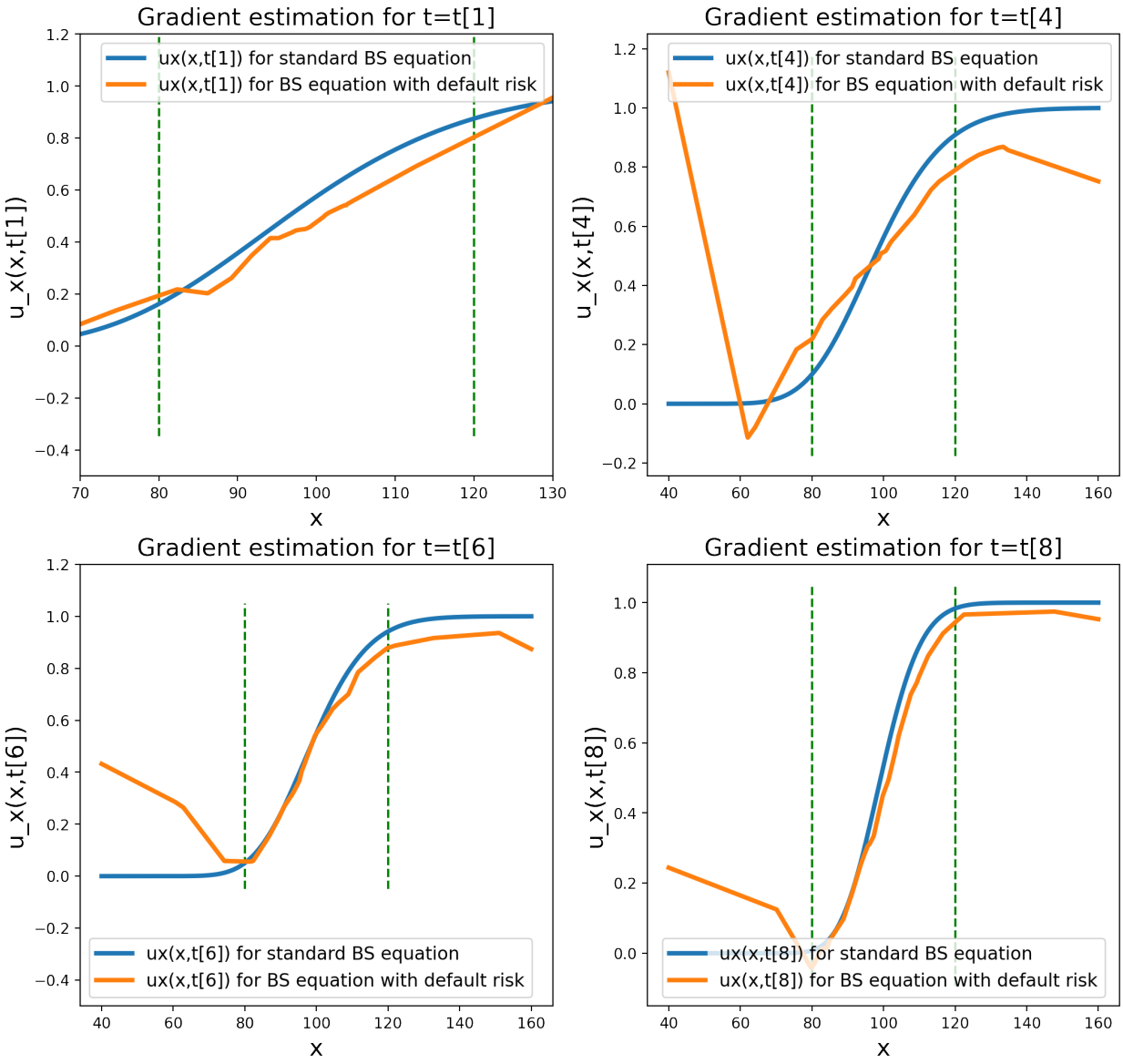

Figure 11: Estimación del gradiente para la ecuación de Black-Scholes con riesgo de default (naranja), comparado con el gradiente exacto de la fórmula de Black-Scholes sin riesgo de default (azul).

5.5 Soluciones para la ecuación de Black-Scholes con distintos riesgos de default

En nuestro modelo, cuando se produce el default, el comprador del instrumento financiero recibe, como mínimo, una fracción δ del valor que recibiría si obviáramos el riesgo de default. Por tanto, en el peor caso posible, tenemos un instrumento cuyo valor es δu. Y si no se produce default, se obtendría todo el valor u. Por tanto, es de esperar que los valores de v se encuentren entre δu y u, tendiendo a u cuando la probabilidad es muy baja y tendiendo a δu cuando la probabilidad es alta.

Para observar este efecto, hemos resuelto el problema 107 para distintos valores constantes de Q, utilizando también el método Deep BSDE descrito en la sección anterior con los parámetros explicados en 8.3 y tomando $\delta = \frac{2}{3}$. En la figura 12 se muestran los valores v_0 para las distintas Q, comparados con los valores δu_0 y u_0.

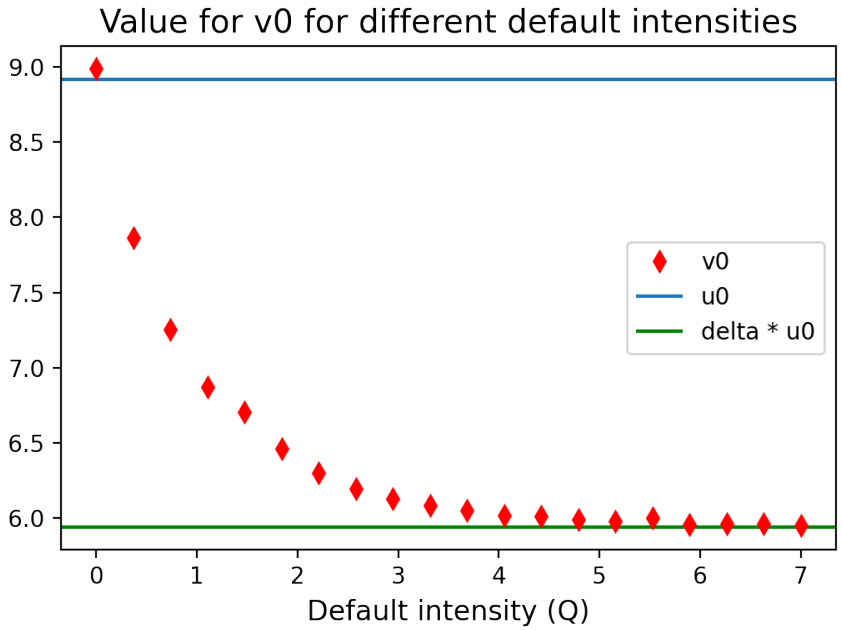

Figure 12: Valores para el precio inicial $v_0 = v(\xi, t_0)$ de un instrumento financiero con riesgo de default de varias intensidades Q constantes. Los datos de ξ, t_0 y del resto de parámetros (excepto la Q) son los mismos que los utilizados en 8.4. Estos valores se comparan con el valor inicial del instrumento sin riesgo de default u_0 y con el de dicho valor multiplicado del δ.

El comportamiento que esperábamos se puede observar claramente en esta figura. Según el riesgo de default aumenta, disminuye el precio v_0 que debemos pagar por el instrumento financiero v, debido a esta posibilidad de default. Pero los valores de v_0 nunca bajan de δu_0, ya que, ocurra default o no, siempre tenemos asegurada la cantidad δu. Para valores muy altos de la intensidad de Q, el default ocurre casi seguro, por lo que v_0 está muy próximo a δu_0 (por ejemplo, para $Q = 4$, $P\{\text{Default}\} = 1 - e^{-Q(T-t_0)} \simeq 0.98$).

También es importante notar cómo cambian los valores iniciales del gradiente $\partial_x v_0$, que se muestran en la gráfica 13. Como se puede apreciar, tienen un comportamiento parecido a los valores del precio v_0. Estos valores son esenciales en finanzas, ya que están relacionados con la cobertura (hedging)

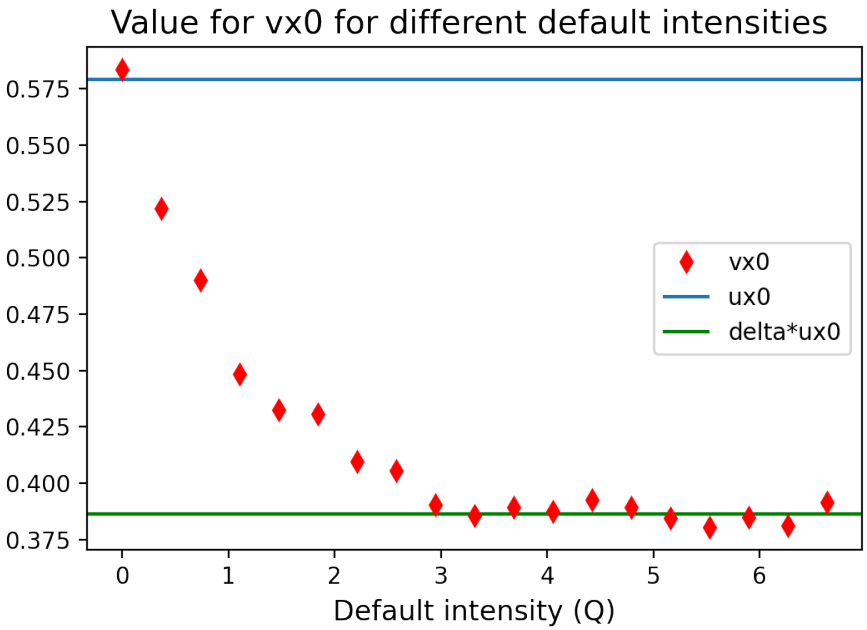

Figure 13: Valores para el gradiente precio inicial $\partial_x v_0 = \partial_x v(\xi, t_0)$ $(vx0)$ de un instrumento financiero con riesgo de default de varias intensidades Q, comparados con el gradiente inicial solución de la ecuación de Black-Scholes sin riesgo de default $(ux0)$ y dicho valor multiplicado por δ.

del riesgo en carteras financieras.

6 La ecuación de Black-Scholes con multiplicadores

En la sección anterior, hemos tratado el caso en el que un vendedor de una opción entra en default y paga sólo una fracción δ de lo que debería pagar. En esta sección, derivaremos otro modelo para la ecuación de Black-Scholes, que trata el caso de que el valor del instrumento se pueda ver modificado debido a la ocurrencia de eventos que pueden suceder varias veces, no sólo una vez como el caso de default anterior.

6.1 Deducción de la ecuación de Black-Scholes con multiplicadores

La deducción del modelo de Black-Scholes con multiplicadores es parecido a la del modelo con riesgo de default, (sección 5.2). Volvemos a suponer un instrumento financiero u, cuyo vendedor debe pagar una cantidad $u(x,T)$, que depende de un activo subyacente x, en un tiempo de expiración T distinto del actual. Ahora trataremos de deducir el precio de otro instrumento financiero v cuyo valor en el tiempo de expiración es un múltiplo $\delta^{Y_T}u$, con Y_T un número entero positivo que especifica el número de veces que ha ocurrido un cierto evento entre la compra del instrumento financiero t_0 y el tiempo de expiración T.

Modelizamos este problema de la siguiente manera.

1. Al igual que en la deducción de 5.2, definimos un proceso de Poisson Y_t con tasa Q. En este caso, Y_t indica el número de veces que ha ocurrido un evento entre t_0 y T, que llamaremos *evento multiplicador*. Tomamos $Y_{t_0} = 0$. Se tiene que:

$$P\{\text{Evento multiplicador en}[t, t+dt]\} \approx Qdt \tag{110}$$

2. Llamaremos $u(x,t)$ al valor de un instrumento financiero estándar, que no depende del número de eventos multiplicadores que ocurren en $[t_0, T]$. El instrumento que tratamos ahora es $v(x,t,y)$, donde y toma valores naturales positivos. Este y viene dado por el proceso de Poisson anterior, por lo que en realidad queremos conocer los valores de $u(x,t,Y_t)$. En el tiempo de expiración, tenemos la relación:

$$v(x,T,y) = \delta^y u(x,T) \tag{111}$$

   Donde $\delta \in [0, +\infty)$. Es decir, cada vez que ocurre un evento multiplicador, el valor de v en el tiempo de expiración se multiplica por δ. Además, señalamos que $v(x,t,0) \neq u(x,t)$, ya que $v(x,t,0)$ es el valor del instrumento asumiendo que no ha ocurrido ningún suceso multiplicador en $[t_0, t)$, pero puede ocurrir algún suceso multiplicador en $[t, T)$. u siempre se calcula ignorando los sucesos multiplicadores.

   Trataremos de derivar una modificación a la ecuación de Black-Scholes que gobierne el valor de $v(x,t,0)$, que llamaremos ecuación de Black-Scholes con multiplicadores.

3. Como en la sección 5.2, asumimos que el movimiento del activo subyacente x es independiente de los sucesos multiplicadores. Por tanto, no modificamos nuestra fórmula para X_t (80).

Ahora deducimos la variante de la ecuación de Black-Scholes para este modelo. Basándonos de nuevo en el Primer Teorema Fundamental de Valoración de Activos [9], asumimos que existe una

medida de valoración que permite escribir el precio actual del instrumento dado por $v\left(x,t,0\right)$ en función de sus posibles valores $v\left(x+dX_t,t+dt,dY_t\right)$ en el instante $t+dt$ como:

$$v\left(x,t,0\right)=\frac{\tilde{E}\left[v\left(x+dX_t,t+dt,dY_t\right)|X_t=x,Y_t=0\right]}{1+rdt} \tag{112}$$

$\tilde{E}$ denota el valor esperado según esta medida de valoración de riesgo neutro, que tiene en cuenta todos los escenarios posibles, sea cual sea el número de eventos multiplicadores. La esperanza se calcula respecto a las variables aleatorias dX_t, (la variación estocástica del precio del subyacente) y dY_t, que es la que indica el número de eventos multiplicadores. r es el tipo de interés sin riesgo, que aparece porque la medida utilizada es una medida de riesgo neutro.

Recordamos que $v(x,t,0)$ es el precio del instrumento financiero dado que no ha ocurrido ningún evento multiplicador hasta t, y para calcular este valor correctamente debemos tener en cuenta la posibilidad de que ocurran eventos multiplicadores en tiempos posteriores. Ahora desarrollamos la esperanza:

$$\begin{aligned}\tilde{E}\left[v\left(x+dX_t,t+dt,dY_t\right)|X_t=x,Y_t=0\right]=&\underbrace{P(dY_t=1)}_{Qdt}\tilde{E}\left[v\left(x+dX_t,t+dt,dY_t\right)|X_t=x,dY_t=1\right]\\&+\underbrace{P(dY_t=0)}_{1-Qdt}\tilde{E}\left[v\left(x+dX_t,t+dt,dY_t\right)|X_t=x,dY_t=0\right]\end{aligned} \tag{113}$$

Donde estamos tomando un intervalo dt lo suficientemente pequeño como para que las probabilidades de $dY_t>1$ sean despreciables. En el caso de que no ocurra ningún evento multiplicador en dt $(dY_t=0)$, nos encontraremos en $t+dt$ con los valores $v\left(x+dx,t+dt,0\right)$, pues en ese caso no habría ocurrido default hasta $t+dt$ y por lo tanto los valores vienen dados por la función $v\left(\cdot,\cdot,0\right)$. En el caso de que ocurra un evento multiplicador en el intervalo $[t,t+dt)$ $(dY_t=1)$, se tiene que:

$$\tilde{E}\left[v\left(x+dX_t,t+dt,dY_t\right)|X_t=x,dY_t=1\right]=\delta\hat{E}\left[v\left(x+dX_t,t+dt,0\right)|X_t=x\right] \tag{114}$$

Ya que, una vez que condicionamos a que ocurre el evento multiplicador en dt, el valor del instrumento se ve multiplicado por δ al finalizar este intervalo. $\hat{E}$ se refiere a los escenarios posibles una vez haya sucedido o no el evento multiplicador en dt, y dichas esperanzas se calculan condicionadas a $X_t=x$, aunque no lo escribamos explícitamente. Esta esperanza $\tilde{E}$ se calcula según la medida de riesgo neutro para el activo subyacente introducida en 80. Esto es válido siempre que la evolución del activo subyacente sea independiente del evento multiplicador, que es la tercera hipótesis de nuestro modelo. Sustituyendo en 113, se llega a:

$$\begin{aligned}\tilde{E}\left[v\left(x+dX_t,t+dt,dY_t\right)|X_t=x,Y_t=0\right]=&Qdt\hat{E}\left[\delta v\left(x+dX_t,t+dt,0\right)\right]\\&+\left(1-Qdt\right)\hat{E}\left[v\left(x+dX_t,t+dt,1\right)\right]\end{aligned} \tag{115}$$

Reescribiendo:

$$\begin{aligned}\tilde{E}\left[v\left(x+dX_t,t+dt,dY_t\right)\right]&=\left(\delta Q\,dt+\left(1-Q\,dt\right)\right)\hat{E}\left[v\left(x+dX_t,t+dt,0\right)\right]\\&=\left(1-\left(1-\delta\right)Qdt\right)\hat{E}\left[v\left(x+dX_t,t+dt,0\right)\right]\end{aligned} \tag{116}$$

La fórmula de valoración 89 es ahora:

$$v(x,t,0) = \frac{(1-(1-\delta)\,Qdt)\,\hat{E}\left[v\left(x+dX_t,t+dt,0\right)\right]}{1+rdt} \approx \frac{\hat{E}\left[v\left(x+dX_t,t+dt,0\right)\right]}{(1+rdt)\,(1+(1-\delta)\,Qdt)}$$
$$\approx \frac{\hat{E}\left[v\left(x+dX_t,t+dt,0\right)\right]}{1+(r+(1-\delta)\,Q)\,dt} \tag{117}$$

Donde en las aproximaciones simplemente ignoramos términos dt^2. Esta ecuación tiene la misma fórmula que utilizamos en la valoración de $u(x,t)$ en el caso de Black-Scholes estándar (83), sólo que ahora el denominador cambia de $1+rdt$ a $1+(r+(1-\delta)\,Q)\,dt$. Por tanto, podemos seguir el mismo desarrollo que en 5.1, llegando a la ecuación de Black-Scholes con multiplicadores:

$$\partial_t v + xr\partial_x v + x^2\frac{\sigma^2}{2}\partial_{xx}v - [r+(1-\delta)\,Q]\,v = 0 \tag{118}$$

Donde el v de la ecuación hace referencia a $v(x,t,0)$. A partir de ahora, usamos la notación $v(x,t) \stackrel{\text{def}}{=} v(x,t,0)$. Es importante notar que Q puede depender de x, t, v e incluso otros parámetros del mercado como la tasa de interés r, o la volatilidad σ_{BS}. En nuestro modelo sólo tomaremos Q dependiente de v según la fórmual 98, pero la ecuación deducida es válida para cualquier Q.

6.2 El método Deep BSDE aplicado a la ecuación de Black-Scholes con multiplicadores

Sea el siguiente problema:

$$\begin{array}{cc} \partial_t v + xr\partial_x v + x^2\frac{\sigma^2}{2}\partial_{xx}v - [r+(1-\delta)\,Q\,(v)]\,v = 0 & (x,t)\in\mathbb{R}\times[t_0,T] \\ v(x,T) = \max(x-E,0) & x\in\mathbb{R} \end{array} \tag{119}$$

Se trata de una opción "Call" con multiplicadores: su precio se puede multiplicar por δ dependiendo de la ocurrencia de los sucesos multiplicadores. Para el valor de $Q(v)$, utilizamos la fórmula 98 que también utilizamos en el riesgo de default, y tomamos $\delta = \frac{2}{3}$. Este problema se trata de una especie de riesgo de default, en el cual el valor de la opción se puede multiplicar varias veces por δ, no sólo una vez. Es decir, sería una especie de riesgo de default variable, en el cual no sabemos exactamente la fracción del valor inicial que vamos a recibir, como ocurría en el riesgo de default anterior, que tenía una fracción fija. El resto de parámetros y los detalles de la implementación son los mismos que los explicados en 8.3 y 8.4. A continuación, exponemos los resultados para los precios iniciales, comparándolos con los precios para la ecuación de Black-Scholes estándar sin multiplicadores, dada por 101. De nuevo advertimos que los valores no deben de coincidir, ya que la u se obtiene sin tener en cuenta los eventos multiplicadores.

Para el valor de v_0, obtenemos $v_0 = 6.872$, mientras que el valor de la ecuación de Black-Scholes estándar es $u_0 = 8.916$, con una diferencia del 22.9%. Para el gradiente, se obtiene $\partial_x v_0 = 0.445$, mientras que el valor del gradiente de la ecuación de Black-Scholes estándar es $\partial_x v_0 = 0.5792$, con una diferencia relativa del 23.3%.

Estas diferencias tan grandes se deben de nuevo a la alta probabilidad de que ocurran sucesos multiplicadores, y por tanto a las probabilidades de que el valor del instrumento se multipliquen por

δ, δ^2, etc... La intensidad viene dada por la Q de la fórmula 98 con los parámetros descritos en 8.4. Para hacernos una idea de la posibilidad de que ocurran los sucesos multiplicadores, aproximamos $Q \approx 0.75$. Entonces:

$$P\{\text{No ocurra ningún suceso multiplicador}\} = P\{Y_T = 0\} \approx e^{-Q(T-t_0)} \simeq 0.47 \tag{120}$$

$$P\{\text{Ocurra un suceso multiplicador}\} = P\{Y_T = 1\} \approx Qe^{-Q(T-t_0)} \simeq 0.35 \tag{121}$$

$$P\{\text{Ocurran dos sucesos multiplicadores}\} = P\{Y_T = 1\} \approx \frac{Q^2}{2}e^{-Q(T-t_0)} \simeq 0.13 \tag{122}$$

Es decir, hay aproximadamente una probabilidad de 0.47 de que obtengamos el pago de la ecuación de Black-Scholes estándar, una probabilidad de 0.35 de que obtengamos δ por dicho pago, una probabilidad de 0.13 de obtener δ^2 por dicho pago, etc... Como $\delta = \frac{2}{3} < 1$, es lógico que el precio inicial v_0 que paguemos por dicho instrumento sea menor a u_0. Además, es menor al valor solución de la ecuación de Black-Scholes con riesgo de default obtenido en la sección 5.4, debido a que con el modelo de default sólo permitimos que el precio se pueda multiplicar por δ una vez.

6.3 Soluciones a la ecuación de Black-Scholes con multiplicadores para distintas intensidades

Ahora mostraremos los resultados para la solución de 119 tomando distintas intensidades Q constantes, los parámetros para el método Deep BSDE dados por 8.3 y $\delta = \frac{2}{3}$. En la figura 14 se muestran los valores del precio inicial $v_0 = v(\xi, t_0)$ y en la figura 15 se muestran los valores del gradiente inicial $\partial_x v_0 = \partial_x v(\xi, t_0)$. En este caso los valores decrecen con la intensidad, debido a que cada vez que ocurre un suceso multiplicador (que ocurre según un proceso de Poisson con intensidad Q), nuestro pago se multiplica por $\delta < 1$. Pero ahora, los valores no tienden a δu_0 o $\delta \partial_x u_0$ como ocurría en la ecuación de Black-Scholes con riesgo de default (sección 5.5), sino que los valores tienden a cero, debido a que con cada suceso multiplicador vamos multiplicando por δ nuestro posible beneficio. Así, si $Q \longrightarrow \infty$, entonces $v(x, t_0) \longrightarrow \delta^\infty u(x, t_0) = 0$.

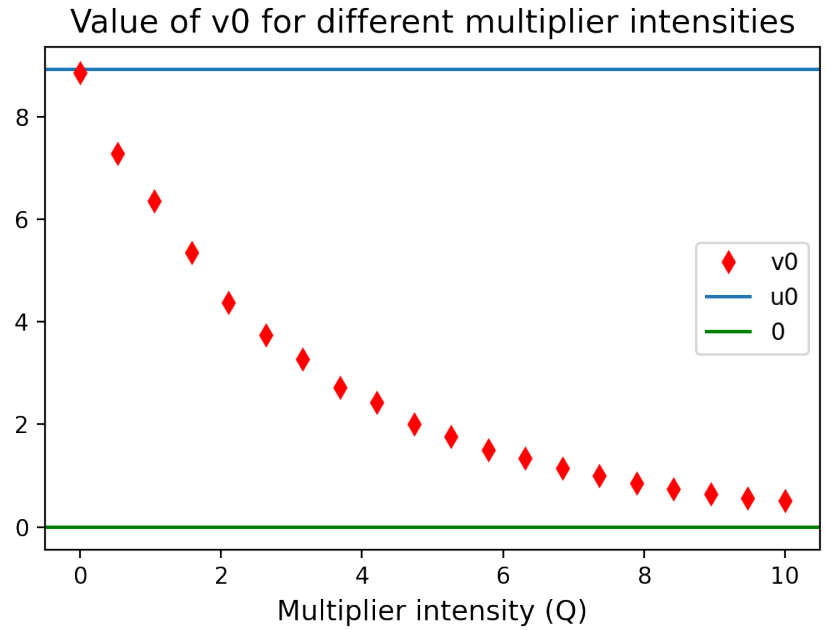

Figure 14: Valores de $v_0 = v(\xi, t_0)$ $(v0)$ para el problema 119 con los parámetros de 8.4, $\delta = \frac{2}{3}$ y varios valores de Q constantes.

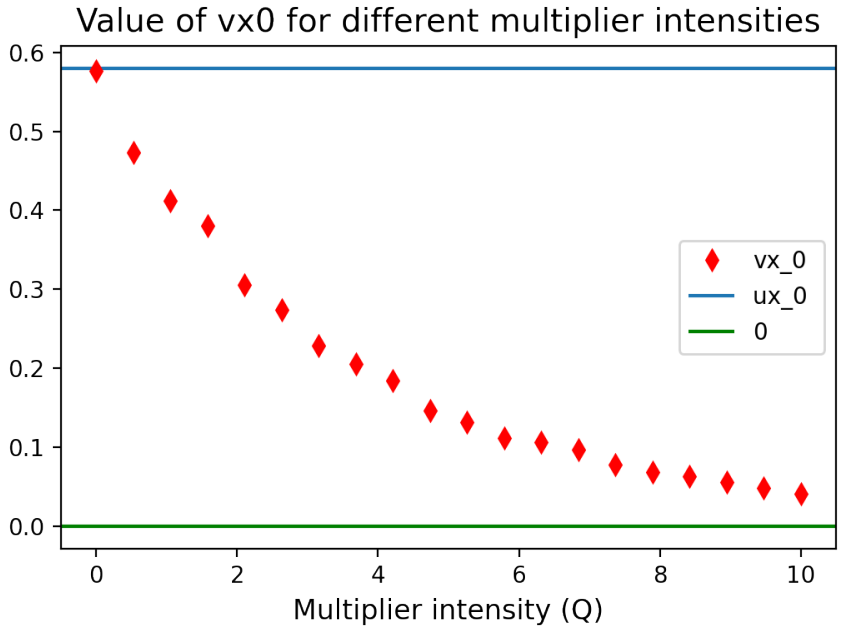

Figure 15: Valores de $\partial_x v_0 = \partial_x v(\xi, t_0)$ $(vx0)$ para el problema 119. Ver 14.

7 Conclusiones

En este TFG hemos analizado el método Deep BSDE propuesto en el artículo [1], que sirve para resolver una gran clase de ecuaciones diferenciales parabólicas no lineales. Para ello, hemos aplicado el método a dos ecuaciones lineales (la ecuación de difusión y la ecuación de Black-Scholes estándar), para las cuales existe solución analítica conocida, que nos han servido para testear la eficiencia del método. Después, hemos aplicado el método a dos ecuaciones no lineales (ecuación de Black-Scholes con riesgo de default y ecuación de Black-Scholes con multiplicadores), y hemos analizado el efecto que producía esta no-linealidad en la solución de las ecuaciones.

La primera conclusión del TFG es que el método Deep BSDE es capaz de predecir los valores iniciales de la solución de una ecuación diferencial y de su gradiente (u_0 y $\partial_x u_0$) con mucha precisión a partir de la condición final. En los ejemplos analizados, hemos escogido un timestep relativamente largo (0.1)[11], pero a pesar de ello la precisión ha sido muy alta (diferencias relativas en torno al 1% o incluso menos en los casos de la ecuación de difusión y la ecuación de Black-Scholes estándar), que se pueden reducir aumentando el número de pasos temporales N. Por tanto, es esperable que el método Deep BSDE sirva para hallar con precisión u_0 y $\partial_x u_0$ en cualquier ecuación diferencial parabólica semilineal del tipo 63. Además, según la observación que hacemos en la sección 4.2, se puede calcular no sólo el valor de la solución en tiempo t_0 en un punto x, sino todo el perfil de la solución en el tiempo inicial.

Aparte de los gradientes iniciales, con el método Deep BSDE es posible aproximar el gradiente de la solución u para cualquier tiempo discretizado t_i entre t_0 (tiempo inicial) y T (tiempo final). Para comprobar esto, hemos utilizado el error cuadrático entre el gradiente real y nuestra estimación del gradiente en las ecuaciones de difusión y la ecuación de Black-Scholes estándar. Además, hemos normalizado estos errores con el error cuadrático de la esperanza del gradiente exacto, para entender cómo de buenos son los estimadores (62). El resultado es que los estimadores son en torno a 10 veces mejores que la media para inferir valores del gradiente, llegando a poder ser hasta casi 20 veces mejores. Por ello, podemos concluir que el método Deep BSDE es útil para hallar estos valores. Cabe destacar que estos estimadores del gradiente son válidos para un continuo de puntos, y que se han hallado simplemente optimizando la función de coste sobre la condición final de los caminos (77).

En este TFG, también hemos tratado problemas de Matemáticas Financieras. Hemos diseñado dos modelos novedosos para tratar dos instrumentos financieros: uno de ellos que incorpora el riesgo de default de un vendedor (ecuación de Black-Scholes con riesgo de default), y el otro incorpora la posibilidad de que el valor de un instrumento se multiplique por una cantidad δ una o varias veces (ecuación de Black-Scholes con multiplicadores).

- La ecuación de Black-Scholes con riesgo de default (95) puede ser especialmente útil en finanzas bajo varios escenarios. Supongamos que compramos una opción financiera a una entidad, y que se teme que dicha entidad entre en default y no cumpla con su obligación por entrar en quiebra. En el caso de que una empresa entre en quiebra, es común celebrar un concurso de acreedores para liquidar el patrimonio de la entidad y con ello pagar las deudas que tiene con todos sus acreedores. Por lo general, los acreedores (en este caso, el comprador de la opción sería uno de ellos) reciben una fracción $\delta \in [0, 1)$ de lo que debieran recibir si no se

[11] Esto equivale a 0.1 años en la ecuación de Black-Scholes.

hubiese producido la quiebra. Por tanto, este modelo se puede aplicar tomando como δ una estimación de esta fracción del pago. Se puede incluso tomar $\delta = 0$, si se estima que no se recibe ningún pago cuando el vendedor entra en quiebra. Otro ejemplo de aplicación de este modelo es que el comprador de la acción contrate un seguro que, bajo el caso de quiebra del vendedor, le pague una fracción $\delta \in [0, 1)$ de lo que le debiera pagar. Esto es un tipo de instrumento financiero parecido a los *Credit Default Swap*, y en este caso u_0 corresponde al precio que deberíamos pagar al vendedor de la opción (*premium*) más el precio del seguro.

- La ecuación de Black-Scholes con multiplicadores (118) también puede ser muy útil bajo otros escenarios. El ejemplo más típico es el de una opción exótica, cuyo *payoff* (pago en el tiempo final) se ve multiplicado por una cantidad $\delta \in [0, +\infty)$ en el caso de que ocurra un cierto evento, que denominamos evento multiplicador. Si estos eventos multiplicadores tienen más posibilidad de ocurrir cuanto mayor sea el intervalo de tiempo entre el momento de compra t_0 y el momento de ejercicio T (siguiendo un proceso de Poisson), entonces nuestra ecuación 118 modeliza este tipo de opción. Por ejemplo, supongamos que un inversor posee acciones en un activo x', y quiere cubrirse frente a posibles pérdidas del valor en bolsa de x'. En ese caso, puede contratar una opción sobre otro activo x, tal que cada vez que ocurra una caída en el precio de x', reciba el doble del valor de la opción que tiene sobre x. En este caso se utilizaría nuestro modelo tomando $\delta = 2$ y suponiendo que las caídas del valor en x' siguen un proceso de Poisson. Instrumentos de este tipo permiten una cobertura sobre carteras financieras complejas, que son muy habituales en la actualidad.

Una observación importante sobre el método Deep BSDE es que no sólo permite hallar el precio de la opción u_0 en el momento inicial, sino que también permite hallar su gradiente $\partial_x u_0$ y los gradientes para tiempos posteriores. Esto es tremendamente útil en finanzas, ya que dichos gradientes se utilizan para cubrir riesgos (*hedging*) [7]. Cuando añadimos términos no lineales a la ecuación de Black-Scholes (como el riesgo de default o los multiplicadores), el valor del gradiente cambia, y por ello la cobertura es diferente. No introducir estos términos en la ecuación supone no conocer el gradiente exacto, y por ello verse expuesto a riesgos que pueden conllevar grandes pérdidas de dinero para las entidades financieras y para la sociedad, como ocurrió en la crisis de 2008. El método Deep BSDE permite resolver esta ecuación de Black-Scholes con los términos no lineales para conseguir así precios y coberturas más adecuados, y prevenirse mejor ante movimientos adversos del mercado. Por ejemplo, en la ecuación de Black-Scholes con riesgo de default, podemos incluso hacer depender la probabilidad de default Q de parámetros del mercado como la tasa de interés, y así prepararnos frente a posibles quiebras de empresas por condiciones desfavorables del mercado, como colapsos o crisis financieras.

8 Apéndice

La implementación de los códigos está disponible en este repositorio de Github (link).

8.1 Implementación del método Deep BSDE en Keras y Tensorflow

A la hora de implementar el método Deep BSDE, es importante notar que el algoritmo de optimización Adam [3] necesita conocer el gradiente de la función de coste. En Machine Learning, la función de coste es una composición de muchas funciones elementales, por lo que su gradiente se puede calcular aplicando la regla de la cadena una gran cantidad de veces (técnica conocida como *backpropagation* [4]). La ventaja de utilizar la librería Tensorflow [6] es que esta computación del gradiente se hace automáticamente (diferenciación automática), siempre que se utilizen las estructuras de datos propias de Tensorflow (tensores) y funciones implementadas en las librerías de Tensorflow. Por ejemplo, para resolver la ecuación de Black-Scholes con riesgo de default 95, es necesario introducir la función $u(x,t)$ que sigue la fórmula de Black-Scholes (ecuación 101), que a su vez involucra a la función de distribución de una normal. En Tensorflow, no es necesario indicar el gradiente de dicha función, ni cómo aplicar la regla de la cadena para computar el gradiente final, sino que estas operaciones están ya incorporadas. Esta diferenciación automática es imprescindible para tratar estos problemas, por lo que es necesario un software que tenga una diferenciación muy eficiente como Tensorflow.

8.2 Detalles de la implementación del método de Deep BSDE para la ecuación de difusión homogénea

En este apartado explicaremos los detalles de la implementación del Método Deep BSDE aplicado a la resolución del problema 1 en la sección 3.3, basándonos en el método descrito en 3.2.

Como hiperparámetros, escogemos:

- $\xi = 1$ como punto inicial, $t_0 = 0$ y $T = 1$.
- $N = 10$ como número de pasos del tiempo discretizado. Por tanto, $\Delta t = 0.1$. Si se necesita mayor precisión, tomando valores más pequeños de Δt se conseguirían mejores resultados, a cambio de un incremento en el coste computacional.
- El algoritmo de optimización empleado es el Adam [3] con un "learning rate" de 0.001.
- $N_{iter} = 6000$ como número de iteraciones con las que se entrena la red con el algoritmo Adam.
- $M = 3000$ como número de caminos con lo que entrenamos la red neuronal. De ellos, la mitad se utilizarán como set de entrenamiento (los que se utilizan para entrenar la red mediante el algoritmo de optimización), y la otra mitad como set de validación (los que se utilizan para hallar cuál es la función de coste en caminos en los que no se ha entrenado la red). La razón por la que realizamos esta división 50-50 (en vez de otras comunes como 80-20 ó 90-10) es porque la función de coste aumenta mucho en los caminos que se separan mucho del punto inicial ξ (caminos "outlier"). Esto se debe a que, como se aprecia en la figura 5, los aproximadores del gradiente de la red neuronal $h_i(\cdot|\theta_i)$ no son buenos lejos de ξ, porque no hay muchos caminos en los se entrenen, ya que la mayoría cae entre $\xi - \sigma$ y $\xi + \sigma$ (figura 4). Por tanto, para tener valores comparables de la función de coste en el set de entrenamiento y

el set de validación, hacemos una separación 50-50, con la que conseguimos que en ambos sets haya aproximadamente el mismo número de caminos outlier. Si hiciéramos una separación 80-20, habría muchos menos caminos outlier en el set de validación, por lo que la función de coste en dicho set sería mucho menor que la función de coste en el set de entrenamiento, lo que puede llevar a confusión.

- Para cada una de las redes neuronales que aproximan el gradiente en cada punto $h_i(\cdot|\theta_i)$, utilizamos $H = 2$ capas ocultas de tipo "Denso" [4], cada una con $d = 10$ neuronas. La función de activación empleada es la "ReLU" [4], y se aplica normalización por lotes ("Batch Normalization") [4] después de la activación.

8.2.1 Forma de la función de coste para el set de validación

Sobre el set de entrenamiento de la figura 1 (curva azul) la función de coste decrece al principio y después se mantiene casi constante en torno al valor 0.388. Sin embargo, sobre el set de validación (curva naranja), existen grandes variaciones en la función de coste, y la curva no es decreciente. Dicho comportamiento atípico no se debe a ningún fallo de la red neuronal, ni a que la red neuronal sólo sea capaz de ajustar los valores que entrena (overfitting). Su causa está relacionada con que el set de validación contiene caminos sobre puntos de x para los cuales no hay suficiente información en el set de entrenamiento como para que las funciones $h(\cdot|\theta_i)$ aproximen bien el gradiente. Esto se puede observar en la figura 4. Existen ciertos caminos en el set de validación con puntos x que se salen del intervalo $(\xi - 2\sigma, \xi + 2\sigma)$, para los cuales no hay suficientes puntos x cercanos en el set de entrenamiento. Estos caminos "outlier" incrementan mucho la función de coste sobre el set de validación para iteraciones en las cuales las aproximaciones $h_i(\cdot|\theta_i)$ pueden llegar a ser especialmente malas. Pero como son puntos fuera del intervalo $(\xi - 2\sigma, \xi + 2\sigma)$, que es nuestra región de interés, no afectan a nuestros estimadores, a pesar de que la función de coste aumente.

8.3 Detalles de la implementación del método Deep BSDE a la ecuación de Black-Scholes

Para aplicar el método Deep BSDE a la ecuación de Black Scholes, utilizamos los siguientes parámetros:

- $\xi = 100$ y $E = 100$, siendo ξ el valor inicial del activo subyacente y E el precio de ejercicio.
- $t_0 = 0$ y $T = 1$, donde estamos midiendo el tiempo en años. $r = 0.02$ es el tipo de interés (un 2% anual, que es un valor típico).
- $\sigma_{BS} = 0.02$ es la volatidad del activo subyacente que aparece en la ecuación de Black-Scholes.
- $N = 10$, lo que implica $\Delta t = 0.1$. A pesar de ser un intervalo temporal bastante grande (t se mide en años, ya que r es el tipo de interés anual), los resultados son bastante buenos.
- El algoritmo de optimización vuelve a ser el Adam, con un "learning rate" de 0.001.
- Utilizamos $N_{iter} = 6000$ iteraciones y $M = 5000$ caminos, la mitad de los cuales se utilizan para entrenar la red (set de entrenamiento) y la otra mitad para validar resultados (set de validación).

- Para cada subred neuronal que aproxima el gradiente $h(\cdot|\theta_i)$, utilizamos $H = 3$ capas ocultas de tipo "Denso" cada una con $d = 10$ neuronas. La activación vuelve a ser "ReLU" y se aplica "Batch Normalization".
- Para mejorar la convergencia, el peso inicial de θ_{u_0} se inicializa en $\max(\xi - E, 0)$, que se espera que esté cerca de un valor razonable del premium. También es muy importante inicializar $\theta_{\partial_x u_0}$ entre 0 y 1, ya que la derivada de la fórmula de Black-Scholes (106) siempre toma valores en este intervalo.

8.4 Detalles de la implementación del método Deep BSDE a la ecuación de Black-Scholes con riesgo de default y con multiplicadores

La configuración de la implementación del método Deep BSDE a la ecuación Black-Scholes con default risk o con multiplicadores es la misma que la del caso estándar (sección 8.3), sólo que ahora introducimos una función $Q(v)$ y un valor δ. La función $Q(v)$ viene dada por la fórmula 98 con los parámetros: $\gamma^{(l)} = 0.6$, $\gamma^{(h)} = 0.9$, $\beta = 0.5$ y $z \simeq 8.916$. z se ha tomado igual al valor de u_0 en el caso sin default. Para δ tomamos $\delta = 2/3$. Para acelerar la convergencia, inicializamos el peso θ_{u_0} en el valor $u(\xi, t_0)$ dado por la fórmula 101 y el peso $\theta_{\partial_x u_0}$ en el valor $\partial_x u(\xi, t_0)$ dado por la fórmula 106.

References

[1] Jiequn Han, Arnulf Jentzen, y Weinan E (2018). *Solving high-dimensional partial differential equations using deep learning*. Proceedings of the National Academy of Sciences Aug 2018, 115 (34) 8505-8510; DOI: 10.1073/pnas.1718942115.

[2] Sandro Salsa (2007). *Partial Differential Equations in Action. From modelling to theory*. Springer Verlag (Primera Edición).

[3] Diederik P. Kingma y Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980

[4] Aurélien Géron (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc.

[5] Chollet, François y otros (2015). *Keras*. `https://keras.io`.

[6] Martín Abadi y otros (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. Software disponible en `tensorflow.org`.

[7] Paul Wilmott, Sam Howison y Jeff Dewynne (1995). *Mathematics of Financial Derivatives*. Cambridge University Press, Primera edición.

[8] Thomas Mikosch (1998). *Elementary Stochastic Calculus with Finance in view*. World Scientific, Primera edición.

[9] Andrea Pascucci (2011). *PDE and Martingale Methods in Option Pricing*. Springer-Verlag

[10] Tomas Björk (2004). *Arbitrage Theory in Continuous Time* Oxford University Press. Primera edición.