Universidad de San Andrés

Trabajo de Graduación

Escuela de Negocios

Licenciatura en Finanzas

Redes Neuronales Profundas para la Valuación de Derivados Financieros en Altas Dimensiones

Autores:
Federico Matías Glancszpigel
(legajo 29070)
Facundo Ignacio Bonfanti
Borgia (legajo 29026)

Mentor:
Pablo Alejandro Macri

Provincia de Buenos Aires, Argentina,

29 de Julio de 2021.

Índice

1. Introducción **2**

2. Revisión literaria **4**

3. Metodología **6**
3.1. Ecuaciones con generadores lineales 6
3.2. Ecuaciones con generadores no lineales 8
3.2.1. Problema con distintas tasas de interés para tomar y prestar dinero 8
3.2.2. Problema incorporando riesgo de *default* 10
3.3. Método *Forward Deep BSDE* 10
3.4. Método *Backward Deep BSDE* 12
3.4.1. Valuación con ejercicio anticipado 14

4. Redes Neuronales **16**
4.1. Estructura de una red neuronal 16
4.2. Optimizadores 17
4.3. Normalización de lotes 18
4.4. Tasa de aprendizaje con decaimiento exponencial 19
4.5. Estructura de las sub-redes 20

5. Resultados numéricos **22**
5.1. Problemas con generadores lineales 24
5.1.1. Pruebas de convergencia 25
5.1.2. Ajuste de parámetros 26
5.1.3. Pruebas de Eficiencia 32
5.2. Problemas con generadores no lineales 35

6. Conclusión **41**

Appendices **46**

Apéndice A. Demostración de la PDE de Black, Scholes y Merton multidimensional **46**

Índice de figuras

1. Diagrama del método *Forward Deep BSDE* (FDB) 12
2. Diagrama del método *Backward Deep BSDE* (BDB) 14
3. diagrama de una capa dentro de una ANN. 16
4. diagrama de una red neuronal profunda con dos capas ocultas. 17
5. diagrama de una sub-red. 20
6. Convergencia de los métodos *Forward* y *Backward* 23
7. Variación del ECM con el aumento de TBS e ITs en los métodos FDB y BDB 27
8. Variación del tiempo de convergencia con el aumento de TBS e ITs en los métodos FDB y BDB 27
9. Variación del ECM con el aumento de LR en los métodos FDB y BDB 28
10. Variación del tiempo de convergencia con el aumento de LR en los métodos FDB y BDB 28
11. Variación del ECM y el tiempo de convergencia con el aumento de la discretizacion temporal en los métodos FDB y BDB . . . 29
12. Variación del ECM con el aumento de la cantidad de neuronas y la cantidad de capas ocultas en la DNN en los métodos FDB y BDB 29
13. Variación del tiempo de convergencia con el aumento de la cantidad de neuronas y la cantidad de capas ocultas en la DNN en los métodos FDB y BDB 30
14. Variación del ECM y el tiempo de convergencia con el aumento de TBS e ITs en el método LSBD 30
15. Variación del ECM y el tiempo de convergencia con el aumento de LR en el método LSBD 31
16. Variación del ECM y el tiempo de convergencia con el aumento de la cantidad de neuronas y la cantidad de capas ocultas en la DNN en el método LSBD. 31

Índice de tablas

1. Resultados de la valuación del problema A. Métodos FDB y BDB. 25
2. Resultados de la valuación del problema B. Métodos FDB y BDB. 26
3. Resultados de la valuación del problema C. Método LSBD. . . 26
4. Resultados de la valuación del problema D. Método LSBD. . . 26
5. Prueba de eficiencia en comparación con MC. Métodos FDB y BDB 32
6. Resultados de la prueba de eficiencia para MC 33
7. Prueba de eficiencia aumentando la cantidad de dimensiones. Métodos FDB, BDB y MC 33
8. Valores de referencia para el cálculo del error relativo en la Tabla 7 33
9. Prueba de eficiencia del método LSBD en comparación LSMC. 34
10. Prueba de eficiencia aumentando la cantidad de dimensiones. Métodos LSBD y LSMC 34
11. Valuación de una opción 100D, problema de distintas tasas de interés. Métodos FDB y BDB 37
12. Valuación de una opción 1D, problema de distintas tasas de interés. Métodos FDB y BDB 37
13. Valuación de una opción 1D, problema de diferentes tasas de interés. Método LSBD 38
14. Valuación de una opción 100D, problema de riesgo de *default*. Métodos FDB y BDB 38
15. Valuación de una opción 1D, problema de riesgo de *default*. Métodos FDB y BDB 38
16. Valuación de una opción 1D, problema de riesgo de *default*. Método LSBD 39
17. Valuación de una opción 100D, problema de distintas tasas de interés. Método LSBD 39
18. Valuación de una opción 100D, problema de riesgo de *default*. Método LSBD 39

Siglas

BDB *Backward Deep BSDE*.

BSDE Ecuaciones diferenciales estocásticas hacia atrás o *Backward Stochastic Differential Equations* en inglés.

BSM Black, Scholes y Merton.

DNN Red neuronal profunda o *Deep Neural Network* en inglés.

ECM Error cuadrático medio.

FDB *Forward Deep BSDE*.

FDM Diferencias Finitas o *Finite Difference Method* en inglés.

ITs Cantidad/numero de iteraciones.

LR Tasa de aprendizaje o *Learning Rate* en inglés.

LSBD *Least Squares Backward DNN*.

LSMC *Least Squares* Monte Carlo.

MC Monte Carlo.

PDE Ecuaciones en derivadas parciales o *Partial Differential Equations* en inglés.

TBS Tamaño del lote de entrenamiento o *Training Batch Size* en inglés.

Resumen El presente trabajo propone estudiar una serie de métodos numéricos basados en Aprendizaje Profundo, o *Deep Learning* en inglés, para la valuación de derivados financieros en contextos más generales, con el objetivo de determinar la factibilidad y eficiencia de dichos métodos. En particular, interesa determinar la conveniencia de utilizar estos métodos para la valuación de instrumentos en altas dimensiones. Para ello, hemos realizado experimentos numéricos detallados donde pudimos comprobar la precisión y eficiencia de los métodos propuestos para resolver ecuaciones diferenciales estocásticas:

- con generadores lineales como las que surgen al valuar derivados financieros bajo los supuestos de Black, Scholes y Merton.
- con generadores no lineales como las que surgen al valuar derivados en condiciones más generales que la anterior como, por ejemplo, se encuentran al usar distintas tasas de interés para colocar o para pedir dinero, o al considerar escenarios de *default*.

Los métodos de Aprendizaje Profundo que hemos estudiado ofrecen una de las pocas alternativas para la valuación de derivados con generadores no lineales en altas dimensiones. En esta tesis, además de comprobar su eficacia comparando con resultados conocidos, hemos sido capaces de calcular por primera vez derivados de ejercicio temprano con generadores no lineales en alta dimensión.

1. Introducción

Numerosa es la cantidad y tipos de derivados financieros que existen hoy en día, tanto en los mercados primarios como secundarios. A lo largo de la historia, estos instrumentos fueron ganando popularidad en el mundo de las finanzas y, al mismo tiempo, nuevos y más complejos instrumentos fueron desarrollándose.

En general, la valuación de derivados financieros ha sido ampliamente estudiada en la academia y en la industria a través de métodos numéricos como el árbol binomial, Monte Carlo (MC) y métodos para resolver ecuaciones en derivadas parciales o *Partial Differential Equations* en inglés (PDEs) como el método de diferencias finitas o *Finite Difference Method* en inglés (FDM). No obstante, tanto el árbol binomial como FDM son inviables para resolver problemas de más de 2 dimensiones, esto es conocido en la literatura como "la maldición de la dimensionalidad" y genera que MC sea el único método utilizado para valuar instrumentos multi-dimensionales.

En años recientes, surgieron métodos numéricos basados en redes neuronales para resolver PDEs en altas dimensiones reformulándolas como ecuaciones diferenciales estocásticas hacia atrás o *Backward Stochastic Differential Equations* en inglés (BSDEs). La razón principal para estudiar BSDEs en finanzas cuantitativas es que, por un lado, permiten valuar derivados financieros con subyacentes de dimensión alta y, por otro, porque permiten hacerlo en condiciones más generales que las que establece el formalismo de Black, Scholes y Merton (BSM). Por ejemplo, en mercados incompletos o cuando el interés al que se pide es distinto al interés al que se presta dinero, entre otros casos. En general, el generador de una BSDE es no lineal y su solución no puede escribirse de forma explícita por medio de una expectación condicionada. En esos casos cuando la dimensión del subyacente es alta, solo se pueden usar las BSDEs.

Distintos trabajos presentaron resultados alentadores donde mostraban la efectividad de estos métodos para resolver el problema de la dimensionalidad asegurando que eran candidatos para remplazar a MC en diversas tareas de valuación. No obstante, la literatura es todavía acotada y no termina de esclarecerse la verdadera efectividad de estos algoritmos.

En esta línea, el presente trabajo estudia la capacidad de los nuevos métodos basados en Aprendizaje Profundo o *Deep Learning* en inglés para resolver BSDEs y valuar derivados financieros en altas dimensiones. En primer lugar, analizamos la factibilidad de estos algoritmos en problemas donde el generador de la ecuación es lineal (en el formalismo de BSM) y comparamos la eficiencia computacional con los métodos de MC para opciones europeas y *Least Squares* Monte Carlo (LSMC) (Longstaff and Schwartz, 2001) para op-

ciones americanas/bermuda. En segundo lugar, estudiamos la factibilidad de los métodos propuestos para la valuación de problemas donde el generador de la ecuación es no lineal. De esta manera, el objetivo es determinar si dichos métodos resultan ser factibles y lo suficientemente eficientes en comparación con los métodos de Monte Carlo como para justificar el despliegue técnico que requiere su construcción y, por otro lado, si dicha factibilidad y eficiencia puede extenderse a problemas con generadores no lineales.
Este documento esta estructurado de la siguiente manera: la Sección 2 contiene la revisión de la literatura; en la Sección 3 detallamos la metodología y los fundamentos teóricos detrás de los algoritmos; en la Sección 4 describimos la estructura y herramientas a utilizar para la construcción de la red neuronal; en la Sección 5 presentamos resultados numéricos para problemas lineales y no lineales, y, por último, la Sección 6 contiene las conclusiones a partir de los resultados obtenidos.

2. Revisión literaria

La literatura de BSDEs se remonta al trabajo de Pardoux and Peng (1992), en el cual los autores plantean que PDEs parabólicas semi-lineales pueden reformularse como BSDEs, estableciendo una generalización del conocido teorema de Feynman-Kac, que juega un rol fundamental en la teoría básica de valuación de derivados financieros (Shreve, 2005). A partir de este trabajo, surge una extensa literatura de métodos numéricos para resolver BSDEs (Chevance, 1997; Zhang et al., 2004; Chassagneux, 2014; Chassagneux and Richou, 2015; E et al., 2019; por mencionar algunos).

En esta tesis, nos interesa enfocarnos en una serie de trabajos recientes que utilizan algoritmos basados en aprendizaje profundo para resolver PDEs en altas dimensiones a partir de la reformulación de Pardoux and Peng (1992). Estos métodos son conocidos como *Deep BSDE* y se explican en detalle en las próximas secciones.

E et al. (2017) y Han et al. (2018) son los primeros trabajos que registramos de la literatura de *Deep BSDE*. Los autores plantean el método conocido como *Forward Deep BSDE* (FDB), que consiste en aproximar para cada paso temporal la solución de la BSDE y su gradiente por medio de una red neuronal. El algoritmo realiza una propagación hacia adelante y minimiza las diferencias al cuadrado entre el valor terminal de la solución estimada por la red neuronal profunda o *Deep Neural Network* en inglés (DNN) y la condición terminal de la BSDE. En este caso, el valor inicial de la función es un parámetro que la red neuronal obtiene mediante la optimización de una función de error.

El método anterior puede ser utilizado como una herramienta de ingeniería financiera para valuar derivados en altas dimensiones, pero únicamente sirve para instrumentos de estilo europeo. Recientemente, Wang et al. (2018) proponen una modificación al algoritmo original para valuar derivados con ejercicio anticipado. Los autores formulan el método *Backward Deep BSDE* (BDB) que a diferencia del método FDB, realiza una propagación hacia atrás partiendo de la condición final de la BSDE y minimiza la varianza del valor inicial de la solución estimada por la DNN. A diferencia del algoritmo propuesto por E et al. (2017) y Han et al. (2018), en el método BDB el valor inicial de la función no es un parámetro a optimizar por la red, sino que se estima como un promedio del conjunto de valores iniciales evaluados en el óptimo de la red.

Más tarde, Liang et al. (2019) proponen modificaciones al algoritmo de Wang et al. (2018), argumentando que los autores no presentan resultados comparativos con otros métodos tradicionales como MC y por lo tanto la validez del método es poco clara. Por un lado, Liang et al. (2019) modifican la dis-

cretización de la BSDE. Por otro lado, incorporan un factor exógeno para determinar el ejercicio anticipado en derivados de estilo bermuda o americano. Particularmente, los autores hacen uso del método de Longstaff and Schwartz (2001) para estimar este factor exógeno. El algoritmo es conocido como *Least Squares Backward DNN* (LSBD).

Los trabajos mencionados anteriormente, son el núcleo de la literatura de *Deep BSDE*. Identificamos además, otros 16 artículos referidos al campo de investigación que se ocupa de la valuación de instrumentos financieros en alta dimensionalidad. En términos generales, la literatura se puede dividir en aquellos trabajos que valúan derivados de estilo europeo en un marco de generadores lineales (Liang et al., 2019, 2021; Yu et al., 2019; Hientzsch, 2019; Fujii et al., 2019; Ganesan et al., 2020), aquellos que abordan la valuación de derivados con ejercicio anticipado en un marco de generadores lineales (Wang et al., 2018; Liang et al., 2019, 2021; Hientzsch, 2019; Huré et al., 2020; Fujii et al., 2019; Chen and Wan, 2021; Becker et al., 2019) y aquellos que valuan derivados de estilo europeo en un marco de generadores no lineales (E et al., 2017; Han et al., 2018; Yu et al., 2020; Takahashi et al., 2021; Fujii et al., 2019; Henry-Labordere, 2017; Gnoatto et al., 2020; Güler et al., 2019; Zhang and Cai, 2020; Kapllani and Teng, 2020; Raissi, 2018; Beck et al., 2019; Huré et al., 2020). En esta línea, nuestro trabajo no solo incluye resultados numéricos de los problemas mencionados anteriormente sino que también es uno de los primeros en abordar la valuación de derivados con ejercicio anticipado en un marco de generadores no lineales.

3. Metodología

3.1. Ecuaciones con generadores lineales

Consideramos un mercado con $n+1$ activos $A_t^0, A_t^1, ..., A_t^n$, representados por procesos estocásticos $A_t^i : [0,T] \times \Omega \to \mathbb{R}$, para todo $i \geq 1$. A_t^0 es el activo libre de riesgo, y evoluciona de acuerdo a

$$dU_t^0 = \frac{dA_t^0}{A_t^0} = r_t dt,$$

donde r_t es la tasa libre de riesgo.
El resto de los activos, $A_t^1, A_t^2, ..., A_t^n$, son los activos riesgosos del mercado y, por conveniencia, los agrupamos como $X_t = (A_t^1, ..., A_t^n)' \in \mathbb{R}^{n\times 1}$ (el símbolo $'$ hace referencia a la transpuesta). Estos activos obedecen la siguiente ecuación diferencial estocástica:

$$dU_t = \frac{dX_t}{X_t} = \mu_t dt + \sigma_t dW_t, \tag{1}$$

donde $\mu_t = (\mu_t^1, ..., \mu_t^n) \in \mathbb{R}^{n\times 1}$, $\sigma_t = \begin{pmatrix} \sigma_{11} & ... & \sigma_{1m} \\ ... & ... & ... \\ \sigma_{n1} & ... & \sigma_{nm} \end{pmatrix} \in \mathbb{R}^{n\times m}$ y $W_t = (W_t^1, ..., W_t^m)' \in \mathbb{R}^{m\times 1}$ es un movimiento Browniano m-dimensional sobre un espacio de probabilidad $(\Omega, \mathcal{F}, \mathbb{P})$ y en un horizonte temporal finito $[0,T]$, $T > 0$. Además, denotamos con $(\mathcal{F}_t)_{0\leq t\leq T}$ la filtración aumentada generada por W_t. En particular, n (i.e. la cantidad de activos riesgosos) y m (i.e. la dimensión del movimiento Browniano) no tienen que ser necesariamente iguales, pero en esta tesis utilizamos $n = m$.
A continuación, consideramos la hipótesis H (ver detalles y definiciones en Karatzas et al. (1987)):

- r_t y μ_t son procesos predecibles y acotados.
- σ_t es predecible, acotado, e invertible con σ_t^{-1} acotado.
- $\exists\ \theta_t \in \mathbb{R}^{mx1}$ predecible y acotado dado que $\mu_t - r_t 1 = \sigma_t \theta_t$.

Bajo H, el mercado es completo.
Sea Y_t la cartera de un pequeño inversor que no puede afectar al mercado

$$Y_t = \pi_t^0 + \pi_t' 1,$$

donde π_t^0 es su posición en el activo libre de riesgo y $\pi_t = (\pi_t^1, ..., \pi_t^n)' \in \mathbb{R}^{n\times 1}$ son las posiciones en los activos riesgosos. Tanto π_t^0 como π_t son predecibles. Se dice que la estrategia Y_t es auto-financiada cuando

$$dY_t = \pi_t^0 dU_t^0 + \pi_t' dU_t. \tag{2}$$

Esto es, cuando el cambio en el valor de la cartera proviene solamente de la variación del precio de los activos. De esta manera, obtenemos

$$dY_t = (Y_t - \pi_t' 1) r_t dt + \pi_t'(\mu_t dt + \sigma_t dW_t) = [Y_t r_t + \pi_t'(\mu_t - r_t 1)]dt + \pi_t' \sigma_t dW_t.$$

Ahora, definimos el derecho contingente de estilo europeo ξ establecido a tiempo T es una variable aleatoria $\mathcal{F}_T$-medible que representa un contrato que paga ξ a tiempo T. Así, una estrategia de cobertura contra ξ es una estrategia auto-financiada $(Y, \pi)|Y_T = \xi$. De manera que si asumimos que el inversor quiere replicar el *payoff* ξ, es necesario resolver

$$dY_t = [Y_t r_t + \pi_t'(\mu_t - r_t 1)]dt + \pi_t' \sigma_t dW_t,\ Y_T = \xi, \tag{3}$$

que es una BSDE para (Y, π) con condición final $Y_T = \xi$. El precio justo Y_T es el valor a tiempo t que tiene que tener Y para cubrir a ξ. Si se cumple la hipótesis H y ξ es de cuadrado integrable, siempre es posible encontrar (Y, π) que satisfaga (3) de manera que el precio justo del derecho contingente a tiempo t es Y_t.

En el caso particular de (3), que tiene un *drift*, *driver*, o generador afín en π_t, Y_t, la BSDE se puede resolver explícitamente como (Harrison and Pliska, 1981)

$$Y_t = Y(t, X(t)) = \mathbb{E}^{\mathbb{Q}}[e^{-\int_t^T r_s ds}\xi|\mathcal{F}_t], \tag{4}$$

donde la medida de probabilidad $\mathbb{Q}$ esta definida por

$$\frac{d\mathbb{Q}}{d\mathbb{P}} = exp\left[-\int_t^T \theta_s' dW_s - \frac{1}{2}\int_t^T |\theta_s|^2 ds\right].$$

Es bien conocido que (4) puede computarse directamente por el método de Monte Carlo. Además, el teorema de Feynman-Kac (Shreve, 2005) muestra que (4) es solución también de una PDE lineal, lo cual ofrece un método de cálculo eficiente y preciso en bajas dimensiones, pero impracticable en altas dimensiones por requerir una memoria computacional que crece exponencialmente con la dimensión (ver Apéndice A).

La ecuación (3) puede generalizarse en la siguiente BSDE:

$$\begin{gathered} -dY_t = f(t, X_t, Y_t, Z_t)dt - Z_t' dW_t,\ para\ t \in [0, T], \\ Y_T = \xi = g(X_T), \end{gathered} \tag{5}$$

donde el generador f es un mapeo predecible

$$f : [0, T] \times \Omega \times \mathbb{R}^d \times \mathbb{R}^{m \times d} \to \mathbb{R}^d.$$

Bajo la hipótesis H, en (3) $f(t, X_t, Y_t, Z_t) = -r_t Y_t - Z_t' \theta_t$ y $Z_t' = \pi_t' \sigma_t$.
En esta tesis buscamos modelar un contrato derivado por vez y, en general consideramos $d = 1$. La condición terminal ξ es una variable aleatoria $\mathcal{F}_T$-medible de cuadrado integrable y valores en $\mathbb{R}^d$.
Una solución de la BSDE es el par (Y, Z) que satisface (5) con

- Y_t un proceso adaptado en $\mathbb{R}^d$.
- Z_t un proceso predecible en $\mathbb{R}^{m \times d}$.

Una observación importante es que al resolver la ecuación (5) obtenemos simultáneamente Y y Z. En términos financieros, esto es obtener tanto el precio del contrato derivado como sus griegas delta para cada activo subyacente. Esto representa una enorme ventaja sobre otros métodos que solo calculan el precio del derivado, reduciendo el tiempo de cómputo en $3n$, si las griegas se calculan por *bump and reprice*. Esto significa que, si el tiempo de cálculo del riesgo de mercado de una cartera con contratos multidimensionales típicamente se hace *overnight*, este método permite reducirlo a unos pocos minutos.

3.2. Ecuaciones con generadores no lineales

En el formalismo de Black, Scholes y Merton (BSM), el generador f es una función lineal de (Y, π) y, como vimos eso conduce a que la Ecuación (3) tenga una solución explícita dada por la fórmula (4). En condiciones más generales, f es una función no lineal y se hace necesario abordar la solución numérica de la BSDE (2). En esta tesis abordamos dos escenarios que generalizan en algún aspecto las suposiciones de BSM y que conducen a ecuaciones con generadores no lineales: 1) valuación de derivados con distintas tasas de interés para tomar y prestar dinero, y 2) valuación de derivados incorporando riesgo de *default*.

3.2.1. Problema con distintas tasas de interés para tomar y prestar dinero

Consideramos un mercado donde el inversor tiene permitido tomar dinero prestado a la tasa r_t^b y prestar dinero a la tasa r_t^l, siendo $r_t^b > r_t^l$. Es decir

$$\frac{dA_t^0}{A_t^0} = \left\{ \begin{array}{c} r_t^l,\ \pi_t^0 \geq 0 \\ -r_t^b,\ \pi_t^0 < 0 \end{array} \right\}, \quad \pi_t^0 = Y_t - \pi_t'.$$

Luego, definimos la BSDE asociada a este problema como

$$dY_t = (Y_t - \pi_t' 1)^+ r_t^l dt - (Y_t - \pi_t' 1)^- r_t^b dt + \pi_t'(\mu_t dt + \sigma_t dW_t),$$
$$dY_t = [(Y_t - \pi_t' 1)^- + (Y_t - \pi_t' 1)] r_t^l dt - (Y_t - \pi_t' 1)^- r_t^b dt + \pi_t'(\mu_t dt + \sigma_t dW_t),$$
$$dY_t = r_t^l Y_t dt - (r_t^b - r_t^l)(Y_t - \pi_t' 1)^- dt + \pi_t'(\mu_t - r_t^l 1) dt + \pi_t' \sigma_t dW_t.$$

Bajo la hipótesis H, obtenemos

$$dY_t = r_t^l Y_t dt - (r_t^b - r_t^l)(Y_t - \pi_t' 1)^- dt + \pi_t' \sigma_t (\theta_t dt + dW_t). \tag{6}$$

Finalmente, la Ecuación (6) puede escribirse en la forma de (5), donde $Z_t' = \pi_t' \sigma_t$ y el generador f es equivalente a

$$f(t, X_t, Y_t, Z_t) = -r_t^l Y_t + (r_t^b - r_t^l)(Y_t - Z_t' \sigma^{-1})^- - Z_t' \theta_t.$$

Debido a Pardoux and Peng (1992), que generalizan el teorema de Feynman-Kac, la solución de la BSDE es también la solución de la PDE asociada con sus correspondientes condiciones de contorno y condición terminal (Bergman, 1995)

$$\frac{\partial Y}{\partial t} + \frac{\sigma^2}{2} X^2 \frac{\partial^2 Y}{\partial X^2} + \left(X \frac{\partial Y}{\partial X} - Y \right) h(X, Y) = 0,$$
$$h(X, Y) = \begin{Bmatrix} r^b, \ (X \frac{\partial Y}{\partial X} - Y) > 0 \\ r^l, \ (X \frac{\partial Y}{\partial X} - Y) \leq 0 \end{Bmatrix}, \tag{7}$$

donde el termino fundamental de (7) es $X \frac{\partial Y}{\partial X} - Y$. Cuando $X \frac{\partial Y}{\partial X} > Y$, (7) se convierte en la PDE de Black, Scholes y Merton con $r = r^b$, caso contrario (7) se transforma en la PDE de Black, Scholes y Merton con $r = r^l$.
La PDE (7) puede ser resuelta con el método de diferencias finitas para dimensiones muy bajas, menores que 2 para una computadora personal actual. Debido a que el tamaño de las matrices involucradas en FDM u otros métodos de discretización de PDEs crece exponencialmente con la dimensión del subyacente, es necesario recurrir a otros esquemas de cómputo numérico. Los métodos basados en BSDEs proveen justamente de una posibilidad de cómputo que hasta ahora estaba vedada para problemas de alta dimensionalidad. En la Sección 5.2 mostramos resultados para este problema con los métodos presentados en las Secciones 3.3 y 3.4.

3.2.2. Problema incorporando riesgo de *default*

De manera análoga al desarrollo del problema anterior, definimos la BSDE de este problema como

$$dY_t = ((1-\delta_t)Q(Y_t) + r_t)Y_t dt + \pi_t' \sigma_t(\theta_t dt + dW_t), \tag{8}$$

$$Q(Y_t) = \mathbb{1}_{(-\infty, \upsilon_t^h)}(Y_t)\gamma_t^h + \mathbb{1}_{(\upsilon_t^l, \infty)}(Y_t)\gamma_t^l$$

$$+\mathbb{1}_{(\upsilon_t^h, \upsilon_t^l)}(Y_t)\left[\frac{(\gamma_t^h - \gamma_t^l)}{(\upsilon_t^h - \upsilon_t^l)}(Y_t - \upsilon_t^h) + \gamma_t^h\right],$$

donde δ_t es la proporción recuperable del activo en caso de $default$, υ_t^h y υ_t^l son umbrales que separan tres tipos de estado de riesgo: alto, intermedio y bajo; y finalmente, γ_t^h y γ_t^l son las probabilidades asociadas a dichos estados de riesgo (notar que para el estado de riesgo intermedio se utiliza una interpolación).
La Ecuación (8) puede escribirse en la forma de (5), donde $Z_t' = \pi_t' \sigma_t$ y el generador f es equivalente a

$$f(t, X_t, Y_t, Z_t) = -((1-\delta_t)Q(Y_t) + r_t)Y_t - Z_t'\theta_t,$$

Debido a Pardoux and Peng (1992), que generalizan el teorema de Feynman-Kac, la solución de la BSDE es también la solución de la PDE asociada con sus correspondientes condiciones de contorno y condición terminal (Han et al., 2018)

$$\frac{\partial Y}{\partial t} + rX\frac{\partial Y}{\partial X} + \frac{\sigma^2}{2}X^2\frac{\partial^2 Y}{\partial X^2} - (1-\delta)Q(Y)Y - rY = 0. \tag{9}$$

Así como en el problema de diferentes tasas de interés, para el problema que incluye riesgo de $default$, la PDE (9) puede ser resuelta mediante FDM para 1 y 2 dimensiones. Pero, como bien mencionamos anteriormente, este método se torna inviable para derivados de grandes dimensiones y por ello es necesario recurrir a esquemas basados en BSDEs. En la Sección 5.2 mostramos resultados para este problema con los métodos presentados en las Secciones 3.3 y 3.4.

3.3. Método *Forward Deep BSDE*

E et al. (2017) y Han et al. (2018) proponen un algoritmo basado en redes neuronales artificiales (ANNs) para obtener la solución de (5) en $t = 0$. Resumimos su método a continuación:

a. Se simulan M caminos de un subyacente d-dimensional a partir de la discretización de la ecuación (1) con el esquema de Euler-Maruyama. El supra índice j indica el camino y X_t^j es un vector d-dimensional

$$X_{t+1}^j = X_t^j + \mu\Delta t + \sigma\Delta W_t^j,\ t = 0, 1, 2, ..., T.$$

Si bien existen distintas discretizaciones de la ecuación (1), en este trabajo utilizaremos en todos los casos la solución exacta al movimiento Browniano geométrico

$$X_{t+1}^j = X_t^j e^{\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\Delta W_t^j}.$$

b. A tiempo $t = 0$, se eligen de forma aleatoria las condiciones iniciales Y_0 y Z_0, únicas para los M caminos.

c. Para cada tiempo t, se aproxima la variable Z_t mediante una red neuronal de H capas (a esto se le llama una sub-red). Cada capa tiene asociado un parámetro θ_t^h, $h = 1, 2, ..., H$, que contiene los ponderadores y sesgos/constantes asociados a las neuronas de la capa. Especificamos la arquitectura de una sub-red en la Sección 4.1.

$$X_t^j \to h_t^1(\theta_t^1) \to h_t^2(\theta_t^2) \to ... \to h_t^H(\theta_t^H) \to \tilde{Z_t^j}.$$

d. Se discretiza la ecuación (5) usando la aproximación $\tilde{Z}_t$ de Z_t y el esquema de Euler-Maruyama, y se propaga hacia adelante en el tiempo.

$$Y_{t+1}^j = Y_t^j - f(t, X_t^j, Y_t^j, \tilde{Z_t^j})\Delta t + \tilde{Z_t^j}'\Delta W_t^j.$$

e. Se repiten los pasos c. y d. para los M caminos simulados del subyacente hasta llegar al nodo terminal. En $t = T$ y $j = M$, se obtiene un vector $\vec{Y_T} = [Y_T^1, ..., Y_T^M]$. El conjunto de sub-redes descritas en el punto c. conforman la DNN que tiene un vector de parámetros asociado $\vec{p} = [\vec{\theta_1}, ..., \vec{\theta_{T-1}}, Y_0, Z_0]$. En total, existen $T - 2$ sub-redes y $(T - 2) \times H$ parámetros θ_t^h.

f. Se computa la función de error como el valor esperado de las diferencias cuadráticas entre $\vec{Y_T}$ y la condición terminal $\vec{g}(X_T) = [g(X_T^1), ..., g(X_T^M)]$.

$$L_{forward} = E\left[\left(\vec{Y_T} - \vec{g}(X_T)\right)^2\right]. \tag{10}$$

g. Finalmente, se minimiza (12) variando los parámetros de la DNN utilizando el optimizador Adam (Estimación de Momento Adaptativo) propuesto por Kingma and Ba (2014), el cual detallamos en la Sección 4.2. Como

resultado de la minimización, se obtienen los parámetros óptimos. En particular interesa el Y_0 óptimo denotado como Y_0^*.

$$Y_0^* = \arg\min_{Y_0}(L_{forward}). \tag{11}$$

A modo ilustrativo, la Figura 1 presenta una descripción gráfica del método FDB.

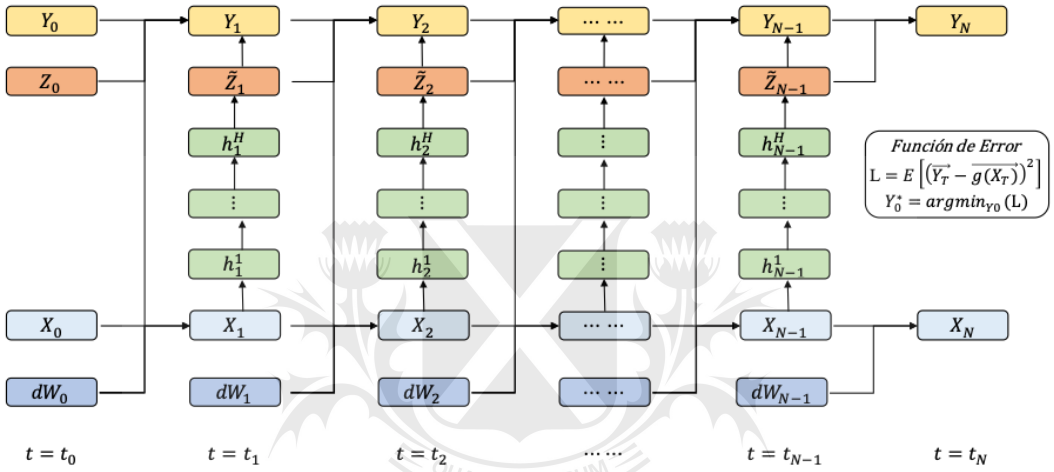

Figura 1: Diagrama del método *Forward Deep BSDE* (FDB). Los nodos en azul y celeste representan el movimiento Browniano y el proceso hacia adelante del subyacente X. Los nodos en verde y naranja representan la sub-redes que generan la aproximación $\tilde{Z}_t$ de Z_t para cada t. Por último, los nodos en amarillo simbolizan el proceso hacia adelante de Y.

3.4. Método *Backward Deep BSDE*

Según el Principio de Optimalidad de Bellman, para valuar un instrumento con ejercicio anticipado el valor de continuación de dicho instrumento debe ser conocido para cualquier tiempo futuro. Ocurre que los métodos numéricos con propagación hacia adelante son inviables para la valuación de instrumentos de tipo americano o bermuda, ya que no pueden estimar valores de continuación sin introducir costosos cómputos de MC anidados que son impracticables. Dado que en el método FDB se propaga hacia adelante comenzando desde una condición inicial, sirve únicamente para valuar derivados de estilo europeo. Por lo tanto, es necesario utilizar métodos que partan de una condición final y propaguen hacia atrás en el tiempo donde el valor de continuación se obtiene mas eficientemente regresando iterativamente en el tiempo sobre valores conocidos del futuro. Cabe aclarar que nos referimos

únicamente a la BSDE, ya que el activo subyacente siempre se propaga hacia adelante.
En esta línea, Wang et al. (2018) desarrollaron uno de los primeros algoritmos para valuar derivados con ejercicio anticipado realizando algunas modificaciones al método *Forward* original. Por otro lado, debido a la falta de resultados consistentes, Liang et al. (2019) modificaron ligeramente el método de Wang. La principal diferencia es que discretizan el proceso Y_t utilizando una expansión de Taylor de 1er. orden cuando se propaga hacia atrás en el tiempo. A continuación, describimos el método propuesto por Liang et al. (2019, 2021):

a. Se simulan M caminos de un subyacente d-dimensional a partir de la discretización de la ecuación (1) con el esquema de Euler-Maruyama. El supra índice j indica el camino y X_t^j es un vector d-dimensional

$$X_{t+1}^j = X_t^j + \mu\Delta t + \sigma\Delta W_t^j,\ t = 0, 1, 2, ..., T.$$

b. Para cada tiempo t, se aproxima la variable Z_t mediante una red neuronal de H capas (a esto se le llama una sub-red). Cada capa tiene asociado un parámetro θ_t^h, $h = 1, 2, ..., H$, que contiene los ponderadores y sesgos/-constantes asociados a las neuronas de la capa.

$$X_t^j \to h_t^1(\theta_t^1) \to h_t^2(\theta_t^2) \to ... \to h_t^H(\theta_t^H) \to \tilde{Z_t^j}.$$

c. Se discretiza la ecuación (5) usando la aproximación $\tilde{Z}_t$ de Z_t y el esquema de Euler-Maruyama, y se propaga hacia atrás en el tiempo comenzando desde la condición terminal $g(X_t^j)$ y utilizando una expansión de Taylor de 1er orden.

$$Y_t^j = Y_{t+1}^j + \frac{f(t, X_t^j, Y_{t+1}^j, Z_t^j)\Delta t - Z_t^{j\prime}\Delta W_t^j}{1 - \frac{\partial f}{\partial y}(t, X_t^j, Y_{t+1}^j, Z_t^j)}. \tag{12}$$

d. Se repiten los pasos b. y c. para los M caminos simulados del subyacente hasta llegar al nodo terminal. En $t = 0$ y $j = M$, se obtiene un vector $\vec{Y_0} = [Y_0^1, ..., Y_0^M]$. El conjunto de sub-redes descritas en el punto c. conforman la DNN que tiene un vector de parámetros asociado $\vec{p} = [\vec{\theta_1}, ..., \vec{\theta_T}]$. En total, existen $T - 1$ sub-redes y $(T - 1) \times H$ parámetros θ_t^h.

e. Se computa la función de error como la varianza del vector $\vec{Y_0}$.

$$L_{backward} = E\left[\left(\vec{Y_0} - E[\vec{Y_0}]\right)^2\right].$$

f. Finalmente, se minimiza (11) variando los parámetros de la DNN por medio del optimizador Adam. Como resultado de la minimización, se obtienen los parámetros óptimos. La diferencia con el método FDB es que Y_0 y Z_0 no forman parte de $\vec{p}$, sino que Y_0^* se estima como el valor esperado del vector $\vec{Y_0}$ evaluado en $\vec{p}$ óptimo.

$$\vec{p^*} = \arg\min_{\vec{p}}(L_{backward}),$$
$$Y_0^* = E[\vec{Y_0}(\vec{p^*})].$$

A modo ilustrativo, la Figura 2 presenta una descripción gráfica del método BDB.

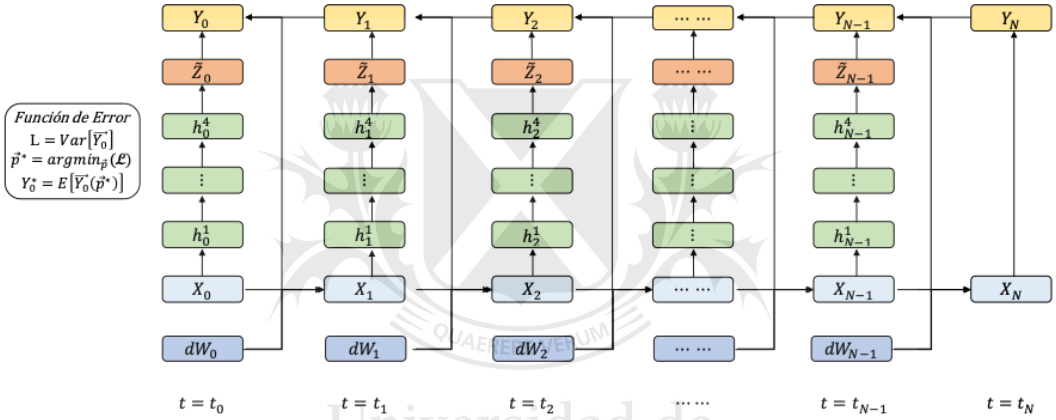

Figura 2: Diagrama del método *Backward Deep BSDE* (BDB). Los nodos en azul y celeste representan el movimiento Browniano y el proceso hacia adelante del subyacente X. Los nodos en verde y naranja representan la sub-redes que generan la aproximación $\tilde{Z}_t$ de Z_t para cada t. Por último, los nodos en amarillo simbolizan el proceso hacia atras de Y.

3.4.1. Valuación con ejercicio anticipado

Para la valuación de instrumentos con ejercicio anticipado es necesario hacer una leve modificación en el punto c. del algoritmo *Backward*. Liang et al. (2019, 2021) plantean la necesidad de un factor exógeno al valor de continuación de la DNN para determinar el ejercicio anticipado, es decir, estiman un valor de continuación A_t^j por un método distinto a la DNN. De esta manera, para cada tiempo t y camino j el algoritmo determina el ejercicio anticipado según la siguiente ecuación:

$$Y_t^j = \begin{Bmatrix} Y_t^j, si\ A_t^j \geq g(X_t^j) \\ g(X_t^j), si\ A_t^j < g(X_t^j) \end{Bmatrix},$$

donde $g(X_t^j)$ es el valor intrínseco de Y_t^j. En particular, los autores estiman A_t^j mediante el método de Longstaff and Schwartz (2001).

4. Redes Neuronales

4.1. Estructura de una red neuronal

En la literatura existen varias arquitecturas de redes neuronales artificiales (ANNs) dadas. Particularmente, en el contexto de *Deep BSDE*, Chan-Wai-Nam et al. (2019) presentan distintas alternativas, siendo una de las configuraciones más sencillas la de una red neuronal profunda pre-alimentada o *feed-forward* en inglés. Definimos una red neuronal profunda pre-alimentada $\eta(\theta, x)$ como un conjunto de transformaciones afines φ_h[1]

$$\eta(\theta, x) = \alpha_H \circ \varphi_H \circ \alpha_{H-1} \circ \varphi_{H-1} \circ ... \circ \alpha_1 \circ \varphi_1 \circ \alpha_0(x),$$

donde α_h es un conjunto de funciones de activación que suelen ser no lineales. Por otro lado, $\varphi_h = \omega_h' x + b_h$, ω_h los ponderadores de cada neurona de la capa y b_h los sesgos o constantes. Todos los parámetros ω_h y b_h de la red se agrupan en el parámetro θ y son aquellos a optimizar por la red. A modo ilustrativo, la Figura 3 presenta un diagrama del funcionamiento de una capa simple dentro de la red.

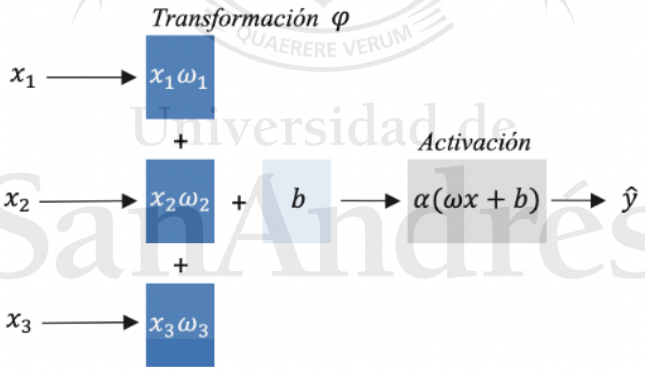

Figura 3: diagrama de una capa dentro de una ANN.

El funcionamiento es simple. Las neuronas reciben una información x a la cual se le aplica una transformación φ_h y luego una función de activación α_h. Finalmente, la función de activación retorna un resultado $\hat{y}$. Una red neuronal profunda (DNN) se caracteriza por tener varias capas intermedias u ocultas, como muestra la Figura 4.

[1]Aclaración: la notación de esta sección no tiene relación con la notación del resto de las secciones de este trabajo.

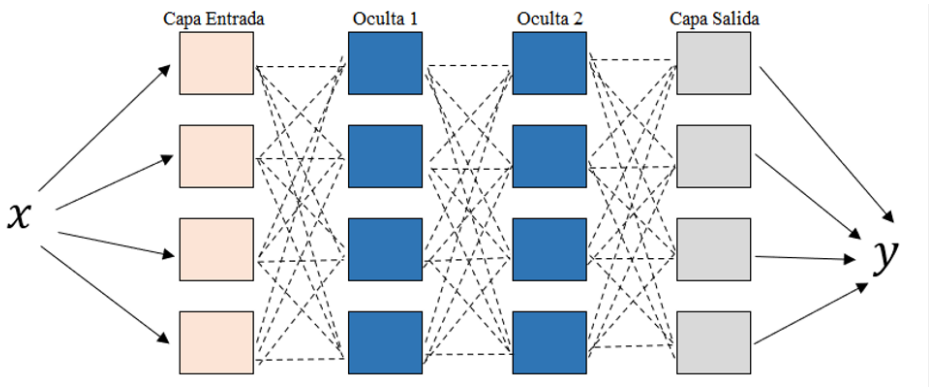

Figura 4: diagrama de una red neuronal profunda con dos capas ocultas.

Al resultado y de la capa de salida, se le aplica una función de error L y luego se minimiza L para obtener el conjunto de parámetros óptimos θ^*.

4.2. Optimizadores

Para minimizar la función de error, existen distintos algoritmos de optimización que requieren el cómputo de derivadas parciales. El proceso para obtener el gradiente de la función de error se conoce como propagación hacia atrás o *Backpropagation* en inglés. Profundizaremos en tres algoritmos: Descenso de Gradiente o *Gradient Descent* en inglés (GD), Descenso de Gradiente en mini-lotes o *Mini-batch Gradient Descent* en inglés (MBGD) y Estimación de Momento Adaptativo o *Adaptive Moment Estimation* en inglés (Adam). Tanto el GD, como el MBGD, consisten en actualizar los ω_h y b_h en dirección opuesta a su gradiente

$$\theta_{n+1} = \theta_n - \delta \nabla_{\theta_n},$$

donde δ es la tasa de aprendizaje o *learning rate* en inglés (LR). El proceso de actualización de θ se repite por una cantidad de iteraciones hasta lograr la convergencia de la función de error al mínimo global. Previo a explicar la diferencia entre los optimizadores GD y MBGD, es necesario introducir el concepto de lote o *batch* en inglés. Un lote refiere a una fracción del conjunto total de datos, mientras que un mini-lote o *mini-batch* en inglés, es simplemente un lote de tamaño reducido. Entonces, la principal diferencia entre ambos optimizadores es que GD utiliza la totalidad de los datos para actualizar los ponderadores y sesgos de cada neurona, lo cual requiere un uso mayor de la memoria RAM al igual que disminuye la velocidad de entrenamiento. En cambio, MBGD permite la actualización de parámetros y sesgos utilizando lotes (o mini-lotes), lo cual mejora la velocidad de entrenamiento y libera espacio en la memoria RAM. Sin embargo, el principal problema de

MBGD es que presenta en muchos casos un comportamiento ruidoso debido a que, al usar menos datos, la superficie de la función objetivo es menos suave. Además, si bien el ruido introducido por MBGD le permite escapar, en algunos casos el algoritmo puede estancarse en un mínimo local.
Para resolver este problema, se desarrollaron algoritmos basados en estimadores de momento, es decir que combinan el valor del gradiente en la iteración actual con valores en iteraciones pasadas. En esta línea, Kingma and Ba (2014) propusieron el optimizador de momento Adam, que se ha convertido rápidamente en el más utilizado debido a su gran eficiencia. Este optimizador requiere un parámetro de aprendizaje δ (LR), dos parámetros β_1 y β_2 que determinan la tasa de decaimiento exponencial de los estimadores de momento y por último, un parámetro ϵ que evita la división por cero. Acá, m_n y ν_n son los estimadores de momento que se inicializan en cero. El optimizador de Adam actualiza el conjunto θ de la siguiente manera:

$$m_n = \beta_1 m_{n-1} + (1-\beta_1)\nabla_{\theta_n},$$
$$\nu_n = \beta_2 \nu_{n-1} + (1-\beta_2)\nabla^2_{\theta_n},$$
$$\widehat{m_n} = \frac{m_n}{1-\beta_1^n},$$
$$\widehat{\nu_n} = \frac{\nu_n}{1-\beta_2^n},$$
$$\theta_{n+1} = \theta_n - \delta \frac{\widehat{m_n}}{\sqrt{\widehat{\nu_n}} + \epsilon}$$

4.3. Normalización de lotes

El algoritmo de propagación hacia atrás, que caracteriza a las redes neuronales, muchas veces puede sufrir de lo que en la literatura se denomina Problema del Desvanecimiento del Gradiente. Ocurre cuando el gradiente se va reduciendo a medida que el algoritmo de propagación hacia atrás computa las derivadas parciales en cada capa de la red. Cuando el algoritmo llega a las capas inferiores, el gradiente es cercano a cero y por esa razón estas capas resultan ser más costosas de entrenar. Existen distintas soluciones a este problema, entre ellas la técnica de Normalización de Lotes o *Batch Normalization* en inglés, propuesta por Ioffe and Szegedy (2015), que consiste en adicionar una operación entre φ_h y α_h para normalizar las entradas y centrarlas en cero. Definimos el algoritmo a continuación:

$$\mu_B = \frac{1}{B}\sum_{b=1}^{B} x_b, \tag{13}$$

$$\sigma_B^2 = \frac{1}{B}\sum_{b=1}^{B}(x_b - \mu_b)^2, \tag{14}$$

$$\widehat{x_b} = \frac{x_b - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{15}$$

$$y_b = \gamma\widehat{x_b} + \beta, \tag{16}$$

donde B refiere a un lote y γ y β son los parámetros a optimizar en cada capa. Las ecuaciones (13-16) son válidas para el entrenamiento de la red, mientras que para la etapa de evaluación o inferencia, este sistema de ecuaciones es reemplazado por

$$y = \frac{\gamma}{\sqrt{\sigma_I^2 + \epsilon}}x + \left(\beta - \frac{\gamma\mu_I}{\sqrt{\sigma_I^2 + \epsilon}}\right),$$

donde σ_I^2 y μ_I se computan sobre todo el conjunto de datos de prueba.

4.4. Tasa de aprendizaje con decaimiento exponencial

Existen ciertas formas de elegir una tasa de aprendizaje óptima para distintas arquitecturas de red. Una manera, es establecer un esquema de tasas de aprendizaje que varía con el tiempo. En particular, Senior et al. (2013) demuestran que un esquema de decaimiento exponencial aumenta considerablemente la eficiencia de la red, convergiendo rápidamente.
En dicho esquema, δ se va actualizando cada cierta cantidad de iteraciones de entrenamiento de la red como indica la siguiente fórmula:

$$\delta_i = \delta_0 \ \pi^{\frac{i}{N}}, \tag{17}$$

donde i es la iteración de entrenamiento actual, π es la tasa de decaimiento que toma valores entre 0 y 1, indicando la velocidad de decaimiento de la tasa de aprendizaje, y N son los pasos de decaimiento, que indican la frecuencia con la que se actualiza la tasa de aprendizaje. Dado un π determinado, y seleccionando una tasa de aprendizaje inicial y otra final, podemos despejar N reformulando la ecuación (17) de la siguiente manera:

$$N = \frac{\ln(\pi)I}{\ln\left(\frac{\delta_I}{\delta_0}\right)}, \tag{18}$$

donde δ_0 es la tasa de aprendizaje con la que inicializamos el algoritmo, I es la cantidad total de iteraciones de entrenamiento y δ_I es la tasa de aprendizaje con la que finalizamos el algoritmo. Es decir, la ecuación (18) resuelve la frecuencia con la que actualizamos la tasa de aprendizaje para terminar en la iteración I con una tasa de aprendizaje δ_I. En las secciones siguientes nos referiremos al proceso de decaimiento exponencial como $\phi(\delta_0; \delta_I)$.

4.5. Estructura de las sub-redes

En la Sección 3 presentamos la metodología del trabajo mencionando la utilización de sub-redes para aproximar la variable Z. E et al. (2017) y Han et al. (2018) proponen sub-redes de 4 capas con función de activación ReLu (*Rectified Linear Unit*), incluyendo la Normalización de Lotes en cada capa. Resumimos la estructura de una sub-red en la Figura 5.

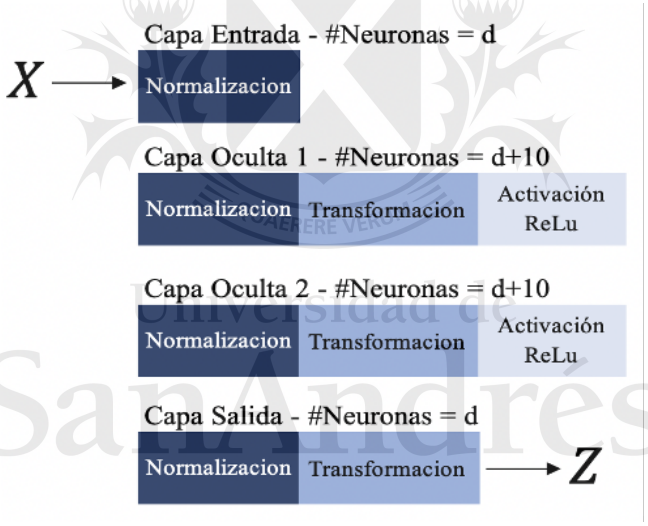

Figura 5: diagrama de una sub-red.

donde formalmente la función de activación ReLU se define como:

$$ReLU(x_i) = \begin{Bmatrix} x_i, & x_i \geq 0 \\ 0, & x_i < 0 \end{Bmatrix}.$$

En primer lugar, la capa de entrada solamente tiene un proceso de normalización, la cantidad de neuronas es igual a las dimensiones de X. Las siguientes 2 capas incorporan los 3 procesos: normalización, transformación φ y activación. Cada capa tiene una cantidad de neuronas equivalente a las dimensiones

de X más 10. Por último, la capa de salida incorpora el proceso de normalización y transformación φ, pero no la activación. La última capa tiene la misma cantidad de neuronas que la capa de entrada. Esta arquitectura se replica en varios trabajos de la literatura con algunas modificaciones (Liang et al., 2019, 2021; Yu et al., 2020; Huré et al., 2020; Hientzsch, 2019; Henry-Labordere, 2017; Gnoatto et al., 2020; Becker et al., 2019; Wang et al., 2018; Kapllani and Teng, 2020), mientras que en otros se plantean arquitecturas distintas (Yu et al., 2019; Ganesan et al., 2020; Güler et al., 2019; Chen and Wan, 2021; Zhang and Cai, 2020; Raissi, 2018).

5. Resultados numéricos

La sección de resultados está dividida en dos partes. En la Sección 5.1 presentamos resultados de la valuación de opciones de estilo europeo y americano/bermuda en el formalismo de Black, Scholes y Merton (BSM) con los métodos propuestos en este trabajo. En la Sección 5.2 presentamos resultados para la valuación de opciones en condiciones más generales que las del formalismo de BSM que involucran generadores no lineales como los descriptos en la Sección 3.2. Desarrollamos los algoritmos en lenguaje Python utilizando la librería Tensorflow 1, mientras que implementamos todas las pruebas en la plataforma Google Colab utilizando GPU y una memoria RAM de 12mb. En la misma línea, implementamos los algoritmos de Monte Carlo (MC) y *Least Squares* Monte Carlo (LSMC) en Tensorflow 1 para que fuesen comparables con los métodos *Deep BSDE*, esto se debe a que Tensorflow permite el cómputo paralelo de operaciones con arreglos numéricos mediante el uso de GPUs o TPUs. Cabe aclarar que desarrollamos la simulación de caminos del activo subyacente en la librería *Numpy* para todos los métodos.
Otra consideración a tener en cuenta es que utilizamos parámetros de los contratos (μ_t, σ_t, r_t, r_t^l y r_t^b, δ_t, υ_t^h, υ_t^l, γ_t^h y γ_t^l) constantes para todo $t \in [0, T]$. Los denotamos como μ, σ, r, r^l y r^b, δ, υ^h, υ^l, γ^h y γ^l.
Para obtener la solución de la BSDE procedemos de la siguiente manera:

1. Seleccionamos los parámetros del contrato/opción (el vencimiento T, el precio de ejercicio K, la tasa libre de riesgo r, σ y μ del activo subyacente, etc.) y del entrenamiento de la DNN (tamaño del lote de entrenamiento (TBS), cantidad de iteraciones (ITs), cantidad de pasos temporales y tasa de aprendizaje (LR)).

2. Corremos el algoritmo para un lote de entrenamiento y computamos Y_0^*.

3. Repetimos 2 por n iteraciones.

4. Al finalizar las iteraciones obtenemos un vector $\vec{Y_0^*} = [Y_0^{*1}, ..., Y_0^{*n}]$. La solución de la BSDE, denotada como Y_0^{sol}, es calculada de manera distinta para cada método, como explicamos a continuación.

En la Figura 6, podemos observar la convergencia de los métodos *Forward Deep BSDE* (FDB) y *Backward Deep BSDE* (BDB) a una solución dada. Como puede verse, el método FDB converge a medida que aumentan las iteraciones (esto se debe en gran medida a la utilización de una LR con decaimiento exponencial como se muestra en la Sección 5.1.2) y el Y_0^{*n} es el valor más representativo de la solución del problema. En cambio, en el método

BDB observamos un comportamiento oscilatorio en relación a la solución del problema. Es decir, en el método *Backward* Y_0^* oscila simétricamente sobre la solución del problema y por lo tanto, el valor más representativo de la solución del problema es el promedio de $\vec{Y}_0^*$ (esto también sucede para el método *Least Squares Backward DNN* (LSBD)).

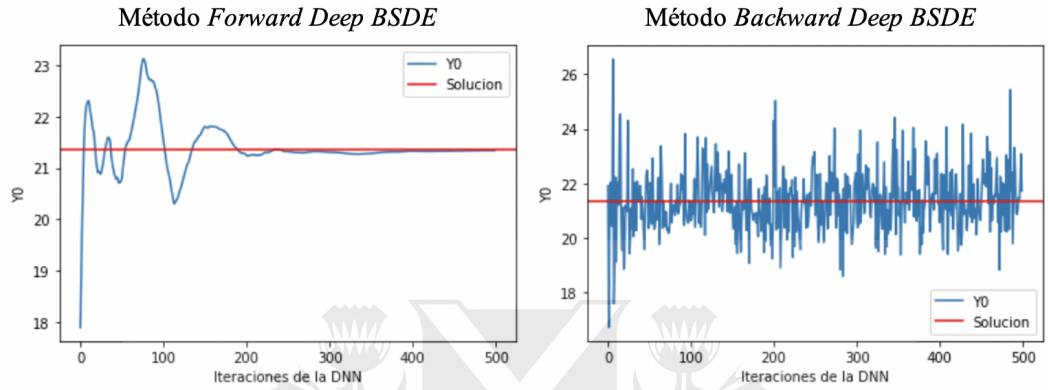

Figura 6: Convergencia de los métodos FDB (panel izquierdo) y BDB (panel derecho) a una solución dada.

Por lo tanto, en el método FDB, $Y_0^{sol} = Y_0^{*n}$ y en los métodos BDB y LSBD, $Y_0^{sol} = \frac{1}{n}\sum_{i=1}^{n} Y_0^{*i}$. Para determinar el intervalo de confianza de Y_0^{sol}, ejecutamos 30 veces el algoritmo y computamos la media ($\mu_{Y_0^{sol}}$), el desvío estándar ($\sigma_{Y_0^{sol}}$) y el intervalo de confianza empírico al 95 % ($IC95\,\%_{Y_0^{sol}}$).
A continuación enumeramos una serie de aclaraciones sobre el contenido de las tablas de resultados y figuras que se presentan a lo largo de esta sección:

- Error Relativo: refiere al siguiente cálculo:

$$ER = \frac{\mu_{Y_0^{sol}}}{sol} - 1,$$

  donde *sol* es la solución de referencia para un problema determinado. Para la BSDE en el formalismo de BSM, *sol* es el valor de expectación de MC con 1 millón de caminos o la fórmula de Black, Scholes y Merton (para un activo subyacente de 1 dimensión). En cambio, para BSDEs con generadores no lineales *sol* equivale al valor obtenido por otros autores o por el método de diferencias finitas (FDM) (cuando el activo subyacente tiene 1 dimensión).

- I.C. 95 % Error Relativo: refiere al intervalo de confianza 95 % del error relativo para un método y problema determinado. La fórmula de cálculo es la misma que en el error relativo, remplazando $\mu_{Y_0^{sol}}$ por $IC95\,\%_{Y_0^{sol}}$.

- Tiempo Converg.: refiere al tiempo promedio de convergencia de las 30 veces que se ejecutó el algoritmo; Es decir, el tiempo promedio que requiere el algoritmo para converger a la solución de un problema dado.
- ECM: refiere al error cuadrático medio de Y_0^{sol}.
- Desvío/Media: cociente entre $\mu_{Y_0^{sol}}$ y $\sigma_{Y_0^{sol}}$.
- Cant. de Caminos: refiere a la cantidad de caminos utilizados en MC.
- Lote: en los casos donde la memoria RAM colapsa debido a la dimensionalidad del problema en MC o LSMC, se generan los caminos en lotes y luego se calcula el valor de expectación.
- Media: en MC y LSMC refiere al valor de expectación.
- Dims.: refiere a las dimensiones del activo subyacente.

5.1. Problemas con generadores lineales

En esta sección presentamos resultados para la valuación de opciones dentro del formalismo de BSM para los métodos propuestos. A lo largo de esta sección se tratan 4 problemas a los que nos referimos como A, B, C, y D. A continuación describimos cada problema:

- Problema A: opción de compra sobre un activo subyacente uni-dimensional, a dinero (ATM) y de estilo europeo. $T = 1$ año, $K = \$100$, $r = 1\,\%$ y $\sigma = 20\,\%$. La solución según la fórmula de BSM es \$8,43.
- Problema B: opción de compra sobre 100 activos subyacentes, a dinero (ATM) y de estilo europeo. $T = 1$ año, $K = \$100$, $r = 1\,\%$, σ está dado por un vector aleatorio con distribución uniforme $U_{[0,1;1]}$ (se fija la semilla aleatoria, $numpy.random.seed = 1$, para que los resultados sean reproducibles) y la correlación entre todos los activos es 0,3. La solución según MC con 1 millón de caminos es \$21,4.
- Problema C: opción de venta sobre un activo subyacente uni-dimensional, a dinero (ATM) y de estilo bermuda. $T = 0{,}5$ años, $K = \$45$, $r = 5\,\%$ y $\sigma = 30\,\%$. La opción puede ejercerse cada 9 días ($dt = 1/20$). La solución según LSMC con 1 millón de caminos es \$3,32.
- Problema D: opción de venta sobre una canasta de 10 activos subyacentes, a dinero (ATM) y de estilo bermuda. $T = 0{,}5$ años, $K = \$45$,

$r = 5\,\%$, σ está dado por un vector aleatorio con distribución uniforme $U_{[0,1;1]}$ (se fija la semilla aleatoria, $numpy.random.seed = 2$, para que los resultados sean reproducibles) y la correlación entre todos los activos es 0,4. La opción puede ejercerse cada 18 días ($dt = 1/10$). La solución según LSMC con 1 millón de caminos es \$3,17.

5.1.1. Pruebas de convergencia

En esta sección demostramos la convergencia de los métodos propuestos en este trabajo en la valuación de opciones de tipo europeo y americano/bermuda en el marco de Black, Scholes y Merton (BSM).
Las tablas 1 y 2 contienen los resultados de la valuación de los problemas A y B. En ambos casos y tanto para el método FDB como para el BDB, $TBS = 256$, $dt = T/2$ (es decir utilizamos una única sub-red para estimar Z) y $LR = \phi(1; 5 \times 10^{/4})$.
Las tablas 3 y 4 corresponden a los resultados de la valuación de los problemas C y D con el método LSBD. En ambos casos, $TBS = 256$ y $LR = \phi(1; 5 \times 10^{/4})$. Para la Tabla 3 utilizamos 20 pasos temporales ($dt = T/20$) y en la tabla 4 utilizamos 10 pasos temporales ($dt = T/10$), donde en cada t está permitido el ejercicio.
Podemos observar que, en todos los casos, los métodos propuestos en este trabajo convergen a las soluciones de referencia. En este sentido, notamos que los métodos son no-sesgados y que los intervalos de confianza son considerablemente acotados.

	Forward Deep BSDE			*Backward Deep BSDE*		
ITs	**Error Relativo**	**I.C. 95 % Error Relativo**	**Tiempo Converg.**	**Error Relativo**	**I.C. 95 % Error Relativo**	**Tiempo Converg.**
10	89,60 %	[89,58 %; 89,62 %]	1,73	0,27 %	[−4,63 %; 5,92 %]	3,17
30	58,14 %	[57,79 %; 58,61 %]	1,85	0,15 %	[−2,32 %; 2,52 %]	3,42
50	29,10 %	[28,31 %; 30,30 %]	2,02	0,49 %	[−0,81 %; 2,15 %]	3,67
100	−4,97 %	[−6,55 %; −3,63 %]	2,41	0,02 %	[−0,70 %; 0,70 %]	4,32
250	0,08 %	[−1,45 %; 1,15 %]	3,45	−0,04 %	[−0,32 %; 0,31 %]	6,24
500	−0,08 %	[−0,97 %; 0,80 %]	5,28	0,00 %	[−0,31 %; 0,33 %]	9,31
5.000	0,05 %	[−0,39 %; 0,39 %]	37,16	0,03 %	[−0,08 %; 0,12 %]	64,19

Tabla 1: Resultados de la valuación del problema A. Métodos FDB y BDB.

ITs	*Forward Deep BSDE* Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.	*Backward Deep BSDE* Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.
10	−16,88 %	[−16,96 %; −16,81 %]	1,79	−0,20 %	[−2,94 %; 3,74 %]	2,84
30	−5,91 %	[−6,38 %; −5,16 %]	2,46	0,27 %	[−1,65 %; 1,63 %]	3,64
50	0,79 %	[−1,30 %; 2,72 %]	3,30	−0,22 %	[−1,32 %; 1,36 %]	4,52
100	0,79 %	[−0,60 %; 2,75 %]	5,51	−0,06 %	[−0,87 %; 0,55 %]	7,24
250	0,03 %	[−0,90 %; 1,16 %]	7,69	−0,07 %	[−0,35 %; 0,22 %]	9,44
500	0,05 %	[−0,63 %; 0,48 %]	16,66	0,00 %	[−0,18 %; 0,14 %]	18,30
5.000	−0,01 %	[−0,22 %; 0,25 %]	129,78	−0,01 %	[−0,06 %; 0,03 %]	109,88

Tabla 2: Resultados de la valuación del problema B. Métodos FDB y BDB.

ITs	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.
10	0,85 %	[−0,52 %; 2,00 %]	27,85
30	0,34 %	[−0,02 %; 0,64 %]	28,93
50	0,15 %	[−0,06 %; 0,34 %]	30,45
100	0,06 %	[−0,02 %; 0,16 %]	34,04
250	0,02 %	[−0,03 %; 0,08 %]	45,87
500	0,02 %	[−0,02 %; 0,04 %]	68,36
5.000	−0,01 %	[−0,02 %; 0,01 %]	427,97

Tabla 3: Resultados de la valuación del problema C. Método LSBD.

ITs	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.
10	3,05 %	[2,19 %; 4,30 %]	14,72
30	1,82 %	[1,34 %; 2,29 %]	16,23
50	0,84 %	[0,54 %; 1,07 %]	18,31
100	0,22 %	[0,10 %; 0,32 %]	23,58
250	0,11 %	[0,03 %; 0,19 %]	32,59
500	0,05 %	[0,02 %; 0,11 %]	49,92
5.000	−0,01 %	[−0,02 %; 0,00 %]	338,92

Tabla 4: Resultados de la valuación del problema D. Método LSBD.

5.1.2. Ajuste de parámetros

En esta sección exponemos la sensibilidad de los métodos *Deep BSDE* a la variación de algunos parámetros de la DNN. En particular, estudiamos cómo afecta a la convergencia el cambio en: el número/cantidad de iteraciones (ITs), el tamaño del lote de entrenamiento (TBS), la tasa de aprendizaje (LR), si la discretización temporal (dt) corresponde exactamente, la cantidad de capas ocultas y la cantidad de neuronas por capa oculta. En las Figuras 7-13, analizamos los métodos *Forward* y *Backward* mediante el problema B, mientras que en las Figuras 14-16 analizamos el método *Least Squares Backward* mediante el problema D. Elegimos los problemas B y D ya que

son problemas de altas dimensiones, a diferencia de los problemas A y C. Detallamos las especificaciones de los parámetros de la DNN en cada prueba en el epígrafe de la figura[2].

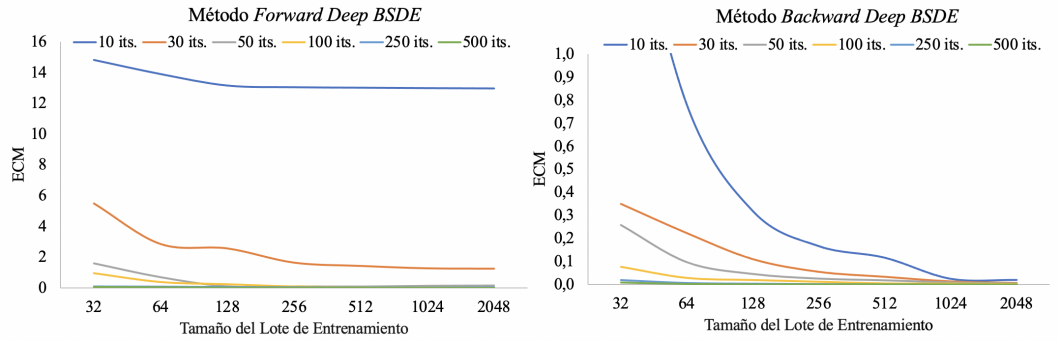

Figura 7: Variación del ECM con el aumento de TBS e ITs. Métodos FDB (panel izquierdo) y BDB (panel derecho). La solución de la BSDE se computa con $dt = 1/2$ y $LR = \phi(1; 5 \times 10^{/4})$.

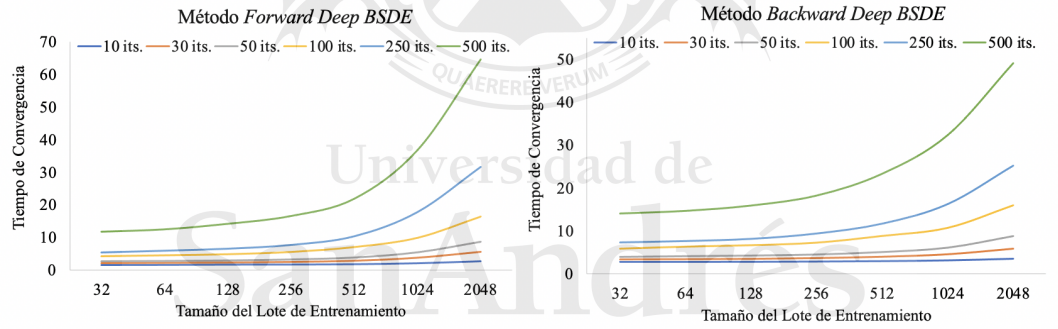

Figura 8: Variación del tiempo de convergencia con el aumento de TBS e ITs. Métodos FDB (panel izquierdo) y BDB (panel derecho). La solución de la BSDE se computa con $dt = 1/2$ y $LR = \phi(1; 5 \times 10^{/4})$.

Con respecto la Figura 7, observamos que en ambos métodos un aumento del TBS genera una reducción en el ECM. La diferencia radica en que para el método FDB, un aumento en las ITs reduce en mayor medida el ECM que un aumento en el TBS, mientras que en el método BDB ocurre lo contrario. Por otro lado, hay que destacar que un aumento en las ITs es más costoso computacionalmente que un aumento en el TBS, como podemos apreciar en la Figura 8. También, se destaca que los tiempos de convergencia de ambos

[2]Excepto en las figuras 12, 13 y 16, la cantidad de capas ocultas es 2 y la cantidad de neuronas por capa oculta es $d + 10$ en todos los casos.

métodos son comparables, no hay uno que sea sustancialmente menor que el otro.

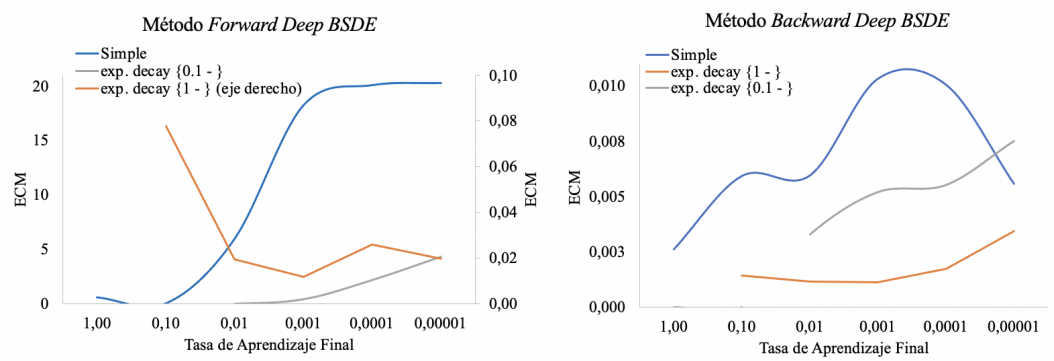

Figura 9: Variación del ECM con el aumento de LR. Métodos FDB (panel izquierdo) y BDB (panel derecho). La solución de la BSDE se computa con $ITs = 250$, $TBS = 256$ y $dt = 1/2$.

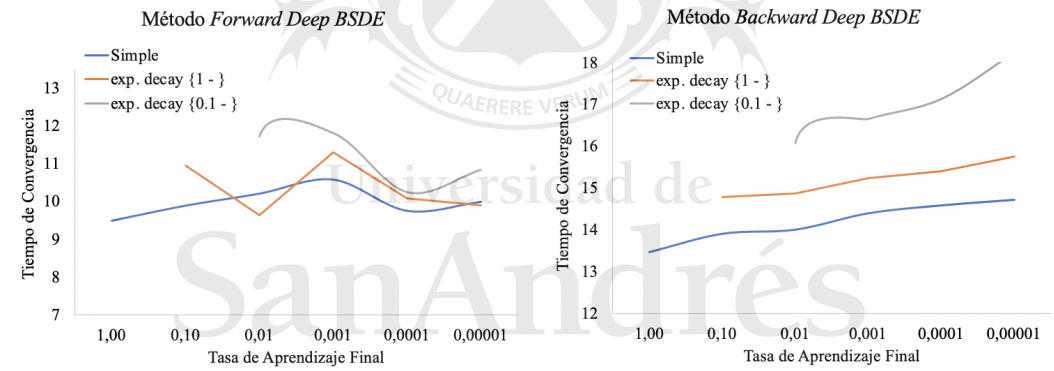

Figura 10: Variación del tiempo de convergencia con el aumento de LR. Métodos FDB (panel izquierdo) y BDB (panel derecho). La solución de la BSDE se computa con $ITs = 250$, $TBS = 256$ y $dt = 1/2$.

En la Figura 9 podemos constatar que un esquema de LR exponencial mejora sustancialmente la convergencia del método FDB, en especial un esquema exponencial que comienza en 1. En el caso del método BDB, observamos que LR no tiene un gran impacto sobre la convergencia del algoritmo.
Con respecto al tiempo de convergencia de los algoritmos, la Figura 10 muestra que en el FDB no hay una diferencia considerable entre los distintos esquemas. En el BDB, en cambio, el esquema simple es el menos costoso computacionalmente.

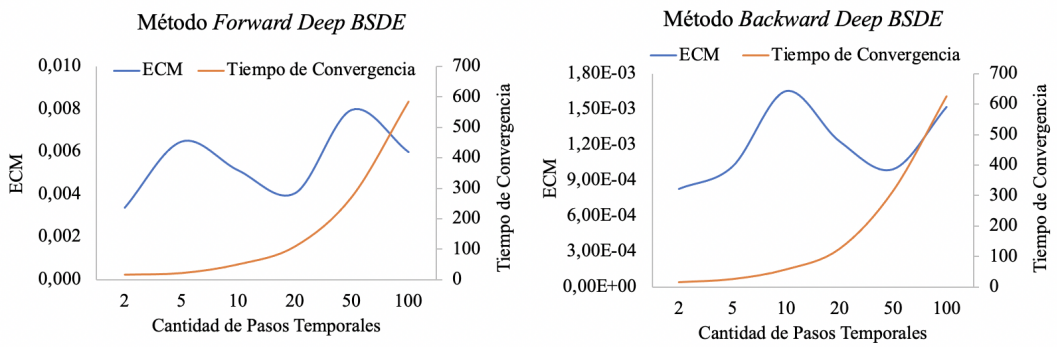

Figura 11: Variación del ECM y el tiempo de convergencia con el aumento de la discretizacion temporal. Métodos FDB (panel izquierdo) y BDB (panel derecho). La solución de la BSDE se computa con $ITs = 100$, $TBS = 2048$ y $LR = \phi(1; 5{\times}10^{/4})$.

La Figura 11 muestra claramente que la cantidad de pasos temporales no afecta a la convergencia de los métodos, posiblemente porque la BSDE europea tiene solución exacta (esta variable si es fundamental en la valuación de opciones de estilo americano/bermuda porque se relaciona con las oportunidades de ejercicio anticipado). Por otro lado, el aumento de la cantidad de pasos temporales si resulta en un aumento exponencial del tiempo de convergencia.

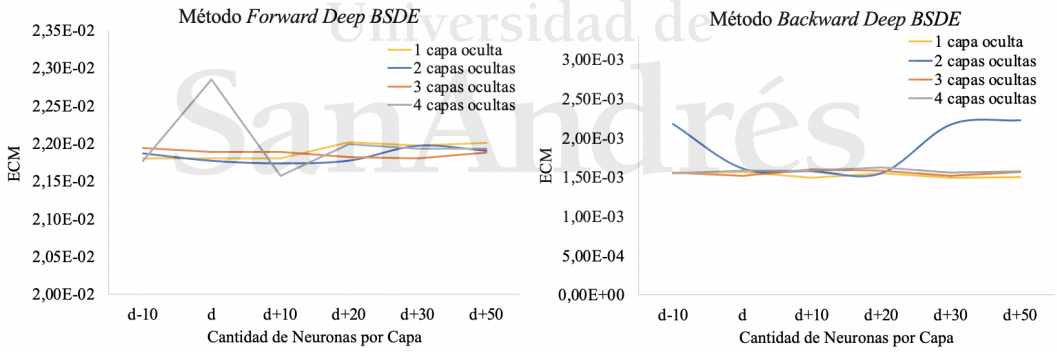

Figura 12: Variación del ECM con el aumento de la cantidad de neuronas y la cantidad de capas ocultas en la DNN. Métodos FDB (panel izquierdo) y BDB (panel derecho). La solución de la BSDE se computa con $ITs = 250$, $TBS = 256$, $dt = 1/2$ y $LR = \phi(1; 5 \times 10^{/4})$.

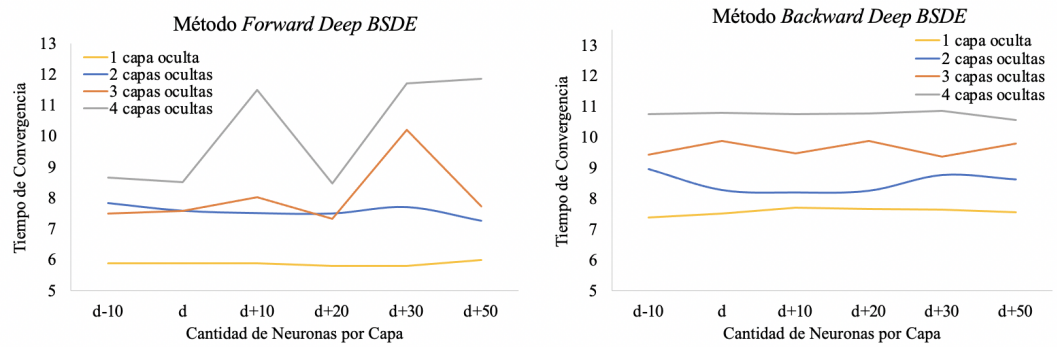

Figura 13: Variación del tiempo de convergencia con el aumento de la cantidad de neuronas y la cantidad de capas ocultas en la DNN. Métodos FDB (panel izquierdo) y BDB (panel derecho). La solución de la BSDE se computa con $ITs = 250$, $TBS = 256$, $dt = 1/2$ y $LR = \phi(1; 5 \times 10^{/4})$.

Podemos observar otros resultados interesantes en las figuras 12 y 13. En primer lugar, observamos que tanto un aumento en la cantidad de capas ocultas como un aumento en la cantidad de neuronas por capa oculta, no tienen prácticamente ningún efecto sobre la convergencia ambos algoritmos. Además, observamos que el tiempo de convergencia de los métodos aumenta con la cantidad de capas ocultas.

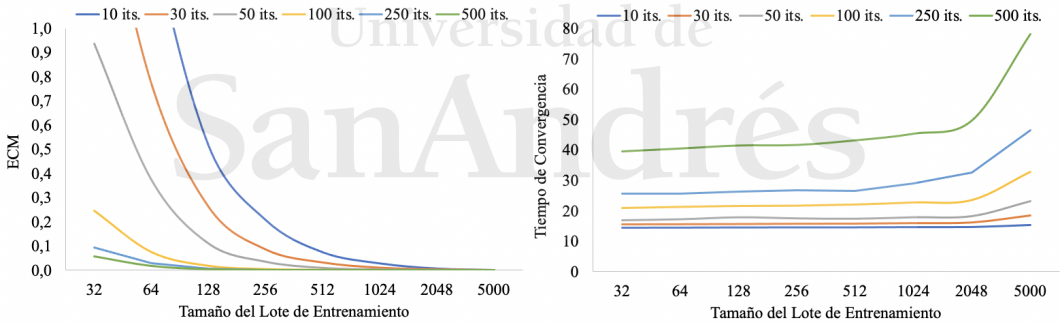

Figura 14: Variación del ECM (panel izquierdo) y el tiempo de convergencia (panel derecho) con el aumento de TBS e ITs. Método LSBD. La solución de la BSDE se computa con $dt = 1/10$ y $LR = \phi(1; 5 \times 10^{/4})$.

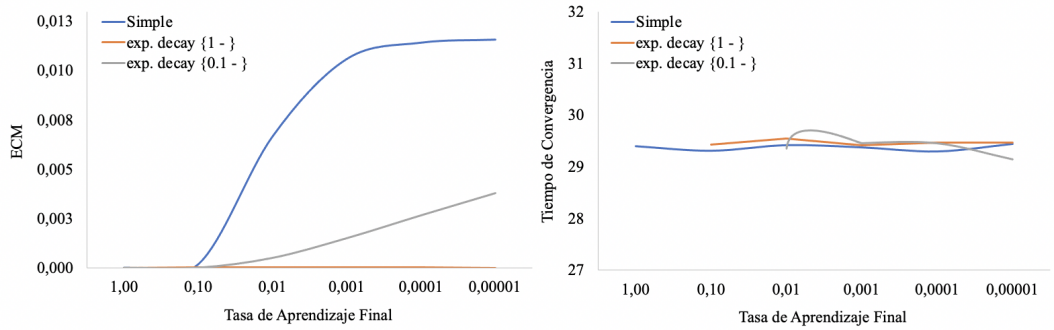

Figura 15: Variación del ECM (panel izquierdo) y el tiempo de convergencia (panel derecho) con el aumento de LR. Método LSBD. La solución de la BSDE se computa con $ITs = 250$, $TBS = 256$ y $dt = 1/10$.

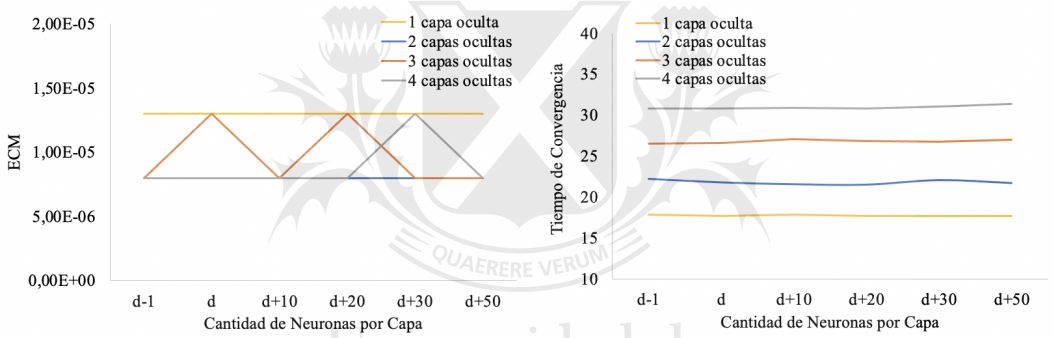

Figura 16: Variación del ECM (panel izquierdo) y el tiempo de convergencia (panel derecho) con el aumento de la cantidad de neuronas y la cantidad de capas ocultas en la DNN. Método LSBD. La solución de la BSDE se computa con $ITs = 250$, $TBS = 2048$, $dt = 1/10$ y $LR = \phi(1; 5 \times 10^{/4})$.

En el caso del método LSBD, obtuvimos resultados muy similares a los del método BDB. El TBS es una variable fundamental en el LSBD para reducir el ECM (Figura 14), así como lo es en el BDB. En la Figura 14 vemos que el LSBD tiene un tiempo de convergencia mayor que el BDB debido al ejercicio anticipado. También, la Figura 16 muestra que un aumento de la cantidad de capas y neuronas en la DNN no mejora la convergencia del método mientras que aumenta el tiempo de convergencia del algoritmo, lo mismo ocurre con los métodos FDB y BDB.
En resumen, la variables fundamentales para reducir el ECM en el método FDB son la cantidad de iteraciones y la tasa de aprendizaje, como podemos observar en las figuras 7 y 9. En cambio, para los métodos BDB y LSBD, la variable que contribuye a reducir considerablemente el ECM es el lote de entrenamiento (figuras 7 y 14). Por otro lado, observamos en las figuras 13 y

16 que un aumento en la cantidad de capas y/o neuronas en la DNN no es relevante para la reduccion del ECM en los métodos *Deep BSDE*, mientras que genera un aumento en el tiempo de convergencia.

5.1.3. Pruebas de Eficiencia

En esta sección comparamos la eficiencia de los métodos propuestos en este trabajo en con los métodos de Monte Carlo (MC) y *Least Squares* Monte Carlo (LSMC).
En tablas 5-8 y 9-10 consignamos los resultados de la valuación de los problemas B y D, respectivamente.
En las tablas 5 y 6, $TBS = 64$, $dt = T/2$ y $LR = \phi(1; 5 \times 10^{/4})$. El objetivo es comparar los métodos FDB y BDB con MC para una misma cantidad de caminos simulados. Es decir, la cantidad de caminos simulados en MC es equivalente a $ITs \times TBS$, excepto en el caso de 5000 iteraciones donde la cantidad de caminos simulados en MC es $1,000,000$ (solución de referencia para el problema). En la tabla 7, $ITs = 250$, $TBS = 256$, $dt = T/2$ y $LR = \phi(1; 5 \times 10^{/4})$. Esta tabla tiene como objetivo exponer la eficiencia de los métodos FDB y BDB en comparación con MC cuando aumenta la dimensionalidad del problema. En esta prueba, también, los caminos simulados en MC son equivalentes a $ITs \times TBS$. La Tabla 8 contiene las soluciones de referencia para la Tabla 7 ya que tienen la mayor cantidad de caminos con las que hemos computado y, por lo tanto, son las de mayor precisión que disponemos.

ITs	*Forward Deep BSDE*			*Backward Deep BSDE*		
	Error Relativo	**Desvio/ Media**	**Tiempo de Converg.**	**Error Relativo**	**Desvio/ Media**	**Tiempo de Converg.**
10	−17,37 %	1,51 %	1,68	0,22 %	4,11 %	2,75
30	−7,60 %	2,22 %	2,17	−0,29 %	2,19 %	3,38
50	−2,40 %	3,09 %	2,84	−0,15 %	1,44 %	4,13
100	1,50 %	2,33 %	4,47	0,12 %	0,76 %	6,31
250	−0,09 %	1,17 %	5,90	−0,07 %	0,36 %	7,69
500	0,10 %	0,76 %	12,56	−0,05 %	0,21 %	14,74
5.000	−0,03 %	0,17 %	110,91	0,00 %	0,06 %	93,79

Tabla 5: Prueba de eficiencia en comparación con MC. Opción de estilo europea con un subyacente de 100 dimensiones, métodos FDB y BDB.

		Método de Monte Carlo		
Cant. de Caminos	**Media**	**Error Relativo**	**Desvio/ Media**	**Tiempo de Converg.**
640	21,09	−1,43 %	4,42 %	0,56
1.920	21,15	−1,15 %	2,27 %	0,63
3.200	22,06	3,10 %	1,82 %	0,66
64.00	21,14	−1,23 %	1,26 %	0,73
16.000	21,55	0,69 %	0,81 %	1,03
32.000	21,47	0,31 %	0,57 %	1,54
1.000.000	21,40	0,00 %	0,10 %	31,14

Tabla 6: Resultados de la prueba de eficiencia para MC. Opción de estilo europea con un subyacente de 100 dimensiones. Esta tabla es para comparar MC con *Deep BSDE* (Tabla 5).

Dims.	*Forward Deep BSDE*			*Backward Deep BSDE*			Método de Monte Carlo		
	Error Relativo	**Desvio/ Media**	**Tiempo de Converg.**	**Error Relativo**	**Desvio/ Media**	**Tiempo de Converg.**	**Error Relativo**	**Desvio/ Media**	**Tiempo de Converg.**
1	0,30 %	0,95 %	2,52	0,16 %	0,46 %	4,64	0,84 %	0,78 %	1,77
2	0,14 %	0,82 %	2,60	0,17 %	0,49 %	4,75	−0,22 %	0,71 %	1,80
5	0,05 %	0,83 %	2,66	−0,01 %	0,34 %	4,75	−0,76 %	0,54 %	1,80
10	0,02 %	0,67 %	2,75	0,02 %	0,17 %	4,83	−0,08 %	0,44 %	1,96
25	0,07 %	0,60 %	2,88	0,04 %	0,22 %	4,98	0,35 %	0,43 %	2,19
50	0,10 %	0,43 %	4,08	−0,02 %	0,19 %	6,53	0,14 %	0,42 %	2,79
100	−0,02 %	0,56 %	5,61	0,01 %	0,19 %	7,86	0,30 %	0,41 %	3,88
250	0,10 %	0,58 %	17,19	−0,16 %	0,18 %	20,60	−0,65 %	0,40 %	6,75
500	−0,15 %	0,64 %	74,42	−0,03 %	0,17 %	80,35	−0,37 %	0,40 %	12,07

Tabla 7: Prueba de eficiencia aumentando la cantidad de dimensiones. Métodos FDB, BDB y MC. En el caso del método de MC, la cantidad de caminos es equivalente a $ITs \times TBS$ de los métodos *Deep BSDE* (64.000 en todos los casos).

			Método de Monte Carlo		
Dims.	**Cant. de Caminos**	**Lote**	**Media**	**Desvio/ Media**	**Tiempo de Converg.**
1	1.000.000	1.000.000	19,15	0,20 %	0,52
2	1.000.000	1.000.000	24,32	0,18 %	0,90
5	1.000.000	1.000.000	15,60	0,14 %	1,82
10	1.000.000	1.000.000	15,54	0,11 %	3,30
25	1.000.000	1.000.000	19,82	0,11 %	7,72
50	1.000.000	1.000.000	20,74	0,11 %	15,02
100	1.000.000	500.000	21,40	0,10 %	31,14
250	1.000.000	200.000	21,84	0,10 %	82,71
500	600.000	30.000	22,16	0,13 %	105,41

Tabla 8: Valores de referencia para el cálculo del error relativo en la Tabla 7. Estos valores corresponden a la expectación de MC en su máxima potencia computacional, es decir se simulan la máxima cantidad de caminos que permite la memoria RAM.

La Tabla 5 muestra que a igualdad de condiciones, el método BDB consigue un error relativo de 0 % y un cociente de desvió estándar sobre media menor a MC. A su vez, observamos que MC es sustancialmente más veloz que los métodos FDB y BDB. En la misma línea, la Tabla 7 exhibe resultados similares. El método BDB tiene un error relativo y un desvío sobre media menor que MC en todos los casos, mientras que MC converge siempre en menor tiempo. En el caso del método FDB, observamos que es menos eficiente en comparación con BDB y MC en ambas pruebas.
La Tabla 9 tiene el mismo objetivo que las tablas 5 y 6 pero compara el método LSBD con LSMC. Utilizamos $TBS = 2048$, $dt = T/10$ y $LR = \phi(1; 5 \times 10^{/4})$. En el mismo sentido, la Tabla 10 tiene el mismo objetivo que las tablas 7 y 8 pero comparando el método LSBD con LSMC. A diferencia de la Tabla 7, la cantidad de caminos simulados en LSMC no es igual a $ITs \times TBS$ sino que utilizamos la máxima potencia computacional del algoritmo, es decir generamos la cantidad máxima de caminos que nos permite la memoria RAM. Utilizamos $ITs = 100$, $TBS = 2048$, $dt = T/10$ (donde la opcion puede ejercerse en cada t) y $LR = \phi(1; 5 \times 10^{/4})$.

ITs	*Least Squares Backward DNN*					*Least Squares* Monte Carlo		
	Error Relativo	**Desvio/ Media**	**Tiempo de Converg.**	**Cant. de Caminos**	**Media**	**Error Relativo**	**Desvio/ Media**	**Tiempo de Converg.**
10	3,05 %	0,69 %	14,59	20.480	3,23	1,84 %	0,83 %	5,55
30	1,82 %	0,32 %	15,86	61.440	3,20	0,96 %	0,48 %	10,46
50	0,84 %	0,18 %	17,18	102.400	3,19	0,76 %	0,37 %	19,70
100	0,22 %	0,07 %	20,63	204.800	3,18	0,19 %	0,26 %	15,68
250	0,11 %	0,05 %	30,11	512.000	3,17	0,06 %	0,17 %	26,99
500	0,05 %	0,03 %	46,96	1.000.000	3,17	0,00 %	0,12 %	57,99

Tabla 9: Prueba de eficiencia del método LSBD en comparación LSMC.

Dims.	*Least Squares Backward DNN*					*Least Squares* Montecarlo		
	Media	**Desvio/ Media**	**Tiempo de Converg.**	**Cant. de Caminos**	**Lote**	**Media**	**Desvio/ Media**	**Tiempo de Converg.**
1	5,69	0, 06 %	17,95	1.000.000	1.000.000	5,68	0, 11 %	11,48
2	2,99	0, 07 %	18,47	1.000.000	1.000.000	2,99	0, 12 %	18,66
5	3,55	0, 08 %	19,54	1.000.000	1.000.000	3,55	0, 12 %	26,36
10	3,17	0, 07 %	20,88	1.000.000	1.000.000	3,17	0, 12 %	57,99
25	3,28	0, 08 %	29,89	1.000.000	250.000	3,30	0, 12 %	337,02
50	3,65	0, 08 %	47,73					
100	3,68	0, 06 %	20,63		Fuera de Memoria			
250	3,90	0, 09 %	167,88					
500	4,05	0, 08 %	347,76					

Tabla 10: Prueba de eficiencia aumentando la cantidad de dimensiones. Métodos LSBD y LSMC

Las tablas 9 y 10 muestran una clara superioridad del método LSBD en comparación con LSMC. Por un lado, en la Tabla 9 vemos que LSBD consigue un error relativo bajo y un desvío sobre media menor a LSMC. También, en 500 iteraciones vemos que LSBD converge más rápidamente que LSMC. Por otro lado, la Tabla 10 es distinta a las tablas 7 y 8 ya que LSBD no degrada su rendimiento notablemente al aumentar la dimensión del problema, como si lo hace LSMC. Y si la dimensión del problema es considerablemente alta, LSMC se hace impracticable. En este sentido, el objetivo de esta tabla es mostrar la clara superioridad de LSBD que puede valuar derivados en alta dimensionalidad con un costo computacional sustancialmente menor. Cabe aclarar que en este trabajo utilizamos la versión secuencial de LSMC, donde los caminos se generan sucesivamente. Es decir, a pesar de generar los caminos en lotes en algunos casos, el procesador de la computadora produce un lote detrás de otro, no distribuye su cómputo en distintos núcleos. Por otro lado, existen métodos computacionales para paralelizar el algoritmo. Es decir, el procesador distribuye la generación de caminos entre sus distintos núcleos, reduciendo considerablemente el tiempo de convergencia del algoritmo. Este es el caso de Liang et al. (2019, 2021), aunque estos exceden el alcance de esta tesis.
En resumen, BDB obtiene un error relativo y un desvío sobre media menores a MC, mientras que FDB resulta ser menos eficiente, obteniendo un error relativo y un desvío sobre media por encima de MC. En cambio, MC continua siendo más veloz que los métodos *Deep BSDE*. Por otro lado, LSBD consigue resultados superiores a LSMC, obteniendo un error relativo menor, un desvió sobre media menor y, en algunos casos, un tiempo de convergencia menor. También, demuestra ser ampliamente superador para valuar opciones en alta dimensionalidad.

5.2. Problemas con generadores no lineales

En esta sección presentamos resultados de la valuación de opciones en marcos teóricos más generales que el de Black, Scholes y Merton (BSM) como los descriptos en la Sección 3.2. En particular nos interesa analizar la convergencia de los métodos propuestos en problemas con generadores no lineales. A continuación detallamos el contenido de las tablas de esta sección:

- Tabla 11: resultados de la valuación de un *spread* de dos opciones de compra de estilo europeo con un activo subyacente de 100 dimensiones, con distintas tasas de interés para prestar y pedir prestado dinero. El pago al vencimiento de esta opción esta dado por la siguiente ecuación:

$$\text{máx}(\text{máx}(X) - \$120, 0) - 2\,\text{máx}(\text{máx}(X) - \$150, 0),$$

donde X es un activo subyacente de 100 dimensiones. Esta opción tiene $T = 0{,}5$ años, $X_0 = \$100$, $r^l = 4\,\%$, $r^b = 6\,\%$, $\mu = 6\,\%$, $\sigma = 20\,\%$ para todos los activos subyacentes y la correlación entre ellos es 0. La solución de referencia es \$21,3, la cual fue obtenida en E et al. (2017) para valuar el mismo problema.

- Tabla 12: resultados de la valuación de una opción de venta de estilo europeo con un activo subyacente uni-dimensional, con distintas tasas de interés para prestar y pedir prestado. La opción tiene $T = 0{,}5$ años, $K = \$100$ (ATM), $r^l = 4\,\%$, $r^b = 6\,\%$, $\mu = 6\,\%$ y $\sigma = 20\,\%$. La solución de referencia es \$4,2, la cual obtenemos por FDM explicito.

- Tabla 13: resultados de la valuación de una opción de venta de estilo bermuda con un activo subyacente uni-dimensional, con distintas tasas de interés para prestar y pedir prestado. Las características de esta opción son las mismas que en la Tabla 12. La opción puede ejercerse cada 9 días, es decir $dt = 1/20$. La solución de referencia es \$4,49, la cual obtenemos por FDM explicito incluyendo el ejercicio anticipado.

- Tabla 14: resultados de la valuación de una opción de estilo europeo con un activo subyacente de 100 dimensiones, incorporando riesgo de *default*. El pago al vencimiento de esta opción es mín(X), donde X es un activo subyacente de 100 dimensiones. Esta opción tiene $T = 1$ año, $X_0 = \$100$, $r = 2\,\%$, $\sigma = 20\,\%$ para todos los activos y la correlación es igual a 0. Por otro lado, los parámetros de *default* (Ecuación (8)) son $\delta = 2/3$, $\gamma^h = 0{,}2$, $\gamma^l = 0{,}02$, $\upsilon^h = 50$ y $\upsilon^l = 70$. La solución de referencia es \$57,3, la cual fue obtenida en Han et al. (2018) para valuar el mismo problema.

- Tabla 15: resultados de la valuación de una opción de venta de estilo europeo con un activo subyacente uni-dimensional, incorporando riesgo de *default*. Las características de esta opción son las mismas que en la Tabla 14. La solución de referencia es \$6,48, la cual obtenemos por FDM explícito.

- Tabla 16: resultados de la valuación de una opción de venta de estilo bermuda con un activo subyacente uni-dimensional, incorporando riesgo de *default*. Las características de esta opción son las mismas que en las tablas 14 y 16. La opción puede ejercerse cada 18 días, es decir $dt = 1/20$. La solución de referencia es \$6,81, la cual obtenemos nuevamente por FDM explícito.

Respecto a las características de la DNN, en todas las tablas de esta sección utilizamos $TBS = 256$ y $LR = \phi(1; 5 \times 10^{/4})$. Para el caso de las opciones de estilo europeo (tablas 11-12, 14-15) $dt = T/2$ mientras que para las opciones bermuda (tablas 13 y 16) $dt = T/20$.
En complemento a las tablas 11 y 14, incluimos al final las tablas 17 y 18 que corresponden a los mismos problemas pero en su versión bermuda como un aporte a la literatura, debido a que estos problemas no tienen solución de referencia. En la Tabla 17, la opción puede ejercerse cada 25 días ($dt = 1/7$) y en la Tabla 18 el ejercicio esta permitido cada 36 días ($dt = 1/10$).

	Forward Deep BSDE			*Backward Deep BSDE*		
ITs	**Error Relativo**	**I.C. 95 % Error Relativo**	**Tiempo Converg.**	**Error Relativo**	**I.C. 95 % Error Relativo**	**Tiempo Converg.**
10	−16,44 %	[−16, 46 %; −16, 42 %]	1,77	−0,11 %	[−0, 92 %; 0, 72 %]	3,12
30	−4,81 %	[−5, 02 %; −4, 56 %]	2,06	−0,05 %	[−0, 38 %; 0, 25 %]	3,44
50	2,27 %	[1, 74 %; 2, 61 %]	2,37	−0,19 %	[−0, 60 %; 0, 13 %]	3,80
100	0,39 %	[−0, 11 %; 0, 78 %]	3,08	−0,44 %	[−0, 70 %; −0, 25 %]	4,76
250	−0,11 %	[−0, 34 %; 0, 05 %]	5,25	−0,87 %	[−1, 00 %; −0, 76 %]	7,58
500	0,01 %	[−0, 14 %; 0, 15 %]	8,87	−0,95 %	[−1, 05 %; −0, 83 %]	12,29
5.000	0,02 %	[−0, 04 %; 0, 09 %]	73,46	−0,94 %	[−0, 97 %; −0, 91 %]	97,55

Tabla 11: Resultados de la valuación de un *spread* de dos opciones de compra con un activo subyacente de 100 dimensiones, para el problema de distintas tasas de interés. Métodos FDB y BDB.

	Forward Deep BSDE			*Backward Deep BSDE*		
ITs	**Error Relativo**	**I.C. 95 % Error Relativo**	**Tiempo Converg.**	**Error Relativo**	**I.C. 95 % Error Relativo**	**Tiempo Converg.**
10	280,80 %	[280,78 %; 280,82 %]	2,13	−0,14 %	[−4,19 %; 3,22 %]	3,83
30	216,68 %	[216,48 %; 216,88 %]	2,30	−0,10 %	[−1,48 %; 1,34 %]	4,11
50	154,08 %	[153,55 %; 154,44 %]	2,46	−0,06 %	[−1,03 %; 0,87 %]	4,35
100	35,07 %	[34,21 %; 36,16 %]	2,88	0,10 %	[−0,46 %; 0,70 %]	5,06
250	0,04 %	[−0,30 %; 0,28 %]	4,14	0,10 %	[−0,34 %; 0,53 %]	7,15
500	0,06 %	[−0,28 %; 0,38 %]	6,19	0,08 %	[−0,12 %; 0,31 %]	10,79
5.000	0,09 %	[−0,12 %; 0,26 %]	42,72	0,10 %	[0,03 %; 0,20 %]	75,97

Tabla 12: Resultados de la valuación de una opción de venta con un activo subyacente uni-dimensional, para el problema de distintas tasas de interés. Métodos FDB y BDB.

ITs	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.
10	5,11 %	[1,58 %; 8,69 %]	40,40
30	3,31 %	[2,00 %; 4,84 %]	43,43
50	2,05 %	[0,97 %; 2,96 %]	46,30
100	0,46 %	[0,10 %; 0,69 %]	53,03
250	0,15 %	[0,00 %; 0,29 %]	71,61
500	0,07 %	[−0,10 %; 0,18 %]	99,94
5.000	0,01 %	[−0,02 %; 0,04 %]	560,01

Tabla 13: Resultados de la valuación de una opción de venta con un activo subyacente uni-dimensional, para el problema de diferentes tasas de interés. Método LSBD.

	Forward Deep BSDE			*Backward Deep BSDE*		
ITs	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.
10	−68,93 %	[−68,93 %; −68,93 %]	1,73	−0,05 %	[−0,43 %; 0,28 %]	3,07
30	−64,18 %	[−64,19 %; −64,18 %]	2,04	−0,08 %	[−0,20 %; 0,10 %]	3,53
50	−59,36 %	[−59,37 %; −59,35 %]	2,33	−0,05 %	[−0,16 %; 0,12 %]	3,93
100	−47,88 %	[−47,91 %; −47,84 %]	3,16	−0,04 %	[−0,12 %; 0,04 %]	5,18
250	−20,20 %	[−20,31 %; −20,08 %]	5,56	−0,05 %	[−0,1 %; −0,01 %]	8,19
500	−2,99 %	[−3,08 %; −2,91 %]	9,42	−0,06 %	[−0,1 %; −0,02 %]	13,37
5.000	−0,25 %	[−0,28 %; −0,22 %]	79,94	−0,06 %	[−0,08 %; −0,05 %]	110,00

Tabla 14: Resultados de la valuación de una opción mínimo con un activo subyacente de 100 dimensiones, para el problema de riesgo de *default*. Métodos FDB y BDB.

	Forward Deep BSDE			*Backward Deep BSDE*		
ITs	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.
10	146,27 %	[146,24 %; 146,29 %]	1,42	−0,51 %	[−4,08 %; 3,16 %]	2,84
30	105,01 %	[104,72 %; 105,29 %]	1,53	0,28 %	[−1,59 %; 1,76 %]	3,05
50	65,54 %	[64,84 %; 66,14 %]	1,63	0,18 %	[−0,51 %; 1,15 %]	3,25
100	2,38 %	[1,41 %; 3,92 %]	1,89	0,26 %	[−0,26 %; 0,90 %]	3,84
250	0,52 %	[−0,05 %; 0,89 %]	2,67	0,15 %	[−0,22 %; 0,70 %]	5,41
500	0,17 %	[−0,07 %; 0,56 %]	4,00	0,16 %	[−0,11 %; 0,47 %]	8,08
5.000	0,11 %	[−0,07 %; 0,36 %]	27,46	0,13 %	[0,08 %; 0,21 %]	55,30

Tabla 15: Resultados de la valuación de una opción de venta con un activo subyacente uni-dimensional, para el problema de riesgo de *default*. Métodos FDB y BDB.

ITs	Error Relativo	I.C. 95 % Error Relativo	Tiempo Converg.
10	4,34 %	[2,14 %; 7,25 %]	38,31
30	3,18 %	[2,15 %; 4,71 %]	40,39
50	1,75 %	[0,93 %; 2,63 %]	42,00
100	0,42 %	[0,14 %; 0,79 %]	47,09
250	0,14 %	[−0,04 %; 0,30 %]	63,41
500	0,08 %	[−0,01 %; 0,17 %]	89,23
5.000	0,00 %	[−0,02 %; 0,03 %]	628,71

Tabla 16: Resultados de la valuación de una opción de venta con un activo subyacente uni-dimensional, para el problema de riesgo de *default*. Método LSBD.

ITs	Media	Desvio Estandar	IC 95 % (1)	IC 95 % (2)	Tiempo Converg.
10	22,55	0,04	22,48	22,59	20,03
30	22,51	0,02	22,48	22,54	26,63
50	22,47	0,02	22,45	22,50	33,15
100	22,39	0,01	22,38	22,42	49,69
250	22,37	0,01	22,36	22,38	98,77
500	22,36	0,00	22,36	22,37	181,58

Tabla 17: Resultados de la valuación de un *spread* de dos opciones de compra con un activo subyacente de 100 dimensiones, para el problema de distintas tasas de interés. Método LSBD.

ITs	Media	Desvio Estandar	IC 95 % (1)	IC 95 % (2)	Tiempo Converg.
10	85,13	0,02	85,11	85,16	32,86
30	85,13	0,01	85,11	85,14	43,28
50	85,13	0,01	85,12	85,14	49,32
100	85,13	0,01	85,12	85,14	73,65
250	85,13	0,00	85,12	85,13	146,77
500	85,13	0,00	85,12	85,13	264,53

Tabla 18: Resultados de la valuación de una opción mínimo con un activo subyacente de 100 dimensiones, para el problema de riesgo de *default*. Método LSBD.

Observamos que, así como en los problemas lineales, en todos los casos analizados, los métodos FDB, BDB y LSBD convergen a las soluciones de referencia. Existe un único caso donde el error relativo es más elevado en relación a los niveles que observamos en la gran mayoría de las pruebas, este corresponde a BDB en la Tabla 11. Debido a esto, realizamos un análisis completo de ajuste de parámetros para ambos problemas no lineales y en los métodos FDB y BDB, para determinar el origen de la diferencia remanente. Este

análisis arrojó los mismos resultados que la Sección 5.1.2, por lo tanto no encontramos un comportamiento generalizado de los métodos *Deep BSDE* en la valuación de problemas con generadores no lineales que indique un problema en la convergencia. En este sentido, considerando que la magnitud del error es reducida (debajo del 1 %) y que en las tablas 12 y 13, el método BDB converge con un error relativo compatible con lo observado en la gran mayoría de las pruebas, concluimos que se debe a una particularidad del problema en sí y no refleja un sesgo de BDB.

6. Conclusión

De la Sección 5 destacamos que los métodos *Deep BSDE* convergen a las soluciones de referencia en todos los casos analizados, tanto en instrumentos europeos como americanos/bermuda, en problemas lineales y no lineales. En relación a los parámetros de la DNN, encontramos que las variables fundamentales del método *Forward Deep BSDE* (FDB) son la cantidad/número de iteraciones y la tasa de aprendizaje, mientras que en los métodos *Backward Deep BSDE* (BDB) y *Least Squares Backward DNN* (LSBD) es el lote de entrenamiento. Por otro lado, las pruebas de eficiencia muestran que BDB obtiene un error relativo y un desvío estándar sobre media menores a Monte Carlo (MC), aunque MC continua siendo más veloz incluso en problemas de 500 dimensiones. Por otro lado, el método FDB obtiene un error relativo y un desvío sobre media por encima de MC y BDB. En el caso de instrumentos con ejercicio anticipado, el método LSBD obtiene resultados ampliamente mejores que *Least Squares* Monte Carlo (LSMC), llegando a valuar opciones de 500 dimensiones mientras que LSMC llega a 25 dimensiones. Consideramos que la alta eficiencia de los métodos *Deep BSDE* en la valuación de problemas con generadores lineales puede extenderse al campo de lo no lineal. De esta manera, concluimos que los 3 métodos propuestos (FDB, BDB y LSBD) son factibles y considerablemente eficientes en la valuación de derivados financieros en altas dimensiones en contextos lineales y no lineales, de estilo europeo y americano/bermuda. Como una futura linea de investigación, proponemos profundizar el análisis de estos métodos en la valuación de derivados con ejercicio anticipado, en contextos no lineales y de alta dimensionalidad. No obstante, cabe resaltar que en esta tesis hemos obtenido valores originales por primera vez en la literatura para estos problemas.

Referencias

Christian Beck, Weinan E, and Arnulf Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29(4):1563–1619, 2019.

Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Timo Welti. Solving high-dimensional optimal stopping problems using deep learning. *arXiv preprint arXiv:1908.01602*, 2019.

Yaacov Z Bergman. Option pricing with differential interest rates. *The Review of Financial Studies*, 8(2):475–500, 1995.

Quentin Chan-Wai-Nam, Joseph Mikael, and Xavier Warin. Machine learning for semi linear pdes. *Journal of Scientific Computing*, 79(3):1667–1712, 2019.

Jean-François Chassagneux. Linear multistep schemes for bsdes. *SIAM Journal on Numerical Analysis*, 52(6):2815–2836, 2014.

Jean-François Chassagneux and Adrien Richou. Numerical stability analysis of the euler scheme for bsdes. *SIAM Journal on Numerical Analysis*, 53 (2):1172–1193, 2015.

Yangang Chen and Justin WL Wan. Deep neural network framework based on backward stochastic differential equations for pricing and hedging american options in high dimensions. *Quantitative Finance*, 21(1):45–67, 2021.

David Chevance. Numerical methods for backward stochastic differential equations. *Numerical methods in finance*, 232, 1997.

Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

Weinan E, Martin Hutzenthaler, Arnulf Jentzen, and Thomas Kruse. On multilevel picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. *Journal of Scientific Computing*, 79 (3):1534–1571, 2019.

Masaaki Fujii, Akihiko Takahashi, and Masayuki Takahashi. Asymptotic expansion as prior knowledge in deep learning method for high dimensional bsdes. *Asia-Pacific Financial Markets*, 26(3):391–408, 2019.

Narayan Ganesan, Jessica Yajie Yu, and Bernhard Hientzsch. Pricing barrier options with deepbsdes. *arXiv preprint arXiv:2005.10966*, 2020.

Alessandro Gnoatto, Christoph Reisinger, and Athena Picarelli. Deep xva solver–a neural network based counterparty credit risk management framework. *Available at SSRN 3594076*, 2020.

Batuhan Güler, Alexis Laignelet, and Panos Parpas. Towards robust and stable deep learning algorithms for forward backward stochastic differential equations. *arXiv preprint arXiv:1910.11623*, 2019.

Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

J Michael Harrison and Stanley R Pliska. Martingales and stochastic integrals in the theory of continuous trading. *Stochastic processes and their applications*, 11(3):215–260, 1981.

Pierre Henry-Labordere. Deep primal-dual algorithm for bsdes: Applications of machine learning to cva and im. *Available at SSRN 3071506*, 2017.

Bernhard Hientzsch. Introduction to solving quant finance problems with time-stepped fbsde and deep learning. *arXiv preprint arXiv:1911.12231*, 2019.

Côme Huré, Huyên Pham, and Xavier Warin. Deep backward schemes for high-dimensional nonlinear pdes. *Mathematics of Computation*, 89(324): 1547–1579, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Lorenc Kapllani and Long Teng. Deep learning algorithms for solving high dimensional nonlinear backward stochastic differential equations. *arXiv preprint arXiv:2010.01319*, 2020.

Ioannis Karatzas, John P Lehoczky, and Steven E Shreve. Optimal portfolio and consumption decisions for a "small investor" on a finite horizon. *SIAM journal on control and optimization*, 25(6):1557–1586, 1987.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jian Liang, Zhe Xu, and Peter Li. Deep learning-based least square forward-backward stochastic differential equation solver for high-dimensional derivative pricing. *arXiv preprint arXiv:1907.10578*, 2019.

Jian Liang, Zhe Xu, and Peter Li. Deep learning-based least squares forward-backward stochastic differential equation solver for high-dimensional derivative pricing. *Quantitative Finance*, pages 1–15, 2021.

Francis A Longstaff and Eduardo S Schwartz. Valuing american options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147, 2001.

Etienne Pardoux and Shige Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In *Stochastic partial differential equations and their applications*, pages 200–217. Springer, 1992.

Maziar Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *arXiv preprint arXiv:1804.07010*, 2018.

Andrew Senior, Georg Heigold, Marc'aurelio Ranzato, and Ke Yang. An empirical study of learning rates in deep neural networks for speech recognition. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6724–6728. IEEE, 2013.

Steven Shreve. *Stochastic calculus for finance I: the binomial asset pricing model*. Springer Science & Business Media, 2005.

Akihiko Takahashi, Yoshifumi Tsuchida, and Toshihiro Yamada. A new efficient approximation scheme for solving high-dimensional semilinear pdes: control variate method for deep bsde solver. *arXiv preprint arXiv:2101.09890*, 2021.

Haojie Wang, Han Chen, Agus Sudjianto, Richard Liu, and Qi Shen. Deep learning-based bsde solver for libor market model with application to bermudan swaption pricing and hedging. *arXiv preprint arXiv:1807.06622*, 2018.

Bing Yu, Xiaojing Xing, and Agus Sudjianto. Deep-learning based numerical bsde method for barrier options. *arXiv preprint arXiv:1904.05921*, 2019.

Yajie Yu, Bernhard Hientzsch, Narayan Ganesan, Corporate Model Risk, and Wells Fargo. Backward deep bsde methods and applications to nonlinear problems. *arXiv preprint arXiv:2006.07635*, 2020.

Jianfeng Zhang et al. A numerical scheme for bsdes. *Annals of Applied Probability*, 14(1):459–488, 2004.

Wenzhong Zhang and Wei Cai. Fbsde based neural network algorithms for high-dimensional quasilinear parabolic pdes. *arXiv preprint arXiv:2012.07924*, 2020.

A. Demostración de la PDE de Black, Scholes y Merton multidimensional

La ecuación (4) surge de observar que el proceso descontado satisface

$$e^{-\int_0^t r_s ds} Y_t = \mathbb{E}^{\mathbb{Q}}[e^{-\int_0^T r_s ds} Y_t | \mathcal{F}_t],$$

es decir, es martingala y, en la medida $\mathbb{Q}$, su *drift* debe ser nulo. Usando el Lema de Ito sobre el proceso descontado, se obtiene

$$d[e^{-\int_0^T r_s ds} Y(t, X(t))] = e^{-\int_0^T r_s ds} \left\{ \left[-r(t)Y(t, X(t)) + \frac{\partial Y(t, X(t))}{\partial t} \right.\right.$$
$$\left. + r(t)X(t) \cdot \nabla Y(t, X(t))) + \frac{1}{2} \operatorname{Tr}[\tilde{\sigma}\tilde{\sigma}(t, X(t)) Hess_x Y(t, X(t))] \right] dt$$
$$\left. + \nabla Y(t, X(t)) \cdot \tilde{\sigma}(t, X(t)) dW(t) \right\},$$

donde $\tilde{\sigma}(t, X(t)) = diag(X(t))\sigma(t, X(t))$. Si pedimos que el *drift* entre corchetes se anule, obtenemos la ecuación multidimensional de Black, Scholes y Merton.