Resolución de la ecuación diferencial parcial de Black-Scholes mediante redes neuronales físicamente informadas

Solving the Black-Scholes partial differential equation using physically-informed neural networks

John Freddy Moreno Trujillo*

* Estudiante de Doctorado en Ciencias Económicas; magíster en Matemática Aplicada. Docente-Investigador, Observatorio de Economía y Operaciones Numéricas (ODEON), Universidad Externado de Colombia, Bogotá (Colombia). [jhon.moreno@uexternado.edu.co], [ORCID ID: 0000-0002-2772-6931].
Artículo recibido: 01 de junio de 2023.
Aceptado: 07 de julio de 2023.
Para citar este artículo:
Moreno, J. (2023). Resolución de la ecuación diferencial parcial de Black-Scholes mediante redes neuronales físicamente informadas. *Odeon*, 24, 71-92.
DOI: https://doi.org/10.18601/17941113.n24.05

Resumen

Artículo conmemorativo por los 50 años del modelo Black-Scholes, en el cual se presenta la deducción de la ecuación diferencial parcial de valoración en el contexto de un modelo de mercado en tiempo continuo. Se propone como método de resolución el uso de una red neuronal físicamente informada (PINN), como una novedosa técnica del denominado aprendizaje de máquina científico, que permite resolver este tipo de ecuaciones sin la necesidad de un gran número de datos de entrenamiento. Se presenta la implementación del método y los resultados de valoración para el caso de opciones de compra europeas.

Palabras clave: ecuación diferencial parcial; Black-Scholes; valoración; redes neuronales.
Clasificación JEL: C89, G13, G19.

Abstract

Commemorative article for the 50th anniversary of the Black-Scholes model presenting the derivation of the partial differential equation for pricing in the context of a continuous-time market model. The use of a physically-informed neural network (PINN) is proposed as a resolution method, as a novel technique in the field of scientific machine learning, which allows solving these types of equations without the need for a large amount of training data. The article includes the implementation of the method and the valuation results for the case of European call options.

Key words: partial differential equation; Black-Scholes; pricing; neural networks.
JEL classification: C89, G13, G19.

Introducción

El éxito de las redes neuronales en tareas de aprendizaje automático, así como la posibilidad de aprovechar la potencia informática de *hardware* especializado como las unidades de procesamiento tensorial de Google o el motor neural de Apple, diseñados para ejecutar eficientemente redes neuronales, han llevado a la comunidad científica a investigar su idoneidad para tareas de computación de alto rendimiento. Como resultado, se ha desarrollado un nuevo campo de investigación conocido como *aprendizaje automático científico*, donde se aplican técnicas como redes neuronales profundas y aprendizaje estadístico a problemas clásicos de las matemáticas aplicadas. En este documento se busca proporcionar una introducción sencilla a los desarrollos recientes en el campo de la solución numérica de ecuaciones diferenciales parciales (EDP), lineales y no lineales, utilizando técnicas de aprendizaje automático. Específicamente se muestra cómo este tipo de técnicas pueden ser aplicadas para resolver la famosa ecuación diferencial parcial de valoración de derivados de Black-Scholes.

Después de décadas de investigación sobre la solución numérica de EDP, aún persisten numerosos desafíos. Uno que se tiene en prácticamente todos los esquemas de discretización clásicos es la llamada *maldición de la dimensionalidad*, formulada por primera vez por Bellman en la década de los cincuenta en el contexto de problemas de control óptimo (Bellman, 1966). En términos simples, esta noción establece que duplicar el número de grados de libertad en cada una de las d direcciones de discretización aumenta la complejidad de la solución (al menos) por un factor de $2d$. Dado que las redes neuronales son una potente alternativa para aproximar funciones altamente no lineales debido a su su naturaleza compositiva, que contrasta con métodos más convencionales de tipo aditivo, en los cuales se construyen aproximaciones de soluciones de EDP mediante métodos de Galerkin, colocación o volúmenes finitos, estas resultan ser cada vez más adecuadas para resolver EDP lineales, no lineales y de alta dimensión, superando el problema de la dimensionalidad.

Aunque los métodos de aproximación a la solución de EDP basados en redes neuronales (profundas) generalmente no pueden competir con los métodos numéricos clásicos en dimensiones bajas o moderadas, especialmente porque resolver una ecuación algebraica es, por lo general, más simple que resolver los problemas de optimización de gran escala y altamente no lineales asociados con el entrenamiento de redes neuronales, la facilidad con la que se pueden aplicar métodos como las redes neuronales físicamente informadas (PINN - Physics-

informed neural networks) a prácticamente cualquier ecuación diferencial, los hace atractivos para prototipos rápidos cuando la eficiencia y la alta precisión no son la principal preocupación.

El objetivo central de este documento es presentar la aplicación de las PINN para aproximar la solución de la EDP de Black-Scholes, no solo como una forma alternativa que explota herramientas y desarrollos recientes, sino también como un homenaje a los 50 años del seminal trabajo de Fisher Black y Myron Scholes (Black y Scholes, 1973) que ha impactado de forma más que significativa la teoría y práctica financiera desde su publicación. El documento está organizado en cuatro partes: en la primera parte se presenta el modelo de mercado financiero en tiempo continuo y se realiza la deducción por argumentos de no arbitraje de la EDP de Black-Scholes; en la segunda parte se describen las redes neuronales físicamente informadas (PINN); en la tercera parte se realiza la implementación de las PINN en la EDP de Black-Scholes para el caso de opciones de compra europeas; en la última parte se presentan algunas conclusiones y se discuten posibles extensiones.

1. La ecuación diferencial parcial de Black-Scholes

En este apartado se realiza la deducción de la ecuación diferencial parcial de Black-Scholes (EDP-BS) utilizando argumentos de no arbitraje. Consideramos un mercado en tiempo continuo conformado por tres tipos de activos modelados sobre un espacio de probabilidad filtrado $(\Omega, \mathcal{F}, \mathbb{P}, (\mathcal{F}_t)_{t\in[0,T]})$. Los activos son:

- Un activo libre de riesgo (B_t) que satisface la ecuación diferencial ordinaria (EDO):

$$dB_t = rB_t dt \quad ; \quad B_t = B_0 e^{rt} \tag{1}$$

  donde: r es una tasa de interés libre de riesgo constante y conocida.

- Un activo riesgoso (S_t) que satisface la ecuación diferencial estocástica (EDE):

$$dS_t = \alpha S_t dt + \sigma S_t dW_t \quad ; \quad S_t = S_0 e^{\left(\alpha - \frac{\sigma^2}{2}\right)t + \sigma W_t} \tag{2}$$

  donde: α y $\sigma > 0$ son constantes que representan la tasa de rentabilidad y volatilidad instantáneas del activo respectivamente. W_t es un movimiento

Browniano estándar sobre $(\Omega, \mathcal{F}, \mathbb{P}, (\mathcal{F}_t)_{t\in[0,T]})$ que representa los choques aleatorios que afectan el precio.

- Un derivado ($V(T-t, S_t; \sigma, K) \equiv V(t, S_t) \equiv V_t$) que satisface, por la aplicación del lema de Itô, la EDE:

$$\begin{aligned} dV_t &= \frac{\partial V_t}{\partial t}dt + \frac{\partial V_t}{\partial S_t}dS_t + \frac{1}{2}\frac{\partial^2 V_t}{\partial S_t^2}dS_t^2 \\ &= \frac{\partial V_t}{\partial t}dt + \frac{\partial V_t}{\partial S_t}(\alpha S_t dt + \sigma S_t dW_t) + \frac{1}{2}\frac{\partial^2 V_t}{\partial S_t^2}\sigma^2 S_t^2 dt \\ &= \left(\frac{\partial V_t}{\partial t} + \alpha S_t \frac{\partial V_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2}\right) dt + \sigma S_t \frac{\partial V_t}{\partial S_t} dW_t \end{aligned} \tag{3}$$

El objetivo es determinar el valor del derivado en el instante t, para lo cual se considera un portafolio de replicación con valor en t denotado por Π_t, compuesto por Δ_t unidades del activo riesgoso y Ψ_t unidades monetarias invertidas o prestadas a la tasa libre de riesgo r, es decir:

$$\Pi_t = \Delta_t S_t + \Psi_t B_t \tag{4}$$

Bajo la condición de autofinanciamiento[1] se tiene que:

$$\begin{aligned} d\Pi_t &= \Delta_t dS_t + \Psi_t dB_t \\ &= \Delta_t(\alpha S_t dt + \sigma S_t dW_t) + \Psi_t(r B_t dt) \\ &= (\alpha \Delta_t S_t + r \Psi_t B_t)\, dt + \Delta_t \sigma S_t dW_t \end{aligned} \tag{5}$$

Se está considerando, entonces, una estrategia de negociación compuesta por:

- El derivado (V_t).
- Posición en el activo riesgoso (subyacente del derivado) (Δ_t).
- Posición en el activo libre de riesgo (Ψ_t).

[1] Esta condición establece que los cambios en el valor del portafolio se deben solamente a los cambios en los precios de los activos que lo conforman y no a la entrada o salida de capital.

Entonces, el cambio en el valor de la estrategia es:

$$
\begin{aligned}
&dV_t + d\Pi_t \\
&= \left(\frac{\partial V_t}{\partial t} + \alpha S_t \frac{\partial V_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2}\right) dt + \sigma S_t \frac{\partial V_t}{\partial S_t} dW_t + \left(\alpha \Delta_t S_t + \Psi_t r B_t\right) dt + \Delta_t \sigma S_t dW_t \\
&= \left(\frac{\partial V_t}{\partial t} + \alpha S_t \frac{\partial V_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} + \alpha \Delta_t S_t + \Psi_t r B_t\right) dt + \left(\sigma S_t \frac{\partial V_t}{\partial S_t} + \Delta_t \sigma S_t\right) dW_t
\end{aligned} \tag{6}
$$

A partir de la ecuación (6) se tiene que esta estrategia se puede hacer libre de riesgo, es decir, se puede eliminar el término asociado con dW_t, si $\sigma S_t \left(\frac{\partial V_t}{\partial S_t} + \Delta_t\right) = 0$, de donde $\Delta_t = -\frac{\partial V_t}{\partial S_t}$. Esta estrategia se conoce como cobertura delta.

De lo anterior resulta:

$$
\begin{aligned}
d(V_t + \Pi_t) &= \left(\frac{\partial V_t}{\partial t} + \alpha S_t \frac{\partial V_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} - \alpha \frac{\partial V_t}{\partial S_t} S_t + \Psi_t r B_t\right) dt \\
&= \left(\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} + \Psi_t r B_t\right) dt
\end{aligned} \tag{7}
$$

y

$$
\Pi_t = \Delta_t S_t + \Psi_t B_t = -\frac{\partial V_t}{\partial S_t} S_t + \Psi_t B_t \tag{8}
$$

Por no arbitraje[2] se tiene que:

$$
d(V_t + \Pi_t) = (V_t + \Pi_t) r dt = \left(V_t - \frac{\partial V_t}{\partial S_t} S_t + \Psi_t B_t\right) r dt \tag{9}
$$

entonces:

$$
\begin{aligned}
\left(\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} + \Psi_t r B_t\right) dt &= \left(V_t - \frac{\partial V_t}{\partial S_t} S_t + \Psi_t B_t\right) r dt \\
\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} + \Psi_t r B_t &= r V_t - r S_t \frac{\partial V_t}{\partial S_t} + r \Psi_t B_t
\end{aligned}
$$

[2]Una estrategia de negociación es de arbitraje si: i. no tiene inversión inicial. ii. no tiene riesgo. iii. genera ganancias con probabilidad estrictamente positiva. En un mercado sin arbitraje, si un activo o combinación de activos es libre de riesgo su rentabilidad debe ser igual a la del activo libre de riesgo (r).

de donde se llega a la EDP de Black-Scholes:

$$\boxed{\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} + rS_t \frac{\partial V_t}{\partial S_t} - rV_t = 0} \tag{10}$$

con la condición de frontera $V(T, S_T) = \Phi(S_T)$, donde la función $\Phi(S_T)$ describe los posibles pagos del derivado al vencimiento.

2. Redes neuronales físicamente informadas (PINN)

La flexibilidad de las redes neuronales profundas, como técnica universal para la aproximación de funciones, viene acompañada de la necesidad de calcular una gran cantidad de parámetros en la fase de aprendizaje supervisado, por lo que generalmente requieren de un gran volumen de datos de entrenamiento. Las redes neuronales físicamente informadas (PINN) son una técnica de aprendizaje automático científico que permite resolver EDP en situaciones donde los datos son limitados, es decir, cuando solo se dispone de la EDP y condiciones de frontera, en lugar de una gran cantidad de pares de valores de las variables independientes y dependientes. Las PINN generan soluciones aproximadas de las EDP mediante el entrenamiento de una red neuronal que minimiza una función de pérdida conformada por términos que representan el desajuste de las condiciones iniciales y de contorno a lo largo de la frontera del dominio espacio-temporal, así como el residual de la EDP en puntos seleccionados del interior. Si bien los precursores de este enfoque se remontan a principios de la década de los noventa (ver Lagaris et al., 1998; Lagaris et al., 2000; Lee y Kang, 1990), el término PINN, así como un aumento en la actividad de investigación posterior, fueron iniciados por el informe en dos partes (Raissi et al., 2017a, 2017b), posteriormente publicado en Raissi et al. (2019).

Describimos el enfoque PINN para aproximar la solución $u : [0, T] \times D \to \mathbb{R}$ de una ecuación de evolución:

$$\frac{\partial u}{\partial t}(t, x) + \mathcal{N}[u](t, x) = 0 \quad ; \quad (t, x) \in (0, T] \times D \tag{11}$$

$$u(0, x) = u_0(x) \quad ; \quad x \in D \tag{12}$$

donde: $\mathcal{N}$ es un operador diferencial no lineal que actúa sobre u, $D \in \mathbb{R}^d$ es un dominio acotado, T denota el tiempo final y $u_0 : D \to \mathbb{R}$ son los datos iniciales conocidos. Tenemos que:

$$u(t,x) = u_b(t,x) \quad ; \quad (t,x) \in (0,T] \times \partial D \tag{13}$$

donde: ∂D denota la frontera del dominio D y $u_b : (0,T] \times \partial D \to \mathbb{R}$ son los datos conocidos en la frontera. El método construye una red neuronal de aproximación $u_\theta(t,x) \approx u(t,x)$ de la solución de (11 y 12), donde $u_\theta : [0,T] \times D \to \mathbb{R}$ denota una función realizada por una red neuronal con parámetros θ.

A diferencia de otros métodos de aprendizaje que intentan inferir la solución mediante un enfoque puramente basado en datos, es decir, ajustando una red neuronal a un conjunto de valores $\{(t_i, x_i, u(t_i, x_i))\}_{i=1}^N$, las PINN tienen en cuenta la EDP subyacente (*la física*), y aprovechando las capacidades de diferenciación automática que ofrecen los *software* modernos para las funciones realizadas por las redes neuronales, la solución aproximada u_θ se deriva respecto a las variables de tiempo y espacio, lo que permite evaluar el residuo de la EDP (11) en un conjunto de puntos colocados. De esta manera, la física codificada en la ecuación diferencial está disponible para construir una función de pérdida que establece en qué medida el problema (11)-(12) se satisface con u_θ.

Mientras que el enfoque de otros métodos que utilizan redes neuronales para resolver EDP se centra en mitigar la maldición de la dimensionalidad, la fortaleza de las PINN radica en su flexibilidad, ya que se pueden aplicar a una gran variedad de EDP desafiantes, mientras que las aproximaciones numéricas clásicas suelen requerir adaptación a los detalles de una EDP específica.

La aproximación en tiempo continuo para la EDP parabólica (11) (12) está basada en el residual de la red neuronal $u_\theta : [0,T] \times D \to \mathbb{R}$ de la solución u con respecto a (11), el cual se define como:

$$r_\theta := \frac{\partial u_\theta}{\partial t}(t,x) + \mathcal{N}[u_\theta](t,x) \tag{14}$$

La clase de redes neuronales consideradas son aquellas con alimentación hacia adelante de múltiples capas, también conocidas como perceptrones multicapa. Estas redes son composiciones de funciones lineales afines alternadas $W^l \ \cdot \ + b^l$ y funciones no lineales $\sigma^l(\cdot)$ llamadas funciones de activación, es decir:

$$u_\theta(z) := W^L \sigma^L (W^{L-1} \sigma^{L-1} (\cdots \sigma^1 (W^0 z + b^0) \cdots) + b^{L-1}) + b^L \tag{15}$$

donde: W^l y b^l son matrices de pesos y vectores de sesgo respectivamente, y $z = [t, x]^T$. Esta estructura, altamente no lineal y compuesta de la función de aproximación u_θ, es el núcleo de muchos métodos de aprendizaje automático basados en redes neuronales.

En general, entrenar una red neuronal, es decir, determinar los parámetros θ utilizando métodos de optimización basados en gradientes, como el gradiente estocástico descendente, el optimizador Adam o AdaGrad, requieren la derivada de u_θ respecto a los parámetros desconocidos W^l y b^l. Para incorporar el residual de la EDP (14) en la función de pérdida por minimizar, las PINN requieren una diferenciación adicional para evaluar los operadores diferenciales $\frac{\partial u_\theta}{\partial t}$ y $\mathcal{N}[u_\theta]$. Se tiene entonces que el residual r_θ comparte los mismos parámetros que la red original $u_\theta(t, x)$, pero incorpra la *física* de (11). Ambos tipos de derivadas se pueden obtener fácilmente mediante diferenciación automática con bibliotecas de aprendizaje automático de última generación, como TensorFlow o PyTorch.

En el enfoque PINN, para la solución de la EDP (11-12) se procede a la minimización de la función de pérdida:

$$\phi_\theta := \phi_\theta^r(X^r) + \phi_\theta^0(X^0) + \phi_\theta^b(X^b) \tag{16}$$

donde: X denota la colección de datos de entrenamiento y la función de pérdida ϕ_θ contiene los siguientes términos:

- La media cuadrática de los residuales:

$$\phi_\theta^r(X^r) := \frac{1}{N_r} \sum_{i=1}^{N_r} |r_\theta(t^r, x_i^r)|^2 \tag{17}$$

  en un conjunto de puntos colocados $X^r := \{(t^r, x_i^r)\}_{i=1}^{N_r} \subset (0, T] \times D$.

- La media cuadrática de los desajustes respecto a las condiciones iniciales:

$$\phi_\theta^0(X^0) := \frac{1}{N_0} \sum_{i=1}^{N_0} |u_\theta(t^0, x_i^0) - u_0(x_i^0)|^2 \tag{18}$$

  en un conjunto de puntos $X^0 := \{(t^0, x_i^0)\}_{i=1}^{N_0} \subset \{0\} \times D$.

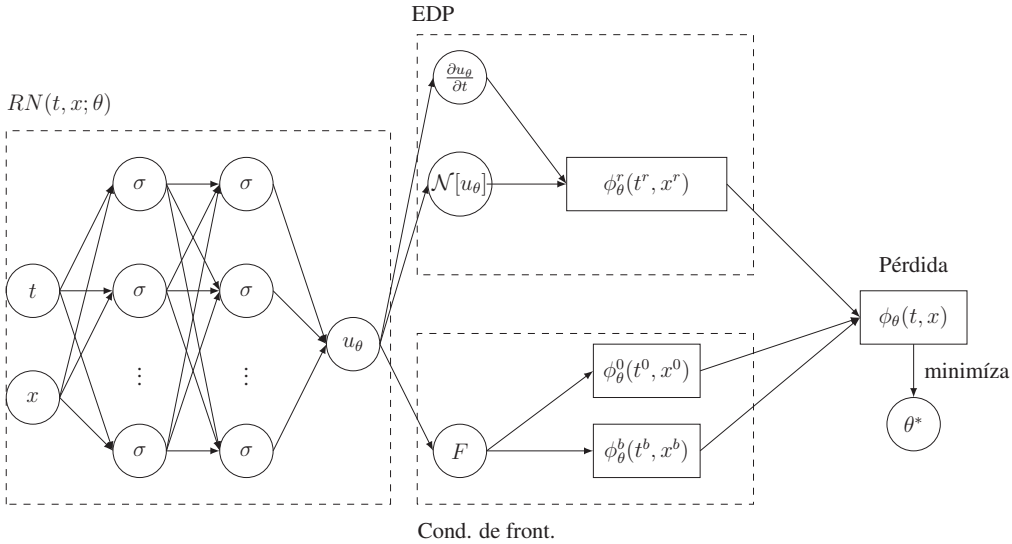
- La media cuadrática de los desajustes respecto a las condiciones de frontera:

$$\phi_\theta^b(X^b) := \frac{1}{N_b}\sum_{i=1}^{N_b} |u_\theta(t^b, x_i^b) - u_b(t_i^b, x_i^b)|^2 \tag{19}$$

en un conjunto de puntos $X^b := \{(t^b, x_i^b)\}_{i=1}^{N_b} \subset (0, T] \times \partial D$.

La figura 1 representa el funcionamiento del esquema PINN.

Figura 1: Arquitectura considerada para la implementación de la técnica PINN en el proceso de resolución de una EDP



3. Resolución de la EDP de Black-Scholes

Consideramos la EDP de valoración de Black-Scholes en el caso específico de opciones europeas de compra[3]:

[3]Una opción europea de compra es un tipo de derivado que otorga a su poseedor el derecho, mas no la obligación, de comprar una determinada cantidad del activo subyacente en una fecha futura específica T y por un valor establecido K.

$$\frac{\partial V_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V_t}{\partial S_t^2} + rS_t \frac{\partial V_t}{\partial S_t} - rV_t = 0 \tag{20}$$

con $V(T, S_T) = \max\{S_T - K; 0\} \equiv (S_T - K)^+$. Trabajamos sobre un dominio acotado de las variables tiempo (t) y precio (S_t), de forma que:

$$t \in [0, T] \tag{21}$$

y

$$S_t \in [0, S^{\max}] \tag{22}$$

donde: S^{max} es un valor constante que denota el máximo precio posible que podría tomar el activo en el intervalo $[0, T]$. Se tienen entonces las siguientes condiciones asociadas a la EDP:

- Si $S_t = 0$ en cualquier instante t:

$$V(t, 0) = 0 \tag{23}$$

  Si en cualquier momento de tiempo el precio del activo subyacente alcanza el valor cero, en ese momento la opción de compra sobre este activo no tendrá ningún valor.

- Si el precio del activo es grande ($S^{\max}$) podemos asegurar que la opción será ejercida por su poseedor, luego $V(T, S_T) = S_T - K$. El valor de esta condición en t implica descontar el precio de ejercicio K, y dado que el precio de no arbitraje del activo en t es simplemente S_t, se tiene la condición de frontera:

$$V(t, S^{\max}) = S^{\max} - Ke^{-r(T-t)} \tag{24}$$

- En el instante de tiempo T el valor de la opción está determinado por el flujo de caja que genera la opción al vencimiento:

$$V(T, S_T) = \max\{S_T - K; 0\} \tag{25}$$

Denotamos por $\hat{u} \equiv \hat{u}(t, S_t; \theta)$ a la función de predicción de nuestra red neuronal para las entradas t, S_t y los parámetros θ. El residual de esta red neuronal es entonces:

$$r_\theta := \frac{\partial \hat{u}}{\partial t} + \mathcal{N}[\hat{u}] \tag{26}$$

donde:

$$\mathcal{N}[\hat{u}] = \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 \hat{u}}{\partial S_t^2} + r S_t \frac{\partial \hat{u}}{\partial S_t} - r\hat{u} \tag{27}$$

Para construir la función de pérdida utilizamos las condiciones (23), (24), (25) y el residual (26), recordando que la pérdida será la suma de las medias cuadráticas de las pérdidas individuales basadas en estas condiciones.

- La media cuadrática de los residuales:

$$\phi_\theta^r(t^r, S^r(t)) := \frac{1}{N_r}\sum_{i=1}^{N_r} |r_\theta(t_i^r, S_i^r(t^r))|^2 \tag{28}$$

  en un conjunto de puntos colocados $\{(t_i^r, S_i^r(t_i^r))\}_{i=1}^{N_r} \subset (0, T] \times [0, S^{\max}]$ aleatoriamente generados.

- La media cuadrática de los desajustes respecto a las condiciones cuando el precio del activo es cero:

$$\phi_\theta^0(t^0, 0) := \frac{1}{N_0}\sum_{i=1}^{N_0} |\hat{u}(t_i^0, 0; \theta)|^2 \tag{29}$$

  en un conjunto de puntos $\{(t_i^0, 0)\}_{i=1}^{N_0} \subset [0, T] \times \{0\}$.

- La media cuadrática de los desajustes respecto a las condiciones en la fecha de expiración del derivado:

$$\phi_\theta^b(T, S^b(T)) := \frac{1}{N_b}\sum_{i=1}^{N_b} |\hat{u}(T, S_i^b(T); \theta) - (S_i^b(T) - K)^+|^2 \tag{30}$$

  en un conjunto de puntos $\{(T, S_i^b(T))\}_{i=1}^{N_b} \subset \{T\} \times (0, S^{\max})$ aleatoriamente generados del precio del activo en la fecha de expiración.

- La media cuadrática de los desajustes respecto a las condiciones cuando el precio del activo alcanza el valor máximo considerado:

$$\phi_\theta^m(t^m, S^{\max}) := \frac{1}{N_m}\sum_{i=1}^{N_m}\left|\hat{u}(t_i^m, S^{\max};\theta) - \left(S^{\max} - Ke^{-r(T-t_i^m)}\right)\right|^2 \tag{31}$$

en un conjunto de puntos $\{(t_i^m, S^{\max})\}_{i=1}^{N_m} \subset (0,T]\times\{S_{\max}\}$ aleatoriamente generados.

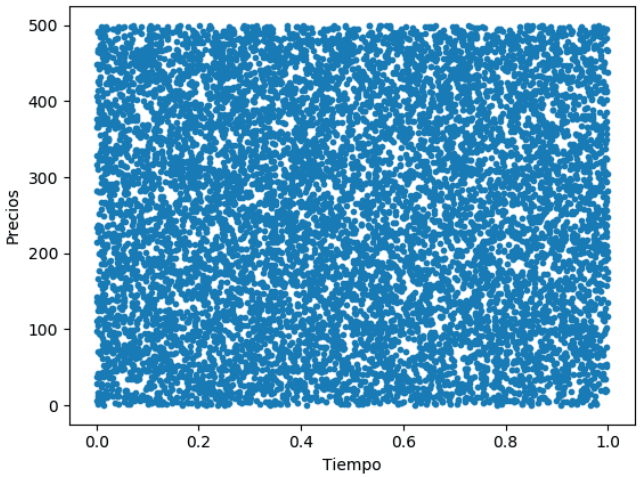
Con lo cual la función de pérdida es:

$$\phi_\theta(t, S(t)) = \phi_\theta^r(t^r, S^r(t)) + \phi_\theta^0(t^0, 0) + \phi_\theta^b(T, S^b(T)) + \phi_\theta^m(t^m, S^{\max}) \tag{32}$$

3.1. Generación de datos de entrenamiento

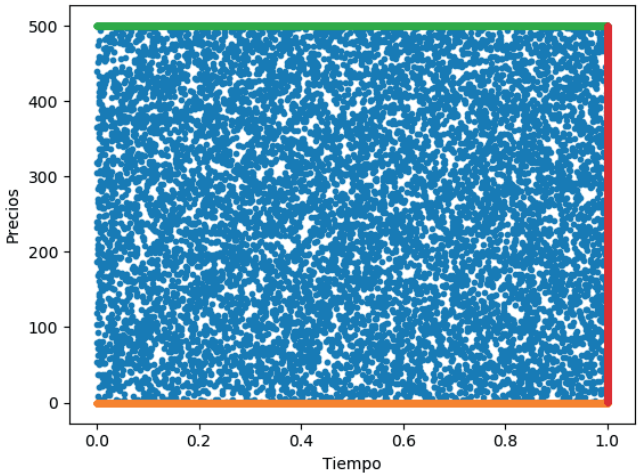
El siguiente paso es la creación del conjunto de datos de entrenamiento de la red neuronal, es decir, la construcción del conjunto de puntos en los cuales se evaluará cada una de las funciones de pérdida. Comenzamos generando una muestra aleatoria de puntos colocados al interior del dominio tiempo-precio. Para el precio se generan 10000 valores aleatorios de una distribución uniforme sobre el intervalo $[0, 500]$ y la misma cantidad para el tiempo sobre el intervalo $[0, 1]$. La figura 2 representa el conjunto de puntos aleatoriamente generados.

Figura 2: Conjunto de 10000 puntos generados en el interior del dominio tiempo-precio



Adicionamos ahora el conjunto de 10000 puntos generados por cada una de las condiciones de frontera de la EDP.

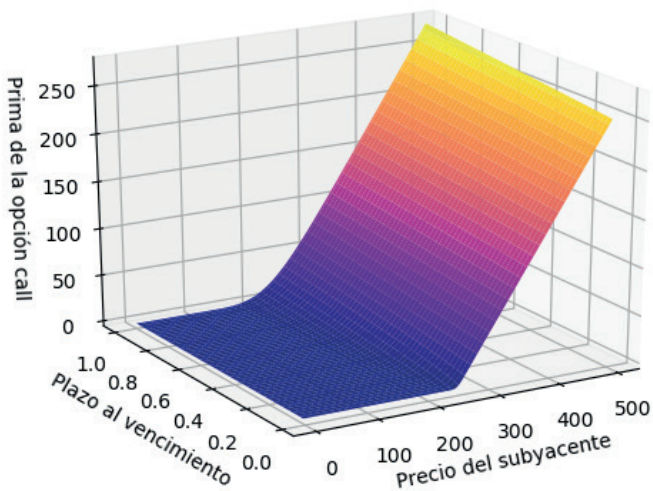
Figura 3: Conjunto total de puntos de entrenamiento generados



3.2. Implementación de PINN en Python

En el anexo se encuentra el código en Python que implementa esta técnica sobre el conjunto datos de entrenamiento. La figura 4 muestra el valor de la prima de una opción calculada por la red neural de aproximación propuesta.

Figura 4. Primas calculadas por la red neuronal de aproximación (PINN) para $S_t \in [0, 500]$, $T \in [0, 1]$, $K = 250$, $r_f = 0,1$ y $\sigma = 0,2$



Se puede observar que las primas encontradas coinciden con los valores que se tienen si se utiliza la fórmula de Black-Scholes para este tipo de opciones, la cual establece que el valor de una opción *call* en t (C_t), para una opción con precio de ejercicio K y vencimiento en T, pactada sobre un activo con valor en t denotado por S_t y volatilidad σ y tasa libre de riesgo en el mercado r_f, está dada por:

$$C_t = S_t N(d_1) + K e^{r_f(T-t)} N(d_2) \tag{33}$$

donde:

$$d_1 = \frac{\ln\left(\frac{S_t}{K}\right) + \left(r_f + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} \quad ; \quad d_2 = d_1 - \sigma\sqrt{T-t} \tag{34}$$

y $N(x)$ denota la probabilidad acumulada de una normal estándar hasta el valor x.

4. Conclusiones y extensiones

Se cumplen 50 años de la publicación del magistral trabajo de Fisher Balck y Myron Scholes sobre la valoración de derivados financieros, y hoy más que nunca es ampliamente reconocida su importancia en la teoría y práctica financiera Sus postulados, desarrollos y resultados marcaron para siempre la historia de las finanzas modernas, y han sido, y seguramente seguirán siendo, fuente de inspiración y punto de partida esencial para el desarrollo de las finanzas cuantitativas, particularmente en lo relacionado con la gestión del riesgo. Como se mencionó en la introducción, este documento tiene como uno de sus objetivos rendir un homenaje a este excepcional desarrollo.

Sobre la implementación de PINN para la resolución de la EDP de Black-Scholes es importante destacar que, aunque no es un procedimiento analítico, sus resultados son muy buenos comparados con la fórmula de Black-Scholes para el caso de opciones europeas, y se constituye en una importante herramienta al momento de considerar extensiones del problema de valoración a contextos en los que no necesariamente se cumplen todos los supuestos, lo que lleva, en la mayoría de los caos, a EDP de valoración semilineales o completamente no lineales, como en los casos de múltiples activos subyacentes, mercados ilíquidos, costos de transacción, volatilidades estocásticas, entre otros.

Podemos concluir, entonces, que las técnicas de aprendizaje científico de máquina para la resolución de EDP se configuran como una alternativa poderosa para el tratamiento de problemas de valoración en el contexto de modelos de mercado más miméticos de la realidad. Que sea esta la oportunidad de invitar al lector a profundizar sobre este tipo de herramientas.

Referencias

Bellman, R. (1966). Dynamic programming. *Science*, *153*(3731), 34-37. https://doi.org/10.1126/science.153.3731.34

Black, F., y Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637-654. https://doi.org/10.1086/260062

Lagaris, I. E., Likas, A., y Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, *9*(5), 987-1000. https://doi.org/10.1109/72.712178

Lagaris, I. E., Likas, A. C., y Papageorgiou, D. G. (2000). Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, *11*(5), 1041-1049. https://doi.org/110.1109/72.870037

Lee, H., y Kang, I. S. (1990). Neural algorithm for solving differential equations. *Journal of Computational Physics*, *91*(1), 110-131. https://doi.org/10.1016/0021-9991(90)90007-N

Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, 141-183.

Raissi, M., Perdikaris, P., y Karniadakis, G. E. (2017a). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.

Raissi, M., Perdikaris, P., y Karniadakis, G. E. (2017b). Physics informed deep learning (part ii): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*.

Raissi, M., Perdikaris, P., y Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686-707. https://doi.org/10.1016/j.jcp.2018.10.045

Anexo

Código en Python para la resolución de la EDP de Balck-Scholes por PINN.

```
import tensorflow as tf
import numpy as np

# Tipo de datos
DTYPE='float32'
tf.keras.backend.set_floatx(DTYPE)

r=0.1
T=1
K=10
sigma=0.2

# Constantes
r = tf.constant(r, dtype=DTYPE)
K = tf.constant(K, dtype=DTYPE)
T = tf.constant(T, dtype=DTYPE)
sigma = tf.constant(sigma, dtype=DTYPE)

# Condición de precio cero
def fun_v_0(t, x):
n = x.shape[0]
return tf.zeros((n,1), dtype=DTYPE)

# Condición de precio máximo
def fun_v_max(t,x):
return x-K*tf.exp(-r*(T-t))

# Condición de frontera al vencimiento
def fun_v_ven(t,x):
return tf.math.maximum(x-K, 0)


# Residual de la EDP
def fun_r(t, x, u, u_t, u_x, u_xx):
```

```
return u_t + 0.5*sigma*(x**2)*u_xx+r*x*u_x+r*u


# Numero de puntos generados
N_0 = 50
N_max = 50
N_ven = 50
N_r = 10000

# Conjunto de condiciones
tmin = 0
tmax = 1
xmin = 0
xmax = 200

# Cotas inferiores
lb = tf.constant([tmin, xmin], dtype=DTYPE)
# Cotas superiores
ub = tf.constant([tmax, xmax], dtype=DTYPE)


# Puntos generados para las condiciones inciales
t_0 = tf.ones((N_0,1), dtype=DTYPE)*lb[0]
x_0 = tf.random.uniform((N_0,1), lb[1], ub[1], dtype=DTYPE)
X_0 = tf.concat([t_0, x_0], axis=1)


# Evaluación de las condicines

u_0 = fun_v_0(t_0,x_0)

t_b = tf.random.uniform((N_max,1), lb[0], ub[0], dtype=DTYPE)
x_b = lb[1] + (ub[1] - lb[1]) * tf.keras.backend.random_bernoulli((N_m
X_b = tf.concat([t_b, x_b], axis=1)

u_b = fun_v_ven(t_b, x_b)
```

```
# Puntos generados del interior
t_r = tf.random.uniform((N_r,1), lb[0], ub[0], dtype=DTYPE)
x_r = tf.random.uniform((N_r,1), lb[1], ub[1], dtype=DTYPE)
X_r = tf.concat([t_r, x_r], axis=1)

X_data = [X_0, X_b]
u_data = [u_0, u_b]

#Modelo PINN
def init_model(num_hidden_layers=8, num_neurons_per_layer=20):

model = tf.keras.Sequential()


model.add(tf.keras.Input(2))


scaling_layer = tf.keras.layers.Lambda(
lambda x: 2.0*(x - lb)/(ub - lb) - 1.0)
model.add(scaling_layer)


for _ in range(num_hidden_layers):
model.add(tf.keras.layers.Dense(num_neurons_per_layer,
activation=tf.keras.activations.get('tanh'),
kernel_initializer='glorot_normal'))


model.add(tf.keras.layers.Dense(1))

return model

#Evaluacion del gradiente

def get_r(model, X_r):

with tf.GradientTape(persistent=True) as tape:
```

```
t, x = X_r[:, 0:1], X_r[:,1:2]

tape.watch(t)
tape.watch(x)

# Residual
u = model(tf.stack([t[:,0], x[:,0]], axis=1))

u_x = tape.gradient(u, x)

u_t = tape.gradient(u, t)
u_xx = tape.gradient(u_x, x)

del tape

return fun_r(t, x, u, u_t, u_x, u_xx)

def compute_loss(model, X_r, X_data, u_data):

# Funcion de pérdida
r = get_r(model, X_r)
phi_r = tf.reduce_mean(tf.square(r))

loss = phi_r

for i in range(len(X_data)):
u_pred = model(X_data[i])
loss += tf.reduce_mean(tf.square(u_data[i] - u_pred))

return loss

def get_grad(model, X_r, X_data, u_data):

with tf.GradientTape(persistent=True) as tape:
tape.watch(model.trainable_variables)
loss = compute_loss(model, X_r, X_data, u_data)
```

```
g = tape.gradient(loss, model.trainable_variables)
del tape

return loss, g

#Selección de los valores utilizado para la tasa de aprendizaje.

model = init_model()

lr = tf.keras.optimizers.schedules.PiecewiseConstantDecay([1000,3000],

# Optimizador
optim = tf.keras.optimizers.Adam(learning_rate=lr)


#Entrenamiento

@tf.function
def train_step():
loss, grad_theta = get_grad(model, X_r, X_data, u_data)

optim.apply_gradients(zip(grad_theta, model.trainable_variables))

return loss

N = 1000
hist = []

for i in range(N+1):

loss = train_step()

hist.append(loss.numpy())
```