Neural PDE Methods for Financial Free-Boundary Problems

From Optimal Stopping to Neural PDE Optimization

Chinonso Nwankwo[1] **Nneka Umeorah**[2] Tony Ware[1] Weizhong Dai[3]

[1] University of Calgary, Canada
[2] Cardiff University, UK
[2] Louisana Tech University, USA

Mathematical Optimization and Control for Industrial and Societal Challenges in Sub-Saharan Africa

7–10 September, 2026
Pretoria, South Africa



MARKETS DON'T WAIT.

When should we **act?**

When should we **wait?**

When is it **no longer optimal to wait?**

OPTIMAL STOPPING

The art and science of timing under uncertainty.



American Options: The Right to Choose *When*

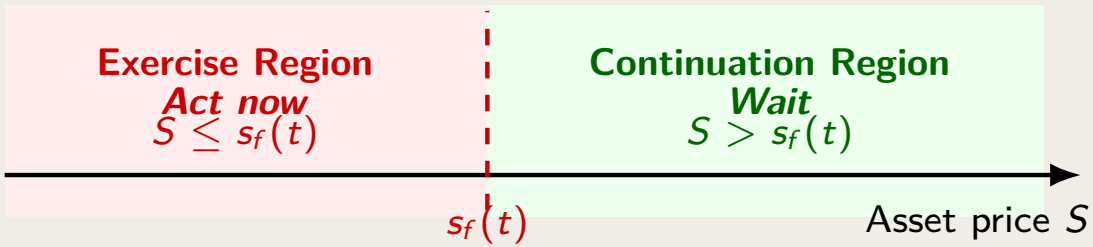
An American option gives the holder the flexibility to buy or sell an asset at a predetermined price, **at any time before or at maturity.**

Optimal stopping formulation

$$V(t,S) = \sup_{\tau \in [t,T]} \mathbb{E}^{\mathbb{Q}} \left[e^{-r(\tau - t)} (K - S_\tau)^+ \,\middle|\, S_t = S \right]$$

Objective: Determine the optimal exercise strategy that maximises the expected discounted payoff.

Decision at a given time t



From Optimal Stopping to a Free-Boundary PDE

Under the risk-neutral measure $\mathbb{Q}$, let the underlying asset follow

$$dS_u = (r - \delta)S_u\, du + \sigma S_u\, dW_u^{\mathbb{Q}}.$$

Optimal stopping divides the domain

$$\begin{cases} V(t,S) = K - S, & S \le s_f(t), \quad \text{(exercise region)}, \\ V_t + \frac{1}{2}\sigma^2 S^2 V_{SS} + (r-\delta)SV_S - rV = 0, & S > s_f(t), \quad \text{(continuation region)}. \end{cases}$$

Conditions at the unknown boundary $S = s_f(t)$

$$\boxed{V(t, s_f(t)) = K - s_f(t)} \qquad \boxed{V_S(t, s_f(t)) = -1} \qquad \boxed{V(T,S) = (K-S)^+}$$

Value matching *Smooth pasting* *Terminal condition*

Why is this a free-boundary problem? Both the option value $V(t,S)$ and the exercise boundary $s_f(t)$ are unknown and must be determined **simultaneously**.

Why Is the Problem Computationally Difficult?

Unknown solution

We seek the option value

$$V(t, S),$$

satisfying the PDE in the continuation region.

Unknown boundary

The exercise boundary

$$s_f(t)$$

is not known in advance and evolves with time.

$$\boxed{V(t, S) \iff s_f(t)}$$

The option value and exercise boundary must be determined **simultaneously**.

- The computational domain depends on the unknown boundary $s_f(t)$.
- The PDE is coupled to $s_f(t)$ through value matching and smooth pasting.

Transform the moving-boundary problem onto a **fixed computational domain**.

Fixing the Moving Boundary: Landau Transformation

Step 1: Transform the moving spatial domain into a fixed domain.

Logarithmic Landau transformation

$$x = \ln S - \ln s_f(t), \qquad V(t,S) = U(t,x)\,.$$

The unknown exercise boundary is now mapped to a fixed location:

$$\boxed{S = s_f(t) \quad \Longrightarrow \quad x = 0.}$$

The transformed PDE becomes

$$U_t - \frac{\sigma^2}{2} U_{xx} - \left(r - \delta + \frac{s_f'(t)}{s_f(t)} - \frac{\sigma^2}{2} \right) U_x + rU = 0, \qquad x > 0.$$

At the fixed boundary $x = 0$,

$$U(t,0) = K - s_f(t), \qquad U_x(t,0) = -s_f(t).$$

The moving boundary has been **fixed geometrically** ($(t,x) \in ([0,T] \times [0,\infty))$), but $s_f(t)$ remains an **unknown function** coupled to the PDE.

Normalising the Computational Domain

Step 2: Map the fixed physical domain to a unit square.

After truncating the far-field boundary at $x = x_{\max}$,

$$(t, x) \in [0, T] \times [0, x_{\max}].$$

Domain transformation

$$\tau = \frac{t}{T}, \qquad y = \frac{x}{x_{\max}}, \qquad P(\tau, y) = \frac{1}{x_{\max}} U(t, x).$$

Hence,

$$\boxed{[0, T] \times [0, x_{\max}] \quad \longrightarrow \quad [0, 1] \times [0, 1]}$$

The normalised fixed-boundary PDE becomes

$$\frac{1}{T}\frac{\partial P}{\partial \tau} - \frac{\sigma^2}{2x_{\max}^2}\frac{\partial^2 P}{\partial y^2} - \frac{1}{x_{\max}}\left(r - \delta - \frac{\sigma^2}{2}\right)\frac{\partial P}{\partial y} - \frac{1}{x_{\max} T}\frac{s_f'(\tau)}{s_f(\tau)}\frac{\partial P}{\partial y} + rP = 0, \qquad y > 0.$$

The exercise boundary is mapped to the fixed computational location

$$S = s_f(t) \quad \Longrightarrow \quad x = 0 \quad \Longrightarrow \quad y = 0.$$

Constructing the Auxiliary Function

Step 3: Build the boundary conditions directly into the neural approximation.

Auxiliary-function idea

Following the neural trial-function approach of Lagaris et al. (1998), write the approximation as

$$\boxed{P(\tau, y; \nu) = A(\tau, y) + B(\tau, y)\, N(\tau, y; \nu)}$$

where $N(\tau, y; \nu)$ is a feedforward neural network.

Fixed part: $A(\tau, y)$

Contains **no trainable parameters**

Constructed to satisfy the prescribed initial and boundary information.

Neural part: $B(\tau, y)N(\tau, y; \nu)$

Contains the **trainable parameters** ν

Learns the remaining structure required to satisfy the governing PDE.

Boundary-Aware Neural Representation

Step 3: Construct an auxiliary function that embeds both conditions.

Neural auxiliary function

$$P(\tau, y; \nu) = e^{-\frac{y^2\gamma}{2\tau}}\,\mathcal{M}(\tau, y; \nu), \qquad (\tau, y) \in ((0,1],[0,1]), \quad \text{where}$$

$$\mathcal{M}(\tau, y; \nu) = \sum_{i=0}^{2} \frac{f_i(\tau;\nu)y^i}{i!} + y x_{\max} N(\tau, y; \nu), \qquad \gamma = \frac{r\phi}{\sigma^2}, \phi \in \mathbb{R}_+ .$$

Boundary terms

$$f_0(\tau;\nu) = \frac{K - \bar{s}_f(\tau;\nu)}{x_{\max}},$$

$$f_1(\tau;\nu) = -K.$$

$m = 2$, the highest-order derivative.

Second-order term

$$f_2(\tau;\nu) = x_{\max}\left[\left(\frac{2rK}{\sigma^2} - \left(\frac{2\delta}{\sigma^2} + 1\right)\bar{s}_f\right) + \frac{\gamma}{\tau x_{\max}^2}(K - \bar{s}_f) - 2N_y(\tau, 0; \nu)\right].$$

Key coupling : $\boxed{\bar{s}_f(\tau;\nu) = K - x_{\max} N_y(\tau, 0; \nu)}$

Learning the Free Boundary

Step 4: Recover the unknown exercise boundary from the trained neural representation.

Under the front-fixing transformation, the exercise boundary is located at $y = 0$.

The boundary conditions embedded in the auxiliary function yield

Free boundary induced by the neural network

$$\boxed{\bar{s}_f(\tau;\nu) = K - x_{\max}\frac{\partial N}{\partial y}(\tau, 0;\nu)}$$

Hence, the same trainable parameters ν determine both

$$\boxed{\nu} \quad \Longrightarrow \quad \begin{cases} P(\tau, y;\nu), & \text{transformed option solution,} \\ \bar{s}_f(\tau;\nu), & \text{exercise boundary.} \end{cases}$$

From the PDE to an Optimization Problem

Step 5: Determine the network parameters by minimising the PDE residual at collocation points, with the conditions embedded in the trial solution.

PDE residual

$$\mathcal{R}(\tau, y; \nu) = \frac{1}{T} P_\tau - \frac{\sigma^2}{2x_{\max}^2} P_{yy} - \frac{1}{x_{\max}} \psi(\tau; \nu) P_y + rP,$$

where

$$\psi(\tau; \nu) = r - \delta - \frac{\sigma^2}{2} + \frac{1}{T \bar{s}_f(\tau; \nu)} \frac{d\bar{s}_f(\tau; \nu)}{d\tau}.$$

At collocation points $\{(\tau_i, y_j)\}$, minimise the mean-squared PDE residual:

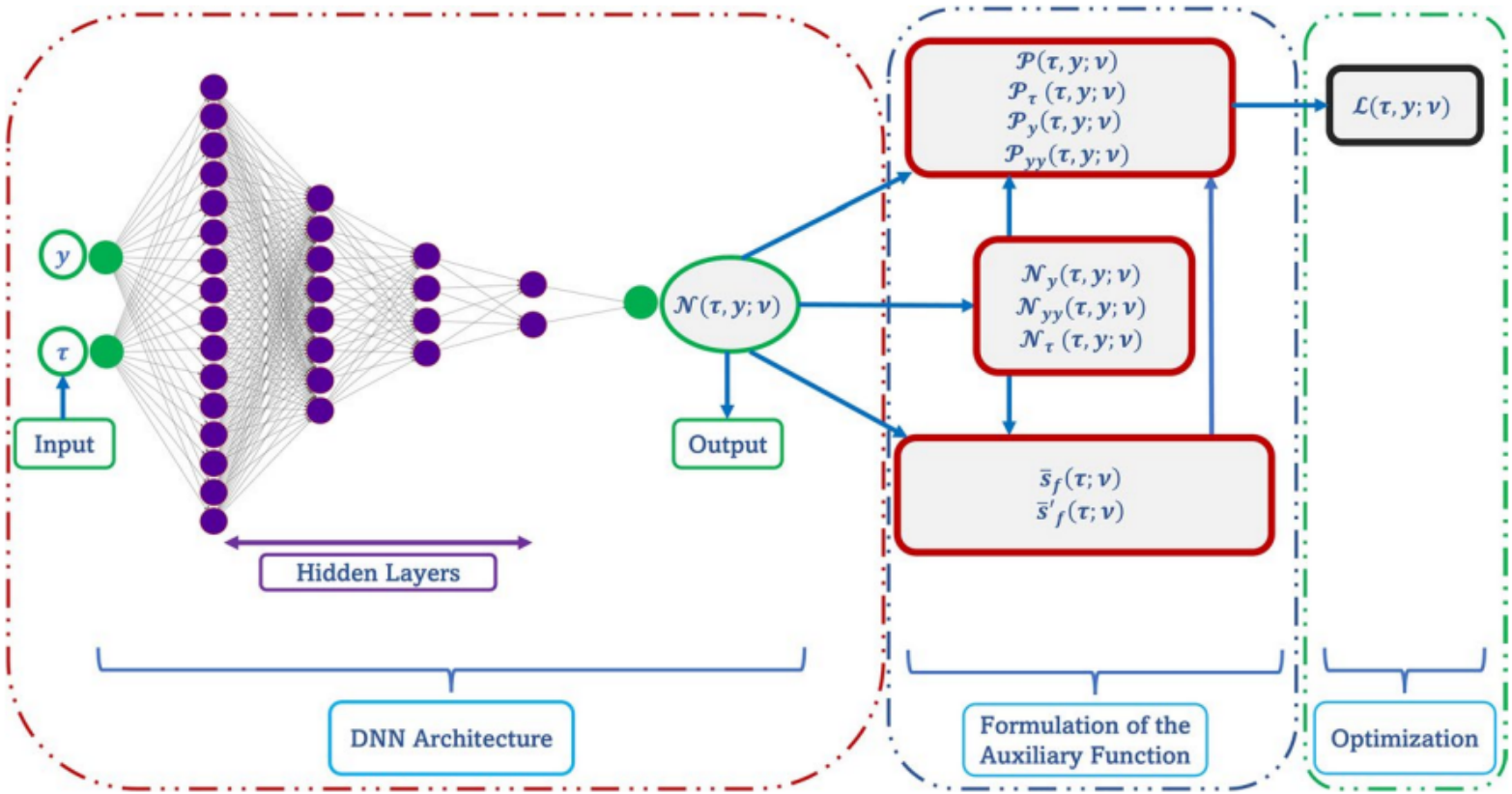
$$\mathcal{L}(\nu) = \frac{1}{N_\tau N_y} \sum_{i=1}^{N_\tau} \sum_{j=1}^{N_y} \left| \mathcal{R}(\tau_i, y_j; \nu) \right|^2.$$

Neural PDE optimization

$$\boxed{\nu^\star = \arg\min_{\nu} \mathcal{L}(\nu)}$$

DNN Architecture

The DNN couples the option solution and the unknown exercise boundary within a single optimization framework.



Where Should We Enforce the PDE?

Step 7: Select collocation points on the computational domain $(\tau, y) \in [0,1]^2$.

The residual loss is evaluated only at the selected points:

$$\mathcal{L}(\nu) = \frac{1}{N} \sum_{(\tau_i, y_i) \in \mathcal{C}} \left| \mathcal{R}(\tau_i, y_i; \nu) \right|^2,$$

where $\mathcal{C}$ denotes the set of collocation points.

Uniform

Evenly distributed points across the domain.

Stretched

Higher concentration near the free boundary.

Random

Points sampled across the computational domain.

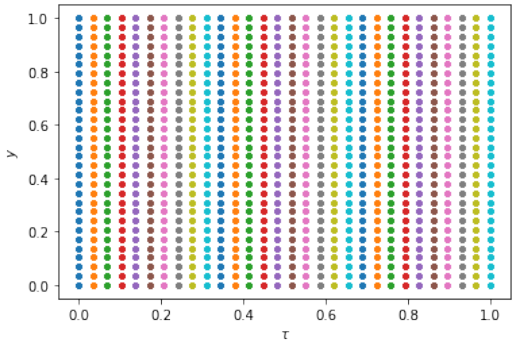
For a stretched spatial grid, for example,

$$y_j = \left(\frac{j}{N_y} \right)^p, \qquad p > 1,$$

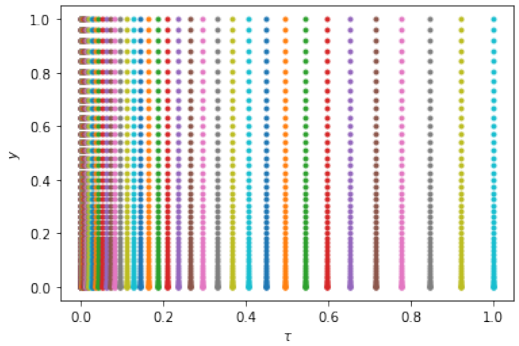
which concentrates points near $y = 0$.

Collocation Strategies

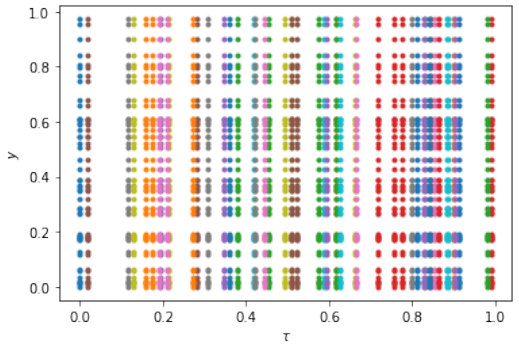
Different sampling schemes were considered on $(\tau, y) \in [0, 1]^2$.



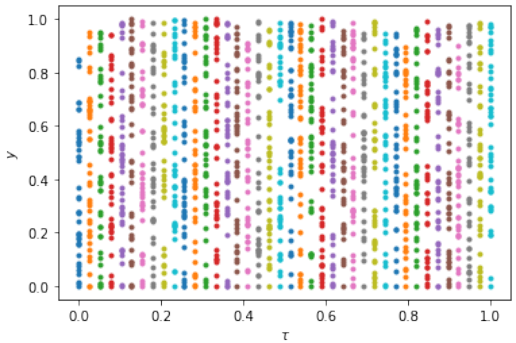
(a) Uniform (y & τ)



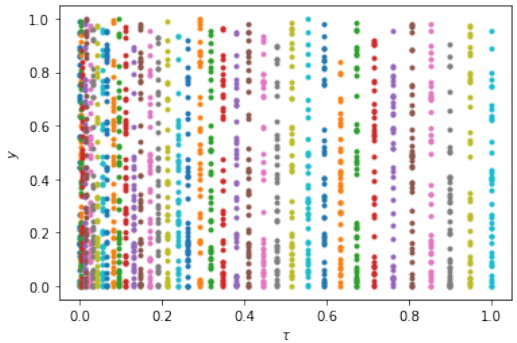
(b) Uniformly stretched (y & τ)



(c) Randomly structured (y & τ)



(d) Random sampling (random in y & uniform in τ)



(e) Random stretched (random in y & uniform in τ)

Numerical Setup

Option test cases

	K	T	r	σ
STM	100	0.5	0.05	0.20
MTM	100	1.0	0.10	0.30
LTM	100	3.0	0.08	0.20

Neural solver

Architecture	2–256–128–64–32–1
Activation	Sigmoid
Optimizer	Adam
$x_{\max}$	6

Reference solution

Sixth-order compact finite-difference scheme

$$h = 0.005$$

Used as a high-accuracy numerical benchmark for the learned exercise boundary.

Evaluation

Can the neural method recover $s_f(\tau)$ accurately across different maturities?

Effect of Collocation Strategy

How does collocation-point placement affect free-boundary recovery?

Effect of sampling strategy: 100×100 grid

Case	Uniform	Randomly structured	Uniformly stretched ($p = 2.0$)	6th-order scheme
STM	84.27	84.52	84.02	83.92
MTM	76.19	76.23	**76.16**	76.16
LTM	81.84	**81.78**	**81.78**	81.78

Effect of stretching parameter: 60×60 grid

Case	$p = 1.2$	$p = 1.5$	$p = 1.7$
STM	84.20	84.04	**83.91**
MTM	76.26	76.17	**76.16**
LTM	81.88	81.80	**81.78**

NB: Collocation design matters: concentrating points near the free boundary generally improves boundary recovery, but the degree of stretching must be chosen appropriately.

Validation Against Numerical Benchmarks

How well does the neural method recover the free boundary?

For the benchmark problem

$$K = 1, \qquad T = 1, \qquad r = 0.1, \qquad \sigma = \sqrt{0.2}, \qquad \delta = 0,$$

we compare the estimated exercise boundary at maturity.

Free-boundary comparison

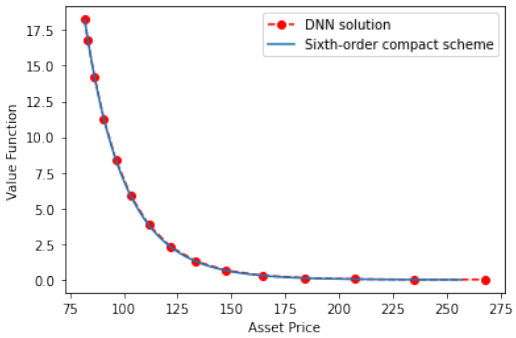
Method	$s_f(T)$
Fazio (2015)	0.8628
Egorova et al. (2016)	0.8627
6th-order compact scheme	0.8628
Neural PDE method	**0.8628**

Main result

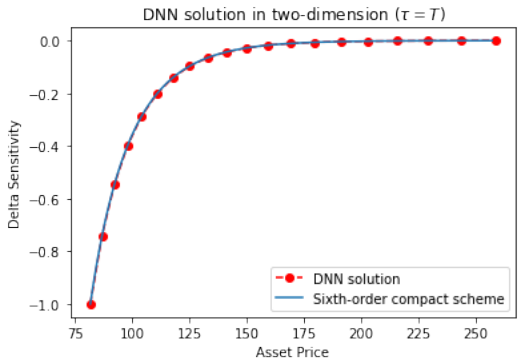
The neural PDE method recovers the exercise boundary to **four decimal places** relative to the high-order numerical benchmark.

Beyond the Free Boundary: Value and Greeks

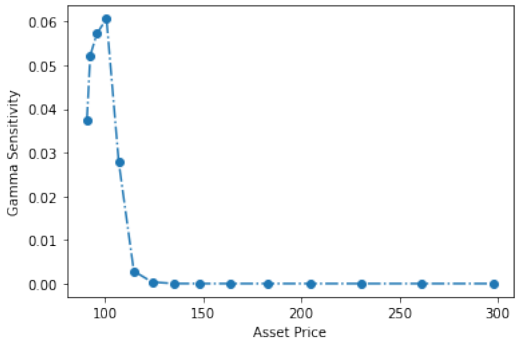
The trained neural solution provides both prices and sensitivities.



Option Value



Delta



Gamma

From one trained neural representation

$$P(\tau, y; \nu^\star) \quad \Longrightarrow \quad \begin{cases} V(t,S), & \text{option value,} \\ \Delta = V_S, & \text{first-order sensitivity,} \\ \Gamma = V_{SS}, & \text{second-order sensitivity.} \end{cases}$$

The same differentiable neural representation provides the **value function, free boundary, and Greeks**.

Conclusion

- Free-boundary formulation: American option pricing combines an **optimal-stopping problem** with an unknown exercise boundary $s_f(t)$.
- Neural PDE solution: Front-fixing and a boundary-aware DNN transform the problem into

$$\boxed{\nu^\star = \arg\min_{\nu} \mathcal{L}(\nu)},$$

  allowing the **option solution and free boundary** to be learned simultaneously.
- Numerical insight: **Collocation design matters:** refinement near the free boundary can improve boundary recovery.

Optimal stopping $\longrightarrow$ Free-boundary PDE $\longrightarrow$ Neural optimization

Selected References

- Nielsen, B. F., Skavhaug, O., & Tveito, A. (2002). Penalty and front-fixing methods for the numerical solution of American option problems. *Journal of Computational Finance*, 5(4), 69–98.
- Nwankwo, C., & Dai, W. (2022). On the efficiency of 5(4) RK-embedded pairs with high order compact scheme and Robin boundary condition for options valuation. *Japan Journal of Industrial and Applied Mathematics*, 39(2), 753–775.
- Fazio, R., Insana, A., & Jannelli, A. (2021). A front-fixing implicit finite difference method for the American put options model. *Mathematical and Computational Applications*, 26(2), 30.
- Egorova, V. N., Tan, S.-H., Lai, C.-H., Company, R., & Jodar, L. (2017). Moving boundary transformation for American call options with transaction cost: Finite difference methods and computing. *International Journal of Computer Mathematics*, 94(2), 345–362.
- Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5), 987–1000.
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear PDEs. *Journal of Computational Physics*, 378, 686–707.

Thank You

Questions & Discussion