Deep Learning for Solving and Estimating Dynamic Macro-Finance Models

Benjamin Fan[1,†], Edward Qiao[2,†], Anran Jiao[3], Zhouzhou Gu[4], Wenhao Li[4], and Lu Lu[3,*]

[1]University High School, Irvine, CA
[2]The Bishop's School, La Jolla, CA
[3]Department of Chemical and Biomolecular Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA
[4]Marshall School of Business, University of Southern California, Los Angeles, CA
[†]These authors contributed equally to this work.
[*]Corresponding author. Email: lulu1@seas.upenn.edu

Abstract

Deep learning has been shown to be an effective method for solving partial differential equations (PDEs) by embedding the PDE residual into the neural network loss function. In this paper, we design a methodology that utilizes deep learning to simultaneously solve and estimate canonical continuous-time general equilibrium models in financial economics, including (1) industrial dynamics of firms and (2) macroeconomic models with financial frictions. Through these applications, we illustrate the advantages of our method:

Keywords: Dynamic macro-finance models; Industrial dynamics of firms; Macroeconomic models with financial frictions; Partial differential equations; Deep learning; Parameter estimation

1 Introduction

Dynamic equilibrium models are the cornerstones of the fast-growing macro-finance literature that tries to understand how financial frictions and asset prices influence economic dynamics, in addition to addressing important policy questions including the design and impact of financial regulation, industrial policy, and monetary policy [1, 2, 3, 4, 5, 6, 7, 8, 9]. These models feature high degrees of nonlinearity originating from either agents' financial constraints or preferences, which make the linearization methods widely used in the macro literature infeasible.

The literature has thus far mostly focused on highly tractable models with a small number of state variables (typically one or two). Furthermore, since solving these models is quite time-consuming, model parameters are usually picked by calibration, which involves intensive model evaluation. Matching moments involves solving the model, simulating the model for a long period and calculating the moment value, and repeating the same procedure for a large number of parameter combinations. Although simulated methods of moment have been applied to corporate-finance models [10, 11, 12, 13], those more explicit and standard methods applied to dynamic equilibrium models are restricted by the curse of dimensionality. Additionally, taking expectations is typical in dynamic problems, but it incurs a significant computational burden. Finally, for different problems, researchers typically need to tailor their numerical methods, which limits the accessibility of the literature, and these methods do not automatically take advantage of the rapidly evolving computational tools.

In this paper, we apply deep-learning methods for solving partial differential equations [14] to economic settings, simultaneously solving and estimating model parameters and allowing for heterogeneous agents.

Our methods have the following advantages: (1) the model is solved globally; (2) the method allows for higher dimensionality; (3) deep learning is a proven and effective method in solving PDEs; (4) estimation and solution are in one step; (5) differentiation is handled analytically which increases numerical accuracy; (6) the underlying package automatically applies the state-of-art machine-learning algorithm, so the method keeps improving itself; (7) the method is versatile and can be applied to a vast variety of problems.

Machine learning (ML) models have already been used in economics and finance, but mainly for the purpose of better predicting economic and financial outcomes in markets such as stocks, insurance, corporate bankruptcy, cryptocurrency, etc. Recent papers, including [15, 16, 17], use deep learning methods to solve economic models. Our paper contributes to this growing literature by combining equilibrium-model estimation with solution and building on the rapidly-improving deep-learning-based solution methods for PDEs in physics and science.

The machine-learning method in [14] is a type of physics-informed neural networks (PINNs). PINNs work by embedding PDE residual into the loss function of the neural network via automatic differentiation [18]. As such, approximating the PDE is no more than minimizing the loss function, which can be done with gradient descent techniques. This method of solving PDEs is mesh-free and simple, and it can be applied to a wide variety of PDE types. In addition to solving forward problems, PINNs can be easily implemented to solve inverse PDE problems. These involve predicting the values of parameters given a set of measurements of the function. PINNs have achieved success with both forward and inverse problems in a diverse range of fields, including optics [19], systems biology [20], fluid mechanics [21, 22], and biomedicine [23, 24]. However, there have been fewer applications to problems in economics, which this paper will explore.

In this paper, we will consider two models. First, we solve a model of industrial dynamics with financial frictions, which considers an industrial equilibrium of a banking sector that takes deposits, makes loans, and uses labor input to manage deposits and loans. Then, we consider a macroeconomic model with the financial sector featuring binding constraints, non-linear financial amplifications, and boundary singularity. We demonstrate the advantage of our methodology in providing a general framework for solving PDEs, as opposed to traditional methods, which require designing different algorithms for different problems. We also display the ease of solving inverse problems, in which we assume parameters are unknown and impose moment conditions. Thus, we are able to simultaneously solve and estimate parameters.

In Section 2, we introduce the physics-informed neural network method for solving both forward and inverse partial differential equations, along with the Python library DeepXDE. Following this introduction, in Sections 3 and 4 we apply the PINN method to solve several equations in economics.

2 Machine learning for PDEs

We first describe general deep neural networks, allowing us to present the framework of physics-informed neural networks (PINNs), which will be leveraged to solve forward and inverse PDEs (i.e., model solution and model estimation). Afterward, overview the hyperparameters used in the following sections.

2.1 Deep neural networks

Although there are several different types of deep neural networks, throughout this paper, we use the feed-forward neural network (FNN). We define an L-layer FNN as a function $\mathcal{N}^L(\mathbf{x})\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ and say that there are $L-1$ hidden layers such that the ℓ-th layer has N_ℓ neurons. Clearly, $N_0 = d_{\text{in}}$ and $N_L = d_{\text{out}}$. Furthermore, for each $1 \leq \ell \leq L$, we define a weight matrix $\mathbf{W}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and bias vector $\mathbf{b}^\ell \in \mathbb{R}^{N_\ell}$. Then, letting $T^\ell(\mathbf{x}) = \mathbf{W}^\ell\mathbf{x} + \mathbf{b}^\ell$ be the affine transformation in the ℓ-th layer, for some non-linear activation function σ, we say

$$\mathcal{N}^L(\mathbf{x}) = T^L \circ \sigma \circ T^{L-1} \circ \ldots \circ \sigma \circ T^1(\mathbf{x}).$$

A network with $L = 4$ can be visualized in Fig. 1. There are several possible activation functions σ, and in this paper, we use both the hyperbolic tangent (tanh) and the swish [25] activation functions. The former is the most typical activation function in machine learning, while the latter can better deal with problems with steep gradients, which are features of the problems we will discuss.

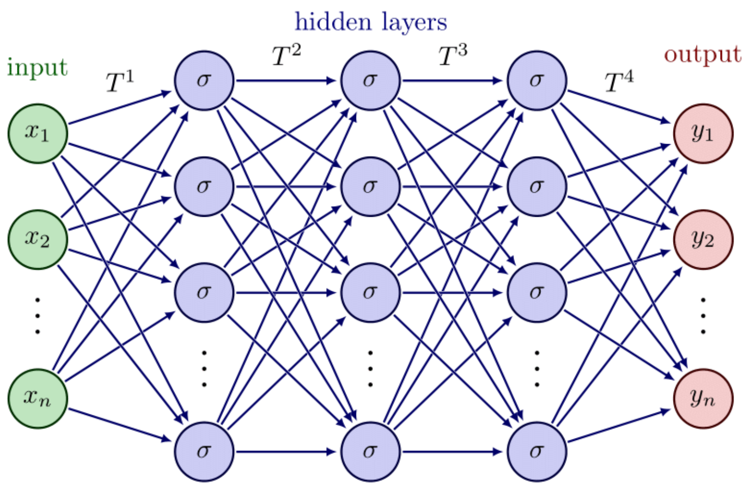

Figure 1: **Visualization of a deep neural network described in Section 2.1.** In this diagram, the number of layers L is 4.

2.2 PINNs for solving forward PDEs

We first discuss the use of PINNs to solve forward PDEs. Consider the following PDE parameterized by $\boldsymbol{\lambda}$ with solution $u(\mathbf{x}, t)$ for $\mathbf{x} = (x_1, \dots, x_d)$ over the domain $\Omega \subset \mathbb{R}^d$:

$$f\left(\mathbf{x}; \frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \dots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \dots; \boldsymbol{\lambda}\right) = 0, \quad \mathbf{x} \in \Omega \tag{2.1}$$

with boundary conditions

$$\mathcal{B}(u, \mathbf{x}) = 0 \quad \text{on} \quad \partial\Omega.$$

To find the solution, we create a neural network $\hat{u}(\mathbf{x}; \boldsymbol{\theta})$ with trainable parameters $\boldsymbol{\theta}$. When training the network, we use $\mathcal{T}_f$ points inside the domain and $\mathcal{T}_b$ points on the boundary. Then, the loss function is defined as

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) \tag{2.2}$$

with

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, \dots, \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \dots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_d}; \dots; \boldsymbol{\lambda}\right) \right\|_2^2,$$
$$\mathcal{L}_b(\boldsymbol{\theta}, \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}, \mathbf{x})\|_2^2,$$

for weights w_f and w_b. Now that we have defined a loss function, where the two losses are the L^2 norms of the residuals, we can train like any normal neural network using gradient-based optimizers such as Adam [26] and L-BFGS [27]. Training the network produces an approximation of $u(\mathbf{x}, t)$.

2.3 PINNs for solving inverse PDEs

Next, we discuss using PINNs to solve inverse PDEs. Inverse problems involve some unknown parameters λ in Eq. (2.1) to be solved for, but we are given some extra information about some points $\mathcal{T}_i \in \Omega$ besides the PDE and boundary conditions [28, 14]:

$$\mathcal{I}(u, \mathbf{x}) = 0 \quad \text{for} \quad \mathbf{x} \in \mathcal{T}_i.$$

Training the PINN in the inverse method is almost identical to training the forward PINN, except that the loss function Eq. (2.2) has an extra term:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_b) + w_i \mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_i),$$

where

$$\mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T}_i) = \frac{1}{|\mathcal{T}_i|} \sum_{\boldsymbol{x} \in \mathcal{T}_i} \|\mathcal{I}(\hat{u}, \boldsymbol{x})\|_2^2,$$

the L^2 norm of the residual. Optimizing $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ together, our solution is $\boldsymbol{\theta}^*, \boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\theta}, \boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathcal{T})$.

The above approach incorporates model solution (solving function u) and model estimation (solving parameter λ) together in a consistent way. Furthermore, we can easily generalize the above to a PDE system with multiple functions to be solved.

2.4 Implementation

We will apply PINNs to solve several forward and inverse PDE problems in economics, and we implement these with the Python library DeepXDE [14]. In all examples, we use the Glorot uniform initializer. In addition, we occasionally enforce boundary conditions automatically through an output transform [29]. Throughout the paper, we use a learning rate decay with the formula

$$\gamma_n = \frac{\gamma_0}{1 + \alpha n / S},$$

where n is the number of iterations, γ_n is the learning rate after n iterations, γ_0 is the initial learning rate, α is the decay rate, and S is the decay step. The other hyperparameters are listed in Table 1.

Table 1: **Hyperparameters used for each problem.**

Section	Depth	Width	Activation Function	Optimizer	Learning Rate	#Iterations
3.2	7	64	tanh	Adam	5×10^{-4}	7.5×10^4
3.3.2	7	64	tanh	Adam	1×10^{-3}	1.5×10^5
3.3.3	7	64	tanh	Adam	1×10^{-3}	1.5×10^5
4.2	7	128	swish	Adam	1×10^{-3}, 1×10^{-5}	3.0×10^5
4.3.1	6	64	swish	Adam	1×10^{-4}	2.0×10^5
4.3.2	6	64	swish	Adam	5×10^{-4}	1.5×10^5

3 A model of industrial dynamics with financial frictions

In this section, we consider an industrial equilibrium of a banking sector that takes deposits, makes loans, and uses labor input to manage deposits and loans. Labor productivity determines the number of loans and deposits that each bank can take. Both loan rate and deposit rates are endogenously determined via the competitive market equilibrium that features bank entry, exit, and equity payouts. The challenges of this problem are twofold: first, banks endogenously determine whether or not to enter or exit the market, so the problem features endogenous entry and exit boundaries; second, we need to track the entire distribution of banks in order to clear the market.

3.1 Problem setup

Time evolves continuously. All bank assets and debts are modeled as short-term. Banks can borrow via deposits at a rate r^d, and via capital market at a rate r (think of this as the policy rate, e.g., FFR). Banks can lend at loan rate r^l, or the capital market rate r. Banks need to hire workers to serve their deposits and loans. For banks with productivity z, the number of loans that l units of labor can serve is

$$f(z, l) = z l^{\alpha}, \quad \alpha \in (0, 1)$$

which features decreasing return to scale. A rationale for this assumption is that as banks get bigger, it is increasingly difficult to find new depositors and new borrowers on which they can earn a profit. Similarly, the amount of deposits that d units of labor can serve is

$$f(z, d) = z d^{\alpha}.$$

Because banks can borrow and lend freely in the capital market at a rate r, they are not constrained by lending and deposit-taking choices, as long as they are nonnegative.

The stochastic process for z_t is given exogenously as

$$dz_t = \mu(z_t)dt + \sigma(z_t)dB_t,$$

with two reflecting boundaries $\underline{z}$ and $\bar{z}$.

Denote bank equity as e. We impose the financial friction as follows:

$$f(z,l) \leq \phi e,$$

$$f(z,d) \leq \phi e.$$

Banks incur a fixed operating cost c_f, and thus the instantaneous profit is

$$\pi(e_t, z_t, l_t, d_t) = \underbrace{r_t^l z_t l_t^\alpha}_{\text{lending revenue}} - \underbrace{r_t^d z_t d_t^\alpha}_{\text{deposits interest expense}} + \underbrace{(z_t d_t^\alpha + e_t - z_t l_t^\alpha) r_t}_{\text{net capital market lending}} - \underbrace{w_t \cdot (l_t + d_t)}_{\text{labor cost}} - \underbrace{c_f}_{\text{fixed cost}}.$$

To keep the problem simple, we assume full symmetry between the deposit market and the loan market. Both loan demand and deposit demand functions are in the same form:

$$r^l - r = \beta(L + L_0)^{-\varepsilon},$$

$$r - r^d = \beta(D + D_0)^{-\varepsilon},$$

where D and L are the aggregate amount of deposits and loans, respectively. We assume $L_0 = D_0$ for simplicity. Due to the full symmetry, the loan spread is equal to the deposit spread, $r^l - r = r - r^d$. In what follows, we will only use the notations on the loan side.

Since the bank can freely adjust its labor input and scales of operations at each instance, the optimal decisions are,

$$l^* = d^* = \min\left\{ \left(\frac{(r^l - r)z\alpha}{w} \right)^{\frac{1}{1-\alpha}}, \left(\frac{\phi e}{z} \right)^{1/\alpha} \right\}.$$

Therefore, the optimized profit function is

$$\pi^*(e,z) = 2\left(r^l - r\right) z \left(l^*\right)^\alpha + e \cdot r - 2wl^* - c_f.$$

Denote $v(e,z)$ as the value function for a bank with equity e and productivity z, with $\underline{v}(e) = e$ the reservation value if a bank exists. We assume that banks pay out equity when they are financially unconstrained, and the equity payout function is

$$\zeta(e,z) = \max\left\{ \kappa\left(\phi e - f(z,l^*)\right), 0 \right\}.$$

In other words, equity payout smoothly increases as the bank gets further away from the financial constraint. Then we can write the bank equity dynamics as

$$de_t = \underbrace{\left(\pi^*(e_t,z_t) - 1_{(e_t,z_t)\in\mathcal{C}^c} \cdot \zeta(e_t,z_z)\right)}_{\equiv \mu_e(e_t,z_t)} dt - 1_{v(e_t,z_t)<\underline{v}(e_t)} \cdot e_t,$$

where the last term reflects the immediate exit when the bank continuation value is smaller than the liquidation value.

Banks optimize the expected discounted cash flows at the rate r. The HJB equation is:

$$r \cdot v(e,z) = \max\left\{ \pi^*(e,z)(1 + v_e') + (1 - v_e')\zeta(e,z)1_{(e,z)\in\mathscr{C}^c} + v_z'\mu(z) + \frac{1}{2}v_{zz}''\sigma(z)^2, r\underline{v}(e) \right\},$$

where the set $\mathscr{C}$ is the region where the bank is financially constrained, and the unconstrained indicator $1_{(e,z)\in\mathscr{C}^c}$ is

$$1_{(e,z)\in\mathscr{C}^c} = 1\left\{\left(\frac{\left(r^l - r\right)z\alpha}{w}\right)^{\frac{1}{1-\alpha}} < \left(\frac{\phi e}{z}\right)^{1/\alpha}\right\}.$$

At the reflecting barriers, we have

$$\partial_z v(e,\underline{z}) = \partial_z v(e,\bar{z}) = 0, \quad \text{for any } e.$$

We also have the boundary condition

$$v(e,z) = \underline{v}(e) = 0.01 \quad \text{when } e = 0.01.$$

Banks will exit the market with zero equity, because $de_t \leq 0$ for $e_t = 0$, and $\pi(0) = -c_f < 0$, i.e., zero equity is an absorbing state, and continuing the operations when equity is zero will guarantee negative profit and thus is worse than exiting the market.

Finally, we describe entry dynamics. Banks first decide whether or not to enter, and then draw their productivity from the distribution $\psi(z)$. We assume that entry incurs a one-time cost c_e, and each entrant has the same initial equity of e_0. The mass of firms entering the market is determined by

$$m = \bar{m}\exp\left(\beta_M\left(\iint_{e,z} v(e,z)\psi(e,z)dzde - c_e\right)\right).$$

The above is a softer version of the free-entry condition. When $\beta_M \to \infty$, entry incentive with respect to entry benefit is going to infinity, so the present value of entry must be zero and we arrive at the free-entry condition

$$\iint_{e,z} v(e,z)\psi(e,z)dzde - c_e = 0.$$

Solve for invariant distribution and estimation: Denote the stationary bank distribution as $g(e,z)$. This distribution does not include banks that exit the market, so we have

$$g(e,z) = 1_{v(e,z)>\underline{v}(e)}g(e,z),$$

where the assumption is that when banks are indifferent between staying or exiting the market, they choose to exit the market.

The Kolmogorov Forward Equation (KFE) for the stationary distribution in banking industrial dynamic model is

$$0 = -\frac{\partial}{\partial z}\left(\mu_z(z)g(e,z)\right) - \frac{\partial}{\partial e}\left(\mu_e(e,z)g(e,z)\right) + \frac{1}{2}\frac{\partial^2}{\partial z^2}\left(\sigma(z)^2 g(e,z)\right) + m\psi(e,z),\ v(e,z) > \underline{v}(e). \tag{1}$$

With the stationary distribution, we can get the aggregate loan

$$L = \iint g(e,z)f(z,l^*(e,z))dzde.$$

Furthermore, the equilibrium loan spread is determined by the household loan demand function,

$$r^l - r = \beta(L + D_0)^{-\varepsilon} \tag{2}$$

for $D_0 = \iint_{e,z} d(e,z)g(e,z)dzde$ (here g is not normalized), where $d = z(l^*)^\alpha$. We assume that $\psi(e,z)$ is a truncated normal distribution ($\Phi(\cdot)$ is the c.d.f. of normal distribution)

$$\psi(e,z) = \frac{1}{\bar{e}\times\left(\Phi\left(\frac{\bar{z}-z_m}{\sigma_\psi}\right) - \Phi\left(\frac{\underline{z}-z_m}{\sigma_\psi}\right)\right)}\frac{1}{\sqrt{2\pi\delta_\psi^2}}\exp\left(-\frac{(z-z_m)^2}{2\delta_\psi^2}\right)\times 1_{e<\bar{e}}\times 1_{z\in[\underline{z},\bar{z}]}$$

Boundary conditions: (1) Banks exit at $\underline{v}(e) = e$ (absorbing boundary), which means $g(e,z) = 0$ when $v(e,z) = e$. (2) Reflecting boundary for stochastic productivity z: $-\mu_z(z)g(e,z) + \frac{1}{2}\frac{\partial}{\partial z}(\sigma(z)^2 g(e,z)) = 0$, when $z = \underline{z}, \overline{z}$.

Specification: Model parameter specifications are shown in Table 2, and additional problem setup details can be found in Appendix A.

Table 2: **Parameter specification of model in Section 3.1.**

Description	Value
Bank equity payout rate	$\kappa = 0.005$
Share of labor	$\alpha = 0.3$
Leverage constraint parameter	$\phi = 10$
Benchmark interest rate	$r = 0.03$
Fixed operating cost	$c_f = 0.03$
Entry cost	$c_e = 0.1$
Boundary and mean of productivity	$\underline{z} = 0.2, \overline{z} = 10, z_m = 5$
Lower and upper bound of state space	$e_{\text{min}} = 0.01, e_{\text{max}} = 1.2$
Drift and volatility of z	$\mu(z) = -0.005(z - z_m), \sigma(z) = 0.08$
Deposit/Loan supply function's constant	$D_0 = L_0 = 1.0$
Entrance distribution parameters	$\sigma_\psi = \frac{\overline{z}-\underline{z}}{4}, \bar{e} = 0.15, \bar{m} = 0.1.$
Entrance elasticity	$\beta_M = 1 \times 10^3$

3.2 Solving the model

Our goal is to solve for $v(e,z)$ and $g(e,z)$. To do so, we employ a technique simplifying the model by eliminating the role of m from training.

3.2.1 Elimination of m

The only equation in which m directly shows up is the KFE (Eq. (1)), in which $g(e,z)$ scales linearly in m. Linear scaling of $g(e,z)$ does not affect the values of the moment targets because $g(e,z)$ is normalized in those calculations. However, scaling $g(e,z)$ affects Eq. (2), in which

$$L = \iint_{e,z} g(e,z) f(z, l^*(e,z)) dz de$$

is scaled by the same factor. Our goal is to exactly satisfy Eq. (2) by scaling $g(e,z)$. Afterwards, in order to still satisfy the KFE, we scale m by the same factor. To implement this idea with a PINN, we fix $m = 1$ throughout training. After training, let $\mathcal{N}_g(e,z)$ be the PINN-predicted value of $g(e,z)$. We calculate

$$\mathcal{N}_L = \iint_{e,z} \mathcal{N}_g(e,z) f(z, l^*(e,z)) dz de.$$

To automatically satisfy Eq. (2), we set m to be

$$r^l - r = \beta(\mathcal{N}_L m - D_0)^{-\varepsilon}$$
$$\implies m = \frac{\frac{\beta}{r^l - r} - D_0}{L},$$

in which we use $\varepsilon = 1$. After solving for m, our final prediction of $g(e,z)$ is $g(e,z) = m\mathcal{N}_g(e,z)$.

3.2.2 Other technical details

We solve the model with the unknown endogenous boundary r^l. When training the PDE, we enforce the Dirichlet boundary condition on v via a soft boundary condition and the Dirichlet boundary condition on g through a hard boundary condition. Furthermore, we enforce the Neumann boundary condition on g via a soft boundary condition. We use loss weights of 10^6 for the HJB residual, 5×10^4 for the KFE residual, 10^3 for the free-entry condition, 10^2 for the Dirichlet boundary condition on v, 10^3 for the Neumann boundary condition on v, and 10^5 for the Neumann boundary condition on g.

Additionally, we train with 2^{16} training points sampled inside the domain, 2^{10} training points sampled on the boundary, and 2^{16} points sampled inside the domain for testing. When estimating r^l, we scale it up 100 times while training and scale it back down afterwards. Lastly, we use a learning rate scheduler, with an initial learning rate of 5×10^{-4}, decay rate 1.0, and decay step 6000.

The training results are displayed in Fig. 2 while the predicted value of r^l and m after training is displayed in Table 3. The training loss decreases steadily (Fig. 2A), and the endogenous variable r^l converges to its true value. The L^2 relative errors for v and g_c[1] are 0.54% and 4.32%, respectively.

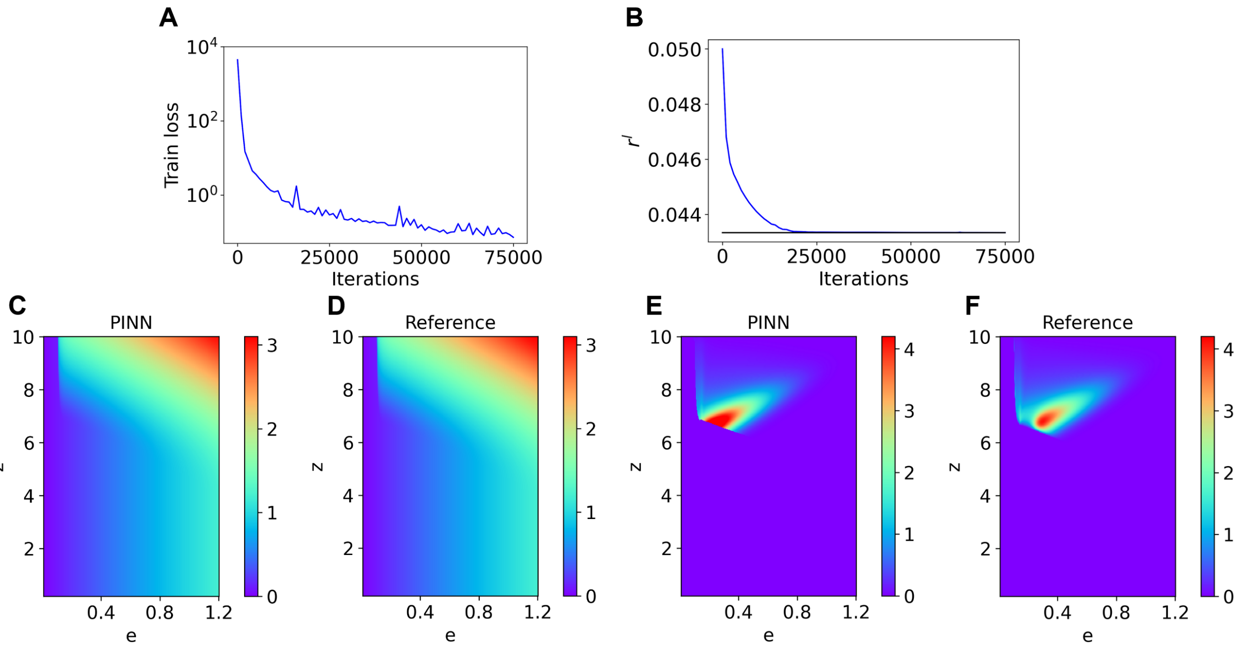

Figure 2: **Example in Section 3.2: PINN results for forward HJB equation.** (**A**) The trajectory of train loss throughout training. (**B**) The convergence of r^l throughout training. (**C**) 2D heatmap for predicted solution for v using PINN. (**D**) 2D heatmap for reference solution for v using Matlab. (**E**) 2D heatmap for predicted solution for g using PINN. (**F**) 2D heatmap for reference solution for g using Matlab.

Table 3: **Example in Section 3.2: Predicted values of endogenous variables r^l and m after training.**

Variable	True Value	Predicted Value	Error
r^l	0.043337	0.043343	0.01%
m	0.2194	0.2242	2.12%

3.3 Simultaneously solving and estimating the model

In practice, we may want to estimate some unknown parameters of the model, which requires additional information in the form of moment targets. First, we describe techniques which will make estimation easier.

[1] g_c is defined as the c.d.f of g, i.e., $g_c(e, z) = \int_{\underline{z}}^{z} \int_0^e g(e', z')de'dz'$.

3.3.1 Elimination of free-entry condition in training

In Subsections 3.3.2 and 3.3.3, we predict the value of c_e, along with several other parameters. Note that c_e only appears in the free-entry condition. To take advantage of this, we train the PINN without the free-entry condition, and after training, we predict c_e by calculating

$$c_e = \iint_{e,z} v(e,z)\psi(e,z)dzde.$$

3.3.2 Estimation of two parameters

In this example, we estimate the two parameters α and c_e. Define the density $g'(e,z)$ as $\max\{g(e,z),0\}$ discounted by normalization factor[2]: $\frac{1}{\iint_{e,z}\max\{g(e,z),0\}dzde}$, representing the normalized density. In estimation, we adjust the unknown parameters to match the average productivity z_{target} and average labor l^*_{target}:

$$z_{\text{target}} = \iint_{e,z} zg'(e,z)dzde,$$
$$l^*_{\text{target}} = \iint_{e,z} l^* g'(e,z)dzde.$$

When training the PDE, we enforce the Dirichlet boundary condition on v via a soft boundary condition and the Dirichlet boundary condition on g through an output transform. Furthermore, we enforce the Neumann boundary condition on g via a soft boundary condition. We use loss weights of 10^6 for the HJB residual, 5×10^4 for the KFE residual, 10^1 for z_{target}, 10^6 for l^*_{target}, 10^2 for the Dirichlet boundary condition on v, 10^3 for the Neumann boundary condition on v, and 10^5 for the Neumann boundary condition on g. Furthermore, we train with 2^{14} training points sampled inside the domain, 2^9 training points sampled on the boundary, and 2^{14} points sampled inside the domain for testing. When estimating α, we scale it up 10 times while training and scale it back down afterwards. For c_e, we employ the technique described in Section 3.3.1. Lastly, we use a learning rate scheduler, with an initial learning rate of 10^{-3}, decay rate 1.0, and decay step 2500.

The training results are displayed in Fig. 3. In addition, the predictions and errors for c_e and α are displayed in Table 4, along with the predictions and errors for z_{target} and l^*_{target}. The training loss decreases steadily (Fig. 3A), and the L^2 relative error of v ends at 3.28%. The trajectory of α is displayed in Fig. 3B. As shown in Figs. 3C and D, the PINN predicts the solution for v accurately. Furthermore, Figs. 3E and F demonstrate that the PINN predicts the solution for g accurately. The errors of both the predicted parameters and the moments of the predicted solution are under 5% (Table 4). Finally, the L^2 relative errors of v and g_c are 3.28% and 4.24%, respectively.

[2]As KFE operator $\hat{L}^*$ is the Markov process's generator, the distribution is always positive when evolving over time. However, in numerical exercise, $g(e,z)$ at some point can be negative.

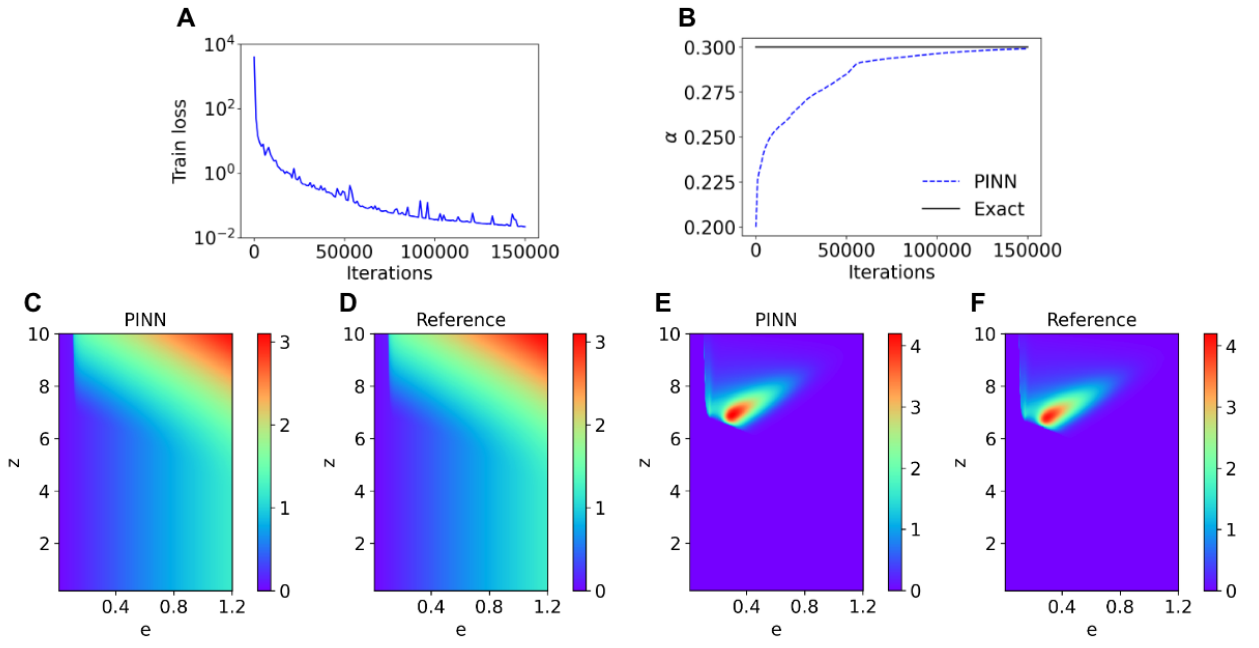

Figure 3: **Example in Section 3.3.2: PINN results for inverse HJB equation with estimation of two parameters.** (**A**) The trajectory of train loss throughout training. (**B**) The convergence of α throughout training. (**C**) 2D heatmap for predicted solution for v using PINN. (**D**) 2D heatmap for reference solution for v using Matlab. (**E**) 2D heatmap for predicted solution for g using PINN. (**F**) 2D heatmap for reference solution for g using Matlab.

Table 4: **Example in Section 3.3.2: PINN results for inverse HJB equation with estimation of two parameters.** Both unknown parameters c_e and α and moment targets z_{target} and l^*_{target} are shown.

	True Value	Predicted Value	Error
c_e	0.1	0.0959	4.10%
α	0.3	0.299	0.31%
z_{target}	7.524	7.611	1.15%
l^*_{target}	0.00647	0.00637	1.46%

3.3.3 Estimation of three parameters

Now, we estimate three parameters: α, c_e, and c_f. We will match four moment conditions: average equity e_{target}, average productivity z_{target}, average leverage ℓ_{target}, and average labor l^*_{target}. They are defined as

$$\begin{aligned} e_{\text{target}} &= \iint_{e,z} e g'(e,z) dz de, \\ z_{\text{target}} &= \iint_{e,z} z g'(e,z) dz de, \\ \ell_{\text{target}} &= \iint_{e,z} \frac{f(z, l^*(e,z))}{e} g'(e,z) dz de, \\ l^*_{\text{target}} &= \iint_{e,z} l^* g'(e,z) dz de. \end{aligned}$$

When training the PDE, we enforce the Dirichlet boundary condition on v via a soft boundary condition and the Dirichlet boundary condition on g through an output transform. Furthermore, we enforce the Neumann boundary condition on g via a soft boundary condition. We use loss weights of 10^6 for the HJB residual, 5×10^4 for the KFE residual, 10^2 for e_{target}, 10^1 for z_{target}, 10^0 for ℓ_{target}, 10^6 for l^*_{target}, 10^2 for the

Dirichlet boundary condition on v, 10^3 for the Neumann boundary condition on v, and 10^5 for the Neumann boundary condition on g. Furthermore, we train with 2^{16} training points sampled inside the domain, 2^{10} training points sampled on the boundary, and 2^{16} points sampled inside the domain for testing. When estimating α and c_f, we scale them up 10 and 100 times, respectively. After training, we scale them back down. For c_e, we employ the technique described in Section 3.3.1. Lastly, we use a learning rate scheduler, with an initial learning rate of 1×10^{-3}, decay rate 1.0, and decay step 2500.

The training results are displayed in Fig. 4. In addition, the predictions and errors for c_e, α, and c_f are displayed in Table 5, along with the predictions and errors for e_{target}, z_{target}, ℓ_{target}, and l^*_{target}. The training loss decreases steadily (Fig. 4A). We see that both α and c_f converge to a values close to the true values (Figs. 4B and C). As shown in Figs. 3C and D, the PINN predicts the solution for v accurately. Furthermore, Figs. 3E and F demonstrate that the PINN predicts the solution for g accurately. The errors of both the predicted parameters and the moments of the predicted solution are under 10% (Table 5). Finally, the L^2 relative errors of v and g_c are 8.53% and 13.21%, respectively.

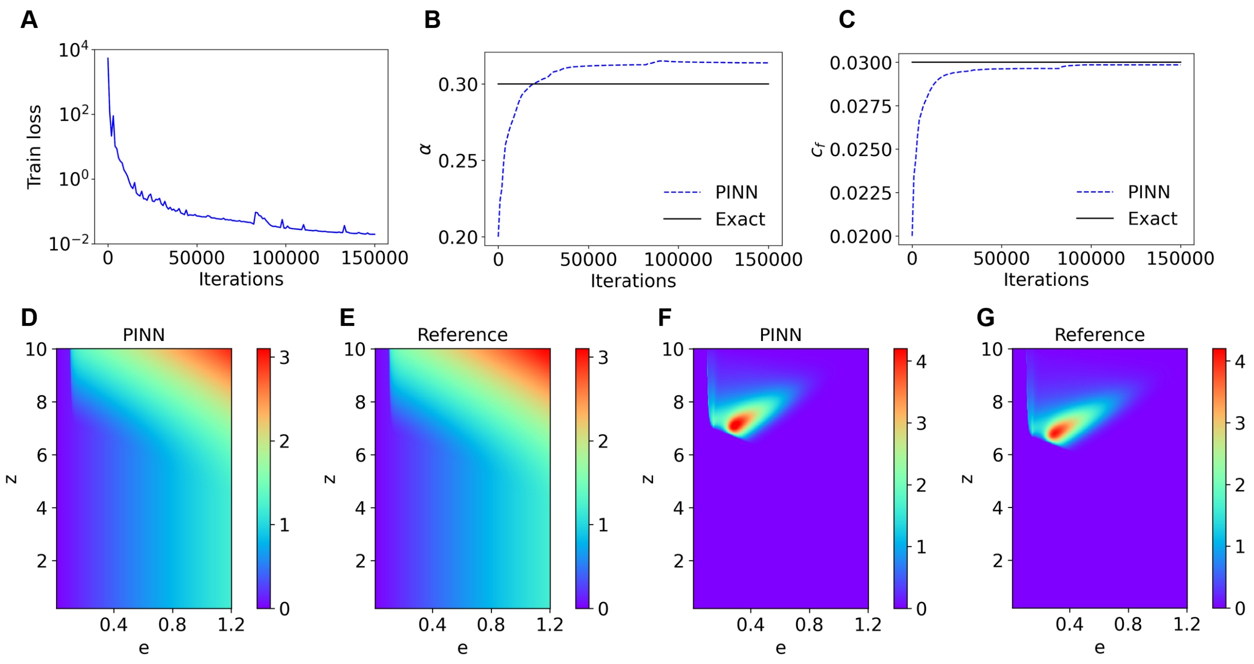

Figure 4: **Example in Section 3.3.3: PINN results for inverse HJB equation with estimation of three parameters.** (**A**) The trajectory of train loss throughout training. (**B**) The convergence of α throughout training. (**C**) The convergence of c_f throughout training. (**D**) 2D heatmap for predicted solution for v using PINN. (**E**) 2D heatmap for reference solution for v using Matlab. (**F**) 2D heatmap for predicted solution for g using PINN. (**G**) 2D heatmap for reference solution for g using Matlab.

Table 5: **Example in Section 3.3.3: PINN results for inverse HJB equation with estimation of three parameters.** Unknown parameters c_e, α, and c_f as well as moment targets e_{target}, z_{target}, ℓ_{target} and l^*_{target} are shown.

	True Value	Predicted Value	Error
c_e	0.1	0.0934	6.60%
α	0.3	0.314	4.59%
c_f	0.03	0.0299	0.49%
e_{target}	0.418	0.385	7.96%
z_{target}	7.524	7.786	3.47%
ℓ_{target}	4.697	4.955	5.50%
l^*_{target}	0.00647	0.00677	4.70%

4 A macroeconomic model with the financial sector

The next problem we discuss describes the model in **(author?)** [2]. The problem features occasional binding constraints, non-linear financial amplifications, and singularity at the boundary. We describe the model only briefly—see [2] for the detailed model setup.

4.1 Problem setup

There are two types of agents, a continuum mass of bankers and a continuum mass of households, both with risk-neutral utility. There are two types of assets: productive capital and risk-free asset. Per unit of capital, banker productivity is a while household productivity is $\underline{a} < a$. Both bankers and households can borrow and lend at the risk-free rate. Banker discount rate δ is smaller than the household discount rate, which motivates them to borrow from households.

To solve for the equilibrium, we need to solve the value of capital $q(\omega)$ and the endogenous marginal value of wealth $\theta(\omega)$ for the banker, where ω is the fraction of the banker wealth among total wealth and is the state variable that determines asset prices and allocations. There is an endogenous boundary η^* where banks pay dividends. The function $q(\eta)$ is increasing and $\theta(\eta)$ is decreasing over $[0, \eta^*]$, with boundary conditions

$$q(0) = \underline{q}, \quad \theta(\eta^*) = 1, \quad q'(\eta^*) = 0, \quad \theta'(\eta^*) = 0, \quad \text{and} \quad \lim_{\eta \to 0} \theta(\eta) = \infty,$$

where the boundary value $\underline{q}$ is determined by

$$\underline{q} = \max_{q} \frac{\underline{a} - \iota(x)}{r - (\Phi(x) - \underline{\delta})}. \tag{3}$$

Note that the maximum possible value of $q(\eta)$ is determined by the following equation:

$$\frac{a - \iota(\bar{q})}{\bar{q}} + \Phi(\bar{q}) - \delta - r = 0,$$

where $\iota(\cdot)$ and $\Phi(\cdot)$ are given by

$$\Phi(x) = \frac{x-1}{\kappa} \quad \text{and} \quad \iota(x) = \Phi(x) + \frac{1}{2}\kappa\Phi(x)^2.$$

Functions $q(\eta)$ and $\theta(\eta)$ are second-order ODEs. Thus, we need two boundary conditions for $q(\eta)$ and two boundary conditions for $\theta(\eta)$. The above are five conditions, but we also have an unknown boundary η^*.

To solve the model, we first find $\psi \in (\eta, \eta + q(\eta)/q'(\eta))$, such that

$$\frac{a - \underline{a}}{q(\eta)} + \underline{\delta} - \delta + (\sigma + \sigma^q(\eta))\sigma^\theta(\eta) = 0, \tag{4}$$

where

$$\sigma^\eta(\eta)\eta = \frac{(\psi - \eta)\sigma}{1 - (\psi - \eta)q'(\eta)/q(\eta)}, \quad \sigma^q(\eta) = \frac{q'(\eta)}{q(\eta)}\sigma^\eta(\eta)\eta, \quad \text{and} \quad \sigma^\theta(\eta) = \frac{\theta'(\eta)}{\theta(\eta)}\sigma^\eta(\eta)\eta.$$

We can easily prove that the left side of equation (4) is monotonic in ψ and has a unique solution. If the above solution indicates $\psi > 1$, then we set $\psi = 1$ and recalculate $\sigma^\eta(\eta)$, $\sigma^q(\eta)$, and $\sigma^\theta(\eta)$.

Next, we compute the second-order derivatives, via

$$q''(\eta) = \frac{2\left(\mu^q(\eta)q(\eta) - q'(\eta)\mu^\eta(\eta)\eta\right)}{\sigma^\eta(\eta)^2\eta^2}, \tag{5}$$

$$\theta''(\eta) = \frac{2\left(\mu^\theta(\eta)\theta(\eta) - \theta'(\eta)\mu^\eta(\eta)\eta\right)}{\sigma^\eta(\eta)^2\eta^2}, \tag{6}$$

where

$$\mu^\eta(\eta) = -\frac{(\psi - \eta)(\sigma + \sigma^q(\eta))(\sigma + \sigma^q(\eta) + \sigma^\theta(\eta))}{\eta} + \frac{a - \iota(q(\eta))}{q(\eta)} + (1 - \psi)(\underline{\delta} - \delta),$$

$$\mu^q(\eta) = r - \frac{a - \iota(q(\eta))}{q(\eta)} - \Phi(q(\eta)) + \delta - \sigma\sigma^q(\eta) - \sigma^\theta(\eta)\cdot(\sigma + \sigma^q(\eta)), \quad \text{and} \quad \mu^\theta = \rho - r.$$

Specification: Model parameter specifications are shown in Table 6 and additional problem setup details and derivations can be found in Appendix B.

Table 6: **Parameter specification of model in Section 4.1**

Description	Value
Capital productivity (experts)	$a = 0.11$
Capital productivity (households)	$\underline{a} = 0.05$
Discount rate (experts)	$\rho = 0.06$
Discount rate (households)	$r = 0.06$
Unit volatility of capital	$\sigma = 0.025$
Capital depreciation rate (experts)	$\delta = 0.03$
Capital depreciation rate (households)	$\underline{\delta} = 0.08$
Loss factor in investment	$\kappa = 10$

4.2 Solving the model

Our goal is to solve for the functions q, θ, and ψ. To do this, we use an assortment of techniques.

4.2.1 Change of variable to deal with singularity

First, when training the model, we perform a change of variable for θ. We define the function $\hat{\theta}(\eta) = \frac{1}{\theta(\eta)}$. The motivation behind this is to deal with the singularity for θ at $\eta = 0$. With this change of variable, we rewrite all the boundary conditions:

$$\lim_{\eta\to 0} \theta(\eta) = \infty \longrightarrow \hat{\theta}(0) = 0,$$
$$\theta(\eta^*) = 1 \longrightarrow \hat{\theta}(\eta^*) = 1,$$
$$\theta'(\eta^*) = 0 \longrightarrow \hat{\theta}'(\eta^*) = 0.$$

We now rewrite the ODEs in the problem setup with the new variable $\hat{\theta}$. We compute

$$\sigma^\theta(\eta)\hat{\theta}(\eta) = -\hat{\theta}(\eta)'\sigma^\eta(\eta)\eta.$$

This allows us to rewrite (4) as

$$\left(\frac{a - \underline{a}}{q} + \underline{\delta} - \delta\right)\hat{\theta}(\eta) + (\sigma + \sigma_q)\sigma^\theta(\eta)\hat{\theta}(\eta) = 0.$$

We also rewrite these definitions as

$$\mu^\eta(\eta)\eta\hat{\theta}(\eta) = -(\psi - \eta)(\sigma + \sigma^q)(\hat{\theta}(\eta)(\sigma + \sigma^q) + \sigma^\theta(\eta)\hat{\theta}(\eta)) + \eta\hat{\theta}(\eta)\left(\frac{a - \iota}{q} + (1 - \psi)(\underline{\delta} - \delta)\right),$$
$$\mu^q(\eta)\hat{\theta}(\eta) = \hat{\theta}(\eta)\left(r - \frac{a - \iota(q(\eta))}{q} - \Phi(q(\eta)) + \delta - \sigma\sigma^q\right) - \sigma^\theta(\eta)\hat{\theta}(\eta)\cdot(\sigma + \sigma^q(\eta)).$$

Lastly, (5) and (6) are now rewritten as

$$q''(\eta)\sigma^\eta(\eta)^2\eta^2\hat{\theta}(\eta) = 2(\mu^q(\eta)\hat{\theta}(\eta)q(\eta) - q'(\eta)\mu^\eta(\eta)\eta\hat{\theta}(\eta)),$$
$$\sigma^\eta(\eta)^2\eta^2\cdot(2\hat{\theta}'(\eta)^2 - \hat{\theta}(\eta)\hat{\theta}''(\eta)) = 2(\mu^\theta\hat{\theta}(\eta)^2 + \hat{\theta}'(\eta)\mu^\eta(\eta)\hat{\theta}(\eta)).$$

4.2.2 Explicitly solving for ψ

While training, we simultaneously solve for q and $\hat{\theta}$. We train ψ as an auxiliary function that updates every 1000 iterations, given that q and $\hat{\theta}$ are increasing. Our initial guess for ψ is $\psi(\eta) = 1$. This guess for ψ is flexible since it is updated once q and $\hat{\theta}$ are increasing, which generally happens quickly. To compute ψ when training, we solve for the function explicitly. We have

$$\begin{aligned}\sigma^q(\eta) &= \frac{q'(\eta)}{q(\eta)}\sigma^\eta(\eta)\eta \\ &= \frac{q'(\eta)(\psi-\eta)\sigma}{q(\eta)-(\psi-\eta)q'(\eta)}.\end{aligned}$$

Then,

$$\begin{aligned}\sigma^q(\eta)+\sigma &= \frac{q'(\eta)(\psi-\eta)\sigma}{q(\eta)-(\psi-\eta)q'(\eta)}+\sigma \\ &= \frac{q'(\eta)(\psi-\eta)\sigma+\sigma q(\eta)-\sigma(\psi-\eta)q'(\eta)}{q(\eta)-(\psi-\eta)q'(\eta)} \\ &= \frac{\sigma q(\eta)}{q(\eta)-(\psi-\eta)q'(\eta)}.\end{aligned}$$

Furthermore,

$$\begin{aligned}(\sigma+\sigma^q(\eta))\sigma^\theta(\eta) &= \left(\frac{\sigma q(\eta)}{q(\eta)-(\psi-\eta)q'(\eta)}\right)\frac{\theta'(\eta)(\psi-\eta)\sigma}{\theta(\eta)(1-(\psi-\eta)q'(\eta)/q(\eta))} \\ &= \left(\frac{\sigma q(\eta)}{q(\eta)-(\psi-\eta)q'(\eta)}\right)\frac{q(\eta)\theta'(\eta)(\psi-\eta)\sigma}{\theta(\eta)(q(\eta)-(\psi-\eta)q'(\eta))} \\ &= \frac{\theta'(\eta)}{\theta(\eta)}\left(\frac{(\sigma q(\eta))^2(\psi-\eta)}{(q(\eta)-(\psi-\eta)q'(\eta))^2}\right).\end{aligned}$$

The condition on ψ becomes

$$\begin{aligned}&\frac{a-\underline{a}}{q(\eta)}+\underline{\delta}-\delta+(\sigma+\sigma^q(\eta))\sigma^\theta(\eta)=0 \\ \Longrightarrow\ &\frac{a-\underline{a}}{q(\eta)}+\underline{\delta}-\delta+\frac{\theta'(\eta)}{\theta(\eta)}\left(\frac{(\sigma q(\eta))^2(\psi-\eta)}{(q(\eta)-(\psi-\eta)q'(\eta))^2}\right)=0 \\ \Longrightarrow\ &\theta(\eta)(q(\eta)-(\psi-\eta)q'(\eta))^2(a-\underline{a}+q(\eta)(\underline{\delta}-\delta))+\sigma^2q(\eta)^3(\psi-\eta)\theta'(\eta)=0,\end{aligned}$$

from which we may solve for ψ using the quadratic formula, after performing the change of variable $\hat{\theta}(\eta) = \frac{1}{\theta(\eta)}$.

4.2.3 Other technical details

We explicitly calculate $\underline{q}$. Solving Eq. (3), we find that $\underline{q} = 0.4862$.

We employ several additional techniques when constructing the model to improve its results. To satisfy the boundary conditions, we enforce hard boundary conditions in addition to soft boundary conditions [29]. We construct the solution

$$\begin{aligned}q(\eta) &= (\eta-\eta^*)^2\mathcal{N}_q(\eta)+C, \\ \hat{\theta}(\eta) &= \eta(\eta-\eta^*)^2\mathcal{N}_{\hat{\theta}}(\eta)-(\eta/\eta^*)^2+2\eta/\eta^*,\end{aligned}$$

where C is a trainable variable that represents $q(\eta^*)$ while $\mathcal{N}_q(\eta)$ and $\mathcal{N}_{\hat{\theta}}(\eta)$ are the first and second components of the network output, respectively. This automatically satisfies the boundary conditions $q'(\eta^*) = 0$, $\hat{\theta}(0) = 0$, $\hat{\theta}(\eta^*) = 1$, and $\hat{\theta}'(\eta^*) = 0$. For the final boundary condition $q(0) = \underline{q}$, we use a soft boundary condition.

As discussed, we train two variables $C = q(\eta^*)$ and η^*, and we set initial guesses of $C = 1.0$ and $\eta^* = 0.4$. While the value of C is flexible, the initial guess for η^* comes from experimentation.

Lastly, initial testing suggests that $q(\eta)$ features a steep gradient for small η. As a result, we employ two techniques to capture this gradient better. We first add a feature layer via an input transform [20]. The feature layer used enables us to stretch the input domain, making it easier to capture the sharp increase for small η. Another technique used involves adding an additional ten anchor training points in the interval $[0, 10^{-4}]$.

Next, we want to ensure that q is increasing and θ is decreasing. By the change of variable, θ decreasing is equivalent to $\hat{\theta}$ increasing, so to enforce q increasing and $\hat{\theta}$ increasing, we add loss terms

$$\mathcal{L}_q = \int_0^{\eta^*} (\min(q'(\eta), 0))^2 d\eta \quad \text{and} \quad \mathcal{L}_{\hat{\theta}} = \int_0^{\eta^*} (\min(\hat{\theta}'(\eta), 0))^2 d\eta,$$

representing loss from the mean squared value of $\min(q'(\eta), 0)$ and $\min(\hat{\theta}'(\eta), 0)$. This pushes the functions to have positive derivatives, which would mean they are increasing. We use loss weights of 10^6 for the ODE system, q increasing, and $\hat{\theta}$ increasing losses, and a loss weight of 1 for the boundary condition $q(0) = \underline{q}$. Furthermore, we train with 10^3 training points sampled inside the domain, 2 points sampled on the boundary, and 10^3 points sampled inside the domain for testing.

While training, we use two different learning rate schedulers. The first learning rate scheduler has an initial learning rate of 1×10^{-3} with a decay rate of 0.5 and decay step of 1500. We use this scheduler for the first 1×10^5 iterations. Then, we switch to a learning rate scheduler with initial learning rate 1×10^{-5}, decay rate 0.5, and decay step 1500 for another 2×10^5 iterations.

The training results are displayed in Fig. 5 while the predicted value of η^* is displayed in Table 7. When calculating the L^2 error for these results, we only consider the portion of the graph where $\eta < \eta^*$. This is an accurate measurement since the final result features a prediction for η^* that is very close to the reference value. The prediction for q (Fig. 5A) has an L^2 relative error of 1.14%. The prediction for θ (Fig. 5B) has an L^2 relative error of 0.258%. Finally, the prediction for ψ (Fig. 5C) has an L^2 relative error of 0.273%.

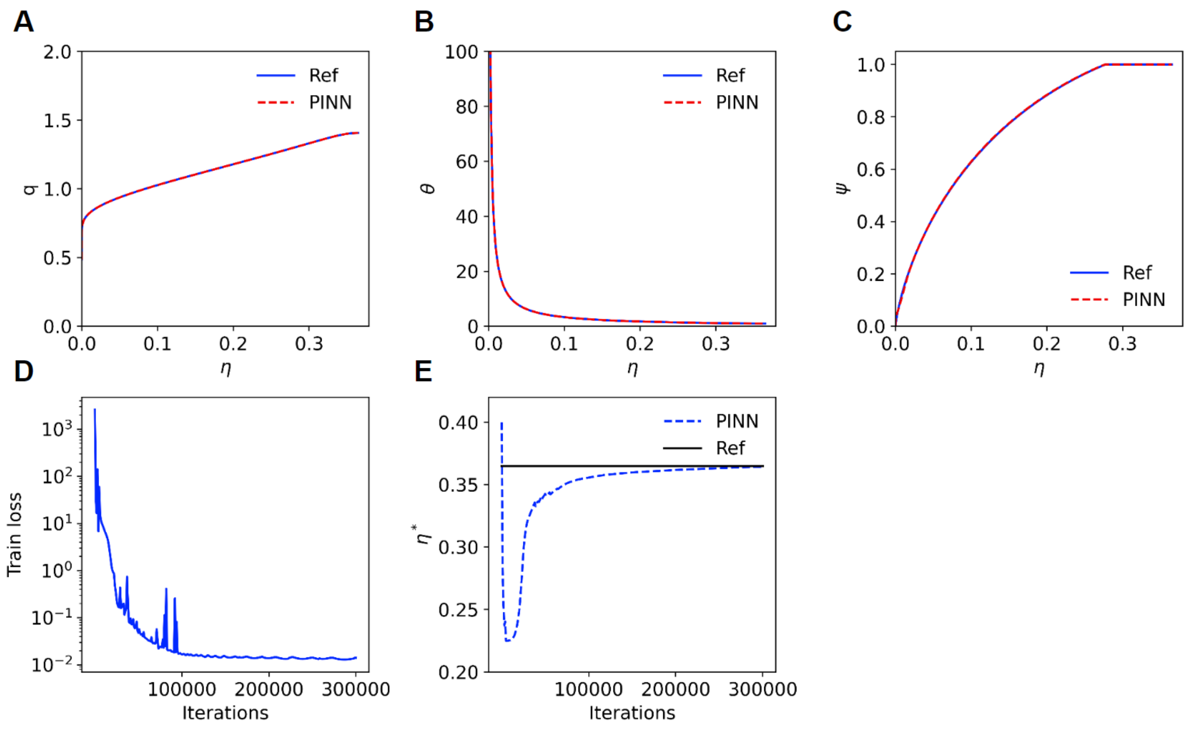

Figure 5: **Example in Section 4.1: PINN results for forward B&S model.** (**A**) The reference value and prediction for q after training. (**B**) The reference value and prediction for θ after training. (**C**) The reference value and prediction for ψ after training. (**D**) The trajectory of train loss throughout training. (**E**) The convergence of η^* throughout training.

Table 7: **Example in Section 4.1: Predicted value of η^* after training.**

Variable	Ref	Prediction
η^*	0.3648	0.3644

4.3 Simultaneously solving and estimating the model

In this section, we simultaneously solve the model and also estimate one or two parameters of the model.

4.3.1 Estimation of one parameter

First, we estimate only one parameter. Specifically, we assume a is unknown and impose a moment condition. There are two ways to match the moment condition.

The first method is simulation. We simply simulate η_t according to its law of motion,

$$\frac{d\eta_t}{\eta_t} = \mu^\eta(\eta_t)dt + \sigma^\eta(\eta_t)dB_t,$$

and then take the average moment value to match the target.

A more efficient method is through the Kolmogorov Forward Equation (KFE). We can solve for the KFE that describes the ODE of the stationary density function of η_t, $f(\eta)$. The equation is

$$-\frac{\partial}{\partial\eta}\left(\mu^\eta(\eta)\eta f(\eta)\right) + \frac{\partial^2}{\partial\eta^2}\left(\frac{1}{2}\sigma^\eta(\eta)^2\eta^2 f(\eta)\right) \equiv \frac{\partial}{\partial\eta}J(\eta) = 0,$$

with boundary condition $-\mu^\eta \eta f(\eta) + \frac{1}{2}\frac{\partial}{\partial\eta}\left(\sigma^\eta(\eta)^2 f(\eta)\right)|_{\eta=\eta^*} = 0$. For simplicity, we define the function Q as

$$Q(\eta) = \frac{2\mu^\eta(\eta)\eta - \frac{\partial}{\partial\eta}(\sigma^\eta(\eta)^2\eta^2)}{\sigma^\eta(\eta)^2\eta^2}.$$

Then, according to Appendix C, the density function can be written as

$$f(\eta) = A\exp\left(-\int_\eta^{\eta^*} Q(\eta')d\eta'\right),$$

where A is the normalization factor

$$\frac{1}{A} = \int_0^{\eta^*} \exp\left(-\int_{\eta'}^{\eta^*} Q(\eta)d\eta\right) d\eta'.$$

We match the moment target

$$a_{\text{target}} = \int_0^{\eta^*} [\psi(\eta)a + (1-\psi(\eta))\underline{a}]f(\eta)d\eta$$

where we define $a_{\text{target}} = 0.1095$. In this paper, we use method 2, employing the KFE.

For the inverse problem, we maintain the same techniques used in the forward problem. While training, we scale a up by 10, and we scale it back down for the results. Furthermore, we restrict η^* to $(0.3, 0.5)$. Also, we use loss weights: 5×10^4 for the ODE on q'', 1×10^4 for the ODE on θ'', 1×10^5 for q increasing, 10^3 for θ decreasing, 10^4 for the a_{target} boundary condition, and 10^0 for the $q(0)=\underline{q}$ boundary condition. We train with $2^{13}-2$ points in the domain, 2 points on the boundary, and 2^{13} points for testing. Lastly, with the learning rate scheduler, we have 1×10^{-4} as the initial learning rate, 1.0 as the decay rate, and 2000 as the decay step.

The detailed training results for one trial is displayed in Fig. 6. The figure shows that the PINN achieves accurate estimations of q, θ, ψ, and a. Furthermore, running several trials, we find that the solution consistently converges to the global minimum no matter the initial guess of a. To demonstrate this consistent convergence, we run ten trials in which we randomly select a from the range $(0.06, 0.16)$. The parameter a is accurately estimated, and the final value is $a = 0.1099 \pm 0.0007$ (Table 8) with L^2 errors of less than 1% for q, θ, and ψ.

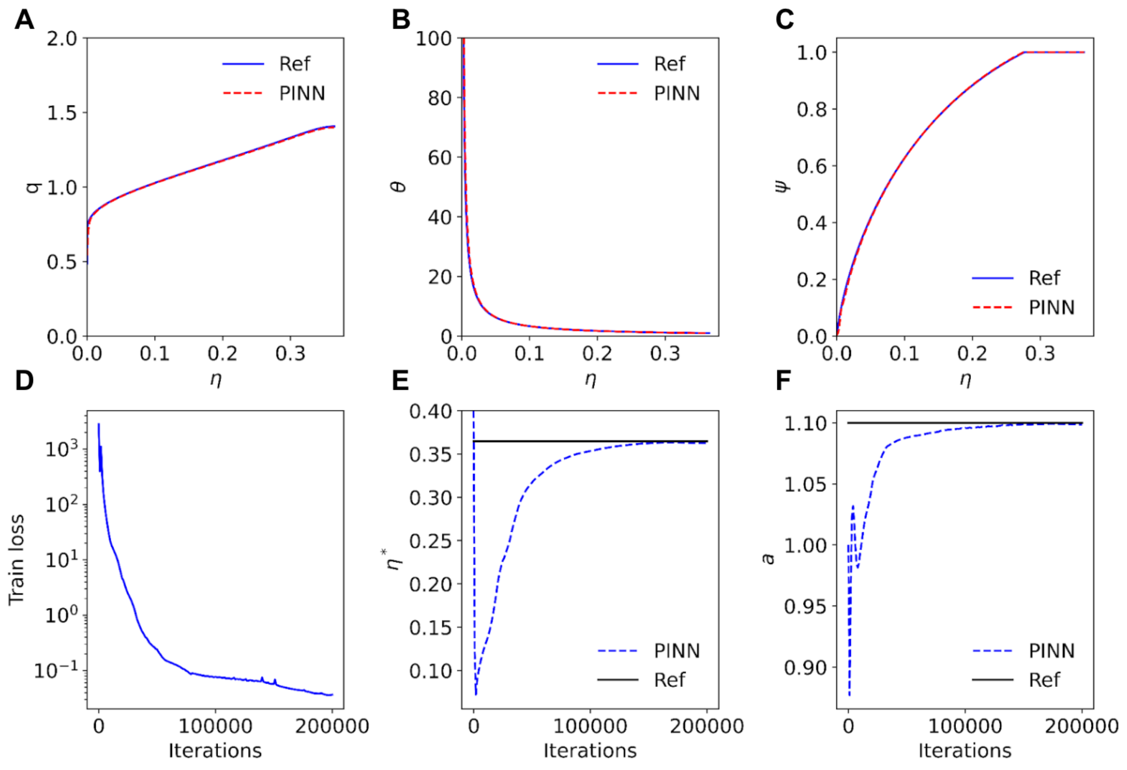

Figure 6: **Example in Section 4.3.1: PINN results for inverse B&S model with one unknown parameter.** (**A**) The reference value and prediction for q. (**B**) The reference value and prediction for θ. (**C**) The reference value and prediction for ψ. (**D**) The trajectory of train loss throughout training. (**E**) The convergence of η^* throughout training. (**F**) The convergence of a throughout training.

Table 8: **Example in Section 4.3.1: Results from several trials of inverse B&S model with one unknown parameter.** The choices of a are random in the range $(0.06, 0.16)$, and the table depicts the final results for each trial.

Trial	a		a_{target}		L^2 relative errors		
	Initial	Final	Value	Error	q	θ	ψ
1	0.1500	0.1098	0.1093	0.16%	0.17%	0.53%	0.06%
2	0.0921	0.1098	0.1093	0.22%	0.22%	0.69%	0.08%
3	0.1472	0.1100	0.1095	0.01%	0.01%	0.02%	0.00%
4	0.1230	0.1099	0.1094	0.11%	0.12%	0.36%	0.04%
5	0.1236	0.1098	0.1094	0.14%	0.14%	0.45%	0.05%
6	0.1005	0.1098	0.1093	0.22%	0.23%	0.71%	0.09%
7	0.1509	0.1099	0.1094	0.08%	0.08%	0.26%	0.03%
8	0.1415	0.1099	0.1094	0.07%	0.07%	0.21%	0.03%
9	0.0889	0.1099	0.1094	0.09%	0.09%	0.29%	0.03%
10	0.0645	0.1098	0.1093	0.15%	0.16%	0.49%	0.06%

4.3.2 Estimation of two parameters

Now, we estimate parameters a and σ. We define our moment conditions:

$$a_{\text{target}} = \int_0^{\eta^*} [\psi(\eta)a + (1-\psi(\eta))\underline{a}]f(\eta)d\eta \quad \text{and} \quad \text{Vol}_{\text{target}} = \frac{\sqrt{\int_0^{\eta^*}(q-\bar{q})^2 f(\eta)d\eta}}{\bar{q}},$$

where $\bar{q}$ is defined as $\int_0^{\eta^*} q(\eta)f(\eta)d\eta$, $a_{\text{target}} = 0.1095$, and $\text{Vol}_{\text{target}} = 0.04126$. Again, we employ the KFE.

To implement the integrals in the moment conditions, we use numerical integration. When creating the model, we use the same change of variable for θ. However, instead of explicitly solving for ψ, noting that the left hand side of equation (4) is monotonic, we utilize a basic bisection algorithm provided by the Python library scipy to solve for ψ. While training, we scale a up by 10 and σ up by 100, and we scale them back down for the results. Once again, we restrict η^* to $(0.3, 0.5)$. We also use loss weights: 10^6 for the ODE on q'', 2×10^5 for the ODE on θ'', 10^6 for q increasing, 10^4 for θ decreasing, 10^5 for the a_{target} moment condition, 5×10^3 for the $\text{Vol}_{\text{target}}$ moment condition, and 10^1 for the $q(0) = \underline{q}$ boundary condition. Furthermore, we train with $2^{13}-2$ training points sampled inside the domain, 2 points sampled on the boundary, and 2^{13} points sampled inside the domain for testing. Lastly, with the learning rate scheduler, we have 5×10^{-4} as the initial learning rate, 1.0 as the decay rate, and 1500 as the decay step. We train this model for 1.5×10^5 iterations.

With our model, we find that the convergence and final values are sensitive to the initial guess of σ but not to the initial guess of a. Furthermore, we find that the prediction of a is consistently accurate, regardless of the initial guess. Although the predictions of σ are less accurate, there are initial guesses that result in accurate predictions of σ. An example is demonstrated in Fig. 7. In this example, the train loss decreases steadily (Fig. 7D). Furthermore, with the final predicted values for a and σ, the predictions of a_{target} and $\text{Vol}_{\text{target}}$ are very close to the true values (Figs. 7E and F). Similarly, the graphs of q, θ, and ψ are very close to their true values (Figs. 7A, B, and C).

To illustrate this convergence, the trajectories of five trials that converge to the global minimum are displayed in Fig. 8A. Furthermore, the trajectories of five trials that converge to a local minimum instead are displayed in Fig. 8B.

In practice, trials that converge to a local minimum can be identified because a_{target} and $\text{Vol}_{\text{target}}$ are farther from the true values (Table 9). We consider the results with three runs of ten random trials each, in which we select a and σ at random in the ranges $(0.6, 1.6)$ and $(0.01, 0.04)$. We generally receive satisfactory results (Fig. 9A). For the next runs, we run five random trials and then apply Bayesian optimization to determine the next five initial guesses. Bayesian optimization is a method that uses Bayes' theorem to maximize a continuous function without information about derivatives [30]. Through this, we find that Bayesian optimization consistently produces better results than random trials (Fig. 9B).

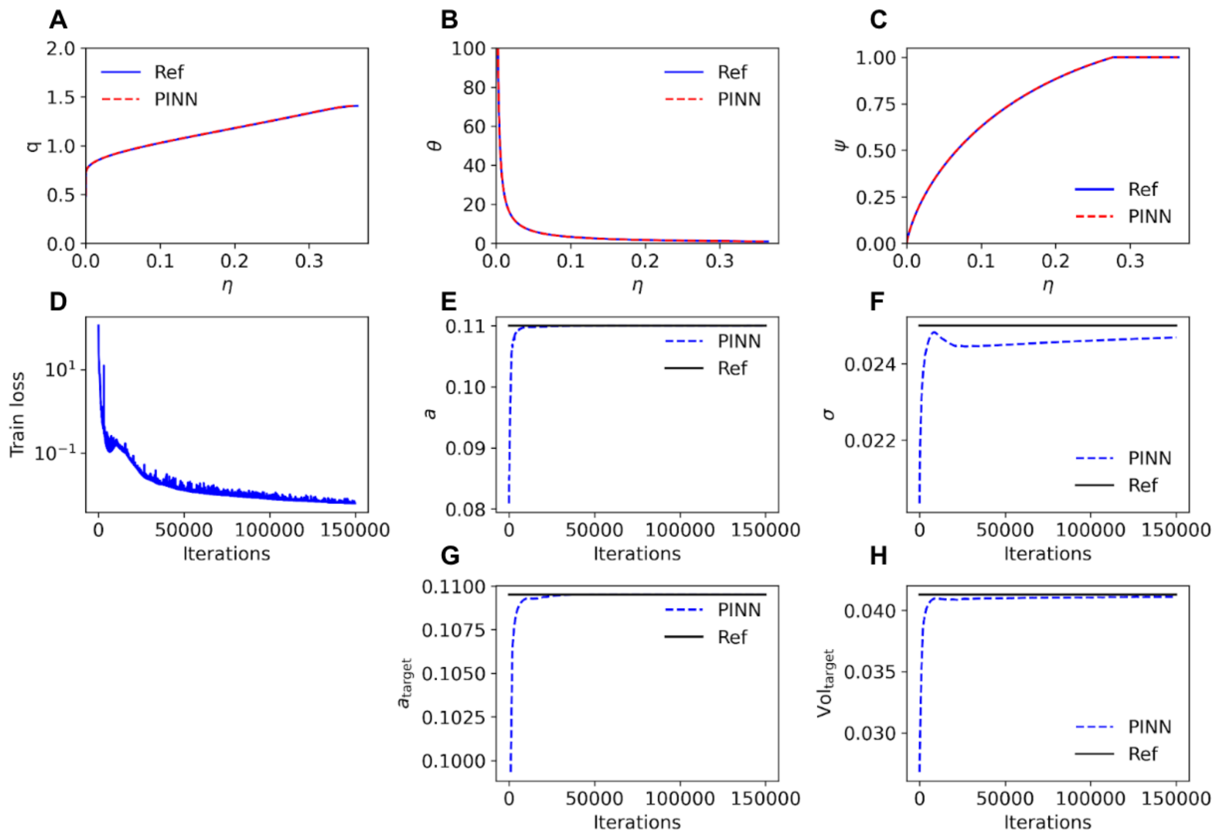

Figure 7: **Example in Section 4.3.2: PINN results for inverse B&S model with two unknown parameters.** (**A**) The reference value and prediction for q. (**B**) The reference value and prediction for θ. (**C**) The reference value and prediction for ψ. (**D**) The trajectory of train loss throughout training. (**E**) The convergence of a throughout training. (**F**) The convergence of σ throughout training. (**G**) The trajectory of a_{target} throughout training. (**H**) The trajectory of $\text{Vol}_{\text{target}}$ throughout training.

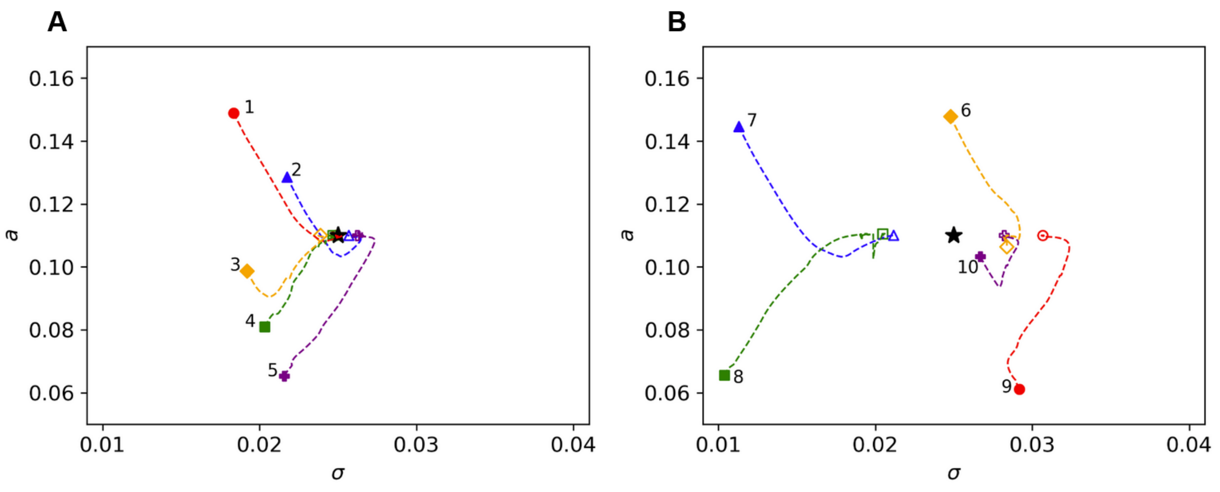

Figure 8: **Example in Section 4.3.2: Trajectories in the parameter space** (σ, a)**.** The black star represents the true value (σ, a), the shaded-in shape represents the initial guess, the empty shape represents the final estimation, and the path connecting them is the trajectory. (**A**) Five trajectories that converged to the global minimum. (**B**) Five trajectories that converged to a local minimum.

Table 9: **Example in Section 4.3.2: PINN parameter estimation from different initial values of a and σ.** Trials 1–5 converge to the global minimum. Trials 6–10 converge to a local minimum.

Trial	Initial values		Final values		a_{target}		$\text{Vol}_{\text{target}}$		L^2 relative errors		
	a	σ	a	σ	Value	Error	Value	Error	q	θ	ψ
1	0.1489	0.0183	0.1100	0.0249	0.1095	0.00%	0.0412	0.19%	0.06%	0.10%	0.05%
2	0.1286	0.0217	0.1100	0.0257	0.1095	0.01%	0.0416	0.88%	0.33%	0.56%	0.22%
3	0.0809	0.0203	0.1100	0.0247	0.1095	0.01%	0.0411	0.37%	0.15%	0.26%	0.10%
4	0.0987	0.0192	0.1100	0.0239	0.1095	0.01%	0.0407	1.43%	0.55%	0.87%	0.37%
5	0.0653	0.0216	0.1099	0.0262	0.1094	0.07%	0.0419	1.47%	0.65%	1.20%	0.38%
6	0.0612	0.0292	0.1100	0.0307	0.1096	0.04%	0.0392	5.00%	1.96%	2.79%	1.31%
7	0.1446	0.0113	0.1100	0.0212	0.1100	0.42%	0.0391	5.23%	2.71%	4.45%	1.41%
8	0.0655	0.0104	0.1104	0.0205	0.1059	3.32%	0.0408	1.23%	4.71%	12.35%	0.35%
9	0.1478	0.0248	0.1064	0.0284	0.1095	0.03%	0.0429	4.05%	1.50%	2.71%	1.03%
10	0.1033	0.0267	0.1100	0.0282	0.1094	0.06%	0.0442	7.09%	2.60%	4.97%	1.79%

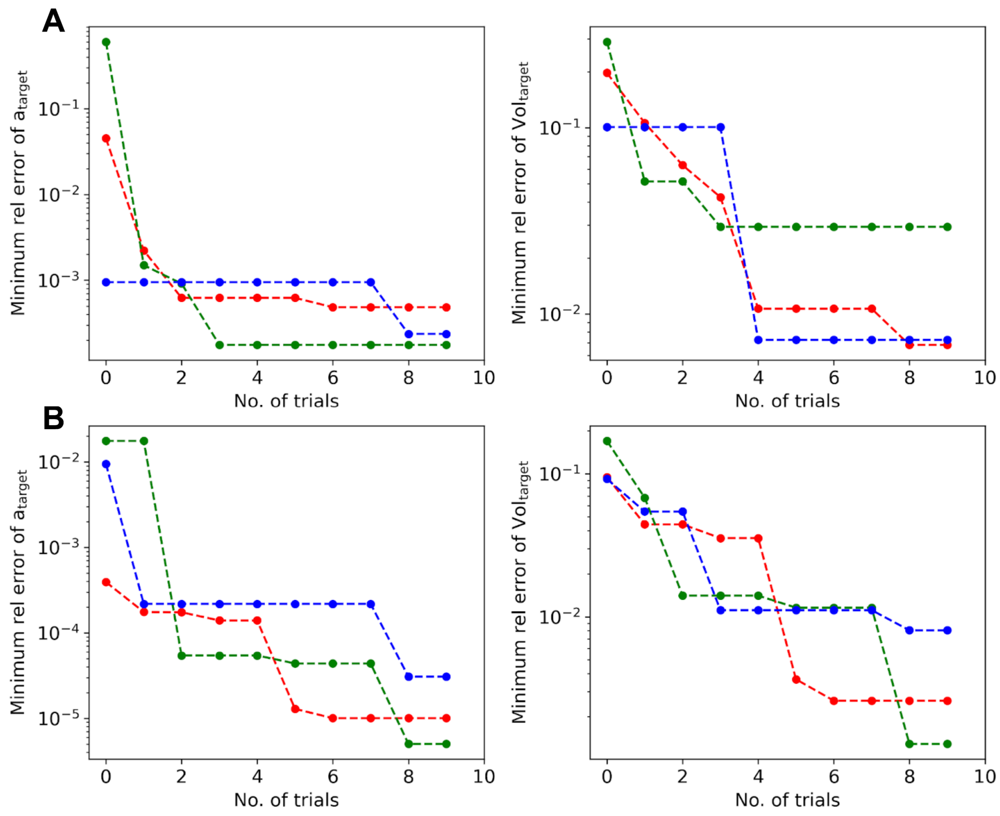

Figure 9: **Example in Section 4.3.2: PINN results from varying initial guesses using Bayesian optimization.** The results from three runs of ten initial guesses, plotting the minimum relative error of a_{target} and $\text{Vol}_{\text{target}}$ so far. (**A**) The initial guesses of a and σ are chosen randomly from the intervals $(0.06, 0.16)$ and $(0.01, 0.04)$, respectively. (**B**) The first five initial guesses of a and σ are chosen once again at random from the intervals $(0.06, 0.16)$ and $(0.01, 0.04)$, respectively. The five guesses afterwards are chosen using Bayesian optimization.

5 Conclusion

In this paper, we proposed a new method for solving economics models with deep learning through physics-informed neural networks (PINNs). We demonstrated the advantages of this method: generality, simultaneous solution and estimation, leveraging the state-of-art machine-learning techniques, and handling large state space. Our proposed method provides a general framework for solving PDEs, whereas in traditional methods, we usually need to design different algorithms for different problems. Furthermore, the inverse problem can be solved very easily and is no more than adding an extra loss term for additional information. We showed the effectiveness of the PINN method in solving two models, each featuring its own challenges.

There are a couple of directions for future research. First, currently we only allow for heterogeneous agent with stationary population distribution. Our method can be generalized to deal with mean-field games where the distribution is a state variable that changes over time. Second, we currently only solve exactly identified estimation problems where the number of moment conditions is equal to the number of parameters to be estimated. A more generalized method should be able to handle overidentified systems and provide error bounds on the estimated coefficients, similar to the classical GMM estimations. Finally, our method can be applied to a broader variety of economics and finance problems.

Acknowledgements

Benjamin Fan, Edward Qiao, and Lu Lu thank the MIT PRIMES program, under which this research was conducted.

References

[1] Zhiguo He and Arvind Krishnamurthy. Intermediary asset pricing. *The American Economic Review*, 103(2):732–770, 2013.

[2] Markus K. Brunnermeier and Yuliy Sannikov. A macroeconomic model with a financial sector. *American Economic Review*, 104(2):379–421, 2014.

[3] Itamar Drechsler, Alexi Savov, and Philipp Schnabl. A model of monetary policy and risk premia. *Journal of Finance*, 73(1):317–373, 2018.

[4] Javier Bianchi and Saki Bigio. Banks, liquidity management and monetary policy. *NBER Working Paper No. 20490*, 2018.

[5] Mark Gertler and Nobuhiro Kiyotaki. Banking, liquidity, and bank runs in an infinite horizon economy. *The American Economic Review*, 105(7):2011–2043, 2015.

[6] Ji Huang. Banking and shadow banking. *Journal of Economic Theory*, 178:124–152, 2018.

[7] Sebastian Di Tella. Optimal regulation of financial intermediaries. *American Economic Review*, 109(1):271–313, 2019.

[8] Arvind Krishnamurthy and Wenhao Li. Dissecting mechanisms of financial crises: Intermediation and sentiment. Technical report, National Bureau of Economic Research, 2020.

[9] Peter Maxted. A macro-finance model with sentiment. 2020.

[10] Joao Gomes, Leonid Kogan, and Lu Zhang. Equilibrium cross section of returns. *Journal of Political Economy*, 111(4):693–732, 2003.

[11] Toni M Whited and Guojun Wu. Financial constraints risk. *The review of financial studies*, 19(2):531–559, 2006.

[12] Christopher A Hennessy and Toni M Whited. How costly is external financing? evidence from a structural estimation. *The Journal of Finance*, 62(4):1705–1745, 2007.

[13] Gregor Matvos and Amit Seru. Resource allocation within firms and financial market dislocation: Evidence from diversified conglomerates. *The Review of Financial Studies*, 27(4):1143–1189, 2014.

[14] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[15] Jesus Fernandez-Villaverde, Galo Nuno, George Sorg-Langhans, and Maximilian Vogler. Solving high-dimensional dynamic programming problems using deep learning. *Unpublished working paper*, 2020.

[16] Hui Chen, Antoine Didisheim, and Simon Scheidegger. Deep structural estimation: With an application to option pricing. *arXiv preprint arXiv:2102.09209*, 2021.

[17] Victor Duarte. Machine learning for continuous-time economics. *Available at SSRN 3012602*, 2018.

[18] Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *arXiv preprint arXiv:2111.02801*, 2021.

[19] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8):11618–11633, 2020.

[20] Alireza Yazdani, Lu Lu, Maziar Raissi, and George Em Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS Computational Biology*, 16(11):e1007575, 2020.

[21] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[22] A. M. Tartakovsky, C. Ortiz Marrero, Paris Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano. Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56(5):e2019WR026731, 2020.

[23] Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R. Witschey, John A. Detre, and Paris Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, 2020.

[24] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E. Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8, 2020.

[25] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

[27] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[28] M. Raissi, P. Perdikaris, and G.E. Karniadakis. 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, 2019.

[29] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[30] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. page 2951–2959, 2012.

[31] H Risken. *The Fokker-Planck equation: Methods of solution and applications*, volume 18. Springer, 1996.

[32] Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*, volume 113, *Graduate Texts in Mathematics.* Springer, 1998.

[33] Xavier Gabaix, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll. The dynamics of inequality. *Econometrica*, 84(6):2071–2111, 2016.

[34] Grigorios A Pavliotis. *Stochastic processes and applications: diffusion processes, the Fokker-Planck and Langevin equations*, volume 60. Springer, 2014.

Appendices

A Setups for the firm industrial dynamic model

Proposition 1. *In bank industrial dynamic models, stationary distribution exists.*

Proof. As the volatility $\sigma(z)$ is constant, and Ornstein–Uhlenbeck process is recurrent, we only need to check if the linear growth condition for (μ_e, μ_z) is satisfied:

$$\begin{aligned}||\vec{\mu}(e,z)|| \leq ||\mu_e(e,z)|| + ||\mu_z(e,z)|| &= ||\theta_0(z-z_m)|| + ||\pi^*(e,z) - \mathbf{1}_{(e_t,z_t)\in\mathscr{C}^c}\kappa(\phi e - f(z,l^*))^+|| \\ &\leq ||\theta_0(z-z_m)|| + ||\pi^*(e,z)|| + ||\kappa(\phi e - f(z,l^*))^+|| \\ &\leq ||\theta_0(z-z_m)|| + ||\pi^*(e,z)|| + |\kappa|(||(\phi e)|| + ||f(z,l^*)||) \\ &\leq ||\theta_0(z-z_m)|| + ||\pi^*(e,z)|| + |\kappa|(|\phi||x| + ||f(z,l^*)||)\end{aligned}$$

We can conclude our proof by noting that both instantaneous profit $\pi^*(e,z)$, and deposit production $f(z,l^*)$ are scaled by productivity z. □

B Setup and derivations for the macroeconomic model with a financial sector

The model in [2] considers an economy populated with one unit of experts (indexed by $i, i \in \mathbb{I} = [0,1]$) and one unit of households (indexed by $j, j \in \mathbb{J} = [1,2]$), in an infinite horizon setup. Time is continuous here. The physical capital k_t held by experts produces output at rate $y_t = ak_t$, while households produces at rate $\underline{y}_t = \underline{a}k_t$. Denoting the investment rate per unit of capital as $\iota_t, \underline{\iota}_t$ for experts and households respectively (i.e., $\iota_t k_t$ is the total investment rate of experts), the corresponding capital evolves as

$$\begin{cases} dk_t = (\Phi(\iota_t) - \delta)k_t dt + \sigma k_t dZ_t, \\ d\underline{k}_t = (\Phi(\underline{\iota}_t) - \underline{\delta})\underline{k}_t dt + \sigma \underline{k}_t dZ_t, \end{cases}$$

where Z_t here is exogenous aggregate Brownian shocks, $\Phi(\cdot)$ is standard investment technology with convex adjustment costs, i.e., $\Phi(0) = 0, \Phi'(0) = 1, \Phi'(\cdot) > 0$ and $\Phi''(\cdot) < 0$, $\delta, \underline{\delta}$ are depreciation rate (assume $\underline{\delta} > \delta$), and σ is the volatility. Experts and households are risk neutral, households have discount rate r and they may have positive or negative consumption, which means households provide fully elastic lending at risk-free rate r. The equilibrium market price of capital is postulated as a GBM:

$$dq_t = \mu_t^q q_t dt + \sigma_t^q q_t dZ_t,$$

where μ_t can be viewed as the time dependent price drift and σ_t^q is the price volatility. Both μ_t^q and σ_t^q are determined by market equilibrium. We focus on the case that equilibrium price $q \in [\underline{q}, \overline{q}]$, where $\underline{q} = \max_\iota \frac{\underline{a}-\iota}{r-(\Phi(\iota)-\underline{\delta})}$ and $q = \max_\iota \frac{a-\iota}{r-(\Phi(\iota)-\delta)}$. The return of the capital managed by experts and households can then be expressed by

$$\begin{aligned} dr_t^k &= \frac{dD_t + d(k_t q_t)}{k_t q_t} = \frac{(a-\iota_t)\ \cancel{k_t}}{\cancel{k_t} q_t} + (\Phi(\iota_t) - \delta + \mu_t^q + \sigma\sigma_t^q)\, dt + (\sigma + \sigma_t^q) dZ_t, \\ d\underline{r}_t^k &= \frac{d\underline{D}_t + d(k_t q_t)}{k_t q_t} = \frac{(\underline{a}-\underline{\iota}_t)\ \cancel{k_t}}{\cancel{k_t} q_t} + (\Phi(\underline{\iota}_t) - \underline{\delta} + \mu_t^q + \sigma\sigma_t^q)\, dt + (\sigma + \sigma_t^q) dZ_t. \end{aligned}$$

The cumulative consumption of a household and an expert are denoted as $\underline{c}_t$ and c_t, respectively. Then utilities are given by

$$\mathbb{E}\left[\int_0^\infty e^{-rt} d\underline{c}_t\right] \text{ (households)} \quad \text{and} \quad \mathbb{E}\left[\int_0^\infty e^{-\rho t} dc_t\right] \text{ (experts).}$$

The net worth of an expert n_t evolves as

$$\frac{dn_t}{n_t} = x_t dr_t^k + (1 - x_t) r dt - \frac{dc_t}{n_t},$$

where x_t is the fraction of capital. the first part is return on risky assets, the second part is return on safe assets, and the third part is consumption. Similarly, for households, we have

$$\frac{d\underline{n}_t}{\underline{n}_t} = \underline{x}_t d\underline{r}_t^k + (1 - \underline{x}_t) r dt - \frac{d\underline{c}_t}{\underline{n}_t},$$

where $\underline{x}_t$ is the fraction of capital. Both households and experts maximize their utility. Households can have negative consumption while experts cannot, i.e., $dc_t \geq 0$. Households and experts' problems can be written as

$$\max_{\underline{x}_t \geq 0, d\underline{c}_t, \underline{\iota}_t} \mathbb{E}\left[\int_0^\infty e^{-rt} d\underline{c}_t\right] \text{ (households)}, \quad \max_{x_t \geq 0, dc_t \geq 0, \iota_t} \mathbb{E}\left[\int_0^\infty e^{-\rho t} dc_t\right] \text{ (experts),}$$

subject to net worth's equation of motion.

Definition for equilibrium. *Given initial wealth distribution, k_0^i, k_0^j, an equilibrium is described by the stochastic process $\{q_t, n_t^i, \underline{n}_t^i \geq 0, n_t^i, \underline{k}_t^i \geq 0, \iota_t^i, \underline{\iota}_t^i, dc_t^i \geq 0, d\underline{c}_t^i\}$ such that: (1) initial net worth is $n_0^i = k_0^i q_0, \underline{n}_0^j = k_0^j q_0$; (2) each expert and agent solves their problems, given capital price q_t; (3) markets for consumption goods and capital are clear, i.e.,*

$$\int_{\mathbb{I}} dc_t^i di + \int_{\mathbb{J}} d\underline{c}_t^j dj = \int_{\mathbb{I}} (a - \iota_t^i) k_t^i di + \int_{\mathbb{J}} (\underline{a} - \underline{\iota}_t^j) \underline{k}_t^j dj,$$

$$\int_{\mathbb{I}} k_t^i di + \int_{\mathbb{J}} \underline{k}_t^j dj = K_t, dK_t \textit{ is: } \left(\int_{\mathbb{I}} (\Phi(\iota_t^i)) - \delta) k_t^i di + \int_{\mathbb{J}} (\Phi(\underline{\iota}_t^i)) - \underline{\delta}) \underline{k}_t^j dj\right) dt + \sigma K_t dZ_t.$$

Solution. First, households and experts' investment choices $\iota_t, \underline{\iota}_t$ are solved by maximizing $dr_t^k, d\underline{r}_t^k$, which means

$$\iota_t, \underline{\iota}_t \in \arg\max_{\iota} \Phi(\iota) - \iota/q_t \Rightarrow \iota_t = \underline{\iota}_t = \Phi'^{-1}(1/q_t).$$

Second, denoting ψ_t as the fraction of capital held by experts, we are led to equilibrium condition $\mathbb{E}_t[d\underline{r}_t^k]/dt \leq r$, with equality if $1 - \psi_t > 0$. This condition means that when expected return is less than risk-free asset, households will not hold any capital. Also, risk-neutral households will hold a fraction of capital when the return equals the risk-free rate.

Third, to solve the experts' problem, we introduce multiplier θ_t for experts' future utility, i.e., $\theta_t n_t \equiv \mathbb{E}_t\left[\int_0^\infty e^{-\rho(s-t)} dc_s\right]$. Introducing unit worth's consumption $d\zeta_t$, the experts optimal trading strategy is

$$\rho \theta_t n_t = \max_{\hat{x}_t, d\zeta_t} n_t d\zeta_t + \mathbb{E} d[\theta_t n_t]$$

Considering a finite process $d\theta_t/\theta_t = \mu_t^\theta dt + \sigma_t^\theta dZ_t$, for optimal strategy, the solution of θ_t features: (1) it is always true that $\theta_t \geq 1$, $d\zeta_t > 0$ only when $\theta_t = 1$; (2) $\mu_t = \rho - r$; (3) either $x_t > 0$ when $\mathbb{E}_t[dr_t^k]/dt - r = -\sigma_t^\theta(\sigma + \sigma_t^q)$ (the risk premium), or $x_t = 0$ when $\mathbb{E}_t[dr_t^k]/dt - r < -\sigma_t^\theta(\sigma + \sigma_t^q)$.

Denote the experts' wealth share as $\eta_t \equiv \frac{N_t}{q_t K_t} \in [0, 1]$, where all functions can be functions of η_t. By applying Itô's lemma, we get ($\langle \cdot, \cdot \rangle$ is the quadratic variation)

$$d\eta_t = \frac{dN_t}{q_t K_t} + N_t d\left(\frac{1}{q_t K_t}\right) + \left\langle dN_t, d\left(\frac{1}{q_t K_t}\right)\right\rangle \equiv \eta_t(\mu_t^\eta dt + \sigma_t^\eta dZ_t)$$

$$\Rightarrow \mu_t^\eta = -\sigma_t^\eta(\sigma + \sigma_t^q + \sigma_t^\theta) + \frac{a - \iota(q_t)}{q_t} + (1 - \psi_t)(\underline{\delta} - \delta), \ \sigma_t^\eta = \frac{\psi_t - \eta_t}{\eta_t}(\sigma + \sigma_t^q).$$

Equilibrium conditions. Optimal strategies of households and experts imply

$$\frac{\mathbb{E}_t[dr_t^k - d\underline{r}_t^k]}{dt} - \sigma_t^\theta(\sigma + \sigma_t^q) = 0 \Rightarrow \psi(\eta),$$

which means for experts, the opportunity cost of holding capital is the risk premium, if the equilibrium has an interior solution. When the implied solution $\psi > 1$, the above equation does not necessarily hold, as it is always profit profitable to hold capital in this case. Next, by Itô's formula, we can solve the equilibrium price $q(\eta)$ and multiplier θ_t from

$$\begin{cases} \mu_t^q q(\eta) = q'(\eta)\mu_t^\eta \eta + \frac{1}{2}(\sigma_t^\eta)^2\eta^2 q''(\eta), \\ \mu_t^\theta \theta(\eta) = \theta'(\eta)\mu_t^\eta \eta + \frac{1}{2}(\sigma_t^\eta)^2\eta^2 \theta''(\eta). \end{cases}$$

Given the stochastic process of $d\eta_t = \eta_t \mu_t^\eta dt + \eta_t \sigma_t^\eta$, the distribution $f(\eta, t)$ evolves as

$$\frac{\partial}{\partial t} f(\eta, t) = -\frac{\partial}{\partial \eta}\left(\mu^\eta f(\eta, t)\right) + \frac{1}{2}\frac{\partial^2}{\partial \eta^2}\left(\sigma_\eta^2(\eta) f(\eta, t)\right).$$

Prior to solving the invariant distribution, we need to show that the distribution function exists[3].

Proposition 2. *In [2], the stationary distribution exists, if* $2(\rho - r)\sigma^2 < \Lambda^2, \Lambda = \frac{\mathbb{E}_t[dr_t^k - d\underline{r}_t^k]}{dt} = \frac{a - \underline{a}}{\underline{q}} - (\delta - \underline{\delta})$.

Proof. Stochastic process η_t's recurrence is equivalent to the inequality, $\mu^\eta > \frac{(\sigma^\eta)^2}{2}$, when $\eta = 0^+$ (see one dimensional case in [31]). Asymptotic ansatz when $\eta \to 0$ (in the online appendix of [2]): $\mu_t^\eta = \hat{\mu} + o(1), \sigma_t^\eta = \hat{\sigma} + o(1), \psi(\eta) = C_\psi \eta + o(\eta), q(\eta) = \underline{q} + C_q \eta^\alpha + o(\eta^\alpha), \theta(\eta) = C_\theta \eta^{-\beta} + o(\eta^{-\beta})$ $(\alpha, \beta > 0)$. By plugging into equilibrium condition, we have $\hat{\sigma} = \Lambda/\beta\sigma$. From the equations for $q(\eta)$ and $\theta(\eta)$, we find

$$2\frac{(\rho - r)\beta^2\sigma^2}{\Lambda^2} = -\beta\frac{2\hat{\mu}}{\hat{\sigma}^2} + \beta(\beta + 1) \to \frac{2\hat{\mu}}{\hat{\sigma}^2} = \beta + 1 - \frac{2(\rho - r)}{\Lambda^2}\sigma^2\beta.$$

□

C Order reduction in forward equation

This section is a technical note. We first discuss the structure of Kolmogorov Forward Equation (KFE), and then give a continuity equation's interpretation of it. Denote the density function as f; the KFE can be heuristically written as

$$\frac{\partial}{\partial t} f = -\sum_i \frac{\partial}{\partial x_i}\left(\mu_i(x) f\right) + \sum_{i,j} \frac{\partial^2}{\partial x_i \partial x_j}\left(\left(\sigma^2\right)_{ij}(x) f\right) \equiv \hat{L}^* f = -\nabla \cdot \vec{J},$$

where $\hat{L}^*$ is the Kolmogorov Forward Operator and the *density flux* $\vec{J}$ is defined as

$$J_i = \frac{\partial}{\partial x_i}\left(\mu_i(x) f\right) - \sum_j \frac{\partial^2}{\partial x_i \partial x_j}\left(\left(\sigma^2\right)_{ij}(x) f\right), \quad \text{for the } i\text{–th column of } \vec{J}.$$

To study the stationary distribution we are interested in, we assume the existence of stationary distribution, which means the following assumption holds.

Assumption 1. *Regularity assumptions in 32, Ch. 5.*

[3]For example, consider a Geometric Brownian Motion: $dX_t = \mu X_t dt + \sigma X_t dZ_t$ with reflecting boundary at $0, D$, the stationary distribution solved from Appendix C is: $f(x) = \frac{\frac{2\mu}{\sigma^2} - 1}{D^{\frac{2\mu}{\sigma^2} - 1}} x^{\frac{2\mu}{\sigma^2} - 2} \times \mathbf{1}_{x \in [0, D]}$, we can see that it cannot be an invariant distribution when $\frac{2\mu}{\sigma^2} - 1 < 0$, as density is negative.

1. *$\sigma^2(x)$ is uniform elliptic, i.e.*
$$\vec{y}^T[\sigma^2(x)]\vec{y} \geq \alpha|\vec{y}|^2, \ \forall \vec{y} \in \mathbb{R}^n.$$
2. *Coefficients are smooth and satisfy linear growth conditions*
$$\exists M \in \mathbb{R}, \ s.t. \ ||\sigma^2(x)|| \leq M, \ ||a(x)|| \leq M(1+||x||), \ ||b(x)|| \leq M(1+||x||),$$
where $a(x), b(x)$ are defined as
$$a_j(x) = -\mu_j(x) + \sum_{i=1}^{n} \partial_{x_i}[\sigma^2(x)]_{ij},$$
$$b_j(x) = \frac{1}{2}\sum_{i,k} \partial^2_{x_i,x_j}[\sigma^2(x)]_{i,k} - \sum_i \partial_{x_i}\mu_i(x).$$
3. *The stochastic process X_t is **recurrent**.*

For the reflecting boundary[4] in our problem, we have $\vec{J}\cdot\hat{n} \equiv 0$ at $\partial\Omega$. The **solution determination problem** can be formally written as

$$\begin{cases} \nabla \cdot \vec{J} = 0, \\ \vec{J}\cdot\hat{n}|_{\partial\Omega} = 0, \\ J_i = -\mu_i(x) f + \sum_j \frac{1}{2}\frac{\partial}{\partial x_j}((\sigma^2(x))_{ij} f), \end{cases}$$

with normalization condition $\int_\Omega f dV = 1$.

Proposition 3. *Under assumption 1, the solution determination problem is equivalent to the problem*

$$\vec{J} = 0 \ and \ \int_\Omega f dV = 1.$$

Proof. We first show that in the one dimensional case[5], the probability flux is always zero under the reflecting boundary condition. This is because

$$J(x) \equiv -\mu(x)f(x) + \frac{1}{2}\frac{d}{dx}\left(\sigma^2(x)f(x)\right) = \int_{\underline{x}}^{x} \hat{L}^* f(x')dx' + J(\underline{x}) = 0,$$

where $\underline{x}$ is the lower boundary.

In higher dimensional cases, $\sigma^2(x)$ is the covariance matrix and $\vec{\mu}$ is the drift vector. According to [34], define $Q(x)$ as: $\vec{Q}(x) = \left(\sigma^2(x)\right)^{-1}(2\vec{\mu}(x) - \nabla\sigma^2(x))$, where $\nabla\sigma^2(x)$ is defined as $\sum_{i,j}\frac{\partial}{\partial x_j}\sigma^2(x)_{ij}\vec{e}_i$. Then[6] $\vec{J} \equiv 0$ **if and only if** $\frac{\partial Q_j}{\partial x_i} = \frac{\partial Q_i}{\partial x_j}$, for all $x \in \Omega$. The density f can be solved as

$$f(\vec{x}) = A\exp\left(-\frac{1}{2}\int_{x_0\in\partial\Omega}^{\vec{x}} \vec{Q}\cdot d\vec{x}\right) \quad \text{and} \quad \frac{1}{A} = \int_\Omega \exp\left(-\int_{x_0\in\partial\Omega}^{x} \vec{Q}(\vec{x}')\cdot d\vec{x}'\right) d^n\boldsymbol{x}.$$

The above formula implies that once we obtain the solution for the KFE with reflecting boundary, we can conclude the solution is **unique** if assumption 1 holds here. □

[4]Indifference condition for value function at the boundary implies the reflecting boundary for distribution. Usually three boundary conditions are considered. They are: (1) *refecting boundary* $\vec{J}\cdot\hat{n}|_{\partial\Omega} = 0$; (2) *absorbing boundary*: $f|_{\partial\Omega} = 0$; (3) *periodic boundary*: $\vec{J}|_{x=a} = \vec{J}|_{x=b}$.

[5]Also see in [33].

[6]This condition implies that the path integral exists, or $d\vec{Q}$ is integrable. Intuitively, for a constant $\sigma^2(x)$, this condition means that the drift term is *curl-free*. Accordingly, a counter example in 2D can be constructed as: $\vec{\mu} = \frac{A}{r}\hat{\tau}$, where $\hat{\tau} = \frac{-y}{\sqrt{x^2+y^2}}\vec{e}_x + \frac{x}{\sqrt{x^2+y^2}}\vec{e}_y$.