Deep Learning Solutions of Discrete-Time Investment Models

Yat Fan Lau

November 24, 2025

Abstract

This report applies deep learning methods to the solution and estimation of dynamic corporate finance models. We employ a neural network-based solver in TensorFlow for a discrete-time investment model with AR(1) shocks, leveraging a Monte Carlo stochastic gradient framework with an All-in-One(AiO) expectation operator. The methodology is then extended to a risky-debt model with endogenous default and bond pricing, where risky debt prices are determined by imposing the zero-profit condition, allowing us to recover state-dependent default regions, leverage dynamics etc. across the ergodic distribution of firms. In the future, we shall further implement and compare estimation techniques including the Generalized Method of Moments (GMM), the Simulated Method of Moments (SMM), and a Bayesian alternative using Hamiltonian Monte Carlo (HMC) within TensorFlow Probability.

Contents

1 Introduction and Literature Review

Dynamic corporate finance models provide a rigorous framework for understanding how firms jointly make investment and financing decisions over time with the goal of maximizing shareholder value. Importantly, dynamic models are particularly well-suited to capturing the inherently forward-looking nature of firm behavior. Their ability to produce both time-series and cross-sectional implications enhances their empirical relevance and allows for more robust evaluation against real-world data than static models. This makes them powerful tools for developing and testing economic theories of corporate decision-making. A foundational approach within this topic is the discrete-time dynamic investment model, in which firms optimally choose next-period capital to maximize the present value of cash flows, accounting for adjustment frictions. This framework, rooted in early contributions by Lucas and Prescott [1, 2], has since been extended and refined in various directions. In this report, we primarily follow the formulation in section 3 of a review by Strebulaev and Whited [3], who outline three core components of such models: **exogenous stochastic state variables**, an **objective function**, and a set of **endogenous state variables** governed by control decisions.

The exogenous state variables typically capture random shocks to firm fundamentals, such as changes in demand or productivity. The objective function often reflects the goal of maximizing shareholder value, quantified as the expected discounted stream of net cash flows. Endogenous state variables include the firm's current capital stock, and may also encompass labor, cash holdings, or debt levels. These evolve dynamically based on the firm's choices regarding future investment, employment, liquidity, and leverage.

Although solving these models analytically is often infeasible due to their complexity, advances in numerical methods have made it possible to approximate solutions and generate rich quantitative predictions. Researchers have developed a rich toolkit of methods for solving this kind of models, including traditional methods and machine-learning methods. In the following text we first review and outline some common traditional approaches to solve this kind of models, followed by the more modern machine learning based methods(such as the deep learning approach) which will be our main focus of this report.

A natural starting point is the **Bellman-operator dynamic programming**. It solves the Bellman equation on discretized grids for the state vector and for the shock process. For example, value function iteration (VFI) approximates the stochastic process with a finite-state Markov chain [4, 5] and iterates on the value function by maximizing over feasible controls at each grid point, delivering policy functions and simulated dynamics. Variants such as Howard's policy improvement and Modified Policy Iteration accelerate convergence by alternating policy evaluation and policy improvement [6, 7, 8, 9] . These methods are robust to non-smooth payoffs and occasionally binding constraints(OBCs). Their main limitation is the curse of dimensionality, which makes dense grids costly as the number of state variables grows.

Next approach is **Euler-Equation Time Iteration (Coleman operator)**, including the **Endogenous Grid Method** (EGM). Time iteration solves for decision rules directly by enforcing the Euler and envelope/Kuhn–Tucker conditions rather than iterating on the value function. EGM constructs "endogenous" grids of next-period states and maps them back to today's states using the Euler equation, eliminating costly root-finding and yielding large speed gains in continuous-state accumulation problems [10, 11, 12]. For many dynamic models, time iteration/EGM is especially effective in the smooth, concave core and has generalized variants that accommodate non-concavities and occasionally binding constraints via piecewise regimes or complementarity conditions [13, 14, 15]. Strengths include speed and accuracy for continuous problems; limitations are that standard EGM assumes invertibility/monotonicity of First-Order Conditions(FOCs) and smoothness—so with kinks or OBCs one must adopt generalized EGM or hybridize with grid or complementarity formulations.

Furthermore, **Projection/collocation (global approximation)**, with sparse-grid (Smolyak) variants. Projection methods approximate unknown value or policy functions with global basis functions (e.g., Chebyshev polynomials) and choose coefficients to minimize the Euler/Bellman errors at collocation nodes [16, 17, 18]. They deliver high global accuracy for smooth problems and scale more gracefully than dense grids when state dimension rises—especially with Smolyak sparse grids that drastically reduce nodes in multi-dimensional models such as cash-and-debt or risky-debt environments [19]. Their strengths are accuracy and better scalability; limitations are sensitivity to non-smoothness (basis approximations can struggle with kinks/OBCs, requiring piecewise bases or local refinement) and the need for careful error diagnostics to ensure reliability near boundaries and constraint switches.

Despite the fact that the above projection or value-function iteration workflows are precise in small dimensions, but they become brittle and slow as states and shocks increase, require hand-crafted derivatives, and struggle with kinks/OBCs. Fortunately, researchers recently have developed deep learning method for solving the models of interest[20]. Maliar *et al.* develop a unified DL framework for solving high-dimensional dynamic economic models. Their approach transforms lifetime reward, Euler, or Bellman equations into nonlinear regression objectives estimated on simulated data via stochastic gradient descent. A key ingredient is an *all-in-one* (AiO) operator that merges nested expectations and thereby makes training tractable, even with many shocks.

We bring these strands together: we implement the basic discrete-time and also risky-debt investment models with an AR(1) shock and solve it using the Euler and Bellman residual minimization respectively. We prefer DL to traditional projection/VFI/Euler-grid methods because it scales to high-dimensional state and shock spaces without tensor grids, is robust to non-smooth features/OBCs, needs no hand-coded derivatives thanks to automatic differentiation, collapses nested expectations into a single simulation step, and reuses the same modular training across different model variants, delivering competitive quantitative accuracy with far less brittleness and engineering overhead.

For the basic investment model of section 3.1 in [3], we employ the Euler residual minimization while in the risky-debt model in section 3.6, we choose the Bellman residual minimization. In the basic model, the firm solves a smooth dynamic optimization problem with interior choices and no OBCs. In this setting, the Euler equations fully characterize the optimum and provide a low-dimensional set of conditions that are convenient to implement. By contrast, the risky-debt model in Section 3.6 features default, limited liability, and regime changes between solvent and default states, so the value function is kinked and the optimal policy can involve corners. In such environments Euler conditions become numerically fragile, because they mix expectations over discontinuous objects, while the Bellman equation remains valid and naturally encodes the max operator over “continue” versus “default” decisions. Using a Bellman-residual objective for the risky-debt case therefore makes it easier to incorporate default, recovery etc. and to let the neural network learn both the value function and policy in the presence of non-smoothness, whereas for the basic model the simpler Euler-equation approach is more efficient.

The rest of this report is organized as follows: in section 2 we present the formulations, implementations and results of part 1. More specifically, Section 2.1 mainly focus on the basic investment model while section 2.2 extends to the risky-debt model. Hopefully, we shall finish part 2 in section 3 about the estimation methods in the future and also some bonus parts if time is allowed.

2 Part 1: Applying Deep Learning to Structural Corporate Finance Model

2.1 Basic Investment Model without Risky-Debt

2.1.1 Mathematical formulation

We consider the basic discrete-time infinite horizon investment problem described in Section 3.1 of [3]. The state comprises beginning-of-period capital $k_t > 0$ and an external shock $z_t > 0$ and we assume the shock follows an AR(1) in logs:

$$\ln z_{t+1} = \rho \ln z_t + \sigma \varepsilon_{t+1}, \quad \varepsilon_{t+1} \sim \mathcal{N}(0, \sigma_\varepsilon^2), \tag{1}$$

with $|\rho| < 1$ and $\sigma > 0$.

The parameter ρ measures the persistence of productivity shocks. If $\rho = 0$, a positive shock today may disappear by tomorrow — the shock is short-lived.

In such cases, firms are unlikely to make large-scale investments just for a temporary spike in profits, because they know the good times won’t last.

In contrast, if ρ is high (i.e., the shock is persistent), firms believe that today’s high productivity will likely continue into the future. In this case, investing makes sense, because the additional capital can generate high returns over many future periods. As a result, firms respond more strongly to shocks.

This is something unique to dynamic models: static models (which only consider a single period) cannot capture the concept of “persistence” because there is no “future.” Only dynamic models can reveal that a firm’s response to current shocks fundamentally depends on its expectations about the future.

The firm produces profits $\pi(k_t, z_t)$ and invests I_t with convex adjustment costs $\psi(I_t, k_t)$. The one-period cash flow to shareholders is

$$e(k_t, I_t, z_t) = \pi(k_t, z_t) - \psi(I_t, k_t) - I_t. \tag{2}$$

Before we proceed, we first provide an intuitive understanding of the above equation. Here k represents how much productive capacity a firm currently has (e.g., the size of machines, stores, or servers). This capacity can be increased through investment (e.g., buying new equipment), but it also depreciates over time at a rate δ with the capital stock accounting identity given by

$$k_{t+1} = (1-\delta)k_t + I_t, \qquad \delta \in (0,1), \tag{3}$$

On the other hand, z represent today’s “business weather”—that is, how favorable or unfavorable demand, efficiency, and the overall economy are. It fluctuates according to some stochastic process (e.g. AR(1) as in our case).

When z is high and expected to remain high, the return on each unit of k is also expected to be higher, which encourages more investment and leads to a higher future capital stock k'. Conversely, when z is low, firms tend to invest less.

However, adjusting k too quickly incurs costs(adjustment costs). Thus, firms weigh today’s cost against the expected future value $E[V(k', z')]$ and choose the optimal k' accordingly. This is essentially the central idea of the task we are going to solve in the following.

The value of the firm at time t can be given by the expected discounted sum of the cash flow over time

$$V_t = \max_{\{k_{t+j}\}_{j=1}^{\infty}} \mathbb{E}_t \left[\sum_{j=0}^{\infty} \left(\frac{1}{1+r} \right)^j e\left(k_{t+j}, I_{t+j}, z_{t+j}\right) \right] \tag{4}$$

where $\mathbb{E}_t$ denotes the expectation conditional on information known at time t, and r is the constant risk-free interest rate. Thus, the above problem of finding an optimal set of $\{k_{t+j}\}_{j=1}^{\infty}$ such that V_t attains its maximum under the constraint of Eqn.(3) is the **lifetime reward** formulation. As is mentioned in Ref[3], this model has no analytical solution. Thus, it would be helpful to recast the model in the Euler or the Bellman formulation in our DL implementation later.

The Euler formulation

To find the extrema of a function under certain constraints, we can introduce Lagrange multipliers χ_{t+j}’s to construct the Lagrangian

$$\mathcal{L} = \mathbb{E}_t \left[\sum_{j=0}^{\infty} \left(\frac{1}{1+r} \right)^j \left(e\left(k_{t+j}, I_{t+j}, z_{t+j}\right) - \chi_{t+j}\left(k_{t+j+1} - k_{t+j}(1-\delta) - I_{t+j}\right) \right) \right] \tag{5}$$

The first-order condition thus can be obtain by taking the derivative of $\mathcal{L}$ w.r.t. I_{t+j}

$$\frac{\partial \mathcal{L}}{\partial I_{t+j}} = 0 \quad \Rightarrow \quad 1 + \frac{\partial \psi}{\partial I_{t+j}} = \chi_{t+j} \tag{6}$$

We note that some of the equations in [3] contains typos in the indexing.

Next

$$\frac{\partial \mathcal{L}}{\partial k_{t+j+1}} = 0 \tag{7}$$

$$\mathbb{E}_t\left[-\chi_{t+j}(\frac{1}{1+r})^j + (\frac{1}{1+r})^{j+1}\left(\frac{\partial \pi}{\partial k_{t+j+1}} - \frac{\partial \psi}{\partial k_{t+j+1}} + (1-\delta)\chi_{t+j+1}\right)\right] = 0 \tag{8}$$

$$\mathbb{E}_t\left[(\frac{1}{1+r})\left(\frac{\partial \pi}{\partial k_{t+j+1}} - \frac{\partial \psi}{\partial k_{t+j+1}} + (1-\delta)\chi_{t+j+1}\right)\right] = \chi_{t+j} \tag{9}$$

Let $\beta = \frac{1}{1+r}$ and define

$$A_t := \frac{\partial \pi}{\partial k_t} - \frac{\partial \psi}{\partial k_t}.$$

The Euler equation is obtained by using Eq.(6)

$$\chi_{t+j} = \beta\,\mathbb{E}_t\left[A_{t+j+1} + (1-\delta)\chi_{t+j+1}\right]$$

$$1 + \frac{\partial \psi}{\partial I_{t+j}} = \beta\,\mathbb{E}_t\left[\frac{\partial \pi}{\partial k_{t+j+1}} - \frac{\partial \psi}{\partial k_{t+j+1}} + (1-\delta)\left(1 + \frac{\partial \psi}{\partial I_{t+j+1}}\right)\right] \tag{10}$$

Forward-iterating S times gives

$$\chi_{t+j} = \mathbb{E}_t\left[\sum_{s=1}^{S}\beta^s(1-\delta)^{s-1}A_{t+j+s} + \beta^S(1-\delta)^{S-1}\chi_{t+j+S}\right].$$

Under the usual transversality/no-bubbles condition

$$\lim_{S\to\infty}\mathbb{E}_t\left[\beta^S(1-\delta)^{S-1}\chi_{t+j+S}\right] = 0,$$

we obtain the closed-form solution with no χ on the right-hand side:

$$\chi_{t+j} = \mathbb{E}_t\left[\sum_{s=1}^{\infty}\beta^s(1-\delta)^{s-1}\left(\frac{\partial \pi}{\partial k_{t+j+s}} - \frac{\partial \psi}{\partial k_{t+j+s}}\right)\right].$$

Split the sum into the $j = 0$ term and the rest:

$$V_t = \max\left\{e(k_t, I_t, z_t) + \beta\,\mathbb{E}_t\left[\sum_{j=1}^{\infty}\beta^{j-1}e(k_{t+j}, I_{t+j}, z_{t+j})\right]\right\}.$$

Recognize the continuation value:

$$\sum_{j=1}^{\infty}\beta^{j-1}e(\cdot) = V_{t+1},$$

so

$$V_t = \max\left\{e(k_t, I_t, z_t) + \beta\,\mathbb{E}_t\left[V_{t+1}\right]\right\}.$$

Imposing the time stationarity and Markov property, we obtain

$$V(k,z) = \max_{k'}\left\{\pi(k,z) - \psi\left(k' - (1-\delta)k, k\right) - \left[k' - (1-\delta)k\right] + \frac{1}{1+r}\int V(k', z')\,dg(z' \mid z)\right\} \tag{11}$$

where

$$\mathbb{E}_t\left[V(k_{t+1}, z_{t+1})\right] = \int V(k', z')\,dg(z' \mid z).$$

This is known as the **Bellman equation**. Maximizing over k' means the firm is choosing next period's capital stock k'(or investment I) as its control and the arg max over k' defines the policy function $k' = \varphi(k, z)$: for each current state (k, z), choose the investment level that maximizes the firm's value (subject to any feasibility constraints like $k' \geq 0$).

The objective is to maximize the expected present value of cash flows,

$$V(k,z) = \max_{\{k' \geq 0\}} \left\{ \pi(k,z) - \psi(k' - (1-\delta)k, k) - \big(k' - (1-\delta)k\big) + \beta \mathbb{E}\big[V(k',z') \,\big|\, z\big] \right\}, \quad (12)$$

with $\beta = \frac{1}{1+r} \in (0,1)$ and we use prime to denote next-period quantities. We note the equivalence between Eq. (11) and Eq. (4), namely that Eq. (11) is the Bellman (recursive) formulation of the sequential problem in Eq. (4). In Eq. (4), the firm chooses the entire future path $\{k_{t+1}, k_{t+2}, \ldots\}$ to maximize expected discounted cash flows, whereas in Eq. (11) it chooses only next period's capital $k' \equiv k_{t+1}$ (equivalently I_t via $k_{t+1} = (1-\delta)k_t + I_t$), and the continuation value $V(k', z')$ embeds all future optimal choices.

By Bellman's principle of optimality—and because k_t is predetermined—maximizing over k' at time t delivers the same policy and value as maximizing over the whole sequence in Eq. (4).

To implement the model concretely, we first need to choose the specific expressions for the production and cost functions:

Functional forms In line with Strebulaev *et al.*[3], we use

$$\pi(k,z) = zk^{\theta}, \qquad \theta \in (0,1), \quad (13)$$

$$\psi(I,k) = \frac{\phi}{2}\frac{(I-k\delta)^2}{k} = \frac{\phi}{2}k(\iota - \delta)^2, \qquad \phi > 0. \quad (14)$$

where we define $\iota = I/k$. These imply

$$\pi_k(k,z) = \theta z k^{\theta-1}, \qquad \psi_I(I,k) = \phi\frac{I-k\delta}{k} = \phi(\iota-\delta), \qquad \psi_k(I,k) = \frac{\phi}{2}(\delta^2 - \iota^2). \quad (15)$$

Before we proceed, we give intuitive meaning of the above functions: the parameter θ measures the rate at which returns to capital diminish. A smaller θ implies that the production function is more concave, meaning that the marginal return from additional capital declines more rapidly.

As a result, when productivity z changes, its impact on the marginal return to capital is smaller. Firms perceive that "investing a bit more doesn't bring much additional benefit," so their response to shocks is more muted, and investment behavior becomes more stable.

2.1.2 Deep learning method and algorithm

In this section, we outline the important ingredients of our implementation of the DL method in Maliar *et al.* (2021) [20] on the model in Strebulaev *et al.*[3].

Training distribution and ergodic domain

States sampling Just like any other conventional DL training, the first step is to get the data for the training. In our case, the data are the states (k, z) generated by ourselves. Maliar *et al.* (2021) suggest two complementary approaches: (i) *exogenous coverage sampling* from a fixed solution domain to guarantee broad coverage, and (ii) *on-policy (ergodic) sampling* from the stationary distribution generated by simulating the model under the current policy, so that training concentrates on the region visited in equilibrium.

Although our current model is small (two state variables), so a fixed grid or random coverage would be computationally sufficient, we adopt a hybrid scheme: we pretrain with coverage draws around the steady state for k and from the stationary AR(1) process for z, then switch to mini-batches drawn primarily from a replay buffer of simulated (k, z) while still retain a small fraction of coverage sampling for exploration.

This keeps accuracy higher on the ergodic set and scales naturally to higher-dimensional problems where exogenous grids become impractical.

Notation

- State vector: $s_t = (k_t, z_t)$.
- Shock process: $\ln z_{t+1} = \rho \ln z_t + \varepsilon_{t+1}, \quad \varepsilon_{t+1} \sim \mathcal{N}(0, \sigma_\varepsilon^2)$.
- Policy network: $k_{t+1} = \varphi_w(k_t, z_t)$.

Exogenous Monte-Carlo Grid We start from the steady state where we choose an initial k by setting $z = z^* = 1$ and the Euler equation Eq.(3.12) in Strebulaev *et al.*(2012) reduces to $k = k^* = (\frac{\theta}{r+\delta})^{1/(1-\theta)}$. We select two scalars $m_- < 1 < m_+$ (e.g. 0.2 and 5) and set $k_{\min} = m_- k^*,\ k_{\max} = m_+ k^*$.. Then generate k via

$$\ln k \sim \text{Uniform}\big(\ln k_{\min}, \ln k_{\max}\big), \qquad k = e^{\ln k}.$$

To sample z we could sample

$$\ln z \sim \mathcal{N}\big(0, \sigma_{\ln z}^2\big), \qquad z = e^{\ln z}.$$

where $\sigma_{\ln z}^2 = \frac{\sigma_\varepsilon^2}{1-\rho^2}$ comes from taking the expectation on both sides of the AR(1) shock process. We truncate at $\pm 3\sigma_{\ln z}$

Endogenous Simulation with Replay Buffer Besides exogenous Monte Carlo sampling, we also use endogenous (on-policy) simulation. Note that we need to know the current policy to generate on-policy states.

M1. **Initialisation.**

- Select initial state(s), e.g. $k_0 = k^*,\ z_0 = 1$.
- Create an empty replay buffer $\mathcal{B} = \varnothing$ of maximum size N_{buf}.

M2. **Roll-out under current policy.**
For n_{roll} consecutive periods and for every parallel path:

$$\begin{aligned}&\text{store current state } s_t \text{ in } \mathcal{B} \quad (\text{discard the oldest element if } |\mathcal{B}| > N_{\text{buf}});\\ &\varepsilon_{t+1} \sim \mathcal{N}(0, \sigma_\varepsilon^2), \quad \ln z_{t+1} = \rho \ln z_t + \varepsilon_{t+1}, \quad z_{t+1} = e^{\ln z_{t+1}};\\ &k_{t+1} = \varphi_w(k_t, z_t) \text{ (network output)}.\end{aligned}$$

M3. **Sampling for stochastic gradient.**
Draw a mini-batch $\{s^{(i)}\}_{i=1}^B$ *uniformly at random* from the buffer $\mathcal{B}$.

M4. **Parameter update.**
Compute the loss, e.g. $L(w) = \dfrac{1}{B}\displaystyle\sum_{i=1}^{B} \xi\big(s^{(i)}; w\big)$, where ξ is the chosen residual, and perform

$$w \leftarrow w - \eta_t \nabla_w L(w) \qquad \text{(ADAM or another SGD rule)}.$$

M5. **Iterate.**
Return to step M2 with the updated parameters. Data generation (M2) and learning (M3–M4) alternate until convergence of w.

Remarks

- After a short burn-in, the replay buffer contains draws from the stationary distribution induced by the *current* policy; training therefore focuses on the state region relevant for the optimal solution.
- Coverage sampling is convenient for debugging and for the first few iterations of training; once the network is stabilized, ergodic sampling typically yields faster and more accurate convergence.

Loss functions and All-in-One(AiO) operator

Euler Residual

For a given state (k, z) and a draw ε for z', the (one-step) Euler residual under policy $\varphi(\cdot; w)$ is

$$\mathcal{R}\big(k, z, \varepsilon; w\big) \;\equiv\; \frac{1}{1+r}\Big(\pi_k(k', z') \;-\; \psi_k(I', k') \;+\; (1-\delta)\big(1+\psi_I(I', k')\big)\Big) \;-\; \big(1+\psi_I(I, k)\big), \tag{16}$$

where $k' = \varphi(k, z; w)$, $z' = \exp\{\rho \ln z + \varepsilon\}$, and (I', k', I) are as defined above. At the exact solution, the conditional expectation $\mathbb{E}[\,\mathcal{R}(k, z, \varepsilon; w^*) \mid k, z\,] = 0$ for all relevant (k, z).

A natural objective for training the policy is the squared conditional Euler error averaged over the state distribution,

$$L(w) \;\equiv\; \mathbb{E}_{(k,z)}\Big[\big(\mathbb{E}_{\varepsilon}\left[\mathcal{R}\big(k, z, \varepsilon; w\big)\right]\big)^2\Big]\,. \tag{17}$$

Computing (17) by Monte Carlo requires, for each state (k, z) in a batch, an inner simulation over many shocks ε to approximate $\mathbb{E}_\varepsilon[r(k, z, \varepsilon; \theta)]$, followed by squaring it and averaging over states. This "nested" structure is computationally expensive: if n_s states and n_ε shocks per state are used, each gradient step costs $O(n_s n_\varepsilon)$ model evaluations. To avoid nested simulation, Maliar suggests the AiO expectation which uses two independent draws $\varepsilon_1, \varepsilon_2$ of the shock for the *same* state (k, z). Conditional on (k, z),

$$\big(\mathbb{E}_\varepsilon[\mathcal{R}]\big)^2 \;=\; \mathbb{E}_{\varepsilon_1}\big[\mathcal{R}(k, z, \varepsilon_1; w)\big]\,\mathbb{E}_{\varepsilon_2}\big[\mathcal{R}(k, z, \varepsilon_2; w)\big] \;=\; \mathbb{E}_{\varepsilon_1,\varepsilon_2}\big[\,\mathcal{R}(k, z, \varepsilon_1; w)\,\mathcal{R}(k, z, \varepsilon_2; w)\,\big],$$

because ε_1 and ε_2 are independent. Hence, the theoretical loss (17) can be written without nesting as a single expectation over a *composite* random vector $\omega = (k, z, \varepsilon_1, \varepsilon_2)$:

$$L(w) \;=\; \mathbb{E}_{(k,z),\,\varepsilon_1,\,\varepsilon_2}\big[\,\mathcal{R}\big(k, z, \varepsilon_1; w\big)\;\mathcal{R}\big(k, z, \varepsilon_2; w\big)\,\big]\,. \tag{18}$$

Equation (18) is the *AiO Euler loss*: it replaces the square of an inner expectation by the product of two independent one-step residuals, enabling unbiased stochastic gradients with only two shock draws per state. Given a mini-batch of states $\{(k_i, z_i)\}_{i=1}^N$ and, for each i, two independent shocks $\varepsilon_{i,1}, \varepsilon_{i,2}$, define the empirical AiO loss

$$\widehat{L}_N(w) \;\equiv\; \frac{1}{N}\sum_{i=1}^{N}\Big[\mathcal{R}\big(k_i, z_i, \varepsilon_{i,1}; w\big)\;\mathcal{R}\big(k_i, z_i, \varepsilon_{i,2}; w\big)\Big], \tag{19}$$

where each $\mathcal{R}(\cdot)$ is computed via (16) using the policy-induced k', I and k'', I' for the corresponding shocks. Stochastic-gradient algorithms (e.g., ADAM) directly minimize (19) over w.

What to do given a batch of states (k, z). For each (k_i, z_i) in the batch:

(a) Compute $k'_i = \varphi(k_i, z_i; w)$ and $I_i = k'_i - (1-\delta)k_i$.

(b) Draw two independent shocks $\varepsilon_{i,1}, \varepsilon_{i,2} \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ and set $z'_{i,j} = \exp\{\rho \ln z_i + \varepsilon_{i,j}\}$ for $j = 1, 2$.

(c) For each $j \in \{1, 2\}$, compute $k''_{i,j} = \varphi(k'_i, z'_{i,j}; w)$, $I'_{i,j} = k''_{i,j} - (1-\delta)k'_i$, and the residual

$$\mathcal{R}_{i,j} = \frac{1}{1+r}\Big(\pi_k(k'_i, z'_{i,j}) \;-\; \psi_k(I'_{i,j}, k'_i) \;+\; (1-\delta)\big(1+\psi_I(I'_{i,j}, k'_i)\big)\Big) \;-\; \big(1+\psi_I(I_i, k_i)\big).$$

(d) Form the per-state AiO contribution $\mathcal{R}_{i,1}\,\mathcal{R}_{i,2}$ and average across i as in (19).

This procedure yields an unbiased Monte Carlo estimator of the theoretical loss (18) with only two shock draws per state, avoiding the cost of nested expectations while remaining faithful to the Euler condition.

DL implementation We parameterize the policy $\iota(k, z; w)$ by a two-layer feed-forward neural network with 64 hidden units per layer and tanh activations. The network takes $(\ln k, \ln z)$ as inputs and outputs a raw scalar which is then mapped into the admissible interval $[\iota_{\min}, \iota_{\max}]$ through a scaled tanh transformation. We set $\iota_{\max} = 0.5$ and $\iota_{\min} = -0.99(1-\delta)$, which guarantees

$$k_{t+1} = \big(1 - \delta + \iota_t\big)k_t \;\geq\; 0.01\,k_t$$

and thus strictly positive capital. In terms of the notation used above, this investment-rate policy $\iota(\cdot; w)$ plays the role of the policy function $\varphi(\cdot; w)$: given (k_t, z_t), next-period capital is

$$k_{t+1} \;=\; \varphi(k_t, z_t; w) \;\equiv\; \big(1 - \delta + \iota(k_t, z_t; w)\big)\,k_t.$$

The AiO Euler loss function follows [20] and is based on the one-step Euler residual $\mathcal{R}(k, z, \varepsilon; w)$ defined in (16). For each state (k_t, z_t) we draw two independent innovations $\varepsilon^{(1)}_{t+1}$ and $\varepsilon^{(2)}_{t+1}$, and, for variance reduction, use antithetic pairs $\pm\varepsilon^{(j)}_{t+1}$ for $j = 1, 2$. For each shock and each sign $s \in \{+, -\}$ we compute the associated one-step-ahead Euler residual

$$\mathcal{R}^{(j,s)}(k_t, z_t; w) \;\equiv\; \mathcal{R}\big(k_t, z_t, s\,\varepsilon^{(j)}_{t+1}; w\big), \qquad j = 1, 2,$$

where $\mathcal{R}(\cdot)$ is evaluated using the policy-induced $k' = \varphi(k_t, z_t; w)$, $I = k' - (1-\delta)k_t$ and $k'' = \varphi(k', z_{t+1}; w)$, $I' = k'' - (1-\delta)k'$ as in (16). For each branch j we average over the antithetic pair,

$$\bar{\mathcal{R}}^{(j)}(k_t, z_t; w) \;\equiv\; \tfrac{1}{2}\Big(\mathcal{R}^{(j,+)}(k_t, z_t; w) + \mathcal{R}^{(j,-)}(k_t, z_t; w)\Big), \qquad j = 1, 2.$$

The AiO loss then approximates the theoretical objective $\mathbb{E}\big[(\mathbb{E}_\varepsilon \mathcal{R})^2\big]$ in (17) by

$$L(w) = \mathbb{E}\Big[\bar{\mathcal{R}}^{(1)}(k_t, z_t; w)\,\bar{\mathcal{R}}^{(2)}(k_t, z_t; w)\Big],$$

where the outer expectation is taken over the joint distribution of $(k_t, z_t, \varepsilon^{(1)}_{t+1}, \varepsilon^{(2)}_{t+1})$. Intuitively, minimizing $L(w)$ drives the conditional expectation $\mathbb{E}_\varepsilon\big[\mathcal{R}(k_t, z_t, \varepsilon; w)\big]$ towards zero for all states in the training support, thus enforcing the Euler condition in an L^2 sense.

We train the network with the Adam optimizer at learning rate 10^{-4}. The training scheme mixes exogenously sampled *coverage* states with *on-policy* states drawn from long simulations under the current policy $\iota(\cdot; w)$. In total we use 1,000 pre-training steps on pure coverage samples followed by 60,000 main training steps with mini-batches of size 4,096.

2.1.3 Results and Validity of the DL Method

In this section, we summarize our result and we demonstrate that a single fully-connected neural network, trained by stochastic gradient descent on an all-in-one (AiO) Euler residual, is able to solve the model with high accuracy.

Policy shape and deterministic dynamics The learned policy has the expected qualitative properties. Figure 1 plots a heatmap of $\iota(k, z)$ on a grid covering $k \in [0.2k^*, 5k^*]$ and the stationary 2.5-standard-deviation band of $\ln z_t$, where k^* is the deterministic steady state solving $\theta k^{\theta-1} = r + \delta$. For any fixed z, the investment rate is strictly decreasing in k, while for any fixed k it is increasing in z. Thus, high productivity and low capital jointly induce a high investment rate, and conversely.

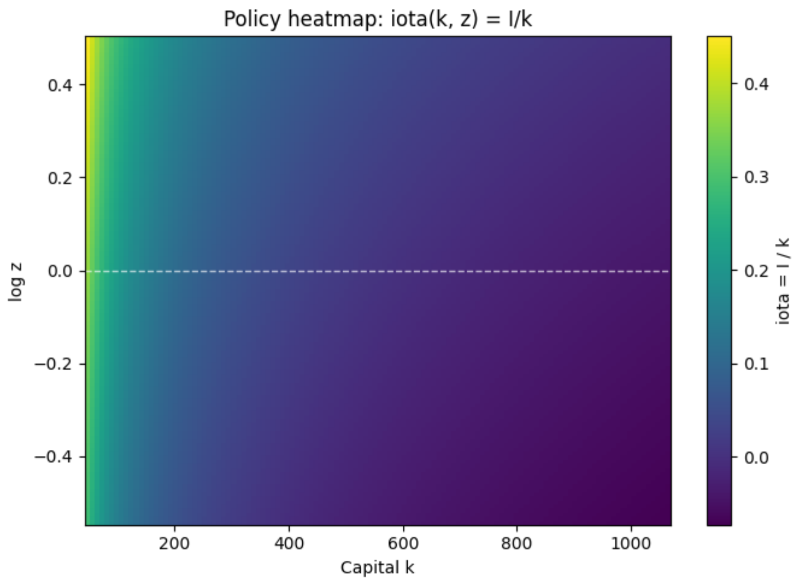

Figure 1:

Figure 2 investigates the deterministic transition dynamics when $z_t \equiv 1$. Starting from three initial conditions $k_0 \in \{0.2k^*, k^*, 5k^*\}$, the corresponding capital paths converge monotonically to a neighborhood of k^*, while the investment rate $\iota_t = I_t/k_t$ converges to the depreciation rate $\delta = 0.1$. These dynamics are exactly those implied by the analytical steady-state condition of the model and provide a qualitative check of the policy network.

Euler-equation accuracy and effectiveness measures To assess the quantitative accuracy of the solution we follow the practice in Maliar *et al.* [20] and compute high-precision Euler residuals on independent test sets. For any given state (k, z) we consider the *conditional* Euler residual

$$\bar{\mathcal{R}}(k, z; w) \;\equiv\; \mathbb{E}_\varepsilon\big[\mathcal{R}(k, z, \varepsilon; w) \mid k, z\big],$$

which is the conditional expectation appearing in the Euler equation formulation. We approximate this conditional expectation using n-node Gauss–Hermite (GH-n) quadrature for the Normal innovation $\varepsilon_{t+1} \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Denote by $\widehat{\mathbb{E}}_n[\cdot]$ the resulting quadrature operator and define

$$\widehat{\bar{\mathcal{R}}}_n(k, z; w) \;\equiv\; \widehat{\mathbb{E}}_n\big[\mathcal{R}(k, z, \varepsilon; w)\big] \approx \bar{\mathcal{R}}(k, z; w).$$

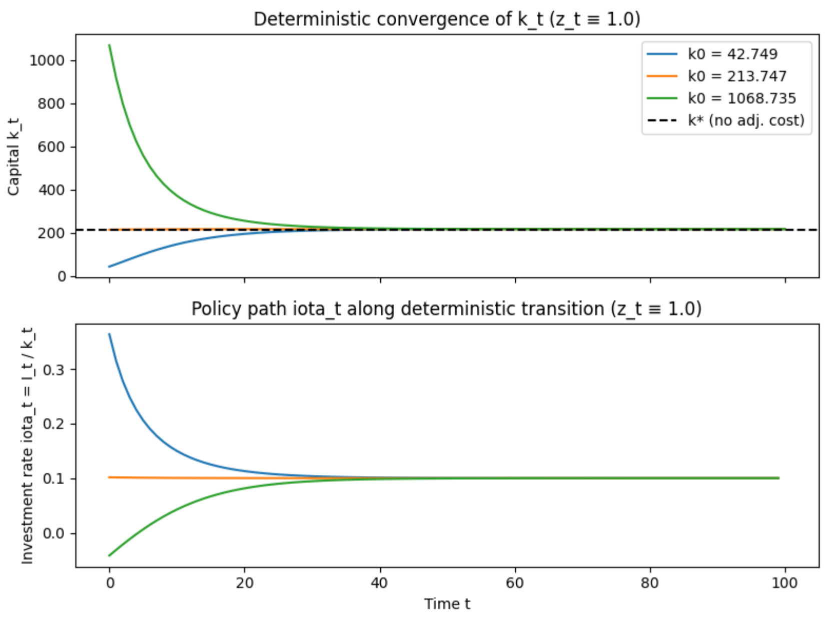

Figure 2:

Our primary accuracy measures are:

(a) **Absolute Euler residuals.** For a test set $\{(k_i, z_i)\}_{i=1}^N$ we compute

$$\bar{\mathcal{R}}_i \;\equiv\; \widehat{\bar{\mathcal{R}}}_n(k_i, z_i; w), \qquad \left|\bar{\mathcal{R}}_i\right| = \left|\widehat{\bar{\mathcal{R}}}_n(k_i, z_i; w)\right|,$$

and summarize the collection $\{|\bar{\mathcal{R}}_i|\}_{i=1}^N$ by the mean absolute error (MAE), the root mean squared error (RMSE), the median, the 95th percentile, and the maximum. Following the quantitative macroeconomics literature, we consider errors below 10^{-3} in absolute value to be negligible.

(b) **Relative (dimensionless) residuals.** Since $\bar{\mathcal{R}}(k, z; w)$ is a linear combination of economically meaningful terms from the Euler equation, we define the relative residual

$$r_i \;\equiv\; \frac{|\bar{\mathcal{R}}_i|}{\left|1 + \psi_I(I_i, k_i)\right| + \left|\beta\, \widehat{\mathbb{E}}_n[\mathrm{term}(k_i, z_i; w)]\right|},$$

where I_i is the investment implied by the policy at (k_i, z_i) and "term" denotes the bracketed expectation term on the right-hand side of the Euler equation. This quantity measures the Euler error as a fraction of the typical scale of the left- and right-hand sides.

(c) **Share-of-states statistics.** We report the share of states satisfying $|\bar{\mathcal{R}}_i| \leq 10^{-3}$ and $|\bar{\mathcal{R}}_i| \leq 10^{-4}$, both on the ergodic set and on a wide coverage region that contains the ergodic set with high probability.

(d) **GH-node robustness.** To verify that Euler errors are not driven by an inaccurate quadrature rule, we compare the median and 95th percentile of $|\bar{\mathcal{R}}_i|$ when increasing the number of nodes from 10 to 15 and 20. Relative changes of only a few tenths of a percent indicate that GH–10 is already sufficiently accurate.

We perform three types of tests using GH–10 as the baseline integration rule:

Ergodic on-policy test. We simulate 10,000 paths of length 10,000 periods under the final policy, discard a 10,000-period burn-in, and then collect $N = 100{,}000$ states (k_t, z_t) to approximate the ergodic distribution. On this on-policy set we obtain

$$\mathrm{MAE}(|\bar{\mathcal{R}}|) = 1.7{\times}10^{-5}, \quad \mathrm{RMSE} = 2.1{\times}10^{-5}, \quad \mathrm{P95} = 3.5{\times}10^{-5}, \quad \max_i |\bar{\mathcal{R}}_i| = 1.0{\times}10^{-3}.$$

The corresponding relative residuals have mean 8.7×10^{-6} and 95th percentile 1.8×10^{-5}. All states satisfy $|\bar{\mathcal{R}}_i| \leq 10^{-3}$ and 99.9% satisfy $|\bar{\mathcal{R}}_i| \leq 10^{-4}$.

Coverage test. We construct an outer *coverage box* in $(\ln k, \ln z)$ space from the 1st–99th percentiles of the ergodic distribution, expanded by 5% in each dimension, and draw $N = 20{,}000$ states uniformly inside this box. On this wider region we obtain

$$\mathrm{MAE}(|\bar{\mathcal{R}}|) = 2.6{\times}10^{-5}, \quad \mathrm{RMSE} = 3.8{\times}10^{-5}, \quad \mathrm{P95} = 8.8{\times}10^{-5}, \quad \max_i |\bar{\mathcal{R}}_i| = 1.7{\times}10^{-4},$$

and 96.1% of points satisfy $|\bar{\mathcal{R}}_i| \leq 10^{-4}$ (all satisfy $|\bar{\mathcal{R}}_i| \leq 10^{-3}$). Edge and corner points at the boundary of the coverage box have slightly larger but still small errors, with MAE 4.7×10^{-5} and 95th percentile 1.5×10^{-4}.

Quadrature robustness. On a random subsample of 5,000 coverage points we recompute Euler residuals using GH–15 and GH–20. Relative to GH–10, the median and 95th percentile of $|\bar{\mathcal{R}}_i|$ change by at most 0.2%, confirming that quadrature error is negligible.

Finally, Figure 3 shows the distribution of the conditional Euler residuals $\bar{\mathcal{R}}(k, z; w)$ on a 50×50 regular grid covering the coverage box, using GH–20 nodes. The histogram is sharply centered around zero, with almost all mass inside $\pm 2 \times 10^{-4}$ and a thin left tail down to approximately -6×10^{-4}. Combined with the summary statistics above, this indicates that the deep-learning Euler-equation solver delivers an approximation whose Euler errors are below commonly used tolerances both on and slightly outside the ergodic set. In summary, the proposed deep-learning approach yields a policy function that is economically sensible and numerically accurate. Relative Euler equation errors are on the order of 10^{-5} or less across the relevant state space, suggesting that the method can serve as a reliable benchmark for discrete-time investment models of the type studied in Strebulaev *et al.* [3].

2.2 Investment model with risky-debt

The basic investment model (Sec. 3.1 of Strebulaev *et al.*) has a representative firm choosing next period capital k' given current capital k and a productivity shock z. There is no corporate debt, no taxes, and therefore no financing margin: the decision is purely real. The Bellman equation is standard and the state is $s \equiv (k, z)$. This benchmark is ideal for isolating investment dynamics, but it cannot address capital structure, default, credit spreads, or tax policy because the cost of funds is exogenous and flat, there is no interest tax shield, and limited liability never binds.

To bring the model closer to corporate finance reality, we consider adding a one-period risky debt choice b', corporate taxes with an interest tax shield, and limited liability so equity can default endogenously when equity value hits zero(Sec. 3.6 of Strebulaev *et al.*). Lenders are risk-neutral and impose a zero-profit condition, so the debt yield $\tilde{r}(z, k', b')$ is priced *inside* the model. The state expands to $s \equiv (k, b, z)$ and the firm jointly chooses (k', b'). The problem becomes a coupled system: the firm's dynamic program interacts

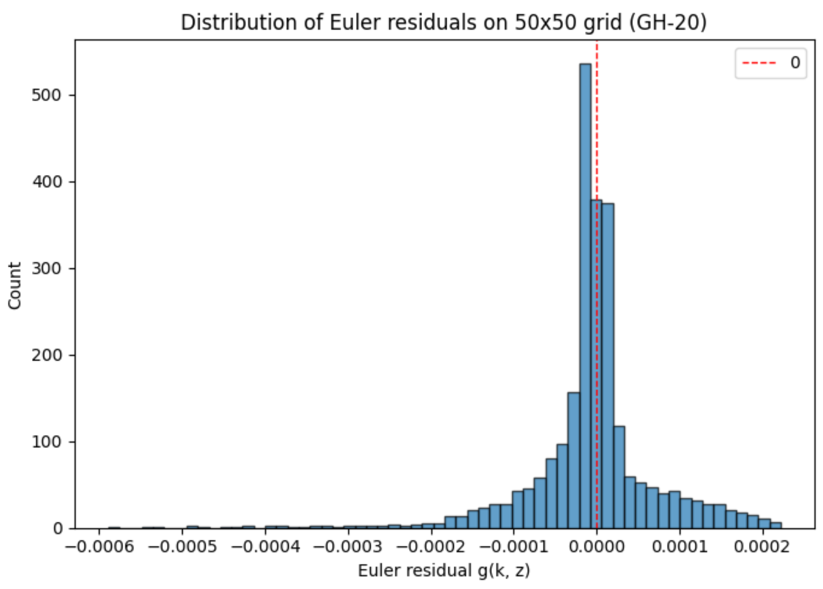

Figure 3:

with a debt-pricing fixed point. Intuitively, leverage is chosen by trading off tax benefits of interest deductibility against expected deadweight default costs, under an endogenous borrowing rate that rises with default risk.

The main computational difficulty in the risky–debt model is a "loop within a loop." The endogenous debt yield $\tilde{r}(z, k', b')$ must satisfy lenders' zero–profit condition, which depends on default probabilities and recoveries implied by the firm's optimal policy and value; but the firm's optimal (k', b') choices themselves depend on $\tilde{r}$. A traditional solution begins by solving the value function under the assumption that debt is priced at the risk-free rate. Once the value function has been obtained, use it to identify default regions and compute the corresponding interest rate as a function of the model's state variables. This updated interest rate is then used to re-solve the value function. The process is repeated iteratively until the value function converges(see Ref [3]). While transparent, this approach is computationally heavy: the state (k, b, z) suffers the curse of dimensionality(especially when more state variables are included), limited liability induces kinks and nonconvexities that demand fine grids/interpolation, and accurate pricing requires costly nested integration over shocks.

We instead extend the DL framework(Maliar *et al.* 2021) we employed in previous section to tackle the risky debt model. The idea is to cast the whole equilibrium system into a single expectation–based loss and train neural networks to approximate the decision rules/value function. Concretely, we parameterize the policy functions by $\{k', b'\} = \{\varphi_k(k, b, z; w_k), \varphi_b(k, b, z; w_b)\}$ and the value function $V = \varphi_V(k, b, z; w_V)$. We then minimize an empirical loss that stacks (i) the Bellman–equation residuals (computed via the AiO product trick to avoid bias) and (ii) the FOC residuals. The lenders' zero–profit condition is enforced in the forward pass when computing the risky interest rate $\tilde{r}$, and thus its residual is not explicitly included in the loss function. This DL approach removes the explicit nested fixed point, so policies and pricing co–evolve during training until the residuals jointly vanish.

2.2.1 Model Formulation and training with Risky Debt

In this section, we outline the basic formulation of risky-debt model.

Financing: one–period risky debt and net cash

At date t the firm chooses next–period net debt b' and capital k'. If $b' > 0$, it issues a one–period risky bond with face value b' to be repaid at $t{+}1$; if $b' \le 0$, it is a net lender holding a risk-free asset. We denote by $q_t(k', b', z)$ the time–t price per unit of (nominal) face value, so that the proceeds are

$$\text{debt proceeds}_t \;=\; q_t(k', b', z)\, b'.$$

In the implementation we treat very small positions as risk–free:

$$\text{if } b' \le \varepsilon_b, \qquad q_t(k', b', z) \;=\; \frac{1}{1+r},$$

for a small threshold $\varepsilon_b > 0$. This both avoids numerical problems when dividing by b' in the risky–debt pricing formula, and consistently treats net savings (and tiny debt positions) as risk–free claims.

Default, limited liability, and recovery

Equity has limited liability: if the continuation value is negative, equity defaults and receives zero. Let $C(k, b, z)$ denote the (pre–truncation) continuation value. The equity value is

$$V(k, b, z) \;=\; \max\{C(k, b, z), 0\}.$$

In the code this is implemented via a smooth softplus projection:

$$V(k, b, z) \;\approx\; \tau_V \, \log\!\Big(1 + e^{C(k,b,z)/\tau_V}\Big), \tag{20}$$

with a temperature parameter $\tau_V > 0$ that is gradually annealed during training. For small τ_V, (20) approximates $V = \max\{C, 0\}$.

If default occurs at $t{+}1$, debtholders seize the firm and receive a recovery value that is a fraction $1 - \alpha$ of the after–tax operating cash flow plus undepreciated capital:

$$R(k', z') \;:=\; (1-\alpha)\Big((1-\tau)\,\pi(k', z') + (1-\delta)\,k'\Big), \qquad \alpha \in [0, 1). \tag{21}$$

Pricing of risky debt

Lenders are risk–neutral and discount at rate r. At date t, for $b' > \varepsilon_b$ (true risky debt), a bond with face value b' is sold at price

$$P_t \;=\; q_t(k', b', z)\, b', \qquad q_t(k', b', z) \;\equiv\; \frac{1}{1 + \tilde r(z, k', b')}.$$

At $t{+}1$, if there is no default the lender receives the promised amount $(1 + \tilde r(z, k', b'))\, b'$, while in default it receives $R(k', Z')$ as in Eq. (21). Risk–neutral pricing (as in Ref[3], Eq. (3.27)) requires that the expected gross payoff, discounted at the risk–free rate, equals the issue proceeds b':

$$b'(1+r) \;=\; \mathbb{E}\big[D' R(k', Z') + (1-D')\,(1 + \tilde r(z, k', b'))\, b' \,\big|\, Z = z\big]\,. \tag{22}$$

Dividing by b' and solving for $1 + \tilde r$ yields

$$1 + \tilde r(z, k', b') \;=\; \frac{1 + r \;-\; \mathbb{E}\left[D' \,\dfrac{R(k', Z')}{b'} \,\middle|\, Z = z\right]}{\mathbb{E}[1 - D' \mid Z = z]}. \tag{23}$$

In our implementation, the conditional expectations in (23) are approximated with M_{pricing} Monte Carlo draws of Z', using the *target* value network to evaluate $C(k', b', Z')$ and a smooth default gate

$$D' \approx \sigma\left(-\frac{C_{\bar{w}}(k', b', Z')}{\tau_D}\right), \tag{24}$$

where $\sigma(\cdot)$ is the logistic function, $\tau_D > 0$ is a temperature parameter, and $C_{\bar{w}}$ denotes the target continuation–value network (parameters frozen by Polyak averaging). When $b' \leq \varepsilon_b$ we bypass (23) and set

$$q_t(k', b', z) = \frac{1}{1+r},$$

so that net saving is priced at the risk–free rate. In all cases, the implementation clips extreme payoff ratios R'/b' to mitigate numerical outliers, and clips q_t into $[0.01,\ 1/(1+r)]$.

Cash flow and costly external finance

Let $q = q_t(k', b', z)$. The gross cash flow to equity at date t *before* external financing costs, denoted $e(k, k', b, b', z)$, is

$$\begin{aligned} e(k, k', b, b', z) \equiv (1-\tau)\,\pi(k, z) \;-\; \Big(k' - (1-\delta)k\Big) \;-\; \psi\big(k' - (1-\delta)k, k\big) \\ +\; q\,b' \;+\; \tau\,(1-q)\,b'\,\beta \;-\; b. \end{aligned} \tag{25}$$

The first line is after–tax operating profit net of investment and quadratic adjustment costs. The second line contains (i) the time–t proceeds from issuing (or repurchasing) debt with face value b', and (ii) a discrete–time approximation of the tax effect of interest, implemented as $\tau(1-q)b'\beta$. For $b' > 0$, this term acts as a tax shield on expected interest payments; for $b' < 0$ it captures the tax on interest income from net saving.

As in Ref[3], we allow for costly external equity finance and specify

$$\eta(e) \equiv \big(\eta_0 + \eta_1 |e|\big)\,\mathbf{1}_{\{e<0\}}, \tag{26}$$

so that equity injections ($e < 0$) are penalized by a fixed component η_0 and a linear component proportional to $|e|$, while distributions ($e \geq 0$) are costless. The *net* payout to shareholders is then

$$d(k, k', b, b', z) \equiv e(k, k', b, b', z) - \eta\big(e(k, k', b, b', z)\big).$$

Bellman equation and continuation value

Let $C(k, b, z)$ denote the pre–truncation continuation value at state (k, b, z) and $V(k, b, z)$ the equity value under limited liability. The exact dynamic program can be written as

$$C(k, b, z) = \max_{k', b'} \left\{ d(k, k', b, b', z) + \beta\,\mathbb{E}\big[V(k', b', Z') \,\big|\, Z = z\big] \right\}, \tag{27}$$

$$V(k, b, z) = \max\{C(k, b, z), 0\}. \tag{28}$$

In the implementation we parameterize:

- a *policy* network π_w mapping normalized features $s = (\log k,\ b/k,\ \log z)$ to controls (k', b'):

  $$(k', b') = \pi_w(s),$$

  where $k' > 0$ is enforced by a softplus activation plus a small floor ($k' \geq 0.1$), and b' is unconstrained (can be positive or negative);

- a *continuation–value* network C_w mapping x to $C_w(k, b, z)$;
- the corresponding equity value via the softplus limited–liability transform in Eq. 20.

In the Bellman right–hand side we use a slowly moving *target* copy $C_{\bar{w}}$ (and $V_{\bar{w}}$) for the expectation over Z', updated by Polyak averaging. Given a state (k, b, z) and controls $(k', b') = \pi_w(k, b, z)$, the model–implied continuation value used in the loss is

$$\text{RHS}(k, b, z; w) \;:=\; d(k, k', b, b', z) \;+\; \beta\, \mathbb{E}\big[V_{\bar{w}}(k', b', Z') \,\big|\, Z = z\big], \tag{29}$$

where the expectation is approximated via Monte Carlo. The Bellman equation then requires

$$C_w(k, b, z) \;\approx\; \text{RHS}(k, b, z; w),$$

while Eq. (28) is enforced by the softplus mapping from C_w to V_w.

Soft default indicator

The exact default indicator at $t+1$ is $D' = \mathbf{1}\{C(k', b', Z') \leq 0\}$. For differentiability we use the smooth gate

$$D' \;\approx\; \sigma\!\left(-\frac{C_{\bar{w}}(k', b', Z')}{\tau_D}\right),$$

as in (24). This D' enters the risky–debt pricing kernel (23) and the diagnostic default–probability computation.

First–order conditions and FOC residuals

Optimality of (k', b') requires that the Bellman right–hand side be maximized with respect to these controls. For interior solutions,

$$\frac{\partial}{\partial k'}\text{RHS}(k, b, z; w) \;=\; 0, \qquad \frac{\partial}{\partial b'}\text{RHS}(k, b, z; w) \;=\; 0.$$

Analytical expressions are cumbersome, so in the code we compute these derivatives by automatic differentiation. To reduce Monte Carlo noise, we still use the AiO product estimator. For each state (k, b, z) we form two independent Monte Carlo evaluations

$$\text{RHS}_a(k, b, z; w), \qquad \text{RHS}_b(k, b, z; w),$$

. We then compute the corresponding action–gradients

$$g_k^a \;:=\; \frac{\partial \text{RHS}_a}{\partial k'}, \quad g_b^a \;:=\; \frac{\partial \text{RHS}_a}{\partial b'}, \qquad g_k^b \;:=\; \frac{\partial \text{RHS}_b}{\partial k'}, \quad g_b^b \;:=\; \frac{\partial \text{RHS}_b}{\partial b'},$$

using a nested `GradientTape`. The FOC loss is then

$$L_{\text{FOC}} \;:=\; \mathbb{E}\Big[\text{stop\_grad}(g_k^a)\, g_k^b \;+\; \text{stop\_grad}(g_b^a)\, g_b^b\Big], \tag{30}$$

where the expectation is over the training batch and Monte Carlo draws. Because g^a and g^b are independent stochastic estimates of the true gradient, (30) is an AiO estimator of the squared gradient up to sampling error, while gradients flow only through g^b. The implementation replaces non–finite entries in g^a, g^b by zeros for numerical robustness.

Our training enforces (i) Bellman consistency between the continuation–value network and the risky–debt environment induced by the current policy (via L_{Bell}), (ii) approximate optimality of the policy (via L_{FOC}), (iii) limited liability (via the softplus mapping), and (iv) internally consistent pricing of risky debt (via the pricing kernel and soft default indicator). In practice, jointly updating policy and value in this way is akin to an actor–critic method and converges to a fixed point approximating the optimal policy and value function for the risky–debt problem.

To illustrate the key differences, we summarize in the following table.

State, controls, constraints, and pricing

	Basic Model	Risky-Debt Model
States	$k \in \mathbb{R}_+$ (capital), z (productivity/demand shock, Markov)	(k, b, z) with $b \in \mathbb{R}$ (net debt; $b < 0$ interpreted as cash)
Controls	k' (equivalently $I = k' - (1 - \delta)k$)	(k', b') (next-period capital and debt)
Cash flow (sources/uses)	$e = \pi(k, z) - \psi(I, k) - I$	$e = (1 - \tau)\pi(k, z) - \psi(I, k) - I + \dfrac{b'}{1 + \tilde{r}} + \dfrac{\tau \tilde{r}\, b'}{(1 + \tilde{r})(1 + r)} - b$
Law of motion	$k' = (1 - \delta)k + I$	same for k', plus choice of b'
Debt pricing	not applicable (no debt pricing)	risky rate $\tilde{r} = \tilde{r}(z, k', b')$ via lender zero-profit
Default	none	Equity value V satisfies $V = \max\{0, \text{going-concern value}\}$; default $\iff V = 0$
Recovery	not applicable	$R(k', z') = (1 - \alpha)\big[(1 - \tau)\pi(k', z') + (1 - \delta)k'\big]$

2.2.2 Results and Discussion

This section reports the numerical performance of the deep–learning solution to the risky–debt model, as well as some economical consistency checks.

Training dynamics and accuracy We train the model for 2000 epochs with a warm–up phase of 400 epochs, during which the state sampling is purely from a coverage distribution and the FOC penalty is kept small. After warm–up, the batch is a mixture of coverage states and on–policy states generated by an ergodic simulation, and the FOC weight is gradually increased.

Figure 4 shows that the total loss falls monotonically from order unity to about 10^{-2} before stabilizing. Because the AiO loss uses products of residuals rather than squared residuals, it occasionally becomes negative; this is expected in the [20] formulation.

To assess accuracy we compute out–of–sample diagnostics on separate sets of coverage and on–policy states. The main statistic is a relative Bellman error,

$$\varepsilon^B(s) = \frac{\big|C(s) - \mathbb{E}[\mathrm{RHS}(s)]\big|}{\big|\mathbb{E}[\mathrm{RHS}(s)]\big| + 10^{-3}},$$

where $C(s)$ is the network's continuation value and $\mathbb{E}[\mathrm{RHS}(s)]$ is a Monte Carlo approximation of the right–hand side of the Bellman equation. We also report an FOC diagnostic $\varepsilon^F(s) = \|\nabla_{(k',b')}\mathrm{RHS}(s)\|^2$ and a limited–liability diagnostic comparing the softplus equity value V with $\max\{0, C\}$.

Table 1 summarizes the final diagnostics. Mean relative Bellman errors are around one percent, while the limited–liability error is negligible. The mean squared FOC residual corresponds to an average first–order violation of about 0.04 in absolute value. Figure 4 also plots the evolution of the coverage and on–policy Bellman errors during training. Both decline from above one to about 10^{-2} and then fluctuate around this level, with on–policy errors slightly below coverage errors.

Table 1: Final accuracy diagnostics

	Coverage states	On–policy states
Mean relative Bellman error	1.13×10^{-2}	8.39×10^{-3}
Max. relative Bellman error	5.96×10^{-2}	8.19×10^{-2}
Mean squared FOC residual	1.78×10^{-3}	1.45×10^{-3}
Max. absolute FOC residual	1.38×10^{-1}	1.40×10^{-1}
Mean relative limited–liability error	4.63×10^{-6}	5.25×10^{-7}
Max. relative limited–liability error	6.80×10^{-4}	7.21×10^{-5}

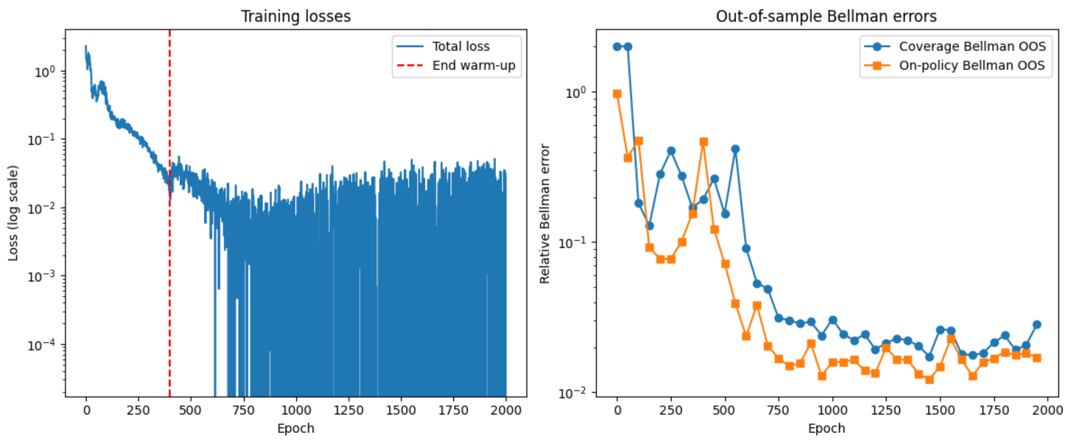

Figure 4: Training loss and out–of–sample Bellman error diagnostics. Notes: The figure shows the evolution of the AiO training loss and the coverage and on–policy relative Bellman errors over epochs. The loss decreases to roughly 10^{-2} and then stabilizes, while both Bellman error measures fall from above one to the order of 10^{-2}. Occasional negative values of the loss are due to the product form of the AiO residuals.

Ergodic distributions of leverage and default probability After training, we simulate an ergodic panel of 256 firms for $T = 400$ periods and discard the first 100 periods as burn–in. Let k_t and b_t denote capital and debt in levels and define period–t leverage by $\ell_t = b_t/k_t$.

Table 2 reports descriptive statistics of the ergodic distribution of leverage and the one–step–ahead default probability π_t^{def}. The mean leverage is about 0.11 with a median of 0.112 and a standard deviation of 0.136. Around 10% of the mass has negative leverage, corresponding to net savings ($b_t < 0$), which are treated as risk–free positions in the pricing kernel. The upper tail is moderate: the 90th percentile of ℓ_t is 0.278, and the maximum observed leverage is 0.596.

The one–period default probability is very low on average ($\mathbb{E}[\pi_t^{\text{def}}] \approx 0.0011$) and highly right–skewed: the median is 0.0007, the 90th percentile 0.0023, and the maximum about 0.0569. If a model period is interpreted as a year, these numbers are of the same order of magnitude as historical investment–grade default rates and are consistent with the low to moderate leverage levels implied by the calibration.

Figure 5 displays histograms of the ergodic distributions of ℓ_t and π_t^{def}. Leverage is approximately bell–shaped and centered around 0.1–0.2, while default probabilities are concentrated close to zero with a thin upper tail. These features are consistent with a dynamic trade–off between tax benefits and expected bankruptcy costs in a setting with costly external equity, as emphasized in [3].

Table 2: Ergodic leverage and default probability statistics

Statistic	Leverage $\ell_t = b_t/k_t$	Default probability π_t^{def}
Mean	0.1060	0.0011
Median	0.1118	0.0007
Std. dev.	0.1363	0.0014
10th pct.	−0.0731	0.0002
50th pct.	0.1118	0.0007
90th pct.	0.2778	0.0023
Maximum	0.5965	0.0569

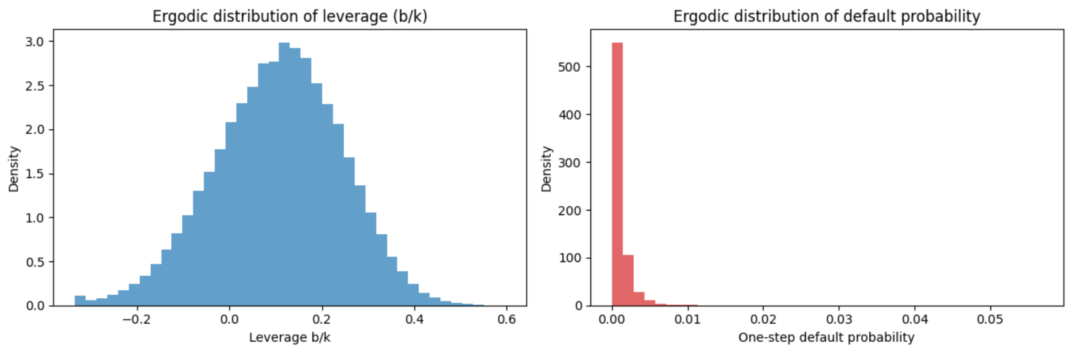

Figure 5: Ergodic distributions of leverage and default probability. Notes: The figure plots histograms of the simulated ergodic distribution of leverage $\ell_t = b_t/k_t$ (left panel) and one–period default probability π_t^{def} (right panel). Leverage is concentrated between 0 and 0.3 with a small negative left tail, while default probabilities are highly concentrated near zero with a thin upper tail.

Cross–sectional patterns on the (k, b) grid To visualize the cross–sectional implications of the policy and value functions, we compute heat maps on a (k, b) grid with productivity fixed at $z = 1$; see Figure 6. The left panel plots the equity value $V(k, b, z = 1)$. As expected, equity value is increasing in capital and decreasing in debt, with smooth iso–value contours and no spurious local extrema. On this grid, V is strictly positive, which is consistent with the low average default probabilities: default occurs primarily in states with low future productivity, rather than in the deterministic slice $z = 1$.

The middle panel of Figure 6 shows the one–step–ahead default probability as a function of (k, b) at $z = 1$. The probability is highest in regions with low capital and high debt, and it decreases monotonically in k and (up to Monte Carlo noise) in the opposite direction of b. Quantitatively the values on this slice are small (around 10^{-3}), in line with the aggregate ergodic statistics.

The right panel plots next–period leverage $\ell' = b'/k'$ implied by the policy network, again for $z = 1$. Leverage is increasing in current b and decreasing in current k, with typical values between 0 and 0.35 and no region where ℓ' approaches unity or explodes. This indicates that the learned policy does not exploit the boundaries of the state space in an unreasonable way and is compatible with a stable ergodic distribution.

Debt policy and partial adjustment Finally, we examine a one–dimensional policy slice $b' = b'(b)$ holding $k = 1.5$ and $z = 1$ fixed; see Figure 7. The debt policy is increasing in current debt, but it lies well below the 45° line: for b in $[0, 2]$, next–period debt rises only from about 0.18 to 0.40. This implies that the firm adjusts debt toward an internal target

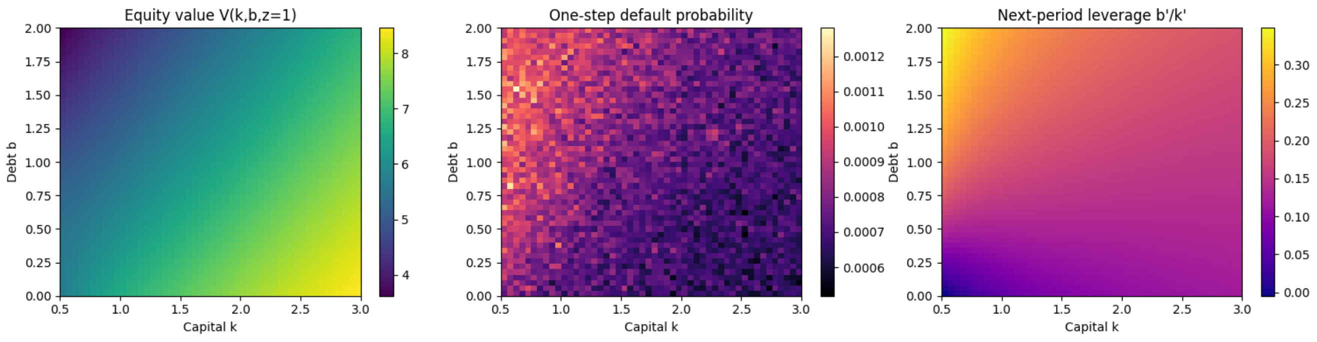

Figure 6: Cross–sectional heat maps on the (k, b) grid at $z = 1$. Notes: The panels show (from left to right) the equity value $V(k, b, 1)$, the one–period default probability, and next–period leverage $\ell' = b'/k'$ implied by the policy functions. Equity increases with capital and decreases with debt; default risk is concentrated in the region of low k and high b; and future leverage is higher for high current debt and low current capital, but remains in a moderate range.

rather than fully inheriting past leverage. In other words, the model generates partial adjustment and mean reversion in leverage, a prominent empirical feature emphasized in [3].

Overall assessment From a numerical perspective, the deep–learning implementation achieves Bellman and FOC residuals of economically small magnitude, both in coverage and along the ergodic distribution. The implied leverage and default probability distributions are internally consistent and fall in plausible ranges given the current parameter choice. Cross–sectional patterns of equity value, default risk, and future leverage on the (k, b) grid exhibit the expected monotonicities, and the debt policy slice reveals realistic partial adjustment toward a target leverage ratio.

In sum, the results suggest that the AiO deep–learning approach of [20] can solve the risky–debt model of [3] with reasonable accuracy and efficiency, producing policy functions and ergodic implications that are qualitatively and quantitatively in line with dynamic trade–off theory.

2.2.3 Issues and practices in NN and training design

As we have already noticed, the DL implementation here is quite different from conventional DL training in most data science applications using real-world data. Thus, in this section, we summarize the issues and practices that we have considered or need to be aware of in our DL implementation for the dynamic models discussed in this report.

- **State scaling and parameterization**
  - Use log states to stabilize gradients and enforce positivity:
    - Basic model: inputs $(\log, \log z)$.
    - Risky debt: inputs $(\log k,\ b/k,\ \log z)$ to work with leverage directly.
  - Parameterize controls in scale-free form:
    - Basic model: predict $\iota = I/k$ and recover $k' = (1 - \delta + \iota)k$.
    - Risky debt: predict k' and b' directly, but always feed states in normalized form.
- **Feasibility-by-construction output maps**

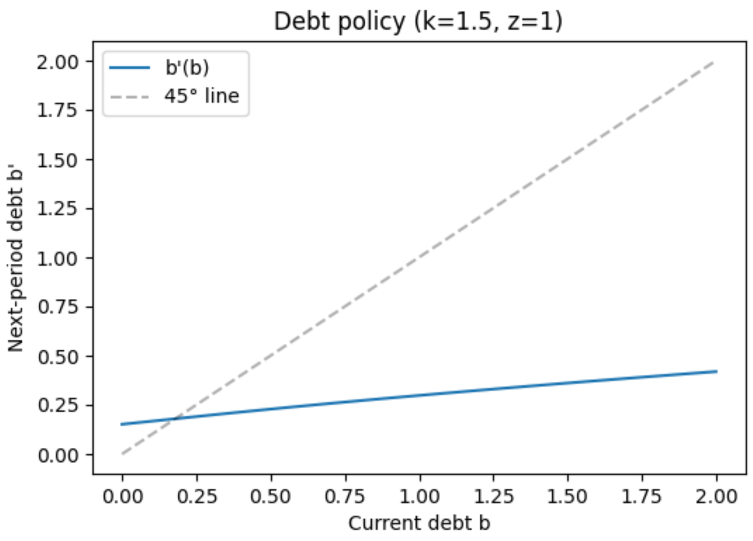

Figure 7: Debt policy function $b'(b)$ at $k = 1.5$ and $z = 1$. Notes: The solid curve shows optimal next–period debt as a function of current debt for fixed capital and productivity. The dashed 45° line indicates full persistence. The gap between the two illustrates partial adjustment of leverage toward an internal target.

  - Basic model: map raw score s to bounded $\iota(s)$ via

$$\iota(s) = \iota_{\text{min}} + \frac{\iota_{\text{max}} - \iota_{\text{min}}}{2}\big(\tanh s + 1\big),$$

    with $\iota_{\text{min}} = -\varepsilon(1-\delta)$ so that $1 - \delta + \iota > 0$ and hence $k' > 0$.
  - Risky debt:
    - Capital: $k' = \texttt{softplus}(\tilde{k}')$.
    - Limited liability: $V \approx \tau_V \log(1 + e^{C/\tau_V})$ as a smooth version of $V = \max\{0, C\}$.
    - Default: $D' \approx \sigma(-C'/\tau_D)$ as a smooth default indicator.
  - Clip only for robustness (e.g. $q \in [0.01, 1/(1+r)]$, state box), not as the primary way to enforce constraints.
- **Sampling of training states**
  - *Coverage sampler*:
    - Draw $(\log k, \log z)$ (and b/k) uniformly from a pre-defined coverage box, with z at its stationary AR(1) distribution.
  - *On-policy / ergodic sampling*:
    - Basic model: replay buffer of visited (k, z) states under current policy; sample from buffer.
    - Risky debt: maintain an on-policy panel (k_t, b_t, z_t) and update it each epoch using current (k', b') and AR(1) shocks, with occasional random resets.
  - *Hybrid schedule*:
    - Warm-up: train only on coverage states.
    - Then ramp up on-policy share linearly to some cap (e.g. 85%), keeping a positive coverage share for global accuracy.
- **Expectation approximation and AiO operator**
  - Training uses Monte Carlo expectations with All-in-One (AiO) structure:
    - Replace $E[\mathcal{R}]^2$ by $E[\mathcal{R}(\varepsilon_1)\mathcal{R}(\varepsilon_2)]$ with independent draws $\varepsilon_1, \varepsilon_2$.

    * Basic model: use two independent shocks plus antithetic variates $(\varepsilon, -\varepsilon)$.
    * Risky debt: two independent calls to the Bellman RHS and pricing kernel inside nested gradient tapes.
  - Evaluation uses accurate Gauss–Hermite quadrature:
    * Default: GH-10 for $E[\mathcal{R}]$ and $E[\text{RHS}]$ on fixed test sets.
    * Robustness: compare GH-15/20 to GH-10 on subsamples.
- **Loss design**
  - Basic model: AiO Euler loss

$$L = \mathbb{E}\big[\mathcal{R}_1(\varepsilon_1)\,\mathcal{R}_2(\varepsilon_2)\big],$$

    where $\mathcal{R}$ is the Euler residual.
  - Risky debt:
    * Bellman AiO loss on continuation value $C(k, b, z)$ (value network), with RHS gradients stopped.
    * FOC AiO loss on (k', b') (policy network) using gradients of RHS wrt controls under two independent shock draws.
    * Overall loss: $L = L_{\text{Bell}} + \lambda_{\text{FOC}} L_{\text{FOC}}$, with λ_{FOC} ramped up after warm-up.
  - Risky debt uses a target value network (Polyak averaging) for RHS and pricing, to stabilize Bellman iteration.
- **Optimization and numerical stability**
  - Optimizer: Adam with small fixed learning rate; single or few Monte-Carlo draws per state (SGD style).
  - Gradient clipping: clip all gradients to a fixed norm (e.g. 1.0); zero out non-finite gradients from nested tapes.
  - Anneal soft temperatures (τ_V, τ_D) during training to move from smooth to sharp limited-liability / default.
  - Set global seeds for Python/NumPy/TF for reproducibility; generate evaluation coverage sets with independent RNG.
- **Evaluation and convergence diagnostics**
  - No separate validation set or early stopping; convergence is judged by *economic* residuals.
  - Two fixed GH-based test sets:
    * *Coverage test set*: uniform over coverage box.
    * *On-policy test set*: ergodic sample under trained policy.

3 Part 2 Estimation methods: SMM/GMM

References

[1] Jr. Lucas, Robert E. Asset prices in an exchange economy. *Econometrica*, 46(6):1429–1446, 1978.

[2] Jr. Lucas, Robert E. and Edward C. Prescott. Investment under uncertainty. *Econometrica*, 39(5):659–681, September 1971.

[3] Ilya A Strebulaev, Toni M Whited, et al. Dynamic models and structural estimation in corporate finance. *Foundations and Trends in Finance*, 6(1–2):1–163, 2012.

[4] George Tauchen. Finite state markov-chain approximations to univariate and vector autoregressions. *Economics Letters*, 20(2):177–181, 1986.

[5] George Tauchen and Robert Hussey. Quadrature-based methods for obtaining approximate solutions to nonlinear asset pricing models. *Econometrica*, 59(2):371–396, 1991.

[6] Ronald A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA, 1960.

[7] John Rust. Numerical dynamic programming in economics. In H. M. Amman, David A. Kendrick, and John Rust, editors, *Handbook of Computational Economics*, volume 1, pages 619–729. Elsevier, Amsterdam, 1996.

[8] Mario J. Miranda and Paul L. Fackler. *Applied Computational Economics and Finance.* MIT Press, Cambridge, MA, 2002.

[9] Nancy L. Stokey, Jr. Lucas, Robert E., and Edward C. Prescott. *Recursive Methods in Economic Dynamics.* Harvard University Press, Cambridge, MA, 1989.

[10] II Coleman, Wilbur John. Solving the stochastic growth model by policy-function iteration. *Journal of Business Economic Statistics*, 8(1):27–29, January 1990.

[11] II Coleman, Wilbur John. Equilibrium in a production economy with an income tax. *Econometrica*, 59(4):1091–1104, July 1991.

[12] Christopher D. Carroll. The method of endogenous gridpoints for solving dynamic stochastic optimization problems. *Economics Letters*, 91(3):312–320, June 2006.

[13] Francisco Barillas and Jesus Fernandez-Villaverde. A generalization of the endogenous grid method. *Journal of Economic Dynamics and Control*, 31(8):2698–2712, August 2007.

[14] Lawrence J. Christiano and Jonas D. M. Fisher. Algorithms for solving dynamic models with occasionally binding constraints. *Journal of Economic Dynamics and Control*, 24(8):1179–1232, July 2000.

[15] Giulio Fella. A generalized endogenous grid method for non-smooth and non-concave problems. *Review of Economic Dynamics*, 17(2):329–344, April 2014.

[16] Kenneth L. Judd. Projection methods for solving aggregate growth models. *Journal of Economic Theory*, 58(2):410–452, December 1992.

[17] Kenneth L. Judd. *Numerical Methods in Economics.* MIT Press, Cambridge, MA, 1998.

[18] S. Borağan Aruoba, Jesús Fernández-Villaverde, and Juan F. Rubio-Ramírez. Comparing solution methods for dynamic equilibrium economies. *Journal of Economic Dynamics and Control*, 30(12):2477–2508, December 2006.

[19] Kenneth L. Judd, Lilia Maliar, Serguei Maliar, and Rafael Valero. Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. *Journal of Economic Dynamics and Control*, 44:92–123, 2014.

[20] Lilia Maliar, Serguei Maliar, and Pablo Winant. Deep learning for solving dynamic economic models. *Journal of Monetary Economics*, 122:76–101, September 2021.