CESifo WORKING PAPERS

9161
2021

June 2021

# Exploiting Symmetry in High-Dimensional Dynamic Programming

*Mahdi Ebrahimi Kahou, Jesús Fernández-Villaverde, Jesse Perla, Arnav Sood*

**Impressum:**

CESifo Working Papers
ISSN 2364-1428 (electronic version)
Publisher and distributor: Munich Society for the Promotion of Economic Research - CESifo GmbH
The international platform of Ludwigs-Maximilians University's Center for Economic Studies and the ifo Institute
Poschingerstr. 5, 81679 Munich, Germany
Telephone +49 (0)89 2180-2740, Telefax +49 (0)89 2180-17845, email office@cesifo.de
Editor: Clemens Fuest
https://www.cesifo.org/en/wp
An electronic version of the paper may be downloaded
· from the SSRN website: www.SSRN.com
· from the RePEc website: www.RePEc.org
· from the CESifo website: https://www.cesifo.org/en/wp

CESifo *Working Paper No. 9161*

# Exploiting Symmetry in High-Dimensional Dynamic Programming

## Abstract

We propose a new method for solving high-dimensional dynamic programming problems and recursive competitive equilibria with a large (but finite) number of heterogeneous agents using deep learning. The „curse of dimensionality“ is avoided due to four complementary techniques: (1) exploiting symmetry in the approximate law of motion and the value function; (2) constructing a concentration of measure to calculate high-dimensional expectations using a single Monte Carlo draw from the distribution of idiosyncratic shocks; (3) sampling methods to ensure the model fits along manifolds of interest; and (4) selecting the most generalizable over-parameterized deep learning approximation without calculating the stationary distribution or applying a transversality condition. As an application, we solve a global solution of a multi-firm version of the classic Lucas and Prescott (1971) model of „investment under uncertainty.“ First, we compare the solution against a linear-quadratic Gaussian version for validation and benchmarking. Next, we solve nonlinear versions with aggregate shocks. Finally, we describe how our approach applies to a large class of models in economics.

JEL-Codes: C450, C600, C630.

Keywords: dynamic programming, deep learning, breaking the curse of dimensionality.

*Mahdi Ebrahimi Kahou*
*University of British Columbia*
*Vancouver / BC / Canada*
*mekahou@alumni.ubc.ca*

*Jesús Fernández-Villaverde*
*University of Pennsylvania*
*Philadelphia / PA / USA*
*jesusfv@econ.upenn.edu*

*Jesse Perla*
*University of British Columbia*
*Vancouver / BC / Canada*
*jesse.perla@ubc.ca*

*Arnav Sood*
*Carnegie Mellon University*
*Pittsburgh / PA / USA*
*arnav@arnavsood.com*

June 21, 2021
We would like to thank Benjamin Bloem-Reddy, Samira Ebrahimi Kahou, Kevin Leyton-Brown, Yaniv Plan, Kevin Song, and Vincent Michalski for useful comments.

## 1 Introduction

We propose a new method for solving high-dimensional dynamic programming problems and recursive competitive equilibria with a large (but finite) number of heterogeneous agents. The key to our approach is to exploit symmetry in dynamic programming and concentration of measure to build a deep neural network that is particularly efficient and easy to train.

Models with many finite agents are very popular in economics. Think about models of industry dynamics with many firms, models with many regions or countries, or models with many households. In fact, one can argue that we only use models with a continuum of agents in macroeconomics or international trade as a convenient abstraction: the U.S. economy had around 128.45 million households in 2020, not an infinite number of them.

Unfortunately, dealing with multi-agent models is challenging, as we quickly bump into the "curse of dimensionality" (Bellman, 1958, p. ix). As soon as we have more than a few agents, it becomes nearly impossible to solve these models or take them to the data. As noted by Powell (2007), there are two sources for the "curse of dimensionality" in dynamic programming. First, the cardinality of the state space is enormous. Second, it is difficult to compute highly-dimensional conditional expectations.

To illustrate these two sources, consider a simple economy. We start with the case that there is only one agent that produces a single perishable good. Since the good is perishable, there are no savings. The agent can have high or low productivity in every period and work with high or low effort. Productivity has persistence: high productivity today makes high productivity tomorrow more likely. Work effort yesterday affects the disutility of effort today: high effort yesterday means higher disutility today. The dynamic programming problem of this agent is trivial. The state has a cardinality of 4 (low productivity today with low work effort yesterday, low productivity today with high work effort yesterday, high productivity today with low work effort yesterday, and high productivity today with high work effort yesterday). The agent only needs to decide whether to work with high or low effort in each of these 4 states.

Let us next modify the model. We will have $N$ agents, with the same environment as before, except that now each agent produces a differentiated, perishable good but consumes all the $N$ goods in the economy (to simplify, we assume that each agent behaves parametrically with respect to all prices, including the price of the good she produces). By introducing many agents, we have gone from having 4 states to having $4^N$ states. Each agent needs to keep track of the individual states of all the $N$ agents in the economy: these states determine the current supply and prices of goods and the (conditional) future distribution of productivities and work effort. Without that information, the agent cannot make an optimal decision regarding her work effort today (and hence, her disutility tomorrow). If the economy has 133 agents (a small village), the cardinality of the state space is $4^{133}$, which is larger than the Eddington number, the estimate of protons in the universe, of $10^{80}$.

However, not only must the agent keep track of $4^N$ states, but she also must compute the conditional expectation of her continuation value function over these $4^N$ states. This computation requires either large sums (if the states, as here, are finite) or complex highly-dimensional integrals (if the states are continuous). Both tasks become prohibitively expensive as soon as $N$ grows above the single digits.

This simple example might seem discouraging: we cannot even compute a simple economy! But it also suggests intriguing possibilities. For example, does an agent need to keep track of the states of every single agent? Under an appropriate symmetry assumption, it might only be necessary to keep track of how many agents are in each of the 4 individual states. Imagine, for instance, that all the goods enter into the agent's utility function with equal weight. The only information the agent requires, beyond her own states, is how many agents are in each of the four possible states. The indices (e.g., whether agent 45 or 79 is the one who has high productivity today with high or low work effort yesterday) are irrelevant. The cardinality of the new state space when $N = 133$ is $\approx 1.56$ million points, something we can easily handle with a laptop.

We will show that this result is much more general: it holds in each case where we can find symmetry in the problem. Since we can include heterogeneity among the agents (e.g., different discount factors or utility weights) in the definition of the states, there will be plenty of cases where symmetry holds. In fact, most heterogeneous agent models in economics have a latent but obvious symmetry imposed by the presence of a Walrasian auctioneer. In general equilibrium, the auctioneer collects all the excess supply of each agent, aggregates them, and sells them. This aggregation removes the indices of agents in the economy and the solution of the model is invariant under all the permutations of other agents' states. Mathematically speaking, any function representing the solution belongs to the family of symmetric functions, i.e., the solution is a function on a set rather than on a vector space (see Macdonald, 1998, for an introduction to symmetric functions).

Notice that, for symmetry to hold, we do not require agents to have the same state values. We only require that they behave the same when they happen to have the same state values. To formalize this idea, we will introduce the notion of permutation-invariant dynamic programming and describe how we can use powerful results to represent functions invariant to permutations.

But our argument goes even further. Suppose we have many agents and there is an underlying symmetry assumption. Will we not have something that resembles a law of large numbers when computing the conditional expectation of the next-period value function? Intuitively, suppose the continuation utility does not depend too much on the states of any given agent. In that case, the continuation utility will become essentially constant for all draws of (independent) idiosyncratic shocks. Hence, we can calculate it with a single Monte Carlo draw from the distribution of idiosyncratic shocks.

We can show this possibility more carefully by relying on some recent developments in high-dimensional probability theory, concentration of measure.[1] Loosely speaking, concentration of measure states that a "well-behaved" (Lipschitz continuous) function that depends on lots of independent random variables is essentially constant. Applying this idea to the expected continuation value function, if we have many agents and their idiosyncratic shocks are independent, each draw from the distribution of idiosyncratic shocks will deliver, approximately, the same result. In models with a continuum of heterogeneous agents, we use an extreme form of this result: usually, we assume that a law of large numbers holds in the economy regarding the idiosyncratic shocks.[2]

The beauty of this result is that it reduces an $N + 1$ dimensional integration (each of the $N$ idiosyncratic shocks plus the aggregate shock) to a two-dimensional integration: the idiosyncratic shock of the agent of interest and the aggregate shock. The argument works, a fortiori, when the idiosyncratic shock is a multidimensional vector, not a scalar. This drastic dimensionality reduction allows us to use deterministic integration methods such as Gaussian quadrature, as opposed to Monte Carlo methods, on the remaining variation in the aggregate shock. In other words: while normally we think of a high-dimensional state space as making integrals more difficult, we show that a large number of shocks make expectations easier to calculate.

Finally, we train the neural network that implements the permutation-invariant dynamic programming by choosing training points inspired by the symmetry of the problem. For instance, consider an economy where the profit is a function of linear aggregate demand. In this case, a firm's profit is invariant under the permutation of other firms' production level: the profit isoquants are hyperplanes in a simplex. Hence, the sampling is reduced to a two-dimensional problem: sampling from the space of the firm of interest and the space of hyperplanes in a simplex. In this paper, we can sample from those hyperplanes by simulating the aggregate dynamics from initial conditions of interest—and using the equilibrium policy for the dynamics of each agent. In such a way, our sampling process provides an especially accurate approximation starting from a finite set of (distributional) initial conditions.

With all these results, we define a general class of recursive competitive equilibrium models with heterogeneous agents, called *permutation-invariant economies*, and establish that the value function and optimal policy of each agent are invariant under permutation of other agents in the economy. We provide a theorem on how to represent permutation-invariant functions that can lead to a substantial reduction in the dimensionality of the state space. This functional

[1]See Vershynin (2018) and Barvinok (2005) for an introduction to high-dimensional probability theory and concentration of measure.

[2]As we will explain in detail at the end of Section 3, drawing from the distribution of idiosyncratic shocks, even just once, is the right thing to do. Instead, setting the shocks to zero (as a naive intuition would suggest if the idiosyncratic shocks are Gaussian) would miss the typical set of the distribution, which in high dimensions does not include the mode of the distribution (i.e., zero). Also, notice that our argument of drawing once from the distribution of idiosyncratic shocks, thanks to concentration of measure, is different from Judd et al. (2017), who propose precomputing the integrals required in Bellman or Euler equations.

representation sheds light on why in some heterogeneous agent models, such as Krusell and Smith (1998), the first moment of the aggregate distribution is enough to explain the perceived law of motion, and provides an extension for models that requires more information on the distribution of aggregates. The spirit of our method is, therefore, to take the 'big K, little k' approach and generalize it to a 'many big K, one little k' where each 'k' can accurately forecast functions of the evolution of the other 'K' that are invariant to relabeling and permutation.

We illustrate our arguments with a variation of the classic Lucas and Prescott (1971) model of investment under uncertainty. We pick this model for two reasons. First, this model generates one of the simplest nontrivial recursive competitive equilibria. For instance, Ljungqvist and Sargent (2018, ch. 7) use this model to introduce recursive competitive equilibria to students. Thus, the model is very well understood. Second, a particular case of this model, when the pricing function is linear, has a known linear-quadratic (LQ) solution, which gives us a classical optimal control solution against which we can benchmark our results and verify the numerical error. The classical control solution (which ignores symmetry) requires computations that scale with the cube of the number of variables (i.e., $\mathcal{O}(N^3)$), whereas our approximation method is independent of the scale (i.e., $\mathcal{O}(1)$ for reasonable $N$).

Furthermore, we can solve the global solution of the LQ model by only fitting it to two data points, regardless of the number of firms. Given that the closed-form model with linearity only requires two parameters, this should not be a surprise. The interesting point is that we can do so even in cases where the guessed solution is heavily over-parameterized (e.g., if we pretend not to know the exact functional form of the solution and, instead, we try to fit a nonlinear solution with 10,000 parameters). An even more exciting result is that this solution is found without directly applying a transversality condition or even using a stationary solution as a special case of a boundary. Despite possibly tens of thousands of parameters, the process finds the "right" equilibrium, which generalizes well outside of the small number of points used in the approximation.[3]

We also use the LQ solution to show the concentration of measure and permutation invariance exactly. While, ex-post, these results should not be surprising, they are more general than one might suspect. To show this, we solve a non-LQ version of the model using neural networks to represent the optimal policy and value function. This strategy is justified by the observation that neural networks are universal approximators, i.e., they can get $\epsilon$-close to any continuous function (Cybenko, 1989, and Hornik, 1991). Also, neural networks are flexible in their design, and we have efficient and fast software (e.g., Pytorch and TensorFlow) and hardware (e.g., GPUs) platforms for training them. Notice, nonetheless, that the neural networks do not cause reduction

[3]The mathematical theory of highly over-parameterized models (when the number of parameters $\gg$ the data points) shows that this class of models, when fitted with stochastic optimization algorithms, generalize by converging to a minimum-norm solution within the space of approximating functions (see Belkin et al., 2019, and Advani et al., 2020). This minimum-norm solution coincides with the equilibrium we are interested in.

of dimensionality: symmetry does.

To frame our paper within the literature, remember that symmetry is a property of a mathematical system that remains invariant under a set of operations or transformations. The existence of symmetry in a problem can lead to a reduction in dimensionality and, thus, alleviate the curse of dimensionality, enhance accuracy, and reduce complexity. For instance, symmetry plays a crucial role in reducing of dimensionality in studying ordinary and partial differential equations (Walcher, 2019, and Bluman et al., 2010).

Symmetry has been applied in economics before. For instance, Hartford et al. (2016) use permutation-equivariant symmetry in order to reduce dimensionality to study the best responses in a normal form game. Sato (1985) and Samuelson (1990) use the Lie groups symmetry to reduce the dimensionality of income space required to describe aggregate demand. Symmetry is also implicitly used every time economists solve models with a mean-field approximation whenever a continuum of agents is assumed.[4]

But it has been the recent developments in the machine learning literature that have proved that exploiting symmetry can significantly alleviate the curse of dimensionality in regression and classification problems. This has been done for data augmentation and designing architectures for neural networks.

Regarding data augmentation, we can create pseudo-data that reflect the symmetry of the problem. For instance, if the function of interest is invariant to any permutation of its inputs, the pseudo-data are all the permutations of independent variables with the same dependent variable. Chen et al. (2019) provide a comprehensive group-theoretical treatment of data augmentation. In terms of architecture design, we can design a parametric approximating family of functions to preserve the symmetry of the problem, such as a neural network that is invariant under the rotation of its input arguments. This design reduces the size of the set of parameters in the optimization problem and alleviates over-fitting.[5]

Our use of permutation-invariant symmetry falls under the architectural approach. We follow the recent developments in the representation of functions that are invariant under permutation groups. Zaheer et al. (2017) establish that using permutation invariance and equivariant symmetry can lead to a significant reduction in dimensionality. Yarotsky (2018) provides a detailed treatment of representing permutation-invariant functions with neural networks and provides a representation that is a universal approximator for permutation-invariant functions. Wagstaff et al. (2019) illustrate cases where this functional representation fails to reduce dimensionality. Our choice of functional representation is also closely linked to the literature of symmetric

[4]The Walrasian auctioneer uses symmetry, which manifests as an equilibrium condition on a distribution rather than individual states. Thus, agents can use the forecast of the distribution to forecast the scalar prices they care about. Similarly, without aggregate shocks, the evolution of the distribution is deterministic. Hence, agents can forecast the scalar functions of the distribution imposed by the auctioneer perfectly.

[5]Lyle et al. (2020) compares the benefits of data augmentation versus architectural design for deep neural networks.

functions, see Prasad (2018) and Zabrocki (2015).

Exploiting symmetry in Markov decision processes (MDPs) to reduce dimensionality was introduced by Ravindran and Barto (2001), who treat symmetry as a homomorphism between two MDPs that commutes with transition dynamics and preserves the reward function. Similarly, in our work, the permutation symmetry can be thought of as a homomorphism between two economies. However, earlier work regarding symmetry has mainly focused on discovering symmetries in MDPs rather than exploiting a priori knowledge of symmetry; see Narayanamurthy and Ravindran (2008).[6]

Using neural networks as a family of approximating functions to solve a stochastic dynamic problem in economics can be traced back to Duffy and McNelis (2001). Due to recent substantial hardware and software advances in training deep neural networks, this method has become popular in solving dynamic models in economics. See, among others, Maliar et al. (2019), Fernández-Villaverde et al. (2019), Duarte (2018), and Azinovic et al. (2019).

The rest of the paper is organized as follows. Section 2 formally introduces the ideas of permutation-invariant dynamic programming and concentration of measure. Section 3 presents our application. Section 4 solves the special case with a linear inverse-demand using the classic control solution using LQ Gaussian techniques. Section 5 introduces our deep learning implementation to solve both the LQ and non-LQ cases. Finally, Section 6 explores the full generality of these techniques.

## 2 Permutation-Invariant Dynamic Programming and Concentration of Measure

This section describes the two ideas that will efficiently tackle highly-dimensional dynamic programs: permutation-invariant dynamic programming and concentration of measure. Permutation invariance will give us a structure in the solution of dynamic programming problems that we can exploit, even when the number of states is large. Concentration of measure will let us deal with the complex conditional expectations that appear in highly-dimensional dynamic programs. In Section 3, we will present a model of investment under uncertainty where we can apply these two ideas.

But, before introducing permutation invariance and concentration of measure, we briefly discuss the concept of a permutation matrix and the associated symmetric group of permutation matrices under matrix multiplication.

[6]There is also an emerging literature in deep reinforcement learning that uses symmetry to reduce dimensionality (van der Pol et al., 2020).

## 2.1 The symmetric group of permutation matrices

We denote a column vector of length $N$ of 1s or 0s as $\mathbf{1}_N$ and $\mathbf{0}_N$ respectively. Similarly, we write matrices of size $M \times N$ of 0s or 1s as $\mathbf{0}_{MN}$ and $\mathbf{1}_{MN}$ respectively, and an identity matrix of size $N$ as $\mathbf{I}_N$. Notice that $\mathbf{1}_{NN} = \mathbf{1}_N \mathbf{1}_N^\top$, a result that will be helpful momentarily.

A permutation matrix is a square matrix with a single 1 in each row and column and zeros everywhere else. These matrices are called "permutation" because, when they premultiply (postmultiply) a conformable matrix $A$, they permute the rows (columns) of $A$.

Let $\mathcal{S}_N$ be the set of all $n!$ permutation matrices of size $N \times N$. For example,

$$S_2 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}.$$

The first element of $S_2$ is the identity matrix, which leaves rows and columns of a multiplied matrix $A$ unchanged, while the second element swaps the rows or columns of $A$. The set $\mathcal{S}_N$ forms a symmetric group under matrix multiplication. See Appendix A for further notation.

## 2.2 Permutation-invariant dynamic programming

We want to study a class of dynamic programming problems where the aggregate state vector $X$ can be permuted. To do so, we start by defining a general dynamic programming problem for environments with many agents and using recursive notation.

**Definition 1** (A 'big $X$, little $x$' dynamic programming problem)**.** *Consider the dynamic programming problem:*

$$v(x, X) = \max_u \left\{ r\big(x, u, X\big) + \beta \mathbb{E}\left[v(x', X')\right] \right\} \tag{1}$$

$$s.t.\ x' = g(x, u) + \sigma w + \eta \omega \tag{2}$$

$$X' = G(X) + \Omega W + \eta \omega \mathbf{1}_N. \tag{3}$$

*Here, $x$ is the individual state of the agent, $X$ is a vector stacking the individual states of all of the $N$ agents in the economy, $u$ is the control, $w$ is random innovation to the individual state of the agent, which gets stacked in $W \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$ where, without loss of generality $w = W_1$, and $\omega \sim \mathcal{N}(0, 1)$ is a random aggregate innovation to all the individual states. Also, $v : \mathbb{R}^{N+1} \to \mathbb{R}$ is the value function, $r : \mathbb{R}^{N+2} \to \mathbb{R}$ is the return function, $g : \mathbb{R}^2 \to \mathbb{R}$ is the deterministic component of the law of motion for $x$, $G : \mathbb{R}^N \to \mathbb{R}^N$ is the the deterministic component of the law of motion for $X$, and $\mathbb{E}\left[\cdot\right]$ is the conditional expectations operator.*

Extending the previous notation to the case where every agent has several state or control variables is straightforward but tedious. To simplify derivations, from now on, we focus on the case where $r(\cdot)$, $g(\cdot)$, and $G(\cdot)$ are differentiable.

**Definition 2** (Permutation-invariant dynamic programming)**.** *A 'big $X$, little $x$' dynamic programming problem is a permutation-invariant dynamic programming problem if, for all $(x, X) \in \mathbb{R}^{N+1}$ and all permutations $\pi \in \mathcal{S}_N$, the reward function $r$ is permutation invariant:*

$$r(x, u, \pi X) = r(x, u, X), \tag{4}$$

*the deterministic component of the law of motion for $X$ is permutation equivariant:*

$$G(\pi X) = \pi G(X), \tag{5}$$

*and the covariance matrix of the idiosyncratic shocks satisfies*

$$\pi\Omega = \Omega\pi. \tag{6}$$

The intuition behind the previous definition is that we are interested in dynamic programming problems where we care, for example, about the distribution of capital among the agents in the economy, but not whether the rich agent is agent 12 or agent 27. If agent 12 swaps her capital with agent 27 (and other relevant states such as her preference shock or income level), the equilibrium dynamics of the economy would still be the same.

The next proposition shows a basic property of permutation-invariant dynamic programming.

**Proposition 1** (Permutation invariance of the optimal solution)**.** *The optimal solution of a permutation-invariant dynamic programming problem is permutation invariant. That is, for all $\pi \in \mathcal{S}_N$:*

$$u(x, \pi X) = u(x, X) \tag{7}$$

*and*

$$v(x, \pi X) = v(x, X). \tag{8}$$

*Proof.* Appendix C.1 □

Building on our previous example, Proposition 1 tells us that, if the equilibrium dynamics of the economy do not change if agent 12 swaps her capital with agent 27, the policy and value functions of agent 1 will not change either. This straightforward property allows us to write the solution of the problem using a remarkable result regarding the representation of permutation-invariant functions.

**Proposition 2** (Representation of permutation-invariant functions)**.** *Let $f : \mathbb{R}^{N+1} \to \mathbb{R}$ be a* *continuous permutation-invariant function under $\mathcal{S}_N$, i.e., for all $(x, X) \in \mathbb{R}^{N+1}$ and all $\pi \in \mathcal{S}_N$:*

$$f(x, \pi X) = f(x, X).$$

*Then, there exist $L \leq N$ and continuous functions $\rho : \mathbb{R}^{L+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^L$ such that:*

$$f(x, X) = \rho\left(x, \frac{1}{N}\sum_{i=1}^{N} \phi(X_i)\right).$$

*Proof.* The proof can be found in Wagstaff et al. (2019). $\square$

This result is powerful: if we can find functions $\rho$ and $\phi$ with a small $L$, we can represent highly-dimensional functions, such as the policy and value functions of models with high $N$ in a much more efficient way and, therefore, exploit the symmetric structure of the dynamic programming problem to solve it quickly and accurately.

## 2.3 Concentration of measure

We complete this section by introducing the idea of concentration of measure. First, we need to have an appropriate notion of boundedness and how it applies to gradients of functions of Gaussian random variables.

**Definition 3** (Bounded functions in $N$)**.** *Let:*

$$\mathcal{H}(M) \equiv \{y \in \mathbb{R}^N : |y_i| \leq M \ \forall i = 1, ..., N\}$$

*be an $N$-dimensional hypercube in $\mathbb{R}^N$. A function $f : \mathbb{R}^N \to \mathbb{R}$ is bounded in $N$ if for every $M$ there exists $K_M$ such that*

$$\sup_{y \in \mathcal{H}(M)} |f(y)| < K_M$$

*where $K_M$ is a constant that does not depend on $N$, but may depend on $M$.*

A simple example of a symmetric function fulfilling Definition 3 is the mean, $f(X) = \frac{1}{N}\sum_{i=1}^{N} X_i$ since $\sup_{y \in \mathcal{H}(M)} |f(y)| < M$ for all $N$. The main purpose of this definition is to reject functions that explode in the number of states, such as the $f(X) = \sum_{i=1}^{N} X_i$ since $\sup_{y \in \mathcal{H}(M)} |f(y)| = NM$.

**Definition 4** (Expected gradient bounded in $N$)**.** *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a bounded function in $N$ and $z \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$ be a normalized Gaussian random vector. The function $f$ has its expected gradient bounded in $N$ if there exists a $C$ such that:*

$$\mathbb{E}\left[\|\nabla f(z)\|^2\right] \leq \frac{C}{N}.$$

*where $C$ does not depend on $N$.*

Loosely speaking, Definition 4 tells us that a function has an expected gradient bounded in $N$ when its value does not change too fast when its Gaussian random arguments vary by a small amount.

The next proposition delivers the intuitive result that, for a large $N$, a function $f(\cdot)$ with an expected gradient bounded in $N$ is essentially a constant. That is, the measure of its values is "concentrating."

**Proposition 3** (Concentration of measure when expected gradients are bounded in $N$)**.** *Suppose $z \sim \mathcal{N}(\mathbf{0}_N, \Sigma)$, where the spectral radius of $\Sigma$, denoted by $\rho(\Sigma)$, is independent of $N$ and $f : \mathbb{R}^N \to \mathbb{R}$ is a function with expected gradient bounded in $N$. Then:*

$$\mathbb{P}\left(\left|f(z) - \mathbb{E}\left[f(z)\right]\right| \geq \epsilon\right) \leq \frac{\rho(\Sigma)C}{\epsilon^2}\frac{1}{N}.$$

*Proof.* Appendix C.2 □

As Ledoux (2001) puts it: "A random variable that depends in a Lipschitz way on many independent variables (but not too much on any of them) is essentially constant."

## 3 An Application: A Model of Investment

To illustrate how permutation-invariant dynamic programming and concentration of measure operate in a common economic application, we propose an extension of the classic model of investment under uncertainty by Lucas and Prescott (1971). We will consider an industry with a large number of firms and follow the recursive formulation in Prescott and Mehra (1980). In contrast to Lucas and Prescott (1971), we will not require that all firms have symmetric states, just that they have symmetric policy functions.

More concretely, we study an industry consisting of $N \geq 1$ price-taking firms, each producing a single good using capital under constant returns to scale. By picking appropriate units, we can consider that a firm $i$ produces output $x$ with $x$ units of capital. We can stack the production (or capital) of the whole industry in the vector $X \equiv [x_1, \ldots x_N]^\top$.

The capital of the firm depreciates at a rate $\delta \in [0, 1]$, is increased by investment $u$, and is shifted by an i.i.d. idiosyncratic shock $w \sim \mathcal{N}(0, 1)$ and a single i.i.d. aggregate shock, $\omega \sim \mathcal{N}(0, 1)$, common to all firms (adding persistence for the shocks is trivial but complicates notations). Thus, the law of motion for capital is:

$$x' = (1 - \delta)x + u + \sigma w + \eta\omega.$$

Due to adjustment frictions, investing $u$ has a cost, in valuation terms, $c(u) = \frac{\gamma}{2}u^2$.

We assume that the (inverse) demand function for the industry is, for some $\nu > 0$ and $\zeta > 0$:

$$p(X) = \alpha_0 - \alpha_1 \left( \frac{1}{N} \sum_{i=1}^{N} \max\{0, x_i\}^{\nu} \right)^{\zeta}. \tag{9}$$

When $\zeta = 1$, this function –an affine transformation of the power-mean– is differentiable almost everywhere. The max operator ensures the inverse demand function is well defined even for negative outputs.

Since we will be looking at competitive equilibria where a firm is a price taker, the firm does not consider the impact of its individual decisions on $p(X)$. Also, all firms use their cash flow to finance their investment, $u$. Hence, the period profits of firm $x$ are:

$$\text{profits} = p(X)x - \frac{\gamma}{2}u^2.$$

The dynamic problem of the firm is to choose $u$ to maximize the expected present discounted value of the firm at discount rate $\beta$.

Since the firm is a price taker, its Bellman equation takes the evolution of $X$ as given and determined by the equilibrium policy of all firms, including itself. In the case of a single representative firm, where $N = 1$, this formulation nests the classic 'big $K$, little $k$' approach.

First, we write such a Bellman equation considering $X$ as a vector, as it is the most common practice in economics.

**Bellman equation with vector** $X$ The firm of interest with capital $x$ takes a symmetric policy $\hat{u}(\cdot, X)$ as given for all other firms in $X$. Without loss of generality, we assume that the firm is the first in the vector, i.e., $x = X_1$. With this assumption, we can write the recursive problem of the firm taking the exogenous policy $\hat{u}(\cdot, X)$ as:

$$v(x, X) = \max_u \left\{ p(X)x - \frac{\gamma}{2}u^2 + \beta \mathbb{E}\left[v(x', X')\right] \right\} \tag{10}$$

$$\text{s.t. } x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{11}$$

$$X_i' = (1-\delta)X_i + \hat{u}(X_i, X) + \sigma W_i + \eta\omega, \quad \text{for } i \in \{2, ..., N\} \tag{12}$$

$$X_1' = (1-\delta)X_1 + \hat{u}(X_1, X) + \sigma w + \eta\omega. \tag{13}$$

Four points deserve further explanation. First, the expectation in equation (10) is taken over the i.i.d. idiosyncratic shocks $\{w, W_2, \ldots W_N\}$ and the aggregate shock $\omega$. Second, equation (11) is just the law of motion of capital of the firm given its control $u$. Third, equation (12) is the forecast the firm makes over the evolution of the distribution of capital of all firms given the exogenous policy. Fourth, equation (13) takes care of the contribution of the firm's own policy to the aggregate state $X$. When $\sigma > 0$, we can let the firm consider that $w = W_1$. When $N$ is sufficiently large, this last point becomes irrelevant for the optimal $u$ and we could remove the special case of that correlated shock, combining (13) into (12). In the case of $N = 1$ and $\mu = 0$, this formulation exactly nests the textbook 'big-K, little-k' example derived in Ljungqvist and Sargent (2018), Section 7.2.

Next, we show that it is easy to rewrite the previous problem considering $X$ as a set. While both formulations look similar cosmetically, the conditions for treating the aggregate distribution as a set rather than a vector are essential to our method.

**Bellman equation with Set $X$** To use our results, we need to show that the investment model described in (10) to (13) satisfies all the conditions of a permutation-invariant dynamic programming problem in Definition 2.

First, to show that the payoff is permutation invariant according to equation (4), it is sufficient for the $p(X)$ function in equation (9) to be permutation invariant. This is easily shown since the function is an affine transformation of a power mean.

Second, the law of motion must be permutation equivariant according to equation (5). This is shown more formally in Appendix D.1, but the intuition is that, conditional on a permutation-invariant $\hat{u}(\cdot, X)$, we can reorder any of the indices in equations (12) and (13), i.e., $X'_i = (1-\delta)X_i + \hat{u}(X_i, X)$ to $X'_j = (1-\delta)X_j + \hat{u}(X_j, \pi X)$ for the permutation matrix $\pi$ since $\hat{u}(X_j, \pi X) = \hat{u}(X_j, X)$.

Finally, to show that the covariance matrix fulfills condition (6) from equations (12) and (13), notice that the idiosyncratic shocks of other agents are embodied in the identity matrix times $\sigma$. Therefore, for all permutation $\pi \in \mathcal{S}_N$, $\pi\sigma\mathbf{I}_N = \sigma\mathbf{I}_N$.

Hence, by Proposition 1, the solution to the investment model $(u, v)$ is permutation invariant and can be treated as a function on sets rather than a vector space (i.e., the ordering of the inputs does not matter). If we create a tuple of the states and the corresponding shocks as $(X, W)$, the set equivalent of equations (12) and (13) becomes:

$$\begin{aligned} X' =& \left\{(1-\delta)\hat{x} + \hat{u}(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W) \setminus (x, w)\right\} \\ &\cup \left\{(1-\delta)x + \hat{u}(x, X) + \sigma w + \eta\omega\right\}. \end{aligned} \tag{14}$$

As before, the special case of the $x$ in expression (14) is to ensure the firm does not take its choice into account when forecasting the evolution, but can still allow for the correlation of shocks when calculating expectations. While this is important for correctness with very small $N$ –and in particular nesting the $N = 1$ case– it has almost no quantitative impact as $N$ increases. For that reason, we will drop the special case in further calculations in this section, effectively decoupling $x$ from the set $X$. Since the forecast $X'$ is only used for expectations, this is equivalent to assuming that the firm does not take into account that its contribution to $X$ is correlated with its own shock when calculating expectations of $X'$.

Using this simplification, we can rewrite equations (10) to (13) as:

$$v(x, X) = \max_u \left\{ p(X)x - \frac{\gamma}{2}u^2 + \beta\mathbb{E}\left[v(x', X')\right] \right\} \tag{15}$$

$$\text{s.t. } x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{16}$$

$$X' = \left\{(1-\delta)\hat{x} + \hat{u}(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W)\right\}. \tag{17}$$

The expectation in (15) is taken over $w \in \mathbb{R}$, $\omega \in \mathbb{R}$, and $W \in \mathbb{R}^N$, all independent normal shocks. This reformulation allows us to solve for the equilibrium with functions on sets rather than vectors and to use the representation in Proposition 2.

Since the problem fulfills the requirements of Definition 2 for any permutation-invariant $\hat{u}(\cdot, \cdot)$, Proposition 1 tells us that we have a permutation-invariant optimal solution. Since in the symmetric equilibrium every agent solves the same policy, the $\hat{u}(x, X)$ is also permutation invariant, which we needed to ensure for the law of motion, (5), to be equivariant. Thus, we are ready to define a symmetric competitive equilibrium.

**Definition 5.** *An equilibrium is a $v(x, X)$ and $\hat{u}(x, X)$ such that:*

- *Given $\hat{u}(x, X)$, $v(x, X)$ is the value function solving (15) for each agent and $u(x, X)$ is the optimal policy function.*
- *The optimal policy is symmetric, i.e., $u(x, X) = \hat{u}(x, X)$*

In this paper, we focus on problems that the optimal solution is unique and can be found via a well-defined Euler equation, though our techniques are not specific to that class of models.

**Proposition 4.** *For a permutation-invariant $p(X)$, the Euler equation takes the form:*

$$\gamma u(X) = \beta\mathbb{E}\left[P(X') + \gamma(1-\delta)u(X')\right] \tag{18}$$

$$X' = \left\{(1-\delta)\hat{x} + u(X) + \sigma\hat{w} + \eta\omega, \text{ for } (\hat{x}, \hat{w}) \in (X, W)\right\}. \tag{19}$$

*Proof.* See Appendix C.3. □

The economic interpretation of the Euler equation (18) is standard. The firm weighs the marginal cost of an additional unit of investment, $\gamma u(X)$, against the benefit of additional profits from that marginal increase in output at a price $p(X')$ plus the value of investment after depreciation. In this application, the policy becomes independent of $x$. That result is, however, incidental. If we had some curvature on $x$ in the revenue function (as we will have in an extension in Section 6.1), the policy will not be independent of $x$.

Later, we will solve our model by minimizing the Euler residuals, using equation (19) for the law of motion of $X'$:

$$\varepsilon(x, X) \equiv \gamma u(x, X) - \beta\mathbb{E}\left[P(X') + \gamma(1-\delta)u(x', X')\right]. \tag{20}$$

**Dimensionality and intuition on the problem structure** While seemingly simple, the dimensionality of the Euler equation (18) is nontrivial. Consider if $N$ was a thousand, then $u(x, X)$ is a policy function on a thousand and one dimensional state-space. Furthermore, even given a $u(x, X)$ function, the expectation is taken over a thousand dimensions of $\hat{w} \in W$ draws and a single dimension for $\omega$, correlated in the evolution of each agent state. It is not feasible to solve this problem with standard methods.

But hope is not lost. First, think about the expectation in the simple case of $\eta = 0$ but $\sigma > 0$: while considering a thousand individual forecasts of $x'$ from (19) is computationally intractable, since $N$ is large, a firm may have an accurate forecast of functions of the distribution of $x'$. In fact, the firm does not care about the individual $x'$ unless its $u(\cdot, X)$ depends a great deal on a single $\hat{x}$ (hence the important of boundedness of the gradient for the concentration of measure).

In the case where firms do not care about the individual $x'$, a forecast of the distribution is good enough. Furthermore, when $\eta = 0$ and $N$ is large, the distribution evolves almost deterministically. Similarly, even when there are aggregate shocks, an agent may have an accurate forecast of functions of the distribution *conditional* on an aggregate shock realization.[7]

Next, consider the structure of $u(\cdot, X)$. Since the price function was invariant to permutations, we know that the policy function will be invariant as well. Hence, even if the firm cannot accurately predict an individual $\hat{x} \in X$ without large amounts of computations, it only needs to predict the general distribution of $X'$ rather than each element in it. This is the essence of the permutation invariance, i.e., $u(\cdot, \pi X) = u(\cdot, X)$ for any permutation $\pi$ of $X$.

The interaction of these provides the intuition for how to simplify the expectations. If $u(\cdot)$ and $p(\cdot)$ are functions of a large number of random $\hat{x}' \in X'$, but do not depend too much on any individual one of them (as defined by the notion of bounded gradients, Definition 4), then the expectation conditional on the aggregate shock may be close to a deterministic function. Conditional on any aggregate shocks, the distribution evolves close to a deterministic function, and the $p(X')$ and $u(\cdot, X')$ are functions of the distribution, so they are fully predictable. In practice, this means that if we do a Monte Carlo expectation drawing random $X'$ consistent with the law of motion, then for a large enough $N$, we only need a single draw of individual shocks according to Proposition 3.

In the case of a linear $p(\cdot)$, we can set the draw of all $\hat{w} \in W$ and $\omega$ to be zero, which relies on certainty equivalence. However, in the more general cases, we will need to draw a random $W$. The intuition is that, in high dimensions, the typical set of a measure does not include the mode of the distribution (i.e., the $N$-dimensional zero). For example, an $N$-dimensional Gaussian random variable concentrates on a hypersphere of radius $\sqrt{N}$, rather than at the origin. Furthermore, the distance between the typical set and the origin increases with the dimensionality of the

[7]The extreme case of this result appears in models à la Aiyagari-Krusell-Smith. With a continuum of agents, we only care about the distribution, not the position of each agent on it.

distribution. See Vershynin (2018) and Fernández-Villaverde and Guerrón-Quintana (2020) for a formal definition of a typical set, examples, and more details.

## 4 A Symmetric Linear-Quadratic Model of Investment

In this section, we focus on the case where the aggregate price $p(X)$ has a linear structure and the revenue has linear dependency on the agents' states, i.e., $\nu = \zeta = 1$. This choice of parameters reduces the problem to a linear-quadratic (LQ) dynamic programming problem as described in Ljungqvist and Sargent (2018, Section 7.2)—except with $N \geq 1$ firms.[8]

The LQ formulation helps us along two dimensions. First, this simple environment lets us formally demonstrate some of the fundamental symmetry and concentration of measure in Section 2 that we will exploit in solving this model with neural network approximations for the non-LQ version of the model. Second, this formulation will provide us with an accurate solution using classical optimal control techniques against which we can benchmark our solution method.

### 4.1 The linear-quadratic regulator formulation

We start by working with a vector. While this is unlike Proposition 4, it simplifies our notation. As before, without loss of generality, assume that the $x$ in (16) is the first element of $X$. Denote the state of all firms and a constant term as $\vec{x} \equiv \begin{bmatrix}1 & x & X^\top\end{bmatrix}^\top \equiv \begin{bmatrix}1 & x & x & X_2 & \dots & X_N\end{bmatrix}^\top \in \mathbb{R}^{N+2}$ and the vector of all Gaussian shocks as $\vec{w} \equiv \begin{bmatrix}\omega & W^\top\end{bmatrix}^\top = \begin{bmatrix}\omega & w & W_2 & \dots & W_N\end{bmatrix}^\top \sim \mathcal{N}(0, \mathbf{I}_{N+1})$.

Next, define $\pi \in \mathcal{S}$ to be a permutation represented by a $(N+2) \times (N+2)$ permutation matrix where the lower-right $N \times N$ block is in $\mathcal{S}_N$, i.e., holding the first and second entries fixed. That is, we are looking at permutations of the form

$$\pi = \left[\begin{array}{cc|c} 1 & 0 & \mathbf{0}_N^\top \\ 0 & 1 & \mathbf{0}_N^\top \\ \hline \mathbf{0}_N & \mathbf{0}_N & \pi_N \end{array}\right], \tag{21}$$

where $\pi_N \in \mathcal{S}_N$ as defined in Section 2.1.

[8] Anderson et al. (1996) offer a detailed mathematical treatment of LQ dynamic programming, dealing with topics such as existence, uniqueness, and convergence.

If we define the matrices:

$$A \equiv \left[\begin{array}{cc|c} 1 & 0 & \mathbf{0}_N^\top \\ 0 & 1-\delta & \mathbf{0}_N^\top \\ \hline H_0 \mathbf{1}_N & \mathbf{0}_N & (1-\delta)\mathbf{I}_N + \frac{H_1}{N}\mathbf{1}_N\mathbf{1}_N^\top \end{array}\right] \tag{22}$$

$$B \equiv \left[\begin{array}{cc|c} 0 & 1 & \mathbf{0}_N^\top \end{array}\right]^\top \tag{23}$$

$$C \equiv \left[\begin{array}{c|c} \eta & \sigma \quad \mathbf{0}_{N-1}^\top \\ \hline \eta \mathbf{1}_N & \sigma \mathbf{I}_N \end{array}\right] \tag{24}$$

$$R \equiv \left[\begin{array}{cc|c} 0 & \frac{\alpha_0}{2} & \mathbf{0}_N^\top \\ \frac{\alpha_0}{2} & 0 & -\frac{\alpha_1}{2N}\mathbf{1}_N^\top \\ \hline \mathbf{0}_N & -\frac{\alpha_1}{2N}\mathbf{1}_N & \mathbf{0}_N\mathbf{0}_N^\top \end{array}\right] \tag{25}$$

$$Q \equiv \left[\frac{\gamma}{2}\right], \tag{26}$$

we can write the problem as an LQ optimal regulator as derived in Appendix D.1.[9]

**Proposition 5** (LQ formulation)**.** *Taking $H_0$ and $H_1$ as given, each firm solves:*

$$v(\vec{x}) = \max_u \left\{-\vec{x}^\top R\vec{x} - u^\top Q u + \beta \mathbb{E}\left[v(\vec{x})\right]\right\} \tag{27}$$

$$s.t. \vec{x}' = A\vec{x} + Bu + C\vec{w} \tag{28}$$

*With solutions characterized by a $P$, $d$, and $F$:*

$$v(\vec{x}) = -\vec{x}^\top P\vec{x} - d \tag{29}$$

$$u(\vec{x}) = -F\vec{x}. \tag{30}$$

*A symmetric recursive competitive equilibrium is defined as $H_0$ and $H_1$ such that:*

$$F = -\left[H_0 \quad 0 \quad \frac{H_1}{N} \quad \frac{H_1}{N} \quad \dots \quad \frac{H_1}{N}\right]^\top = -\left[\begin{array}{cc|c} H_0 & 0 & \frac{H_1}{N}\mathbf{1}_N^\top \end{array}\right]^\top. \tag{31}$$

*Furthermore, both the control $u(\cdot)$ and value function $v(\cdot)$:*

1. *are invariant to permutations $\pi \in \mathcal{S}$ as defined by (21);*

2. *have expected gradients bounded in $N$, according to Definition 4:*

$$\nabla_W u(\vec{x'}) = \frac{\sigma H_1}{N}\mathbf{1}_N; \tag{32}$$

[9]See also Ljungqvist and Sargent (2018, Ch. 5 and 7), in particular, Section 5.3 and exercises 5.11, 5.12, 7.1, and 7.2.

3. *have a concentration of measure according to Proposition 3 when calculating with a Monte Carlo draw rather than certainty equivalence, which leads to —for some $C_v$ independent of $N$:*

$$\mathbb{P}\left(\left|u(\vec{x}') - \mathbb{E}\left[u(\vec{x}') \mid w, \omega, \vec{x}\right]\right| \geq \epsilon\right) \leq \frac{\sigma^2 H_1^2}{\epsilon^2}\frac{1}{N} \tag{33}$$

$$\mathbb{P}\left(\left|v(\vec{x}') - \mathbb{E}\left[v(\vec{x}') \mid w, \omega, \vec{x}\right]\right| \geq \epsilon\right) \leq \frac{\sigma^2 C_v^2}{\epsilon^2}\frac{1}{N}; \tag{34}$$

*Besides, the Euler residual $\varepsilon(X; u)$ is a Gaussian random variable, when calculating with a Monte Carlo draw rather than certainty equivalence, as follows*

$$\varepsilon(X; u) \sim \mathcal{N}\left(0, \frac{\beta^2\sigma^2\left[\alpha_1 - \gamma(1-\delta)H_1\right]^2}{N}\right). \tag{35}$$

*Proof.* See Appendix D.1 for the proof using classic LQ-Gaussian control, Appendix D.2 for permutation invariance, and Appendix D.3 for concentration of measure

As a demonstration of some of the symmetry properties, we will sketch out the proof for $u(\cdot)$ in Proposition 5, where we take the functional form of $F$ as given.

1. Since $F$ as defined in (31) is identical for all but the first two elements, it aligns with the block structure of the permutations. Hence, for any of the $\pi$ in (21), $\pi F = F$, which ensures that $u(\cdot)$ is symmetric in all changes to the $X$ vector.

2. The gradient of an affine function is constant. Hence, for any $\vec{w}$, the policy next period is $u(A\vec{x} + Bu + C\vec{w})$. Given the block structure of $F$ and $C$, we differentiate with respect to $W$ in the $\vec{w}$ to find:

$$\nabla_W u(A\vec{x} + Bu + C\vec{w}) = \frac{\sigma H_1}{N}\mathbf{1}_N \tag{36}$$

   Therefore, the square of the norm of the gradient of the policy can be bounded as:

$$\mathbb{E}\left[\|\nabla_W u(\vec{x}')\|^2\right] = \left(\sum_{i=1}^{N}\frac{\sigma H_1}{N}\right)^2 = \frac{\sigma^2 H_1^2}{N}. \tag{37}$$

3. Since $W \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$, the spectral radius of $\mathbf{I}_N$ is 1 and the concentration of measure follows from (32) and Proposition 3.

□

Recall that, in classic LQ control, certainty equivalence allows us to calculate the policy $u(\cdot)$ setting $\vec{w} = \mathbf{0}_{N+2}$ in the maximization (27), while the variance of the shock only enters into the $d$

of the solution (29). The $d$ can be decomposed into a component related to the $\omega$ aggregate shock and another for the $W$ associated with the finite number of firms. Consistent with intuition, as $N$ goes to infinity, the component of $d$ associated with the idiosyncratic shocks goes to 0. That is, conditioning on the aggregate shock, the firm can get closer and closer to the deterministic forecast of the distribution of $X'$, at which point certainty equivalence is no longer necessary.

Given that we know the exact solution, certainty equivalence gives us a closed-form expression for the numerical error associated with using a single Monte Carlo draw for the expectation, which can serve as a guide for the quality of the approximation in the many cases of interest where certainty equivalence would not hold.

To illustrate some of these ideas, we fix the model parameters at conventional levels. Unless stated otherwise, we will have $\sigma = 0.005, \eta = 0.001, \alpha_0 = 1, \alpha_1 = 1, \beta = 0.95, \gamma = 90, \zeta = 1, \nu = 1$, and $\mu = 0$. In terms of solving for an equilibrium in Proposition 5, we guess an $H_0$ and $H_1$, solve the LQ problem as a matrix Ricatti equation, and then find the fixed point of $H_0$ and $H_1$.

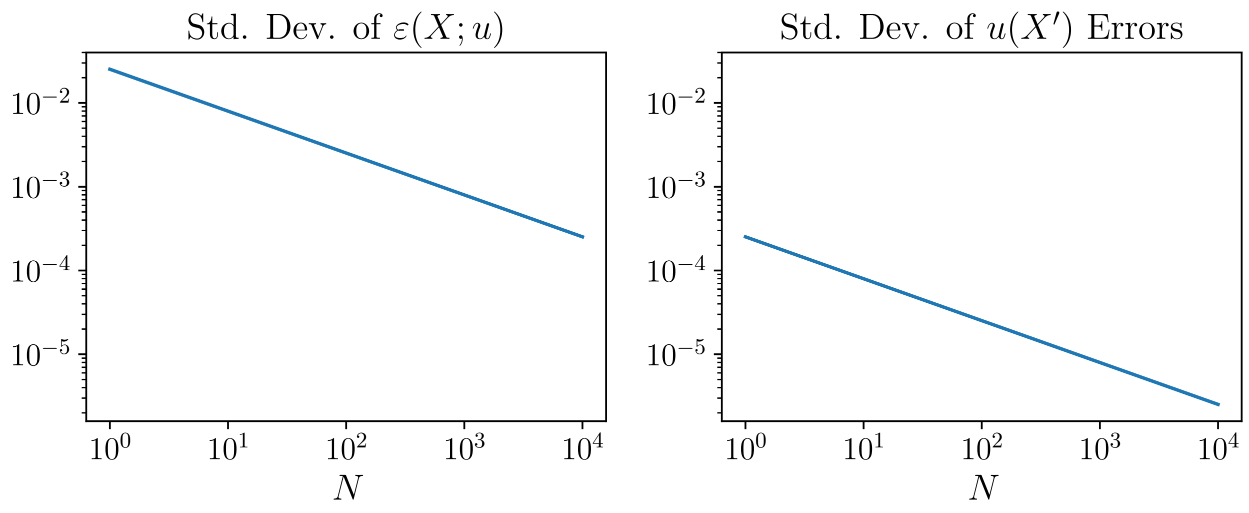

Figure 1: The concentration of Euler residuals $\varepsilon(X; u)$ and the optimal policy $u(X')$ for $\nu = \zeta = 1$.

The left panel of Figure 1 plots the standard deviation of the Euler residuals $\varepsilon(X; u)$ induced by the concentration of measure approximation as we increase the number of firms in the industry and we solve the model with the algorithm described above. By the time we get to 10,000 firms, the standard deviation of the Euler errors is $2.4 \times 10^{-4}$. In the right panel of Figure 1, we draw the standard deviation of the optimal policy $u(X')$ for the linear aggregate prices given the same Monte Carlo approximation and find the standard deviation of the policy error is $2.5 \times 10^{-6}$ for $N = 10,000$ and $2.2 \times 10^{-5}$ for our baseline of $N = 128$.

## 4.2 Why do we need a functional representation?

While the algorithm outlined in the previous subsection works in delivering accurate solutions, as $N$ becomes large, it is too slow, as it requires computations of order $\mathcal{O}(N^3)$). In comparison, the specialized algorithms we will introduce in Section 5 that exploit the symmetry of the dynamic programming problem will only require computations of order $\mathcal{O}(1)$ for reasonable $N$.

For instance, if it is known that optimal policy is permutation invariant, linear, and does not depend on $x_0$, Proposition 2 tells us that:

$$u(x, X) = H_0 + \frac{1}{N} H_1 \sum_{i=1}^{N} X_i \tag{38}$$

$$= \rho\left(\frac{1}{N}\sum_{i=1}^{N}\phi(X_i)\right). \tag{39}$$

In other words, $\phi : \mathbb{R} \to \mathbb{R}^1$ (i.e., $L = 1$) is the identity function $\phi(X) = \begin{bmatrix} X \end{bmatrix}$ and $\rho : \mathbb{R}^1 \to \mathbb{R}$ such that $\rho(y) = H_0 + H_1 y$ for some undetermined $H_0$ and $H_1$. In this case, regardless of the number of firms $N$, the problem is two dimensional in the space of parameters. See Appendix D.4 for the functional representation of $v(x, X)$.

We move now to provide a detailed treatment of the representation of the optimal policy function when the only a priori knowledge available is the permutation invariance of the optimal policy function, but while certainty equivalence is not assumed or does not hold.

## 5 A Deep Learning Implementation

Our goal is to search for a policy function, $u(X)$, that satisfies Proposition 4 and works much faster than the standard LQ solution. We are interested in a global solution that is not just accurate in a ball around some particular $X^{\text{ss}} \in \mathcal{X}$ (usually, but not necessarily, the steady state of the model), but beyond it as well. However, we will not require that the solution needs to be accurate for any $X \in \mathcal{X}$. In particular, we want the $u(\cdot)$ to generalize well on paths $\left\{X_t^1, X_t^2, \ldots X_t^J\right\}_{t=0}^{T}$ that are likely were we to start from a particular (or finite number) of distributional initial conditions $\left\{X_0^1, X_0^2, \ldots X_0^J\right\}$. An extreme version of this idea would be to focus our effort on getting an accurate solution along the transition path after an unexpected shock. Fortunately, we will see that the solutions will do much better in many cases and will be accurate in a much larger set of points.

## 5.1 Solution method

Our solution algorithm assumes a function form $u(X;\theta)$. A common choice might be a polynomial with a set of weights $\theta$. Then, we pick points in the $\mathcal{X}$ space and minimize a loss function (e.g., mean squared error of the Euler residuals, as in equation (20)) with respect to $\theta$. The points can be some collocation nodes, simulated points à la Rust, or many others.

Thus, the algorithm starts by following the spirit of the classical collocation-style approaches. We diverge in using relatively few points in $\mathcal{X}$ and in ignoring stationarity in both the simulation and the solution. In particular, we use the concentration of measure described in Appendix B to calculate the integral over the $W$ shocks in (18). To do so, though, we must condition on the aggregate $\omega$ shock to ensure that Definition 4 holds.

## 5.2 Symmetric approximations

We will consider any parametric specification for the $u(\cdot)$ function that can be implemented in a machine learning framework as a neural network. It can be as simple as a 2-parameter specification to find the $H_0$ and $H_1$ that implement the affine $\rho$ and $\phi$ in equation (38), or it can be flexible representations with many parameters. Symmetry within this model is encoded in the structure of the approximating function. For example, the LQ solution in equation (38) has a built-in symmetry on the $X$ vector by taking its mean. We will use the representation theorem in Proposition 2 to implement approximations where the functional form is not known and choose $L$, and neural networks for $\rho$ and $\phi$ separately.

Symmetry will allow us to tackle a potential problem of deep learning: its reliance on heavily over-parameterized models, indexed by a number of parameters frequently several orders of magnitude greater than the number of available data points. Recall that deep learning approximates a function $f(\cdot;\theta)$ by composing a set of activation functions. For example, we can choose the activation functions $\{f_1(\cdot;\theta_1), f_2(\cdot;\theta_2), f_3(\cdot;\theta_3)\}$ and then approximate $f(\cdot;\theta) \equiv f_1(f_2(f_3(\cdot);\theta_3);\theta_2);\theta_1)$. Suppose we can impose some structure on the approximation. In that case, we can get a nesting of functions that is considerably more efficient than blind nesting.[10]

Symmetry is one clear example of structure we can encode in the approximation by using the representation theorem in Proposition 2 to design our neural network. For instance, we can use rectified linear units (ReLU), a common nonlinear specification in deep learning, to approximate the functions $\phi$ and $\rho$. In particular, our (one-layer) approximating function will be $f(x;\theta_1,\theta_2) = \theta_1 \max(\theta_2 x, 0)$, where the max is the point-wise maximum, $\theta_2 \in \mathbb{R}^{N\times M}$ and

[10]For instance, Krizhevsky et al. (2012) use the fact that images are invariant under translational and reflection transformations to increase the accuracy of image recognition algorithms (e.g., a mirror image of a cat must be a cat). Clark and Storkey (2015) substantially enhance the performance of learning Go by encoding symmetry under rotation in their algorithm since Go is a board game where the board is invariant under rotation. Hartford et al. (2016) use permutation-equivariant symmetry to reduce dimensionality in the approximation of best responses in a normal form game.

$\theta_1 \in \mathbb{R}^{M\times 1}$ are matrices of unknown coefficients found in the fitting procedure, and $M \geq N$ is the width of the model. To make the network "deeper," we could have multiple layers of these, e.g., $f(x;\theta_1,\theta_2,\theta_3) = \theta_1 \max(\theta_3 \max(\theta_2 x, 0))$.[11]

A surprising benefit of a high-dimensional approximation is the "double-descent" phenomenon in machine learning (see Belkin et al., 2019, and Advani et al., 2020). In classical statistics, the bias-variance trade-off tells us that the generalization error of a function approximation becomes large when the number of free parameters approaches the number of data points (at the limit, when the number of free parameters is equal to the number of data points, we have a simple interpolation). For that reason, one typically wants to choose a smaller number of free parameters than the number of data points. The double-descent phenomenon, which is shown with many machine learning algorithms, is that if one *further* increases the number of parameters far above this interpolation threshold, then the generalization error begins to decrease again. That is, often the cure to over-fitting is to add more parameters. We explore this topic in Section 5.5, where we consider the lack of treatment of transversality conditions and stationarity in more detail.

## 5.3 Solving the LQ case with neural networks

Now, we solve our investment model in the LQ case using a neural network that takes advantage of symmetry. Then, we will compare that solution against the closed-form LQ solution solved using classic control in Section 4. Finally, we will use the same neural network for the case that is not LQ.

First, we will consider an approximation with $\phi$ as a finite set of moments in the same spirit as Krusell and Smith (1998) or the deep-learning model of Fernández-Villaverde et al. (2019). Second, we consider approximating $\phi$ by a flexible ReLU network. In both cases, $\rho$ is left as a highly parameterized neural network. This model is fit using simulations from a single initial condition and only 16 trajectories from $t = 0$ to 63 (including both aggregate and idiosyncratic shocks). Interestingly, it is not important that the transition is very close to any stationary solution (i.e., this does not require a hidden transversality condition or some form of backward induction from a steady state) or that a small or large number of trajectories are used. In fact, in Section 5.5, we solve a version with only a few of those data points.

**Baseline specifications** We call the approximation of $\phi$ using the first moment $\phi$(Identity), using the higher moments (four in our case), $\phi$(Moments), and the more flexible ReLU-based neural networks (with two layers each with 128 nodes) $\phi$(ReLU). Across all of their layers, the baseline $\phi$(Identity), $\phi$(Moments), and $\phi$(ReLU) have 49.4K, 49.8K, and 66.8K parameters respectively. In these three approximations, $\rho$ is represented by a neural network with four layers,

[11]Experience suggests it is better to be deep than wide for the same number of parameters, though the computer science theory explaining this result is in its infancy.

each with 128 nodes.

Our first check is on the accuracy of the solution. Figure 2 shows the mean squared error of the Euler residuals at each time step on the equilibrium path (averaged over 256 different simulated trajectories) and the corresponding 95th percentile bandwidth of those errors, and linear demand (i.e., $\nu = \zeta = 1$). The left panel shows the results for $\phi$(Identity), the center panel shows the results for $\phi$(Moments), and the right panel shows the result for $\phi$(ReLU). Both models are highly accurate. However, the model with ReLU is especially stable. This result can be interpreted as the function generalizing very consistently because of —rather than in spite of— the extra parameters.

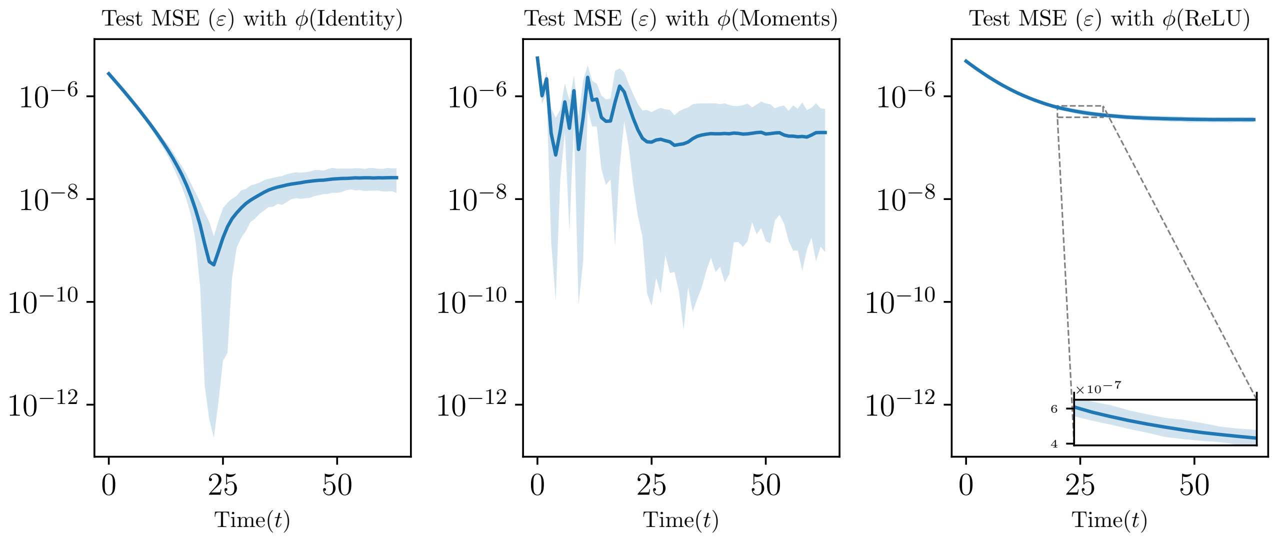

Figure 2: The Euler residuals for $\nu = \zeta = 1$ and $N = 128$ for $\phi$(Identity), $\phi$(Moments), and $\phi$(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles.

To interpret the scale of this Euler error, Figure 3 compares a simulated economy with aggregate shocks using the same approximations as in Figure 2 and the analytical solution provided by the algorithm described in Section 4.1. The red dashed trajectory provides the analytical solution, the orange trajectory depicts the solution obtained by $\phi$(Moments), the blue trajectory depicts the solution obtained by $\phi$(ReLU), and the green trajectory depicts the solution obtained by $\phi$(Identity). All four policies are almost identical.

While these figures show that the solution is very accurate, we need to gauge whether the performance is practical, and—in particular—whether the "curse of dimensionality" has been sidestepped. Figure 4 shows the performance for $\phi$(ReLU) as a function of $N$. The left panel shows the computation time in seconds.[12] The right panel shows the MSE of Euler residuals

[12]The time is calculated from the beginning of the fitting process until the mean squared Euler error reaches $1 \times 10^{-6}$, which is chosen to be in the same order of magnitude as the error coming from the concentration of measure approximation of the integral in Figure 1 when $N$ =128.

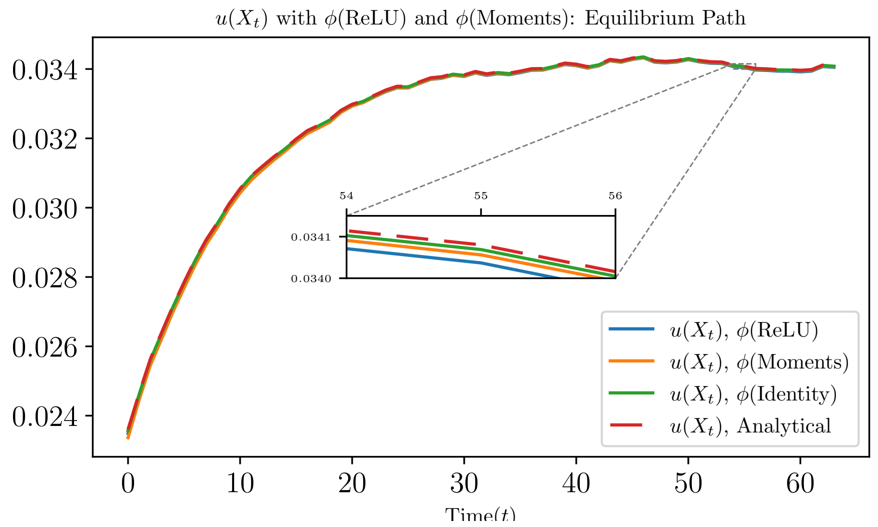

Figure 3: Comparison between baseline approximate solutions and the LQ-regulator solution (Section 4.1) for the case with $\nu = \zeta = 1$ and $N = 128$.

(i.e., $\varepsilon$) on the test data. The reported numbers are the median values of 21 trials with different seeds. The algorithm is finding the global solution of the model with high accuracy in just around a minute. Furthermore, this computation is not a systematic function of $N$. The variation in solution time is largely due to the use of random initial conditions in a high dimensional space (the 66.8K parameters), which is—counterintuitively—essential for convergence when fitting deep learning models.

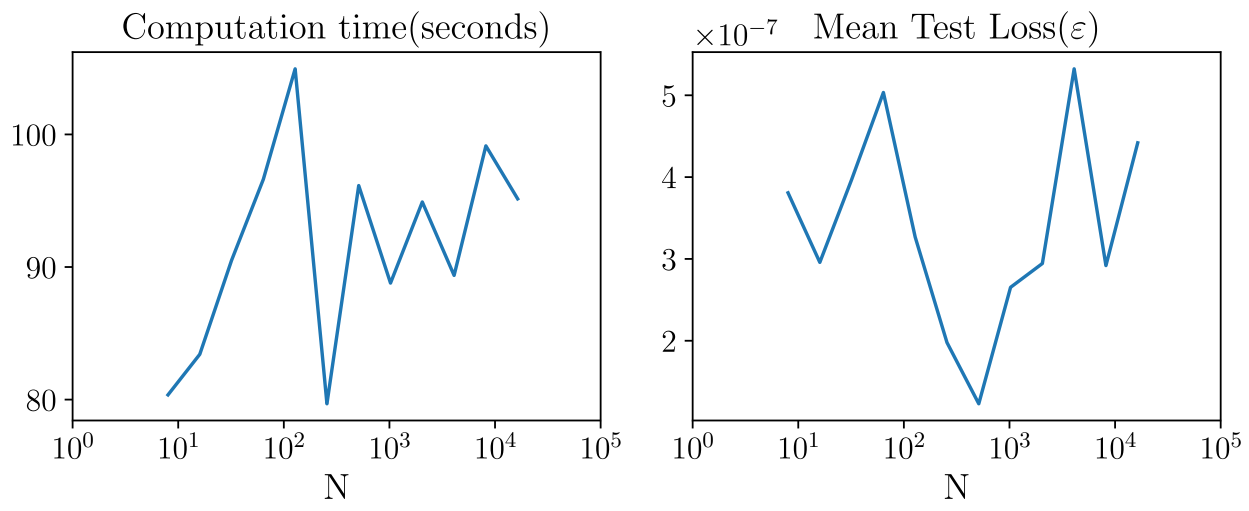

Figure 4: Performance of the $\phi$(ReLU) for different $N$.

Finally, Table 1 shows the results for different variations in the approximations, $\phi$(Identity), $\phi$(ReLU), and $\phi$(Moments). The columns are as follows: Time is measured in seconds. Params represents the number of parameters ($\theta$) in functional approximation, in thousands. Train MSE reports the mean squared error of the Euler residuals for the simulated trajectories used for training. Test MSE documents the mean squared error of the Euler residuals on simulated data after the training is complete, and which internally uses the approximation itself during the simulation. Val MSE displays the mean squared error on the validation data, that is, trajectories used to check the quality of the generalization during the training process, but where the data are not used to fit the model directly. Policy error ($|u - u_{\text{ref}}|$) is the mean absolute value of the approximate solution and the accurate solution ($u_{\text{ref}}$) on the test data. Policy error $\left(\frac{|u-u_{\text{ref}}|}{u_{\text{ref}}}\right)$ is the mean absolute value of the relative error of the approximate solution and the accurate solution ($u_{\text{ref}}$) on the test data. The results in Table 1 are the median among 21 trials with different seeds. See Table 2 in Appendix E.1 for additional experiments.

Table 1: Performance of Different Networks in Solving the Linear Model

| group | description | Time (s) | Params (K) | Train MSE ($\varepsilon$) | Test MSE ($\varepsilon$) | Val MSE ($\varepsilon$) | Policy Error ($\lvert u - u_{\text{ref}}\rvert$) | Policy Error $\left(\frac{\lvert u-u_{\text{ref}}\rvert}{u_{\text{ref}}}\right)$ |
|---|---|---|---|---|---|---|---|---|
| $\phi$(Identity) | Baseline | 42 | 49.4 | 4.1e-06 | 3.3e-07 | 3.3e-07 | 2.9e-05 | 0.10% |
| | Thin (64 nodes) | 33 | 12.4 | 3.7e-06 | 2.7e-07 | 2.7e-07 | 3.4e-05 | 0.10% |
| $\phi$(ReLU) | Baseline | 107 | 66.8 | 3.7e-06 | 3.3e-07 | 3.3e-07 | 2.7e-05 | 0.09% |
| | L = 2 | 86 | 66.3 | 1.3e-05 | 2.1e-07 | 2.2e-07 | 2.6e-05 | 0.08% |
| | L = 16 | 91 | 69.9 | 5.5e-06 | 1.5e-07 | 1.5e-07 | 2.1e-05 | 0.07% |
| | Shallow($\phi$ : 1 layer, $\rho$ : 2 layers) | 79 | 17.7 | 2.0e-06 | 5.5e-07 | 5.5e-07 | 3.2e-05 | 0.11% |
| | Deep($\phi$ : 4 layers, $\rho$ : 8 layers) | 242 | 165.1 | 2.1e-03 | 2.2e-03 | 2.1e-03 | 2.7e-03 | 8.50% |
| | Thin($\phi, \rho$ : 64 nodes) | 87 | 17.0 | 1.1e-05 | 4.5e-07 | 4.5e-07 | 3.0e-05 | 0.10% |
| $\phi$(Moments) | Baseline | 55 | 49.8 | 1.4e-06 | 7.6e-07 | 7.6e-07 | 2.8e-05 | 0.09% |
| | Moments (1,2) | 211 | 49.5 | 2.4e-06 | 1.1e-06 | 2.3e-06 | 4.4e-05 | 0.14% |
| | Very Shallow(1 layer) | 241 | 0.6 | 1.1e-05 | 8.4e-06 | 7.9e-06 | 1.1e-02 | 34.00% |
| | Thin (64 nodes) | 82 | 12.6 | 1.6e-06 | 9.1e-07 | 9.2e-07 | 3.8e-05 | 0.12% |

The first lesson from Table 1 is that the results are mainly robust to variations in the networks' architecture. However, for small networks, as can be seen in the case of the *Very Shallow* example in $\phi$(Moments), sometimes the optimization procedure finds a local minimum that is not close to the global minimum solution. Not coincidentally, this is the model with the fewest number of parameters—which should give us pause since it is still orders of magnitude above the dimensionality of 2 parameters required for the analytical solution.

The second lesson from Table 1 is that the results are, in general, good when we use a larger number of moments (equivalently, a higher dimension of $L$) than when we use only the first moment ($L = 1$, which we know from the analytical solution is a sufficient statistic for payoff-relevant values). Compare, for example, the $\phi$(Identity) case with $\phi(\text{ReLU}(L = 16))$. The latter solves the model with many more parameters, but the neural network finds the lower-dimensional manifold in about the same amount of time.

## 5.4 Nonlinear demand function

In this section, we move to the case where $\nu > 1$, i.e., the (inverse) demand function is nonlinear. In this situation, the LQ solution and certainty equivalence no longer hold and we do not have an analytical solution. We find the solution to this nonlinear case using exactly the same algorithm, code, and functional approximations as in Section 5.3. As before, we fit the model and then check its generalization with a set of simulated trajectories.

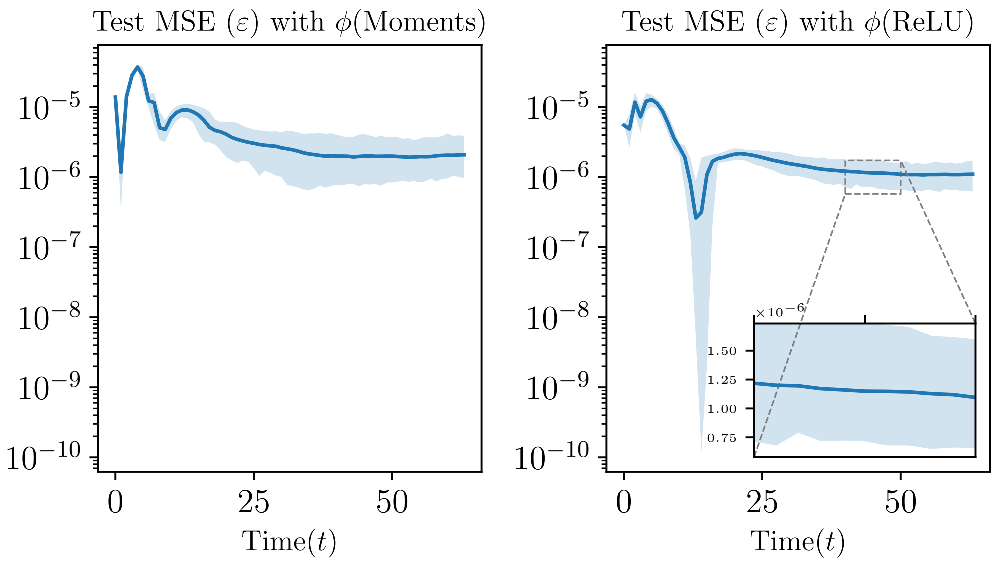

Figure 5: The Euler residuals for $\nu = 1.5$, $\zeta = 1.0$, and $N = 128$ for $\phi$(Identity), $\phi$(Moments), and $\phi$(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles.

Figure 5 shows the average Euler residuals over 256 simulated trajectories, with the corresponding 95th percentile bandwidth. The economy uses $\zeta = 1$, $\nu = 1.5$, and $N = 128$. The left panel shows the results for baseline $\phi$(Moments). The right panel shows the results for baseline $\phi$(ReLU). As in Figure 2, the mean squared error is fairly low and tight. Table 3 in Appendix E.2 provides additional information on this experiment and different variations on it. The main conclusion is that the performance is roughly the same as in the linear case because it is using the same algorithm.

We can also investigate the qualitative behavior of the approximate optimal solution, $u(\cdot)$, along the transition path as $\nu$ approaches one, which coincides with the linear case where an accurate solution exists. Figure 6 depicts the optimal policy along the equilibrium paths for $\nu = [1.0, 1.05, 1.1, 1.5]$ and $\zeta = 1$, for an economy with $N = 128$. In this experiment, the baseline $\phi$(ReLU) is utilized to approximate the optimal policies. Figure 6 shows how the optimal policy for a trajectory smoothly approaches the known analytical solution as $\nu$ approaches one.

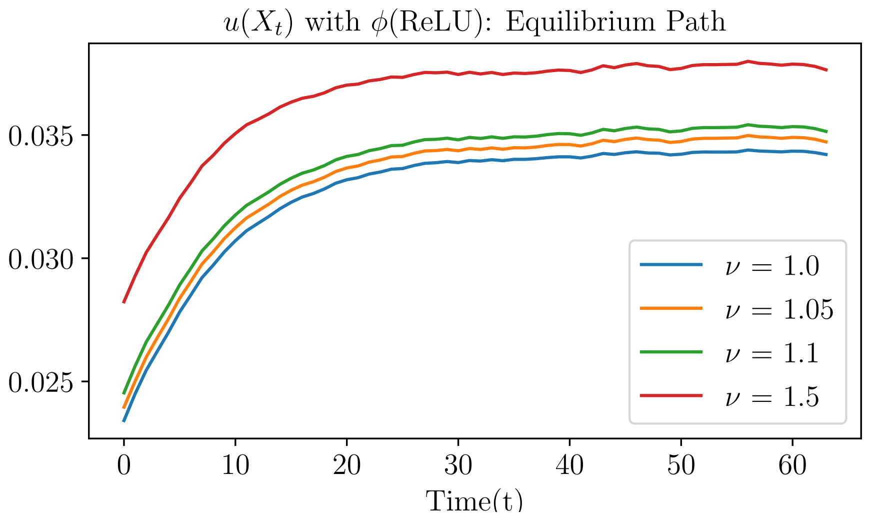

Figure 6: The optimal policy $u$ along the equilibrium paths for prices, $\nu = [1.0, 1.05, 1.1, 1.5]$, $\zeta = 1$, and $N = 128$. Each path shows the optimal policy for a single trajectory.

## 5.5 Generalization: Where is the transversality condition?

Intriguingly, to get our approximation, we neither applied a transversality condition nor solved a stationary problem. While we fitted the data to a $T$ where the equilibrium was getting close to some ergodic distribution, we did not use that behavior as a special case. However, in general, we can have many candidate solutions that satisfy the first-order conditions of a dynamic model if we do not constrain the long-run behavior of these solutions in one form or another. In fact, that is what we did to get the LQ solution: the classic control solutions implicitly applied a transversality condition.[13] And, yet, in Section 5.3, we saw that our neural networks achieved the same solution as the LQ one, i.e., they consistently picked the solution that generalizes best.[14] Why would we so carelessly disregard what has been thought of as an essential boundary condition for solving models?

This question can be recast in terms of generalization and regularization. First: Why would our algorithm choose the "right" solution to the functional equation in the absence of explicit handling of the long-run behavior? Second, if the algorithm chooses a particular solution (rather than a random one), is it the one we want (as measured by its ability to generalize)?

In terms of answering the first question, one might suspect that this is simply because we are simulating with a great deal of data, but this turns out not to be the case. To show this, we can conduct an even more extreme example of over-parameterization by fitting a nonlinear network with a parameter vector in $\mathbb{R}^{17700}$ using only 3 random data points in $\mathbb{R}^{512}$ for the linear

[13]Or, alternatively, we could think of this as being a guess-and-verify approach where we assumed a functional form consistent with a transversality condition.

[14]Of course, with shorter simulations, the generalization error for very large $t > T$ becomes worse. But that is a consequence of there being insufficient numbers of data points in a domain of interest, not a consequence of whether the system is ergodic or not.

demand function. In this case, the solution consistent with a transversality condition or imposing a stationary solution boundary condition only needs 2 points to interpolate.

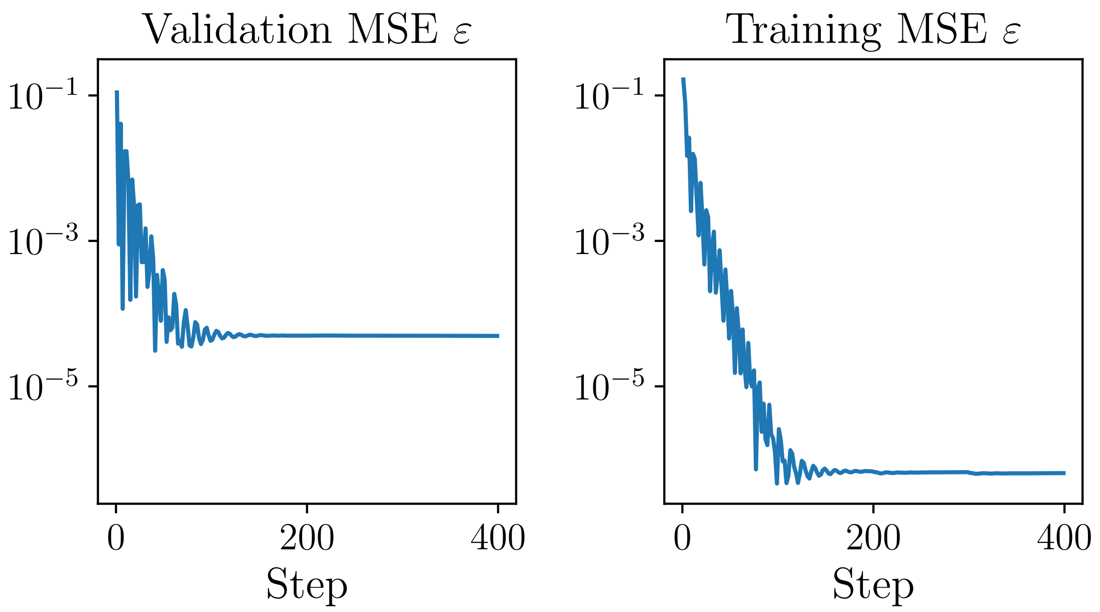

Figure 7: Training and validation loss of the Euler residuals $\varepsilon$. The training procedure utilizes 3 data points in the state space for an economy with $\nu = \zeta = 1$ and $N = 512$.

The results are shown in Figure 7. In this experiment, we fit a model with both $\rho$ and $\phi$ represented by a neural network with two and one hidden layers, each with 128 nodes and $\nu = \zeta = 1$. The output dimension of $\phi$ is $L = 4$. Each step is one gradient updating in stochastic gradient descent optimization. We draw 3 random data points within $\mathbb{R}^{512}$ from simulated trajectories. The model fits using random "batches" of data of size 2, where the randomization in the optimization algorithm (e.g., stochastic gradient descent) is an essential part of regularizing the solutions. The right panel of Figure 7 shows the convergence of the MSE of the training data, which the optimizer is minimizing. The left panel of Figure 7 shows the MSE of the Euler errors on "validation" data, i.e., random data used to test the generalization error during training, but which does not enter the optimization objective directly. The training error drops to within the numerical tolerance, as expected. But rather than over-fitting, the validation error also drops to less than $5 \times 10^{-5}$, which corresponds to a relative policy error of 0.06%, despite a large number of parameters and the minimal data.

While the theoretical understanding of this over-fitting phenomenon is still an active area of research within computer science, it is robustly found in many functional approximation problems. In application after application, using stochastic optimization methods with highly over-parameterized models tends to robustly find solutions that fulfill a minimum norm solution in the space of approximating functions. Recent work in understanding convergence and generalization of over-parameterized models includes Arora et al. (2018), Allen-Zhu et al. (2019), and Nakkiran et al. (2019).

For our specific problem, the minimum norm solution seems to coincide with the most generalizable solution—and in particular, it seems to automatically fulfill the boundary-value conditions (e.g., transversality or imposing a boundary at a terminal period) we often need to apply directly when approximating with lower-dimensional models.

## 6 Extensions and Generalizations

Did we choose a setup that was particularly amenable to our methods? In this section, we consider how these general techniques can be used for a much larger class of models. Most broadly, our tools are useful for solving any high-dimensional functional equations with some degree of symmetry—and are especially useful when they contain high-dimensional expectations. The functional equation could come from static problems, Bellman equations, or Euler equations, and can be solved by various approximation techniques.[15]

### 6.1 Decreasing returns to scale

The constant returns to scale in production and the linear marginal cost of investment in equation (15) led to the value function in Proposition 4 being independent from $x$. As the primary interest was in analyzing the high-dimensional state, this was a convenient result.

But what if the value function (10) has decreasing returns to scale and, thus, the policy becomes a function of $x$? Imagine, for example:

$$\text{profits} = p(X)\frac{x^{1-\mu}}{1-\mu} - \frac{\gamma}{2}u^2.$$

This formulation leads to an Euler equation:

$$\gamma u(x, X) = \beta \mathbb{E}\left[P(X')x^{-\mu} + \gamma(1-\delta)u(x', X')\right].$$

Given this equation, the solution techniques are identical except that $\mathbb{E}[\cdot]$ now depends strongly on the idiosyncratic shock associated with state $x$. Extending the previous approach where we condition on random variables that strongly influence the gradient, as defined by Definition 4, the expectation can be approximated by a two-dimensional Gaussian quadrature in $w$ and $\Omega$, where the $W$ for the aggregate evolution is still performed with a Monte Carlo draw.

[15]For example, we could also use reinforcement learning (also known as approximate dynamic programming or neuro-dynamic programming). See Sutton and Barto (2018) and Bertsekas (2019) for a survey of methods, and van der Pol et al. (2020) for an example applying symmetry to reinforcement learning.

The only other modification required is to ensure we are using a representation in Proposition 2 where $\rho(x, y)$ and $\phi(x)$ where $y \in \mathbb{R}^L$.

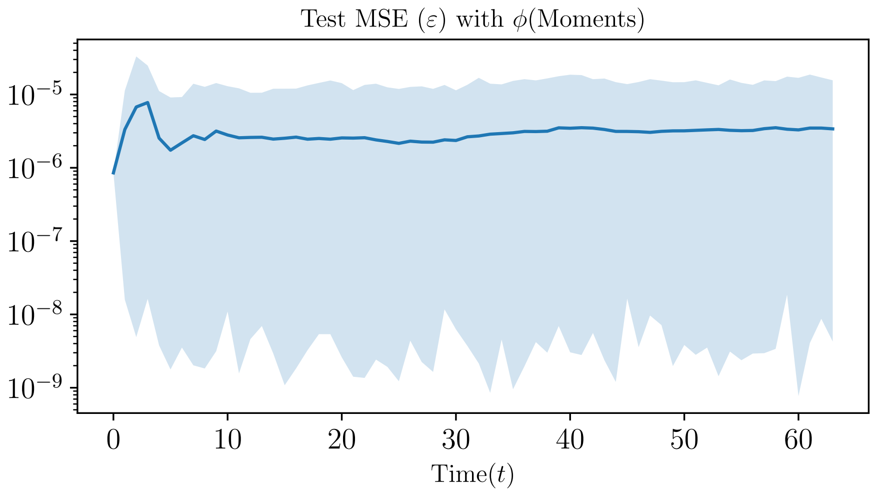

Figure 8: The Euler residuals for $\nu = \zeta = 1$, $\mu = 0.05$, and $N = 128$, over 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles. In this case, the baseline $\phi$(Moments) is used to approximate the optimal policy.

Figure 8 depicts the average Euler residuals along the equilibrium path (averaged over 256 trajectories) and the corresponding 95th percentile bandwidth for an economy with $\nu = \zeta = 1$, $\mu = 0.05$, and $N = 128$. In this experiment baseline $\phi$(Moments) is used to approximate the optimal policy. The generalization properties are excellent, and the mean squared Euler error is on the order of $1 \times 10^{-6}$ for most of the trajectories and periods.

## 6.2 Multiple productivity types

We consider now the case when firms have $\text{profits}_j = \zeta_j p(X)\frac{x^{1-\mu}}{1-\mu} - \frac{\gamma}{2}u^2$ for some finite set of productivities, e.g., $\zeta_1, \zeta_2$. Then, rather than having permutation invariance based on the entire $X$, we could group them such that the policy becomes $u(x, X^1, X^2)$, where $X_1$ contains all firms with $\zeta_1$, etc. That is, the elements of $X$ could be symmetric in blocks, and this could be implemented in the network with a representation such as

$$u(x, X^1, X^2) \approx \rho\left(x, \frac{1}{N_1}\sum_{i=1}^{N_1}\phi_1(X_i^1), \frac{1}{N_2}\sum_{i=1}^{N_2}\phi_2(X_i^2)\right).$$

If there were reasons in the problem structure to think that the $\phi_1$ and $\phi_2$ could be proportional to each other, then the representation might even be

$$u(x, X^1, X^2) \approx \rho\left(x, \frac{1}{N_1}\sum_{i=1}^{N_1}\phi(X_i^1), \frac{1}{N_2}\sum_{i=1}^{N_2}\phi(X_i^2)\right).$$

## 6.3 Complex idiosyncratic states

Next, we study the case where firms also have a stochastic productivity $y$, such that the firm is identified by the tuple $z \equiv (x, y)$ and the aggregate state is a collection of tuples $Z \equiv \{(X_i, Y_i)\}_{i=1}^N$. We can apply our techniques by considering that it is the symmetry with respect to the $z$ tuple of states, rather than permutation within that tuple, that is required. We can then implement this symmetry in an approximation along the lines of

$$u(x, y, Z) \approx \rho\left(x, y, \frac{1}{N}\sum_{i=1}^{N}\phi(X_i, Y_i)\right),$$

for some $\phi : \mathbb{R}^2 \to \mathbb{R}^L$ and $(x, y)$ the idiosyncratic state of interest.[16] Hence our techniques are easily applyed to models with exchangeable "agents" containing a finite number of states.

Our approach can even be extended to cases where the collection of underlying states is itself exchangeable. For example, consider if, in addition to its $x$, each firm has up to $M$ customers of varying sizes, $y^j$, where those customers are otherwise exchangeable in the firm's payoffs.[17] Then we can denote the customers themselves as a set, and use $z \equiv (x, \{y^j\}_{j=1}^M)$ as an individual firm's state. If we let $Y_i \equiv \{y_i^j\}_{j=1}^M$, the aggregate distribution is $Z \equiv \{X_i, Y_i\}$.

A neural network imposing this symmetry might be

$$u(x, Y, Z) \approx \rho_1\left(x, \frac{1}{M}\sum_{j=1}^{M}\phi_1(y^j), \frac{1}{N}\sum_{i=1}^{N}\rho_2\left(X_i, \frac{1}{M}\sum_{j=1}^{M}\phi_2(Y_i^j)\right)\right),$$

where the training process would fit parameters for $\rho_1, \phi_1, \rho_2$, and $\phi_2$, and the $\rho_2$ function itself could have a latent dimension greater than one. This last example demonstrates why deep learning, in the sense of many layers of highly-parameterized functions, is so important for dealing with symmetry. Without modern (and easily available) machine learning software, it would be impractical to fit this nested approximation.

Given the flexibility of working with states that are themselves complicated structures, a natural application of these methods is to solve dynamic models with networks—expanding the computational feasibility of solving extensions of models surveyed in Carvalho and Tahbaz-Salehi

[16]In a model with a continuum of agents, this would simply lead to a two-dimensional distributional state, and the modeler would typically write payoffs as expectations using that distribution.

[17]See Hartford et al. (2018) for methods to represent permutation equivariance with interactions between sets.

(2019). In particular, it holds the promise of making multi-period dynamic programming in models such as Oberfield (2018) possible by enabling network connections as a persistent state variable and enabling firms to form expectations of the network's evolution.

### 6.4 Global solutions with transitions and aggregate shocks

In many applications, researchers use a continuum of agents. Unfortunately, as soon as we add several aggregate shocks to them, these models are difficult to tackle. Since the distributional state becomes a stochastic partial differential equation (or its discrete-time equivalent), agents need to solve optimal control problems with an infinite-dimensional state, which in turn depends on their decisions. The forward equation for the evolution is difficult enough to simulate, even as an initial value problem, but becomes intractable when it is part of an infinite-dimensional optimal control problem. The continuum approximation is then the source of the numerical intractability rather than the solution to it, hinting that leaving things discrete might help.

In that sense, our methods offer a path forward by offering the possibility of simulating economies with a large number of finite agents (on the order of thousands or tens of thousands), without even having to worry about stationarity properties. In comparison, in perturbation solutions such as Kaplan et al. (2018) in continuous time and Den Haan (2010) in discrete time one would require appropriate stationarity properties.

This insight might be particularly fruitful when solving growth models with heterogeneous agents, which often have steady states and balanced growth paths that are notoriously sensitive to parameters, multiplicity of equilibria, and hysteresis (i.e., they may not even be ergodic). By using a finite number of agents—along the models discussed in Buera and Lucas (2018)—models with endogeneity of both innovation and diffusion as in Benhabib et al. (2021) without ergodicity could be solved from arbitrary initial conditions.

## 7 Conclusion

In this paper, we have shown how to exploit symmetry in dynamic programming problems and the solution of recursive competitive equilibria with a large (but finite) number of heterogeneous agents using deep learning. Symmetry, together with concentration of measure, allowed us to break the "curse of dimensionality." The method is easy to implement using standard, open-source software libraries and fast to run on a standard laptop. Our results open the door to many applications of interest, including macro, finance, international finance, industrial organization, and other fields.

## References

ADVANI, M. S., A. M. SAXE, AND H. SOMPOLINSKY (2020): "High-dimensional dynamics of generalization error in neural networks," *Neural Networks*, 132, 428–446.

ALLEN-ZHU, Z., Y. LI, AND Z. SONG (2019): "A convergence theory for deep learning via over-parameterization," in *International Conference on Machine Learning*, PMLR, 242–252.

ANDERSON, E. W., E. R. MCGRATTAN, L. P. HANSEN, AND T. J. SARGENT (1996): "Mechanics of forming and estimating dynamic linear economies," *Handbook of Computational Economics*, 1, 171–252.

ARORA, S., R. GE, B. NEYSHABUR, AND Y. ZHANG (2018): "Stronger generalization bounds for deep nets via a compression approach," in *Proceedings of the 35th International Conference on Machine Learning*, ed. by J. Dy and A. Krause, vol. 80, 254–263.

AZINOVIC, M., L. GAEGAUF, AND S. SCHEIDEGGER (2019): "Deep equilibrium nets," *Available at SSRN 3393482*.

BARVINOK, A. (2005): "Math 710: Measure concentration," *Lecture notes.*

BELKIN, M., D. HSU, S. MA, AND S. MANDAL (2019): "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, 116, 15849–15854.

BELLMAN, R. (1958): *Dynamic Programming*, Princeton University Press.

BENHABIB, J., J. PERLA, AND C. TONETTI (2021): "Reconciling models of diffusion and innovation: A theory of the productivity distribution and technology frontier," *Econometrica (Forthcoming).*

BERTSEKAS, D. P. (2019): *Reinforcement Learning and Optimal Control*, Athena Scientific.

BLUMAN, G. W., A. F. CHEVIAKOV, AND S. C. ANCO (2010): *Applications of Symmetry Methods to Partial Differential Equations*, Springer.

BOUCHERON, S., G. LUGOSI, AND P. MASSART (2013): *Concentration Inequalities: A Nonasymptotic Theory of Independence*, Oxford University Press.

BUERA, F. J. AND R. E. LUCAS (2018): "Idea flows and economic growth," *Annual Review of Economics*, 10, 315–345.

CARVALHO, V. M. AND A. TAHBAZ-SALEHI (2019): "Production networks: A primer," *Annual Review of Economics*, 11, 635–663.

Chen, S., E. Dobriban, and J. H. Lee (2019): "Invariance reduces variance: Understanding data augmentation in deep learning and beyond," *arXiv preprint arXiv:1907.10905*.

Clark, C. and A. Storkey (2015): "Training deep convolutional neural networks to play Go," in *International Conference on Machine Learning*, 1766–1774.

Cybenko, G. (1989): "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, 2, 303–314.

Den Haan, W. J. (2010): "Comparison of solutions to the incomplete markets model with aggregate uncertainty," *Journal of Economic Dynamics and Control*, 34, 4–27.

Duarte, V. (2018): "Machine learning for continuous-time finance," Tech. rep., Gies College of Business.

Duffy, J. and P. D. McNelis (2001): "Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm," *Journal of Economic Dynamics and Control*, 25, 1273–1303.

Fernández-Villaverde, J. and P. A. Guerrón-Quintana (2020): "Estimating DSGE Models: Recent Advances and Future Challenges," Working Paper 27715, National Bureau of Economic Research.

Fernández-Villaverde, J., S. Hurtado, and G. Nuño (2019): "Financial frictions and the wealth distribution," Working Paper 26302, National Bureau of Economic Research.

Hartford, J., D. R. Graham, K. Leyton-Brown, and S. Ravanbakhsh (2018): "Deep models of interactions across sets," *arXiv preprint arXiv:1803.02879*.

Hartford, J. S., J. R. Wright, and K. Leyton-Brown (2016): "Deep learning for predicting human strategic behavior," in *Advances in Neural Information Processing Systems*, 2424–2432.

Hornik, K. (1991): "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, 4, 251–257.

Judd, K. L., L. Maliar, S. Maliar, and I. Tsener (2017): "How to solve dynamic stochastic models computing expectations just once," *Quantitative Economics*, 8, 851–893.

Kaplan, G., B. Moll, and G. L. Violante (2018): "Monetary policy according to HANK," *American Economic Review*, 108, 697–743.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012): "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 1097–1105.

Krusell, P. and A. A. Smith, Jr (1998): "Income and wealth heterogeneity in the macroeconomy," *Journal of Political Economy*, 106, 867–896.

Ledoux, M. (2001): *The Concentration of Measure Phenomenon*, 89, American Mathematical Society.

Ljungqvist, L. and T. J. Sargent (2018): *Recursive Macroeconomic Theory*, MIT Press, 4 ed.

Lucas, R. E. and E. C. Prescott (1971): "Investment under uncertainty," *Econometrica*, 659–681.

Lyle, C., M. van der Wilk, M. Kwiatkowska, Y. Gal, and B. Bloem-Reddy (2020): "On the Benefits of Invariance in Neural Networks," *arXiv preprint arXiv:2005.00178*.

Macdonald, I. G. (1998): *Symmetric Functions and Hall Polynomials*, Oxford University Press.

Maliar, L., S. Maliar, and P. Winant (2019): "Will artificial intelligence replace computational economists any time soon?" Tech. Rep. 14024, C.E.P.R. Discussion Papers.

Nakkiran, P., G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever (2019): "Deep double descent: Where bigger models and more data hurt," *arXiv preprint arXiv:1912.02292*.

Narayanamurthy, S. M. and B. Ravindran (2008): "On the hardness of finding symmetries in Markov decision processes," in *Proceedings of the 25th International Conference on Machine Learning*, 688–695.

Oberfield, E. (2018): "A theory of input-output architecture," *Econometrica*, 86, 559–589.

Powell, W. B. (2007): *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, vol. 703, John Wiley & Sons.

Prasad, A. (2018): "An introduction to Schur polynomials," *arXiv preprint arXiv:1802.06073*.

Prescott, E. C. and R. Mehra (1980): "Recursive competitive equilibrium: The case of homogeneous households," *Econometrica*, 48, 1365–1379.

Ravindran, B. and A. G. Barto (2001): *Symmetries and Model Minimization in Markov Decision Processes*, University of Massachusetts.

Samuelson, P. A. (1990): "Law of conservation of the capital-output ratio in closed von Neumann systems," in *Conservation Laws and Symmetry: Applications to Economics and Finance*, ed. by R. Sato and R. V. Ramachandran, Springer Netherlands, 53–56.

Sato, R. (1985): "The invariance principle and income-wealth conservation laws: Application of Lie groups and related transformations," *Journal of Econometrics*, 30, 365–389.

Sutton, R. S. and A. G. Barto (2018): *Reinforcement Learning: An Introduction*, MIT press.

van der Pol, E., D. E. Worrall, H. van Hoof, F. A. Oliehoek, and M. Welling (2020): "MDP homomorphic networks: Group symmetries in reinforcement learning," *arXiv preprint arXiv:2006.16908.*

Vershynin, R. (2018): *High-Dimensional Probability: An Introduction with Applications in Data Science*, vol. 47, Cambridge university press.

Wagstaff, E., F. B. Fuchs, M. Engelcke, I. Posner, and M. Osborne (2019): "On the limitations of representing functions on sets," *arXiv preprint arXiv:1901.09006.*

Walcher, S. (2019): "Symmetries of ordinary differential equations: A short introduction," *arXiv preprint arXiv:1911.01053.*

Yarotsky, D. (2018): "Universal approximations of invariant maps by neural networks," *arXiv preprint arXiv:1804.10306.*

Zabrocki, M. (2015): "Introduction to symmetric functions," *unpublished work http://garsia. math. yorku. ca/ghana03/mainfile. pdf.*

Zaheer, M., S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola (2017): "Deep sets," in *Advances in Neural Information Processing Systems*, 3391–3401.

## Appendix A Mathematics Definitions and Background

**Definition 6** (The permutation group and the permutation operator). *Define:*

- *A permutation group is a group $G$ whose elements are permutations of a given set $\mathcal{I}$ and whose group operation is the composition of permutations in $G$.*
- *A symmetric group of a set $\mathcal{I}$ is the group of all permutations of set $\mathcal{I}$, denoted by $\mathcal{S}^{\mathcal{I}}$.*

**Definition 7** (Permutation invariance). *A function $f : \mathbb{R}^N \to \mathbb{R}$ is permutation invariant if, for all permutations $\pi \in \mathcal{S}_N$:*

$$f(\pi X) = f(X).$$

**Definition 8** (Permutation equivariance). *A function $f : \mathbb{R}^N \to \mathbb{R}^N$ is permutation equivariant if, for all permutations $\pi \in \mathcal{S}_N$:*

$$f(\pi X) = \pi f(X).$$

**Proposition 6.** *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^N$ and $f : \mathcal{X} \subseteq \mathbb{R}^N \to \mathbb{R}$ be a continuous permutation invariant function under $\mathcal{S}^{\mathbb{N}}$, i.e., for all $\vec{Y} \in \mathbb{R}^N$ and all $\pi \in \mathcal{S}^{\mathbb{N}}$:*

$$f(\pi\vec{Y}) = f(\vec{Y}).$$

*Then, there exist $L \leq N$, and continuous functions $\varrho : \mathbb{R}^L \to \mathbb{R}$ and $\varphi : \mathbb{R} \to \mathbb{R}^L$, such that*

$$f\left(\vec{Y}\right) = \varrho\left(\frac{1}{N}\sum_{i=1}^{N}\varphi(y_i)\right).$$

*Proof.* See Wagstaff et al. (2019) and Zaheer et al. (2017). □

## Appendix B Concentration of Measure

**Proposition 7** (Gaussian Poincaré inequality). *Suppose $f : \mathbb{R}^N \to \mathbb{R}$ is a continuously differentiable function and $z \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$ is a standard Gaussian random vector. Then:*

$$\mathrm{Var}[f(z)] \leq \mathbb{E}\left[\|\nabla f(z)\|^2\right],$$

*Moreover, if $z \sim \mathcal{N}(\mathbf{0}_N, \Sigma)$ is a Gaussian random vector with mean zero and covariance matrix $\Sigma$, then:*

$$\mathrm{Var}[f(z)] \leq \mathbb{E}\left[\nabla f^\top(z)\Sigma\nabla f(z)\right].$$

*Proof.* See Boucheron et al. (2013), Theorem 3.20. □

# Appendix C Proofs

## C.1 Proof of Proposition 1

*Proof.* Let $\pi \in \mathcal{S}_N$ be a permutation and $\vec{Y} \equiv \pi X$. The Bellman equation can be written as:

$$v(x, \vec{Y}) = \max_u \left\{ r(x, u, \vec{Y}) + \beta \mathbb{E}\left[ v(x', \vec{Y}') \right] \right\} \tag{C.1}$$

$$\text{s.t. } x' = g(x, u) + \sigma w + \eta \omega \tag{C.2}$$

$$\vec{Y}' = G(\vec{Y}) + \Omega \pi W + \eta \omega \pi \mathbf{1}_N. \tag{C.3}$$

Assuming $\frac{\partial g(x,u)}{\partial u} \neq 0$, the Euler equation can be written as

$$\frac{r_u\big(x, u(x, \vec{Y}), \vec{Y}\big)}{g_u\big(x, u(x, \vec{Y})\big)} + \beta \mathbb{E}\left[ r_x\big(x', u(x', \vec{Y}'), \vec{Y}'\big) - r_u\big(x', u(x', \vec{Y}'), \vec{Y}'\big) \frac{g_x\big(x', u(x', \vec{Y}')\big)}{g_u\big(x', u(x', \vec{Y}')\big)} \right] = 0,$$

where $r_u \equiv \frac{\partial r}{\partial u}$, $r_x \equiv \frac{\partial r}{\partial x}$, $g_x \equiv \frac{\partial g}{\partial x}$, and $g_u \equiv \frac{\partial g}{\partial u}$ . Since $r$ is permutation invariant with respect to $X$, we have

$$r_u\big(x, u(x, \vec{Y}), \vec{Y}\big) = r_u\big(x, u(x, \vec{Y}), X\big)$$
$$r_x\big(x, u(x, \vec{Y}), \vec{Y}\big) = r_x\big(x, u(x, \vec{Y}), X\big).$$

Using the fact that $G(\vec{Y}) = \pi G(X)$ and $\Omega \pi = \pi \Omega$

$$\vec{Y}' = \pi\big(G(X) + \Omega W + \eta \omega \mathbf{1}_N\big) = \pi X'. \tag{C.4}$$

Defining $u(x, \pi X) = u \circ \pi(x, X) \equiv u_\pi(x, X)$, the Euler equation becomes:

$$\frac{r_u\big(x, u_\pi(x, X), X\big)}{g_u\big(x, u_\pi(x, X)\big)} + \beta \mathbb{E}\left[ r_x\big(x', u_\pi(x', X'), X'\big) - r_u\big(x', u_\pi(x', X'), X'\big) \frac{g_x\big(x', u_\pi(x', X')\big)}{g_u\big(x', u_\pi(x', X')\big)} \right] = 0,$$

where:

$$x' = g(x, u) + \sigma w + \eta \omega \tag{C.5}$$

$$X' = G(X) + \Omega W + \eta \omega \mathbf{1}_N. \tag{C.6}$$

Since the solution of the Euler equation is unique, then:

$$u_\pi(x, X) = u(x, \pi X) = u(x, X).$$

Given the permutation invariant optimal policy, $u$, and reward function, $r$, the Bellman equation can be written as:

$$v(x, \vec{Y}) = r\big(x, u(x, X), X\big) + \beta \mathbb{E}\left[v(x', \vec{Y}')\right],$$

where $x'$ and $\vec{Y}'$ are described by equations (C.2) and (C.3). Using equation (C.4) and defining $v(x, \pi X) = v \circ \pi(x, X) \equiv v_\pi(x, X)$, the Bellman equation can be written as:

$$v_\pi(x, X) = r\big(x, u(x, X), X\big) + \beta \mathbb{E}\left[v_\pi(x', X')\right],$$

where $x'$ and $X'$ are described by equations (C.5) and (C.6). By the contraction mapping theorem, the solution of the Bellman equation is unique. Therefore:

$$v_\pi(x, X) = v(x, \pi X) = v(x, X).$$

□

## C.2 Proof of Proposition 3

*Proof.* By the Chebyshev's inequality:

$$\mathbb{P}\left(\big|f(z) - \mathbb{E}\left[f(z)\right]\big| \geq \epsilon\right) \leq \frac{\mathrm{Var}\big[f(z)\big]}{\epsilon^2}.$$

By the Gaussian Poincaré inequality (Proposition 7):

$$\mathrm{Var}\big[f(z)\big] \leq \mathbb{E}\left[\nabla f(z)^\top \Sigma \nabla f(z)\right].$$

Therefore:

$$\mathbb{P}\left(\big|f(z) - \mathbb{E}\left[f(z)\right]\big| \geq \epsilon\right) \leq \frac{1}{\epsilon^2}\mathbb{E}\left[\nabla f(z)^\top \Sigma \nabla f(z)\right].$$

Since a covariance matrix is symmetric and positive semi-definite, there exists an orthogonal matrix $U$ (i.e. $U^\top = U^{-1}$) and a diagonal matrix $\Lambda$ such that:

$$\Sigma = U \Lambda U^\top,$$

where $\Lambda$ is the diagonal matrix of the eigenvalues of $\Sigma$. Hence:

$$\nabla f(z)^\top \Sigma \nabla f(z) = \sum_{i=1}^{N} \lambda_i \big[U^\top \nabla f(z)\big]_i^2 \leq \max_{i=1,..,N}\{\lambda_i\} \|\nabla f(z)\|^2 = \rho(\Sigma)\|\nabla f(z)\|^2,$$

where $\rho(\Sigma) \equiv \max_{i=1,..,N} \{\lambda_i\}$ is the spectral radius of $\Sigma$. Therefore,

$$\mathbb{P}\left(\left|f(z) - \mathbb{E}\left[f(z)\right]\right| \geq \epsilon\right) \leq \frac{\rho(\Sigma)C}{\epsilon^2}\frac{1}{N}.$$

□

## C.3 Proof of Proposition 4

*Proof.* Since the price function $p(X)$ is permutation invariant and $\hat{u}$ is differentiable, the agent problem can be written as:

$$v(x, X) = \max_u \left\{ p(X)\frac{x^{1-\mu}}{1-\mu} - \frac{\gamma}{2}u^2 + \beta\mathbb{E}\left[v(x', X')\right]\right\} \tag{C.7}$$

$$\text{s.t. } x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{C.8}$$

$$X' = \left\{(1-\delta)\hat{x} + \hat{u}(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W)\right\}. \tag{C.9}$$

The first-order condition is:

$$-\gamma u + \beta\mathbb{E}\left[\frac{\partial v(x', X')}{\partial x'}\right] = 0. \tag{C.10}$$

□

Given that the other agents' law of motion is independent of the agent of interest's choice, the envelope condition is:

$$\frac{\partial v(x', X')}{\partial x'} = P(X')(x')^{-\mu} + \gamma(1-\delta)u', \tag{C.11}$$

where $u'$ is the optimal policy in the next period. Combining the first-order condition (C.10) and the envelope condition (C.11) yields:

$$-\gamma u(x, X) + \beta\mathbb{E}\left[P(X')(x')^{-\mu} + \gamma(1-\delta)u(x', X')\right] = 0. \tag{C.12}$$

In a symmetric equilibrium, all the agents use the same policy function, i.e., $\hat{u}(.,.) = u(.,.)$, and the laws of motion can be rewritten as:

$$x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{C.13}$$

$$X' = \left\{(1-\delta)\hat{x} + u(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W)\right\}. \tag{C.14}$$

For the case of $\mu = 0$, the Euler equation becomes:

$$-\gamma u(x, X) + \beta \mathbb{E}\left[P(X') + \gamma(1-\delta)u(x', X')\right] = 0. \tag{C.15}$$

In this case, the $x$ drops out of the Euler equation. Hence, $u$ does not depend on $x$, and the Euler equation can be rewritten as:

$$-\gamma u(X) + \beta \mathbb{E}\left[P(X') + \gamma(1-\delta)u(X')\right] = 0, \tag{C.16}$$

and the law of motion for $X$ is described by equation (C.9).

# Appendix D LQ Formulations

This section solves the special case of the LQ model. The purpose is to demonstrate the symmetry and concentration of measure when we know the closed-form solution of the problem.

## D.1 Proof of Proposition 5: Linear-quadratic Gaussian control

With an LQ Gaussian model, we know that the policy will be affine. Therefore, $\hat{u}$ for other firms can be written as

$$\hat{u}(X_i, X) = H_0 + H_2 X_i + \frac{H_1}{N}\sum_{i=1}^{N} X_i \quad i = 1, ..., N, \tag{D.1}$$

for some undetermined $H_0$, $H_1$, and $H_2$. As shown in Proposition 4, in a permutation invariant symmetric equilibrium, $u(\hat{x}, X)$ does not depend on $\hat{x}$ and, hence, $H_2 = 0$. The symmetric constants, which we arbitrarily define as $\frac{H_1}{N}$ for all of the $X$ vector, are the only ones that can fulfill permutation invariance with respect to the $\pi$ permutations defined in equation (21). The law of motion for $\vec{x}$ is then:

$$\vec{x}' = A\vec{x} + Bu + C\vec{w} \tag{D.2}$$

where $A$, $B$, and $C$ are defined in equations (22), (23), and (24). The reward and policy $u$ of the firm of interest at state $\vec{x}$ take the form:

$$-\vec{x}^{\top} R\vec{x} - u^{\top} Q u, \tag{D.3}$$

where $R$ and $Q$ are defined in equations (25) and (26).

For a given pair of $(H_0, H_1)$, the solution of this LQ problem has the form:

$$\vec{v}(\vec{x}) = -\vec{x}^\top P \vec{x} - d \tag{D.4}$$
$$\vec{u}(\vec{x}) = -F\vec{x}, \tag{D.5}$$

where $P$ solves the following discounted algebraic matrix Riccati equation:

$$P = R + \beta A^\top P A - \beta^2 A^\top P B \big(Q + \beta B^\top P B\big)^{-1} B^\top P A, \tag{D.6}$$

$d$ can be found via:

$$d = \frac{\beta}{1-\beta} \text{trace}\big(PCC^\top\big), \tag{D.7}$$

and $F$ satisfies:

$$F = \beta\big(Q + \beta B^\top P B\big)^{-1} B^\top P A. \tag{D.8}$$

Ljungqvist and Sargent (2018, ch. 5) derive equations (D.6), (D.7), and (D.8).

## D.2 Proof of Proposition 5: Permutation invariance

Now, we illustrate the permutation invariance of the optimal policy and value function of the linear quadratic model described in Proposition 5. Define $\vec{y} \equiv \pi\vec{x}$. The value function associated with $\vec{y}$ in (29) can be written as:

$$v(\vec{y}) = -\vec{y}^\top P \vec{y} - d = -\vec{x}^\top \big(\pi^\top P \pi\big)\vec{x} - d,$$

where $P$ solves the discounted algebraic Riccati equation:

$$P = R + \beta A^\top P A - \beta^2 A^\top P B \big(Q + \beta B^\top P B\big)^{-1} B^\top P A. \tag{D.9}$$

Sandwiching both sides of Equation (D.9) by $\pi^\top$ and $\pi$ and using the fact that $R$ is permutation invariant (i.e., $\pi^\top R \pi = R$) yields:

$$\pi^\top P \pi = R + \beta \pi^\top A^\top P A \pi - \beta^2 \pi^\top A^\top P B \big(Q + \beta B^\top P B\big)^{-1} B^\top P A \pi.$$

Since the deterministic part of the laws of motion $A$ is permutation equivariant (i.e., $A\pi = \pi A$) and the vector $B$ remains unchanged under a permutation (i.e., $\pi B = B$), the above equation can be rewritten as:

$$\pi^\top P\pi = R + \beta A^\top\big(\pi^\top P\pi\big)A - \beta^2 A^\top\big(\pi^\top P\pi\big)B\big[Q + \beta B^\top(\pi^\top P\pi)B\big]^{-1}B^\top(\pi^\top P\pi)A. \tag{D.10}$$

Notice that $\pi^\top P\pi$ solves the same discounted algebraic Riccati equation (D.9) as $P$. Since the solution of equation (D.9) is unique:

$$\pi^\top P\pi = P.$$

The constant term $d$ can be written as

$$\begin{aligned} d &= \beta(1-\beta)^{-1}\mathrm{trace}(P\pi CC^\top\pi^\top) \\ &= \beta(1-\beta)^{-1}\mathrm{trace}(\pi^\top P\pi CC^\top) \\ &= \beta(1-\beta)^{-1}\mathrm{trace}(PCC^\top). \end{aligned} \tag{D.11}$$

As shown above, $d$ remains unchanged under permutation and, therefore:

$$v(\vec{y}) = v(\pi\vec{x}) = v(\vec{x}).$$

The optimal policy $\vec{u}(\vec{y})$ can be written as

$$u(\vec{y}) = -\beta\big(Q + \beta B^\top PB\big)^{-1}B^\top PA\vec{y} = -\beta\big(Q + \beta B^\top PB\big)^{-1}B^\top(\pi^\top P\pi)A\vec{x}.$$

Since $(\pi^\top P\pi) = P$, then:

$$u(\vec{y}) = u(\pi\vec{x}) = u(\vec{x}).$$

**Structure of the $P$ Quadratic Form.** When solving for Proposition 5, you can further find that the permutation invariant matrix $P$ is of the form

$$P \equiv \left[\begin{array}{cc|c} p_1 & p_2 & \frac{p_3}{N}\mathbf{1}_N^\top \\ p_2 & 0 & \frac{p_4}{N}\mathbf{1}_N^\top \\ \hline \frac{p_3}{N}\mathbf{1}_N & \frac{p_4}{N}\mathbf{1}_N & \frac{p_5}{N^2}\mathbf{1}_N\mathbf{1}_N^\top \end{array}\right] \tag{D.12}$$

for some $p_1, \ldots p_5$. The lower block of this matrix is $\frac{p_5}{N^2}\mathbf{1}_N\mathbf{1}_N^\top$ (the $N \times N$ matrix of 1s is then the outer-product $\mathbf{1}_N\mathbf{1}_N^\top$.) This shows the symmetry with respect to the $i = 2, \ldots N$ states in $\vec{x}$ due to the block structure similar to than in equation (21), and the magnitude of their interaction (i.e., the lower block decreases at rate $N^2$ rather than $N$).

## D.3 Proof of Proposition 5: Concentration of measure

The optimal policy for $\vec{x}'$ can be written as:

$$u(\vec{x}') = -F\vec{x}' = -F(A\vec{x} + Bu + C\vec{w}).$$

The optimal policy defined in Proposition 5 for a given $H_0$ and $H_1$ is $u(\vec{x}) = H_0 + \frac{H_1}{N}\sum_{i=1}^{N} X_i = H_0 + H_1\mathbb{E}[x]$, and, hence, a bounded function in $N$ according to Definition 3. For any finite $H_1$ (including the equilibrium value), the gradient of $u(\vec{x}')$ with respect to the idiosyncratic shocks $W \equiv \{W_1, \ldots W_N\}$ becomes:

$$\nabla_W u(\vec{x}') = \frac{\sigma H_1}{N}\mathbf{1}_N.$$

Therefore, the square of the norm of the gradient of the policy can be bounded by:

$$\mathbb{E}\left[\|\nabla_W u(\vec{x}')\|^2\right] = \frac{\sigma^2 H_1^2}{N}.$$

Using Proposition 3 and conditioning on the aggregate shock $\omega$:

$$\mathbb{P}\left(\left|u(\vec{x}') - \mathbb{E}\left[u(\vec{x}') \mid w, \omega, \vec{x}\right]\right| \geq \epsilon\right) \leq \frac{\sigma^2 H_1^2}{\epsilon^2}\frac{1}{N}. \tag{D.13}$$

For the $\mu = 0$ case, the optimal policy $u(\cdot)$ does not depend on $x$ and, thus, conditioning on the idiosyncratic shock of the firm of interest is not required.

The value function can be written as:

$$v(\vec{x}') = -\vec{x}'^{\top} P \vec{x}' - d. \tag{D.14}$$

Recall that the vector of all states next period within the $\vec{x}'$ is $X' \equiv \begin{bmatrix} x_1' & \ldots & x_N' \end{bmatrix}^{\top}$. Given the symmetry in equations (31) and (D.12), we will drop the vector notation and instead use the $\vec{x}' \equiv \begin{bmatrix} 1 & x' & X' \end{bmatrix}$ notation where possible (i.e., let $x_0' = x'$ for the firm of interest).

Using the chain rule and additive structure of the idiosyncratic shocks, the gradient of the value function is:

$$\nabla_W v(\vec{x}') = -\frac{2\sigma(p_3 + p_4 x')}{N}\mathbf{1}_N - \frac{2p_5\sigma}{N^2}\mathbf{1}_N\mathbf{1}_N^{\top}X',$$

and then:

$$\|\nabla_W v(\vec{x}')\|^2 = \frac{\left[2\sigma(p_3 + p_4 x')\right]^2}{N} + \frac{8\sigma^2 p_5(p_3 + p_4 x')}{N^2}\mathbf{1}_N^{\top}X' + \frac{(2\sigma p_5)^2}{N^3}X'^{\top}\mathbf{1}_N\mathbf{1}_N^{\top}X'.$$

Since conditional on $(w, \omega, \vec{x})$, $x'$ it is not a stochastic variable, we have:

$$\mathbb{E}\left[\|\nabla_W v(\vec{x}')\|^2 \mid w, \omega, \vec{x}\right] = \frac{\left[2\sigma(p_3 + p_4 x')\right]^2}{N} + \frac{8\sigma^2 p_5(p_3 + p_4 x')}{N^2}\mathbf{1}_N^\top \mathbb{E}\left[X' \mid w, \omega, \vec{x}\right]$$
$$\frac{\left(2\sigma p_5\right)^2}{N^3}\mathbb{E}\left[X'^\top \mathbf{1}_N \mathbf{1}_N^\top X' \mid w, \omega, \vec{x}\right].$$

Since $X'$ is a normal random vector with conditional mean $\mathbb{E}\left[X' \mid w, \omega, \vec{x}\right] = X + \big(u(X) + \eta\omega\big)\mathbf{1}_N$ and conditional covariance $\operatorname{cov}\left(X' \mid \; w, \omega, \vec{x}\right) = \sigma^2 \mathbf{I}_N$, we have:[18]

$$\mathbb{E}\left[\|\nabla_W v(\vec{x}')\|^2 \mid w, \omega, \vec{x}\right] = \frac{\left[2\sigma(p_3 + p_4 x')\right]^2}{N} + \frac{8\sigma^2 p_5(p_3 + p_4 x')}{N}\left[\frac{\mathbf{1}_N^\top X}{N} + u(X) + \eta\omega\right]$$
$$\frac{(2\sigma p_5)^2}{N}\left[\frac{\operatorname{trace}\left(\sigma^2 \mathbf{1}_N \mathbf{1}_N^\top\right)}{N^2} + \left(\frac{\mathbf{1}_N X}{N} + u(X) + \eta\omega\right)^2\right].$$

Since $\frac{\mathbf{1}_N X}{N}$ and $u(X)$ are bounded functions in $N$ (Definition 3), there exists $K_1$ such that:

$$\left|\frac{\mathbf{1}_N X}{N} + u(X) + \eta\omega\right| \leq K_1.$$

Also note that:

$$\frac{\operatorname{trace}\left(\sigma^2 \mathbf{1}_N \mathbf{1}_N^\top\right)}{N^2} = \frac{\sigma^2}{N} \leq \sigma^2$$

Therefore, by the Cauchy-Schwarz inequality, there exists a positive $C_v$ such that

$$\mathbb{E}\left[\|\nabla_W v(\vec{x}')\|^2 \mid w, \omega, \vec{x}\right] \leq \frac{\sigma^2 C_v^2}{N}.$$

Hence, by Proposition 3:

$$\mathbb{P}\left(\left|v(\vec{x}') - \mathbb{E}\left[v(\vec{x}') \mid w, \omega, \vec{x}\right]\right| \geq \epsilon\right) \leq \frac{\sigma^2 C_v^2}{\epsilon^2}\frac{1}{N}. \tag{D.15}$$

The inequalities in (D.13) and (D.15) show that even with a single draw, we can calculate the expectation as $N$ increases. A high-dimensional state space is a help rather than a hindrance when calculating expectations (as long as there is sufficient independence in the states) because of the concentration of measure phenomenon. Inequality (33) shows that, with a single draw of $\vec{w}$, we know the precise distribution of the deviation from the expectation, and with a large enough $N$, it converges to 0. We could take multiple draws if required. But since Monte Carlo integration converges very slowly in general, the gains in extra accuracy will be minor even if we increase the number of draws substantially.

[18] If $\vec{z}$ is a random vector with mean $\vec{m}$ and covariance matrix $\Sigma$, then, for any matrix $\Lambda$, $\mathbb{E}\left[\vec{z}^\top \Lambda \vec{z}\right] = \operatorname{trace}(\Lambda\Sigma) + \vec{m}^\top \Lambda \vec{m}$.

The Euler residuals for a linear policy of the form $u(X) = H_0 + \frac{H_1}{N}\sum_{i=1}^{N} X_i$ are:

$$\begin{aligned}\varepsilon(X,u) =& \gamma u(X) - \beta\Big[\big(\alpha_0 - \frac{\alpha_1}{N}\sum_{i=1}^{N} X_i + u(X) + \sigma w_i\big) + \\ &\gamma(1-\delta)\big(H_0 + \frac{H_1}{N}\sum_{i=1}^{N} X_i + u(X) + \sigma w_i\big)\Big].\end{aligned}$$

From the Euler equation, equation (19) becomes:

$$\gamma u(X) - \beta\Big[\big(\alpha_0 - \frac{\alpha_1}{N}\sum_{i=1}^{N} X_i + u(X)\big) + \gamma(1-\delta)\big(H_0 + \frac{H_1}{N}\sum_{i=1}^{N} X_i + u(X)\big)\Big] = 0.$$

The Euler residuals are then:

$$\varepsilon(X,u) = \beta\sigma\big[\alpha_1 - \gamma(1-\delta)H_1\big]\frac{1}{N}\sum_{i=1}^{N} w_i.$$

Using the fact that $w_i \sim \mathcal{N}(0,1)$, the Euler residual is a Gaussian random variable

$$\varepsilon(X,u) \sim \mathcal{N}\big(0, \frac{\beta^2\sigma^2\big[\alpha_1 - \gamma(1-\delta)H_1\big]^2}{N}\big).$$

## D.4 Representation of the $v$ function

If it is known that the LQ problem satisfies certainty equivalence and the value function is quadratic and permutation invariant, $v$ can be represented as:

$$\begin{aligned}v(x,X) &= -\Big(p_1 + d + 2p_2 x + 2p_3\frac{\sum_{i=1}^{N} X_i}{N} + 2p_4 x\frac{\sum_{i=1}^{N} X_i}{N} + p_5\big(\frac{\sum_{i=1}^{N} X_i}{N}\big)^2\Big) \\ &= \rho_v\big(x, \frac{1}{N}\sum_{i=1}^{N}\phi_v(X_i)\big)\end{aligned}$$

In other words, $\phi_v : \mathbb{R} \to \mathbb{R}^1$ (i.e., $L = 1$) is the identity function $\phi_v(X) = \Big[X\Big]$, and $\rho_v : \mathbb{R}\times\mathbb{R}^1 \to \mathbb{R}$ such that:

$$\rho_v(x,y) = -(p_1 + d + 2p_2 x + 2p_3 y + 2p_4 xy + p_5 y^2).$$

# Appendix E Additional Numerical Results

## E.1 More network variations

Table 2 reports a more complete list of experiments, including those of Table 1.

Table 2: Linear Model Performance in Additional Experiments

| group | description | Time (s) | Params (K) | Train MSE ($\varepsilon$) | Test MSE ($\varepsilon$) | Val MSE ($\varepsilon$) | Policy Error ($\lvert u - u_{\text{ref}}\rvert$) | Policy Error $\left(\frac{\lvert u-u_{\text{ref}}\rvert}{u_{\text{ref}}}\right)$ |
|---|---|---|---|---|---|---|---|---|
| $\phi$(Identity) | Baseline | 42 | 49.4 | 4.1e-06 | 3.3e-07 | 3.3e-07 | 2.9e-05 | 0.10% |
| | Thin (64 nodes) | 33 | 12.4 | 3.7e-06 | 2.7e-07 | 2.7e-07 | 3.4e-05 | 0.10% |
| | Shallow (2 layers) | 159 | 16.6 | 3.7e-06 | 7.8e-07 | 7.6e-07 | 9.4e-03 | 33.53% |
| $\phi$(ReLU) | Baseline | 107 | 66.8 | 3.7e-06 | 3.3e-07 | 3.3e-07 | 2.7e-05 | 0.09% |
| | L = 1 | 88 | 66.0 | 1.8e-05 | 2.3e-07 | 2.4e-07 | 2.8e-05 | 0.10% |
| | L = 2 | 86 | 66.3 | 1.3e-05 | 2.1e-07 | 2.2e-07 | 2.6e-05 | 0.08% |
| | L = 8 | 70 | 67.8 | 3.0e-05 | 5.9e-07 | 5.9e-07 | 3.3e-05 | 0.11% |
| | L = 16 | 91 | 69.9 | 5.5e-06 | 1.5e-07 | 1.5e-07 | 2.1e-05 | 0.07% |
| | Shallow($\phi$ : 1 layer, $\rho$ : 2 layers) | 79 | 17.7 | 2.0e-06 | 5.5e-07 | 5.5e-07 | 3.2e-05 | 0.11% |
| | Shallow($\phi$ : 1 layer) | 58 | 50.4 | 8.7e-06 | 1.5e-07 | 1.5e-07 | 2.5e-05 | 0.08% |
| | Shallow($\rho$ : 2 layers) | 89 | 34.0 | 3.1e-06 | 4.2e-07 | 4.2e-07 | 2.7e-05 | 0.09% |
| | Deep($\phi$ : 4 layers, $\rho$ : 8 layers) | 242 | 165.1 | 2.1e-03 | 2.2e-03 | 2.1e-03 | 2.7e-03 | 8.50% |
| | Very Thin($\phi, \rho$ : 16 nodes) | 45 | 1.2 | 1.2e-05 | 4.9e-07 | 4.9e-07 | 3.2e-05 | 0.10% |
| | Thin($\phi, \rho$ : 64 nodes) | 87 | 17.0 | 1.1e-05 | 4.5e-07 | 4.5e-07 | 3.0e-05 | 0.10% |
| | Wide($\phi, \rho$ : 256 nodes) | 115 | 264.7 | 5.4e-06 | 4.0e-07 | 4.0e-07 | 4.1e-05 | 0.13% |
| $\phi$(Moments) | Baseline | 55 | 49.8 | 1.4e-06 | 7.6e-07 | 7.6e-07 | 2.8e-05 | 0.09% |
| | Moments (1,2) | 211 | 49.5 | 2.4e-06 | 1.1e-06 | 2.3e-06 | 4.4e-05 | 0.14% |
| | Very Shallow(1 layer) | 241 | 0.6 | 1.1e-05 | 8.4e-06 | 7.9e-06 | 1.1e-02 | 34.00% |
| | Shallow (2 layers) | 137 | 17.0 | 1.6e-06 | 9.9e-07 | 9.5e-07 | 1.8e-02 | 59.41% |
| | Deep(8 layers) | 241 | 115.3 | 2.8e-06 | 1.2e-06 | 1.0e-06 | 5.2e-05 | 0.16% |
| | Thin (64 nodes) | 82 | 12.6 | 1.6e-06 | 9.1e-07 | 9.2e-07 | 3.8e-05 | 0.12% |
| | Wide (256 nodes) | 61 | 197.9 | 1.8e-06 | 8.7e-07 | 8.0e-07 | 4.3e-05 | 0.13% |

## E.2 Nonlinear demand function

The results in Table 3 confirm the conventional wisdom in the deep learning literature. First, *Very Shallow* in $\phi$(ReLU), and *Very Shallow* and *Shallow* in $\phi$(Moments) highlight the role of depth and generalizability. In these experiments, the MSE of the Euler residuals on both the training and validation data is very low. However, as seen in Test MSE ($\varepsilon$), the approximated policy is incapable of generalization. Second, for cases where there is nonlinearity in the function of interest, depth plays a more important role than pure capacity (number of parameters). For instance, in $\phi$(Moments), *Thin* has way fewer parameters than *Shallow*, but it has more depth. *Thin* has an extraordinarily higher generalization power than *Shallow*.

Table 3: Nonlinear Model Performance

| group | description | Time (s) | Params (K) | Train MSE ($\varepsilon$) | Test MSE ($\varepsilon$) | Val MSE ($\varepsilon$) |
|---|---|---|---|---|---|---|
| $\phi$(ReLU) | Baseline | 60 | 67.1 | 1.4e-05 | 4.7e-06 | 3.3e-06 |
| | L = 1 | 109 | 66.3 | 9.4e-06 | 1.3e-05 | 4.5e-06 |
| | L = 2 | 73 | 66.6 | 1.0e-05 | 3.3e-06 | 2.3e-06 |
| | L = 8 | 73 | 68.1 | 1.1e-05 | 4.9e-06 | 2.0e-06 |
| | L = 16 | 72 | 70.2 | 1.5e-05 | 5.4e-06 | 1.7e-06 |
| | Very Shallow($\phi, \rho$ : 1 layer) | 136 | 1.4 | 8.9e-06 | 4.8e+06 | 4.9e-06 |
| | Shallow($\phi, \rho$ : 2 layers) | 47 | 34.3 | 1.0e-05 | 9.2e-06 | 2.8e-06 |
| | Thin($\phi, \rho$ : 32 nodes) | 52 | 4.5 | 1.3e-05 | 6.0e-06 | 2.7e-06 |
| $\phi$(Moments) | Baseline | 26 | 49.8 | 6.0e-06 | 5.0e-06 | 3.8e-06 |
| | Moments (1) | 24 | 49.4 | 2.7e-05 | 6.5e-06 | 3.4e-06 |
| | Moments (1,2) | 27 | 49.5 | 8.0e-06 | 5.1e-06 | 3.6e-06 |
| | Very Shallow (1 layer) | 252 | 0.6 | 8.3e-06 | 1.4e+00 | 5.0e-06 |
| | Shallow (2 layers) | 35 | 17.0 | 5.8e-06 | 1.2e+00 | 4.4e-06 |
| | Thin (32 nodes) | 66 | 3.2 | 1.1e-05 | 9.7e-06 | 4.4e-06 |