# Training Neural Networks Embedded in Dynamic Discrete Choice Models

Ecenur Oğuz and Robert L. Bray

Kellogg School of Management, Northwestern University

April 20, 2026

**Abstract**

We develop the first general-purpose estimator for infinite-horizon dynamic discrete choice models whose estimation problem, after pre-computation, is unencumbered by large systems of linear equations—either imposed as constraints, or embedded in the objective function. Our unnested fixed point (UFXP) and optimal unnested fixed point (OUFXP) estimators exploit a dual representation of Bellman's equation to separate the utility parameters from the dynamic programming fixed point. We establish the consistency and asymptotic normality of UFXP and OUFXP, as well as the efficiency of the latter. Our estimators enable researchers to model utility functions non-parametrically via flexible neural-network approximations.

**Keywords:** Dynamic discrete choice, nested fixed point (NFXP), conditional choice probability (CCP), structural estimation, neural networks

## 1 Introduction

Dynamic programs are notoriously difficult to estimate. To reduce the computational burden, classic estimators leverage two properties commonly found in dynamic discrete choice models: likelihood concavity and utility function linearity. When the empirical likelihood is concave, the optimizer finds the (pseudo) MLE on the first try—it doesn't get led astray by local optima or saddle points. And when the utility function is linear, the estimation problem decouples into two much easier problems: one that solves a set of Bellman-like equations, and the other that maximizes the empirical likelihood, given the corresponding fixed point solutions. Further, utility function linearity almost always implies likelihood function concavity, so we generally enjoy both benefits when the utility function is affine.

Unfortunately, affine functions have an Achilles heel: dimensionality. Splines and polynomial basis functions yield excellent approximations—when there's a single state variable. However, the number of "factors" or linear basis functions required to achieve a good approximation grows *exponentially* with the dimension of the state space.

Neural networks mitigate the problem of dimensionality, as they can recognize patterns in high-dimensional spaces with a parsimonious set of parameters. Breathtaking advances in neural networks mean that we must no longer model organic human preferences with flat hyperplanes. Indeed, a growing literature demonstrates the efficacy of neural networks to model utility functions in discrete choice models.[1] Han et al. (2022) and Sifringer et al. (2020) replace fixed coefficients with neural network layers, allowing preferences to vary flexibly with observed characteristics and capturing heterogeneity that linear specifications cannot accommodate. Hsieh et al. (2025) further formalize this approach by proposing neural utility representations that preserve economic structure while accommodating behavioral nonlinearities. Aouad and Désir (2025) and Arkoudi et al. (2023) extend these ideas to general random utility maximization frameworks and high-dimensional choice environments. However, all these recent works correspond to *static models*.

We bring neural networks to *dynamic models* by introducing them to the celebrated dynamic discrete choice problem.[2] This model was an instant classic, winning its creator, John Rust, the 1992 Ragnar Frisch Medal, and inspiring four survey articles: Rust (1986), Aguirregabiria and Mira (2010), Arcidiacono and Ellickson (2011), and Kasahara and Shimotsu (2018). We update this model by casting its utility function *non-parametrically*: rather than a pre-specified functional form, we allow utilities to arise from a neural network of arbitrary complexity (we make no other changes besides this one substitution: e.g., the same Bellman equation governs the system dynamics).[3]

Unfortunately, embedding a neural network into Rust's framework sacrifices the two key properties that facilitate estimation: concavity and linearity. Losing concavity means running the opti-

[1] This structural approach is distinct from earlier work that applies machine learning primarily as a predictive classifier in discrete choice settings (van Cranenburgh et al., 2022), or from approaches that use neural networks to map moments to parameters (Wei and Jiang, 2025).

[2] A notable recent utilization of neural networks in dynamic settings appears in van der Laan et al. (2025) for the estimation of choice probabilities and continuation values. In contrast, we embed the neural network directly into the economic model as a structural primitive to recover the underlying utility function.

[3] This non-parametric reframing echoes Rust's (1987a, p. 1001) original paper, which cautioned against an over-reliance on "closed-form" specifications:

> The second objective of this paper is to illustrate a new estimation method that allows me to compute maximum likelihood estimates of the primitive parameters of a class of controlled stochastic processes, even though there is no analytic formula for the associated likelihood function. ... I argue, however, that models with closed-form solutions have certain inherent limitations which make them poor candidates for empirical work.

mizer from many starting points, as few gradient ascent paths will end at the global optimum. And losing linearity means nesting the dynamic programming problem within the estimation problem, so that the difficulty of the first multiplies, rather than adds to, the difficulty of the second (contrary to popular belief, even CCP estimators suffer from nested fixed points, in which additional likelihood gradient ascents necessitate additional Bellman solutions: see Section 3.1). Further, losing these two speed-ups is especially burdensome when training neural networks, given their high intrinsic training costs. Indeed, estimating a neural network utility function with traditional methods would require solving at least one Bellman equation for every neural network backpropagation.

We develop a new estimator that can efficiently train a neural network embedded in a dynamic discrete choice problem. Rather than maximize a (pseudo) likelihood problem, as other estimators do, we minimize the norm of a random compression of the first-order conditions. Whereas the old maximum likelihood problem decouples from the underlying dynamic programming problem only when the utility function is linear, our new norm minimization problem decouples *always*. Put differently, for the maximum likelihood problem, we must solve a Bellman equation every time we evaluate the objective (when the utility function is non-linear), but for the norm minimization problem, we must solve upfront slightly more Bellman equations than there are utility parameters (e.g., neural network weights), after which we can evaluate the objective with ease. This change shortens the time between neural network backpropagations from hours to seconds (see Section 4.2). Our estimator thus frees us from a reliance on linearity.

Our estimator also frees us from a reliance on concavity (or convexity, in the norm minimization case) by enabling rapid and inexpensive exploration of the optimization landscape—precisely what is required when optimizing a non-concave objective, where exhaustive search becomes unavoidable. The curse of dimensionality ensures that almost all general-purpose dynamic discrete choice estimators spend almost all of their time solving Bellman-style fixed points.[4] Our estimator is no exception: the norm minimization time is negligible relative to the fixed point solution time. And since we can recycle our fixed point solutions across optimization runs, we can thus blanket the objective function with many inexpensive optimization runs. For example, initializing our optimization routine with 1000 different starting values scales the estimation time by only a factor of 7.7 (see Section 5.3). In contrast, initializing a traditional estimator with 1000 different starting values would take 1000 times as long.

Further, our Bellman-style fixed points are *independent of the neural network configuration*,

[4]We write 'general-purpose' estimators to exclude the special-purpose estimators designed for the rare cases in which the econometrician can circumvent Bellman's equation. See Section 4.2.

which means that estimating many versions of the model is essentially no more costly than estimating a single version. Accordingly, our estimators enable the econometrician to iteratively craft the neural network to the data—assigning the most flexibility where the sample is the richest—rather than blindly specifying its configuration upfront.

# 2 Model

## 2.1 Discrete Choice Preliminaries

We begin with some basic discrete choice theory. Consider an agent that chooses one action from a set of actions $\mathbb{A} \equiv \{1, \ldots, A\}$. The utility of choosing action $a \in \mathbb{A}$ is $e_a + c_a$, where $c \in \mathbb{R}^A$ is a vector of deterministic values and $e \in \mathbb{R}^A$ is a vector of stochastic shocks that resolves from continuous density function $g$, which has full support over $\mathbb{R}^A$. McFadden's social surplus function denotes the expected utility of the top choice, unconditional on the random component:

$$\nu(c) \equiv \int_{e \in \mathbb{R}^A} \max_{a \in \mathbb{A}} (e_a + c_a) g(e) de.$$

The Williams-Daly-Zachary theorem establishes that the gradient of the social surplus function yields the corresponding choice probabilities:

$$\begin{aligned} \nabla \nu(c) &= p(c), \\ \text{where} \quad p_a(c) &\equiv \Pr\left(a = \arg\max_{j \in \mathbb{A}} e_j + c_j\right). \end{aligned}$$

Choice probability vector $p(c)$ resides in $(A-1)$-simplex $\mathbb{P}$.

Next, define $I^+$ as the matrix that maps $(v_1, \ldots, v_{A-1})'$ to $(v_1, \ldots, v_{A-1}, 0)'$, and define $\Delta$ as the matrix that maps $(v_1, \ldots, v_A)'$ to $(v_1 - v_A, \ldots, v_{A-1} - v_A)'$. Composing these two matrices yields a normalizing operator: matrix $I^+\Delta$ maps $(v_1, \ldots, v_A)'$ to $(v_1 - v_A, \ldots, v_A - v_A)'$. Note that normalized value vector $I^+\Delta c$ yields the same choice probabilities as initial value vector $c$:

$$\begin{aligned} p_a(I^+\Delta c) &= \Pr\left(a = \arg\max_{j \in \mathbb{A}} e_j + c_j - c_A\right) \\ &= \Pr\left(a = \arg\max_{j \in \mathbb{A}} e_j + c_j\right) \\ &= p_a(c). \end{aligned}$$

Hence, we have

$$\nabla\nu(I^+\Delta c) = \nabla\nu(c).$$

Next, for $v \in \mathbb{R}^{A-1}$ define

$$\rho(v) \equiv \nabla\nu(I^+ v).$$

This function maps differenced value vector $\Delta c \in \mathbb{R}^{A-1}$ to choice probability vector $p(c) \in \mathbb{P}$:

$$\rho(\Delta c) = \nabla\nu(I^+\Delta c) = \nabla\nu(c) = p(c).$$

In fact, this mapping is invertible (Hotz and Miller, 1993), so there exists $\rho^{-1}$ that satisfies

$$\rho^{-1}(p(c)) = \rho^{-1}(\rho(\Delta c)) = \Delta c.$$

The following result establishes that the random discrete problem admits an equivalent deterministic continuous problem.

**Proposition 1.** *For any $c \in \mathbb{R}^A$, the social surplus satisfies*

$$\begin{aligned} \nu(c) &= \sup_{p\in\mathbb{P}} \epsilon(p) + p'c, \\ \textit{where} \quad \epsilon(p) &\equiv \nu(I^+\rho^{-1}(p)) - p' I^+ \rho^{-1}(p) \\ &= \int_{e\in\mathbb{R}^A} \Big( \sum_{a\in\mathbb{A}} \mathbb{1}\{a = \arg\max_{j\in\mathbb{A}} e_j + I_j^+ \rho^{-1}(p)\} e_a \Big) g(e) de. \end{aligned}$$

*Moreover, this optimization problem's maximizer,*

$$p^* \equiv \arg\max_{p\in\mathbb{P}} \epsilon(p) + p'c,$$

*satisfies the following first-order conditions*

$$\rho^{-1}(p^*) = \Delta c.$$

For a concrete example, suppose the elements of $e$ are independent standard Gumbels. In this case, $\nu$ and $\epsilon$ respectively coincide with the log-sum-exp function and the information entropy

function (both increased by Euler–Mascheroni constant $\gamma$), and $\rho$ and $\rho^{-1}$ respectively coincide with the softmax function and the log-odds function:

$$\begin{aligned}
\nu(c) &= \gamma + \log\Big(\sum_{a\in\mathbb{A}} \exp(c_a)\Big), \\
\epsilon(p) &= \gamma - \sum_{a\in\mathbb{A}} p_a \log(p_a), \\
\rho_a(\Delta c) &= \frac{\exp(c_a)}{\sum_{j\in\mathbb{A}} \exp(c_j)}, \\
\text{and} \quad \rho_a^{-1}(p) &= \log(p_a) - \log(p_A), \\
\text{when} \quad e_a &\overset{\text{i.i.d.}}{\sim} \text{EV1}(0,1).
\end{aligned}$$

It is straightforward to verify that Proposition 1 holds in this case.

## 2.2 Dynamic Program

We are now ready to define the dynamic discrete choice problem. We observe an agent confronted with a discrete-time, infinite-horizon Markov decision process. In each period, the agent chooses action variable $a \in \mathbb{A} \equiv \{1, \ldots, A\}$ in response to state variables $e$ and $x$. The agent observes $e \in \mathbb{R}^A$ and $x \in \mathbb{X} \equiv \{1, \ldots, X\}$, but we do not observe the former, which will serve as our statistical error term. Variable $e$ is a multivariate random vector with continuous density function $g$, which has full support over $\mathbb{R}^A$. Taking action $a$ in state $(e, x)$ sets the probability density of the next-period state, $(e', x')$, to $f_a(x'|x)g(e')$, and yields contemporaneous utility $e_a + u_a^\theta(x)$. Our objective is to estimate the utility parameter $\theta \in \Theta \subset \mathbb{R}^t$, which will ultimately represent the weights and biases of a neural network. In each period, the agent chooses the action that yields the largest expected discounted utility, under discount factor $\beta \in [0, 1)$.

We could frame $\beta$ as a parameter to estimate, but doing so slightly complicates our procedure (see Appendix Section B), so we will follow the usual practice and take it as given. We similarly omit unobserved heterogeneity from our baseline model; however, accommodating it is straightforward, as our estimators are compatible with standard two-stage procedures (see Appendix Section C). Also, the transition kernel $f$ will almost certainly be parameterized, but these parameters are trivial to pre-estimate, so we take the transition probabilities as given, without loss of generality.

The following Bellman equation specifies the agent's value function, which quantifies their ex-

pected discounted utility:

$$\bar{v}^\theta(e,x) \equiv \max_{a\in\mathbb{A}} e_a + u_a^\theta(x) + \beta \sum_{x'\in\mathbb{X}} f_a(x'|x) \int_{e'\in\mathbb{R}^A} \bar{v}^\theta(e',x')g(e')de'. \tag{1}$$

Our dynamic program has state space $\mathbb{R}^A \times \mathbb{X}$ and action space $\mathbb{A}$. But infinite state spaces are tricky, since they require interpolation, and discrete action spaces are a nuisance, since they do not yield first-order conditions, as Rust (1987b, p. 1000-1) explains:

> Unfortunately, since the controlled process $\{i_t, x_t\}$ is the solution to a discrete stochastic control problem, one will rarely find a closed-form solution for its probability density or any sort of "first order condition" convenient for estimation.

Fortunately, Proposition 1 enables us to translate our finite-action-space, infinite-state-space dynamic program to an analogous infinite-action-space, finite-state-space dynamic program. This transformation will yield first-order conditions and will allow us to disregard the unobserved state variable. We drop this error term by integrating over it, defining a new value function that characterizes the agent's expected discounted utility, conditional on $x$, but not on $e$:

$$\begin{aligned}
v^\theta(x) &\equiv \int_{e\in\mathbb{R}^A} \bar{v}^\theta(e,x)g(e)de \\
&= \int_{e\in\mathbb{R}^A} \Big( \max_{a\in\mathbb{A}} e_a + u_a^\theta(x) + \beta \sum_{x'\in\mathbb{X}} f_a(x'|x) \int_{e'\in\mathbb{R}^A} \bar{v}^\theta(e',x')g(e')de' \Big) g(e)de \\
&= \int_{e\in\mathbb{R}^A} \Big( \max_{a\in\mathbb{A}} e_a + u_a^\theta(x) + \beta \sum_{x'\in\mathbb{X}} f_a(x'|x) v^\theta(x') \Big) g(e)de \\
&= \nu\Big( u^\theta(x) + \beta \sum_{x'\in\mathbb{X}} f(x'|x) v^\theta(x') \Big) \\
&= \sup_{p\in\mathbb{P}} \epsilon(p) + p' u^\theta(x) + \beta \sum_{x'\in\mathbb{X}} p' f(x'|x) v^\theta(x'),
\end{aligned} \tag{2}$$

where $u^\theta(x)$ and $f(x'|x)$ are length-$A$ vectors with $a$th elements $u_a^\theta(x)$ and $f_a(x'|x)$.

Whereas Bellman equation (1) characterizes a dynamic program with state $(e,x) \in \mathbb{R}^A \times \mathbb{X}$ and action $a \in \mathbb{A}$, Bellman equation (2) characterizes a dynamic program with state $x \in \mathbb{X}$ and action $p \in \mathbb{P}$. Thus, rather than elements of $\mathbb{A}$, we can equivalently suppose that the agent chooses *conditional choice probability* (CCP) measures over $\mathbb{A}$. Under this framing, the policy functions that govern the agent's behavior are mappings from $\mathbb{X}$ to $\mathbb{P}$. We will use $\mathbb{X}^\mathbb{P}$ to denote the set of such mappings. Further, we will use $p$ to denote a generic element of $\mathbb{P}$ (i.e., a single CCP vector),

and use $P$ to denote a generic element of $\mathbb{X}^{\mathbb{P}}$ (i.e., a collection of CCP vectors indexed by $\mathbb{X}$).

Adhering to the convention of using capital letters for objects that span over $\mathbb{X}$, define $E(P)$, $P_a$, $U_a^\theta$, and $V^\theta$ as the length-$X$ vectors with $x$th elements $\epsilon(P(x))$, $P_a(x)$, $u_a^\theta(x)$, and $v^\theta(x)$, and define $F_a$ as the $X \times X$ matrix with $xx'$th element $f_a(x'|x)$. Next, define $U_P^\theta \equiv E(P) + \sum_{a\in\mathbb{A}} \text{diag}(P_a)U_a^\theta$ and $F_P \equiv \sum_{a\in\mathbb{A}} \text{diag}(P_a)F_a$ as the utility function and transition kernel associated with policy $P$. And, finally, define $C_V^\theta$ as the $X \times A$ matrix with $a$th column $U_a^\theta + \beta F_a V$. Each element of $C_{V^\theta}^\theta$ is a *choice specific value* (CSV)—i.e., an expected discounted utility, net of the current error term.

Proposition 1 establishes that these CSVs underlie the first-order conditions associated with the Bellman equation (2):

$$\rho^{-1}(P^\theta) = C_{V^\theta}^\theta \Delta'. \tag{3}$$

The expression above relies on the convention that $\rho^{-1}$ and $\rho$ are evaluated row-wise when applied to a matrix. For example, the $x$th row of system (3) specifies that

$$\rho^{-1}(P^\theta(x)) = \Delta u^\theta(x) + \beta \sum_{x'\in\mathbb{X}} \Delta f(x'|x) v^\theta(x'),$$

and the $a$th column specifies that

$$\rho_a^{-1}(P^\theta) = U_a^\theta - U_A^\theta + \beta(F_a - F_A)V^\theta.$$

System (3) implies the following optimal policy:

$$\begin{aligned} P^\theta &\equiv P_{V_\theta}^\theta \\ \text{where} \quad P_V^\theta &\equiv \rho(C_V^\theta \Delta'). \end{aligned} \tag{4}$$

In the expression above, $P_V^\theta \in \mathbb{X}^{\mathbb{P}}$ specifies the optimal policy to follow today when $V \in \mathbb{R}^X$ specifies the value tomorrow. And whereas $P_V^\theta$ maps value functions to policy functions, $V_P^\theta$ maps policy functions to value functions:

$$\begin{aligned} V_P^\theta &\equiv U_P^\theta + \beta F_P V_P^\theta \\ &= (I - \beta F_P)^{-1} U_P^\theta. \end{aligned} \tag{5}$$

Specifically, $V_P^\theta$ characterizes the value of following policy $P$ forever. Composing $P_V^\theta$ and $V_P^\theta$ yields

the policy iteration operator:

$$P_P^\theta \equiv P_{V_P^\theta}^\theta.$$

The expression above yields the optimal policy for the agent to follow today, given that they will follow policy $P$ forever thereafter. It underlies the policy iteration algorithm, which is second only to the celebrated value iteration algorithm, which iteratively applies the following operator:

$$T_V^\theta \equiv U_{P_V^\theta}^\theta + \beta F_{P_V^\theta} V.$$

If $V$ characterizes tomorrow's value, then $T_V^\theta$ characterizes today's value. Bellman equation (2) establishes that $V^\theta$ is the unique solution to the following fixed point equation:

$$V = T_V^\theta. \tag{6}$$

# 3 Estimation

## 3.1 Nested Estimators

Rust (1987b) devised the first dynamic discrete choice estimator. His nested fixed point (NFXP) algorithm maximizes the empirical likelihood of the observed actions:

$$\hat{\theta}_{\text{NFXP}} \equiv \arg\max_{\theta \in \Theta} \mathcal{L}_{\text{NFXP}}^\theta$$
$$\text{where} \quad \mathcal{L}_{\text{NFXP}}^\theta \equiv \sum_{a \in \mathbb{A}} n_a' \log(P_a^\theta)$$

and $n_a$ is a length-$X$ vector whose $x$th element specifies the number of action-$a$, state-$x$ observations in our sample. Since this log likelihood depends on the optimal policy, which depends on the solution of (6), Rust's algorithm solves a Bellman equation every time it evaluates the empirical likelihood function. In other words, it nests a dynamic programming fixed point problem within a maximum likelihood problem.

However, (6) is not the only nested fixed point equation that bedevils the NFXP algorithm, as

the gradient of the likelihood function depends on the following $t$ fixed points:[5]

$$\tfrac{\partial}{\partial\theta_i}V^\theta = \sum_{a\in\mathbb{A}} \mathrm{diag}(P_a^\theta)\tfrac{\partial}{\partial\theta_i}U_a^\theta + \beta F_{P^\theta}\tfrac{\partial}{\partial\theta_i}V^\theta. \tag{7}$$

Specifically,

$$\begin{aligned}
\tfrac{\partial}{\partial\theta}\mathcal{L}^\theta_{\mathrm{NFXP}} &= \sum_{a\in\mathbb{A}} n'_a \mathrm{diag}(P_a^\theta)^{-1}\tfrac{\partial}{\partial\theta}P_a^\theta,\\
\text{where}\quad \tfrac{\partial}{\partial\theta}P_a^\theta &= \sum_{j\in\mathbb{A}\setminus\{A\}} \big(\tfrac{\partial}{\partial j}\rho_a(C^\theta_{V^\theta}\Delta')\big)\big(\tfrac{\partial}{\partial\theta}U_j^\theta - \tfrac{\partial}{\partial\theta}U_A^\theta + \beta(F_j - F_A)\tfrac{\partial}{\partial\theta}V^\theta\big)
\end{aligned}$$

and $\frac{\partial}{\partial j}\rho_a(M)$ denotes the $X \times X$ Jacobian of $\rho_a(M)$ with respect to the $j$th column of matrix $M$. Further, the Hessian matrix of the likelihood function depends on the $t(t+1)/2$ distinct fixed points defined in the following:

$$\begin{aligned}
\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}V^\theta = \sum_{a\in\mathbb{A}} \Big(&\mathrm{diag}(P_a^\theta)\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}U_a^\theta + \mathrm{diag}(\tfrac{\partial}{\partial\theta_j}P_a^\theta)\tfrac{\partial}{\partial\theta_i}U_a^\theta + \beta\mathrm{diag}(\tfrac{\partial}{\partial\theta_j}P_a^\theta)F_a\tfrac{\partial}{\partial\theta_i}V^\theta\Big)\\
&+ \beta F_{P^\theta}\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}V^\theta.
\end{aligned} \tag{8}$$

Specifically,

$$\begin{aligned}
\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^\theta_{\mathrm{NFXP}} =& \sum_{a\in\mathbb{A}} n'_a\mathrm{diag}(P_a^\theta)^{-1}\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}P_a^\theta - n'_a\mathrm{diag}(P_a^\theta)^{-2}\mathrm{diag}(\tfrac{\partial}{\partial\theta_j}P_a^\theta)\tfrac{\partial}{\partial\theta_i}P_a^\theta,\\
\text{where}\quad \tfrac{\partial^2}{\partial\theta_i\partial\theta_j}P_a^\theta =& \sum_{\ell\in\mathbb{A}\setminus\{A\}}\sum_{k\in\mathbb{A}\setminus\{A\}} \big(\tfrac{\partial^2}{\partial\ell\partial k}\rho_a(C^\theta_{V^\theta}\Delta')\big)\\
&\mathrm{diag}\big(\tfrac{\partial}{\partial\theta_j}U_\ell^\theta - \tfrac{\partial}{\partial\theta_j}U_A^\theta + \beta(F_\ell - F_A)\tfrac{\partial}{\partial\theta_j}V^\theta\big)\\
&\big(\tfrac{\partial}{\partial\theta_i}U_k^\theta - \tfrac{\partial}{\partial\theta_i}U_A^\theta + \beta(F_k - F_A)\tfrac{\partial}{\partial\theta_i}V^\theta\big)\\
&+ \sum_{k\in\mathbb{A}\setminus\{A\}} \big(\tfrac{\partial}{\partial k}\rho_a(C^\theta_{V^\theta}\Delta')\big)\\
&\big(\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}U_k^\theta - \tfrac{\partial^2}{\partial\theta_i\partial\theta_j}U_A^\theta + \beta(F_k - F_A)\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}V^\theta\big).
\end{aligned}$$

Accordingly, applying a Newton step to the NFXP likelihood function has traditionally required solving the $t + t(t+1)/2$ fixed points defined in (7) and (8). Rust's (2000, p. 21) algorithm avoids such Newton steps, since

[5] When calculating this derivative, it is useful to note that $\frac{\partial}{\partial\theta}V^\theta = \frac{\partial}{\partial\theta}V_P^\theta\big|_{P=P^\theta}$, which follows from the envelope theorem.

Given that the likelihood function $L(\theta) = L(\theta, EV_\theta)$ depends on the expected value function $EV_\theta$ (see formula (2.13) in chapter 2), it is easy to see that use of Newton's method to determine the direction vector in (3.7) would require computing the second derivative of the expected value function, $\partial^2 EV_\theta / \partial\theta\partial\theta'$, which is quite difficult to compute.

However, Rust's algorithm does solve the $t$ fixed points in (7) for each gradient ascent step.

To avoid solving a full dynamic program for each likelihood evaluation, Hotz and Miller (1993, p. 498) developed the conditional choice probability (CCP) estimator, which "does not require econometricians to explicitly solve the valuation functions used to characterize optimal decision rules via backwards recursion methods." There are now many CCP estimators, but essentially all of them solve some version of the following problem: [6]

$$\hat{\theta}_{\text{CCP}} \equiv \arg\max_{\theta\in\Theta} \mathcal{L}^\theta_{\text{CCP}}$$
$$\text{where} \quad \mathcal{L}^\theta_{\text{CCP}} \equiv \sum_{a\in\mathbb{A}} n'_a \log(P^\theta_{\hat{P}a}).$$

It is generally understood that CCP estimators do not suffer from nested fixed points. This belief

[6] Although often expressed in slightly different forms, equation (5) is a core component of all general-purpose, infinite-horizon CCP estimators. For example, it corresponds to Aguirregabiria and Mira's (2002) equation (8), Aguirregabiria and Mira's (2007) equation (14), Pakes et al.'s (2007) equation (9), Pesendorfer and Schmidt-Dengler's (2008) equation (6), Bajari et al.'s (2007) first equation on page 1343, Hotz et al.'s (1994) equation (4.2), and to the infinite-horizon analog of Hotz and Miller's (1993) equation (3.12). Aguirregabiria and Mira (2002, 2007), Pakes et al. (2007) and Pesendorfer and Schmidt-Dengler (2008) re-express fixed point (5) in terms of $(I - \beta F_{\hat{P}})^{-1}$. But for large problems, it would generally be faster to repeatedly solve (5) than explicitly compute $(I - \beta F_{\hat{P}})^{-1}$. Bajari et al. (2007) and Hotz et al. (1994) use forward simulation to approximate the equivalent sequential representation, $V^\theta_P = \sum_{t=0}^\infty \beta^t F^t_P U^\theta_P$. Finally, Hotz and Miller (1993) works with the finite-horizon analog of (5). To see this with their notation, note that

$$\begin{aligned} v_j(H_t) &= \sum_{s=t+1}^{T} \sum_{H_s\in\mathscr{A}_s(H_t)} U(p_s, H_s) F_j(H_{t+1}|H_t) \prod_{r=t+1}^{s-1} \Big(\sum_{k=1}^{J} p_{rk} F_k(H_{r+1}|H_r)\Big) \qquad (3.12) \\ &= \sum_{H_{t+1}\in\mathscr{A}_{t+1}(H_t)} F_j(H_{t+1}|H_t)\Big[U(p_{t+1}, H_{t+1}) + \sum_{k=1}^{J} p_{(t+1)k} v_k(H_{t+1})\Big]. \end{aligned}$$

Denote the expected continuation value given $H_t$ with $V(H_t) \equiv \sum_{k=1}^J p_{tk} v_k(H_t)$, and the policy induced transition with $F_{pt}(H_{t+1}|H_t) \equiv \sum_{j=1}^J p_{tj} F_j(H_{t+1}|H_t)$. Then

$$V(H_t) = \sum_{H_{t+1}\in\mathscr{A}_{t+1}(H_t)} F_{pt}(H_{t+1}|H_t)\Big[U(p_{t+1}, H_{t+1}) + V(H_{t+1})\Big].$$

Index feasible histories at time $t$ as as $\{H^1_t, \ldots, H^{n_t}_t\}$. Let $V_t \in \mathbb{R}^{n_t}$ have $i$th entry $V(H^i_t)$, and $U_{t+1} \in \mathbb{R}^{n_{t+1}}$ have $j$th entry $U(p_{t+1}, H^j_{t+1})$. Define $F_{pt}$ as the $n_t \times n_{t+1}$ matrix with $ij$th entry $F_{pt}(H^j_{t+1}|H^i_t)$. Then $V_t = F_{pt}U_{t+1} + F_{pt}V_{t+1}$. In a stationary environment with generic states, including geometric discounting $\beta \in [0, 1)$, letting $T \to \infty$ gives the infinite–horizon analog: $V = \beta F_p U + \beta F_p V$.

is incorrect. For example, evaluating the CCP likelihood requires evaluating $P^{\theta}_{\hat{P}}$, which in turn requires solving the fixed point equation (5), with $P = \hat{P}$. However, the CCP fixed point (5) is simpler than the NFXP fixed point (6): the former solves the value function that corresponds to policy $\hat{P}$, which is pre-estimated, whereas the latter solves the value function that corresponds to policy $P^{\theta}$, which is computed. Put differently, evaluating the CCP likelihood function requires implementing a single policy-iteration step, whereas solving the NFXP likelihood requires iterating policy-iteration steps to convergence.

The CCP likelihood function also has similar gradient and Hessian nested fixed points. Specifically, $\frac{\partial}{\partial\theta}\mathcal{L}^{\theta}_{\text{CCP}}$ and $\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^{\theta}_{\text{CCP}}$ depend on the solutions to

$$\frac{\partial}{\partial\theta_i}V^{\theta}_{\hat{P}} = \sum_{a\in\mathbb{A}} \text{diag}(\hat{P}_a)\frac{\partial}{\partial\theta_i}U^{\theta}_a + \beta F_{\hat{P}}\frac{\partial}{\partial\theta_i}V^{\theta}_{\hat{P}} \tag{9}$$

$$\text{and} \quad \frac{\partial^2}{\partial\theta_i\partial\theta_j}V^{\theta}_{\hat{P}} = \sum_{a\in\mathbb{A}} \left(\text{diag}(\hat{P}_a)\frac{\partial^2}{\partial\theta_i\partial\theta_j}U^{\theta}_a\right) + \beta F_{\hat{P}}\frac{\partial^2}{\partial\theta_i\partial\theta_j}V^{\theta}_{\hat{P}}, \tag{10}$$

as $\frac{\partial}{\partial\theta}\mathcal{L}^{\theta}_{\text{NFXP}}$ and $\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^{\theta}_{\text{NFXP}}$ depend on the solutions to (7) and (8). In fact, (9) and (10) are no easier to solve than (7) and (8), conditional on a fixed policy. Thus, essentially all of the computational gains from the CCP estimator stem from avoiding the repeated computation of $P^{\theta}$.

Bray's (2019b) strong convergence (SC) estimator is the same as the CCP estimator, except it replaces nested fixed point (5) with the following nested fixed point:

$$\Psi V^{\theta}_{\hat{P}} \equiv \Psi U^{\theta}_{\hat{P}} + \beta\Psi F_{\hat{P}}\Psi V^{\theta}_{\hat{P}}, \tag{11}$$

where $\Psi$ is the matrix that maps $(v_1, \cdots, v_X)$ to $(v_1 - \bar{v}, \cdots, v_X - \bar{v})$, for $\bar{v} \equiv \sum_{i=1}^{X} v_i/X$. Specifically,

$$\begin{aligned} \hat{\theta}_{\text{SC}} &\equiv \arg\max_{\theta\in\Theta} \mathcal{L}^{\theta}_{\text{SC}} \\ \text{where} \quad \mathcal{L}^{\theta}_{\text{SC}} &\equiv \sum_{a\in\mathbb{A}} n'_a \log(P^{\theta}_{\Psi V^{\theta}_{\hat{P}} a}). \end{aligned}$$

However, it is simple to show that $P^{\theta}_{\Psi V} = P^{\theta}_{V}$, which implies that $\mathcal{L}^{\theta}_{\text{SC}} = \mathcal{L}^{\theta}_{\text{CCP}}$. Thus, SC is simply a different implementation of CCP. But this implementation will be faster when the Markov chain under policy $\hat{P}$ is ergodic, in which case successive approximations of (11) will converge strictly faster than successive approximations of (5), a property known as *strong convergence* (see Bray, 2019a).

We can leverage strong convergence for computing gradients and Hessians as well, since $\frac{\partial}{\partial\theta}\mathcal{L}^{\theta}_{\text{SC}}$

and $\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^\theta_{\text{SC}}$ depend on the solutions to

$$\begin{aligned}
\Psi\tfrac{\partial}{\partial\theta_i}V^\theta_{\hat{P}} &= \sum_{a\in\mathbb{A}}\Psi\text{diag}(\hat{P}_a)\tfrac{\partial}{\partial\theta_i}U^\theta_a + \beta\Psi F_{\hat{P}}\Psi\tfrac{\partial}{\partial\theta_i}V^\theta_{\hat{P}} \\
\text{and}\quad \Psi\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}V^\theta_{\hat{P}} &= \sum_{a\in\mathbb{A}}\Psi\Big(\text{diag}(\hat{P}_a)\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}U^\theta_a\Big) + \beta\Psi F_{\hat{P}}\Psi\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}V^\theta_{\hat{P}},
\end{aligned}$$

which converge faster than the solution to (9) and (10) when the state variables follow an ergodic Markov chain.

So far, we have made only incremental progress: the SC fixed points are easier to solve than the CCP fixed points, which in turn are easier to solve than the NFXP fixed points. However, a real breakthrough arises in one special case: when the utility function is linear. If $U^\theta_a = W_a\theta$, for some $X \times t$ matrix $W_a$, then the CCP and SC fixed points unnest, and these estimators become substantially more efficient (Aguirregabiria, 1999, p. 296; Aguirregabiria and Mira, 2010, p. 52; Bray, 2019b, p. 53). For the CCP case, note that $U^\theta_a = W_a\theta$ implies

$$\begin{aligned}
V^\theta_{\hat{P}} &= V^E_{\hat{P}} + V^W_{\hat{P}}\theta, \\
\tfrac{\partial}{\partial\theta}V^\theta_{\hat{P}} &= V^W_{\hat{P}}, \\
\text{and}\quad \tfrac{\partial^2}{\partial\theta\partial\theta}V^\theta_{\hat{P}} &= 0, \\
\text{where}\quad V^E_{\hat{P}} &\equiv E(\hat{P}) + \beta F_{\hat{P}}V^E_{\hat{P}} \\
\text{and}\quad V^W_{\hat{P}} &\equiv \sum_{a\in\mathbb{A}}\text{diag}(\hat{P}_a)W_a + \beta F_{\hat{P}}V^W_{\hat{P}}.
\end{aligned}$$

And for the SC case, we apply $\Psi$ where appropriate. As you see, there are still fixed points in the system above, but these fixed points do not depend on $\theta$. Hence, we could solve $V^E_{\hat{P}}$ and $V^W_{\hat{P}}$ upfront, after which $V^\theta_{\hat{P}}$, $\frac{\partial}{\partial\theta}V^\theta_{\hat{P}}$, and $\frac{\partial^2}{\partial\theta\partial\theta}V^\theta_{\hat{P}}$ are available in closed form. Hence, after solving the two fixed point equations above, we could evaluate $\mathcal{L}^\theta_{\text{CCP}}$, $\frac{\partial}{\partial\theta}\mathcal{L}^\theta_{\text{CCP}}$, and $\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^\theta_{\text{CCP}}$ for any number of $\theta$ values, with negligible marginal effort. Computing $V^E_{\hat{P}}$ and $V^W_{\hat{P}}$ may be costly in large problems, but once obtained, maximizing the CCP likelihood is trivial.[7]

Our contribution will be to extend this fixed point unnesting technique to dynamic programs with non-linear utility functions.

[7] Another way to unnest the CCP fixed points would be to compute $(I-\beta F_{\hat{P}})^{-1}$ upfront. But this matrix inversion is not practical in general, as it requires $O(X^3)$ operations with standard algorithms (or $O(X^{2.373})$ operations with esoteric algorithms), whereas solving for $V^\theta_{\hat{P}}$ requires only $O(X^2)$ operations (Judd, 1998, p. 77).

### 3.2 Dual Representation

The following proposition will underpin our estimation approach.

**Proposition 2.** *Computing $w'V_P^\theta$ for one weighting vector $w \in \mathbb{R}^X$ and many parameter vectors $\theta \in \Theta$ only requires solving a* single *fixed point equation, since*

$$w'V_P^\theta = \lambda' U_P^\theta, \tag{12}$$

$$\textit{for} \quad \lambda \equiv w + \beta F_P' \lambda. \tag{13}$$

*Proof.* We can express $w'V_P^\theta$ as the objective of the following linear program:

$$\begin{aligned} \max_{V \in \mathbb{R}^X} \quad & w'V \\ \text{s.t.} \quad & V = U_P^\theta + \beta F_P V. \end{aligned}$$

The strong duality theorem asserts that this program has the same objective as the following dual linear program:

$$\begin{aligned} \min_{\lambda \in \mathbb{R}^X} \quad & \lambda' U_P^\theta \\ \text{s.t.} \quad & \lambda = w + \beta F_P' \lambda. \end{aligned}$$

□

We will use this proposition to create estimators whose fixed points unnest for all utility functions in the next section. But before that, we will illustrate the value of Proposition 2 by showing how it accelerates the computation of $\frac{\partial}{\partial \theta_i}\mathcal{L}_{\text{NFXP}}^\theta$, $\frac{\partial^2}{\partial \theta_i \partial \theta_j}\mathcal{L}_{\text{NFXP}}^\theta$, $\frac{\partial}{\partial \theta_i}\mathcal{L}_{\text{CCP}}^\theta$, and $\frac{\partial^2}{\partial \theta_i \partial \theta_j}\mathcal{L}_{\text{SC}}^\theta$.

To begin, consider a dynamic program with transition function $\tilde{F}_P \equiv F_P$ and utility function $\tilde{U}_P^\theta \equiv \sum_{a \in \mathbb{A}} \text{diag}(P_a) \frac{\partial}{\partial \theta_i} U_a^\theta$. Line (7) establishes that restricting this dynamic program to policy $\tilde{P} \equiv P^\theta$ yields value function $\tilde{V}_{\tilde{P}} \equiv \frac{\partial}{\partial \theta_i} V^\theta$. Next, define $\lambda$ as the solution to (13) under policy $\tilde{P}$ and weighting vector

$$w' \equiv \sum_{a \in \mathbb{A}} n_a' \text{diag}(P_a^\theta)^{-1} \sum_{j \in \mathbb{A} \setminus \{A\}} \left(\tfrac{\partial}{\partial j} \rho_a(C_{V^\theta} \Delta')\right) \beta (F_j - F_A).$$

In this case, line (12) establishes that

$$w' \tfrac{\partial}{\partial \theta_i} V^\theta = w' \tilde{V}_{\tilde{P}} = \lambda' \tilde{U}_{\tilde{P}} = \lambda' \sum_{a \in \mathbb{A}} \text{diag}(P_a^\theta) \tfrac{\partial}{\partial \theta_i} U_a^\theta.$$

And, with this, we can express the NFXP likelihood gradient in a way that depends on the single fixed point in (13), rather than $t$ fixed points in (7):

$$\begin{aligned} \tfrac{\partial}{\partial \theta_i} \mathcal{L}^\theta_{\text{NFXP}} =& \lambda' \sum_{a \in \mathbb{A}} \text{diag}(P_a^\theta) \tfrac{\partial}{\partial \theta_i} U_a^\theta \\ &+ \sum_{a \in \mathbb{A}} n_a' \text{diag}(P_a^\theta)^{-1} \sum_{j \in \mathbb{A} \setminus \{A\}} \big( \tfrac{\partial}{\partial j} \rho_a(C^\theta_{V^\theta} \Delta') \big) ( \tfrac{\partial}{\partial \theta_i} U_j^\theta - \tfrac{\partial}{\partial \theta_i} U_A^\theta). \end{aligned}$$

We can speed up the computation of $\tfrac{\partial^2}{\partial \theta_i \partial \theta_j} \mathcal{L}^\theta_{\text{NFXP}}$ with the same trick. For this case, consider a dynamic program with state transition matrix $\breve{F}_P \equiv F_P$ and utility function

$$\breve{U}^\theta_P \equiv \sum_{a \in \mathbb{A}} \text{diag}(P_a) \tfrac{\partial^2}{\partial \theta_i \partial \theta_j} U_a^\theta + \text{diag}(\tfrac{\partial}{\partial \theta_j} P_a^\theta) \tfrac{\partial}{\partial \theta_i} U_a^\theta + \beta \text{diag}(\tfrac{\partial}{\partial \theta_j} P_a^\theta) F_a \tfrac{\partial}{\partial \theta_i} V^\theta.$$

Line (8) establishes that restricting this dynamic program to policy $\breve{P} \equiv P^\theta$ yields value function $\breve{V}_{\breve{P}} \equiv \tfrac{\partial^2}{\partial \theta_i \partial \theta_j} V^\theta$. Thus, for $w$ and $\lambda$ defined above, (12) establishes that

$$\begin{aligned} w' \tfrac{\partial^2}{\partial \theta_i \partial \theta_j} V^\theta &= w' \breve{V}_{\breve{P}} \\ &= \lambda' \breve{U}_{\breve{P}} \\ &= \lambda' \sum_{a \in \mathbb{A}} \Big( \text{diag}(P_a^\theta) \tfrac{\partial^2}{\partial \theta_i \partial \theta_j} U_a^\theta + \text{diag}(\tfrac{\partial}{\partial \theta_j} P_a^\theta) \tfrac{\partial}{\partial \theta_i} U_a^\theta + \beta \text{diag}(\tfrac{\partial}{\partial \theta_j} P_a^\theta) F_a \tfrac{\partial}{\partial \theta_i} V^\theta \Big). \end{aligned}$$

And, with this, we can express the NFXP likelihood Hessian in a way that depends on the one

fixed point in (13) and the $t$ fixed points in (7), rather than the $t(t+1)/2$ fixed points in (8):

$$\begin{aligned}
\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^\theta_{\text{NFXP}} =& \lambda' \sum_{a\in\mathbb{A}} \Big(\text{diag}(P^\theta_a)\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}U^\theta_a + \text{diag}(\tfrac{\partial}{\partial\theta_j}P^\theta_a)\tfrac{\partial}{\partial\theta_i}U^\theta_a + \beta\text{diag}(\tfrac{\partial}{\partial\theta_j}P^\theta_a)F_a\tfrac{\partial}{\partial\theta_i}V^\theta\Big) \\
&- \sum_{a\in\mathbb{A}} n'_a \text{diag}(P^\theta_a)^{-2}\text{diag}(\tfrac{\partial}{\partial\theta_j}P^\theta_a)\tfrac{\partial}{\partial\theta_i}P^\theta_a \\
&+ \sum_{a\in\mathbb{A}} n'_a \text{diag}(P^\theta_a)^{-1}\Big(\sum_{\ell\in\mathbb{A}\setminus\{A\}}\sum_{k\in\mathbb{A}\setminus\{A\}}\big(\tfrac{\partial^2}{\partial\ell\partial k}\rho_a(C^\theta_{V^\theta}\Delta')\big) \\
&\text{diag}\big(\tfrac{\partial}{\partial\theta_j}U^\theta_\ell - \tfrac{\partial}{\partial\theta_j}U^\theta_A + \beta(F_\ell - F_A)\tfrac{\partial}{\partial\theta_j}V^\theta\big) \\
&\big(\tfrac{\partial}{\partial\theta_i}U^\theta_k - \tfrac{\partial}{\partial\theta_i}U^\theta_A + \beta(F_k - F_A)\tfrac{\partial}{\partial\theta_i}V^\theta\big) \\
&+ \sum_{k\in\mathbb{A}\setminus\{A\}}\big(\tfrac{\partial}{\partial k}\rho_a(C^\theta_{V^\theta}\Delta')\big)\big(\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}U^\theta_k - \tfrac{\partial^2}{\partial\theta_i\partial\theta_j}U^\theta_A\big)\Big).
\end{aligned}$$

The $\frac{\partial}{\partial\theta_i}\mathcal{L}^\theta_{\text{CCP}}$, $\frac{\partial}{\partial\theta_i}\mathcal{L}^\theta_{\text{SC}}$, $\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^\theta_{\text{CCP}}$, and $\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}^\theta_{\text{SC}}$ simplifications are analogous: Proposition 2 decreases the number of fixed points required to evaluate the gradients from $t$ to 1, and decreases the number of fixed points required to evaluate the Hessians from $t(t+1)/2$ to $t+1$. These reductions are meaningful when the utility function is parameterized by the weights of a neural network, in which case $t$ could be in the hundreds or thousands.

### 3.3 Unnested Fixed Point (UFXP) Estimator

Proposition 2 reduces the number of nested fixed points in NFXP and CCP, but it does not eliminate them. For example, the $\lambda_i$ used to streamline the computation of $\frac{\partial}{\partial\theta_i}\mathcal{L}^\theta_{\text{NFXP}}$ depends on $\theta$, and therefore must be recomputed at each gradient ascent step. The difficulty is that NFXP, CCP, and SC do not naturally align with Proposition 2, since their objective functions involve many distinct linear combinations of the value function. For example, the CCP likelihood depends on $(F_a(x) - F_A(x))V^\theta_{\hat{P}}$ for every $x \in \mathbb{X}$ and every $a \in \mathbb{A}\setminus\{A\}$, where $F_a(x)$ denotes the $x$th row of $F_a$. Consequently, unnesting the CCP fixed points via Proposition 2 would require solving (13) for $X(A-1)$ different weighting vectors $w$.

Rather than fit Proposition 2 to pre-existing estimators, as we did in the previous section, we will now fit an estimator to Proposition 2. Whereas NFXP and CCP operate in the CCP space—depending on $P^\theta$ or $P^\theta_{\hat{P}}$—our estimator operates in the CSV space—depending on $C^\theta_{V^\theta_{\hat{P}}}$. Put differently, NFXP and CCP exploit the optimal policy, line (4), while we exploit the associated first-order conditions, line (3). This shift is crucial: the optimal policy is nonlinear in the value function, whereas the first-order conditions are linear in the value function. This linearity will enable

us to construct an empirical objective that depends on only a small number of linear combinations of the value function.

The first-order conditions in (3) imply that

$$\rho^{-1}(\hat{P}) - C^{\theta}_{V^{\theta}_{\hat{P}}}\Delta' \to 0 \quad \text{as} \quad \hat{P} \to P^{\theta}.$$

Our unnested fixed point (UFXP) estimator minimizes the sum of squares of $m > t$ random linear functionals of these limit conditions (although $m = t + 1$ suffices, we usually set $m$ slightly larger):

$$\begin{aligned} \hat{\theta}_{\text{UFXP}} &\equiv \arg\min_{\theta \in \Theta} \mathcal{Q}^{\theta}_{Z} \\ \text{where} \quad \mathcal{Q}^{\theta}_{Z} &\equiv \sum_{i=1}^{m} \text{tr}(Z_i'(\rho^{-1}(\hat{P}) - C^{\theta}_{V^{\theta}_{\hat{P}}}\Delta'))^2 \end{aligned}$$

and $Z_i$ is an $X \times (A-1)$ matrix comprised of independent standard normal random variables, for $i \in \{1, \cdots, m\}$.

Objective $\mathcal{Q}^{\theta}_{Z}$ fits seamlessly with Proposition 2. To see this, note that

$$\begin{aligned} &\text{tr}(Z_i'(\rho^{-1}(\hat{P}) - C^{\theta}_{V^{\theta}_{\hat{P}}}\Delta')) \\ &\quad = \sum_{a \in \mathbb{A}\setminus\{A\}} Z_{ia}'\big(\rho_a^{-1}(\hat{P}) - U^{\theta}_{a} + U^{\theta}_{A} - \beta(F_a - F_A)V^{\theta}_{\hat{P}}\big) \\ &\quad = w_i' V^{\theta}_{\hat{P}} + \sum_{a \in \mathbb{A}\setminus\{A\}} Z_{ia}'(\rho_a^{-1}(\hat{P}) - U^{\theta}_{a} + U^{\theta}_{A}) \\ &\quad = \lambda_i' U^{\theta}_{\hat{P}} + \sum_{a \in \mathbb{A}\setminus\{A\}} Z_{ia}'(\rho_a^{-1}(\hat{P}) - U^{\theta}_{a} + U^{\theta}_{A}), \\ \text{where} \quad w_i' &\equiv - \sum_{a \in \mathbb{A}\setminus\{A\}} \beta Z_{ia}'(F_a - F_A), \end{aligned} \tag{14}$$

$\lambda_i$ is the corresponding fixed point of (13), and $Z_{ia}$ is the $a$th column of $Z_i$. The key, of course, is replacing $w_i' V^{\theta}_{\hat{P}}$ with $\lambda_i' U^{\theta}_{\hat{P}}$, as doing so exchanges $\theta$-dependent fixed point (5) for $\theta$-independent fixed point (13). Accordingly, after solving $\lambda_1, \cdots, \lambda_m$, we can evaluate our estimator's objective for any number of $\theta$ values, with negligible marginal cost. These $m$ dual-variable vectors also yield the objective's gradient and Hessian matrix, which also do not require any fixed point computations to evaluate (see Appendix Section A for the explicit expressions of $\frac{\partial}{\partial \theta}\mathcal{Q}^{\theta}_{Z}$ and $\frac{\partial^2}{\partial \theta_j \partial \theta_k}\mathcal{Q}^{\theta}_{\text{Z}}$).

The following theorem establishes the consistency and asymptotic normality of our UFXP estimator.

**Theorem 1.** *Define $N(x) \equiv \sum_{a \in \mathbb{A}} n_a(x)$ as the number of state-x observations, $N \equiv \sum_{x \in \mathbb{X}} N(x)$ as the total number of observations, and $\eta(x) \equiv \lim_{N \to \infty} N(x)/N$ as the limiting proportion of state-x observations. Suppose the following conditions hold:*

*Compactness: The true parameter $\theta$ is an interior point of the compact space $\Theta \subset \mathbb{R}^t$.*

*Smoothness: The utility function, $u^\theta$, is continuously differentiable in $\theta$ over $\Theta$.*

*Error-distribution regularity: The additive error $e \in \mathbb{R}^A$ has a strictly positive and continuously differentiable joint density $g$ on $\mathbb{R}^A$ with integrable first moment and gradient.*

*Identification: The map $\theta \mapsto C^\theta_{V^\theta_{P^\theta}} \Delta'$ is injective.*

*Full-rank: The Jacobian $\frac{\partial}{\partial \theta} \text{vec}(C^\theta_{V^\theta_P} \Delta') \Big|_{P=P^\theta}$ has full rank $t$ over $\Theta$.*

*Random projection: $Z_1, \ldots, Z_m$ (where $m > t$) are a collection of $X \times (A-1)$ matrices whose entries are i.i.d. draws from a distribution that is absolutely continuous with respect to Lebesgue measure. For each state $x \in \mathbb{X}$, let $z(x)$ be the $(A-1) \times m$ matrix where the $i$-th column is the transposed $x$-th row of $Z_i$.*

*In this case, with probability one over the distribution of $\{Z_i\}_{i=1}^m$, we have the following as $N \to \infty$*

1. *$\hat{\theta}_{UFXP} \xrightarrow{P} \theta$ and*
2. *$\sqrt{N}(\hat{\theta}_{UFXP} - \theta) \xrightarrow{d} \mathcal{N}(0, \Sigma_{UFXP})$,*

*where*

$$\begin{aligned}
\Sigma_{UFXP} &\equiv \Big((D^\theta)' D^\theta\Big)^{-1} (D^\theta)' \left( \sum_{x \in \mathbb{X}} \frac{z(x)' \Gamma(x) \Sigma(x) \Gamma(x)' z(x)}{\eta(x)} \right) D^\theta \Big((D^\theta)' D^\theta\Big)^{-1}, \\
D^\theta &\equiv \sum_{x \in \mathbb{X}} z(x)' \Big( \Delta \tfrac{\partial}{\partial \theta} u^\theta(x) + \beta \sum_{x' \in \mathbb{X}} \Delta f(x'|x) \tfrac{\partial}{\partial \theta} v^\theta_{P^\theta}(x') \Big), \\
\Sigma(x) &\equiv \text{diag}(P^\theta(x)) - P^\theta(x) P^\theta(x)', \\
\Gamma(x) &\equiv \tfrac{\partial}{\partial P(x)} \rho^{-1}(P^\theta(x)).
\end{aligned}$$

### 3.4 Optimal Unnested Fixed Point (OUFXP) Estimator

Cameron and Trivedi's (2005) equation (6.24) suggests that we can improve our estimator's asymptotic performance by replacing random weights $z$ with the following "optimal" weights:

$$z^\theta(x) \equiv \Big( \frac{\hat{\Gamma}(x) \hat{\Sigma}(x) \hat{\Gamma}(x)'}{N(x)/N} \Big)^{-1} \Big( \Delta \tfrac{\partial}{\partial \theta} u^\theta(x) + \beta \sum_{x' \in \mathbb{X}} \Delta f(x'|x) \tfrac{\partial}{\partial \theta} v^\theta_{\hat{P}}(x') \Big),$$

where $\hat{\Sigma}(x)$ and $\hat{\Gamma}(x)$ are analogs to $\Sigma(x)$ and $\Gamma(x)$ with $\hat{P}(x)$ replacing $P^\theta(x)$.

Unfortunately, the optimal weights depend on $\theta$, so using them requires a two-step estimator. In the first step, we calculate $\hat{\theta}_{\text{UFXP}}$, and in the second step we calculate

$$\hat{\theta}_{\text{OUFXP}} \equiv \arg\min_{\theta\in\Theta} \mathcal{Q}^{\theta}_{Z^{\hat{\theta}_{\text{UFXP}}}}.$$

Note, if the utility function is linearly parameterized ($U^{\theta}_{a} = W_a\theta$) then the OUFXP estimator simplifies to:

$$\hat{\theta}_{\text{OUFXP}} \equiv \Big(\sum_{x\in\mathbb{X}} z^{\hat{\theta}_{\text{UFXP}}}(x)'\Delta W(x) - \Lambda\sum_{a\in\mathbb{A}}\text{diag}(\hat{P}_a)W_a\Big)^{-1}\Big(\sum_{x\in\mathbb{X}} z^{\hat{\theta}_{\text{UFXP}}}(x)'\rho^{-1}(\hat{P}(x)) + \Lambda E(\hat{P})\Big),$$

where $\Lambda$ is the $t \times X$ matrix whose $i$th row is $\lambda_i$, and $W(x)$ is the $A \times t$ matrix whose $a$th row corresponds to the $x$th row of $W_a$.[8]

The following result justifies the name of our new estimator.

**Theorem 2.** *The optimal unnested fixed point estimator (OUFXP) is as asymptotically efficient as maximum likelihood.*

When interpreting Theorem 2, keep in mind that we have treated the model's state transition matrices as given. Thus, this theorem indicates that OUFXP is as asymptotically efficient as any other dynamic discrete choice estimator that first pre-estimates state transition probabilities (which is essentially all of them).

# 4 Numerical Experiments

We will compare our estimators to other prominent dynamic discrete choice estimators with a simplified version of the structural econometric model of Section 5. Our estimators will tackle this toy model with ease, but traditional estimators will struggle. The difficulty traditional estimators face with this simplified model suggests that the more elaborate version we present in Section 5 would be effectively unestimatable without our approach.

Our model concerns a grocery store's infinite-horizon inventory management problem. The key object of interest is the inventory holding cost. Most stylized inventory models—such as the newsvendor, linear-quadratic, and $(s, S)$ specifications—capture inventory costs with one or two parameters. But with over a million observations in our actual empirical sample (which we

[8] See Miessi Sanches et al. (2016) and Dearing (2019) for other closed-form estimators of dynamic programs with linearly parameterized utility functions.

will introduce in the next section), we can afford to estimate the inventory holding cost function nonparametrically.

The store's holding cost function depends on three state variables: $r \in \{1, 2, 3, 4, 5, 6\}$ indexes the expectation of the current daily demand, $o \in \{1, 2, 3\}$ indexes the amount of other-product inventory held, and $i \in \{0, \cdots, 29\}$ records the number of units of the focal product held. The holding cost combines a baseline linear cost, a step penalty, and a weighted combination of concave and convex tails (see Appendix Section D for further details).

Figure 1 depicts holding cost $h(r, o, i)$ (and our estimates of it, which we will discuss later). To be clear, this function is ad hoc: there is no economic rationale for this particular functional form. However, we believe this function represents roughly the correct level of complexity, as many operational factors could kink and twist a store's holding cost function: e.g., the focal-product holding cost could increase with the other-product inventory level, due to crowding in the backroom; the holding cost could increase with the demand level, as more popular products are allocated more valuable real estate in the back room; and the holding cost could be discontinuous, as overflow inventory is stored elsewhere, once the designated shelf is full.

Importantly, the holding cost function is not only irregular, but is *irregular in three dimensions*. This makes holding costs difficult to capture with a linear basis (e.g., splines perform poorly in higher dimensions). We believe most true, organic utilities would exhibit such multi-dimensional idiosyncrasies—a belief that motivates our embrace of neural networks.

The holding cost is just one part in a four-part utility function:

$$u_q(r, o, i) + e_q \equiv -h(r, o, i) - \eta \mathrm{E}((d - i)^+ | r) - \kappa \mathbb{1}\{q > 0\} + e_q.$$

The second term in the expression above is the cost of unfulfilled demand, the third term is the cost of shipping, and the final term is a statistical error term. More specifically, $\eta$ is the loss associated with one unsatisfied customer, $\kappa$ is the cost of receiving one shipment of inventory, $d$ is the daily customer demand, $q$ is the amount of inventory the store orders from its supplier, and $e_q$ is an independent Gumbel shock associated with order quantity $q$. The utility function above is analogous to the one presented in Section 5, which we develop in more detail.

The action variable is order quantity $q \in \mathbb{Q} \equiv \{0, 6, 12, 18\}$, which the store manager chooses with the aim of maximizing expected discounted utility, under discount factor $\beta = 0.9997$ (which corresponds to an annual discount factor of $0.9997^{365} = 0.896$).[9] Finally, our model has the following

[9]Fixing this parameter is not necessary, as we could estimate it with the procedure outlined in Appendix Section

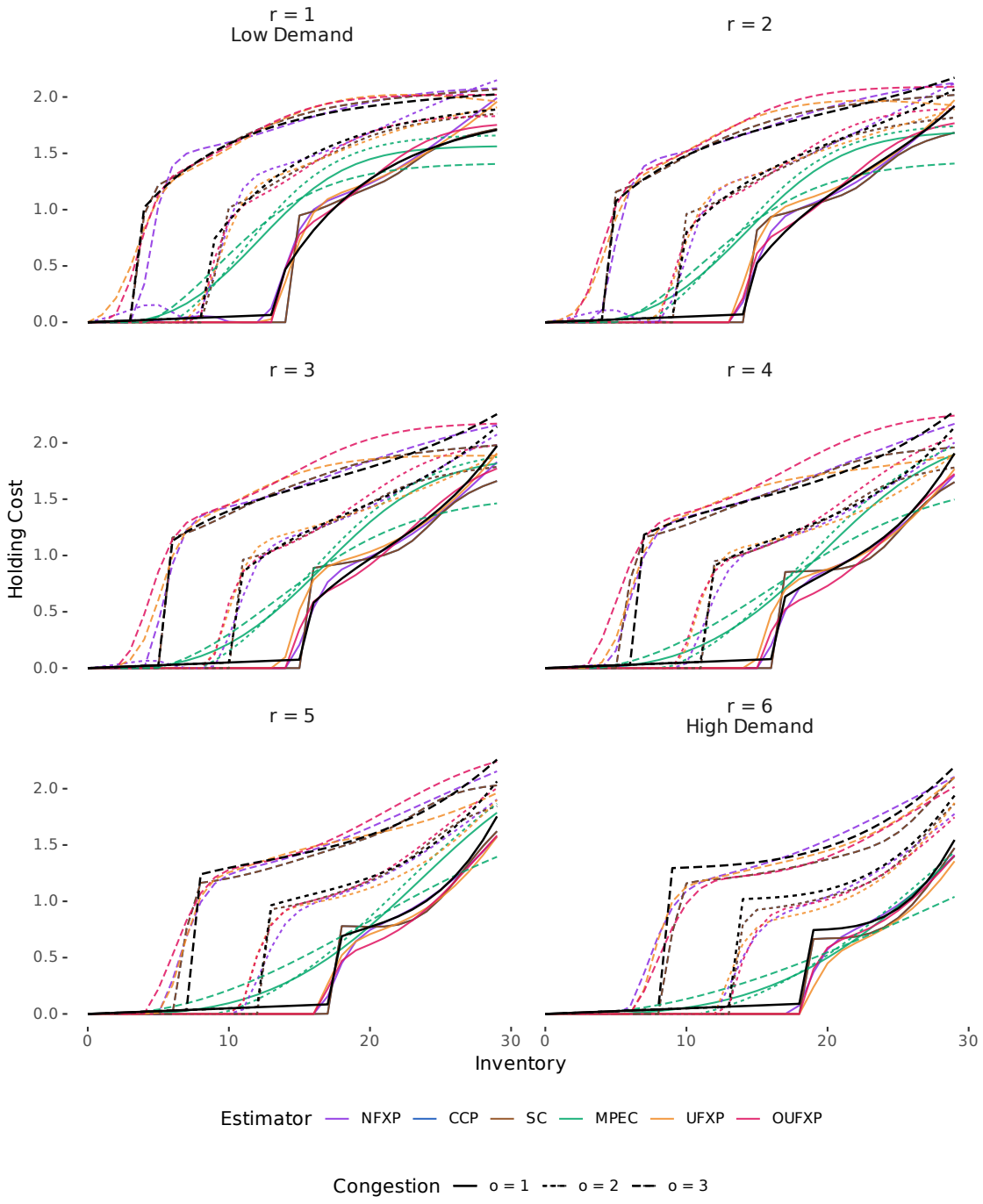



Figure 1: True and estimated holding cost functions for the 540-state model. The black lines depict the true holding cost function, $h(r, o, i)$, defined in Appendix Section D. The colored lines plot the best estimate, $\hat{h}$, produced by each estimator. We define an estimator's best estimate as the minimizer of $\|\hat{h} - h\|_1$ across optimization runs. The six panels represent demand states $r \in \{1, \ldots, 6\}$. Within each panel, line types denote the congestion state $o \in \{1, 2, 3\}$.

transition kernel

$$f_q(r', o', i'|r, o, i) \equiv f^r(r'|r, i) f^o(o'|o) f_q^i(i'|r, i).$$

In addition to $h$, we formally define $f^r$, $f^o$, and $f_q^i$ in Appendix Section D. The model we have described has $6 \cdot 3 \cdot 30 = 540$ distinct states. To gauge how estimation times vary with the size of the state space, we also consider a version of this model with $6 \cdot 3 \cdot 300 = 5400$ states (see Appendix Section D).

### 4.1 Estimation Details

For a given version of our model, we begin by solving the agent's optimal policy, and using that to simulate a sample with 1,073,679 observations (the size of the panel we will use in Section 5). Next, we use this random sample to pre-estimate CCPs and transition probabilities (see Appendix Section D for estimation details). And finally, we estimate $\eta$, $\kappa$, and $h$ from our simulated sample, $\hat{P}$, $\hat{f}_q$, and $\beta$.

Note, we treat $h$ as a model primitive, rather than the parameters that define it. We estimate this function with a multi-layer perceptron that receives six features—$r$, $o$, $i$, $ro$, $ri$, $oi$—and returns a corresponding holding cost. Our neural network has one output layer of width 1 and $\gamma_1$ hidden layers of width $\gamma_2$. We try three configurations: a wide architecture with $\gamma = (2, 6)$, a balanced architecture with $\gamma = (4, 4)$, and a deep architecture with $\gamma = (6, 3)$. The wide network comprises 91 parameters (78 weights and 13 biases), the balanced network comprises 93 parameters (76 weights and 17 biases), and the deep network comprises 85 parameters (66 weights and 19 biases). For each architecture, we consider two possible activation functions, ReLU and softplus, for a total of six neural network specifications. Finally, we apply an output transformation that enforces nonnegativity and makes the function evaluate to zero when $i = 0$. More specifically, we set $h(r, o, i) = \text{ReLU}(n(r, o, i) - n(r, o, 0))$, where $n(r, o, i)$ is the output of the neural network.

Overall, specifying the neural network entails several design choices: the architecture, the activation function, the set of inputs (e.g., interactions $ro$, $ri$, and $oi$ rather than quadratics $r^2$, $o^2$, and $i^2$), and the final transformation function (e.g., the imposition of economic restrictions $h(r, o, 0) = 0$ and $h(r, o, i) \geq 0$). Neural networks are highly customizable.

We estimate $\eta$, $\kappa$, and $h$ with the UFXP, OUFXP, NFXP, CCP, and SC estimators presented in Section 3, as well as the mathematical program with equilibrium constraints (MPEC) estimator

B)

of Su and Judd's (2012).[10] NFXP, CCP, SC, and MPEC are general-purpose estimators, capable of estimating any model that conforms to the framework presented in Section 2.2. These four are therefore natural benchmarks for our estimators, which are also general-purpose. We do not compare UFXP and OUFXP to specialized estimators that leverage structural properties of the dynamic program.[11] The model we estimate does not exhibit any exploitable special structure (while common in single-dimensional problems, such structure is rare in higher dimensions).

We maximize $\mathcal{L}^{\theta}_{\text{NFXP}}$, $\mathcal{L}^{\theta}_{\text{CCP}}$, $\mathcal{L}^{\theta}_{\text{SC}}$, $-\mathcal{Q}^{\theta}_{Z}$, and $-\mathcal{Q}^{\theta}_{Z^{\hat{\theta}_{\text{UFXP}}}}$ with four optimizers: Knitro Interior-Point Direct barrier method (Byrd et al., 2006), Adam (Adaptive Moment Estimation) adaptive gradient descent method (Kingma and Ba, 2017), L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) quasi-Newton method (Liu and Nocedal, 1989), and Trust-region Newton method (Nocedal and Wright, 2006). (Following Su and Judd (2012), we maximize $\mathcal{L}^{\theta,V}_{\text{MPEC}}$ with Knitro only, as the other three optimizers have difficulty with MPEC's non-linear constraint).

We provide all four optimizers with analytical objectives and gradients, and additionally provide Trust and Knitro with analytical Hessians. We compute the NFXP, CCP, and SC gradients and Hessians with the dual fixed point representations facilitated by Proposition 2 (see Section 3.2). Doing so reduces the number of fixed points solved by a factor of at least $t/2 \geq 85/2$. Consequently, we implement these benchmark estimators more efficiently than usual.

Since the estimators' objective functions are not concave, multiple starts will be required to achieve global optimality (or close to it). For all estimations of the 540-state model, we use 100 random starting points. We also use 100 random starting points for the 5400-state model; however,

[10] The MPEC estimator avoids nested fixed points by recasting Bellman's problem from a subroutine required to evaluate the likelihood function into a constraint that needs only to hold at the optimum:

$$
\begin{aligned}
(\hat{\theta}_{\text{MPEC}}, V^{\hat{\theta}_{\text{MPEC}}}) \equiv & \underset{\theta \in \Theta, V \in \mathbb{R}^{\mathcal{X}}}{\arg\max} \; \mathcal{L}^{\theta,V}_{\text{MPEC}} \\
& \text{s.t.} \quad V \quad \text{satisfies} \quad (6), \\
\text{where} \quad \mathcal{L}^{\theta,V}_{\text{MPEC}} \equiv & \sum_{a \in \mathbb{A}} n'_a \log(P^{\theta}_{Va}).
\end{aligned}
$$

Whereas NFXP enforces (6) each time the likelihood is evaluated, MPEC enforces it only for the final parameters. Since it computes only one Bellman equation, this formulation seems easier to solve, but that is not necessarily the case, as Iskhakov et al. (2016) demonstrated.

[11] Examples of such special-purpose estimators are those that leverage *finite dependence*, which a dynamic program exhibits if there exists a policy under which the distribution of the state variables after a set number of transitions is independent of the current action (Hotz et al., 1994; Arcidiacono and Miller, 2011). Another example is the Euler equation approach of Aguirregabiria and Magesan (2013), which replaces the inversion of $I - \beta F_{\hat{P}}$ with local first-order conditions that offer computational advantages only when the transition kernel exhibits strong local structure (such as sparse or low-dimensional action-dependent dynamics) so that the set of states that are reachable within two steps is small. Similarly, the pseudo-value function estimator of Dearing (2019) is computationally advantageous only if some policy $P$ arranges matrix $I - \beta F_P$ in a form that is easy to invert, such as triangular, banded, block-diagonal, or arrow.

since this model is much more difficult to estimate, we limit our focus in this case to the L-BFGS and Trust optimizers, and to the balanced-architecture, softplus-activation network.[12]

Following standard neural network training practices, we draw our initial neural network weights from a Kaiming uniform distribution (He et al., 2015), draw our neural network biases from a uniform distribution with bounds defined by the inverse square root of the number of inputs of each respective layer, and draw $\eta$ and $\kappa$ from a Uniform(0, 10) distribution. Finally, for MPEC, we draw the initial elements of $V$ independently from the standard normal distribution.

For the UFXP estimator's weighting matrices, $Z_1, \ldots, Z_m$, we set $m = 100$. To give more weight to the more precisely estimated elements of $\hat{P}$, we draw the $(x, q)$th element of matrix $Z_i$ from an independent normal distribution with mean 0 and variance $\frac{n_{x,q} n_{x,0}}{n_{x,q}+n_{x,0}}$, where $n_{x,q}$ denotes the number of times action $q$ is observed in state $x$. For the OUFXP estimator, we construct a weighting matrix $Z^{\hat{\theta}_{\text{UFXP}}}$ from the best first-stage UFXP estimate for each neural network configuration. Further, we warm-start each OUFXP optimization from its respective UFXP estimate rather than from random starting points.

We execute each estimation as an independent, single-threaded job allocated to a single CPU core on Northwestern University's Quest HPC cluster, using nodes equipped with Intel Xeon processors. We allocate 4 GB of RAM for the 540-state model and 64 GB for the 5400-state model. To ensure a fair comparison of computational time, we submit the jobs in parallel and execute them in a randomized order across all combinations of estimators, neural network configurations, optimizers, and random seeds for initializations. Finally, the computer cluster subjects every estimation run to a seven-day time limit, from the start of the job.

### 4.2 Results

We execute 12600 optimization runs for the 540-state model: 2400 runs (100 starting values times six neural network configurations times four optimizers) for each of NFXP, CCP, SC, UFXP, and OUFXP, and 600 runs (100 starting values times six neural network configurations times one optimizer) for MPEC. And we execute 1000 runs for the 5400-state model: 200 runs (100 starting values times one neural network configuration times two optimizers) for each of NFXP, CCP, SC, UFXP, and OUFXP. Only UFXP and OUFXP successfully return estimates for all optimization runs across both models. NFXP, CCP, and SC failed to converge within the seven-day time limit for 3.6%, 5.3%, and 0.4% of the 540-state runs, respectively, and for 62%, 60%, and 13% of the

[12] We exclude MPEC from the 5400-state experiments because it is incompatible with these optimizers, and casual experimentation suggests it performs as poorly with the 5400-state model as it does with the 540-state model.

5400-state runs. While all 600 MPEC runs for the 540-state model finished within the time limit, 7.2% of them failed to converge to a feasible solution satisfying (6).

Figure 1 demonstrates that our neural network approximations can accurately recover the retailer's complex holding cost function. The plots compare the true $h$ for the 540-state model with the best estimate, $\hat{h}$, from all successful runs of each estimator. We define "best" as the estimate that minimizes $\|\hat{h} - h\|_1$, where $\|\cdot\|_1$ denotes the $\ell_1$ norm. The functional $R^2$ that treats each state as one observation indicates that the best NFXP, CCP, SC, UFXP, and OUFXP neural network approximations explain 98.2%, 98.4%, 98.9%, 97.3%, and 97% of the variation in $h$ for the 540-state model, and 95.4%, 97.4%, 98.6%, 97.7%, and 97.4% for the 5400-state model, respectively.[13] In contrast, the best MPEC approximation for the 540-state model explains only 57.8% of the variation in $h$. Hence, out of 600 MPEC runs, not a single one yielded a reasonable estimate of $h$.

Figure 2 plots the empirical CDF of the 540-state model's holding cost error across the 100 $\hat{\theta}$ initializations, for each neural network, optimizer, and estimator. These empirical CDFs demonstrate that we cannot attribute MPEC's poor performance to Knitro, as that optimizer performs well for the other five estimators. The MPEC objective—with its 540 additional parameters and constraints—simply seems harder to maximize. However, the MPEC estimator is not the only one that yields poor estimates: 49%, 48%, 50%, 66%, and 67% of NFXP, CCP, SC, UFXP, and OUFXP estimates are worse than the poor MPEC estimate depicted in Figure 1.

This performance inconsistency underscores the need for multiple optimization starts. To make this point clear, we will define an estimate as "inadequate" if its holding cost error, $\|\hat{h} - h\|_1$, is 50% larger than the smallest error for the given (neural network, optimizer, estimator) triple. For example, the minimum (Softplus Balanced, L-BFGS, NFXP) error is 0.0609, so the (Softplus Balanced, L-BFGS, NFXP) estimates for which $\|\hat{h} - h\|_1 \geq 1.5 \cdot 0.0609 = 0.0913$ are inadequate. Overall, 81%, 80%, 81%, 85% and 86% of NFXP, CCP, SC, UFXP, and OUFXP estimates are inadequate. Accordingly, ensuring less than a 1% probability of reporting an inadequate estimate would require running the optimizer from at least $\lceil \log(0.01)/\log(0.80) \rceil = 21$ starting points.

Our unnested estimators are relatively more competitive when we initialize the optimizer with $R = 21$ starting points than when we initialize them with $R = 1$ starting point, as Table 1 demonstrates. This table reports the ratios of the average estimation times of NFXP, CCP, SC, and MPEC divided by the average estimation times of UFXP[14] (recall that we truncate all estimation

[13] The seven-day computational limit may have lowered the $R^2$ values for the NFXP and CCP estimators of the 5400-state model, as optimization runs that explored the parameter space more thoroughly were more likely to time out.

[14] See Table 3 in Appendix for an OUFXP analog.

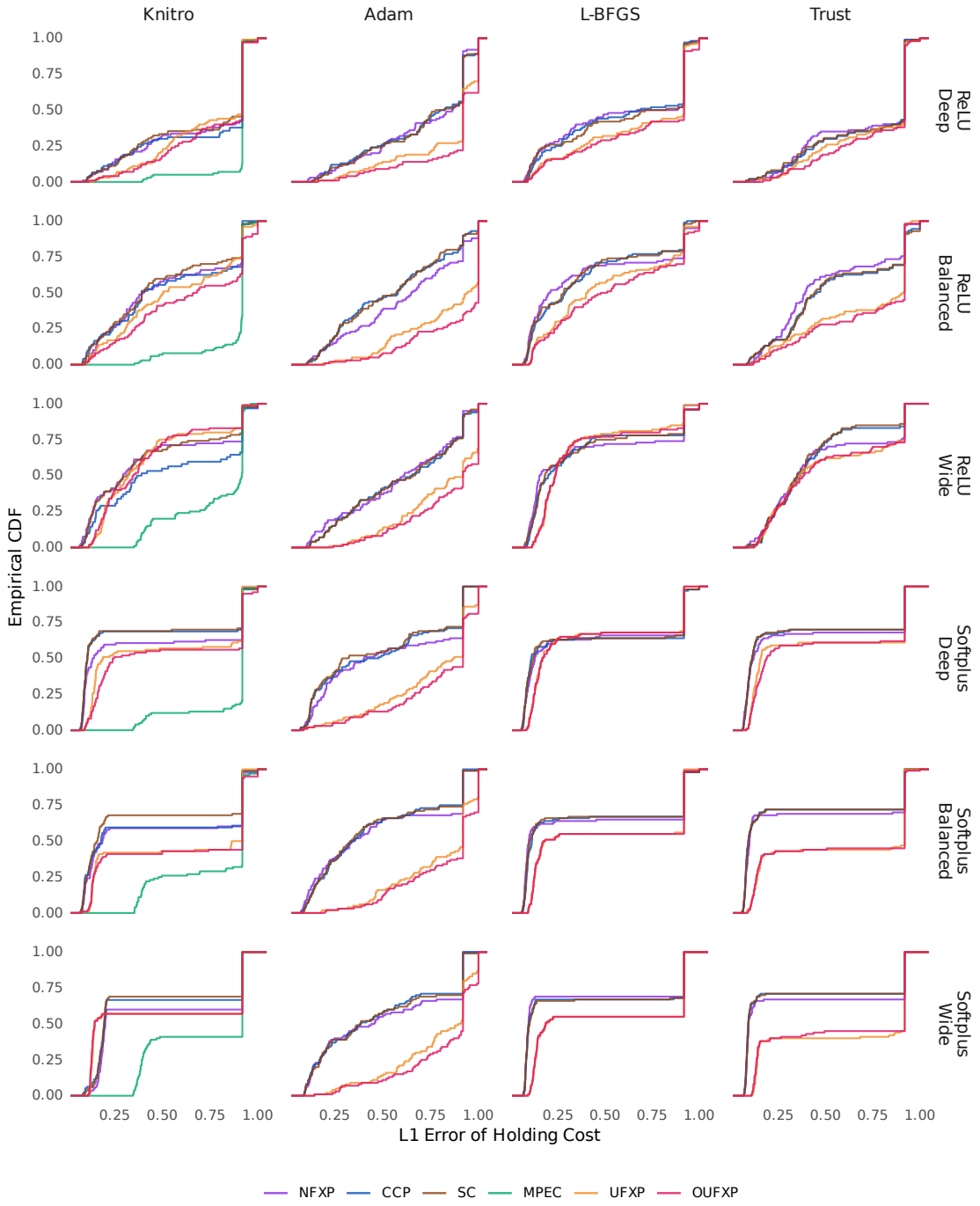



Figure 2: Empirical cumulative distribution functions (CDFs) of the holding cost estimation errors, $\|\hat{h} - h\|_1$, for the 540-state model. Each panel corresponds to the 100 random initializations that correspond to a given optimizer and neural network configuration. Line colors distinguish the estimators. The horizontal axis is capped at 1.0 for visual clarity.

Table 1: Performance Ratios: Average Estimation Time of
Benchmark Estimators Divided by Average Estimation Time of UFXP.

| | | R=1 | | | | R=21 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NFXP | CCP | SC | MPEC | NFXP | CCP | SC | MPEC |
| 540 State Model | | | | | | | | | |
| Adam | ReLU Balanced | 1681 | 2146 | 310 | | 2077 | 2652 | 384 | |
| | ReLU Deep | 1402 | 1914 | 293 | | 1709 | 2332 | 357 | |
| | ReLU Wide | 1841 | 2308 | 334 | | 2349 | 2944 | 426 | |
| | Softplus Balanced | 1560 | 2081 | 296 | | 1908 | 2546 | 362 | |
| | Softplus Deep | 1311 | 1814 | 267 | | 1566 | 2167 | 319 | |
| | Softplus Wide | 1703 | 2352 | 354 | | 2174 | 3002 | 452 | |
| Knitro | ReLU Balanced | 2003 | 2562 | 826 | 9 | 2312 | 2957 | 954 | 11 |
| | ReLU Deep | 990 | 1141 | 276 | 7 | 1152 | 1327 | 321 | 9 |
| | ReLU Wide | 1526 | 1948 | 625 | 10 | 1662 | 2120 | 680 | 11 |
| | Softplus Balanced | 4738 | 4926 | 792 | 23 | 6530 | 6790 | 1091 | 32 |
| | Softplus Deep | 2622 | 2302 | 321 | 25 | 3551 | 3117 | 435 | 34 |
| | Softplus Wide | 7575 | 5669 | 712 | 37 | 13234 | 9903 | 1243 | 64 |
| L-BFGS | ReLU Balanced | 590 | 699 | 98 | | 2154 | 2553 | 357 | |
| | ReLU Deep | 449 | 419 | 61 | | 2226 | 2073 | 303 | |
| | ReLU Wide | 642 | 815 | 94 | | 1885 | 2393 | 277 | |
| | Softplus Balanced | 1239 | 1528 | 193 | | 3989 | 4920 | 622 | |
| | Softplus Deep | 1043 | 1332 | 159 | | 2845 | 3634 | 433 | |
| | Softplus Wide | 1576 | 1809 | 222 | | 5462 | 6271 | 769 | |
| Trust | ReLU Balanced | 1491 | 2358 | 473 | | 1929 | 3050 | 612 | |
| | ReLU Deep | 1490 | 1608 | 679 | | 2322 | 2506 | 1058 | |
| | ReLU Wide | 2072 | 2506 | 495 | | 2580 | 3120 | 617 | |
| | Softplus Balanced | 662 | 629 | 63 | | 1226 | 1166 | 117 | |
| | Softplus Deep | 726 | 701 | 69 | | 1248 | 1205 | 119 | |
| | Softplus Wide | 435 | 456 | 59 | | 861 | 903 | 116 | |
| 5400 State Model | | | | | | | | | |
| L-BFGS | Softplus Balanced | 486 | 497 | 307 | | 6998 | 7160 | 4414 | |
| Trust | Softplus Balanced | 468 | 456 | 104 | | 4067 | 3959 | 906 | |

*Note:* Values report the average estimation times of the benchmark estimators, divided by the average estimation time of UFXP, both when the optimizers are initialized with a single random start ($R = 1$), and multiple optimization starts ($R = 21$). These ratios combine three statistics: the Average Benchmark Estimation Time, the UFXP Fixed Point Time, and the Average UFXP Optimization Time. The Average Benchmark Estimation Time is the total time required for benchmark estimators (NFXP, CCP, SC, and MPEC) to obtain the parameters $\hat{\theta}$, truncated to a maximum of seven days, and averaged across 100 optimization starts. The UFXP Fixed Point Time is the total time required to solve for the dual variables $\lambda_1, \ldots, \lambda_m$, which is done only once prior to optimization and shared across all starts. The Average UFXP Optimization Time is the total time required to obtain the parameters, $\hat{\theta}_{\text{UFXP}}$, given $\lambda_1, \ldots, \lambda_m$, averaged across 100 optimization starts. No truncation was needed for either of the UFXP time components, as all were completed within one week. Finally, we tabulate:

$$\frac{R \cdot \text{Average Benchmark Estimation Time}}{\text{UFXP Fixed Point Time} + R \cdot \text{Average UFXP Optimization Time}}.$$

times to seven days, a transformation that works against us, since no UFXP run lasted longer than a week). Increasing $R$ makes UFXP relatively more efficient, as it enables the estimator to amortize the time spent solving dual-variable fixed points across more optimization starts. For example, for the (Softplus Balanced, L-BFGS, 5400-state) case, computing $\lambda_1, \ldots, \lambda_m$ requires 872 seconds, whereas maximizing $-\mathcal{Q}_Z^\theta$ requires an average of 20 seconds. Accordingly, the average estimation time when we initialize UFXP with $R$ starts is $872 + 20R$. In contrast, the corresponding average estimation times when we initialize NFXP, CCP, and SC with $R$ starts are $433586R$, $443627R$, and $273470R$. Note that whereas the average UFXP time is *affine* in $R$, the average NFXP, CCP, SC, and MPEC times are *linear* in $R$, since each optimization run solves an independent set of fixed points.

But even without this multi-start advantage, the unnested estimator is still markedly more efficient: across all $R = 1$ cases, UFXP is between 435 and 7575 times faster than NFXP, between 419 and 5669 times faster than CCP, between 59 and 826 times faster than SC, and between 7 and 37 times faster than MPEC. And increasing the number of optimization starts accentuates this advantage: across all $R = 21$ cases, UFXP is between 861 and 13234 times faster than NFXP, between 903 and 9903 times faster than CCP, between 116 and 4414 times faster than SC, and between 9 and 64 times faster than MPEC. (The MPEC times are moot, since that estimator did not yield sensible estimates.)

Table 2 compares the estimators with two other performance metrics: the *workload* and the *span*. The workload is the average number of fixed points solved per optimization run. For example, UFXP has a workload of $m = 100$, since it solves $\lambda_1, \cdots, \lambda_m$ upfront and recycles these values across all $R$ jobs. The OUFXP workload is between 270 and 286, since the number of fixed points associated with $\frac{\partial}{\partial \theta} V_P^\theta$ and with the second-stage dual variables varies with the neural network configuration.[15] In contrast, the NFXP workload is between 10946 and 300925,[16] and the CCP workload is between 1624 and 188398. (And these workloads would have been much larger had we not used the dual-fixed point technique outlined in Section 3.2 to streamline the corresponding gradient and Hessian computations.)

The second metric reported in Table 2 reflects the fact that equations that can be solved in

[15]The OUFXP workload calculation is predicated on solving the second-stage fixed points for only the best of the first-stage estimates.

[16]This actually understates the computational burden of NFXP. Following Rust's (2000) manual, we implement a "polygorithm" that combines value iterations (contraction mappings) and policy iterations (Newton-Kantorovich steps), only the latter of which comprise fixed point equations that contribute to our tables. For example, the average (Softplus Balanced, L-BFGS) NFXP run not only sequentially solves 77709 fixed point equations, but also sequentially implements an additional 21764 Bellman contractions.

Table 2: Fixed Point Counts: Span and Workload by Estimator.

| | | Workload | | | | | Span | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NFXP | CCP | SC | UFXP | OUFXP | NFXP | CCP | SC | UFXP | OUFXP |
| 540 State Model | | | | | | | | | | | |
| Adam | ReLU Balanced | 254031 | 25974 | 25921 | 100 | 286 | 254031 | 25974 | 25921 | 1 | 3 |
| | ReLU Deep | 238470 | 24516 | 24453 | 100 | 270 | 238470 | 24516 | 24453 | 1 | 3 |
| | ReLU Wide | 252468 | 25189 | 25292 | 100 | 282 | 252468 | 25189 | 25292 | 1 | 3 |
| | Softplus Balanced | 260990 | 26209 | 26277 | 100 | 286 | 260990 | 26209 | 26277 | 1 | 3 |
| | Softplus Deep | 257027 | 25967 | 26441 | 100 | 270 | 257027 | 25967 | 26441 | 1 | 3 |
| | Softplus Wide | 250916 | 25728 | 25595 | 100 | 282 | 250916 | 25728 | 25595 | 1 | 3 |
| Knitro | ReLU Balanced | 53214 | 18527 | 72955 | 100 | 286 | 27387 | 1285 | 5561 | 1 | 3 |
| | ReLU Deep | 30427 | 7894 | 17149 | 100 | 270 | 16464 | 553 | 1330 | 1 | 3 |
| | ReLU Wide | 70940 | 30664 | 88138 | 100 | 282 | 38042 | 2239 | 6737 | 1 | 3 |
| | Softplus Balanced | 219570 | 126818 | 222086 | 100 | 286 | 87850 | 6219 | 11122 | 1 | 3 |
| | Softplus Deep | 118774 | 54905 | 68766 | 100 | 270 | 48223 | 2942 | 3666 | 1 | 3 |
| | Softplus Wide | 300925 | 188398 | 207529 | 100 | 282 | 117483 | 9127 | 9639 | 1 | 3 |
| L-BFGS | ReLU Balanced | 29520 | 2998 | 3389 | 100 | 286 | 29520 | 2998 | 3389 | 1 | 3 |
| | ReLU Deep | 19304 | 1624 | 1751 | 100 | 270 | 19304 | 1624 | 1751 | 1 | 3 |
| | ReLU Wide | 37951 | 3861 | 3610 | 100 | 282 | 37951 | 3861 | 3610 | 1 | 3 |
| | Softplus Balanced | 77709 | 7718 | 7566 | 100 | 286 | 77709 | 7718 | 7566 | 1 | 3 |
| | Softplus Deep | 72577 | 7343 | 6521 | 100 | 270 | 72577 | 7343 | 6521 | 1 | 3 |
| | Softplus Wide | 100997 | 9113 | 8649 | 100 | 282 | 100997 | 9113 | 8649 | 1 | 3 |
| Trust | ReLU Balanced | 20265 | 12349 | 21137 | 100 | 286 | 4562 | 423 | 715 | 1 | 3 |
| | ReLU Deep | 10946 | 6922 | 13156 | 100 | 270 | 2613 | 257 | 490 | 1 | 3 |
| | ReLU Wide | 24971 | 15795 | 27104 | 100 | 282 | 5587 | 547 | 926 | 1 | 3 |
| | Softplus Balanced | 20247 | 17239 | 15527 | 100 | 286 | 4632 | 590 | 534 | 1 | 3 |
| | Softplus Deep | 27822 | 18491 | 16760 | 100 | 270 | 6557 | 691 | 630 | 1 | 3 |
| | Softplus Wide | 14407 | 13207 | 14265 | 100 | 282 | 3420 | 466 | 502 | 1 | 3 |
| 5400 State Model | | | | | | | | | | | |
| L-BFGS | Softplus Balanced | 4177 | 392 | 4410 | 100 | 286 | 4177 | 392 | 4410 | 1 | 3 |
| Trust | Softplus Balanced | 1985 | 2498 | 11082 | 100 | 286 | 465 | 88 | 384 | 1 | 3 |

*Note:* Workload measures the number of fixed points solved per optimization run. Span measures the number of fixed points that must be solved sequentially. All reported values are averages over 100 starts. For benchmark estimators using Trust or Knitro, the workload exceeds the span because analytical Hessians require $t$ additional fixed points, which can be solved in parallel. OUFXP counts include both the UFXP counts and two additional parallelizable fixed point computations required to build the optimal weights and the corresponding dual fixed points, whose sizes depend on the neural network. MPEC is excluded as it does not solve fixed points. We also exclude counts from any runs that failed to converge within the seven-day time limit. Because all UFXP and OUFXP runs successfully converged, these timeouts exclusively affected the benchmark estimators. Consequently, this exclusion biases the benchmark averages downward, as the runs with higher fixed point counts are more likely to time out.

parallel are less burdensome than those that must be solved in sequence. Specifically, the span records the average length of an optimization run's computational critical path, under maximal parallelization—that is, the expected number of batches of fixed point equations that must be solved in sequence. For example, UFXP always has a span of 1, since $\lambda_1, \cdots, \lambda_m$ can be solved in parallel. In contrast, OUFXP always has a span of 3, because it solves the fixed points that define (i) the first-stage dual variables, (ii) the Jacobian matrix $\frac{\partial}{\partial \theta} V_{\hat{P}}^{\theta}$ (which influences $Z^{\hat{\theta}_{\text{UFXP}}}$), and (iii) the second-stage dual variables. The fixed points in each of these three stages can be parallelized, but one stage can't begin until the preceding stage has ended, so switching from UFXP to OUFXP triples the span. However, switching from OUFXP to NFXP or CCP increases the span by more than *three orders of magnitude*.

# 5 Empirical Application

## 5.1 Model

We now present an empirical model that we believe would be intractable to estimate without our approach. Our model is an extension of the toy example studied in Section 4. Whereas we previously estimated a single-stage inventory model, we will now estimate a multi-echelon supply chain model. And whereas we previously had three state variables, we will now have six (the three from before, plus in-transit inventory, upstream inventory, and upstream demand).

We will estimate our model with a 1,073,679-observation panel of daily data, spanning from February 2, 2011 to December 31, 2014. The sample describes the movement of 38 laundry detergent products through a supply chain that comprises one upstream distribution center (DC) and 67 downstream stores. (We focus on detergents, since their bulk makes them particularly difficult to store.) For each store–product–day, we observe: (i) retail prices, wholesale prices, and discounts, (ii) customer sales, (iii) the amount of inventory at the store, at the DC, and in the pipeline in between, and (iv) the store's inventory order, and the size and timing of the corresponding shipment from the DC.

A large Shanghai retailer owns all stores in our sample, but each store is overseen by a manager whose performance bonuses and promotion prospects depend on the functioning of their particular store. The managers, therefore, prioritize local store performance over global chain performance. For example, managers will generally squabble over scarce upstream inventory (Bray et al., 2019).

As before, the action variable is order quantity, $q \in \mathbb{Q} \equiv \{0, 6, 12, 20\}$, which the store manager

chooses with the aim of maximizing expected discounted utility, under discount factor $\beta = 0.9997$. Each possible order quantity corresponds to an idiosyncratic utility shock, which we collect in a vector of independent Gumbels, $e \in \mathbb{R}^{|\mathbb{Q}|}$. Unobserved state vector $e$ is accompanied by observed state vector $x \equiv (r, o, i, j, k, \ell)$. The first three observed state variables are analogous to their Section 4 counterparts: $r \in \{1, 2, 3, 4, 5\}$ indexes the mean demand for the focal detergent at the given store, $o \in \{1, 2, 3\}$ indexes the aggregate inventory of all but the focal detergent at the given store, and $i \in \mathbb{I}$ denotes the inventory level of the focal detergent at the given store, where $|\mathbb{I}| = 100$ and $\max \mathbb{I} = 149$.[17]

The final three state variables describe the supply chain for the focal product: $j \in \mathbb{Q}$ specifies the amount of inventory in transit between the DC and the given store, $k \in \{1, 2, 3\}$ indexes the amount of inventory held at the distribution center, and $\ell \in \{1, 2, 3\}$ indexes the mean demand faced by the DC (i.e., the expected sum of store orders). Upstream demand $\ell$ is relevant because it is predictive of future upstream inventory: the DC roughly follows an $(s, S)$ inventory model, with demand state $\ell$ determining the slope of the line from $S$ to $s$. Upstream inventory $k$ is relevant because the DC is less likely to fulfill an order when its stocks run low (Bray et al., 2019, p. 456): e.g., the DC fulfills 91% of orders when its inventory level is above the first decile but only 34% when its inventory level is below the first decile. And pipeline inventory $j$ is relevant because there is a lag between an order's placement and its fulfillment (unlike in the toy model): conditional on fulfillment, orders arrive after one day with probability 0.13, after two days with probability 0.67, and after three or more days with probability 0.20. To simplify our model dynamics, we suppose that all fulfillments take two days, so that today's $j$ reports the inventory the store will receive tomorrow in response to the order it placed yesterday.

The total utility from the focal product on the given day is

$$\tilde{u}_q(r, o, i, j, d) + e_q \equiv -h(r, o, i) + \eta \min(d, i) - \kappa \mathbb{1}\{j > 0\} + e_q,$$

where $d$ is the daily customer demand, $e_q$ is an independent Gumbel shock associated with order quantity $q$, $\eta$ is the value of a sale, $h(r, o, i)$ is the inventory holding cost, and $\kappa$ is the cost of receiving a shipment of inventory. Unfortunately, this utility does not fit into the framework presented in

[17] Capping $i$ at 149 truncates this variable's top 10%. We perform this truncation because the right tail of $i$ spans an extremely large range. Indeed, we believe modeling the tail behavior of $i$ would likely require an entirely different approach. Thus, rather than estimate $h$ over the full domain of $i$, we more modestly estimate this function for non-tail values of $i$. To mitigate distortions associated with this truncation, we set the rows of $Z_1, \ldots, Z_m$ that correspond to $i > \max \mathbb{I} - \max \mathbb{Q} = 149 - 20 = 129$ to zero. In other words, we assign zero empirical weight to observations for which the agent could order inventory beyond 149 units.

Section 2.2, as it depends on demand $d$. However, we can integrate over this random variable to create the following equivalent utility function:

$$u_q(r, o, i, j) + e_q \equiv -h(r, o, i) - \eta \mathrm{E}((d - i)^+ | r) - \kappa \mathbb{1}\{j > 0\} + e_q.$$

We can replace $\tilde{u}$ with $u$, because maximizing the expected discounted sum of $\tilde{u}_q(r, o, i, j, d) + e_q$ yields the same policy as maximizing the expected discounted sum of $u_q(r, o, i, j) + e_q$.

Utility function $u$ is the same as that used in Section 4, except the shipping cost depends on $j$ rather than $q$, since not all orders translate into shipments. As before, the object of interest is the holding cost function $h$, which we will model with neural networks. Note that state variables $k$ and $\ell$ do not influence current utilities, but they do influence future utilities, via the following transition kernel:

$$f_q(x'|x) \equiv f^r(r'|r) f^o(o'|o) f^i(i'|r, i, j) f^j(j'|k, q) f^k(k'|k, \ell) f^\ell(\ell'|\ell).$$

As you see, today's $\ell$ influences tomorrow's $k$, today's $k$ influences tomorrow's $j$, and today's $j$ influences tomorrow's $i$.

Transition kernels $f^r$, $f^o$, $f^k$, $f^\ell$ are general, but transition kernels $f^i$ and $f^j$ reflect the following laws of motion:

$$\begin{aligned} & i' = \min(\max \mathbb{I}, (i - d)^+ + j), \\ \text{and} \quad & j' = bq, \end{aligned}$$

where random demand $d$ resolves from general distribution $f^d(d|r)$ and order fulfillment indicator variable $b$ resolves from a Bernoulli distribution $f^b$ whose mean depends on $k$.

## 5.2 Estimation Details

Our sample does not include mean demands, so we begin by pre-estimating the expected inventory draw-downs of the DC and each store, by product and day.

We estimate the mean downstream demand using a neural network: First, we input a rich set of demand features ($y$) into a multilayer perceptron with two hidden layers of widths 64 and 32, and an output layer of width 1. Second, we exponentiate the neural network's output and treat the result as the mean of a Poisson distribution to evaluate the empirical likelihood of the

observed sales. Specifically, we model the reduced-form probability of observing sales quantity $s$ given features $y$ as $\exp(m(y))^s \exp(-\exp(m(y)))/s!$, where $m(y)$ is the network's output for feature vector $y$. We train this model by maximizing the corresponding Poisson log-likelihood, $\sum_y \sum_s n_{ys} \log(\exp(m(y))^s \exp(-\exp(m(y)))/s!)$, where $n_{ys}$ is the number of observations with feature set $y$ and sales $s$. Finally, we set the expected mean demands to the corresponding fitted values, $\exp(m(y))$, and then map each mean demand to a demand index $r \in \{1, 2, 3, 4, 5\}$ (see Appendix Section E for details).

We estimate the DC's mean demand analogously, but for this case, we start with the upstream demand, which is the sum of store orders, and end with the upstream demand state index $\ell \in \{1, 2, 3\}$.

In addition to pre-estimating demand means, to construct $r$ and $\ell$, we must also pre-estimate CCPs and state transition probabilities. For CCPs, we have too many states to use a simple frequency estimator, so we again use a neural network: First, we input the state vectors $x \in \mathbb{X}$ into a multilayer perceptron with two hidden layers of widths 64 and 32, and an output layer of width 4 (since $|\mathbb{Q}| = 4$). Second, we apply a softmax function to the neural network's output and use the result to evaluate the empirical log-likelihood of the observed order quantities. Specifically, we model the reduced-form probability of ordering $q$ in state $x$ as $\exp(m_q(x))/\sum_{q' \in \mathbb{Q}} \exp(m_{q'}(x))$, where $m_q(x)$ is the $q$th element of the network's output vector at state $x$. We train this model by maximizing the corresponding log-likelihood, $\sum_{x \in \mathbb{X}} \sum_{q \in \mathbb{Q}} n_{xq} \log(\exp(m_q(x))/\sum_{q' \in \mathbb{Q}} \exp(m_{q'}(x)))$, where $n_{xq}$ is the number of state-$x$ and order-$q$ observations. Finally, we set $\hat{P}$ to the corresponding fitted values.

To be clear: we do *not* need our unnested estimators to accommodate these incidental networks, which are trained in an offline pre-estimation stage.

After pre-estimating CCPs, we then pre-estimate state transition probabilities. We estimate Poisson arrival rates for $f^d$ and Bernoulli probabilities for $f^b$, then combine these with the corresponding laws of motion to estimate $f^i$ and $f^j$ on the discretized state space (see Appendix Section E for details). We estimate the other transition kernels, $f^r$, $f^o$, $f^k$, and $f^\ell$ nonparametrically with their empirical distributions. For example, upstream inventory $k$ gradually decreases, as the DC sends shipments to stores, and then jumps up, when the DC receives a shipment from its supplier; rather than separately estimate the DC's inbound and outbound shipments, we estimate the aggregate inventory adjustment—i.e., the final next-period inventory level, $k'$, given the current-period inventory level, $k$, and demand rate, $\ell$. As before, we combine estimates $\hat{f}^r$, $\hat{f}^o$, $\hat{f}^k$, $\hat{f}^\ell$, $\hat{f}^d$, and $\hat{f}^b$ into a corresponding $\hat{f}_q$ estimate.

After estimating the CCPs and the transition kernel, we are ready to estimate $\theta$. We estimate these structural parameters with UFXP. We decompose $h$ into a base inventory cost and a multiplicative congestion penalty: $h(r, o, i) = h^i(r, i)(1 + h^o(r, o))$, where functions $h^i(\cdot)$ and $h^o(\cdot)$ stem from separate neural networks, which together comprise 77 parameters (60 weights and 17 biases).

We set $h^i(r, i) \equiv \sum_{j=1}^{i} w_j n(r, j)^2$, where $n(r, j)$ is the output of a multilayer perceptron that uses a single hidden layer of width 11 to map $r$, $i$, and interaction $ri$ to a scalar output, and $w_j$ is the fraction of discrete inventory levels that correspond to inventory state $j$. For example, $w_{80} = 2/150$: the denominator reflects the 150 possible inventory levels (0–149 units) and the numerator reflects the two levels (80 and 81 units) that map to state $i = 80$. By design, $h^i(r, i)$ is non-decreasing in $i$, and is zero when $i$ is zero.

We set $h^o(r, o) \equiv \sum_{k=2}^{o} m(r, k)^2$, where $m(r, k)$ is the output of another multilayer perceptron that uses a single hidden layer of width 4 to map $r$, $o$, and interaction $ro$ to a scalar output. Similarly, by design, $h^o(r, o)$ is non-decreasing in $o$, and is zero when $o$ is one.

We initialize our neural networks with 1000 random starting values, following the methodology outlined in Section 4.1. We use the Trust optimizer and recycle the same $\lambda_1, \cdots, \lambda_m$ dual fixed points for each of the $R = 1000$ minimizations of $\mathcal{Q}_Z^\theta$. We construct the $m = 300$ weighting matrices, $Z_1, \ldots, Z_m$, by drawing their elements from a standard normal distribution (with the proviso that the rows that correspond to $i > \max \mathbb{I} - \max \mathbb{Q} = 149 - 20 = 129$ are zeroed out, as explained in footnote 17).

We use the computer setup described in Section 4.1 to make the estimation times roughly comparable to those reported in Section 4.2.

### 5.3 Results

We run the estimation with 1000 random starts. Our unnesting approach makes this multi-start strategy computationally feasible: it takes 208.4 minutes to solve dual fixed points $\lambda_1, \cdots, \lambda_m$, but only 1.4 minutes, on average, to convert them into a set of estimates, $\hat{\theta}_{\text{UFXP}}$.

Our estimates illustrate the intrinsic strangeness of neural networks. In most traditional settings, the estimator's criterion function will feature a unique global optimum and a smattering of clearly inferior local optima (if any). This is not the case for neural networks. Rather than a single point, neural networks often have a high-dimensional valley of nearly equivalent solutions. And this is precisely what we find: out of our 1000 optimization runs, 383 yield a value that is within 0.5% of the overall minimum $\mathcal{Q}_Z^{\hat{\theta}}$. Because optimality encompasses a region rather than a point,

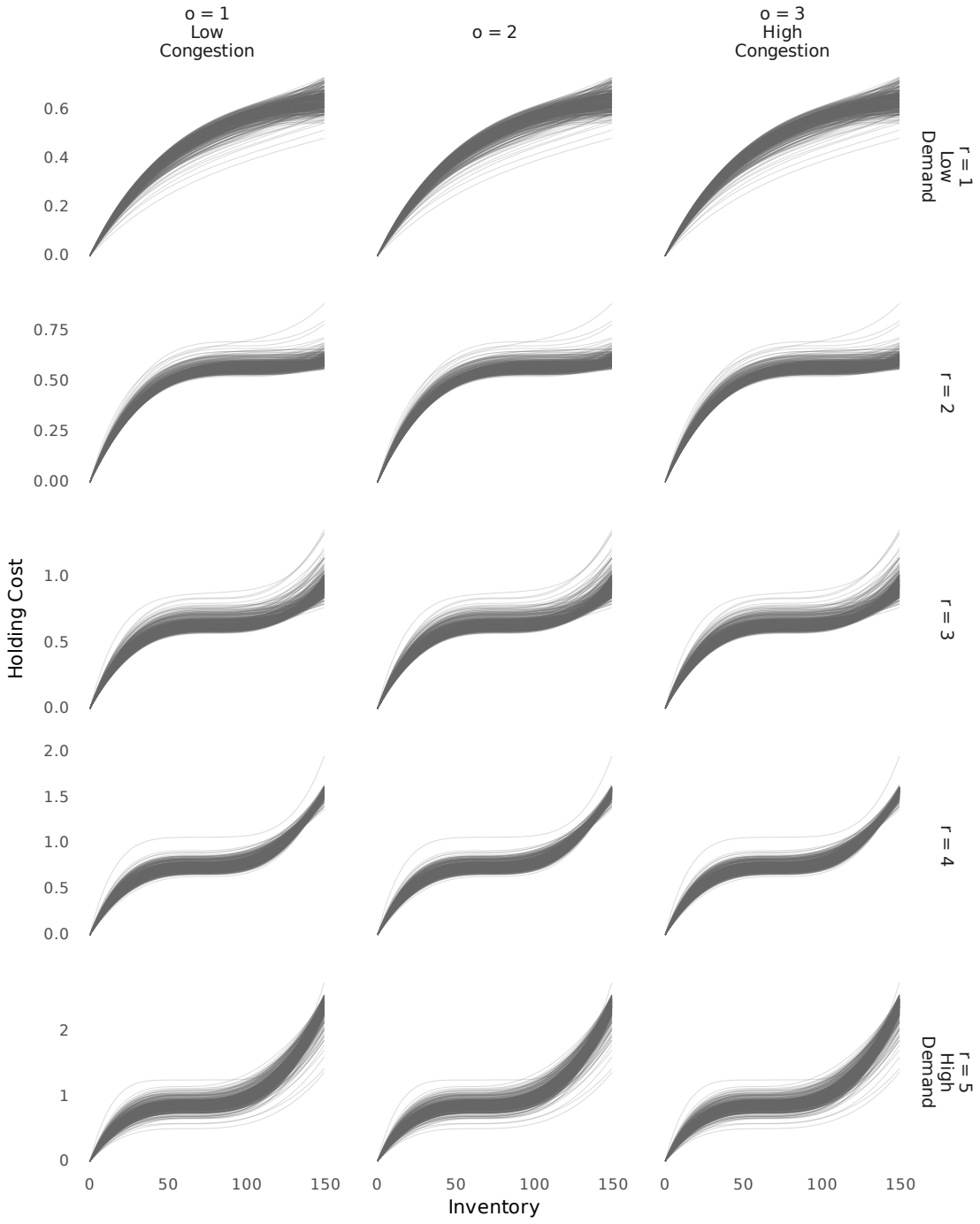



Figure 3: Ensemble of 383 estimated holding cost functions whose UFXP objectives fall within 0.5% of the minimum attained across 1000 random starts. Each panel depicts $\hat{h}(r, o, i)$ for a fixed demand state, $r$, a fixed congestion state, $o$, and all inventory states $i \in \mathbb{I}$.

an honest assessment of the utility function requires an *ensemble* of estimates, which reinforces the need for many optimization starts.

Across the ensemble, the estimated shortage cost $\hat{\eta}$ averages 0.41 with a standard deviation of 0.008, and the fixed ordering cost $\hat{\kappa}$ averages 6.79 with a standard deviation of 0.003. We plot the holding cost functions that correspond with our ensemble of 383 near-optimal estimates in Figure 3. Despite some variation across the ensemble, an overall pattern clearly emerges: congestion state $o$ has a negligible impact on costs, whereas the demand state $r$ significantly alters the shape of $h$. The influence of $r$ is unsurprising because this variable captures structural heterogeneity across stores, with larger stores exhibiting consistently higher $r$ values.

Overall, the estimated holding cost function proves to be neither convex nor concave, let alone linear or quadratic. This last fact contradicts most of the literature on inventories, which generally imposes linear or quadratic forms.

# 6 Conclusion

The key to this work is the switch from the primal linear program

$$\max_{V \in \mathbb{R}^X} \quad w'V \quad \text{s.t.} \quad V = U^\theta_{\hat{P}} + \beta F_{\hat{P}} V$$

to the dual linear program

$$\min_{\lambda \in \mathbb{R}^X} \quad \lambda' U^\theta_{\hat{P}} \quad \text{s.t.} \quad \lambda = w + \beta F'_{\hat{P}} \lambda.$$

In the primal problem, the utility parameter $\theta$ enters the constraint; in the dual, it enters the objective. Consequently, the primal solution varies with $\theta$, whereas the dual solution does not. By strong duality, both programs have the same value, implying

$$w' V^\theta_{\hat{P}} = \lambda' U^\theta_{\hat{P}}.$$

Thus, after solving the dual variable vector $\lambda$, we can evaluate $w' V^\theta_{\hat{P}}$ for many values of $\theta$ with ease.

This observation motivates a dynamic discrete choice estimator that depends on value function $V^\theta_{\hat{P}}$ only through inner products $w_i' V^\theta_{\hat{P}}$, for $i = 1, \ldots, m$, where $m$ is not much more than the length of the utility parameter vector, $\theta$. After computing the corresponding dual variables, $\lambda_1, \ldots, \lambda_m$, such an estimator would be trivial to evaluate.

Restricting attention to linear functionals of $V^\theta_{\hat{P}}$ precludes likelihood-based estimators, which are intrinsically nonlinear. Thus, we base our estimator on the following first-order conditions:

$$\rho_a^{-1}(P^\theta) - U_a^\theta + U_A^\theta - \beta(F_a - F_A)V^\theta = 0.$$

First, we approximate these first-order conditions by replacing $P^\theta$ and $V^\theta$ with $\hat{P}$ and $V^\theta_{\hat{P}}$. And then we minimize the squared Euclidean norm of a random low-dimensional compression of this approximation:

$$\hat{\theta}_{\text{UFXP}} \equiv \arg\min_{\theta\in\Theta} \sum_{i=1}^{m} \Big( \sum_{a\in\mathbb{A}\setminus\{A\}} Z'_{ia}\big(\rho_a^{-1}(\hat{P}) - U_a^\theta + U_A^\theta - \beta(F_a - F_A)V^\theta_{\hat{P}}\big)\Big)^2.$$

In the expression above, it suffices to set $Z_{ia}$ to a length-$X$ vector of independent standard normal random variables, for $i \in \{1, \cdots, m\}$, and to set $m$ as small as $t + 1$, where $t$ is the length of $\theta$. Expressed this way, our objective does not appear any easier to evaluate than the CCP estimator's objective. However, unlike the CCP objective, we can entirely remove the value function from ours, since

$$\hat{\theta}_{\text{UFXP}} = \arg\min_{\theta\in\Theta} \sum_{i=1}^{m} \Big(\lambda'_i U^\theta_{\hat{P}} + \sum_{a\in\mathbb{A}\setminus\{A\}} Z'_{ia}(\rho_a^{-1}(\hat{P}) - U_a^\theta + U_A^\theta)\Big)^2,$$

$$\text{where} \quad \lambda_i \equiv w_i + \beta F'_{\hat{P}}\lambda_i$$

$$\text{and} \quad w'_i \equiv -\sum_{a\in\mathbb{A}\setminus\{A\}} \beta Z'_{ia}(F_a - F_A).$$

As you see, our estimator still depends on the $m$ fixed points that define $\lambda_1, \cdots, \lambda_m$, but it does not depend on any *nested* fixed points (i.e., those whose solutions change with $\theta$). Of course, UFXP could not be efficient, as its random compression is ad hoc. However, the inefficient UFXP estimates give way to efficient OUFXP estimates.

Our unnested estimators are motivated by a related unnesting of the CCP fixed point that arises when the utility function is linear. In that case, the UFXP and CCP unnesting schemes are comparable. Our approach, therefore, meaningfully improves upon the state of the art only when the utility function is nonlinear. Fortunately, a promising class of nonlinear utility functions is drawing increasing attention: those arising from neural networks.

Embedding a neural network in a dynamic discrete choice model sacrifices not just linearity, but all semblance of smoothness—needless to say, the objective function will not be concave. This means

the estimation problem will require brute force—initializing the optimizer with many different starting points. Our technique makes this practical, as each start reuses the same dual fixed point solutions, $\lambda_1, \ldots, \lambda_m$. In contrast, traditional estimators must start from scratch for each set of initial values. To make this concrete, note that solving the dual fixed points took 208.4 minutes for our 54000-state model, whereas converting these fixed points into utility parameter estimates took only 1.4 minutes, on average. Hence, initializing our estimator with 1000 starting points took only $(1000 \cdot 1.4 + 208.4)/(1.4 + 208.4) = 7.7$ times longer than initializing it with a single starting point. Compare this increase with the 1000-fold increase that would beset NFXP, CCP, SC, or MPEC if we started them with 1000 starting points. Our approach enables us to *swarm* the objective function with hundreds of cheap gradient ascent paths. UFXP likewise amortizes fixed point computations across optimizers and neural network specifications, making large-scale optimization ensembles computationally tractable.

And our estimators not only enable faster ensembles—they also enable faster individual runs. First, the UFXP fixed points can be computed in parallel, whereas the NFXP and CCP fixed points must be computed in sequence. Second, UFXP solves little more than $t$ fixed points, whereas traditional NFXP and CCP implementations solve $t$ fixed points for *each gradient ascent*, of which there could be hundreds. (However, our dual fixed point technique reduces the number of fixed points required to evaluate the NFXP and CCP gradients from $t$ to one.)

Now, for convenience, we will gather our empirical results:

- Initializing a nested estimator (NFXP, CCP, or SC) with 1000 starting values would increase the computational burden by a factor of 1000; in contrast, initializing UFXP with 1000 starting values increased the computational burden of our 54000-state model by a factor of 7.7.

- For the 540-state model, the median workload was 25444 with the nested estimators, 100 with UFXP, and 282 with OUFXP.

- For the 540-state model, the median span was 7642 with the nested estimators, 1 with UFXP, and 3 with OUFXP.

- For the 540-state model initialized with one starting value, the median estimation time was 6 hours with the nested estimators, 0.009 hours with UFXP, and 0.016 hours with OUFXP.

- For the 540-state model initialized with 21 starting values, the median estimation time was 123 hours with the nested estimators, 0.099 hours with UFXP, and 0.16 hours with OUFXP.

- For the 5400-state model initialized with one starting value, the median estimation time was 115 hours with the nested estimators, 0.248 hours with UFXP, and 0.748 hours with OUFXP.
- For the 5400-state model initialized with 21 starting values, the median estimation time was 2416 hours with the nested estimators, 0.369 hours with UFXP, and 0.962 hours with OUFXP.

## References


Aguirregabiria, Victor. 1999. The dynamics of markups and inventories in retailing firms. *The Review of Economic Studies* **66**(2) 275–308.

Aguirregabiria, Victor, Arvind Magesan. 2013. Euler equations for the estimation of dynamic discrete choice structural models. *Structural Econometric Models*, *Advances in Econometrics*, vol. 31. Emerald Group Publishing Limited, 3–44.

Aguirregabiria, Victor, Pedro Mira. 2002. Swapping the nested fixed point algorithm: A class of estimators for discrete markov decision models. *Econometrica* **70**(4) 1519–1543.

Aguirregabiria, Victor, Pedro Mira. 2007. Sequential estimation of dynamic discrete games. *Econometrica* **75**(1) 1–53.

Aguirregabiria, Victor, Pedro Mira. 2010. Dynamic discrete choice structural models: A survey. *Journal of Econometrics* **156**(1) 38–67.

Aouad, Ali, Antoine Désir. 2025. Representing random utility choice models with neural networks. *Management Science* **0**(0) null.

Arcidiacono, Peter, Paul B. Ellickson. 2011. Practical methods for estimation of dynamic discrete choice models. *Annual Review of Economics* **3**(Volume 3, 2011) 363–394.

Arcidiacono, Peter, Robert A. Miller. 2011. Conditional choice probability estimation of dynamic discrete choice models with unobserved heterogeneity. *Econometrica* **79**(6) 1823–1867.

Arkoudi, Ioanna, Rico Krueger, Carlos Lima Azevedo, Francisco C. Pereira. 2023. Combining discrete choice models and neural networks through embeddings: Formulation, interpretability and performance. *Transportation Research Part B: Methodological* **175** 102783.

Bajari, Patrick, C. Lanier Benkard, Jonathan Levin. 2007. Estimating dynamic models of imperfect competition. *Econometrica* **75**(5) 1331–1370.

Bray, Robert L. 2019a. Markov Decision Processes with Exogenous Variables. *Management Science* **65**(10) 4598–4606.

Bray, Robert L. 2019b. Strong convergence and dynamic economic models. *Quantitative Economics* **10**(1) 43–65.

Bray, Robert L., Yuliang Yao, Yongrui Duan, Jiazhen Huo. 2019. Ration gaming and the bullwhip effect. *Operations Research* **67**(2) 453–467.

Byrd, Richard H., Jorge Nocedal, Richard A. Waltz. 2006. Knitro: An integrated package for nonlinear optimization. G. Di Pillo, M. Roma, eds., *Large-Scale Nonlinear Optimization*. Springer US, Boston, MA, 35–59.

Cameron, A. Colin, Pravin K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. Cambridge University Press.

Dearing, Adam. 2019. Pseudo-value functions and closed-form ccp estimation of dynamic discrete choice models, working paper.

Han, Yafei, Francisco Camara Pereira, Moshe Ben-Akiva, Christopher Zegras. 2022. A neural-embedded discrete choice model: Learning taste representation with strengthened interpretability. *Transportation Research Part B: Methodological* **163** 166–186.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 1026–1034.

Hotz, V. Joseph, Robert A. Miller. 1993. Conditional Choice Probabilities and the Estimation of Dynamic Models. *The Review of Economic Studies* **60**(3) 497–529.

Hotz, V. Joseph, Robert A. Miller, Seth Sanders, Jeffrey Smith. 1994. A Simulation Estimator for Dynamic Models of Discrete Choice. *The Review of Economic Studies* **61**(2) 265–289.

Hsieh, Sung-Lin, Shaowei Ke, Zhaoran Wang, Chen Zhao. 2025. Logit neural-network utility. *Journal of Economic Behavior & Organization* **236** 107054.

Iskhakov, Fedor, Jinhyuk Lee, John Rust, Bertel Schjerning, Kyoungwon Seo. 2016. Comment on "constrained optimization approaches to estimation of structural models". *Econometrica* **84**(1) 365–370.

Judd, Kenneth. 1998. *Numerical Methods in Economics*, vol. 1. 1st ed. The MIT Press.

Kasahara, Hiroyuki, Katsumi Shimotsu. 2018. Estimation of Discrete Choice Dynamic Programming Models. *The Journal of Japanese Economic Association* **69**(1) 28–58.

Kingma, Diederik P., Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Lee, John M. 2012. *Introduction to Smooth Manifolds*, *Graduate Texts in Mathematics*, vol. 218. 2nd ed. Springer, New York, NY.

Liu, Dong C., Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming* **45** 503–528.

Miessi Sanches, Fabio A., Daniel Junior Silva, Sorawoot Srisuma. 2016. Ordinary least squares estimation of a dynamic game model. *International Economic Review* **57**(2) 623–634.

Newey, Whitney K., Daniel McFadden. 1994. Large sample estimation and hypothesis testing. *Handbook of Econometrics*, *Handbook of Econometrics*, vol. 4, chap. 36. Elsevier, 2111–2245.

Nocedal, Jorge, Stephen J. Wright. 2006. *Numerical Optimization*. 2nd ed. Springer Series in Operations Research and Financial Engineering, Springer, New York.

Pakes, Ariel, Michael Ostrovsky, Steven Berry. 2007. Simple estimators for the parameters of discrete dynamic games (with entry/exit examples). *The RAND Journal of Economics* **38**(2) 373–399.

Pesendorfer, Martin, Philipp Schmidt-Dengler. 2008. Asymptotic Least Squares Estimators for Dynamic Games1. *The Review of Economic Studies* **75**(3) 901–928.

Rust, John. 1986. Structural estimation of markov decision processes. *Handbook of Econometrics*, *Handbook of Econometrics*, vol. 4, 1st ed., chap. 51. Elsevier, 3081–3143.

Rust, John. 1987a. Optimal replacement of GMC bus engines: An empirical model of Harold Zurcher. *Econometrica* **55**(5) 999–1033.

Rust, John. 1987b. Optimal replacement of gmc bus engines: An empirical model of harold zurcher. *Econometrica* **55**(5) 999–1033.

Rust, John. 2000. Nested fixed point algorithm documentation manual, unpublished manuscript.

Shastri, Anant R. 2011. *Elements of Differential Topology*. CRC Press, Boca Raton, FL.

Sifringer, Brian, Virginie Lurkin, Alexandre Alahi. 2020. Enhancing discrete choice models with representation learning. *Transportation Research Part B: Methodological* **140** 236–261.

Su, Che-Lin, Kenneth L. Judd. 2012. Constrained optimization approaches to estimation of structural models. *Econometrica* **80**(5) 2213–2230.

van Cranenburgh, Sander, Shenhao Wang, Akshay Vij, Francisco Pereira, Joan Walker. 2022. Choice modelling in the age of machine learning - discussion paper. *Journal of Choice Modelling* **42** 100340.

van der Laan, Lars, Aurelien Bibaut, Nathan Kallus. 2025. Efficient inference for inverse reinforcement learning and dynamic discrete choice models.

Wei, Yanhao (Max), Zhenling Jiang. 2025. Estimating parameters of structural models using neural networks. *Marketing Science* **44**(1) 102–128.

# Appendix

## A UFXP derivatives

In Section 3.3 of the main text, we established the UFXP objective function $\mathcal{Q}_Z^\theta$ and defined the $m$ dual-variable vectors $\lambda_i$. In this section, we provide the explicit derivations showing how these vectors yield the objective's gradient and Hessian matrix:

$$\begin{aligned}
\tfrac{\partial}{\partial\theta}\mathcal{Q}_Z^\theta &= 2\sum_{i=1}^{m}\operatorname{tr}(Z_i'(\rho^{-1}(\hat{P}) - C^\theta_{V^\theta_{\hat{P}}}\Delta')) \\
&\quad \Big(\lambda_i'\sum_{a\in\mathbb{A}}\operatorname{diag}(\hat{P}_a)\tfrac{\partial}{\partial\theta}U_a^\theta - \sum_{a\in\mathbb{A}\setminus\{A\}} Z_{ia}'(\tfrac{\partial}{\partial\theta}U_a^\theta - \tfrac{\partial}{\partial\theta}U_A^\theta)\Big) \\
\text{and}\quad \tfrac{\partial^2}{\partial\theta_j\partial\theta_k}\mathcal{Q}_Z^\theta &= 2\sum_{i=1}^{m}\Big(\lambda_i'\sum_{a\in\mathbb{A}}\operatorname{diag}(\hat{P}_a)\tfrac{\partial}{\partial\theta_k}U_a^\theta - \sum_{a\in\mathbb{A}\setminus\{A\}} Z_{ia}'(\tfrac{\partial}{\partial\theta_k}U_a^\theta - \tfrac{\partial}{\partial\theta_k}U_A^\theta)\Big) \\
&\quad \Big(\lambda_i'\sum_{a\in\mathbb{A}}\operatorname{diag}(\hat{P}_a)\tfrac{\partial}{\partial\theta_j}U_a^\theta - \sum_{a\in\mathbb{A}\setminus\{A\}} Z_{ia}'(\tfrac{\partial}{\partial\theta_j}U_a^\theta - \tfrac{\partial}{\partial\theta_j}U_A^\theta)\Big) \\
&\quad + 2\sum_{i=1}^{m}\operatorname{tr}(Z_i'(\rho^{-1}(\hat{P}) - C^\theta_{V^\theta_{\hat{P}}}\Delta')) \\
&\quad \Big(\lambda_i'\sum_{a\in\mathbb{A}}\operatorname{diag}(\hat{P}_a)\tfrac{\partial^2}{\partial\theta_j\partial\theta_k}U_a^\theta - \sum_{a\in\mathbb{A}\setminus\{A\}} Z_{ia}'\Big(\tfrac{\partial^2}{\partial\theta_j\partial\theta_k}U_a^\theta - \tfrac{\partial^2}{\partial\theta_j\partial\theta_k}U_A^\theta\Big)\Big).
\end{aligned}$$

## B Unobserved Discount Factor

We have thus far taken $\beta$ as given. But we can implement versions of UFXP and OUFXP when $\beta$ is a parameter to be estimated. For simplicity, we will restrict attention to the case in which the Markov chain is ergodic under policy $\hat{P}$, in which case

$$|||(F_{\hat{P}}')^t - \pi_{\hat{P}}\mathbf{1}'||| = o(\gamma^t) \quad \text{as} \quad t\to\infty \quad \text{for any} \quad \gamma > \gamma_2(F_{\hat{P}}),$$

where $\pi_{\hat{P}}$ is the stationary distribution associated with $F_{\hat{P}}$, $\mathbf{1}$ is a vector of ones, $\gamma_2(F_{\hat{P}}) < 1$ is the modulus of the second largest eigenvalue of $F_{\hat{P}}$, and $|||\cdot|||$ denotes the matrix spectral norm (Bray,

2019b). Further, $F_a$ being a stochastic matrix implies $F_a\mathbf{1} = \mathbf{1}$, which in turn implies

$$\begin{aligned}\pi_{\hat{P}}\mathbf{1}'w_i &= -\sum_{a\in\mathbb{A}\setminus\{A\}} \beta\pi_{\hat{P}}\mathbf{1}'(F_a' - F_A')Z_{ia} \\ &= -\sum_{a\in\mathbb{A}\setminus\{A\}} \beta\pi_{\hat{P}}(\mathbf{1}' - \mathbf{1}')Z_{ia} \\ &= 0.\end{aligned}$$

Now if we define

$$\ell_i^t \equiv (F_{\hat{P}}')^t w_i = F_{\hat{P}}'\ell_i^{t-1}, \tag{15}$$

then the preceding two results establish that

$$\begin{aligned}\|\ell_i^t\| &= \|(F_{\hat{P}}')^t w_i\| \\ &= \|((F_{\hat{P}}')^t - \pi_{\hat{P}}\mathbf{1}')w_i\| \\ &\leq |||((F_{\hat{P}}')^t - \pi_{\hat{P}}\mathbf{1}')|||\|w_i\| \\ &= o(\gamma^t).\end{aligned} \tag{16}$$

This is noteworthy, because the dual vector corresponding to the weighting vector defined in (14) satisfies

$$\lambda_i = \sum_{t=0}^{\infty} \beta^t \ell_i^t.$$

With this, (16) indicates that

$$\begin{aligned}\left\|\lambda_i - \sum_{t=0}^{T-1} \beta^t \ell_i^t\right\| &= \left\|\sum_{t=T}^{\infty} \beta^t \ell_i^t\right\| \\ &\leq \sum_{t=T}^{\infty} \|\beta^t \ell_i^t\| \\ &= \sum_{t=T}^{\infty} o(\beta^t \gamma^t) \\ &= o\Big(\frac{\beta^T \gamma^T}{1-\beta\gamma}\Big) \\ &= o\Big(\frac{\gamma^T}{1-\gamma}\Big).\end{aligned}$$

It follows that

$$\left\|\lambda_i - \sum_{t=0}^{T-1} \beta^t \ell_i^t\right\| \lesssim \epsilon \quad \text{when} \quad T > \frac{\log(\epsilon) + \log(1-\gamma)}{\log(\gamma)}.$$

Thus, we can implement UFXP by iterating (15) to compute $\ell_i^0, \ldots, \ell_i^{T-1}$ and then saving these $T-1$ fixed vectors to quickly approximate $\lambda_i$ for a given value of $\beta$ with

$$\hat{\lambda}_i^{\beta} \equiv \sum_{t=0}^{T} \beta^t \ell_i^t.$$

The OUFXP estimator admits the same approximation.

In principle, one could bound $T$ with a mixing-time argument such as Dobrushin's condition; in practice, it should suffice to iterate (15) until $\|\ell_i^t\| < 10^{-9}$.

# C Unobserved Heterogeneity

Accommodating unobserved heterogeneity is straightforward, as our UFXP and OUFXP estimators are compatible with the two-stage procedure outlined in Arcidiacono and Miller (2011, Section 6). This approach first applies the EM algorithm to the reduced-form CCPs to compute an ex post probability of each observation that belongs to each heterogeneity group. The approach then estimates the structural parameters separately under each group-specific weighting scheme. Because the first-stage classification is performed in reduced form, it is both fast and agnostic to the estimator used in the second stage.

# D Numerical Experiment Details

The store's holding cost function is as follows:

$$\begin{aligned} h(r,o,i) \equiv & ci + (k_r^1 + k_o^2)\mathbb{1}\{i \geq \tau_r^1 + \tau_o^2\} \\ & + \omega_r \nu_o \Big(1 - \exp\Big(-\delta\big(i - (\tau_r^1 + \tau_o^2)\big)^+\Big)\Big) \\ & + (1-\omega_r)\Big(\alpha\Big(i - (\tau_r^1 + \tau_o^2)\Big)^+ + \psi_o\Big(\big(i - (\tau_r^1 + \tau_o^2)\big)^+\Big)^3\Big). \end{aligned}$$

This cost function depends on three state variables: $r \in \{1,2,3,4,5,6\}$ indexes the expectation of current daily demand at the beginning of the day, $o \in \{1,2,3\}$ indexes the amount of other-product inventory held at the beginning of the day, and $i \in \{0, \cdots, 29\}$ records the number of units of the focal product held at the beginning of the day. The holding cost combines a baseline linear cost, a step penalty, and a weighted combination of concave and convex tails.

As you can see above, the focal-product storage cost depends on the focal-product demand via the indices of $\tau^1 \equiv (0,1,2,3,4,5)$, $k^1 \equiv (0.0, 0.05, 0.10, 0.15, 0.20, 0.25)$, and $\omega \equiv (1.0, 0.8, 0.6, 0.4, 0.2, 0.0)$, and depends on the other-product inventory via the indices of $\tau^2 \equiv (14, 9, 4)$, $k^2 \equiv (0.4, 0.7, 1.0)$, $\nu \equiv (1.3, 1.1, 0.9)$, $\psi \equiv (7.5 \times 10^{-4}, 2.5 \times 10^{-4}, 10^{-4})$. Finally, we set $c \equiv 0.005$, $\delta \equiv 0.15$, and $\alpha \equiv 2 \times 10^{-5}$.

The action variable is order quantity $q \in \mathbb{Q} \equiv \{0, 6, 12, 18\}$, which the store manager chooses with the aim of maximizing expected discounted utility, under discount factor $\beta = 0.9997$. After $q$ is chosen, demand $d$ realizes from a Poisson distribution with mean $\mu_r$, where $\mu \equiv (4.5, 6, 7.5, 9, 10.5, 12)$. Next, the following day's state vector, $(r', o', i')$, resolves. The demand state $r$ follows one of two Markov chains, depending on whether the current focal inventory is below the threshold of 14 units:

$$\Pi_{\text{low}} = \begin{bmatrix} 0.95 & 0.05 & 0 & 0 & 0 & 0 \\ 0.25 & 0.70 & 0.05 & 0 & 0 & 0 \\ 0 & 0.25 & 0.70 & 0.05 & 0 & 0 \\ 0 & 0 & 0.05 & 0.70 & 0.25 & 0 \\ 0 & 0 & 0 & 0.05 & 0.70 & 0.25 \\ 0 & 0 & 0 & 0 & 0.05 & 0.95 \end{bmatrix}, \quad \Pi_{\text{high}} = \begin{bmatrix} 0.90 & 0.10 & 0 & 0 & 0 & 0 \\ 0.25 & 0.65 & 0.10 & 0 & 0 & 0 \\ 0 & 0.25 & 0.65 & 0.10 & 0 & 0 \\ 0 & 0 & 0.05 & 0.65 & 0.30 & 0 \\ 0 & 0 & 0 & 0.05 & 0.65 & 0.30 \\ 0 & 0 & 0 & 0 & 0.05 & 0.95 \end{bmatrix}$$

The congestion level $o'$ is set to $o$ with probability $p_o \equiv 0.9$, and is drawn uniformly from $\{1, 2, 3\}$ otherwise, and $i' = \min(29, (i - d)^+ + q)$ resolves from the classic inventory law of motion with a maximum inventory state cap. In summary, our model has the following transition kernel

$$
\begin{aligned}
f_q(r', o', i' | r, o, i) &\equiv f^r(r'|r, i) f^o(o'|o) f_q^i(i'|r, i), \\
\text{where} \quad f^r(r'|r, i) &\equiv \begin{cases} \Pi_{\text{low}}, & \text{if } i < 14 \\ \Pi_{\text{high}}, & \text{if } i \geq 14, \end{cases} \\
f^o(o'|o) &\equiv p_o \mathbb{1}\{o' = o\} + (1 - p_o)/3, \\
f_q^i(i'|r, i) &\equiv \sum_d f^d(d|r) \mathbb{1}\{i' = (i - d)^+ + q\}, \\
\text{and} \quad f^d(d|r) &\equiv \text{Poisson}(d|\mu_r).
\end{aligned}
$$

We derive $\hat{P}$ in three steps. First, we compute the empirical state-action frequency estimates, applying a Laplace smoothing factor of 0.1. Second, we compute another set of probabilities using a Gaussian kernel with bandwidth 1.0. This smooths the probabilities across inventory $i$ while holding other states fixed, weighting the neighboring estimates by their empirical observation counts. Finally, we compute a convex combination of the empirical frequency estimates and kernel-smoothed estimates. We weight the frequency estimates by $n_x/(n_x + s)$, where $n_x$ is the number of times state $x$ is observed, and $s$ is set to 2500. Thus, $\hat{P}$ relies more on the empirical frequencies where data is rich, and relies more on smoothed kernel estimates where data is sparse. We derive $\hat{f}_q$ from pre-estimates of $f^r$ and $f^o$ and $\mu$, the first two of which we glean from the empirical measures of $r'$ and $o'$ given $r$, $i$ and $o$, and the last of which we glean by maximizing the Poisson likelihood of $d$ given $r$.

To evaluate how estimation times scale with the size of the state space, we also construct a 5400-state version of this model. This larger model preserves the exact economic structure of the baseline model, but measures inventory in finer units, effectively scaling the inventory space by a factor of 10. Specifically, we expand the inventory state space to $i \in \{0, \ldots, 299\}$, scale the order quantities to $\mathbb{Q} \equiv \{0, 60, 120, 180\}$, and scale the Poisson demand means to $\mu \equiv (45, 60, 75, 90, 105, 120)$.

To keep the holding cost economically comparable across the two models, we adjust the parameters of $h(r, o, i)$. The inventory step thresholds scale up by a factor of 10: $\tau^1 \equiv (0, 10, 20, 30, 40, 50)$ and $\tau^2 \equiv (140, 90, 40)$. Conversely, to offset the larger $i$ values, the linear and exponential

cost coefficients scale down by a factor of 10 ($c \equiv 0.0005$, $\delta \equiv 0.015$, $\alpha \equiv 2 \times 10^{-6}$), and the cubic coefficient scales down by a factor of 1,000 to offset the cubed inventory terms: $\psi \equiv (7.5 \times 10^{-7}, 2.5 \times 10^{-7}, 10^{-7})$. Finally, the inventory threshold governing the demand state transition ($f^r$) shifts from 14 units to 149 units. All other parameters and probability distributions remain identical to the 540-state model.

# E Empirical Application Details

We estimate store demand means with a neural network. First, we winsorize sales at 12, and discard observations with fewer than 12 units of inventory at the beginning of the day. Dropping these observations mitigates demand censoring due to stock outs: demand is unlikely to exceed available inventory of 12, as only 4.8% of sales quantities exceed 12 units. Second, we derive nine sales features from the resulting sample: (i) month dummy variables; (ii) store dummy variables; (iii) item dummy variables; (iv) today's sales of the given item in the given store; (v) today's sales of the given item across all stores; (vi) today's sales of the given store across all items; (vii) tomorrow's retail price; (viii) tomorrow's wholesale price, and (ix) tomorrow's discount. Third, we input these features into a multilayer perceptron with two hidden layers of widths 64 and 32 and an output layer of width 1. Fourth, we exponentiate the neural network's output, and treat the result as the mean of a Poisson distribution, which we use to evaluate the empirical likelihood of tomorrow's sales quantity, given our nine input features. Finally, we find the model weights that yield the maximum likelihood, and define mean demand as the corresponding fitted values.

We estimate DC demand means analogously, using the sum of store orders as the DC's demand. But for this case, we need no censoring correction, as we observe both the DC's realized demand (orders) and satisfied demand (shipments), unlike for the stores, where we observe only the satisfied demand (sales). We model the DC demand with the same 64-to-32-to-1 neural network, but with a different set of inputs: (i) month dummy variables; (ii) item dummy variables; (iii) tomorrow's average retail price across stores; (iv) tomorrow's average wholesale price across stores, and (v) tomorrow's average discount across stores. As before, we train this model by maximizing the empirical likelihood of the DC's demand, assuming a Poisson distribution with a feature-varying mean. And, as before, we set the mean demands to the fitted values.

After estimating demand means for the stores and DC, we construct the state space with discretization. We set $\mathbb{Q}$ to the top four order quantities, which represent 94.2% of orders (the remaining 5.8% are scattered across 254 different values). Because the downstream demand has

Table 3: Ratios of average estimation times of benchmark estimators to average OUFXP estimation time.

| | | R=1 | | | | R=21 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NFXP | CCP | SC | MPEC | NFXP | CCP | SC | MPEC |
| 540 State Model | | | | | | | | | |
| Adam | ReLU Balanced | 1157 | 1477 | 214 | | 1707 | 2179 | 315 | |
| | ReLU Deep | 938 | 1280 | 196 | | 1224 | 1670 | 256 | |
| | ReLU Wide | 1329 | 1665 | 241 | | 1933 | 2423 | 351 | |
| | Softplus Balanced | 1035 | 1380 | 197 | | 1392 | 1858 | 264 | |
| | Softplus Deep | 867 | 1200 | 177 | | 1105 | 1529 | 225 | |
| | Softplus Wide | 1276 | 1761 | 265 | | 1897 | 2620 | 394 | |
| Knitro | ReLU Balanced | 591 | 755 | 244 | 3 | 654 | 836 | 270 | 3 |
| | ReLU Deep | 466 | 537 | 130 | 4 | 535 | 616 | 149 | 4 |
| | ReLU Wide | 681 | 869 | 279 | 4 | 734 | 937 | 300 | 5 |
| | Softplus Balanced | 2794 | 2905 | 467 | 14 | 4255 | 4425 | 711 | 21 |
| | Softplus Deep | 738 | 648 | 90 | 7 | 866 | 760 | 106 | 8 |
| | Softplus Wide | 4474 | 3348 | 420 | 22 | 9142 | 6841 | 859 | 44 |
| L-BFGS | ReLU Balanced | 264 | 313 | 44 | | 1303 | 1544 | 216 | |
| | ReLU Deep | 237 | 221 | 32 | | 1318 | 1228 | 179 | |
| | ReLU Wide | 366 | 465 | 54 | | 1491 | 1893 | 219 | |
| | Softplus Balanced | 667 | 822 | 104 | | 3131 | 3862 | 489 | |
| | Softplus Deep | 605 | 773 | 92 | | 2283 | 2917 | 348 | |
| | Softplus Wide | 866 | 994 | 122 | | 4128 | 4740 | 582 | |
| Trust | ReLU Balanced | 614 | 971 | 195 | | 798 | 1261 | 253 | |
| | ReLU Deep | 404 | 436 | 184 | | 499 | 538 | 227 | |
| | ReLU Wide | 729 | 881 | 174 | | 846 | 1023 | 202 | |
| | Softplus Balanced | 353 | 336 | 34 | | 737 | 700 | 70 | |
| | Softplus Deep | 402 | 388 | 38 | | 747 | 722 | 71 | |
| | Softplus Wide | 231 | 243 | 31 | | 495 | 519 | 67 | |
| 5400 State Model | | | | | | | | | |
| L-BFGS | Softplus Balanced | 162 | 166 | 102 | | 2891 | 2958 | 1824 | |
| Trust | Softplus Balanced | 159 | 155 | 35 | | 1912 | 1861 | 426 | |

*Note:* Values report the average estimation time of the benchmark estimator (numerator), divided by the average estimation time of OUFXP (denominator), for single ($R = 1$) and multiple ($R = 21$) optimization starts. We define estimation time as the total computational cost of obtaining the parameters $\hat{\theta}$. For benchmark estimators, this consists entirely of optimization time—the time required to search the parameter space for the $\theta$ that maximizes the objective. For OUFXP, however, estimation time is the sum of the initial UFXP estimation and the subsequent second-stage estimation. Each stage comprises two parts: a one-time fixed point time to solve the dual variables $\lambda_1, \ldots, \lambda_m$, and the subsequent optimization time. Accordingly, the average estimation time for OUFXP with $R$ optimization starts is the sum of the average stage-one (UFXP) and stage-two estimation times: (fixed point time$_1$ + fixed point time$_2$) + (average optimization time$_1$ + average optimization time$_2$)$R$, whereas the average estimation time for NFXP, CCP, SC, and MPEC with $R$ optimization starts is: (average optimization time)$R$, as their fixed points are nested within their optimization. All estimation times are truncated to seven days. This truncation did not increase any ratios, since all OUFXP runs completed within one week.

a right-skewed distribution, we group these values into five equal-width bins after applying a log transformation, and index them with $r$. We group $o$, $k$, and $\ell$ into tertiles based on their empirical distributions.

We allocate the finest grid to downstream focal-product inventory, since it is the most important variable for our purposes (nonparametrically estimating holding costs). We cap the observed inventories at the 90th percentile (149) and group them into 100 levels based on empirical quantiles.

We estimate the state transition kernel over the discretized space. This discretization necessitates a careful treatment of the inventory transition, $f^{\bar{i}}$, to ensure it accurately reflects the underlying law of motion: $i' = \min(\max \mathbb{I}, (i-d)^+ + j)$. We define $\mathbb{B}_{\bar{i}}$ as the set (bin) of exact inventory levels $i$ that map to the discrete state $\bar{i}$, and $\mathbb{B}_r$ as the set of mean demand values $s$ that map to the discrete demand state $r$.

While the model assumes demand $d$ follows a Poisson distribution for a given mean $s$, the effective distribution in state $r$ is a mixture across the empirical values $s \in \mathbb{B}_r$. We estimate the mixture probability mass function using the empirical frequencies. We use the resulting distribution, $\hat{f}^d(d|r)$, to compute both the expected probability of transitioning to a next-period state $\bar{i}'$, and the expected shortage:

$$
\begin{aligned}
f^{\bar{i}}(\bar{i}'|r,\bar{i},j) &\equiv \mathrm{E}_{\bar{i}}(\Pr(i' \in \mathbb{B}_{\bar{i}'}|r,i,j)) \\
\mathrm{E}((d-i)^+|r) &\equiv \mathrm{E}_{\bar{i}}(\sum_{d=0}^{\infty} \hat{f}^d(d|r)(d-i)^+) \\
\text{where} \quad \Pr(i' \in \mathbb{B}_{\bar{i}'}|r,i,j) &\equiv \sum_{d=0}^{\infty} \hat{f}^d(d|r) \mathbb{1}\{\min(\max \mathbb{I}, (i-d)^+ + j) \in \mathbb{B}_{\bar{i}'}\}.
\end{aligned}
$$

The expectation $\mathrm{E}_{\bar{i}}$ is taken with respect to the empirical frequency of exact inventory levels $i$ observed within the bin $\mathbb{B}_{\bar{i}}$.

# F Proofs

*Proof of Proposition 1.* Define $\mathbb{P}_a(c) \equiv \mathrm{E}\big(e_a | a = \arg\max_{j\in\mathbb{A}} e_j + c_j\big)$, in which case the social surplus can be expressed as follows:

$$\begin{aligned}
\nu(c) &= \mathrm{E}\Big(\max_{a\in\mathbb{A}} e_a + c_a\Big) \\
&= \mathrm{E}\Big(\sum_{a\in\mathbb{A}} \mathbb{1}\{a = \arg\max_{j\in\mathbb{A}} e_j + c_j\}(e_a + c_a)\Big) \\
&= \sum_{a\in\mathbb{A}} \mathrm{E}\big(\mathbb{1}\{a = \arg\max_{j\in\mathbb{A}} e_j + c_j\}(e_a + c_a)\big) \\
&= \sum_{a\in\mathbb{A}} \Pr\big(a = \arg\max_{j\in\mathbb{A}} e_j + c_j\big)\mathrm{E}\big(e_a + c_a | a = \arg\max_{j\in\mathbb{A}} e_j + c_j\big) \\
&= \rho(\Delta c)'(\mathbb{P}(c) + c).
\end{aligned}$$

And since $\Delta I^+ = I$, it follows that

$$\begin{aligned}
\epsilon(p) &= \nu(I^+\rho^{-1}(p)) - p'I^+\rho^{-1}(p) \\
&= \rho(\Delta I^+\rho^{-1}(p))'(\mathbb{P}(I^+\rho^{-1}(p)) + I^+\rho^{-1}(p)) - p'I^+\rho^{-1}(p) \\
&= p'(\mathbb{P}(I^+\rho^{-1}(p)) + I^+\rho^{-1}(p)) - p'I^+\rho^{-1}(p) \\
&= p'\mathbb{P}(I^+\rho^{-1}(p)).
\end{aligned}$$

Thus, choosing $v$ so that $p = \rho(\Delta v)$ yields the following:

$$\begin{aligned}
\epsilon(p) + p'c &= p'\mathbb{P}(I^+\rho^{-1}(p)) + p'c \\
&= \rho(\Delta v)'(\mathbb{P}(I^+\Delta v) + c) \\
&= \sum_{a\in\mathbb{A}} \Pr\left(a = \arg\max_{j\in\mathbb{A}} e_j + v_j - v_A\right)\Big(\mathrm{E}\big(e_a | a = \arg\max_{j\in\mathbb{A}} e_j + v_j - v_A\big) + c_a\Big) \\
&= \sum_{a\in\mathbb{A}} \Pr\left(a = \arg\max_{j\in\mathbb{A}} e_j + v_j\right)\Big(\mathrm{E}\big(e_a | a = \arg\max_{j\in\mathbb{A}} e_j + v_j\big) + c_a\Big) \\
&= \sum_{a\in\mathbb{A}} \mathrm{E}\Big(\mathbb{1}\{a = \arg\max_{j\in\mathbb{A}} e_j + v_j\}(e_a + c_a)\Big) \\
&= \mathrm{E}\Big(\sum_{a\in\mathbb{A}} \mathbb{1}\{a = \arg\max_{j\in\mathbb{A}} e_j + v_j\}(e_a + c_a)\Big) \\
&\leq \mathrm{E}\big(\max_{a\in\mathbb{A}} e_a + c_a\big) \\
&= \nu(c).
\end{aligned}$$

Further, the inequality in the penultimate line holds with equality if and only if $\arg\max_{a\in\mathbb{A}} e_a + v_a = \arg\max_{a\in\mathbb{A}} e_a + c_a$, which holds if and only if $\Delta v = \Delta c$. This proves that the objective is maximized at $p = \rho(\Delta c)$.

To establish the first-order conditions, consider the following Lagrangian:

$$\mathcal{L}(p, \lambda) \equiv \epsilon(p) + p'c + \lambda(p'\mathbf{1} - 1).$$

Setting the partial derivative with respect to $p$ to zero yields:

$$\begin{aligned}
0 &= \tfrac{\partial}{\partial p}\mathcal{L}(p, \lambda) \\
&= \nabla\nu(I^+\rho^{-1}(p))'\tfrac{\partial}{\partial p}\big(I^+\rho^{-1}(p)\big) - \tfrac{\partial}{\partial p}\big(p'I^+\rho^{-1}(p)\big) + c' + \lambda\mathbf{1}' \\
&= \rho(\rho^{-1}(p))'\tfrac{\partial}{\partial p}\big(I^+\rho^{-1}(p)\big) - (I^+\rho^{-1}(p))' - p'\tfrac{\partial}{\partial p}\big(I^+\rho^{-1}(p)\big) + c' + \lambda\mathbf{1}' \\
&= p'\tfrac{\partial}{\partial p}\big(I^+\rho^{-1}(p)\big) - (I^+\rho^{-1}(p))' - p'\tfrac{\partial}{\partial p}\big(I^+\rho^{-1}(p)\big) + c' + \lambda\mathbf{1}' \\
&= -(I^+\rho^{-1}(p))' + c' + \lambda\mathbf{1}'.
\end{aligned}$$

Finally, post-multiply the result above by $\Delta'$ yields the first-order conditions, since $\Delta\mathbf{1} = 0$ and $\Delta I^+ = I$. □

*Proof of Theorem 1.* Let $\theta_0$ denote the true parameter. We define the empirical objective function

and its population counterpart as follows:

$$\mathcal{Q}^{\theta}_{Z,N} \equiv \sum_{i=1}^{m} \text{tr}(Z_i'(\rho^{-1}(\hat{P}) - C^{\theta}_{V^{\theta}_{\hat{P}}}\Delta'))^2,$$
$$\text{and} \quad \mathcal{Q}^{\theta}_{Z,0} \equiv \sum_{i=1}^{m} \text{tr}(Z_i'(\rho^{-1}(P^{\theta_0}) - C^{\theta}_{V^{\theta}_{P^{\theta_0}}}\Delta'))^2,$$

where $\hat{P}$ is a consistent estimator of the true CCPs, $P^{\theta_0}$, estimated from a sample of $N$ observations. To invoke the extremum-estimator consistency theorem of Newey and McFadden (1994), we must establish the following four conditions:

(i) $\Theta$ is compact,

(ii) $\mathcal{Q}^{\theta}_{Z,0}$ is continuous,

(iii) $\sup_{\theta\in\Theta} |\mathcal{Q}^{\theta}_{Z,N} - \mathcal{Q}^{\theta}_{Z,0}| \xrightarrow{P} 0$,

(iv) $\theta_0$ is the unique minimizer of $\mathcal{Q}^{\theta}_{Z,0}$.

The first condition holds by the *compactness* assumption, and the second condition holds by the *smoothness* assumption.

For the third condition, it is sufficient to show $\rho^{-1}(\hat{P}) - C^{\theta}_{V^{\theta}_{\hat{P}}}\Delta'$ converges uniformly to $\rho^{-1}(P^{\theta_0}) - C^{\theta}_{V^{\theta}_{P^{\theta_0}}}\Delta'$. By the triangle inequality,

$$\sup_{\theta\in\Theta} \|(\rho^{-1}(\hat{P}) - C^{\theta}_{V^{\theta}_{\hat{P}}}\Delta') - (\rho^{-1}(P^{\theta_0}) - C^{\theta}_{V^{\theta}_{P^{\theta_0}}}\Delta')\| \leq \|\rho^{-1}(\hat{P}) - \rho^{-1}(P^{\theta_0})\| + \sup_{\theta\in\Theta} \|C^{\theta}_{V^{\theta}_{\hat{P}}}\Delta' - C^{\theta}_{V^{\theta}_{P^{\theta_0}}}\Delta'\|.$$

By the *error distribution regularity* assumption, the true CCPs lie in the interior of the probability simplex and $\rho$ is continuously differentiable. The inverse function theorem implies that $\rho^{-1}$ is continuously differentiable and therefore locally Lipschitz on the set of interior CCPs. Thus, it follows that $\|\rho^{-1}(\hat{P}) - \rho^{-1}(P^{\theta_0})\| \xrightarrow{P} 0$.

Moreover, the *smoothness* assumption ensures that $C^{\theta}_{V^{\theta}_{P}}$ is continuous with respect to both $P$ and $\theta$. Because $C^{\theta}_{V^{\theta}_{\hat{P}}}$ converges to $C^{\theta}_{V^{\theta}_{P^{\theta_0}}}$ pointwise and is continuous on a compact set by the *compactness* assumption, it converges uniformly over $\Theta$. Thus, the third condition follows.

For the fourth condition, note that the limit objective can be rewritten as $\mathcal{Q}^{\theta}_{Z,0} = \|Z'h_0(\theta)\|^2$, where $Z \in \mathbb{R}^{D\times m}$ is formed by concatenating vectors $\text{vec}(Z_i)$ for $i = 1, \ldots, m$, and $h_0(\theta) \equiv$

$\text{vec}(\rho^{-1}(P^{\theta_0}) - C^{\theta}_{V^{\theta}_{P^{\theta_0}}} \Delta')$, where $D = X(A-1)$. Because $h_0(\theta_0) = \mathbf{0}$, $\mathcal{Q}^{\theta_0}_{Z,0} = 0$. By the *identification* assumption, $h_0(\theta) = \mathbf{0}$ if and only if $\theta = \theta_0$. Thus, $\theta_0$ is the unique minimizer of $\mathcal{Q}^{\theta}_{Z,0}$ provided that $Z' h_0(\theta) \neq \mathbf{0}$ for all $\theta \neq \theta_0$. This holds almost surely by the following lemma.

**Lemma 1.** *For $m > t$, define the projection matrix $Z \in \mathbb{R}^{D \times m}$ to comprise* i.i.d. *entries from a distribution that is absolutely continuous with respect to Lebesgue measure. Then $Z$ satisfies* $\ker(Z') \cap h_0(\Theta) = \{\mathbf{0}\}$ *almost surely.*

*Proof.* We will use the following results:

**Theorem 3** (Theorem 6.30 in Lee (2012))**.** *Suppose $N$ and $M$ are smooth manifolds and $S \subseteq M$ is an embedded submanifold. If $F : N \to M$ is a smooth map that is transverse to $S$, then $F^{-1}(S)$ is an embedded submanifold of $N$ whose codimension is equal to the codimension of $S$ in $M$.*

**Theorem 4** (Theorem 6.35 (Parametric Transversality Theorem) in Lee (2012))**.** *Suppose $N$ and $M$ are smooth manifolds, $X \subseteq M$ is an embedded submanifold, and $\{F_s : s \in S\}$ is a smooth family of maps from $N$ to $M$. If the map $F : N \times S \to M$ is transverse to $X$, then for almost every $s \in S$, the map $F_s : N \to M$ is transverse to $X$.*

Mapping $h_0$ is a smooth, proper, injective immersion, by the *smoothness*, *identification*, and *full-rank* assumptions. Because $\Theta$ is compact, $h_0$ is also a proper map; therefore, the image space, $h_0(\Theta)$, is a smooth $t$-dimensional embedded submanifold of $\mathbb{R}^D$, by Theorem 3.4.1 of Shastri (2011). Let $M^0 \equiv h_0(\Theta) \setminus \{\mathbf{0}\}$. Since $M^0$ is open in $h_0(\Theta)$, $M^0$ is also a smooth manifold.

Let us define the functions $f_Z : M^0 \to \mathbb{R}^m$ and $F : M^0 \times \mathbb{R}^{D \times m} \to \mathbb{R}^m$ as $F(u, Z) = f_Z(u) = Z'u$, for $Z \in \mathbb{R}^{D \times m}$. It is straightforward that $F$ is a smooth map, and since restricting a smooth map to a smooth manifold preserves smoothness, $\{f_Z : Z \in \mathbb{R}^{D \times m}\}$ is a smooth family of maps $M^0 \to \mathbb{R}^m$.

Finally, we will show that $F$ is transverse to $\{\mathbf{0}\}$. Since $F$ is bilinear, its differential at $(u, Z)$ in the direction $(\delta u, \Delta Z) \in T_u M^0 \times \mathbb{R}^{D \times m}$ is

$$dF_{(u,Z)}(\delta u, \Delta Z) \;=\; Z'\,\delta u \;+\; \Delta Z'\,u \;\in\; \mathbb{R}^m.$$

Fix any $y \in \mathbb{R}^m$. Because $u \neq \mathbf{0}$, the linear map $\Delta Z \mapsto \Delta Z' u$ is surjective: choose

$$\Delta Z \;=\; \frac{u\,y'}{\|u\|^2} \;\in\; \mathbb{R}^{D \times m},$$

for which $\Delta Z'u = y$. Taking $\delta u = 0$ we get

$$dF_{(u,Z)}(0, \Delta Z) \;=\; \Delta Z'u \;=\; y.$$

Thus $dF_{(u,Z)}$ is onto $\mathbb{R}^m$ at every $(u, Z)$ with $u \in M^0$ and $F(u, Z) = \mathbf{0}$, so $F$ is transverse to $\{\mathbf{0}\} \subset \mathbb{R}^m$. Then, by Theorem 4, $f_Z$ is transverse to $\{\mathbf{0}\}$ almost surely, and by Theorem 3, the image space $f_Z^{-1}(\{\mathbf{0}\})$ has dimension $t-m$. Thus, for $m > t$, $f_Z^{-1}(\{\mathbf{0}\})$ is empty and $\ker(Z') \cap h_0(\Theta) = \{\mathbf{0}\}$ holds for almost every $Z$.

□

We will now establish the asymptotic normality of our estimator. First note that $N(x)\hat{P}(x)$ follows a multinomial distribution with parameters $N(x)$ and $P^{\theta_0}(x)$, thus $\sqrt{N(x)}(\hat{P}(x) - P^{\theta_0}(x)) \xrightarrow{d} \mathcal{N}(0, \Sigma(x))$ for any state $x \in \mathbb{X}$ by the multivariate central limit theorem.

Applying the delta method yields

$$\sqrt{N(x)}(\rho^{-1}(\hat{P}(x)) - C^{\theta_0}_{V^{\theta_0}_{\hat{P}}}(x)\Delta') \xrightarrow{d} \mathcal{N}(0, \Gamma(x)\Sigma(x)\Gamma(x)').$$

By the envelope theorem, the first-order effect of a change in the policy on the value function is zero at the optimal policy ($\frac{\partial}{\partial P} V_P\big|_{P=P^{\theta_0}} = \mathbf{0}$). Consequently, the derivative of the choice-specific value matrix with respect to $P(x)$ vanishes, and the Jacobian $\Gamma(x)$ simplifies to

$$\begin{aligned}\Gamma(x) &\equiv \tfrac{\partial}{\partial P(x)}\big(\rho^{-1}(P(x)) - C^{\theta_0}_{V^{\theta_0}_P}(x)\Delta'\big)\Big|_{P=P^{\theta_0}} \\ &= \tfrac{\partial \rho^{-1}}{\partial P(x)}(P^{\theta_0}(x)).\end{aligned}$$

The first-order conditions yield

$$\mathbf{0} = (\tfrac{\partial}{\partial \theta}\mathcal{Q}^{\theta_0}_{Z,N})' + \tfrac{\partial^2}{\partial\theta\partial\theta'}\mathcal{Q}^{\bar{\theta}}_{Z,N}(\hat{\theta}_{\text{UFXP}} - \theta_0),$$

where $\bar{\theta}$ lies on the line segment between $\hat{\theta}_{\text{UFXP}}$ and $\theta_0$. Note that $\bar{\theta} \xrightarrow{P} \theta_0$, and $\mathcal{Q}_{Z,N}$ is continuous, thus $\frac{\partial^2}{\partial\theta\partial\theta'}\mathcal{Q}^{\bar{\theta}}_{Z,N} \xrightarrow{P} \frac{\partial^2}{\partial\theta\partial\theta'}\mathcal{Q}^{\theta_0}_{Z,N}$. Moreover, $\hat{P} \xrightarrow{P} P^{\theta_0}$, thus $\operatorname{tr}(Z_i'(\rho^{-1}(\hat{P}) - C^{\theta_0}_{V^{\theta_0}_{\hat{P}}}\Delta')) \xrightarrow{P} 0$ for all $i = 1, \ldots, m$. Therefore,

$$\frac{\partial}{\partial\theta}\mathcal{Q}^{\theta_0}_{Z,N} = 2\sum_{i=1}^{m}\text{tr}(Z_i'(\rho^{-1}(\hat{P}) - C^{\theta_0}_{V^{\theta_0}_{\hat{P}}}\Delta'))(D^{\theta_0}_i + o_p(1)),$$

$$\frac{\partial^2}{\partial\theta\partial\theta'}\mathcal{Q}^{\bar{\theta}}_{Z,N} = 2\sum_{i=1}^{m}(D^{\theta_0}_i)'D^{\theta_0}_i + o_p(1),$$

$$\text{where}\quad D^{\theta_0}_i = \lambda_i'\sum_{a\in\mathbb{A}}\text{diag}(P^{\theta_0}_a)\frac{\partial}{\partial\theta}U^{\theta_0}_a - \sum_{a\in\mathbb{A}\setminus\{A\}}Z_{ia}'(\frac{\partial}{\partial\theta}U^{\theta_0}_a - \frac{\partial}{\partial\theta}U^{\theta_0}_A).$$

It follows that

$$\hat{\theta}_{\text{UFXP}} - \theta_0 = \left((D^{\theta_0})'D^{\theta_0}\right)^{-1}(D^{\theta_0})'\sum_{x\in\mathbb{X}}z(x)'(\rho^{-1}(\hat{P}(x)) - C^{\theta_0}_{V^{\theta_0}_{\hat{P}}}(x)\Delta') + o_p(\|\hat{\theta}_{\text{UFXP}} - \theta_0\|),$$

where $D^\theta$ is the $m \times t$ matrix formed by stacking the vectors $D^\theta_i$ for $i \in \{1,\ldots,m\}$, $z(x)'$ is the $m \times (A-1)$ matrix formed by stacking the $x$th row of $Z_i$ for $i \in \{1,\ldots,m\}$, and $C^\theta_V(x)$ is the $x$th row of $C^\theta_V$. Multiplying both sides by $\sqrt{N}$ and rewriting the scaling factor yields

$$\sqrt{N}(\hat{\theta}_{\text{UFXP}} - \theta_0) = \left((D^{\theta_0})'D^{\theta_0}\right)^{-1}(D^{\theta_0})' \sum_{x\in\mathbb{X}}\sqrt{\frac{N}{N(x)}}z(x)'\sqrt{N(x)}(\rho^{-1}(\hat{P}(x)) - C^{\theta_0}_{V^{\theta_0}_{\hat{P}}}(x)\Delta') + o_p(1).$$

Because observations across states are independent, the state-specific limit conditions $\sqrt{N(x)}(\rho^{-1}(\hat{P}(x)) - C^{\theta_0}_{V^{\theta_0}_{\hat{P}}}(x)\Delta')$ are asymptotically independent normal vectors. Since $N(x)/N \to \eta(x)$, Slutsky's theorem implies $\sqrt{N/N(x)} \xrightarrow{P} 1/\sqrt{\eta(x)}$. Therefore, the summation converges in distribution to a mean-zero normal random vector with variance

$$\sum_{x\in\mathbb{X}}\frac{1}{\eta(x)}z(x)'\Gamma(x)\Sigma(x)\Gamma(x)'z(x).$$

Pre-multiplying and post-multiplying this inner variance by the Jacobian sandwich matrices yields $\sqrt{N}(\hat{\theta}_{\text{UFXP}} - \theta_0) \xrightarrow{d} \mathcal{N}(0, \Sigma_{\text{UFXP}})$, completing the proof.

□

*Proof of Theorem 2.* We begin by characterizing the asymptotic variance of the OUFXP estimator. As $\hat{P} \xrightarrow{P} P^\theta$ and $\hat{\theta}_{\text{UFXP}} \xrightarrow{P} \theta$, the Continuous Mapping Theorem implies that the state-specific OUFXP weights converge in probability to:

$$\begin{aligned} z^\theta(x) &\equiv \Big(\frac{\Gamma(x)\Sigma(x)\Gamma(x)'}{\eta(x)}\Big)^{-1} J(x), \\ \text{where} \quad J(x) &\equiv \tfrac{\partial}{\partial\theta}(C^\theta_{V^\theta_{P^\theta}}(x)\Delta'). \end{aligned}$$

Substituting $z^\theta(x)$ yields

$$\begin{aligned} \sqrt{N}(\hat{\theta}_{\text{OUFXP}} - \theta) &\xrightarrow{d} \mathcal{N}(0, \Sigma_{\text{OUFXP}}), \\ \text{where} \quad \Sigma_{\text{OUFXP}} &\equiv \Big(\sum_{x\in\mathbb{X}} \eta(x) J(x)' \Big(\Gamma(x)\Sigma(x)\Gamma(x)'\Big)^{-1} J(x)\Big)^{-1}. \end{aligned}$$

The maximum likelihood estimator has the following standard limit:

$$\begin{aligned} \sqrt{N}(\hat{\theta}_{\text{MLE}} - \theta) &\xrightarrow{d} \mathcal{N}(0, \Sigma_{\text{MLE}}), \\ \text{where} \quad \Sigma_{\text{MLE}} &\equiv \Big(\lim_{N\to\infty} \tfrac{1}{N}\mathbb{E}\Big[\big(\tfrac{\partial}{\partial\theta}\mathcal{L}^\theta\big)'\big(\tfrac{\partial}{\partial\theta}\mathcal{L}^\theta\big)\Big]\Big)^{-1} \\ &= \Big(\sum_{x\in\mathbb{X}} \eta(x)\big(\tfrac{\partial}{\partial\theta}P^\theta(x)\big)' \operatorname{diag}(P^\theta(x))^{-1} \tfrac{\partial}{\partial\theta}P^\theta(x)\Big)^{-1} \\ &= \Big(\sum_{x\in\mathbb{X}} \eta(x) J(x)' \tfrac{\partial\rho}{\partial v}(C^\theta_{V^\theta}(x)\Delta')' \operatorname{diag}(P^\theta(x))^{-1} \tfrac{\partial\rho}{\partial v}(C^\theta_{V^\theta}(x)\Delta') J(x)\Big)^{-1}. \end{aligned}$$

To establish that $\Sigma_{\text{OUFXP}} = \Sigma_{\text{MLE}}$, what is left to show is

$$\Big(\tfrac{\partial\rho}{\partial v}(C^\theta_{V^\theta}(x)\Delta')' \operatorname{diag}(P^\theta(x))^{-1} \tfrac{\partial\rho}{\partial v}(C^\theta_{V^\theta}(x)\Delta')\Big)\Big(\Gamma(x)\Sigma(x)\Gamma(x)'\Big) = I_{A-1},$$

where $I_{A-1}$ is the $(A-1)\times(A-1)$ identity matrix. We will use the following three properties:

1. $\Gamma(x)\tfrac{\partial\rho}{\partial v}(C^\theta_{V^\theta}(x)\Delta') = I_{A-1}$.
   Because the mappings $\rho$ and $\rho^{-1}$ are continuously differentiable inverses of one another over the interior of the probability simplex, the Inverse Function Theorem implies that their Jacobians are inverses when evaluated at corresponding points. Since $P^\theta(x) = \rho(C^\theta_{V^\theta}(x)\Delta')$, the product of $\Gamma(x) \equiv \tfrac{\partial\rho^{-1}}{\partial P(x)}(P^\theta(x))$ and the Jacobian of $\rho$ is the $(A-1)\times(A-1)$ identity matrix.

2. $\tfrac{\partial\rho}{\partial v}(C^\theta_{V^\theta}(x)\Delta')\Gamma(x)\Sigma(x) = \Sigma(x)$.

The matrix product $\frac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')\Gamma(x)$ acts as the identity mapping for any vector residing in the tangent space of the probability simplex (i.e., any vector whose elements sum to zero). Because $\Sigma(x) \equiv \text{diag}(P^{\theta}(x)) - P^{\theta}(x)P^{\theta}(x)'$, the columns of $\Sigma(x)$ sum to zero ($\mathbf{1}'\Sigma(x) = P^{\theta}(x)' - P^{\theta}(x)' = \mathbf{0}$). Therefore, applying this operator to $\Sigma(x)$ returns $\Sigma(x)$.

3. $\frac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'\mathbf{1} = \mathbf{0}$.
The mapping $\rho$ inherently outputs a valid probability vector in $\mathbb{P}$, meaning its elements must always sum to 1. Consequently, any marginal change in the input values must induce changes in the choice probabilities that sum to exactly 0. Thus, the columns of the Jacobian $\frac{\partial \rho}{\partial v}$ sum to zero ($\mathbf{1}'\frac{\partial \rho}{\partial v} = \mathbf{0}$), and the transpose of this condition yields the property.

$$
\begin{aligned}
&\Big(\tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'\,\text{diag}(P^{\theta}(x))^{-1}\tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')\Big)\Big(\Gamma(x)\Sigma(x)\Gamma(x)'\Big)\\
&= \tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'\,\text{diag}(P^{\theta}(x))^{-1}\Sigma(x)\Gamma(x)'\\
&= \tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'\,\text{diag}(P^{\theta}(x))^{-1}\big(\,\text{diag}(P^{\theta}(x)) - P^{\theta}(x)P^{\theta}(x)'\big)\Gamma(x)'\\
&= \tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'(I - \text{diag}(P^{\theta}(x))^{-1}P^{\theta}(x)P^{\theta}(x)')\Gamma(x)'\\
&= \tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'\Gamma(x)' - \tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'\,\text{diag}(P^{\theta}(x))^{-1}P^{\theta}(x)P^{\theta}(x)'\Gamma(x)'\\
&= I_{A-1} - \tfrac{\partial \rho}{\partial v}(C^{\theta}_{V^{\theta}}(x)\Delta')'\mathbf{1}P^{\theta}(x)'\Gamma(x)'\\
&= I_{A-1}.
\end{aligned}
$$

The first equality follows from Property 2. The fifth equality follows from Property 1 and the identity $\text{diag}(P^{\theta}(x))^{-1}P^{\theta}(x) = \mathbf{1}$. The final equality follows from Property 3.

□