Best Practices for Differentiated Products Demand Estimation with `pyblp`

Christopher Conlon* Jeff Gortmaker†

February 18, 2019

Abstract

Differentiated products demand systems are a workhorse for understanding the price effects of mergers, the value of new goods, and the contribution of products to seller networks. Berry et al. (1995) provide a flexible workhorse model which accounts for the endogeneity of prices and is based on the random coefficients logit. While popular, there exists no standardized generic implementation for the Berry et al. (1995) estimator. Because the estimation algorithm involves a nested fixed point within a nonconvex optimization problem, estimation can be computationally challenging. This paper reviews and combines several recent advances related to the estimation of BLP-type problems and implements an extensible generic interface via the `pyblp` package.

To install `pyblp` if python is installed on your computer just type:

```
pip install pyblp
```

Up to date documentation for the package is available at `http://pyblp.readthedocs.io`.

*New York University, Stern School of Business: cconlon@stern.nyu.edu [Corresponding Author]
†New York Fed: jeff@jeffgortmaker.com

1 Introduction

Empirical of supply and demand for differentiated products are one of the most important achievements of the New Empirical Industrial Organization (NEIO) literature of the last thirty years. Perhaps the most influential paper in that literature is Berry et al. (1995). The authors provide an estimator which allows for flexible substitution patterns across products while also addressing the potential endogeneity of price. It is popular, because it addresses many of the unrealistic features of the previous literature such as fixed markups (as in CES demands), or representative agents (as Deaton and Muellbauer (1980)). It also has the advantage that it both scales well for large numbers of potential products, and can be applied to both aggregated and dis-aggregated data. The BLP model and its variants have been used in a wide variety of applications: understanding the value of new goods Petrin (2002), the price effects of mergers Nevo (2001) Nevo (2000a), and two-sided markets Lee (2013). The BLP approach has been applied to a wide number of different questions and industries including hospital demand and negotiations with insurers Ho and Pakes (2014) and students' choice of schools Bayer et al. (2007), and even asset pricing Koijen and Yogo (2018). Moreover, the BLP approach has been extremely influential in the practice of prospective merger evaluation, particularly in recent years.

The model itself is both quite simple to understand, and quite challenging to estimate. At its heart, the model involves a nonlinear change of variables from the space of observed marketshares to the space of mean utilities for products. After this nonlinear change of variables, the BLP problem is simply either a single linear instrumental variables regression problem, or a two equation (supply and demand) linear IV regression problem. This means that a wide variety of tools for that class of problems are available to researchers.

The main disadvantage of the BLP estimator is that it is a non-linear, non-convex optimization problem with a simulated objective function. This means that the problem must be solved iteratively using nonlinear optimization software, and because of the non-convexity, there is no mathematical guarantee that a solution will always be found. This has led to some frustration with the BLP approach (see Knittel and Metaxoglou (2014)). There is also the fear that when estimation is slow or complicated, researchers may cut corners in undesirable ways and sacrifice modeling richness for computational speed.

There is a growing recent literature which focuses on methodological innovations for BLP-type problems, or the econometric properties of the BLP estimator. For example, Knittel and Metaxoglou (2014) estimate the BLP model many times using the automobile data from Berry et al. (1999) data (with only a demand side specified), and the simulated fake cereal data from Nevo (2000b). The authors apply several different numerical optimization routines to the same data, and obtain a wide range of parameter estimate and implied elasticities. When we attempt replicate these difficulties with `pyblp`, our parameter estimates do not appear affected by our choice of optimization algorithm, and appear (for the most part) stable, we attribute this primarily to

numerical adjustments specific to our implementation.

Relatedly, Armstrong (2016) shows that absent exogenous cost shifting instruments, identification can be weak in these sorts of problems. Conlon (2017) shows that what appears to be a weak instrument problem, may actually be a many instrument problem in the language of Newey and Smith (2004). Recent work has further examined the role of instruments by Gandhi and Houde (2017) and Reynaert and Verboven (2014). The former construct *differentiation IV*, while the latter construct *optimal IV* in the sense of Amemiya (1977) or Chamberlain (1987).[1] We provide routines to construct both sets of instruments. Our results with respect to *optimal instruments* are broadly similar with those of Reynaert and Verboven (2014), in that the performance gains from the *optimal instruments* are substantial both in terms of bias and efficiency.

However, our simulations indicate, even absent strong cost-shifting instruments, it is possible to obtain unbiased (and relatively precise) parameter estimates when one includes a properly specified supply side. This supports the "folklore" around the original Berry et al. (1995) paper, that supply restrictions were a necessary component in pinning down parameter estimates. Our finding appears to provide a partial solution to the problem posed by Armstrong (2016), yet is somewhat different from the results presented in Reynaert and Verboven (2014) which suggest that once optimal demand side instruments are included, the addition of a supply side has limited benefit.

There is another strand of the literature which focuses on the role of simulation error in the BLP problem. Freyberger (2015) shows how to bias correct parameter estimates for simulation error. Judd and Skrainka (2011) and Heiss and Winschel (2008) consider a variety of integration techniques and suggest *sparse grids quadrature* as the best alternative. We implement a variety of routines: Pseudo Monte Carlo, Quasi Monte Carlo, Quadrature, and Sparse Grids Quadrature and our findings mirror those in Heiss and Winschel (2008).

Despite all of these methodological improvements, what is still lacking from this literature is a standardized implementation that is sufficiently general to encompass a wide range of potential problems and use cases. Instead, nearly every researcher implements the estimator on their own with problem-specific tweaks and adjustments. This makes replication extremely challenging, and also makes it hard to evaluate various different methodological and statistical improvements to the estimator.

The goal of this paper is to present best practices in the estimation of BLP-type models, some of which are well-known in the literature, others of which are lesser known, and other still are novel to this paper. In addition to presenting these best practices, we provide a common framework `pyblp` which offers a general implementation of the BLP approach in `Python 3`. This software is general, extensible, and open-source so that it can be modified and extended by scholars as new methodological improvements become available. The hope is that these best practices, along with this standardized and extensible software implementation reduce some of the barriers to BLP-type

[1] It is worth mentioning that the actual *optimal IV* are well-known to be infeasible. See Berry et al. (1995) or Berry et al. (1999).

estimation and make these techniques accessible to a wider range of researchers. In addition, we hope that by providing a common framework, this makes structural estimation of demand and supply easier to replicate for researchers and practicioners.

2 Model

Notation

j	products
t	markets
i	"individuals"
f	firms
J_t	set/number of products in market t
T	number of markets
θ_1	exogenous linear demand parameters
θ_2	endogenous (nonlinear) parameters (including price)
$\widetilde{\theta}_2$	endogenous (nonlinear) parameters
θ_3	exogenous linear supply parameters
K_1	$\dim(X_1)$
K_2	$\dim(X_2)$
K_3	$\dim(X_3)$
s_{jt}	the calculated market share of j in market t
$\mathcal{S}_{jt}$	the observed market share of j in market t
p_{jt}	the price of j in market t
δ_{jt}	the mean utility of j in market t
η_{jt}	the markup of j in market t
O	a $J_t \times J_t$ matrix of 0's and 1's where 1 denotes same owners
Ω	a $J_t \times J_t$ matrix of demand derivatives $\Omega_{(j,k)} = \frac{\partial q_j}{\partial p_k}$

2.1 Demand

Berry et al. (1995) begin with the following problem. An individual i in market $t = 1, \ldots, T$ receives indirect utility from selecting a particular product j:

$$u_{ijt} = \delta_{jt} + \mu_{ijt} + \epsilon_{ijt} \tag{1}$$

Consumers then choose among $J_t = \{0, 1, \ldots, J_t\}$ discrete alternatives and select exactly one option

which provides the most utility (including the outside alternative, denoted $j = 0$):

$$d_{ij} = \begin{cases} 1 \text{ if } u_{ijt} > u_{ij't} \text{ for } j \neq j' \\ 0 \text{ if otherwise} \end{cases} \tag{2}$$

Aggregate marketshares are given by integrating over heterogeneous consumer choices:

$$s_{jt}(\delta_{\cdot t}) = \int d_{ij}(\delta_{\cdot t}, \mu_{it})\, d\mu_{it} d\epsilon_{it}$$

When ϵ_{ijt} is distributed IID type I extreme value (Gumbel) this becomes:[2]

$$s_{jt}(\delta_{\cdot t}, \theta_2) = \int \frac{\exp[\delta_{jt} + \mu_{ijt}]}{\sum_{k \in J_t} \exp[\delta_{kt} + \mu_{ikt}]} f(\mu_{it}|\widetilde{\theta}_2) d\,\mu_{it} \tag{3}$$

This is often referred to as a *mixed logit* or *random coefficients logit* because each individual i's demands are given by a multinomial logit kernel, while $f(\mu_{it}|\widetilde{\theta}_2)$ denotes the *mixing distribution* over the heterogeneous types i and θ_2 parametrizes this heterogeneity. Indeed θ_2 contains all of the endogenous parameters of the model (the heterogenous tastes) as well as the endogenous parameter on price α.

The key insight is that we can perform a nonlinear change of variables (see Berry and Haile (2014)): $\delta_t = D_t^{-1}(\mathcal{S}_t, \widetilde{\theta}_2)$ (where $\mathcal{S}_t$ denotes the J_t vector of *observed marketshares*). For each market t, equation (3) represents a $J_t \times J_t$ system of equations in $\delta_{\cdot t}$. Given the $\delta_{\cdot t}(\mathcal{S}_t, \widetilde{\theta}_2)$ which solves that system of equations; along with some covariates: x_{jt}, prices: p_{jt}, and a structural error: ξ_{jt}; under an additivity assumption, one can write the index:

$$\delta_{jt}(\mathcal{S}_t, \widetilde{\theta}_2) = x_{jt}\beta - \alpha p_{jt} + \xi_{jt} \tag{4}$$

With the addition of some instruments Z_{jt}^D (including the exogenous regressors x_{jt}), one can form moment restrictions conditions of the form $E[\xi_{jt}' Z_{jt}^D] = 0$.

2.2 Supply

We can derive an additional set of supply moments from the multi-product differentiated Bertrand first order conditions. Consider the profits of firm f which controls several products $\mathcal{J}_f$ and sets

[2]Identification generally requires normalizing one of the option. Often the outside option $u_{i0t} = 0$ is the preferred normalization.

prices p_j:

$$\begin{aligned}\arg\max_{p_j \in \mathcal{J}_f} \sum_{j \in \mathcal{J}_f} (p_j - c_j) \cdot s_j(\mathbf{p}) \\ s_j(\mathbf{p}) + \sum_{k \in \mathcal{J}_f} \frac{\partial s_k}{\partial p_j}(\mathbf{p}) \cdot (p_k - c_k) \quad &= \quad 0\end{aligned}$$

It is helpful to write the FOC in matrix form so that for a single market t:

$$\begin{aligned}s(\mathbf{p}) &= (O \odot \Omega(\mathbf{p})) \cdot (\mathbf{p} - \mathbf{mc}) \\ (\mathbf{p} - \mathbf{mc}) &= \underbrace{(O \odot \Omega(\mathbf{p}))^{-1} s(\mathbf{p})}_{\eta}\end{aligned} \tag{5}$$

Here the multi-product Bertrand markup η depends on the element-wise (Hadamard) product $\odot$ of two $J_t \times J_t$ matrices: the matrix of demand derivatives $\Omega(\mathbf{p})$ where each (j,k) entry is given by is $\frac{\partial q_j}{\partial p_k}$, and the ownership matrix O:[3]

$$O_{(j,k)} = \left\{ \begin{array}{ll} 1 & \text{for } (j,k) \in \mathcal{J}_f \text{ for any } f \\ 0 & \text{o.w} \end{array} \right\}$$

This enables us to recover an estimate of $mc_{jt} = p_{jt} - \eta_{jt}$, which in turn allows us to construct additional *supply side moments*. We can parametrize marginal cost as:[4]

$$f(p_{jt} - \eta_{jt}) = f(mc_{jt}) = x_{jt}\gamma_1 + w_{jt}\gamma_2 + \omega_{jt} \tag{6}$$

and construct moment conditions of the form $E[\omega'_{jt} Z^S_{jt}] = 0$. The idea is that we can use observed prices, along with information on demand derivatives $\Omega_t(\theta_2, \mathbf{p_t})$ and firm conduct to recover markups η_{jt} and then marginal costs mc_{jt}. This also imposes a functional form restriction on marginal costs, which depends on both product characteristics x_{jt} and the marginal cost shifters w_{jt} which are excluded from demand.

2.3 The Estimator

We can construct a GMM estimator using our supply and demand moments. Our demand moment restrictions have sample analogue: $g^d(\theta) = \frac{1}{N} \sum_{j,t} \xi_{jt} z^D_{jt}$ and our supply moment restrictions have

[3] We can easily consider alternative forms of conduct: e.g Single Product Oligopoly, Multiproduct Oligopoly, Multiproduct Monopoly. Miller and Weinberg (2017) consider estimating a single parameter $O(\kappa)$ and Backus et al. (2018) test various forms of $O(\kappa)$.

[4] The most common choice of $f(\cdot)$ is the identity function, but some authors also consider $f(\cdot) = \log(\cdot)$ so that

$$\log(p_{jt} - \eta_{jt}) \equiv \log(mc_{jt}) = x_{jt}\gamma_1 + w_{jt}\gamma_2 + \omega_{jt}$$

In practice this constrains marginal costs to be always positive.

sample analogue $g^s(\theta) = \frac{1}{N}\sum_{j,t}\omega_{jt}z^S_{jt}$.[5] We can stack both sets of moments up to form $g(\theta) = \begin{bmatrix} g^d(\theta) \\ g^s(\theta) \end{bmatrix}$ and construct a nonlinear GMM estimator for $\theta = [\beta, \alpha, \widetilde{\theta}_2, \gamma]$ with some weighting matrix W:

$$\min_{\theta} \qquad q(\theta) \equiv g(\theta)' W g(\theta) \tag{7}$$

For clarity, we partition the parameter space θ into three parts: $\theta_1 : (K_1 \times 1)$ contains β the exogenous demand parameters; $\theta_3 : (K_3 \times 1)$ contains γ the exogenous supply parameters; and the remaining (endogenous) parameters are contained in $\theta_2 : (K_2 \times 1)$, including the price coefficient α. The endogenous parameters are endogenous in the literal sense that they appear in both equations for demand and supply.

To be explicit we write the entire program as follows:[6]

$$\begin{aligned}
\min_{\theta} \qquad & q(\theta) \equiv g(\theta)' W g(\theta) \\
g(\theta) &= \begin{bmatrix} Z^{D\prime}\xi \\ Z^{S\prime}\omega \end{bmatrix} / N \\
\xi_{jt} &= \delta_{jt} - x_{jt}\beta + \alpha p_{jt} \\
\omega_{jt} &= f(p_{jt} - \eta_{jt}) - x_{jt}\gamma_1 - w_{jt}\gamma_2 \\
\eta_{\cdot t} &= (O \odot \Omega(\mathbf{p_t}, \widetilde{\theta}_2))^{-1}\mathbf{q_t} \\
\mathcal{S}_{jt} = s_{jt}(\delta, \theta_2) &\equiv \int \frac{\exp[\delta_{jt} + \mu_{ijt}]}{\sum_{k \in J_t}\exp[\delta_{kt} + \mu_{ikt}]} f(\mu_{it}|\widetilde{\theta}_2) d\mu_{it}
\end{aligned} \tag{8}$$

This estimator and its econometric properties are discussed in Berry et al. (1995) and Berry et al. (2004). Our focus is not going to be on the econometric properties of $\hat{\theta}$ but rather various algorithms by which one might obtain $\hat{\theta}$. Technically, we need to solve this program twice. Once to obtain a consistent estimate for $\hat{W}(\hat{\theta})$ and a second time to obtain the efficient GMM estimator.

2.4 The Nested Fixed Point Algorithm of Berry et al. (1995)

In addition to providing an estimator Berry et al. (1995) provide an algorithm to solving (8) which attempts to simplify the problem. One can concentrate out $[\theta_1, \theta_3]$ and perform a nonlinear search over just θ_2. Our approach differs somewhat from the previous literature and we provide details in Appendix A.1. The intuition is that the exogenous parameters enter the transformed problem linearly, and thus one can construct $\widehat{\theta}_1(\theta_2), \widehat{\theta}_3(\theta_2)$. This works because the markup term can be

[5] Here $N = \sum_t \dim(\mathcal{J}_t)$.

[6] It is helpful to define $N = \sum_t J_t$ (the overall number of observations for all products in all markets).

written as a function of only the θ_2 parameters: $\eta_{jt}(\mathbf{q_t}, \mathbf{p_t}, \delta_{\cdot t}, \theta_2)$ and does not depend on $[\theta_1, \theta_3]$.[7]

In this case parameters are explicit functions of other parameters and solved for in a particular order. For each guess of θ_2:

(a) For each market t: solve $\mathcal{S}_{jt} = s_{jt}(\delta_{\cdot t}, \theta_2)$ for $\widehat{\delta}_{\cdot t}(\mathcal{S}_{jt}, \theta_2)$.

(b) For each market t: use $\widehat{\delta}_{\cdot t}(\mathcal{S}_{jt}, \theta_2)$ to construct $\eta_{\cdot t}(\mathbf{q_t}, \mathbf{p_t}, \widehat{\delta}_{\cdot t}(\theta_2), \theta_2)$

(c) For each market t: Recover $\widehat{mc}_{jt}(\widehat{\delta}_{\cdot t}(\theta_2), \theta_2) = p_{jt} - \eta_{jt}(\widehat{\delta}_{\cdot t}(\theta_2), \theta_2)$

(d) Stack up $\widehat{\delta}_{\cdot t}(\mathcal{S}_{jt}, \theta_2)$ and $\widehat{mc}_{jt}(\widehat{\delta}_{\cdot t}(\theta_2), \theta_2)$ and use linear IV-GMM to recover $[\widehat{\theta}_1(\theta_2), \widehat{\theta}_3(\theta_2)]$ following the recipe in Appendix A.1. *Note: this is different from the typical formulation so that we can absorb fixed effects.*

$$\begin{aligned} \widehat{\delta}_{jt}(\mathcal{S}_{jt}, \theta_2) + \alpha p_{jt} &= x_{jt}'\beta + \xi_{jt} \\ \widehat{mc}_{jt}(\theta_2) &= (x_{jt} \;\; w_{jt})'\gamma + \omega_{jt} \end{aligned}$$

(e) Construct the residuals:

$$\begin{aligned} \widehat{\xi}_{jt}(\theta_2) &= \widehat{\delta}_{jt}(\theta_2) - x_{jt}\widehat{\beta}(\theta_2) + \alpha p_{jt} \\ \widehat{\omega}_{jt}(\theta_2) &= \widehat{mc}_{jt}(\theta_2) - [x_{jt}\, w_{jt}]\widehat{\gamma}(\theta_2) \end{aligned} \tag{9}$$

(f) Construct sample moments

$$\begin{aligned} g_n^D(\theta_2) &= \frac{1}{N}\sum_{jt} Z_{jt}^{D\prime}\widehat{\xi}_{jt}(\theta_2) \\ g_n^S(\theta_2) &= \frac{1}{N}\sum_{jt} Z_{jt}^{S\prime}\widehat{\omega}_{jt}(\theta_2) \end{aligned} \tag{10}$$

(g) Construct GMM objective $Q_n(\theta_2) = \underbrace{\begin{bmatrix} g_n^d(\theta_2) \\ g_n^s(\theta_2) \end{bmatrix}'}_{g_n(\theta_2)} W \underbrace{\begin{bmatrix} g_n^d(\theta_2) \\ g_n^s(\theta_2) \end{bmatrix}}_{g_n(\theta_2)}$

We provide analytic gradients for the BLP problem with supply and demand in A.2. One advantage of the BLP algorithm is that is performs a nonlinear search over only K_2 *nonlinear parameters*. Consequently, the Hessian matrix is only $K_2 \times K_2$. This implies relatively minimal memory requirements. Also, the IV-GMM regression in step (d) concentrates out the *linear* parameters $[\theta_1, \theta_3]$. This implies that large numbers of linear parameters β are essentially for free which is important

[7] Why? We already know we can invert the shares to solve for $\delta(\theta_2)$. Because Ω_t, the matrix of demand derivatives $\frac{\partial q_{kt}}{\partial p_{jt}} = -\int \alpha_i \left[s_{ikt} \cdot I[j=k] - s_{ikt}s_{ijt}\right] f(\mu_{it}, \alpha_i|\theta_2)$ again depends only on θ_2 (which contains the price coefficient α) and θ_2.

if one includes a large number of fixed effects such as product or market level fixed effects.[8] In fact, other than (a) the remaining steps are computationally trivial. As is well known (a)-(c) can be performed separately for each market across multiple processors.

The main disadvantage is that all parameters are implicit functions of other parameters, particularly of θ_2. The objective is now a complicated implicit function of parameters θ_2. Once we incorporate any heterogeneous tastes, the resulting optimization problem is non-convex. In general the complexity of this problem grows rapidly with the number of nonlinear θ_2 parameters, while (θ_1, θ_3) parameters are more or less for free.

2.5 Nested Logit and RCNL Variants

The RCNL model of Brenkers and Verboven (2006) instead places a GEV structure on ϵ_{ijt}. This model is popular in applications where the most important set of characteristics governing substitution is categorical. This has made it popular in alcoholic beverages: distilled spirits Conlon and Rao (2017), Miravete et al. (2018) and beer: Miller and Weinberg (2017).

Much like the random coefficients model integrates out over a heterogeneous distribution where each individual type follows a logit distribution. The RCNL model integrates out over a heterogenous distribution where each individual now follows a *nested logit distribution*. We expand our definition of the endogenous parameters θ_2 to include the nesting parameter ρ.

$$u_{ijt} = \delta_{jt}(\theta_1) + \mu_{ijt}(\theta_2) + \varepsilon_{ijt}(\rho)$$

The primary difference from the nested logit is that the inclusive value term for all products in nest J_{gt} now depends on the consumer's type i:

$$s_{jt}(\delta_{jt}(\theta), \theta) = \int \frac{\exp[(\delta_{jt} + \mu_{ijt}(\widetilde{\theta}_2))/(1-\rho)]}{\exp[I_{ig}/(1-\rho)]} \cdot \frac{\exp[I_{ig}]}{\exp[IV_i]} f(\mu_{ijt}|\widetilde{\theta}_2) \partial\widetilde{\theta}_2 \tag{11}$$

$$I_{ig}(\theta_1, \theta_2) = (1-\rho) \ln \sum_{j \in J_{gt}} \exp\left(\frac{\delta_{jt} + \mu_{ijt}(\widetilde{\theta}_2)}{1-\rho}\right), \quad IV_i(\theta_1, \theta_2) = \ln\left(1 + \sum_{g=1}^{G} \exp I_{ig}\right)$$

The challenge for estimation is that $\log \delta_{\cdot t} \leftarrow \ln \delta_{\cdot t} + \ln \mathcal{S}_{\cdot t} - \ln \mathbf{s_t}(\delta_{\cdot t}, \widetilde{\theta}_2)$ is no longer a contraction. Instead, the contraction must be dampened so that:[9]

$$\log \delta_{\cdot t} \leftarrow \ln \delta_{\cdot t} + (1-\rho)\left[\ln \mathcal{S}_{\cdot t} - \ln \mathbf{s_t}(\delta_{\cdot t}, \widetilde{\theta}_2)\right] \tag{12}$$

This creates an additional challenge where the rate of convergence for the contraction in (12) can

[8] For example Nevo (2001) or Nevo (2000b) includes product fixed effects.

[9] See Grigolon and Verboven (2014) for a derivation. The expression in (11) does not precisely match Grigolon and Verboven (2014) because of a typographical error in their final expression.

become arbitrarily slow as $\rho \to 1$.[10] Thus as more consumers substitute within the nest, this model becomes much harder to estimate. Our simulations will demonstrate that this can be quite problematic.

As is well known, the relation in (12) has an analytic solution in the absence of random coefficients: $\mu_{ijt} = 0$ for all (i, j, t). This model reduces to the regular nested logit. This expression was derived in Berry (1994):

$$\ln \delta_{jt} \equiv \ln \mathcal{S}_{jt} - \ln \mathcal{S}_{0t} - \rho \ln \mathcal{S}_{j|gt}$$

3 Algorithmic Improvements in `pyblp`

3.1 Solving for the Shares

The main challenge of the Nested Fixed Point (NFXP) algorithm is solving the system of market-shares: $\mathcal{S}_{jt} = s_{jt}(\delta, \theta_2)$. In the NFXP approach, θ_2 is treated as fixed, which has the immediate implication that rather than solving a system of N nonlinear equations and N unknowns δ, we solve T systems of J_t equations and J_t unknowns $\delta_{\cdot t}$. Independent of how we solve each system of equations, we can solve each market t in parallel.[11]

Consider a single market t, where we search for the J_t vector $\delta_{\cdot t}$ which satisfies:

$$\mathcal{S}_{jt} = s_{jt}(\delta_{\cdot t}|\theta_2) \quad = \quad \int \frac{\exp[\delta_{jt} + \mu_{ijt}]}{\sum_{k \in J_t} \exp[\delta_{kt} + \mu_{ikt}]} f(\mu_{it}|\widetilde{\theta}_2) d\,\mu_{it} \tag{13}$$

It is impossible to choose a vector $\delta_{\cdot t}$ which solves (13) exactly. Instead, we must solve the system of equations to some tolerance. We express the tolerance in terms of the log difference in shares:

$$\left\| \ln \mathcal{S}_{jt} - \ln s_{jt}(\delta_{\cdot t}; \theta_2) \right\|_\infty \leq \epsilon^{tol} \tag{14}$$

If the tolerance is too loose, the numerical error propagates to the rest of the estimation routine, and the tolerance should be chosen as small as possible.[12] It is also possible to set a tolerance which is too tight and thus can never be satisfied, this is particularly problematic when we sum over a large number of elements. We prefer $\epsilon^{tol} \in [10^{-12}, 10^{-14}]$ as the *machine epsilon* for 64-bit doubles is generally $\epsilon^{mach} \in [10^{-16}, 10^{-15}]$.[13]

[10] This can be formalized in terms of the modulus of the contraction mapping or the Lipschitz constant. See Dubé et al. (2012) for more details.

[11] This is the same idea that provides the sparsity of the share constraints in the MPEC approach.

[12] Tolerance issues abound in Knittel and Metaxoglou (2014). Dubé et al. (2012) show how error propagates from the solution of (13) to the estimates of $\widehat{\theta}$. Lee and Seo (2016) provide a more precise characterization of this error when Newton's method is used.

[13] Choosing a smaller tolerance may be impossible for the machine to satisfy when the scaling of the problem changes, and choosing a larger tolerance can lead to numerical errors.

Newton Approaches

A direct approach would be to solve the system of J_t equations and J_t unknowns using Newton-Type methods. Consider Newton-Raphson iteration below:[14]

$$\delta_{\cdot t}^{h+1} \leftarrow \delta_{\cdot t}^{h} - \lambda J_{\mathbf{s}}^{-1}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2) \cdot \mathbf{s_t}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2) \tag{15}$$

Each Newton-Raphson iteration would require computation of both the J_t vector of marketshares $\mathbf{s_t}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2)$, the $J_t \times J_t$ Jacobian matrix $J_{\mathbf{s}}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2) = \frac{\partial \mathbf{s_t}}{\partial \delta_{\cdot t}}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2)$, as well as its inverse $J_{\mathbf{s}}^{-1}(\delta_{\cdot t}^{h})$. As in all iterative solution methods there are two primary costs: (a) the cost per iteration; (b) the number of iterations until convergence. The costs per iteration are driven primarily by the cost of computing (rather than inverting) the Jacobian matrix which involves $J_t \times J_t$ numerical integrals.[15] The algorithm in (15) is quadratically convergent. This tells us that convergence will be rapid for some $\delta_{\cdot t}^{k}$ within some *basin of attraction*, but characterizing the basin of attraction is extremely difficult.

There are some alternative *Quasi-Newton* methods which solve variants of (15). These variants generally involve modifying the step-size λ or approximating $J_{\mathbf{s}}^{-1}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2)$ in ways that avoid calculating the inverse Jacobian at each step such as *Powell's Method*. *Broyden's Method* avoids calculating the Jacobian at all, and instead constructs an approximation.[16] Quasi-Newton methods are often relatively fast when they work, though they need not converge (they may oscilate, reach a resting point, etc.) and may be sensitive to starting values. Though the BLP problem is a non-convex system of nonlinear equations, there are some features which make it amenable Quasi-Newton methods. The marketshare function is a real-analytic function of $\delta_{\cdot t}$, in other words: $s_{jt}(\delta_{\cdot,t}|\widetilde{\theta}_2)$ is $\mathbb{C}^\infty$ and bounded between $[0,1]$ and thus it agrees with its Taylor approximation at any $\delta_{\cdot t}$. This guarantees that at least within some *basin of attraction* Quasi-Newton methods will be quadratically convergent.[17] The other useful property is that $J_{\mathbf{s}}(\delta_{\cdot t}, \widetilde{\theta}_2)$ is strictly diagonally dominant, which guarantees that it is always non-sinsgular.

Among the Quasi-Newton methods considered, we find that *Levenberg-Marquardt*, which minimizes (15) in a least-squares sense using an exact version of the analytic Jacobian provides the fastest and most reliable alternative. We provide details in Appendix A.3

Fixed Point Approaches

Berry et al. (1995) also propose a fixed-point approach to solve the $J_t \times J_t$ system of equations in

[14] In practice it is generally faster to solve the linear system: $J_{\mathbf{s}}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2)\left[\delta_{t}^{h+1} - \delta_{t}^{h}\right] = -\mathbf{s_t}(\delta_{\cdot t}^{h}, \theta_2)$.

[15] A typical entry is $J_{jk} = \int \left(-s_{ijt}(\mu_{it})s_{ikt}(\mu_{it}) + I[j \neq k]s_{ijt}(\mu_{it})\right) f(\mu_{it}|\widetilde{\theta}_2)\partial\mu_{it}$. The primary cost arises from the numerical integration over heterogeneous types. Even for a large market with $J_t = 1000$ products, inverting a 1000×1000 matrix is relatively easy (relative to J_t^2 numerical integrals).

[16] Reynaerts et al. (2012) show that Broyden's method is outperformed by standard Newton-Raphson root finding methods for the BLP problem.

[17] For an example of a Quasi-Newton solution to (15) see Houde (2012).

(13). They show that the following is a contraction mapping $f(\delta) = \delta$:

$$f : \delta_{\cdot t}^{h+1} \leftarrow \delta_{\cdot t}^{h} + \ln \mathcal{S}_{\cdot t} - \ln \mathbf{s}_{\cdot \mathbf{t}}(\delta_{\cdot t}^{h}, \widetilde{\theta}_2) \tag{16}$$

This kind of contraction mapping is linearly convergent where the rate of convergence is proportional to $\frac{L(\widetilde{\theta}_2)}{1-L(\widetilde{\theta}_2)}$ where $L(\widetilde{\theta}_2)$ is the Lipschitz constant. Because (16) is a contraction, we know that $L(\widetilde{\theta}_2) < 1$. Dubé et al. (2012) show that for the BLP contraction the Lipschitz constant is defined as $L(\widetilde{\theta}_2) = \max_{\delta \in \Delta} \left\| \mathbf{I}_{J_t} - \frac{\partial \log \mathbf{s}_{\cdot \mathbf{t}}}{\partial \delta_{\cdot t}}(\delta_{\cdot t}, \widetilde{\theta}_2) \right\|_\infty$.

A smaller Lipschitz constant implies that (16) converges more rapidly. Dubé et al. (2012) show in simulation that all else being equal, a larger outside good share generally implies a smaller Lipschitz constant.[18] Conversely, as the outside good share becomes smaller, the convergence of the fixed point relationship takes increasingly many steps.

Accelerated Fixed Points

Given a fixed point relationship there may be faster ways to obtain a solution to $f(\delta) = \delta$ than direct iteration on the fixed point relation as in (16). There is a large literature on acceleration methods for fixed points. Most of these methods use information from mutliple iterations $(\delta^h, \delta^{h+1}, \delta^{h+2}, f(\delta^h), f(f(\delta^h)))$ to approximate $J_{\mathbf{s}}$ or $J_{\mathbf{s}}^{-1}$.[19]

Reynaerts et al. (2012) conduct extensive testing of various fixed point acceleration methods and find that the `SQUAREM` algorithm of Varadhan and Roland (2008) works well on the BLP contraction in (16). That algorithm is described below:[20]

$$\begin{aligned} \delta_{\cdot t}^{h+1} &= \delta_{\cdot t}^{h} - 2\alpha^h r^h + (\alpha^h)^2 v^h, \quad \alpha^h = \frac{(v^h)' r^h}{(v^h)' v^h} \\ r^h &= f(\delta_{\cdot t}^{h}) - \delta_{\cdot t}^{h}, \quad v^h = f(f(\delta_{\cdot t}^{h})) - 2f(\delta_{\cdot t}^{h}) + \delta_{\cdot t}^{h} \end{aligned} \tag{17}$$

In general the `SQUAREM` method is $4-8\times$ as fast as direct iteration on the BLP contraction in (16). The idea is to take roughly the same number of steps as in the Newton-Raphson iteration, but to have much less expensive steps as we calculating the Jacobian directly. In fact, all of the terms in ((17)) are computed as a matter of course, because these are just iterations of x^h and $f(x^h)$. Unlike direct iteration on ((16)), there is technically no convergence guarantee as the iteration on ((17)) is no longer a contraction.

The are alternative acceleration methods in addition to `SQUAREM`. Reynaerts et al. (2012) also consider `DF-SANE` which takes on the form: $\delta_{\cdot t}^{h+1} \leftarrow \delta_{\cdot t}^{h} - \alpha^h f(\delta_{\cdot t}^{h})$ with a different choice of the

[18] A simple but novel derivation. Consider an element (j,k) of $\frac{\partial \log \mathbf{s}_{\cdot \mathbf{t}}}{\partial \delta_{\cdot t}}$: $\mathcal{I}[j = k] - \frac{1}{s_{jt}} \int s_{ijt}(\mu_{it}) \cdot s_{ikt}(\mu_{it}) f(\mu_{it}|\widetilde{\theta}_2) \partial \mu_{it} = \mathcal{I}[j = k] - \text{diag}(1/s_j) \cdot \Gamma(\widetilde{\theta}_2)$. This implies that $L(\widetilde{\theta}_2) = \max_\delta \left[\max_j \frac{1}{s_j} \sum_{k=1}^{K} |\Gamma_{jk}(\delta_{\cdot t}, \widetilde{\theta}_2)| \right]$. A rough approximation to the term inside the square braces: $\max_j \sum_{k=1}^{K} |s_{kt}| \cdot |\rho(s_{ijt}, s_{ikt})| < 1 - s_{0t}$.

[19] Many of these algorithms are special cases of *Anderson Mixing*.

[20] `pyblp` includes a port of the `SQUAREM` package from `R`.

step-size α^h. They find performance is similar to `SQUAREM` though it can be slightly slower and less robust. Consistent with Reynaerts et al. (2012), we find the accelerated fixed point approaches (`SQUAREM` in particular) seem to best balance reducing the number of fixed point iterations with increasing the potential cost per iteration and use this as our default algorithm.

3.2 Many Fixed Effects

There is a long tradition of extending the demand side utility to incorporate product or market fixed effects. For example Nevo (2001), Nevo (2000a) allow for product fixed effects d_j, so that:

$$\delta_{jt} = d_j + \lambda_t - \alpha p_{jt} + \Delta\xi_{jt}$$

These can manageably be incorporated as dummy-variables in the linear part of the regression as there are only $J = 24$ products.

Backus et al. (2018) and Conlon and Rao (2017) allow for product (j) store (or chain) (s) specific fixed effects d_{js}. Using weekly store-upc level Nielsen scanner data it is not uncommon for there to be more that $J = 3500$ products (in both distilled spirits and ready-to-eat cereal). If one wants to allow for store-upc specific fixed effects d_{js} this can explode to 100,000 or more fixed effects. Indeed, in Backus et al. (2018) the authors incorporate more than 50,000 such fixed effects. Similarly, there are approximately $T = 500$ weeks of Nielsen scanner data from 2006-2016. If one wanted to incorporate store × week fixed effects for only 100 stores this too could reach the order of 50,000 such fixed effects.

$$\delta_{jst} + \alpha p_{jst} = d_{js} + \lambda_{st} + \Delta\xi_{jst}$$

There are several differencing algorithms for removing the fixed effects. For simplicitly let's assume there are two dimensions of fixed effects N and T where $N >> T$:

$$\tilde{y}_{it} = y_{it} - \overline{y}_{i\cdot} - \overline{y}_{\cdot t}$$
$$\tilde{x}_{it} = x_{it} - \overline{x}_{i\cdot} - \overline{x}_{\cdot t}$$

The simplest approach might be to *iteratively demean*, that is remove $\overline{x}_{i\cdot}$ and update x_{it} and then remove $\overline{x}_{\cdot t}$ and to repeat this process until $\overline{x}_{i\cdot} = \overline{x}_{\cdot t} = 0$. This can be done in a single iteration if $Cov(\overline{x}_{\cdot t}, \overline{x}_{i\cdot}) = 0$. However, when the two dimensions of fixed effects are correlated this can take many iterations and can be quite burdensome.

The least squares dummy variables (LSDV) approach requires constructing the annihilator matrix which removes all of the fixed effects while handling the correlation. This approach requires inverting an $(N + T) \times (N + T)$ matrix. Often constructing (and inverting) such a large matrix is infeasible because of memory requirements. Several algorithms have been proposed to deal with

this problem. The most popular algorithm is the `reghdfe` (Stata) approach of Correia (2016) which iteratively regresses Y on X in order to residualize Y.

The BLP application is a bit unusual in that we re-use the x_{jt}, w_{jt} variables over and over. However, the left hand side variables $\widehat{\delta}_{jt}(\theta_2) + \alpha p_{jt}, \widehat{mc}(\theta_2)$ change with each guess of θ_2, which means the entire procedure needs to be repeated for each update of θ_2.

For two-dimensional fixed effects, `pyblp` uses the algorithm of Somaini and Wolak (2016). This has the advantage that the linear explanatory variables in (X_1, X_3) can be residualized only once, while the left hand side variables (which are $N \times 1$ vectors) still need to be residualized for each guess of θ_2. The savings are particularly large when the dimension $\dim(X_1) = K_1$ or $\dim(X_3) = K_3$ are large. For three or more dimensional fixed effects `pyblp` uses the iterative de-meaning algorithm of Rios-Avila (2015) similar to the one described above.

3.3 Optimization and Numerical Issues

As documented by Knittel and Metaxoglou (2014), optimization of the GMM objective function for the BLP problem can be challenging. Best practices involve considering a number of different starting values and optimization routines, verifying that $\widehat{\theta}$ satisfies both the first order conditions (gradient is within tolerance of zero) and second-order conditions (Hessian matrix has all positive eigenvalues). Both of these are reported by default in `pyblp`. Some optimization routines also allow the user to input constraints on parameters. Sometimes these constraints can speed up estimation, or prevent the optimization routine from considering "unreasonable" values.

The BLP problem itself in (8) represents a non-convex optimization problem.[21] This means that the Hessian need not be positive-semi-definite at all values of θ_2. In practice this means that no optimization routine is guaranteed to return a global minimum in a fixed amount of time. This critique applies both to the Simplex/Nelder-Mead type approaches and to the derivative based Quasi-Newton approaches.

The `pyblp` package uses `scipy.optimize` routines for optimization as well as interfacing with commercial routines such as `KNITRO`. The `KNITRO` software incorporates four different gradient based search routines, and has been recommended for the BLP problem previously by Dubé et al. (2012). Table 10 lists the nine different optimization routines currently supported by `pyblp`. Though non-derivative based routines such as `nelder-mead` have been frequently used in previous papers, they are not recommended.[22]

Indeed the most important aspect of `pyblp` optimization is that it calculates the analytic gradient for any user provided model, including models which incorporate both supply and demand

[21] Absent unobserved heterogeneity or random coefficients the problem is globally convex.

[22] Both Knittel and Metaxoglou (2014) and Dubé et al. (2012) find that derivative based routines outperform the simplex based `nelder-mead` routine both in terms of speed and reliability. Our own tests concur with this prior opinion.

moments or fixed effects.[23] Analytic gradients provide both a major speedup in computational time, while also generally improving the chances that the algorithm converges to a valid minimum.

Another important feature is that many routines allow the user to place bounds or "box constraints" on the parameter space. That is we can restrict components of θ_2 which we call $\theta_l \in [\underline{\theta}_l, \overline{\theta}_l]$. Some typical constraints are that: demand slopes down, random coefficients have positive but bounded variances, etc. One of the most common challenges for estimation routines is when the optimization software considers such a large value for a random coefficient that $s_{jt}(\delta_{\cdot t}, \theta_2) \to 1$ while $s_{kt}(\delta_{\cdot t}, \theta_2) \to 0$ for $k \neq j$. This constitutes a form of *overflow* error and represents one of the major reasons these optimization routines can fail.[24] By restricting variance and covariance from becoming too large we can prevent many of these sorts of errors.

Another way to guarantee overflow safety is to use a protected version of the $\log - \text{sum} - \exp$ (LSE) function. This is accomplished by $LSE(x_1, \ldots, x_K) = \log \sum_k \exp x_k = a + \log \sum_k \exp(x_k - a)$. By choosing $a = \max_k x_k$ this helps ensure overflow safety. This is a well known trick from the applied mathematics and machine-learning literatures, but there does appear to be evidence of it in the literature on BLP models.

The optimization interface to `pyblp` is generic in the sense that any optimizer which uses the same format as `scipy.optimize` package should work fine. To illustrate this, in the documentation we give an example of such a "custom" routine where we construct a simple brute force solver which searches over a grid of parameter values. This should allow users to experiment with routines without much difficulty or "upgrade" if better routines are developed.

Our generic recommendation is that researchers try `KNITRO` first if available, and then try `slsqp` with constraints on the parameter space.[25] If a user is unwilling to place constraints on the parameter space we find that `bfgs` seems to be the least likely to choose values for which *overflow* errors are an issue. As a generic recommendation, we suggest placing relatively restrictive bounds on the parameter space, and do not recommend non-derivative based optimization routines (ie: Simplex or Nelder-Mead) under any circumstances. We also recommend trying multiple different optimizers and starting values to check for agreement.

[23] See A.2 in the Appendix.

[24] In this case $s_{jt}(\delta_{\cdot t}, \theta_2) = s_{jt}$ cannot be solved for $\delta_{\cdot t}$, thus the inversion for the mean utilities fails.

[25] The main disadvantage of `KNITRO` is that it is not freely available and must be purchased and installed by end-users.

3.4 Heterogeneity and Integration

An important aspect of the BLP model is that it incorporates heterogeneity via random coefficients. The challenge is that the integration in (3) needs to be performed numerically:

$$s_{jt}(\delta_t, \theta_2) = \int \frac{\exp[\delta_{jt} + \mu_{ijt}]}{\sum_{k \in J_t} \exp[\delta_{kt} + \mu_{ikt}]} f(\mu_{it}|\theta_2) d\,\mu_{it} \approx \sum_{i=1}^{I_t} w_i \underbrace{\frac{\exp[\delta_{jt} + \widehat{\mu}_{ijt}(\nu_{it})]}{\sum_{k \in J_t} \exp[\delta_{kt} + \widehat{\mu}_{ikt}(\nu_{it})]}}_{s_{ijt}(\mu_{it}(\theta_2))} \tag{18}$$

The most common approach is Monte-Carlo integration where random draws are taken from some candidate distribution $\nu_{it} \sim f(\nu_{it}|\theta_2)$ and then used to calculate $\mu_{ijt}(\nu_{it}, \theta_2)$. We can then take a weighted sum of individuals $s_{jt}(\delta_t, \theta_2) = \sum_{i=1}^{I_t} w_i s_{ijt}(\delta_t, \mu_{it}(\theta_2))$. The main advantage of Monte-Carlo integration (actually pseudo-Monte Carlo integration) is that it avoids the *curse of dimensionality*. While the accuracy for low dimensional integrals increases slowly in the number of points of approximation, the accuracy does not decline quickly as one increases the dimension of integration.

The second approach is the so-called *Gaussian Quadrature* approach. Here the integrand is approximated with a polynomial and then integrated exactly as polynomial integration is trivial. In practice this is far simpler as it still amount to a weighted sum in (18) over a particular choice of (ν_i, w_i). The main choice the user makes is the *polynomial order* of the rule. In effect the user chooses the order of the polynomial used to approximate the integrand. As the order grows, more nodes are required by the accuracy of the approximation improves.

Gaussian Quadrature works best when certain conditions are met: that the integrand is bounded and continuously differentiable. Thankfully, the logit kernel in (18) is always bounded on $[0, 1]$ (because it is a marketshare) and is infinitely continuously differentiable (real analytic). This lends itself to quadrature type approaches. There are a number of different flavors of quadrature rules designed to best approximate integrals under different weighting schemes. The *Gauss-Hermite* family of rules work best when $f(\mu_{it}) \propto exp[-\mu_{it}^2]$, which (with a change of variables) includes integrals over a normal density. The *Gauss-Kronod* rules offer an alternative where for a given level of polynomial accuracy p, it reuses the set of nodes from the $p-1$ polynomial accuracy integration rule. This allows for *adaptive* accuracy without wasting calculations, when the error from numerical integration is large the the polynomial degree can be expanded. Both the advantage and disadvantage of *Gaussian Quadrature* rules is that they do a better job covering the "tail" of the probability distribution. While this increases the accuracy of the approximation, it can also lead to very large values which create *overflow* issues.[26] We prefer to use quadrature rules, but to be careful of potential overflow/underflow issues when computing shares.

The Gaussian quadrature rules apply only to a single dimension. One way to estimate higher dimensional integrals, is to construct the product of single dimensional integrals. The disadvantage

[26] Essentially for some simulated individual i we have that $s_{ijt} \to 1$ and $s_{ikt} \to 0$. This problem has been previously documented by Judd and Skrainka (2011) and Skrainka (2012b).

of *product rules* is that if one needs I_t points to approximate the integral in dimension one, then one needs I_t^d points to approximate the integral in dimension d. This is the so-called *curse of dimensionality*.

The *curse of dimensionality* is a well-known problem in numerical analysis and several off-the-shelf solutions exist. There are several clever algorithms for improving upon the product rule for higher dimensional integration. Judd and Skrainka (2011) explores *monomial cubature rules* from Stroud (1971) while Heiss and Winschel (2008) use *Sparse Grids* methods to selectively eliminate nodes from the product rules. One disadvantage of these methods is that they often involve weights which are negative $w_i < 0$, which can create problems when trying to decompose the distribution of heterogeneity (particularly for counterfactuals).

Though `pyblp` allows for arbitrary (user-supplied) distributions of random coefficients, by far the most commonly employed choices in the literature are the independent normal and correlated normal distributions for $f(\nu_i|\theta_2)$. Here `pyblp` provides some specialized routines to handle these integrals with limited user intervention. There are number of different methods one can use to generate (w_i, ν_i) for the (correlated) normal case:

1. `monte_carlo`: Draw ν_{ik} from the standard normal distribution `numpy.random.normal()` for each dimension of heterogeneity. Set $w_i = \frac{1}{I_t}$ where I_t is the number of simulated "individuals".

2. `product`: Construct the *Gauss-Hermite* quadrature rules (ν_i, w_i) for a single dimension and build product rules to obtain a set of weights and nodes in higher dimensions d.

3. `nested_product`: Constructs (ν_i, w_i) using the product rule in dimension d but beginning with the basis of *Gauss-Kronod* rules instead of *Gauss-Hermite* rules.

4. `grid`: Constructs (ν_i, w_i) using the *Sparse Grids* rules of Heiss and Winschel (2008) for dimension d using the *Gauss-Hermite* rules as an initial basis.

5. `nested_grid`: Constructs (ν_i, w_i) using the *Sparse Grids* rules of Heiss and Winschel (2008) for dimension d using the *Gauss-Kronod* rules as an initial basis.

[Should we incorporate low-discrepancy QMC rules like Halton draws and Sobol Sequences? These are popular but SGI is probably better.]

In general, the best practice in low dimensions is probably to use the `product` rules to a relatively high degree of polynomial accuracy. In higher dimensions *Sparse Grids* appear to scale the best both in our own Monte Carlo studies and those of Judd and Skrainka (2011) and Heiss and Winschel (2008).

3.5 Solving for Pricing Equilibria

Many counterfactuals of BLP type problems involve perturbing either the market structure, marginal costs, or both and solving for counterfactual equilibrium prices. The Bertrand-Nash first order conditions are defined by (5) for each market t:

$$\mathbf{mc} = \mathbf{p} - \underbrace{(O \odot \Omega(\mathbf{p}))^{-1} s(\mathbf{p})}_{\eta(\mathbf{p},O)}$$

During estimation, in order to recover marginal costs, one need only invert the $J_t \times J_t$ matrix: $(O \odot \Omega(\mathbf{p}))^{-1} s(\mathbf{p})$. Solving for counterfactual pricing equilbria is more difficult as now we must solve the $J_t \times J_t$ nonlinear system of equations, often after replacing the ownership matrix with a post-merger counterpart: $O \to \tilde{O}$.

$$\mathbf{p} = \mathbf{mc} + \eta^{post}(\mathbf{p}, \tilde{O}) \tag{19}$$

In general, solving this problem is hard as it represents a non-convex nonlinear system of equations, where one must simulate in order to compute $\eta(\mathbf{p}, \tilde{O})$ and its derivatives.[27] Once one incorporates both multi-product firms and arbitrary coefficients into the problem both existence and uniqueness of an equilibrium become challenging to establish.[28]

One approach might be to solve the system using Newton's method, which requires calculating the $J_t \times J_t$ Jacobian: $\frac{\partial \eta(\mathbf{p})}{\partial \mathbf{p}}$. We provide (possibly novel) analytic expressions for this Jacobian in Appendix A.2.[29] The expression for the Jacobian involves the Hessian matrix of demand $\frac{\partial^2 s_k}{\partial p_j^2}$ as well as tensor products, and can be computationally challenging.[30]

The second, and perhaps most common approach in the literature is treating (19) as a fixed point and iterating on $\mathbf{p} \leftarrow\!\shortmid \mathbf{mc} + \eta^{post}(\mathbf{p}, \tilde{O})$.[31] The problem is that while a fixed point of (19) may represent the Bertrand-Nash equilibrium of (6), it is not necessarily a contraction. In fact, as part of Monte Carlo experiments conducted in Armstrong (2016), the author finds that iterating

[27] The first Monte Carlo studies which evaluate price and quantity equilibria as part of the data generating process are likely Armstrong (2016), Skrainka (2012a) and Conlon (2017).

[28] Caplin and Nalebuff (1991) and Gallego et al. (2006) have results which apply to single product firms and linear in price utility under logit demands. Konovalov and Sandor (2010) generalizes these to logit demands with linear in price utility and multi-product firms. With the addition of random coefficients, it is possible that the resulting model will violate the quasi-concavity of the profit function these results requie. Morrow and Skerlos (2010) avoids some of these restrictions but places other restrictions on indirect utilities. Existence and uniqueness are beyond the scope of this paper, we instead focus on calculating solutions to the system of first order conditions asusming such a solution exists.

[29] For example Knittel and Metaxoglou (2014) don't update $\mathbf{s}(\mathbf{p})$ and thus avoid fully solving the system of equations.

[30] An alternative might be to try a Newton-type approach without providing analytic formulas for the Jacobian. This seems slow and ill-advised as there are J_t markups each with J_t derivatives and each derivative involves integration in order to compute both $\Omega(\mathbf{p})$ and $\mathbf{s}(\mathbf{p})$.

[31] See Miravete et al. (2018) for a recent example.

on (19) does not always lead to a solution and at least some fraction of the time leads to cycles. We were able to replicate this finding for similarly constructed Monte Carlo experiments between 1-5% of the time. Were this relation to be a contraction, we should always be able to iterate on $\mathbf{p} \mapsto \mathbf{mc} + \eta^{post}(\mathbf{p}, \tilde{O})$ until we reached the solution.

Our preferred approach follows Morrow and Skerlos (2011). This approach does not appear to be well known in the Industrial Organization literature, but in our experiments appears to be quite effective. They reformulate the solution to (5) by breaking up $\Omega(\mathbf{p})$ into two parts: a $J_t \times J_t$ diagonal matrix Λ, and a $J_t \times J_t$ dense matrix Γ, where $\alpha_i = \frac{\partial u_{ijt}}{\partial p_{jt}}$ the marginal (dis)-utility of price. We can write their reformulation:

$$\begin{aligned} \Omega(\mathbf{p}) =& \Lambda(\mathbf{p}) - \Gamma(\mathbf{p}) \\ \Lambda_{jj} =& \int \alpha_i s_{ijt}(\mu_{it}) f(\mu_{it}|\widetilde{\theta}_2) \partial\mu \\ \Gamma_{jk} =& \int \alpha_i s_{ijt}(\mu_{it}) s_{ikt}(\mu_{it}) f(\mu_{it}|\widetilde{\theta}_2) \partial\mu \end{aligned} \tag{20}$$

They reformulate the problem as a different fixed point:[32]

$$\mathbf{p} = \mathbf{c} + \zeta(\mathbf{p}) \quad \text{where} \quad \zeta(\mathbf{p}) = \Lambda^{-1}(\tilde{O} \odot \Gamma(\mathbf{p}))(\mathbf{p} - \mathbf{c}) - \Lambda(\mathbf{p})^{-1}\mathbf{s}(\mathbf{p}) \tag{21}$$

The fixed point in (21) is entirely different from that in (19) and coincides *only* at the resting point(s). Consistent with results reported in Morrow and Skerlos (2011), we find that (21) is around 3-12x faster than Newton-type approaches and reliably finds an equilibrium.

[Accelerated FP methods don't speed this calculation up, right?]

Perhaps most consequentially, the ability to solve for a pricing equilibrium rapidly and reliably makes it possible to generate the Amemiya (1977) or Chamberlain (1987) "optimal instruments".

3.6 Optimal Instruments

Though this section may seem pedantic, the expression we derive differs slightly from the prior literature in a way which may have substantial impact on the usefullness of these instruments.

As a way to improve efficiency, we can construct optimal instruments in the spirit of Amemiya (1977) or Chamberlain (1987). These optimal instruments were featured in both Berry et al. (1995) and Berry et al. (1999) but are not commonly employed in many subsequent studies using the BLP approach in part because they are challenging to construct. While the procedure itself is quite involved, the good news is that does not require much in the way of user input, and can be fully implemented by the `pyblp` software.

[32] This resembles a well known "folklore" solution to the pricing problem which is to rescale each equation by its own share s_{jt}. See Skrainka (2012a). For the plain logit $\Lambda^{-1}_{(jj)} = \frac{1}{\alpha \cdot s_j}$.

For the GMM problem, Chamberlain (1987) tells us that the optimal instruments are related to the expected Jacobian of the moment conditions where the expectation is taken conditional on the exogenous regressors (z_t) only. We can write this expectation for each product j and market t as $E[D_j(z_t)\Omega^{-1}|z_t]$. We derive the components of the optimal instruments under the assumption that (ξ_{jt}, ω_{jt}) are jointly i.i.d:[33]

$$D_j := \underbrace{\begin{bmatrix} \frac{\partial \xi_j}{\partial \beta} & \frac{\partial \omega_j}{\partial \beta} \\ \frac{\partial \xi_j}{\partial \alpha} & \frac{\partial \omega_j}{\partial \alpha} \\ \frac{\partial \xi_j}{\partial \widetilde{\theta}_2} & \frac{\partial \omega_j}{\partial \widetilde{\theta}_2} \\ \frac{\partial \xi_j}{\partial \gamma} & \frac{\partial \omega_j}{\partial \gamma} \end{bmatrix}}_{(K_1+K_2+K_3)\times 2} = \begin{bmatrix} -x_j & 0 \\ \frac{\partial \xi_j}{\partial \alpha} & \frac{\partial \omega_j}{\partial \alpha} \\ \frac{\partial \xi_j}{\partial \widetilde{\theta}_2} & \frac{\partial \omega_j}{\partial \widetilde{\theta}_2} \\ 0 & -x_j \\ 0 & -w_j \end{bmatrix}, \quad \Omega := \underbrace{\begin{bmatrix} \sigma_\xi^2 & \sigma_{\xi,\omega} \\ \sigma_{\xi,\omega} & \sigma_\omega^2 \end{bmatrix}}_{2\times 2} \tag{22}$$

A little calculation shows that for each market t and each observation j:

$$D_j\Omega^{-1} = \frac{1}{\sigma_\xi^2\sigma_\omega^2 - (\sigma_{\xi,\omega})^2} \times \begin{bmatrix} -\sigma_\omega^2 x_j & \sigma_{\xi\omega} x_j \\ \sigma_\omega^2 \frac{\partial \xi_j}{\partial \alpha} - \sigma_{\xi,\omega}\frac{\partial \omega_j}{\partial \alpha} & \sigma_\xi^2 \frac{\partial \omega_j}{\partial \alpha} - \sigma_{\xi,\omega}\frac{\partial \xi_j}{\partial \alpha} \\ \sigma_\omega^2 \frac{\partial \xi_j}{\partial \widetilde{\theta}_2} - \sigma_{\xi,\omega}\frac{\partial \omega_j}{\partial \widetilde{\theta}_2} & \sigma_\xi^2 \frac{\partial \omega_j}{\partial \widetilde{\theta}_2} - \sigma_{\xi,\omega}\frac{\partial \xi_j}{\partial \widetilde{\theta}_2} \\ \sigma_{\xi,\omega} x_j & -\sigma_\xi^2 x_j \\ \sigma_{\xi,\omega} w_j & -\sigma_\xi^2 w_j \end{bmatrix} \tag{23}$$

Clearly rows 1 and 4 are co-linear. Let Θ be a conformable matrix of zeros and ones such that:

$$(D_j\Omega^{-1}) \circ \Theta = \frac{1}{\sigma_\xi^2\sigma_\omega^2 - (\sigma_{\xi\omega})^2} \times \begin{bmatrix} -\sigma_\omega^2 x_{jt} & 0 \\ \sigma_\omega^2 \frac{\partial \xi_j}{\partial \alpha} - \sigma_{\xi,\omega}\frac{\partial \omega_j}{\partial \alpha} & \sigma_\xi^2 \frac{\partial \omega_j}{\partial \alpha} - \sigma_{\xi,\omega}\frac{\partial \xi_j}{\partial \alpha} \\ \sigma_\omega^2 \frac{\partial \xi_j}{\partial \widetilde{\theta}_2} - \sigma_{\xi,\omega}\frac{\partial \omega_j}{\partial \widetilde{\theta}_2} & \sigma_\xi^2 \frac{\partial \omega_j}{\partial \widetilde{\theta}_2} - \sigma_{\xi,\omega}\frac{\partial \xi_j}{\partial \widetilde{\theta}_2} \\ 0 & -\sigma_\xi^2 x_j \\ \sigma_{\xi,\omega} w_j & -\sigma_\xi^2 w_j \end{bmatrix} \tag{24}$$

We can partition our instrument set by column into "demand" and "supply" instruments:

$$\underbrace{z_j^D}_{K_1+K_2+(K_3-K_1)} := E[(\Delta_j\Omega^{-1} \circ \Theta)_{\cdot 1}|z] \text{ and } \underbrace{z_j^S}_{K_2+K_3} := E[(\Delta_j\Omega^{-1} \circ \Theta)_{\cdot 2}|z] \tag{25}$$

In (25), we have $K_2 + K_3$ instruments for supply: z_j^S and demand: z_j^D, though it is evident from (23) that the instruments for the endogenous parameters are not the same.[34]

Notice that when we include simultaneous supply and demand moments we have overidentifying

[33] We can easily extend this formula to allow for heteroskedastic or clustered standard errors by letting Ω_{jt} rather than Ω for the i.i.d case.

[34] This is true except for the knife-edge case where $\frac{\partial \xi_j}{\partial \theta_l} / \frac{\partial \omega_j}{\partial \theta_l} \propto \frac{\sigma_\xi^2 + \sigma_{\xi,\sigma}}{\sigma_\omega^2 + \sigma_{\xi,\sigma}}$.

restrictions. As shown in (25), we have $2 \cdot (K_2 + K_3)$ restrictions and $(K_1 + K_2 + K_3)$ parameters. This gives us $K_2 + (K_3 - K_1)$ overidentifying restrictions. The first set of K_2 overidentifying restrictions comes from the fact that in (23) we have two restrictions for each of the endogenous/nonlinear parameters $[\alpha, \widetilde{\theta}_2]$. The second set of $K_3 - K_1$ overidentifying restrictions comes from the cost shifters w_{jt} which are excluded from demand.[35] With the exception of derivatives with respect to the nonlinear/endogenous parameters, many of the instruments are simply exogenous regressors re-scaled by covariances.

Remark 1

The set of overidentifying restrictions should be quite intuitive. Both supply and demand moments are informative about the endogenous parameters, and excluded cost shifters provide overidentifying restrictions. It is worth pointing out that our derivation of the optimal instruments appears to vary from those in the prior literature. Some of the prior literature using optimal instruments for BLP type problems suggests the resulting problem is *just identified* rather than *over identified*. Reynaert and Verboven (2014) appear to construct their version of optimal instruments by summing across the rows of (23) and excluding either the first or third row. This gives them $K = K_1 + K_2 + K_3$ instruments and K unknowns so that the model is just identified. However, because they stack (ξ, ω) they effectively have $2 \cdot N$ rather than N observations. Conceptually, one way to view their formulation is that it imposes $E[\xi'_{jt} z_{jt}] + E[\omega'_{jt} z_{jt}] = 0$ rather than separately imposing $E[\xi'_{jt} z_{jt}] = 0$ and $E[\omega'_{jt} z_{jt}] = 0$.

Remark 2

If one assumes that $\sigma_{\xi,\omega} = 0$ or that the supply and demand shocks are uncorrelated, then the expression in (23) clearly shows that the optimal instruments for demand no longer depend on the supply Jacobian $\frac{\partial \xi_j}{\partial \theta_2}$ and likewise in that optimal instruments for supply no longer depend on the demand Jacobian $\frac{\partial \omega_j}{\partial \theta_2}$.[36] We still have overidentifying restrictions in this case as both supply and demand moments provide information on the endogenous parameters.

Constructing Feasible Instruments

The main challenge with implementing the optimal instruments is that the expectations of the Jacobian terms: $E[\frac{\partial \xi_j}{\partial \theta_2}, \frac{\partial \omega_j}{\partial \theta_2} | z]$ are not directly measured. The most obvious example is that $\frac{\partial \xi_j}{\partial \alpha} = p_j$, however because p_j is endogenous we must replace it with $E[p_j | z]$.

Here is what Berry et al. (1995) say about optimal instruments:

> *Unfortunately $D_j(z)$ is typically very difficult, if not impossible, to compute. To cal-*

[35] If not all x_{jt} from the demand side are included in supply we still have overidentifying restrictions from w_{jt}.

[36] In this case the optimal weighting matrix would also have a block diagonal structure with separate blocks for demand and supply moments. The zero covariance restriction is the identification argument in MacKay and Miller (2018).

> *culate $D_j(z)$ we would have to calculate the pricing equilibrium for different (ξ_j, ω_j) sequences, take derivatives at the equilibrium prices, and then integrate out over the distribution of such sequences. In addition, this would require an assumption that chooses among multiple equilibria when they exist, and either additional assumptions on the joint distribution of (ξ, ω), or a method for estimating that distribution.*

The appendix of the NBER working paper version of Berry et al. (1999) is even less positive:

> *Calculating a good estimate of $E[p|z]$ then requires (i) knowing or estimating the density of the unobservables and (ii) solving at some initial guess for θ the fixed point that defines equilibrium prices for each (ξ, ω) and then integrating this implicit function with respect to the density of the unknown parameters. This process is too complicated to be practical.*

Even Reynaert and Verboven (2014) focus primarily on the case of perfect competition where $E[p_{jt}|z] = E[mc_{jt}|z] = x_{jt}\widehat{\gamma}_1 + w_{jt}\widehat{\gamma}_2 + \widehat{\omega}_{jt}$ which avoids implicit functions and solving for equilbrium prices. When they do consider imperfect competition they prefer to construct $E[p_{jt}|z_t]$ by regressing the endogenous variable on a series of exogenous regressors (essentiallly a "first stage" regression).

We follow the (likely) more accurate (but costly) recipe proposed by Berry et al. (1999) and show that given other computational advances in `pyblp` it is feasible to implement. For each market t we can:

1. Obtain an initial estimate $\widehat{\theta} = [\widehat{\beta}, \widehat{\alpha}, \widehat{\theta}_2, \widehat{\gamma}]$.
2. Obtain an initial estimate of $\widehat{\Omega}^{-1}$ by taking covariances of $(\widehat{\xi}, \widehat{\omega})$
3. Draw the $J_t \times 2$ matrix of structural errors: (ξ_t^*, ω_t^*). [See below].
4. Compute $\hat{y}_{jt}^S = \widehat{mc}_{jt} = [x_{jt}\, w_{jt}]\widehat{\gamma} + \omega_{jt}^*$ and the exogenous portion of utility $\hat{y}_{jt}^D = x_{jt}\widehat{\beta} + \xi_{jt}^*$
5. Use $(\hat{y}_{jt}^D, \hat{y}_{jt}^S, \widehat{\alpha}, x_t, w_t)$ to solve for the equilbrium price and quantity $(\hat{\mathbf{p}}_\mathbf{t}, \hat{\mathbf{s}}_\mathbf{t})$ using the ζ-markup approach in Section 3.5. [Note: this does not include any endogenous quantities].
6. Treat the $(\hat{\mathbf{p}}_\mathbf{t}, \hat{\mathbf{s}}_\mathbf{t}, \mathbf{x}_\mathbf{t}, \mathbf{w}_\mathbf{t})$ as data and solve for $\widehat{\xi} = \xi(\hat{\mathbf{p}}_\mathbf{t}, \hat{\mathbf{s}}_\mathbf{t}, \mathbf{x}_\mathbf{t}, \mathbf{w}_\mathbf{t}, \hat{\alpha}, \widehat{\theta}_2)$ and $\widehat{\omega}_t = \omega(\hat{\mathbf{p}}_\mathbf{t}, \hat{\mathbf{s}}_\mathbf{t}, \mathbf{x}_\mathbf{t}, \mathbf{w}_\mathbf{t}, \hat{\alpha}, \widehat{\theta}_2)$ following Section 2.1.
7. Construct the Jacobian terms $[\frac{\partial \widehat{\xi}_j}{\partial \theta_2}(\xi_t^*, \omega_t^*), \frac{\partial \widehat{\omega}_j}{\partial \theta_2}(\xi_t^*, \omega_t^*)]$ and $\hat{D}_j(\xi_t^*, \omega_t^*)\widehat{\Omega}^{-1}$ using the analytic formulas in A.2.
8. Average $\hat{D}_j(\xi_t^*, \omega_t^*)$ or $\hat{D}_j(\xi_t^*, \omega_t^*)\widehat{\Omega}^{-1}$ over several draws of (ξ_t^*, ω_t^*) to construct an estimate of $E[\hat{D}_j|z_t]$.

There are three options for generating (ξ_t^*, ω_t^*) suggested by Berry et al. (1999), and `pyblp` makes all three available:

(a) `approximate`: Replace the structural errors with their expectation: $(\xi_t^*, \omega_t^*) = (E[\xi_t], E[\omega_t]) = (0, 0)$. This is what Berry et al. (1999) do.

(b) `asymptotic`: (default) Estimate an asymptotic normal distribution for $(\xi_{jt}, \omega_{jt}) \sim N(0, \widehat{\Omega})$ and then draw (ξ_t^*, ω_t^*) from that distribution.

(c) `empirical`: Draw (ξ_t^*, ω_t^*) from the joint empirical distribution of $(\widehat{\xi}_{jt}, \widehat{\omega}_{jt})$ given the initial estimates of $\widehat{\theta}$.

In general (b) and (c) are not believed to be computationally feasible, particularly when there are large numbers of draws for (ξ_t^*, ω_t^*). The costly step is (5) above which involves solving for a new equilibrium $(\hat{\mathbf{p}}_\mathbf{t}, \hat{\mathbf{s}}_\mathbf{t})$ for each set of draws. These improved optimal instruments are only really feasible because of the advances in Section 3.5 which drastically reduce the amount of time it takes to solve for equilibria. For relatively large problems, constructing optimal instruments may take several minutes. For smaller problems such as Berry et al. (1995) or Nevo (2000b) it takes only several seconds. Our simulations indicate that `approximate` performs as well as the more expensive `empirical` or `asymptotic` formulations.

Updating `pyblp.results` with the optimal instruments takes only two lines of code:

```
instrument_results = results.compute_optimal_instruments(method='empirical')
updated_problem = instrument_results.to_problem()
```

Demand Side Only

The optimal instruments are much easier in the case where no supply side is specified, because we no longer have a model for marginal costs and cannot solve for equilibrium $(\hat{\mathbf{p}}_\mathbf{t}, \hat{\mathbf{s}}_\mathbf{t})$. Instead the user must supply a vector of expected prices $E[p_t|z_t]$, which can be constructed in a first-stage.

4 Replication Exercises

Nevo (2000) Setup

Here we provide replications using `pyblp` for the two best-known BLP applications. In the first replication we estimate the model of Nevo (2000b). This problem is notable because it includes both observable demographics, unobserved heterogeneity, and product fixed effects.

We demonstrate how to construct the Nevo problem using `pyblp` in Figure 1. We configure the: (1) the linear portion of demand β, (2) the nonlinear portion of demand Σ, and (3) the agent demographic interactions Π. We then put all three formulas together with the data. Replication is straightforward because the original instruments are provided along with the data, and the reported estimates are from one-step GMM with the TSLS $(Z'Z)^{-1}$ weighting matrix.

We estimate the model twice. Once using the set of parameters estimated in the original Nevo (2000b) example, and a second time where we restrict the interaction between income^2 and price $\Pi_{p,inc^2} = 0$. We report our results in Table 1. It is well known that the original reported estimates

```
# Linear demand parameters including product FE
linear_formula = pyblp.Formulation('0 + prices', absorb='C(product_ids)')
# Nonlinear Demand (RC)
nonlinear_formula = pyblp.Formulation('1 + prices + sugar + mushy')
product_formulations = (linear_formula,nonlinear_formula)
# Demographic Interactions
agent_formulation = pyblp.Formulation('0 + income + income_squared + age + child')
# This puts the problem together
nevo_problem = pyblp.Problem(product_formulations, product_data, agent_formulation, agent_data)
#display problem formulation
nevo_problem
```

```
Dimensions:
=====================================
 N     T    K1    K2    D    MD    ED
----  ---  ----  ----  ---  ----  ----
2256  94    1     4     4    20    1
=====================================

Formulations:
======================================================================
       Column Indices:           0           1            2      3
----------------------------  ------  --------------  -----  -----
 X1: Linear Characteristics    prices
X2: Nonlinear Characteristics    1         prices      sugar  mushy
       d: Demographics         income  income_squared   age   child
======================================================================
```

Figure 1: Nevo (2000b) Problem Formulation and Code

in Nevo (2000b) set the tolerance of the contraction mapping too loose. This explains the discrepancy between our *Replication* results and those originally reported. However, our replication results are identical to those reported by Dubé et al. (2012) using the MPEC approach. We prefer the *Restricted* results which have smaller standard errors and lack the multicolinearity problem between *income* and $income^2$ in the demographic interactions.

BLP 95/99 Setup

For our second replication, we consider the problem in Berry et al. (1995). The distinguishing features of this problem are that it lacks demographic interactions and product fixed effects but it adds a supply side. It also allows for the price coefficient $\alpha_i = \frac{\alpha}{y_i}$ where y_i is the income of agent i. This is modeled like a demographic interaction. As in the original paper, we estimate the model twice. The first time to obtain initial consistent estimates of $\widehat{\theta}$, and the second after constructing the optimal instruments in the style of Chamberlain (1987).

Our configuration for Berry et al. (1995)'s problem differs more substantially from the original paper, because parts of the original configuration are not included with the data:

1. We follow BLP (1999) by replacing the original specification's $\log(y_i - p_j)$ term with its first-

order linear approximation, p_j/y_i.[37] To do so, we include the inverse of income, $1/y_i$, as a demographic variable, and configure Π to interact the demographic with prices. Below, we refer to the parameter on this interaction as α. This means that the price parameters are no longer directly comparable.

2. The original instruments are not provided, but $\sum_{j \in J_t} x_{jt}$ are nearly collinear. We take these "BLP instruments" interact them up to the second degree and projected them down to the smallest number of principal components that explain at least 99% of the interactions' variance following Conlon (2017).

3. We start the optimization routine at the published estimates and the new $\alpha = -10$.

4. We constrain the diagonal of Σ to $[0, 10]$ and α to $[-50, -0.1]$.

5. Instead of importance sampling, we use 200 pseudo-Monte Carlo draws in each market.

We provide the `pyblp` formulation of the problem in Figure 2 and the results in Table 2. With the exception of price, which we construct in using the first-order approximation of Berry et al. (1999), we obtain broadly similar parameter estimates. Because the precise form of the instruments and weighting matrix are not included with the data itself, it is more difficult to fully replicate the parameter estimates.

5 Monte Carlo Experiments

5.1 Monte Carlo Configuration

Here we provide some Monte Carlo experiments to illustrate some of the best practices we laid out in Section 3.

Our simulation configurations are loosely based on those of Armstrong (2016). We construct 1,000 different synthetic datasets. There are $F = 5$ firms, and each produces a number of products chosen randomly from $J_f \in \{2, 5, 10\}$. There are $T = 20$ markets, and in each market, the number of firms is chosen randomly from $F_t \in \{F - 2, F - 1, F\} = \{3, 4, 5\}$. This procedure generates variation in the number of firms and products across markets which can be helpful for identification. Sample sizes are generally between $N = 200$ and $N = 600$. In extensions, we also consider simulations in which we scale the number of markets or the number of products per firm.

Unobserved product characteristics, ξ and ω, are drawn from a mean-zero bivariate normal distribution with $\text{Var}(\xi) = \text{Var}(\omega) = 0.1$ and $\text{Corr}(\xi, \omega) = 0.5$. In each market, unobserved agent characteristics, ν, consist of 1,000 draws from the standard normal distribution, and differ across markets.

[37] Otherwise there are consumers for which $p_j > y_i$ which creates a host of problems.

```
product_formulations = (
   # linear demand
   pyblp.Formulation('1 + hpwt + air + mpd + space'),
   # nonlinear demand
   pyblp.Formulation('1 + prices + hpwt + air + mpd + space'),
   # supply model
   pyblp.Formulation('1 + log(hpwt) + air + log(mpg) + log(space) + trend')
)
# agent model
agent_formulation = pyblp.Formulation('0 + I(1 / income)')
#display problem formulation
blp_problem = pyblp.Problem(product_formulations, product_data,agent_formulation, agent_data)
blp_problem
```

```
Dimensions:
============================================
 N     T    K1    K2    K3    D    MD    MS
----  ---  ----  ----  ----  ---  ----  ----
2217  20    5     6     6     1    11    12
============================================

Formulations:
================================================================================
       Column Indices:            0          1        2       3          4         5
----------------------------  --------  ---------  -----  --------  ----------  -----
 X1: Linear Characteristics       1        hpwt      air     mpd       space
X2: Nonlinear Characteristics     1       prices    hpwt     air        mpd      space
  X3: Cost Characteristics        1      log(hpwt)   air   log(mpg)  log(space)  trend
       d: Demographics         1/income
================================================================================
```

Figure 2: BLP95/99 Problem Formulation and Code

Observed linear product characteristics are $X_1 = [1, x, p]$, with a single random coeffcient $X_2 = x$, and linear supply characteristics $X_3 = [1, x, w]$. Exogenous characteristics, (x, w), are drawn from the standard uniform distribution. Endogenous Bertrand-Nash prices and shares $(\mathbf{p}, \mathbf{s})$ are computed via iteration over the ζ-markup equation (21).

Demand-side parameters are $(\beta_0, \beta_1, \alpha) = [-7, 6, -1]'$ and $\sigma_x = 3$. Other linear parameters were chosen to generate realistic outside shares that are generally between $s_{0t} = 0.8$ and $s_{0t} = 0.9$. Supply-side parameters are $(\gamma_0, \gamma_1, \gamma_2) = [2, 1, 1]'$. These parameters enter into a linear functional form for marginal costs, $c = X_3\gamma + \omega$.
We consider three variants on this Monte Carlo design:

Simple This is the baseline problem described above.

Complex adds a random coefficient on price: $X_2 = (x, p)$, $\Sigma_{22} = 0.2$, and $\Sigma_{12} = \Sigma_{21} = 0$.

RCNL adds a nesting parameter: $\rho = 0.5$ and each of the J_f products produced by a firm is randomly assigned to one of $H = 2$ nesting groups.

Two broad classes of models are estimated: demand-only models, which are estimated with single equation GMM, and models that also include the supply side, which are estimated with multiple equation GMM.

[Check that this is how instruments are constructed]

Instruments are constructed in two stages. The initial supply instruments consist of all of the exogenous regressors. The initial demand instruments consist of the exogenous regressors x_j, the cost shifters w_j, and the "BLP Instruments" which are the sum of product characteristics for products controlled by the same firm and those controlled by competing firms:

$$\begin{aligned} Z_t^S &= \{1, x_j, w_j\} \\ Z_t^D &= \left\{1, x_j, w_j, \sum_{k \in \mathcal{J}_f} 1, \sum_{k \notin \mathcal{J}_f} 1, \sum_{k \in \mathcal{J}_f} x_k, \sum_{k \notin \mathcal{J}_f} x_k, \right\} \end{aligned} \tag{26}$$

For some specifications we include the *differentiation IV* of Gandhi and Houde (2017) in place of the BLP instruments:

$$\mathbf{d}_{jk} = \mathbf{x}_k - \mathbf{x}_j, \quad Z_t^{l,Local} = \left\{ \sum_{k \in \mathcal{J}_f} |d_{jk}^{(l)}| < \sigma, \sum_{k \notin \mathcal{J}_f} |d_{jk}^{(l)}| < \sigma \right\}, \quad Z_t^{l,Quad} = \left\{ \sum_{k \in \mathcal{J}_f} d_{jk}^{(l)2}, \sum_{k \notin \mathcal{J}_f} d_{jk}^{(l)2} \right\} \tag{27}$$

The *differentiation IV* come in two flavors: *Local* and *Quadratic*. In both cases, the differentiation IV are constructed by computing the distance in characteristic space between products j and k for each characteristic (l). The *Local* measure counts the number of products within a standard

deviation of product j for characteristic l, while the *Quadratic* measure sums up the aggregate distance between j and other products.

Optimal instruments are constructed in three different variants following Section 3.6. The first set is constructed by setting $\xi^* = \omega^* = 0$. The second set draw from the empirical distribution $(\widehat{\xi}_{jt}, \widehat{\omega}_{jt})$ from the first stage estimates. The third set is constructed by averaging over 100 draws from the estimated bivariate normal distribution of ξ^* and ω^*. When a supply side is included, the vector of expected prices is computed via iteration over the ζ-markup equation following Section 3.5. Absent a supply side, $E[p_{jt}|z_t]$ is estimated from a regression of endogenous prices onto all exogenous variables.

To numerically integrate choice probabilities, we use Gauss-Hermite product rules that exactly integrate polynomials of degree 17 or less.[38] In an extension, we also compare results computed under pseudo-Monte Carlo integration with three times as many nodes.

To solve the standard fixed point for δ in each market, we use the `SQUAREM` acceleration method of Varadhan and Roland (2008) with L_∞ tolerance of 10^{-12}.[39] During the first GMM step, each iteration routine starts at the solution to the logit (or nested logit) model; in the second step, routines start at the estimated first-stage $\hat{\delta}$.

To optimize, we supply objective values and analytic gradients to a Sequential Least SQuares Programming (SLSQP) routine, which is made available by the open-source `SciPy` library. We use an objective tolerance of 10^{-10}.[40] Drawing different starting values from a uniform distribution with support 50% below and above the true parameter values, we solve each simulation three times and keep the solution with the smallest objective value. For each simulation, we use box constraints 1,000% above and below the true values, except for the diagonal of Σ, which we bound from below by zero; for α when a supply side is estimated, which we bound from above by -0.001; and for ρ in the RCNL simulation, which we constrain to $[0, 0.95]$.[41]

5.2 Monte Carlo Results

Fixed Effects

Fixed effects are either included as indicator variables or absorbed. A single fixed effect is absorbed

[38] In the *Simple* specification when there is only one nonlinear product characteristic, the product rule is one-dimensional, with $(17 + 1)/2 = 9$ nodes. In the *Complex* specification, there are 9^2 nodes. If our simulations were of higher dimension, it would be more efficient to use sparse grid integration. We chose to use product rules because for these small simulation problems the gains from sparse grids are minimal. Sparse grids also have negative weights, which can introduce some estimation problems.

[39] We chose not to use a stricter tolerance of, for example, 10^{-14} to allow for comparison of different fixed point formulations. For example, the exponentiated version often requires a looser tolerance. Rarely, the fixed point iteration routine will fail to converge. We set a maximum limit of 5,000 contraction evaluations in each market. To reduce numerical instability, we scale the logit expression by $\exp(-\max_i V_{ijt})$.

[40] Rarely, the optimization routine will fail to converge. We set a maximum limit of 100 major optimization iterations.

[41] These same box constraint exceptions also apply to the support of the distribution from which we draw initial parameter values.

with simple de-meaning; two, with Somaini and Wolak (2016) algorithm; and three, with iterative de-meaning, using an infinity norm tolerance of 10^{-14}.

In Table 3 we compare the computational time as well as the memory usage for Monte Carlo experiments where we either absorbed the fixed effects or included them as dummy variables. As one might expect, absorbing the fixed effects dramatically reduces the memory requirements (by multiple orders of magnitude) and can speed up the computation by as much as 5×. As expected the largest improvements for the 2-D case are when one dimension is much larger than the other. Even absorbing relatively small numbers of fixed effects (125 in a single dimension) leads to a substantial reduction in memory usage, and substantial speedup in computational time. This is likely an important advantage of `pyblp` going forward.

[I don't understand this part. Do we need it?]

Across 100 different simulations and 10 identical runs of the Nevo problem, we report the median seconds and maximum memory needed to estimate a single GMM step. For comparison with Nevo (2000b) problem with 24 product FE we increase the number of simulated markets to $T = 200$. Up to three fixed effects are added to the Simple simulation, which each consist of values randomly chosen from a list of 100 different levels. Associated parameters in β are drawn from the standard uniform distribution.

Fixed Point Iteration

In order to highlight the effects of different iteration schemes from Section 3.1, we explore different methods of solving the system of share equations for $\delta_{\cdot t}(\theta_2)$. We compare the conventional fixed-point iteration scheme proposed by Berry et al. (1995) to several alternative methods of solving the system of nonlinear equations without the Jacobian: *Broyden's Method*, *Anderson Accerlation*, `DF-SANE`, `SQUAREM` and the Jacobian based methods: Powell's Method (`MINPACK-hybr`), and *Levenberg-Marquardt*.

We focus mainly on the computational burden and report the results in Table 5. For the simple and complex simulations we vary the coefficient on the constant term β_0. A smaller value of β_0 leads to a larger share for the outside good. As shown by Dubé et al. (2012), a smaller outside good share implies a larger value for the Lipschitz constant for the contraction and it generally increases the number of iterations. Several patterns emerge. As we shrink the outside good share from $s_0 = 0.91 \to 0.26$ (and increase the Lipschitz constant) the number of contraction evaluations needed increases approximately by a factor of 5× for the *Simple* and *Complex* simulations. As expected, the Jacobian based routines are unaffected by varying the Lipschitz constant.[42] With the `SQUAREM` iteration scheme the number of iterations increases, but by only 70%.

In general, we find *Broyden's method* and the Quasi-Newton *Powell's method* are too unreliable for general use (at least absent additional modifications). Even on simple problems, they can fail to

[42] This is consistent with the findings of Dubé et al. (2012) using the MPEC method.

converge to a fixed point. We also find that *Anderson mixing* is often slower than direct iteration on the contraction mapping. The `DF-SANE` algorithm performs substantially better than standard fixed point iteration, but less well than our two preferred algorithms `SQUAREM` (an accelerated fixed point iteration) and *Levenberg Marquardt* (a Gauss-Newton solver with an analytic Jacobian).

In terms of other speedups, the effects of the `SQUAREM` iteration scheme are substantial. It reduces the number of iterations between $4-8\times$ without substantially increasing the cost per iteration. This is because it approximates the Newton step without actually computing the costly Jacobian. It performs well across all settings, but does particularly well in the BLP example which includes a supply side.

The *Levenberg-Marquardt* algorithm reduces the number of iterations, but at a higher cost per iteration. On some of the more difficult problems (those with a small outside good share) it performs as much as $10\times$ faster than fixed point iteration, and at other times roughly as fast as `SQUAREM`. The speedup is particularly large for the case of the RCNL model, where the contraction is dampened by $(1-\rho)$. In that case when $\rho \to 1$ the contraction becomes arbitrarily slow. This performs somewhat better than other Quasi-Newton routines in Reynaerts et al. (2012) were reported to perform.

We also consider some commonly employed tricks in the literature to speed up the contraction mapping and report the results in Table 4. In all cases, we employ the `SQUAREM` algorithm. We consider two commonly employed "tweaks" from the literature: (1) working with $\exp[\delta_{jt}]$ rather than δ_{jt} to avoid taking logs and eliminating a division in the share computation and (2) "hot-start" where the starting value for the iterative procedure is the value of $\delta_{\cdot t}^{n-1}$ which solved the system of equations for the previous guess of θ_2. In addition, we consider the potential cost of our overflow safe modification to the $\log-\text{sum}-\exp$ function.

The results in Table 4 are largely underwhelming. Once we adopt the `SQUAREM` accelerated iteration the additional tricks to speed up the problem (exponentiated δ's) seem to have little benefit.[43] The "hot-start" approach was able to reduce the number of iterations between by between 10-20% which is possibly worth considering.[44] One clear recommendation is the $(\log\text{sum}\exp)$ trick from Section 3.3 which reduces the chances of *overflow* problems. This seems relatively low-cost and reduces the possibility that the iteration routine fails to find a solution to the system of equations.

Numerical Integration

Given extensive past work by Heiss and Winschel (2008), Judd and Skrainka (2011), Skrainka (2012b), and Freyberger (2015) the choice of integration method has both a striking effect on the small sample performance of the estimator, and is perhaps not surprising. We report the results of our experiments in Table 6. We compare a *Gauss Hermite Product Rule* rule of degree 17 which

[43] Because the scale of the exponentiated problem was different we had to set the tolerance to $\epsilon^{tol} = 10^{-12}$ because of lost precision.

[44] This introduces a potential numerical problem where $Q(\theta^h)$ need not evaluate to precisely the same quantity depending on θ^{h-1} though for sufficient tolerances this should not be an issue. In practice we still recommend a tighter tolerance of $\epsilon^{tol} = 10^{-14}$ as the default.

has $I_t = 9$ nodes in dimension one and $I_t = 9^2 = 81$ nodes in dimension 2 with a Monte Carlo rule which has exactly three times as many nodes $I_t = 27$ or $I_t = 243$. Here we see an order of magnitude reduction in Median Bias (from 37% to 2% for our *Simple* simulation) and reductions in MAD by a factor of $3 - 12\times$ particularly for the random coefficient terms (as one might expect).

[QMC Rules? Halton, etc. Should we increase the number of MC draws to get comparable finite sample performance? Why is computational time the same with 3× draws?]

Choice of Instruments

We also consider several different alternative choices of instruments for our Monte Carlo simulations. We consider models which include a supply side, as well as models estimated using the demand side only. We present a slightly different construction of the Chamberlain (1987) *optimal instruments* for the BLP problem in Section 3.6 than those proposed in Reynaert and Verboven (2014).

In Table 7, we present simulation results using the original BLP instruments from equation (26), as well as both forms of Gandhi and Houde (2017)'s differentiation IV *Local* an *Quadratic* from (27), as well as our *approximate* version of the feasible optimal instruments in the spirit of Chamberlain (1987) from equation (25).

We find that in most settings the "optimal instruments" perform best, consistent with the findings of Reynaert and Verboven (2014). We also find that the differentiation IV substantially outperform the original BLP instruments, as Gandhi and Houde (2017) suggest. The exception is the RCNL model where our formula for the differentiation IV in equation (27) does not account for the group structure.

We also find that the "optimal instruments" perform much better when the supply side is included than with the demand side alone, while this seems to have a limited impact under the other forms of instruments. In fact, we find that with both optimal instruments and a supply side, the bias is all but eliminated in most of our Monte Carlo experiments, while the MAD (particularly for the random coefficients) is substantially reduced. This is not consistent with Reynaert and Verboven (2014) who found that including the supply side had little effect once "optimal instruments" were used.

We compute the optimal instruments using three different techniques: the *Approximate* approach employed by Berry et al. (1999) where $(\xi^*, \omega^*) = (0, 0)$ (their unconditional expectation), and the *Asymptotic* approach where a bivariate normal distribution is estimated for $(\xi_{jt}, \omega_{jg}) \sim N(0, \widehat{\Omega})$ and then 100 draws are taken from that distribution to compute the expected Jacobian of the moment conditions, as well as the *empirical* distribution which samples from replacement from the joint distribution of $(\widehat{xi}_{jt}, \widehat{\omega}_{jt})$. We present results in Table 8. Consistent with the prior literature the *Approximate* version of optimal instruments appear to provide nearly all of the benefits of the *Asymptotic* or *Empirical* version at much lower cost.

We also show the value of both the supply side restrictions and the optimal instruments is largest when the "exogenous cost shifter" is weakest. We report results in Table 9 where we reduce the magnitude of the coefficient γ_2 which governs how responsive marginal costs are to the exogenous cost shifter. As we reduce this coefficient it reduces the correlation between (p, z). When the cost shifting instrument is very weak $\rho \approx 0.05$, this increases both the bias and the variance of the price parameter α consistent with Armstrong (2016). However, if we include the supply restrictions (under optimal instruments) we are effectively able to eliminate the bias and substantially reduce the variance of the estimates. This finding appears to be novel as Reynaert and Verboven (2014) don't find substantial benefits of including supply side restrictions once the optimal instruments are employed.[45]

Consistent with Gandhi and Houde (2017) our recommendation is to start with the *differentiation IV* in the first stage, and then compute the "optimal instruments" in the second stage. The substantial small sample benefits of including optimal instruments suggest that they should be employed more widely, particularly when there are multiple random coefficients. Our hope is that `pyblp` makes these instruments available to a wider set of researchers as it substantially reduces the cost by providing analytic Jacobian calculations and rapidly solving for the pricing equilibrium needed to generate the instruments.

Optimization Algorithms

Similar to Knittel and Metaxoglou (2014) we consider a broad array of different optimization algorithms and report our results regarding convergence in Table 10. We report our results for the parameter estimates for different Appendix Table A-1.

Our findings are somewhat different from those in Knittel and Metaxoglou (2014). Whereas those authors found that many different optimization algorithms found a variety of local minima and often failed to converge to a valid minimum, we obtain basically the opposite result. Using a broad array of optimization routines: three algorithms within `KNITRO` and six algorithms within `scipy.optimize` we find that $> 99\%$ of all simulation runs converge to a local maximum which we define as having an infinity norm of the gradient sufficiently close to zero and a positive semi-definite Hessian matrix (all eigenvalues weakly positive). Even the non-derivative based Nelder-Mead algorithm, which we do not recommend, appears to do reasonably well at finding an optimum.

In general our preferred algorithms are probably `KNITRO-Interior/Direct` and `BFGS` as they provide the best speed and reliability. We also find that `SLSQP` works well when parameter bound constrains are included. We should caution that our simulations are simple enough to be run thousands of times, so it may not be surprising that most optimization software packages appear to work well. It may also be the case that various numerical fixes and improvements to the fixed point iteration may have resolved some of the issues around optimization.

[45] This may have to do with the fact that they construct optimal instruments differently for the simultaneous supply and demand problem so that the model is just identified whereas our approach constructs them to be overidentified.

We find similar success with the Nevo and BLP problems.

[Appendix: Do we have a table of BLP or Nevo under different starting values/optimization routines?]

Problem Size

We might also be interested to how the BLP problem scales as we vary the size of the problem. We report our results in Table 11. We find that computational time is roughly linear in the number of markets T, while it is grows at a rate closer to $\sqrt{J}$ as we increase the number of products when we use only the demand side. When we include both supply and demand the computational time appears to grow more quickly than J.

The problems in our Monte Carlo exercise are fairly simple, but it appears as if using only demand side restrictions, $T = 40$ makrets is roughly enough to obtain "reasonable" parameter estimates in terms of bias and efficiency.

6 Conclusion

Our goal has to be to review recent methodological developments related to the BLP problem, and collect them not only in a single paper, but a single software package `pyblp`. We've provided a list of best practices with numerical evidence to support our recommendations. Our hope is that these practices can now be made available to a wider swath of researchers, and provide a common platform which should facilitate replication.

Some of these results such as those related to numerical integration (Sparse Grids) are well established in the literature Skrainka (2012b) and Heiss and Winschel (2008). In other cases, we find other algorithms (such as *Levenberg-Marquardt* for solving the fixed point relationship) which perform as well as best in class algorithms such as the `SQUAREM` approach Reynaerts et al. (2012) and in some cases (such as the RCNL model) substantially better. We also document how lesser known methodologies such as the reformulation of the pricing equilibria by Morrow and Skerlos (2011) can substantially improve both speed and reliability.

In addition, we present some methodological results which we believe to be novel. We show how with a slight reformulation of the nested-fixed point problem, it is possible to include high dimensional fixed effects in models with simultaneous supply and demand. We also provide a somewhat different expression for optimal instruments than the prior literature (Reynaert and Verboven, 2014) which makes clear the over-identifying restrictions implied by the supply side. Also novel,we find that optimal instruments when combined with a correctly specified supply side are extremely valuable. Consistent with prior work, we find the gains to "optimal instruments" to be substantial such that they should nearly always be employed. Thankfully, we have made this process extremely straightforward in `pyblp`.

		Nevo (2000)		Replication		Restricted	
Parameters	Variable	Estimate	SE	Estimate	SE	Estimate	SE
Means (β)	Price	-32.433	7.743	-62.730	14.803	-32.019	2.304
Standard Deviations (Σ)	Price	1.848	1.075	3.312	1.340	1.803	0.920
	Constant	0.377	0.129	0.558	0.163	0.375	0.120
	Sugar	0.004	0.012	-0.006	0.014	-0.004	0.012
	Mushy	0.081	0.205	0.093	0.185	0.086	0.193
Interactions (Π)	Price × Income	16.598	172.334	588.325	270.441	4.187	4.638
	Price × Income^2	-0.659	8.955	-30.192	14.101		
	Price × Child	11.625	5.207	11.055	4.123	11.755	5.197
	Constant × Income	3.089	1.213	2.292	1.209	3.101	1.054
	Constant × Age	1.186	1.016	1.284	0.631	1.198	1.048
	Sugar × Income	-0.193	0.005	-0.385	0.121	-0.190	0.035
	Sugar × Age	0.029	0.036	0.052	0.026	0.028	0.032
	Mushy × Income	1.468	0.697	0.748	0.802	1.495	0.648
	Mushy × Age	-1.514	1.103	-1.353	0.667	-1.539	1.107
Mean Own Elasticity ($\bar{\varepsilon}_{jj}$)				-3.618		-3.702	
Objective ($\bar{g}'W\bar{g}$)		14.9		4.6		15.4	

Table 1: Nevo (2000) Replication

		BLP (1995)		Replication		PC Instruments		Optimal Instruments	
Parameters	Variable	Estimate	SE	Estimate	SE	Estimate	SE	Estimate	SE
Means (β)	Constant	-7.061	0.941	-7.817	0.941	-8.467	2.205	-8.538	1.016
	HP / weight	2.883	2.019	-3.540	6.760	3.391	2.958	1.853	2.300
	Air	1.521	0.891	1.279	3.911	1.347	3.136	1.069	1.154
	MP$	-0.122	0.320	-2.004	1.108	-1.205	0.797	-1.649	0.798
	Size	3.460	0.610	2.463	1.348	2.305	1.250	1.590	0.784
Standard Deviations (Σ)	Constant	3.612	1.485	0.000	1.915	0.610	4.084	1.319	1.832
	HP / weight	4.628	1.885	10.000	8.257	0.000	5.863	3.315	5.262
	Air	1.818	1.695	0.198	7.780	0.000	6.389	0.846	1.624
	MP$	1.050	0.272	2.563	1.201	1.558	0.937	2.071	0.822
	Size	2.056	0.585	0.000	2.569	0.000	2.371	1.209	1.066
Term on Price (α)	$\ln(y-p)$	43.501	6.427	7.403	0.893	7.555	0.769	7.962	2.519
Supply Side Terms (γ)	Constant	0.952	0.194	2.413	0.207	2.391	0.198	2.423	0.178
	ln(HP / weight)	0.477	0.056	0.778	0.136	0.781	0.144	0.760	0.198
	Air	0.619	0.038	0.969	0.077	0.967	0.069	0.948	0.093
	ln(MPG)	-0.415	0.055	-0.614	0.123	-0.639	0.129	-0.616	0.153
	ln(Size)	-0.046	0.081	0.228	0.176	0.313	0.165	0.249	0.184
	Trend	0.019	0.002	0.015	0.003	0.017	0.004	0.016	0.004
Mean Own Elasticity ($\bar{\varepsilon}_{jj}$)				-2.825		-2.775		-2.939	
Objective ($\bar{g}'W\bar{g}$)				396.9		460.1		96.4	

Table 2: BLP (95/99) Replication

We approximate $\ln(y-p)$ with $\frac{p}{\alpha_i}$ following BLP99. This prevents us from exactly replicating the BLP95 paper.
Replication instruments consist of sums of product characteristics for own and competitor firms.
PC instruments takes all second order interactions of instruments and projects down to principal components.
Optimal instruments computes the approximation to the supply and demand optimal instruments using $\xi_{jt} = \omega_{jt} = 0$

[Add density plot for markups for BLP/Nevo]

Problem	Fixed Effects	Absorbed	Seconds	Megabytes
Simple Simulation	125	No	21.34	92.57
Simple Simulation	125	Yes	13.38	0.70
Simple Simulation	125×125	No	25.97	212.45
Simple Simulation	125×125	Yes	11.98	1.07
Simple Simulation	25×625	No	51.69	644.99
Simple Simulation	25×625	Yes	11.91	1.06
Simple Simulation	$25 \times 25 \times 25$	No	18.38	37.33
Simple Simulation	$25 \times 25 \times 25$	Yes	13.73	1.26
Nevo Example	24	No	18.21	0.04
Nevo Example	24	Yes	17.18	0.00

Table 3: Fixed Effects Absorption

Problem	Median s_0	Overflow Safe	Type	Initial δ	Mean Milliseconds	Mean Evaluations	Percent Converged
Simple Simulation ($\beta_0 = -7$)	0.91	Yes	δ	δ^0	1.99	17.29	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	No	δ	δ^0	1.74	17.28	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	Yes	$\exp(\delta)$	δ^0	1.98	17.29	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	No	$\exp(\delta)$	δ^0	1.74	17.28	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	Yes	δ	δ^{n-1}	1.51	15.32	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	Yes	δ	δ^0	3.88	34.38	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	No	δ	δ^0	3.39	34.37	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	Yes	$\exp(\delta)$	δ^0	3.88	34.38	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	No	$\exp(\delta)$	δ^0	3.39	34.37	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	Yes	δ	δ^{n-1}	2.70	27.38	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	Yes	δ	δ^0	2.93	16.87	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	No	δ	δ^0	2.58	16.87	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	Yes	$\exp(\delta)$	δ^0	2.98	16.87	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	No	$\exp(\delta)$	δ^0	2.58	16.87	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	Yes	δ	δ^{n-1}	2.39	14.70	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	Yes	δ	δ^0	6.03	34.11	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	No	δ	δ^0	5.21	34.11	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	Yes	$\exp(\delta)$	δ^0	5.88	34.11	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	No	$\exp(\delta)$	δ^0	5.29	34.11	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	Yes	δ	δ^{n-1}	4.09	25.77	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	Yes	δ	δ^0	8.78	43.28	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	No	δ	δ^0	7.54	43.35	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	Yes	$\exp(\delta)$	δ^0	8.76	43.28	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	No	$\exp(\delta)$	δ^0	7.66	43.35	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	Yes	δ	δ^{n-1}	7.16	37.42	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	Yes	δ	δ^0	13.25	63.80	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	No	δ	δ^0	11.59	63.86	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	Yes	$\exp(\delta)$	δ^0	13.11	63.80	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	No	$\exp(\delta)$	δ^0	11.45	63.86	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	Yes	δ	δ^{n-1}	10.74	56.57	100.00%
Nevo Example	0.54	Yes	δ	δ^0	3.17	26.16	100.00%
Nevo Example	0.54	No	δ	δ^0	2.91	26.16	100.00%
Nevo Example	0.54	Yes	$\exp(\delta)$	δ^0	3.30	26.16	100.00%
Nevo Example	0.54	No	$\exp(\delta)$	δ^0	2.91	26.16	100.00%
Nevo Example	0.54	Yes	δ	δ^{n-1}	2.55	20.95	100.00%
BLP Example	0.89	Yes	δ	δ^0	15.70	18.55	100.00%
BLP Example	0.89	No	δ	δ^0	13.96	18.56	100.00%
BLP Example	0.89	Yes	$\exp(\delta)$	δ^0	16.24	18.55	100.00%
BLP Example	0.89	No	$\exp(\delta)$	δ^0	14.13	18.56	100.00%
BLP Example	0.89	Yes	δ	δ^{n-1}	12.02	14.83	100.00%

Table 4: Fixed Point Tricks

Problem	Median s_0	Algorithm	Tolerance	Norm	Jacobian	Mean Milliseconds	Mean Evaluations	Percent Converged
Simple Simulation ($\beta_0 = -7$)	0.91	Iteration	Absolute	L^∞	No	4.72	46.79	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	Broyden	Absolute	L^∞	No	18.40	54.90	61.94%
Simple Simulation ($\beta_0 = -7$)	0.91	Anderson	Absolute	L^∞	No	58.21	171.27	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	DF-SANE	Absolute	L^∞	No	2.75	17.54	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	SQUAREM	Absolute	L^∞	No	1.99	17.29	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	SQUAREM	Relative	L^2	No	2.12	16.45	100.00%
Simple Simulation ($\beta_0 = -7$)	0.91	Modified Powell	Relative	L^2	Yes	2.82	16.27	33.21%
Simple Simulation ($\beta_0 = -7$)	0.91	LevenbergMarquardt	Relative	L^2	Yes	2.05	9.03	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	Iteration	Absolute	L^∞	No	22.88	229.19	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	Broyden	Absolute	L^∞	No	24.67	75.12	96.11%
Simple Simulation ($\beta_0 = -1$)	0.26	Anderson	Absolute	L^∞	No	220.88	992.62	71.12%
Simple Simulation ($\beta_0 = -1$)	0.26	DF-SANE	Absolute	L^∞	No	5.83	38.00	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	SQUAREM	Absolute	L^∞	No	3.88	34.38	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	SQUAREM	Relative	L^2	No	4.28	33.69	100.00%
Simple Simulation ($\beta_0 = -1$)	0.26	Modified Powell	Relative	L^2	Yes	3.00	17.39	12.42%
Simple Simulation ($\beta_0 = -1$)	0.26	LevenbergMarquardt	Relative	L^2	Yes	2.05	8.95	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	Iteration	Absolute	L^∞	No	7.81	48.28	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	Broyden	Absolute	L^∞	No	23.70	59.92	70.79%
Complex Simulation ($\beta_0 = -7$)	0.91	Anderson	Absolute	L^∞	No	79.89	201.12	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	DF-SANE	Absolute	L^∞	No	3.87	18.01	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	SQUAREM	Absolute	L^∞	No	2.93	16.87	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	SQUAREM	Relative	L^2	No	3.04	16.08	100.00%
Complex Simulation ($\beta_0 = -7$)	0.91	Modified Powell	Relative	L^2	Yes	3.74	15.28	32.82%
Complex Simulation ($\beta_0 = -7$)	0.91	LevenbergMarquardt	Relative	L^2	Yes	2.64	9.00	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	Iteration	Absolute	L^∞	No	36.93	228.54	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	Broyden	Absolute	L^∞	No	32.56	82.92	96.62%
Complex Simulation ($\beta_0 = -1$)	0.27	Anderson	Absolute	L^∞	No	435.96	1,237.61	59.52%
Complex Simulation ($\beta_0 = -1$)	0.27	DF-SANE	Absolute	L^∞	No	8.26	38.13	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	SQUAREM	Absolute	L^∞	No	6.03	34.11	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	SQUAREM	Relative	L^2	No	6.31	33.48	100.00%
Complex Simulation ($\beta_0 = -1$)	0.27	Modified Powell	Relative	L^2	Yes	4.37	17.54	11.10%
Complex Simulation ($\beta_0 = -1$)	0.27	LevenbergMarquardt	Relative	L^2	Yes	2.65	8.95	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	Iteration	Absolute	L^∞	No	30.77	164.00	92.32%
RCNL Simulation ($\rho = 0.5$)	0.91	Broyden	Absolute	L^∞	No	20.35	49.00	29.00%
RCNL Simulation ($\rho = 0.5$)	0.91	Anderson	Absolute	L^∞	No	283.37	645.20	29.88%
RCNL Simulation ($\rho = 0.5$)	0.91	DF-SANE	Absolute	L^∞	No	10.01	41.20	98.88%
RCNL Simulation ($\rho = 0.5$)	0.91	SQUAREM	Absolute	L^∞	No	8.78	43.28	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	SQUAREM	Relative	L^2	No	8.76	40.17	100.00%
RCNL Simulation ($\rho = 0.5$)	0.91	Modified Powell	Relative	L^2	Yes	4.74	16.85	53.32%
RCNL Simulation ($\rho = 0.5$)	0.91	LevenbergMarquardt	Relative	L^2	Yes	3.46	10.40	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	Iteration	Absolute	L^∞	No	60.42	318.10	92.22%
RCNL Simulation ($\rho = 0.8$)	0.91	Broyden	Absolute	L^∞	No	13.61	35.03	2.50%
RCNL Simulation ($\rho = 0.8$)	0.91	Anderson	Absolute	L^∞	No	521.57	1,279.61	3.50%
RCNL Simulation ($\rho = 0.8$)	0.91	DF-SANE	Absolute	L^∞	No	15.42	61.28	99.04%
RCNL Simulation ($\rho = 0.8$)	0.91	SQUAREM	Absolute	L^∞	No	13.25	63.80	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	SQUAREM	Relative	L^2	No	12.86	59.02	100.00%
RCNL Simulation ($\rho = 0.8$)	0.91	Modified Powell	Relative	L^2	Yes	6.55	23.15	60.98%
RCNL Simulation ($\rho = 0.8$)	0.91	LevenbergMarquardt	Relative	L^2	Yes	4.62	13.87	100.00%
Nevo Example	0.54	Iteration	Absolute	L^∞	No	10.60	95.70	99.97%
Nevo Example	0.54	Broyden	Absolute	L^∞	No	1.34	4.45	1.15%
Nevo Example	0.54	Anderson	Absolute	L^∞	No	90.38	281.15	98.86%
Nevo Example	0.54	DF-SANE	Absolute	L^∞	No	4.75	27.33	100.00%
Nevo Example	0.54	SQUAREM	Absolute	L^∞	No	3.17	26.16	100.00%
Nevo Example	0.54	SQUAREM	Relative	L^2	No	3.49	24.72	100.00%
Nevo Example	0.54	Modified Powell	Relative	L^2	Yes	3.41	17.57	29.69%
Nevo Example	0.54	LevenbergMarquardt	Relative	L^2	Yes	2.26	9.50	100.00%
BLP Example	0.89	Iteration	Absolute	L^∞	No	31.59	39.13	100.00%
BLP Example	0.89	Broyden	Absolute	L^∞	No	79.71	74.30	15.00%
BLP Example	0.89	Anderson	Absolute	L^∞	No	229.17	210.16	97.25%
BLP Example	0.89	DF-SANE	Absolute	L^∞	No	29.92	34.27	95.00%
BLP Example	0.89	SQUAREM	Absolute	L^∞	No	15.70	18.55	100.00%
BLP Example	0.89	SQUAREM	Relative	L^2	No	14.93	17.21	100.00%
BLP Example	0.89	Modified Powell	Relative	L^2	Yes	26.51	18.49	32.35%
BLP Example	0.89	LevenbergMarquardt	Relative	L^2	Yes	25.86	12.10	99.41%

Table 5: Fixed Point Algorithms

					True Value				Median Bias				Median Absolute Error			
Simulation	Supply	Integration	I_t	Seconds	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ
Simple	No	Monte Carlo	100	2.0	-1	3			0.165	-0.649			0.281	0.652		
Simple	No	Monte Carlo	1,000	6.1	-1	3			0.152	-0.109			0.253	0.195		
Simple	No	Product Rule	9^2	1.5	-1	3			0.157	-0.026			0.246	0.181		
Simple	Yes	Monte Carlo	100	6.7	-1	3			0.107	-0.659			0.227	0.660		
Simple	Yes	Monte Carlo	1,000	23.1	-1	3			0.030	-0.091			0.178	0.186		
Simple	Yes	Product Rule	9^2	4.8	-1	3			0.012	0.013			0.171	0.160		
Complex	No	Monte Carlo	100	3.6	-1	3	0.2		0.205	-0.821	-0.059		0.299	0.827	0.106	
Complex	No	Monte Carlo	1,000	12.3	-1	3	0.2		0.079	-0.195	-0.007		0.258	0.274	0.126	
Complex	No	Product Rule	9^1	4.0	-1	3	0.2		0.001	-0.124	0.069		0.262	0.233	0.159	
Complex	Yes	Monte Carlo	100	11.1	-1	3	0.2		0.094	-0.676	-0.129		0.256	0.711	0.142	
Complex	Yes	Monte Carlo	1,000	37.6	-1	3	0.2		0.010	-0.118	-0.046		0.197	0.230	0.119	
Complex	Yes	Product Rule	9^1	10.7	-1	3	0.2		-0.066	-0.047	0.050		0.192	0.178	0.133	
RCNL	No	Monte Carlo	100	9.7	-1	3		0.5	0.198	-0.705		0.032	0.295	0.706		0.053
RCNL	No	Monte Carlo	1,000	36.8	-1	3		0.5	0.126	-0.152		-0.001	0.242	0.224		0.026
RCNL	No	Product Rule	9^2	7.5	-1	3		0.5	0.102	-0.009		-0.006	0.229	0.199		0.025
RCNL	Yes	Monte Carlo	100	27.1	-1	3		0.5	0.097	-0.614		0.037	0.129	0.615		0.046
RCNL	Yes	Monte Carlo	1,000	105.0	-1	3		0.5	0.026	-0.097		0.005	0.109	0.189		0.020
RCNL	Yes	Product Rule	9^2	17.5	-1	3		0.5	0.009	-0.013		0.002	0.110	0.174		0.018

Table 6: Alternative Integration Methods

					True Value				Median Bias				Median Absolute Error			
Simulation	Supply	Z_D	Z_S	Seconds	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ
Simple	No	BLP		1.0	-1	3			0.111	0.022			0.227	0.418		
Simple	No	Local		0.6	-1	3			0.077	0.021			0.240	0.288		
Simple	No	Quadratic		0.6	-1	3			0.090	-0.026			0.250	0.373		
Simple	No	Approximate	Approximate	1.5	-1	3			0.157	-0.026			0.246	0.181		
Simple	Yes	BLP		2.7	-1	3			0.111	0.022			0.226	0.418		
Simple	Yes	BLP	BLP	3.1	-1	3			0.026	-0.001			0.231	0.404		
Simple	Yes	Local		1.3	-1	3			0.077	0.021			0.240	0.287		
Simple	Yes	Quadratic		1.3	-1	3			0.089	-0.026			0.250	0.373		
Simple	Yes	Approximate	Approximate	4.8	-1	3			0.012	0.013			0.171	0.160		
Complex	No	BLP		2.7	-1	3	0.2		-0.020	-0.478	0.019		0.263	0.845	0.200	
Complex	No	Local		1.7	-1	3	0.2		-0.201	-0.235	0.037		0.428	0.452	0.200	
Complex	No	Quadratic		1.6	-1	3	0.2		-0.136	0.256	-0.155		0.437	0.547	0.200	
Complex	No	Approximate	Approximate	4.0	-1	3	0.2		0.001	-0.124	0.069		0.262	0.233	0.159	
Complex	Yes	BLP		6.9	-1	3	0.2		-0.025	-0.481	0.031		0.262	0.854	0.200	
Complex	Yes	BLP	BLP	7.2	-1	3	0.2		-0.100	-0.402	0.103		0.275	0.549	0.163	
Complex	Yes	Local		4.4	-1	3	0.2		-0.201	-0.218	0.038		0.423	0.444	0.200	
Complex	Yes	Quadratic		4.0	-1	3	0.2		-0.157	0.283	-0.044		0.452	0.552	0.200	
Complex	Yes	Approximate	Approximate	10.7	-1	3	0.2		-0.066	-0.047	0.050		0.192	0.178	0.133	
RCNL	No	BLP		4.7	-1	3		0.5	0.139	-0.715		0.028	0.274	1.174		0.063
RCNL	No	Local		4.3	-1	3		0.5	0.236	-0.809		0.105	0.426	1.631		0.200
RCNL	No	Quadratic		5.3	-1	3		0.5	0.320	-0.769		0.151	0.484	1.363		0.226
RCNL	No	Approximate	Approximate	7.5	-1	3		0.5	0.102	-0.009		-0.006	0.229	0.199		0.025
RCNL	Yes	BLP		9.7	-1	3		0.5	0.138	-0.701		0.028	0.272	1.171		0.064
RCNL	Yes	BLP	BLP	11.3	-1	3		0.5	0.057	-0.624		0.023	0.234	1.154		0.060
RCNL	Yes	Local		8.0	-1	3		0.5	0.232	-0.775		0.103	0.418	1.591		0.196
RCNL	Yes	Quadratic		9.5	-1	3		0.5	0.309	-0.726		0.141	0.473	1.363		0.223
RCNL	Yes	Approximate	Approximate	17.5	-1	3		0.5	0.009	-0.013		0.002	0.110	0.174		0.018

Table 7: Alternative Instruments

BLP instruments are sums of product characteristics from eqn (26).
Local and Quadratic instruments follow eqn (27) and Gandhi and Houde (2017).
The *Approximate* form of the feasible optimal instruments are included.

				True Value				Median Bias				Median Absolute Error			
Simulation	Supply	Optimality	Seconds	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ
Simple	No	Approximate	1.5	-1	3			0.157	-0.026			0.246	0.181		
Simple	No	Asymptotic	4.3	-1	3			0.157	-0.024			0.246	0.181		
Simple	No	Empirical	4.3	-1	3			0.157	-0.022			0.245	0.178		
Simple	Yes	Approximate	4.8	-1	3			0.012	0.013			0.171	0.160		
Simple	Yes	Asymptotic	19.0	-1	3			0.029	0.015			0.180	0.153		
Simple	Yes	Empirical	19.0	-1	3			0.012	0.011			0.177	0.159		
Complex	No	Approximate	4.0	-1	3	0.2		0.001	-0.124	0.069		0.262	0.233	0.159	
Complex	No	Asymptotic	7.8	-1	3	0.2		0.000	-0.122	0.069		0.266	0.235	0.159	
Complex	No	Empirical	7.7	-1	3	0.2		-0.003	-0.121	0.065		0.252	0.237	0.154	
Complex	Yes	Approximate	10.7	-1	3	0.2		-0.066	-0.047	0.050		0.192	0.178	0.133	
Complex	Yes	Asymptotic	34.6	-1	3	0.2		-0.038	-0.046	0.029		0.212	0.222	0.123	
Complex	Yes	Empirical	34.6	-1	3	0.2		-0.040	-0.058	0.028		0.213	0.213	0.119	
RCNL	No	Approximate	7.5	-1	3		0.5	0.102	-0.009		-0.006	0.229	0.199		0.025
RCNL	No	Asymptotic	12.0	-1	3		0.5	0.108	-0.011		-0.006	0.228	0.198		0.025
RCNL	No	Empirical	11.8	-1	3		0.5	0.106	-0.014		-0.006	0.223	0.199		0.024
RCNL	Yes	Approximate	17.5	-1	3		0.5	0.009	-0.013		0.002	0.110	0.174		0.018
RCNL	Yes	Asymptotic	53.8	-1	3		0.5	0.010	0.000		0.001	0.115	0.179		0.019
RCNL	Yes	Empirical	53.7	-1	3		0.5	0.014	-0.005		0.001	0.112	0.173		0.019

Table 8: Form of Optimal Instruments

					True Value				Median Bias				Median Absolute Error			
Simulation	γ_2	Corr(p,z)	Supply	Seconds	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ
Simple	0.1	0.052	No	1.5	-1	3			0.267	-0.051			0.350	0.190		
Simple	0.1	0.052	Yes	5.1	-1	3			0.010	0.005			0.205	0.162		
Simple	0.2	0.102	No	1.5	-1	3			0.157	-0.026			0.246	0.181		
Simple	0.2	0.102	Yes	4.8	-1	3			0.012	0.013			0.171	0.160		
Simple	0.4	0.199	No	1.5	-1	3			0.051	-0.001			0.126	0.170		
Simple	0.4	0.199	Yes	4.6	-1	3			0.008	0.017			0.111	0.148		
Simple	0.8	0.376	No	1.5	-1	3			0.014	0.011			0.062	0.167		
Simple	0.8	0.376	Yes	4.7	-1	3			-0.000	0.009			0.060	0.143		
Complex	0.1	0.054	No	4.0	-1	3	0.2		0.103	-0.138	0.062		0.319	0.243	0.166	
Complex	0.1	0.054	Yes	10.8	-1	3	0.2		-0.053	-0.047	0.040		0.207	0.188	0.139	
Complex	0.2	0.104	No	4.0	-1	3	0.2		0.001	-0.124	0.069		0.262	0.233	0.159	
Complex	0.2	0.104	Yes	10.7	-1	3	0.2		-0.066	-0.047	0.050		0.192	0.178	0.133	
Complex	0.4	0.204	No	4.1	-1	3	0.2		-0.064	-0.089	0.060		0.183	0.219	0.158	
Complex	0.4	0.204	Yes	10.7	-1	3	0.2		-0.049	-0.039	0.043		0.157	0.167	0.127	
Complex	0.8	0.384	No	4.1	-1	3	0.2		-0.061	-0.067	0.045		0.143	0.216	0.147	
Complex	0.8	0.384	Yes	10.9	-1	3	0.2		-0.045	-0.031	0.036		0.125	0.164	0.122	
RCNL	0.1	0.050	No	7.3	-1	3		0.5	0.200	-0.017		-0.011	0.306	0.201		0.027
RCNL	0.1	0.050	Yes	18.0	-1	3		0.5	0.009	-0.006		0.002	0.121	0.175		0.019
RCNL	0.2	0.097	No	7.5	-1	3		0.5	0.102	-0.009		-0.006	0.229	0.199		0.025
RCNL	0.2	0.097	Yes	17.5	-1	3		0.5	0.009	-0.013		0.002	0.110	0.174		0.018
RCNL	0.4	0.189	No	7.7	-1	3		0.5	0.038	-0.001		-0.002	0.133	0.193		0.022
RCNL	0.4	0.189	Yes	17.5	-1	3		0.5	0.010	-0.011		0.001	0.089	0.172		0.019
RCNL	0.8	0.358	No	7.8	-1	3		0.5	0.007	-0.000		-0.002	0.079	0.181		0.021
RCNL	0.8	0.358	Yes	17.6	-1	3		0.5	0.005	-0.012		0.001	0.063	0.169		0.017

Table 9: Varying Instrument Strength

							Percent of Runs		Median, First GMM Step			
Simulation	Supply	P	Software	Algorithm	Bounds	Gradient	Converged	PSD Hessian	Seconds	Evaluations	$q = \bar{g}'W\bar{g}$	$\|\nabla q\|_\infty$
Simple	No	1	Knitro	Interior/Direct	Yes	Yes	100.00%	99.90%	0.7	13	9.63E-19	5.26E-10
Simple	No	1	Knitro	Active Set	Yes	Yes	100.00%	100.00%	0.5	12	9.17E-19	5.33E-10
Simple	No	1	Knitro	SQP	Yes	Yes	100.00%	100.00%	0.6	13	8.78E-19	5.29E-10
Simple	No	1	SciPy	SLSQP	Yes	Yes	100.00%	99.90%	0.6	7	7.45E-17	2.05E-08
Simple	No	1	SciPy	TNC	Yes	Yes	100.00%	100.00%	0.7	9	1.12E-18	1.87E-09
Simple	No	1	SciPy	L-BFGS-B	Yes	Yes	100.00%	100.00%	0.5	7	3.86E-17	1.20E-08
Simple	No	1	SciPy	BFGS	No	Yes	100.00%	99.90%	0.6	8	2.46E-17	9.55E-09
Simple	No	1	SciPy	Nelder-Mead	No	No	100.00%	100.00%	2.6	39	6.79E-12	8.80E-06
Simple	Yes	2	Knitro	Interior/Direct	Yes	Yes	100.00%	100.00%	1.9	18	4.12E-01	7.37E-12
Simple	Yes	2	Knitro	Active Set	Yes	Yes	100.00%	100.00%	1.7	17	4.10E-01	7.58E-12
Simple	Yes	2	Knitro	SQP	Yes	Yes	100.00%	99.90%	1.8	17	4.09E-01	9.14E-12
Simple	Yes	2	SciPy	SLSQP	Yes	Yes	100.00%	99.90%	1.5	11	4.09E-01	2.73E-06
Simple	Yes	2	SciPy	TNC	Yes	Yes	99.90%	100.00%	2.9	21	4.11E-01	2.29E-06
Simple	Yes	2	SciPy	L-BFGS-B	Yes	Yes	100.00%	100.00%	1.4	10	4.11E-01	9.43E-07
Simple	Yes	2	SciPy	BFGS	No	Yes	100.00%	100.00%	1.5	11	4.12E-01	1.03E-06
Simple	Yes	2	SciPy	Nelder-Mead	No	No	100.00%	100.00%	8.0	87	4.12E-01	2.17E-05
Complex	No	2	Knitro	Interior/Direct	Yes	Yes	100.00%	100.00%	1.4	20	8.48E-18	4.68E-09
Complex	No	2	Knitro	Active Set	Yes	Yes	100.00%	76.90%	1.4	18	3.16E-04	1.30E-09
Complex	No	2	Knitro	SQP	Yes	Yes	100.00%	92.70%	1.7	20	5.89E-17	2.85E-09
Complex	No	2	SciPy	SLSQP	Yes	Yes	100.00%	95.60%	1.5	12	2.66E-13	1.21E-06
Complex	No	2	SciPy	TNC	Yes	Yes	99.50%	96.60%	4.1	28	1.55E-13	2.79E-07
Complex	No	2	SciPy	L-BFGS-B	Yes	Yes	100.00%	92.50%	1.7	13	5.73E-14	5.16E-07
Complex	No	2	SciPy	BFGS	No	Yes	100.00%	100.00%	1.4	13	1.47E-14	3.52E-07
Complex	No	2	SciPy	Nelder-Mead	No	No	100.00%	100.00%	7.9	90	2.58E-11	1.83E-05
Complex	Yes	3	Knitro	Interior/Direct	Yes	Yes	100.00%	99.50%	4.0	25	5.95E-01	1.67E-11
Complex	Yes	3	Knitro	Active Set	Yes	Yes	99.50%	73.20%	4.1	23	5.16E-01	1.24E-11
Complex	Yes	3	Knitro	SQP	Yes	Yes	100.00%	98.40%	4.3	27	5.43E-01	3.35E-11
Complex	Yes	3	SciPy	SLSQP	Yes	Yes	99.70%	99.00%	5.6	17	5.62E-01	7.14E-06
Complex	Yes	3	SciPy	TNC	Yes	Yes	99.20%	99.10%	7.7	37	5.60E-01	1.91E-05
Complex	Yes	3	SciPy	L-BFGS-B	Yes	Yes	99.30%	98.60%	3.3	16	5.48E-01	2.03E-06
Complex	Yes	3	SciPy	BFGS	No	Yes	99.20%	99.20%	4.2	18	5.95E-01	1.76E-06
Complex	Yes	3	SciPy	Nelder-Mead	No	No	99.60%	99.20%	20.0	166	5.93E-01	3.18E-05
RCNL	No	2	Knitro	Interior/Direct	Yes	Yes	100.00%	99.50%	3.9	20	4.54E-19	2.60E-09
RCNL	No	2	Knitro	Active Set	Yes	Yes	100.00%	98.20%	3.1	19	5.39E-19	2.66E-09
RCNL	No	2	Knitro	SQP	Yes	Yes	100.00%	98.10%	3.5	21	5.25E-19	3.07E-09
RCNL	No	2	SciPy	SLSQP	Yes	Yes	100.00%	98.30%	3.0	13	2.09E-14	3.30E-06
RCNL	No	2	SciPy	TNC	Yes	Yes	99.90%	98.40%	5.6	26	2.01E-16	3.23E-07
RCNL	No	2	SciPy	L-BFGS-B	Yes	Yes	100.00%	98.50%	3.0	13	7.13E-17	1.72E-07
RCNL	No	2	SciPy	BFGS	No	Yes	100.00%	99.60%	2.5	13	4.05E-17	9.54E-08
RCNL	No	2	SciPy	Nelder-Mead	No	No	100.00%	99.70%	15.0	88	2.07E-11	1.13E-04
RCNL	Yes	3	Knitro	Interior/Direct	Yes	Yes	100.00%	100.00%	6.7	25	6.07E-01	1.54E-11
RCNL	Yes	3	Knitro	Active Set	Yes	Yes	100.00%	100.00%	6.6	26	5.77E-01	2.34E-11
RCNL	Yes	3	Knitro	SQP	Yes	Yes	100.00%	100.00%	7.0	26	5.78E-01	3.71E-11
RCNL	Yes	3	SciPy	SLSQP	Yes	Yes	100.00%	100.00%	5.8	17	5.79E-01	2.01E-05
RCNL	Yes	3	SciPy	TNC	Yes	Yes	99.60%	100.00%	11.1	34	5.64E-01	7.07E-05
RCNL	Yes	3	SciPy	L-BFGS-B	Yes	Yes	100.00%	100.00%	5.6	17	5.74E-01	2.50E-06
RCNL	Yes	3	SciPy	BFGS	No	Yes	99.90%	100.00%	4.8	16	6.08E-01	1.43E-06
RCNL	Yes	3	SciPy	Nelder-Mead	No	No	100.00%	100.00%	31.9	156	6.09E-01	1.49E-04

Table 10: Optimization Algorithms

					True Value				Median Bias				Median Absolute Error			
Simulation	Supply	T	J_f	Seconds	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ
Simple	No	20	$\{2,5,10\}$	1.5	-1	3			0.157	-0.026			0.246	0.181		
Simple	No	40	$\{2,5,10\}$	2.7	-1	3			0.091	-0.017			0.167	0.115		
Simple	No	100	$\{2,5,10\}$	6.6	-1	3			0.040	0.006			0.102	0.078		
Simple	No	20	$\{4,10,20\}$	1.8	-1	3			0.087	0.003			0.165	0.141		
Simple	No	20	$\{10,25,50\}$	2.7	-1	3			0.035	-0.022			0.116	0.102		
Simple	Yes	20	$\{2,5,10\}$	4.8	-1	3			0.012	0.013			0.171	0.160		
Simple	Yes	40	$\{2,5,10\}$	9.4	-1	3			0.013	0.008			0.122	0.115		
Simple	Yes	100	$\{2,5,10\}$	23.1	-1	3			0.005	0.010			0.079	0.073		
Simple	Yes	20	$\{4,10,20\}$	7.0	-1	3			0.009	0.021			0.149	0.134		
Simple	Yes	20	$\{10,25,50\}$	39.1	-1	3			0.008	-0.024			0.113	0.093		
Complex	No	20	$\{2,5,10\}$	4.0	-1	3	0.2		0.001	-0.124	0.069		0.262	0.233	0.159	
Complex	No	40	$\{2,5,10\}$	8.2	-1	3	0.2		0.031	-0.058	0.025		0.174	0.142	0.110	
Complex	No	100	$\{2,5,10\}$	21.2	-1	3	0.2		0.007	-0.023	0.012		0.114	0.092	0.064	
Complex	No	20	$\{4,10,20\}$	5.7	-1	3	0.2		-0.024	-0.076	0.053		0.211	0.158	0.150	
Complex	No	20	$\{10,25,50\}$	11.9	-1	3	0.2		-0.019	-0.074	0.034		0.163	0.123	0.172	
Complex	Yes	20	$\{2,5,10\}$	10.7	-1	3	0.2		-0.066	-0.047	0.050		0.192	0.178	0.133	
Complex	Yes	40	$\{2,5,10\}$	21.0	-1	3	0.2		-0.012	-0.016	0.016		0.139	0.116	0.096	
Complex	Yes	100	$\{2,5,10\}$	53.5	-1	3	0.2		-0.003	-0.013	0.006		0.093	0.077	0.056	
Complex	Yes	20	$\{4,10,20\}$	18.7	-1	3	0.2		-0.054	-0.039	0.046		0.180	0.133	0.139	
Complex	Yes	20	$\{10,25,50\}$	110.4	-1	3	0.2		-0.027	-0.042	0.022		0.142	0.090	0.123	
RCNL	No	20	$\{2,5,10\}$	7.5	-1	3		0.5	0.102	-0.009		-0.006	0.229	0.199		0.025
RCNL	No	40	$\{2,5,10\}$	15.2	-1	3		0.5	0.058	-0.003		-0.004	0.163	0.137		0.016
RCNL	No	100	$\{2,5,10\}$	42.5	-1	3		0.5	0.025	0.006		-0.001	0.099	0.083		0.010
RCNL	No	20	$\{4,10,20\}$	9.4	-1	3		0.5	0.089	-0.042		0.003	0.183	0.237		0.025
RCNL	No	20	$\{10,25,50\}$	14.3	-1	3		0.5	0.065	-0.075		0.008	0.144	0.309		0.033
RCNL	Yes	20	$\{2,5,10\}$	17.5	-1	3		0.5	0.009	-0.013		0.002	0.110	0.174		0.018
RCNL	Yes	40	$\{2,5,10\}$	35.7	-1	3		0.5	0.004	-0.002		0.000	0.076	0.116		0.014
RCNL	Yes	100	$\{2,5,10\}$	92.6	-1	3		0.5	-0.005	0.000		0.001	0.051	0.072		0.008
RCNL	Yes	20	$\{4,10,20\}$	25.5	-1	3		0.5	0.022	-0.074		0.006	0.121	0.209		0.023
RCNL	Yes	20	$\{10,25,50\}$	112.2	-1	3		0.5	0.029	-0.093		0.011	0.123	0.259		0.030

Table 11: Problem Scale Results

References

Amemiya, T. (1977): "The Maximum Likelihood and the Nonlinear Three-Stage Least Squares Estimator in the General Nonlinear Simultaneous Equation Model," *Econometrica*, 45, 955–968.

Armstrong, T. (2016): "Large Market Asymptotics for Differentiated Product Demand Estimators with Economic Models of Supply," *Econometrica*, 84, 1961–1980.

Backus, M., C. Conlon, and M. Sinkinson (2018): "Common Ownership and Competition in the Ready-To-Eat Cereal Industry," Working Paper.

Bayer, P., F. Ferreira, and R. McMillan (2007): "A Unified Framework for Measuring Preferences for Schools and Neighborhoods," *Journal of Political Economy*, 115, 588–638.

Berry, S. (1994): "Estimating discrete-choice models of product differentiation," *RAND Journal of Economics*, 25, 242–261.

Berry, S., J. Levinsohn, and A. Pakes (1995): "Automobile Prices in Market Equilibrium," *Econometrica*, 63, 841–890.

——— (1999): "Voluntary Export Restraints on Automobiles: Evaluating a Trade Policy," *American Economic Review*, 89, 400–430.

Berry, S., O. B. Linton, and A. Pakes (2004): "Limit Theorems for Estimating the Parameters of Differentiated Product Demand Systems," *Review of Economic Studies*, 71, 613–654.

Berry, S. T. and P. A. Haile (2014): "Identification in Differentiated Products Markets Using Market Level Data," *Econometrica*, 82, 1749–1797.

Brenkers, R. and F. Verboven (2006): "LIBERALIZING A DISTRIBUTION SYSTEM: THE EUROPEAN CAR MARKET," *Journal of the European Economic Association*, 4, 216–251.

Caplin, A. and B. Nalebuff (1991): "Aggregation and Imperfect Competition: On the Existence of Equilibrium," *Econometrica*, 59, 25–59.

Chamberlain, G. (1987): "Asymptotic Efficiency in Estimation with Conditional Moment Restrictions," *Journal of Econometrics*, 34, 305–334.

Conlon, C. (2017): "The MPEC Approach to Empirical Likelihood Estimation of Demand," Working Paper.

Conlon, C. and N. Rao (2017): "The Price of Liquor is Too Damn High: The Effects of Post and Hold Pricing," Working Paper.

Correia, S. (2016): "Linear Models with High-Dimensional Fixed Effects: An Efficient and Feasible Estimator," Tech. rep., working Paper.

Deaton, A. S. and J. Muellbauer (1980): "An Almost Ideal Demand System," *American Economic Review*, 70, 312–26.

Dubé, J.-P. H., J. T. Fox, and C.-L. Su (2012): "Improving the Numerical Performance of BLP Static and Dynamic Discrete Choice Random Coefficients Demand Estimation," *Econometrica*, 80, 2231–2267.

Freyberger, J. (2015): "Asymptotic theory for differentiated products demand models with many markets," *Journal of Econometrics*, 185, 162 – 181.

Gallego, G., W. T. Huh, W. Kang, and R. Phillips (2006): "Price Competition with the Attraction Demand Model: Existence of Unique Equilibrium and Its Stability," *Manufacturing & Service Operations Management*, 8, 359–375.

Gandhi, A. and J. Houde (2017): "Measuring Substitution Patterns in Differentiated Products Industries," Working Paper.

Grigolon, L. and F. Verboven (2014): "Nested Logit or Random Coefficients Logit? A Comparison of Alternative Discrete Choice Models of Product Differentiation," *The Review of Economics and Statistics*, 96, 916–935.

Heiss, F. and V. Winschel (2008): "Likelihood approximation by numerical integration on sparse grids," *Journal of Econometrics*, 144, 62 – 80.

Ho, K. and A. Pakes (2014): "Hospital Choices, Hospital Prices, and Financial Incentives to Physicians," *American Economic Review*, 104, 3841–84.

Houde, J.-F. (2012): "Spatial Differentiation and Vertical Mergers in Retail Markets for Gasoline," *American Economic Review*, 102, 2147–82.

Judd, K. L. and B. Skrainka (2011): "High performance quadrature rules: how numerical integration affects a popular model of product differentiation," CeMMAP working papers CWP03/11, Centre for Microdata Methods and Practice, Institute for Fiscal Studies.

Knittel, C. R. and K. Metaxoglou (2014): "Estimation of Random-Coefficient Demand Models: Two Empiricists' Perspective," *Review of Economics and Statistics*, 96.

Koijen, R. S. and M. Yogo (2018): "A Demand System Approach to Asset Pricing," *Journal of Political Economy*, 0, null.

Konovalov, A. and Z. Sandor (2010): "On price equilibrium with multi-product firms," *Economic Theory*, 44, 271–292.

Lee, J. and K. Seo (2016): "Revisiting the nested fixed-point algorithm in BLP random coefficients demand estimation," *Economics Letters*, 149.

Lee, R. S. (2013): "Vertical Integration and Exclusivity in Platform and Two-Sided Markets," *American Economic Review*, 103, 2960–3000.

MacKay, A. and N. Miller (2018): "Demand Estimation in Models of Imperfect Competition," Working Paper.

Miller, N. and M. Weinberg (2017): "Understanding the Price Effects of the MillerCoors Joint Venture," *Econometrica*, 85, 1763–1791.

Miravete, E. J., K. Seim, and J. Thurk (2018): "Market Power and the Laffer Curve," *Econometrica*, 86, 1651–1687.

Morrow, W. R. and S. J. Skerlos (2010): "On the Existence of Bertrand-Nash Equilibrium Prices Under Logit Demand," *CoRR*, abs/1012.5832.

——— (2011): "Fixed-Point Approaches to Computing Bertrand-Nash Equilibrium Prices Under Mixed-Logit Demand," *Operations Research*, 59, 328–345.

Nevo, A. (2000a): "Mergers with differentiated products: the case of the ready-to-eat cereal industry," *RAND Journal of Economics*, 31, 395–421.

——— (2000b): "A Practitioner's Guide to Estimation of Random Coefficients Logit Models of Demand (including Appendix)," *Journal of Economics and Management Strategy*, 9, 513–548.

——— (2001): "Measuring Market Power in the Ready-to-Eat Cereal Industry," *Econometrica*, 69, 307–342.

Newey, W. K. and R. J. Smith (2004): "Higher Order Properties of GMM and Generalized Empirical Likelihood Estimators," *Econometrica*, 72, 219–255.

Petrin, A. (2002): "Quantifying the Benefits of New Products: The Case of the Minivan," *Journal of Political Economy*, 110, 705–729.

Reynaert, M. and F. Verboven (2014): "Improving the performance of random coefficients demand models: The role of optimal instruments," *Journal of Econometrics*, 179, 83–98.

Reynaerts, J., R. Varadhan, and J. C. Nash (2012): "Enhancing the Convergence Properties of the BLP (1995) Contraction Mapping," Vives discussion paper series 35, Katholieke Universiteit Leuven, Faculteit Economie en Bedrijfswetenschappen, Vives, http://ideas.repec.org/p/ete/vivwps/35.html.

Rios-Avila, F. (2015): "Feasible fitting of linear models with N fixed effects," *The Stata Journal*, 15, 881–898.

Skrainka, B. (2012a): "Three Essays on Product Differentiation," PhD dissertation, University College London.

Skrainka, B. S. (2012b): "A Large Scale Study of the Small Sample Performance of Random Coefficient Models of Demand," Working Paper.

Somaini, P. and F. Wolak (2016): "An Algorithm to Estimate the Two-Way Fixed Effects Model," *Journal of Econometric Methods*, 5, 143–152.

Varadhan, R. and C. Roland (2008): "Simple and Globally Convergent Methods for Accelerating the Convergence of Any EM Algorithm," *Scandinavian Journal of Statistics*, 35, 335–353.

Appendices

A Appendix

A.1 Concentrating out Linear Parameters

Our objective is to concentrate out $[\widehat{\beta}(\theta_2), \widehat{\gamma}(\theta_2)]$. Define $y^D_{jt}, y^S_{jt}, x^D_{jt}, x^S_{jt}$ as follows:

$$\begin{aligned} y^D_{jt} &:= \widehat{\delta}_{jt}(\theta_2) + \alpha p_{jt} = \quad x'_{jt}\beta + \xi_t =: x^{D\prime}_{jt}\beta + \xi_{jt} \\ y^S_{jt} &:= \widehat{mc}_{jt}(\theta_2) = \quad (x_{jt}\ w_{jt})'\gamma + \omega_t =: x^{S\prime}_{jt}\gamma + \omega_{jt} \end{aligned} \tag{1}$$

Stacking the system across observations yields:[46]

$$\underbrace{\begin{bmatrix} y_D \\ y_S \end{bmatrix}}_{2N\times 1} = \underbrace{\begin{bmatrix} X_D & 0 \\ 0 & X_S \end{bmatrix}}_{2N\times(K_1+K_3)} \underbrace{\begin{bmatrix} \beta \\ \gamma \end{bmatrix}}_{(K_1+K_3)\times 1} + \underbrace{\begin{bmatrix} \xi \\ \omega \end{bmatrix}}_{2N\times 1} \tag{2}$$

Adding the instruments $z^D : (N \times q^D)$ and $z^S : (N \times q^S)$ yields $q = q^D + q^S$ moment restrictions:

$$g(\beta, \gamma) = E_n \begin{bmatrix} z^{D\prime}_n (y^D_n - x^{D\prime}_n \beta) \\ z^{S\prime}_n (y^S_n - x^{S\prime}_n \gamma) \end{bmatrix} = 0 \tag{3}$$

$$\underbrace{g_n}_{q\times 1} = \underbrace{\frac{1}{N}\underbrace{\begin{bmatrix} Z'_D & 0 \\ 0 & Z'_S \end{bmatrix}}_{q\times 2N}\underbrace{\begin{bmatrix} y_D \\ y_S \end{bmatrix}}_{2N\times 1}}_{=:\tilde{Y}\quad (q\times 1)} - \underbrace{\frac{1}{N}\underbrace{\begin{bmatrix} Z'_D X_D & 0 \\ 0 & Z'_S X_D \end{bmatrix}}_{q\times(K_1+K_3)}}_{=:\tilde{X}} \underbrace{\begin{bmatrix} \beta \\ \gamma \end{bmatrix}}_{(K_1+K_3)\times 1} \tag{4}$$

Now we can simply perform a GMM regression of $\tilde{Y}$ on $\tilde{X}$ where W is the $(q \times q)$ GMM weighting matrix:[47]

$$\begin{bmatrix} \widehat{\beta}(\theta_2) \\ \widehat{\gamma}(\theta_2) \end{bmatrix} = (\tilde{X}'W\tilde{X})^{-1}\tilde{X}'W\tilde{Y} \tag{5}$$

A.2 Analytic Derivative Calculations

This derivation appears to be novel to the literature for the case of simultaneous estimation of supply and demand.

[46] Note: we cannot perform independent regressions unless we are willing to assume that $Cov(\xi_{jt}, \omega_{jt}) = 0$.

[47] Observe this is the same GMM weighting matrix as for the overall problem. Also note that we require that $q > K_1 + K_3$ so that we have overidentifying restrictions, we need at least $K_2 = \dim(\theta_2)$ such restrictions.

The gradient of the GMM objective function $Q(\theta_2)$ is given by:

$$2 \cdot G_n(\theta_2)' \cdot W \cdot g_n(\theta_2)$$

The challenging piece here is the Jacobian of the GMM objective $G_n(\theta_2) = \frac{1}{N} \underbrace{\begin{bmatrix} Z_D' & 0 \\ 0 & Z_S' \end{bmatrix}}_{q \times 2N} \underbrace{\begin{bmatrix} \frac{\partial \xi}{\partial \theta_2} \\ \frac{\partial \omega}{\partial \theta_2} \end{bmatrix}}_{2N \times K_2}$ which is a $(q \times K_2)$ matrix. Because (9) are linear we can write:

$$\begin{bmatrix} \frac{\partial \xi}{\partial \theta_2} \\ \frac{\partial \omega}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \delta}{\partial \theta_2} \\ -f'(\cdot) \frac{\partial \eta}{\partial \theta_2} \end{bmatrix} \tag{6}$$

For the demand moments, after invoking the implicit function theorem, this has a convenient block structure which can be separated market by market t:[48]

$$\underbrace{\frac{\partial \delta_{\cdot t}}{\partial \theta_2}(\theta_2)}_{J_t \times K_2} = \underbrace{\left[\frac{\partial s_{\cdot t}}{\partial \delta_{\cdot t}}(\theta_2)\right]^{-1}}_{J_t \times J_t} \times \underbrace{\left[\frac{\partial s_{\cdot t}}{\partial \theta_2}(\theta_2)\right]}_{J_t \times K_2}$$

For the supply moments, the $f'(\cdot)$ in the (6) comes from (6) where $f(\cdot)$ is typically linear $f'(\cdot) = 1$ or logarithmic $f'(\cdot) = \frac{1}{mc}$. Differentiating the supply moments is challenging because the matrix of demand derivatives $\Omega(\mathbf{p}, \xi(\theta_2), \theta_2)$ depends on both $\xi(\theta_2)$ and θ_2 directly. It can be helpful to write $A(\xi(\theta_2), \theta_2) = O \odot \Omega$, and $\eta = A^{-1} \cdot \mathbf{s}$. In order to avoid tensor product notation, let's consider taking the derivative with respect to an element within θ_2 which we call θ_l:[49]

$$\begin{aligned} \underbrace{\frac{\partial \eta_{\cdot t}}{\partial \theta_l}}_{J_t \times 1} &= -A_t^{-1} \frac{\partial A_t}{\partial \theta_l} A_t^{-1} \mathbf{s_{\cdot t}} + A_t^{-1} \frac{\partial A_t}{\partial \xi} \frac{\partial \xi_{\cdot t}}{\partial \theta_l} A_t^{-1} \mathbf{s_{\cdot t}} + A_t^{-1} \underbrace{\frac{\partial \mathbf{s_{\cdot t}}}{\partial \theta_l}}_{=0 \text{ (data)}} \\ &= -\underbrace{A_t^{-1}}_{(J_t \times J_t)} \underbrace{\frac{\partial A_t}{\partial \theta_l}}_{(J_t \times J_t)} \underbrace{\eta_{\cdot t}}_{(J_t \times 1)} + \underbrace{A_t^{-1}}_{(J_t \times J_t)} \underbrace{\frac{\partial A_t}{\partial \xi_{\cdot t}}}_{(J_t \times J_t \times J_t)} \underbrace{\frac{\partial \xi_{\cdot t}}{\partial \theta_l}}_{(J_t \times 1)} \underbrace{\eta_{\cdot t}}_{(J_t \times 1)} \end{aligned}$$

Because the supply error $\omega_{\cdot t}$ and the markup $\eta_{\cdot t}$ depend both directly on θ_2 and indirectly on θ_2 through $\xi_{\cdot t}$ this expression is more complicated.

[48] $\frac{\partial \xi}{\partial \delta} = I_N$ so that $\frac{\partial \xi}{\partial \theta_2} = \frac{\partial \delta}{\partial \theta_2}$ The matrix inverse $\left[\frac{\partial s_{\cdot t}}{\partial \delta_{\cdot t}}(\theta_2)\right]^{-1}$ is guaranteed by the diagonal dominance of system of equations with respect to δ_{jt}. As long as the outside good has positive share, we have that for each j: $|\frac{\partial s_{jt}}{\partial \delta_{jt}}| > \sum_{k \neq j} |\frac{\partial s_{kt}}{\partial \delta_{jt}}|$. A square matrix and its transpose have the same eigenvalues and thus are both are non-singular. In practice, as shares become small, there may still be numerical issues.

[49] In the markup η, $\mathbf{s_{\cdot t}}$ is data and thus does not depend on parameters.

[Still have a Tensor Product]

A.3 Levenberg-Marquardt Algorithm for $\delta_{\cdot t}$.

The Levenberg-Marquardt (LM) algorithm minimizes the following least-squares problem in order to solve the $J_t \times J_t$ system of nonlinear equations:

$$\min_{\delta} \sum_{j=1}^{J} (\mathcal{S}_j - s_j(\delta_{\cdot t}, \theta_2)^2$$

The idea is to update our guess of δ to to $\delta + x$ where x is a $J \times 1$ vector. The Levenberg-Marquardt update is given by the solution to the linear system of equations:

$$\underbrace{[J_{\mathbf{s}}' J_{\mathbf{s}} + \lambda \operatorname{diag}(J_{\mathbf{s}}' J_{\mathbf{s}})]^{-1}}_{\approx \mathbf{H^{-1}}} J_{\mathbf{s}}' x = J_{\mathbf{s}}'(\mathcal{S}_j - s_j(\delta_{\cdot t}, \theta_2)$$

This has the advantage that for $\lambda = 0$ the algorithm takes a full Gauss-Newton step, while for λ large it takes a step in the direction of the gradient (gradient-descent). The additional diagonal term also guarantees the invertibility of approximate Hessian.

B Additional Tables and Figures

								True Value				Median Bias				Median Absolute Error			
Simulation	Supply	P	Software	Algorithm	Bounds	Gradient	Seconds	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ	α	Σ_{00}	Σ_{11}	ρ
Simple	No	1	Knitro	Interior/Direct	Yes	Yes	1.5	-1	3			0.157	-0.028			0.246	0.181		
Simple	No	1	Knitro	Active Set	Yes	Yes	1.4	-1	3			0.157	-0.024			0.246	0.181		
Simple	No	1	Knitro	SQP	Yes	Yes	1.5	-1	3			0.157	-0.024			0.246	0.181		
Simple	No	1	SciPy	SLSQP	Yes	Yes	1.4	-1	3			0.157	-0.024			0.246	0.180		
Simple	No	1	SciPy	TNC	Yes	Yes	1.9	-1	3			0.157	-0.028			0.246	0.181		
Simple	No	1	SciPy	L-BFGS-B	Yes	Yes	1.4	-1	3			0.157	-0.026			0.246	0.181		
Simple	No	1	SciPy	BFGS	No	Yes	1.4	-1	3			0.157	-0.028			0.246	0.181		
Simple	No	1	SciPy	Nelder-Mead	No	No	4.5	-1	3			0.157	-0.028			0.246	0.181		
Simple	Yes	2	Knitro	Interior/Direct	Yes	Yes	4.6	-1	3			0.018	0.006			0.161	0.185		
Simple	Yes	2	Knitro	Active Set	Yes	Yes	4.4	-1	3			0.019	0.007			0.161	0.186		
Simple	Yes	2	Knitro	SQP	Yes	Yes	4.5	-1	3			0.018	0.007			0.162	0.185		
Simple	Yes	2	SciPy	SLSQP	Yes	Yes	4.1	-1	3			0.019	0.007			0.161	0.186		
Simple	Yes	2	SciPy	TNC	Yes	Yes	6.6	-1	3			0.018	0.006			0.160	0.185		
Simple	Yes	2	SciPy	L-BFGS-B	Yes	Yes	3.9	-1	3			0.018	0.007			0.161	0.186		
Simple	Yes	2	SciPy	BFGS	No	Yes	4.0	-1	3			0.018	0.006			0.160	0.185		
Simple	Yes	2	SciPy	Nelder-Mead	No	No	14.6	-1	3			0.018	0.006			0.160	0.185		
Complex	No	2	Knitro	Interior/Direct	Yes	Yes	3.9	-1	3	0.2		0.075	-0.079	0.018		0.238	0.202	0.156	
Complex	No	2	Knitro	Active Set	Yes	Yes	3.6	-1	3	0.2		0.053	-0.095	-0.083		0.283	0.246	0.200	
Complex	No	2	Knitro	SQP	Yes	Yes	4.0	-1	3	0.2		-0.033	-0.174	0.096		0.276	0.280	0.200	
Complex	No	2	SciPy	SLSQP	Yes	Yes	3.9	-1	3	0.2		-0.002	-0.131	0.073		0.261	0.236	0.164	
Complex	No	2	SciPy	TNC	Yes	Yes	7.9	-1	3	0.2		0.013	-0.109	0.058		0.259	0.231	0.165	
Complex	No	2	SciPy	L-BFGS-B	Yes	Yes	4.1	-1	3	0.2		-0.069	-0.167	0.114		0.287	0.264	0.197	
Complex	No	2	SciPy	BFGS	No	Yes	4.2	-1	3	0.2		0.075	-0.080	-0.146		0.236	0.201	0.200	
Complex	No	2	SciPy	Nelder-Mead	No	No	15.3	-1	3	0.2		0.077	-0.079	0.017		0.238	0.201	0.158	
Complex	Yes	3	Knitro	Interior/Direct	Yes	Yes	9.8	-1	3	0.2		-0.039	-0.051	0.030		0.181	0.190	0.126	
Complex	Yes	3	Knitro	Active Set	Yes	Yes	8.6	-1	3	0.2		-0.012	-0.032	-0.071		0.188	0.194	0.200	
Complex	Yes	3	Knitro	SQP	Yes	Yes	10.4	-1	3	0.2		-0.046	-0.062	0.044		0.186	0.198	0.137	
Complex	Yes	3	SciPy	SLSQP	Yes	Yes	9.2	-1	3	0.2		-0.047	-0.059	0.044		0.189	0.194	0.131	
Complex	Yes	3	SciPy	TNC	Yes	Yes	17.8	-1	3	0.2		-0.044	-0.059	0.040		0.184	0.197	0.131	
Complex	Yes	3	SciPy	L-BFGS-B	Yes	Yes	9.6	-1	3	0.2		-0.050	-0.061	0.044		0.185	0.197	0.130	
Complex	Yes	3	SciPy	BFGS	No	Yes	9.8	-1	3	0.2		-0.037	-0.051	-0.088		0.181	0.190	0.200	
Complex	Yes	3	SciPy	Nelder-Mead	No	No	37.7	-1	3	0.2		-0.035	-0.048	0.021		0.182	0.192	0.133	
RCNL	No	2	Knitro	Interior/Direct	Yes	Yes	7.6	-1	3		0.5	0.105	-0.005		-0.005	0.222	0.193		0.024
RCNL	No	2	Knitro	Active Set	Yes	Yes	7.4	-1	3		0.5	0.116	-0.019		-0.008	0.226	0.214		0.025
RCNL	No	2	Knitro	SQP	Yes	Yes	7.8	-1	3		0.5	0.111	-0.013		-0.006	0.226	0.203		0.025
RCNL	No	2	SciPy	SLSQP	Yes	Yes	7.0	-1	3		0.5	0.107	-0.005		-0.007	0.226	0.202		0.025
RCNL	No	2	SciPy	TNC	Yes	Yes	12.5	-1	3		0.5	0.102	-0.002		-0.006	0.229	0.208		0.026
RCNL	No	2	SciPy	L-BFGS-B	Yes	Yes	7.9	-1	3		0.5	0.102	-0.006		-0.006	0.229	0.203		0.025
RCNL	No	2	SciPy	BFGS	No	Yes	6.8	-1	3		0.5	0.108	-0.010		-0.005	0.225	0.191		0.024
RCNL	No	2	SciPy	Nelder-Mead	No	No	28.9	-1	3		0.5	0.113	-0.011		-0.005	0.222	0.192		0.024
RCNL	Yes	3	Knitro	Interior/Direct	Yes	Yes	15.7	-1	3		0.5	0.010	-0.016		0.000	0.111	0.180		0.021
RCNL	Yes	3	Knitro	Active Set	Yes	Yes	16.4	-1	3		0.5	0.014	-0.021		0.000	0.112	0.188		0.022
RCNL	Yes	3	Knitro	SQP	Yes	Yes	16.7	-1	3		0.5	0.011	-0.021		0.001	0.112	0.188		0.022
RCNL	Yes	3	SciPy	SLSQP	Yes	Yes	14.8	-1	3		0.5	0.011	-0.019		0.001	0.110	0.182		0.022
RCNL	Yes	3	SciPy	TNC	Yes	Yes	25.6	-1	3		0.5	0.011	-0.022		0.001	0.112	0.193		0.022
RCNL	Yes	3	SciPy	L-BFGS-B	Yes	Yes	15.6	-1	3		0.5	0.012	-0.019		0.000	0.109	0.183		0.022
RCNL	Yes	3	SciPy	BFGS	No	Yes	13.7	-1	3		0.5	0.011	-0.018		0.000	0.109	0.180		0.021
RCNL	Yes	3	SciPy	Nelder-Mead	No	No	62.1	-1	3		0.5	0.010	-0.019		0.001	0.109	0.180		0.021

Table A-1: Optimization Algorithms Parameter Estimates