SUPPLEMENT TO "IMPROVING THE NUMERICAL PERFORMANCE OF STATIC AND DYNAMIC AGGREGATE DISCRETE CHOICE RANDOM COEFFICIENTS DEMAND ESTIMATION"

(*Econometrica*, Vol. 80, No. 5, September 2012, 2231–2267)

BY JEAN-PIERRE DUBÉ, JEREMY T. FOX, AND CHE-LIN SU

S1. IMPLEMENTATION DETAILS FOR MPEC AND NFP

IN THIS SECTION, we discuss the implementation details for MPEC and NFP applied to the BLP demand estimation problem. For a gradient-based solver to perform effectively, researchers need to provide the first-order and second-order closed-form derivatives along with their respective sparsity patterns. Without this additional information, both MPEC and NFP may fail to converge or may require considerably more iterations (and hence more computation time) to find a solution to the optimization problem. We provide derivatives and sparsity patterns in our publicly available MATLAB code. Below, we provide formulas for the derivatives and sparsity patterns. We also use specific numerical examples to explore the implications of derivatives and sparsity patterns for the performance of the optimization algorithms.

S1.1. *Supplying First- and Second-Order Derivatives and Their Sparsity Patterns*

In principle, the KNITRO solver, the one we use, can perform MPEC or NFP with only the first-order derivatives. Supplying the second-order closed-form derivatives increases the accuracy of the search direction and decreases the number of iterations needed to find a solution. Therefore, for large-dimensional problems or for problems with a large number of simulation draws to evaluate the share equations, supplying the second-order closed-form derivatives and the Hessian to the solver leads to substantial speed improvements. Supplying second-order closed-form derivatives can also lead to large speed improvements for NFP. We have seen examples where providing the second-order closed-form derivatives to KNITRO decreases NFP's computational time by 80% or more. Consequently, second-order derivatives can reduce the computational time for NFP applied to large-dimensional demand estimation problems from a matter of days to a matter of hours; see the example with 250 markets, 25 products, and 3000 draws reported in Table VI in the paper.

A naive implementation of MPEC could lead some researchers to conclude that it cannot handle large-dimensional problems with many variables and many constraints. However, the BLP demand estimation problem is typically very sparse. Most state-of-the-art solvers (e.g., KNITRO) are able to exploit the sparsity structure of an optimization problem, reducing the amount of

 DOI: 10.3982/ECTA8585

memory required. Thus, supplying the sparsity pattern of the constraint Jacobian and the Hessian decreases the memory required and enables the solver to manage problems with a large number of variables and constraints. We have successfully implemented MPEC for problems with 12,552 variables and 12,542 constraints, and observed it to perform 26 times faster than NFP.

Some researchers may incorrectly conclude that supplying the sparsity pattern of the Hessian is not important for the NFP algorithm because its outer loop involves a small number of parameters. However, as we show below, computing the gradient and the Hessian for NFP requires the inverse of the matrix $[\frac{\partial s(\xi;\theta)}{\partial \xi}]$. When the numbers of markets and products are large, the inverse matrix $[\frac{\partial s(\xi;\theta)}{\partial \xi}]^{-1}$ term needed to evaluate the gradient and the Hessian can become computationally prohibitive unless the sparsity structure of the matrix $[\frac{\partial s(\xi;\theta)}{\partial \xi}]$ is provided.

S1.2. *Formulae for the Gradients and Hessians of NFP and MPEC*

We derive the gradients and Hessians of NFP and MPEC, respectively, applied to the normal random coefficients logit demand system used in BLP (1995) and many other empirical papers. To simplify the notation, we denote the market share equation for product j in market t as

$$\begin{aligned} s_j(\xi_t;\theta) &= \int \exp\Bigg(x_{j,t}'\overline{\beta} - \bar{\alpha}p_{j,t} + \xi_{j,t} + \sum_k x_{j,t,k}\nu_k\sigma_{\beta_k} - p_{j,t}\nu_{K+1}\sigma_\alpha\Bigg) \\ &\quad \Bigg/\Bigg(1 + \sum_{i=1}^{J}\exp\Bigg(x_{i,t}'\overline{\beta} - \bar{\alpha}p_{i,t} + \xi_{i,t} \\ &\quad + \sum_k x_{i,t,k}\nu_k\sigma_{\beta_k} - p_{i,t}\nu_{K+1}\sigma_\alpha\Bigg)\Bigg) dF(\nu) \\ &= \int \mathcal{T}_j(\xi_t,\nu;\theta)\,dF(\nu), \end{aligned}$$

where $\theta = (\overline{\beta}, \overline{\alpha}, \sigma_\beta, \sigma_\alpha)'$, and $\nu \sim N(0, I_{K+1})$.

S1.2.1. *Derivatives of MPEC*

The MPEC formulation is

$$\begin{aligned} &\min_{(\theta,\xi,g)} g'Wg \\ &\quad \text{subject to} \quad s(\xi;\theta) = S, \\ &\qquad\qquad\quad\ \ g = Z'\xi. \end{aligned}$$

The Lagrangian for MPEC is

$$\mathcal{L}(\theta, \xi, g, \lambda_s \lambda_g) = g'Wg + \lambda_s'\big(s(\xi; \theta) - S\big) + \lambda_g'(g - Z'\xi).$$

The gradient of the MPEC objective function is

$$\nabla_{(\theta,\xi,g)} g'Wg = \begin{bmatrix} 0 \\ 0 \\ 2Wg \end{bmatrix}.$$

The constraint Jacobian for MPEC is

$$\begin{bmatrix} \dfrac{\partial s}{\partial \theta} & \dfrac{\partial s}{\partial \xi} & 0 \\ 0 & -Z' & I_g \end{bmatrix},$$

where the components of the constraint Jacobian are

$$\frac{\partial s_j(\xi_t; \theta)}{\partial \overline{\beta}_k} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\left(x_{j,t,k} - \sum_i \mathcal{T}_i(\xi_t, \nu; \theta) x_{i,t,k}\right) dF(\nu),$$

$$\frac{\partial s_j(\xi_t; \theta)}{\partial \overline{\alpha}} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\left(p_{j,t} - \sum_i \mathcal{T}_i(\xi_t, \nu; \theta) p_{i,t}\right) dF(\nu),$$

$$\frac{\partial s_j(\xi_t; \theta)}{\partial \sigma_{\beta_k}} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\left(x_{j,t,k} - \sum_i \mathcal{T}_i(\xi_t, \nu; \theta) x_{i,t,k}\right) \nu_k \, dF(\nu),$$

$$\frac{\partial s_j(\xi_t; \theta)}{\partial \sigma_{\alpha}} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\left(p_{j,t} - \sum_i \mathcal{T}_i(\xi_t, \nu; \theta) p_{i,t}\right) \nu_{K+1} \, dF(\nu),$$

$$\frac{\partial s_j(\xi_t; \theta)}{\partial \xi_{j,t}} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\big(1 - \mathcal{T}_j(\xi_t, \nu; \theta)\big) \, dF(\nu),$$

$$\frac{\partial s_j(\xi_t; \theta)}{\partial \xi_{i,t}} = -\int \mathcal{T}_j(\xi_t, \nu; \theta) \mathcal{T}_i(\xi_t, \nu; \theta) \, dF(\nu),$$

and

I_g is an identity matrix.

The Hessian of the Lagrangian is

$$\nabla^2 \mathcal{L}(\theta, \xi, g, \lambda_s \lambda_g) = \nabla^2\big(g'Wg\big) + \sum_{j=1}^{J \times T} \lambda_{sj} \nabla^2 s_j(\xi, \theta)$$

$$= \begin{bmatrix} \dfrac{\partial^2 \mathcal{L}}{\partial \theta^2} & \dfrac{\partial^2 \mathcal{L}}{\partial \theta\, \partial \xi} & 0 \\ \dfrac{\partial^2 \mathcal{L}}{\partial \xi\, \partial \theta} & \dfrac{\partial^2 \mathcal{L}}{\partial \xi^2} & 0 \\ 0 & 0 & \dfrac{\partial^2 \mathcal{L}}{\partial g^2} \end{bmatrix},$$

where

$$\frac{\partial^2 \mathcal{L}}{\partial \theta^2} = \sum_{j=1}^{J*T} \lambda_{sj} \frac{\partial^2 s_j(\xi, \theta)}{\partial \theta^2},$$

$$\frac{\partial^2 \mathcal{L}}{\partial \theta\, \partial \xi} = \sum_{j=1}^{J*T} \lambda_{sj} \frac{\partial^2 s_j(\xi, \theta)}{\partial \theta\, \partial \xi},$$

$$\frac{\partial^2 \mathcal{L}}{\partial \xi^2} = \sum_{j=1}^{J*T} \lambda_{sj} \frac{\partial^2 s_j(\xi, \theta)}{\partial \xi^2},$$

$$\frac{\partial^2 \mathcal{L}}{\partial g^2} = 2W,$$

and

$$\frac{\partial^2 s_j(\xi_t; \theta)}{\partial \xi_{jt}^2} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\big(1 - \mathcal{T}_j(\xi_t, \nu; \theta)\big) \times \big(1 - 2\mathcal{T}_j(\xi_t, \nu; \theta)\big)\, dF(\nu),$$

$$\frac{\partial^2 s_j(\xi_t; \theta)}{\partial \xi_{jt}\, \partial \xi_{it}} = -\int \mathcal{T}_j(\xi_t, \nu; \theta)\mathcal{T}_i(\xi_t, \nu; \theta)\big(1 - 2\mathcal{T}_j(\xi_t, \nu; \theta)\big)\, dF(\nu),$$

$$\frac{\partial^2 s_i(\xi_t; \theta)}{\partial \xi_{jt}\, \partial \xi_{it}} = -\int \mathcal{T}_j(\xi_t, \nu; \theta)\mathcal{T}_i(\xi_t, \nu; \theta)\big(1 - 2\mathcal{T}_i(\xi_t, \nu; \theta)\big)\, dF(\nu),$$

$$\frac{\partial^2 s_j(\xi_t; \theta)}{\partial \xi_{it}^2} = -\int \mathcal{T}_j(\xi_t, \nu; \theta)\mathcal{T}_i(\xi_t, \nu; \theta)\big(1 - 2\mathcal{T}_i(\xi_t, \nu; \theta)\big)\, dF(\nu),$$

$$\frac{\partial^2 s_j(\xi_t; \theta)}{\partial \xi_{it}\, \partial \xi_{kt}} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\mathcal{T}_i(\xi_t, \nu; \theta)\big(2\mathcal{T}_k(\xi_t, \nu; \theta)\big)\, dF(\nu),$$

$$\frac{\partial^2 s_j(\xi_t; \theta)}{\partial \sigma_{\beta_k}^2} = \int \mathcal{T}_j(\xi_t, \nu; \theta)\left(x_{j,t,k} - \sum_{i=1}^{J} \mathcal{T}_i(\xi_t, \nu; \theta) x_{i,t,k}\right)^2 \nu_k^2\, dF(\nu)$$

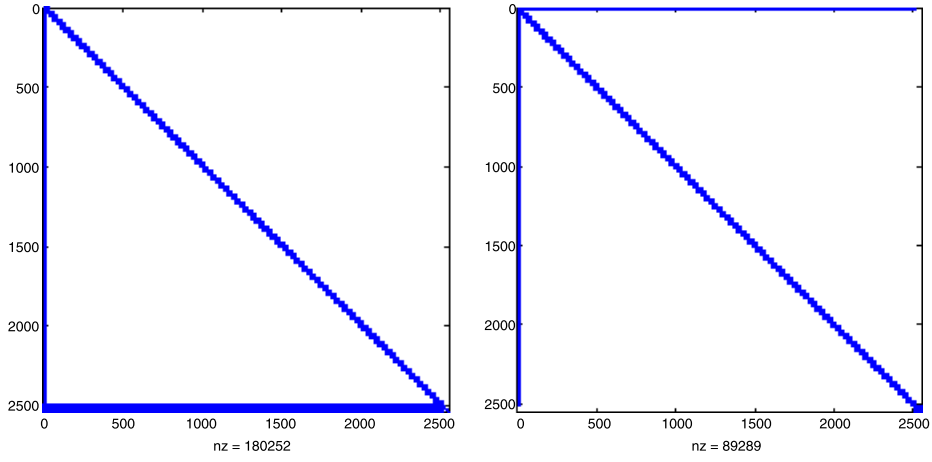
$$+\int \mathcal{T}_j(\xi_t,\nu;\theta)\Bigg[-\sum_{i=1}^{J}\Bigg\{\mathcal{T}_i(\xi_t,\nu;\theta)$$
$$\times\Bigg(x_{i,t,k}-\sum_{l=1}^{J}\mathcal{T}_l(\xi_t,\nu;\theta)x_{l,t,k}\Bigg)\nu_k\Bigg\}x_{i,t,k}\Bigg]\nu_k\,dF(\nu),$$

$$\frac{\partial^2 s_j(\xi_t;\theta)}{\partial\sigma_{\beta_k}\,\partial\sigma_{\beta_m}}=\int \mathcal{T}_j(\xi_t,\nu;\theta)\Bigg(x_{j,t,m}-\sum_{i=1}^{J}\mathcal{T}_i(\xi_t,\nu;\theta)x_{i,t,m}\Bigg)$$
$$\times\nu_m\Bigg(x_{j,t,k}-\sum_{i=1}^{J}\mathcal{T}_i(\xi_t,\nu;\theta)x_{i,t,k}\Bigg)\nu_k\,dF(\nu)$$
$$+\int \mathcal{T}_j(\xi_t,\nu;\theta)\Bigg[-\sum_{i=1}^{J}\Bigg\{\mathcal{T}_i(\xi_t,\nu;\theta)$$
$$\times\Bigg(x_{i,t,m}-\sum_{l=1}^{J}\mathcal{T}_l(\xi_t,\nu;\theta)x_{l,t,m}\Bigg)\nu_m\Bigg\}x_{i,t,k}\Bigg]\nu_k\,dF(\nu),$$

$$\frac{\partial^2 s_j(\xi_t;\theta)}{\partial\sigma_{\beta_k}\,\partial\xi_{jt}}=\int \mathcal{T}_j(\xi_t,\nu;\theta)\big(1-2\mathcal{T}_j(\xi_t,\nu;\theta)\big)$$
$$\times\Bigg(x_{j,t,k}-\sum_{i=1}^{J}\mathcal{T}_i(\xi_t,\nu;\theta)x_{i,t,k}\Bigg)\nu_k\,dF(\nu),$$

$$\frac{\partial^2 s_j(\xi_t;\theta)}{\partial\sigma_{\beta_k}\,\partial\xi_{it}}=\int \mathcal{T}_j(\xi_t,\nu;\theta)$$
$$\times\Bigg(x_{j,t,k}+x_{i,t,k}-2\sum_{l=1}^{J}\mathcal{T}_l(\xi_t,\nu;\theta)x_{l,t,k}\Bigg)\nu_k\,dF(\nu).$$

We now discuss the sparsity structure of the Jacobian and Hessian in MPEC. In BLP demand estimation problems, markets are assumed not to exhibit any unobserved interdependence. Consequently, the unobserved demand shocks in one market do not appear in the share equations of other markets. This feature leads to a sparse Jacobian and Hessian in the MPEC formulation. Let $n(y)$ denote the number of elements in the vector y. Recall that T and J are the number of markets and products, respectively, and hence $n(\xi)=T\times J$. The size of the Jacobian matrix is $(n(\xi)+n(g))\times(n(\theta)+n(\xi)+n(g))$. The number of nonzero elements in the Jacobian is around $(n(\theta)+n(g))\times n(\xi)+T\times J^2$. Since $n(\xi)$ is typically much larger than $n(\theta)+n(g)$, the density of the Jacobian matrix (defined as the ratio of the number of nonzero elements and the number of total elements) is in the order of $\frac{1}{T}$. For the Hessian, the



(a) Sparsity pattern of the Jacobian (b) Sparsity pattern of the Hessian

FIGURE S1.—Sparsity patterns of the Jacobian and the Hessian.

total number of elements is $(n(\theta)+n(\xi)+n(g))^2$, and the number of nonzero elements is around $2\times n(\theta)\times n(\xi)+T\times J^2+n(g)^2$. The density of the Hessian is also in the order of $\frac{1}{T}$. We illustrate graphically the sparsity patterns of the constraint Jacobian and the Hessian for $T=50$ markets and $J=25$ products in Figure S1(a). The y-axis indicates the position of rows, the x-axis indicates the position of columns, and the cells in blue color indicate the location of nonzero elements. nz is the number of nonzero elements in the matrix.

S1.2.2. *Derivatives of NFP*

The NFP formulation is

$$\min_{\theta} Q\big(\xi(\theta)\big) = \xi(\theta)' Z W Z' \xi(\theta),$$

where $\xi(\theta) = s^{-1}(S,\theta)$. By the implicit function theorem, we have

$$\frac{d\xi(\theta)}{d\theta} = -\left[\frac{\partial s(\xi;\theta)}{\partial \xi}\right]^{-1} \frac{\partial s(\xi;\theta)}{\partial \theta}.$$

The gradient of NFP is

$$\nabla_\theta Q(\theta) = \frac{d\xi(\theta)'}{d\theta}\frac{dQ(\xi(\theta))}{d\xi} = -2\frac{\partial s(\xi;\theta)'}{\partial \theta}\left[\frac{\partial s(\xi;\theta)'}{\partial \xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi}.$$

The Hessian of NFP is

$$\nabla_\theta^2 Q(\theta) = \left[\frac{d^2 Q(\theta)}{d\theta_k\, d\theta_m}\right],$$

where

$$\begin{aligned}
\frac{d^2 Q(\xi(\theta))}{d\theta_k\, d\theta_m} &= -\frac{d}{d\theta}\left(\frac{\partial s(\xi;\theta)'}{\partial \theta}\right)\left[\frac{\partial s'}{\partial \xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi} \\
&\quad - \frac{\partial s(\xi;\theta)'}{\partial \theta}\frac{d}{d\theta}\left(\left[\frac{\partial s(\xi;\theta)'}{\partial \xi}\right]^{-1}\right)\frac{dQ(\xi(\theta))}{d\xi} \\
&\quad - \frac{\partial s(\xi;\theta)'}{\partial \theta}\left[\frac{\partial s(\xi;\theta)'}{\partial \xi}\right]^{-1}\frac{d}{d\theta}\left(\frac{dQ(\xi(\theta))}{d\xi}\right) \\
&= -\left(\frac{\partial^2 s}{\partial\theta_k\,\partial\theta_m} - \frac{\partial^2 s}{\partial\theta_k\,\partial\xi}\left[\frac{\partial s}{\partial\xi}\right]^{-1}\frac{\partial s}{\partial\theta_m}\right)'\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi} \\
&\quad + \frac{\partial s'}{\partial\theta_k}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\left(\frac{\partial^2 s}{\partial\xi\,\partial\theta_m} + \sum_{j=1}^{JT}\left(\frac{d\xi}{d\theta_m}\right)_j\frac{\partial^2 s}{\partial\xi\,\partial\xi_j}\right)' \\
&\quad \times \left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi} \\
&\quad + \frac{\partial s'}{\partial\theta_k}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{d^2 Q(\xi(\theta))}{d\xi^2}\left[\frac{\partial s}{\partial\xi}\right]^{-1}\frac{\partial s}{\partial\theta_m} \\
&= -\frac{\partial^2 s'}{\partial\theta_k\,\partial\theta_m}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi} \\
&\quad + \frac{\partial s'}{\partial\theta_m}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{\partial^2 s'}{\partial\theta_k\,\partial\xi}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi} \\
&\quad + \frac{\partial s'}{\partial\theta_k}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{\partial^2 s'}{\partial\xi\,\partial\theta_m}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi} \\
&\quad - \frac{\partial s'}{\partial\theta_k}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{\partial^2 s'}{\partial\xi\,\partial\xi_j}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{dQ(\xi(\theta))}{d\xi}\left[\frac{\partial s}{\partial\xi}\right]^{-1}\frac{\partial s}{\partial\theta_m} \\
&\quad + \frac{\partial s'}{\partial\theta_k}\left[\frac{\partial s'}{\partial\xi}\right]^{-1}\frac{d^2 Q(\xi(\theta))}{d\xi^2}\left[\frac{\partial s}{\partial\xi}\right]^{-1}\frac{\partial s}{\partial\theta_m}.
\end{aligned}$$

Computing the gradient and the Hessian for NFP requires the evaluation of the terms $[\frac{\partial s(\xi;\theta)}{\partial\xi}]^{-1}\frac{\partial s(\xi;\theta)}{\partial\theta}$ and $[\frac{\partial s(\xi;\theta)'}{\partial\xi}]^{-1}\frac{\partial Q(\xi(\theta))}{\partial\xi}$. When the numbers of markets

and products are large, calculating $[\frac{\partial s(\xi;\theta)}{\partial \xi}]^{-1}\frac{\partial s(\xi;\theta)}{\partial \theta}$ and $[\frac{\partial s(\xi;\theta)'}{\partial \xi}]^{-1}\frac{\partial Q(\xi(\theta))}{\partial \xi}$ can become computationally prohibitive unless the sparsity structure of the matrix $[\frac{\partial s(\xi;\theta)}{\partial \xi}]$ is provided.

S1.3. *A Numerical Example*

In this section, we investigate the implications of supplying different levels of derivative information for the computational times of NFP and MPEC. We begin with a simple demand estimation example consisting of a single simulated data set with 50 markets, 25 products, and 1000 simulation draws. The demand parameter values are the same as those used in the paper. We compare the performance of NFP and MPEC using the KNITRO solver in two different implementations.

In the first implementation, we provide KNITRO with only first-order derivatives. For MPEC, we also provide the sparsity pattern of the constraint Jacobian. In the second implementation, we supply KNITRO with both first-order and second-order derivatives. For MPEC, we also provide the sparsity patterns of the constraint Jacobian and the Hessian. For each implementation, we use five different starting values for the parameter search.

We report the numerical results in Table S.I. For NFP, the computational time required for the implementation with both first- and second-order derivatives is only around 30% of the time required for the implementation with only first-order derivatives. The provision of the second-order derivatives reduces the numbers of iterations, function/gradient evaluations, and even contraction mapping iterations required for solving the estimation problem. In turn, the second-order derivatives reduce the overall computational time. MPEC exhibits a similar computational time improvement, with the implementation using both first- and second-order derivatives requiring only 30% of the time needed for the implementation with only first-order derivatives. When the second-order derivatives are provided, the solver does not need to approximate the Hessian with the gradient information. As a result, the number of

TABLE S.I

NUMERICAL RESULTS FOR SUPPLYING DIFFERENT LEVELS OF DERIVATIVE INFORMATION[a]

Algorithm	Derivatives	CPU Time (min.)	Major Iterations	Function Evaluations	Gradient Evaluations	Contraction Iterations
NFP	1st order	261	105	202	110	392,701
	1st and 2nd order	83	48	72	53	137,629
MPEC	1st order	49	125	163	1256	
	1st and 2nd order	14	102	145	107	

[a]The results are based on one data set and five starting values for each case. The mean intercept is 2. MPEC and NFP produce the same lowest objective value. See the footnote to Table III in the main text for other details.

gradient evaluations needed for estimating the model is reduced dramatically, reducing the overall computational time.

As we discussed above, for MPEC to solve demand estimation problems with a few thousand market-product pairs, the researcher needs to provide the sparsity patterns of the constraint Jacobian and the Hessian. In our experiment using 2000 markets and 15 products, both NFP and MPEC ran out of memory on a computer with 12 GB of RAM when sparsity information was not supplied. However, when sparsity information was supplied, MPEC was able to solve examples with 2000 markets and products using only 500 MB RAM.

S2. KNITRO OUTPUTS FOR MPEC AND NFP

In comparing the relative performance of different computational algorithms (e.g., NFP and MPEC), one should consider the computational burden associated with function evaluations. In this section, we provide sample KNITRO output for MPEC and NFP to illustrate the potential computational time cost associated with the evaluation of the functions (objective and constraints) and their derivatives. We intentionally focus on an example embodying a large-dimensional optimization problem. We use the Monte Carlo experiment reported in Section 7 of the paper, with a data set consisting of 500 markets, 25 products, and 1000 simulation draws. For MPEC, there are 12,552 variables and 12,542 constraints, which suggests that the numbers of elements in the constraint Jacobian and the Hessian are 157,427,184 ($= 15{,}452 \times 15{,}552$) and 157,552,794 ($= 15{,}552 \times 15{,}552$), respectively. However, the numbers of nonzero elements in the Jacobian and the Hessian are much smaller: 900,252 for the Jacobian and 225,918 for the Hessian. The Jacobian and the Hessian are highly sparse, with densities of 0.5% for the Jacobian and 0.14% for the Hessian. By supplying the sparsity patterns of the Jacobian and the Hessian, the solver does not need to store the full Jacobian and Hessian matrices in memory and, hence, can use sparse numerical linear algebra routines in the iteration process.

For this example, the computational times for NFP and MPEC are 26,854 CPU seconds and 1010 CPU seconds, respectively. A striking feature of the output for both MPEC and NFP is the fact that the majority of the computational time is spent on evaluating the functions and their derivatives. For MPEC, around 97% of the total computational time is spent on evaluating the objective function, the constraints (share equations), and their gradients and Hessians; only 3% of the 1009 seconds is used for solving the optimization problem. For NFP, almost all (99.99%) of the total computational time is spent on evaluating the GMM objective function and its gradient and Hessian; less than 1 second is spent on solving the optimization problem. The output also shows that NFP needs fewer iterations and fewer evaluations of the objective function, the gradient, and the Hessian in comparison with MPEC. However, NFP needs around 50,000 contraction mapping iterations in the solution process, which amounts to evaluating the share equations 50,000 times

(while MPEC needs to evaluate the share equations only 26 times). In summary, when comparing different computational methods, one should take into account the time and effort needed to evaluate functions and their derivatives. In this example, despite the high-dimension optimization problem, MPEC still turns out to be computationally lighter, and hence faster, than NFP due to the number of function evaluations.

KNITRO output for MPEC

```
=====================================
KNITRO 6.0.1
=====================================
algorithm: 1
maxit: 100
Problem Characteristics
-----------------------
Objective goal: Minimize
Number of variables: 12552
Number of constraints: 12542
Number of nonzeros in Jacobian: 900252
Number of nonzeros in Hessian: 225918
Iter Objective FeasError OptError ||Step|| CGits
-------- ------------- ---------- ---------- ---------- --
0  1.676037e+03 2.555e-05
1  3.913200e+02 5.336e-02 1.647e+03 3.708e+04 0
2  5.582143e+01 4.518e-02 5.407e+02 3.179e+04 0
3  5.490694e+01 6.111e-02 7.338e+02 9.727e+03 0
4  5.798389e+01 4.086e-02 2.650e-02 8.044e+02 18
5  4.232980e+01 5.781e-02 3.982e-02 5.688e+02 1
6  2.163262e+01 1.063e-02 4.623e+02 6.109e+03 0
7  2.103637e+01 2.238e-02 1.775e+02 6.327e+03 0
8  2.294612e+01 1.665e-02 1.652e-02 3.670e+02 9
9  1.928737e+01 2.025e-02 3.636e+00 6.691e+02 0
10 1.846350e+01 3.030e-03 6.082e+00 2.021e+03 0
11 1.829924e+01 1.249e-03 1.957e+00 1.332e+03 0
12 1.835531e+01 1.009e-05 1.301e-01 1.984e+02 0
13 1.835618e+01 2.535e-08 7.653e-05 7.147e+00 0
14 1.835618e+01 1.569e-14 1.000e-08 7.046e-03 0
EXIT: Locally optimal solution found.
Final Statistics
----------------
Final objective value = 1.83561823632553e+01
Final feasibility error (abs / rel) = 1.57e-14 / 1.57e-14
Final optimality error (abs / rel) = 1.00e-08 / 1.00e-08
# of iterations = 14
# of CG iterations = 28
# of function evaluations = 26
# of gradient evaluations = 15
# of Hessian evaluations = 14
```

```
Total program time (secs) = 1008.97015 ( 1010.558 CPU time)
Time spent in evaluations (secs) = 982.10559
============================================================
```

KNITRO output for NFP

```
====================================
KNITRO 6.0.1
Ziena Optimization, Inc.
====================================
algorithm: 1
maxit: 100
Problem Characteristics
-----------------------
Objective goal: Minimize
Number of variables: 5
Number of constraints: 0
Number of nonzeros in Jacobian: 0
Number of nonzeros in Hessian: 15
Iter Objective FeasError OptError ||Step|| CGits
-------- -------------- ---------- ---------- ---------- --
0 1.675984e+03 0.000e+00
1 7.598950e+02 0.000e+00 3.045e+01 8.941e-01 3
2 1.222743e+02 0.000e+00 3.258e+01 5.827e-01 0
3 2.846721e+01 0.000e+00 7.568e+00 1.314e-01 0
4 1.938900e+01 0.000e+00 1.233e+00 1.589e-01 0
5 1.840068e+01 0.000e+00 3.342e-01 1.952e-01 0
6 1.835634e+01 0.000e+00 1.483e-02 4.430e-02 0
7 1.835618e+01 0.000e+00 6.258e-05 2.657e-03 0
8 1.835618e+01 0.000e+00 9.762e-07 8.256e-06 0
EXIT: Locally optimal solution found.
Final Statistics
----------------
Final objective value = 1.83561823709107e+01
Final feasibility error (abs / rel) = 0.00e+00 / 0.00e+00
Final optimality error (abs / rel) = 9.76e-07 / 9.76e-07
# of iterations = 8
# of CG iterations = 3
# of function evaluations = 9
# of gradient evaluations = 9
# of Hessian evaluations = 8
Total program time (secs) = 26781.08008 ( 26854.102 CPU
  time)
Time spent in evaluations (secs) = 26780.39648
============================================================
```

S3. EXTENSION: MAXIMUM LIKELIHOOD ESTIMATION

In this section, we outline how a researcher would adapt static MPEC to a likelihood-based estimation of random-coefficients-logit demand. Some researchers prefer to work with likelihood-based estimators and, more specifically, with Bayesian MCMC estimators (Yang, Chen, and Allenby (2003), Jiang, Manchanda, and Rossi (2009)) based on the joint density of observed prices and market shares.[1] Besides efficiency advantages, the ability to evaluate the likelihood of the data could be useful for testing purposes. The trade-off relative to GMM is the need for additional modeling structure, which, if incorrect, could lead to biased parameter estimates. Like GMM, the calculation of the density of market shares still requires inverting the system of market share equations. Once again, MPEC can be used to circumvent the need for inverting the shares, thereby offsetting a layer of computational complexity and a potential source of numerical error. Below, we outline the estimation of a limited information approach that models the data-generating process for prices in a "reduced form" (this motivation is informal, as we do not specify a supply-side model and solve for a reduced form). However, one can easily adapt the estimator to accommodate a structural (full-information) approach that models the data-generating process for supply-side variables, namely prices, as the outcome of an equilibrium to a game of imperfect competition (assuming the equilibrium exists and is unique).

Recall that the system of market shares is defined by

$$\text{(S1)} \qquad s_j(x_t, p_t, \xi_t; \theta) = \int_\beta \frac{\exp(\beta^0 + x'_{j,t}\beta^x - \beta^p p_{j,t} + \xi_{j,t})}{1 + \sum_{k=1}^{J} \exp(\beta^0 + x'_{k,t}\beta^x - \beta^p p_{k,t} + \xi_{k,t})} dF_\beta(\beta; \theta).$$

We assume, as in a triangular system, that the data-generating process for prices is

$$\text{(S2)} \qquad p_{j,t} = z'_{j,t}\gamma + \eta_{j,t},$$

where $z_{j,t}$ is a vector of price-shifting variables and $\eta_{j,t}$ is a mean-zero, i.i.d. shock. To capture the potential endogeneity in prices, we assume that the supply and demand shocks have the joint distribution $(\xi_{j,t}, \eta_{j,t})' \equiv u_{j,t} \sim N(0, \Omega)$, where $\Omega = \begin{bmatrix} \sigma_\xi^2 & \sigma_{\xi,\eta} \\ \sigma_{\xi,\eta} & \sigma_\eta^2 \end{bmatrix}$. Let $\rho = \frac{\sigma_{\xi,\eta}}{\sigma_\xi \sigma_\eta}$.

[1] One can also think of Jiang, Manchanda, and Rossi (2009) as an alternative algorithm for finding the parameters. The Markov chain Monte Carlo (MCMC) approach is a stochastic search algorithm that might perform well if the BLP model produces many local optima, because MCMC will not be as likely to get stuck on a local flat region. Because our goal is not to study the role of multiple local minima, we do not explore the properties of a Bayesian MCMC algorithm.

The system defined by equations (S1) and (S2) has the joint density function

$$f_{s,p}(s_t, p_t; \Theta) = f_{\xi|\eta}(s_t \mid x_t, p_t; \theta, \Omega)|J_{\xi\to s}|f_\eta(p_t \mid z_t; \gamma, \Omega),$$

where $\Theta = (\theta, \gamma, \sigma_\xi^2, \sigma_{\xi,\eta}, \sigma_\eta^2)$ is the vector of model parameters, $f_{\xi|\eta}(\cdot|\cdot)$ is the marginal density of ξ conditional on η, $f_\eta(\cdot|\cdot)$ is a Gaussian density with variance σ_η^2, and $J_{\xi\to s}$ is the Jacobian matrix corresponding to the transformation of variables of $\xi_{j,t}$ to shares. The density of $\xi_{j,t}$ conditional on $\eta_{j,t}$ is

$$f_{\xi|\eta}(s_t \mid x_t, p_t; \theta, \Omega) = \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi}\sigma_\xi\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2} \frac{\left(\xi_{j,t} - \rho\dfrac{\sigma_\xi}{\sigma_\eta}\eta_{j,t}\right)^2}{\sigma_\xi^2(1-\rho^2)} \right).$$

Note that the evaluation of $\xi_{j,t}$ requires inverting the market share equations, (S1).

The element $J_{j,k}$ in row l and column k of the Jacobian matrix, $J_{\xi\to s}$, is

$$J_{j,l} = \begin{cases} \displaystyle\int_\beta \left(1 - \frac{\exp(\beta^0 + x_{j,t}'\beta^x - \beta^p p_{j,t} + \xi_{j,t})}{1 + \sum_{k=1}^{J} \exp(\beta^0 + x_{k,t}'\beta^x - \beta^p p_{k,t} + \xi_{k,t})} \right) \\ \quad \times \dfrac{\exp(\beta^0 + x_{j,t}'\beta^x - \beta^p p_{j,t} + \xi_{j,t})}{1 + \sum_{k=1}^{J} \exp(\beta^0 + x_{k,t}'\beta^x - \beta^p p_{k,t} + \xi_{k,t})} dF_\beta(\beta; \theta), \\ \quad j = l, \\ \displaystyle -\int_\beta \frac{\exp(\beta^0 + x_{j,t}'\beta^x - \beta^p p_{j,t} + \xi_{j,t})}{1 + \sum_{k=1}^{J} \exp(\beta^0 + x_{k,t}'\beta^x - \beta^p p_{k,t} + \xi_{k,t})} \\ \quad \times \dfrac{\exp(\beta^0 + x_{l,t}'\beta^x - \beta^p p_{l,t} + \xi_{l,t})}{1 + \sum_{k=1}^{J} \exp(\beta^0 + x_{k,t}'\beta^x - \beta^p p_{k,t} + \xi_{k,t})} dF_\beta(\beta; \theta), \\ \quad j \neq l. \end{cases}$$

Standard maximum likelihood estimation would involve searching for parameters, Θ^{LISML}, that maximize the log-likelihood function

$$l(\Theta) = \sum_{t=1}^{T} \log\bigl(f_{s,p}(s_t, p_t; \Theta)\bigr).$$

This would consist of a nested inner loop to compute the demand shocks, $\xi_{j,t}$, via numerical inversion (the NFP contraction mapping).

The equivalent MPEC approach entails searching for the vector of parameters (Θ, ξ) that maximizes the constrained optimization problem

$$\max l(\Theta, \xi) = \sum_{t=1}^{T} \log\bigl(f_{\xi|\eta}(s_t \mid x_t, p_t; \theta, \Omega)|J_{\xi\to s}|f_{\eta}(p_t \mid z_t; \gamma, \Omega)\bigr)$$
$$\text{subject to} \quad s(\xi; \theta) = S.$$

S4. MONTE CARLO: VARYING THE QUALITY OF THE DATA

In principle, the quality of the data could influence the convexity of the objective function and, hence, the trajectory of the outer-loop search. To assess the role of "data quality," we construct a set of sampling experiments that manipulate the power of the instruments (the correlation between the prices, p,

TABLE S.II

MONTE CARLO RESULTS VARYING THE DATA QUALITY: THE POWER OF INSTRUMENTS[a]

Instrument Power (ν)	R^2	Implementation	Runs Converged (fraction)	CPU Time (s)	Elasticities		
					Bias	RMSE	Value
1/2	0.75	NFP	0.50	619	0.028	0.173	−8.16
		MPEC	1	118	0.023	0.173	−8.16
1/4	0.69	NFP	1	342	0.058	0.140	−8.16
		MPEC	1	129	0.058	0.140	−8.16
1/6	0.62	NFP	1	447	−0.020	0.158	−8.15
		MPEC	1	135	−0.020	0.158	−8.15
1/8	0.57	NFP	1	376	−0.022	0.186	−8.13
		MPEC	1	135	−0.022	0.186	−8.13
1/16	0.46	NFP	1	512	−0.111	0.312	−8.07
		MPEC	1	159	−0.090	0.323	−8.05

[a]There are 20 replications for each experiment. Each replication uses ten starting values to ensure a global minimum is found. The NFP-tight implementation has $\epsilon_{\text{in}} = 10^{-14}$ and $\epsilon_{\text{out}} = 10^{-6}$. There is no inner loop in MPEC; $\epsilon_{\text{out}} = 10^{-6}$ and $\epsilon_{\text{feasible}} = 10^{-6}$. The same 100 simulation draws are used to generate the data and to estimate the model. The column R^2 reports the (mean across replications) R^2 from the regression of price on all instruments, treating each product in each market as a separate observation. The meaning of ν is described in the text. These simulations were run on a different computer than the simulations in the main text.

and the instruments, z). Let $z_{j,t,d} = \tilde{u}_{j,t,d} + \nu u_{j,t}$, where $u_{j,t}$ is a random shock that also affects price and ν is a measure of the power of the instruments. Higher ν's result in more powerful instruments. By generating the prices before the instruments, we ensure that prices, shares, and product characteristics are unaffected by the power of the instruments. Thus, the NFP inner loop and the MPEC market share constraints are identical when ν varies. Only the instruments in the moment conditions vary. We use the identity matrix for the GMM weighting matrix to avoid instrument power affecting the choice of weighting matrix.

Table S.II lists the results from five runs with differing levels of instrument power. The table lists the value of ν and the resulting R^2 from a regression of price on all excluded-from-demand and non-excluded instruments. We see that R^2 decreases as the power of the instruments decreases. In all specifications, MPEC is faster than NFP. MPEC's speed decreases with instrument power,

TABLE S.III

MONTE CARLO RESULTS FOR DYNAMIC BLP WITH ONE CONSUMER TYPE FOR $\delta = 0.96$: NFP VERSUS MPEC[a]

	MPEC		NFP		
Speeds:	335.55 secs.		553.50 secs.		
Parameters	Mean	RMSE	Mean	RMSE	Truth
Utility intercept, product 1	3.9557	0.1780	3.9556	0.1780	4.0000
Utility intercept, product 2	2.9572	0.2015	2.9572	0.2015	3.0000
Utility price coefficient, type 1	−1.0030	0.0101	−1.0030	0.0101	−1.0000
Price, product 1, constant	0.2111	0.0345	0.2111	0.0345	0.2000
Price, product 1, lagged price of product 1	0.7962	0.0136	0.7962	0.0136	0.8000
Price, product 1, lagged price of product 2	0.0026	0.0098	0.0026	0.0098	0.0000
Price, product 2, constant	0.2071	0.0378	0.2071	0.0378	0.2000
Price, product 2, lagged price of product 1	0.0037	0.0168	0.0037	0.0168	0.0000
Price, product 2, lagged price of product 2	0.7935	0.0156	0.7935	0.0156	0.8000
Demand shocks, Cholesky variance term	0.9958	0.0173	0.9958	0.0173	1.0000
Covariance btw supply and demand, Cholesky variance term	0.5015	0.0215	0.5015	0.0215	0.5000
Supply shocks, Cholesky variance term	0.8647	0.0152	0.8647	0.0152	0.8660
% of replications routine reports convergence	100%		100%		

[a]There are 20 replications for each of MPEC and NFP. The same synthetic data are used for both MPEC and NFP. Each replication uses five starting values to do a better job at finding a global minimum. The NFP implementation has $\epsilon_{\text{in}}^{\xi} = 10^{-14}$, $\epsilon_{\text{in}}^{V} = 10^{-14}$, and $\epsilon_{\text{out}} = 10^{-6}$. There is no inner loop in MPEC; $\epsilon_{\text{out}} = 10^{-6}$ and $\epsilon_{\text{feasible}} = 10^{-6}$. The data have $T = 50$ periods and $M = 20$ distinct markets. Each market has two competing products. The Chebyshev regression approximation to the value function uses a fourth-order polynomial and five interpolation nodes. The numerical integration of future states uses Gauss–Hermite quadrature with three nodes. NFP uses numerical derivatives, as coding the derivatives of dynamic BLP is infeasible for many problems and automatic differentiation slowed NFP considerably in the results in the main text. MPEC uses automatic differentiation in the form of the package MAD. The percentage of replications where the routine reports convergence is the fraction of the 20 replications where the lowest objective function coincided with an exit flag of 0 from KNITRO.

TABLE S.IV

MONTE CARLO RESULTS FOR DYNAMIC BLP WITH ONE CONSUMER TYPE FOR $\delta = 0.99$: NFP VERSUS MPEC[a]

	MPEC		NFP		
Speeds:	671.49 secs.		1295.50 secs.		
Parameters	Mean	RMSE	Mean	RMSE	Truth
Utility intercept, product 1	4.0684	0.7235	3.3907	1.7473	4.0000
Utility intercept, product 2	3.0692	0.7316	2.3913	1.7844	3.0000
Utility price coefficient, type 1	−0.9885	0.0380	−0.9987	0.0152	−1.0000
Price, product 1, constant	0.1929	0.0682	0.2171	0.0655	0.2000
Price, product 1, lagged price of product 1	0.8170	0.0532	0.8022	0.0546	0.8000
Price, product 1, lagged price of product 2	−0.0044	0.0295	0.0000	0.0520	0.0000
Price, product 2, constant	0.1770	0.1102	0.2058	0.0813	0.2000
Price, product 2, lagged price of product 1	−0.0195	0.0557	−0.0065	0.0436	0.0000
Price, product 2, lagged price of product 2	0.8330	0.0860	0.8089	0.0585	0.8000
Demand shocks, Cholesky variance term	1.0139	0.0468	1.0053	0.0334	1.0000
Covariance btw supply and demand, Cholesky variance term	0.4985	0.0255	0.5050	0.0219	0.5000
Supply shocks, Cholesky variance term	0.8652	0.0152	0.8640	0.0159	0.8660
% of replications routine reports convergence	100%		90%		

[a]See footnote to Table S.III for details.

although the decrease from 118 to 159 seconds is not large compared to the total run time of NFP, which ranges from 342 to 619 seconds. We have no theoretical explanation for the pattern relating NFP's speed and instrument power. To some extent, the pattern is driven by the fact that NFP encounters convergence problems as we increase the power of the instruments. When $\nu = 0.5$, only 50% (10 out of 20) of the replications fail to converge for all five starting values, for NFP. Naturally, this convergence problem could be a practical concern if the researcher mistakenly interprets the point at which the solver stops as a local minimum even though the gradient-based solver is unable to detect convergence.

S5. MONTE CARLO: DYNAMIC BLP WITH ONE CONSUMER TYPE

Results from the Monte Carlo experiments using $\delta = 0.96$ and $\delta = 0.99$ are presented in Tables S.III and S.IV, respectively.

REFERENCES

JIANG, R., P. MANCHANDA, AND P. E. ROSSI (2009): "Bayesian Analysis of Random Coefficient Logit Models Using Aggregate Data," *Journal of Econometrics*, 149, 136–148. [12]

YANG, S., Y. CHEN, AND G. M. ALLENBY (2003): "Bayesian Analysis of Simultaneous Demand and Supply," *Quantitative Marketing and Economics*, 1, 251–304. [12]

University of Chicago Booth School of Business, Chicago, IL 60637, U.S.A. and NBER; jdube@chicagobooth.edu,

Dept. of Economics, University of Michigan, Ann Arbor, MI 48109, U.S.A. and NBER; jtfox@umich.edu,

and

University of Chicago Booth School of Business, Chicago, IL 60637, U.S.A.; Che-Lin.Su@chicagobooth.edu.

Manuscript received May, 2009; final revision received October, 2011.