

An MM Algorithm for Fixed Effects Multinomial Logit Models

Mingli Chen	Ian Meeker
University of Warwick	Charles River Associates
Marc Rysman	Shuang Wang
Boston University	Charles River Associates*

October 9, 2026

Abstract

Estimating multinomial logit models with many fixed effects or individual-specific coefficients faces prohibitive computational challenges. We develop a new framework to address this issue that leverages the Minorization-Maximization (MM) algorithm. This method transforms the logit likelihood into a sequence of tractable least-squares regressions, allowing the use of linear methods for handling fixed effects. When combined with nonlinear acceleration methods, our approach is

*The opinions expressed in this article are the authors' own and do not reflect the views of Charles River Associates. The empirical application relies on data from the U.S. Federal Trade Commission and was completed during Meeker's tenure at the Commission. The views expressed here are solely those of the authors and do not necessarily reflect those of the Commission or its Commissioners. We thank Devesh Raval and Ted Rosenbaum for their feedback and for sharing the data used in the empirical application. We also thank Keith Brand and Daniel Grodzicki for their comments.

highly memory-efficient and dramatically faster than existing alternatives for estimating fixed effects, while remaining highly competitive and easier to implement for individual-specific coefficients. We demonstrate the practical value and flexibility of our estimator through robust numerical simulations and an empirical application to hospital choice.

Keywords: MM algorithm, multinomial logit, fixed effects, jack-knife bias correction, acceleration methods

1 Introduction

Implementing models with many fixed effects is critical to modern empirical work and plays a central role in what Angrist and Pischke (2010) term the credibility revolution in economics. In linear models, methods for addressing large numbers of unobserved individual effects, such as demeaning, are well-known. However, these approaches do not directly extend to non-linear models, such as the multinomial logit. Instead, standard implementations of multinomial estimation typically rely on estimating dummy variable coefficients, which becomes computationally slow and memory intensive, or even infeasible, as the number of fixed effects grows large. While there are a number of methods for applying fixed effects approaches to binary outcome models, extending these methods to multinomial models has remained a significant challenge.

In this paper, we introduce a new framework for calculating maximum likelihood estimates for a multinomial logit model with fixed effects. Our approach draws on the Minorization-Maximization (MM) algorithm, which can be seen as a generalization of the Expectation-Maximization (EM) algorithm (Hunter and Lange, 2004). While the MM algorithm was developed in the statistics literature (Hunter and Lange, 2004; Lange, 2016) and was introduced for multinomial logit estimation by Böhning (1992), who does not consider fixed effects, it has seen little application in econometrics to date. Applications of quadratic-bound MM to multinomial logistic regression have, to our knowledge, been confined to settings with a shared, low-dimensional coefficient vector and none confront the incidental parameters problem

that arises from individual, time, or multi-way fixed effects, nor the resulting need for bias-corrected point estimates and valid inference.

We build on this statistical foundation by providing a new MM algorithm based on a deliberately looser quadratic bound—a simple scalar multiple of the identity matrix, in place of Böhning’s tighter, correlation-aware matrix bound. As we show, for alternative-specific parameters like fixed effects, our algorithm using this simple scalar bound mathematically matches the exact parameter trajectory of Böhning’s method. However, our scalar relaxation is not merely a simplification. While Böhning’s bound takes better advantage of curvature, our scalar identity bound yields a fast Ordinary Least Squares (OLS) update rather than the more involved Generalized Least Squares (GLS) implied by Böhning’s bound. Because our method is easier to compute, it is thus faster than the matrix bound approach.

The scalar bound is the key that unlocks our primary contribution: transforming non-linear logit estimation into a sequence of tractable linear regressions to absorb high-dimensional fixed effects. This theoretical mapping allows us to seamlessly import linear panel data techniques, such as Frisch-Waugh-Lovell demeaning, enabling us to estimate models with millions of fixed effects. We extend the approach to a block-coordinate variant that accommodates individual-specific coefficients, and pair the resulting estimator with a split-panel jackknife bias correction (Dhaene and Jochmans, 2015). We also show how to efficiently compute cluster-robust standard errors that avoid ever forming or inverting the massive multi-way Hessian.

We use simulations to demonstrate the computational advantages of our new methods over competing alternative methods. We consider three sets of methods. The first set consists of MM-based methods, which includes the scalar-bound MM algorithm that we propose and Böhning’s MM algorithm. For this set of methods, we also show how the algorithms perform with an acceleration method that is applicable to EM and MM settings, and a conditional MM algorithm that maximizes subsets of parameters using sequential conditioning arguments. The second set of methods is based on Newton’s method. Standard implementations of multinomial logit estimation, such as those available in Stata or R, rely on Newton’s method and handle fixed effects by introducing indicator variables for each group,

the so-called dummy variables approach. These implementations create enormous Hessian matrices to invert and so are extremely slow or run into memory problems in our simulations. To advantage Newton’s method, we also implement a highly optimized version that exploits the sparsity and structure of the Hessian, following Hospido (2012). The third approach uses the optimization algorithm Adam, as recommended for the multinomial logit by Du, Kanodia, and Athey (2023).

We explore how these methods scale with the number of fixed effects and the number of individual-specific coefficients on a continuous explanatory variable. In terms of coefficients, all methods return essentially the same coefficients. In terms of speed, we find that the acceleration methods pair well with the MM algorithms. Our methods, particularly with acceleration, outperform competing ones when handling fixed effects. Our algorithm outperforms some alternatives when handling large numbers of individual-specific coefficients, but struggles more as the scalar bound does not take into account dependencies between alternatives.

While highly-optimized Newton’s method is a strong competitor, any speed improvement over methods is small relative to the level of bespoke programming required to achieve it. Generally applicable code for implementing our MM algorithm with and without acceleration methods is available on author websites.¹ We also find that conditional maximization can further improve computational speed in the case of individual-specific coefficients, where demeaning is less effective. This approach performs well in our simulations because the parameter blocks are largely independent. We discuss below conditions under which that may not hold.

We conclude with an empirical application to hospital choice based on Raval, Rosenbaum, and Tenn (2017). Their method has been applied in several recent papers (i.e., Barrette, Gowrisankaran, and Town, 2022; Brot et al., 2024) and has been influential in antitrust settings (i.e., Tenn and Vandergrift, 2017). This line of research emphasizes highly flexible choice models in order to accurately determine the relevant market for competition policy purposes, which can be implemented with large numbers of fixed effects. In part because of the estimation challenge of estimating so many fixed effects, Raval, Rosenbaum, and Tenn (2017) estimate separate multinomial logit models for each sub-population in the data with just

¹See in particular: imeeker.com/research

an intercept for each hospital as the only explanatory variable. This approach means the researcher cannot hold any coefficients constant across the whole population, and cannot include any other explanatory variables, such as time dummies, a continuous variable, or multi-way fixed effects. In this context, we show our estimator accommodates population-wide estimation of all of their parameters, and can easily handle these additional explanatory variables as well.

Overall, our paper makes several contributions. We provide an attractive new framework to estimate the multinomial discrete choice models with fixed effects or individual-specific coefficients. Unlike many alternative approaches to handling fixed effects in nonlinear models, our methods allow the researcher to recover individual-specific fixed effects and coefficients, which is valuable if the researcher wants to make use of them in predicting outcomes or constructing summary variables. Within the MM literature, we provide a new minorization for the multinomial logit model and show how it can be used to efficiently address panel data estimation. We also show how to reformulate Böhning’s (1992) algorithm as a regression problem, unlocking the use of linear panel methods to this existing algorithm. Our general approach could likely be extended to a number of linear-index likelihood models, allowing a wide range of non-linear models to handle many fixed effects.

Literature Review Our paper contributes to the literature on fixed effects estimation of nonlinear panel data models. Several other papers precede us in this regard. A classic contribution is Chamberlain’s conditional logit model (Chamberlain, 1980), which relies on conditioning arguments to avoid estimating the fixed effects. Typical implementations handle only the binary outcome case, and extending the model to multinomial outcomes creates significant combinatoric complexities. Furthermore, the model does not naturally deliver estimates for the fixed effects, which our approach does. A recent example of a relevant applied paper is Dubois, Griffith, and O’Connell (2020), which estimates a model of soda taxes in a multinomial logit with individual-specific coefficients on price and several indicator variables. They use the parameter results to construct compensating variation, which requires recovering the individual-level coefficients, which our model allows.

D’Haultfœuille and Iaria (2016) and Hospido (2012) introduce efficient methods for estimating the dummy variables model and the latter provides a bias-corrected likelihood approach. D’Haultfœuille and Iaria (2016) rely on simulation (which introduces integration error) of the choice set to cheaply compute the Hessian of the objective function, while Hospido (2012) exploits the sparsity of the fixed effects. When calculating the Hessian for Newton’s method, we adapt Hospido (2012) to our setting.

Stammann, Hei, and McFadden (2016) is similar to us in that they advocate for iterative demeaning to obtain estimates, and then ex-post bias correction, in their case based on Hahn and Newey (2004). Several techniques rely on concentrating out fixed effects so the researcher can maximize over the remaining parameters, such as Hinz, Hudle, and Wanner (2019) and Stammann (2018). Our understanding is that all three of these approaches have been developed only for binary outcomes and do not easily extend to multinomial settings.

Another approach relies on differencing out fixed effects in a way that leads to estimation with moment inequalities, such as Ho and Pakes (2014) and Shum, Song, and Shi (2018). Our approach differs in that it generates point identification and delivers estimates of the fixed effects, and we view our approach as computationally less challenging than estimation with moment inequalities. It is possible to replace the multinomial logit model with the Poisson model, as in Guimaraes, Figueirdo, and Woodward (2003), which is attractive because the Poisson model more easily accommodates fixed effects. However, this technique requires groups of agents to have the same explanatory variables.

Our discussion focuses on estimation methods. For identification, Mugnier and Wang (2024) study identification in a linear index model with a non-parametric link function. Similar to our paper, they study the cases of fixed effects, two-way fixed effects, and individual-specific slope coefficients, and so cover our applications.

The Minorization-Maximization algorithm is sometimes called the Majorization-Minimization algorithm, but is the MM algorithm in either case. It has appeared under many other names as well, often depending on what function was used. Relative to the EM algorithm, the MM algorithm expands the set of functions

that can be used in the first step (the E-step of the EM algorithm). Böhning and Lindsay (1988) is an important early citation, which dates the origin of the MM algorithm similar to EM algorithm. Hunter and Lange (2004) and Lange (2016) provide a helpful overview and history.

The closest paper to ours in the statistics literature is probably Böhning (1992), which also develops an MM algorithm for the multinomial logit. While Böhning (1992) does not mention fixed effects, we show how his method can be adapted to a panel setting and connects to our algorithms. Another related MM paper is James (2017), who shows that the MM algorithm can be advantageous in the context of the mixed multinomial logit model of McFadden and Train (2000) but does not discuss fixed effects.

We are aware of several papers making use of our algorithm. Sahai (2023) embeds our MM algorithm into an EM algorithm to allow for the estimation of many fixed effects in a mixed logit model. He applies his EM algorithm to estimate a model of school choice in India. Rysman, Wang, and Wozniak (2025) applies our model to study a panel of consumers choosing between payment type (such as card, cash, or check). They estimate a multinomial logit model with more than 1.4 million fixed effects.

Our paper is closely related to that of Chen (2019). She proposes a new EM algorithm for nonlinear panel data models with interactive effects. She uses the EM algorithm to transform the model into a linear form (with generalized residuals), enabling the application of existing techniques for handling interactive fixed effects in the linear case. Additionally, she derives the bias formula, and with its known analytic form, one can perform ex-post bias correction. Notably, her implementation of the EM algorithm closely resembles the MM algorithm. However, she does not consider multinomial models. Building on her ideas, our model could be extended to accommodate interactive fixed effects in a multinomial logit model. However, caution is required, as the log-likelihood function would no longer be concave, and the MM algorithm would, at best, converge to a local maximum.

2 The MM Algorithm

The central idea in our paper is to use the MM algorithm to estimate a nonlinear discrete choice model in an iterative series of two steps. First, in each iteration, we construct a simpler concave surrogate function that minorizes the complicated log-likelihood function, such that the surrogate function is less than the log-likelihood function everywhere but at the current best guess of the parameters, where it is equal. Second, we maximize the surrogate function instead of the log-likelihood function. The ascent of the log-likelihood is guaranteed by the properties of minorization. By alternating between these steps of minorization and maximization, the MM algorithm finds the parameters that maximize the original log-likelihood function.

Before diving into the details of our multinomial logit model, we first discuss the definition of the MM algorithm in general and provide conditions required for convergence. Relative to the standard mathematical definition of a minorizing function, we add an extra condition that makes the function suitable for use in a maximization problem. We refer to our minorization as a *transfer minorization*. Our name is based on the terminology of Lange, Hunter, and Yang (2000), which refers to the minorization as the *transfer function*. That is, we transfer optimization from the function of interest to the minorization of this function.

Definition 1. Let $\Omega \subseteq \mathbb{R}^p$ be a closed subset, $\mathcal{L} : \Omega \rightarrow \mathbb{R}$ be a twice differentiable function, and $S : \Omega \times \Omega \rightarrow \mathbb{R}$ be a real-valued function. We say that S is a transfer minorization of $\mathcal{L}$ if the following conditions hold for all $\boldsymbol{\theta}, \boldsymbol{\vartheta} \in \Omega$:

- (a) $S(\boldsymbol{\theta}; \boldsymbol{\vartheta}) \leq \mathcal{L}(\boldsymbol{\theta})$;
- (b) $S(\boldsymbol{\vartheta}; \boldsymbol{\vartheta}) = \mathcal{L}(\boldsymbol{\vartheta})$;
- (c) $\nabla_{\boldsymbol{\theta}}^2 S(\boldsymbol{\theta}; \boldsymbol{\vartheta})$ —which denotes the second derivative of S with respect to $\boldsymbol{\theta}$ —exists and is negative definite at $\boldsymbol{\theta}$.

Analogously, S is a *transfer majorization* of $\mathcal{L}$ if $-S$ is a transfer minorization of $-\mathcal{L}$. The first two conditions of Definition 1 are from Leeuw and Lange (2009)

and are standard for defining a minorization. The third condition ensures that the minorization is well-behaved around the focal point.

As shown in Leeuw and Lange (2009), Definition 1 implies:

Corollary 1. *If S is a transfer minorization of $\mathcal{L}$, then for all $\boldsymbol{\theta} \in \Omega$:*

$$\nabla_{\boldsymbol{\theta}} S(\boldsymbol{\theta}; \boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta}).$$

This follows because, at $\boldsymbol{\theta} = \boldsymbol{\vartheta}$, the gradients of S and $\mathcal{L}$ must coincide for S to be a minorization.

Intuitively, when maximizing a challenging likelihood function $\mathcal{L}$, we instead iteratively maximize a surrogate function S . Each maximizer of $S(\cdot; \boldsymbol{\theta}^{(k)})$ should improve the objective, and S is chosen to be easier to optimize than $\mathcal{L}$. Figure 1 provides an example for a scalar parameter θ (we write θ instead of $\boldsymbol{\theta}$ as we are considering a scalar). Starting from an initial guess $\theta^{(k)}$, we construct a transfer minorization $S(\theta; \theta^{(k)})$, which lies below $\mathcal{L}$ and matches it at $\theta^{(k)}$. The concavity of S ensures a well-defined maximizer $\theta^{(k+1)}$. Repeating this process converges (under suitable regularity conditions) to a local maximum of $\mathcal{L}$.

The general MM algorithm starts from an initial guess $\boldsymbol{\theta}^{(0)}$. The MM algorithm iteratively applies the following two-step procedure. In the first step, *the minorization step*, the surrogate function $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$ is computed for the current iterate $\boldsymbol{\theta}^{(k)}$. In the second step, *the maximization step*, calculate the next iterate $\boldsymbol{\theta}^{(k+1)}$, which is the value $\boldsymbol{\theta} \in R^p$ that maximize $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$. At each step, let $\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k+1)}$ and repeat the procedure until $\boldsymbol{\theta}^{(k)}$ converges.

Lange (2013, Chapter 8) establishes convergence of the MM algorithm.² Thus, any function that satisfies Definition 1 for some function $\mathcal{L}$ can be used in the MM algorithm to find an optimum to $\mathcal{L}$. This approach allows substantial freedom in selecting the transfer minorization. From the perspective of the MM algorithm, the EM algorithm is a special case and it works because the conditional expectation

²Lange (2013) does not include part (c) of Definition 1 in the definition of a minorization but then later imposes it for the proof of convergence.

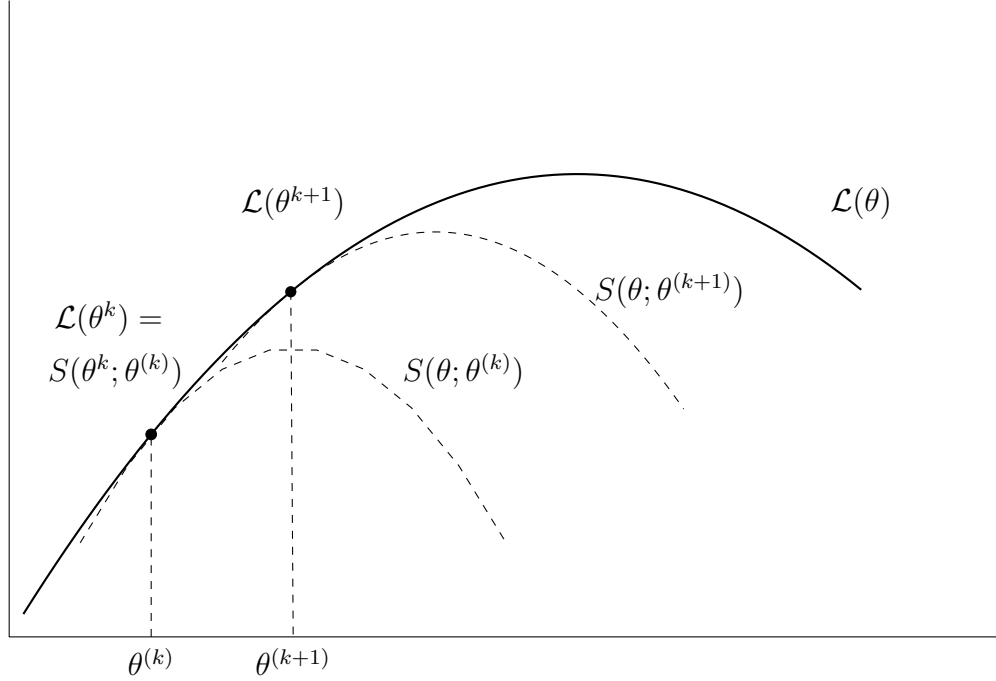


Figure 1: One-dimensional illustration of MM algorithm as a minorization or optimization transfer strategy.

of $\mathcal{L}$ used in the EM algorithm is a transfer minorization. The MM algorithm is more general because the researcher is free to use any minorizing function. Hunter and Lange (2004) discuss several approaches for constructing MM algorithms. In the next section, we focus on a Taylor expansion, which we use to deliver a least-squares optimization problem in our context.

3 Model

In the multinomial logit model, an individual makes a discrete choice among several alternatives, each of which draws an Extreme Value error. The distinguishing feature of the model is the closed-form logistic function for the probability of each choice.

We observe I individuals make a discrete choice among J alternatives in each of

T time periods.³ Individual i in period t who chooses alternative j obtains utility u_{ijt} . Utility is defined as:

$$u_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij} + \varepsilon_{ijt}, \quad (1)$$

where x_{it} is a set of individual-specific observable characteristics; w_{jt} is a set of alternative-specific characteristics; and α_{ij} is a set of fixed effects at the household-by-alternative level. For instance, if we observe households choosing the method of transportation to work, x_{it} might represent household income, w_{jt} might represent the cost of various transportation modes, and α_{ij} would be separate fixed effects for every household and alternative combination. We typically normalize one alternative to have a utility of zero, so α_{ij} represents $I \times (J - 1)$ fixed effects. It is possible to allow w_{jt} to vary across individuals (i.e., w_{ijt}), for instance, if prices were specific to the household.

Continuing with Eq.(1), let ε_{ijt} be an *i.i.d* idiosyncratic shock following a Type I Extreme Value distribution. The variable y_{ijt} is a binary indicator for the alternative that consumer i chooses in t , where:

$$y_{ijt} = \mathbb{1}[u_{ijt} \geq u_{ikt}, \forall k], \quad \forall i, t.$$

Define the parameter vector $\boldsymbol{\theta} = \{\beta_j, \gamma_i, \alpha_{ij} : i = 1, \dots, I; j = 1, \dots, J\}$. Let

$$\psi_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij}, \quad \boldsymbol{\psi}_{it} = (\psi_{i1t}, \dots, \psi_{iJt}),$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_J)$, $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_I)$, and $\boldsymbol{\alpha}_i = (\alpha_{i1}, \dots, \alpha_{iJ})$. The log-likelihood is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i,t} \log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}), \quad (2)$$

with $\mathbf{y}_{it} = \{y_{i1t}, \dots, y_{iJt}\}$ and

$$l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}) = \prod_{j=1}^J p_{ijt}(\boldsymbol{\psi}_{it})^{y_{ijt}}, \quad p_{ijt}(\boldsymbol{\psi}_{it}) = \frac{\exp(\psi_{ijt})}{\sum_{m=1}^J \exp(\psi_{imt})}.$$

³The number of alternatives can differ across time and consumers. For ease of presentation, we consider a balanced panel.

In practice, we normalize the mean utility of one of the alternatives to be zero, as is standard in the multinomial logit.

4 An MM Algorithm for the Multinomial Logit

4.1 A New Transfer Minorization

We now derive a function S satisfying Definition 1 as a transfer minorization of the multinomial logit log-likelihood $\mathcal{L}$. We start from the second-order Taylor expansion of the log-likelihood around an iterate $\theta^{(k)}$.⁴ We construct an MM surrogate by replacing the Hessian in the third term with the scalar-identity $-0.5I$ and completing the square. Thus, we use a lower bound for the third term in the Taylor expansion. Our approximation has less curvature than the Taylor expansion, although we choose it to be relatively easy to compute.

Theorem 1. *Let $\mathcal{L}(\theta)$ be in Eq.(2). Define*

$$S(\theta; \theta^{(k)}) = \mathcal{L}(\theta^{(k)}) + \frac{1}{4} \sum_{i,t} \mathbf{h}(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it})' \mathbf{h}(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it}) - \frac{1}{4} \sum_{i,t} \left(\mathbf{v}_{it}^{(k)} - \boldsymbol{\psi}_{it} \right)' \left(\mathbf{v}_{it}^{(k)} - \boldsymbol{\psi}_{it} \right) \quad (3)$$

where

$$\begin{aligned} h_j(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it}) &= 2 \left(y_{ijt} - p_{ijt}(\boldsymbol{\psi}_{it}^{(k)}) \right), \\ \mathbf{h}(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it}) &= \{h_1(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it}), \dots, h_J(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it})\} \\ p_{ijt}(\boldsymbol{\psi}_{it}) &= \frac{\exp(\psi_{ijt})}{\sum_{m=1}^J \exp(\psi_{imt})}, \\ \psi_{ijt}^{(k)} &= x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)}, \\ \psi_{ijt} &= x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij}, \\ \boldsymbol{\psi}_{it}^{(k)} &= \{\psi_{i1t}^{(k)}, \dots, \psi_{iJt}^{(k)}\} \\ \boldsymbol{\psi}_{it} &= \{\psi_{i1t}, \dots, \psi_{iJt}\} \\ \mathbf{v}_{it}^{(k)} &= \boldsymbol{\psi}_{it}^{(k)} + \mathbf{h}(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it}) \end{aligned}$$

Then $S(\theta; \theta^{(k)})$ is a transfer minorization of $\mathcal{L}(\theta)$.

⁴We write out the full Taylor expansion in Equation 38 in Appendix A.1 for interested readers.

Therefore $\boldsymbol{\theta}^{(k+1)} = \arg \max_{\boldsymbol{\theta}} S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$, $k = 0, 1, 2, \dots$ converges to a local, if not global, maximum of $\mathcal{L}(\boldsymbol{\theta})$.

The parameters that we are maximizing over MM algorithm, $\boldsymbol{\theta}$, appear only in the third part of the right-hand side of Eq.(3). Focusing on this third part, we can view the optimization of $S(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k)})$ as a linear regression of the pseudo-data $v_{ijt}^{(k)} = x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it})$ on x_{it} , w_{jt} , and household-alternative fixed effects. Maximizing this surrogate function with respect to the parameters $\boldsymbol{\theta}$ is equivalent to solving the OLS problem:

$$\min_{\boldsymbol{\theta}} \sum_{i,t} \left(\mathbf{v}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right)' \left(\mathbf{v}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right) \quad (4)$$

where: $\mathbf{Z}_{it}$ is the matrix formed by the independent variables for individual i at time t and $\boldsymbol{\psi}_{it} = \mathbf{Z}_{it}\boldsymbol{\theta}$.

The central benefit of our MM approach to the multinomial logit is that we convert a non-linear optimization problem to a series of sequential linear optimization. This allows us to apply linear methods like demeaning to a nonlinear estimation problem.

The functional form of $h_j(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it})$ is clearly important to our technique. For this bound, this function takes on a particularly simple form: $2 \left(y_{ijt} - p_{ijt}(\boldsymbol{\psi}_{it}^{(k)}) \right)$. Thinking of $x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)}$ as the expectation of u_{ijt} at iteration k , our iterative linear regression uses a dependent variable above the expectation for observations with $y_{ijt} = 1$ and below the expectation for observations with $y_{ijt} = 0$.

If we could observe u_{ijt} , we could estimate our coefficients and fixed effects directly by linear techniques rather than relying on discrete outcome methods such as the multinomial logit. In this sense, the minorization step of the MM algorithm is a form of *data augmentation* (Tanner, 1996). That is, we calculate v_{ijt} as an approximation of the unobserved u_{ijt} . That is the intuition behind many applications of the EM algorithm.

In the multinomial logit, v_{ijt} does not coincide with the expectation of u_{ijt} . That is, $v_{ijt}^{(k)} \neq E[u_{ijt} | x_{it}, w_{jt}, \alpha_{ij}, y_{ijt}, \boldsymbol{\theta}_j^{(k)}]$. Indeed, substituting $v_{ijt}^{(k)}$ with $E[u_{ijt} | x_{it}, w_{jt}, \alpha_{ij}, y_{ijt}, \boldsymbol{\theta}_j^{(k)}]$

would lead to biased results because the likelihood of the expectation is not equal to the expectation of the likelihood. Generating the expectation of the likelihood function for the case of the multinomial logit is more complicated than our minorization approach, so the EM algorithm is relatively unattractive in our context.

Remark 1 (A Digression on the Binary Probit). In the case of the binary probit, the MM and EM algorithms coincide. Chen (2019) estimates the binary probit by the EM algorithm and derives a functional form equivalent to our MM algorithm. Ruud (1991) and Greene (2018) also discuss applying the EM to the binary probit model. In the case of binary probit:

$$l(x_{it}\beta + w_t\gamma_i + \alpha_i; y_{it}) = \begin{cases} \Phi(x_{it}\beta + w_t\gamma_i + \alpha_i), & y_{it} = 1 \\ 1 - \Phi(x_{it}\beta + w_t\gamma_i + \alpha_i), & y_{it} = 0 \end{cases}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, $\phi(\cdot)$ is its probability density function. We have suppressed j subscripts relative to the previous equations as there are only two alternatives.

For ease of notation, let $\psi_{it} = x_{it}\beta + w_t\gamma_i + \alpha_i$. Accordingly:

$$h(\psi_{it}; y_{it}) = \frac{(y_{it} - \Phi(\psi_{it})) \phi(\psi_{it})}{\Phi(\psi_{it}) (1 - \Phi(\psi_{it}))}$$

Thus, we see that $h(\psi_{it}; y_{it})$ corresponds to the well-known Mills ratio and, as a result, $h(\psi_{it}; y_{it}) = E[\epsilon_{it} | \psi_{it}, y_{it}]$, where ϵ_{it} corresponds to the normally distributed error from the probit model. In this sense, the MM and EM algorithms correspond in the case of the binary probit.

Remark 2 (Normalization). The multinomial logit is identified up to a constant and requires a normalization. The typical normalization is to set the linear index of the base alternative equal to 0. This means estimating parameters for only $J - 1$ alternatives. The OLS problem, however, updates the parameters based on the full space of J alternatives. As a result, our MM algorithm generally produces a non-zero update for the base option fixed effects, which we show in Appendix D. The update is a function of the starting value and the difference between the observed choice and the predicted probabilities, adjusted for any movement in the base slope. Although this does not create problems for convergence, we still normalize ex post. That is, we estimate the model allowing the algorithm to update all of

the fixed effects and then subtract off the base alternative fixed effects from the other fixed effects at the end, so we still report the base alternative fixed effects equal to zero.

4.2 Connection to Böhning (1992)

Böhning (1992) provides an alternative MM algorithm for the multinomial logit. He also starts from the Taylor series approximation, but replaces the Hessian with a matrix bound, which is tighter than our scalar bound. Böhning's bound takes into account dependencies between alternatives. Working in the space of $J - 1$ alternatives, Böhning's bound is:⁵

$$B = \frac{1}{2}(I_{J-1} - \frac{1}{J}\mathbf{1}\mathbf{1}') \quad (5)$$

Using this bound, we obtain the surrogate function:

$$\dot{S}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) = \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \frac{1}{2} \sum_{i,t} \left(\dot{\mathbf{h}}_{it}^{(k)} \right)' B^{-1} \dot{\mathbf{h}}_{it}^{(k)} - \frac{1}{2} \sum_{i,t} \left(\dot{\mathbf{v}}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right)' B \left(\dot{\mathbf{v}}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right) \quad (6)$$

where the pseudo-data target are defined as:

$$\dot{\mathbf{v}}_{it}^{(k)} = \boldsymbol{\psi}_{it}^{(k)} + B^{-1} \dot{\mathbf{h}}_{it}^{(k)}, \quad (7)$$

$$\dot{\mathbf{h}}_{it}^{(k)} = \mathbf{y}_{it} - \mathbf{p}_{it}^{(k)} \quad (8)$$

While Böhning (1992) presents the solution of his surrogate as a modified Newton step, treating the update from a purely numerical optimization perspective, we can instead think of this surrogate from a regression perspective. As with our surrogate, the parameters to be estimated $\boldsymbol{\theta}$ appear only in the last term. Maximizing

⁵For ease of exposition, we consider the case where all consumers face the same choice set over time. With varying choice sets, we would need to index Böhning's bound B and the number of alternatives J by i and t

Böhning’s surrogate is equivalent to the generalized least squares problem:

$$\min_{\boldsymbol{\theta}} \sum_{i,t} \left(\dot{\mathbf{v}}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right)' \mathbf{B} \left(\dot{\mathbf{v}}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right) \quad (9)$$

Translating Böhning’s MM algorithm into a regression framework makes it clear how we can apply linear methods to handle fixed effects. Relative to our method, Böhning’s better approximates the curvature of the objective function and thus converges in fewer iterations. However, it uses a GLS maximization step, which is slower than our OLS maximization.

Böhning’s pseudo-data bears a close resemblance to the pseudo-data for our MM algorithm. If we multiply by $\mathbf{B}^{-1}$, Böhning’s pseudo-data simplifies to:

$$\dot{\mathbf{v}}_{it}^{(k)} = \boldsymbol{\psi}_{it}^{(k)} - \boldsymbol{\psi}_{i1t}^{(k)} \mathbf{1} + 2(\mathbf{y}_{it} - \mathbf{p}_{it}^{(k)}) - 2(y_{i1t} - p_{i1t}) \mathbf{1} \quad (10)$$

The first two terms are the same as in our pseudo-data. The reason is that the inverse of Böhning’s bound contains the same scalar-identity component. The difference lies in the third term. The off-diagonal terms in Böhning’s bound translate to subtracting off the base alternative. This ensures that the base alternative is normalized to 0 and that we can estimate the parameters in the space of $J - 1$ alternatives.

In contrast, our MM algorithm uses the space of J alternatives and ex-post normalization. Alternatively, we could impose the normalization by subtracting the value of the base options at each iteration for both the regressors and our pseudo-data. Doing this yields Böhning’s pseudo-data. Because this base-differencing creates correlation across observations, it requires estimating the maximization step via GLS rather than OLS, where the weight matrix is exactly Böhning’s. Thus, applying this $J - 1$ differencing and GLS transformation to our scalar surrogate mathematically produces Böhning’s exact update. However, this means our baseline OLS algorithm and Böhning’s algorithm take different mathematical paths in the general case. When a model includes shared parameters, estimating the unconstrained OLS surrogate in J dimensions and estimating the differenced GLS surrogate in $J - 1$ dimensions will yield slightly different intermediate step trajectories, as the full-space OLS surrogate penalizes absolute movement along shared

utility components.

This geometric connection reveals a powerful equivalence: for models containing only alternative-specific coefficients and fixed effects, our full-space OLS algorithm and Böhning’s GLS algorithm yield algebraically identical relative parameter trajectories (See Appendix A.2 for mathematical proof). When parameters are strictly alternative-specific, the regressors are identical across all alternatives. Under this condition, a standard property of Seemingly Unrelated Regressions (SUR) applies: the GLS weight matrix analytically cancels out, causing Böhning’s differenced system to collapse mathematically into equation-by-equation OLS. Consequently, the relative updates produced by our unconstrained scalar MM algorithm exactly match Böhning’s updates at every iteration. Consequently, for this broad class of models, our unconstrained scalar MM algorithm achieves the exact same optimal, curvature-adjusted path as Böhning’s method, but does so natively, avoiding the need to base-difference the data or apply GLS whitening transformations

5 The Algorithms

5.1 MM Algorithm

Our MM algorithm starts from an initial guess $\boldsymbol{\theta}^{(0)}$. From this initial guess, each iteration follows a two-step procedure. In the minorization step, we compute $v_{ijt} = x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)}))$. In the maximization step, we update the parameters by linearly regressing v_{ijt} on x_{it} , w_{jt} , and household-choice fixed effects. These steps repeat until convergence.

For our base algorithm, we use the dummy variables approach to handle fixed effects. Let d_{ij} be an $I \times J$ set of dummy variables indicating individual i and alternative j ; d_i be the set of dummy variables indicating individual i ; and d_j be the set of dummy variables indicating alternative j . We interact the regressors x_{it} and w_{jt} with the appropriate dummy variables and combine with d_{ij} into a single vector $z_{ijt} = [x_{it} \otimes d_j, w_{jt} \otimes d_i, d_{ij}]$. We then stack the observations to form the

matrix $\mathbf{Z}$. We write out the steps in Algorithm 1.

Algorithm 1 (MM Algorithm)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} \ j=1,\dots,J$

repeat

Minorization Step:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\boldsymbol{\theta}^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,I; j=1,\dots,J; t=1,\dots,T}$$

Maximization Step: (Update $\boldsymbol{\theta}^k$ by OLS)

$$\boldsymbol{\theta}^{(k+1)} \leftarrow (\mathbf{Z}'\mathbf{Z})^{-1} \mathbf{Z}'\mathbf{v}^{(k)}$$

$$k \leftarrow k + 1$$

until convergence

In the algorithm, we did not specify a convergence criterion. The MM algorithm accommodates a number of different convergence criteria. The researcher should use the criterion that makes sense with their application. A natural choice is the difference in the successive values of the parameters $\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\|$. Another possible choice is the difference in successive values of the log-likelihood $|\mathcal{L}(\boldsymbol{\theta}^{(k+1)}) - \mathcal{L}(\boldsymbol{\theta}^{(k)})|$. This criterion requires an additional computational step since the MM algorithm does not compute the likelihood.

We can improve the speed of the algorithm with some minor adjustments. Because the independent variables do not change from iteration to iteration, we need to compute the pseudo-inverse $(\mathbf{Z}'\mathbf{Z})^{-1} \mathbf{Z}'$ only once at the beginning. This converts the maximization step into simple matrix multiplication. The best way to compute the pseudo-inverse will depend on the problem. When the number of regressors is small, computing the pseudo-inverse with matrix multiplication and standard inversion will often be the fastest approach, but also the least stable. Where stability is needed, QR decomposition strikes a good balance between stability and speed.

Reuse of the linear index can also improve the algorithm's speed. The linear index $x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)}$ appears both in the expression for the minorization and in the logit probabilities. So we need to compute this term only once per minorization step, resulting in small savings.

5.2 Demeaning

We now turn to handling fixed effects. If some of the fixed effects have many categories, updating the parameters using OLS with dummy variables is inefficient or even infeasible. Our MM algorithms accommodate two approaches to handling fixed effects. The first approach is to use standard techniques for handling fixed effects in linear models such as demeaning. The second approach is to transform the maximization step into a conditional maximization step, updating the fixed coefficients and fixed effects sequentially.

5.2.1 One-Way Fixed Effects

As the model is linear at each iteration, we can make use of the Frisch-Waugh-Lovell theorem to absorb the fixed effects. Partialling out the fixed effects greatly reduces the dimensionality of the problem and hence the computational cost of the maximization step.

In the case of one-way fixed effects, we can residualize the regressors by demeaning the variables. In the maximization step, we first recover a subset of the parameters $\boldsymbol{\eta} = \{\{\beta_j\}_{j=1,\dots,J}, \{\gamma_i\}_{i=1,\dots,I}\}$ using demeaned OLS. The asymptotic runtime of OLS helps us understand why demeaned OLS can lead to large speed improvements. OLS has a complexity of $\mathcal{O}(NK^2 + K^3)$ where N is the number of observations and K is the number of regressors (i.e., the dimensions of x_{it} and w_{jt}).⁶ Because runtime scales superlinearly with the number of regressors, absorbing the fixed effects has a major impact on the speed of the algorithm relative to the dummy variables approach, which introduces $I \times J$ additional regressors. This suggests that our method will be most effective when the number of additional covariates is small and the number of fixed effects is large.

After recovering $\boldsymbol{\eta}$, we then recover the fixed effects in a separate, additional step. The computational cost of this additional step is small relative to the gain from

⁶The Big-O calculation assumes schoolbook matrix multiplication for all multiplications and Gauss-Jordan elimination for computing the inverse. Other methods will have different complexities. QR decomposition, for example has a complexity of $\mathcal{O}(NK^2)$.

using partitioned regression.

For ease of notation, we combine the variables x_{jt} and w_{it} into a single matrix and stack observations to form the matrix $\mathbf{Z}$. We denote the demeaned variables with a tilde, such as $\tilde{\mathbf{Z}}$.

Using partitioned regression, our MM algorithm becomes:

Algorithm 2 (MM Algorithm - Fixed Effects)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1, \dots, I, j=1, \dots, J}$

repeat

Minorization Step:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1, \dots, N; j=1, \dots, J; t=1, \dots, T}$$

$$\tilde{\mathbf{v}}^{(k)} \leftarrow \text{Demean } \mathbf{v}^{(k)}$$

Maximization Step:

1. Update $\boldsymbol{\eta}^{(k)}$ by demeaned OLS:

$$\boldsymbol{\eta}^{(k+1)} \leftarrow \left(\tilde{\mathbf{Z}}' \tilde{\mathbf{Z}} \right)^{-1} \tilde{\mathbf{Z}}' \tilde{\mathbf{v}}^{(k)}$$

2. Update the fixed effects: $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha}^{(k+1)} \leftarrow \mathbf{v}^{(k)} - \mathbf{Z}\boldsymbol{\eta}^{(k+1)} - \left(\tilde{\mathbf{v}}^{(k)} - \tilde{\mathbf{Z}}\boldsymbol{\eta}^{(k+1)} \right)$$

$$k \leftarrow k + 1$$

until convergence

Our update treats the partialled out parameters like nuisance parameters. It recovers the contribution of these parameters. For one-way fixed effects, this alternative update returns a vector with a length equal to the number of observations N . This vector repeats α_{ij} for each time period. The advantage of this update step is that it extends naturally to multi-way fixed effects and many individual-specific coefficients.

An alternative way to update the fixed effects is to take the average of the residuals:

$$\alpha_{ij}^{(k+1)} = \frac{1}{T} \sum_t v_{ijt}^{(k)} - z_{ijt} \boldsymbol{\eta}^{(k+1)} \quad (11)$$

This update is often simpler but does not extend to more complicated problems with multiple nuisance parameters.

Note that the independent variables $\mathbf{Z}$ do not change from iteration to iteration. To avoid repetition, we can then pull out the pseudo-inverse $(\tilde{\mathbf{Z}}' \tilde{\mathbf{Z}})^{-1} \tilde{\mathbf{Z}}'$ and save it for later iterations, transforming the maximization step into a simple matrix multiplication.

5.2.2 Multi-Way Fixed Effects

The above algorithms also apply to the case of multi-way fixed effects with minor modification. The algorithms can accommodate both individual-alternative fixed effects and alternative-time fixed effects. For example, utility can have the form:

$$u_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij} + \tau_{jt} + \varepsilon_{ijt}, \quad (12)$$

where τ_{jt} are the alternative-time fixed effects. The other variables are the same as in (1).⁷

The main difference with multi-way fixed effects is the method for residualizing the regressors. With multi-way fixed effects, the annihilator matrix for projecting out the fixed effects is not equivalent to demeaning and beyond two fixed effects, directly computing the annihilator matrix is normally impossible. A number of papers provide methods for absorbing multi-way fixed effects in linear models. Notably, Guimaraes and Portugal (2010), Gaure (2013), and Correia (2017) provide iterative estimators based on the method of alternating projections. Fong and Saunders (2011) provide an alternative estimator for solving sparse linear systems more generally. Finally, Somaini and Wolak (2016) provide a method for computing the annihilator matrix in the case of two-way fixed effects. These methods are available in many commonly-used statistical software.⁸ Researchers can use any of

⁷Because the alternative-time fixed effects absorb the main effect of the characteristic w_{jt} , a location normalization on the slopes such as $\gamma_1 = 0$ is needed.

⁸All of the methods mentioned are available in the python package PyHDFE. Gaure (2013) is also available in the R packages lfe and collapse, and Correia (2017) is available in the Stata package reghdfe.

these methods to residualize the regressors in the maximization step.

In the case of multi-way fixed effects, the update recovers the sum of the fixed effects $\alpha_{ij} + \tau_{jt}$ rather than the actual coefficients. So in the maximization step, we are updating the sum as:

$$[\alpha_{ij} + \tau_{jt}]^{(k+1)} = v_{ijt}^{(k)} - z_{ijt}\hat{\eta}^{(k+1)} - \left(\tilde{v}_{ijt}^{(k)} - \tilde{z}_{ijt}\hat{\eta}^{(k+1)}\right) \quad (13)$$

Recovering the coefficients involves solving the system defined by:

$$\alpha_{ij} + \tau_{jt} = v_{ijt} - z_{ijt}\hat{\eta} - (\tilde{v}_{ijt} - \tilde{z}_{ijt}\hat{\eta}) \quad (14)$$

where $\hat{\eta}$ is the final estimates. Uniquely solving for the separate coefficients requires a normalization. With high-dimensional fixed effects, this system will be too large to solve directly and must be solved using numerical routines. Gaure (2013) and Stammann (2018) describes how to solve this particular system using the Kaczmarz method (Kaczmarz, 1993).⁹ Solving for the coefficients ex-post will reduce some of the computational gains from our MM algorithm.

5.3 Conditional Maximization

An alternative approach to handling the fixed effects that avoids demeaning is to rewrite the maximization step as two conditional maximization steps. In the case of one-way fixed effects, we first maximize the surrogate with respect to $\boldsymbol{\eta}$ conditional on the fixed effects $\boldsymbol{\alpha}$ and then maximize the surrogate function with respect to $\boldsymbol{\alpha}$ conditional on the updated value of $\boldsymbol{\eta}$.

$$\boldsymbol{\eta}^{(k+1)} = \arg \max_{\boldsymbol{\eta}} S(\boldsymbol{\eta}, \boldsymbol{\alpha}^{(k)}; \boldsymbol{\theta}^{(k)}) \quad (15)$$

$$\boldsymbol{\alpha}^{(k+1)} = \arg \max_{\boldsymbol{\alpha}} S(\boldsymbol{\eta}^{(k+1)}, \boldsymbol{\alpha}; \boldsymbol{\theta}^{(k)}) \quad (16)$$

To find the update, we simply need to solve the first-order conditions one at a time. The first-order conditions have simple closed-form solutions. They yield the

⁹This is the English translation of Kaczmarz's original 1937 paper in German.

least squares solution with the parameter that is fixed substituted into the model as a constant. While we are not fully maximizing the surrogate function, we are improving so we preserve the ascent property of the MM algorithm. We have:

$$\mathcal{L}(\boldsymbol{\theta}^{(k)}) = S(\boldsymbol{\theta}^{(k)}; \boldsymbol{\theta}^{(k)}) \leq S(\boldsymbol{\eta}^{(k+1)}, \boldsymbol{\alpha}^{(k)}; \boldsymbol{\theta}^{(k)}) \leq S(\boldsymbol{\eta}^{(k+1)}, \boldsymbol{\alpha}^{(k+1)}; \boldsymbol{\theta}^{(k)}) \leq \mathcal{L}(\boldsymbol{\theta}^{(k+1)}) \quad (17)$$

The first equality and the last inequality follow from the fact that S is a transfer minorization. The middle inequalities hold from the conditional maximization process. As we do not alter the minorization, the gradient alignment condition still holds (1). Provided the regressors exhibit sufficient variation (as detailed in Appendix A) to ensure full rank, each block’s design matrix has full column rank. Under this condition, the negative quadratic surrogate ensures each conditional least-squares problem is strictly concave and yields a unique update. Consequently, provided standard regularity conditions hold to ensure a finite maximum likelihood estimator exists, convergence to a local if not global maximizer is guaranteed by standard block-coordinate ascent and Block-MM theorems (e.g., Tseng, 2001; Razaviyayn, Hong, and Luo, 2013).

Conditional maximization yields the following minorization-conditional-maximization (MCM) algorithm.¹⁰

¹⁰We choose to use terminology along the lines of Chauveau and Hunter (2013) to emphasize that this is a generalization of the ECM approach of Meng and Rubin (1993). In the optimization literature, these types of methods are often referred to as Block-MM algorithms (Li et al., 2026).

Algorithm 3 (MCM Algorithm)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} j=1,\dots,J$

repeat

Minorization Step:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\boldsymbol{\theta}^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,I; j=1,\dots,J; t=1,\dots,T}$$

Conditional Maximization Steps:

1. Update $\boldsymbol{\eta}^{(k)}$ by OLS:

$$\boldsymbol{\eta}^{(k+1)} \leftarrow (\mathbf{Z}'\mathbf{Z})^{-1} \mathbf{Z}'(\mathbf{v}^{(k)} - \boldsymbol{\alpha}^{(k)})$$

2. Update the fixed effects: $\boldsymbol{\alpha}$

$$\alpha_{ij}^{(k+1)} = \frac{1}{T} \sum_t v_{ijt}^{(k)} - z_{ijt} \boldsymbol{\eta}^{(k+1)}$$

$k \leftarrow k + 1$

until convergence

The real difference between the MM and the MCM versions is in the update of $\boldsymbol{\eta}^{(k)}$. The MM algorithm uses the demeaned versions of all of the variables and fully accounts the fixed effects, whereas the MCM algorithm relies only on “demeaned” pseudo-data.

While updating the parameters in blocks simplifies the maximization, it comes at the cost of requiring more iterations. The MCM algorithm effectively ignores the information contained in cross-terms. When blocks are highly correlated and the cross-terms are important, the MCM will exhibit slow convergence as the MCM update will be a poor approximation to the full solution. Whether the simpler updates outweigh the slower convergence will depend on the problem.

5.4 Handling Individual-Specific Coefficients

The dimension of the individual-specific coefficients γ_i can also pose computational problems. Updating these parameters using OLS with interactions between regressors and dummies is inefficient and at worst infeasible. Just as we did with the fixed effects, we can solve this problem by using demeaning or by turning to

conditional maximization.

5.4.1 Demeaning

Partialling out the alternative-specific variables reduces the dimensionality of the problem and offers significant speed advantages. To partial out the alternative-specific variables, we need to use a generalized demeaning algorithm that accommodates heterogeneous slopes, such as Bergé, Butts, and McDermott (2026). Projecting out the individual-specific variables in addition to the fixed effects requires only a slight modification to algorithm 2. As with the fixed effects, demeaning affects the maximization step, not the minorization step. In the maximization step, we first recover the parameters $\beta = (\{\beta_j\}_{j=1,\dots,J})$ using demeaned OLS. After recovering β , we then recover the contribution of alternative-specific variables and the fixed effects in an additional step. The estimation procedure can be found in Algorithm 4.

Algorithm 4 (Fixed Effects and Individual-Specific Coefficients)

Require: initial guess $\theta^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} \ j=1,\dots,J$

repeat

Minorization Step:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\psi_{it}^{(k)}; \mathbf{y}_{it})$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,N; j=1,\dots,J; t=1,\dots,T}$$

$$\tilde{\mathbf{v}}^{(k)} \leftarrow \text{Demean } \mathbf{v}^{(k)}$$

Maximization Step:

1. Update $\beta^{(k)}$ by demeaned OLS:

$$\beta^{(k+1)} \leftarrow (\tilde{\mathbf{X}}' \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}' \tilde{\mathbf{v}}^{(k)}$$

2. Update the contribution $\mathbf{w}\gamma + \alpha$:

$$\mathbf{w}\gamma^{(k+1)} + \alpha^{(k+1)} \leftarrow \mathbf{v}^{(k)} - \mathbf{X}\beta^{(k+1)} - (\tilde{\mathbf{v}}^{(k)} - \tilde{\mathbf{X}}\beta^{(k+1)})$$

$$k \leftarrow k + 1$$

until convergence

As with multi-way fixed effects, our algorithm recovers the overall contribution of alternative-specific variables and the fixed effects rather than the coefficients. It is

straightforward to solve the first-order conditions to obtain individual parameter values as we do in the MCM algorithm.

5.4.2 Conditional Maximization

Just as we did with fixed effects, we can replace the maximization step by a series of conditional maximization steps.¹¹ We separate the parameters into following three blocks: β , γ , and α . This simplifies the update as we can ignore the cross-dependencies.

Alternatively, we could separate the parameters into two blocks, one with β and the other with (γ, α) . Under this partitioning of the parameters, we need to jointly solve the first-order conditions for γ and α , which involves solving a large system of equations.

We update each set of parameters solving the first-order conditions one at a time giving the following algorithm:

¹¹We thank an anonymous referee for this helpful point.

Algorithm 5 (MCM - Fixed Effects and Individual-Specific Coefficients)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} j=1,\dots,J$

repeat

Minorization Step:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it})$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,N; j=1,\dots,J; t=1,\dots,T}$$

Conditional Maximization Steps:

1. Update $\boldsymbol{\beta}^{(k)}$ by OLS:

$$\boldsymbol{\beta}^{(k+1)} \leftarrow (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'(\mathbf{v}^{(k)} - \mathbf{W}\boldsymbol{\gamma}^{(k)} - \boldsymbol{\alpha}^{(k)})$$

2. Update $\boldsymbol{\gamma}$ as:

$$\boldsymbol{\gamma}^{(k+1)} = (\mathbf{W}'\mathbf{W})^{-1} \mathbf{W}'(\mathbf{v}^{(k)} - \mathbf{X}\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\alpha}^{(k)})$$

3. Update $\boldsymbol{\alpha}$ as:

$$\alpha_{ij}^{(k+1)} = \frac{1}{T} \sum_t v_{ijt}^{(k)} - x_{it}\beta_j^{(k+1)} - w_{jt}\gamma_i^{(k+1)}$$

$$k \leftarrow k + 1$$

until convergence

As written, the update for $\boldsymbol{\gamma}$ is inefficient. $\mathbf{W}$ is the stacked vectors w_{jt} interacted with dummies variables for individuals. Let $\mathbf{W}_i$ be the submatrices of $\mathbf{W}$, the stacked w_{jt} corresponding to individual i 's choice set. We can update $\boldsymbol{\gamma}$ as:

$$\boldsymbol{\gamma}_i^{(k+1)} = (\mathbf{W}_i' \mathbf{W}_i)^{-1} \mathbf{W}_i' (\mathbf{v}_i^{(k)} - \mathbf{X}_i \boldsymbol{\beta}^{(k+1)} - \boldsymbol{\alpha}_i^{(k)}) \quad (18)$$

The term $(\mathbf{W}_i' \mathbf{W}_i)^{-1} \mathbf{W}_i'$ does not vary across iterations so we can compute this once at the beginning. When choice sets do not vary across individuals, $\mathbf{W}_i$ is fixed across all individuals and we just need to compute the pseudo-inverse for one individual. In the case where w_{jt} is a scalar, the update formula simplifies to:

$$\gamma_i^{(k+1)} = \frac{\sum_{jt} (v_{ijt}^{(k)} - x_{it}\beta_j^{(k+1)} - \alpha_{ij}^{(k)}) w_{jt}}{\sum_{jt} w_{jt}^2} \quad (19)$$

As before, the MCM algorithm avoids the need to demean, but can take more iterations to converge. The tradeoff between the full maximization solution computed via demeaning or the MCM algorithm depends on the problem. Where demeaning is most expensive and cross-terms are not important, conditional maximization

offers greater benefits.

We can also use conditional maximization to handle individual-specific coefficients in Böhning’s MM algorithm. The algorithm is similar to our MM algorithm, but we need to whiten the fixed effects. We provide pseudo-code in the appendix.

5.5 Böhning’s MM algorithm

We provide analogous algorithms for the MM algorithm of Böhning (1992) in Appendix B.3. While Böhning (1992) derives his update as an explicit mathematical map from one iterate to the next, we demonstrated how to recast his solution as a sequence of regressions. For one-way fixed effects, the algorithm for Böhning is very similar to the MM algorithm we present above, but the partitioned regression uses the demeaned whitened data. In the case of conditional maximization, incorporating the whitening element of Böhning’s algorithm is less straightforward. That is the case for individual fixed effects as well.

The main difficulty with handling individual-specific coefficients in Böhning’s MM algorithm is that updates recover the whitened version of the contribution of the nuisance parameters. To compute the linear index for the next iteration, we then need to unwhiten the contribution, making Böhning’s MM algorithm more cumbersome to implement.

5.6 Acceleration Methods

The low cost per iteration of the MM algorithm relative to gradient approaches comes at the expense of more iterations. Many MM algorithms can exhibit slow convergence close to the maximum (Zhou, Alexander, and Lange, 2011). In these cases, acceleration methods can be used to improve the rate of convergence.

Acceleration methods use additional information to improve the parameter updates. There are many acceleration methods for MM and fixed-point algorithms.

We highlight SQUAREM (Varadhan and Roland, 2008), an iterative method well-suited to our algorithms.

SQUAREM approaches the MM algorithm from the perspective of a fixed point equation.¹² We can view the MM algorithm as a mapping F from the current iterate $\boldsymbol{\theta}^{(k)}$ to the next iterate $\boldsymbol{\theta}^{(k+1)}$, $\boldsymbol{\theta}^{(k+1)} = F(\boldsymbol{\theta}^{(k)})$. In this formulation, the goal of the MM algorithm is to find the fixed point $\theta^* = F(\theta^*)$, or equivalently, to find the root of the function $G(\boldsymbol{\theta}) = \boldsymbol{\theta} - F(\boldsymbol{\theta})$. This can be done with Newton’s method updating the parameters as:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \nabla G(\boldsymbol{\theta}^{(k)})^{-1} G(\boldsymbol{\theta}^{(k)}) \quad (20)$$

Rather than compute $\nabla G(\boldsymbol{\theta}^{(k)})^{-1}$ directly, SQUAREM relies on an approximation.¹³

Before giving the update formula, we first define the quantities $u_k = F(\boldsymbol{\theta}^{(k)}) - \boldsymbol{\theta}^{(k)}$, and $w_k = F \circ F(\boldsymbol{\theta}^{(k)}) - F(\boldsymbol{\theta}^{(k)})$. These quantities are differences in the parameter values between successive steps of the MM algorithm. With these defined, the SQUAREM update formula is:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - 2s_k u_k + s_k^2 (w_k - u_k) \quad (21)$$

where $s_k = -\frac{\|u_k\|}{\|w_k - u_k\|}$.¹⁴

As the update formula shows, a single step of each acceleration method really consists of three steps. The method first performs two steps of the MM algorithm, saving the parameter values at each step. Rather than taking another step of the MM algorithm, SQUAREM extrapolates the next value from the two MM iterations. By using information from two MM steps, the method can produce larger steps or better search directions.

¹²This discussion is close to that in Zhou, Alexander, and Lange (2011).

¹³We also experimented with the acceleration method of Zhou, Alexander, and Lange (2011), which uses a different approximation than SQUAREM. We found that this method performed similarly to SQUAREM, although slightly worse. Thus, we focus on SQUAREM in this paper.

¹⁴There are three variants of SQUAREM that calculate the step size s_k in different ways (Varadhan and Roland, 2008) We focus on the third variant, which generally performs the best (Du and Varadhan, 2020).

While SQUAREM can result in noticeable speed improvements over the base algorithm, it breaks the ascent property of the MM algorithm. As a result, this method will not necessarily give parameter values that increase the likelihood at every iteration. In this case, the researcher can fall back to the second MM update $F \circ F(\boldsymbol{\theta}^{(k)})$ to ensure reliability. The extrapolation can similarly disrupt the convergence of the other MM-based algorithms. Without safeguards, the extrapolation step may cause the algorithm to diverge or oscillate endlessly.

5.7 Bias Correction

Our focus so far has been on obtaining point estimates. However, estimates from the MM algorithm for nonlinear panel data models may still be biased, as with any maximum likelihood estimator, due to the incidental parameters problem. To reduce the bias, we recommend that researchers use split-panel jackknife estimation following Dhaene and Jochmans (2015). The split-panel jackknife removes the leading bias and reduces the bias from higher-order terms.

Split-panel jackknife estimation involves splitting the dataset into two subpanels. In the simplest case, the first subpanel runs from time $t = 0$ to $t = \frac{T}{2}$ and the second runs from $t = \frac{T}{2} + 1$ to $t = T$. We use the MM algorithm to obtain an estimate for the full sample $\hat{\boldsymbol{\theta}}$, an estimate for the first subpanel $\hat{\boldsymbol{\theta}}_1$, and an estimate for the second subpanel $\hat{\boldsymbol{\theta}}_2$. The split-panel jackknife estimator $\tilde{\boldsymbol{\theta}}$ is then $\tilde{\boldsymbol{\theta}} = 2\hat{\boldsymbol{\theta}} - \frac{1}{2}(\hat{\boldsymbol{\theta}}_1 + \hat{\boldsymbol{\theta}}_2)$.

6 Inference

Although our focus has been on how to generate point estimates rapidly and reduce incidental parameter bias, researchers also require reliable standard errors. For estimation, the key insight of our algorithms is that translating the multinomial logit into a sequence of least-squares regressions completely avoids the need to invert massive, intractable parameter matrices. As we show below, the computational

benefits of this linearization extend directly to inference as well.

6.1 The Hessian and the Schur Complement

In a model with common parameters β and a high-dimensional vector of nuisance parameters $\xi = (\alpha, \gamma)$, writing the Hessian in a block-partitioned form is helpful:

$$\mathcal{H} = \begin{bmatrix} \mathbf{H}_{\beta\beta} & \mathbf{H}_{\beta\xi} \\ \mathbf{H}'_{\beta\xi} & \mathbf{H}_{\xi\xi} \end{bmatrix} \quad (22)$$

where $\mathbf{H}_{\beta\beta}$ corresponds to the curvature over β , $\mathbf{H}_{\xi\xi}$ corresponds to the curvature over the fixed effects ξ , and $\mathbf{H}_{\beta\xi}$ captures the cross-derivatives.

The covariance matrix of the maximum likelihood estimators is the negative inverse of the Hessian evaluated at the optimum $\text{Var}(\hat{\beta}) = -H^{-1}$. To conduct inference on our parameters of interest β , we require the top-left block of the inverse of this massive information matrix. From block matrix inversion, this variance block evaluates to the inverse of the Schur complement:

$$\text{Var}(\hat{\beta}) = - \left(\mathbf{H}_{\beta\beta} - \mathbf{H}_{\beta\xi} \mathbf{H}_{\xi\xi}^{-1} \mathbf{H}'_{\beta\xi} \right)^{-1} \quad (23)$$

In standard multinomial logit software, evaluating this variance requires constructing and inverting the high-dimensional nuisance parameter block $\mathbf{H}_{\xi\xi}$. With one-way fixed effects, $\mathbf{H}_{\xi\xi}$ has a block diagonal structure that makes direct inversion computationally convenient. Multi-way fixed effects destroys this block-independence, rendering direct inversion of $\mathbf{H}_{\xi\xi}$ computationally difficult for large datasets. However, we can bypass this inversion entirely by recognizing the geometric relationship between Newton's method and Iteratively Reweighted Least Squares (IRLS) method not unlike our MM algorithm.

6.2 Connection to Newton's Method

While the MM algorithms discussed earlier use constant matrix bounds to avoid re-weighting the data at every step, a direct second-order approximation reveals the exact curvature of the log-likelihood. Operating in the $J - 1$ dimensional space normalized against the base alternative, the second-order Taylor expansion of the log-likelihood in $\boldsymbol{\psi}$ around the current linear predictor $\boldsymbol{\psi}^{(k)}$ is:

$$\begin{aligned}\tilde{S}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) &= \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \sum_{i,t} \mathbf{h}(\boldsymbol{\psi}_{it}^{(k)})' (\boldsymbol{\psi}_{it} - \boldsymbol{\psi}_{it}^{(k)}) \\ &\quad - \frac{1}{2} \sum_{i,t} \left(\boldsymbol{\psi}_{it} - \boldsymbol{\psi}_{it}^{(k)} \right)' \boldsymbol{\Omega}_{it} \left(\boldsymbol{\psi}_{it} - \boldsymbol{\psi}_{it}^{(k)} \right)\end{aligned}\tag{24}$$

where $\boldsymbol{\Omega}_{it} = \text{diag}(\mathbf{p}_{it}) - \mathbf{p}_{it}\mathbf{p}_{it}'$ represents the true logit probability weights, and $\mathbf{h}_{it}^{(k)} = \mathbf{y}_{it} - \mathbf{p}_{it}^{(k)}$ is the residual gradient. Maximizing this function is mathematically equivalent to solving the generalized least squares problem:

$$\min_{\boldsymbol{\theta}} \sum_{i,t} \left(\mathbf{v}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right)' \boldsymbol{\Omega}_{it}^{(k)} \left(\mathbf{v}_{it}^{(k)} - \mathbf{Z}_{it}\boldsymbol{\theta} \right)\tag{25}$$

where the pseudo-data is constructed using the inverse logit weights, $\mathbf{v}_{it}^{(k)} = \boldsymbol{\psi}_{it}^{(k)} + \boldsymbol{\Omega}_{it}^{(k)-1} \mathbf{h}(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it})$, and $\mathbf{Z}_{it}$ is the matrix of independent regressors for individual i at time t . Combining the weight matrices and design matrices across the sample, the closed-form solution to this problem is:

$$\boldsymbol{\theta}^{(k+1)} = (\mathbf{Z}'\boldsymbol{\Omega}^{(k)}\mathbf{Z})^{-1} \mathbf{Z}'\boldsymbol{\Omega}^{(k)}\mathbf{v}^{(k)}\tag{26}$$

Substituting the expression for the pseudo-data and simplifying yields the classic Newton-Raphson update:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + (\mathbf{Z}'\boldsymbol{\Omega}^{(k)}\mathbf{Z})^{-1} \mathbf{Z}'(\mathbf{y} - \mathbf{p}^{(k)})\tag{27}$$

This iteratively reweighted least squares (IRLS) algorithm is just the regression formulation of Newton's method. Thus, evaluating the exact inverse of the Hessian is structurally equivalent to executing a weighted linear projection.

In general, because the weight matrix $\boldsymbol{\Omega}^{(k)}$ changes at every iteration, repeatedly

forming and inverting the massive parameter-space matrix $\mathbf{Z}'\boldsymbol{\Omega}^{(k)}\mathbf{Z}$ makes Newton's method computationally costly. We can leverage this exact equivalence one single time at convergence to secure analytical standard errors.¹⁵

6.3 Analytical and Clustered Standard Errors

Once an MM-based algorithm has successfully converged to the MLE, the predicted probabilities $\hat{\mathbf{p}}$ and their corresponding weight matrices $\hat{\boldsymbol{\Omega}}$ are fixed.

By the Frisch-Waugh-Lovell theorem, the Schur complement $\mathbf{H}_{\beta\beta} - \mathbf{H}_{\beta\xi}\mathbf{H}_{\xi\xi}^{-1}\mathbf{H}'_{\beta\xi}$ is algebraically identical to the weighted cross-product matrix of the residualized covariates, which are obtained by projecting the covariates of interest onto the nuisance fixed-effects space using GLS. We therefore do not need to invert the massive, multi-way $\mathbf{H}_{\xi\xi}$ matrix. Instead, we can simply use weighted demeaning.

To obtain the standard errors for β . We start by evaluating the model at the converged MLE to calculate the exact logit weight matrix $\hat{\boldsymbol{\Omega}}$. We can then regress the covariates $\mathbf{X}$ onto the fixed effects space ξ using these true logit weights, and extract the residualized covariates, denoted $\tilde{\mathbf{X}}$. We provide additional details on how to compute the weighted projection in Appendix C. For multi-way fixed effects, this weighted projection can be accomplished using the method of alternating projections. Finally, we can calculate the asymptotic variance matrix of the parameters of interest as $\text{Var}(\hat{\beta}) = (\tilde{\mathbf{X}}'\hat{\boldsymbol{\Omega}}\tilde{\mathbf{X}})^{-1}$.

This framework also yields cluster-robust standard errors at virtually no additional computational cost. By utilizing the residualized covariates $\tilde{\mathbf{x}}_{it}$, we construct the effective observation-level score gradient as $\mathbf{g}_{it} = \tilde{\mathbf{x}}'_{it}(\mathbf{y}_{it} - \hat{\mathbf{p}}_{it})$. Summing these score vectors within clusters (e.g., individuals or regions) provides the target matrix $\mathbf{S} = \sum_c \mathbf{g}_c \mathbf{g}'_c$. The exact multi-way, cluster-robust standard errors are then computed via the standard sandwich estimator, $\mathbf{H}^{-1}\mathbf{S}\mathbf{H}^{-1}$, where $\mathbf{H}^{-1} = (\tilde{\mathbf{X}}'\hat{\boldsymbol{\Omega}}\tilde{\mathbf{X}})^{-1}$.

¹⁵Calculating Newton's method using IRLS is inefficient. For the multinomial logit, it is better to exploit the Hessian's structure following Hospido, 2012 when implementing Newton's method.

While the analytical and cluster-robust standard errors derived above offer significant computational savings, researchers may also rely on traditional bootstrapping to conduct inference. Because our MM algorithms drastically reduce the time required to compute the maximum likelihood estimates, re-estimating the model across hundreds of bootstrap resamples is highly feasible. Bootstrapping is particularly attractive in empirical settings where the multinomial logit model is embedded within a more complex, multi-step estimation procedure.

For researchers interested in transformations of β alone, standard errors can be calculated using the delta method with the covariance matrix derived above. For transformations that also depend on estimated fixed effects, such as marginal effects, the delta method requires either forming the full joint covariance matrix or solving the joint information system directly against the target’s gradient. Because these joint calculations can be computationally demanding, researchers may find it much simpler to compute standard errors for these quantities using the bootstrap approach.

6.4 Bias-Corrected Standard Errors

As proven by Dhaene and Jochmans, 2015, the split-panel jackknife removes the leading $O(1/T)$ bias and when the data environment is stationary across the two subpanels leaves the asymptotic variance of the estimator entirely unchanged. Because the asymptotic variance of the bias-corrected estimate $\tilde{\beta}$ is identical to that of the uncorrected estimator, it is fully valid asymptotically to simply report the exact full-sample analytical or clustered standard errors calculated in the previous section alongside the bias-corrected point estimates. However, if the distribution of covariates changes significantly over time, these variances can diverge. To robustly account for this temporal heterogeneity, researchers can compute standard errors using the combined influence function detailed below.

For researchers wishing to compute asymptotic cluster-robust standard errors that explicitly account for the mechanical covariance between the full panel and the subpanels, we can recycle our residualized covariates to calculate the estimator’s

Influence Functions (IF).

An influence function represents the cluster-level score multiplied by the inverse Information Matrix. Having run the IRLS projection once each for the full sample and the two sub-panels, we construct the influence function for cluster c under model $m \in \{full, 1, 2\}$ as:

$$\mathbf{IF}_{c,m} = \mathbf{I}_m^{-1} \mathbf{g}_{c,m} \quad (28)$$

where $\mathbf{g}_{c,m}$ is the effective score for cluster c in that specific sample. Because the SPJ estimator is a strict linear combination, its joint influence function is identically:

$$\mathbf{IF}_{c,SPJ} = 2\mathbf{IF}_{c,full} - \frac{1}{2}\mathbf{IF}_{c,1} - \frac{1}{2}\mathbf{IF}_{c,2} \quad (29)$$

Note that if an observation belongs only to the first half of the panel, its observation-level score contribution to the second subpanel is zero, and vice versa. However, because individuals typically span both halves of the panel, the cluster-level components $\mathbf{IF}_{c,1}$ and $\mathbf{IF}_{c,2}$ will generally both be non-zero and must be retained for the combined influence function. The asymptotic cluster-robust variance matrix for the bias-corrected estimates is simply the sum of outer products of these combined vectors:

$$\text{Var}_{\text{cluster}}(\tilde{\boldsymbol{\beta}}) = \sum_{c=1}^C \mathbf{IF}_{c,SPJ} \mathbf{IF}_{c,SPJ}' \times \left(\frac{C}{C-1} \frac{N-1}{N-K} \right) \quad (30)$$

7 Performance of the MM Algorithm

In this section, we demonstrate the performance of the MM algorithm relative to existing methods through two simulations. The first simulation considers how our method scales with the number of fixed effects and the second considers how our method scales with the number of individual-specific coefficients. First, we outline some similarities across the methods, which we further explain in the appendix.

7.1 Iterative Ascent Methods

In the simulations, we assess the performance of the MM algorithm against commonly used first and second-order ascent methods. Specifically, we consider Adam (Kingma and Ba, 2014) and Newton’s method. These particular methods are available in existing software implementations of logit models.¹⁶

To compare these methods, it is helpful to recognize that they, along with our least-squares MM updates, all belong to the family of preconditioned gradient ascent methods. Preconditioned gradient ascent modifies the standard optimization step by multiplying the gradient search direction by a preconditioning matrix P_k , which helps correct for the curvature of the parameter space. It updates the parameters iteratively as:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \sigma_k P_k \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)}) \quad (31)$$

To see that our MM algorithm fits this structure, substituting the definition of our pseudo-data into the OLS update yields:

$$\boldsymbol{\theta}^{(k+1)} = (\mathbf{Z}'\mathbf{Z})^{-1} \mathbf{Z}'[\mathbf{Z}\boldsymbol{\theta}^{(k)} + 2(\mathbf{y} - \mathbf{p}^{(k)})] \quad (32)$$

which simplifies precisely to:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + 2(\mathbf{Z}'\mathbf{Z})^{-1} \mathbf{Z}'(\mathbf{y} - \mathbf{p}^{(k)}) \quad (33)$$

Because the term $\mathbf{Z}'(\mathbf{y} - \mathbf{p}^{(k)})$ is exactly the gradient of the multinomial logit log-likelihood, our MM algorithm effectively takes a deterministic score step modified by a constant preconditioner $P_k = (\mathbf{Z}'\mathbf{Z})^{-1}$. Ultimately, these iterative methods differ primarily in what type of information they use to construct this preconditioning matrix P_k and how they set the step size σ_k .

We compare the preconditioning terms and their construction across the different the alternative methods in Appendix E. The core computational innovation of

¹⁶The R package `mlogit` and the Stata commands `cmlogit` use Newton’s method. The recent Python package `torch-choice` described in Du, Kanodia, and Athey (2023) includes a variety of first-order methods, including Adam, which the paper recommends for large-scale problems.

our framework is recognizing that the least-squares regression equivalence unlocks the use of linear panel data techniques. Demeaning allows us to evaluate these potentially massive preconditioned gradient steps without constructing the full multi-way cross-product.

7.2 Fixed Effects

We consider a setting where the number of covariates is small and the number of fixed effects grows with the sample size. In this simulation, I consumers choose from three alternatives, $J = \{1, 2, 3\}$ in $T = 20$ time periods. Consumers choose the alternative that yields the highest utility. Consumer i 's utility from choosing option j in time period t is:

$$u_{ijt} = x_{it}\beta_j + \alpha_{ij} + \varepsilon_{ijt} \quad (34)$$

where x_{it} is a single individual-level characteristic; α_{ij} are individual-alternative fixed effects; and ε_{ijt} is a scalar *i.i.d* idiosyncratic shock distributed according to a Type I Extreme Value distribution. We draw x_{it} and α_{ij} from standard normals and set $\beta = (\beta_1, \beta_2, \beta_3) = (0, 0.5, 1)$. We normalize the utility of the base option to be zero so $\alpha_{i1} = 0$, $\forall i$.

We allow the number of individuals I to vary from 500 to 500,000. Since the model contains individual-alternative fixed effects, the number of fixed effects is twice the number of individuals $2I$. Thus, in our most extreme scenario, we are estimating 1,000,000 fixed effects. For each value of I , we simulate 250 trials. We drop individuals who failed to choose all options at least once as these can create problems for convergence, as would be the case for any fixed effects estimation.

We estimate the model parameters using Adam, Newton's method, our MM algorithm (Algorithm 2), and Böhning's MM algorithm (Algorithm 7). Because this simulation features only alternative-specific coefficients, our MM algorithm and Böhning's MM algorithm take mathematically identical steps in the relative parameter space. The only distinction lies in the implementation: our method uses standard OLS in the full space of J alternatives, whereas Böhning's method re-

quires differencing and whitening the data to perform OLS in the space of $J - 1$ alternatives. Therefore, comparing these two MM algorithms in this specific simulation captures the net computational overhead of Böhning’s MM algorithm, balancing the added cost of the whitening step against its smaller $(J - 1)$ -dimensional regression.

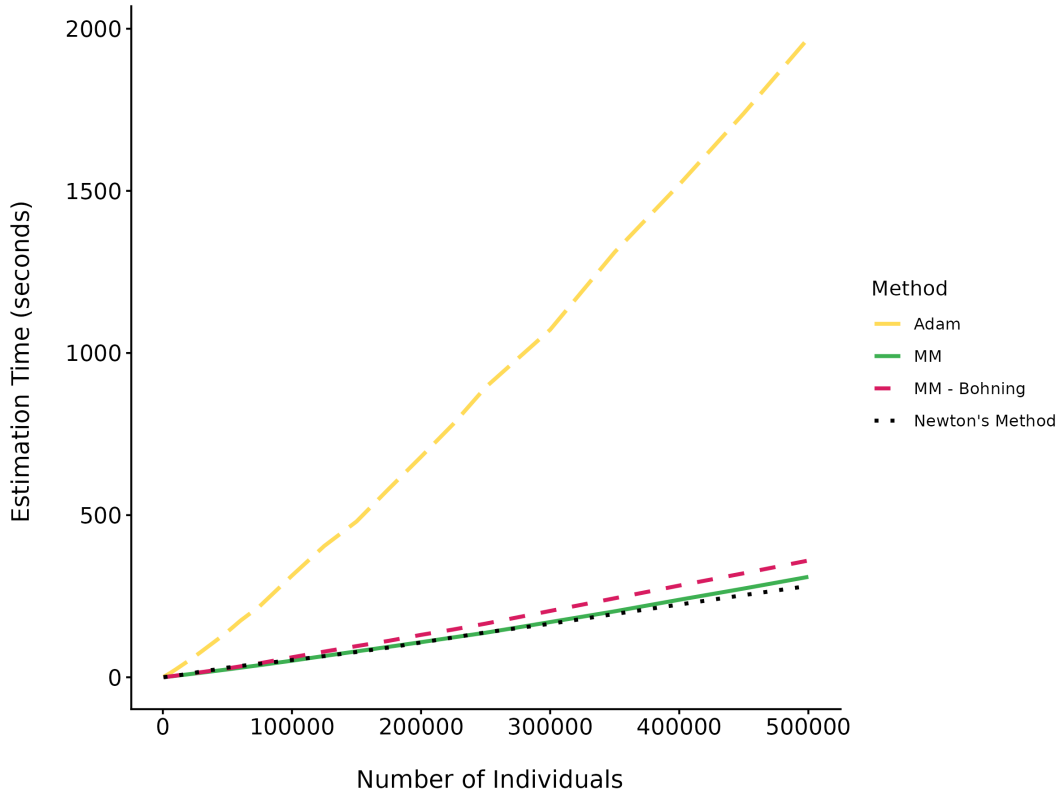
We also estimate SQUAREM accelerated versions of the MM-based algorithms. We consider the performance of the MCM algorithm in Appendix H. Using conditional maximization offers similar performance as full maximization via demeaning. There is not a reason to prefer one approach to the other.

We use the same convergence criterion across all of the methods. In particular, we stop the algorithms when the difference in successive values of the log-likelihood is less than 10^{-8} . The choice of the stopping condition does not have a major effect on the results. In addition, we start all of the algorithms from the same initial guess of $\mathbf{0}$.

Figure 2 shows the runtimes for the unaccelerated methods against the number of individuals I .¹⁷ The figure shows the median runtime for each method. Since we drop individuals who chose an alternative with zero probability, there is some variation in the number of observations across trials.

¹⁷We performed all calculations on a 2.8 GHz Intel Gold 6242 using 8 cores with 8 GB of memory per code.

Figure 2



Notes: The lines show the median runtime.

All of the methods yield roughly the same parameter estimates across runs. The differences between the estimates are less than 10^{-6} and lack any practical significance. The minuscule differences confirm that the MM algorithms are indeed converging to the same estimates as Newton's method and Adam, the maximum likelihood estimates. Thus, the differences in runtimes reflect the performance of the different algorithms.

Newton's method outperforms the unaccelerated algorithms. Newton's method is more than 7 times faster than Adam on average. Newton's method takes only 4 minutes to estimate 1,000,000 fixed effects. In contrast, Adam takes around 30 minutes.¹⁸ The unaccelerated versions of MM-based algorithms are only slightly slower than Newton's method with all of the algorithms having similar performance

¹⁸There may exist a step size (or schedule of step sizes) that allows Adam to perform better. However, the time spent finding this step size would likely far exceed any reduction in computation time.

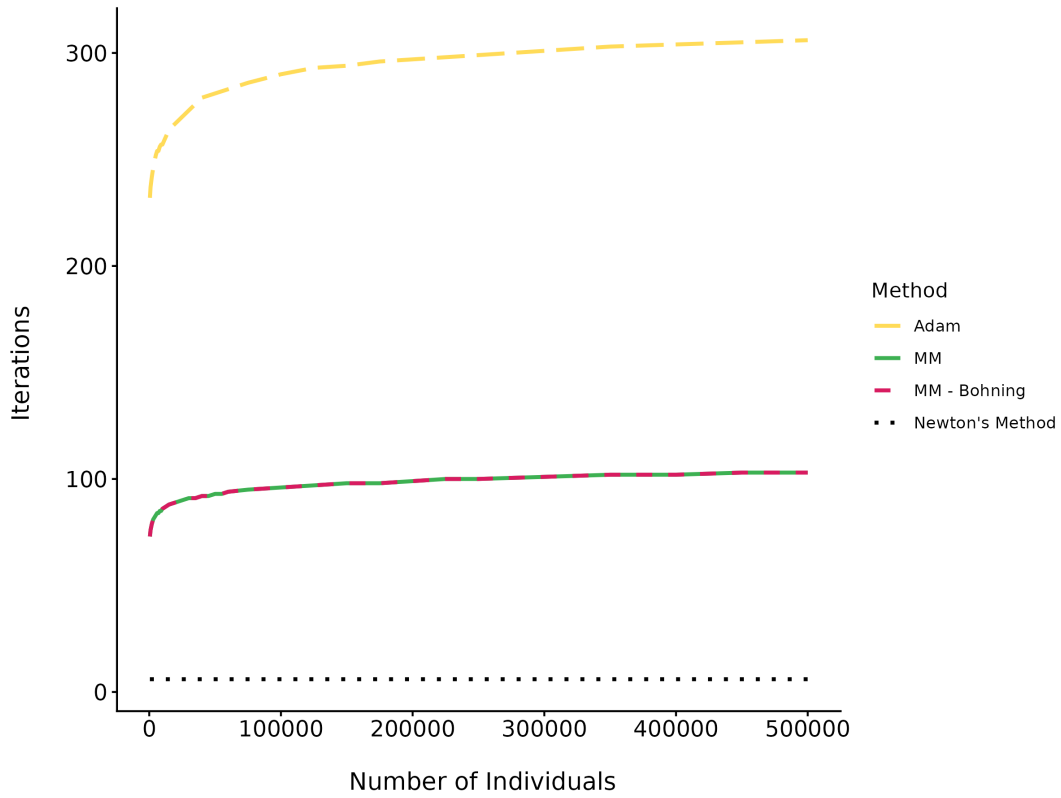
on average.

Figure 2 reveals only a slight difference in runtime between our MM algorithm and Böhning’s MM algorithm. This confirms that the computational overhead introduced by the whitening step is minor. Such a small penalty is expected here, as the uniform choice set across individuals represents the optimal scenario for fast, vectorized whitening.

The number of iterations to convergence provides additional clarity on the performance of the different methods. Figure 3 shows the median number of iterations taken by each method plotted against the number of individuals. As expected from our theoretical discussion, our MM algorithm and Böhning’s MM algorithm require the exact same number of iterations to converge, perfectly confirming their mathematical equivalence for this model specification.

Newton’s method takes the fewest iterations. In this simulation, Newton’s method always converges in 6 iterations. The greater convergence speed of Newton’s method comes at the expense of making each iteration costly to compute. In contrast, the MM algorithms features a low cost per iteration but take many iterations. Both take just around 100 iterations. Finally, Adam takes 3 times the number of iterations as the MM algorithms.

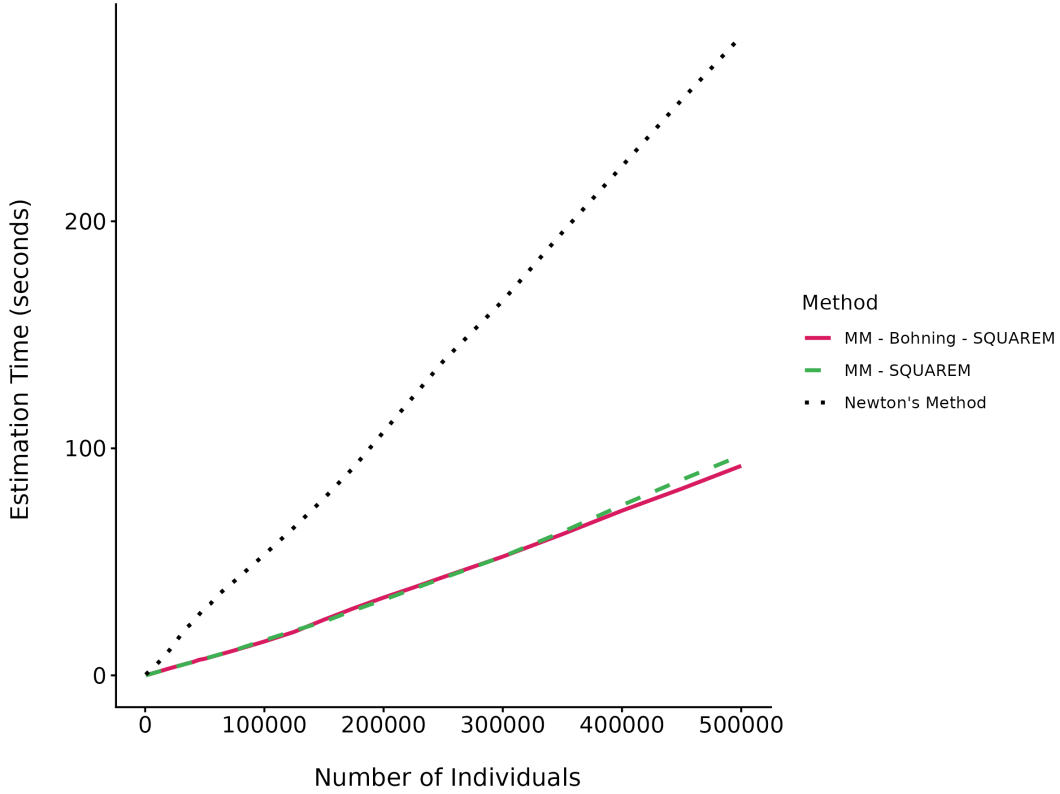
Figure 3



Notes: The lines show the median number of iterations until convergence.

Acceleration using SQUAREM offers additional speed improvements and allows the MM-based algorithms to outperform Newton's method as shown in figure 4. The extrapolation step allows these algorithms to jump ahead, saving iterations. The accelerated versions all show similar performance, beating highly optimized Newton's method by more than a factor of 2. The accelerated versions are capable of estimating 1,000,000 fixed effects in around 1.5 minutes.

Figure 4



Notes: The lines show the median runtime.

Thus far, we have evaluated the methods based solely on their runtimes. However, programming complexity is another important consideration. The extra time spent coding a faster algorithm may outweigh any runtime savings.

Out of the algorithms considered, we found Newton's method to be the most involved to code. For Newton's method to perform well, the calculation of the Hessian needs to be efficient, which means using analytical expressions for the building blocks of the Hessian and avoiding any unnecessary calculations. Simply following the approach to exploiting the sparsity of the Hessian described in Hospido (2012) is not enough for Newton's method to outperform the MM algorithm. In fact, our early implementations performed worse than the base MM algorithm. Adding SQUAREM acceleration to our MM algorithm is far easier and offers additional speed than implementing highly optimized Newton's method.

For models with alternative-specific coefficients, there is no reason to estimate the general case for Böhning’s MM algorithm. Our MM algorithm produces identical steps while avoiding the complications due to the whitening step and the need to subtract off the base option. It is both as fast and simpler to code.

7.3 Individual-Specific Coefficients

We now consider how the MM algorithm scales with the number of individual-specific coefficients. This simulation is similar to the previous one, but we replace the fixed effect with an alternative-specific variable that has individual-specific coefficients. As before, I consumers choose from three alternatives, $J = \{1, 2, 3\}$, but now in $T = 50$ time periods.¹⁹ Consumers choose the alternative that yields the highest utility. Consumer i ’s utility from choosing option j in time period t is now:

$$u_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \varepsilon_{ijt}, \quad (35)$$

where x_{it} is a single individual-level characteristic; w_{jt} is a single alternative-level characteristic; and ε_{ijt} is a scalar *i.i.d* idiosyncratic shock distributed according to a Type I Extreme Value distribution. We draw x_{it} and w_{jt} from standard normals. The individual-specific coefficients γ_i is drawn from a normal $\mathcal{N}(0, 0.5^2)$. As in the previous simulation, we set $\beta = (\beta_1, \beta_2, \beta_3) = (0, 0.5, 1)$.

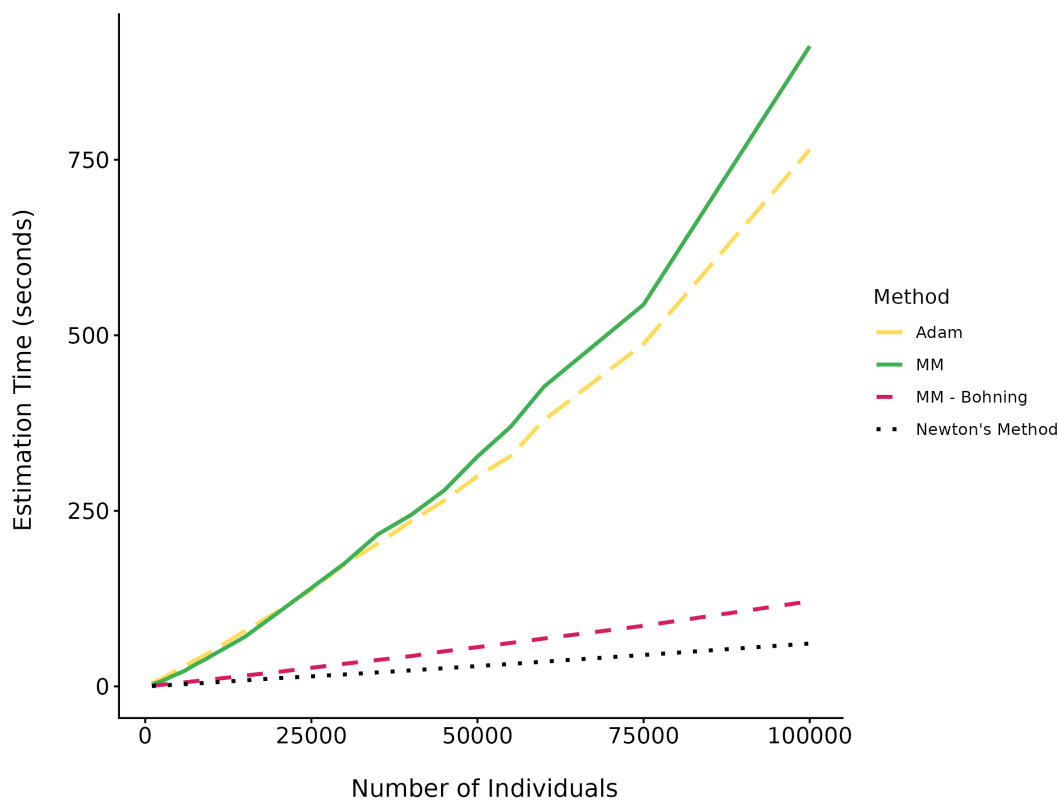
We allow the number of individuals I to vary from 1,000 to 100,000. For each value of I , we simulate 250 trials. As in the previous simulation, we drop individuals who do not choose all of the options at least once.

We estimate the model parameters using the same algorithms as before. Figure 5 displays the median runtimes across the 250 simulation trials. As detailed in Appendix H, the MCM algorithm slightly outperforms the baseline MM algorithm in this setting. This advantage is unsurprising as the demeaning is more expensive, and the parameter blocks are highly independent by construction. We expect

¹⁹We increase the length of the time-series to reduce the number of times that a consumer chooses an alternative no more than one time. These cases can cause convergence problems for all of the algorithms.

that dependence typically occurs in severely unbalanced panels or when covariates correlate strongly with fixed-effect groupings. Under these conditions, full maximization via demeaning may be preferred.

Figure 5



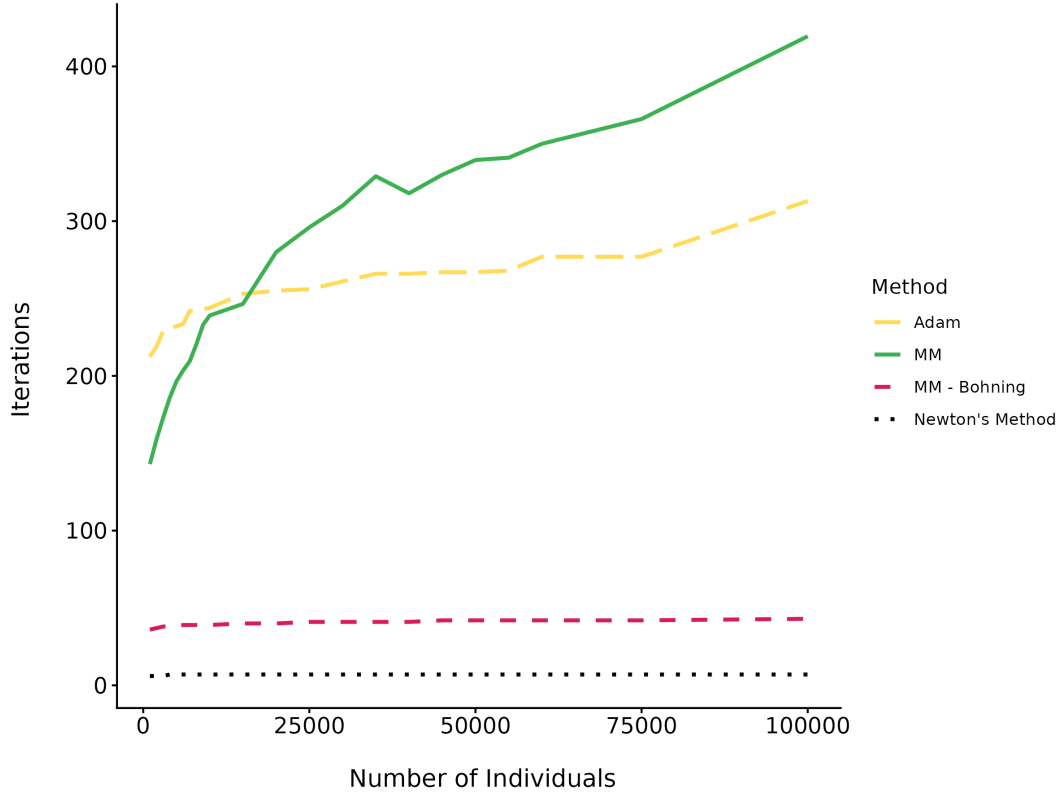
Notes: The line shows the median runtime.

As before, Newton's method outperforms the unaccelerated algorithms and Adam is still relatively slow. Böhning's MM algorithm is only slightly slower than Newton's method. However, in contrast to the previous simulation, our MM algorithm is the slowest algorithm.

The reason for the poor performance of our MM algorithm stems from our scalar-identity bound. With individual-specific coefficients, increasing the value of one coefficient affects all of the alternatives. Taking into account the dependence between alternatives is important for generating better steps. Böhning's bound allows for the dependencies; our scalar-identity bound cannot.

To avoid overshooting, our MM algorithm takes small steps and, as a result, takes more iterations, as we see in Figure 6. Figure 6 shows the number of iterations taken by each method plotted against the number of individuals.

Figure 6



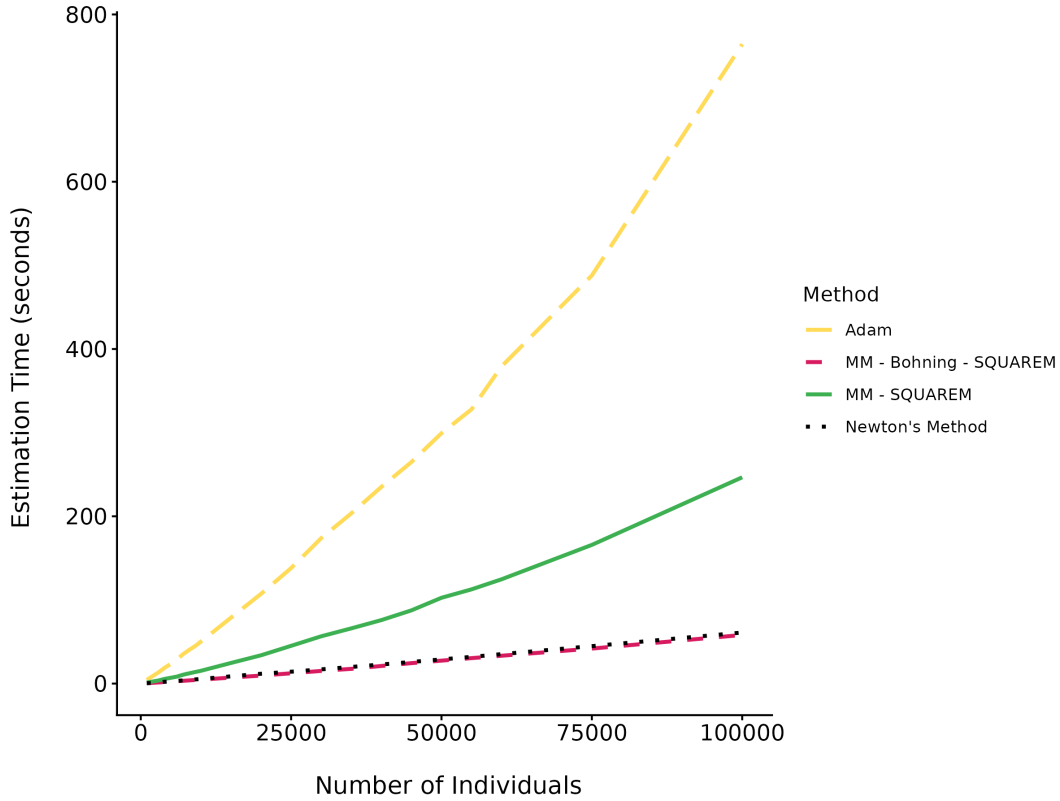
Notes: The lines shows the median number of iterations until convergence.

For the MM algorithm, the number of iterations until convergence is increasing with the number of individuals. With 100,000 individuals, our MM algorithm takes 400 iterations on average far more than even Adam, which takes around 300. By taking into account dependencies between alternatives, Böhning's MM algorithm consistently converges in around 50 iterations. Newton's method still consistently converges in around 6 to 7 iterations.

Acceleration helps to offset the slow convergence of our MM algorithm (See Figure 7). With SQUAREM acceleration, our MM algorithm outperforms Adam, but is still short of Newton's method. Accelerating Böhning's MM algorithm allows it to match Newton's method. Since Böhning's MM algorithm is taking only around 50

iterations to begin with, there is less scope for accelerations to improve performance by saving iterations.

Figure 7



Notes: The lines shows the median runtimes.

While Newton's method is fastest, it also takes significantly longer to program and optimize. For this simulation, we found that the increased programming time far outweighed any runtime savings. Newton's method is faster only when using well-optimized code, optimized for the specific data structure in a way that limits extensibility. When factoring programming time, the accelerated version of our MM algorithm still compares favorably to Newton's method.

When dependencies between alternatives matter more, Böhning's MM algorithm is a good choice. The improved rate of convergence justifies the cost of implementing the whitening step. Given that there are only three alternatives, the dependency is high. For larger choice sets where dependency is weaker, the gap between our algorithm and Böhning's may narrow. However, our MM algorithm still offers

reasonable performance while being simple to implement.

8 Empirical Application

In this section, we further demonstrate the utility of our method by providing an empirical application to hospital choice. Economists frequently rely on discrete choice models to identify the degree of substitution between hospitals and study competition in healthcare markets (e.g. Gaynor, Kleiner, and Vogt, 2013; Gowrisankaran, Nevo, and Town, 2015; Garmon, 2017). When evaluating hospital mergers, antitrust practitioners use multinomial logit models to estimate diversion ratios that measure the level of competition between the merging parties (e.g. Capps et al., 2018). In a hospital setting, these diversions represent the share of patients who would switch from one hospital to another if the first hospital went out of network. Due to the availability of patient-level discharge data, practitioners often use individual-level logit models.

In the logit model, the functional form of the utilities dictates the allowed substitution patterns. Greater flexibility is therefore crucial for recovering realistic substitution patterns. Due to the shortcomings of existing estimation methods, (Raval, Rosenbaum, and Tenn, 2017) provide a semiparametric bin estimator that can capture rich substitution patterns.²⁰ Their method partitions individuals into groups based on a set of G characteristics and a minimum group size. They estimate separate multinomial logit models for each group, with hospital intercepts as the only explanatory variables. Their estimator is equivalent to a multinomial logit with hospital-group fixed effects.

The semiparametric approach of Raval, Rosenbaum, and Tenn (2017) has two main drawbacks. The first is that it can accommodate only discrete variables. Continuous variables, like travel time or the acuity weight, need to be discretized. The second drawback comes from the binning process, which prevents groups from overlapping. This process effectively rules out some forms of multi-way fixed effects.

²⁰The method has seen use in litigated hospital mergers, e.g., Advocate-Northshore (Tenn and Vandergrift, 2017).

These drawbacks could be addressed by estimating on the whole population simultaneously and treating intercepts as hospital-group fixed effects. However, doing so leads to a large number of fixed effects. This leads to the use of the MM algorithm. We allow for multinomial logit models that have continuous variables, many fixed effects, and multi-way fixed effects. The MM algorithm, therefore, expands the set of models that practitioners can consider and gives additional flexibility beyond what the semiparametric method can offer.

8.1 Diversion from Natural Disasters

In order to explore the contribution of our algorithm in discrete choice estimation and Raval, Rosenbaum, and Tenn (2017) in particular, we revisit Raval, Rosenbaum, and Wilson (RRW, 2022), which compares diversion ratios estimated from logit models to observed diversions from hospital closures due to natural disasters. We consider one of the four natural disasters considered in that paper, the Americus Tornado. The Americus Tornado swept through Americus, GA on March 1, 2007, damaging Sumter Regional Hospital. The hospital remained closed for a year, eventually reopening on April 1, 2008. Prior to closure, Sumter hospital accounted 50.4% of patient admissions within its service area (RRW), forcing a large number of patients to divert to surrounding hospitals.

The data consists of all hospital admissions from the state of Georgia in 2006, 2007, and 2008. The data set includes information about each patient such as age, zip code, and gender, among other variables. It also includes information about treatment including the area of diagnosis and length of stay. We do not use the full dataset, but instead use the estimation sample from RRW. This sample consists of admissions within the 90% service area of Sumter hospital, the smallest set of zip codes that accounts for 90% of patient admissions.

Following RRW, we calculate the observed diversions using the difference in shares between the period prior to closure and the period just after closure. The pre-period runs from January 1, 2006 to February 28, 2007 and the post-period runs from March 1, 2007 to March 31, 2008. The observed diversions from Sumter

hospital to hospital k are:

$$D_{observed}^k = \frac{s_k^{post} - s_k^{pre}}{s_{Sumter}^{pre}} \quad (36)$$

where s is the observed share of admissions. We compare the observed diversions to predicted diversions from a logit model estimated on the pre-period data. Given the logit form, the predicted diversions from Sumter to hospital k are:

$$\hat{D}^k = \sum_i \frac{\hat{s}_{ik}}{1 - \hat{s}_{i,Sumter}} \frac{\hat{s}_{i,Sumter}}{\sum_i \hat{s}_{i,Sumter}} \quad (37)$$

where i consists of the pre-period patients and $\hat{s}$ are the predicted shares from the logit model. We examine how well different models perform in predicting the observed diversions.

8.2 Models

We consider four logit models. We take the first three models from RRW. The first model *Semipar* uses the semiparametric method, grouping individuals according to their zip code, disease acuity, age group, and area of diagnosis. Under *Semipar*, we estimate a logit model with no explanatory variables, just hospital-specific constant terms, separately for each group. The model is equivalent to a logit model with 816 hospital-group indicators. The second model *GNT* follows Gowrisankaran, Nevo, and Town (2015) and includes travel time, travel time squared, hospital fixed effects, and interactions of demographic and hospital characteristics with travel time. The third model *Ho* follows Ho (2006) and includes time, time squared, hospital fixed effects, and interactions between hospital and patient characteristics.

The final model *Time+ZIP* is our own specification. It includes travel time, travel time squared, and hospital fixed effects. It also includes zip code-hospital indicators for three zip codes with a large number of patients who received treatment at Sumter hospital in the pre-period. Together with hospital fixed effects, our model has 32 indicator variables. While not as many as *Semipar*, *Time+ZIP* has continuous variables, which the semiparametric method cannot accommodate.

We estimate *Ho*, *GNT*, and *Time+ZIP* using the MM algorithm paired with SQUAREM. The flexibility of the MM algorithm allows us to accommodate all of these different specifications. While we could have estimated *Semipar* with the MM algorithm, we chose to use the estimation approach in Raval, Rosenbaum, and Tenn (2017) since it was easily available to us.

8.3 Predictive Performance

We evaluate the four models based on their prediction errors, $e_k = \hat{D}^k - D_{observed}^k$. In particular, we focus on the mean squared prediction errors. We use bootstrapping to compute 95% confidence intervals. As in RRW, we bootstrap the pre- and post-period datasets to account for variation in the observed and predicted diversions. While the predicted diversions depend only on the variation in the pre-period data, the observed diversions depend on the variation in both periods.

Figure 8

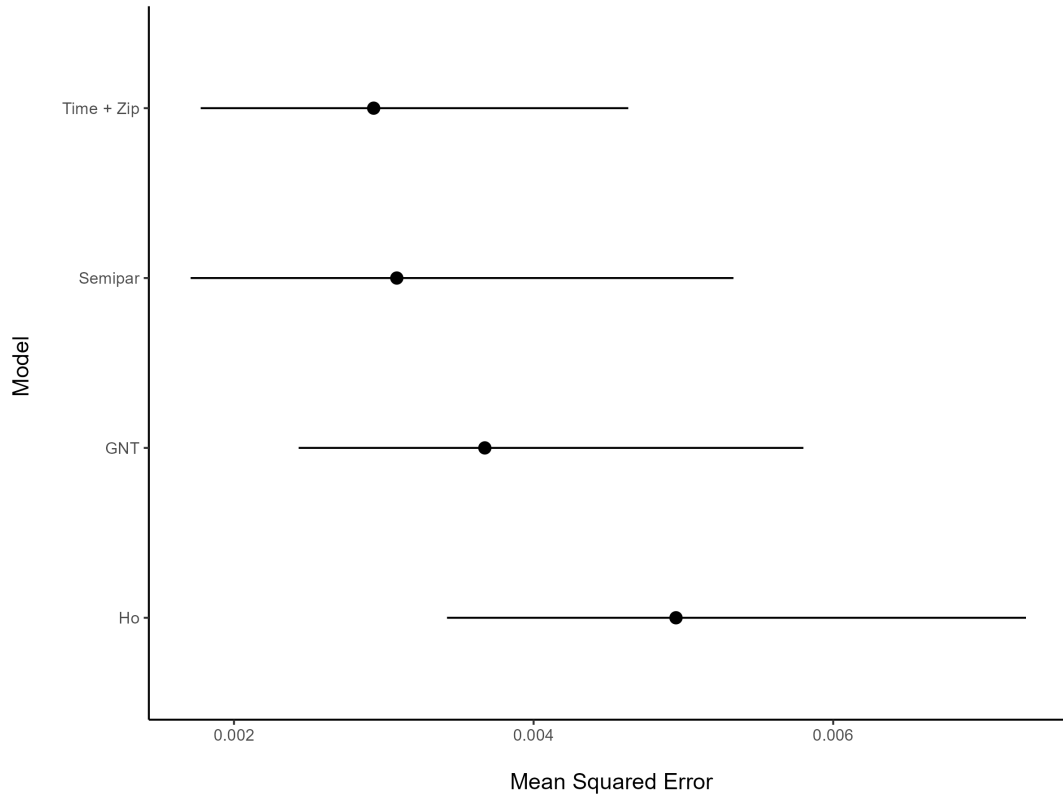


Figure 8 shows the 95% bootstrapped confidence intervals for the different models. From the figure, we see that *Time+ZIP* is best performing model, slightly edging out *Semipar*. *GNT* and *Ho* are the worst performing models on average.

As RRW note, including patient characteristics and interactions does not necessarily help us predict diversions. *GNT* and *Ho* perform the worst even though they include many patient characteristics and various interactions. The results also highlight that we do not need hundreds of group-hospital indicators to capture rich substitution patterns. A model that incorporates travel time with many fixed effects may perform just as well as the semiparametric approach. Another approach we could take would be to add a continuous term, such as patient distance interacted with age, to the 816 hospital-group interactions in RRW, which naturally would allow us to improve fit.

The MM algorithm allows practitioners to estimate a wide range of logit models. We were able to estimate models with a variety of covariates and fixed effects. In this specific example, the MM algorithm allowed us to estimate a logit model that incorporated continuous variables and fixed effects, allowing us to outperform the semiparametric method.

9 Conclusion

We present a new method for estimating multinomial models with many fixed effects. Existing implementations rely on non-linear estimation of many dummy variable coefficients, which is computationally slow and memory-intensive. Our approach relies on the Minorization-Maximization (MM) Algorithm, which can be seen as a generalization of the better-known Expectation-Maximization (EM) algorithm. Our implementation maximizes the likelihood through an iterative series of linear regressions, allowing fixed effects to be handled through the use of linear panel methods or through conditional maximization. We also show how the existing MM algorithm of Böhning (1992) can be adapted to better handle fixed effects. Our MM algorithm is particularly well suited for fixed effects estimation.

We show the performance of our algorithms through two Monte Carlo simulations. Our method perform as well as existing methods when estimating fixed effects while being easier to program. Moreover, adding acceleration allows our methods to outperform the alternatives. With individual-specific coefficients, our method paired with acceleration performs slightly worse, but is still a good option due to its low implementation cost.

Our method provides a new way to estimate multinomial logit models in a variety of settings. We demonstrate one potential use to estimating models of hospital choice and show how our method provides greater flexibility for estimating diversions relative to Raval, Rosenbaum, and Tenn (2017), a recent popular approach. Overall, our method expands the set of specifications that practitioners can consider given their limited time and computational resources.

Appendices

Appendix A Proofs

A.1 Proof of Theorem 1

We show that $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$ is a minorization of $\mathcal{L}(\boldsymbol{\theta}^{(k)})$ at $\boldsymbol{\theta}^{(k)}$ by verifying:

$$(1) \quad \underline{S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) \leq \mathcal{L}(\boldsymbol{\theta})}$$

We start from a representative individual i . Let $\boldsymbol{\psi}_{it} = x_{it}\boldsymbol{\beta} + \mathbf{w}_t\gamma_i + \boldsymbol{\alpha}_i$ where $\boldsymbol{\alpha}_i = (\alpha_{i1}, \dots, \alpha_{iJ})$. The multinomial logit likelihood for observation (i, t) is:

$$l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}) = \prod_{j=1}^J \left(\frac{\exp(\psi_{ijt})}{\sum_{k=1}^J \exp(\psi_{ikt})} \right)^{y_{ijt}}.$$

The scaled score function and Hessian are:

$$\begin{aligned} h_j(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}) &= 2(y_{ijt} - p_{ijt}(\psi_{ijt})), \\ h_{jk}(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}) &= \begin{cases} -p_{ijt}(\psi_{ijt})(1 - p_{ijt}(\psi_{ijt})), & j = k \\ p_{ijt}(\psi_{ijt})p_{ikt}(\psi_{ikt}), & j \neq k \end{cases} \end{aligned}$$

Note that $h_j(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})$ and $h_{jk}(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})$ are scalar valued even though $\boldsymbol{\psi}_{it}$ and $\mathbf{y}_{it}$ are vector valued.

The second-order Taylor expansion of $\log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})$ at $\tilde{\boldsymbol{\psi}}_{it} = \boldsymbol{\psi}_{it}^{(k)}$ is:

$$\log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}) = \log l(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) + (\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it})' \nabla \log l(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) + \frac{1}{2} (\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it})' \nabla^2 \log l(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) (\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it}) \quad (38)$$

We have that:

$$\nabla^2 \log l(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) = \begin{bmatrix} h_{11}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) & h_{12}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) & \dots & h_{1J}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) \\ h_{21}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) & h_{22}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) & \dots & h_{2J}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) \\ \vdots & \vdots & & \vdots \\ h_{J1}(\boldsymbol{\psi}_{it}^*) & h_{J2}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) & \dots & h_{JJ}(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) \end{bmatrix} \quad (39)$$

$$= \begin{bmatrix} -p_{i1t}(1 - p_{i1t}) & p_{i1t}p_{i2t} & \dots & p_{i1t}p_{iJt} \\ p_{i2t}p_{i1t} & -p_{i2t}(1 - p_{i2t}) & \dots & p_{i2t}p_{iJt} \\ \vdots & \vdots & & \vdots \\ p_{iJt}p_{i1t} & p_{iJt}p_{i2t} & \dots & -p_{iJt}(1 - p_{iJt}) \end{bmatrix}, \quad (40)$$

Letting $\Delta\boldsymbol{\psi}_{it} = \boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it}$ and $\mathbf{p}_{it}$ be the J vector with elements p_{ijt} , we can write the negative of the Hessian matrix in (46) as $\mathbf{W}_{it} = \text{Diag}(\mathbf{p}_{it}) - \mathbf{p}_{it}\mathbf{p}_{it}'$. Note that for any vector $\Delta\boldsymbol{\psi}_{it}$:

$$\Delta\boldsymbol{\psi}_{it}' \mathbf{W}_{it} \Delta\boldsymbol{\psi}_{it} = \sum_{j=1}^J p_{ijt} \Delta\psi_{ijt}^2 - \left(\sum_{j=1}^J p_{ijt} \Delta\psi_{ijt} \right)^2 \leq \frac{1}{2} \sum_{j=1}^J \Delta\psi_{ijt}^2 \quad (41)$$

This implies $\mathbf{W}_{it} \preceq \frac{1}{2}\mathbf{I}$. Thus, because the Hessian is $\nabla^2 \log l(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) = -\mathbf{W}_{it}$, we have $\nabla^2 \log l(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) \succeq -\frac{1}{2}\mathbf{I}$, meaning $\nabla^2 \log l(\boldsymbol{\psi}_{it}^*; \mathbf{y}_{it}) + \frac{1}{2}\mathbf{I}$ is positive semi-definite.

Therefore:

$$\begin{aligned} \log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}) &\geq \log l(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) + (\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it})' \nabla \log l(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) - \frac{1}{4} (\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it})' (\boldsymbol{\psi}_{it} - \tilde{\boldsymbol{\psi}}_{it}) \\ &= \log l(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) + \sum_j (y_{ijt} - p_{ijt}(\tilde{\boldsymbol{\psi}}_{it})) (\psi_{ijt} - \tilde{\psi}_{ijt}) - \frac{1}{4} \sum_j (\tilde{\psi}_{ijt} - \psi_{ijt})^2 \end{aligned}$$

Substituting $h_j(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) = 2(y_{ijt} - p_{ijt}(\tilde{\boldsymbol{\psi}}_{it}))$, we have:

$$\begin{aligned}
\log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it}) &\geq \log l(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) + \frac{1}{2} \sum_j h_j(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) (\psi_{ijt} - \tilde{\psi}_{ijt}) - \frac{1}{4} \sum_j (\tilde{\psi}_{ijt} - \psi_{ijt})^2 \\
&= \log l(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) + \frac{1}{4} \sum_j h_j(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it})^2 - \frac{1}{4} \sum_j (\tilde{\psi}_{ijt} + h_j(\tilde{\boldsymbol{\psi}}_{it}; \mathbf{y}_{it}) - \psi_{ijt})^2
\end{aligned}$$

After algebraic manipulation and summing over all observations, we obtain:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &\geq \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \frac{1}{4} \sum_{i,j,t} h_j(x_{it}\boldsymbol{\beta}^{(k)} + \mathbf{w}_t\gamma_i^{(k)} + \boldsymbol{\alpha}_i^{(k)}; \mathbf{y}_{it})^2 \\
&\quad - \frac{1}{4} \sum_{i,j,t} (x_{it}\boldsymbol{\beta}^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + h_j(x_{it}\boldsymbol{\beta}^{(k)} + \mathbf{w}_t\gamma_i^{(k)} + \boldsymbol{\alpha}_i^{(k)}; \mathbf{y}_{it}) \\
&\quad - (x_{it}\boldsymbol{\beta} + w_{jt}\gamma_i + \alpha_{ij}))^2
\end{aligned} \tag{42}$$

where the RHS of the above inequality as $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$.

$$(2) \quad \underline{S(\boldsymbol{\theta}^{(k)}; \boldsymbol{\theta}^{(k)}) = \mathcal{L}(\boldsymbol{\theta}^{(k)})}$$

Direct substitution shows:

$$\begin{aligned}
S(\boldsymbol{\theta}^{(k)}; \boldsymbol{\theta}^{(k)}) &= \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \frac{1}{4} \sum_{i,j,t} h_j(\boldsymbol{\psi}^{(k)}; \mathbf{y}_{it})^2 - \\
&\quad \frac{1}{4} \sum_{i,j,t} (\boldsymbol{\psi}^{(k)} + h_j(\boldsymbol{\psi}^{(k)}; \mathbf{y}_{it}) - \boldsymbol{\psi}^{(k)})^2 \\
&= \mathcal{L}(\boldsymbol{\theta}^{(k)}) + \frac{1}{4} \sum_{i,j,t} h_j(\boldsymbol{\psi}^{(k)}; \mathbf{y}_{it})^2 - \frac{1}{4} \sum_{i,j,t} h_j(\boldsymbol{\psi}^{(k)}; \mathbf{y}_{it})^2 \\
&= \mathcal{L}(\boldsymbol{\theta}^{(k)})
\end{aligned}$$

$$(3) \quad \underline{\nabla_{\boldsymbol{\theta}}^2 S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) \text{ exists, and } \nabla_{\boldsymbol{\theta}}^2 S(\boldsymbol{\theta}^{(k+1)}; \boldsymbol{\theta}^{(k)}) \text{ is negative definite}}$$

To ease notation, we consider $\boldsymbol{\alpha}_i$ as coefficients on indicator variables, combine these indicator variables with x_{it} and w_{jt} and denote the combined vector as z_{ijt} . Then by the definition of $S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$ in Eq.(42), $\nabla_{\boldsymbol{\theta}}^2 S(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}) = -\frac{1}{2} \mathbf{Z}' \mathbf{Z}$ where $\mathbf{Z}$ is the matrix with z_{ijt} , $\forall i, j, t$ stacked by rows, and does not depend on $\boldsymbol{\theta}$. Assuming

that Z is full rank, then $Z'Z$ is positive definite, so we have our result. For Z to be full rank, we need sufficient variation of x_{it} and a normalization on fixed effects parameters α_i , such as that $\alpha_{ij} = 0$ for one value of j for all i .

A.2 Proof of the Equivalence for Models with Alternative-Specific Parameters

Consider the multinomial logit model with only alternative-specific parameters and fixed effects:

$$u_{ijt} = x_{it}\beta_j + \alpha_{ij} + \epsilon_{ijt} \quad (43)$$

We show that our MM algorithm produces relative parameter updates that are algebraically identical to Böhning's MM algorithm at every iteration.

Böhning's MM algorithm operates in the space of $J - 1$ alternatives normalized against the base alternative $j = 1$. As shown in Section 4.2, maximizing Böhning's surrogate is equivalent to a generalized least-squares regression. The resulting parameter step is:

$$\Delta\theta_B = (\dot{Z}'(\mathbf{I}_{NT} \otimes \mathbf{B})\dot{Z})^{-1} \dot{Z}'\dot{\mathbf{h}}^{(k)} \quad (44)$$

where $\dot{Z}$ is the stacked design matrix for the $J - 1$ non-base alternatives, $\mathbf{B} = \frac{1}{2}(\mathbf{I}_{J-1} - \frac{1}{J}\mathbf{1}\mathbf{1}')$, and $\dot{\mathbf{h}}^{(k)}$ is the stacked score vector with observation-level components $\dot{\mathbf{h}}_{it}^{(k)} = \mathbf{y}_{it} - \mathbf{p}_{it}^{(k)}$.

Because all coefficients are alternative-specific, the identical regressors are used across all alternatives. Let $\mathbf{X}$ be the $NT \times K$ matrix containing all observation-level regressors x_{it} and individual dummy variables for α_{ij} . We can write the non-base design matrix using Kronecker products as $\dot{Z} = \mathbf{X} \otimes \mathbf{I}_{J-1}$. Consequently:

$$\dot{Z}'(\mathbf{I}_{NT} \otimes \mathbf{B})\dot{Z} = (\mathbf{X}'\mathbf{X}) \otimes \mathbf{B} \quad (45)$$

Using standard properties of Kronecker products, the Böhning update simplifies to:

$$\begin{aligned} \Delta\theta_B &= [(\mathbf{X}'\mathbf{X})^{-1} \otimes \mathbf{B}^{-1}] [\mathbf{X}' \otimes \mathbf{I}_{J-1}] \dot{\mathbf{h}}^{(k)} \\ &= [(\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}' \otimes \mathbf{B}^{-1}] \dot{\mathbf{h}}^{(k)} \end{aligned} \quad (46)$$

Recall that $\mathbf{B}^{-1} = 2(\mathbf{I}_{J-1} + \mathbf{1}\mathbf{1}')$. Multiplying this inverse bound by the observation-level score vector $\dot{\mathbf{h}}_{it}^{(k)}$ yields $2\dot{\mathbf{h}}_{it}^{(k)} + 2(\mathbf{1}'\dot{\mathbf{h}}_{it}^{(k)})\mathbf{1}$. Because the logit probabilities sum to one across all J alternatives, the sum of the non-base scores equals the negative of the base score: $\mathbf{1}'\dot{\mathbf{h}}_{it}^{(k)} = -(y_{i1t} - p_{i1t}^{(k)})$. Therefore, Böhning's update for a specific non-base alternative j is:

$$\Delta\boldsymbol{\theta}_{B,j} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'[2(\mathbf{y}_j - \mathbf{p}_j^{(k)}) - 2(\mathbf{y}_1 - \mathbf{p}_1^{(k)})] \quad (47)$$

where $\mathbf{y}_j$ and $\mathbf{p}_j^{(k)}$ are the stacked vectors for alternative j .

In contrast, our scalar MM algorithm operates in the full space of J alternatives. The update step derived from our OLS surrogate is:

$$\Delta\boldsymbol{\theta}_{MM} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{h}^{(k)} \quad (48)$$

where $\mathbf{Z} = \mathbf{X} \otimes \mathbf{I}_J$ is the full design matrix, and $\mathbf{h}^{(k)}$ is the stacked vector of our scores defined in Theorem 1 as $h_{ijt}^{(k)} = 2(y_{ijt} - p_{ijt}^{(k)})$. Using Kronecker products, this simplifies to:

$$\Delta\boldsymbol{\theta}_{MM} = [(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \otimes \mathbf{I}_J]\mathbf{h}^{(k)} \quad (49)$$

For a specific alternative j , the absolute update step is simply:

$$\Delta\boldsymbol{\theta}_{MM,j} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'[2(\mathbf{y}_j - \mathbf{p}_j^{(k)})] \quad (50)$$

To compare this with Böhning's normalized update, we difference the step for alternative j against the base alternative 1:

$$\Delta\boldsymbol{\theta}_{MM,j} - \Delta\boldsymbol{\theta}_{MM,1} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'[2(\mathbf{y}_j - \mathbf{p}_j^{(k)}) - 2(\mathbf{y}_1 - \mathbf{p}_1^{(k)})] \quad (51)$$

Comparing (47) and (51), the relative parameter steps are perfectly identical. Consequently, for models with strictly alternative-specific coefficients, our unconstrained OLS MM algorithm and Böhning's GLS MM algorithm generate the exact same relative parameter updates, predicted probabilities, and log-likelihood values at every iteration.

Appendix B Algorithms to implement Böhning, 1992

B.1 Böhning's Algorithm

While Böhning (1992) derives his update as an explicit mathematical map from one iterate to the next, we demonstrated how to recast his solution as a sequence of regressions. Here, we demonstrate how to computationally implement this regression approach. Recall that Böhning's bound yields the following GLS problem:

$$\min_{\theta} \sum_{i,t} \left(\dot{\mathbf{v}}_{it}^{(k)} - \mathbf{Z}_{it}\theta \right)' \mathbf{B} \left(\dot{\mathbf{v}}_{it}^{(k)} - \mathbf{Z}_{it}\theta \right) \quad (52)$$

where the pseudo-data is:²¹

$$\dot{v}_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)})) - 2(y_{i1t} - p_{i1t}(\theta^{(k)})) \quad (53)$$

Rather than computing GLS solution directly, it is generally more efficient to whiten the data and transform the problem into an OLS problem in the space of $J - 1$ alternatives.

Let L be the symmetric square root of the Böhning matrix such that $\mathbf{B} = \mathbf{L}\mathbf{L}'$. The whitened pseudo-data is $\ddot{\mathbf{v}} = L\dot{\mathbf{v}}$ and the whitened regressor matrix is $\ddot{\mathbf{Z}} = \mathbf{L}\mathbf{Z}$. The GLS update is then equivalent to the OLS problem:

$$\min_{\theta} \left(\ddot{\mathbf{v}}^{(k)} - \ddot{\mathbf{Z}}\theta \right)' \left(\ddot{\mathbf{v}}^{(k)} - \ddot{\mathbf{Z}}\theta \right) \quad (54)$$

Since Böhning's bound is constant, we need to compute the symmetric root only once at the beginning. Moreover, L has a simple closed-form that makes whitening

²¹Because Böhning's MM algorithm operates in the normalized space of $J - 1$ alternatives, any regressors associated with common coefficients must be differenced against the base alternative. For notational convenience in the formulas and algorithms for Böhning's method, we let w_{jt} denote the already base-differenced characteristic ($w_{jt} - w_{1t}$).

the data straightforward. For a vector $\mathbf{x}$, the whitened version is:

$$\ddot{\mathbf{x}} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J + \sqrt{J}} \mathbf{1}\mathbf{1}' \right) \mathbf{x} \quad (55)$$

Thus, we can calculate the whitened data from simple vector operations and avoid the matrix multiplication entirely. This closed-form formula assumes a uniform choice set of size J across all observations. In settings where choice sets vary, the transformation remains identical but uses the specific choice set size J_{it} for each observation, increasing the computational cost of the whitening step.

The base algorithm for Böhning's approach is then:

Algorithm 6 (Böhning's MM Algorithm)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1, \dots, I} \ j=2, \dots, J$

repeat

 Minorization Step:

1. Compute the pseudo-data:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)})) - 2(y_{i1t} - p_{i1t}(\theta^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1, \dots, N; j=2, \dots, J; t=1, \dots, T}$$

2. Whiten the pseudo-data:

$$\ddot{\mathbf{v}}^{(k)} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J + \sqrt{J}} \mathbf{1}\mathbf{1}' \right) \mathbf{v}^{(k)}$$

 Maximization Step (Update $\boldsymbol{\theta}^{(k)}$ by OLS):

$$\boldsymbol{\theta}^{(k+1)} \leftarrow \left(\ddot{\mathbf{Z}}' \ddot{\mathbf{Z}} \right)^{-1} \ddot{\mathbf{Z}}' \ddot{\mathbf{v}}^{(k)}$$

$$k \leftarrow k + 1$$

until convergence

Because the whitened regressor matrix does not change from iteration to iteration, we can compute the pseudo-inverse $\left(\ddot{\mathbf{Z}}' \ddot{\mathbf{Z}} \right)^{-1} \ddot{\mathbf{Z}}'$ once at the beginning.

B.2 Handling one-way fixed effects in Böhning's MM Algorithm

The algorithm for Böhning's algorithm is very similar to the MM algorithm we present above for one-way fixed fixed effects, but the partitioned regression uses the demeaned whitened data $\ddot{\mathbf{Z}}$. We also use the version of the fixed effect update that takes the mean over the residuals. This version allows us to write the update in terms of the original data, rather the whitened data. While we can use the update that relies on the difference between the residuals and demeaned residuals, that update will produce a whitened version of the fixed effects, which then requires unwhitening to update the linear index.

Algorithm 7 (Böhning's MM Algorithm - Fixed Effects)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} j=2,\dots,J$

repeat

Minorization Step:

1. Compute the pseudo-data:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)})) - 2(y_{i1t} - p_{i1t}(\theta^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,N; j=2,\dots,J; t=1,\dots,T}$$

2. Whiten the pseudo-data:

$$\ddot{\mathbf{v}}^{(k)} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J+\sqrt{J}} \mathbf{1}\mathbf{1}' \right) \mathbf{v}^{(k)}$$

3. Demean the whitened pseudo-data:

$$\ddot{\mathbf{v}}^{(k)} \leftarrow \text{Demean } \ddot{\mathbf{v}}^{(k)}$$

Maximization Step:

1. Update $\boldsymbol{\eta}^{(k)}$ by demeaned OLS:

$$\boldsymbol{\eta}^{(k+1)} \leftarrow \left(\ddot{\mathbf{Z}}' \ddot{\mathbf{Z}} \right)^{-1} \ddot{\mathbf{Z}}' \ddot{\mathbf{v}}^{(k)}$$

2. Update the fixed effects: $\boldsymbol{\alpha}$

$$\alpha_{ij}^{(k+1)} = \frac{1}{T} \sum_t v_{ijt}^{(k)} - z_{ijt} \boldsymbol{\eta}^{(k+1)}$$

$k \leftarrow k + 1$

until convergence

B.2.1 Böhning's MCM Algorithm

We can also apply conditional maximization to Böhning's algorithm, but it is not as straightforward as with our algorithm. In the MCM version of Böhning's algorithm, we need to whiten the fixed effects in the conditional maximization step.

Algorithm 8 (Böhning's MCM Algorithm)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} \ j=2,\dots,J$

repeat

Minorization Step:

1. Compute the pseudo-data:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)})) - 2(y_{i1t} - p_{i1t}(\theta^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,N; j=2,\dots,J; t=1,\dots,T}$$

2. Whiten the pseudo-data:

$$\ddot{\mathbf{v}}^{(k)} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J+\sqrt{J}} \mathbf{1}\mathbf{1}' \right) \mathbf{v}^{(k)}$$

3. Whiten the fixed effects:

$$\ddot{\boldsymbol{\alpha}}^{(k)} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J+\sqrt{J}} \mathbf{1}\mathbf{1}' \right) \boldsymbol{\alpha}^{(k)}$$

Conditional Maximization Steps:

1. Update $\boldsymbol{\eta}^{(k)}$ by OLS:

$$\boldsymbol{\eta}^{(k+1)} \leftarrow \left(\ddot{\mathbf{Z}}' \ddot{\mathbf{Z}} \right)^{-1} \ddot{\mathbf{Z}}' (\ddot{\mathbf{v}}^{(k)} - \ddot{\boldsymbol{\alpha}}^{(k)})$$

2. Update the fixed effects: $\boldsymbol{\alpha}$

$$\alpha_{ij}^{(k+1)} = \frac{1}{T} \sum_t v_{ijt}^{(k)} - z_{ijt} \boldsymbol{\eta}^{(k+1)}$$

$k \leftarrow k + 1$

until convergence

B.3 Handling Individual-Specific Coefficients in Böhning's MM Algorithm

In this section, we show how to handle individual-specific coefficients in Böhning's MM Algorithm through demeaning. Böhning's MM Algorithm recovers the

whitened contribution of the nuisance parameters rather than the raw, unwhitened contribution. The algorithm is:

Algorithm 9 (Böhning's MM Algorithm -Fixed Effects and Individual-Specific Coefficients)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} \ j=2,\dots,J$

repeat

 Minorization Step:

1. Compute the pseudo-data:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)})) - 2(y_{i1t} - p_{i1t}(\theta^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,N; j=2,\dots,J; t=1,\dots,T}$$

2. Whiten the pseudo-data:

$$\ddot{\mathbf{v}}^{(k)} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J+\sqrt{J}} \mathbf{1}\mathbf{1}' \right) \mathbf{v}^{(k)}$$

3. Demean the whitened pseudo-data:

$$\ddot{\mathbf{v}}^{(k)} \leftarrow \text{Demean } \ddot{\mathbf{v}}^{(k)}$$

 Maximization Step:

1. Update $\beta^{(k)}$ by demeaned OLS:

$$\boldsymbol{\beta}^{(k+1)} \leftarrow \left(\ddot{\mathbf{X}}' \ddot{\mathbf{X}} \right)^{-1} \ddot{\mathbf{X}}' \ddot{\mathbf{v}}^{(k)}$$

2. Update the whitened contribution $\ddot{\xi} = \mathbf{L}(\mathbf{w}\boldsymbol{\gamma} + \boldsymbol{\alpha})$:

$$\ddot{\xi}^{(k+1)} \leftarrow \ddot{\mathbf{v}}^{(k)} - \ddot{\mathbf{X}}\boldsymbol{\beta}^{(k+1)} - \left(\ddot{\mathbf{v}}^{(k)} - \ddot{\mathbf{X}}\boldsymbol{\beta}^{(k+1)} \right)$$

3. Unwhiten the contribution:

$$\boldsymbol{\xi}^{(k+1)} = \sqrt{2} \left(\mathbf{I}_{J-1} + \frac{1}{\sqrt{J+1}} \mathbf{1}\mathbf{1}' \right) \ddot{\xi}^{(k+1)}$$

$k \leftarrow k + 1$

until convergence

We could instead use conditional maximization to find the parameters. This requires whitening the fixed effects for use in the OLS update.

Algorithm 10 (Böhning's MCM Algorithm -Fixed Effects and Individual-Specific Coefficients)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} \ j=2,\dots,J$

repeat

Minorization Step:

1. Compute the pseudo-data:

$$v_{ijt}^{(k)} \leftarrow x_{it}\beta_j^{(k)} + w_{jt}\gamma_i^{(k)} + \alpha_{ij}^{(k)} + 2(y_{ijt} - p_{ijt}(\theta^{(k)})) - 2(y_{i1t} - p_{i1t}(\theta^{(k)}))$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,N; j=2,\dots,J; t=1,\dots,T}$$

2. Whiten the pseudo-data:

$$\ddot{\mathbf{v}}^{(k)} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J+\sqrt{J}} \mathbf{1}\mathbf{1}' \right) \mathbf{v}^{(k)}$$

3. Whiten the fixed effects:

$$\ddot{\boldsymbol{\alpha}}^{(k)} = \frac{1}{\sqrt{2}} \left(\mathbf{I}_{J-1} - \frac{1}{J+\sqrt{J}} \mathbf{1}\mathbf{1}' \right) \boldsymbol{\alpha}^{(k)}$$

Conditional Maximization Steps:

1. Update $\boldsymbol{\beta}^{(k)}$ by OLS:

$$\boldsymbol{\beta}^{(k+1)} \leftarrow \left(\ddot{\mathbf{X}}' \ddot{\mathbf{X}} \right)^{-1} \ddot{\mathbf{X}}' (\ddot{\mathbf{v}}^{(k)} - \ddot{\mathbf{W}} \boldsymbol{\gamma}^{(k)} - \ddot{\boldsymbol{\alpha}}^{(k)})$$

2. Update $\boldsymbol{\gamma}$ as:

$$\boldsymbol{\gamma}^{(k+1)} = (\ddot{\mathbf{W}}' \ddot{\mathbf{W}})^{-1} \ddot{\mathbf{W}}' (\ddot{\mathbf{v}}^{(k)} - \ddot{\mathbf{X}} \boldsymbol{\beta}^{(k+1)} - \ddot{\boldsymbol{\alpha}}^{(k)})$$

3. Update $\boldsymbol{\alpha}$ as:

$$\alpha_{ij}^{(k+1)} = \frac{1}{T} \sum_t v_{ijt}^{(k)} - x_{it}\beta_j^{(k+1)} - w_{jt}\gamma_i^{(k+1)}$$

$k \leftarrow k + 1$

until convergence

Appendix C Implementing IRLS

In this appendix, we discuss details of implementing the IRLS, which has implications for how to compute standard errors.

C.1 The General Algorithm

Recall that $\psi_{it} = x_{it}\beta + w_t\gamma_i + \alpha_i$. Normalize the mean utility of the base option $j = 1$ to zero, $\psi_{i1t} = 0$. The second-order Taylor expansion in ψ around $\psi^{(k)}$ is:

$$\begin{aligned} \tilde{S}(\theta; \theta^{(k)}) &= \mathcal{L}(\theta^{(k)}) + \sum_{i,t} h(\psi_{it}^{(k)})' (\psi_{it} - \psi_{it}^{(k)}) \\ &\quad - \frac{1}{2} \sum_{i,t} \left(\psi_{it} - \psi_{it}^{(k)} \right)' \Omega_{it} \left(\psi_{it} - \psi_{it}^{(k)} \right) \end{aligned} \tag{56}$$

where $\Omega_{it} = \text{diag}(p_{it}) - p_{it}p_{it}'$ and $h_{ijt}^{(k)} = y_{ijt} - p_{ijt}^{(k)}$. We can rewrite this approximation as:

$$\begin{aligned} \tilde{S}(\theta; \theta^{(k)}) &= \mathcal{L}(\theta^{(k)}) + \frac{1}{2} \sum_{i,t} h(\psi_{it}^{(k)})' \Omega_{it}^{-1} h(\psi_{it}^{(k)}) \\ &\quad - \frac{1}{2} \sum_{i,t} \left(\psi_{it}^{(k)} + \Omega_{it}^{-1} h(\psi_{it}^{(k)}) - \psi_{it} \right)' \Omega_{it} \left(\psi_{it}^{(k)} + \Omega_{it}^{-1} h(\psi_{it}^{(k)}) - \psi_{it} \right) \end{aligned} \tag{57}$$

Maximizing this surrogate function is equivalent solving to the generalized least squares problem:

$$\min \sum_{it} \left(v_{it}^{(k)} - \psi_{it}(\theta) \right) \Omega_{it}^{(k)} \left(v_{it}^{(k)} - \psi_{it}(\theta) \right) \tag{58}$$

where $v_{it}^{(k)} = \psi_{it}^{(k)} + \Omega_{it}^{(k)-1} h(\psi_{it}^{(k)}; \mathbf{y}_{it})$.

Let $\mathbf{Z}$ be the matrix of independent regressors and $\Omega^{(k)}$ be the full weight matrix

at the iteration k .

$$\Omega^{(k)} = \begin{bmatrix} \Omega_{11}^{(k)} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Omega_{12}^{(k)} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & & \Omega_{it}^{(k)} & & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \Omega_{IT}^{(k)} \end{bmatrix} \quad (59)$$

Then the closed-form solution to this problem is:

$$\boldsymbol{\theta}^{(k+1)} = (\mathbf{Z}'\Omega^{(k)}\mathbf{Z})^{-1}\mathbf{Z}'\Omega^{(k)}\mathbf{v}^{(k)} \quad (60)$$

The algorithm is then:

Algorithm 11 (Iterative Reweighted Least Squares Algorithm)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \gamma_i^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} \ j=2,\dots,J$

repeat

Working-Data Step:

$$p_{ijt} \leftarrow \frac{\exp(\psi_{ijt})}{1 + \sum_k \exp(\psi_{ikt})}$$

$$\Omega_{it}^{(k)} \leftarrow \text{diag}(p_{it}) - p_{it}p'_{it}$$

$$\mathbf{v}_{it}^{(k)} \leftarrow x_{it}\boldsymbol{\beta}^{(k)} + \mathbf{w}_t\gamma_i^{(k)} + \boldsymbol{\alpha}_i^{(k)} + (\Omega_{it}^{(k)})^{-1}\mathbf{h}(\psi_{it}^{(k)}; \mathbf{y}_{it})$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,I; j=2,\dots,J; t=1,\dots,T}$$

Maximization Step: (Update θ^k by GLS)

$$\boldsymbol{\theta}^{(k+1)} = (\mathbf{Z}'\Omega^{(k)}\mathbf{Z})^{-1}\mathbf{Z}'\Omega^{(k)}\mathbf{v}^{(k)}$$

$$k \leftarrow k + 1$$

until convergence

C.2 Computing the GLS Solution

The weight matrix $\mathbf{\Omega}$ is quite large. It is a $(J-1) \cdot IT \times (J-1) \cdot IT$ block diagonal with individual blocks $\mathbf{\Omega}_{it}^{(k)}$. Given its size, naively storing and multiplying $\mathbf{\Omega}^{(k)}$ to compute the maximization step is slow and memory consuming. We recommend storing the individual non-zero blocks of $\mathbf{\Omega}^{(k)}$ and then using block matrix multiplication to compute the GLS update.

The matrices $\mathbf{Z}$ and $\mathbf{v}^{(k)}$ have elements Z_{it} and $v_{it}^{(k)}$ corresponding to $\mathbf{\Omega}_{it}^{(k)}$. Using block matrix multiplication, we can update $\boldsymbol{\theta}$ as:

$$\begin{aligned}\boldsymbol{\theta}^{(k+1)} &= (\mathbf{Z}'\mathbf{\Omega}^{(k)}\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{\Omega}^{(k)}\mathbf{v}^{(k)} \\ &= \left(\sum_{i,t} (\mathbf{Z}_{it}'\mathbf{\Omega}_{it}^{(k)}\mathbf{Z}_{it}) \right)^{-1} \left(\sum_{i,t} \mathbf{Z}_{it}'\mathbf{\Omega}_{it}^{(k)}\mathbf{v}_{it}^{(k)} \right)\end{aligned}\quad (61)$$

C.3 Handling Fixed Effects

Fixed effects cause problems for the dimensionality of $\mathbf{Z}$. We still want to make use of demeaning to project out the fixed effects. However, the weight matrix complicates the demeaning and makes it more costly. Consider the case without alternative specific variables where utility has the form:

$$u_{ijt} = x_{it}\beta_j + \alpha_{ij} + \varepsilon_{ijt} \quad (62)$$

Let $\mathbf{D}_\alpha$ be the matrix of indicator variables for the non-base fixed effects for individual. In this case, $\mathbf{Z} = [\mathbf{X}, \mathbf{D}_\alpha]$. The projection matrix for the fixed effects is:

$$\mathbf{P}_{\mathbf{D}_\alpha} = \mathbf{D}_\alpha(\mathbf{D}_\alpha'\mathbf{\Omega}^{(k)}\mathbf{D}_\alpha)^{-1}\mathbf{D}_\alpha'\mathbf{\Omega}^{(k)} \quad (63)$$

The matrix $\mathbf{D}_\alpha$ is $(J-1) \cdot IT \times (I \cdot (J-1))$. Given the structure of $\mathbf{D}_\alpha$, we can simplify the projection matrix to a weighted mean.

Let $\mathbf{D}_i$ represent the block of $\mathbf{D}_\alpha$ corresponding to individual i . This block is just

a series of stacked $(J - 1) \times (J - 1)$ identity matrices:

$$\mathbf{D}_i = \begin{bmatrix} \mathbf{D}_{i1} \\ \mathbf{D}_{i2} \\ \vdots \\ \mathbf{D}_{iT} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{J-1} \\ \mathbf{I}_{J-1} \\ \vdots \\ \mathbf{I}_{J-1} \end{bmatrix} \quad (64)$$

The columns correspond to the fixed effects $\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{iJ-1}$. Given this structure, we can calculate the projected value of $\mathbf{v}^{(k)}$ for individual i as:

$$\bar{\mathbf{v}}_i^{(k)} = \sum_t \left[\left(\sum_{\tau} \boldsymbol{\Omega}_{i\tau}^{(k)} \right)^{-1} \boldsymbol{\Omega}_{it}^{(k)} \mathbf{v}_{it}^{(k)} \right] \quad (65)$$

We just end up with essentially a weighted average. The demeaned value is then:

$$\tilde{\mathbf{v}}_{it} = \mathbf{v}_{it} - \bar{\mathbf{v}}_i \quad (66)$$

The expression for the group mean depends on $\boldsymbol{\Omega}^{(k)}$ so we need to demean the design matrix $\mathbf{X}$ at every iteration. This stands in contrast to our algorithm where the design matrix can be demeaned once at the beginning.

Once we demean the variables, we can update the coefficients as:

$$\begin{aligned} \boldsymbol{\beta}^{(k+1)} &= (\tilde{\mathbf{X}}' \boldsymbol{\Omega}^{(k)} \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}' \boldsymbol{\Omega}^{(k)} \tilde{\mathbf{v}}^{(k)} \\ &= \left(\sum_{i,t} (\tilde{\mathbf{X}}_{it}' \boldsymbol{\Omega}_{it}^{(k)} \tilde{\mathbf{X}}_{it}) \right)^{-1} \left(\sum_{i,t} \tilde{\mathbf{X}}_{it}' \boldsymbol{\Omega}_{it}^{(k)} \tilde{\mathbf{v}}_{it}^{(k)} \right) \end{aligned} \quad (67)$$

Finally, we can recover the fixed effects as:

$$\boldsymbol{\alpha}_i^{(k+1)} = \bar{\mathbf{v}}_i^{(k)} - \bar{\mathbf{X}}_i \boldsymbol{\beta}^{(k+1)} \quad (68)$$

Algorithm 12 (Iterative Reweighted Least Squares Algorithm - Fixed Effects)

Require: initial guess $\boldsymbol{\theta}^{(0)} = \{\beta_j^{(0)}, \alpha_{ij}^{(0)}\}_{i=1,\dots,I} j=2,\dots,J$

repeat

Working-Data Step:

$$p_{ijt} \leftarrow \frac{\exp(\psi_{ijt})}{1 + \sum_k \exp(\psi_{ikt})}$$

$$\boldsymbol{\Omega}_{it}^{(k)} \leftarrow \text{diag}(p_{it}) - p_{it} p_{it}'$$

$$v_{it}^{(k)} \leftarrow x_{it} \boldsymbol{\beta}^{(k)} + \boldsymbol{\alpha}_i^{(k)} + (\boldsymbol{\Omega}_{it}^{(k)})^{-1} \mathbf{h}(\boldsymbol{\psi}_{it}^{(k)}; \mathbf{y}_{it})$$

$$\mathbf{v}^{(k)} \leftarrow \{v_{ijt}^{(k)}\}_{i=1,\dots,I; j=2,\dots,J; t=1,\dots,T}$$

Maximization Step:

1. Demean the variables:

$$\bar{v}_i \leftarrow \sum_t \left[\left(\sum_{\tau} \boldsymbol{\Omega}_{i\tau}^{(k)} \right)^{-1} \boldsymbol{\Omega}_{it}^{(k)} \mathbf{v}_{it}^{(k)} \right]$$

$$\bar{\mathbf{X}}_i \leftarrow \sum_t \left[\left(\sum_{\tau} \boldsymbol{\Omega}_{i\tau}^{(k)} \right)^{-1} \boldsymbol{\Omega}_{it}^{(k)} \mathbf{X}_{it} \right]$$

$$\tilde{v}_{it} \leftarrow v_{it} - \bar{v}_i$$

$$\tilde{\mathbf{X}}_{it} \leftarrow \mathbf{X}_{it} - \bar{\mathbf{X}}_i$$

2. Update $\boldsymbol{\beta}$ by demeaned GLS:

$$\boldsymbol{\beta}^{(k+1)} \leftarrow \left(\sum_{i,t} (\tilde{\mathbf{X}}_{it}' \boldsymbol{\Omega}_{it}^{(k)} \tilde{\mathbf{X}}_{it}) \right)^{-1} \left(\sum_{i,t} \tilde{\mathbf{X}}_{it}' \boldsymbol{\Omega}_{it}^{(k)} \tilde{v}_{it}^{(k)} \right)$$

3. Update the fixed effects: $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha}_i^{(k+1)} = \bar{v}_i^{(k)} - \bar{\mathbf{X}}_i \boldsymbol{\beta}^{(k+1)}$$

$k \leftarrow k + 1$

until convergence

The algorithm extends naturally to the case of multi-way fixed effects. We can sweep out the multi-way fixed effects using the method of alternating projections applying the individual projections repeatedly until convergence. The projections would be similar to (65) and therefore depend on $\Omega^{(k)}$. So group means need to be calculated at each iteration.

C.4 Implications for Inference

Conducting inference on the parameters of interest does not require implementing the full IRLS algorithm only elements of it. The variance of the parameters of interests is just the inverse of the Schur complement which equals: $\text{Var}(\hat{\beta}) = (\tilde{\mathbf{X}}'\hat{\Omega}\tilde{\mathbf{X}})^{-1}$ This expression requires us to demean X according to our expression in (65) and then use block matrix multiplication as we do in the update for β in the IRLS algorithm.

Appendix D Normalization

In general, our MM algorithm will produce non-zero updates for the base option that is supposed to be normalized to zero. Consider the general case where utility has the form:

$$u_{ijt} = x_{it}\beta_j + \gamma_i w_{jt} + \alpha_{ij} + \epsilon_{ijt} \quad (69)$$

We would like to normalize the fixed effect for alternative 1 to zero, $\alpha_{i1} = 0$. Since utilities are relative, we can subtract off w_{1t} from all options to remove it from the equation for the base updated.

On the k th iteration, the MM algorithm will update the fixed effect for the base option as:

$$\alpha_{i1}^{(k+1)} = \frac{1}{T} \sum_t v_{i1t}^{(k)} - x_{it}\beta_1^{(k+1)} \quad (70)$$

$$\begin{aligned}
&= \bar{x}_i \left(\beta_1^{(k)} - \beta_1^{(k+1)} \right) + \frac{1}{T} \sum_t \alpha_{i1}^{(k)} + 2 \left(y_{i1t} - p_{i1t}^{(k)} \right) \\
&= \bar{x}_i \left(\beta_1^{(k)} - \beta_1^{(k+1)} \right) + \alpha_{i1}^{(k)} + \frac{1}{T} \sum_t 2 \left(y_{i1t} - p_{i1t}^{(k)} \right)
\end{aligned}$$

Recursively substituting for $\alpha_{i1}^{(k)}$, we can express the update at iteration $k + 1$ as sum of the starting value plus the sum of the average difference between the choices and the predicted probabilities in all previous iterations.

$$\alpha_{i1}^{(k+1)} = \bar{x}_i \left(\beta_1^{(0)} - \beta_1^{(k+1)} \right) + \alpha_{i1}^{(0)} + \sum_{n=0}^k \frac{1}{T} \sum_t 2 \left(y_{i1t} - p_{i1t}^{(n)} \right) \quad (71)$$

By default, the unrestricted MM algorithm updates the fixed effect of the base alternative to the value of the previous iteration plus the average difference between the observed choices and the predicted probabilities, adjusted for any movement in the base slope: $\alpha_{i1}^{(k)} + \frac{1}{T} \sum_t 2 \left(y_{i1t} - p_{i1t}^{(k)} \right) + \bar{x}_i \left(\beta_1^{(k)} - \beta_1^{(k+1)} \right)$. As a result, the MM algorithm will generally produce non-zero updates for the base alternative. There are two ways to impose the normalization of $\alpha_{i1} = 0$. The first is to subtract off the newly calculated base intercept, $\alpha_{i1}^{(k+1)}$, from all $\alpha_{ij}^{(k+1)}$ at each iteration. The other way is to let the algorithm run unconstrained and subtract off the final estimate of α_{i1} from the other fixed effects only after convergence. If the MM algorithm takes K iterations to converge, this approach is equivalent to subtracting off the final accumulated value $\alpha_{i1}^{(K)} = \bar{x}_i \left(\beta_1^{(0)} - \beta_1^{(K)} \right) + \alpha_{i1}^{(0)} + \sum_{n=0}^{K-1} \frac{1}{T} \sum_t 2 \left(y_{i1t} - p_{i1t}^{(n)} \right)$. The second method is generally more efficient as it avoids repeated subtractions.

The normalization does not affect the updates to β because we calculate β using demeaned OLS, $\beta^{(k+1)} = \left(\tilde{\mathbf{x}}' \tilde{\mathbf{x}} \right)^{-1} \tilde{\mathbf{x}} \tilde{\mathbf{v}}^{(k)}$. Normalizing against the base alternative simply shifts the intercept for each individual-alternative pair by a constant α_{i1} . Because the variables are time-demeaned within each individual-alternative group prior to the regression, this constant shift completely differences out, leaving the slope updates entirely unchanged.

Appendix E Alternative Estimation Methods

As described in Section 7.1, the MM algorithm and the rivals we compare it to belong to the family of preconditioned gradient ascent methods, which update the parameters iteratively as:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \sigma_k P_k \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)}) \quad (72)$$

Repeating Equation 31, we can write the MM update as:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + 2(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'(\mathbf{y} - \mathbf{p}^{(k)})$$

Because the term $\mathbf{Z}'(\mathbf{y} - \mathbf{p}^{(k)})$ is exactly the gradient of the multinomial logit log-likelihood, our MM algorithm can be thought of as a deterministic score step modified by a constant preconditioner $P_k = (\mathbf{Z}'\mathbf{Z})^{-1}$. Böhning's MM algorithm shares this same underlying structure in the normalized space, though it uses a different constant metric that incorporates cross-alternative correlations to better approximate true second-order curvature. Ultimately, these iterative methods differ primarily in what type of information they use to construct this preconditioning matrix P_k and how they set the step size σ_k . We compare the alternative methods in this framework below.

E.1 First-Order Methods

First-order methods, also called gradient methods, construct the preconditioner using only information about the gradient. In the case of the multinomial logit, the gradient is easy to compute, so these methods have a low cost per iteration and scale well with the number of fixed effects.

Standard gradient ascent simply uses the identity matrix as the preconditioner ($P_k = \mathbf{I}$) and a constant step size $\sigma_k = \sigma$. This approach can be highly sensitive to the step size. If the step is too large, the method overshoots the maximum, and if

too small, it is slow to converge. Adam effectively constructs a dynamic, diagonal preconditioner; however, rather than applying this preconditioner directly to the current gradient, Adam applies it to an exponentially weighted moving average of past gradients (momentum). It takes an exponentially weighted average over past gradients and squared gradients, updating the parameters as:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \sigma \frac{\frac{\mathbf{m}^{(k)}}{1-\phi_1^k}}{\sqrt{\frac{\mathbf{v}^{(k)}}{1-\phi_2^k} + \epsilon}} \quad (73)$$

where $\mathbf{m}^{(k)}$ is the weighted average of the gradients $\mathbf{m}^{(k)} = \phi_1 \mathbf{m}^{(k-1)} + (1 - \phi_1) \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)})$; $\mathbf{v}^{(k)}$ is the weighted average of the squared gradients $\mathbf{v}^{(k)} = \phi_2 \mathbf{v}^{(k-1)} + (1 - \phi_2) \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)})^2$; σ is the base step size; ϕ_1 and ϕ_2 govern the exponential decay; and ϵ is a hyperparameter to avoid division by zero.

While the diagonal preconditioner in Adam makes each iteration computationally cheap, it ignores structural dependencies between parameters. Furthermore, the choices of the hyperparameters σ, ϕ_1, ϕ_2 , and ϵ heavily affect its performance. While the default values often perform well, they are not appropriate for every problem, and Adam can fail to converge due to a poor choice of hyperparameters (Reddi, Kale, and Kumar, 2019; Zhang et al., 2022). Appendix F describes the choices we use in our simulations. From the perspective of researcher time, tuning the hyperparameters for optimal performance can easily negate any speed advantage. In contrast to first-order methods, the MM algorithm does not depend on any hyperparameters. Because our constant preconditioner is derived from a strict minorizing bound, the ascent property of the MM algorithm guarantees monotonic improvement in the likelihood with a fixed step size, which avoids overshooting and contributes to our algorithm’s robust numerical stability.

E.2 Second-Order Methods

Second-order methods construct the preconditioning matrix using explicit information about the second-order derivatives to capture the curvature of the parameter space. Newton’s method computes the search direction by using the exact inverse

Hessian as its dynamic preconditioner and moves in the direction:

$$P_k \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)}) = -\nabla^2 \mathcal{L}(\boldsymbol{\theta}^{(k)})^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}^{(k)}) \quad (74)$$

When the number of parameters is large, brute-force calculation and inversion the Hessian is costly. The cost of inverting the Hessian scales cubically with the number of parameters.²² With fixed effects, the Hessian has a sparse structure that facilitates direct calculation of the Newton (Hospido, 2012). The form of the multinomial logit imposes additional structure on the Hessian that facilitates direct computation.

As we will show later, even exploiting the structure of the Hessian, each iteration of Newton’s method is considerably more expensive than an iteration of the MM algorithm. Quasi-Newton methods, like BFGS, replace the inverse Hessian in (74) with an approximation. By avoiding the inversion, BFGS is normally less computationally burdensome than Newton’s method. However, due to the structure of the Hessian, we found the BFGS update to be more expensive than directly calculating the Newton step, so we study Newton’s method rather than quasi-Newton methods for the remainder of the paper.

Although second-order methods have a high cost per iteration, they exhibit faster convergence. Newton’s method converges at a quadratic rate, whereas MM algorithms typically converge at a linear rate. The MM algorithm therefore takes more iterations, but each iteration is less costly. We investigate this tradeoff in the next section.

²²Calculating the inversion through Gauss-Jordan elimination has a complexity of $\mathcal{O}(P^3)$ where P is the number of parameters.

Appendix F Implementation Details for Simulation 1

This section describes the implementation details for Newton’s method and Adam used in the simulation in section 7.2.

F.1 Newton’s Method

To better advantage Newton’s method, our implementation of Newton’s method follows Hospido (2012) and exploits structure and sparsity of the Hessian to efficiently calculate the Newton step by making use of block inversion and block matrix multiplication. In addition to exploiting the structure of the Hessian, we tailor the block matrix multiplication to the specific structure of the data in our simulation. All of this allows us to directly compute the Newton step at a low cost while using little memory. We first show the structure of the Hessian and then show how we exploit that structure to calculate the Newton step.

F.1.1 The Gradient and Hessian

For background, recall that the parameters to be estimated are $\boldsymbol{\theta} = (\beta_2, \beta_3, \{\alpha_{i2}, \alpha_{i3}\}_{i=1, \dots, I})$. We rewrite the likelihood function as $l(x_{it}\boldsymbol{\beta} + \boldsymbol{\alpha}_i; \mathbf{y}_{it}) = l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})$. We suppress w_{jt} and γ_i from our notation because we do not use these terms in this simulation. For each individual i and each time period t ,

$$\frac{\partial \log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})}{\partial \boldsymbol{\theta}} = \frac{\partial \log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})}{\partial \boldsymbol{\psi}_{it}} \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} = (\mathbf{y}_{it} - \mathbf{p}_{it})' \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} \quad (75)$$

where $\frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}}$ is a $2 \times (1 + I)$ matrix.

Let $\text{diag}_2(a)$ be a 2×2 matrix with diagonal elements as a and off-diagonal elements

as 0,

$$\begin{aligned} \left[\frac{\partial \psi_{it}}{\partial \boldsymbol{\theta}} \right]_{:,1:2} &= \frac{\partial \psi_{it}}{\partial \boldsymbol{\beta}} = \text{diag}_2(x_{it}) \\ \left[\frac{\partial \psi_{it}}{\partial \boldsymbol{\theta}} \right]_{:,2 \times i + 1:2} &= \frac{\partial \psi_{it}}{\partial \boldsymbol{\alpha}_i} = \text{diag}_2(1) \end{aligned}$$

while the other elements are zero because $\boldsymbol{\alpha}_h$ is not in $l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})$ if $h \neq i$.

Therefore, Eq.(75) can be rewritten as:

$$((y_{i2t} - p_{i2t})x_{it}, (y_{i3t} - p_{i3t})x_{it}, 0, \dots, 0, y_{i2t} - p_{i2t}, y_{i3t} - p_{i3t}, 0, \dots, 0)$$

The gradient of the likelihood $\nabla \mathcal{L}(\theta)$ is:

$$\nabla \mathcal{L}(\theta) = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}}; \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_1}, \dots, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_I} \right) = \quad (76)$$

$$\left(\sum_{i,t} (y_{i2t} - p_{i2t})x_{it}, \sum_{i,t} (y_{i3t} - p_{i3t})x_{it}; \sum_t y_{12t} - p_{12t}, \sum_t y_{13t} - p_{13t}, \dots, \sum_t y_{I2t} - p_{I2t}, \sum_t y_{I3t} - p_{I3t} \right) \quad (77)$$

For ease of notation, we define:

$$H_{\psi_{it}} = \begin{bmatrix} -p_{i2t}(1 - p_{i2t}) & p_{i2t}p_{i3t} \\ p_{i3t}p_{i2t} & -p_{i3t}(1 - p_{i3t}) \end{bmatrix} = \begin{bmatrix} H_{i,11} & H_{i,12} \\ H_{i,21} & H_{i,22} \end{bmatrix} \quad (78)$$

Then the matrix $\frac{\partial^2 \log l(\boldsymbol{\psi}_{it}(\boldsymbol{\theta}); \mathbf{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'}$ can be written as:

$$\begin{bmatrix} H_{\psi_{it}} x_{it}^2 & \mathbf{0} & \cdots & \mathbf{0} & H_{\psi_{it}} x_{it} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ H_{\psi_{it}} x_{it} & \mathbf{0} & \cdots & \mathbf{0} & H_{\psi_{it}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}$$

And then the Hessian matrix $\nabla^2 \mathcal{L}(\theta)$ is:

$$\sum_{i,t} \frac{\partial^2 \log l(\psi_{it}; \mathbf{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} =$$

$$\left[\begin{array}{c|cccccc} \sum_{i,t} H_{\psi_{it}} x_{it}^2 & \sum_t H_{\psi_{1t}} x_{1t} & \sum_t H_{\psi_{2t}} x_{2t} & \cdots & \sum_t H_{\psi_{it}} x_{it} & \cdots & \sum_t H_{\psi_{It}} x_{It} \\ \hline \sum_t H_{\psi_{1t}} x_{1t} & \sum_t H_{\psi_{1t}} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \sum_t H_{\psi_{2t}} x_{2t} & \mathbf{0} & \sum_t H_{\psi_{2t}} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \sum_t H_{\psi_{it}} x_{it} & \mathbf{0} & \mathbf{0} & \cdots & \sum_t H_{\psi_{it}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \sum_t H_{\psi_{It}} x_{It} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \sum_t H_{\psi_{It}} \end{array} \right] \quad (79)$$

We partition the Hessian into blocks according to the lines in (79). The blocks are:

$$A = \sum_{i,t} H_{\psi_{it}} x_{it}^2 = \sum_i A_i$$

$$B = \left[\sum_t H_{\psi_{1t}} x_{1t} \quad \sum_t H_{\psi_{2t}} x_{2t} \quad \cdots \quad \sum_t H_{\psi_{it}} x_{it} \quad \cdots \quad \sum_t H_{\psi_{It}} x_{It} \right]$$

$$= [B_1 \ B_2 \ \cdots \ B_i \ \cdots \ B_I]$$

$$C = B'$$

$$D = \begin{bmatrix} \sum_t H_{\psi_{1t}} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \sum_t H_{\psi_{2t}} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \sum_t H_{\psi_{it}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \sum_t H_{\psi_{It}} \end{bmatrix} \quad (80)$$

$$= \begin{bmatrix} D_1 & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & D_2 & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & D_i & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & D_I \end{bmatrix}$$

F.1.2 Block Inversion

To calculate the inverse of the Hessian $\nabla^2 \mathcal{L}(\boldsymbol{\theta})^{-1}$, we use block inversion. With block inversion, we can write the inverse Hessian as:

$$\nabla^2 \mathcal{L}(\boldsymbol{\theta})^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{bmatrix} \quad (81)$$

Because D is block diagonal, we can find its inverse by inverting each of the 2×2 diagonal matrices D_i . This simplification greatly reduces the computation burden of inversion.

F.1.3 Newton Step

The Newton step is $\Delta\theta = -\sigma\nabla^2\mathcal{L}(\theta)^{-1}\nabla\mathcal{L}$. For the step size, we use $\sigma = 1$. This is the optimal step size in a neighborhood of the solution (Nocedal and Wright, 2006). Outside of that area, the step size must satisfy the Wolfe conditions to guarantee convergence (Nocedal and Wright, 2006). Most implementations of Newton's method start with the unit step size and then use backtracking line search in the event that the Wolfe conditions do not hold. In our simulations, we did not find a need for backtracking so we hard-code the step size as one. The Newton step is simply:

$$\begin{aligned}\Delta\theta &= -\nabla^2\mathcal{L}(\theta)^{-1}\nabla\mathcal{L} \\ &= -\begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\beta} \\ \frac{\partial\mathcal{L}}{\partial\alpha_i} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\alpha_I} \end{bmatrix}\end{aligned}\tag{82}$$

Brute-force multiplication is inefficient and slow. Instead, Hospido (2012) uses block matrix multiplication to simplify the Newton step into:

$$\Delta\beta = -\left(\sum_i (A_i - B_i D_i^{-1} C_i)\right)^{-1} \left(\frac{\partial\mathcal{L}}{\partial\beta} - \sum_i B_i D_i^{-1} \frac{\partial\mathcal{L}}{\partial\alpha_i}\right)\tag{83}$$

and

$$\Delta\alpha_i = -D_i^{-1} \left(\frac{\partial\mathcal{L}}{\partial\alpha_i} + B_i \Delta\beta\right)\tag{84}$$

The matrices A_i , B_i , C_i and D_i^{-1} are all 2×2 . We never construct the full inverse Hessian so this formulation is also memory efficient.

To compute the Newton step, we start by calculating the choice probabilities as

these are the common building elements of both the gradient and the Hessian. With the probabilities, the calculation of the gradient is straightforward and follows (77). We then compute the building blocks of the inverse Hessian.

Rather than construct $H_{\psi_{it}}$ for each individual, we instead construct the individual elements separately using vector operations. We end up with three vectors: one for $H_{i,11}$, one for $H_{i,12}$, and one for $H_{i,22}$. We do not compute $H_{i,21}$ as $H_{i,12} = H_{i,21}$. We then multiply these vectors by the corresponding values of x_{it} and x_{it}^2 and sum over time to obtain the individual elements of the matrices A_i , B_i , and D_i . To sum over time, we make use of fast grouped sums for vectors.

We loop over individuals and piece together the individual vector elements to form the final matrices A_i , B_i and D_i . From these components, we can calculate $\Delta\beta$ using (83). With $\Delta\beta$, we loop over individuals again to calculate the steps for the fixed effects using (84).

F.2 Adam

Our implementation relies on a modified version of 73 suggested by Kingma and Ba (2014). We iterate on:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \sigma^{(k)} \frac{m^{(k)}}{\sqrt{\mathbf{v}^{(k)} + \hat{\epsilon}}} \quad (85)$$

where $\sigma^{(k)} = \sigma \frac{\sqrt{1-\phi_2^k}}{1-\phi_1^k}$ and $\hat{\epsilon} = \epsilon \sqrt{1-\phi_2^k}$. We use the full data set to compute the gradient. This is slightly unusual. Most implementations compute the gradient from a random subsample of data or mini-batch, often consisting of 32 to 512 observations. The small sample reduces the computational burden of each iteration. Mini-batching also improves generalizability by avoiding sharp solutions (Keskar et al., 2017). That is great when the goal is prediction, but here the goal is estimation. In our context, small subsamples lead to slow convergence since each iteration updates only a limited number of individual-alternative fixed effects.

The performance of Adam ultimately depends on the choice of its hyperparameters.

Out of the hyperparameters, the step size σ usually has the most influence on the speed of convergence. We use a step size of $\sigma = 0.1$.

While an exhaustive grid search might yield a better choice, performing a grid search for each trial of the simulation is simply not practical. To determine the step size, we instead started with the recommended step size from Kingma and Ba (2014) of $\sigma = 0.001$ and the considered alternative step sizes based on powers of 10. We evaluated the step sizes 0.001, 0.01, 0.1, and 1 on the 250 datasets generated for $I = 1000$. We capped the maximum number of iterations at 5000. Table 1 shows the mean, minimum, and maximum number of iterations that Adam took before terminating.

Table 1: Adam: Number of Iterations

σ	Mean	Min	Max
1	4961.72	204	5000
0.1	239.06	229	276
0.01	2320.02	1944	2452
0.001	5000	5000	5000

With the default step size $\sigma = 0.001$, Adam fails to converge before reaching the iteration limit of 5000. This step size is simply too small. Larger step sizes reduce the number of iterations needed to achieve convergence. For $\sigma = 0.01$, Adam takes around 2300 iterations and for $\sigma = 0.1$, takes around 230 iterations. Increasing the step size further to 1 results in Adam failing to converge most of the time. Based on powers of 10, we settled a step size $\sigma = 0.1$, which seems to work well in practice.

Appendix G Implementation Details for Simulation 2

This section describes the implementation details for Newton’s method used in the simulation in section 7.3. As in the first simulation, we rely on Hospido (2012). While the individual-specific coefficients change the structure of Hessian, they do not alter the basic approach.

G.1 The Gradient and Hessian

In second simulation, we estimate the parameters $\boldsymbol{\theta} = (\beta_2, \beta_3, \{\gamma_i\}_{i=1, \dots, I})$. As before, we rewrite the likelihood function as $l(x_{it}\boldsymbol{\beta} + w_{jt}\gamma_i; \mathbf{y}_{it}) = l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})$. We suppress the fixed effects α_{ij} as they do not appear in this simulation. For each individual i and each time period t ,

$$\frac{\partial \log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})}{\partial \boldsymbol{\theta}} = \frac{\partial \log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})}{\partial \boldsymbol{\psi}_{it}} \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} = (\mathbf{y}_{it} - \mathbf{p}_{it})' \times \frac{\partial \boldsymbol{\psi}_{it}}{\partial \boldsymbol{\theta}} \quad (86)$$

which reduces to:

$$\left((y_{i2t} - p_{i2t})x_{it}, (y_{i3t} - p_{i3t})x_{it}, 0, \dots, 0, \sum_j (y_{ij t} - p_{ij t})w_{jt}, 0, \dots, 0 \right)$$

The gradient of the likelihood $\nabla \mathcal{L}(\boldsymbol{\theta})$ is therefore:

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}}; \frac{\partial \mathcal{L}}{\partial \gamma_1}, \dots, \frac{\partial \mathcal{L}}{\partial \gamma_I} \right) = \quad (87)$$

$$\left(\sum_{i,t} (y_{i2t} - p_{i2t})x_{it}, \sum_{i,t} (y_{i3t} - p_{i3t})x_{it}, \sum_{j,t} (y_{1jt} - p_{1jt})w_{jt}, \dots, \sum_{j,t} (y_{Ijt} - p_{Ijt})w_{jt} \right) \quad (88)$$

Then the matrix $\frac{\partial^2 \log l(\boldsymbol{\psi}_{it}(\boldsymbol{\theta}); \mathbf{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'}$ can be written as:

$$\begin{bmatrix} -p_{i2t}(1-p_{i2t})x_{it}^2 & p_{i2t}p_{i3t}x_{it}^2 & \dots & \mathbf{0} & (p_{i2t}\bar{w}_{it} - p_{i2t}w_{2t})x_{it} & \mathbf{0} & \dots & \mathbf{0} \\ p_{i3t}p_{i2t}x_{it}^2 & -p_{i3t}(1-p_{i3t})x_{it}^2 & \dots & \mathbf{0} & (p_{i3t}\bar{w}_{it} - p_{i3t}w_{3t})x_{it} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ (p_{i2t}\bar{w}_{it} - p_{i2t}w_{2t})x_{it} & (p_{i3t}\bar{w}_{it} - p_{i3t}w_{3t})x_{it} & \dots & \mathbf{0} & (\bar{w}_{it})^2 - \overline{w_{it}^2} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}$$

where

$$\begin{aligned} \bar{w}_{it} &= \sum_j p_{ij t} w_{jt} \\ \overline{w_{it}^2} &= \sum_j p_{ij t} w_{jt}^2 \end{aligned}$$

And then the Hessian matrix $\nabla^2 \mathcal{L}(\boldsymbol{\theta})$ is:

$$\sum_{i,t} \frac{\partial^2 \log l(\boldsymbol{\psi}_{it}; \mathbf{y}_{it})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad (89)$$

where the individual blocks are The individual blocks are:

$$A = \sum_{i,t} H_{\boldsymbol{\psi}_{it}} x_{it}^2 = \sum_i A_i$$

$$B = \begin{bmatrix} \sum_t (p_{12t} \bar{w}_{it} - p_{12t} w_{2t}) x_{1t} \cdots \sum_t (p_{i2t} \bar{w}_{it} - p_{i2t} w_{2t}) x_{it} \cdots \sum_t (p_{I2t} \bar{w}_{it} - p_{I2t} w_{2t}) x_{It} \\ \sum_t (p_{13t} \bar{w}_{it} - p_{13t} w_{3t}) x_{1t} \cdots \sum_t (p_{i3t} \bar{w}_{it} - p_{i3t} w_{3t}) x_{it} \cdots \sum_t (p_{I3t} \bar{w}_{it} - p_{I3t} w_{3t}) x_{It} \\ \vdots \\ \sum_t (p_{1It} \bar{w}_{it} - p_{1It} w_{It}) x_{1t} \cdots \sum_t (p_{iIt} \bar{w}_{it} - p_{iIt} w_{It}) x_{it} \cdots \sum_t (p_{IIt} \bar{w}_{it} - p_{IIt} w_{It}) x_{It} \end{bmatrix}$$

$$= [B_1 \cdots B_i \cdots B_I]$$

$$C = B'$$

$$D = \begin{bmatrix} \sum_t (\bar{w}_{1t})^2 - \overline{w_{1t}^2} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \sum_t (\bar{w}_{2t})^2 - \overline{w_{2t}^2} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \sum_t (\bar{w}_{it})^2 - \overline{w_{it}^2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \sum_t (\bar{w}_{It})^2 - \overline{w_{It}^2} \end{bmatrix} \quad (90)$$

$$= \begin{bmatrix} d_1 & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & d_2 & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & d_i & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & d_I \end{bmatrix}$$

Note that the matrix D is a diagonal matrix so inverting it is trivial, which makes block inversion an efficient strategy. We can apply the same formula as in (81).

G.2 Newton Step

The Newton step is:²³

$$\begin{aligned}\Delta\theta &= -\nabla^2\mathcal{L}(\theta)^{-1}\nabla\mathcal{L} \\ &= -\begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\beta} \\ \frac{\partial\mathcal{L}}{\partial\gamma_i} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\gamma_I} \end{bmatrix}\end{aligned}\tag{91}$$

From Hospido (2012), the Newton step simplifies to:

$$\Delta\beta = \left(\sum_i (A_i - \frac{1}{d_i}B_iC_i)\right)^{-1} \left(\frac{\partial\mathcal{L}}{\partial\beta} - \sum_i \frac{1}{d_i}B_i\frac{\partial\mathcal{L}}{\partial\gamma_i}\right)\tag{92}$$

and

$$\Delta\gamma_i = \frac{1}{d_i} \left(\frac{\partial\mathcal{L}}{\partial\gamma_i} + B_i'\Delta\beta\right)\tag{93}$$

As in the previous simulation, we first compute the individual components and then find $\Delta\beta$ which we use to calculate each $\Delta\gamma_i$.

Appendix H Performance of MCM Algorithm

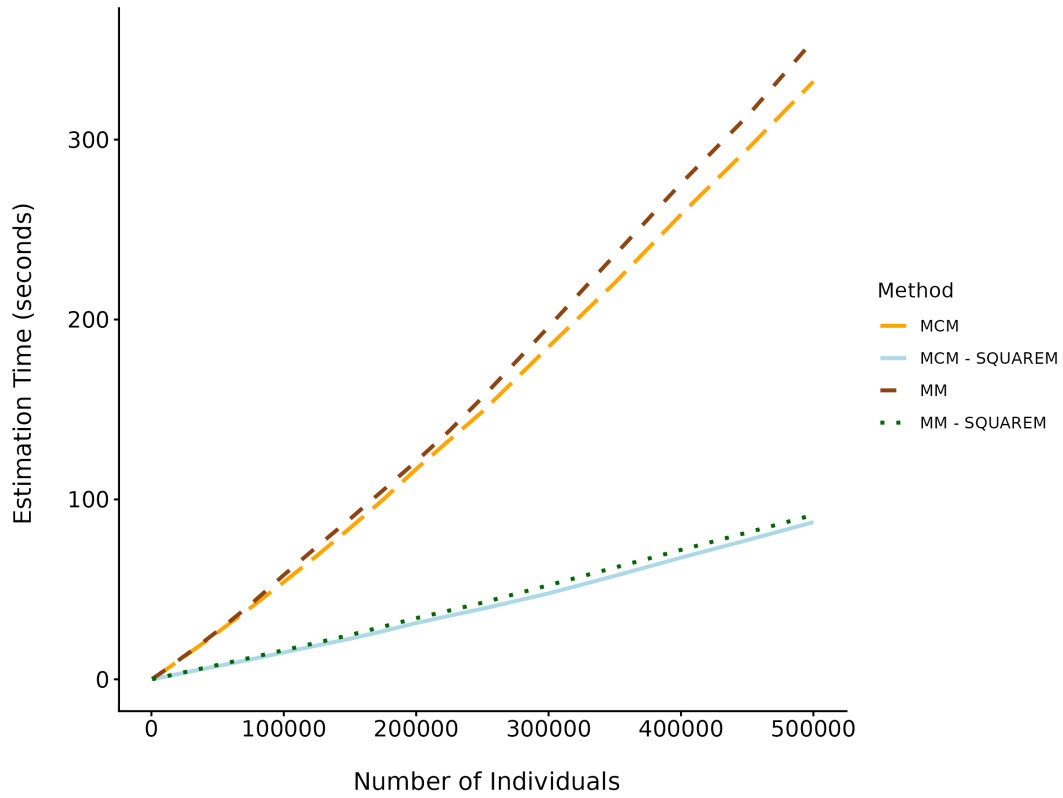
This section compares the performance of the MM algorithm to the MCM algorithm with and without acceleration for the two simulations. Both algorithms give identical point estimates in both simulations.

H.1 Simulation 1

Figure 9 shows the runtimes for each algorithm. In the simulation, the MCM algorithm is slightly faster than its MM variant, but both algorithms offer similar performance. There is a fair amount of variation across runs so the MCM algorithm is not always faster than the MM algorithm.

²³As in the previous simulation, we set $\sigma = 1$.

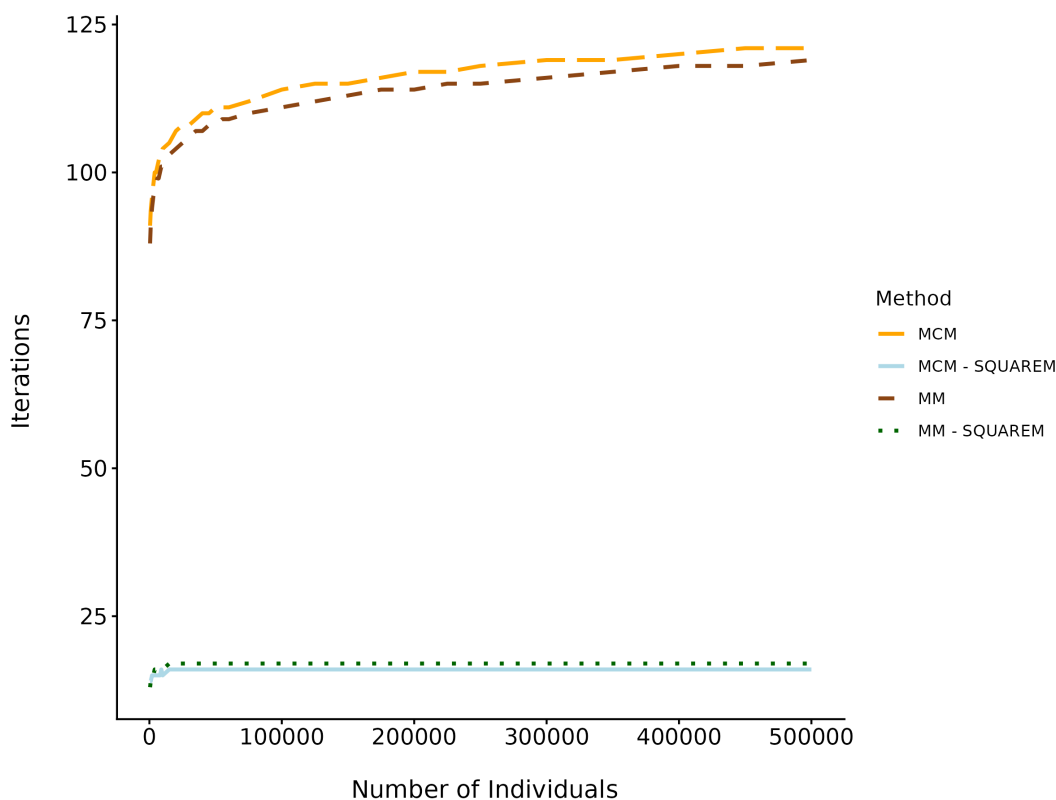
Figure 9



Notes: The lines shows the median runtime.

Comparing the number of iterations, we see that both algorithms take a similar number of iterations to converge. The non-accelerated version of the MCM algorithm takes slightly more iterations, but not many. Cross-terms are not important here so the MCM algorithm will take similar steps to the MM algorithm.

Figure 10

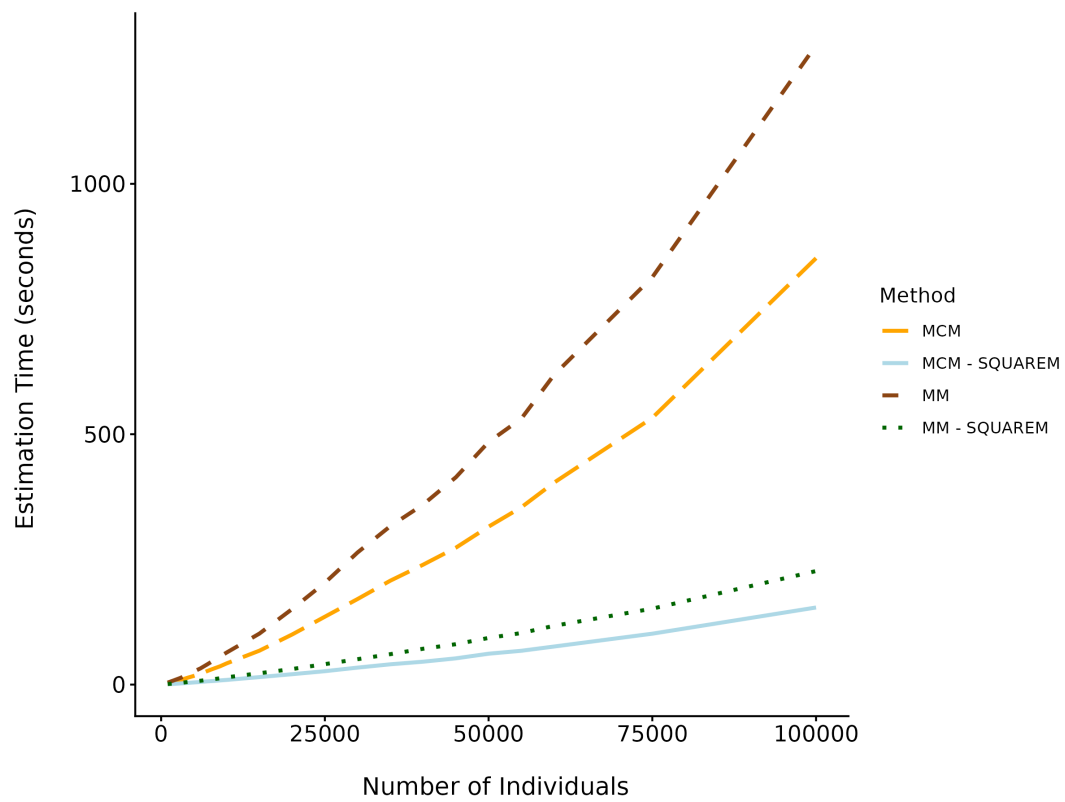


Notes: The lines show the median number of iterations until convergence.

H.2 Simulation 2

The MCM algorithm outperforms the MM algorithm although the difference narrows after acceleration. Both algorithms take the same number of iterations on average. In our well-behaved simulations, cross-terms are not important since variables are drawn from i.i.d. normals. For this simulation, the differences in speed mainly comes from the cost of the respective update steps. The generalized demeaning used in the update for the MM algorithm is more expensive than the simple subtraction used in the MCM update. For cases with individual-specific updates, conditional maximization offers more of a benefit than with fixed effects estimation.

Figure 11



Notes: The lines show the median runtimes.

References

- Angrist, J. D. and J.-S. Pischke (2010). “The Credibility Revolution in Empirical Economics: How Better Research Design Is Taking the Con out of Econometrics”. In: *Journal of Economic Perspectives* 24, pp. 3–30.
- Barrette, E., G. Gowrisankaran, and R. Town (2022). “Countervailing Market Power and Hospital Competition”. In: *The Review of Economics and Statistics* 104, pp. 1351–1360.
- Bergé, L. R., K. Butts, and G. McDermott (2026). “Fast and user-friendly econometrics estimations: The R package *fixest*”. In: *arXiv preprint arXiv:2601.21749*.
- Böhning, D. (1992). “Multinomial Logistic Regression Algorithm”. In: *Annals of the Institute of Statistical Mathematics* 44, pp. 197–200.
- Böhning, D. and B. G. Lindsay (1988). “Monotonicity of Quadratic-Approximation Algorithms”. In: *Annals of the Institute of Statistical Mathematics* 40, pp. 641–663.
- Brot, Z., Z. Cooper, S. V. Craig, and L. Klarnet (2024). “Is There Too Little Antitrust Enforcement in the US Hospital Sector?” In: *American Economic Review: Insights* 6, pp. 526–42.
- Capps, C., L. Kmitch, Z. Zabinski, and S. Zayats (2018). “The Continuing Saga of Hospital Merger Enforcement”. In: *Antitrust LJ* 82, p. 441.
- Chamberlain, G. (1980). “Analysis of Covariance with Qualitative Data”. In: *Review of Economic Studies* 47, pp. 225–238.
- Chauveau, D. and D. R. Hunter (2013). “ECM and MM algorithms for normal mixtures with constrained parameters”. In.
- Chen, M. (2019). *Estimation of Nonlinear Panel Models with Multiple Unobserved Effects*. Unpublished manuscript, University of Warwick.
- Correia, S. (2017). “Linear models with high-dimensional fixed effects: An efficient and feasible estimator”. In: *Working Paper*.
- D’Haultfœuille, X. and A. Iaria (2016). “A Convenient Method for the Estimation of the Multinomial Logit Model with Fixed Effects”. In: *Economics Letters* 141, pp. 77–79.
- Dhaene, G. and K. Jochmans (2015). “Split-panel Jackknife Estimation of Fixed-effect Models”. In: *Review of Economic Studies* 82, pp. 991–1030.
- Du, T., A. Kanodia, and S. Athey (2023). “Torch-Choice: A PyTorch Package for Large-Scale Choice Modelling with Python”. In: *arXiv preprint arXiv:2304.01906*.

- Du, Y. and R. Varadhan (2020). “SQUAREM: An R Package for Off-the-Shelf Acceleration of EM, MM and Other EM-Like Monotone Algorithms”. In: *Journal of Statistical Software* 92.7, pp. 1–41. DOI: 10.18637/jss.v092.i07. URL: <https://www.jstatsoft.org/index.php/jss/article/view/v092i07>.
- Dubois, P., R. Griffith, and M. O’Connell (2020). “How Well Targeted are Soda Taxes?” In: *American Economic Review* 110, pp. 3661–3704.
- Fong, D. C.-L. and M. Saunders (2011). “LSMR: An iterative algorithm for sparse least-squares problems”. In: *SIAM Journal on Scientific Computing* 33.5, pp. 2950–2971.
- Garmon, C. (2017). “The accuracy of hospital merger screening methods”. In: *The RAND Journal of Economics* 48.4, pp. 1068–1102.
- Gaure, S. (2013). “OLS with multiple high dimensional category variables”. In: *Computational Statistics & Data Analysis* 66, pp. 8–18.
- Gaynor, M. S., S. A. Kleiner, and W. B. Vogt (2013). “A structural approach to market definition with an application to the hospital industry”. In: *The Journal of Industrial Economics* 61.2, pp. 243–289.
- Gowrisankaran, G., A. Nevo, and R. Town (2015). “Mergers When Prices Are Negotiated: Evidence from the Hospital Industry.” In: *American Economic Review* 105, pp. 172–203. ISSN: 00028282. URL: <http://search.ebscohost.com.ezproxy.bu.edu/login.aspx?direct=true&db=ecn&AN=1467554&site=ehost-live&scope=site>.
- Greene, W. H. (2018). *Econometric Analysis*. 8th. Prentice Hall.
- Guimaraes, P., O. Figueirdo, and D. Woodward (2003). “A Tractable Approach to the Firm Location Decision Problem”. In: *The Review of Economics and Statistics*, pp. 201–204.
- Guimaraes, P. and P. Portugal (2010). “A simple feasible procedure to fit models with high-dimensional fixed effects”. In: *The Stata Journal* 10.4, pp. 628–649.
- Hahn, J. and W. Newey (2004). “Jackknife and Analytical Bias Reduction for Nonlinear Panel Models”. In: *Econometrica* 72, pp. 1295–1319.
- Hinz, J., A. Hudle, and J. Wanner (2019). *Separating the Wheat from the Chaff: Fast Estimation of GLMs with High-Dimensional Fixed Effects*. Unpublished manuscript, European University Institute.
- Ho, K. and A. Pakes (2014). “Hospital Choices, Hospital Prices, and Financial Incentives to Physicians”. In: *American Economic Review* 104, pp. 3841–3884.

- Ho, K. (2006). “The welfare effects of restricted hospital choice in the US medical care market”. In: *Journal of Applied Econometrics* 21.7, pp. 1039–1079. DOI: <https://doi.org/10.1002/jae.896>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.896>.
- Hospido, L. (2012). “Estimating nonlinear models with multiple fixed effects: A computational note”. In: *Oxford Bulletin of Economics and Statistics* 74.5, pp. 760–775.
- Hunter, D. R. and K. Lange (2004). “A Tutorial on MM Algorithms”. In: *The American Statistician* 58, pp. 30–37.
- James, J. (2017). “MM Algorithm for General Mixed Multinomial Logit Models”. In: *Journal of Applied Econometrics* 32, pp. 841–857.
- Kaczmarz, S. (1993). “Approximate solution of systems of linear equations”. In: *International Journal of Control* 57.6, pp. 1269–1271. DOI: 10.1080/00207179308934446. eprint: <https://doi.org/10.1080/00207179308934446>. URL: <https://doi.org/10.1080/00207179308934446>.
- Keskar, N. S., D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang (2017). *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. arXiv: 1609.04836 [cs.LG].
- Kingma, D. P. and J. Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.
- Lange, K. (2013). *Optimization*. Second Edition. New York: Springer.
- Lange, K. (2016). *MM Optimization Algorithms*. SIAM.
- Lange, K., D. R. Hunter, and I. Yang (2000). “Optimization transfer using surrogate objective functions”. In: *Journal of Computational and Graphical Statistics* 9, pp. 1–20.
- Leeuw, J. de and K. Lange (2009). “Sharp Quadratic Majorization in One Dimension”. In: *Computational Statistics & Data Analysis* 53, pp. 2471–2484.
- Li, Y., L. Balzano, D. Needell, and H. Lyu (2026). “Convergence and complexity of block majorization-minimization for constrained block-Riemannian optimization”. In: *Journal of Machine Learning Research* 27.42, pp. 1–77.
- McFadden, D. and K. Train (2000). “Mixed MNL models for discrete response”. In: *Journal of Applied Econometrics* 15, pp. 447–470.
- Meng, X.-L. and D. B. Rubin (1993). “Maximum likelihood estimation via the ECM algorithm: A general framework”. In: *Biometrika* 80.2, pp. 267–278.

- Mugnier, M. and A. Wang (2024). *Fixed Effects Nonlinear Panel Models with Heterogeneous Slopes: Identification and Consistency*. SSRN Paper No. 4186349.
- Nocedal, J. and S. J. Wright (2006). *Numerical optimization*. Springer.
- Raval, D., T. Rosenbaum, and S. A. Tenn (2017). “A semiparametric discrete choice model: An application to hospital mergers”. In: *Economic Inquiry* 55.4, pp. 1919–1944.
- Raval, D., T. Rosenbaum, and N. E. Wilson (2022). “Using disaster-induced closures to evaluate discrete choice models of hospital demand”. In: *The RAND Journal of Economics* 53.3, pp. 561–589.
- Razaviyayn, M., M. Hong, and Z.-Q. Luo (2013). “A unified convergence analysis of block successive minimization methods for nonsmooth optimization”. In: *SIAM Journal on Optimization* 23.2, pp. 1126–1153.
- Reddi, S. J., S. Kale, and S. Kumar (2019). “On the convergence of Adam and beyond”. In: *arXiv preprint arXiv:1904.09237*.
- Ruud, P. A. (1991). “Extensions of Estimation Methods Using the EM Algorithm”. In: *Journal of Econometrics* 49, pp. 305–341.
- Rysman, M., S. Wang, and K. Wozniak (2025). *Empirics of Payment Method Choice in Scanner Data*. Unpublished manuscript, Boston University.
- Sahai, H. (2023). “The Welfare Implications of School Voucher Design: Evidence from India”. In: .
- Shum, M., W. Song, and X. Shi (2018). “Estimating Semi-parametric Panel Multinomial Choice Models using Cyclic Monotonicity”. In: *Econometrica* 86, pp. 737–761.
- Somainsi, P. and F. A. Wolak (2016). “An algorithm to estimate the two-way fixed effects model”. In: *Journal of Econometric Methods* 5.1, pp. 143–152.
- Stammann, A. (2018). *Fast and Feasible Estimation of Generalized Linear Models with High-Dimensional k-way Fixed Effects*. Unpublished manuscript, arXiv:1707.01815.
- Stammann, A., F. Heiß, and D. McFadden (2016). *Estimating Fixed Effects Logit Models with Large Panel Data*. Beiträge zur Jahrestagung des Vereins für Socialpolitik 2016: Demographischer Wandel - Session: Microeconometrics, No. G01-V3. URL: <http://hdl.handle.net/10419/145837>.
- Tanner, M. (1996). *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*. Springer.

- Tenn, S. and S. Vandergrift (2017). “Geographic Market Definition in Urban Hospital Mergers: Lessons from the Advocate-NorthShore Litigation”. In: *Antitrust Source* 17.3, pp. 1–10.
- Tseng, P. (2001). “Convergence of a block coordinate descent method for nondifferentiable minimization”. In: *Journal of optimization theory and applications* 109.3, pp. 475–494.
- Varadhan, R. and C. Roland (2008). “Simple and globally convergent methods for accelerating the convergence of any EM algorithm”. In: *Scandinavian Journal of Statistics* 35.2, pp. 335–353.
- Zhang, Y., C. Chen, N. Shi, R. Sun, and Z.-Q. Luo (2022). “Adam can converge without any modification on update rules”. In: *Advances in Neural Information Processing Systems* 35, pp. 28386–28399.
- Zhou, H., D. Alexander, and K. Lange (2011). “A quasi-Newton acceleration for high-dimensional optimization algorithms”. In: *Statistics and Computing* 21, pp. 261–273.