An MM Algorithm for General Mixed Multinomial Logit Models*

Jonathan James†

California Polytechnic State University

March 10, 2016

Abstract

This paper develops a new technique for estimating mixed logit models with a simple minorization-maximization (MM) algorithm. The algorithm requires minimal coding and is easy to implement for a variety of mixed logit models. Most importantly, the algorithm has a very low cost per iteration relative to current methods, producing substantial computational savings. In addition, the method is asymptotically consistent, efficient and globally convergent.

Keywords: Discrete Choice, Mixed Logit, EM Algorithm, MM Algorithm

JEL Classification Codes: C02, C13, C25, C35

*Thanks to Peter Arcidiacono, Fabian Lange, Kenneth Train, Thierry Magnac, and three anonymous referees for helpful comments.

†Department of Economics, Orfalea College of Business, California Polytechnic State University, San Luis Obispo, CA 93407. Email: jjames04@calpoly.edu. For updated versions of this paper please visit my homepage: http://www.calpoly.edu/~jjames04
1 Introduction

Accommodating unobserved heterogeneity is critical for studying discrete response models and more importantly in using these models to construct counterfactuals. The mixed logit model is a flexible discrete choice framework that allows the unobserved error structure of the utility maximization problem to be arbitrarily correlated across choices and time. In theory, the mixed logit model is extremely powerful as McFadden and Train (2000) show that, with a flexible enough error structure, it can approximate any random utility model with arbitrary degree of accuracy.

However, estimating mixed logit models is challenging because this more complicated error structure produces a difficult to maximize integrated likelihood function. Maximizing the likelihood with quasi-Newton methods is unattractive because they are either difficult to implement, only being applicable to models where it is realistic to derive and code problem-specific analytical gradients, or they have a high cost per iteration, either through the computation of the analytical gradient or conducting many evaluations of the likelihood to numerically approximate the gradient. An alternative approach for solving difficult maximization problems is to transfer optimization to simpler surrogate functions (Lange et al., 2000). The use of surrogate functions is the engine for the expectation-maximization (EM) algorithm (Dempster et al., 1977), a popular alternative to quasi-Newton methods which has been adopted for estimating mixed logit models (Bhat, 1997; Train, 2007, 2008). The EM algorithm forms a surrogate function that is an expected complete data likelihood. This makes the EM algorithm very easy to implement because maximizing the surrogate function only requires complete data maximum likelihood which often has a closed form solution. Train (2007) illustrates this simplicity with an EM algorithm for estimating a mixed logit model with normally distributed random coefficients that entails iteratively computing a sample mean and a sample covariance.
While the EM algorithm is unmatched in its simplicity, its application to mixed logit models has a major shortcoming. In many situations, maximizing the surrogate function requires finding the maximum to a standard (complete data) logit model, which does not have a closed form solution, requiring numerical optimization, in some cases multiple numerical optimizations, at each iteration\footnote{In the latent class random coefficients model, each iteration of the EM algorithm requires solving a separate standard logit for each of the latent classes\cite{Bhat,Train}.} This expensive numerical search at each iteration of the outer search algorithm produces iterations that are extremely slow and may even rule out the EM algorithm as a method to estimate these models.

The contribution of this paper is to establish a new surrogate function for estimating mixed logit models that characteristically has a simple closed form solution. This new approach extends the simplicity and flexibility of the EM algorithm but produces iterations that are substantially faster. With minor adjustments, the algorithm can be applied to a broad set of mixed logit models because the existence of the new surrogate function is based off of the nature of discrete choice models, not the form of the unobserved heterogeneity.

The method constitutes a minorization-maximization (MM) algorithm\cite{Lange et al., Hunter and Lange}, which makes it consistent, efficient, and globally convergent. The minorization is accomplished with a simple quadratic lower bound derived from B"ohning and Lindsay\cite{Bohning and Lindsay}, which makes it consistent, efficient, and globally convergent. The primary strength of the MM algorithm is that the cost per iteration is very low, making the algorithm potentially much faster than other methods. This paper uses Monte Carlo experiments to compare the computational performance of the MM algorithm with two other popular methods: the EM algorithm and maximum likelihood with a quasi-Newton algorithm. In all of these experiments, the MM algorithm was 5 to 8 times faster than the EM algorithm, which suggests that replacing the EM algorithm in practice with the MM algorithm will result in significant computational savings. Compared to the quasi-Newton
algorithm, the MM algorithm is most effective with panel data, where the fast iterations combine with a faster rate of convergence for the MM algorithm (McLachlan and Krishnan 2007, Page 101). In the Monte Carlo experiments with panel data, the MM algorithm was typically as fast and in some cases faster than the quasi-Newton algorithm. Given its simple implementation and strength in panel data settings, the MM algorithm is ideal for more complex mixed logit models where analytical gradients are difficult to derive and Fiebig et al. (2010, Footnote 12) remark that identification is likely only possible with repeated observations.

A secondary benefit of the MM algorithm is that the maximum of the surrogate function is attained with easy to compute sufficient statistics, which enables estimation of a broader set of models that are not possible with the EM algorithm. In particular, when the error structure of the mixed logit model has a continuous distribution, estimation necessitates numerically simulating the integrated likelihood function. This requires computations on a much larger simulated dataset. The challenge facing the EM algorithm is that the maximum of its surrogate function lacks a sufficient statistic and requires numerical optimization over the entire simulated dataset. Storing the simulated dataset is not feasible in many situations and may force researchers to reduce the number of simulation draws. This is problematic for complex models that require many simulation draws to overcome the curse of dimensionality associated with numerical integration (Cherchi and Guevara 2012, Chiu and Walker 2007). In contrast, the MM algorithm has very low storage demands because the maximum of its surrogate function uses simple sufficient statistics that are additive across data points and do not require storing the simulated dataset. Furthermore, the use of sufficient statistics conveniently facilitates parallel processing, which the EM algorithm cannot. In this way, the MM algorithm is more tractable than the EM algorithm because it can be implemented faster with many orders of magnitude more simulation draws.

This paper proceeds in Section 2 by developing the MM algorithm for a mixed logit
model with both fixed coefficients and normally distributed random coefficients. To illustrate further applications of the algorithm, Appendix A extends the MM algorithm to a scale heterogeneity model (Fiebig et al., 2010), and Appendix B extends the MM algorithm to a latent class model. Section 3 provides a discussion of the strengths of the MM algorithm relative to other estimators. Section 4 studies the performance of the MM algorithm through Monte Carlo experiments. Finally, Section 5 concludes.

2 An MM Algorithm for Mixed Logit Models

The MM algorithm developed in this paper can be applied to a variety of mixed logit models. To develop the intuition and insight of the method, this section focuses on its application to a mixed logit model with fixed coefficients and normally distributed random coefficients. Let the utility for individual $i$, from making choice $j$, in choice situation $t$, be characterized as $U_{itj} = x_{itj}'\alpha + z_{itj}'\beta_i + \varepsilon_{itj}$, for $j = 1, 2, \ldots, J$ and $t = 1, 2, \ldots, T$. The choice-specific characteristics are separated into $x_{itj}$, which have fixed utility weights $\alpha$, and $z_{itj}$, which have individual specific utility weights $\beta_i$. The taste heterogeneity, $\beta_i$, is unobserved and in this model assumed to be distributed in the population multivariate normal with mean $\gamma$ and full covariance $\Delta$, i.e. $\beta_i \sim \mathcal{N}(\gamma, \Delta)$. Individuals make the choice that yields the highest utility, so the discrete outcome $d_{itj} = 1$ is observed if $j = \arg\max_{k \in \{1, \ldots, J\}} U_{itk}$ and zero otherwise.

Under the assumption that the unobserved random utility shock, $\varepsilon_{itj}$, is independent across individuals, choices and time and distributed type-I extreme value, McFadden (1974) provide a survey of different types of mixing models that have been used in the literature.
shows that the choice probability for outcome $j$ conditional on $\beta_i = \beta$ is

$$P_{itj}(\alpha, \beta) = \frac{\exp(x_{itj}'\alpha + z_{itj}'\beta)}{\sum_{k=1}^J \exp(x_{itk}'\alpha + z_{itk}'\beta)}$$

(1)

For any observed sequence of choices for individual $i$, the conditional (on unobserved $\beta$) likelihood is a product of the choice probabilities,

$$L_i(\alpha, \beta) = \prod_{t=1}^T \prod_{j=1}^J P_{itj}(\alpha, \beta)^{d_{itj}}.$$  

Letting $f(\beta|\gamma, \Delta)$ denote the probability density function for a multivariate normal distribution with mean $\gamma$ and covariance $\Delta$, the log-likelihood function is defined by integrating each of the individual conditional likelihoods over all possible values of the unobserved data,

$$LL(\Psi) = \sum_{i=1}^N \ln \left[ \int_{\beta} L_i(\alpha, \beta) f(\beta|\gamma, \Delta) d\beta \right],$$

where $\Psi = \{\alpha, \gamma, \Delta\}$. The integral in the log-likelihood does not have a known closed form expression and is typically simulated numerically in estimation.\(^3\) Because the log operator is non-linear, when estimating the model using maximum simulated likelihood (MSL), a sufficiently large number of simulation draws is required to reduce the simulation bias (Lee, 1995).

Rather than trying to directly optimize $LL(\Psi)$, the strategy of the MM algorithm is to transfer optimization to a simpler surrogate function that guarantees ascent of the log-likelihood. The MM algorithm refines the surrogate function put forth by the EM algorithm\(^4\) The EM surrogate function is denoted by $Q(\Psi|\Psi^m)$ and bounds the log-likelihood around a particular value of the parameters, $\Psi^m$. Train (2008) shows that the EM surrogate function for mixed logit models takes the form

$$Q(\Psi|\Psi^m) = \sum_{i=1}^N \int_{\beta} \ln \left[ L_i(\alpha, \beta) f(\beta|\gamma, \Delta) \right] h_i(\beta|\Psi^m) d\beta - \text{constant}$$

This function is an expected complete data log-likelihood. In this expression, $h_i(\beta|\Psi^m)$ is

\(^3\)Alternatively, Harding and Hausman (2007) and Bhat (2011) develop closed form approximations that can be used with certain distributions to avoid simulation.

an individual-specific probability density function of the unobserved data, $\beta_i$, conditional on individual $i$’s observed data and the parameters taking the value $\Psi^m$ and is defined using Bayes’ rule:

$$ h_i(\beta | \Psi^m) = \frac{L_i(\alpha^m, \beta) f(\beta | \gamma^m, \Delta^m)}{\int_{\beta'} L_i(\alpha^m, \beta') f(\beta' | \gamma^m, \Delta^m) d\beta'} $$

Evaluating the integral in the EM surrogate function requires simulation. Given the difficulty of drawing values of $\beta$ from $h_i(\beta | \Psi^m)$, [Train (2007)] proposes drawing values of $\beta$ from $f(\beta | \gamma^m, \Delta^m)$ and weighting these draw by $L_i(\alpha^m, \beta) / [\int_{\beta'} L_i(\alpha^m, \beta') f(\beta' | \gamma^m, \Delta^m)]$. The normalizing constant in the denominator is the integrated likelihood function, which as mentioned cannot be calculated explicitly. However, it can be approximated just as it is approximated when estimating the model with MSL. To simulate $Q(\Psi | \Psi^m)$, take $R$ draws for each individual from $\mathcal{N}(\gamma^m, \Delta^m)$, each labeled $\beta_{ir}$, and construct the weight for each draw using the conditional likelihood functions:

$$ w_{ir}^m = \frac{L_i(\alpha^m, \beta_{ir})}{\sum_{r'=1}^{R} L_i(\alpha^m, \beta_{ir'})} $$

The superscript $m$ on the weights indicates that they are a function of $\Psi^m$. As with maximum simulated likelihood, $R$ must be sufficiently large to eliminate the bias from simulating the denominator of the weights ([Train 2007]).

---

5 Technically the denominator of this weight should be multiplied by $(1/R)$ as it is in [Train 2007]. This is omitted here so that the weights for each individual sum to 1, and the sample size remains $N$ instead of $N \times R$. 

---

7
Using the approximation to $h_i(\beta|\Psi^m)$, the EM surrogate function becomes

$$Q(\Psi|\Psi^m) = \sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \ln [L_i(\alpha, \beta_{ir}) f(\beta_{ir}|\gamma, \Delta)] - \text{constant}$$

$$= \sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \ln [L_i(\alpha)] + \sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \ln [f(\beta_{ir}|\gamma, \Delta)] - \text{constant}$$

By exchanging the expectation and log operator in the original log-likelihood function, maximizing the EM surrogate function is much simpler for two reasons. First, the log operator makes the surrogate function additively separable in the parameters in the utility function and the parameters for the distribution of taste heterogeneity. This leads to a much simpler maximization problem over separate independent functions, rather than a single joint maximization. Second, maximization only requires complete data maximum likelihood that often has a closed form expression. For example, $Q(\gamma, \Delta|\Psi^m)$ is the likelihood function for a multivariate normal distribution with weighted observations on $\beta$, where the solution for the mean and covariance is simply the sample analogs [Train 2007]. So, $\gamma^{m+1}, \Delta^{m+1} = \text{argmax}_{\gamma, \Delta} Q(\gamma, \Delta|\Psi^m)$ has a closed form solution

$$\gamma^{m+1} = \left[ \frac{\sum_{i=1}^{N} \left( \sum_{r=1}^{R} w_{ir}^m \beta_{ir} \right) }{N} \right] / N$$

$$\Delta^{m+1} = \left[ \frac{\sum_{i=1}^{N} \left( \sum_{r=1}^{R} w_{ir}^m \beta_{ir} \beta_{ir}' \right) }{N - \gamma^{m+1} \gamma^{m+1}'} \right] / N$$

If the estimates for the parameters in the mixing distribution have a closed form in the complete data problem, then they will also have a closed form in the EM algorithm. [Train 2008] underscores this appeal by developing EM estimators for many non-parametric mixing distributions, each of which require iteratively computing different sample moments. The
drawback of the EM algorithm is that the other part of the surrogate function, \( Q(\alpha|\Psi^m) \), which represents a weighted standard (complete data) logit log-likelihood with fixed utility weights \( \alpha \), cannot be maximized in closed form and necessitates numerical optimization. Furthermore, in some models, each iteration of the EM algorithm requires solving multiple standard logits\(^6\). Because this function has to be maximized at every step of the EM algorithm, the computational burden can be overwhelming.

A lack of a closed form solution to the surrogate function is a major shortcoming of the EM algorithm and in some settings inhibits the use of the EM algorithm all together\(^7\). This paper derives an alternative surrogate function, \( \tilde{Q}(\Psi|\Psi^m) \), that has a closed form solution for these other parameters. The main insight is to exploit a natural feature of discrete choice models and recognize that \( Q(\alpha|\Psi^m) \), the weighted standard logit log-likelihood, can be minorized with a quadratic lower bound function. This new surrogate function will be linear in parameters and have a simple closed form solution for the maximum. Modifying the EM surrogate function in this way produces a minorization-maximization (MM) algorithm (Hunter and Lange, 2004), which is sketched in Figure 1. Beginning with \( \Psi^m \), then choosing \( \Psi^{m+1} = \arg\max_{\Psi} \tilde{Q}(\Psi|\Psi^m) \), and repeating will guarantee convergence to the maximum of the log-likelihood.

The new surrogate function is formed by using the lower bound quadratic approximation in Böhning and Lindsay (1988, Theorem 4.1) that shows that a function can be bounded below by a quadratic approximation if there exists a global lower bound to the second

\(^6\)See Footnote 1.

\(^7\)Numerically optimizing \( Q(\alpha|\Psi^m) \) requires repeated computations on the entire simulated dataset. When the simulated dataset cannot be stored in memory, then not only will the weights in Eq. 2 need to be computed at each iteration of the EM algorithm, they will also need to be computed at each iteration of the inner numerical search.
Figure 1: Sketch of MM Algorithm

The quadratic approximation requires the gradient and lower bound of the Hessian for the original EM objective function. It is evaluated at \( \alpha^m \) to maintain the equality \( LL(\Psi^m) = Q(\Psi^m | \Psi^m) = \tilde{Q}(\Psi^m | \Psi^m) \). The maximum of the new surrogate function is \( \alpha^{m+1} = \alpha^m - [B^m]^{-1}g^m \), which is a single Newton step where the Hessian is replaced by its lower bound. This step is guaranteed to improve the likelihood, and the algorithm can continue to the next iteration.

Since \( Q(\alpha | \Psi^m) \) is a weighted standard logit, the gradient and Hessian evaluated at \( \alpha = \alpha^m \)
The requirement on the lower bound matrix of the Hessian is that it is global with respect to \( \alpha \). This means that it cannot be a function of the probabilities \( P_{itj}(\alpha^m, \beta_{ir}) \). However, it can be a function of the weights and simulated draws because they are treated as observed data in the maximization. The insight in Böhning and Lindsay (1988) is that, in discrete choice models, the choice probabilities are bounded, i.e. \( P_{itj}(\alpha^m, \beta_{ir}) \in [0, 1] \), which makes possible a bound on the Hessian. As an example, in the binary logit, the Hessian is defined as

\[
H = \sum_i \left[ -P_i (1 - P_i) \right] x_i x_i' - \frac{1}{J} \left( \sum_{j=1}^J x_{itj} \right) \left( \sum_{j=1}^J x_{itj} \right)'
\]

(3)

Because the weights sum to one, the lower bound is the same at every iteration, so it can be computed and inverted outside of the algorithm.

Table 1 outlines the MM algorithm for the mixed logit model with fixed coefficients and

\[
\begin{align*}
g^m &= \sum_{i=1}^N \sum_{r=1}^R w_{ir}^m \left[ \sum_{t=1}^T \sum_{j=1}^J x_{itj} \left( d_{itj} - P_{itj}(\alpha^m, \beta_{ir}) \right) \right] \\
H^m &= -\sum_{i=1}^N \sum_{r=1}^R w_{ir}^m \left[ \sum_{t=1}^T \left( \sum_{j=1}^J x_{itj} x'_{itj} P_{itj}(\alpha^m, \beta_{ir}) \right) - \left( \sum_{j=1}^J x_{itj} P_{itj}(\alpha^m, \beta_{ir}) \right) \left( \sum_{j=1}^J x_{itj} P_{itj}(\alpha^m, \beta_{ir}) \right)' \right] \\
\end{align*}
\]

(4)
Table 1: MM Algorithm: random and fixed coefficient model

- Outside of the algorithm
  - Compute $B^{-1}$ using Eq. (4)
  - Initialize with starting values $\Psi^0 = \{\alpha^0, \gamma^0, \Delta^0\}$
- Repeat the following iteration until converged, i.e. $||\Psi^{m+1} - \Psi^m||_\infty < \kappa$
  
  **Step 1:** Simulate and calculate weights from likelihood components
  - Given $\gamma^m$ and $\Delta^m$, for each $i$, draw $R$ values of $\beta$ from $N(\gamma^m, \Delta^m)$ labeled $\beta_{ir}$
  - Calculate $P_{itj}(\alpha^m, \beta_{ir})$ for each $\beta_{ir}$ using Eq. (1)
  - Calculate $L_i(\alpha^m, \beta_{ir}) = \prod_{t=1}^{T} \prod_{j=1}^{J} P_{itj}(\alpha^m, \beta_{ir})^{d_{itj}}$
  - Using $L_i(\alpha^m, \beta_{ir})$, compute weights $w_{ir}^m$ using Eq. (2)

  **Step 2:** Update parameters
  - $\gamma^{m+1} = \frac{\sum_{i=1}^{N} \left( \sum_{r=1}^{R} w_{ir}^m \beta_{ir} \right)}{N}$
  - $\Delta^{m+1} = \frac{\sum_{i=1}^{N} \left( \sum_{r=1}^{R} w_{ir}^m \beta_{ir} \beta_{ir}' \right)}{N - \gamma^{m+1} \gamma^{m+1}'}$
  - $\alpha^{m+1} = \alpha^m - B^{-1} g^m$
    - $g^m = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j=1}^{J} x_{itj} \left( d_{itj} - \hat{P}_{itj}^m \right)$
    - $\hat{P}_{itj}^m = \sum_{r=1}^{R} w_{ir} P_{itj}(\alpha^m, \beta_{ir})$

multivariate normal random coefficients. Standard errors can be computed with bootstrapping or from an estimate of the information matrix (Ruud, 1991). The information matrix is estimated by taking the cross product of the individual simulated scores. Train (2007) shows the components of the simulated scores for $\gamma$ and $\Delta$. The simulated score for $\alpha$ is given in Table 1 as $\sum_{t=1}^{T} \sum_{j=1}^{J} x_{itj} (d_{itj} - \hat{P}_{itj})$, where $\hat{P}_{itj}$ uses the formula in Table 1 at the solution. Having now outlined the general idea and implementation of the MM algorithm, the next section provides a deeper discussion of the algorithm in practice.

---

8If the seed for the random number generator is fixed, then the algorithm will converge deterministically, and traditional convergence criteria can be used. Otherwise the sequence of parameters will form a Markov Chain. For further discussion see Nielsen (2000).
3 Discussion

Understanding when the MM algorithm should be employed instead of other estimators, e.g., the EM algorithm or MSL with a quasi-Newton algorithm, is an important question. The iterative procedure outlined in Table 1 showcases three main advantages of the MM algorithm. First, in many applications, the lower bound matrix used to update the parameters will only need to be computed and inverted once outside of the algorithm. In contrast, both the EM algorithm and MSL will require the repeated computation and inversion (or at least rank one update of the inverted Hessian) at each iteration. Avoiding this potentially costly and repeated matrix inversion can result in substantial computational savings. This makes the MM algorithm attractive when the dimension of $x$ is large, e.g. models with many market or product specific fixed effects. The second benefit of the algorithm is that $g^m$, which is used for the parameter update, is a function of the same choice probabilities used to calculate the weights. Because the choice probabilities are reused, the iterations of the MM algorithm are extremely fast, requiring about one evaluation of the simulated likelihood function in total.\(^9\) With the EM algorithm, the choice probabilities will be evaluated multiple times within the inner iteration which will only be used to update $\alpha$. The MM algorithm has the advantage that every time the choice probabilities are calculated, all of the parameters are updated. The final benefit of the algorithm is that it provides a sufficient statistic for each of the parameter updates. Updating $\alpha$ only requires $g^m$, which can be written as the sum of the individual contributions, $g^m = \sum_{i=1}^N g_i^m$. This is useful because updating the parameters does not require storing all of the weights, and the iteration can be written looping over individuals. Since the order of the sum does not matter, this is favorable to parallel processing.

Carrying out the MM algorithm in other settings requires identifying a suitable lower

\(^9\)The denominator of the weight in Eq. (2) is a simulated likelihood.
bound matrix. The lower bound matrix in Eq. (4) did not depend on the form of the unobserved heterogeneity, so it can be used for any functional form assumption on the mixing distribution. Using similar ideas for the creation of the bound in Eq. (4), lower bound matrices have been derived for other discrete choice models that can be incorporated into the MM algorithm as well: Böhning (1992) derives a lower bound for multinomial logit models and Böhning (1999) derives a bound for probit models. In general, the existence of a simple lower bound only depends on the utility function being linear in parameters, even if the parameters are interacted with the unobserved taste heterogeneity in a non-linear way. If the parameters in the utility function are not interacted with the taste heterogeneity, as is the case with fixed coefficients in the utility function, then the lower bound matrix will be the same at every iteration of the algorithm, so it can be computed and inverted prior to estimation. In cases where the unobserved heterogeneity is interacted with the parameters in the utility function, there will likely be a different lower bound matrix at each iteration. However, typically the main components of the lower bound matrix can be computed outside of the algorithm and only need to be re-weighted and inverted at each iteration, which is computationally light compared to repeatedly calculating the actual Hessian. Appendix A and Appendix B extend the MM algorithm to a scale heterogeneity model (Fiebig et al., 2010) and a latent class model which illustrates how to construct the lower bound matrix when the parameters in the utility function interact with the taste heterogeneity.

For models whose utility function has a combination of fixed parameters and parameters interacted with the unobserved taste heterogeneity, the MM algorithm can be implemented either using a single lower bound for all of the parameters to update them simultaneously every iteration, or employing a conditional-maximization (CM) approach (Meng and Rubin, 1993). With the CM approach, the fixed parameters are updated in one iteration with a lower bound matrix that is inverted outside of the algorithm, and the parameters that are interacted with the taste heterogeneity are updated with a different lower bound matrix.
in the next iteration, making sure that the weights and the distributional parameters are updated at every iteration. The later approach is more favorable if the number of fixed coefficients is large since the inversion of the lower bound matrix for these parameters would only need to be done once.

One potential concern with the MM algorithm is that taking parameter steps that are smaller than the EM algorithm may lead to too many iterations. Rai and Matthews (1993) argue against this point showing that methods that only partially maximize the EM surrogate function at each iteration typically yield a minimal number of extra iterations and can produce significant computational savings. For the MM algorithm, the number of extra iterations will depend on two factors. First, the global rate of convergence for the algorithm is determined by the component with the slowest rate of convergence (Meng, 1994). For mixed logit models, the parameters for the distribution of taste heterogeneity will likely dictate the overall number of iterations. Since the MM algorithm makes a full update on these parameters and is only partially updating the fixed coefficients, the effect of this partial updating is likely to be negligible. The second factor is how close the lower bound matrix is to the actual Hessian. Because the lower bound replaces all of the choice probabilities with 1/2, the distance from the actual Hessian will depend on how far these probabilities are from 1/2. If many individuals have probabilities that are close to one, then the MM algorithm will require more iterations. Given the low cost per iteration of the MM algorithm, it is highly unlikely that these extra iterations will lead to an overall increase in computation time.

Finally, while the MM algorithm offers a fast and tractable method for maximizing the likelihood, it does not directly address the other major challenge of mixed logit models which

---

10An alternative to the MM algorithm for partially maximizing the EM surrogate function, $Q(\alpha|\Psi^m)$, would be to take a single scaled Newton step (Rai and Matthews, 1993). The scaling is necessary to maintain monotonicity since Newton-Raphson is an unstable algorithm and a single cycle does not guarantee that the objective function will be improved (Meng and Rubin, 1993). This approach is likely inferior because this requires an additional calculation of all of the choice probabilities to check for an increase in the likelihood at the new parameters, which also requires storage of all of the weights and simulated data. In addition, it requires the repeated computation and inversion of the actual Hessian.
is the curse of dimensionality associated with numerical integration. In comparison to methods like [Harding and Hausman (2007)] and [Bhat (2011)] that use closed form expressions to approximate the integrated likelihood function, the MM algorithm requires numerical simulation. While the closed form approximation methods provide more tractability by avoiding simulation, they require coding beyond simple complete data maximum likelihood calculations, and they are, so far, limited in their applications. Even though the MM algorithm still relies on numerical simulation, by providing an algorithm with very fast iterations, which also makes possible massively parallel processing, it can alleviate the problem of the curse of dimensionality by realistically accommodating substantially more simulation draws than other methods.

4 Monte Carlo Experiments

This section uses simulated data from the normally distributed random coefficients model in Section 2 to compare the computational performance of the MM algorithm with two other methods: the EM algorithm and maximum simulated likelihood (MSL). The first set of experiments has two random coefficients and varies the number of fixed coefficients from 0, 1, 10, 100, and 1,000. Table 2 reports for each estimator the total number of iterations to convergence, the number of inner iterations for estimators with any inner sub-routine, and the total computation time in seconds for each model. The MM algorithm and the EM

---

11 The data is generated so that the total variance of the explained portion of utility is constant across models. The regressors, $x$ and $z$, are drawn from a standard normal distribution. The two random coefficients are normally distributed with mean zero, variance 0.5 and correlation 0.25, which sets $\text{Var}(z' \beta) = 1$. For each model, the $\alpha$’s are chosen so that $\text{Var}(x' \alpha) = 1$. For the model with a single fixed coefficient, $\alpha = 1$. For the model with 10, each element of $\alpha$ equals $1/\sqrt{10}$. For the models with 100 and 1,000 coefficients, each element of $\alpha$ is set to $1/\sqrt{10}$, and the vector $x$ is created so that it only has 10 nonzero elements randomly chosen for each observation.

12 For strict comparison, the same convergence criteria was applied to each of the estimators, which was when the change in the parameters was within a pre-specified tolerance, i.e. $\| \Psi^{m+1} - \Psi^m \|_\infty < 1e^{-6}$. For the surrogate based estimators, the vector epsilon algorithm [Kuroda and Sakakihara (2006)] was applied to accelerate convergence, which is detailed in Appendix C.
algorithm converge to the same solution because they solve the same problem. Train (2007) notes that because simulation enters the optimization problem of the EM algorithm and MSL differently, for a finite number of simulation draws, they will not converge to exactly the same solution but will be identical as the number of draws are increased. These experiments used 1,000 pseudorandom draws for the integral, such that all methods essentially converged to the same answer.

Panel A of Table 2 shows the results for the MM algorithm. Since the MM algorithm has no inner subroutine, only the total iterations and total computation time are reported. As a baseline, the model with zero fixed parameters takes 21 seconds. Adding a single fixed coefficient increases the computation time to 32 seconds. As expected, increasing the number of fixed parameters had very little effect on computation time, where the model with 1,000 fixed coefficients took on average 42 seconds.

Panels B and C in Table 2 show the results for the EM algorithm using two different methods for the inner optimization: The first is the Newton-Raphson algorithm that requires the gradient and Hessian in the repeated estimation of the standard logit, and the second is a quasi-Newton algorithm that only requires the gradient with the inverse of the Hessian being approximated with the BFGS algorithm. It is ambiguous which method should be used in practice. The Newton-Raphson algorithm has quadratic convergence, requiring fewer inner iterations, while the quasi-Newton algorithm offers faster iterations because it does not require calculating or inverting the actual Hessian. In the model with no fixed coefficients, the EM algorithm does not require any inner numerical optimization, so it is identical to the MM algorithm. Including a single fixed coefficient introduces numerical optimization, which required on average 82 Newton steps for the EM algorithm with Newton-Raphson and 202 inner iterations for the EM algorithm with the quasi-Newton optimizer, resulting in an 8 and 7 fold increase in computation time respectively. Strikingly, including even a single parameter that requires numerical optimization has a deleterious affect on the EM
Table 2: Two Random Coefficients and Various Number of Fixed Coefficients

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Inner Iterations</td>
<td>Time (sec.)</td>
<td>Function Evaluations</td>
<td>Time (sec.)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>100</td>
<td>1,000</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>39</td>
<td>39</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>21s</td>
<td>32s</td>
<td>32s</td>
<td>33s</td>
<td>42s</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>33</td>
<td>33</td>
<td>34</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>82</td>
<td>82</td>
<td>87</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>21s</td>
<td>185s</td>
<td>184s</td>
<td>200s</td>
<td>349s</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>202</td>
<td>294</td>
<td>314</td>
<td>416</td>
</tr>
<tr>
<td></td>
<td>21s</td>
<td>145s</td>
<td>202s</td>
<td>218s</td>
<td>364s</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>11</td>
<td>16</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>14</td>
<td>24</td>
<td>22</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>25s</td>
<td>28s</td>
<td>50s</td>
<td>45s</td>
<td>137s</td>
</tr>
<tr>
<td></td>
<td>84</td>
<td>111</td>
<td>401</td>
<td>2602</td>
<td>60209</td>
</tr>
<tr>
<td></td>
<td>55s</td>
<td>73s</td>
<td>262s</td>
<td>1713s</td>
<td>39611s</td>
</tr>
</tbody>
</table>

Note: \( N = 1,000, \ T = 10 \) and \( J = 3 \). Each model uses 1,000 pseudorandom numbers for simulation. Each experiment is replicated 20 times and averages are reported. The replications where conducted on a shared computing cluster. To benchmark the computation time, each experiment was run five times on an isolated computer with a 2.7 GHz 12-Core Intel Xeon E5 processor to calculate the average time per function evaluation and then multiplied by the average number of function evaluations for the 20 replications. The exception is MSL with numerical gradients and 1,000 fixed coefficients. This model was only run one iteration and the information was then multiplied by the number of iterations required for MSL with analytical gradients. All models use the estimates from a standard logit as starting values.
algorithm, regardless of how the inner optimization is carried out. The MM algorithm always required a few more total iterations than the EM algorithm, but because it avoids the inner numerical optimization, the MM algorithm was always significantly faster, with computational speedups increasing in the number of fixed parameters: 4 to 5 times for the smaller model and 8 to 9 times for the biggest model.

Panels D and E in Table 2 show the results for MSL using quasi-Newton algorithms with analytical and numerical gradients and the BFGS algorithm to approximate the Hessian. For MSL with analytical gradients, the number of function evaluations differs from the number of iterations because the algorithm had to occasionally scale the parameter step to ensure monotonicity. For MSL with numerical gradients, the number of function evaluations includes the number of times the likelihood needed to be evaluated to approximate the gradient. Not surprisingly, MSL with analytical gradients typically required fewer iterations than the MM algorithm because it has a faster rate of convergence. However, the iterations of the MM algorithm where much faster, which resulted in smaller total computation times in many of the specifications. In fact, in the model with 1,000 fixed coefficients, the quasi-Newton algorithm had difficulty approximating the 1,005 by 1,005 Hessian, resulting in more iterations than the MM algorithm and taking three times longer.

Table 3 extends the baseline experiment with 10 fixed coefficients (Column 3 in Table 2) to compare the estimators under different specifications of the model: changing the number of choices, changing the number of random coefficients, and using a shorter panel and cross-sectional data. This table restricts attention to the MM algorithm, the EM algorithm using Newton-Raphson for the inner optimization and MSL with analytical gradients. Only the computation time in seconds and total number of iterations are reported.

Panel A in Table 3 studies the effect of increasing the choice set, expanding it from $J = 3$ to $J = 6$ and $J = 12$. When the choice set is increased, the EM algorithm converges in fewer iterations. This is expected since the rate of convergence of EM based methods
Table 3: Alternative Specifications: Time in seconds [number of iterations]

<table>
<thead>
<tr>
<th></th>
<th>MM Algorithm</th>
<th>EM Algorithm, Newton-Raphson inner optimization</th>
<th>MSL, quasi-Newton algorithm with analytical gradients</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Panel A:</strong> Changing ( J ) (( N = 1,000, T = 10, 2 ) random coefficients, 10 fixed coefficients)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( J = 6 )</td>
<td>50s [35]</td>
<td>279s [24]</td>
<td>52s [16]</td>
</tr>
<tr>
<td>( J = 12 )</td>
<td>100s [38]</td>
<td>587s [20]</td>
<td>58s [17]</td>
</tr>
<tr>
<td><strong>Panel B:</strong> Changing number of random coefficients (( N = 1,000, T = 10, J = 3, 10 ) fixed coefficients)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 rand coef</td>
<td>44s [51]</td>
<td>263s [49]</td>
<td>40s [15]</td>
</tr>
<tr>
<td>4 rand coef</td>
<td>59s [67]</td>
<td>336s [65]</td>
<td>40s [17]</td>
</tr>
<tr>
<td><strong>Panel C:</strong> Changing ( T ) (( N = 1,000, J = 3, 2 ) random coefficients, 10 fixed coefficients)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T = 5 )</td>
<td>35s [80]</td>
<td>198s [77]</td>
<td>47s [16]</td>
</tr>
<tr>
<td>( T = 1 )</td>
<td>559s [3343]</td>
<td>939s [3334]</td>
<td>10s [20]</td>
</tr>
</tbody>
</table>

Note: Each model uses 1,000 pseudorandom numbers for simulation. Each experiment is replicated 20 times and averages are reported. The replications were conducted on a shared computing cluster. To benchmark the computation time, each experiment was run five times on an isolated computer with a 2.7 GHz 12-Core Intel Xeon E5 processor to calculate the average time per function evaluation and then multiplied by the average number of function evaluations for the 20 replications. All models use the estimates from a standard logit as starting values.
depends on how informative the observed data is about the unobserved data (McLachlan and Krishnan 2007, Page 101) and increasing the choice set provides more information about the unobserved taste heterogeneity. For the MM algorithm, this effect is counterbalanced because a large \( J \) necessarily increases the distance between the lower bound matrix and the actual Hessian, resulting in smaller steps at each iteration and more total iterations\(^\text{13}\). Consequently, when \( J = 3 \), the MM algorithm required 20% more iterations than the EM algorithm, when \( J = 6 \), the MM algorithm required 50% more iterations, and when \( J = 12 \), the MM algorithm required nearly twice as many iterations. Despite requiring more iterations, the MM algorithm remained 5 times faster than the EM algorithm, even when \( J = 12 \) because the time per iteration is substantially faster.

Panel B of Table 3 shows the results for increasing the number of random coefficients in the model from 2 to 3 and 4\(^\text{14}\). As the dimension of the unobserved data increases, the EM algorithm, and the MM algorithm took longer to converge. This is expected since the rate of convergence for these methods is slower with a greater fraction of unobserved data. Consequently, while the MM algorithm was faster than MSL with only two random coefficients, MSL performed slightly better than the MM algorithm for 3 and 4 random coefficients. Comparing the EM algorithm and the MM algorithm, both required more iterations, but the MM algorithm was consistently 5 times faster.

Finally, Panel C in Table 3 changes the number of observations per individual by looking at a shorter panel (\( T = 5 \)) and cross-sectional data (\( T = 1 \)). The results when \( T = 5 \) are very consistent with the results from \( T = 10 \) in Table 2. The MM algorithm was 5 times faster than the EM algorithm and slightly faster than MSL. However, as predicted,

\(^{13}\text{In the lower bound matrix, all of the choice probabilities are substituted with the value 1/2, which becomes cruder for larger } J \text{ because most of the choice probabilities will be very far from 1/2.}\)

\(^{14}\text{To preserve the variance of the explained portion of utility across models, with three random coefficients the variance for each random coefficient was set to 1/3, and for four they were set to 0.25. The mean and correlation for these random coefficients remained at 0 and 0.25 respectively, as they were in the previous models. This set } \text{Var}(z' \beta) = 1 \text{ for each model.}\)
the EM algorithm and the MM algorithm both performed poorly relative to MSL with the cross-sectional data. When there is only a single observation per individual, the data is less informative about the individual’s unobserved preferences, and the surrogate based estimators suffer from a slow rate of convergence.

The experiments show that the repeated numerical optimization is a significant limitation of the conventional EM algorithm. The MM algorithm addresses this problem by proposing a closed form updating that results in very fast iterations and significant computational savings. In fact, in certain situations, models with many fixed coefficients or many repeated observations per individual, the MM algorithm was as fast or faster than MSL with analytical gradients.

There are three additional aspects of the MM algorithm that are not captured in these experiments. First, for the random coefficients model studied in this paper, the steps for the MM algorithm in Table 1 are much simpler to code than the analytical gradient for MSL and the Newton-Raphson algorithm for the EM algorithm. This aspect is not accounted for in a strict comparison of computation times. Second, efficiently implementing the EM algorithm hinges on the ability to store all of the simulated data and weights to update the parameters, which was possible in these experiments. In situations where this is not possible, the computational gains from the MM algorithm relative to the EM algorithm will be substantially larger because these large storage demands are not necessary for the MM algorithm. Finally, while the normally distributed random coefficients model is an important and popular framework, the simplicity of implementing the MM algorithm suggests it would be most advantageously applied to models where analytical gradients for the log-likelihood are too complex and the only alternative is numerical gradient based methods. Given the burden of numerically approximating the gradient, the computational savings of the MM algorithm could be substantial.\[15\] For example, the MM algorithm was almost 1,000 times faster.
5 Conclusion

The use of simple surrogate functions is a powerful tool for maximizing complicated likelihood functions. The EM algorithm is a popular surrogate function method. However, the EM algorithm has a major shortcoming when applied to the estimation of mixed logit models which is that it requires the repeated numerical optimization of a standard logit model at each iteration of the algorithm. This paper proposes an MM algorithm that uses a new surrogate function that can be maximized in closed form, avoiding costly repeated numerical optimization. This innovation results in an estimator that is very easy to implement and has a much lower cost per iteration than most other methods, offering substantial savings in coding and computation time. In the Monte Carlo experiments, the MM algorithm was 5 to 8 times faster than the EM algorithm and in some panel data settings faster than the much more difficult to implement quasi-Newton algorithm.

One useful extension of the estimator would be to situations where other outcome variables are observed in addition to the discrete choice data, for example, hybrid choice models (Ben-Akiva et al., 2002) or quasi-structural models (Bernal and Keane, 2010). Not only do these models introduce many more diverse parameters but the unobserved heterogeneity appears in multiple outcome equations which makes implementing quasi-Newton algorithms even more difficult. The MM algorithm is ideal for this setting because the surrogate function will be additively separable in many of the parameters, maximizing the surrogate function will only require complete data maximum likelihood, and the iterations of the algorithm will
be very fast. By providing a simple and fast method to maximize mixed logit models, the MM algorithm significantly broadens the types of behavioral choice models that researchers can estimate.

References


A MM algorithm for scale heterogeneity

In the scale heterogeneity model [Fiebig et al., 2010], preference heterogeneity is modeled by using an individual-specific parameter to equally scale up or down a constant set of utility weights. The utility function can be written as,

\[ U_{itj} = z'_{itj} [s(\phi_i)\beta] + \varepsilon_{itj}, \]

where \( s(\phi_i) \) is a transformation function of the unobserved variable \( \phi_i \), and \( \phi_i \) is assumed to be normally distributed in the population, i.e. \( \phi_i \sim \mathcal{N}(\gamma, \delta) \). The output of \( s(\phi_i) \) scales the utility weights, \( \beta \), up or down in a constant proportion. Typically, the function \( s(\cdot) \) is chosen so that the scale parameter is positive. Fiebig et al. (2010) choose \( s(\cdot) = \exp(\cdot) \) so that the scale parameter is log-normal, but other functions are possible, for example, \( s(\cdot) = \max(0, \cdot) \) would produce a scale parameter that is distributed from a censored normal, or \( s(\cdot) = \exp(\cdot)/(1 + \exp(\cdot)) \) would generate a scale parameter that is distributed between 0 and 1 from Johnson’s \( S_B \) distribution. The scale heterogeneity model is not identified without normalizing the mean of \( \phi \), so we will fix \( \gamma = \overline{\phi} \).

Given a type-I extreme value assumption on \( \varepsilon \), the choice probability conditional on the unobserved \( \phi \) is

\[ P_{itj}(\beta, \phi) = \exp(z'_{itj}s(\phi)\beta)/\left[ \sum_{k=1}^{J} \exp(z'_{itk}s(\phi)\beta) \right]. \]

The MM algorithm seeks to find a solution to

\[ LL(\Psi) = \sum_{i=1}^{N} \ln \left[ \int \phi L_i(\beta, \phi) f(\phi|\gamma, \delta) d\phi \right], \]

where \( f(\phi|\gamma, \delta) \) is the probability density function of the normal distribution with mean \( \overline{\phi} \) and variance \( \delta \), and the conditional likelihood is defined as \( L_i(\beta, \phi) = \prod_{t=1}^{T} \prod_{j=1}^{J} [P_{itj}(\beta, \phi)]^{d_{itj}} \).

To apply the MM algorithm, we use the current guess of the parameters, \( \beta^m \) and \( \delta^m \), and begin with the simulated EM surrogate function

\[ Q(\Psi|\Psi^m) = \sum_{i=1}^{N} \sum_{r=1}^{R} w^m_{ir} \ln \left[ L_i(\beta, \phi_{ir}) \right] + \sum_{i=1}^{N} \sum_{r=1}^{R} w^m_{ir} \ln \left[ f(\phi_{ir}|\overline{\phi}, \delta^m) \right] - \text{constant} \]

where there are \( R \) draws labeled \( \phi_{ir} \) for each individual that is from a normal distribution, \( \mathcal{N}(\overline{\phi}, \delta^m) \), and the weights are defined as \( w^m_{ir} = L_i(\beta^m, \phi_{ir})/\left[ \sum_{r'=1}^{R} L_i(\beta^m, \phi_{ir'}) \right]. \)
The maximum of $Q(\delta|\Psi^m)$ has a closed form solution. However, maximizing $Q(\beta|\Psi^m)$ would require numerical optimization. To implement the MM algorithm we need to define the new surrogate function, which requires the gradient and the lower bound matrix of the Hessian. The gradient and Hessian of $Q(\beta|\Psi^m)$ evaluated are $\beta = \beta^m$ are:

$$g^m = \sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \left[ \sum_{t=1}^{T} \sum_{j=1}^{J} s(\phi_{ir}) z_{itj} \left( d_{itj} - P_{itj}(\beta^m, \phi_{ir}) \right) \right]$$

$$H^m = -\sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \left\{ \sum_{t=1}^{T} \left[ \sum_{j=1}^{J} \left( s(\phi_{ir})^2 z_{itj} z_{itj}' P_{itj}(\beta^m, \phi_{ir}) \right) \right] - \left( \sum_{j=1}^{J} s(\phi_{ir}) z_{itj} P_{itj}(\beta^m, \phi_{ir}) \right) \right\}$$

We can apply a similar lower bound to the one in Eq. (4).

$$B^m = -\frac{1}{2} \sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \left\{ \sum_{t=1}^{T} \left[ \sum_{j=1}^{J} \left( s(\phi_{ir})^2 z_{itj} z_{itj}' \right) \right] - \frac{1}{J} \left( \sum_{j=1}^{J} s(\phi_{ir}) z_{itj} \right) \left( \sum_{j=1}^{J} s(\phi_{ir}) z_{itj} \right)' \right\}$$

$$= \sum_{i=1}^{N} \left[ \sum_{r=1}^{R} w_{ir}^m s(\phi_{ir})^2 \right] \left\{ -\frac{1}{2} \sum_{t=1}^{T} \left[ \sum_{j=1}^{J} \left( z_{itj} z_{itj}' \right) \right] - \frac{1}{J} \left( \sum_{j=1}^{J} z_{itj} \right) \left( \sum_{j=1}^{J} z_{itj} \right)' \right\}$$

Unlike the lower bound matrix in Section 2, this lower bound is a function of the unobserved data, $\phi$, so it must be computed and inverted at each iteration of the algorithm. However, the computation can be simplified drastically by computing $B_i$ outside of the algorithm and simply re-weighting this matrix at each iteration using $s(\phi_i)^2$. Table 4 outlines the MM algorithm for the scale heterogeneity model.
Table 4: MM Algorithm: scale heterogeneity model

- Outside of the algorithm
  - For each $i = 1, 2, \ldots, N$, compute and store $B_i$ using Eq. 5
  - Initialize with starting values $\Psi^0 = \{\beta^0, \delta^0\}$
- Repeat the following iteration until converged, i.e. $||\Psi^{m+1} - \Psi^m||_\infty < \kappa$

**Step 1:** Simulate and calculate weights from likelihood components
- Given $\gamma$ and $\delta^m$, for each $i$, draw $R$ values of $\phi$ from $N(\gamma, \delta^m)$ labeled $\phi_{ir}$
- Compute $P_{itj}(\beta^m, \phi_{ir}) = \exp(z_{itj}'s(\phi_{ir})\beta^m)/\sum_{k=1}^{J} \exp(z_{itk}'s(\phi_{ir})\beta^m)$
- Compute $L_i(\beta^m, \phi_{ir}) = \prod_{t=1}^{T} \prod_{j=1}^{J} [P_{itj}(\beta^m, \phi_{ir})]^{d_{itj}}$
- Compute $w_{ir}^m = L_i(\beta^m, \phi_{ir})/\sum_{r'=1}^{R} L_i(\beta^m, \phi_{ir'})$

**Step 2:** Update parameters
- $\gamma$ is not updated since it is a normalization
- $\delta^{m+1} = \left[\sum_{i=1}^{N} \left( \sum_{r=1}^{R} w_{ir}^m \phi_{ir}^2 \right) \right] / N - \hat{\gamma} \hat{\gamma}' - \gamma \hat{\gamma}' + \gamma'$
- $\hat{\gamma}' = \left[\sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \phi_{ir} \right] / N$
- $\beta^{m+1} = \beta^m - \left[\sum_{i=1}^{N} s(\phi_i)\beta B_i\right]^{-1} \mathbf{g}^m$
  - $\mathbf{g}^m = \sum_{i=1}^{N} \sum_{r=1}^{R} w_{ir}^m \sum_{t=1}^{T} \sum_{j=1}^{J} s(\phi_{ir})z_{itj} (d_{itj} - P_{itj}(\beta^m, \phi_{ir}))$
  - $s(\phi_i)^2 = \sum_{r=1}^{R} w_{ir}^m s(\phi_{ir})^2$

B MM algorithm for latent class

In the latent class model, individual decision makers are affiliated with an unobserved class where all individuals in that class have the same preferences. The utility function can be written as, $U_{itj} = z'_{itj}\beta_r + \varepsilon_{itj}$, for individual $i$ if they belong to class $r$ for $r = 1, \ldots, R$.

The choice probabilities and conditional likelihood if the individual belong to class $r$ are defined as $P_{itj}(\beta_r) = \exp(z'_{itj}/\sum_{k=1}^{J} \exp(z'_{itk}/\beta_r))$ and $L_i(\beta_r) = \prod_{t=1}^{T} \prod_{j=1}^{J} [P_{itj}(\beta_r)]^{d_{itj}}$. The MM algorithm seeks to find a solution to $LL(\Psi) = \sum_{i=1}^{N} \ln[\sum_{r=1}^{R} \pi_r L_i(\beta_r)]$, where the unobserved data is the class assignment and the parameters to be estimated are each of the class specific utility weights, $[\beta_1, \beta_2, \ldots, \beta_R]$ and the share in the population belonging to each class, $[\pi_1, \pi_2, \ldots, \pi_R]$. 
To apply the MM algorithm, we use the current guess of the parameters, $\beta_1^m, \beta_2^m, \ldots, \beta_R^m$ and $\pi_1^m, \pi_2^m, \ldots, \pi_R^m$, and form the weights $w_{ir}^m = \pi_r^m L_i(\beta_r^m) / \left[ \sum_{r'=1}^R \pi_{r'}^m L_i(\beta_{r'}^m) \right]$. The EM surrogate function is

$$Q(\Psi|\Psi^m) = \sum_{i=1}^N w_{i1}^m \ln [L_i(\beta_1)] + \cdots + \sum_{i=1}^N w_{iR}^m \ln [L_i(\beta_R)] + \sum_{i=1}^N \sum_{r=1}^R w_{ir}^m \ln [\pi_r] - \text{constant}$$

The maximum of $Q(\pi|\Psi^m)$ has a closed form solution. However, maximizing $Q(\beta_r|\Psi^m)$ for each $r = 1, \ldots, R$ would require $R$ separate numerical optimizations. To implement the MM algorithm we need to define the new surrogate function, for each $Q(\beta_r|\Psi^m)$ which requires the gradient and the lower bound matrix of the Hessian. The gradient and Hessian of each $Q(\beta_r|\Psi^m)$ are

$$g_{ir}^m = \sum_{i=1}^N w_{ir}^m \left[ \sum_{t=1}^T \sum_{j=1}^J z_{itj} (d_{itj} - P_{itj}(\beta_r^m)) \right]$$

$$H_{ir}^m = -\sum_{i=1}^N w_{ir}^m \left\{ \sum_{t=1}^T \left[ \sum_{j=1}^J (z_{itj} z'_{itj} P_{itj}(\beta_r^m)) - \left( \sum_{j=1}^J z_{itj} P_{itj}(\beta_r^m) \right) \left( \sum_{j=1}^J z_{itj} P_{itj}(\beta_r^m) \right)' \right] \right\}$$

We can apply a similar lower bound to the one in Eq. (4).

$$B_{ir}^m = \sum_{i=1}^N w_{ir}^m \left\{ -\frac{1}{2} \sum_{t=1}^T \left[ \sum_{j=1}^J (z_{itj} z'_{itj}) - \frac{1}{J} \left( \sum_{j=1}^J z_{itj} \right) \left( \sum_{j=1}^J z_{itj} \right)' \right] \right\}$$

Similar to the scale heterogeneity model, this bound must be computed and inverted at each iteration. However, this can be simplified by computing $B_i$, which is used for all of the bounds, outside of the algorithm and simply re-weighting these values with $w_{ir}^m$ at each
iteration. Using this bound, Table 5 outlines the MM algorithm for the latent class model.

Table 5: MM Algorithm: latent class model

- **Outside of the algorithm**
  - For each $i = 1, 2, \ldots, N$, compute $B_i$ using Eq. (6)
  - Initialize with starting values $\Psi^0 = \{\beta_1^0, \ldots, \beta_R^0, \pi_1^0, \ldots, \pi_R^0\}

- **Repeat the following iteration until converged**, i.e. $||\Psi^{m+1} - \Psi^m||_\infty < \kappa$

  **Step 1:** Calculate weights from likelihood components
  - Given $\beta_r^m$ compute $P_{itj}(\beta_r^m) = \exp(z'_{itj}\beta_r^m) / \left(\sum_{k=1}^J \exp(z'_{itk}\beta_r^m)\right)$ for each $r = 1, \ldots, R$
  - Compute $L_i(\beta_r^m) = \prod_{t=1}^T \prod_{j=1}^J [P_{itj}(\beta_r^m)]^{d_{it}}$
  - Compute $w_{ir}^m = \pi_r^m L_i(\beta_r^m) / \left(\sum_{r'=1}^R \pi_r^m L_i(\beta_r^m)\right)$

  **Step 2:** Update parameters
  - $\pi_r^{m+1} = \frac{\sum_{i=1}^N w_{ir}^m}{N}$
  - $\beta_r^{m+1} = \beta_r^m - \left[\sum_{i=1}^N w_{ir}^m B_i\right]^{-1} g_r^m$
  - $g_r^m = \sum_{i=1}^N w_{ir}^m \left[\sum_{t=1}^T \sum_{j=1}^J z_{itj} (d_{itj} - P_{itj}(\beta_r^m))\right]

An alternative lower bound for this model that can be computed and inverted outside of the algorithm can be constructed by noting that $w_{ir}^m \leq 1$ and replacing the weights in Eq. (6) with $w_{ir}^m = 1$.

$$B = \sum_{i=1}^N \left\{-\frac{1}{2} \sum_{t=1}^T \left[\sum_{j=1}^J \left(z_{itj} z'_{itj}\right) - \frac{1}{J} \left(\sum_{j=1}^J z_{itj}\right)^2 \left(\sum_{j=1}^J z_{itj}\right)\right]\right\}$$

This bound could replace all of the $B_r^m$ in Table 5. This bound is less sharp than the one in Eq. (6), which will result in more iterations. However, it might result in a shorter computation time because it avoids taking the repeated matrix inverse.

### C Convergence Criteria

EM and MM algorithms are fixed point algorithms that produce a sequence of estimates $\Psi^1, \Psi^2, \ldots, \Psi^m$ which converge to a value $\Psi^*$. The algorithm is typically terminated when
the difference or percentage change between successive parameter estimates is less than some tolerance. Kuroda and Sakakihara (2006) propose a novel convergence criteria for the EM algorithm which exploits a vector epsilon algorithm. Given a linearly converging vector sequence, the vector epsilon algorithm generates an alternative sequence that converges to the same stationary value. Using the sequence of EM parameter estimates, $\Psi^m$, the vector epsilon sequence, $\psi^m$ is defined as,

$$\psi^{m+1} = \Psi^m + \left[ \left( \Psi^{m-1} - \Psi^m \right)^{-1} + \left( \Psi^{m+1} - \Psi^m \right)^{-1} \right]^{-1}$$

Where $x^{-1} = x/||x||^2$ with the denominator representing the dot product of vector $x$. Kuroda and Sakakihara (2006) show that this sequence, $\psi^1, \psi^2, \ldots, \psi^m$, converges to the same stationary value as the EM sequence, $\Psi^*$, but uses the magnitude and direction of the EM sequence to extrapolate to the convergence point, requiring significantly fewer iterations. Implementing this requires no modification to the EM code, and the sequence $\psi^m$ never enters any of the part of the algorithm. It is a sequence which is computed in tandem with the algorithm and provides a stopping criteria in the same manner one would use the EM sequence to form a stopping criteria, i.e. once $||\psi^{m+1} - \psi^m||_\infty < \kappa$, then $\psi^{m+1} \approx \Psi^*$. 

32