

Implementing An MM Algorithm for Fixed Effects Multinomial Logit Models

Mingli Chen
University of Warwick

Ian Meeker
Charles River Associates

Marc Rysman
Boston University

Shuang Wang
Charles River Associates*

November 14, 2025

I. Introduction

This document explains the sample code for Chen et al. (2025) and walks through two examples that show how to handle fixed effects and individual-specific coefficients. We also show how to incorporate the acceleration method in Zhou et al. (ZAL, 2011).

The sample code includes two files:

- `main.R`: provides the code for the two examples
- `helper_fcns.R`: provides the code for the estimators

There are 5 functions within `helper_fcns.R`.

- `mm_fe`: estimator for handling single- and multi-way fixed effects
- `mm_coef`: estimator for handling individual-specific coefficients
- `mm_coef_zal`: accelerated version of `mm_coef`
- `calc_prob`: function that calculates the logit probabilities
- `calc_loglik`: function that calculates the log-likelihood

All of the estimation functions are fairly general and take in a response variable y and a design matrix X . The functions should be able to accommodate most specifications with minor modifications.

*The opinions expressed in this article are the authors' own and do not reflect the views of Charles River Associates. We thank Erin Eidschun for spotting an error in an earlier version.

II. Simulated Data

Our examples rely on a simulated dataset. We consider 50,000 individuals choosing among three alternatives $j \in \{1, 2, 3\}$ in each of 36 time periods. We consider a single individual-specific variable x_{it} and a single alternative specific variable w_{jt} . We generate the choices from a multinomial logit model where utility has:

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

where ε_{ijt} is the Type I extreme value shock, γ_i is an individual specific coefficient, and α_{ij} are individual-alternative fixed effects. We set $(\beta_1, \beta_2, \beta_3) = (0, 0.5, 1)$ and we draw the variables and fixed effects from standard normals. We draw γ_i from a normal distribution with mean 0 and variance 0.25.

We drop individuals that never choose all of the alternatives at least once, leaving 48,725 individuals in the final data. The data set looks like:

	obs_id	individual	alternative	period	y	x	w	fe
	<int>	<int>	<int>	<int>	<lgcl>	<num>	<num>	<char>
1:	1	1	1	1	FALSE	-1.3825069	0.18038413	1_1
2:	1	1	2	1	FALSE	-1.3825069	0.48686661	1_2
3:	1	1	3	1	TRUE	-1.3825069	-0.06150710	1_3
4:	2	1	1	2	FALSE	0.9951642	0.02223641	1_1
5:	2	1	2	2	FALSE	0.9951642	-0.76774236	1_2
6:	2	1	3	2	TRUE	0.9951642	-0.52700006	1_3

The field `obs_id` identifies the choice setting. It is an integer that represents a unique combination of an individual and a time period.

III. Fixed Effects

A. Algorithm

Suppose that we want to estimate a model of the form:

$$u_{ijt} = x_{it}\beta_j + \alpha_{ij} + \varepsilon_{ijt} \quad (2)$$

The fixed effects are not identified without imposing a normalization. The most common normalization is to set one of the fixed effects to 0, for example $\alpha_{i1} = 0 \forall i$. By default, the MM algorithm imposes a normalization that is a function of the starting values. We discuss the normalization in next section.

Given starting parameters, we can estimate the above specification using the MM algorithm through repeated approximations and demeaned OLS. The algorithm is:

Algorithm 1 (Fixed Effects)

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

repeat

$k \leftarrow k + 1$

Minorization Step:

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

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

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

Maximization Step:

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

$$\boldsymbol{\beta}^{(k+1)} \leftarrow \left(\tilde{\boldsymbol{\mathcal{X}}}^{\prime} \tilde{\boldsymbol{\mathcal{X}}} \right)^{-1} \tilde{\boldsymbol{\mathcal{X}}} \tilde{\mathbf{v}}^{(k)}$$

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

$$\boldsymbol{\alpha}^{(k+1)} \leftarrow \mathbf{v}^{(k)} - \boldsymbol{\mathcal{X}} \boldsymbol{\beta}^{(k+1)} - \left(\tilde{\mathbf{v}}^{(k)} - \tilde{\boldsymbol{\mathcal{X}}} \boldsymbol{\beta}^{(k+1)} \right)$$

until convergence

As we note in the paper, the regressors in the maximization step never change from iteration to iteration so we can pull $\left(\tilde{\boldsymbol{\mathcal{X}}}^{\prime} \tilde{\boldsymbol{\mathcal{X}}} \right)^{-1} \tilde{\boldsymbol{\mathcal{X}}}$ out and calculate it once at the beginning. With this formulation, the algorithm becomes:

Algorithm 2 (Fixed Effects Modified)

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

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

$A \leftarrow (\tilde{\mathbf{X}}' \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}$

repeat

$k \leftarrow k + 1$

Minorization Step:

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

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

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

Maximization Step:

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

$$\boldsymbol{\beta}^{(k+1)} \leftarrow A\tilde{\mathbf{v}}^{(k)}$$

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

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

until convergence

Our code implements the modified version of the algorithm. For the stopping criteria, we use the change in successive log-likelihoods.

B. Normalization

Consider estimating the model in (2) with MM algorithm. We will view alternative 1 as the base option. Suppose that we are iteration k . The update to the fixed effects is

$$\alpha_{i1}^{(k+1)} = v_{i1t} - x_{it}\beta_1 - (\tilde{v}_{i1t} - \tilde{x}_{it}\beta_1) \quad (3)$$

In our setting, β_1 is 0 so the equation simplifies to:

$$\alpha_{i1}^{(k+1)} = v_{i1t} - \tilde{v}_{i1t} \quad (4)$$

$$= v_{i1t} - (v_{i1t} - \bar{v}_{i1t}) \quad (5)$$

$$= \bar{v}_{i1t} \quad (6)$$

$$= \frac{1}{T} \sum_t x_{it}\beta_1 + \alpha_{i1}^{(k)} + y_{ijt} - p_{ijt}^{(k)} \quad (7)$$

$$= \frac{1}{T} \sum_t 0 + \alpha_{i1}^{(k)} + y_{ijt} - p_{ijt}^{(k)} \quad (8)$$

$$= \alpha_{i1}^{(k)} + \frac{1}{T} \sum_t y_{ijt} - p_{ijt}^{(k)} \quad (9)$$

$$= \alpha_{i1}^{(0)} + \sum_{n=1}^k \frac{1}{T} \sum_t y_{ijt} - p_{ijt}^{(n)} \quad (10)$$

By default, the MM algorithm normalizes the fixed effect of the base alternative for individual i to value of the

previous iteration plus the average difference between the observed choices and the predicted probabilities, $\alpha_{i1}^{(k)} + \frac{1}{T} \sum_t y_{ijt} - p_{ijt}^{(k)}$. Working backward, we can express the update at iteration $k+1$ as the initial value plus the sum of the average difference between the choices and the predicted probabilities in all previous iterations.

This normalization does not affect the updates to β as we calculate β using demeaned OLS, $\beta^{(k+1)} = (\tilde{\mathbf{x}}' \tilde{\mathbf{x}})^{-1} \tilde{\mathbf{x}} \tilde{\mathbf{v}}^{(k)}$. When the variables are demeaned, the constant from the normalization differences out.

At each iteration, the MM algorithm will produce non-zero updates for the base alternative. The only exception is when starting value is zero and the average predicted probability is equal to the share of the base option for every individual. There are two ways to impose the usual normalization of $\alpha_{i1} = 0$. The first is to subtract off $\frac{1}{T} \sum_t y_{ijt} - p_{ijt}^{(k)}$ at each iteration. This just means calculating the update $\alpha^{(k+1)}_{i1}$ and then subtracting it off of the appropriate fixed effect. The second way is to subtract off the final estimate for α_{i0} from the fixed effects. This method is more efficient and avoids repeated calculations.

C. Estimation

In this section, we show how to use our estimation function for fixed effects. In section VI, we walk through each section of the code and explain its relation to the algorithm.

To estimate the model, we use the function `mm_fe(X, y, fe, index, tol)`. This function assume that all inputs are in long form. It takes in the following arguments:

- **y**: a logical vector that defines the choices
- **X**: a matrix containing the values of the regressors
- **fe**: for single-way fixed effects, a factor variable that define the levels of the fixed effect that will be absorbed and for multi-way fixed effects, a matrix of factor variables that define the levels of each fixed effect.
- **index**: a factor variable that identifies the choice setting. This is needed to calculate the logit probabilities in the minorization step.
- **tol**: the desired accuracy. The default is 1×10^{-8} .

We now setup the inputs

```
> X <- model.matrix(~-1 + x:as.factor(alternative), data = dt)
> X <- X[, -1]
> y <- as.matrix(dt[, y])
> index <- dt[, as.factor(obs_id)]
> fe <- dt[, as.factor(as.character(fe))]
```

We estimate the model with

```
> start <- list(beta = rep(0, ncol(X)), fe = 0, loglik = 0)
> res <- mm_fe(X, y, fe, index, start, tol = 1e-8)
> res$beta
```

```

              [,1]
x:as.factor(alternative)2 0.4911566
x:as.factor(alternative)3 0.9642876
```

```
> head(res$fe)
```

```

              [,1]
1 -0.4129061
2 -0.8636705
3  1.3771345
4 -0.4129061
5 -0.8636705
6  1.3771345
```

```
> res$iter
```

```
[1] 296
```

The function returns a list containing: the point estimates for β_j , the fixed effects, and the number of iterations. We return the fixed effects in long format with repeated entries for each choice setting. Subsetting gives the unique values.

IV. Individual-Specific Coefficients

A. Model

Suppose now that we want to estimate a logit model where utility has the form:

$$u_{ijt} = x_{it}\beta_j + w_{jt}\gamma_i + \alpha_{ij} + \varepsilon_{ijt} \quad (11)$$

We are now trying to estimate 97,450 fixed effects and 48,725 individual-specific coefficients. To do this efficiently, we need to absorb the fixed effects and the alternative-specific variables with the individual-specific coefficients using generalized demeaning. In this case the algorithm becomes:

Algorithm 3 (Individual-Specific Coefficients)

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

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

$A \leftarrow (\tilde{\mathbf{X}}' \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}$

repeat

$k \leftarrow k + 1$

 Minorization Step:

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

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

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

 Maximization Step:

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

$$\boldsymbol{\beta}^{(k+1)} \leftarrow A \tilde{\mathbf{v}}^{(k)}$$

2. Update absorbed variables: $\boldsymbol{\alpha}$

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

until convergence

Note that the algorithm is not keeping track of the individual γ_i 's and α_{ij} 's but instead is keeping track of the total contribution of the absorbed variables $w_{jt}\gamma_i + \alpha_{ij}$.

B. Estimation

To estimate the model, we can use the function `mm_coef(X, y, fe, coef, index, tol)`. This function assume that all inputs are in long form. It takes in the following arguments:

- **y**: a vector that defines the choice
- **X**: a matrix containing the values of the regressors
- **fe**: a factor variable that define the levels of the fixed effect that will be absorbed
- **coef**: a factor variable that define the levels of the individual slope coefficients
- **index**: a factor variable that identifies the choice setting. This is needed to calculate the logit probabilities in the minorization step.
- **tol**: the desired accuracy. The default is 1×10^{-8} .

We first setup the inputs. Many are the same as before.

```
> X <- model.matrix(~-1 + x:as.factor(alternative), data = dt)
> X <- X[, -1]
> y <- as.matrix(dt[, y])
```

```

> w <- dt[, w]
> fe <- dt[, .(fe = as.factor(as.character(fe)), individual = as.character(individual))]
> index <- dt[, as.factor(obs_id)]

```

and then estimate the model using

```

> start <- list(beta = rep(0, ncol(X)), fe = 0, loglik = 0)
> res <- mm_coef_zal(X, y, w, index, fe, start, tol = 1e-8)
> res$beta
> head(res$fe)
> res$iter

```

The function `mm_coef` returns the point estimates for β_j , the total contribution of the absorbed factors $w_{jt}\gamma_i + \alpha_{ij}$, and the number of iterations.

`mm_coef` can only handle one variable with individual-specific coefficients. Incorporating additional variables requires modifying the two lines that demean variables. Those lines are:

```

> X_dm <- fixest::demean(X = X, f = fe, slope.vars = w, slope.flag = c(0, -1))
> minor_dm <- fixest::demean(X = minor, f = fe, slope.vars = w, slope.flag = c(0, -1))

```

For this function, we are using `fixest::demean` to demean the variables. Incorporating another variable amounts to changing the arguments `f`, `slope.vars`, and `slope.flag` to ensure that we absorb the right variables.

We are using different functions to handle the demeaning in the case with just fixed effects and this case. With just fixed effects, we use `collapse::fhdwithin` as it is faster than `fixest::demean`. However, `collapse::fhdwithin` is less general. Researchers looking to explore a range of specification using only one estimation function should use `fixest::demean`.

V. Acceleration

As we show in the paper, our MM algorithm can exhibit slow convergence, especially when absorbing individual-specific coefficients. To speed up estimation, researchers can pair the MM algorithm with acceleration methods. Rather than use `mm_coef`, we recommend that researchers use an accelerated version `mm_zal`. We provide `mm_coef` mainly for illustrative purposes.

In the paper, we recommend that researchers use either SQUAREM (Varadhan and Roland, 2008) or ZAL (Zhou et al., 2011). Adding either method is simple as both take two steps of the MM algorithm and then extrapolate the next value from the previous two. Starting from the value for the k -th iteration $\theta^{(k)}$, we take one step to obtain θ_1 and then another step to obtain θ_2 . ZAL predicts the next value using:

$$\theta^{(k+1)} = (1 - c_k)\theta_1 + c_k\theta_2 \quad (12)$$

where $c_k = -\frac{(\theta_1 - \theta^{(k)})^T (\theta_1 - \theta^{(k)})}{(\theta_1 - \theta^{(k)})^T ((\theta_2 - \theta_1) - (\theta_1 - \theta^{(k)}))}$.

To implement the accelerated version, we essentially copy and paste the code for a single iteration and just store the values from the two iterations. These are then inputs in the extrapolation step.

As we note in the paper, the extrapolation step breaks the ascent property of the MM algorithm so we are not guaranteed to get a value that improves the log-likelihood. To improve numerical stability, researchers can revert back to θ_2 when the extrapolation step fails to improve the likelihood. We do not do this in our code.

In the sample code, we provide an accelerated version of `mm_coef`, `mm_coef_zal`. This function takes in the same arguments as `mm_coef`. We can estimate the model using:

```
> res <- mm_coef_zal(X, y, w, index, fe, start, tol = 1e-8)
> res$beta

[1] 0.5253344 1.0522039

> head(res$fe)

[1] -0.4555975 -0.6856417 1.4319154 -0.5166486 -1.1699687 1.2522174

> res$iter

[1] 1137
```

The accelerated function returns the same output as the base version.

VI. Detailed Breakdown of the Code

We now provide a detailed breakdown `mm_fe` for those interested in better understanding the code. The first line initializes the process

```
> new_values <- start
```

The next two lines demean the design matrix and create the pseudoinverse for the demeaned OLS, which we save for use in the maximization step.

```
> X_dm <- collapse::fhdwithin(X, fe)
> A <- solve(crossprod(X_dm)) %*% t(X_dm)
```

We then calculate the components for the minorization in the first iteration, the linear index and the logit probabilities.

```
> lin_pred <- X %*% values_new$beta + values_new$fe
> P <- calc_prob(lin_pred, index)
```

The loop covers the repeated minorization and maximization.

```
> iter <- 0
> repeat{
+   iter <- iter + 1
+
+   minor <- lin_pred + y - P
+   minor_dm <- collapse::fhdwithin(minor, fe)
+   values <- values_new
+
+   values_new$beta <- A %*% minor_dm
+   Xb <- X %*% values_new$beta
+   Xb_dm <- X_dm %*% values_new$beta
+   values_new$fe <- minor - Xb - (minor_dm - Xb_dm)
+   lin_pred <- Xb + values_new$fe
+   P <- calc_prob(lin_pred, index)
+
+   values_new$loglik <- calc_loglik(y, P)
+   diff <- abs(values_new$loglik - values$loglik)
+   if (diff < tol) break
+ }
```

The first two lines in the loop perform the minorization step. The first line calculates the minorization and the second one demeans it.

```
> minor <- lin_pred + y - P
> minor_dm <- collapse::fhdwithin(minor, fe)
```

The next lines cover the maximization step. The first line of this code chunk updates β and last three line update the fixed effects.

```
> values_new$beta <- A %*% minor_dm
> Xb <- X %*% values_new$beta
> Xb_dm <- X_dm %*% values_new$beta
> values_new$fe <- minor - Xb - (minor_dm - Xb_dm)
```

We then update the linear index and the probabilities for use in the next minorization step.

```
> lin_pred <- Xb + values_new$fe
> P <- calc_prob(lin_pred, index)
```

The final lines update the log-likelihood and check the stopping criteria.

```
> values_new$loglik <- calc_loglik(y, P)
> diff <- abs(values_new$loglik - values$loglik)
> if (diff < tol) break
```

While we do not discuss it here, the code for `mm_coef` is very similar. The main difference is the function used for the demeaning.

References

- Chen, M., I. Meeker, M. Rysman, and S. Wang (2025). An MM algorithm for fixed effects multinomial logit models. Working Paper.
- Varadhan, R. and C. Roland (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics* *35*(2), 335–353.
- Zhou, H., D. Alexander, and K. Lange (2011). A quasi-Newton acceleration for high-dimensional optimization algorithms. *Statistics and Computing* *21*, 261–273.