# Random Starting Values and Multistage Optimization

*Tihomir Asparouhov and Bengt Muthén*

November 8, 2019

## 1 Random Starting Values

Model estimation in Mplus is typically based on a fit function maximization where the arguments of the fit function are the model parameters. All optimization algorithms require a set of starting values for these parameters. Mplus provides a set of default starting values which are constructed from the sample statistics and can be found in the *Technical output* 1 section. Alteratively, Mplus can use a set of starting values specified in the Mplus input file using the * symbol. Another alternative for the starting value is the randomly generated starting values. To request randomly generated starting values, the STARTS option must be specified. The random starting values are useful in resolving convergence problems that can occur in certain situations. They can be used also in certain estimation problems that yield multiple local maxima, such as mixture models and EFA models. Using multiple sets of randomly generated starting value can ensure that the global maximum is found rather than an inferior local maximum. Random starting values are used by defaults for every Mixture model estimation.

The generation of the random starting value is described as follows. Let $v_{ij}$ be the random starting value for the $i-$th parameter in the $j-$th starting value set. The value $v_{ij}$ is computed as

$$v_{ij} = w_i + sr_{ij}b_i, \tag{1}$$

where $w_i$ is the original unperturbed starting value, either given in the input file or obtained as the Mplus default starting value, $r_{ij}$ is a uniformly distributed random number in the interval [-0.5, 0.5], $s$ is a scale variable, and $b_i$ is the base scale for that parameter. The scale variable $s$ determines the strength of the random perturbations. By default $s$ is set to 5. A larger $s$ value will trigger a more aggressive optimization search but may result in more non-convergence problems and more local maxima that are inferior to the global maximum. A smaller $s$ value will likely produce starting values that yield the same local maxima and may not explore the parameter space to the needed extend. The most optimal value of $s$ will certainly depend on the data and the model. The option STSCALE can be used to change the value of $s$.

The random numbers $r_{ij}$ are obtained as follows. A sequence of random numbers is generated with a special numerical routine that depends on an integer value called the seed of the random sequence. Changing the seed of the sequence, changes the random numbers in the sequence. Suppose that we want to generate $L$ sets of random staring values As a first step, Mplus uses a random seed $c$ to generate $L$ random integers $c_j$ between 1 and 1000000. The integer $c_j$ is then used as a seed number to generate the $j$-th random sequence $r_{ij}$. This is done so that the random sequence $r_{ij}$ can easily be reproduced and so that it is independent from the other random sequences. The sequence may need to be reproduced for two reasons. First, in a multistage optimization process, only a particular set of starting values may need to be rerun, for example, if a set of starting values is selected for a second stage optimization. Second, after an extensive optimization search, we may choose to rerun the best solution as a stand alone run for additional output features. This will save computational time as the optimization search would be reduced to that particular set of starting values. We can accomplish that using the option OPTSEED where the seed number $c_j$ is specified. The seed numbers are available in the Mplus output and are usually reported next to the maximum log-likelihood obtained with that seed. The global seed $c$ can be changed to produce a different set of seed values $c_j$ which in turn will produce different random sequences $r_{ij}$ as well as different random starting values. To change the global seed $c$, we specify the option STSEED.

The base scale values $b_i$ are computed as follows. For most parameters $b_i = 2$. The exceptions are as follows. For variance and covariance parameters $b_i = 0$, i.e., variance covariance parameter are actually not perturbed. For all mean and intercept parameter

$$b_i = 2max\{1, \sqrt{var}\} \tag{2}$$

where $var$ is the starting value of the residual variance for that variable. If the model has multiple latent class variables and the parametrization is the default logit parametrization, for all parameter in the latent class regression $b_i = .4$.

The above construction is used when the generating seed $c_j$ is an odd integer. When the seed is an even integer, we take a different approach for the thresholds parameters and intercepts of the latent class variables. This alternative approach produces better results in certain situations and is described as follows. For the thresholds of an observed categorical variable $U$ we use the following construction. If $U$ has $T + 1$ categories, i.e., it has $T$ thresholds for each latent class, we generate $r_1$,...,$r_T$ random numbers in the interval [0,1]. Let $r_{(1)}$, ..., $r_{(T)}$ be the ordered statistics for these random numbers, i.e., let r(1) be the smallest number etc. We use as a starting value for the $i$-th threshold $\tau_i = log(r_{(i)}/(1 - r_{(i)}))$. Note that unlike the previous construction, these starting values are independent of the unperturbed starting values. Similarly we provide starting values for the means of the latent class variable $C$. If $C$ takes $K$ values, we generate $K-1$ random numbers in the interval [0, 1] and the starting value are obtained as $\alpha_k = log((r_{(k)} - r_{(k-1)})/(1 - r_{(K-1)}))$, where we set r(0) = 0. These formulas set the distribution of $U$ and $C$ to be directly defined by the random numbers, i.e., $P(U \leq i) = P(C \leq i) = r_{(i)}$. The fact that we use this alternate random starting values when $c_j$ is even amounts to having this alternative perturbation algorithm be used with a probability of 50%.

In the presence of multiple latent class variables, when the probability parametrization is used for the latent class variables, we use a different algorithm with probability 1/3, i.e., when the number $c_i$ is divisible by 3. This alternate algorithm is specified as follows. For each conditional or unconditional probability distribution $P$ with $K$ values we generate random $r_1$, ...,$r_K$ in the interval of [0,1] and use as a starting value for the $k$-th probability the value $r_k/(r_1 + ...r_K)$.

## 2 Multi-stage optimization process for latent class model estimation

The general optimization process in Mplus is a multistage process. Mplus version 8.3 and earlier used a two-stage process. Mplus 8.4 introduced a third stage in the optimization. As discussed below, this introduces a third setting for the STARTS option and a second setting for the STITER option. Thus, the STARTS option can be specified as STARTS=$L_1 L_2 L_3$, where $L_1$ refers to the number of random starting values used in stage I, $L_2$ refers to the number of random starting values used in stage II, and $L_3$ refers to the number of random starting values used in stage III. STITER can be specified as STITER=$I_1 I_2$, where $I_1$ refers to the cutoff iteration used in Stage I, and $I_2$ refers to the cutoff iteration used in Stage II. These additional settings are optional, i.e., the STARTS and STITER commands can be used the same way they are used in Mplus 8.3 or earlier. If the commands are used without the additional arguments, Mplus assumes the default values for these settings $L_3 = 10$ and $I_2 = 75$. This means that even if the options are used without their additional arguments, three-stage optimization may be performed. It is possible to reproduce Mplus 8.3 two-stage optimization in Mplus 8.4 by using the specification STARTS=$L_1 L_2 L_2$.

In this section we describe the three stages of the optimization and the ANALYSIS options that can be used to customize it.

## 2.1 Stage I

In the first stage of the computation for each set of starting values, we begin the EM optimization process where the maximum number of EM iterations is set to a low value. By default that value is 10 but can be changed with the STITER option. The convergence criteria are made more lenient in this stage of the optimization and are increased to 1. If those convergence criteria are satisfied we terminate the estimation even before the 10-th iteration is reached. The log-likelihood values obtained from each set of starting values are recorded and used in stage two. If the number of starting values requested with the starts command is $L_1$, at this stage of the estimation we perform $L_1$ limited optimizations to obtain $L_1$ log-likelihood values.

## 2.2 Stage II

In Stage II, we continue the estimation only for a limited set of random starting value sets. The number of such sets is controlled by the second number in the STARTS option. Suppose that this number is $L_2$. The log-likelihood values obtained in stage I are ordered and only the top $L_2$ starting values set are allowed to continue in the optimization process. All stage two optimizations are allowed to continue now beyond the 10-th EM iteration and may be completed until convergence unless a selection process implemented in Stage III removes or terminates the optimization.

## 2.3 Stage III

Stage III is controlled by the third entry in the STARTS option: $L_3$. By default $L_3 = 10$. If $L_2 \leq L_3$ then all Stage II optimizations are completed until convergence. If $L_2 > L_3$ then some of the second stage optimizations might not be completed. The optimization continues as follows. The first $L_3$ optimizations are completed until convergence. For the remaining $L_2 - L_3$ optimizations we continue the estimation until the 75-th iteration (this number is controlled by the second entry in the STITER option). At that point, the optimization is allowed to continue only if at the 75-th iteration mark it obtains a log-likelihood value that is better than any one of the top $L_3$ optimizations at the 75-th mark that have already been completed. If the log-likelihood is not higher than the top $L_3$ optimizations at the 75-iteration mark, we conclude that this set of random starting values is unlikely to produce the best solution, as $L_3$ of the previous optimizations were better at that point in the optimization. Therefore, we terminate the optimization process for that starting values set and move on to the next starting value set obtained in Stage II. This process is sequential and is intended to complete to convergence the best $L_3$ optimizations. However, the number of completed optimizations is likely to be higher. That is because some of the early optimizations that were completed may not be among the best $L_3$ optimizations at the end if better optimizations were found later on in the search process. Mplus will report the log-likelihood values for all optimizations that were completed. Stage III is intended to weed out Stage II optimizations that are clearly underperforming and thus reduce the computational burden, while preserving the efficiency of the random starting value search process.

## 2.4 Additional point

The STARTS option is designed to control the full optimization process using the numbers $L_1$, $L_2$ and $L_3$. However, we do not recommend changing $L_3$. The default value of $L_3 = 10$ is designed to obtain the correct top two log-likelihood values. We need the top two values, and not just the top one log-likelihood value, because in general we want to see that the best log-likelihood value is replicated at least one more time. If it is not replicated at least once, that is an indication that more random starting values should be used in the search process.

Note also that the third stage optimization is designed to help those situations where a large number of $L_1$ and $L_2$ starting values are used. If the option STARTS= $L_1 L_2$ is specified then the default value of $L_3$ is unchanged, i.e., $L_3 = 10$. If $L_2$ is smaller than 10, the third stage will have no effect on the optimization at all. If $L_2$ is less than 20, the effect of the third stage will likely be very small. If the model has several latent class variables, or a single class variable with a large number of classes, typically a large number of random startling values is needed such as STARTS=300 100. In those situations, the third stage will be very helpful in reducing the computational burden. In summary, the size of the optimization search should be driven by the first two numbers in the STARTS option, just as this was done prior to Mplus 8.4.

If multiple processors are available for the computation, the PROCESSOR=T option should be specified. This will have the following effect on the multistage optimization. Each of the $T$ computational threads will divide the number of stage I and II random starting value sets approximately equally. Thus each of the $T$ copies of Mplus will conduct a stage I optimization with $L_1/T$ random starting value sets and a stage II optimization with $L_2/T$ random starting value sets. Stage III however is not divided. The threads will communicate, and will maintain a joint record of the top 10 optimizations at the 75-iteration mark. Thus an optimization may be discarded if another thread has produced optimizations that outperformed it at the 75-iteration mark.

## 3 Timing comparison

In this section we present some timing comparisons to illustrate the advantages of the three-stage estimation implemented in Mplus 8.4 over the two-stage estimation implemented in Mplus 8.3 and earlier versions. Example 1 is a modification of Mplus User's Guide example 10.8. Examples 2-5 are discussed in detail in Muthén and Asparouhov (2019). Here we provide a brief summary of the examples to illustrate the size of the computation.

- Example 1. This a two-level growth mixture model with 1 latent class variable with 4 categories, 4 observed continuous variables, 2 covariates, 2 dimensions of numerical integration, sample size=1500, and STARTS=250 100

- Example 2. This is an LTA model with 3 latent class variables with 5 categories each, 12 observed categorical variables, 0 dimensions of numerical integration, sample size=2933, and STARTS=100 20
- Example 3. This is a binary RI-LTA model with 4 latent class variables, one has 2 categories and the other three have 5 categories, 12 observed categorical variables, 0 dimensions of numerical integration, sample size=2933, and STARTS=320 80
- Example 4. This is a continuous RI-LTA model with 3 latent class variables with 5 categories each, 12 observed categorical variables, 1 dimension of numerical integration, sample size=2933, and STARTS=400 80
- Example 5. This is a continuous RI-LTA model with 3 latent class variables with 5 categories each, 9 observed categorical variables, 1 dimension of numerical integration, sample size=2933, and STARTS=320 80

We use Mplus 8.3 and Mplus 8.4 to compare the computational time for these examples on two different Intel CPUs: i7-7700k and i9-9900k, and varying number of processors used in the computation. The results are presented in Table 1. Example 1 has only one latent class variable and the gains in the computational speed in that example are entirely due to the three-stage estimation. This is not the case for examples 2-5 which have multiple latent class variables.

In addition to the three-stage estimation, Mplus 8.4 has computational improvements for LTA analysis and more generally for models with multiple latent class variables. These improvements are based on ideas from the well known work of Baum and Welch on hidden Markov models, see Baum (1970). We have adapted their work to apply for more general models, such as non-Markov hidden processes, as well as models with continuous latent variables requiring numerical integration. Thus, examples 2-5 benefit not only from the three-stage estimation but also from the Baum-Welch algorithm improvements. Nevertheless, we have included these examples here because models with multiple latent class variables are the type of models

| i7-7700k | | i9-9900k | | | | | |
|---|---|---|---|---|---|---|---|
| 8.3 Proc=8 | 8.4 Proc=8 | 8.3 Proc=8 | 8.3 Proc=10 | 8.3 Proc=12 | 8.4 Proc=8 | 8.4 Proc=10 | 8.4 Proc=12 |
| Example 1 | | | | | | | |
| 23:12 | 12:55 | 13:21 | 13:16 | 12:21 | 7:26 | 7:36 | 6:41 |
| Example 2 | | | | | | | |
| 6:08 | 00:28 | 3:37 | 5:09 | 05:43 | 00:18 | 00:17 | 00:17 |
| Example 3 | | | | | | | |
| 1:30:08 | 10:36 | 1:01:46 | 47:34 | 45:26 | 6:03 | 5:36 | 5:47 |
| Example 4 | | | | | | | |
| 12:30:04 | 14:36 | 8:47:42 | 7:36:54 | 7:15:02 | 8:11 | 7:44 | 7:40 |
| Example 5 | | | | | | | |
| 20:25:17 | 27:22 | 15:32:38 | 12:17:19 | 12:17:12 | 15:01 | 14:51 | 14:47 |

Table 1: Computational time comparison

that are typically estimated with a large number of random starting values. These are the types of examples where we can expect that the three-stage estimation will have a very large impact on the computational time. In all of the above examples we have verified that the top two solutions found with all estimation settings are the same.

The results are stunning. A 20 hours computation in Mplus 8.3 can be done in Mplus 8.4 in less than 15 minutes, by utilizing the advantages of the three-stage estimation, the Baum-Welch algorithm, as well as updated hardware.

## References

[1] Baum, L. E. (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains, Ann. Math. Statist., vol. 41, pp. 164–171.

[2] Muthén, B., & Asparouhov, T. (2019) What Multilevel Modeling Can Teach Us About Single-Level Modeling: Latent Transition Analysis With Random Intercepts (RI-LTA). http://www.statmodel.com/download/RI-LTA.pdf