David Kreiberg, Katerina Marcoulides & Ulf Henning Olsson (2021) A Faster Procedure for Estimating CFA Models Applying Minimum Distance Estimators with a Fixed Weight Matrix, Structural Equation Modeling: A Multidisciplinary Journal, 28:5, 725-739, DOI: 10.1080/10705511.2020.1835484

A faster procedure for estimating CFA models applying Minimum Distance Estimators with a fixed weight matrix

David Kreiberg[1], Katerina Marcoulides[2] and Ulf Henning Olsson[1]

[1]BI Norwegian Business School

[2]University of Minnesota

David Kreiberg is a Lecturer of Statistics

Katerina Marcoulides is Assistant Professor of Quantitative/Psychometric Methods

Ulf Henning Olsson is Professor Statistics

Abstract

This paper presents a numerically more efficient implementation of the quadratic form minimum distance (MD) estimator with fixed weight matrix for confirmatory factor analysis (CFA) models. In structural equation modeling (SEM) computer software, such as EQS, lavaan, LISREL and Mplus, various MD estimators are available to the user. Standard procedures for implementing MD estimators involves a one-step approach applying non-linear optimization techniques. Our implementation differs from the standard approach by utilizing a two-step estimation procedure. In the first step, only a subset of the parameters are estimated using non-linear optimization. In the second step, the remaining parameters are obtained using numerically efficient Linear Least Squares (LLS) methods. Through examples, it is demonstrated that the proposed implementation of MD estimators may be considerably faster than what the standard implementation offer. The proposed procedure will be of particular interest in computationally intensive applications such as simulation, bootstrapping and other procedures involving re-sampling.

Key words

CFA, quadratic form fit function, minimum distance estimator, estimation time

Introduction

Structural equation modeling (SEM) is a well-established and effective statistical modeling approach used by researchers in many scientific disciplines for the examination of complex relationships among observed and latent variables. The modeling approach includes as special cases path analysis, factor analysis, growth curve modeling, survival analysis, latent class analysis, and latent transition analysis (Marcoulides & Schumacker, 1996; Muthén & Asparouhov, 2011). A key element in SEM is parameter estimation. To date, several estimation procedures have been suggested in the literature. Among the most commonly used procedures are maximum likelihood (ML), unweighted least squares (ULS) and generalized (weighted) least squares (GLS) (see e.g., Bollen, 1989).

Although the specific functional form may differ across estimation procedures, almost all of them can be considered variations of a minimum distance (MD) estimator (Browne, 1982). In its most general formulation, the MD estimator is written using the quadratic form

$$F = \big(\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\big)^T \boldsymbol{V} \big(\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\big) \tag{1}$$

where $\boldsymbol{s}$ and $\boldsymbol{\sigma}(\boldsymbol{\vartheta})$ are the vector elements in the observed sample variance-covariance and the model implied variance-covariance matrices, respectively. The matrix $\boldsymbol{V}$ is either a fixed positive definite weight matrix or a weight matrix that converges in probability to a positive definite matrix. By appropriately choosing the form of $\boldsymbol{V}$, different estimators can be obtained (for details, see e.g., Jöreskog et al., 2016). For a specific choice of $\boldsymbol{V}$, estimation is performed by minimizing (1) w.r.t. the unknown parameters using non-linear optimization techniques. We will refer to this estimation procedure as Non-Linear Least Squares (NLLS).

Due to a variety of complexities, such as violations of distributional assumptions and small sample behavior, extensive research over the past few decades has focused on improving the estimation of SEMs. However, not much research has been done in terms of computational efficiency when applying MD estimators, at least not within the confines of SEM. A faster implementation of the MD estimator may be of particular value in applications where some form of simulation or re-sampling is involved. Obvious examples are simulation based estimation, bootstrapping and, more generally, simulation research. Fast estimation may also be useful in applied research when the number of observed variables is large and the implemented model contain a large number of free parameters.

In this study, we introduce a numerically more efficient, but otherwise equivalent, procedure for how to minimize the MD quadratic form fit function when the model takes the form of a CFA model. The proposed procedure is based on a study by Golub and Pereyra (1973), who termed it Separable Non-Linear Least Squares (SNLLS). It is emphasized that we are not proposing a new estimator, but a faster implementation of existing estimators derived from (1). Thus, applying SNLLS will yield the exact same estimation results as the ones obtained from applying NLLS (i.e., the same parameter estimates, standard errors and fit statistics). Estimating CFA models using NLLS involves numerically searching a solution for the entire set of free parameters. In contrast, SNLLS is a two-step process where, in the first step, only a subset of the parameters are estimated applying non-linear optimization techniques. The remaining parameters are estimated in a second step using LLS. Since the first estimation step represents a lower dimensional optimization problem, fewer iterations and function evaluations are required before arriving at the solution. As a result, considerable time gains may be realized when estimating the model. Using the same notation as Golub and Pereyra (1973), Appendix A provides a brief mathematical description of the SNLLS framework and its justification.

Over the years, estimation using SNLLS has become popular in a wide range of scientific disciplines such as system identification, signal analysis, robotics, telecommunications and other areas of applied engineering. Golub and Pereyra (2002), alongside a presentation of various technical aspects, provided an overview of several real world applications applying SNLLS. More directly related to the current study, Kreiberg et. al. (2016) outlined a CFA modeling approach to the error-in-variables (EIV) problem. Estimating the CFA-EIV model, they outlined a SNLLS implementation of the quadratic form fit function. The aim of the current study is to further generalize the work in Kreiberg et. al. (2016) to include all CFA models.

Implementing SNLLS requires a modification of the objective function. In the following sections, we outline the complete analytical framework for how to appropriately modify (1). Through examples and simulations, we illustrate the potential benefits and some possible challenges of applying SNLLS. One minor modification to the original framework by Golub and Pereyra is the introduction of the weight matrix $\boldsymbol{V}$. The SNLLS procedure works with different forms of $\boldsymbol{V}$, e.g. when the aim is to handle non-normal or ordinal variables. However, cases in which $\boldsymbol{V}$ is a function of the parameters (i.e., $\boldsymbol{V} = \boldsymbol{V}(\boldsymbol{\vartheta})$) will pose a problem, since estimation applying SNLLS requires $\boldsymbol{V}$ to be a known matrix. That effectively rules out estimation using ML. We therefore limit the further presentation to only consider applications of SNLLS for which $\boldsymbol{V}$ is a fixed matrix. Specifically, in the examples, we use the GLS estimator with weight matrix given by

$$\boldsymbol{V} = 2^{-1}\boldsymbol{L}^{T}(\boldsymbol{S}^{-1} \otimes \boldsymbol{S}^{-1})\boldsymbol{L} \tag{2}$$

where $\boldsymbol{S}$ is a $(p \times p)$ sample covariance matrix, $\otimes$ denotes the Kronecker product and $\boldsymbol{L}$ is the so-called duplication matrix (see Magnus and Neudecker, 1999).

The remainder of the study is organized in the following way. First, to improve overall readability and establish a general notation, some standard mathematical details describing the modeling framework are provided. Next, the theoretical background and detailed mathematical derivations necessary for the proposed procedure is presented. This section is followed by illustrations comparing the numerical efficiency of applying NLLS and SNLLS using empirical examples and simulations. The first illustration is the well-known Nine Psychological Test model from the Holzinger and Swineford (1939) study. The second illustration is taken from Miller et. al. (2018), who used a CFA approach to analyze Multitrait Multimethod Matrices. The specific model is a 3–Traits by 3–Methods Correlated Traits/Correlated Methods model. Such models represent a challenge since their estimation often leads to improper solutions (see e.g. Marsh and Bailey, 1991). The third illustration uses the dependent part (the so-called democracy part) of Bollen's (1989) panel model for political democracy and industrialization in 75 countries. This particular part of Bollen's (1989) panel model provides the ideal illustration for examining a repeated measurement model with correlated errors. The fourth illustration is a brief simulation study. The design allows us to study to what extent numerical efficiency of NLLS and SNLLS is affected by model complexity and the relative number of latent variance-covariance parameters to the number of factor loading parameters.

Analytical Review and Derivations

We begin by considering the common CFA modeling framework. Following this overview, we then turn our attention to the mathematical details and derivations needed for implementing the SNLLS procedure.

Let $\boldsymbol{x}$ be a stochastic $(p \times 1)$ zero mean vector of observed variables, and let $\boldsymbol{\Sigma}$ denote the associated $(p \times p)$ covariance matrix. Due to symmetry, the number of non-duplicated elements in $\boldsymbol{\Sigma}$ is equal to $h = 2^{-1}p(p+1)$.

Suppose that the underlying process of $\boldsymbol{x}$ takes the simple form

$$\boldsymbol{x} = \boldsymbol{\Lambda}\boldsymbol{\xi} + \boldsymbol{\delta} \tag{3}$$

where $\boldsymbol{\xi}$ is an $(m \times 1)$ vector of common factors, $\boldsymbol{\delta}$ is a $(p \times 1)$ vector of unique factors (sometimes referred to as the errors) and $\boldsymbol{\Lambda}$ is a $(p \times m)$ parameter matrix relating the elements of $\boldsymbol{\xi}$ to the elements of $\boldsymbol{x}$. It is generally assumed that the elements of $\boldsymbol{\delta}$ are mutually uncorrelated with the elements of $\boldsymbol{\xi}$.

If the above model holds, we can express $\boldsymbol{\Sigma}$ as a matrix function of the parameters characterizing the process of $\boldsymbol{x}$ as

$$\begin{aligned} \boldsymbol{\Sigma} &= \boldsymbol{\Sigma}(\boldsymbol{\vartheta}) \\ &= \boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Lambda}^T + \boldsymbol{\Theta} \end{aligned} \tag{4}$$

where T is the transpose operator, $\boldsymbol{\Phi} = E\{\boldsymbol{\xi}\boldsymbol{\xi}^T\}$ is an $(m \times m)$ covariance matrix associated with the common factors, $\boldsymbol{\Theta} = E\{\boldsymbol{\delta}\boldsymbol{\delta}^T\}$ is a $(p \times p)$ covariance matrix associated with the unique factors, and $\boldsymbol{\vartheta}$ denotes a $(t_\vartheta \times 1)$ parameter vector composed of the (free) elements of $\boldsymbol{\Lambda}$, $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$.

Suppose that a sample of n data points, denoted $\boldsymbol{x}_i$ (for $i = 1, \ldots, n$), are observed from a population of interest. Given the observed sample data, the aim is to estimate the true, but

unknown, parameter vector $\boldsymbol{\vartheta}_0$. Let $F = F\big(\boldsymbol{S}, \boldsymbol{\Sigma}(\boldsymbol{\vartheta})\big)$ be a scalar function that expresses the distance between $\boldsymbol{S}$ and $\boldsymbol{\Sigma}(\boldsymbol{\vartheta})$. An estimate of $\boldsymbol{\vartheta}_0$ is then obtained by

$$\widehat{\boldsymbol{\vartheta}} = \arg\min_{\boldsymbol{\vartheta}} F\big(\boldsymbol{S}, \boldsymbol{\Sigma}(\boldsymbol{\vartheta})\big) \tag{5}$$

where $\boldsymbol{S}$, the sample covariance matrix, is obtained from the observed data using

$$\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T \tag{6}$$

As outlined in the introduction, we will be concerned with the quadratic form fit function, given by

$$F = \big(\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\big)^T \boldsymbol{V} \big(\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\big) \tag{7}$$

where $\boldsymbol{V}$ is an $(h \times h)$ symmetric weighting matrix, which is typically computed from the data. Moreover, $\boldsymbol{s}$ and $\boldsymbol{\sigma}(\boldsymbol{\vartheta})$, which are both of size $(h \times 1)$, are vectors composed of the non-duplicated elements of $\boldsymbol{S}$ and $\boldsymbol{\Sigma}(\boldsymbol{\vartheta})$, respectively. Specifically, $\boldsymbol{s}$ and $\boldsymbol{\sigma}(\boldsymbol{\vartheta})$ are respectively obtained from

$$\begin{aligned} \boldsymbol{s} &= \mathrm{vech}(\boldsymbol{S}) \\ &= \boldsymbol{K}^T \mathrm{vec}(\boldsymbol{S}) \end{aligned} \tag{8}$$

and

$$\begin{aligned} \boldsymbol{\sigma}(\boldsymbol{\vartheta}) &= \mathrm{vech}\big(\boldsymbol{\Sigma}(\boldsymbol{\vartheta})\big) \\ &= \boldsymbol{K}^T \mathrm{vec}\big(\boldsymbol{\Sigma}(\boldsymbol{\vartheta})\big) \end{aligned} \tag{9}$$

In (8) and (9), $\boldsymbol{K}$ is a $(p^2 \times h)$ matrix computed from $\boldsymbol{K} = \boldsymbol{L}(\boldsymbol{L}^T\boldsymbol{L})^{-1}$, where $\boldsymbol{L}$ is a $(p^2 \times h)$ matrix containing only ones and zeros (see the Appendix for mathematical details concerning the derivation of $\boldsymbol{L}$). Moreover, vec is the operation of vectorising the elements of a matrix and vech is the operation of vectorising the non-duplicated elements of a symmetric matrix.

For later, it will be useful to express (7) in terms of the Euclidean norm (which is essentially the least squares objective function expressing the sum of the squared residuals)

$$F = \|\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\|_{\boldsymbol{V}}^2 \tag{10}$$

The function F is minimized by numerically searching the parameter space, using a suitable algorithm, until some convergence criteria is satisfied.

Modifying the Quadratic Fit Function F

Next, we outline how to appropriately modify F to accommodate the SNLLS estimation procedure. Before doing so, we first define some new and necessary notation that is needed for completing the computational processes.

Let $\boldsymbol{\vartheta}_\lambda$ $(t_{\vartheta_\lambda} \times 1)$ be a partition of $\boldsymbol{\vartheta}$ containing the free parameters relating the elements of $\boldsymbol{\xi}$ to the elements of $\boldsymbol{x}$, and let $\boldsymbol{\vartheta}_{\varphi,\theta} = \begin{pmatrix}\boldsymbol{\vartheta}_\varphi^T & \boldsymbol{\vartheta}_\theta^T\end{pmatrix}^T$ $(t_{\vartheta_{\varphi,\theta}} \times 1)$ be a partition of $\boldsymbol{\vartheta}$ containing the free non-repeated elements of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$. We then have

$$\boldsymbol{\vartheta} = \begin{pmatrix}\boldsymbol{\vartheta}_\lambda^T & \boldsymbol{\vartheta}_{\varphi,\theta}^T\end{pmatrix}^T \tag{11}$$

Our approach will be to rewrite F in the following way

$$
\begin{aligned}
F &= \|\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\|_{\boldsymbol{V}}^2 \\
&= \left\|\boldsymbol{s} - \boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta}\right\|_{\boldsymbol{V}}^2
\end{aligned}
\tag{12}
$$

where $\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)$ $(h \times t_{\vartheta_{\varphi,\theta}})$ is a tall matrix function (i.e., $h > t_{\vartheta_{\varphi,\theta}}$) assumed to have full column rank. We immediately see that for some value of $\boldsymbol{\vartheta}_\lambda$, the solution to the problem of minimizing F w.r.t to $\boldsymbol{\vartheta}_{\varphi,\theta}$ is an application of LLS, from which the solution immediately follows

$$
\boldsymbol{\vartheta}_{\varphi,\theta} = \left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \tag{13}
$$

In order to implement the LLS solution for estimating $\boldsymbol{\vartheta}_{\varphi,\theta}$, it is necessary to obtain $\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)$ for some estimate of $\boldsymbol{\vartheta}_\lambda$. To do so, two related problems must be solved. The first problem is to derive the functional form of $\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)$. The second problem is to outline a procedure for how to estimate $\boldsymbol{\vartheta}_\lambda$ without directly involving $\boldsymbol{\vartheta}_{\varphi,\theta}$.

First, we derive the functional expression for $\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)$. Suppose that there exist matrices $\boldsymbol{L}_\Phi$ and $\boldsymbol{L}_\Theta$ such that the following expressions hold (details on how to obtain $\boldsymbol{L}_\Phi$ and $\boldsymbol{L}_\Theta$ are provided in the Appendix)

$$
\mathrm{vec}(\boldsymbol{\Phi}) = \boldsymbol{L}_\Phi\boldsymbol{\vartheta}_\varphi \tag{14}
$$

and

$$
\mathrm{vec}(\boldsymbol{\Theta}) = \boldsymbol{L}_\Theta\boldsymbol{\vartheta}_\theta \tag{15}
$$

where $\boldsymbol{L}_\Phi$ and $\boldsymbol{L}_\Theta$ are selection matrices containing zeros and ones. Their exact formulation will depend on the actual structure of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$. Given the common CFA model, we can write

$$\begin{aligned}\boldsymbol{\sigma}(\boldsymbol{\vartheta}) &= \boldsymbol{K}^T \mathrm{vec}\big(\boldsymbol{\Sigma}(\boldsymbol{\vartheta})\big) \\ &= \boldsymbol{K}^T \mathrm{vec}(\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Lambda}^T + \boldsymbol{\Theta}) \\ &= \boldsymbol{K}^T \big(\mathrm{vec}(\boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Lambda}^T) + \mathrm{vec}(\boldsymbol{\Theta})\big) \\ &= \boldsymbol{K}^T \big((\boldsymbol{\Lambda}^T \otimes \boldsymbol{\Lambda})\mathrm{vec}(\boldsymbol{\Phi}) + \mathrm{vec}(\boldsymbol{\Theta})\big)\end{aligned} \tag{16}$$

In this expression, we made use of the matrix identity: $\mathrm{vec}(\boldsymbol{ABC}) = (\boldsymbol{C}^T \otimes \boldsymbol{A})\mathrm{vec}(\boldsymbol{B})$, where $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are matrices of compatible sizes.

Inserting the expressions for $\mathrm{vec}(\boldsymbol{\Phi})$ and $\mathrm{vec}(\boldsymbol{\Theta})$, as stated in (14) and (15), into (16) gives

$$\begin{aligned}\boldsymbol{\sigma}(\boldsymbol{\vartheta}) &= \boldsymbol{K}^T \left((\boldsymbol{\Lambda}^T \otimes \boldsymbol{\Lambda})\boldsymbol{L}_\Phi \boldsymbol{\vartheta}_\varphi + \boldsymbol{L}_\Theta \boldsymbol{\vartheta}_\theta\right) \\ &= \boldsymbol{K}^T ((\boldsymbol{\Lambda}^T \otimes \boldsymbol{\Lambda})\boldsymbol{L}_\Phi \quad \boldsymbol{L}_\Theta) \begin{pmatrix}\boldsymbol{\vartheta}_\varphi \\ \boldsymbol{\vartheta}_\theta\end{pmatrix} \\ &:= \boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta}\end{aligned} \tag{17}$$

Next, we outline an approach that will allow us to obtain an estimate of $\boldsymbol{\vartheta}_\lambda$. Again, consider the function F, which is now written using the alternative formulation

$$\begin{aligned}F &= \big(\boldsymbol{s} - \boldsymbol{G}(\boldsymbol{\theta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta}\big)^T \boldsymbol{V}\big(\boldsymbol{s} - \boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta}\big) \\ &= \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta} - \boldsymbol{\vartheta}_{\varphi,\theta}^T\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} + \boldsymbol{\vartheta}_{\varphi,\theta}^T\boldsymbol{G}^T(\boldsymbol{\theta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta} \\ &= \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - 2\boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta} + \boldsymbol{\vartheta}_{\varphi,\theta}^T\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\boldsymbol{\vartheta}_{\varphi,\theta}\end{aligned} \tag{18}$$

Inserting (13) into (18) leads to

$$
\begin{aligned}
F = \; & \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - 2\boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \\
& + \left(\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s}\right)^T \boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda) \\
& \times \left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \\
= \; & \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - 2\boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \\
& + \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda) \\
& \times \left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \\
= \; & \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - 2\boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \\
& + \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \\
= \; & \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s}
\end{aligned}
\tag{19}
$$

Relying on the mathematical results in Golub and Pereyra (1973), minimizing (19) w.r.t $\boldsymbol{\vartheta}_\lambda$ leads to the exact same minimum function value as the one obtained from minimizing (7) w.r.t $\boldsymbol{\vartheta}$.

Expressing F entirely as a function of $\boldsymbol{\vartheta}_\lambda$ is key in reducing the estimation time. To specifically highlight the benefits of rewriting F, as done in (18) and (19), let us again consider the function $F = F(\boldsymbol{\vartheta})$ written as

$$
F(\boldsymbol{\vartheta}) = \left(\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\right)^T\boldsymbol{V}\left(\boldsymbol{s} - \boldsymbol{\sigma}(\boldsymbol{\vartheta})\right) \tag{20}
$$

Given an optimization algorithm and a set of initial values, minimizing $F(\boldsymbol{\vartheta})$ w.r.t $\boldsymbol{\vartheta}$ involves searching the entire parameter space (i.e., the whole space of $\boldsymbol{\vartheta}$).

Next, consider the function $F = F(\boldsymbol{\vartheta}_\lambda)$ given by

$$F(\boldsymbol{\vartheta}_\lambda) = \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\left(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\right)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \tag{21}$$

where $\boldsymbol{\vartheta}_\lambda$ is a parameter vector of lower dimension as compared to the full parameter vector $\boldsymbol{\vartheta}$. Minimizing $F(\boldsymbol{\vartheta}_\lambda)$ involves searching a reduced parameter space, which is inevitably less computationally demanding. Thus, it is reasonable to conjecture that a reduction in estimation time can be realized from applying $F(\boldsymbol{\vartheta}_\lambda)$.

When an estimate of $\boldsymbol{\vartheta}_\lambda$ is obtained, the remaining parameters are estimated by a straightforward, and computationally efficient, application of LLS

$$\widehat{\boldsymbol{\vartheta}}_{\varphi,\theta} = \left(\boldsymbol{G}^T(\widehat{\boldsymbol{\vartheta}}_\lambda)\boldsymbol{V}\boldsymbol{G}(\widehat{\boldsymbol{\vartheta}}_\lambda)\right)^{-1}\boldsymbol{G}^T(\widehat{\boldsymbol{\vartheta}}_\lambda)\boldsymbol{V}\boldsymbol{s} \tag{22}$$

Depending therefore on the model, it appears that the computational benefits of minimizing $F(\boldsymbol{\vartheta}_\lambda)$, rather than $F(\boldsymbol{\vartheta})$, far outweighs the additional step needed to estimate $\boldsymbol{\vartheta}_{\varphi,\theta}$ when applying SNLLS (we present further supporting evidence on this issue in the next section).

As indicated in the previous discussion, the relative performance obtained by applying the two different formulations of F will, to a large extent, depend on the dimension of $\boldsymbol{\vartheta}_\lambda$ relative to the dimension of $\boldsymbol{\vartheta}_{\varphi,\theta}$. For enhanced precision we express the relative dimension of these two parameter vectors using the fraction

$$\begin{aligned} PR &= \frac{\dim\left(\boldsymbol{\vartheta}_{\varphi,\theta}\right)}{\dim(\boldsymbol{\vartheta}_\lambda)} \\ &= \frac{t_{\vartheta_{\varphi,\theta}}}{t_{\vartheta_\lambda}} \end{aligned} \tag{23}$$

As will be demonstrated in the next section, when PR is large, estimation by SNLLS will typically be multiple times faster than estimation by NLLS.

Illustrations and Results

In this section, we provide four examples illustrating the difference in numerical efficiency across the two implementations of the quadratic form fit function F. As it turns out, it is surprisingly difficult to measure numerical efficiency. One place to start is simply to count the number of iterations and function evaluations until the optimizer reaches a satisfactory solution. However, the end goal is a faster implementation. Thus, our focus will be to evaluate numerical efficiency in terms of estimation time (i.e., the time used by the optimizer to reach a solution). Evaluating estimation time, one has to be aware that the hardware processor is at any time point busy with ongoing processes. Thus, the allocated resources assigned to the optimizer will vary, which in turn leads to variation in estimation time. It is therefore recommended to "simulate" the time by re-estimating the model a large number of times. Measures of computational efficiency are then obtained as the mean (or median) estimation time.

The first three example are pure timing experiments. In each of these examples, the model is re-estimated 1000 times using the same empirical covariance matrix as input. Thus, the only source of variation is the time it takes to estimate the model. The fourth example additionally involves simulated multivariate data. At each run in the loop, a new empirical covariance matrix is generated. The same (default) seed was used for both procedures.

Estimation is performed using Matlab (2019) and the complementary optimization toolbox. This toolbox contains a number of flexible and robust algorithms, which are well suited for our

purpose. Below, we outline the conditions under which the two procedures, SNLLS and NLLS, are implemented and evaluated:

- *Algorithm*: As in other SEM software, non-linear optimization is performed using a Quasi-Newton design. The specific implementation makes use of the BFGS (Broyden–Fletcher–Goldfarb–Shanno) Hessian update mechanism, which is the default in Matlab.

- *Gradient*: For ease of implementation, the gradient is computed using numerical differentiation. A centered design is applied, which is more time consuming, but helps prevent potential numerical problems.

- *Tolerances*: All tolerances are set to their default values (see the Matlab documentation for more details).

- *Starting values*: The numerical search procedure requires a set of starting values that are not "too far" away from the optimal solution. There are various ways to obtain such values. We apply two different types, one labeled "*fabin 3*" and the other labeled "*simple*". As indicated by the label, when the *fabin 3* type starting values are applied, all non-zero elements of $\mathbf{\Lambda}$ are determined by the non-iterative fabin 3 estimator (see Hägglund, 1982). When the *simple* type starting values are used, all non-zero elements of $\mathbf{\Lambda}$ are set to one. For both types, all elements of $\mathbf{\Phi}$ and $\mathbf{\Theta}$ are set to zero, except the variances of latent variables (set to 0.05), and the unique variances (set to half the observed variance).

  In the first three examples, we report results for both types of starting values. In the fourth example, we simplify matters by only considering the *simple* type. Note that when SNLLS is applied, no starting values for the elements of $\mathbf{\Phi}$ and $\mathbf{\Theta}$ are required.

The "timing loop" consists of the following steps:

1. The clock starts.
2. Starting values are obtained.
3. Estimation is performed. When SNLLS is applied, two functions are involved. When NLLS is applied, only one function is involved.
4. The clock stops and the recorded time is stored in a vector for later analysis.

The GLS estimator is used throughout all examples. To ensure that estimation results were similar across the two procedures, we compared the parameter estimates and function values at the minimum. For the first three examples, we also compared the estimation results to the output of established SEM software such as Mplus (Muthén & Muthén, 2017) and lavaan (Rosseel, 2012). Results were consistent across software packages.

Example 1

In the first example, we use the classic data from the Holzinger and Swineford (1939) study. The data ($n = 301$) is well known among SEM users, and consist of mental ability test scores of seventh- and eighth-grade children from two different schools in Chicago (see e.g., Jöreskog et al., 2016). As common in the literature and elsewhere (see e.g., Joreskog, 1969 or Jöreskog et al., 2016) only a subset of the data is used in this example.

The model is a 9-indicator 3-factor model with three indicators per factor. Indicators 1–3 are measures of the common factor Visual, indicators 4–6 are measures of the common factor Textual and indicators 7–9 are measures of the common factor Speed. As illustrated in the path diagram below, there are no cross loadings, but the unique factors belonging to indicators 7 and 8 are set to correlate (see Sörbom, 1989).

Insert Figure 1 About Here

As indicated in the path diagram, for identification purposes, we let X_1, X_4 and X_7 be marker variables (reference) by setting $\lambda_{11} = \lambda_{42} = \lambda_{73} = 1$. Appendix D provides an R code for estimating the model using the SNLLS procedure.

Table 1a and Table 1b below summarize the results obtained from applying the two procedures, SNLLS and NLLS, in combination with the two types of starting values described above:

Insert Table 1a and 1b About Here

In terms of iterations (*It*) and function evaluations (*Fe*), the results indicate that minimizing F using SNLLS is numerically more efficient than using NLLS. This pattern of results is recurrent throughout all the examples.

The timing results show that for the SNLLS procedure, the mean estimation times are 0.0136 sec. and 0.0153 sec., and for the NLLS procedure, the times are 0.0429 sec. and 0.0447 sec. Thus, when starting values are obtained using the *fabin 3* estimator, NLLS takes on average $0.0429/0.0136 \approx 3.15$ times longer to reach the solution. When the *simple* type starting values are used, the corresponding number is $0.0447/0.0153 \approx 2.92$.

The main conclusion that can be reached from the first illustrative example is that estimation using SNLLS is numerically more efficient than estimation using NLLS. Moreover, in this case, estimation times are not much affected by how the starting values are generated.

Example 2

Next, we consider the CFA approach to analyze Multitrait Multimethod Matrices (MTMMs). Without going too much into the specific details, the CFA approach involves forming a model based on a set of Trait factors and a set of Method factors (Campbell and Fiske, 1959). The following path diagram shows the structure of a 3–Traits by 3–Methods Correlated Traits/Correlated Methods model (a so-called 3×3 CTCM).

Insert Figure 2 About Here

When analyzing the MTMM, the CTCM model typically forms the baseline, from which several nested models are evaluated.

A recent study by Miller et. al. (2018), analyzed student behavior using a 3×3 CTCM model similar in form to the one illustrated in the path diagram given in Figure 2. The analysis was based on a sample consisting of 831 students. The variables entering the analysis were Academic Engagement (AE), Disruptive Behavior (DB) and Respectful Behavior (termed RS). In the model, indicators 1–3 are student self-reported AE, DB and RS, indicators 4–6 are teacher reported AE, DB and RS, and indicators 7–9 are AE, DB and RS obtained from SDO (a dedicated observer).

As was originally done in the study by Miller et. al. (2018), we let X_1, X_2 and X_3 be marker variables (reference) for the Traits-factors (using $\lambda_{11} = \lambda_{22} = \lambda_{33} = 1$) and X_1, X_4 and X_7 be marker variables (reference) for the Method-factors (using $\lambda_{14} = \lambda_{45} = \lambda_{76} = 1$). The structure of $\boldsymbol{\Lambda}$ in this example prevents us from applying the *simple* type starting values for the SNLLS estimator, since $\boldsymbol{\Lambda}$, and thereby $\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)$, does not have full column rank. We consider two alternative approaches for how to address this issue:

1. Applying a non-iterative procedure such as the *fabin 3* estimator.

2. Applying Ridge regression. Using this approach, the modified fit function takes the form

$$\tilde{F} = \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda) + \rho\mathbf{I})^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \tag{24}$$

   where ρ is a small constant and $\mathbf{I}$ is an $(h \times h)$ identity matrix. The values obtained from implementing $\tilde{F}$, will serve as the starting values when minimizing

$$F = \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\big(\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{G}(\boldsymbol{\vartheta}_\lambda)\big)^{-1}\boldsymbol{G}^T(\boldsymbol{\vartheta}_\lambda)\boldsymbol{V}\boldsymbol{s} \tag{25}$$

Admittedly, the second approach is somewhat ad-hoc involving some necessary user choices. First, one has to determine the value of ρ, which should be close to zero. We found that any number between 0.05 and 0.50 works well. Second, when implementing $\tilde{F}$, it is sufficient to iterate the model only few times, say 5–10. In this example, we used $\rho = 0.1$ and iterated the model 5 times.

The results from applying both procedures in combination with the two types of starting values are given in Table 2a and Table 2b:

Insert Table 2a and 2b About Here

Focusing on estimation time, the results show that when starting values are obtained from the *fabin 3* estimator, the mean estimation time for SNLLS is 0.1063 sec. and for NLLS, the time is 0.1923 sec. If the *simple* type starting values are applied, the comparable numbers are 0.2021 sec. and 0.4291 sec., respectively. Thus, on average, NLLS use about twice the time to reach the solution as compared to SNLLS.

As before, the general conclusion is that estimation using SNLLS is faster by some margin. In contrast to the previous example, for both procedures, it is evident that estimation time is somewhat sensitive to the choice of starting values.

Example 3

In this next example, data on the topics of industrialization and political democracy are considered. The data have been used extensively in many books and papers (e.g., Bollen, 1989 and Jöreskog et al., 2016), and is available from various online sources. The data are collected from 75 developing countries ($n = 75$), consisting of four indicators of political democracy, measured at two time points (1960 and 1965), and three indicators of industrialization, measured at one time point (1960).

Using only the indicators of political democracy, we form an 8-indicator 2-factor model. Indicators 1–4 are measures of the common factor Political Democracy at time 1 (1960) and indicators 5–8 are measures of the common factor Political Democracy at time 2 (1965). There are no cross loadings, but due to the repeated measurement design, each of the unique factors

at time 1 are set to correlate with their equivalents at time 2. Moreover, the unique factors belonging to indicators 2 and 4 are set to correlate and, correspondingly, the unique factors belonging to indicators 6 and 8 are set to correlate. Indicators X_1 and X_5 are marker variables (reference). The path diagram is displayed in Figure 3.

Insert Figure 3 About Here

The results are given in Table 3a and Table 3b:

Insert Table 3a and 3b About Here

The structure of the model in this example commands a slightly higher PR ($= 2.83$). Thus, a greater difference in estimation time across the two procedures is expected. The timing results reveal that when the *fabin 3* estimator is used, the mean estimation time for SNLLS is 0.0099 sec. and for NLLS, the time is 0.0885 sec. Correspondingly, using the *simple* type starting values, the results are 0.0104 sec. and 0.0746 sec. Thus, dependent on the choice of starting values, average estimation time for NLLS is in this case 7–9 times longer than for SNLLS.

In this example, we find that the time difference is quite substantial in favor of SNLLS. It is additionally observed that NLLS performs better when the *simple* type starting values are employed, whereas SNLLS performs close to the same.

Example 4

The purpose of the following example is to compare numerical efficiency between SNLLS and NLLS along two dimensions. First, we consider efficiency by simply comparing the estimation time. As the number of elements in $\boldsymbol{\vartheta}$ increases (decreases), the time it takes to estimate the model also increases (decreases) accordingly. It is of interest to study how the number of free parameters affects estimation time across the two procedures. Second, we consider numerical efficiency under conditions of changing the number of elements in $\boldsymbol{\vartheta}_{\varphi,\theta}$, relative to the number of elements in $\boldsymbol{\vartheta}_{\lambda}$. As previously conjectured, SNLLS has a comparable advantage over NLLS when the number of elements in $\boldsymbol{\vartheta}_{\varphi,\theta}$ is large relative to the number of elements in $\boldsymbol{\vartheta}_{\lambda}$ (i.e., when PR is large).

In this example, we investigate numerical performance by forming a series of models based on a repeated measurement design. The idea is to create a general modeling framework that allows us to gradually increase t_{ϑ} by adding indicators, while at the same time systematically altering PR. The models are simulated 1000 times using normal data ($n = 500$). The path diagram below illustrates the general structure of the simulated models.

Insert Figure 4 About Here

Based on the structure shown in Figure 4, a series of models is obtained by increasing the value q, which is simply the number of indicators per common factor. Letting $q = 2, 3, \ldots, 12$, we have

$$t_\vartheta = 4(q-1) + 10 + 6q\ , \qquad PR = \frac{10+6q}{4(q-1)} \tag{26}$$

A summary describing the characteristics of the simulated models is provided in following table:

Insert Table 4 About Here

From the table, we see that q and PR are inversely related. Thus, for low values of q (i.e., high values of PR), we expect that SNLLS will be relatively fast when compared to NLLS.

Insert Figure 5 About Here

From Figure 5a, as expected, we observe that estimation time increase markedly with the number of estimated parameters (as given by t_ϑ). The increase is far more dramatic when estimation is performed using NLLS. For instance, when $q = 2$, the mean estimation time is 0.0064 sec. for SNLLS and 0.1033 sec. for NLLS. At the opposite end of the scale, when $q = 12$, the corresponding numbers are 20.7761 sec. and 45.7718 sec., respectively. If one were to perform a simulation, for which $q = 12$, estimating the model 1000 times will take a little less than 6 hours for SNLLS and a little less than 13 hours for NLLS, given similar conditions and hardware. In such situations, considerable time gains can be realized by applying SNLLS.

Next, we compare the two procedures in terms of relative numerical efficiency for different values of PR. Figure 5b shows how many times longer, on average, it takes to estimate the model using NLLS over SNLLS. For instance, when $q = 2$ ($PR = 5.50$), it takes a little more

than 16 times longer, whereas when $q = 12$ ($PR = 1.86$), it takes a little more than 2 times longer. These results clearly demonstrate the numerical efficiency of SNLLS when the number of free parameters in $\boldsymbol{\Lambda}$ is small relative to the number of free parameters in $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$.

Concluding Remarks and Limitations

In the current study, we outlined an alternative implementation, labelled SNLLS, of the minimum distance (MD) estimator for estimating CFA models. Through examples and simulations, we demonstrated the potential benefits in terms of estimation time from applying SNLLS. Specifically, it was shown that considerable time gains can be realized when the model is complex and contains a large number of free parameters, or when the number of free elements in $\boldsymbol{\vartheta}_{\varphi,\theta}$ is large compared to the number of free elements in $\boldsymbol{\vartheta}_{\lambda}$. By construction, SNLLS allows the researcher to estimate $\boldsymbol{\vartheta}_{\varphi,\theta}$ using LLS whenever an estimate of $\boldsymbol{\vartheta}_{\lambda}$ is available. A feature that is also useful in connection to single equation estimation techniques, such as instrumental variables estimation (see, among others, Bollen, 1996 and Hägglund, 1982).

Simulation research, and the conclusions drawn from it, are of course potentially limited by the choices made when setting up the simulation. The examples and simulations considered in this study are undoubtedly subject to the same limitation, implying that different choices of estimators, minimization algorithms (and how they are configured), and hardware may have some effect on the realized outcomes. However, our experience from applying SNLLS is that it generally performs consistently under changing conditions.

One major limitation of this study is that we have not considered MD estimators for the case when the weighting matrix $\boldsymbol{V}$ is a function of the free parameters. Estimators of this type

includes the all-important ML estimator. Applying SNLLS in such cases is more involved and entails additional considerations. Another important limitation is that we have focused on estimating CFA models. At this point, it is unknown to what extend SNLLS can be adapted to models that extends the common CFA modeling framework. Extensions of particular interest include the full SEM, multiple group analysis with mean structures and latent growth curve models. Future research may provide solutions to the limitations outlined here.

Acknowledgments

The authors would like to thank the reviewers for their insightful comments and suggestions in the course of preparing this text.

References

Bollen, K. A. (1989). Structural equations with latent variables. Wiley. New York.

Bollen, K. A. (1996). An alternative two stage least squares (2SLS) estimator for latent variable equations. Psychometrika, 61(1), 109-121.

Browne, M. W. (1974). Generalized least squares estimators in the analysis of covariance structures. South African Statistical Journal, 8(1), 1-24.

Browne, M. W. (1977). The analysis of patterned correlation matrices by generalized least squares. British Journal of Mathematical and Statistical Psychology, 30(1), 113-124.

Browne, M. W. (1982), Covariance structures. In D. M. Hawkins (Ed.), Topics in applied multivariate analysis (pp. 72–141). Cambridge: Cambridge University Press.

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. British Journal of Mathematical and Statistical Psychology, 37(1), 62-83.

Campbell, D. T., & Fiske, D. W. (1959). Convergent and discriminant validation by the multitrait-multimethod matrix. Psychological bulletin, 56(2), 81.

Fuller, W. A. (2009). Measurement error models (Vol. 305). John Wiley & Sons.

Golub, G. H., & Pereyra, V. (1973). The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. SIAM Journal on numerical analysis, 10(2), 413-432.

Golub, G. H., & Pereyra, V. (2003). Separable nonlinear least squares: the variable projection method and its applications, Inverse Problems, 19(2).

Hägglund, G. (1982). Factor analysis by instrumental variables methods. Psychometrika, 47(2), 209-222.

Holzinger, K., and Swineford, F. (1939). A study in factor analysis: The stability of a bifactor solution. Supplementary Educational Monograph, no. 48. Chicago: University of Chicago Press.

Joreskog, K. G. (1969). A general approach to confirmatory maximum likelihood factor analysis. Psychometrika, 34, 183-202.

Jöreskog, K. G., Olsson, U. H., & Wallentin, F. Y. (2016). Multivariate analysis with LISREL. Basel, Switzerland: Springer.

Kreiberg, D., Söderström, T., Yang-Wallentin, F. (2016). Errors-in-variables system identification using structural equation modeling. Automatica, 66, 218-230

Magnus, J. R., & Neudecker, H. (1999). Matrix differential calculus with applications in statistics and econometrics. John Wiley & Sons.

Marcoulides, G. A., & Schumacker, R. E. (Eds.). (1996). Advanced structural equation modeling: Issues and techniques. Lawrence Erlbaum Associates, Inc.

Marsh, H., & Bailey, M. (1991). Confirmatory factor analysis of multitrait-multimethod data: A comparison of alternative models. Applied Psychological Measurement, 15, 47-70.

MATLAB. (2019). version R2019b. Natick, Massachusetts: The MathWorks Inc.

Muthén, B., & Asparouhov, T. (2011). Beyond multilevel regression modeling: Multilevel analysis in a general latent variable framework. In Handbook of advanced multilevel analysis (pp. 23-48). Routledge.

Muthén, L.K., & Muthén, B.O. (2017). Mplus User's Guide (8th Edition). Los Angeles, CA: Muthén & Muthén.

R Core Team (2018). R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org/

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. Journal of Statistical Software, 48, 1-36.

Sörbom, D. (1989). Model modification. Psychometrika, 54(3), 371-384.

Ye, K., & Lim, L. H. (2016). Every matrix is a product of Toeplitz matrices. Foundations of Computational Mathematics, 16(3), 577-598.

Appendix

A: Golub and Pereyra (1973)

As mentioned in the introduction, the SNLLS implementation of the quadratic fit function relies on a mathematical framework developed by Golub and Pereyra (1973). Below, we provide a summary of this framework.

Given data (t_i, y_i), for $i = 1, \dots, m$, Golub and Pereyra consider models in which the objective function can be written in the form

$$\begin{aligned} r(\mathbf{a}, \boldsymbol{\alpha}) &= \sum_{i=1}^{m} \left(y_i - \eta(\mathbf{a}, \boldsymbol{\alpha}, t_i)\right)^2 \\ &= \|\boldsymbol{y} - \boldsymbol{\eta}(\mathbf{a}, \boldsymbol{\alpha})\|^2 \end{aligned} \tag{A1}$$

where $\| \, . \|^2$ denotes the Euclidean norm and $\boldsymbol{\eta}(\mathbf{a}, \boldsymbol{\alpha})$ is a non-linear function of the disjoint parameter sets $\mathbf{a}$ and $\boldsymbol{\alpha}$. The idea of parameter separation is that the function $\boldsymbol{\eta}(\mathbf{a}, \boldsymbol{\alpha})$ can be re-expressed in a way that allows $\mathbf{a}$ to enter the objective function in a linear fashion

$$r(\mathbf{a}, \boldsymbol{\alpha}) = \|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\mathbf{a}\|^2 \tag{A2}$$

Here, $\boldsymbol{\Phi}(\boldsymbol{\alpha})$ is a tall matrix function of full column rank. Conditional on $\boldsymbol{\alpha}$, an LLS solution for the minimization of (A2) is given by

$$\mathbf{a}(\boldsymbol{\alpha}) = \left(\boldsymbol{\Phi}^T(\boldsymbol{\alpha})\boldsymbol{\Phi}(\boldsymbol{\alpha})\right)^{-1}\boldsymbol{\Phi}^T(\boldsymbol{\alpha})\boldsymbol{y} \tag{A3}$$

where T is the transpose operator. Inserting (A3) into (A2) gives

$$r_2(\boldsymbol{\alpha}) = \left\|\boldsymbol{y} - \boldsymbol{\Phi}(\boldsymbol{\alpha})\left(\boldsymbol{\Phi}^T(\boldsymbol{\alpha})\boldsymbol{\Phi}(\boldsymbol{\alpha})\right)^{-1}\boldsymbol{\Phi}^T(\boldsymbol{\alpha})\boldsymbol{y}\right\|^2 \tag{A4}$$

Now, the objective function is expressed entirely as a function of $\boldsymbol{\alpha}$.

Golub and Pereyra (1973) provides the theoretical justification underlying the implementation of (A3) and (A4). To do so, they state the following theorem:

a. If the point $\hat{\mathbf{a}}(\hat{\boldsymbol{\alpha}})$, obtained using (A.3), is a global minimizer of $r_2(\boldsymbol{\alpha})$, then the point $(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$ is also a global minimizer of $r(\mathbf{a}, \boldsymbol{\alpha})$.

b. If the point $(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$ is a global minimizer of $r(\mathbf{a}, \boldsymbol{\alpha})$, then the point $\hat{\boldsymbol{\alpha}}$ is a global minimizer of $r_2(\boldsymbol{\alpha})$, and

    1. $r_2(\hat{\boldsymbol{\alpha}}) = r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$

    2. if the point $\hat{\mathbf{a}}(\hat{\boldsymbol{\alpha}})$ is unique, it must satisfy (A3)

In practice, the procedure is first to obtain an estimate of $\boldsymbol{\alpha}$ by minimizing (A4). Once $\hat{\boldsymbol{\alpha}}$ is available, an estimate of $\mathbf{a}$ follows from the use of (A3).

B: Obtaining $\boldsymbol{L}_{\Phi}$ and $\boldsymbol{L}_{\Theta}$

In this section, we describe how to obtain the matrices $\boldsymbol{L}_{\Phi}$ and $\boldsymbol{L}_{\Theta}$. For later use, it will be helpful to outline a framework for how to obtain the matrix $\boldsymbol{K}_{\Sigma}$ $(p^2 \times h)$. The approach taken here follows the one in Fuller (2009, Ch. 4, Appendix 4.a), which deviates from Browne (1974), but leads to the same result. As will be demonstrated below, the generality of the presented framework allows us to handle covariance matrices of various forms.

Let $\boldsymbol{\Sigma}$ $(p \times p)$ be a covariance matrix with elements $\boldsymbol{\Sigma}(i,j) = \sigma_{ij}$ for $i,j = 1, \dots, p$. A vector consisting of the non-duplicated elements of $\boldsymbol{\Sigma}$ is obtained using the expression

$$\begin{aligned} \boldsymbol{\sigma} &= \boldsymbol{K}_{\Sigma}^{T}\mathrm{vec}(\boldsymbol{\Sigma}) \\ &= \left(\sigma_{11}\ \ \sigma_{21}\ \ \dots\ \ \sigma_{p1}\ \ \sigma_{22}\ \ \sigma_{32}\ \ \dots\ \ \sigma_{p2}\ \ \dots\ \ \dots\ \ \sigma_{pp}\right)^{T} \end{aligned} \tag{B1}$$

where $\boldsymbol{\sigma}$ is a vector with length h.

In order to derive $\boldsymbol{K}_{\Sigma}$, we first introduce the matrix $\boldsymbol{E}^{ij}$ $(p \times p)$, for $i,j = 1, \dots, p$. For every distinct element in $\boldsymbol{\Sigma}$, as given by $\boldsymbol{\Sigma}(i,j) = \sigma_{ij}$ for $i \geq j$, there exists an associated matrix $\boldsymbol{E}^{ij}$ with elements

$$\boldsymbol{E}^{ij}(u,v) = \begin{cases} 1 & \text{if } (u,v) = (i,j) \\ 1 & \text{if } (u,v) = (j,i)\,, \\ 0 & \text{Otherwise} \end{cases} \quad \text{for } u,v = 1, \dots, p \tag{B2}$$

Next, we form a new matrix, denoted $\boldsymbol{L}_{\Sigma}$ $(p^2 \times h)$, by concatenating a series of vectors based on $\boldsymbol{E}^{ij}$

$$\begin{aligned} \boldsymbol{L}_{\Sigma} = (&\mathrm{vec}(\boldsymbol{E}^{11})\ \ \mathrm{vec}(\boldsymbol{E}^{21})\ \dots\ \ \mathrm{vec}(\boldsymbol{E}^{p1})\ \ \mathrm{vec}(\boldsymbol{E}^{22})\ \ \mathrm{vec}(\boldsymbol{E}^{32})\ \dots \\ &\mathrm{vec}(\boldsymbol{E}^{p2})\ \ \dots\ \ \dots\ \ \mathrm{vec}(\boldsymbol{E}^{pp})\big) \end{aligned} \tag{B3}$$

This matrix is known as the duplication matrix (see also Magnus and Neudecker, 1999). The matrix $\boldsymbol{K}_{\Sigma}$ is found by the expression

$$\boldsymbol{K}_{\Sigma} = (\boldsymbol{L}_{\Sigma}^{T}\boldsymbol{L}_{\Sigma})^{-1}\boldsymbol{L}_{\Sigma}^{T} \tag{B4}$$

The outlined framework allows us to find $\boldsymbol{L}$ and $\boldsymbol{K}$ for covariance matrices of various forms relevant in covariance analysis.

First, consider the covariance matrix $\boldsymbol{\Phi}$ $(m \times m)$. Given that no other restrictions, apart from symmetry, are placed on the elements of $\boldsymbol{\Phi}$, we have

$$\begin{aligned} \boldsymbol{L}_{\Phi} = (&\mathrm{vec}(\boldsymbol{E}^{11})\ \ \mathrm{vec}(\boldsymbol{E}^{21})\ \dots\ \ \mathrm{vec}(\boldsymbol{E}^{p1})\ \ \mathrm{vec}(\boldsymbol{E}^{22})\ \ \mathrm{vec}(\boldsymbol{E}^{32})\ \dots \\ &\mathrm{vec}(\boldsymbol{E}^{m2})\ \ \dots\ \ \dots\ \ \mathrm{vec}(\boldsymbol{E}^{mm})\big) \end{aligned} \tag{B5}$$

where $\boldsymbol{L}_{\Phi}$ is a matrix with dimension $\left(m^2 \times 2^{-1}m(m+1)\right)$.

Second, consider the covariance matrix $\boldsymbol{\Theta}$ $(p \times p)$. In the simplest (and possibly most common) case, we have that δ_i and δ_j, for $i \neq j$, are uncorrelated. It obviously follows that $\boldsymbol{\Theta}$ is a diagonal matrix with distinct elements only on the diagonal. In such a case, we have

$$\boldsymbol{L}_{\Theta} = \left(\text{vec}(\boldsymbol{E}^{11}) \;\; \text{vec}(\boldsymbol{E}^{22}) \;\; \ldots \;\; \text{vec}(\boldsymbol{E}^{pp})\right) \tag{B6}$$

with dimension $(p^2 \times p)$.

C: Equality constraints

Incorporating equality constraints into $\boldsymbol{\Sigma}$ requires the previously presented framework to be extended somewhat.

As before, consider a generic element $\boldsymbol{\Sigma}(i,j)$ belonging to the lower half (including the diagonal) of $\boldsymbol{\Sigma}$ (i.e., $i \geq j$). Suppose that there are $q \geq 1$ additional elements restricted to be the same value as $\boldsymbol{\Sigma}(i,j)$. Formally, we express this as

$$\boldsymbol{\Sigma}(i,j) = \boldsymbol{\Sigma}(k_1,l_1) = \cdots = \boldsymbol{\Sigma}\left(k_q,l_q\right) \tag{C1}$$

where $(k_1,l_1),\ldots,\left(k_q,l_q\right)$, for $k_1 \geq l_1,\ldots,k_q \geq l_q$, is a series of index sets describing the position of the constrained elements. The elements of $\boldsymbol{E}^{ij}$ are now found by

$$\boldsymbol{E}^{ij}(u,v) = \begin{cases} 1 & \text{if } \; (u,v) \in \left\{(i,j),(k_1,l_1),\ldots,\left(k_q,l_q\right)\right\} \\ 1 & \text{if } \; (u,v) \in \left\{(j,i),(l_1,k_1),\ldots,\left(l_q,k_q\right)\right\} \\ 0 & \text{Otherwise} \end{cases} \tag{C2}$$

for $u,v = 1,\ldots,p$

As in the previous cases, applying the expression for $\boldsymbol{E}^{ij}$, requires knowledge about the exact structure of $\boldsymbol{\Sigma}$. As an example, let $\boldsymbol{\Sigma}$ be a Toeplitz matrix (see e.g., Ye and Lim (2016)) of the form

$$\boldsymbol{\Sigma}_{Toep} = \begin{pmatrix} \sigma_1 & \sigma_2 & \ldots & \sigma_p \\ \sigma_2 & \sigma_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sigma_2 \\ \sigma_p & \ldots & \sigma_2 & \sigma_1 \end{pmatrix} \tag{C3}$$

The number of distinct elements in $\boldsymbol{\Sigma}_{Toep}$ is p. Covariance matrices with a Toeplitz structure are relevant in repeated measurement analysis where individuals' are observed at multiple occasions. Such analysis include multilevel analysis and longitudinal analysis. Obtaining $\boldsymbol{K}_{\Sigma_{Toep}}$ $(p^2 \times p)$ involves

$$\boldsymbol{L}_{\Sigma_{Toep}} = \left(\text{vec}(\boldsymbol{E}^{11}) \;\; \text{vec}(\boldsymbol{E}^{21}) \;\; \ldots \;\; \text{vec}(\boldsymbol{E}^{p1})\right) \tag{C4}$$

Using $\boldsymbol{L}_{\Sigma_{Toep}}$, we now get

$$\boldsymbol{K}_{\Sigma_{Toep}} = \left(\boldsymbol{L}_{\Sigma_{Toep}}^{T}\boldsymbol{L}_{\Sigma_{Toep}}\right)^{-1}\boldsymbol{L}_{\Sigma_{Toep}}^{T} \tag{C5}$$

D: R code for example 1

```
################################################################################
#                                                                              #
#          A FASTER PROCEDURE FOR ESTIMATING CFA MODELS APPLYING MINIMUM       #
#                              DISTANCE ESTIMATORS                             #
#                                                                              #
# The code below demonstrates how to implement the SNLLS estimation procedure. #
# The model used in the demonstration is the well-known 9-indicator 3-factor   #
# model based on the Holzinger and Swineford (1939) data.                      #
#                                                                              #
#                   (created by David Kreiberg 12-07-2020)                     #
#                                                                              #
################################################################################

# FUNCTIONS
# Step 1 SNLLS fit function
fit.SNLLS1 <- function(lambda, p, m, s, V, K.SIGMA, L.PHI, G2) {
  LAMBDA <- matrix(c(      1.0,         0,         0,
                     lambda[1],         0,         0,
                     lambda[2],         0,         0,
                             0,       1.0,         0,
                             0, lambda[3],         0,
                             0, lambda[4],         0,
                             0,         0,       1.0,
                             0,         0, lambda[5],
                             0,         0, lambda[6]),
                   p, m, byrow = TRUE)

  G1 <- t(K.SIGMA)%*%kronecker(LAMBDA, LAMBDA)%*%L.PHI; G <- cbind(G1, G2)
  F <- t(s)%*%V%*%s - t(s)%*%V%*%G%*%solve((t(G)%*%V%*%G), (t(G)%*%V%*%s))
}

# Step 2 SNLLS fit function
fit.SNLLS2 <- function(lambda, p, m, s, V, K.SIGMA, L.PHI, G2) {
  LAMBDA <- matrix(c(      1.0,         0,         0,
                     lambda[1],         0,         0,
                     lambda[2],         0,         0,
                             0,       1.0,         0,
                             0, lambda[3],         0,
                             0, lambda[4],         0,
                             0,         0,       1.0,
                             0,         0, lambda[5],
```

```
                              0,         0, lambda[6]),
                    p, m, byrow = TRUE)

  G1 <- t(K.SIGMA)%*%kronecker(LAMBDA, LAMBDA)%*%L.PHI; G <- cbind(G1, G2)
  var.theta2 <- solve((t(G)%*%V%*%G), (t(G)%*%V%*%s))
}

# Selection matrix L for covariance matrices with
# 0.5*p*(p+1) distinct elements
L1 <- function(p) {
  L <- matrix(0, p^2, 0)
  for(i in 1:p) {
    for(j in 1:p) {
      if (j >= i) {
        E = matrix(0, p, p)
        E[j, i] <- 1; E[i, j] <- 1
        L <- cbind(L, matrix(E, p^2, 1))
      }
    }
  }
  L
}

# Selection matrix L for covariance matrices with
# distinct elements determined by a set of index pairs
L2 <- function(p, index.sets) {
  nr <- dim(index.sets)[1];
  L <- matrix(0, p^2, 0)
  for(i in 1:nr) {
    k <- index.sets[i, 1]; l <- index.sets[i, 2];

    E = matrix(0, p, p)
    E[k, l] <- 1; E[l, k] <- 1
    L <- cbind(L, matrix(E, p^2, 1))
  }
  L
}

# LOAD THE DATA
library(lavaan)
X <- HolzingerSwineford1939
```

```
# MODEL CHRACTERISTICS
# Specify number of observed indicators and latent constructs
p <- 9; m <- 3

# COMPUTATIONS
L.SIGMA <- L1(p); L.PHI <- L1(m)
K.SIGMA <- L.SIGMA%*%solve(t(L.SIGMA)%*%L.SIGMA)

index.sets <- matrix(c(1:9, 8,
                       1:9, 7), 10, 2)
L.THETA <- L2(p, index.sets); G2 <- t(K.SIGMA)%*%L.THETA

# Obtain covariance matrix and covariance vector
S <- cov(X[ ,(7:15)]); s <- t(K.SIGMA)%*%matrix(S, p^2, 1)

# Compute GLS weight matrix
V <- 0.5*t(L.SIGMA)%*%(kronecker(solve(S), solve(S)))%*%L.SIGMA

# ESTIMATION
# Specify starting values (estimator = fabin 3)
lambda0 <- c(0.778, 1.107, 1.133, 0.924, 1.225, 0.854)

# Perform optimization
# Step 1 estimation
Est.step1 <- nlminb(lambda0, fit.SNLLS1, gradient = NULL, hessian = NULL,
                    p, m, s, V, K.SIGMA, L.PHI, G2)
var.theta1 <- Est.step1$par

# Step 2 estimation
var.theta2 <- fit.SNLLS2(var.theta1, p, m, s, V, K.SIGMA, L.PHI, G2)


# PRINT RESULTS
# Prepare results
Est.LAMBDA.visual <- matrix(c(1.0, var.theta1[1:2]), 3, 1,
                            dimnames = list(c("X1", "X2", "X3"),
                                            c("Visual")))

Est.LAMBDA.textual <- matrix(c(1.0, var.theta1[3:4]), 3, 1,
                             dimnames = list(c("X4", "X5", "X6"),
                                             c("Textual")))
```

```
Est.LAMBDA.speed <- matrix(c(1.0, var.theta1[5:6]), 3, 1,
                           dimnames = list(c("X7", "X8", "X9"),
                                           c("Speed")))

Est.PHI <- matrix(var.theta2[1:6], 6, 1,
                  dimnames = list(c("Visual , Visual  :",
                                    "Visual , Textual :",
                                    "Visual , Speed   :",
                                    "Textual, Textual :",
                                    "Textual, Speed   :",
                                    "Speed  , Speed   :"),
                                  c("PHI est.")))

Est.THETA <- matrix(var.theta2[7:16], 10, 1,
                    dimnames = list(c("X1, X1 :",
                                      "X2, X2 :",
                                      "X3, X3 :",
                                      "X4, X4 :",
                                      "X5, X5 :",
                                      "X6, X6 :",
                                      "X7, X7 :",
                                      "X8, X8 :",
                                      "X9, X9 :",
                                      "X8, X7 :"),
                                    c("THETA est.")))

# Display computations and parameter estimates
# Min. fit function value:
print(Est.step1$objective)

# Parameter estimates:
print(Est.LAMBDA.visual)
print(Est.LAMBDA.textual)
print(Est.LAMBDA.speed)
print(Est.PHI)
print(Est.THETA)
```

	SNLLS	NLLS
Mean estimation time (in sec.)	0.0136	0.0429
Median estimation time (in sec.)	0.0135	0.0427
St. dev. estimation time (in sec.)	5.6746e-04	6.3719e-04
Minimum fit function value	0.1778	0.1778
Number of iterations (*It*)	21	51
Number of func. evaluations (*Fe*)	286	2385
PR	2.67	2.67

Table 1a: Timing results: Holzinger and Swineford, Starting values = *fabin 3* estimator

	SNLLS	NLLS
Mean estimation time (in sec.)	0.0153	0.0447
Median estimation time (in sec.)	0.0152	0.0446
St. dev. estimation time (in sec.)	2.4632e-04	6.6542e-04
Minimum fit function value	0.1778	0.1778
Number of iterations (*It*)	22	53
Number of func. evaluations (*Fe*)	325	2475
PR	2.67	2.67

Table 1b: Timing results: Holzinger and Swineford, Starting values = *simple*

	SNLLS	NLLS
Mean estimation time (in sec.)	0.1063	0.1923
Median estimation time (in sec.)	0.1062	0.1914
St. dev. estimation time (in sec.)	0.0089	0.0034
Minimum fit function value	0.0228	0.0228
Number of iterations (*It*)	50	163
Number of func. evaluations (*Fe*)	1300	11055
PR	1.75	1.75

Table 2a: Timing results: Miller et., al., Starting values = *fabin 3* estimator

	SNLLS	NLLS
Mean estimation time (in sec.)	0.2021	0.4291
Median estimation time (in sec.)	0.2029	0.4274
St. dev. estimation time (in sec.)	0.0138	0.0055
Minimum fit function value	0.0228	0.0228
Number of iterations (*It*)	71	359
Number of func. evaluations (*Fe*)	2275	24723
PR	1.75	1.75

Table 2b: Timing results: Miller et., al., Starting values = *simple* with Ridge constant = 0.1

	SNLLS	NLLS
Mean estimation time (in sec.)	0.0099	0.0885
Median estimation time (in sec.)	0.0098	0.0882
St. dev. estimation time (in sec.)	3.5261e-04	0.0018
Minimum fit function value	0.1661	0.1661
Number of iterations (*It*)	13	114
Number of func. evaluations (*Fe*)	195	5405
PR	2.83	2.83

Table 3a: Timing results: Bollen. Starting values $=$ *fabin 3* estimator

	SNLLS	NLLS
Mean estimation time (in sec.)	0.0104	0.0746
Median estimation time (in sec.)	0.0104	0.0745
St. dev. estimation time (in sec.)	1.9914e-04	8.6866e-04
Minimum fit function value	0.1661	0.1661
Number of iterations (*It*)	15	94
Number of func. evaluations (*Fe*)	208	4512
PR	2.83	2.83

Table 3b: Timing results: Bollen. Starting values $=$ *simple*

q	2	3	4	5	6	7	8	9	10	11	12
# indi.	8	12	16	20	24	28	32	36	40	44	48
t_ϑ	26	36	46	56	66	76	86	96	106	116	126
PR	5.50	3.50	2.83	2.50	2.30	2.17	2.07	2.00	1.94	1.90	1.86

Table 4: Values for q, t_ϑ and PR (# indi. $=$ number of indicators)

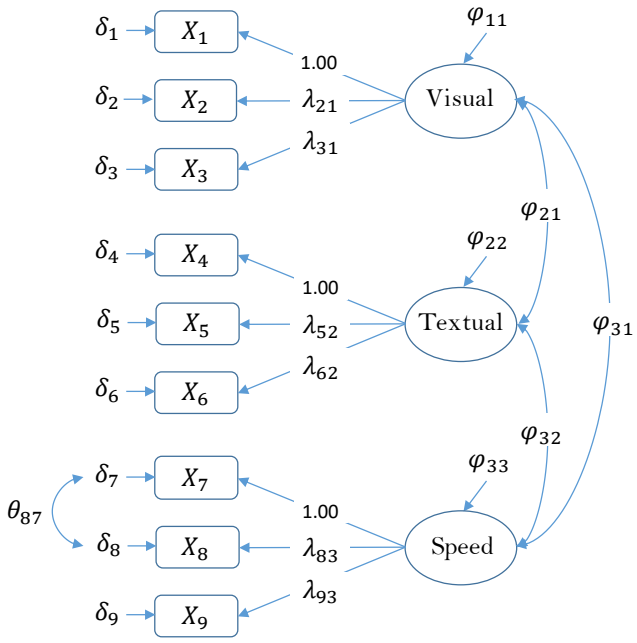

Figure 1: Holzinger and Swineford (1939)

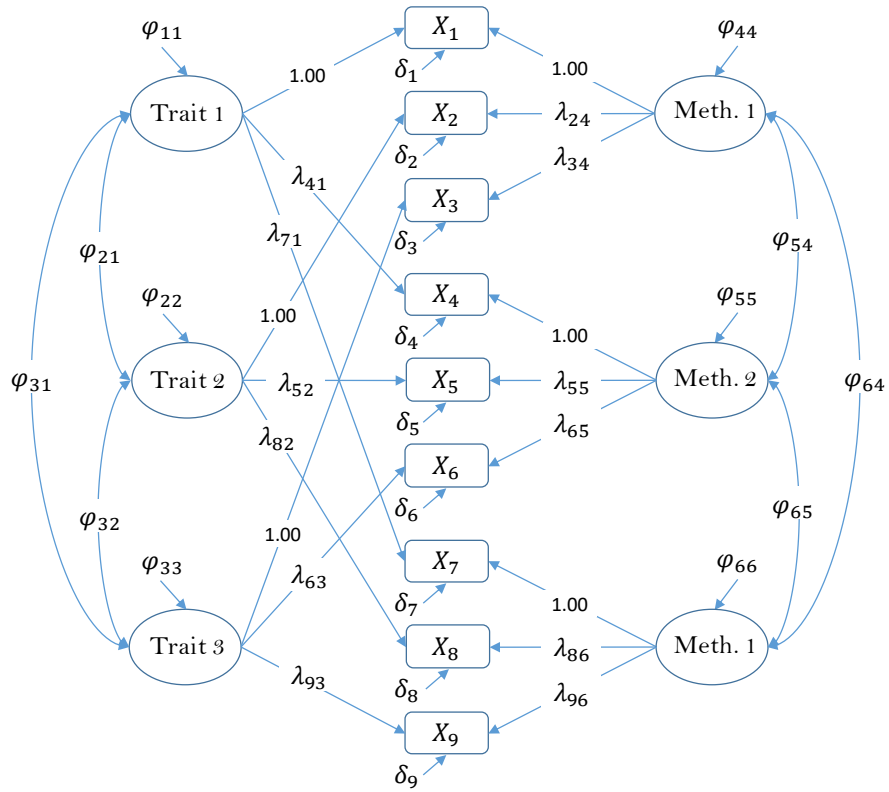

Figure 2: Miller et. al. (2018)

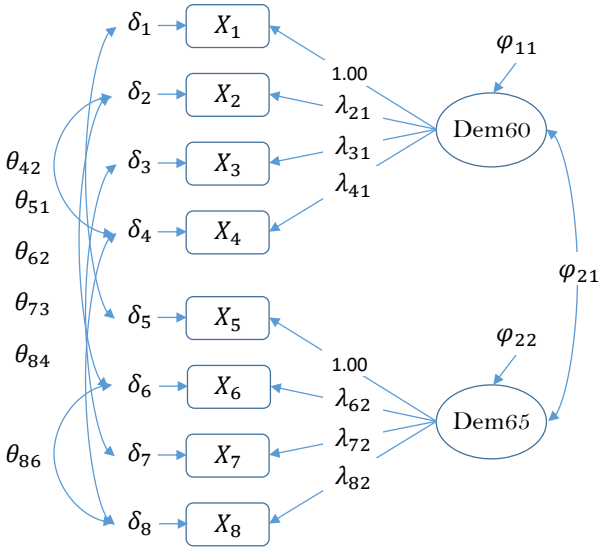

Figure 3: Bollen (1989)

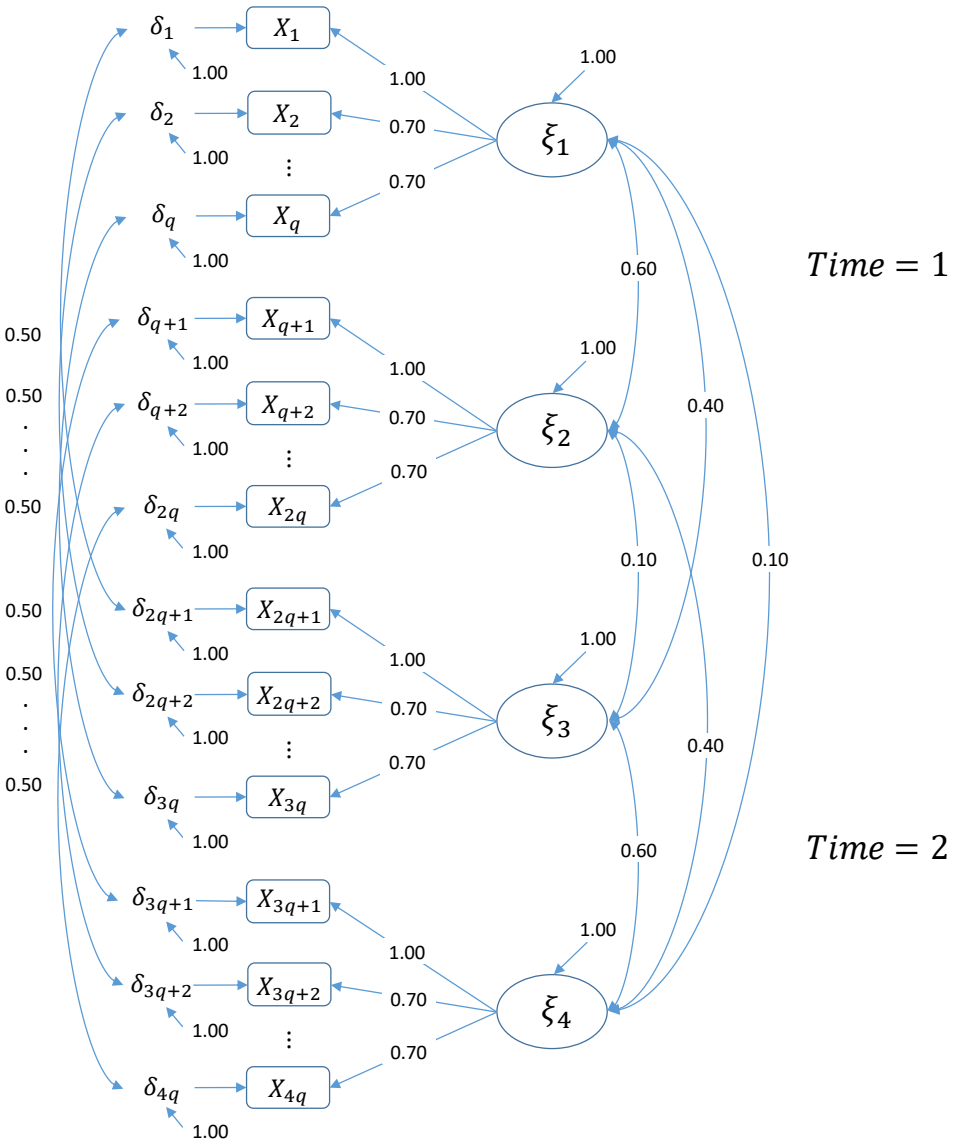

Figure 4: The population Model for the Repeated Measure study

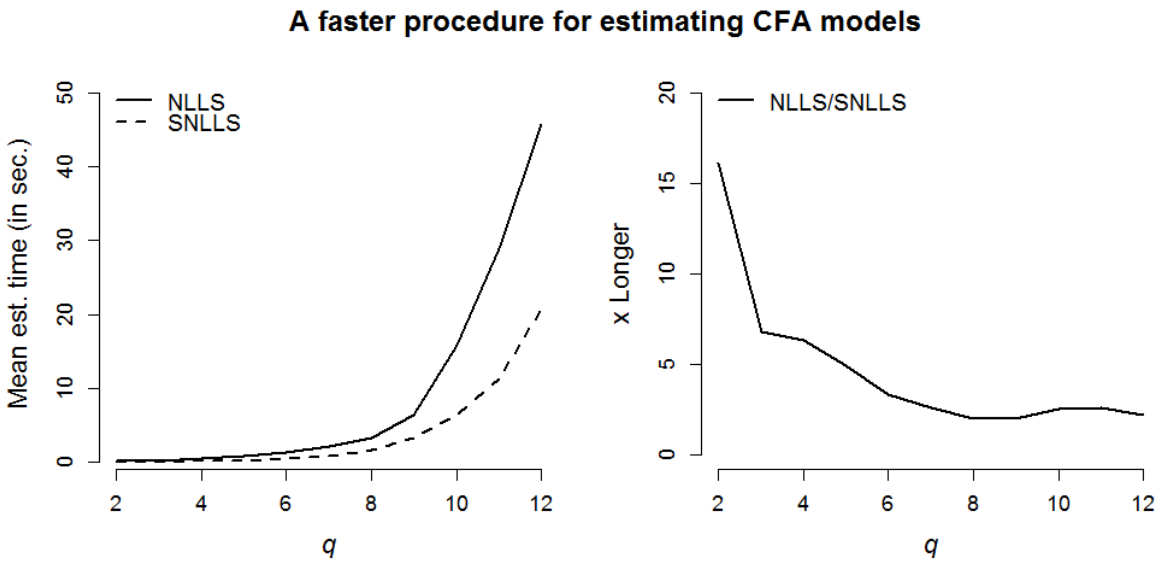

Figure 5a and 5b: Summarizing the simulation results for the two procedures