A Faster Procedure for Estimating SEMs Applying Minimum Distance Estimators With a Fixed Weight Matrix

David Kreiberg & Xingwu Zhou

To cite this article: David Kreiberg & Xingwu Zhou (2022): A Faster Procedure for Estimating SEMs Applying Minimum Distance Estimators With a Fixed Weight Matrix, Structural Equation Modeling: A Multidisciplinary Journal, DOI: 10.1080/10705511.2022.2076093

To link to this article: https://doi.org/10.1080/10705511.2022.2076093

© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC

Published online: 10 Oct 2022.

Submit your article to this journal

View related articles

View Crossmark data
A Faster Procedure for Estimating SEMs Applying Minimum Distance Estimators With a Fixed Weight Matrix

David Kreiberg and Xingwu Zhou

ABSTRACT
This study presents a separable nonlinear least squares (SNLLS) implementation of the minimum distance (MD) estimator employing a fixed-weight matrix for estimating structural equation models (SEMs). In contrast to the standard implementation of the MD estimator, in which the complete set of parameters is estimated using nonlinear optimization, the SNLLS implementation allows a subset of parameters to be estimated using (linear) least squares (LS). The SNLLS implementation possesses a number of benefits, such as faster convergence, better performance in ill-conditioned estimation problems, and fewer required starting values. The present work demonstrates that SNLLS, when applied to SEM estimation problems, significantly reduces the estimation time. Reduced estimation time makes SNLLS particularly useful in applications involving some form of resampling, such as simulation and bootstrapping.

1. Introduction
This study addresses the application of separable nonlinear least squares (SNLLS) when performing covariance structure analysis (CSA). SNLLS was first introduced by Golub and Pereyra (1973), who showed that for a certain type of nonlinear estimation problems, a subset of parameters can be estimated using numerically efficient least squares (LS). As will be discussed below, several studies have shown that parameter separation offers a number of numerical benefits, such as faster convergence, better performance when the estimation problem is ill-conditioned (i.e., problems in which the ratio between the largest and the smallest singular value of the covariance matrix is large), and fewer required starting values.

SNLLS is typically applied to problems involving some form of nonlinear regression analysis, but not exclusively so. A recent study by Kreiberg et al. (2021) suggested an SNLLS implementation of the minimum distance (MD) estimator for estimating confirmatory factor analysis (CFA) models. The motivation for the current study is to generalize the results in Kreiberg et al. (2021) by outlining an SNLLS implementation for estimating structural equation models (SEMs). This is important for several reasons. First, it makes SNLLS applicable to a wider range of models. Second, at this stage, little is known about the potential benefits of applying SNLLS in the context of CSA. The outlined SNLLS implementation may pave the way for future research on how to improve the numerical performance of CSA based estimators.

To make the idea of SNLLS clearer, consider the familiar MD quadratic form objective function

$$F(\theta) = \left(s_\theta - \sigma_\theta(\theta)\right)^T V \left(s_\theta - \sigma_\theta(\theta)\right),$$

where \(s_\theta\) and \(\sigma_\theta(\theta)\) are covariance vectors derived from the sample and the model, respectively, \(\theta\) is the parameter vector and \(V\) is a weighting matrix chosen by the user. We consider the case in which \(V\) is a fixed matrix (i.e., when \(V\) is not a function of \(\theta\)). Such cases include well-known estimators such as unweighted least squares (ULS), generalized least squares (GLS), and weighted least squares (WLS). The standard implementation of Equation (1) is a one-step estimation procedure, here referred to as nonlinear least squares (NLLS), that involves the use of nonlinear optimization techniques. Estimation is performed by searching the parameter space for the value of \(\theta\) that minimizes Equation (1). In contrast, the SNLLS implementation of Equation (1) is a two-step estimation procedure that works by splitting \(\theta\) into two subsets. In the first step, one subset of parameters is estimated using nonlinear optimization. In the second step, based on the estimates obtained in the first step, the remaining subset of parameters is estimated using LS. As demonstrated in Kreiberg et al. (2021), SNLLS provides parameter estimates and a minimum objective function value identical to those obtained using NLLS. It obviously follows that the asymptotic properties of the estimator are...
maintained. The presentation below presents a general framework for how to accomplish parameter separation in the case of SEMs.

Over the years, SNLLS has become popular in applied research across a wide range of scientific disciplines. Golub and Pereyra (2003) compiled a list of real-world examples of SNLLS applications. Mullen (2008) subsequently provided a comprehensive overview of SNLLS for a number of applications in physics and chemistry. SNLLS has also proved useful in systems and control applications. For instance, Söderström et al. (2009), Söderström & Mossberg (2011), and Kreiberg et al. (2016) applied CSA to handle the errors-in-variables (EIV) estimation problem. The work in these studies showed how to implement the MD estimator using SNLLS.

Several studies have documented that the SNLLS implementation of nonlinear estimators offers a number of benefits. For instance, Sjöberg and Viberg (1997) evaluated the numerical performance of SNLLS when applied to neural-network minimization problems. Their main conclusions were that SNLLS provides faster convergence and performs better in cases in which the estimation problem is ill-conditioned. A recent study by Dattner et al. (2020) investigated how to implement the MD estimator using SNLLS.

Before presenting the SEM framework, it will be useful to provide a brief overview of the SEM framework and the associated MD estimator. Section 3 outlines how to modify the MD objective function to accommodate the SNLLS implementation of the estimator when applied to SEMs. Section 4 compares the numerical efficiency of SNLLS and NLLS when applied to real-world estimation problems. Finally, Section 5 presents some concluding remarks.

2. Background

2.1. Notation

Before presenting the SEM framework, it will be useful to introduce the following notation. Let $x$ be a $p \times 1$ zero-mean random vector, and let $\Sigma_x$ be the associated $p \times p$ covariance matrix given by

$$\Sigma_x = E[x x^T],$$

where $E$ is the expectation operator and the superscript $T$ is the transpose of a vector or a matrix. The number of nonredundant elements in $\Sigma_x$ is $h = 2^{-1}p(p+1)$, given that no restrictions other than symmetry are placed on the elements of $\Sigma_x$. A covariance vector containing the nonredundant elements (i.e., the lower half of $\Sigma_x$ including the diagonal) is

$$\sigma_x = \text{vech}(\Sigma_x).$$

In this expression, $\text{vech}$ is the operation of vectorizing the nonredundant elements of $\Sigma_x$. Alternatively, $\sigma_x$ is obtained by

$$\sigma_x = K_x^T \text{vec}(\Sigma_x).$$

Here, vec is the operation of vectorizing the elements of a matrix by stacking its columns, and $K_x$ is a $p^2 \times h$ matrix obtained from

$$K_x = L_x (L_x^T L_x)^{-1},$$

where $L_x$ is a $p^2 \times h$ selection matrix containing only ones and zeros. This matrix has the additional usage

$$\text{vec}(\Sigma_x) = L_x \sigma_x.$$
The nonredundant elements of available. An estimate of 
likewise. The noise vectors 
\( \mathbf{r} \) and its nonredundant elements are required.

Suppose that a sample of data points 
\( x_i \) and 
\( x_2 = A_2 \mathbf{Z} + \mathbf{e}_2 \). 
(15)
The first equation is the structural equation, which specifies 
the causal relationships among the latent variables. In this 
equation, \( \eta \) and \( \xi \) are respectively \( p_x \times 1 \) and 
\( p_\xi \times 1 \) random vectors, \( \delta \) is a \( p_x \times 1 \) random 
noise vector, and \( \mathbf{B} \) and \( \mathbf{G} \) are respectively 
\( p_x \times p_\xi \) and \( p_\xi \times p_\xi \) parameter matrices relating 
the latent random vectors. The last two equations are 
measurement equations. In these equations, \( x_1 \) and \( x_2 \) are 
respectively \( p_x \times 1 \) and \( p_\xi \times 1 \) observed random vectors, \( \mathbf{e}_1 \) 
and \( \mathbf{e}_2 \) are noise vectors of similar dimensions, and \( A_1 \) and 
\( A_2 \) are respectively \( p_x \times p_\xi \) and \( p_\xi \times p_\xi \) parameter 
matrix. Under suitable conditions, and for the right choice 
of \( \mathbf{V} \), the MD estimator is consistent and asymptotically 
normal. Note that consistency does not depend on \( \mathbf{V} \) as 
long as \( \mathbf{V} \) converges in probability to a symmetric positive 
definite matrix.

Using a proper algorithm, Equation (21) is minimized by 
numerically searching the parameter space until some 
convergence criterion is satisfied. For the estimation problem to 
be feasible, it is a necessary condition that the number of 
elements in \( \mathbf{s}_x - \mathbf{\sigma}_x(\mathbf{G}) \) is at least as large as the number 
of free parameters in \( \mathbf{G} \).

3. Modifying the MD Quadratic Form Objective Function

Next, we outline how to modify the objective function in 
Equation (21) to accommodate the SNLLS implementation. 
To do so, we need some additional notation. Let \( \mathbf{\theta}_{\mathbf{p}, \xi, \cdot} \) be a 
\( \mathbf{t}_{\mathbf{p}, \xi, \cdot} \times 1 \) vector containing the free elements in \( \mathbf{B}, \mathbf{G}, \mathbf{A}_1, \) 
and \( \mathbf{A}_2 \), and let \( \mathbf{\sigma}_{\xi, \cdot} \) be a 
\( \mathbf{t}_{\mathbf{p}, \xi, \cdot} \times 1 \) vector containing the free elements in 
\( \mathbf{\sigma}_{\xi}, \mathbf{\sigma}_{\xi}, \mathbf{\sigma}_{\xi}, \mathbf{\sigma}_{\xi}, \) and \( \mathbf{\sigma}_{\xi, \cdot} \), 
and let \( \mathbf{H} = (\mathbf{I} - \mathbf{B})^{-1} \). The covariance matrix implied by Equations 
(13)–(15) is
\[
\Sigma_x(\mathbf{G}) = \begin{pmatrix}
\mathbf{H} \Sigma \mathbf{H}^T + \Sigma_{\xi} & \mathbf{H} \Sigma \mathbf{A}_1^T + \Sigma_T \\
\mathbf{A}_1 \Sigma \mathbf{H}^T + \Sigma_{\xi} & \mathbf{A}_1 \Sigma \mathbf{A}_1^T + \Sigma_{\xi, \cdot}
\end{pmatrix}.
\]
(18)

2.3. The MD Estimator

Suppose that a sample of data points \( x_i \) for \( i = 1, \ldots, N \) 
is available. An estimate of \( \Sigma_x \) is then computed using
\[
\hat{\Sigma}_x = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T.
\]
(19) 
Given \( \Sigma_x \), the aim is to estimate the true parameter vector 
\( \mathbf{G} \). An estimate of \( \mathbf{G} \) is obtained by
\[
\hat{\mathbf{G}} = \arg \min F(\mathbf{G}),
\]
(20)

where \( F(\mathbf{G}) \) is a scalar function that expresses the distance 
between the observed and the model-implied covariance 
structure. Below, we focus on the MD objective function 
given by
\[
F(\hat{\mathbf{G}}) = (\mathbf{s}_x - \mathbf{\sigma}_x(\hat{\mathbf{G}}))^T \mathbf{V}(\mathbf{s}_x - \mathbf{\sigma}_x(\hat{\mathbf{G}})).
\]
(21)

In this expression, \( \mathbf{s}_x \) and \( \mathbf{\sigma}_x(\hat{\mathbf{G}}) \) are vectors containing 
the nonredundant elements of \( \Sigma_x \) and \( \Sigma_x(\hat{\mathbf{G}}) \), respectively. That is,
\[
\mathbf{s}_x = \mathbf{K}_x^T \text{vec}(\Sigma_x), \quad \mathbf{\sigma}_x(\hat{\mathbf{G}}) = \mathbf{K}_x^T \text{vec}(\Sigma_x(\hat{\mathbf{G}})).
\]
(22)

Moreover, the matrix \( \mathbf{V} \) is a positive definite 
definite matrix.
where \( \tilde{s} \) and \( \tilde{V} \) correspond to \( s \) and \( V \), respectively, but with their rows and columns rearranged according to the order in \( \tilde{\sigma} \). For some value of \( G(\theta, \gamma, \lambda) \), the solution to the problem of minimizing Equation (27) w.r.t. \( \sigma_{3,6} \) is a straightforward application of LS

\[
\tilde{\sigma}_{3,6}(\theta, \gamma, \lambda) = \left( G(\theta, \gamma, \lambda) \tilde{V} G(\theta, \gamma, \lambda)^{-\dagger} \right)^{-1} G(\theta, \gamma, \lambda)^{-\dagger} \tilde{V} \tilde{s}.
\]  

(28)

Since \( \tilde{\sigma}_{3,6} \) depends on \( \theta, \gamma, \lambda \), it is necessary to outline how to obtain an estimate \( \hat{\theta}, \hat{\gamma}, \hat{\lambda} \) without directly involving \( \sigma_{3,6} \). Theorem 2.1 in Golub and Pereyra (1973) provides the justification for replacing \( \sigma_{3,6} \) in Equation (27) with the right-hand side of Equation (28). Doing so, leads to the modified objective function

\[
F(\theta, \gamma, \lambda) = \tilde{\sigma}^T \tilde{V} \tilde{s} - \tilde{\sigma}^T \tilde{V} G(\theta, \gamma, \lambda)^{-1} G(\theta, \gamma, \lambda)^{-\dagger} \tilde{V} \tilde{s}.
\]  

(29)

Apart from some slight notational differences, the derivation of Equation (29) is similar to the derivation in Kreiberg et al. (2021). From the preceding presentation, it follows that SNLLS is a two-step procedure. In the first step, \( \hat{\theta}, \hat{\gamma}, \hat{\lambda} \) is obtained by minimizing Equation (29) applying nonlinear optimization. In the second step, using \( \hat{\theta}, \hat{\gamma}, \hat{\lambda} \) from the first step, \( \hat{\sigma}_{3,6} \) is obtained by Equation (28).

The major benefit of the formulation in Equation (29) is that the minimization w.r.t. \( \theta, \gamma, \lambda \) represents a lower dimensional optimization problem. Thus, the computational load when minimizing \( F(\theta, \gamma, \lambda) \) w.r.t. \( \theta, \gamma, \lambda \) is smaller, and in some cases by a considerable margin, than what is the case when minimizing Equation (21) w.r.t. \( \theta \). This is especially the case when the number of elements in \( \sigma_{3,6} \) is large compared with the number of elements in \( \theta, \gamma, \lambda \).

4. Illustrations

This section provides two examples that illustrate the difference in numerical efficiency between the two implementations, SNLLS and NLLS, of the MD estimator when applied to SEMs. Numerical performance is assessed by studying the convergence of the optimizer and the time it takes the optimizer to reach its minimum. Since timing depends on other processes running on the device performing the estimation, it is recommended to compute the average estimation time over multiple runs. Estimation and timing are performed using Matlab (2020, version R2020b). The two implementations are compared under the following conditions:

- **Algorithm:** The optimizer is a Quasi-Newton (QN) design applying the Broyden–Fletcher–Goldfarb–Shanno (BFGS) Hessian update mechanism (default in Matlab).
- **Gradient:** For simplicity, the gradient is computed using a finite difference approach. The computation is based on a centered design, which is supposed to provide greater accuracy at the expense of being more time-consuming.

- **Tolerances:** Tolerances are set to their default values (details are found in the Matlab documentation).
- **Starting values:** Starting values are taken from the open-source R (R Core Team, 2021) package lavaan (Rosseel, 2012). The starting values for the free elements are as follows:
  - \( A_1 \) and \( A_2 \) are computed using the non-iterative fabin 3 estimator (see Hagglund, 1982).
  - \( B \) and \( I \) are set to zero.
  - \( \Sigma_3 \) and \( \Sigma_6 \) are set to zero except for the diagonal elements, which are set to 0.05.
  - \( \Sigma_1 \) and \( \Sigma_5 \) are set to zero except for the diagonal elements, which are set to half the observed variance. For the examples below, no starting values are required for the elements in \( \Sigma_{1,5,6} \).

- **Estimator:** The GLS estimator is used throughout the examples. The GLS estimator uses a weight matrix of the form

\[
\tilde{V} = 2^{-1} L_s^T (S^{-1} \otimes S^{-1}) L_s.
\]  

(30)

- **Timing:** In each example, the model is re-estimated 1000 times using the same empirical covariance matrix as input.

To ensure that our programming is correct, we compared the estimation results to the results obtained using lavaan.

4.1. Example 1

The first example considers a model for the medical illness of depression. The data (\( N = 323 \)) used in this example are taken from Geiser (2012) and consist of six indicators of depression. In the data, \( X_{1,1} \) and \( X_{1,2} \) are indicators of the first-order common factor Depression State 1, \( X_{1,3} \) and \( X_{1,4} \) are indicators of the first-order common factor Depression State 2, and \( X_{1,5} \) and \( X_{1,6} \) are indicators of the first-order common factor Depression State 3. The three factors themselves are indicators of the second-order common trait factor Depression. The model additionally contains an indicator-specific factor labeled IS. Indicators \( X_{1,1}, X_{1,2}, X_{1,3}, X_{1,5}, \) and the factor Depression State 1 serve as marker variables. The path diagram illustrating the structure of the model is shown in Figure 1.

![Figure 1. Geiser (2012).](image-url)
Results of the estimation are presented in Table 1. As seen from the table, the number of iterations and function evaluations is \((It, Fe) = (23, 375)\) for SNLLS and \((It, Fe) = (145, 5439)\) for NLLS. As expected, the required computational load for minimizing \(F(\theta_{\beta, \gamma, \lambda})\) w.r.t. \(\theta_{\beta, \gamma, \lambda}\) is far less than the required load for minimizing \(F(\boldsymbol{\theta})\) w.r.t. \(\boldsymbol{\theta}\). Figure 2 shows the convergence profiles for the two implementations. From the figure, it is clear that the SNLLS objective function \(F(\theta_{\beta, \gamma, \lambda})\) starts at a point much closer to its minimum of 0.0109 than what is seen for the NLLS objective function \(F(\theta)\). In terms of estimation time, the mean time is 0.0334 sec. for SNLLS and 0.1408 sec. for NLLS. Thus, SNLLS is faster by a factor of 0.1408/0.0334 = 4.2153. The results in this example clearly suggest that the SNLLS implementation is numerically more efficient than the standard NLLS implementation.

4.2. Example 2

The second example considers a model for industrialization and political democracy. The model is taken from Bollen (1989), and has been used extensively in books, tutorials, etc. The data consist of 11 indicators of industrialization and political democracy for 75 countries \((N = 75)\). In the data, \(X_{1,1}, ..., X_{1,4}\) are indicators of the common factor Political Democracy at time 1 (1960), \(X_{1,5}, ..., X_{1,8}\) are indicators of the common factor Political Democracy at time 2 (1965) and \(X_{2,1}, ..., X_{2,3}\) are indicators of the common factor Industrialization at time 1 (1960). Due to the repeated measurement design, the unique factors belonging to \(X_{1,i}\) and \(X_{1,i+4}\) for \(i = 1, ..., 4\) are set to correlate. Additionally, the unique factors belonging to \(X_{1,i}\) and \(X_{1,i+2}\) for \(i = 2, 6\) are set to correlate. Indicators \(X_{1,1}\), \(X_{1,5}\) and \(X_{2,1}\) serve as marker variables.

Results of the estimation are presented in Table 2. The results in this example generally confirm the results from the previous example. In this case, the number of iterations and function evaluations are \((It, Fe) = (26, 759)\) for SNLLS and \((It, Fe) = (230, 14742)\) for NLLS. Figure 4 shows the convergence profiles for the two implementations. The patterns in the figure resemble those in Figure 2. Considering the estimation time, the mean time is 0.1257 sec. for SNLLS and 0.8263 sec. for NLLS. In this case, SNLLS proves to be faster by a factor of 0.8263/0.1257 = 6.5736.

5. Concluding Remarks

In this study, we have presented an SNLLS implementation of the MD objective function for estimating SEMs. The outlined framework includes all necessary expressions for applying SNLLS, and represents a generalization of

<table>
<thead>
<tr>
<th>Table 1. Timing results, Geiser (2012).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td><strong>SNLLS</strong></td>
</tr>
<tr>
<td>Mean est. time (in sec.)</td>
</tr>
<tr>
<td>Median est. time in (sec.)</td>
</tr>
<tr>
<td>Standard dev. est. time in (sec.)</td>
</tr>
<tr>
<td>Minimum objective func. value</td>
</tr>
<tr>
<td>Number of iterations ((It))</td>
</tr>
<tr>
<td>Number func. evaluations ((Fe))</td>
</tr>
<tr>
<td>(t_{SNLLS}/t_{NLLS})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Timing results, Bollen (1989).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td><strong>SNLLS</strong></td>
</tr>
<tr>
<td>Mean est. time (in sec.)</td>
</tr>
<tr>
<td>Median est. time in (sec.)</td>
</tr>
<tr>
<td>Standard dev. est. time in (sec.)</td>
</tr>
<tr>
<td>Minimum objective func. value</td>
</tr>
<tr>
<td>Number of iterations ((It))</td>
</tr>
<tr>
<td>Number func. evaluations ((Fe))</td>
</tr>
<tr>
<td>(t_{SNLLS}/t_{NLLS})</td>
</tr>
</tbody>
</table>

Figure 2. Convergence profile, Geiser (2012).

Figure 3. Bollen (1989).
previously known results. Using examples from the SEM literature, we demonstrated that the computational load of applying SNLLS is considerably less than that of applying NLLS. Another benefit of SNLLS is that fewer starting values are required, which may mitigate potential problems due to the somewhat arbitrary choice of starting values for the covariance parameters.

The present work may have several interesting extensions. First, as shown by research, SNLLS may hold a potential for improving numerical performance in situations in which the estimation problem is ill-conditioned. Thus, an interesting case for future research would be to compare the numerical performance of SNLLS and NLLS under more challenging conditions in which the condition number of the observed covariance matrix is large. Second, the SNLLS implementation is not yet available for maximum likelihood (ML) estimation. Some initial work on this topic is underway. This work, combined with the previous point, may lead to an improved implementation of the ML estimator when applied to SEMs.

References


Appendices

A. Deriving L and K

Let x be an arbitrary \( p \times 1 \) random vector, and let \( \Sigma_x = \{\sigma_{ij}\} \) be the associated \( p \times p \) covariance matrix. The purpose of the following presentation is to introduce an algebraic framework that facilitates eliminating the redundancy originating from the structure of \( \Sigma_x \). To do so, let \( K_x \) be a matrix such that

\[
\sigma_x = K_x^T \text{vec}(\Sigma_x),
\]

where \( \sigma_x \) is a covariance vector containing the nonredundant elements of \( \Sigma_x \) and \( K_x \) is a matrix obtained by

\[
K_x = L_x (L_x^T L_x)^{-1}.
\]

In this expression, \( L_x \) is a selection matrix (i.e., a matrix composed of zeros and ones).

Below, we propose a rather general framework that applies to any structure characterizing \( \Sigma_x \). Before presenting some examples on how to obtain \( L_x \), it is necessary to introduce some additional notation. Let \( E(u, v) = \{e_{ij}(u, v)\} \) denote a \( p \times p \) matrix (for \( i, j = 1, ..., p \)) with elements

\[
e_{ij}(u, v) = \begin{cases} 1 & \text{if } \sigma_{ij} = \sigma_{uv} \\ 0 & \text{otherwise} \end{cases}
\]

![Figure 4. Convergence profile, Bollen (1989).](image-url)
Next, we demonstrate how to obtain \( L \) for two standard cases and one case specialized for the SNLSS implementation.

**Case 1:** As a start, consider the case in which \( \Sigma \) is symmetric and no other restrictions are placed on its elements. The covariance vector containing the \( 2^{-1}p(p + 1) \) nonredundant elements of \( \Sigma \) is

\[
\sigma = (\sigma_{11} \ldots \sigma_{p1} \sigma_{22} \ldots \sigma_{p2} \sigma_{33} \ldots \sigma_{pp})^T. \tag{A4}
\]

Applying (A3), the matrix \( L \) is formed by horizontally concatenating \( 2^{-1}p(p + 1) \) vectors using

\[
L = \begin{pmatrix}
\text{vec}(E(1,1)) & \text{vec}(E(p,1)) & \text{vec}(E(2,2)) & \text{vec}(E(p,2)) \\
\text{vec}(E(3,3)) & \ldots & \text{vec}(E(p,p))
\end{pmatrix}. \tag{A5}
\]

**Case 2:** Now, consider the case in which \( \Sigma \) is diagonal. The covariance vector containing the \( p \) nonredundant elements of \( \Sigma \) is given by

\[
\sigma = (\sigma_{11} \ldots \sigma_{22} \ldots \sigma_{pp})^T. \tag{A6}
\]

The matrix \( L \) is now formed by horizontally concatenating \( p \) vectors

\[
L = \begin{pmatrix}
\text{vec}(E(1,1)) & \text{vec}(E(2,2)) & \ldots & \text{vec}(E(p,p))
\end{pmatrix}. \tag{A7}
\]

Before introducing the third and final case, it is necessary to expand the notation. Let \( x_1 \) and \( x_2 \) be respectively \( p_1 \times 1 \) and \( p_2 \times 1 \) random vectors, and let \( x \) be a \( p = p_1 + p_2 \) dimensional column vector obtained by stacking \( x_1 \) and \( x_2 \) in the following way

\[
x = \begin{pmatrix} x_1^T & x_2^T \end{pmatrix}^T. \tag{A8}
\]

The associated \( p \times p \) covariance matrix is given by

\[
\Sigma = \begin{pmatrix}
\Sigma_{i,i} & \Sigma_{i,j} \\
\Sigma_{j,i} & \Sigma_{j,j}
\end{pmatrix}_{(p_1 \times p_2) \times (p_1 \times p_2)} \tag{A9}
\]

**Case 3:** As in the first case, suppose that no other restrictions, apart from symmetry, are placed on the elements of \( \Sigma \). Let the covariance vector containing the \( 2^{-1}p(p + 1) \) nonredundant elements of \( \Sigma \) be given by

\[
\tilde{\sigma} = (\sigma_{11} \ldots \sigma_{p1} \sigma_{22} \ldots \sigma_{p2} \sigma_{33} \ldots \sigma_{pp})^T, \tag{A10}
\]

where

\[
\sigma_{i} = (\sigma_{i1} \ldots \sigma_{ip1} \sigma_{i2} \ldots \sigma_{ip2} \sigma_{i3} \ldots \sigma_{ip})^T, \tag{A11}
\]

\[
\sigma_{ij} = (\sigma_{ip1+pj1} \ldots \sigma_{ip1+pj2} \ldots \sigma_{ip1+pj3} \ldots \sigma_{ip})^T, \tag{A12}
\]

\[
\sigma_{ij} = (\sigma_{ip1+pj1} \ldots \sigma_{ip1+pj2} \ldots \sigma_{ip1+pj3} \ldots \sigma_{ ip})^T. \tag{A13}
\]

Construct a matrix \( \tilde{L} \) by horizontally concatenating three matrices

\[
\tilde{L} = \begin{pmatrix} (\tilde{L}_1)_{1,1} & (\tilde{L}_1)_{1,2} & (\tilde{L}_1)_{1,3} \\
(\tilde{L}_1)_{2,1} & (\tilde{L}_1)_{2,2} & (\tilde{L}_1)_{2,3} \\
(\tilde{L}_1)_{3,1} & (\tilde{L}_1)_{3,2} & (\tilde{L}_1)_{3,3}
\end{pmatrix} \tag{A14}
\]

where the submatrices are given by

\[
(\tilde{L}_1)_{1,1} = \begin{pmatrix}
\text{vec}(E(1,1)) & \text{vec}(E(p,1)) & \text{vec}(E(2,2)) & \text{vec}(E(p,2)) \\
\text{vec}(E(3,3)) & \ldots & \text{vec}(E(p,p))
\end{pmatrix}. \tag{A15}
\]

\[
(\tilde{L}_1)_{1,2} = \begin{pmatrix}
\text{vec}(E(p_1 + 1, p_1 + 1)) & \ldots & \text{vec}(E(p_1 + 1, p_2)) & \text{vec}(E(p_1 + 2, p_1)) & \ldots & \text{vec}(E(p_1 + 2, p_2)) \\
\text{vec}(E(p_1 + 3, p_1)) & \ldots & \text{vec}(E(p_1 + 3, p_2))
\end{pmatrix}. \tag{A16}
\]

\[
(\tilde{L}_1)_{1,3} = \begin{pmatrix}
\text{vec}(E(p_1 + 1, 1)) & \ldots & \text{vec}(E(p_1 + 1, 2)) & \ldots & \text{vec}(E(p_2, 1)) \\
\text{vec}(E(p_1 + 1, 3)) & \ldots & \text{vec}(E(p_2, p))
\end{pmatrix}. \tag{A17}
\]

The number of columns in (A15), (A16), and (A17) is \( 2^{-1}p_1(p_1 + 1) \), \( 2^{-1}p_2(p_2 + 1) \) and \( p_2 \times p_1 \), respectively.

**B. Deriving \( G(\vartheta_{p,y_k}) \)**

The derivation below uses the following matrix identity

\[
\text{vec}(ABC) = (C^T \otimes A) \text{vec}(B), \tag{B1}
\]

where \( A, B, \) and \( C \) are matrices of compatible sizes. In addition, we make use of the following relations

\[
\text{vec}(\Sigma) = L_1 \sigma_{zz}, \quad \text{vec}(\Sigma) = L_2 \alpha_{zz}. \tag{B2}
\]

The model-implied covariance matrix is

\[
\Sigma(\vartheta) = \begin{pmatrix}
\Sigma_{i,i}(\vartheta) & \Sigma_{i,j}(\vartheta) \\
\Sigma_{j,i}(\vartheta) & \Sigma_{j,j}(\vartheta)
\end{pmatrix} = \begin{pmatrix}
A_1 \Sigma_{1,1} \Sigma_{1,1}^T + \Sigma_{1,1} & A_1 \Sigma_{1,1} \Sigma_{2,1}^T + \Sigma_{2,1} \\
A_1 \Sigma_{1,1} \Sigma_{2,1}^T + \Sigma_{2,1} & A_1 \Sigma_{2,2} \Sigma_{2,2}^T + \Sigma_{2,2}
\end{pmatrix} \tag{B3}
\]

Applying SNLSS, the key is to express the model-implied covariance vector using the form

\[
\tilde{\sigma}(\vartheta) = \begin{pmatrix}
\sigma_{i,i}(\vartheta) & \sigma_{i,j}(\vartheta) & \sigma_{j,i}(\vartheta) & \sigma_{j,j}(\vartheta)
\end{pmatrix}^T := G(\vartheta_{p,y_k}) \sigma_{\vartheta,\vartheta}. \tag{B4}
\]

To do so, it is necessary to vectorize the individual blocks of (B3).

Starting with the block \( \Sigma_{i,i}(\vartheta) \), we have

\[
\sigma_{i,i}(\vartheta) = K_i^T \text{vec}(\Sigma_{i,i}(\vartheta)) = K_i^T (A_1 \Sigma_{1,1} \Sigma_{1,1}^T + \Sigma_{1,1} \Sigma_{2,1}^T + \Sigma_{2,1} \Sigma_{2,2} \Sigma_{2,2}^T + \Sigma_{2,2}) \tag{B5}
\]

Using (B1) and (B2), it follows that

\[
\sigma_{i,j}(\vartheta) = K_i^T (A_1 \Sigma_{1,1} \Sigma_{1,1}^T + \Sigma_{1,1} \Sigma_{2,1}^T + \Sigma_{2,1} \Sigma_{2,2} \Sigma_{2,2}^T + \Sigma_{2,2}) \tag{B6}
\]

Next, we consider the block \( \Sigma_{j,i}(\vartheta) \). Using the same procedure as before, we have

\[
\sigma_{j,i}(\vartheta) = K_j^T \text{vec}(\Sigma_{j,i}(\vartheta)) = K_j^T (A_2 \Sigma_{2,2} \Sigma_{2,2}^T + \Sigma_{2,2} \Sigma_{1,1}^T + \Sigma_{1,1}) \tag{B7}
\]

where \( K_i^T = (A_1 \otimes A_2) L_i \) and \( K_j^T = (A_2 \otimes A_1) L_j \).
Finally, for the block $\Sigma_{x,1}(\theta)$, it follows that
\[
\sigma_{x,1}(\theta) = \text{vec}(\Sigma_{x,1}(\theta))
\]
\[
= \text{vec}(A_2 \Sigma_{x,1}^{-1} H^T A_1^T) + \text{vec}(\Sigma_{x,1})
\]
\[
= (A_1 H \Gamma \otimes A_2) \text{vec}(\Sigma_\chi) + \text{vec}(\Sigma_{x,1})
\]
\[
= (A_1 H \Gamma \otimes A_2) L_\chi \sigma_\chi + \sigma_{x,1}
\]
\[
= ((A_1 H \Gamma \otimes A_2) L_\chi 0 0 0 1) \sigma_{x,1}.\]

Putting the pieces together, we obtain
\[
\begin{pmatrix}
\sigma_{x,1}(\theta) \\
\sigma_{x,2}(\theta) \\
\sigma_{x,1,1}(\theta)
\end{pmatrix}
\]
\[
= \begin{pmatrix}
K_{x,1}^T (A_1 H \Gamma \otimes A_1 H \Gamma) L_\zeta & K_{x,2}^T (A_1 H \otimes A_1 H) L_\zeta & K_{x,1}^T L_\chi & 0 & 0 \\
K_{x,2}^T (A_1 H \Gamma \otimes A_2) L_\zeta & 0 & 0 & K_{x,2}^T L_\chi & 0 \\
(A_1 H \Gamma \otimes A_2) L_\chi & 0 & 0 & 0 & 1
\end{pmatrix}
\times \sigma_{x,1,1}.\]