First and second-order derivatives for CP and INDSCAL

Jorge Tendeiro, Mohammed Bennani Dosse, Jos M.F. ten Berge

University of Groningen, The Netherlands

ABSTRACT

In this paper we provide the means to analyse the second-order differential structure of optimization functions concerning CANDECOMP/PARAFAC and INDSCAL. Closed-form formulas are given under two types of constraint: unit-length columns or orthonormality of two of the three component matrices. Some numerical problems that might occur during the computation of the Jacobian and Hessian matrices are addressed. The use of these matrices is illustrated in three applications.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Carroll and Chang [3] and Harshman [5] independently presented two identical methods to analyse three-way arrays. The former is CANDECOMP and the latter is PARAFAC; the method is now well known as CANDECOMP/PARAFAC or simply CP. Given a \( p \times q \times m \) array \( M \) with frontal \( p \times q \) slices \( M_i \), \( i = 1, \ldots, m \), CP aims at finding the component matrices \( X \) \( (p \times r) \), \( Y \) \( (q \times r) \) and \( D \) \( (m \times r) \) that minimize the function

\[
f(X,Y,D) = \sum_{i=1}^{m} ||M_i - XD||^2.
\]

where \( D \) is the diagonal matrix holding row \( i \) of \( D \) in the diagonal. Minimizing \( f \) can be done in various ways. Carroll and Chang [3] and Harshman [5] proposed an alternating least-squares method that has become known as the CP decomposition. However, other approaches have also been proposed. For instance, Paatero [12] has offered a conjugate gradient algorithm.

The CP decomposition starts by initializing \( X \), \( Y \) and \( D \), and alternately updates each component matrix while the others remain constant. Iterations are terminated when the relative improvement in \( f \) is smaller than a predefined threshold. It is not guaranteed that \( f \) converges; if it does converge, it is not guaranteed that the global minimum is reached. To increase the chances of finding the seeked minimum it is desirable to start CP with several initialization values.

For the special case when the array has symmetric frontal slices, say \( S_1, \ldots, S_m \) of order \( p \times p \), Carroll and Chang [3] proposed INDSCAL, which minimizes the function

\[
g(X,D) = \sum_{i=1}^{m} ||S_i - XD||^2.
\]

Since minimizing \( g \) directly seems difficult, Carroll and Chang [3] suggested minimizing \( f \) instead. They conjectured that, after convergence, \( X \) and \( Y \) will be equal or, at least, columnwise proportional (i.e., the columns of \( Y \) can be rescaled to match the columns of \( X \), while the columns of \( D \) absorb the inverse scaling). Such matrices will be referred to as being equivalent.

Carroll and Chang’s conjecture seems to be valid in practical applications. However, counter-examples have already been constructed. Ten Berge and Kiers [16] proved that equivalence may be violated at global minima of \( f \) if the slices \( S_i \) are indefinite. When the slices are non-negative definite and \( r = 1 \) then equivalence can be violated only at stationary points that do not correspond to global minima. Ten Berge and Kiers [16] conjectured that such stationary points would be local minima. However, Bennani Dosse and Ten Berge [1] proved that such stationary points must be saddle points. This was achieved by analysing the first and second-order derivatives of a specific optimization function derived from the loss function of CP. Notice that the result by Bennani Dosse and Ten Berge [1] concerns the case where \( r = 1 \) component is used. The conjecture of Carroll and Chang seems to be an open issue when \( r > 1 \) components are used. In this paper, we aimed at finding a second-order sufficient condition that classifies CP...
function Notice that a similar problem applies also to INDSCAL and its associate determining the nature of stationary points of indeterminacy. Also, the new optimization functions that will be derived might affect the search and quality of an optimal solution for paper extends beyond the study of the equivalence problem. In fact, we settled for two types of constraint: it involves no loss of

Another source of freedom that needs to be controlled is directly related to the fact that the CP model is overparameterized. Namely, given diagonal matrices \( \mathbf{A}_i, \mathbf{A}_j, \mathbf{A}_k \) such that \( \mathbf{A}_i \mathbf{A}_j \mathbf{A}_k = \mathbf{I} \), both \( (\mathbf{X}, \mathbf{Y}, \mathbf{D}) \) and \( (\mathbf{X}_d, \mathbf{Y}_d, \mathbf{D}_d) \) represent the same solution. This scaling indeterminacy is considered to be trivial in CP. Nevertheless it does pose a problem when optimizing \( \mathbf{f} \) using differential tools since once \( \mathbf{f}(\mathbf{X}, \mathbf{Y}, \mathbf{D}) = \mathbf{f}(\mathbf{X}_d, \mathbf{Y}_d, \mathbf{D}_d) \), ie, for each \( (\mathbf{X}, \mathbf{Y}, \mathbf{D}) \) in the domain of \( \mathbf{f} \) there is an infinity of points which are mapped onto \( \mathbf{f}(\mathbf{X}, \mathbf{Y}, \mathbf{D}) \). This has the effect of making the second-order sufficient conditions useless, since in these conditions the Hessian matrix will invariably fail to be non-singular. Therefore, determining the nature of stationary points of \( \mathbf{f} \) via its second-order differential structure becomes unsatisfactory under the current setting. Notice that a similar problem applies also to INDSCAL and its associate function \( g \), since an INDSCAL solution is also characterized by scaling indeterminacy. Also, the new optimization functions that will be derived from \( f \) and \( g \) suffer from the same problem. Since the analysis of second-order structures is one of the goals in this paper, something had to be done to overcome this issue. Constraining the domain of the optimization functions is a possible solution to the problem discussed in the previous paragraph. We settled for two types of constraint: \( \mathbf{X} \) and \( \mathbf{Y} \) constrained to hold unit length columns (Case I), and \( \mathbf{X} \) and \( \mathbf{Y} \) constrained by orthonormality (Case II). The first constraint is a so-called identification constraint; it involves no loss of fit. The second constraint is active, thus a loss of fit is due to happen when compared to the unconstrained situation. Both constraints proved to eliminate the problem of singularity of the Hessian matrix in the vast majority of the cases. Some exceptions were found, as will be discussed in later sections.

The utility of the second-order conditions that we present in this paper extends beyond the study of the equivalence problem. In fact, minimizing \( \mathbf{f} \) is not a straightforward optimization problem. First of all, there is usually no closed-form solution. Moreover, a solution might not even exist. For example, the \( 2 \times 2 \times 2 \) symmetric slice array analysed by Ten Berge, Kiers and De Leeuw [17] showed that the loss function (1.1) has an infimum which is not a minimum. More recently, Stegeman [13] showed that (1.1) does not have a minimum when \( p \times p \times 2 \) arrays of rank \( p + 1 \) or higher are decomposed into \( p \) rank-1 arrays and a residual array (see also Stegeman [14] for a follow-up). Other problems that might affect the search and quality of an optimal solution for \( \mathbf{f} \) are:

2. Derivatives of matrix functions with respect to matrix variables

2.1. Notation

Scalars will be denoted by lower case italic font \((a, x, \lambda)\), vectors by lower case bold-face font \((\mathbf{a}, \mathbf{x}, \lambda)\), matrices by upper case bold-face font \((\mathbf{A}, \mathbf{X}, \Lambda)\), and arrays by underlined upper case bold-face font \((\mathbf{A}, \mathbf{X}, \Lambda)\). Given matrix \( \mathbf{X}, \mathbf{x} \) denotes the \( i \)-th column of \( \mathbf{X} \). The only exceptions to this rule appear in definitions (3.2) and (4.2).

For given matrices \( \mathbf{A} \) and \( \mathbf{B} \), \( \mathbf{A}^\top \) is the transpose of \( \mathbf{A} \); \( \text{tr}(\mathbf{A}) \) is the trace of \( \mathbf{A} \); \( \text{vec}(\mathbf{A}) \) reshapes \( \mathbf{A} \) into a column vector by stacking the columns in sequence, one below the other; \( \mathbf{A} \odot \mathbf{B}, \mathbf{A} \odot \mathbf{B} \) denote the Kronecker, Hadamard and Khatri-Rao products of \( \mathbf{A} \) and \( \mathbf{B} \), respectively; and \( \text{diag}_{m}(\mathbf{A}) \) is the column vector holding the diagonal of \( \mathbf{A} \). Given a vector \( \mathbf{d} \), \( \text{diag}_{m}(\mathbf{d}) \) denotes the diagonal matrix whose diagonal is equal to \( \mathbf{d} \). \( \mathbf{I}_n \) is the identity matrix of order \( n \); \( \mathbf{0}_{m \times n} \) is the zero matrix of order \( m \times n \); \( \mathbf{C}_{mm} \) is the \( mn \) \( \times mn \) commutation matrix, i.e., \( \mathbf{C}_{mn} \text{vec}(\mathbf{A}) = \text{vec}(\mathbf{A}^\top) \); \( \mathbf{T} \) is the \( n^2 \times n \) matrix with unit entries in
derivatives whose entry \( (i, j) \) is the Jacobian matrix of function \( f: \mathbb{R}^n \rightarrow \mathbb{R}^m \) is the \( m \times n \) matrix of partial derivatives whose entry \((i, j) = \frac{\partial f_i(x)}{\partial x_j} \) for \( x \in \mathbb{R}^n \). This notion of Jacobian matrix can be extended to matrix functions with matrix variables: the Jacobian matrix of function \( A: \mathbb{R}^{p \times r} \rightarrow \mathbb{R}^{m \times n} \) is the \( mn \times pr \) matrix given by

\[
A_{ij} = \frac{\partial A^{ij}}{\partial X^{kl}};
\]

Given a scalar function \( f: \mathbb{R}^n \rightarrow \mathbb{R} \), the associated Hessian matrix \( \frac{\partial^2 f}{\partial X^2} \) is the \( n \times n \) matrix whose entry \((i, j) = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \) for \( x \in \mathbb{R}^n \). The concept of Hessian matrix can be extended to scalar functions with scalar variables as follows: the Hessian matrix of function \( f: \mathbb{R}^p \rightarrow \mathbb{R} \) is the \( pr \times pr \) matrix given by

\[
H_f = \frac{\partial^2 f(x)}{\partial X^2}.
\]

This is how the partial derivatives will be arranged in the sequel. For example, the Jacobian matrix of a scalar function will be a row vector. Also, all differential formulae that will be introduced are adapted to this definition. Notice that there exist authors who choose to display the partial derivatives of the Jacobian and Hessian matrices in a different way than the one done in the present paper. See, for example, Magnus and Neudecker [110, Chapter 9] for a discussion on this subject. Therefore, some caution is needed before going into the derivations of the next sections.

2.3. Matrix differentiation formulas

In Table A1 (see Appendix A) we summarize the most important formulas of matrix differentiation that are of use in this paper. In Tables B1 and C1 (see Appendices B and C) we define functions \( F_1 \rightarrow F_8 \) and \( G_1 \rightarrow G_{16} \), for which we present the relevant partial derivatives. These functions appear useful during the differentiation process, as they simplify the presentation of our results. In some functions we add the superscript \((i)\) to the dependency of the function on the value of \( i = 1, \ldots, m \).

3. Optimization of CP

The loss function of CP (1.1) can be written as

\[
f(X, Y; D) = \sum_{i=1}^{m} \left( \| M_i \|^2 + \text{tr}(YD_iX_iD_iY_i) - 2\text{tr}(YM_iX_iD_iY_i) \right).
\]  

At stationary points we have

\[
d_i = \text{transposed row } i \text{ of } D = (X^T X)^{-1} \text{diag}(X^T Y)
\]  

and

\[
\text{tr}(YD_iX_iD_iY_i) = \text{tr}(YM_iX_iD_iY_i).
\]  

Formula (3.2) can be directly derived from the equation \( \frac{\partial}{\partial Y} = 0 \); it allows to express \( D \) in terms of \( X \) and \( Y \). Equality (3.3) can be seen as follows: define \( e_i = \text{diag}(YM_iX_i) \), and verify that \( \text{tr}(YM_iX_iD_i(YD_iX_iD_iY_i)) = e_i, d_i \text{, tr}(YM_iX_iD_iY_i) = d_i, e_i \). Thus, to optimize the loss function of CP we can work with function

\[
f(X, Y; D) = \sum_{i=1}^{m} \left( \| M_i \|^2 - \text{tr}(YM_iX_iD_iY_i) \right).
\]

for \( D \) defined as in Eq. (3.2). Minimizing Eq. (3.4) is equivalent to maximizing

\[
L^C_{\text{CP}}(X, Y) = \sum_{i=1}^{m} \text{tr}(YM_iX_iD_i),
\]

for \( D \) defined by Eq. (3.2).

We wish to describe a sufficient condition for a stationary point of \( L^C_{\text{CP}}(X, Y) \) to be a (local) maximum. In order to do this, we will derive the Jacobian and Hessian matrices for \( L^C_{\text{CP}}(X, Y) \) in two different scenarios: \( XY \) constrained to hold columns of unit length (Case I), and \( XY \) constrained by orthonormality (Case II). The constrained situations will be dealt with by introducing Lagrange multipliers:

\[
L^C_{\text{CP}}(X, Y; \lambda_i) = \sum_{i=1}^{m} \text{tr}(YM_iX_iD_i) - \text{tr}(A(X^T X - I)) - \text{tr}(\lambda_i Y^T Y - I)).
\]

In the case that \( X \) and \( Y \) are constrained to have unit length columns we have that \( A = \text{Diag}([\lambda_i]) \) and \( \Lambda = \text{Diag}([\delta_i]) \) are diagonal \( r \times r \) matrices holding Lagrange multipliers, and \( D \) is given by Eq. (3.2) with the diagonal of \( X^T X + YY^T \) filled with 1’s. If \( X \) and \( Y \) are constrained by orthonormality then \( A = [\lambda_i] \) and \( \Lambda = [\delta_i] \) are symmetric \( r \times r \) matrices holding Lagrange multipliers and \( D \) is given by Eq. (3.2) with \( X^T X + YY^T = I_r \).

3.1. Derivation of the Jacobian of \( L^C_{\text{CP}} \)

Define \( \gamma_i = \text{tr}(YM_iX_iD_i) \); we have

\[
\frac{\partial \gamma_i}{\partial X} = \text{vec}(I_r)^T (D_i \otimes I_r) \left( \frac{\partial (X^T X)^{-1}}{\partial X} \right) + (I_r \otimes Y) \frac{\partial (X^T Y Y^T)^{-1}}{\partial X}.
\]

In Case I the partial derivative of \( d_i \) with respect to \( X \) is

\[
\frac{\partial d_i}{\partial X} = (\text{diag}(X^T Y) \otimes I_r) \frac{\partial (X^T X)^{-1}}{\partial X} + (X^T X)^{-1} T_r (I_r \otimes Y M_i) ;
\]

see Appendix B for the derivation of \( \frac{\partial Y}{\partial X} = (X^T X)^{-1} \). In Case II we have that

\[
\frac{\partial d_i}{\partial X} = T_r (I_r \otimes Y M_i)^T.
\]

Analogously,

\[
\frac{\partial \gamma_i}{\partial Y} = \text{vec}(I_r)^T (I_r \otimes X M_i) \left( \frac{\partial (D_i \otimes I_r)}{\partial X} + (I_r \otimes Y) T_r \frac{\partial d_i}{\partial Y} \right).
\]
In Case I the partial derivative of \( d \) with respect to \( Y \) is

\[
\frac{\partial d}{\partial Y} = (\text{diag}_Y(X'MY) \otimes I_m) \frac{\partial (XX^*YY)^{-1}}{\partial Y} = (XX^*Y'Y)^{-1}'T_r(I \otimes XM); \tag{3.11}
\]

see Appendix B for the derivation of \( \frac{\partial (XX^*YY)^{-1}}{\partial Y} \). In Case II we have that

\[
\frac{\partial d}{\partial Y} = T_r(I \otimes XM). \tag{3.12}
\]

The Jacobian of \( L_c^p \) is the \((p+q)\times r\) row vector

\[
\text{Jac}(L_c^p) = \left[ \sum_{i=1}^m \frac{\partial Y_i}{\partial X_i} \sum_{j=1}^m \frac{\partial Y_j}{\partial X_j} \right] - 2(\text{vec}(X)/(\Lambda \otimes I_p)) \text{vec}(Y)/(\Lambda \otimes I_p). \tag{3.13}
\]

### 3.2. Derivation of the Lagrange multipliers

To find expressions for the Lagrange multipliers as functions of \( X \) and \( Y \) we need to solve \( \frac{\partial F_i^p}{\partial X} = 0 \) and \( \frac{\partial F_i^p}{\partial Y} = 0 \). We shall solve the first equation; the process is the same for the second one. Equation \( \frac{\partial F_i^p}{\partial X} = 0 \) is equivalent to \( \sum_{i=1}^m \frac{\partial Y_i}{\partial X_i} = 2(\text{vec}(X)/(\Lambda \otimes I_p)). \) This implies that

\[
\sum_{i=1}^m \frac{\partial Y_i}{\partial X_i} = 2 \sum_{j=1}^r \lambda_{jk} X_j, \tag{3.14}
\]

for \( k = 1, \ldots, r \). In case I Eq. (3.14) becomes \( \sum_{i=1}^m \frac{\partial Y_i}{\partial X_i} = 2\lambda_{jk} X_j \), which implies that

\[
\lambda_{jk} = 2 \frac{1}{\sum_{i=1}^m} \frac{\partial Y_i}{\partial X_i} X_k. \tag{3.15}
\]

In case II we have

\[
\lambda_{jk} = 2 \frac{1}{\sum_{i=1}^m} \frac{\partial Y_i}{\partial X_i} X_j, \tag{3.16}
\]

for \( j = 1, \ldots, r \).

### 3.3. Derivation of the Hessian of \( L_c^p \)

Next we derive the second-derivative matrices. Define the following constant matrices with respect to \( X \): \( J_1 = \text{vec}(I_m(I \otimes Y'M)) \); \( J_2 = -\text{diag}_m(\text{vec}(YY))^r(E(I_n + C_r)); J_3 = T_r(I \otimes XM) \). It can be seen that

\[
\frac{\partial^2 Y_i}{\partial X^2} = \left[ I_p \otimes J_1 \right] \left[ I_p \otimes (X'MY) \otimes I_m \right] \left[ I_p \text{vec}(I_m) \right] T_r \frac{\partial d}{\partial X} \tag{3.17}
\]

with \( \frac{\partial d}{\partial X} \) given by Eq. (3.8) (in Case I) or Eq. (3.9) (in Case II). The term \( \frac{\partial^2 d}{\partial X^2} \) is \( 0_{pr \times pr} \) in Case II; to derive \( \frac{\partial^2 d}{\partial X^2} \) in Case I we start by rewriting Eq. (3.8):

\[
\frac{\partial d}{\partial X} = F_i^p J_2 (I \otimes X) + (XX^*YY)^{-1}J_3, \tag{3.18}
\]

where \( F_i^p = (\text{diag}_m(X'MY) \otimes I_m)((XX^*YY)^{-1} \otimes (XX^*YY)^{-1}) \), see Appendix B. We can now write

\[
\frac{\partial^2 d}{\partial X^2} = \left[ I_p \otimes X'J_2 \otimes I_m \right] \left[ \frac{\partial F_i^p}{\partial X} \right] \left[ I_p \otimes J_2 \right] \left[ I_p \text{vec}(I_m) \right] T_r \left[ \frac{\partial d}{\partial X} \right] + \left[ I_p \otimes C_p \otimes I_m \right] \left[ \text{vec}(I_m) \otimes I_p \right] \left( J_4 \otimes I_m \right), \tag{3.19}
\]

We proceed in a similar way to derive the second-order derivatives with respect to \( Y \). Define the following constant matrices with respect to \( Y \): \( K_1 = \text{vec}(I_m(I \otimes X'M)); K_2 = -\text{diag}_m(\text{vec}(XY))E(I_n + C_r) ; K_3 = T_r(I \otimes XM) \). It can be seen that

\[
\frac{\partial^2 Y_j}{\partial Y^2} = \left[ I_p \otimes K_1 \right] \left[ I_p \otimes C_p \otimes I_m \right] \left[ I_p \text{vec}(I_m) \otimes I_m \right] T_r \frac{\partial d}{\partial Y} \tag{3.20}
\]

with \( \frac{\partial d}{\partial Y} \) given by Eq. (3.11) (in Case I) or Eq. (3.12) (in Case II). The term \( \frac{\partial^2 d}{\partial Y^2} \) is \( 0_{pr \times pr} \) in Case II; to derive \( \frac{\partial^2 d}{\partial Y^2} \) in Case I we start by rewriting (3.11):

\[
\frac{\partial d}{\partial Y} = F_i^p K_2 (I \otimes Y^*') + (XX^*YY)^{-1}K_4, \tag{3.21}
\]

We can now write:

\[
\frac{\partial^2 d}{\partial Y^2} = \left[ I_p \otimes Y \otimes K_1 \otimes I_m \right] \frac{\partial F_i^p}{\partial Y} \frac{\partial d}{\partial Y} \tag{3.22}
\]

with \( \frac{\partial d}{\partial Y} \) given by Eq. (3.8) (in Case I) or Eq. (3.9) (in Case II). The term \( \frac{\partial^2 d}{\partial Y^2} \) is \( 0_{pr \times pr} \) in Case II; to derive \( \frac{\partial^2 d}{\partial Y^2} \) in Case I we start by rewriting Eq. (3.8):

\[
\frac{\partial d}{\partial Y} = F_i^p J_2 (I \otimes X) + (XX^*YY)^{-1}J_3, \tag{3.23}
\]

In order to derive the crossed derivative define the following constants with respect to \( Y \): \( L_1 = \text{vec}(I_m) \); \( L_2 = (I \otimes X^*T_r); L_3 = -E(I_n + C_r)(I \otimes X) \). We can rewrite \( \frac{\partial Y_j}{\partial X} \):

\[
\frac{\partial Y_j}{\partial X} = \left[ I_p \otimes Y \otimes L_1 \otimes I_m \right] \frac{\partial (I \otimes XM')}{\partial Y} \frac{\partial d}{\partial Y} \tag{3.24}
\]

Differentiating \( \frac{\partial Y_j}{\partial X} \) with respect to \( Y \) gives us

\[
\frac{\partial}{\partial Y} \frac{\partial Y_j}{\partial X} = \left[ I_p \otimes Y \otimes L_2 \otimes I_m \right] \frac{\partial (I \otimes XM')}{\partial Y} \frac{\partial d}{\partial Y} + \left[ I_p \otimes Y \otimes L_1 \otimes I_m \right] \frac{\partial (I \otimes XM')}{\partial Y} \frac{\partial d}{\partial Y}, \tag{3.25}
\]

see Appendix B for the derivations of \( \frac{\partial (I \otimes XM')}{\partial Y} \) and \( \frac{\partial (D \otimes I')}{{\partial Y}} \). We have \( \frac{\partial \mathcal{F}_i^p}{\partial Y} \) left to derive. Start by rewriting \( \frac{\partial d}{\partial Y} \):

\[
\frac{\partial d}{\partial Y} = F_i^p L_1 + (XX^*YY)^{-1}T_r(I \otimes XM'), \tag{3.26}
\]

for Case I
\[
\frac{\partial d}{\partial X} = T_{\ell} (I \otimes Y'M') \text{ for Case II},
\]
(3.26)
where \(E_{\ell} = (\text{diag}_\omega(XM)Y') \otimes (XX + YY)^{-1} \otimes (XX + YY)^{-1} \text{diag}_\omega(\text{vec}(YY)). \)
We have that
\[
\frac{\partial \hat{d}}{\partial Y} = (L_{\ell} \otimes I) \frac{\partial F_{\ell}}{\partial Y} + \left( (I \otimes XM)YT_r \otimes I \right) \frac{\partial (XX + YY)^{-1}}{\partial Y} \left( I_r \otimes YT_r \right) \frac{\partial (I \otimes Y'M')}{\partial Y},
\]
for Case I, and
\[
\frac{\partial \hat{d}}{\partial Y} = \left( I_r \otimes YT_r \right) \frac{\partial (I \otimes Y'M')}{\partial Y},
\]
for Case II.

3.4. Sufficient second-order conditions
A sufficient condition for a stationary point of \(L^C\) to be a maximum depends on the type of constraint:
- in Case I it is sufficient for a maximum for which \(W^C_{\ell} = (p+q)r \times (p+q)^r\) is negative definite, where \(W\) is the \((p+q)r \times (p+q)^r\) matrix whose columns span the subspace orthogonal to \(X\):
- in Case II it is sufficient for a maximum for which \(W^CP_{\ell} = (p+q)r \times (p+q-r-1)r\) is negative definite, where \(W\) is the \((p+q)r \times (p+q-r-1)r\) matrix whose columns span the subspace orthogonal to \(X\).

\[
\begin{bmatrix}
1 \otimes X & 0 & H_1 & 0 \\
0 & 1 \otimes X & 0 & H_2
\end{bmatrix},
\]
where
\[
H_1 = \begin{bmatrix}
x_1 & \ldots & x_r \\
x_1 & x_2 & \ldots & x_r \\
\vdots & \vdots & \ddots & \vdots \\
x_1 & x_2 & \ldots & x_r 
\end{bmatrix},
\]
and \(H_2\) is similar to \(H_1\) with all occurrences of \(x_i\)'s replaced by \(y_i\)'s, Magnus and Neudecker ([10], Chapter 7).

4. Optimization of INDSCAL
In a similar fashion as was done for CP, we can reformulate the problem of minimizing the loss function (1.2) of INDSCAL as equivalent to the problem of maximizing
\[
L^{\text{IND}}(X) = \sum_{i=1}^{m} \text{tr}(XS \tilde{D}_i),
\]
(4.1)
where \(\tilde{D}_i\) is the diagonal matrix holding
\[
\partial d_i = \text{transposed row i of } \tilde{D} = (XX + XX)^{-1} \text{diag}_\omega(XS,X) \text{ in the diagonal.}
\]
The Lagrangean is defined by
\[
L^{\text{IND}}(X) = \sum_{i=1}^{m} \partial d_i = (XX + XX)^{-1} \text{diag}_\omega(XS,X).
\]
(4.2)

4.1. Derivation of the Jacobian of \(L^{\text{IND}}\)
Define \(\partial_\ell = \text{tr}(XS \tilde{D}_i)\). We have
\[
\frac{\partial \partial d_i}{\partial X} = \text{vec}(\partial_\ell)\left( \tilde{D}_i X \otimes I_r \right) (S_i \otimes I_r) C_{pr} + (I \otimes XS) \times \left( \tilde{D}_i I_r \right) + (I \otimes X) \left( I_r \otimes \partial d_i \right) \right).
\]
(4.3)
In Case I the partial derivative of \(\partial d_i\) with respect to \(X\) is
\[
\frac{\partial \partial d_i}{\partial X} = \left( \text{diag}_\omega(XS,X) \right) \frac{\partial (XX + XX)^{-1}}{\partial X} \left( XX + XX \right)^{-1} \text{diag}_\omega(XS,X),
\]
(4.5)
see Appendix C for the derivation of \(\frac{\partial (XX + XX)^{-1}}{\partial X}\). In Case II we have that
\[
\frac{\partial \partial d_i}{\partial X} = T_r (XX \otimes I_r) C_{pr} + (I \otimes X) \left( I_r \otimes \partial d_i \right).
\]
(4.6)
The Jacobian of \(L^{\text{IND}}\) is the \(1 \times r\) row vector
\[
\text{Jac}(L^{\text{IND}}) = \sum_{i=1}^{m} \frac{\partial \partial d_i}{\partial X} - 2\text{vec}(X) (A \otimes I_r).
\]
(4.7)

4.2. Derivation of the Lagrange multipliers
Proceeding in a similar fashion as done in Section 3, it is straightforward to verify that
\[
\lambda_{i\ell} = \frac{1}{2} \sum_{k=1}^{m} \frac{\partial \partial d_i}{\partial X_k} x_k
\]
(4.8)
in Case I, and
\[
\lambda_{i\ell} = \frac{1}{2} \sum_{k=1}^{m} \frac{\partial \partial d_i}{\partial X_k} x_k
\]
(4.9)
in Case II \((j, k = 1, \ldots, r)\).
4.3. Derivation of the Hessian of $L^{2\text{IND}}$

Now define the following matrices which are constant with respect to $X$: $N_i = \text{vec}(L_i)$; $N_2 = (S \otimes L_i)C_{pr}$; $N_i = E_i(L_i + C_r)$. We can rewrite

$$\frac{\partial \gamma}{\partial X} = N_i \left( G_i^T N_2 + G_i^T \frac{\partial \tilde{d}}{\partial X} \right).$$  \hfill (4.10)

It can be seen that

$$\frac{\partial^2 \gamma}{\partial X} = \left( \frac{\partial G_i}{\partial X} \right) \left( \frac{\partial G_i}{\partial X} \right)^T + \left( \frac{\partial \tilde{d}}{\partial X} \right) \left( \frac{\partial \tilde{d}}{\partial X} \right)^T,$$  \hfill (4.11)

with $\frac{\partial \tilde{d}}{\partial X}$ given by Eq. (4.5) (in Case I) or Eq. (4.6) (in Case II). To derive $\frac{\partial^2 \tilde{d}}{\partial X}$ in Case I we start by rewriting Eq. (4.5):

$$\frac{\partial \tilde{d}_i}{\partial X} = 2G_i^T N_i G_5 + G_3 T_i^T G_{i6}.$$  \hfill (4.12)

It can be seen that

$$\frac{\partial^2 \tilde{d}_i}{\partial X} = 2 \left( G_i^T N_i \right) \frac{\partial G_i}{\partial X} + \left( \frac{\partial G_i}{\partial X} \right) \left( \frac{\partial G_i}{\partial X} \right)^T + \left( \frac{\partial \tilde{d}_i}{\partial X} \right) \left( \frac{\partial \tilde{d}_i}{\partial X} \right)^T + \left( \frac{\partial G_i}{\partial X} \right) \left( \frac{\partial G_i}{\partial X} \right)^T \frac{\partial \tilde{d}_i}{\partial X}.$$  \hfill (4.13)

In Case II we have that

$$\frac{\partial \tilde{d}_i}{\partial X} = T_i G_{i6}, \quad \frac{\partial^2 \tilde{d}_i}{\partial X} = \left( \frac{\partial G_i}{\partial X} \right)^T \frac{\partial G_i}{\partial X}.$$  \hfill (4.15)

The Hessian of $L^{2\text{IND}}$ is the $pr \times pr$ symmetric matrix

$$\text{Hess}(L^{2\text{IND}}) = \sum_{i=1}^{m} \frac{\partial^2 \gamma_i}{\partial X^2} - 2 \left( \Lambda \otimes I_r \right).$$  \hfill (4.16)

4.4. Sufficient second-order conditions

A sufficient condition for a stationary point of $L^{2\text{IND}}$ to be a maximum depends on the type of constraint:

- In Case I it is sufficient that $W = \text{vec}(I)$ is of the form $pr \times (pr - r)$ matrix whose columns span the subspace orthogonal to $I_r \otimes X$.
- In Case II it is sufficient that $\tilde{W} = \text{vec}(I)$ is of the form $pr \times (pr - r + 1)$ matrix whose columns span the subspace orthogonal to $I_r \otimes X$.

5. Illustration: the KHL data

Ten Berge, Kiers and De Leeuw [17] analysed a contrived array which they christened “KHL data”, due to previous work by Kruskal, Harshman and Lundy [8,9]. The KHL data is the $2 \times 2 \times 2$ array

$$X = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$  \hfill (5.1)

We ran the ALS algorithm 200 times for $X$ with $r = 2$ components. The component matrices were randomly initialized by orthonormal matrices. In all runs the algorithm halted on solutions with loss $f \approx 2$. We wanted to test the nature of these solutions, i.e., whether these solutions correspond to minima and/or saddle points.

We computed the Jacobian and Hessian for each of the 200 solutions under unit length constraint. In general, each of the 200 solutions displays a similar behaviour: $A$ has rank 1, $B$ and $C$ are orthonormal, Jac is approximately $\Theta_2 \times \Theta_3$ (its entries are usually in the order of $10^{-14}$), and the Hess is of the form

$$\text{Hess}_{\text{CP}} = \begin{bmatrix} 0 & a & b \\ 0 & -a & -b \\ a & -b & 0 \end{bmatrix}.$$  \hfill (5.2)

for real numbers $a, b$. The eigenvalues of Hess$_{\text{CP}}$ are typically $\{0, 0, -\lambda, \lambda\}$, for real $\lambda$. Therefore, it can be concluded that each of the 200 solutions are, indeed, saddle points.

This example shows two things. On one hand, there exist cases for which the occurrence of saddle points is a severe problem, like the KHL data. On the other hand, it is relevant to have a tool available that diagnoses whether a solution is a saddle point. Once spotted, such solutions should be discarded at once.

Ten Berge, Kiers and De Leeuw [17] showed that the CP loss function (1.1) has infimum 1 when 2 components are extracted. This reinforces the fact that none of the 200 solutions that were found could correspond to the global minimum. However, in the absence of this information, the researcher would profit from knowing that all solutions were saddle points and therefore useless. This is possible by analysing the second-order differential structure as we have done here.

The KHL data is a contrived example. The question of whether similar behaviour is to be expected for real data is still unanswered. The applications discussed in Sections 6–8 are intended to better understand what happens in general.

6. Application I: INDSCAL under orthonormality constraint

Ten Berge et al. [18] discussed an algorithm for INDSCAL with orthogonality constraints referred to as the SVD-process. This algorithm was originally devised as a Varimax procedure based on an SVD, but Ten Berge [15] observed that the problem could be reformulated in terms of diagonalizing a set of symmetric matrices simultaneously. The SVD-process provides a direct procedure to fit the INDSCAL model under orthogonality constraints.

The SVD-process attempts to find a columnwise orthonormal $X$ such that $L^{2\text{IND}}(X)$ is maximized; it proceeds as follows:

Step 1 Initialize $X$ ($pr \times r$ orthonormal).
Step 2 Compute $D_i = \text{diag}(\{X_i X_i\}) = 1, \ldots, m$.
Step 3 Compute the SVD $\sum_{i=1}^{m} \sum_{X} = P L Q$, and update $X$ by $X = P Q$.
Step 4 Repeat Steps 2 and 3 until the relative increase in $L^{2\text{IND}}(X)$ is smaller than a predefined convergence criterion.

The SVD-process to INDSCAL has been proved to converge monotonically when the frontal slices of array $S$ are positive or negative semidefinite, Ten Berge et al. [18]. Thus, we will work with arrays holding semidefinite frontal slices in the remaining of this section.

Ten Berge et al. [18] ran some experiments where they argue that the SVD-process to INDSCAL seems to be hampered by the occurrence of local maxima of $L^{2\text{IND}}$. However, the possibility of the occurrence of saddle points was not considered. Notice that there exist contrived examples for which saddle points do occur. For example, consider the $2 \times 3 \times 3$ array with positive semidefinite slices (Ten Berge and Kiers [16])

$$S_1 = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 3 & -1 & 0 \\ -1 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$  \hfill (6.1)
The (orthonormally constrained) INDSCAL optimal solution with \( r = 2 \) components is

\[
X = \begin{bmatrix}
\sqrt{5} & -\sqrt{5} \\
\sqrt{5} & -\sqrt{5} \\
0 & 0
\end{bmatrix}, \quad D = \begin{bmatrix}
4 & 2 \\
2 & 4
\end{bmatrix}.
\] (6.2)

It corresponds to the global minimum 1 of (1.2). There are, however, non-optimal orthonormally constrained INDSCAL solutions corresponding to saddle points. The following four solutions are stationary points of (1.2) that correspond to non-optimal values of (1.2) (5, 20, 22, 22, respectively). They are all saddle points.

\[
\begin{align*}
X^{(1)} &= \begin{bmatrix}
0 & 1 \\
1 & 0 \\
0 & 0
\end{bmatrix}, \quad D^{(1)} = \begin{bmatrix}
3 & 3 \\
3 & 3
\end{bmatrix} \\
X^{(2)} &= \begin{bmatrix}
\sqrt{5} & 0 \\
\sqrt{5} & 0 \\
0 & 1
\end{bmatrix}, \quad D^{(2)} = \begin{bmatrix}
4 & 0 \\
2 & 1
\end{bmatrix} \\
X^{(3)} &= \begin{bmatrix}
0 & 1 \\
0 & 0 \\
1 & 0
\end{bmatrix}, \quad D^{(3)} = \begin{bmatrix}
0 & 3 \\
1 & 3
\end{bmatrix} \\
X^{(4)} &= \begin{bmatrix}
0 & 0 \\
1 & 0 \\
0 & 1
\end{bmatrix}, \quad D^{(4)} = \begin{bmatrix}
3 & 0 \\
3 & 1
\end{bmatrix}.
\end{align*}
\] (6.3-6.6)

The fact that such non-optimal solutions exist does not imply that the SVD-approach algorithm will converge to them. This is precisely the point that we wanted to investigate in this application: is it possible that the SVD-approach algorithm converges to saddle points? The answer to this question can clarify the type of solutions that the SVD-approach usually finds, therefore the interpretation of the solution is further enriched.

A simulation study was carried out to test whether saddle points occur (software: Matlab R2008a). We randomly generated 150 3×3×3 symmetric slice arrays with positive definite slices. Each slice was generated as \( MMM \), where \( M \) is a 3×3 matrix whose entries were uniformly generated from the interval \([-1, 1]\). For each array we ran the SVD-approach to INDSCAL with \( r = 2 \) components using 10 different random initializations for \( X \); each \( X \) was a 3×3 matrix whose entries were uniformly generated from the interval \([-1, 1]\); afterwards, \( X \) was orthonormalized via the Gram–Schmidt procedure.

The convergence criterion was fixed at 1e – 06. After convergence, the Jacobian and Hessian for each INDSCAL solution \((X, D)\) were computed, and we inspected whether \( W^{D^{(r)}D^{(r)}} W \) was negative definite or indefinite (second-order sufficient condition).

The same procedure was repeated, this time for arrays with positive semidefinite slices. Each slice was generated as \( MMM \), where \( M \) is a 2×3 matrix whose entries were uniformly generated from the interval \([-1, 1]\).

All results were numerically stable, as expected. We verified that the SVD-approach for INDSCAL never halted on saddle points. Also, it was verified that local maxima occurred for 12 arrays (8% of the cases) for arrays with positive definite slices, whereas for arrays with positive semidefinite slices local maxima occurred for 17 arrays (11% of the cases). Although these results do not formally prove that convergence to saddle points is impossible, it can be concluded that there are no indications to that effect.

7. Application II: INDSCAL equivalence problem in CP formulation

Carroll and Chang [3] suggested running CP in order to fit INDSCAL because they conjectured that \( X \) and \( Y \) would end up equal or at least columnwise proportional if CP converged. If Carroll and Chang’s conjecture is correct, CP can be used as an algorithm to compute INDSCAL solutions for symmetric slice arrays. This conjecture seems to be valid in practical applications. However, counter-examples have already been constructed. Ten Berge and Kiers [16] proved that equivalence may be violated at global minima if \( f \) the slices \( S_i \) are not positive definite. They considered the array \([\text{Ten Berge and Kiers}\ [16]\).}

Equation (7.1) for which a global minimum of (1.1) with \( r = 2 \) components and \( X \) not equivalent to \( Y \) was presented. We ran CP 500 times with \( r = 2 \) and \( r = 3 \) components for the previous array. In both cases all runs converged to a global minimum of (1.1) with \( X \) and \( Y \) non-equivalent. When \( r = 1 \) the algorithm sometimes did converge to a solution with \( X \) and \( Y \) equivalent.

When the slices are nonnegative definite and \( r = 1 \) then equivalence can be violated only at stationary points that do not correspond to global minima. In this case, Ten Berge and Kiers [16] conjectured that such stationary points would correspond to local minima. However, Bennani Dosse and Ten Berge [1] proved that such stationary points can only be saddle points.

Bennani Dosse, Ten Berge and Tendeiro [2] showed that equivalence occurs when the components are constrained by orthonormality, the slices are positive semidefinite and the saliences are non-negative. It is still not clear whether non-equivalence occurs or not under circumstances different from these, or whether CP converges to saddle points or not. We conducted some simulations to try to understand what happens in cases where components are not orthonormal, slices are not necessarily positive semidefinite, and saliences are unconstrained. Eleven situations were considered, revolving around arrays with 3×3 symmetric slices: \( 2 \times 3 \times 3 (r = 2) \), \( 3 \times 3 \times 3 (r = 2, 3) \), \( 4 \times 3 \times 3 (r = 2, 3) \), \( 5 \times 3 \times 3 (r = 2, 3, 4) \), \( 6 \times 3 \times 3 (r = 2, 3, 4) \). Both positive definite and indefinite slice arrays were considered.

One hundred arrays were generated for each case. The positive definite slices were generated as \( MMM \), where \( M \) is a 3×3 matrix whose entries were uniformly generated from the interval \([-1, 1]\). The entries of the diagonal and the upper-triangular parts of the indefinite slices were nonnegative definite and the saliences are non-negative. It is not clear whether non-equivalence occurs or not under circumstances different from these, or whether CP converges to saddle points or not. We conducted some simulations to try to understand what happens in cases where components are non-orthonormal, slices are not necessarily positive semidefinite, and saliences are unconstrained. Eleven situations were considered, revolving around arrays with 3×3 symmetric slices: \( 2 \times 3 \times 3 (r = 2) \), \( 3 \times 3 \times 3 (r = 2, 3) \), \( 4 \times 3 \times 3 (r = 2, 3) \), \( 5 \times 3 \times 3 (r = 2, 3, 4) \), \( 6 \times 3 \times 3 (r = 2, 3, 4) \). Both positive definite and indefinite slice arrays were considered.

One hundred arrays were generated for each case. The positive definite slices were generated as \( MMM \), where \( M \) is a 2×3 matrix whose entries were uniformly generated from the interval \([-1, 1]\). The entries of the diagonal and the upper-triangular parts of the indefinite slices were uniformly generated from the interval \([-1, 1]\); the lower-triangular part of each slice was filled in such that symmetry would occur. Each array was given 100 different random startups; the convergence criterion was set at 1e – 08. No constraint was imposed on the saliences in \( D \). A solution was declared degenerate when at least one of the non-diagonal entries of the so-called triple cosine matrix was below – 0.05. Our main goal was to check whether non-equivalence occurred or not, and to what extent of stationary point it corresponded (local optimum or saddle point).

The Jacobian matrices associated to non-degenerate solutions were analysed. Its entries were relatively small (usually with modulus smaller than 1e – 05), thus analysing the second-order differential structure seems legitimate. We worked under unit length constraints. Occurrences of degeneracy and of different values for the loss function were registered.

The results found are summarized in Tables 1 and 2. The variables read: NonEquiv = number of arrays for which at least one startup ended up with non-equivalent CP solution; Deg = number of arrays with degenerate solutions, within the 100 startups \((x + y: x = all 100 startups are degenerate; y = < 100 startups are degenerate); \)SadPt = number of arrays for which at least one startup ended in a saddle point, non-degenerate \((x/ y: x = for CP's Hessian; y = for INDSCAL's Hessian); \)DiffPt = number of arrays with at least two different values for CP's loss function within the 100 startups, with at least one non-degenerate solution. Some special situations are marked with asterisks, as follows: (*) = the associated CP...
solution is degenerate (all components are almost proportional); (***)= one or more of the eigenvalues of the Hessian are relatively small in magnitude (smaller than 1e−01), indicating that the Hessian is nearly singular; (***)= for one startup of one array the Hessian for CP was nearly singular, but the Hessian for INDSCAL was negative definite.

The first observation to be made is that non-equivalence was never observed for non-degenerate solutions. Since no non-equivalent solution was found for arrays with positive definite slices, it was not possible to test whether the result of Bennani Dosse and Ten Berge [1] for arrays with positive definite slices does apply to cases with r=1 components. Also, saddle points were rarely observed. In addition, we can observe that arrays with indefinite slices are more prone to suffer from degeneracy, occurrence of saddle points, and multiple fit values.

The cases reported with (**) are situations where it is not clear whether we are facing a saddle point or not, since the Hessian matrix seems to be almost singular. These cases should be treated with care, since the second-order sufficient condition applies to non-singular Hessian matrices. It is not clear why such points occur. An anonymous reviewer suggested that the problem might be originated in rank-deficient component matrices. We verified that this was true for five of the situations reported by (**). It should be noted that these component matrices were estimated rather than randomly generated, and that these decompositions are not degenerate.

8. Application III: CP in general

A simulation study was conducted to inspect the occurrence of saddle points for CP solutions of generic arrays. Twenty nine situations were considered, for which uniqueness is proved to hold due to Kruskal's sufficient condition for uniqueness (Kruskal [7]). One hundred arrays were randomly generated for each situation, the entries being uniformly generated from the interval [−1, 1]. Each array was given 100 different random startups; the convergence criterion was set at 100−3. For arrays with positive definite slices, it was not possible to test whether the result of Harshman and Lundy [6] for arrays with positive definite slices does apply to cases with r=1 components. Also, saddle points were rarely observed. In addition, we can observe that arrays with indefinite slices are more prone to suffer from degeneracy, occurrence of saddle points, and multiple fit values.

The cases reported with (**) are situations where it is not clear whether we are facing a saddle point or not, since the Hessian matrix seems to be almost singular. These cases should be treated with care, since the second-order sufficient condition applies to non-singular Hessian matrices. It is not clear why such points occur. An anonymous reviewer suggested that the problem might be originated in rank-deficient component matrices. We verified that this was true for five of the situations reported by (**). It should be noted that these component matrices were estimated rather than randomly generated, and that these decompositions are not degenerate.

9. Discussion

In this paper we dealt with first and second-order differential structures of optimization functions related to CP and INDSCAL. Our goal was to provide a tool to further characterize three-way solutions. Closed form formulas for the Jacobian and Hessian matrices were derived, under two different types of constraints.

Simulations that highlight the usefulness of Hessian structure were performed. The results of the simulations seem to tell that saddle points do not occur frequently, although they do occur with positive probability. In some cases the Hessian matrix showed to be ill-conditioned. The reasons for this phenomenon are still not clear and need further investigation.

Some numerical problems occur when we consider degenerate CP/INDSCAL solutions (Harshman and Lundy [6]). Typically, a degenerate solution is one where some components become more and more proportional, while some entries of these components become larger and larger, as the algorithm progresses. In a degenerate solution, the contributions of some of the degenerate components nearly cancel the contributions of other degenerate components, while the components together contribute to improve the fit.

The computation of the Jacobian and the Hessian matrices are free of numerical problems for CP/INDSCAL solutions which are not degenerate. However, degenerate solutions do lead to problems. These problems are more or less severe depending on how many degenerate components exist and how strong the degeneracy is. The core of this problem resides in matrix Γ = X′ X X′ (for CP) and Γ = X′ X′ X′ (for INDSCAL), recall Eqs. (3.2) and (4.2). When a CP/INDSCAL solution is degenerate, Γ becomes almost rank deficient, which creates numerical problems when computing Γ−1. Equivalently, the problem is that the optimization function is (almost) non-differentiable at the

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Arrays with positive definite slices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim. array</td>
<td># comps</td>
</tr>
<tr>
<td>2×3×3</td>
<td>r=2</td>
</tr>
<tr>
<td>3×3×3</td>
<td>r=2</td>
</tr>
<tr>
<td>3×3×3</td>
<td>r=3</td>
</tr>
<tr>
<td>4×3×3</td>
<td>r=2</td>
</tr>
<tr>
<td>5×3×3</td>
<td>r=2</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=1</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=2</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=3</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=4</td>
</tr>
</tbody>
</table>

It can be seen that saddle points occur scarcely; almost all these occurrences relate to a nearly singular Hessian matrix. It is not clear why such solutions occur. In addition, we point out that retaining more components seems to have the effect of increasing the number of degenerate solutions.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Arrays with generic slices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim. array</td>
<td># comps</td>
</tr>
<tr>
<td>2×2×2</td>
<td>r=1</td>
</tr>
<tr>
<td>3×2×2</td>
<td>r=1</td>
</tr>
<tr>
<td>3×3×2</td>
<td>r=2</td>
</tr>
<tr>
<td>3×3×3</td>
<td>r=1</td>
</tr>
<tr>
<td>4×2×2</td>
<td>r=2</td>
</tr>
<tr>
<td>4×3×2</td>
<td>r=1</td>
</tr>
<tr>
<td>4×3×3</td>
<td>r=1</td>
</tr>
<tr>
<td>5×2×2</td>
<td>r=1</td>
</tr>
<tr>
<td>5×3×2</td>
<td>r=2</td>
</tr>
<tr>
<td>5×3×2</td>
<td>r=1</td>
</tr>
<tr>
<td>5×3×3</td>
<td>r=1</td>
</tr>
<tr>
<td>6×3×2</td>
<td>r=1</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=2</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=3</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=3</td>
</tr>
<tr>
<td>5×2×2</td>
<td>r=1</td>
</tr>
<tr>
<td>5×2×2</td>
<td>r=3</td>
</tr>
<tr>
<td>5×2×2</td>
<td>r=3</td>
</tr>
<tr>
<td>6×3×3</td>
<td>r=4</td>
</tr>
<tr>
<td>6×3×2</td>
<td>r=2</td>
</tr>
<tr>
<td>5×3×2</td>
<td>r=1</td>
</tr>
<tr>
<td>5×3×2</td>
<td>r=2</td>
</tr>
<tr>
<td>5×3×2</td>
<td>r=3</td>
</tr>
<tr>
<td>5×4×2</td>
<td>r=2</td>
</tr>
<tr>
<td>5×4×2</td>
<td>r=3</td>
</tr>
<tr>
<td>5×4×2</td>
<td>r=4</td>
</tr>
</tbody>
</table>
point corresponding to a degenerate solution. The problem might vary from mild to severe, depending on how close or far is $\Gamma$ from singularity. In some severe situations the computations might need to be completely disregarded. For instance, we observed solutions for which the severity of the deficiency of $\Gamma$ leads to loss of symmetry of the Hessian, which is naturally a serious problem.

A CP solution is never degenerate under an orthonormality constraint on $X$ and $Y$, Harshman and Lundy [6]. Likewise, an INDCAL solution is not degenerate if $X$ is constrained by orthonormality. Therefore, orthonormality constraints typically avoid any numerical problems. In any other case, we advise to first check whether the solution at hand is degenerate or not. If the solution is not degenerate then the use of the formulas to compute the Jacobian and the Hessian is warranted. In case of degeneracy, one should do some prior analysis on the rank deficiency of $\Gamma$. If the problem is not very severe, it is possible that $\Gamma^{-1}$ is relatively well defined, and therefore all the computations will follow safely. A posterior test to the numerical stability of the process is, for example, to compute the Hessian matrix Hess and afterwards compute $\rho = \text{tr}(\text{Hess} - \text{Hess}^2)$; large values of $\rho$ (say, $\rho > 10^{-20}$) indicate that Hess is further from symmetry than it should. Therefore, Hess should be discarded in such cases.

Appendix A. Matrix differentiation formulas

Consider the following matrices: $A$ ($m \times n$); $B$ ($p \times q$); $d$ ($n \times 1$); $D = \text{diag}(d)$.

Table A1 presents the most important formulas of matrix differentiation that are of use in this paper.

| $\frac{\partial \text{tr}(A)}{\partial A}$ | $\text{vec}(A)$ |
| $\frac{\partial \text{vec}(A)}{\partial A}$ | $\text{vec}(A)$ |

The formulas in the first column can be found in Fackler [4]. The formulas in the second column can be obtained as follows:

$$\frac{\partial \text{tr}(A)}{\partial A} = \text{vec}(A) \text{vec}(A)^T = \text{vec}(A) \frac{\partial A}{\partial A}$$

$$\frac{\partial \text{vec}(A)}{\partial A} = \text{vec}(A)$$

$$\frac{\partial \text{vec}(A)}{\partial A} = \text{vec}(A)$$

(continued on next page)
Table C1 (continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_1^1 = -((G_2^2) \otimes I)$</td>
<td>$\frac{\partial G_1^1}{\partial G_2} = -(I \otimes \text{vec}(I)) Y + (X \otimes \text{vec}(I)) (I \otimes S_1)$</td>
</tr>
<tr>
<td>$G_11 = G_0 \otimes G_3$</td>
<td>$\frac{\partial G_11}{\partial G_0} = (I \otimes C_0 \otimes I) (I \otimes \text{vec}(I)) (I \otimes S_1)$</td>
</tr>
<tr>
<td>$G_1^1 = G_0 \otimes G_11$</td>
<td>$\frac{\partial G_1^1}{\partial G_0} = (I \otimes C_0 \otimes I) (I \otimes \text{vec}(I)) (I \otimes S_1)$</td>
</tr>
<tr>
<td>$G_1^1 = \text{diag}(\text{vec}(XX))$</td>
<td>$\frac{\partial G_1^1}{\partial X} = I + C_0 (I \otimes X)$</td>
</tr>
<tr>
<td>$G_13 = \text{diag}(\text{vec}(XX))$</td>
<td>$\frac{\partial G_13}{\partial X} = I + C_0 (I \otimes X)$</td>
</tr>
<tr>
<td>$G_1^1 = G_0 \otimes G_13$</td>
<td>$\frac{\partial G_1^1}{\partial G_0} = (I \otimes C_0 \otimes I) (I \otimes \text{vec}(I)) (I \otimes S_1)$</td>
</tr>
<tr>
<td>$G_1^1 = G_0 \otimes G_{12}$</td>
<td>$\frac{\partial G_1^1}{\partial G_0} = (I \otimes C_0 \otimes I) (I \otimes \text{vec}(I)) (I \otimes S_1)$</td>
</tr>
<tr>
<td>$G_1^1 = G_0 \otimes G_{11}$</td>
<td>$\frac{\partial G_1^1}{\partial G_0} = (I \otimes C_0 \otimes I) (I \otimes \text{vec}(I)) (I \otimes S_1)$</td>
</tr>
</tbody>
</table>

Table C1 summarizes the expressions of functions $G_1 - G_{16}$ with the relevant partial derivatives.

References