Received: 4 September 2018 | Revised: 29 October 2018 | Accepted: 15 November 2018

DOI: 10.1002/nla.2226

RESEARCH ARTICLE

WILEY

Solving bilinear tensor least squares problems and application to Hammerstein identification

Lars Eldén[1] | Salman Ahmadi-Asl[2]

[1]Department of Mathematics, Linköping University, Linköping, Sweden

[2]Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia

Correspondence
Lars Eldén, Department of Mathematics, Linköping University, 581 83 Linköping, Sweden.
Email: lars.elden@liu.se

Funding information
Mega Grant, Grant/Award Number: 14.756.31.0001

AMS Subject Classification: 65F99, 65K10, 15A69, 93E24

Summary

Bilinear tensor least squares problems occur in applications such as Hammerstein system identification and social network analysis. A linearly constrained problem of medium size is considered, and nonlinear least squares solvers of Gauss–Newton-type are applied to numerically solve it. The problem is separable, and the variable projection method can be used. Perturbation theory is presented and used to motivate the choice of constraint. Numerical experiments with Hammerstein models and random tensors are performed, comparing the different methods and showing that a variable projection method performs best.

KEYWORDS

bilinear regression, bilinear tensor least squares problem, Hammerstein identification, Gauss–Newton-type method, separable, variable projection

1 | INTRODUCTION

Multidimensional arrays or tensors are generalizations of vectors and matrices. In this paper, we study the following linearly constrained bilinear tensor least squares (LBLS) problem:

$$\min_{x,y} \|\mathcal{A} \cdot (x, y)_{2,3} - b\|, \qquad e_i^T x = 1, \tag{1}$$

where $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$ is a tensor, and $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, $b \in \mathbb{R}^l$, and e_i is a canonical unit vector, for some i with $1 \le i \le m$. Here, $\|\cdot\|$ is the Euclidean vector norm, and $\mathcal{A} \cdot (x, y)_{2,3}$ denotes tensor–vector–vector multiplication along modes 2 and 3 (to be defined later). Throughout this paper, we assume that the LBLS problem (1) is overdetermined, that is, $l > m + n - 1$.

As we will demonstrate in Section 3, the bilinear problem is singular without the constraint. This is well known in the literature on the Hammerstein model,[1,2] and it is common to impose either a linear or a quadratic constraint on one of the unknown vectors and use an alternating algorithm. Typically, alternating algorithms converge slowly.[3] We show that by imposing the linear constraint as in (1), it is possible to apply standard, nonalternating methods for nonlinear least squares problems, with faster convergence to a stationary point. Thus, we describe the use of Gauss–Newton-type (G-N) methods (see, e.g., chapter 9 of the work of Björck[4]). Because the problem is separable, we can also use variable projection (VP) methods,[3,5] where one of x and y is eliminated and an optimization problem in terms of the other is solved.

The main contribution of this paper is the development of perturbation theory for the LBLS problem and a study of its consequences for algorithm design. We show that the convergence rate depends on the conditioning of the problem, which in turn depends on how the linear constraint is implemented, that is, which component of x or y is constrained to be equal to 1. Thus, in the G-N methods, we choose the constraint that makes the Jacobian as well conditioned as possible. We also present perturbation theory, which shows that the conditioning of the solution parts x and y may be different.

In the VP methods, we constrain the part of the solution that is most well conditioned. A comparison of algorithms is made, where a VP method comes out as winner.

The LBLS problem (1) with linear (or sometimes quadratic) constraint arises in the identification of a Hammerstein model; see, for example, the works of Bai et al.[1] and Wang et al.,[2] where the problem dimensions are usually quite small. There is a rather extensive literature on Hammerstein systems (see the references in the works of Wang et al.[2] and Ding et al.[6]), and the methods of this paper have been used before. For a very brief introduction to Hammerstein system identification, see Section 5.1.

The theory for bilinear systems of equations of the type $\mathcal{A} \cdot (x, y)_{2,3} = b$ is studied in the work of Johnson et al.,[7] and some applications are mentioned. Related bilinear and multilinear problems are used in statistics and chemometrics.[8,9] Much larger problems occur in text analysis of news articles and social media[10,11]; here, the tensor is also sparse.

The problem (1) can also be considered as a *bilinear tensor regression problem*. In the last couple of years, such problems are becoming quite abundant in the literature; see, for example, two recent papers dealing with more general setups.[9,12] However, most such papers are application oriented and do not deal with basic properties and algorithms (in fact, most propose alternating algorithms). In this paper, we are concerned with the simpler problem (1), we introduce perturbation analysis, and we discuss and implement algorithms for problems of small and medium dimensions. This will be the starting point for research concerning the more general problems.

For a comprehensive review of tensor decomposition and applications, see the work of Kolda et al.[13]

This paper is organized as follows. First, in Section 2, some necessary tensor concepts are defined. In Section 3, the nature of the singularity of the unconstrained bilinear least squares problem is discussed. Then, we show how to solve the LBLS problem using Gauss–Newton and VP methods. We present perturbation theory for the LBLS problem in Section 4. There, we also discuss how to implement the linear constraint numerically. Numerical experiments are reported in Section 5.

1.1 | Notation

We let $\|r\|$ be the Euclidean vector norm, and the subordinate matrix norm of A is denoted $\|A\|_2$. The Frobenius norm of a tensor is written $\|\mathcal{A}\|$. The identity matrix of dimension n is denoted I_n, and its column vectors, the canonical unit vectors, are denoted e_i, for $i = 1, 2, \dots, n$. We will use the notational convention

$$\mathbb{R}^m \ni x = \begin{pmatrix} 1 \\ \bar{x} \end{pmatrix}, \qquad \bar{x} \in \mathbb{R}^{m-1}, \tag{2}$$

similar for x with subscripts.

Let $A \in \mathbb{R}^{m\times n}$ with $m > n$. The *thin QR decomposition* (see theorem 5.2.3 of the work of Golub et al.[14]) is $A = QR$, where $Q \in \mathbb{R}^{m\times n}$ has orthonormal columns, and $R \in \mathbb{R}^{n\times n}$ is upper triangular.

The term *tensor* is used to denote multidimensional arrays, denoted by calligraphic letters, $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$. In this paper, we restrict ourselves to three-dimensional tensors. For matrices, we use Roman capital letters, and for vectors, Roman small letters.

2 | TENSOR CONCEPTS

To denote elements in a tensor, we use two equivalent notations, $\mathcal{A}(i, j, k) = a_{ijk}$. A *fiber* is a subtensor, where all indices except one are fixed, for example, $\mathcal{A}(i, :, k)$. A fiber can be identified with a vector.

Multiplication of a tensor by a matrix $U \in \mathbb{R}^{p\times l}$ is defined

$$\mathbb{R}^{p\times m\times n} \ni \mathcal{B} = (U)_1 \cdot \mathcal{A}, \quad b_{ijk} = \sum_{\mu=1}^{l} u_{i\mu} a_{\mu jk}.$$

This is equivalent to multiplying all mode-1 fibers in $\mathcal{A}$ by the matrix U. Multiplication in the other modes is analogous. Simultaneous multiplication in all three modes is denoted $(U, V, W) \cdot \mathcal{A}$. A special notation is used for multiplication by

a transposed matrix; let $\tilde{U} \in \mathbb{R}^{l\times p}$:

$$\mathbb{R}^{p\times m\times n} \ni \mathcal{B} = (\tilde{U}^T)_1 \cdot \mathcal{A} =: \mathcal{A} \cdot (\tilde{U}), \quad b_{ijk} = \sum_{\mu=1}^{l} a_{\mu jk}\tilde{u}_{\mu i}.$$

Multiplication of a tensor by a vector is analogous; for instance, assuming conforming dimensions,

$$\mathcal{A} \cdot (x, y)_{2,3} \in \mathbb{R}^{l\times 1\times 1}.$$

We will also let $\mathcal{A} \cdot (x, y)_{2,3} \in \mathbb{R}^{l}$. The notation $\mathcal{A} \cdot (x, \cdot)_{2,3}$ will be used for a linear operator,

$$\mathcal{A} \cdot (x, \cdot)_{2,3} \ : \quad \mathbb{R}^n \ni y \mapsto \mathcal{A} \cdot (x, y)_{2,3} \in \mathbb{R}^{l}. \tag{3}$$

We will also use the same notation for the matrix in $\mathbb{R}^{l\times n}$ of the operator. The operator and matrix $\mathcal{A} \cdot (\cdot, y)_{2,3}$ are defined analogously.

Matrices are assumed to have rows and columns, and vectors are usually column vectors. However, the different modes of a tensor are not associated with rows or columns, et cetera, until we explicitly do so. This means that $\mathcal{A} \cdot (u, v)_{2,3} \in \mathbb{R}^{l\times 1\times 1}$ is neither a column vector or a row vector, until we do such an association explicitly.

3 | ALGORITHMS FOR SOLVING THE LBLS PROBLEM

Let x and y be arbitrary and choose $\alpha \neq 0$. Clearly, $\mathcal{A} \cdot (x, y)_{2,3} = \mathcal{A} \cdot (\alpha x, 1/\alpha\, y)_{2,3}$. Thus, the residual

$$r(x, y) = \mathcal{A} \cdot (x, y)_{2,3} - b \tag{4}$$

is constant along the curve $(\alpha x, 1/\alpha\, y)$ in $\mathbb{R}^m \times \mathbb{R}^n$. It follows that the Jacobian is rank deficient in the direction of the curve and the problem (1) without constraint is indeterminate. This indeterminacy is well known; see, for example, the works of Bai et al.[1] and Wang et al.[2] In order to remove it, we impose a linear constraint that prevents the solution from moving along the curve. In our description of methods, we will, for simplicity of presentation and without loss of generality, let the constraint be $e_1^T x = 1$. Thus, we consider the constrained minimization problem (1) with $i = 1$. We will discuss in Section 4.3 how to implement the constraint in a numerically sound way. Alternative ways of handling the indeterminacy are described in the works of Bai et al.,[1] Wang et al.,[2] and Ding et al.[6]

In this section, we describe Gauss–Newton and VP methods for solving the LBLS problem.

3.1 | The gradient and Hessian

With the residual defined in (4), let the objective function be

$$f(x, y) = \frac{1}{2} r(x, y)^T r(x, y),$$

where $r(x, y) = (r_1(x, y), r_2(x, y), \dots, r_l(x, y))^T$. Then, the gradient is[4(p340)]

$$\nabla f(x, y) = J(x, y)^T r(x, y),$$

where $J(x, y)$ is the Jacobian of $r(x, y)$. From the bilinearity, we get the identity

$$\begin{aligned} r(x + s, y + t) - r(x, y) &= \mathcal{A} \cdot (s, y)_{2,3} + \mathcal{A} \cdot (x, t)_{2,3} + \mathcal{A} \cdot (s, t)_{2,3} \\ &= \begin{pmatrix} \mathcal{A} \cdot (\cdot, y)_{2,3} & \mathcal{A} \cdot (x, \cdot)_{2,3} \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} + \mathcal{A} \cdot (s, t)_{2,3}, \end{aligned}$$

and it is seen that

$$J(x, y) = \begin{pmatrix} J_x & J_y \end{pmatrix} \in \mathbb{R}^{l\times(m+n)},$$

where J_x and J_y are matrices identified with the operators $\mathcal{A} \cdot (\cdot, y)_{2,3}$ and $\mathcal{A} \cdot (x, \cdot)_{2,3}$, respectively.

The rank deficiency of J is now obvious because

$$J \begin{pmatrix} x \\ -y \end{pmatrix} = 0.$$

Clearly, J is rank deficient everywhere.

The residual for the constrained problem is $\bar{r}(\bar{x}, y) = r(x, y)$, with x given by (2). The Jacobian of $\bar{r}$ is

$$\bar{J}(\bar{x}, y) = (\bar{J}_{\bar{x}} \quad J_y) \in \mathbb{R}^{l\times(m+n-1)}, \tag{5}$$

where the matrix $\bar{J}_{\bar{x}}$ is defined to be J_x with the first column deleted.

The gradient for the constrained problem can be written

$$\nabla \bar{f} = \bar{J}^T r = \bar{J} \cdot (r)_1 = (\bar{\mathcal{A}} \cdot (r, \cdot, y) \quad \mathcal{A} \cdot (r, x, \cdot)).$$

The Hessian becomes

$$\begin{aligned} H &= \bar{J} \cdot (\bar{J})_1 + \begin{pmatrix} \frac{\partial}{\partial \bar{x}} \\ \frac{\partial}{\partial y} \end{pmatrix} \begin{pmatrix} \bar{\mathcal{A}} \cdot (r, \cdot, y) & \mathcal{A} \cdot (r, x, \cdot) \end{pmatrix} \\ &= \bar{J} \cdot (\bar{J})_1 + \begin{pmatrix} 0 & (\bar{\mathcal{A}} \cdot (r)_1) \cdot (\cdot)_3 \\ (\bar{\mathcal{A}} \cdot (r)_1) \cdot (\cdot)_2 & 0 \end{pmatrix}, \end{aligned}$$

which can be identified with

$$H = \bar{J}^T \bar{J} + \begin{pmatrix} 0 & \bar{A}_r \\ \bar{A}_r^T & 0 \end{pmatrix}, \tag{6}$$

where $\bar{A}_r$ is the matrix $\mathbb{R}^{(m-1)\times n} \ni \bar{A}_r = \bar{\mathcal{A}} \cdot (r)_1$ and $\bar{\mathcal{A}} = \mathcal{A}(:, 2 : m, :)$.

3.2 | Gauss–Newton methods

Gauss–Newton methods are iterative schemes to solve nonlinear least-squares problems; see, for example, chapter 9 of the work of Björck.[4] Let $(x_k; y_k)$ be an approximation of the solution. In the *Gauss–Newton method* for solving (1) with the constraint $e_1^T x = 1$, the following linear least squares problem is solved at iteration k:

$$\min_{\bar{p}} \frac{1}{2} \|\bar{J}\bar{p} + r_k\|^2, \qquad \bar{p} = \begin{pmatrix} \bar{p}_x \\ p_y \end{pmatrix}, \tag{7}$$

where $r_k = r(x_k, y_k)$, $\bar{J} = \bar{J}(\bar{x}_k, y_k)$, and $\bar{p}_x \in \mathbb{R}^{m-1}$. The new iterate is

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + p_k, \qquad p_k = \begin{pmatrix} 0 \\ \bar{p} \end{pmatrix} = \begin{pmatrix} 0 \\ \bar{p}_x \\ p_y \end{pmatrix}.$$

The QR or SVD decompositions can be used to solve the linear least squares problem (7). The Gauss–Newton method has fast local convergence rate for problems with small residuals and nonlinear structure; see the work of Björck.[4(p343)]

The *damped Gauss–Newton method*[4(p343)] is designed to be more robust than the Gauss–Newton method; at iteration k, after finding the Gauss–Newton correction $p_k = (p_x, p_y)$, a line search is performed along p_k, by finding an approximate solution α_k of the following one-dimensional minimization problem:

$$\min_{\alpha\geq 0} \|\mathcal{A} \cdot (x_k + \alpha p_x, y_k + \alpha p_y)_{2,3} - b\|^2,$$

and the new iterate is

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \alpha_k p_k.$$

By straightforward calculation, we have

$$\mathcal{A} \cdot (x_k + \alpha p_x, y_k + \alpha p_y)_{2,3} - b = \mathcal{A} \cdot (x_k, y_k)_{2,3} + \alpha \left(\mathcal{A} \cdot (x_k, p_y)_{2,3} + \mathcal{A} \cdot (p_x, y_k)_{2,3} \right) - b + \alpha^2 \mathcal{A} \cdot (p_x, p_y)_{2,3},$$

and

$$\begin{aligned} g(\alpha) &= \|\mathcal{A} \cdot (x_k + \alpha p_x, y_k + \alpha p_y)_{2,3} - b\|^2 \\ &= \|\beta\|^2 \alpha^4 + (2\beta^T \gamma)\alpha^3 + (\gamma^T \gamma + 2r^T \beta)\alpha^2 + (2\gamma^T r)\alpha + \|r\|^2, \end{aligned}$$

where

$$r = \mathcal{A} \cdot (x_k, y_k)_{2,3} - b, \quad \gamma = \mathcal{A} \cdot (x_k, p_y)_{2,3} + \mathcal{A} \cdot (p_x, y_k)_{2,3}, \quad \beta = \mathcal{A} \cdot (p_x, p_y)_{2,3}.$$

The function $g(\alpha)$ is a fourth-degree polynomial in α, so after the coefficients of the polynomial are computed, a line search is very cheap, as we avoid computing extra tensor–vector multiplications. The actual line search is performed using MATLAB's one-dimensional function minimizer `fminbd`.

3.3 | The VP method

As the LBLS problem (1) is bilinear, that is, linear in each of x and y separately, it is a separable nonlinear least squares problem in the sense of the work of Golub et al.,[5] which can be solved by the *VP method*. For notational convenience, we rewrite the residual in (1) as

$$\|\phi_x(y) - b\|,$$

where, for fixed x satisfying (2), $\phi_x(\cdot) = \mathcal{A} \cdot (x, \cdot)_{2,3}$. To formulate the Golub–Pereyra scheme from the work of Golub et al.[5] for our separable problem, we use the Moore–Penrose pseudoinverse of ϕ_x. For fixed x, the solution of $\min_y \|\phi_x(y) - b\|$, is given by $y = \phi_x^+ b$. Therefore, (1) is equivalent to

$$\min_{e_1^T x = 1} \|(I_l - P_{\phi_x})b\|^2, \qquad y = \phi_x^+ b, \tag{8}$$

where $P_{\phi_x} = \phi_x \phi_x^+$, is an orthogonal projector.

The VP method of Golub et al.[5] is essentially the Gauss–Newton method applied to the nonlinear least squares problem (8). Thus, we shall compute the gradient with respect to $\bar{x}$ of

$$f(x) = \frac{1}{2}\|(I_l - P_{\phi_x})b\|^2.$$

As in Section 3.1, we need to compute the Jacobian of $r(x) = (I_l - P_{\phi_x})b$. We have

$$J(x) = -D(P_{\phi_x})b, \tag{9}$$

where $D(P_{\phi_x})$ is the Fréchet derivative with respect to $\bar{x}$ of the orthogonal projector P_{ϕ_x}. Kaufman[15] proposed an approximation, which is demonstrated to be more efficient than the exact formula (see lemma 4.1 of the work of Golub et al.[5]). In the notation of Kaufman,[15]

$$\widehat{J} = P_{\phi_x}^{\perp} D(\phi_x)\phi_x^+ b, \tag{10}$$

where $P_{\phi_x}^{\perp} = I_l - \phi_x \phi_x^+$. Note that $\widehat{J}$ is a matrix/operator*. Clearly, as $\phi_x(\cdot) = \mathcal{A} \cdot (x, \cdot)_{2,3}$, and because we differentiate with respect to $\bar{x}$, the Fréchet derivative is $\bar{\mathcal{A}} \cdot (\cdot)_3$, where $\bar{\mathcal{A}} = \mathcal{A}(:, 2:m, :)$. Thus, because $\phi_x^+ b \in \mathbb{R}^n$, we can write

$$D(\phi_x)\phi_x^+ b = \bar{\mathcal{A}} \cdot (\phi_x^+ b)_3 \in \mathbb{R}^{l \times (m-1)}.$$

Now recall that $\phi_x = J_y$. Let the thin QR decomposition of $\phi_x \in \mathbb{R}^{l \times n}$ be

$$\phi_x = J_y = Q_y R_y, \qquad Q_y \in \mathbb{R}^{l \times n}, \quad R_y \in \mathbb{R}^{n \times n}.$$

Then, assuming that ϕ_x has full column rank, $\phi_x^+ = R_y^{-1} Q_y^T$, and $P_{\phi_x}^{\perp} = I - Q_y Q_y^T$. Now, (10) becomes

$$\begin{aligned} &\widehat{J} : \mathbb{R}^{m-1} \mapsto \mathbb{R}^l, \\ &\widehat{J}(\cdot) = \left(P_{\phi_x}^{\perp} D(\phi_x)\phi_x^+ b\right)(\cdot) = \left(I_l - Q_y Q_y^T\right)_1 \cdot \left(\left[\bar{\mathcal{A}} \cdot \left(\cdot, R_y^{-1} Q_y^T b\right)_{2,3}\right]\right). \end{aligned} \tag{11}$$

This formula makes perfect sense: We eliminate the variable y and project so that the range of $\widehat{J}$ becomes orthogonal to that of J_y. Thus, in the approximate Gauss–Newton method for (8), we compute the correction $\bar{p}_x$ for $\bar{x}$, by solving the least squares problem

$$\min_{p_x} \|\widehat{J}\bar{p}_x + r\|.$$

We will refer to this method as VPx.

By analogous derivations, one can formulate the VP method for the case when the constraint is still imposed on x, but x is eliminated and a minimization problem for y is solved. Here, we write

$$\|\psi_y(\bar{x}) - b(y)\|, \qquad \psi_y(\cdot) = \bar{\mathcal{A}} \cdot (\cdot, y)_{2,3}, \qquad b(y) = b - \mathcal{A}_1 \cdot (y)_3,$$

*The slight confusion in (10) is due to the numerical linear algebra conventions for matrix–vector multiplication. The formula is clarified in (11).

where $\bar{\mathcal{A}} = \mathcal{A}(:, 2:m, :)$ and $\mathcal{A}_1 = \mathcal{A}(:, 1, :)$. Let A_1 be the matrix identified with $\mathcal{A}_1$. The approximate Jacobian becomes

$$-\left(I_l - \psi_y \psi_y^+\right)_1 \cdot \left[\left(\bar{\mathcal{A}} \cdot \left(\psi_y^+ b(y), \cdot\right)_{2,3} + A_1\right)\right].$$

This method will be referred to as VPy.

3.4 | Newton's method

Due to the bilinear nature of the problem, the cost for an iteration with Newton's method is of the same order of magnitude as for the other methods; see Section 3.6. A Newton step amounts to solving $H\bar{p} = -\bar{J}^T r$, where H and $\bar{p}$ are given by (6) and (7), respectively.

As Newton's method is sensitive to the initial approximation, we cannot expect it to converge properly if used as a stand-alone method. We will use it in combination with VPx for problems with slower convergence.

3.5 | Alternating least squares algorithm

Alternating algorithms are often used for solving the bilinear problem.[1,2] First, an initial approximation for y is inserted, and then, the linear least squares problem for x is solved. Then, the newly computed x is inserted and one solves for y. After each such iteration, the solution is normalized, for example, by putting one component of x equal to 1, and rescaling x and y accordingly. Typically, alternating algorithms converge slowly and, in general, they are not guaranteed to converge to a local minimizer.[3,16] We will see below that the convergence rate depends strongly on the conditioning of the problem.

We will refer to the alternating method as ALS. We will also perform a few ALS iterations as a starting method for the G-N and VP methods.

3.6 | Computational work

Here and in Section 5, we will compare the following methods:

- Gauss–Newton (G-N),
- Damped Gauss–Newton (DG-N),
- variable projection (VPx and VPy),
- variable projection followed by Newton (VPxN), and
- alternating iterations (ALS).

We first emphasize that, for the problems that occur, for example, in Hammerstein identification, the dimensions are so small that measuring the computational work in terms of operation counts is irrelevant; only the rate of convergence and accuracy matter.

On the other hand, because *large and sparse* tensors occur in some applications,[10,11] it is of interest to discuss how large problems with *dense* tensors can be solved by the methods of this paper.

Clearly, for large and dense problems, memory may become a concern. For instance, a tensor of dimensions $1000 \times 500 \times 500$ requires about 2 GB of memory, which may be large on some computers.

The operations with highest operation counts are tensor–vector multiplication (`ttv` in the TensorToolbox[17]), QR factorization, and matrix multiplication, which all have third-order complexity, that is, $O(lmn)$, essentially. However, because QR and matrix multiplication are performed with highly optimized code, and because `ttv` for large dense tensors is probably dominated by data movement, `ttv` can be expected to be slower than QR and matrix multiplication. We performed a timing experiment in MATLAB with a $800 \times 300 \times 300$ tensor, and here, the times for tensor–vector multiplication in the first and second modes were 4–5 times longer (0.5 s) than for multiplication in the third mode. In connection with the same tensor, a QR decomposition of a 800×600 matrix (such as in G-N) required 0.03 s, approximately.

Thus, we can get a rough comparison between the methods by counting how many `ttv`'s, QR decompositions, and matrix multiplications are performed per iteration. The residual r is needed for the gradient computation; here, one can reuse the Jacobian so that no extra `ttv` is needed. In VPx, VPy, and ALS, the QR decompositions are for the blocks of the Jacobians. Therefore, to make it a fair comparison, we have counted each such computation as one half QR. We give the numbers in Table 1, which shows that all methods require roughly the same work per iteration.

TABLE 1 The number of operations per iteration

	G-N	DG-N	VPx	VPy	Newton	ALS
`ttv`	2	3	3	3	3	3
QR	1	1	1.5	1.5	1	1
Matrix mult.			1	1		

From the above arguments and Table 1, we see that it makes sense to compare the efficiency of all the methods by counting the number of iterations for convergence.

3.7 | Convergence rate I

It is shown in the work of Ruhe et al.[3] that all the methods mentioned, except ALS, may have asymptotic superlinear convergence (of course, Newton's method has quadratic convergence). The convergence rates of the G-N and VP methods depend in a nontrivial way on the Hessian matrix for the iterations, but the conclusion of Ruhe et al.[3] is that they still have essentially the same convergence rate. We will see this in the numerical experiments.

For problems with residual r close to zero, that is, where the data fit the model well, and the noise level of the data is small, all methods, except ALS, exhibit *almost* quadratic convergence. This is because the second term in the Hessian (6) becomes insignificant as the iterations proceed, and the G-N and VP methods approach Newton's method. For problems with larger residual, we get linear convergence.[3(p320)]

We come back to convergence rate in Section 4.2.

3.8 | Preliminary examples

To illustrate the performance of the algorithms, we solved two numerical examples, which are derived from a Hammerstein identification problem. For more details of the problem, see Section 5.1. A well-conditioned problem with tensor $\mathcal{A}$ of dimension $100 \times 3 \times 5$ was constructed with a corresponding right-hand side b. The problem was solved with the constraint $e_1^T x = 1$. After one initial iteration with an alternating method ALS, the convergence of all methods was fast (see the left graph of Figure 1).

We then made the Hammerstein problem considerably more ill conditioned (for details, see Section 5.1). With the constraint $e_1^T x = 1$ and one initial ALS iteration, the convergence rate of all methods was slower, and for ALS, much slower. VPy could not attain the required tolerance within 50 iterations (see the right graph of Figure 1). If we constrained any of the components of y, then convergence was very fast for all methods except ALS.

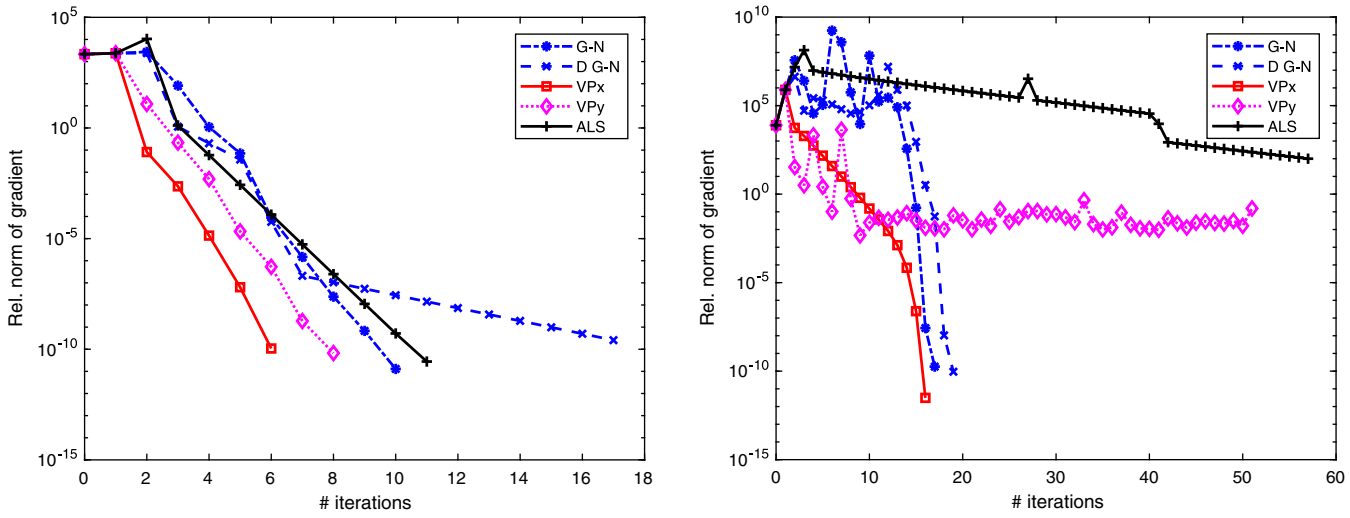

FIGURE 1 Convergence history: relative norm of gradient for different methods. Well-conditioned (left) and ill-conditioned (right) Hammerstein problems

Clearly, the convergence properties depend on the conditioning of the problem and on how the linear constraint is chosen. These two examples demonstrate that an understanding of the conditioning of the LBLS problem is needed.

4 | PERTURBATION THEORY

Consider the first-order condition for the linearly constrained BLS problem,

$$\bar{J}^T r(x, y) = 0,$$

where $\bar{J}$ is defined in (5), and (x, y) is a stationary point. Add a small perturbation to the tensor and the right-hand side, $\widetilde{\mathcal{A}} = \mathcal{A} + \delta\mathcal{A}$ and $\widetilde{b} = b + \delta b$, where

$$\|\delta\mathcal{A}\| \le \epsilon, \qquad \|\delta b\| \le \epsilon. \tag{12}$$

Further, assume that the perturbation of $\mathcal{A}$ is so small that the perturbed Jacobian $\widetilde{J}^T$ has full column rank, and the perturbed Hessian is positive definite. Denote the perturbed solution

$$x + \delta x = \begin{pmatrix} 1 \\ \bar{x} \end{pmatrix} + \begin{pmatrix} 0 \\ \delta\bar{x} \end{pmatrix}, \qquad y + \delta y, \qquad r + \delta r. \tag{13}$$

A straightforward calculation (see Appendix A) shows that the first-order condition, after omitting terms that are $\mathcal{O}(\epsilon^2)$, gives the following equation for the perturbation of the solution:

$$\begin{pmatrix} -I & \bar{J}_{\bar{x}} & J_y \\ \bar{J}_{\bar{x}}^T & 0 & \bar{A}_r \\ J_y^T & \bar{A}_r^T & 0 \end{pmatrix} \begin{pmatrix} \delta r \\ \delta\bar{x} \\ \delta y \end{pmatrix} = \begin{pmatrix} \delta b - \delta\mathcal{A}\cdot(x, y)_{2,3} \\ -\delta\bar{\mathcal{A}}\cdot(r, y)_{1,3} \\ -\delta\mathcal{A}\cdot(r, x)_{1,2} \end{pmatrix}, \tag{14}$$

where $\bar{A}_r$ is identified with $\bar{\mathcal{A}}\cdot(r)_1$, and $\delta\bar{\mathcal{A}}\cdot(r, y)_{1,3} \in \mathbb{R}^{m-1}$ and $\delta\mathcal{A}\cdot(r, x)_{1,2} \in \mathbb{R}^n$ are treated as column vectors. The matrix in (14) is the Hessian when r has been included in the problem formulation.

In order to express the solution conveniently, we rewrite (14) as

$$\begin{pmatrix} -I & \bar{J} \\ \bar{J}^T & T_r \end{pmatrix} \begin{pmatrix} \delta r \\ \delta z \end{pmatrix} = \begin{pmatrix} \delta b - \delta\mathcal{A}\cdot(x, y)_{2,3} \\ \delta\widetilde{\mathcal{A}}(r, x, y) \end{pmatrix}, \tag{15}$$

where

$$T_r = \begin{pmatrix} 0 & \bar{A}_r \\ \bar{A}_r^T & 0 \end{pmatrix}, \qquad \delta z = \begin{pmatrix} \delta\bar{x} \\ \delta y \end{pmatrix}, \qquad \delta\widetilde{\mathcal{A}}(r, x, y) = \begin{pmatrix} -\delta\bar{\mathcal{A}}\cdot(r, y)_{1,3} \\ -\delta\mathcal{A}\cdot(r, x)_{1,2} \end{pmatrix}. \tag{16}$$

The solution is

$$\begin{pmatrix} \delta r \\ \delta z \end{pmatrix} = \begin{pmatrix} -I + \bar{J}H^{-1}\bar{J}^T & \bar{J}H^{-1} \\ H^{-1}\bar{J}^T & H^{-1} \end{pmatrix} \begin{pmatrix} \delta b - \delta\mathcal{A}\cdot(x, y)_{2,3} \\ \delta\widetilde{\mathcal{A}}(r, x, y) \end{pmatrix}, \tag{17}$$

where $H = \bar{J}^T\bar{J} + T_r$ is the Hessian (6).

We can now estimate the perturbations $\|\delta r\|$ and $\|\delta z\|$.

Theorem 1. *Consider the bilinear least squares problem with linear equality constraint (1) with full rank Jacobian $\bar{J}$ and positive definite Hessian H. Let $\mathcal{A} + \delta\mathcal{A}$ and $b + \delta b$ be perturbed data and assume that $\delta\mathcal{A}$ is so small that the perturbed Jacobian has full rank, and the perturbed Hessian is positive definite. Let (x, y) be a stationary point, and denote the perturbed solution as in (13). Then, if second-order terms are neglected, we can estimate from (17)*

$$\|\delta r\| \lessapprox \left(1 + \frac{\sigma_{\max}^2(\bar{J})}{\lambda_{\min}(H)}\right)\left(\|\delta b\| + \|\delta\mathcal{A}\cdot(x, y)_{2,3}\|\right) + \frac{\sigma_{\max}(\bar{J})\|r\|\,\|\delta\mathcal{A}\|}{\lambda_{\min}(H)}\left\|\begin{pmatrix} x \\ y \end{pmatrix}\right\|, \tag{18}$$

$$\left\|\begin{pmatrix} \delta\bar{x} \\ \delta y \end{pmatrix}\right\| \lessapprox \frac{\sigma_{\max}(\bar{J})}{\lambda_{\min}(H)}(\|\delta b\| + \|\delta\mathcal{A}\cdot(x, y)_{2,3}\|) + \frac{\|r\|\,\|\delta\mathcal{A}\|}{\lambda_{\min}(H)}\left\|\begin{pmatrix} x \\ y \end{pmatrix}\right\|, \tag{19}$$

where $\lambda_{\min}(H)$ is the smallest eigenvalue of H and $\sigma_{\max}(\bar{J})$ is the largest singular value of $\bar{J}$.

Proof. The results are proved by straightforward estimation of the terms in (17), using $\|H^{-1}\|_2 \leq 1/\lambda_{\min}(H)$ and $\|\bar{J}\|\|_2 \leq \sigma_{\max}(\bar{J})$. The last term in both right-hand sides follows from

$$\left\|\delta\widetilde{\mathcal{A}}(r,x,y)\right\|^2 = \left\|\begin{pmatrix} \delta\bar{\mathcal{A}}\cdot(r,y)_{1,3} \\ \delta\mathcal{A}\cdot(r,x)_{1,2} \end{pmatrix}\right\|^2 \leq \left(\|\delta\bar{\mathcal{A}}\|^2\ \|y\|^2 + \|\delta\mathcal{A}\|^2\ \|x\|^2\right)\|r\|^2,$$

using $\|\delta\bar{\mathcal{A}}\| \leq \|\delta\mathcal{A}\|$. □

The estimates (18) and (19) are fundamental in the sense that they are based on the geometry of the problem at the stationary point (see section 9.1.2 of the work of Björck[4] and see also the work of Wedin[18]). However, the eigenvalues of H depend on the residual and that dependence is not visible in the estimates. Therefore, in Theorem 2, we derive alternative estimates that are valid for small residuals. First, we give two lemmas.

Lemma 1. *Let T_r be defined by (16).*
Then,

$$\|T_r\|_2 \leq \|\mathcal{A}\|\ \|r\|. \tag{20}$$

Proof. Because the eigenvalues of the symmetric matrix T_r are plus/minus the singular values of $\bar{A}_r$, we have $\|T_r\|_2 = \|\bar{A}_r\|_2$. Let $\bar{A}^{(1)}$ be the mode-1 matricization of $\bar{\mathcal{A}}$, that is, it is the matrix whose columns are all the mode-1 vectors of $\bar{\mathcal{A}}$. Then,

$$\|\bar{A}_r\|_2 = \|\bar{\mathcal{A}}\cdot(r)_1\|_2 = \|r^T\bar{A}^{(1)}\|_2 \leq \|r\|\ \|\bar{A}^{(1)}\|_2 \leq \|r\|\ \|\bar{A}^{(1)}\| = \|r\|\ \|\bar{\mathcal{A}}\| \leq \|r\|\ \|\mathcal{A}\|,$$

where we have used the inequality $\|A\|_2 \leq \|A\|$. □

Lemma 2. *Let $\bar{J} = QR$ be the thin QR decomposition, and let $\sigma_{\min}(\bar{J}) =: \sigma_{\min} > 0$ be the smallest singular value of $\bar{J}$ (and R). Assume that, at a stationary point,*

$$\frac{\|\mathcal{A}\|\ \|r\|}{\sigma_{\min}^2} = \eta < 1, \tag{21}$$

holds. Then,

$$\|H^{-1}\|_2 \leq \frac{1}{\sigma_{\min}^2}\ \frac{1}{1-\eta}, \tag{22}$$

$$\|\bar{J}H^{-1}\|_2 \leq \frac{1}{\sigma_{\min}}\ \frac{1}{1-\eta}, \tag{23}$$

$$\|\bar{J}H^{-1}\bar{J}^T\|_2 \leq \frac{1}{1-\eta}. \tag{24}$$

Proof. Using the QR decomposition, we have

$$H^{-1} = \left(R^TR + T_r\right)^{-1} = R^{-1}\left(I + R^{-T}T_rR^{-1}\right)^{-1}R^{-T},$$

and

$$\|H^{-1}\|_2 \leq \|R^{-1}\|_2^2\ \|\left(I + R^{-T}T_rR^{-1}\right)^{-1}\|_2.$$

By Lemma 1 and assumption (21), we have

$$\left\|R^{-T}T_rR^{-1}\right\|_2 \leq \eta < 1.$$

For any matrix X, which satisfies $\|X\|_2 < 1$, we can estimate $\|(I+X)^{-1}\|_2 < 1/(1-\|X\|_2)$. Therefore, we get

$$\|H^{-1}\|_2 \leq \frac{1}{\sigma_{\min}^2}\ \frac{1}{1-\eta}.$$

The proofs of the other two inequalities are analogous. □

It is seen from section 9.1.2 of the work of Björck[4] that (21) is a sufficient condition for the Hessian to be positive definite and, thus, for the stationary point to be a local minimum. In section 9.1.2 of the work of Björck,[4] a slightly weaker condition is used; however, using (21), we can see in the following theorem the likeness to the perturbation theory for the linear least squares problem.

Theorem 2. *Let the assumptions of Theorem 1 hold, and further assume that (21) is satisfied. Then, if second-order terms are neglected, we can estimate from (17),*

$$\|\delta r\| \lessapprox \frac{1}{1-\eta}\left((2-\eta)(\|\delta b\| + \|\delta\mathcal{A}\cdot(x,y)_{2,3}\|) + \frac{\|r\|\ \|\delta\mathcal{A}\|}{\sigma_{\min}}\left\|\begin{pmatrix} x \\ y\end{pmatrix}\right\|\right), \tag{25}$$

$$\left\|\begin{pmatrix} \delta\bar{x} \\ \delta y\end{pmatrix}\right\| \lessapprox \frac{1}{1-\eta}\left(\frac{1}{\sigma_{\min}}(\|\delta b\| + \|\delta\mathcal{A}\cdot(x,y)_{2,3}\|) + \frac{\|r\|\ \|\delta\mathcal{A}\|}{\sigma_{\min}^2}\left\|\begin{pmatrix} x \\ y\end{pmatrix}\right\|\right). \tag{26}$$

Proof. The results are proved by straightforward estimation of the terms in (17), using Lemma 2. □

The estimates (25)–(26) have (not surprisingly) the same structure as the corresponding estimates for the linear least squares problem; see, for example, theorem 2.2.6 of the work of Björck.[4] For instance, if $r = 0$, then the conditioning of the solution x and y together depends essentially on the condition number of the Jacobian. However, if $r \neq 0$, there is also a dependence on the square of the condition number of the Jacobian in the form $\|r\|/\sigma_{\min}^2$.

The second estimate (19) has the disadvantage that $\delta\bar{x}$ and δy are lumped together. Due to the bilinear and separable nature of the problem, one would like to have separate estimates for these perturbations. We will now derive such a result for the case $r = 0$.

4.1 | Estimates of $\|\delta\bar{x}\|$ and $\|\delta y\|$ in the case $r = 0$

Assuming that the problem is consistent, $r = 0$, we derive explicit expressions for the perturbations $\delta\bar{x}$ and δy. We further assume that $\bar{J}$ has full column rank. Using the partitioned Jacobian (5), we can write the second equation of (17) as

$$S\begin{pmatrix} \delta\bar{x} \\ \delta y\end{pmatrix} = \begin{pmatrix} \bar{J}_x^T\bar{J}_x & \bar{J}_x^T J_y \\ J_y^T\bar{J}_x & J_y^T J_y\end{pmatrix}\begin{pmatrix} \delta\bar{x} \\ \delta y\end{pmatrix} = \begin{pmatrix} \bar{J}_x^T \\ J_y^T\end{pmatrix} c, \tag{27}$$

where $c = \delta b - \delta\mathcal{A}\cdot(x,y)_{2,3}$. Let the thin QR decompositions of the blocks of $\bar{J}$ be

$$\bar{J}_x = Q_x R_x, \qquad Q_x \in \mathbb{R}^{l\times(m-1)}, \quad R_x \in \mathbb{R}^{(m-1)\times(m-1)}, \tag{28}$$

$$J_y = Q_y R_y, \qquad Q_y \in \mathbb{R}^{l\times n}, \quad R_y \in \mathbb{R}^{n\times n}, \tag{29}$$

where the columns of Q_x and Q_y are orthonormal, and R_x and R_y are upper triangular and nonsingular. Inserting the QR factors in (27), we get

$$\begin{pmatrix} R_x^T R_x & R_x^T Q_x^T Q_y R_y \\ R_y^T Q_y^T Q_x R_x & R_y^T R_y\end{pmatrix}\begin{pmatrix} \delta\bar{x} \\ \delta y\end{pmatrix} = \begin{pmatrix} R_x^T Q_x^T \\ R_y^T Q_y^T\end{pmatrix} c. \tag{30}$$

Defining

$$\begin{pmatrix} \delta_x \\ \delta_y\end{pmatrix} = \begin{pmatrix} R_x\delta\bar{x} \\ R_y\delta y\end{pmatrix}, \tag{31}$$

$E = Q_x^T Q_y$, and multiplying by the inverses of R_x^T and R_y^T, (30) becomes

$$\begin{pmatrix} I & E \\ E^T & I\end{pmatrix}\begin{pmatrix} \delta_x \\ \delta_y\end{pmatrix} = \begin{pmatrix} Q_x^T \\ Q_y^T\end{pmatrix} c.$$

It is easy to verify that

$$\begin{pmatrix} I & E \\ E^T & I\end{pmatrix}^{-1} = \begin{pmatrix} I + E\widetilde{S}^{-1}E^T & -E\widetilde{S}^{-1} \\ -\widetilde{S}^{-1}E^T & \widetilde{S}^{-1}\end{pmatrix},$$

where $\widetilde{S} = I - E^T E = Q_y^T(I - Q_x Q_x^T)^T Q_y$. The nonsingularity of the Schur complement $\widetilde{S}$ follows from the fact that the matrix

$$\begin{pmatrix} I & E \\ E^T & I\end{pmatrix}$$

is positive definite. Therefore, we have

$$\delta_x = \left((I + E\widetilde{S}^{-1}E^T)Q_x^T - E\widetilde{S}^{-1}Q_y^T\right) c.$$

After rewriting this and using $\delta_x = R_x \delta\bar{x}$, we get

$$\delta\bar{x} = R_x^{-1} Q_x^T P_x c, \qquad P_x = I - Q_y \left(Q_y^T \left(I - Q_x Q_x^T \right) Q_y \right)^{-1} Q_y^T \left(I - Q_x Q_x^T \right). \tag{32}$$

We recognize P_x as an oblique projection; see, for example, the work of Hansen.[19] Clearly, we have derived part of the Moore–Penrose pseudoinverse of $\bar{J}$.

Proposition 1. *Assume that the matrix $\bar{J} = (\bar{J}_x \quad J_y)$ has full column rank and that the blocks have QR decompositions (28)–(29). Then, its Moore–Penrose pseudoinverse is equal to*

$$\bar{J}^+ = \begin{pmatrix} \bar{J}_x^+ P_x \\ J_y^+ P_y \end{pmatrix} = \begin{pmatrix} R_x^{-1} Q_x^T P_x \\ R_y^{-1} Q_y^T P_y \end{pmatrix}, \tag{33}$$

where P_x is defined in (32) and P_y is symmetrically defined.

The proof is given in Appendix B.

We now have the following theorem.

Theorem 3. *Assume that, at the optimal point (x, y), the Jacobian $\bar{J}$ and the perturbed Jacobian have full column rank and that the residual satisfies $r = 0$. Let $\sigma_{\min}(R_x) > 0$ and $\sigma_{\min}(R_y) > 0$ be the smallest singular values of R_x and R_y, respectively. Then, to first order,*

$$\|\delta\bar{x}\| \lessapprox \frac{\left\| Q_x^T P_x \right\|}{\sigma_{\min}(R_x)} \left(\|\delta b\| + \|\delta\mathcal{A} \cdot (x, y)_{2,3}\| \right), \tag{34}$$

$$\|\delta y\| \lessapprox \frac{\left\| Q_y^T P_y \right\|}{\sigma_{\min}(R_y)} \left(\|\delta b\| + \|\delta\mathcal{A} \cdot (x, y)_{2,3}\| \right), \tag{35}$$

where P_x is defined in (32), and P_y is symmetrically defined.

The theorem shows that, in this case, the conditioning of x and y depends on the condition number of R_x and R_y, respectively, and on the geometric properties of the range spaces of $\bar{J}_x$ and J_y.[20] Note that the norm of an oblique projection is usually larger than 1. If the range of $\bar{J}_x$ is orthogonal to that of J_y, that is, $Q_x^T Q_y = 0$, then P_x becomes an orthogonal projection onto the null space of J_y, and $Q_x^T P_x = Q_x^T$, that is, the perturbations of $\delta\bar{x}$ and δy are completely independent.

If the residual is small and the problem is not too ill conditioned, that is, if

$$\frac{\|\mathcal{A}\| \, \|r\|}{\sigma_{\min}^2} = \eta \ll 1,$$

is satisfied, then the estimates (34)–(35) hold approximately also in the case $r \neq 0$. Therefore, it makes sense to base decisions on the implementation of the linear constraint on the conditioning of $\bar{J}_x$ and J_y, separately.

4.2 | Convergence rate II

From the work of Ramsin et al.[21] and theorem 1 of the work of Wedin,[18] one can prove that the convergence factor ρ of the Gauss–Newton method is bounded from above by the eigenvalue of the largest modulus of the curvature matrix $K = -(\bar{J}^+)^T T_r \bar{J}^+$ at the stationary point[†], where $\bar{J}^+$ is the Moore–Penrose pseudoinverse. Thus,

$$\rho \leq \max_i |\lambda_i(K)| =: \hat{\rho}. \tag{36}$$

As $\bar{J}^+ = R^{-1} Q^T$, the convergence rate depends on the conditioning of $\bar{J}$. Moreover, we have an expression (33) for the pseudoinverse, which, using the block structure of T_r, gives

$$K = -(C + C^T), \qquad C = P_x^T Q_x R_x^{-T} \bar{A}_r R_y^{-1} Q_y^T P_y.$$

Therefore, the convergence rate depends also on the conditioning of the two blocks of the Jacobian. As it would be quite expensive to choose the solution component to constrain so that $\hat{\rho}$ is minimized, we base the decision on the conditioning of the Jacobian or the two blocks of the Jacobian.

[†]The factor ρ is the maximum of a Rayleigh quotient for K over tangent vectors of the surface $\|r(x, y)\|$.

4.3 | Implementation of the linear constraint

In the description of the algorithms, we assumed that the linear constraint is $e_1^T x = 1$. The preliminary experiments in Section 3.8 show clearly that the actual choice of which component of x or y to put equal to 1 can make a difference in the convergence speed of the algorithms, especially when the problem is ill conditioned. This is confirmed by the theoretical arguments in the preceding section.

In the absence of other information, the natural starting value for x and y is a random vector. In order to collect some information about the problem at hand, we propose to start the solution procedure by iterating a small number of steps with ALS and then try to choose the constraint so that the problem becomes as well conditioned as possible, which presumably would lead to fast convergence.

Consider first the G-N methods. Let $(x^{(k)}, y^{(k)})$ be the result after k ALS iterations, and compute the Jacobian $J = (J_x\ J_y) = (\mathcal{A} \cdot (y^{(k)})_3\ \mathcal{A} \cdot (x^{(k)})_2)$. From the thin QR decomposition $J = QR$, we delete the columns of R one at a time and check the condition number of the matrix of remaining columns. We let the best conditioned matrix determine which component of x or y to be put equal to 1.

From Theorem 3, we see that the conditioning x and y may be different. Therefore, it makes sense to consider them separately and check which constraint on x gives the best conditioned sub-Jacobian, and similarly for y. The overall best conditioned constraint is chosen. Using this procedure for VPx, we eliminate the less well-conditioned part of the solution and iterate for the best conditioned part. With VPy, we do the opposite: We eliminate the best conditioned part and iterate for the other. Note, however, that, as is shown in the work of Ruhe et al.[3] and is visible in our experiments, the asymptotic rate of convergence is the same for the G-N and VP methods. We will see that our choice of constraint improves the behavior of VPx in early iterations, as compared with that of G-N.

In view of the slow convergence of ALS, the selection of the constrained component may be improved if, after a few iterations with the G-N and VP methods, we recheck which constraint gives the best conditioned matrix.

Consider now the computational cost for determining the constraint that gives the best conditioned problem for the G-N methods. For small problems (e.g., the Hammerstein problems in Section 5), the computing time for checking the conditioning by computing the singular value decomposition (SVD) of the respective Jacobian with one column deleted would be negligible. However, the cost is $O(l(m + n)^3)$, so for larger problems, this may become significant. In principle, it can be reduced to $O(l(m + n)^2)$ by using algorithms for updating the R factor in the QR decomposition (see section 6.5.2 of the work of Golub et al.[14]), combined with condition estimators (see section 3.5.4 of the work of Golub et al.[14]). Therefore, for large problems, the cost for checking the conditioning would be of the same order of magnitude as that for one iteration of the Gauss–Newton methods (see Section 3.6).

On the other hand, for problems of dimension of the order 200–1000, the efficiency of the function `qr` in MATLAB is so high that it beats a hand-coded function that performs the updates of the R factor. Therefore, in our numerical experiments, we use the function `qr` for each modified R matrix, combined with the condition estimator `condest` in MATLAB, which avoids the SVD.

The computational cost for the corresponding procedure for the VP methods is lower as we deal with smaller matrices.

5 | NUMERICAL EXPERIMENTS

In this section, we investigate the performance of the different algorithms for the LBLS problem, by solving a small Hammerstein problem and a larger problem with random data. All the numerical experiments were performed using MATLAB 2017b together with the TensorToolbox.[17]

As stopping criterion for the iterations, we used the relative gradient, that is, the norm of the gradient divided by $\|b\|$.

To measure the separated conditioning of the problems solved, based on (34) and (35), we computed the approximate condition numbers

$$\kappa_x = \|\bar{J}_x\| \left\|R_x^{-1} Q_x^T P_x\right\|, \qquad \kappa_y = \|J_y\| \left\|R_y^{-1} Q_y^T P_y\right\|. \tag{37}$$

Here, we have normalized the numbers by multiplying by $\|\bar{J}_x\|$ and $\|J_y\|$, respectively.

5.1 | Test 1: A Hammerstein problem

In the theory of system identification, Hammerstein systems are used to model various practical process; see, for example, the references in the work of Wang et al.[2] Our presentation is based on that paper. The system consists of a static

nonlinearity followed by a linear dynamics. Note that, here, we temporarily use notation that is common in the system identification literature: $x(t)$ is the input signal, $u(t)$ is the internal signal, and $y(t)$ and $v(t)$ are output and noise signals, respectively. The following input–output equation describes the Hammerstein system

$$y(k) = \sum_{j=1}^{n} b_j x(k-j) + v(k), \tag{38}$$

where $k = 1, 2, \dots l$. It is assumed that the nonlinear function $x(t)$ can be expressed as a linear combination of known basis functions ϕ_i,

$$x(k) = f[u(k)] = \sum_{i=1}^{m} a_i \phi_i(u(k)), \tag{39}$$

so from (38) and (39), we have

$$y(k) = \sum_{j=1}^{n} \sum_{i=1}^{m} a_i b_j \phi_i(u(k-j)) + v(k).$$

Thus, the Hammerstein system can be written in terms of tensor notations as

$$\mathcal{A} \cdot (a, b)_{2,3} = y + v,$$

where $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$ is a 3-tensor with elements $a_{ijk} = \phi_i(u(k-j))$.

Due to the noise, one usually considers a least squares problem,

$$\min_{a,b} \|\mathcal{A} \cdot (a, b)_{2,3} - \bar{y}\|,$$

for the identification of the parameters.

In the literature, several methods have been proposed to solve Hammerstein systems, among which we mention the normalized iterative method,[22] the Levenberg–Marquardt method and separable least squares,[23] gradient-based algorithms,[24] et cetera. We simulated a system with the following parameters, which are similar to those in the work of Wang et al.[2]

- $u(k)$ is uniformly distributed in the interval $[-3, 3]$. Then, u is filtered as in work of Wang et al.[2(p2631)] (MATLAB's `filter`) with filter function $1/(1 - 0.5q^{-1})$.
- The noise vector is $\eta = \tau \|\hat{b}\| / \|v\| v$, where the components of v are normally distributed with zero mean and variance 1; we use τ to vary the noise level.
- $\phi_k(t) = t^k, \quad k = 1, \dots, 5.$
- $(l, m, n) = (100, 5, 3)$.

We now go back to the notation that is used in the rest of this paper. We chose the "true" parameters in the system to be

$$x = (1, 2, 5, 7, 1), \quad y = (0.4472, -0.8944, 0.6)$$

and generated the right-hand side $\hat{b} = \mathcal{A} \cdot (x, y)_{2,3}$. As initial approximations, we took normally distributed vectors with mean zero and variance 1, and we started with one ALS iteration. The G-N and VP methods chose to impose the linear constraint on the second component of y. The iterations were stopped when the relative residual was below $0.5 \cdot 10^{-9}$. In our first test, we had no noise ($\tau = 0$). The convergence histories are shown in Figure 2.

Clearly, the problem is very well conditioned: The approximate condition numbers (37) were of the order 407 and 1.2, respectively. With no noise, we can check the accuracy of the solution, and all relative errors were of the order 10^{-15} or smaller.

We also ran the same problem with $\tau = 0.1$. The convergence histories were very similar, except that DG-N was more conservative and converged more slowly, in 20 iterations.

We then made the problem more ill conditioned by sampling u in the interval $[2, 4]$, and we started with no noise, $\tau = 0$. (Clearly, the problem would be less ill conditioned if we used polynomials orthogonal on the interval as basis functions instead of monomials.[23] However, the purpose here was to study the behavior of the algorithms for a more ill-conditioned problem.) For G-N and DG-N, the linear constraint was imposed on the fifth component of x; after five iterations, we checked the conditioning of Jacobians, and then, the third component of y was constrained. For VPx and VPy, the constraint was imposed on the second component of y. The convergence histories are illustrated in Figure 3.

The approximate condition numbers (37) were $1.1 \cdot 10^6$ and 3.9, respectively. The relative errors of the solution were of the order 10^{-11} for x and 10^{-13} for y, with all the methods that converged, which confirms that the separate conditioning makes sense.

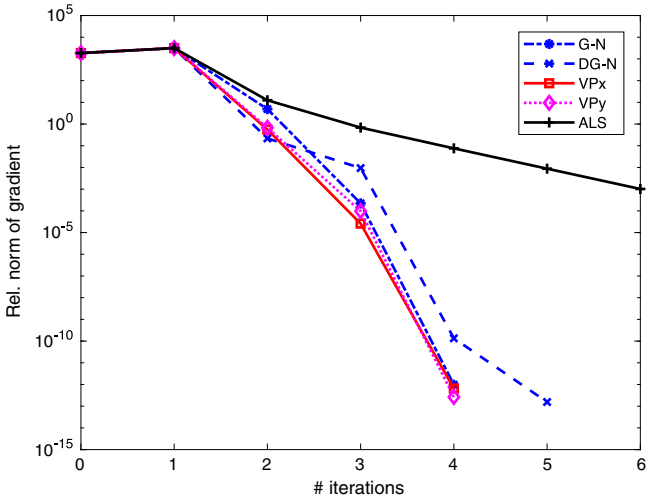

FIGURE 2 Test 1: Well-conditioned Hammerstein problem with no noise. Convergence histories

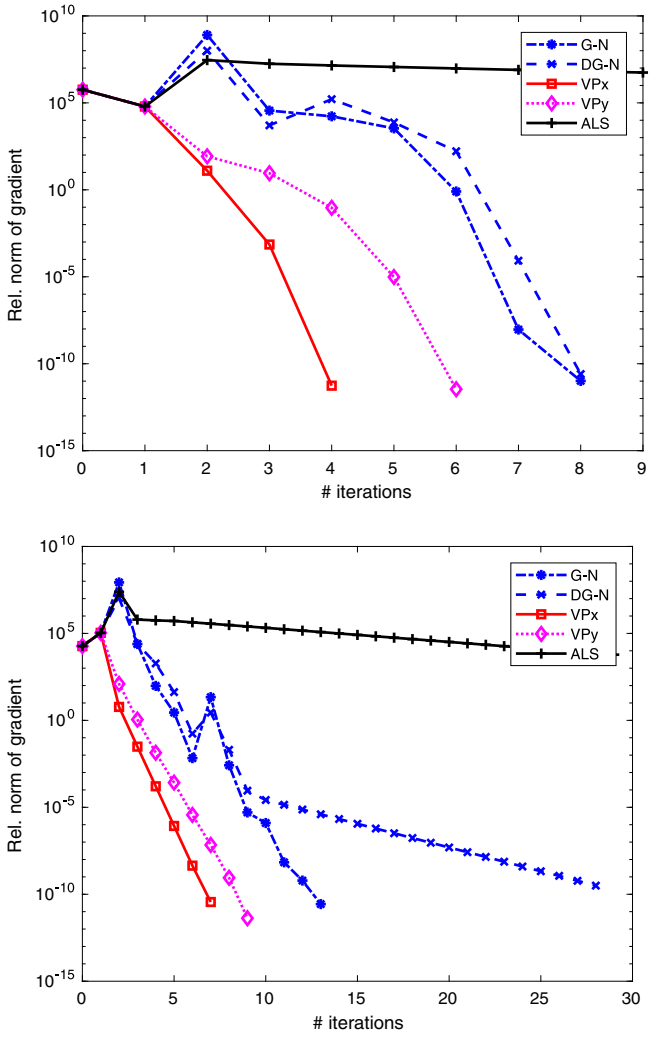

FIGURE 3 Test 1: ill-conditioned Hammerstein problem with no noise. $\tau = 0$ (top) and noise level $\tau = 0.1$ (bottom). Convergence histories

We then added noise with $\tau = 0.1$. For G-N and DG-N, the linear constraint was first imposed on the first component of y and, then, after five iterations, on the fifth component of x. For VPx and VPy, the second component of y was constrained. This problem is very ill conditioned: The condition number of the Hessian was of the order 10^{11}, but it was still positive definite at the solution. The condition (21) was not satisfied, and the separated condition numbers (37) are no longer relevant. The convergence was slower for the Gauss–Newton method (see the bottom plot of Figure 3). Our implementation of the damped Gauss–Newton method seems to be too conservative. On the other hand, VPx takes advantage of the fact that the separated problem for y is well conditioned and converges fast.

Comparing the convergence rates in the two plots of Figure 3, we see that, for $\tau = 0$, the Gauss–Newton and VP methods exhibit superlinear convergence. In the case, when $\tau = 0.1$, the convergence rate is clearly linear. The computed estimate $\hat{\rho}$ of the convergence factor was of the order 0.02, which is consistent with the slope of the curves.

5.2 | Test 2: Random data

We let $\mathcal{A} \in \mathbb{R}^{500\times200\times200}$ be a tensor, whose elements are normally distributed N(0,1); the elements of the vectors x and y are sampled from a uniform distribution in $[0, 1]$. A bilinear least squares problem was constructed with right-hand side

$$b = \bar{b} + \eta, \quad \bar{b} = \mathcal{A} \cdot (x, y)_{2,3}, \quad \eta = \tau \|\bar{b}\| \frac{d}{\|d\|},$$

where the elements of d were normally distributed N(0,1). Before starting the iterations with the tested algorithms, we performed three iterations of the alternating least squares method (starting from the random vectors) to get an initial approximation. The stopping criterion was $0.5 \cdot 10^{-10}$.

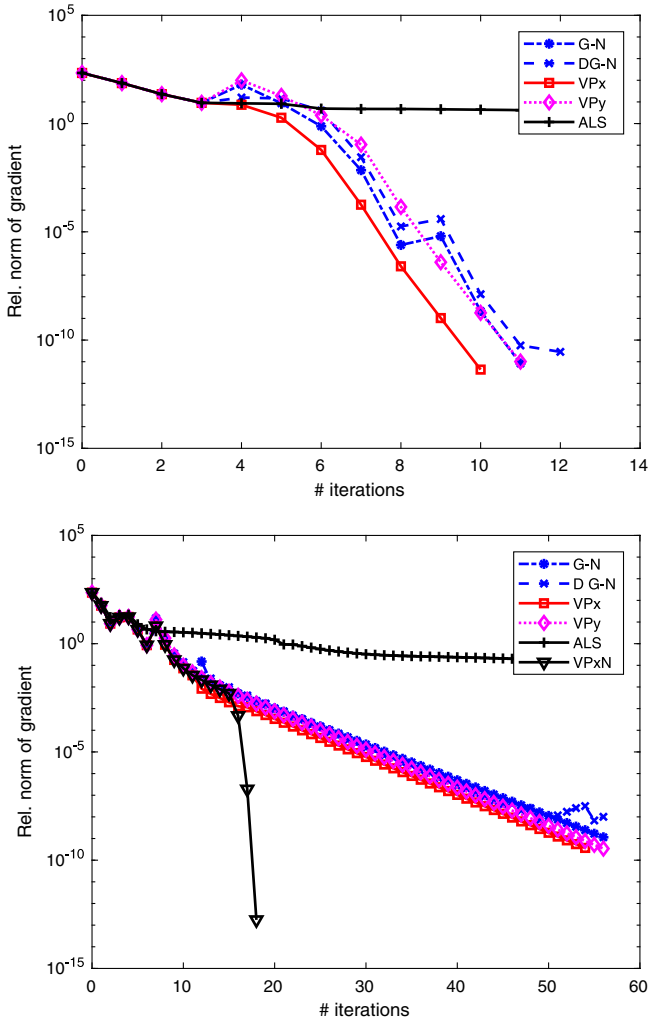

FIGURE 4 Test 2: random tensor of dimensions (500, 200, 200). Convergence histories with $\tau = 0.001$ (top plot) and $\tau = 0.1$ (bottom)

We first let $\tau = 0.001$. As the solution vectors are random, it does not make sense to state which component was chosen to be constrained. It is sufficient to say that, for G-N and DG-N, the constraint was first imposed on a component of y; after five iterations, a component of x was constrained. For VPx and VPy, a component of y was constrained. The convergence is illustrated in Figure 4. The separated condition numbers (37) were of the order 64.

For the examples run so far, the G-N and VP methods all converged reasonably fast and it did not make sense to use Newton's method. However, for the random tensor problem with noise $\tau = 0.1$, the convergence was considerably slower (see Figure 4). The condition number of the Hessian was of the order $5 \cdot 10^3$. The condition (21) was not satisfied. The estimated convergence factor was $\hat{\rho} = 0.69$, which is consistent with the slope of the curves in Figure 4. For this problem, all methods switched constraints.

It did happen for some random tensors with $\tau = 0.1$ that the methods converged to different solutions, corresponding to local minima. To increase robustness, we here performed six ALS iterations initially.

Here, we also tested to use Newton's method in combination with VPx. From the figure, we see that the convergence rate of VPx is linear. If we assume that the norm of the gradient is approximately R^k, where k is the iteration number and R is the convergence rate, then we can estimate R by computing the quotient of two consecutive norms of the gradient. From our experience, this estimate varies in the early iterations and stabilizes as the iterations proceed. When the difference between these estimates for the VPx method was less than 2%, we switched to Newton iterations.

It is difficult to compare the efficiency of algorithms by timing experiments in MATLAB, due to the fact that the programmer does not have control of all the aspects (especially data movement, and optimized functions vs. MATLAB code; cf. Section 3.6) that determine the speed of execution. Therefore, in order to just give a rough estimate of execution times, we here mention that the execution times for the last example were 24, 24, 21, 26, and 8.5 s, for G-N, DG-N, VPx, VPy, and VPxN, respectively (cf. Table 1). The computations were performed on a laptop computer (Intel Core i3-4030U CPU, 1.90 GHz × 4) running Ubuntu Linux.

6 | CONCLUSIONS AND FUTURE WORK

In this paper, we have studied different iterative methods to solve the LBLS problem. The methods were Gauss–Newton methods, G-N and DG-N; VP methods, VPx and VPy; Newton's method; and an alternating least squares method, ALS. Perturbation theory was presented that gave information used in the design of the algorithms. We showed that the work for one iteration was of the same order of magnitude for all the methods, and therefore, we could make a fair comparison by counting the number of iterations for convergence. The methods were tested using a well-conditioned and an ill-conditioned small Hammerstein identification problem and a larger artificial test problem.

Our experiments showed that, overall, the convergence of the ALS method was inferior to all the other methods. VPx converged faster and more consistently than VPy, G-N, and DG-N for the Hammerstein problems and for the well-conditioned artificial problem. For the less well-conditioned artificial problem, with a larger data perturbation, all methods converged more slowly. Here, we used a combination of VPx and Newton's method, which gave a dramatically improved convergence rate.

The results of this paper indicate that the VP method VPx is the method of choice, possibly combined with Newton's method for problems with slower convergence rate. More work is needed to investigate the use of methods with close to quadratic convergence, for example, a robust variant of Newton's method. This could be especially worthwhile because Newton's method requires essentially the same work per iteration as the other methods. We plan to investigate this in the future, with emphasis on problems from actual applications.

The methods in this paper are intended for small- and medium-size problems. Large and sparse problems occur in information sciences; see, for example, the works of Lampos et al.[10,11] We are presently studying algorithms for such problems. By a certain projection, the problem will be reduced to a medium-size LBLS problem, for which the methods of this paper can be used.

More general problems, where the unknowns are matrices, for example,

$$\min_{X,Y} \|\mathcal{A} \cdot (X, Y)_{2,3} - \mathcal{B}\|,$$

are studied in the works of Hoff[9] and Lock.[12] The methods of this paper can be adapted to such problems. This is a topic of our future research.

ACKNOWLEDGEMENT

We are grateful to an anonymous referee for the constructive criticism and helpful comments. The second author was supported by the Mega Grant project (14.756.31.0001).

ORCID


Lars Eldén https://orcid.org/0000-0003-2281-856X

REFERENCES

1. Bai E, Liu D. Least squares solutions of bilinear equations. Syst Control Lett. 2006;55:466–472.
2. Wang J, Zhang Q, Ljung L. Revisiting Hammerstein system identification through the two-stage algorithm for bilinear parameter estimation. Automatica. 2009;45:2627–2633.
3. Ruhe A, Wedin P-Å. Algorithms for separable nonlinear least squares problems. SIAM Rev. 1980;22:318–337.
4. Björck Å. Numerical methods for least squares problems. Philadelphia, PA: SIAM; 1996.
5. Golub G, Pereyra V. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. SIAM J Numer Anal. 1973;10:413–432.
6. Ding F, Liu XP, Liu G. Identification methods for Hammerstein nonlinear systems. Digit Signal Process. 2011:21;215–238.
7. Johnson CR, Smigoc H, Yang D. Solution theory for systems of bilinear equations. Lin Multilin Alg. 2014;62:1553–1566.
8. Linder M, Sundberg R. Second order calibration: bilinear least squares regression and a simple alternative. Chemom Intell Lab Syst. 1998:159–178.
9. Hoff PD. Multilinear tensor regression for longitudinal relational data. Ann Appl Stat. 2015:1169–1193.
10. Lampos V, Preotiuc-Pietro D, Cohn T. A user-centric model of voting intention from social media. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics; 2013 Aug 4–9; Sofia, Bulgaria. Association for Computational Linguistics, Stroudsberg, PA; 2013. p. 993–1003.
11. Lampos V, Preotiuc-Pietro D, Samangooei S, Gelling D, Cohn T. Extracting socioeconomic patterns from the news: modelling text and outlet importance jointly. In: Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science 2014; 2014 Jun 26; Baltimore, MD. Association for Computational Linguistics, Stroudsberg, PA; 2014. p. 13–17.
12. Lock EF. Tensor-on-tensor regression. J Comput Graph Stat. 2017;27:638–647.
13. Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev. 2009;51:455–500.
14. Golub G, Loan CFV. Matrix computations. 4th ed. Baltimore, MD: Johns Hopkins Press; 2013.
15. Kaufman L. A variable projection method for solving separable nonlinear least squares problems. BIT Numer Math. 1975;15:49–57.
16. Powell MJD. On search directions for minimization algorithms. Math Program. 1973;4:193–201.
17. Bader B, Kolda T. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. ACM Trans Math Softw. 2006;32:635–653.
18. Wedin P-Å. On the Gauss-Newton method for the non-linear least squares problem. Stockolm, Sweden: Institutet för Tillämpad Matematik; 1974.
19. Hansen PC. Oblique projections and standard-form transformations for discrete inverse problems. Numer Linear Algebra Appl. 2013;20:250–258.
20. Szyld DB. The many proofs of an identity on the norm of oblique projections. Numer Algor. 2006;42:309–323.
21. Ramsin H, Wedin P-Å. A comparison of some algorithms for the nonlinear least squares problem. BIT Numer Math. 1977;17:72–90.
22. Li G, Wen C. Convergence of normalized iterative identification of Hammerstein systems. Syst Control Lett. 2011;60:929–935.
23. Westwick DT, Kearney RE. Separable least squares identification of nonlinear Hammerstein models: application to stretch reflex dynamics. Ann Biomed Eng. 2001;29:707–718.
24. Ding F, Liu X, Chu J. Gradient-based and least-squares-based iterative algorithms for Hammerstein systems using the hierarchical identification principle. IET Control Theory Appl. 2013;7:176–184.

How to cite this article: Eldén L, Ahmadi-Asl S. Solving bilinear tensor least squares problems and application to Hammerstein identification. *Numer Linear Algebra Appl.* 2018;e2226. https://doi.org/10.1002/nla.2226

APPENDIX A: DERIVATION OF THE PERTURBATION EQUATION 14

Recall that the perturbed quantity $x + \delta x$ is

$$x + \delta x = \begin{pmatrix} 1 \\ \bar{x} \end{pmatrix} + \begin{pmatrix} 0 \\ \delta\bar{x} \end{pmatrix}. \tag{A1}$$

Assume that the perturbations $\delta\mathcal{A}$ and δb satisfy (12). Consider first the perturbation of the residual,

$$\begin{aligned} r + \delta r &= (\mathcal{A} + \delta\mathcal{A}) \cdot (x + \delta x, y + \delta y)_{2,3} - b - \delta b \\ &= r + \mathcal{A} \cdot (\delta x, y)_{2,3} + \mathcal{A} \cdot (x, \delta y)_{2,3} + \delta\mathcal{A} \cdot (x, y)_{2,3} - \delta b + O(\epsilon^2). \end{aligned}$$

Ignoring $O(\epsilon^2)$ and using (A1), we can write

$$-\delta r + \bar{J}_{\bar{x}} \delta\bar{x} + J_y \delta y = \delta b - \delta\mathcal{A} \cdot (x, y)_{2,3},$$

which is the first equation in (14).

The perturbation of the first-order condition $r^T\bar{J} = 0$ can be conveniently written using the notation of Section 2,

$$(\bar{J} + \delta\bar{J}) \cdot (r + \delta r)_1 = \bar{J} \cdot (r)_1 + \delta\bar{J} \cdot (r)_1 + \bar{J} \cdot (\delta r)_1 + O(\epsilon^2) = 0,$$

which, using $\bar{J} \cdot (r)_1 = 0$ and ignoring $O(\epsilon^2)$, gives

$$\delta\bar{J} \cdot (r)_1 + \bar{J} \cdot (\delta r)_1 = 0. \tag{A2}$$

Using $\bar{J} = (\bar{\mathcal{A}} \cdot (\cdot, y)_{2,3} \;\; \mathcal{A} \cdot (x, \cdot)_{2,3})$ and inserting the perturbed quantities, we get

$$\delta\bar{J} = \left(\bar{\mathcal{A}} \cdot (\cdot, \delta y)_{2,3} \;\; \mathcal{A} \cdot (\delta x, \cdot)_{2,3} \right) + \left(\delta\bar{\mathcal{A}} \cdot (\cdot, y)_{2,3} \;\; \delta\mathcal{A} \cdot (x, \cdot)_{2,3} \right) + O(\epsilon^2),$$

and

$$\delta\bar{J} \cdot (r)_1 = \left(\bar{\mathcal{A}} \cdot (r, \cdot, \delta y) \;\; \mathcal{A} \cdot (r, \delta x, \cdot) \right) + \left(\delta\bar{\mathcal{A}} \cdot (r, \cdot, y) \;\; \delta\mathcal{A} \cdot (r, x, \cdot) \right) + O(\epsilon^2). \tag{A3}$$

Inserting $J \cdot (\delta r)_1 = (\bar{\mathcal{A}} \cdot (\delta r, \cdot, y) \;\; \mathcal{A} \cdot (\delta r, x, \cdot))$ and (A3) in (A2) and ignoring $O(\epsilon^2)$, we get two equations,

$$\begin{aligned} \bar{\mathcal{A}} \cdot (\delta r, y)_{1,3} + \bar{\mathcal{A}} \cdot (r, \delta y)_{1,3} &= -\delta\bar{\mathcal{A}} \cdot (r, y)_{1,3} \\ \mathcal{A} \cdot (\delta r, x)_{1,2} + \mathcal{A} \cdot (r, \delta x)_{1,2} &= -\delta\mathcal{A} \cdot (r, x)_{1,2}. \end{aligned}$$

From (A1), we have $\mathcal{A} \cdot (r, \delta x)_{1,2} = \bar{\mathcal{A}} \cdot (r, \delta\bar{x})_{1,2}$. Thus, translating the tensor expressions to matrix form, we have derived the last two equations in (14).

APPENDIX B: PROOF OF PROPOSITION 1

Proof. We put

$$X = \begin{pmatrix} R_x^{-1} Q_x^T P_x \\ R_y^{-1} Q_y^T P_y \end{pmatrix},$$

and verify that the four identities uniquely defining the Moore–Penrose pseudoinverse (see, e.g., section 5.5.2 of the work of Golub et al.[14]),

$$\begin{aligned} &(i) \quad \bar{J}X\bar{J} = \bar{J}, && (ii) \quad X\bar{J}X = X, \\ &(iii) \quad \bar{J}X = (\bar{J}X)^T, && (iv) \quad X\bar{J} = (X\bar{J})^T, \end{aligned}$$

are satisfied. Clearly, $P_xQ_x = Q_x$ and $P_xQ_y = 0$, as well as $P_yQ_y = Q_y$ and $P_yQ_x = 0$. Therefore,

$$X\bar{J} = \begin{pmatrix} R_x^{-1} Q_x^T P_x \\ R_y^{-1} Q_y^T P_y \end{pmatrix} (Q_x R_x \;\; Q_y R_y) = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}.$$

Thus, the identities (i), (ii), and (iv) hold. Then, we have

$$\bar{J}X = Q_x Q_x^T P_x + Q_y Q_y^T P_y.$$

Performing the multiplications using (32), we obtain some terms that are symmetric by inspection and the following two terms:

$$-(T_x + T_y) := Q_x F (I - F^T F)^{-1} Q_y^T + Q_y F^T (I - F F^T)^{-1} Q_x^T,$$

where $F = Q_x^T Q_y$. It is straightforward, using the SVD of F, to show that $(F(I - F^TF)^{-1})^T = F^T(I - FF^T)^{-1}$, and therefore, $T_x^T = T_y$, which implies that $T_x + T_y$ is symmetric. This shows that (iii) holds, and the proposition is proved. □