Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Newton-type Method for the Orthonormal INDSCAL Problem in Metric Multidimensional Scaling

Jiao-fen Li
Guilin University of Electronic Technology
Rui-li Jiang
Guilin University of Electronic Technology
Rui-juan Jiao
Guilin University of Electronic Technology
Xue-lin Zhou
zhouxuelin0309@163.com
Guilin University of Electronic Technology
Ya-qiong Wen
Guilin University of Electronic Technology

Research Article

Keywords: Multidimensional scaling, Individual differences scaling, Riemannian optimization, Newton's method

Posted Date: March 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8746904/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Newton-type Method for the Orthonormal INDSCAL Problem in Metric Multidimensional Scaling

Jiao-fen Li · Rui-li Jiang · Rui-juan Jiao · Xue-lin Zhou · Ya-qiong Wen

Received: date / Accepted: date

Abstract The orthonormal individual differences scaling (O-INDSCAL) model is a fundamental tool for metric multidimensional scaling of multiple doubly centered dissimilarity matrices, but efficient algorithms remain scarce when orthonormality and nonnegativity constraints are imposed in medium- to large-scale settings. In this work,

Research supported by the National Natural Science Foundation of China (12261026, 12361079, 12201149), the Natural Science Foundation of Guangxi (2023GXNSFAA026067, 2024GXNSFAA010521), the Guangxi Key Laboratory of Automatic Detecting Technology and Instruments (YQ23103, YQ24105) and the Science and Technology Project of Guangxi (Guike AD25069086).

Jiao-fen Li
1. School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, 541004, China
2. Center for Applied Mathematics of Guangxi (GUET), Guilin, 541004, China
E-mail: lixiaogui1290@163.com

Rui-li Jiang
School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, 541004, China

Rui-juan Jiao
School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, 541004, China

Xue-lin Zhou
1. School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, 541004, China
2. School of Mathematics and Statistics, Yunan University, Kunming 650000, China
E-mail: zhouxuelin0309@163.com (Corresponding author)

Ya-qiong Wen
1. School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, 541004, China
2. Center for Applied Mathematics of Guangxi (GUET), Guilin, 541004, China
E-mail: pmxdgmcdsy@163.com (Corresponding author)

we revisit O-INDSCAL from a Riemannian optimization perspective and make two main contributions. First, starting from explicit closed-form expressions of the Riemannian gradient and Hessian, we represent the Hessian as a Kronecker-product-based linear operator on the tangent space and reformulate the Riemannian Newton equation as a symmetric linear system whose dimension coincides with that of the underlying product manifold. This dimension-reduced formulation enables the use of Krylov subspace methods and substantially lowers the per-iteration cost of Newton steps while preserving second-order accuracy. Second, we design a hybrid algorithm that couples a globally convergent Riemannian curvilinear search with Barzilai–Borwein step sizes and a locally quadratically convergent Riemannian Newton refinement, thereby combining robust globalization with fast local convergence. Extensive numerical experiments on synthetic and real O-INDSCAL benchmarks show that the proposed method attains high-accuracy solutions and compares favorably with projected gradient flows, ALS/MPE-type schemes, and generic Manopt-based Riemannian solvers in terms of both efficiency and reliability. MATLAB implementations of all methods used in the numerical experiments are available at `Hybrid_Newton_for_OINDSCAL_MATLAB_Codes`.

Keywords Multidimensional scaling · Individual differences scaling · Riemannian optimization · Newton's method

Mathematics Subject Classification (2000) 62-08 · 62R30 · 65F10 · 65H10

1 Introduction

Multidimensional scaling (MDS) is a well-established technique in numerical analysis and data science, aimed at recovering a configuration of n points in a Euclidean space of reduced dimension r from information that quantifies pairwise relationships among the objects [9, 5, 6]. This information is typically encoded through measures of similarity or dissimilarity (denoted s/d) between object pairs. In metric MDS, the goal is to compute an $n \times r$ coordinate matrix X such that the inner product matrix XX^T closely approximates the observed dissimilarity structure. The resulting matrix X enables low-dimensional visualization while preserving as much of the original relational structure as possible. The approach hinges on the principle that Euclidean distances between embedded points can be expressed in terms of scalar products.

A widely adopted extension of MDS to multi-subject or multi-condition data is the INdividual Differences SCALing (INDSCAL) model, originally introduced by Carroll and Chang [10] (see also [15, 29]). INDSCAL operates on a three-way array consisting of m symmetric $n \times n$ similarity or dissimilarity matrices, commonly referred to as slices S_i. The key modeling assumption is that all slices share a common Euclidean space of dimension r, while allowing the relative importance (or salience) of dimensions to vary across slices. The model approximates each slice via the decomposition:

$$S_i = Q\tilde{D}_i Q^T + E_i, \quad i = 1, \ldots, m, \tag{1}$$

where $Q \in \mathbb{R}^{n\times r}$ is a common loading matrix, $\tilde{D}_i \in \mathcal{D}(r)$ is a diagonal matrix representing the dimension-specific weights for the i-th slice, and $E_i \in \mathbb{R}^{n\times n}$ denotes the residual error. This structure implies that the geometric configuration Q is fixed across all conditions or subjects, while the salience matrices D_i encode idiosyncratic variations. When the input matrices S_i are correlation or covariance matrices, the INDSCAL model can be interpreted as a form of simultaneous principal component analysis (PCA)

or, more broadly, as the simultaneous diagonalization of multiple symmetric matrices. The standard formulation seeks to estimate $(Q, D_1, \ldots, D_m)$ by minimizing the total squared reconstruction error:

$$f(Q, \tilde{D}_1, \ldots, \tilde{D}_m) = \sum_{i=1}^{m} \|S_i - Q\tilde{D}_i Q^T\|^2, \tag{2}$$

where $\|\cdot\|$ denotes the Frobenius norm. This formulation leads to an unconstrained nonlinear optimization problem, which is typically addressed via alternating least squares (ALS), most notably through the CANDECOMP algorithm [10]. However, the convergence of this algorithm remains theoretically unproven.

A practically important variant of INDSCAL imposes nonnegativity constraints on the salience matrices $\tilde{D}_i$. This constraint is motivated by interpretability: the factorized representation $Q\tilde{D}_i^{1/2}$ can be used to visualize individual-specific spaces in a meaningful way. However, the CANDECOMP approach does not enforce nonnegativity, potentially compromising interpretability in certain applications. To address this issue, a series of studies [26, 27, 22, 29, 33] have reformulated the INDSCAL problem (2) with nonnegative saliences into a constrained optimization framework by reparameterizing $\tilde{D}_i$ as D_i^2, and enforcing orthonormality on Q. This yields the following equality-constrained formulation:

$$\begin{aligned} \text{minimize} \quad & f(Q, D_1, \ldots, D_m) = \sum_{i=1}^{m} \|S_i - QD_i^2 Q^T\|^2, \\ \text{subject to} \quad & (Q, D_1, \ldots, D_m) \in \mathcal{O}(n, r) \times \mathcal{D}(r)^m, \end{aligned} \tag{3}$$

where $\mathcal{O}(n, r) = \{Q \in \mathbb{R}^{n \times r} : Q^T Q = I_r\}$ denotes the Stiefel manifold of orthonormal matrices, and $\mathcal{D}(r)$ is the set of all $r \times r$ diagonal matrices. The product space $\mathcal{D}(r)^m$ denotes m independent copies of this diagonal matrix set.

Analytical solutions to the problem (3) are generally intractable because the decision variables are strongly coupled through the objective function, making the development of efficient numerical algorithms essential. Despite the importance of O-INDSCAL in multivariate analysis, dedicated optimization methods remain relatively scarce. Early work by Trendafilov [24, 27] proposed a continuous-time projected gradient-flow framework for (3), in which the gradient dynamics of the objective are projected onto the constraint manifold and then discretized to obtain a steepest-descent-type scheme. This approach is conceptually simple and has been extended to related models [24, 25, 27, 30], but, as a first-order method with essentially linear convergence, it typically requires a large number of iterations to reach high-accuracy solutions, especially for medium- and large-scale instances. To improve practical efficiency, Loisel and Takane [17] introduced an approximate alternating least-squares (ALS) procedure reformulated as a fixed-point iteration and further accelerated it by minimal polynomial extrapolation (MPE). The resulting ALS/MPE scheme shows improved empirical performance over the gradient-flow-based method on certain synthetic symmetric datasets. However, its effectiveness depends on a heuristic choice of the extrapolation degree, and its convergence properties remain unestablished in the general nonlinear and constrained O-INDSCAL setting. In parallel, Trendafilov and Gallo [29] emphasized the usefulness of matrix manifolds for formulating multivariate data-analysis problems and briefly surveyed manifold-based optimization methods implemented in the MATLAB toolbox

Manopt [7]. These generic first- and second-order solvers provide a convenient computational platform, but they are not tailored to the specific algebraic and geometric structure of the O-INDSCAL model and may incur avoidable computational overhead, particularly when high accuracy or larger problem sizes are required. Building on this line of work, Zhou et al. [33] developed a Riemannian inexact Newton method specifically for O-INDSCAL, in which the Newton equation is solved approximately by a conjugate-gradient procedure subject to a prescribed inexactness criterion, and the negative Riemannian gradient is used as a fallback direction when the inexactness condition is not satisfied. The method enjoys global convergence and local quadratic convergence under standard assumptions. Nonetheless, numerical evidence indicates that its practical performance is quite sensitive to the choice of the initial iterate: if the starting point is not sufficiently close to a solution, the inexactness condition may fail repeatedly, causing the algorithm to revert to gradient steps for many iterations and thus significantly slowing convergence. More recently, Zhou and Li [32] proposed Riemannian trust-region algorithms for the direct INDSCAL formulation posed on the product manifold $\mathcal{O}_0(n,r) \times \mathbf{D}(r)^m$, where $\mathcal{O}_0(n,r)$ denotes the set of $n \times r$ orthonormal matrices with zero column sums. Although the direct INDSCAL model and its associated manifold differ from the O-INDSCAL setting considered here, the modified trust-region techniques in [32] can, in principle, be adapted to O-INDSCAL. However, trust-region methods generally require the solution of a trust-region subproblem at each iteration, which increases per-iteration cost and can limit scalability. Taken together, these limitations motivate the development of structure-exploiting algorithms that achieve robust globalization and fast local convergence while maintaining low per-iteration complexity for medium- and large-scale O-INDSCAL problems.

Motivated by these developments, we revisit the O-INDSCAL problem (3) from the perspective of Riemannian Newton methods and exploit its specific algebraic structure to design an efficient solver. Starting from explicit expressions for the Riemannian gradient and Hessian, and leveraging recent advances in the literature [3, 19, 16], we first characterize the Hessian as a linear operator on the tangent space of the product manifold and derive a block-structured representation of this operator. By systematically employing Kronecker-product and vectorization techniques, we then dimensionally reduce the Riemannian Newton equation and reformulate it as a standard symmetric linear system $H_\phi x = g$, whose size coincides with the intrinsic dimension of the manifold. This leads to a Newton step that can be computed efficiently by Krylov subspace methods, such as the conjugate residual (CR) algorithm, without forming or storing large dense matrices. To enhance robustness and combine desirable global and local convergence properties, we further propose a hybrid scheme that couples a globally convergent first-order phase with a locally fast Riemannian Newton refinement. In the first phase, a curvilinear search method with Barzilai–Borwein step sizes is employed to generate a high-quality initial point; this scheme is known to be globally convergent with a linear rate [31, 14]. In the second phase, starting from this refined iterate, the Riemannian Newton method is activated to exploit its quadratic local convergence near a solution. Numerical results show that this two-stage strategy achieves a favorable balance between stability and efficiency and provides an effective solver for the O-INDSCAL problem.

Although the O-INDSCAL model on $\mathcal{O}(n,r)\times\mathbf{D}(r)^m$ and its Riemannian geometry are well documented in the literature [e.g. 26, 27, 29, 33], the present work focuses on exploiting this structure to obtain a practically efficient Riemannian Newton framework

with explicit dimensional reduction and a tailored globalization mechanism. The main contributions can be summarized as follows:

1) We derive a dimension-reduced formulation of the Riemannian Newton equation by representing the Hessian as a Kronecker-product-based linear operator on the tangent space. This leads to a symmetric linear system $H_\phi x = g$ whose dimension matches that of the product manifold, yielding substantial savings in storage and computational cost per Newton step and enabling the use of efficient Krylov subspace solvers.
2) We design a hybrid algorithm that integrates a globally convergent curvilinear search with Barzilai–Borwein steps and a locally quadratically convergent Riemannian Newton method. This strategy combines reliable globalization with fast local convergence, and is particularly effective for medium- and large-scale O-INDSCAL instances.
3) We provide a comprehensive numerical study on synthetic and real datasets, demonstrating that the proposed framework attains high-accuracy solutions with competitive or superior runtimes compared to existing gradient-flow, trust-region, and generic manifold-optimization approaches, thereby confirming the practical benefits of the proposed structure-exploiting Riemannian Newton scheme.

The remainder of the paper is structured as follows. Section 2 presents the geometric foundations of the product manifold and provides explicit matrix expressions for the Riemannian gradient and Hessian. In Section 3, we introduce the hybrid algorithm, detailing both the curvilinear search phase and the Newton refinement phase. Section 4 reports numerical results that demonstrate the effectiveness of our approach, including comparisons with existing methods. Finally, Section 5 offers concluding remarks.

2 Geometry of embedded matrix submanifolds

In this section we first recall some notation, definitions and basic properties of Riemannian manifolds used throughout the paper. The tangent space at point x on a manifold $\mathcal{M}$ is denoted $\mathbf{T}_x\mathcal{M}$. For mappings $f : \mathcal{M} \to \mathcal{N}$ between manifolds $\mathcal{M}$ and $\mathcal{N}$, the differential of f at $x \in \mathcal{M}$ is denoted $\mathfrak{D}f(x)$, mapping from $\mathbf{T}_x\mathcal{M}$ to $\mathbf{T}_{f(x)}\mathcal{N}$. For a smooth function f on a manifold $\mathcal{M} \subset \mathbb{R}^{m\times p}$, $\bar{f}$ extends f to the ambient Euclidean space $\mathbb{R}^{m\times p}$. ∇ and grad represent the Euclidean and Riemannian gradients, respectively; hence, for f on $\mathcal{M} \subset \mathbb{R}^{m\times p}$, ∇ and grad operate on $\bar{f}$ and f, respectively. Hess denotes the Riemannian Hessian.

We next introduce some basic geometric properties of the involved product manifold $\mathcal{O}(n,r)\times\mathcal{D}(r)^m$, the reader is referred to [1, 29] for more details of the Stiefel manifold $\mathcal{O}(n,r)$. For convenience of expression, we define the product manifold $\mathcal{M}$ as follows:

$$\mathcal{M} := \mathcal{O}(n,r) \times \mathcal{D}(r)^m, \quad Z := (Q, D_1, \ldots, D_m) \in \mathcal{M},$$

and the corresponding tangent vector as:

$$\Delta Z := (\xi, \eta_1, \ldots, \eta_m) \in \mathbf{T}_Z\mathcal{M},$$

where $\mathbf{T}_Z\mathcal{M}$ denotes the tangent space of $\mathcal{M}$ at $Z \in \mathcal{M}$. The dimensions of the Stiefel manifold $\mathcal{O}(n,r)$, and the linear manifolds $\mathcal{D}(r)$ are given by $\dim \mathcal{O}(n,r) =$

$nr - \frac{1}{2}r(r+1)$ and $\dim \mathcal{D}(r) = r$, respectively. Thus, the dimension of the product manifold $\mathcal{M}$ is:

$$\dim(\mathcal{M}) = \dim(\mathcal{O}(n,r)) + m\dim(\mathcal{D}(r)) = (m+n)r - \frac{1}{2}r(r+1). \tag{4}$$

The tangent spaces at points $Q \in \mathcal{O}(n,r)$ and $D \in \mathcal{D}(r)$ are given by [1] as:

$$\mathbf{T}_Q\mathcal{O}(n,r) = \{P \in \mathbb{R}^{n\times r} \mid P^TQ + Q^TP = 0\}, \quad \mathbf{T}_D\mathcal{D}(r) = \mathcal{D}(r).$$

Therefore, the tangent space of the product manifold $\mathcal{M}$ at $Z \in \mathcal{M}$ is:

$$\mathbf{T}_Z\mathcal{M} \cong \mathbf{T}_Q\mathcal{O}(n,r) \times \mathcal{D}(r)^m,$$

where $\cong$ denotes the identification of the tangent spaces.

Let the Euclidean space $\mathbb{R}^{n\times r} \times (\mathbb{R}^{r\times r})^m$ be equipped with the natural inner product:

$$\langle M, \tilde{M}\rangle = \mathrm{tr}(M_q^T\tilde{M}_q) + \sum_{i=1}^{m}\mathrm{tr}(M_{d,i}^T\tilde{M}_{d,i}),$$

for any $M = (M_q, M_{d,1}, \ldots, M_{d,m})$ and $\tilde{M} = (\tilde{M}_q, \tilde{M}_{d,1}, \ldots, \tilde{M}_{d,m}) \in \mathbb{M}$. This inner product induces a norm $\|\cdot\|$, where $\mathrm{tr}(A)$ denotes the trace of a square matrix A. Since $\mathcal{M}$ is an embedded submanifold of $\mathbb{M}$, it inherits the following induced Riemannian metric:

$$\langle \Delta Z, \Delta\tilde{Z}\rangle_Z = \mathrm{tr}(\xi^T\tilde{\xi}) + \sum_{i=1}^{m}\mathrm{tr}(\eta_i^T\tilde{\eta}_i), \tag{5}$$

where $\Delta Z, \Delta\tilde{Z} := (\tilde{\xi}, \tilde{\eta}_1, \ldots, \tilde{\eta}_m) \in \mathbf{T}_Z\mathcal{M}$. The corresponding norm is $\|\Delta Z\|_Z^2 = \|\xi\|^2 + \sum_{i=1}^m \|\eta_i\|^2$. For simplicity, we use $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ to denote the Riemannian metric on $\mathcal{M}$ and its induced norm, respectively.

Next, we define the orthogonal projection map onto $\mathcal{M}$. The orthogonal projections of $\Xi \in \mathbb{R}^{n\times r}$ and $H_i \in \mathbb{R}^{r\times r}$ onto $\mathbf{T}_Q\mathcal{O}(n,r)$and $\mathbf{T}_D\mathcal{D}(r)$, respectively, are given by [1] as:

$$\mathbf{P}_Q(\Xi) = \Xi - Q\,\mathrm{sym}(Q^T\Xi), \quad \mathbf{P}_D(H_i) = I_r \odot H_i, \tag{6}$$

where $\odot$ denotes the element-wise (Hadamard) matrix product, $\mathrm{sym}(A) = \frac{1}{2}(A + A^T)$ represents the symmetric part of a matrix A, respectively. Therefore, the orthogonal projection of any point $(\Xi, H_1, \ldots, H_m) \in \mathbb{R}^{n\times r} \times (\mathbb{R}^{r\times r})^m$ onto $\mathbf{T}_Z\mathcal{M}$ is:

$$\mathbf{P}_Z(\Xi, H_1, \ldots, H_m) \cong \left(\mathbf{P}_Q(\Xi), \mathbf{P}_D(H_1), \ldots, \mathbf{P}_D(H_m)\right). \tag{7}$$

The retraction on $\mathcal{O}(n,r)$ [1] via QR decomposition is given by:

$$\mathcal{R}_Q(\xi) = \mathrm{qf}(Q + \xi), \quad \forall Q \in \mathcal{O}(n,r),\ \xi \in \mathbf{T}_Q\mathcal{O}(n,r),$$

where $\mathrm{qf}(A)$ represents the Q factor in the economy QR decomposition of $A \in \mathbb{R}^{n\times r}$. The retractions on the linear manifolds $\mathcal{D}(r)$ is defined as $\mathcal{R}_D(\eta) = D + \eta$. Thus, the retraction $\mathcal{R}$ on the product manifold $\mathcal{M}$ takes the form:

$$\mathcal{R}_Z(\Delta Z) \cong \left(\mathcal{R}_Q(\xi), \mathcal{R}_D(\eta_1), \ldots, \mathcal{R}_D(\eta_m)\right). \tag{8}$$

We next derive the representation matrix formulas of Riemannian gradient and Hessian of the objective function in the problem (3). The Riemannian gradient, denoted by grad $f(Z)$, is defined at a point $Z \in \mathcal{M}$ to satisfy the relation:

$$\langle \text{grad } f(Z), \Delta Z \rangle = \mathfrak{D} f(Z)[\Delta Z], \quad \Delta Z \in \mathbf{T}_Z \mathcal{M},$$

where $\mathfrak{D} f(Z)[\Delta Z]$ is the directional derivative of f at Z in the direction of ΔZ. The Riemannian Hessian, denoted by Hess $f(Z)$, is a linear transformation of the tangent space $\mathbf{T}_Z \mathcal{M}$, defined through the covariant derivative $\nabla_{\Delta Z} \text{grad } f$ of the gradient grad f at Z:

$$\text{Hess } f(Z)[\Delta Z] := \nabla_{\Delta Z} \text{grad } f, \quad \Delta Z \in \mathbf{T}_Z \mathcal{M},$$

where the covariant derivative is constructed using the Levi-Civita connection ∇ on $\mathcal{M}$ [1].

In what follows, let f be the objective function of the problem (3), *i.e.*,

$$f(Z) = \frac{1}{4} \sum_{i=1}^{m} \left\| S_i - Q D_i^2 Q^T \right\|^2, \quad Z \in \mathcal{M},$$

where the scaling factor $\frac{1}{4}$ is introduced to simplify the expressions for the gradient and Hessian. Let $\bar{f}$ be a function with the same form as f defined in $\mathbb{R}^{n \times r} \times (\mathbb{R}^{r \times r})^m$; that is,

$$\bar{f}(Z) = \frac{1}{4} \sum_{i=1}^{m} \left\| S_i - Q D_i^2 Q^T \right\|^2, \quad Z \in \mathbb{R}^{n \times r} \times (\mathbb{R}^{r \times r})^m.$$

Thus, the objective function $f(Z)$ are the restrictions of $\bar{f}$ onto $\mathcal{M}$, *i.e.*, $f = \bar{f}|_{\mathcal{M}}$. Let $Z \in \mathcal{M}$, expression for the Riemannian gradient is given in [1, 29] as

$$\text{grad } f(Z) = \mathbf{P}_Z \big(\nabla \bar{f}(Z) \big), \quad Z \in \mathcal{M}. \tag{9}$$

where $\nabla \bar{f}(Z)$ is the Euclidean gradient of $\bar{f}$ on $\mathbb{R}^{n \times r} \times (\mathbb{R}^{r \times r})^m$, which is computed as [33]

$$\nabla \bar{f}(Z) := \big(\nabla_Q \bar{f}(Z),\ \nabla_{D_1} \bar{f}(Z),\ \ldots,\ \nabla_{D_m} \bar{f}(Z) \big)$$

with

$$\nabla_Q \bar{f}(Z) = -\sum_{i=1}^{m} F_i Q D_i^2, \quad \nabla_{D_i} \bar{f}(Z) = -D_i Q^T F_i Q, \qquad i = 1, \ldots, m, \tag{10}$$

where

$$F_i = S_i - Q D_i^2 Q^T \qquad i = 1, \ldots, m. \tag{11}$$

Consequently, by employing the projection $\mathbf{P}_Z$ outlined in (6)-(7), we derive the Riemannian gradient of f at $Z = (Q, D_1, \ldots, D_m) \in \mathcal{M}$ as

$$\text{grad } f(Z) = -\left\{ \mathbf{P}_Q \left(\sum_{i=1}^{m} \left(S_i Q D_i^2 - Q D_i^4 \right) \right), D_1 \odot \left(Q^T S_1 Q - D_1 \right), \ldots, D_m \odot \left(Q^T S_m Q - D_m^2 \right) \right\}. \tag{12}$$

Let $\Delta Z = (\xi, \eta_1, \ldots, \eta_m)$ be a tangent vector at $Z \in \mathcal{M}$, by choosing the covariant derivative $\nabla_{\Delta Z}\zeta := \mathbf{P}_Z(\mathfrak{D}\zeta(Z)[\Delta Z])$ (§5.3.3 of [2]), expression for the Riemannian Hessian is given as

$$\begin{aligned}\operatorname{Hess} f(Z)[\Delta Z] &= \mathbf{P}_Z\Big(\mathfrak{D}(\operatorname{grad} f)(Z)[\Delta Z]\Big)\\ &= \mathbf{P}_Z\Big(\mathfrak{D}(\mathbf{P}_Z(\nabla \bar{f}))(Z)[\Delta Z]\Big)\\ &= \mathbf{P}_Z\Big(\mathfrak{D}\mathbf{P}_Z[\Delta Z](\nabla \bar{f}(Z)) + \mathfrak{D}(\nabla \bar{f})(Z)[\Delta Z]\Big),\end{aligned} \tag{13}$$

where the Frechét derivative $\mathfrak{D}(\nabla \bar{f})(Z)[\Delta Z]$ is written as [33]

$$\begin{aligned}\mathfrak{D}(\nabla \bar{f})(Z)[\Delta Z] &= \lim_{t\to 0}\frac{1}{t}\Big(\nabla \bar{f}(Q + t\xi, D_1 + t\eta_1, \ldots, D_m + t\eta_m) - \nabla \bar{f}(Q, D_1, \ldots, D_m)\Big)\\ &= -\Big(\sum_{i=1}^{m}\big(F_i\xi D_i^2 - W_iQD_i^2 + 2F_iQD_i\eta_i\big),\\ &\qquad D_1Q^TF_1\xi - D_1Q^TW_1Q + D_1\xi^TF_1Q + \eta_1Q^TF_1Q,\\ &\qquad\qquad\vdots\\ &\qquad D_mQ^TF_m\xi - D_mQ^TW_mQ + D_m\xi^TF_mQ + \eta_mQ^TF_mQ\Big),\end{aligned} \tag{14}$$

with F_i being given in (11) and

$$W_i = 2QD_i\eta_iQ^T + QD_i^2\xi^T + \xi D_i^2Q^T, \qquad i = 1, \ldots, m. \tag{15}$$

On the other hand,

$$\begin{aligned}&\mathbf{P}_Z\Big(\mathfrak{D}\mathbf{P}_Z[\Delta Z](\nabla \bar{f}(Z))\Big)\\ &= \Big(\mathbf{P}_Q\big(\mathfrak{D}\mathbf{P}_Q[\xi](\nabla_Q \bar{f}(Z))\big),\ \mathbf{P}_D\big(\mathfrak{D}\mathbf{P}_D[\eta_1](\nabla_{D_1} \bar{f}(Z))\big),\ \ldots,\ \mathbf{P}_D\big(\mathfrak{D}\mathbf{P}_D[\eta_m](\nabla_{D_m} \bar{f}(Z))\big)\Big)\\ &= \Big(-\mathbf{P}_Q\Big(\xi\operatorname{sym}(Q^\top\nabla_Q \bar{f}(Z))\Big),\ 0,\ \ldots,\ 0\Big).\end{aligned} \tag{16}$$

Consequently, by using (13) and (16), we obtain a more concrete expression for $\operatorname{Hess} f(Z)$ acts on ΔZ as

$$\operatorname{Hess} f(Z)[\Delta Z] = \mathbf{P}_Z\Big(\mathfrak{D}(\nabla \bar{f})(Z)[\Delta Z] - \Big(\xi\operatorname{sym}(Q^T\nabla_Q \bar{f}(Z)),\ 0,\ \ldots,\ 0\Big)\Big). \tag{17}$$

where $\nabla_Q \bar{f}(Z)$ be defined in (10). By substituting (14) into (17), together with (11), (15) and the projection $\mathbf{P}_Z$ given in (6)-(7), we obtain the Riemannnian Hessian of f at $Z = (Q, D_1, \ldots, D_m) \in \mathcal{M}$ acts on $\Delta Z = (\xi, \eta_1, \ldots, \eta_m) \in \mathbf{T}_Z\mathcal{M}$ can be expressed by

$$\begin{aligned}\operatorname{Hess} f(Z)[\Delta Z] = -\Big(&\mathbf{P}_Q\Big(\sum_{i=1}^{m}\big(F_i\xi D_i^2 - W_iQD_i^2 + 2F_iQD_i\eta_i\big) + \xi\operatorname{sym}(Q^T\nabla_Q \bar{f}(Z))\Big),\\ &D_1 \odot \Big(Q^TF_1\xi + \xi^TF_1Q - Q^TW_1Q\Big) + \eta_1 \odot \Big(Q^TF_1Q\Big),\\ &\qquad\vdots\\ &D_m \odot \Big(Q^TF_m\xi + \xi^TF_mQ - Q^TW_mQ\Big) + \eta_m \odot \Big(Q^TF_mQ\Big)\Big)\end{aligned} \tag{18}$$

with F_i and W_i being given in (11) and (15), respectively, for $i = 1, \ldots, m$.

3 A hybrid Newton scheme for the problem (3)

Having derived the matrix expressions for grad f and Hess f, along with other necessary components for Riemannian optimization algorithms, we now develop the Riemannian Newton's method for solving the problem (3). In this method, the search direction $\Delta Z^{(k)} \in \mathbf{T}_{Z^{(k)}}\mathcal{M}$ at $Z^{(k)} \in \mathcal{M}$ is determined as the solution to the Newton's equation:

$$\operatorname{Hess} f(Z^{(k)})[\Delta Z^{(k)}] = -\operatorname{grad} f(Z^{(k)}). \tag{19}$$

Once $\Delta Z^{(k)}$ is obtained, the next iterate is computed as

$$Z^{(k+1)} = \mathcal{R}_{Z^{(k)}}(\Delta Z^{(k)}), \tag{20}$$

where, for simplicity, the step size is set to one. Substituting (12) and (18) into (19), the complete Riemannian Newton's equation for the problem (3) at $Z \in \mathcal{M}$ takes the form

$$\begin{cases} -\mathbf{P}_Q \left(\displaystyle\sum_{i=1}^{m} \left(F_i \xi D_i^2 - W_i Q D_i^2 + 2F_i Q D_i \eta_i\right) + \xi \operatorname{sym}\left(Q^T \nabla_Q \bar{f}(Z)\right) \right) \\ \qquad = \mathbf{P}_Q \left(\displaystyle\sum_{i=1}^{m} \left(S_i Q D_i^2 - Q D_i^4\right) \right), \\ -D_i \odot \left(Q^T F_i \xi + \xi^T F_i Q - Q^T W_i Q\right) - \eta_i \odot \left(Q^T F_i Q\right) \\ \qquad = D_i \odot \left(Q^T S_i Q - D_i^2\right), \quad i = 1, \ldots, m, \end{cases} \tag{21}$$

with $\nabla_Q \bar{f}(Z)$, F_i and W_i being given in (10), (11) and (15), respectively, for $i = 1, \ldots, m$.

The key problem now lies in solving the above Newton's equation. This system is inherently complex, as the solution $\Delta Z = (\xi, \eta_1, \ldots, \eta_m) \in \mathbf{T}_Z\mathcal{M}$ must satisfy multiple constraints for a given $Z \in \mathcal{M}$, namely, $\xi^T Q + Q^T \xi = 0$ and $\eta_i \in \mathcal{D}(r)$ for $i = 1, \ldots, m$.

3.1 Alternative representation of the Hessian

To address these problems, our goal is to represent the Riemannian Hessian Hess f as a matrix that characterizes a linear transformation on the tangent space $\mathbf{T}_Z\mathcal{M}$ at an arbitrary point $Z \in \mathcal{M}$. This matrix representation enables us to reformulate the Newton's equation (21) as a standard linear system, which can be efficiently solved using established numerical techniques. Toward this end, we begin with the following lemma.

Lemma 1 *[1] An equivalent form of the tangent space to $\mathcal{O}(n, r)$ at $Q \in \mathcal{O}(n, r)$ is given by*

$$\mathbf{T}_Q\mathcal{O}(n, r) = \left\{ QM + Q_\perp N \,\middle|\, M \in \mathcal{K}(r),\ N \in \mathbb{R}^{(n-r)\times r} \right\}, \tag{22}$$

where $Q_\perp$ is an n-by-$(n-r)$ orthogonal matrix such that $QQ^T + Q_\perp Q_\perp^T = I_n$, and $\mathcal{K}(r)$ is the set of all $r \times r$ skew-symmetric matrices.

From Lemma 1, one can easily to check that the vector space $\mathbf{T}_Z\mathcal{M}$ of all tangent vectors has a dimension of $\dim(\mathbf{T}_Z\mathcal{M}) = \dim(\mathcal{K}(r))+\dim(\mathbb{R}^{(n-r)\times r})+m\dim(\mathcal{D}(r)) = (m+n)r - \frac{1}{2}r(r+1)$, which matches the exact dimension of the product manifold $\mathcal{M}$ in (4). Let $\Delta Z = (\xi, \eta_1, \ldots, \eta_m) \in \mathbf{T}_Z\mathcal{M}$, where $\xi \in \mathbf{T}_Q\mathcal{O}(n,r)$ and $\eta_i \in \mathcal{D}(r)$, from (22), ΔZ can be expressed by

$$\Delta Z = (QM + Q_\perp N, \eta_1, \ldots, \eta_m), \tag{23}$$

where $M \in \mathcal{K}(r)$, $N \in \mathbb{R}^{(n-r)\times r}$. Since $\mathrm{Hess}\, f(Z)[\Delta Z]$ is also determined to be a unique tangent vector in $\mathbf{T}_Z\mathcal{M}$, there exist matrices $M_H \in \mathcal{K}(r)$, $N_H \in \mathbb{R}^{(n-r)\times r}$ and $\Xi_1, \ldots, \Xi_m \in \mathcal{D}(r)$ which satisfy

$$\mathrm{Hess}\, f(Z)[\Delta Z] = (QM_H + Q_\perp N_H,\ \Xi_1, \ldots, \Xi_m). \tag{24}$$

The subsequent lemma demonstrates that we can represent $M_H, N_H, \Xi_1, \ldots, \Xi_m$ in terms of $M, N, \eta_1, \ldots, \eta_n$, correspondingly.

Lemma 2 *Let $Z = (Q, D_1, D_2, \ldots, D_m) \in \mathcal{M}$ with $Q \in \mathcal{O}(n,r)$, $Q_\perp \in \mathcal{O}(n, n-r)$ satisfy $Q^TQ_\perp = 0$. If a tangent vector $\Delta Z \in \mathbf{T}_Z\mathcal{M}$ is expressed as (23), then the Hessian $\mathrm{Hess}\, f(X)$ of the objective function f acts on ΔZ as in (24) with*

$$\begin{aligned}
M_H &= -\mathrm{skew}\left(\sum_{i=1}^m \left(S_i^{(1)}MD_i^2 - MD_i^4\right) + M\mathrm{sym}(Q^T\nabla_Q\bar{f}(Z)) + \sum_{i=1}^m S_i^{(2)}ND_i^2 + \sum_{i=1}^m \left(2S_i^{(1)}D_i - 4D_i^3\right)\eta_i\right),\\
N_H &= -\left(\sum_{i=1}^m {S_i^{(2)}}^T MD_i^2 + \sum_{i=1}^m \left(S_i^{(3)}ND_i^2 - ND_i^4\right) + N\mathrm{sym}(Q^T\nabla_Q\bar{f}(Z)) + \sum_{i=1}^m 2{S_i^{(2)}}^T D_i\eta_i\right),\\
\Xi_i &= -D_i \odot \left(S_i^{(1)}M - MS_i^{(1)}\right) - D_i \odot \left(S_i^{(2)}N + N^T{S_i^{(2)}}^T\right) + \eta_i \odot \left(3D_i^2 - S_i^{(1)}\right)
\end{aligned} \tag{25}$$

for $i = 1, \ldots, m$, and where $\mathrm{skew}(A) = \frac{1}{2}(A - A^T)$ represents the symmetric part of a matrix A, and we have defined, for $i = 1, \ldots, m$,

$$S_i^{(1)} = Q^TS_iQ,\ S_i^{(2)} = Q^TS_iQ_\perp \quad \text{and } S_i^{(3)} = Q_\perp^TS_iQ_\perp. \tag{26}$$

Proof From (18) and (24), we can obtain the following equalities:

$$\begin{aligned}
QM_H + Q_\perp N_H &= -\mathbf{P}_Q\Big(\sum_{i=1}^m \left(F_i\xi D_i^2 - W_iQD_i^2 + 2F_iQD_i\eta_i\right) + \xi\mathrm{sym}(Q^T\nabla_Q\bar{f}(Z))\Big),\\
\Xi_i &= -D_i \odot \left(Q^TF_i\xi + \xi^TF_iQ - Q^TW_iQ\right) - \eta_i \odot \left(Q^TF_iQ\right), \quad i = 1, \ldots, m,
\end{aligned} \tag{27}$$

with F_i and W_i are given in (11) and (15), respectively, for $i = 1, \ldots, m$. Note that for any $W \in \mathbb{R}^{n\times l}$, we have $Q^T\mathbf{P}_Q(W) = \mathrm{skew}(Q^TW)$ and $Q_\perp^T\mathbf{P}_Q(W) = Q_\perp^TW$. By multiplying the first equation of (27) by Q^T on the left-hand side and utilizing the relations $Q^TQ = I_r$ and $Q^TQ_\perp = 0$, we yield

$$M_H = -\,\mathrm{skew}\left(Q^T\left(\sum_{i=1}^m \left(F_i\xi D_i^2 - W_iQD_i^2 + 2F_iQD_i\eta_i\right) + \xi\mathrm{sym}\left(Q^T\nabla_Q\bar{f}(Z)\right)\right)\right). \tag{28}$$

Next, by substituting F_i, W_i as in (11) and (15) respectively, and $\xi = QM + Q_\perp N$ into (28), and using the relations $Q^T\xi = M$ and $Q^TQ_\perp = 0$, we can derive the expression of M_H in (25) by employing $M, N, \eta_1, \ldots, \eta_n$.

Similarly, multiplying the first equation of (27) by $Q_\perp^T$ on the left-hand side yields

$$N_H = -Q_\perp^T \left(\sum_{i=1}^{m} \left(F_i \xi D_i^2 - W_i Q D_i^2 + 2 F_i Q D_i \eta_i \right) + \xi \mathrm{sym}\left(Q^T \nabla_Q \bar{f}(Z) \right) \right). \tag{29}$$

Subsequently, by substituting F_i, W_i and $\xi = QM + Q_\perp N$ into (29), in addition to $Q_\perp^T \xi = N$ and $Q_\perp^T Q = 0$, we obtain N_H in (25) by using $M, N, \eta_1, \ldots, \eta_n$. Lastly, by substituting F_i, W_i and $\xi = QM + Q_\perp N$ into the second to the last equality in (27), we obtain $\Xi_1, \ldots, \Xi_m$ in (25), by using $M, N, \eta_1, \ldots, \eta_n$.

Since the Hessian $\mathrm{Hess}\, f(Z)[\Delta Z]$ acts as a linear transformation with respect to ΔZ, it follows that it is also linear in terms of the set of matrices $(M, N, \eta_1, \ldots, \eta_m)$, where $M \in \mathcal{K}(r)$, $N \in \mathbb{R}^{(n-r)\times r}$ and $\eta_i \in \mathcal{D}(r)$, for $i = 1, \ldots, m$. Furthermore, since $\mathrm{Hess}\, f(Z)[\Delta Z]$ resides in the tangent space $\mathbf{T}_Z\mathcal{M}$, it can be uniquely expressed as in (24) where $M_H \in \mathcal{K}(r)$, $N_H \in \mathbb{R}^{(n-r)\times r}$ and $\Xi_i \in \mathcal{D}(r)$ for $i = 1, \ldots, m$. Let $\phi := \dim(\mathbf{T}_Z\mathcal{M})$ and view the Hessian $\mathrm{Hess}\, f(Z)$ at $Z \in \mathcal{M}$ as a linear transformation $\mathcal{H}$ on $\mathbb{R}^\phi$. Specially, $\mathcal{H}$ transforms a ϕ-dimension vector

$$\left(\mathrm{vec}_K(M)^T, \mathrm{vec}(N)^T, \mathrm{diag}(\eta_1)^T, \ldots, \mathrm{diag}(\eta_m)^T \right)^T \tag{30}$$

into

$$\left(\mathrm{vec}_K(M_H)^T, \mathrm{vec}(N_H)^T, \mathrm{diag}(\Xi_1)^T, \ldots, \mathrm{diag}(\Xi_m)^T \right)^T,$$

where $\mathrm{diag}(D)$ be a column vector composed of diagonal elements of a diagonal matrix D, and $\mathrm{vec}_K(\cdot)$ represents the vec operator that taking skew-symmetric matrices into vectors, which takes the forms of, for any $N = (b_{ij}) \in \mathcal{K}(n)$,

$$\mathrm{vec}_K(N) = \sqrt{2}(b_{21}, b_{31}, \ldots, b_{n1}, b_{32}, b_{42}, \ldots, b_{n2}, \ldots, b_{(n-1)(n-2)}, b_{n(n-2)}, b_{n(n-1)})^T.$$

The following lemma obtains the representation matrix of that linear transformation $\mathcal{H}$.

Lemma 3 *Let $\mathcal{H}$ be a linear transformation on $\mathbb{R}^\phi$ that acts on (30) with $M \in \mathcal{K}(r)$, $N \in \mathbb{R}^{(n-r)\times r}$ and $\eta_i \in \mathcal{D}(r)$ with $i = 1, \ldots, m$ as*

$$\mathcal{H} \begin{pmatrix} \mathrm{vec}_K(M) \\ \mathrm{vec}(N) \\ \mathrm{diag}(\eta_1) \\ \vdots \\ \mathrm{diag}(\eta_m) \end{pmatrix} = \begin{pmatrix} \mathrm{vec}_K(M_H) \\ \mathrm{vec}(N_H) \\ \mathrm{diag}(\Xi_1) \\ \vdots \\ \mathrm{diag}(\Xi_m) \end{pmatrix}, \tag{31}$$

where M_H, N_H, $\Xi_1, \ldots, \Xi_m$ are given in (25). Then, the representation matrix H_ϕ of $\mathcal{H}$ is given by

$$H_\phi = \begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} & \ldots & H_{1,m+2} \\ H_{2,1} & H_{2,2} & H_{2,3} & \ldots & H_{2,m+2} \\ H_{3,1} & H_{3,2} & H_{3,3} & & \\ \vdots & \vdots & & \ddots & \\ H_{m+2,1} & H_{m+2,2} & & & H_{m+2,m+2} \end{pmatrix} \in \mathbb{R}^{\phi\times\phi}, \tag{32}$$

where

$$
\begin{aligned}
H_{1,1} &= -\frac{1}{2}\Phi_r^T \left(I_{r^2} - \Psi_{(r,r)}\right)\left(\sum_{i=1}^{m}\left(D_i^2 \otimes S_1^{(i)} - D_i^4 \otimes I_r\right) + \mathrm{sym}(Q^T\nabla_Q \bar{f}(Z)) \otimes I_r\right)\Phi_r,\\
H_{1,2} &= -\frac{1}{2}\Phi_r^T \left(I_{r^2} - \Psi_{(r,r)}\right)\sum_{i=1}^{m}\left(D_i^2 \otimes S_i^{(2)}\right),\\
H_{1,j} &= -\frac{1}{2}\Phi_r^T \left(I_{r^2} - \Psi_{(r,r)}\right)\sum_{i=1}^{m}\left(I_r \otimes \left(2S_i^{(1)}D_i - 4D_i^3\right)\right)\Gamma_r,\\
H_{2,1} &= -\sum_{i=1}^{m}\left(D_i^2 \otimes {S_i^{(2)}}^T\right)\Phi_r,\\
H_{2,2} &= -\left(\sum_{i=1}^{m}\left(D_i^2 \otimes S_i^{(3)} - D_i^4 \otimes I_{n-r}\right) + \mathrm{sym}(Q^T\nabla_Q \bar{f}(Z)) \otimes I_{n-r}\right),\\
H_{2,j} &= -2\sum_{i=1}^{m}\left(I_r \otimes {S_i^{(2)}}^T D_i\right)\Gamma_r,\\
H_{t+2,1} &= -\Gamma_r^T \Pi_r \left(I_r \otimes D_i S_i^{(1)} - S_i^{(1)} \otimes D_i\right)\Phi_r,\\
H_{t+2,2} &= -\Gamma_r^T \Pi_r \left(I_r \otimes D_i S_i^{(2)} + (S_i^{(2)} \otimes D_i)\Psi_{(r,n-r)}\right),\\
H_{t+2,j} &= \delta_{t+2,j}\Gamma_r^T \Pi_r \left(\left(3D_i^2 - S_i^{(1)}\right) \otimes I_r\right)\Gamma_r,
\end{aligned}
$$

for $j = 3, \ldots, m+2$ *and* $t = 1, \ldots m$, *in which* $\delta_{p,q}$ *is the Kronecker delta, and where* $\Phi_r \in \mathbb{R}^{r^2 \times \frac{r(r-1)}{2}}$, $\Psi_{(r,r)} \in \mathbb{R}^{r^2 \times r^2}$, $\Gamma_r \in \mathbb{R}^{r^2 \times r}$ *and* $\Pi_r \in \mathbb{R}^{r^2 \times r^2}$ *are specific vectorization-related matrices [13] such that*

$$
\begin{aligned}
&\mathrm{vec}(M) = \Phi_r \mathrm{vec}_K(M), \quad M \in \mathcal{K}(r),\\
&\mathrm{vec}(N^T) = \Psi_{(n,m)}\mathrm{vec}(N), \quad N \in \mathbb{R}^{m \times n},\\
&\mathrm{vec}(D) = \Gamma_r\,\mathrm{diag}(D), \quad D \in \mathcal{D}(r),\\
&\mathrm{vec}(D \odot W) = \Pi_r \mathrm{vec}(DW), \quad D \in \mathcal{D}(r), W \in \mathbb{R}^{r \times r}.
\end{aligned}
\tag{33}
$$

with

$$
\Phi_n^T \Phi_n = I_{n(n-1)/2}, \qquad \Gamma_n^T \Gamma_n = I_n \tag{34}
$$

and

$$
\Psi_{(m,n)}\Psi_{(n,m)} = I_{mn} \quad \textit{and} \quad \Psi_{(m,n)}^{-1} = \Psi_{(m,n)}^T = \Psi_{(n,m)}. \tag{35}
$$

Proof Combining equations given in (33)-(35) together with equalities (25) in Lemma 2, $\mathrm{vec}_K(M_H)$, $\mathrm{vec}(N_H)$ and $\mathrm{diag}(\Gamma_{H,i})$ for $i = 1, \ldots, m$ are calculated as follows:

$$
\begin{aligned}
&\mathrm{vec}_K(M_H)\\
=&\Phi_r^T \mathrm{vec}(M_H)\\
=&-\Phi_r^T \mathrm{vec}\Bigg\{ \mathrm{skew}\left(\sum_{i=1}^m \left(S_i^{(1)} M D_i^2 - M D_i^4\right) + M\mathrm{sym}\big(Q^T \nabla_Q \bar{f}(Z)\big) + \sum_{i=1}^m S_i^{(2)} N D_i^2 + \sum_{i=1}^m \left(2S_i^{(1)} D_i - 4D_i^3\right)\eta_i\right)\Bigg\}\\
=&-\frac{1}{2}\Phi_r^T \left(I_{r^2} - \Psi_{(r,r)}\right)\left(\sum_{i=1}^m \left(D_i^2 \otimes S_1^{(i)} - D_i^4 \otimes I_r\right) + \mathrm{sym}(Q^T \nabla_Q \bar{f}(Z)) \otimes I_r\right)\Phi_r \mathrm{vec}_K(M)\\
&-\frac{1}{2}\Phi_r^T \left(I_{r^2} - \Psi_{(r,r)}\right)\sum_{i=1}^m \left(D_i^2 \otimes S_i^{(2)}\right)\mathrm{vec}(N)\\
&-\frac{1}{2}\Phi_r^T \left(I_{r^2} - \Psi_{(r,r)}\right)\sum_{i=1}^m \left(I_r \otimes \left(2S_i^{(1)} D_i - 4D_i^3\right)\right)\Gamma_r \mathrm{diag}(\eta_i)\\
=&H_{1,1}\mathrm{vec}_K(M) + H_{1,2}\mathrm{vec}(N) + \sum_{i=1}^m H_{1,2+i}\,\mathrm{diag}(\eta_i),
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{vec}(N_H)\\
=&-\left(\sum_{i=1}^m {S_i^{(2)}}^T M D_i^2 + \sum_{i=1}^m \left(S_i^{(3)} N D_i^2 - N D_i^4\right) + N\mathrm{sym}\big(Q^T \nabla_Q \bar{f}(Z)\big) + \sum_{i=1}^m 2{S_i^{(2)}}^T D_i \eta_i\right)\\
=&-\sum_{i=1}^m \left(D_i^2 \otimes {S_i^{(2)}}^T\right)\Phi_r \mathrm{vec}_K(M)\\
&-\left(\sum_{i=1}^m \left(D_i^2 \otimes S_i^{(3)} - D_i^4 \otimes I_{n-r}\right) + \mathrm{sym}(Q^T \nabla_Q \bar{f}(Z)) \otimes I_{n-r}\right)\mathrm{vec}(N)\\
&-2\sum_{i=1}^m \left(I_r \otimes {S_i^{(2)}}^T D_i\right)\Gamma_r \mathrm{diag}(\eta_i)\\
=&H_{2,1}\mathrm{vec}_K(M) + H_{2,2}\mathrm{vec}(N) + \sum_{i=1}^m H_{2,2+i}\,\mathrm{diag}(\eta_i),
\end{aligned}
$$

and for $i = 1, \ldots, m$,

$$
\begin{aligned}
&\mathrm{diag}(\Xi_i)\\
=&\Gamma_r^T \mathrm{vec}(\Xi_i)\\
=&-\Gamma_r^T \mathrm{vec}\left(D_i \odot \left(S_i^{(1)} M - M S_i^{(1)}\right) + D_i \odot \left(S_i^{(2)} N + N^T {S_i^{(2)}}^T\right) - \eta_i \odot \left(3D_i^2 - S_i^{(1)}\right)\right)\\
=&-\Gamma_r^T \Pi_r \left(I_r \otimes D_i S_i^{(1)} - S_i^{(1)} \otimes D_i\right)\Phi_r \mathrm{vec}_K(M)\\
&-\Gamma_r^T \Pi_r \left(I_r \otimes D_i S_i^{(2)} + (S_i^{(2)} \otimes D_i)\Psi_{(r,n-r)}\right)\mathrm{vec}(N)\\
&+\Gamma_r^T \Pi_r \left(\left(3D_i^2 - S_i^{(1)}\right) \otimes I_r\right)\Gamma_r \mathrm{diag}(\eta_i)\\
=&H_{2+i,1}\mathrm{vec}_K(M) + H_{2+i,2}\mathrm{vec}(N) + H_{2+i,2+i}\,\mathrm{diag}(\eta_i).
\end{aligned}
$$

This completes the proof.

The following proposition characterizes the structure of the representation matrix H_ϕ; its proof is provided in Appendix B of [33].

Proposition 31 *With respect to the induced inner product (5), the representation matrix H_ϕ is symmetric.*

3.2 Solve Newton's equation (21)

We are now in the position to solve Newton's equation (21). From (24) and (26), Newton's equation (21) is equivalent to the following

$$\begin{cases} QM_H + Q_\perp N_H = \mathbf{P}_Q\left(\sum_{i=1}^{m}\left(S_i Q D_i^2 - Q D_i^4\right)\right), \\ \Xi_i = D_i \odot \left(S_i^{(1)} - D_i^2\right), \quad i = 1, \ldots, m. \end{cases} \tag{36}$$

By multiplying the first equation of (36) from the left by Q^T and $Q_\perp^T$, and utilizing the identities $Q^T\mathbf{P}_Q(W) = \mathrm{skew}(Q^T W)$ and $Q_\perp^T\mathbf{P}_Q(W) = Q_\perp^T W$ for any $W \in \mathbb{R}^{n\times r}$, along with equation (26), Newton's equation (21) can be further rewritten in an equivalent form as follows:

$$\begin{cases} M_H = \mathrm{skew}\left(\sum_{i=1}^{m}\left(S_i^{(1)} {D_i}^2 - {D_i}^4\right)\right), \\ N_H = \sum_{i=1}^{m} {S_i^{(2)}}^T {D_i}^2, \\ \Xi_i = D_i \odot \left(S_i^{(1)} - D_i^2\right), \quad i = 1, \ldots, m. \end{cases} \tag{37}$$

By applying the vec operation to all equations of (37), we obtain

$$\begin{cases} \mathrm{vec}_K(M_H) = \frac{1}{2}\Phi_r^T\left(I_{r^2} - \Psi_{(r,r)}\right)\mathrm{vec}\left(\sum_{i=1}^{m}\left(S_i^{(1)} {D_i}^2 - {D_i}^4\right)\right), \\ \mathrm{vec}(N_H) = \mathrm{vec}\left(\sum_{i=1}^{m} {S_i^{(2)}}^T {D_i}^2\right), \\ \mathrm{diag}(\Xi_i) = \Gamma_r^T \Pi_r \mathrm{vec}\left(D_i S_i^{(1)} - D_i^3\right), \quad i = 1, \ldots, m. \end{cases} \tag{38}$$

Following from Lemma 3, Newton's equation (21) can be represented in the following form:

$$H_\phi x = g, \tag{39}$$

where H_ϕ is given in (32), which is the representation matrix of $\mathrm{Hess}\, f(Z)$, and

$$x = \begin{pmatrix} \mathrm{vec}_K(M) \\ \mathrm{vec}(N) \\ \mathrm{diag}(\eta_1) \\ \vdots \\ \mathrm{diag}(\eta_m) \end{pmatrix}, \quad g = \begin{pmatrix} \frac{1}{2}\Phi_r^T\left(I_{r^2} - \Psi_{(r,r)}\right)\mathrm{vec}\left(\sum_{i=1}^{m}\left(S_i^{(1)} {D_i}^2 - {D_i}^4\right)\right) \\ \mathrm{vec}\left(\sum_{i=1}^{m} {S_i^{(2)}}^T {D_i}^2\right) \\ \Gamma_r^T \Pi_r \mathrm{vec}\left(D_1 S_1^{(1)} - D_1^3\right) \\ \vdots \\ \Gamma_r^T \Pi_r \mathrm{vec}\left(D_m S_m^{(1)} - D_m^3\right) \end{pmatrix}. \tag{40}$$

Thus, we have derived an equivalent formulation of Newton's equation (21) in the form of a standard symmetric linear system. By solving this symmetric linear equation (37), we obtain $\mathrm{vec}_K(M)$, $\mathrm{vec}(N)$ and $\mathrm{diag}(\eta_i)$ for $i = 1, \ldots, m$. Once these vectorized components are determined, we can efficiently reshape them into their respective matrix forms: $M \in \mathcal{K}(r)$, $N \in \mathbb{R}^{(n-r)\times r}$ and $\eta_i \in \mathcal{D}(r)$. This allows us to construct the solution to Newton's equation (21), given by $\Delta Z = (QM + Q_\perp N, \eta_1, \ldots, \eta_m)$.

3.3 Newton's method for solving the problem (3)

Based on the above discussion, we propose the following algorithm for solving the O-INDSCAL problem (3).

Algorithm 1 Newton's method for the problem (3)

1: Choose an initial point $Z^{(0)} = (Q^{(0)}, D_1^{(0)}, \ldots, D_m^{(0)}) \in \mathcal{M}$.
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Compute $Q_\perp^{(k)}$ that satisfies ${Q^{(k)}}^T Q_\perp^{(k)} = 0$ and ${Q_\perp^{(k)}}^T Q_\perp^{(k)} = I_{n-r}$.
4: Compute the partition matrix $H_\phi^{(k)}$ by Lemma 3 and the vector $g^{(k)}$ by (40). Compute $x^{(k)} \in \mathbb{R}^\phi$ that satisfies the symmetric linear equation

$$H_\phi^{(k)} x^{(k)} = g^{(k)}. \tag{41}$$

5: Partition $x^{(k)} \in \mathbb{R}^\phi$ as ${x^{(k)}}^T = \begin{bmatrix} {x_1^{(k)}}^T & {x_2^{(k)}}^T & {x_3^{(k)}}^T & \ldots & {x_{m+2}^{(k)}}^T \end{bmatrix}$, where $x_1^{(k)} \in \mathbb{R}^{\frac{r(r-1)}{2}}$, $x_2^{(k)} \in \mathbb{R}^{(n-r)r}$, $x_3^{(k)} \in \mathbb{R}^r$, …, $x_{m+2}^{(k)} \in \mathbb{R}^r$.
6: Construct $M^{(k)} \in \mathcal{K}(r)$, $N^{(k)} \in \mathbb{R}^{(n-r)\times r}$ and $\eta_i^{(k)} \in \mathcal{D}(r)$ for $i = 1, \ldots, m$ that satisfy $\mathrm{vec}_K(M^{(k)}) = x_1^{(k)}$, $\mathrm{vec}(N^{(k)}) = x_2^{(k)}$ and $\mathrm{diag}(\eta_i^{(k)}) = x_{2+i}^{(k)}$.
7: Compute $\Delta Z^{(k)} = \left(Q^{(k)} M^{(k)} + Q_\perp^{(k)} N_x^{(k)}, \eta_1^{(k)}, \ldots, \eta_m^{(k)}\right)$.
8: Compute the next iterate $Z^{(k+1)} = \mathcal{R}_{Z^{(k)}}(\Delta Z^{(k)})$, where $\mathcal{R}$ is a retraction on $\mathcal{M}$.
9: **end for**

Remark 1 In the Step 3 of Algorithm 1, we can arbitrarily fix $Q_\perp^{(k)}$ to satisfy $(Q^{(k)})^T Q_\perp^{(k)} = 0$ and $(Q_\perp^{(k)})^T Q_\perp^{(k)} = I_{n-r}$. Such $Q_\perp^{(k)}$ can be computed by the full QR decompositions. In practice, using the MATLAB's qr function with input $Q^{(k)} \in \mathcal{O}(n, r)$ returns an orthogonal matrix $\tilde{Q} \in \mathbb{R}^{n\times n}$ such that $Q^{(k)} = \tilde{Q} \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$ where R_1 is a $r \times r$ upper triangular matrix. If we partition $\tilde{Q} = (Q_1, Q_2)$ with $Q_1 \in \mathcal{O}(n, r)$ and $Q_2 \in \mathcal{O}(n, n-r)$, then $Q_2^T Q^{(k)} = (0, I_{n-r}) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = 0$. Therefore, we can choose Q_2 as $Q_\perp^{(k)}$.

If a good approximate solution to the problem is available beforehand, Newton's method is highly effective. This effectiveness stems from the fact that Newton's method generally produces sequences that converge quadratically, albeit locally. This property is formally established in the following lemma, which corresponds to Theorem 6.3.2 in [1].

Lemma 4 (Local convergence of Newton's method) *Consider the problem of minimizing an objective function f on a Riemannian manifold $\mathcal{M}$. Let $x_c \in \mathcal{M}$ be a critical point of f;* $\mathrm{grad}\, f(x_c) = 0$. *Assume that* $\mathrm{Hess}\, f(x_c)$ *is nondegenerate at $x_c \in \mathcal{M}$. Then there exists a neighborhood U of x_c in $\mathcal{M}$ such that for all $x_0 \in U$, the Newton algorithm generates an infinite sequence $\{x_k\}$ converging quadratically to x_c.*

We have several remarks for the representation matrix H_ϕ and Algorithm 1 as follows.

Remark 2 The representation matrix H_ϕ provides a useful tool for verifying the positive definiteness of the Hessian $\operatorname{Hess} f(Z)$, which is essential for determining whether a critical point identified by Algorithm 1 corresponds to a local minimum. In general, the Newton step $\Delta Z^{(k)}$, which solves (19), does not necessarily serve as a descent direction for the function $f(Z^{(k)})$. Specifically, we have

$$\mathcal{D}f(Z)[\Delta Z] = \langle \operatorname{grad} f(Z),\ \Delta Z\rangle = -\langle \operatorname{grad} f(Z),\ \operatorname{Hess} f(Z)^{-1} \operatorname{grad} f(Z)\rangle,$$

which is not guaranteed to be negative unless additional assumptions are imposed on the operator $\operatorname{Hess} f(Z)$. A sufficient condition for $\Delta Z^{(k)}$ to be a descent direction is the positive definiteness of $\operatorname{Hess} f(Z^{(k)})$, meaning that

$$\langle \Delta Z,\ \operatorname{Hess} f(Z)[\Delta Z]\rangle > 0, \quad \forall\ \Delta Z \neq 0.$$

From Lemma 3, it follows that for any $\Delta Z = (QM + Q_\perp N, \eta_1, \ldots, \eta_m)$, where $M \in \mathcal{K}(r)$, $N \in \mathbb{R}^{(n-r)\times r}$ and $\eta_i \in \mathcal{D}(r)$ for $i = 1, \ldots, m$, the quadratic form associated with the Hessian can be expressed as

$$\langle \Delta Z,\ \operatorname{Hess} f(Z)[\Delta Z]\rangle = \begin{pmatrix} \operatorname{vec}_K(M) \\ \operatorname{vec}(N) \\ \operatorname{diag}(\eta_1) \\ \vdots \\ \operatorname{diag}(\eta_m) \end{pmatrix}^T H_\phi \begin{pmatrix} \operatorname{vec}_K(M) \\ \operatorname{vec}(N) \\ \operatorname{diag}(\eta_1) \\ \vdots \\ \operatorname{diag}(\eta_m) \end{pmatrix}.$$

Thus, if the symmetric matrix H_ϕ is positive definite, then $\operatorname{Hess} f(Z)$ is also positive definite, implying that Z is a local minimum.

Remark 3 Similar to unconstrained optimization in the Euclidean space, $\|\operatorname{grad} f(Z^{(k)})\| < \varepsilon$ for some constant $\varepsilon > 0$ is a reasonable stopping criterion for Riemannian optimization approaches to the problem (3). In fact, the Lagrangian of the problem (3) is

$$\mathcal{L}(Q, D_1, \ldots, D_m, \Lambda) = \bar{f}(Z) - \tfrac{1}{2}\operatorname{tr}\left(\Lambda(Q^TQ - I_r)\right), \qquad D_i \in \mathcal{D}(r), \quad i = 1, \ldots m,$$

where Λ is symmetric representing the Lagrange multiplier. Then the first-order optimality conditions under the Euclidean framework are

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial Q} = \nabla_Q \bar{f}(Z) - Q\nabla_Q \bar{f}(Z)^T Q = 0, \\ \mathbf{P}_D\left(\dfrac{\partial \mathcal{L}}{\partial D_i}\right) = \mathbf{P}_D\left(\nabla_{D_i}\bar{f}(Z)\right) = 0, \qquad i = 1, \ldots m, \\ Q^TQ = I_r, \end{cases}$$

with $\Lambda = \nabla_Q \bar{f}(Z)^T Q$. Under the condition $Q^TQ = I_r$, $\frac{\partial \mathcal{L}}{\partial Q} = 0$ is equivalent to $\operatorname{grad}_Q f(Z) = 0$, since it follows from (9), (6) and (7) that

$$\begin{aligned} \mathbf{P}_Q\Big(\nabla_Q \bar{f}(Z)\Big) &= \nabla_Q \bar{f}(Z) - Q\operatorname{sym}\Big(Q^T\nabla_Q \bar{f}(Z)\Big) \\ &= \Big(I_r - \frac{1}{2}QQ^T\Big)\Big(\nabla_Q \bar{f}(Z) - Q\nabla_Q \bar{f}(Z)^T Q\Big) \\ &= \Big(I_r - \frac{1}{2}QQ^T\Big)\frac{\partial \mathcal{L}}{\partial Q}. \end{aligned}$$

Thus, first-order critical points in the Euclidean sense can be interpreted as stationary points under the Riemannian framework.

Remark 4 The symmetric linear system (41) can, in principle, be solved via direct matrix inversion when $H_\phi^{(k)}$ is nonsingular. However, for high-dimensional problems, direct inversion becomes computationally prohibitive. In such cases, iterative Krylov subspace methods provide a more practical alternative. One such method is the Conjugate Residual (CR) algorithm (see [18, page 182]), which is designed to solve linear systems of the form $Hx = b$, where H is symmetric (not necessarily positive definite) and invertible, and b is a nonzero vector. Importantly, since (41) imposes no structural constraints on the solution vector x, a variety of modern Krylov subspace solvers are applicable. When the Riemannian Hessian $\operatorname{Hess} f(Z)$ is known to be positive definite, the Conjugate Gradient (CG) method is a suitable choice. On the other hand, if (41) is interpreted as a general linear system-without requiring symmetry of the representation matrix H_ϕ-then more general-purpose solvers such as the Generalized Minimal Residual (GMRES) method or the BiConjugate Gradient Stabilized (BiCGSTAB) method may also be employed. These methods are widely used in large-scale scientific computing and offer robust performance across a broad range of problem classes.

Remark 5 Building on the component-wise characterization of the Riemannian Hessian on the tangent space, we can extend the classical trust-region framework to the Riemannian setting and thereby obtain a modified Riemannian trust-region (RTR) algorithm for the structured optimization problem (3). At the kth iteration, let the current iterate be denoted by $Z^{(k)} = \big(Q^{(k)}, D_1^{(k)}, \ldots, D_m^{(k)}\big)$. A second-order model of the objective function is constructed on the tangent space $\mathbf{T}_{Z^{(k)}}\mathcal{M}$ via a Riemannian Taylor expansion, leading to the trust-region subproblem

$$\min_{\Delta Z} \ \mathbf{M}_{Z^{(k)}}(\Delta Z) \quad \text{subject to} \quad \Delta Z = (\xi, \eta_1, \ldots, \eta_m) \in \mathbf{T}_{Z^{(k)}}\mathcal{M}, \quad \|\Delta Z\| \le \Delta^{(k)}, \tag{42}$$

where the model function is given by

$$\mathbf{M}_{Z^{(k)}}(\Delta Z) = f\big(Z^{(k)}\big) + \big\langle \operatorname{grad} f\big(Z^{(k)}\big),\, \Delta Z\big\rangle + \frac{1}{2}\big\langle \operatorname{Hess} f\big(Z^{(k)}\big)[\Delta Z],\, \Delta Z\big\rangle.$$

Using (23) and (24), the quadratic term can be written as

$$\begin{aligned}
&\big\langle \operatorname{Hess} f\big(Z^{(k)}\big)[\Delta Z],\, \Delta Z\big\rangle \\
&= \Big\langle \Big(Q^{(k)} M_H^{(k)} + Q_\perp^{(k)} N_H^{(k)},\, \Xi_1^{(k)}, \ldots, \Xi_m^{(k)}\Big),\ \Big(Q^{(k)} M + Q_\perp^{(k)} N,\, \eta_1, \ldots, \eta_m\Big)\Big\rangle \\
&= \operatorname{tr}\Big(-M_H^{(k)} M + {N_H^{(k)}}^T N + \sum_{i=1}^{m} \Xi_i^{(k)} \eta_i\Big),
\end{aligned}$$

where the matrices M_H, N_H, and Ξ_i for $i = 1, \ldots, m$ are defined in Lemma 2. Likewise, by Lemma 1, the Riemannian gradient admits the decomposition

$$\operatorname{grad} f(Z) = \big(QM_G + Q_\perp N_G,\, \Gamma_1, \ldots, \Gamma_m\big),$$

with components

$$M_G = -\operatorname{skew}\left(\sum_{i=1}^{m}\Big(S_i^{(1)} {D_i}^2 - {D_i}^4\Big)\right), \quad N_G = -\sum_{i=1}^{m} {S_i^{(2)}}^T {D_i}^2, \quad \Gamma_i = -D_i \odot \Big(S_i^{(1)} - D_i^2\Big).$$

Substituting these expressions into the gradient term yields

$$\begin{aligned}
&\langle \operatorname{grad} f\big(Z^{(k)}\big),\, \Delta Z\rangle \\
&= \Big\langle \Big(Q^{(k)} M_G^{(k)} + Q_\perp^{(k)} N_G^{(k)},\, \Gamma_1^{(k)}, \ldots, \Gamma_m^{(k)}\Big),\ \Big(Q^{(k)} M + Q_\perp^{(k)} N,\, \eta_1, \ldots, \eta_m\Big)\Big\rangle \\
&= \operatorname{tr}\Big(-M_G^{(k)} M + N_G^{(k)^T} N + \sum_{i=1}^{m} \Gamma_i^{(k)} \eta_i\Big).
\end{aligned}$$

Combining the above, the Riemannian trust-region subproblem (42) is equivalently reformulated in terms of the variables $M \in \mathcal{K}(r)$, $N \in \mathbb{R}^{(n-r)\times r}$, and $\eta_i \in \mathcal{D}(r)$, $i = 1, \ldots, m$, as

$$\begin{aligned}
\min_{M,\, N,\, \eta_1, \ldots, \eta_m} \quad & \mathbf{M}_{Z^{(k)}}(M, N, \eta_1, \ldots, \eta_m) \\
\text{subject to} \quad & \|M\|^2 + \|N\|^2 + \sum_{i=1}^{m} \|\eta_i\|^2 \le \big(\Delta^{(k)}\big)^2,
\end{aligned} \tag{43}$$

where

$$\mathbf{M}_{Z^{(k)}}(M, N, \eta_1, \ldots, \eta_m) = f\big(Z^{(k)}\big) + \operatorname{tr}\Big(-M_t^{(k)} M + N_t^{(k)^T} N + \sum_{i=1}^{m} \Gamma_{t,i}^{(k)} \eta_i\Big),$$

and the intermediate matrices are defined by

$$M_t^{(k)} = M_G^{(k)} + \frac{1}{2} M_H^{(k)}, \quad N_t^{(k)} = N_G^{(k)} + \frac{1}{2} N_H^{(k)}, \quad \Gamma_{t,i}^{(k)} = \Gamma_i^{(k)} + \frac{1}{2} \Xi_i^{(k)}, \quad i = 1, \ldots, m.$$

The overall framework of the modified RTR method and the tCG procedure for solving the resulting structured trust-region subproblem (43) follows the ideas developed for the direct INDSCAL model (see Algorithm 1 and Algorithm 2 in [32]). Fix an outer iteration index k, and let $\Delta Z^j \in \mathbf{T}_{Z^{(k)}}\mathcal{M}$ denote the jth iterate produced by the tCG algorithm for the trust-region subproblem (42). In standard RTR methods, all updates are carried out directly in the tangent space. By contrast, in the modified tCG method, we update only the component matrices $M^j \in \mathcal{K}(r)$, $N^j \in \mathbb{R}^{(n-r)\times r}$, and $\eta_1^j, \ldots, \eta_m^j \in \mathcal{D}(r)$, and implicitly form the tangent vector via $\Delta Z^j = \big(Q^{(k)} M^j + Q_\perp^{(k)} N^j,\ \eta_1^j, \ldots, \eta_m^j\big)$. As a consequence, only the final step ΔZ^j needs to be assembled explicitly at termination; intermediate tangent vectors ΔZ^i for $i < j$ are never formed, which reduces memory traffic and computational overhead. Moreover, classical RTR schemes typically require the full Riemannian Hessian action $\operatorname{Hess} f(Z^{(k)})[\Phi_j]$, where $\Phi_j = \big(Q^{(k)} \delta_{M,j} + Q_\perp^{(k)} \delta_{N,j},\ \delta_{\eta_1,j}, \ldots, \delta_{\eta_m,j}\big) \in \mathbf{T}_{Z^{(k)}}\mathcal{M}$. In our implementation, the modified tCG method works directly with the component-wise quantities $M_{\Phi,j} \in \mathcal{K}(r)$, $N_{\Phi,j} \in \mathbb{R}^{(n-r)\times r}$, and $\Xi_{\Phi i,j} \in \mathcal{D}(r)$ for $i = 1, \ldots, m$, such that $\operatorname{Hess} f(Z^{(k)})[\Phi_j] = \big(Q^{(k)} M_{\Phi,j} + Q_\perp^{(k)} N_{\Phi,j},\ \Xi_{\Phi 1,j}, \ldots, \Xi_{\Phi m,j}\big)$. This component-wise representation avoids repeated construction and manipulation of full tangent vectors and yields a more efficient computation of Hessian–vector products within each tCG iteration.

3.4 A hybrid method for the problem (3)

We have developed a quadratically convergent algorithm for solving the orthonormality-constrained INDSCAL problem (3) based on Riemannian Newton's method. While Newton's method offers rapid local convergence, it lacks global convergence guarantees and therefore requires a sufficiently accurate initial estimate. To address this issue, we incorporate a Riemannian curvilinear search strategy-originally proposed in Wen and Yin [31]-as a means of generating a suitable starting point. This method is computationally lightweight, memory-efficient, and easy to implement. As demonstrated in our numerical experiments, the Riemannian curvilinear search approach converges quickly during the initial iterations. However, its performance tends to plateau as the iterates approach a local minimizer. In contrast, second-order methods such as Riemannian Newton are more effective at achieving high-precision solutions. To harness the strengths of both methods, we propose a hybrid scheme that combines the global convergence of the curvilinear search with the local efficiency of Newton's method.

We first outline the Riemannian curvilinear search algorithm. Let $\mathcal{R}$ denote a retraction operator. The iterative scheme updates the current iterate $Z^{(k)}$ via

$$Z^{(k+1)} = \mathcal{R}_{Z^{(k)}}\left(t_k\big(\Delta Z^{(k)}\big)\right),$$

where $\Delta Z^{(k)} \in \mathbf{T}_{Z^{(k)}}\mathcal{M}$ is a descent direction and $t_k > 0$ is a step size determined by a nonmonotone Armijo-type line search. Specifically, for fixed constants $\rho, \varrho, \delta \in (0,1)$, the step size $t_k = \gamma_k \delta^h$ is chosen to satisfy the inequality

$$f\left(\mathcal{R}_{Z^{(k)}}\big(\Delta Z^{(k)}\big)\right) \le C_k + \rho t_k \left\langle \operatorname{grad} f(Z^{(k)}),\ \Delta Z^{(k)} \right\rangle, \tag{44}$$

where the reference value C_k is updated using the formula

$$C_{k+1} = \frac{1}{P_{k+1}}\left(\varrho P_k C_k + f(Z^{(k+1)})\right),$$

where $C_0 = f(Z^{(0)})$, $P_{k+1} = \varrho P_k + 1$, and $P_0 = 1$. To improve convergence, we incorporate the Barzilai-Borwein (BB) step size rule [4], which is widely used in large-scale unconstrained optimization. Following Wen and Yin [31], the BB step size is computed via either of the two formulas

$$\gamma_k^{(1)} = \frac{\operatorname{tr}(S_{k-1}^T S_{k-1})}{\left|\operatorname{tr}(V_{k-1}^T S_{k-1}))\right|}, \qquad \text{or} \qquad \gamma_k^{(2)} = \frac{\left|\operatorname{tr}(S_{k-1}^T V_{k-1}))\right|}{\operatorname{tr}(V_{k-1}^T V_{k-1}))}, \tag{45}$$

where

$$S_{k-1} = Z^{(k)} - Z^{(k-1)}, \quad V_{k-1} = \operatorname{grad} f(Z^{(k)}) - \operatorname{grad} f(Z^{(k-1)}).$$

We summarize this procedure in Algorithm 2.

Algorithm 2 Riemannian curvilinear search method with BB steps for the problem (3)

1: Choose an initial point $Z^{(0)} \in \mathcal{M}$. Set $k = 0$, $\gamma_{\min} \in [0,1]$, $\gamma_{\max} \ge 1$, $C_0 = f(Z^{(0)})$, $P_0 = 1$.
2: **while** $\|\operatorname{grad} f(Z^{(k)})\| \neq 0$ **do**
3: Compute $\Delta Z^{(k)} = -\operatorname{grad} f(Z^{(k)})$.

4: Calculate γ_k according to (45) and set $\gamma_k = \max\left(\gamma_{\min}, \min(\gamma_k,\ \gamma_{\max})\right)$. Then, compute C_k, P_k and find a step size t_k satisfying (44).
5: Set $Z^{(k+1)} := \mathcal{R}_{Z^{(k)}}\left(t_k\left(\Delta Z^{(k)}\right)\right)$.
6: Set $k := k + 1$.
7: **end while**

The convergence of Algorithm 2 follows from Theorem 3 in Hu et al. [14], which guarantees that under mild conditions-including differentiability of f, boundedness from below, and smoothness of the retraction-the sequence $\{Z^{(k)}\}$ converges to a stationary point.

Lemma 5 (Global convergence of Algorithm 2). *Let $\{Z^{(k)}\}$ be the sequence generated by Algorithm 2 using the nonmonotone line search (44). Then, every accumulation point of $\{Z^{(k)}\}$ is a stationary point of problem (3), i.e.,*

$$\lim_{k\to\infty} \operatorname{grad} f(Z^{(k)}) = 0.$$

To combine the global convergence property of the curvilinear search method with the fast local convergence of Newton's method, we propose the hybrid Algorithm 3. The method initially applies Algorithm 2 and switches to Newton's method (Algorithm 1) when a suitable switching criterion is met.

Algorithm 3 A hybrid method for the problem (3)

1: Choose an initial point $Z^{(0)} \in \mathcal{M}$, and prepare a switching criterion. Set $k := 0$ $\gamma_{\min} \in [0, 1]$, $\gamma_{\max} \geq 1$, $C_0 = f(Z^{(0)})$, $P_0 = 1$.
2: Set $\Delta Z^{(0)} = -\operatorname{grad} f(Z^{(0)})$.
3: **while** the switching criterion is not satisfied **do**
4: Perform Steps 3–5 in Algorithm 2.
5: $k := k + 1$.
6: **end while**
7: Set $Z^{(0)} := Z^{(k)}$.
8: Perform Steps 2–9 in Algorithm 1.

A critical component of Algorithm 3 lies in determining the appropriate point at which to transition from the Riemannian curvilinear search phase to the Newton refinement phase. As suggested in Li et al. [16], where a hybrid method combining the conjugate gradient and Newton algorithms was developed for computing the parameterized generalized eigenvalue problem, one commonly adopted switching criterion involves monitoring the norm of the Riemannian gradient

$$\|\operatorname{grad} f(Z^{(k)})\| \leq \varepsilon_1, \tag{46}$$

where $\varepsilon_1 > 0$ is a user-specified threshold. However, selecting an optimal value for ε_1 a priori is often nontrivial in practice. An alternative and more practical switching strategy is to monitor the relative reduction in the objective function value

$$\frac{|f(Z^{(k)}) - f(Z^{(k+1)})|}{|f(Z^{(k)})|} \leq \varepsilon_2, \tag{47}$$

where $\varepsilon_2 > 0$ is a small prescribed constant. Under this rule, Algorithm 2 switches to the Newton phase once the decrease in the objective function between successive iterations becomes marginal.

Given that both the Riemannian curvilinear search and the Riemannian Newton methods have been rigorously formulated for solving the problem (3), the global convergence of Algorithm 2 follows directly from Lemma 5. That is, the iterates generated by the hybrid algorithm are guaranteed to converge to a critical point of f. Furthermore, by Lemma 4, the Newton phase exhibits quadratic convergence when initialized sufficiently close to an optimal solution. Thus, with an appropriately chosen ε_1 or ε_2, the proposed hybrid scheme enjoys both global convergence and rapid local refinement, combining the robustness of gradient-based search with the efficiency of Newton-type updates.

Proposition 1 *If $\varepsilon_1 > 0$ or $\varepsilon_2 > 0$ is sufficiently small, Algorithm 3, with the switching criterion (46) or (47), generates a sequence that is globally and quadratically convergent.*

This following remark provides a detailed analysis of the computational complexity of the proposed Algorithm 3.

Remark 6 The overall computational complexity of Algorithm 3 is dominated by two phases: a globally convergent curvilinear-search phase and a locally quadratically convergent Riemannian Newton phase. In the first phase, each curvilinear-search iteration requires one Riemannian gradient evaluation and a small number of function evaluations for the nonmonotone line search, leading to a cost of order $\mathcal{O}\big((1+T_{\mathrm{ls}})mn^2r\big)$ flops per iteration, where (m) is the number of slices, (n) the number of stimuli, (r) the embedding dimension, and T_{ls} the average number of backtracking steps. In the second phase, each Newton iteration involves constructing the reduced Hessian representation and right-hand side, of cost $\mathcal{O}(mn^3 + mnr^3)$, followed by the solution of a symmetric linear system of size $\varphi = (m+n)r - \frac{1}{2}r(r+1)$ by a Krylov method (e.g., CR), with complexity $\mathcal{O}\big(L, \varphi^2\big) = \mathcal{O}\big(L(m+n)^2r^2\big)$, where L is the average number of Krylov iterations. Thus, if N_{CS} curvilinear iterations and N_{N} Newton iterations are performed, the total cost of Algorithm 3 is

$$\mathcal{O}\big(N_{\mathrm{CS}}(1+T_{\mathrm{ls}})mn^2r\big) + \mathcal{O}\big(N_{\mathrm{N}}(mn^3 + mnr^3 + L(m+n)^2r^2)\big),$$

with the expensive second-order operations confined to a relatively small number of Newton steps near the solution.

4 Numerical experiments

In this section, we investigate the numerical performance of the proposed hybrid Riemannian Newton method (Algorithm 3, hereafter denoted by `Hybrid-Newton`) for solving the O-INDSCAL fitting problem (3). We first consider an illustrative real-data example involving similarity judgments among ten cola beverages rated by ten subjects. This dataset, originally reported in Schiffman et al. [20] and subsequently analyzed in Trendafilov [28], Zhou et al. [33], corresponds to the setting $n = 10$ and $m = 10$. We then perform a series of synthetic-data experiments to benchmark `Hybrid-Newton` against several state-of-the-art algorithms. In particular, we compare with the Riemannian inexact Newton method and a suitably adapted modified Riemannian trust-region method from our previous work [33, 32] (abbreviated as `Inexact-Newton` and `Modified-RTR`, respectively), the projected gradient method (`PG`) of Trendafilov [26, 27],

which is a widely used algorithm for O-INDSCAL, as well as standard first- and second-order Riemannian optimization schemes implemented in the MATLAB toolbox Manopt [7, 29]. We further include in our comparison two alternating least-squares variants: the original `ALS` algorithm and its accelerated version proposed in Takane et al. [22]. All numerical experiments were carried out in MATLAB R2024b on a desktop computer equipped with an Intel(R) Core(TM)2 Quad CPU at 2.33 GHz and 3.00 GB of RAM. MATLAB implementations of all methods used in the numerical experiments are available at `Hybrid_Newton_for_OINDSCAL_MATLAB_Codes`.

To generate synthetic datasets for benchmarking, we follow the experimental design described in Takane et al. [22]. Three types of datasets are considered: one unstructured (random) and two partially structured. For the random case (RAND), each slice S_i is generated by sampling all entries independently from a standard normal distribution using `randn`(n,n), followed by symmetrization, $S_i \leftarrow \frac{S_i+S_i^T}{2}$. In the structured cases, the data slices S_i are generated according to the model

$$S_i = Q\tilde{D}_i Q^T + E_i,$$

where $Q \in \mathbb{R}^{n\times r}$ and $\tilde{D}_i \in \mathcal{D}(r)$ are random matrices. Specifically, the entries of Q are drawn from a uniform distribution on $[0, 1]$ using `rand`(n,r), and Q is then orthonormalized via the economy-size singular value decomposition (SVD). The diagonal entries of each $\tilde{D}_i$ are sampled from a standard normal distribution. In the "indefinite" case (IND), these diagonal values are retained as generated; in the "nonnegative definite" case (NND), negative diagonal values are replaced by their absolute values. The noise term E_i has entries sampled from a zero-mean normal distribution with variance σ^2, where σ is set to 10% of the standard deviation of the structural component $Q\tilde{D}_i Q^T$. To better reflect practical applications, the number of "subjects" m is taken up to $m \le 500$, while the number of "stimuli" n is restricted to $n \le 200$. For ease of visualization of the resulting MDS configurations, the embedding dimension r is varied over three levels, $r \in \{2, 3, 5\}$.

Initial values $Z^{(0)} = \big(Q^{(0)}, D_1^{(0)}, \ldots, D_m^{(0)}\big)$ for the proposed `Hybrid-Newton` algorithm are chosen either randomly or via a deterministic, structure-exploiting strategy. In the random case, we first generate an $n \times r$ matrix $Q = \texttt{randn}(n, r)$ and orthonormalize it using a QR decomposition to obtain $Q^{(0)}$. For each i, the diagonal entries of $D_i^{(0)}$ are initialized by $D_i^{(0)} = \mathrm{diag}\big(\texttt{randn}(r, 1)\big)$. For structured initialization, we adopt the strategy of Trendafilov [26]: the empirical mean $\sum_{i=1}^m S_i$ is decomposed via SVD as $P\Sigma P^T$, and $Q^{(0)}$ is chosen as the $n \times r$ matrix consisting of the r leading singular vectors P_r. The initial salience matrices are then set as

$$D_i^{(0)^2} = \max\Big\{\mathrm{diag}\big(Q^{(0)^T} S_i Q^{(0)}\big), 0\Big\}, \quad i = 1, \ldots, m.$$

In all experiments, the proposed `Hybrid-Newton` algorithm uses the default switching parameter $\varepsilon_1 = 10^{-1}$ as defined in (46); for large-scale instances, we also consider the tighter choice $\varepsilon_1 = 10^{-2}$. The symmetric linear systems arising in the Newton steps (see (41)) are solved by the conjugate residual (CR) method. At each Newton iteration k, the CR method is initialized from the zero vector, and the iterations are terminated once the relative residual satisfies

$$\frac{\|r_L^{(k)}\|_2}{\|g^{(k)}\|_2} < 10^{-6},$$

where $r_L^{(k)} = g^{(k)} - H_\kappa^{(k)} x_L^{(k)}$ denotes the residual of the CR method at the L-th inner iteration of the k-th Newton step, and $x_L^{(k)}$ is the corresponding CR iterate. The maximum number of CR iterations is set to 1000.

Unless otherwise specified, the global stopping criterion for all compared Riemannian optimization algorithms is based on the Riemannian gradient norm:

$$\| \operatorname{grad} f(Z^{(k)}) \| < \varepsilon, \quad \text{with } \varepsilon = 10^{-6}.$$

All algorithmic parameters in the Riemannian curvilinear search method (Algorithm 2, hereafter denoted by `OptGBB`) are chosen in accordance with the settings recommended by Wen and Yin [31], namely $\delta = 0.2$, $\varrho = 0.85$, $\rho = 10^{-4}$, $\gamma_{\min} = 10^{-20}$, and $\gamma_{\max} = 10^{20}$. The maximum number of iterations for `OptGBB` is set to 1000.

4.1 An application example: ten colas data

We first examine the performance of the proposed `Hybrid-Newton` algorithm on a classical dataset concerning the perceived similarity of ten cola beverages, evaluated by ten subjects. This dataset, originally reported in Schiffman et al. [20] and subsequently analyzed in Trendafilov [28] and Zhou et al. [33], is a standard benchmark for the INDSCAL model. In this setting, $n = 10$ (stimuli) and $m = 10$ (subjects). The stimuli are Diet Pepsi, RC Cola, Yukon, Dr. Pepper, Shasta, Coca-Cola, Diet Dr. Pepper, Tab, Pepsi-Cola, and Diet Rite. The participants, aged 18–21, comprise an equal number of male and female university students. We target a three-dimensional configuration, that is, we set $r = 3$. Each subject's dissimilarity matrix contains pairwise judgments on a scale from 0 ("identical") to 100 ("completely different"). Prior to analysis, all matrices are normalized by dividing their entries by 100. The resulting ten normalized symmetric dissimilarity matrices serve as the input to the O-INDSCAL fitting procedure.

For both random and structured initializations, we apply the proposed `Hybrid-Newton` algorithm, the `Inexact-Newton` and `Modified-RTR` methods developed in our earlier work [33, 32] to solve the O-INDSCAL fitting problem (3). The corresponding numerical results are reported in Table 1, where "`CPU(s)`" denotes the total computation time, "$\|\text{grad} f\|$" the final Riemannian gradient norm, and "`Fvalue`" the objective function value at termination. For the `Hybrid-Newton` algorithm, the "`IT`" column lists, in order, the number of iterations in the curvilinear-search phase (`OptGBB`), the total number of outer iterations, and–in parentheses–the cumulative number of CR iterations used to solve the symmetric linear systems $H_\phi^{(k)} x = g^{(k)}$ arising in the Newton step. For the `Inexact-Newton` algorithm of Zhou et al. [33] and the suitably adapted `Modified-RTR` algorithm of Zhou and Li [32], "`IT`" records the number of outer iterations required to satisfy the stopping criteria, with, in parentheses, the total number of inner CG iterations for the inexact Riemannian Newton equations and the cumulative number of tCG iterations for the trust-region subproblems.

Figure 1 illustrates the evolution of the objective value $f(Z^{(k)})$ (left) and the Riemannian gradient norm $\log_{10} \| \operatorname{grad} f(Z^{(k)}) \|$ (right) with respect to the outer iteration index k for the `Inexact-Newton`, `Hybrid-Newton`, and `Modified-RTR` methods. Figure 2 (left) reports the number of inner iterations of the CG method in `Inexact-Newton` and of the tCG method in `Modified-RTR` as functions of the outer iteration index. The plots clearly indicate that, for both methods, the inner iteration counts adapt to the magnitude of the current gradient: when the iterates are far from the solution (and

Table 1 Comparison of `Hybrid-Newton`, `Inexact-Newton`, and `Modified-RTR` for different initial points

Algorithms	CPU(s)	IT	$\|\mathrm{grad} f\|$	Fvalue
Randomly generated initial point $Z^{(0)} = (Q^{(0)}, D_1^{(0)}, \ldots, D_{10}^{(0)})$				
`Hybrid-Newton`	0.0182	20/7 (239)	3.00×10^{-7}	2.3965
`Inexact-Newton`	0.1412	23 (261)	1.87×10^{-7}	2.3965
`Modified-RTR`	0.0335	18 (241)	1.46×10^{-7}	2.3965
Specially constructed initial point $Z^{(0)} = (Q^{(0)}, D_1^{(0)}, \ldots, D_{10}^{(0)})$				
`Hybrid-Newton`	0.0194	31/6 (214)	1.23×10^{-10}	2.3965
`Inexact-Newton`	0.0400	10 (180)	9.54×10^{-7}	2.3965
`Modified-RTR`	0.0375	12 (291)	1.57×10^{-9}	2.3965

the gradient norm is relatively large), only a few inner iterations are needed, whereas the number of inner iterations increases as the iterates approach the optimum so as to enhance the accuracy of the subproblem solutions. Table 4 provides detailed numerical statistics for the CR solver employed within the Newton steps. The first column records the Newton iteration index, and the second confirms the symmetry of the coefficient matrix $H_\phi^{(k)}$. The remaining columns, for each Newton step, report the number of CR iterations ("`IT`") and the final relative residual norm $\|r_L^{(k)}\|_2/\|g^{(k)}\|_2$. From Table 4, we further observe that, within the `Hybrid-Newton` algorithm, the number of inner CR iterations varies only moderately across outer iterations and remains uniformly small and stable, indicating that the resulting symmetric linear systems are numerically well conditioned and readily solvable. Figure 2 (right) depicts, under the stringent termination criterion $\|\mathrm{grad}\, f(Z^{(k)})\| \le 10^{-8}$, the convergence histories of the proposed `Hybrid-Newton` method and the `OptGBB` scheme. The horizontal axis corresponds to the iteration count, while the vertical axis shows the base-10 logarithm of the Riemannian gradient norm. The convergence profile of `Hybrid-Newton` exhibits two distinct stages: an initial phase of moderate decrease (coinciding with that of the curvilinear search) followed by a pronounced drop in the gradient norm after the switch to the Newton phase. By contrast, the standalone curvilinear method converges more slowly. These observations clearly confirm the effectiveness of the hybrid switching strategy in accelerating convergence once the iterates enter a neighborhood of the solution.

Table 2 Iteration history of the CR method for solving $H_\phi^{(k)} x = g^{(k)}$

k	$\|H_\phi^{(k)} - H_\phi^{(k)^T}\|$	CR method for $H_\phi^{(k)} x = g^{(k)}$	
		IT	$\|r_L^{(k)}\|_2/\|g^{(k)}\|_2$
1	1.14×10^{-15}	38	6.5762×10^{-7}
2	1.51×10^{-15}	39	5.8066×10^{-7}
3	1.31×10^{-15}	40	7.0879×10^{-7}
4	1.24×10^{-15}	42	5.7148×10^{-7}
5	1.17×10^{-15}	43	8.1348×10^{-7}
6	1.44×10^{-15}	37	5.7362×10^{-7}

Using the same random initialization and stopping criteria, all three algorithms converge to the same limit point. The final solution $Z^* = (Q^*, D_1^*, \ldots, D_{10}^*)$ yields the

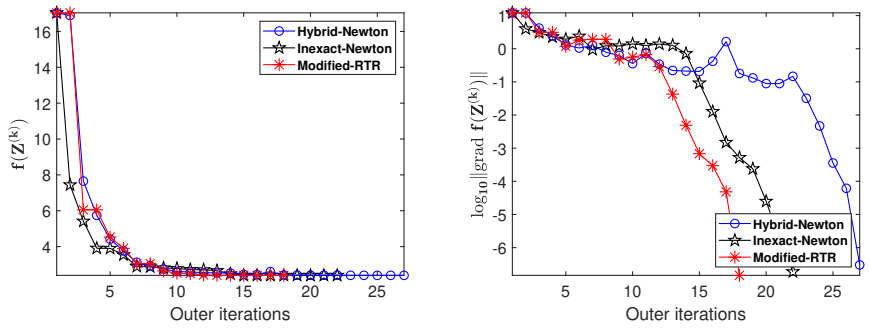


Fig. 1 Evolution of the objective value $f(Z^{(k)})$ (left) and the Riemannian gradient norm $\log_{10} \| \operatorname{grad} f(Z^{(k)})\|$ (right) with respect to the outer iteration index k for the `Inexact-Newton`, `Hybrid-Newton`, and `Modified-RTR` methods.

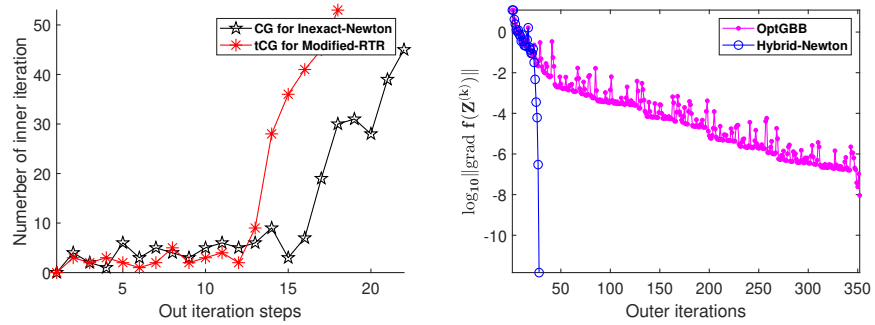


Fig. 2 Left: number of inner iterations of the CG method in `Inexact-Newton` and of the tCG method in `Modified-RTR` versus the outer iteration index. Right: evolution of $\log_{10} \|\operatorname{grad} f(Z^{(k)})\|$ with respect to the outer iteration index for the proposed `Hybrid-Newton` and `OptGBB` methods.

following loading matrix Q^* (rows corresponding to the three dimensions):

$$Q^* = \begin{pmatrix} -0.3707 & -0.3083 & 0.5970 & -0.2078 & 0.0136 & 0.3357 & -0.0396 & 0.3925 & -0.2911 & -0.1214 \\ -0.1696 & -0.2091 & 0.0644 & 0.6410 & -0.1421 & -0.1203 & 0.5792 & -0.1760 & -0.2653 & -0.2024 \\ 0.3442 & -0.2933 & -0.2119 & -0.2468 & -0.2975 & -0.1724 & 0.3248 & 0.3935 & -0.3067 & 0.4659 \end{pmatrix}^T.$$

The associated salience matrices D_i^* are summarized in the matrix below, where each column contains the square roots of the diagonal entries of D_i^* for a given subject:

$$D^* = \begin{pmatrix} -0.4493 & 0.7917 & -0.6467 & 0.7859 & -0.4757 & 0.5299 & 0.6978 & 0.6092 & -0.7756 & 0.4440 \\ -0.7336 & 0.9514 & 1.0696 & 0.6056 & 0.0000 & 0.0000 & -1.0556 & -0.8312 & 0.4749 & 0.9881 \\ 1.0407 & -0.4787 & -0.4637 & -1.0353 & -1.1377 & -1.0883 & 0.5105 & -0.6636 & -0.9895 & 0.4634 \end{pmatrix}^T.$$

The resulting configuration is in close agreement with that reported in Trendafilov [28], differing only in the signs of certain rows of Q^{*T} and D^{*T}. Such sign changes reflect the intrinsic non-uniqueness of MDS-type solutions, which are invariant under orthogonal transformations (rotations and reflections) that preserve interpoint distances. The loading matrix Q^* specifies the coordinates of the stimuli in the common latent space, whereas the diagonal matrices D_i^* encode subject-specific saliences along each dimension. These components are visualized in Figure 5.

To further assess the model fit, we compute the SSTRESS(1) index [8], defined as the normalized residual

$$\sum_{i=1}^{m} \frac{\left\| \frac{1}{2} J_n (D_i \odot D_i) J_n + Q D_i^2 Q^T \right\|}{\left\| \frac{1}{2} J_n (D_i \odot D_i) J_n \right\|} = 6.3397.$$

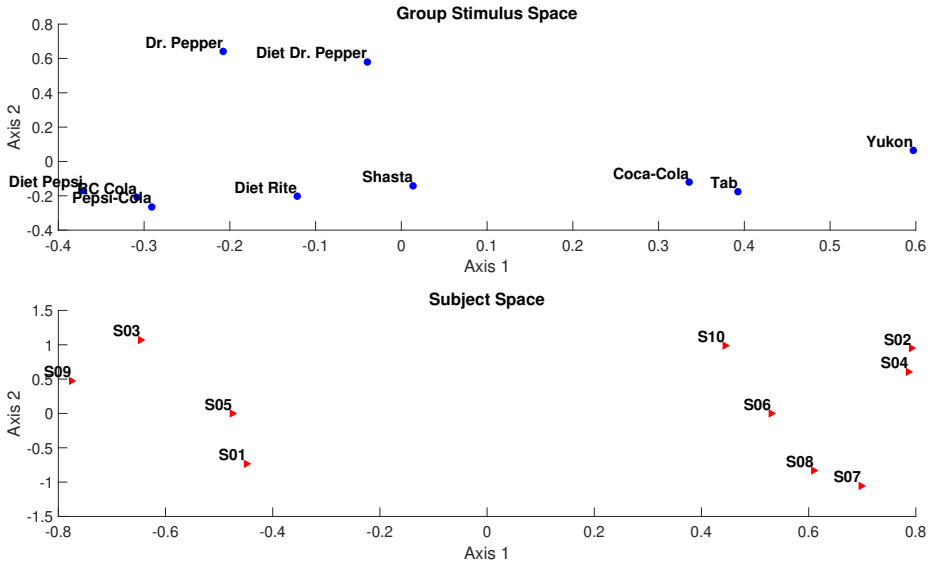


Fig. 3 INDSCAL analysis of ten colas data [20]

4.2 Comparison with `Inexact-Newton` and `Modified-RTR` from our earlier work

In this subsection, we further evaluate the proposed `Hybrid-Newton` algorithm on medium- and large-scale synthetic datasets, and compare its performance with the `Inexact-Newton` and `Modified-RTR` methods from our earlier work [33, 32], as well as with a direct application of the Riemannian curvilinear search scheme `OptGBB`. The comparison is conducted on the three classes of synthetic datasets introduced above: (i) partially structured data with indefinite saliences (IND), (ii) partially structured data with nonnegative definite saliences (NND), and (iii) fully unstructured random data (RAND). For the NND datasets, the number of subjects is fixed at $m = 300$, while the number of stimuli takes values $n \in \{30, 60, 120, 200\}$. The embedding dimension is chosen as $r \in \{3, 5\}$. In the IND experiments, the number of subjects is fixed at $m = 500$, the number of stimuli again takes values $n \in \{30, 60, 120, 200\}$, and the embedding dimension is set to $r \in \{2, 5\}$. For the RAND datasets, we consider moderate problem sizes with $m \leq 100$ and $n \leq 30$, and we take $r \in \{2, 3\}$.

All four algorithms are executed with the same structured initializations and stopping criteria, and the resulting numerical performance is summarized in Table 3. The

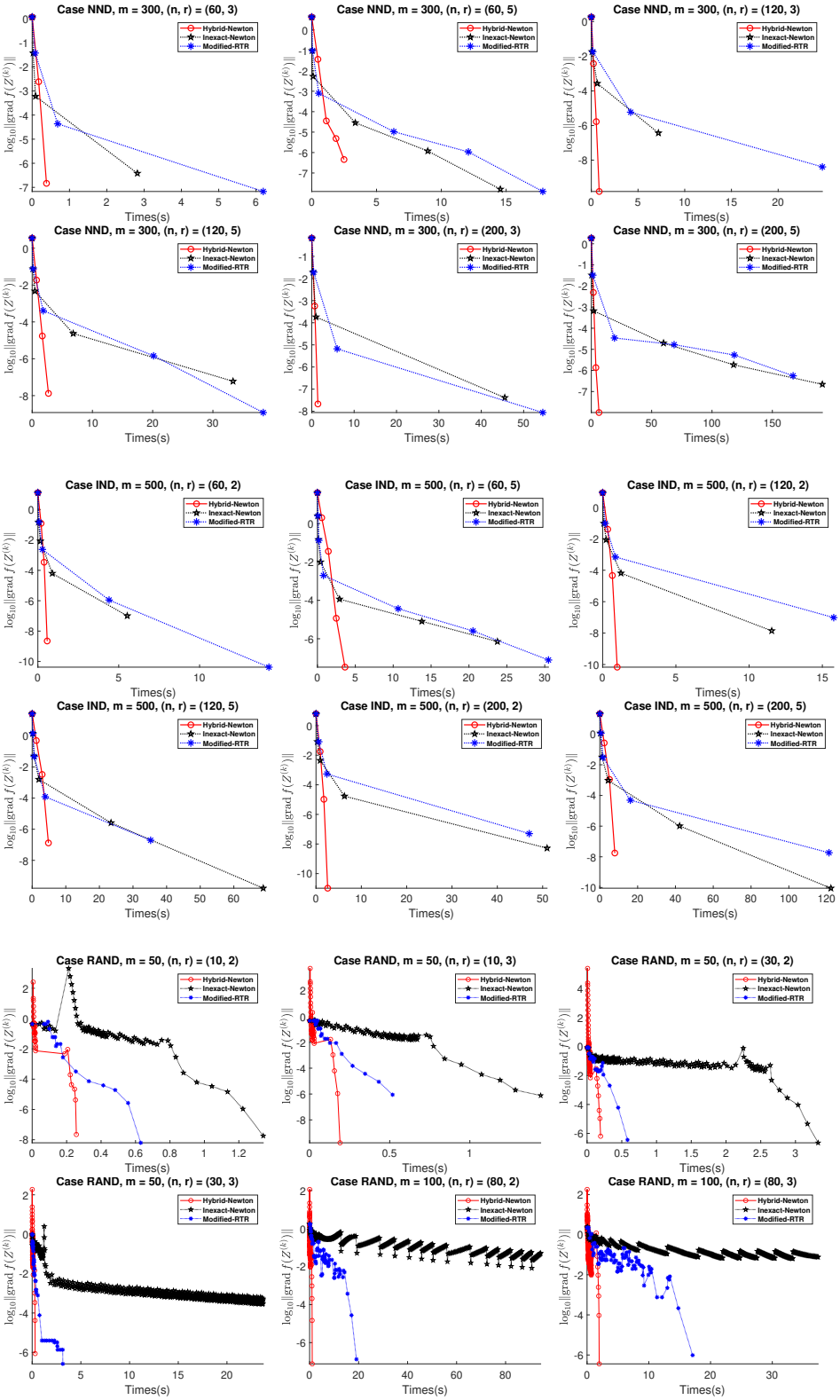


Fig. 4 Comparison among the performances of the proposed `Hybrid-Newton` algorithm with the `Inexact-Newton` and `Modified-RTR` methods developed in our earlier work [33, 32] for various problem dimensions and datasets.

Table 3 Comparison results for the proposed `Hybrid-Newton` algorithm with the `Inexact-Newton` and `Modified-RTR` methods from our earlier work.

Algorithms	m, (n,r)	CPU(s)	IT	$\|\text{grad} f\|$	Fvalue	m, (n,r)	CPU(s)	IT	$\|\text{grad} f\|$	Fvalue
Partially structured with *nnd* weight matrices (NND)										
Hybrid-Newton	300, (30,3)	0.448	5(1076)	2.81×10^{-7}	0.896	300, (30,5)	2.256	7(2451)	3.42×10^{-9}	1.697
Inexact-Newton		3.830	7(644)	2.04×10^{-7}	0.896		11.624	9(1705)	2.58×10^{-7}	1.697
Modified-RTR		4.082	6(1098)	2.72×10^{-8}	0.896		7.324	7(1780)	4.99×10^{-7}	1.697
OptGBB		2.828	1020	9.82×10^{-7}	0.896		7.821	2793	9.40×10^{-7}	1.697
Hybrid-Newton	300, (60,3)	0.388	3(544)	1.47×10^{-7}	0.888	300, (60,5)	2.499	5(2151)	4.56×10^{-7}	1.678
Inexact-Newton		2.820	5(285)	3.83×10^{-7}	0.888		14.555	7(1296)	1.60×10^{-8}	1.678
Modified-RTR		6.193	4(539)	6.76×10^{-8}	0.888		17.831	6(1539)	1.24×10^{-8}	1.678
OptGBB		3.907	974	9.50×10^{-7}	0.888		12.905	3056	9.87×10^{-7}	1.678
Hybrid-Newton	300, (120,3)	0.852	4(855)	1.55×10^{-10}	0.879	300, (120,5)	2.711	4(1263)	1.32×10^{-8}	1.659
Inexact-Newton		7.185	5(284)	3.70×10^{-7}	0.879		33.338	6(628)	5.98×10^{-8}	1.659
Modified-RTR		24.732	4(598)	4.06×10^{-9}	0.879		38.354	5(1046)	1.21×10^{-9}	1.659
OptGBB		6.050	593	9.90×10^{-7}	0.879		33.546	1806	9.70×10^{-7}	1.659
Hybrid-Newton	300, (200,3)	1.460	3(584)	2.14×10^{-8}	0.877	300, (200,5)	6.393	4(2160)	1.01×10^{-8}	1.658
Inexact-Newton		45.543	5(419)	4.11×10^{-8}	0.877		191.439	7(1515)	2.19×10^{-7}	1.658
Modified-RTR		54.517	4(557)	8.74×10^{-9}	0.877		167.188	6(1596)	5.56×10^{-7}	1.658
OptGBB		29.716	652	9.95×10^{-7}	0.877		121.451	2722	9.65×10^{-7}	1.658
Partially structured with *ind* weight matrices (IND)										
Hybrid-Newton	500, (30,2)	0.494	4(539)	9.48×10^{-8}	126.652	500, (30,5)	4.728	9(1790)	2.19×10^{-9}	316.197
Inexact-Newton		2.725	6(288)	8.89×10^{-7}	126.652		11.372	9(1059)	4.12×10^{-8}	316.197
Modified-RTR		4.322	5(732)	5.84×10^{-9}	126.652		12.575	8(1881)	5.94×10^{-9}	316.197
OptGBB		2.991	674	9.99×10^{-7}	126.652		6.051	1376	1.00×10^{-6}	316.197
Hybrid-Newton	500, (60,2)	0.584	4(369)	2.23×10^{-9}	126.645	500, (60,5)	3.667	5(1270)	3.32×10^{-8}	316.217
Inexact-Newton		5.527	6(242)	1.02×10^{-7}	126.645		23.757	8(997)	7.12×10^{-7}	316.217
Modified-RTR		14.288	5(722)	4.23×10^{-11}	126.645		30.495	7(1535)	7.83×10^{-8}	316.217
OptGBB		2.964	319	9.36×10^{-7}	126.645		23.187	2533	9.97×10^{-7}	316.217
Hybrid-Newton	500, (120,2)	1.001	4(394)	6.82×10^{-11}	126.631	500, (120,5)	4.874	4(634)	1.31×10^{-7}	316.038
Inexact-Newton		11.543	6(245)	1.40×10^{-8}	126.631		68.752	7(763)	1.62×10^{-10}	316.038
Modified-RTR		15.777	4(281)	9.66×10^{-8}	126.631		35.242	5(556)	1.95×10^{-7}	316.038
OptGBB		4.663	242	8.92×10^{-7}	126.631		23.730	785	9.89×10^{-7}	316.038
Hybrid-Newton	500, (200,2)	2.586	4(415)	1.03×10^{-11}	126.616	500, (200,5)	7.967	4(492)	1.77×10^{-8}	316.055
Inexact-Newton		50.973	6(279)	5.22×10^{-9}	126.616		122.238	7(614)	8.99×10^{-11}	316.055
Modified-RTR		47.039	4(284)	5.02×10^{-8}	126.616		121.238	5(590)	1.84×10^{-8}	316.055
OptGBB		19.554	260	9.75×10^{-7}	126.616		49.889	643	9.45×10^{-7}	316.055
Completely random data sets (RAND)										
Hybrid-Newton	50, (10,2)	0.238	30/8(668)	2.23×10^{-8}	10.687	50, (10,3)	0.163	51/6(613)	1.68×10^{-10}	9.954
Inexact-Newton		1.342	83(958)	1.82×10^{-8}	10.687		1.442	92(1049)	7.68×10^{-7}	9.953
Modified-RTR		0.631	18(751)	6.30×10^{-9}	10.687		0.518	22(740)	8.81×10^{-7}	9.953
OptGBB		0.392	534	9.91×10^{-7}	10.687		0.229	338	9.78×10^{-7}	9.953
Hybrid-Newton	50, (30,2)	0.148	65/8(1010)	6.41×10^{-7}	116.347	50, (30,3)	0.194	111/6(961)	8.68×10^{-7}	114.946
Inexact-Newton		3.326	150(1803)	2.24×10^{-7}	116.323		23.727	500(13957)	5.23×10^{-4}	114.944
Modified-RTR		0.583	29(495)	3.57×10^{-7}	116.323		3.148	47(3046)	2.60×10^{-7}	114.944
OptGBB		0.745	976	9.79×10^{-7}	116.347		0.548	824	7.09×10^{-7}	114.944
Hybrid-Newton	100, (50,2)	0.486	170/7(1814)	1.08×10^{-8}	669.266	100, (50,3)	0.494	110/6(1289)	8.57×10^{-8}	666.738
Inexact-Newton		20.935	500(3785)	2.31×10^{-2}	669.317		39.990	500(8895)	2.61×10^{-2}	666.740
Modified-RTR		12.578	49(3051)	1.04×10^{-8}	669.257		5.818	40(1146)	1.53×10^{-7}	666.734
OptGBB		2.100	1285	9.83×10^{-7}	669.257		1.386	772	9.89×10^{-7}	666.734
Hybrid-Newton	100, (80,2)	0.775	131/10(2932)	7.63×10^{-8}	1732.191	100, (80,3)	1.284	265/10(2781)	3.61×10^{-7}	1728.764
Inexact-Newton		94.322	500(12831)	5.06×10^{-2}	1732.191		37.419	500(3890)	7.65×10^{-2}	1729.039
Modified-RTR		19.123	80(2487)	1.33×10^{-7}	1732.153		17.099	102(2267)	9.74×10^{-7}	1728.837
OptGBB		4.368	1731	7.30×10^{-7}	1732.153		3.762	1365	7.05×10^{-7}	1728.741

quantities "`CPU(s)`", "`IT`", "$\|\text{grad} f\|$", and "`Fvalue`" are defined as in Table 1. We emphasize that, when the slice matrices S_i are generated under the partially structured IND or NND datasets, the structured initialization produces starting points that are already close to a solution. In this regime, `Hybrid-Newton` can be applied directly, without a preliminary `OptGBB` phase to refine the initial iterate. By contrast, when the slice matrices S_i are generated in a fully random manner (RAND), the same structured initialization may yield starting points of relatively poor quality. In that case, we first employ `OptGBB` to obtain a reasonably accurate initial iterate, and then switch to `Hybrid-Newton` to exploit its fast local convergence.

Figure 4 presents, for the three second-order methods `Inexact-Newton`, `Modified-RTR`, and `Hybrid-Newton`, the evolution of the Riemannian gradient norm as a function of CPU time. From the numerical results in Table 3 and the convergence profiles in Figure 4, we observe that, for both IND and NND data, all three second-order algorithms successfully solve the O-INDSCAL fitting problem with only a small number of outer

iterations. Nevertheless, `Hybrid-Newton` consistently attains the lowest overall computational time. The curves in Figure 4 also clearly show that the per-iteration cost of `Hybrid-Newton` is markedly lower than that of `Inexact-Newton` and `Modified-RTR`, resulting in a visibly faster decay of the gradient norm with respect to CPU time.

For the RAND datasets, `Hybrid-Newton` remains robust and delivers high-accuracy solutions once an adequate starting point has been generated by `OptGBB`. In contrast, the performance of `Inexact-Newton` deteriorates when the initial iterate is not sufficiently close to a solution: the inexactness condition for the Riemannian Newton step may fail repeatedly, forcing the method to revert to pure gradient steps over many iterations and thereby substantially slowing convergence. This behavior underscores the advantage of the proposed hybrid strategy, which combines the global robustness of the curvilinear search with the rapid local convergence of an efficiently implemented Riemannian Newton phase.

4.3 Comparison with the classical projected gradient approach

In this subsection, we compare the proposed `Hybrid-Newton` algorithm with the classical projected gradient (PG) method, originally introduced in Trendafilov [26, 27]. The PG approach is a widely used technique for product-manifold constrained matrix optimization, particularly in multivariate data analysis [12, 11, 23, 25, 24, 30, 28]. It can be implemented either as a discrete-time iterative scheme or as a continuous-time dynamical system governed by ordinary differential equations (ODEs) derived from projected gradient flows.

For a general constrained optimization problem of the form $\min_{X\in\mathcal{M}} E(X)$, the discrete-time PG method generates a sequence $\{X_t\}$ via

$$X_{t+1} = \pi_{\mathcal{M}}\big(X_t - h_t \nabla E(X)\big|_{X=X_t}\big),$$

where h_t denotes the step size and $\pi_{\mathcal{M}}(\cdot)$ is the orthogonal projection onto the feasible manifold $\mathcal{M}$. The continuous-time counterpart evolves along the negative projected gradient direction and is described by the differential equation

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = -\pi_{\mathcal{M}}\big(\nabla E(X(t))\big), \quad X(0) = X_0 \in \mathcal{M}.$$

When applied to the O-INDSCAL problem (3), this framework leads to the following system of ODEs:

$$\begin{cases} \dfrac{\mathrm{d}Q}{\mathrm{d}t} = \mathbf{P}_Q \left(\displaystyle\sum_{i=1}^{m} \big(S_i Q D_i^2 - Q D_i^4\big) \right), \\ \dfrac{\mathrm{d}D_i}{\mathrm{d}t} = D_i \odot \big(Q^T S_i Q - D_i\big), \quad i = 1, \ldots, m, \end{cases}$$

where $\mathbf{P}_Q$ denotes the orthogonal projection onto the tangent space at Q of the Stiefel manifold.

In our implementation, we employ MATLAB's built-in ODE solvers `ode45`, `ode113`, and `ode15s` from the ODE Suite [21]. The choice of solver depends on the stiffness and complexity of the underlying dynamical system: `ode45` is typically preferred for nonstiff problems, `ode113` is advantageous when higher accuracy is required or when function evaluations are expensive, and `ode15s` is well-suited for stiff or differential–algebraic

Table 4 Comparison results for the proposed `Hybrid-Newton` algorithm with `PG-ODEs`

Algorithms	m, (n,r)	CPU(s)	IT	$\|\text{grad} f\|$	Fvalue	m, (n,r)	CPU(s)	IT	$\|\text{grad} f\|$	Fvalue
Partially structured with *nnd* weight matrices (NND)										
Hybrid-Newton	80, (30,2)	0.028	3(288)	7.56×10^{-11}	0.156	50, (30,3)	0.179	19(1715)	1.59×10^{-7}	0.164
PG-ode15s		1.946	275	7.95×10^{-7}	0.156		11.510	936	9.00×10^{-7}	0.156
PG-ode45		4.894	1112	7.95×10^{-7}	0.156		13.177	4640	9.00×10^{-7}	0.156
PG-ode113		3.555	2341	7.95×10^{-7}	0.156		9.049	9750	9.00×10^{-7}	0.156
Hybrid-Newton	80, (60,2)	0.052	3(264)	3.36×10^{-11}	0.156	50, (60,3)	0.178	5(596)	4.02×10^{-10}	0.155
PG-ode15s		5.261	237	9.55×10^{-7}	0.156		21.564	447	9.18×10^{-7}	0.155
PG-ode45		11.768	839	9.55×10^{-7}	0.156		22.946	2482	9.18×10^{-7}	0.155
PG-ode113		8.797	1789	9.55×10^{-7}	0.156		14.906	5183	9.18×10^{-7}	0.155
Hybrid-Newton	50, (150,3)	0.985	9(1078)	2.67×10^{-10}	0.153	50, (200,5)	2.398	7(895)	9.67×10^{-7}	0.295
PG-ode15s		195.810	682	9.80×10^{-7}	0.153		1149.042	630	9.97×10^{-7}	0.295
PG-ode45		228.864	4273	9.80×10^{-7}	0.153		475.429	4935	9.97×10^{-7}	0.295
PG-ode113		189.275	8838	9.80×10^{-7}	0.153		213.977	10259	9.98×10^{-7}	0.295
Partially structured with indefinite weight matrices (IND)										
Hybrid-Newton	80, (20,2)	0.122	4(416)	8.26×10^{-10}	22.195	50, (50,5)	0.352	7(504)	3.58×10^{-8}	31.933
PG-ode15s		21.580	951	9.55×10^{-7}	22.195		71.850	614	9.65×10^{-7}	31.915
PG-ode45		37.474	6904	9.55×10^{-7}	22.195		25.032	1668	9.65×10^{-7}	31.915
PG-ode113		25.142	14304	9.55×10^{-7}	22.195		11.808	3360	9.65×10^{-7}	31.915
Hybrid-Newton	80, (100,2)	0.475	4(440)	5.76×10^{-9}	22.108	50, (100,5)	0.380	4(382)	4.42×10^{-9}	31.874
PG-ode15s		47.547	413	8.88×10^{-7}	22.108		57.561	468	7.56×10^{-7}	31.874
PG-ode45		47.485	2098	8.88×10^{-7}	22.108		13.565	921	7.56×10^{-7}	31.874
PG-ode113		33.851	4398	8.88×10^{-7}	22.108		9.507	1893	7.56×10^{-7}	31.874
Hybrid-Newton	80, (200,2)	1.435	4(448)	1.52×10^{-8}	22.092	50, (200,5)	1.352	4(392)	2.82×10^{-9}	31.888
PG-ode15s		51.018	240	8.37×10^{-7}	22.092		255.902	388	6.51×10^{-7}	31.888
PG-ode45		58.354	553	8.37×10^{-7}	22.092		52.471	801	6.51×10^{-7}	31.888
PG-ode113		42.454	1191	8.48×10^{-7}	22.092		33.343	1646	6.51×10^{-7}	31.888
Completely random data sets (RAND)										
Hybrid-Newton	30, (30,2)	0.079	31/6(431)	1.21×10^{-7}	69.693	30, (30,3)	0.039	42/6(438)	8.98×10^{-8}	68.709
PG-ode15s		139.166	9167	9.98×10^{-7}	69.692		14.024	2245	9.79×10^{-7}	68.709
PG-ode45		54.995	10910	9.98×10^{-7}	69.692		3.329	1919	9.79×10^{-7}	68.709
PG-ode113		11.360	19673	9.98×10^{-7}	69.692		1.889	3179	9.79×10^{-7}	68.709
Hybrid-Newton	20, (50,2)	0.043	49/5(311)	1.30×10^{-9}	131.435	20, (50,3)	0.071	66/7(639)	3.18×10^{-8}	130.373
PG-ode15s		7.586	1788	9.29×10^{-7}	131.435		40.864	3986	9.84×10^{-7}	130.373
PG-ode45		4.656	1890	9.29×10^{-7}	131.435		11.970	4729	9.84×10^{-7}	130.373
PG-ode113		2.802	3230	9.29×10^{-7}	131.435		8.226	8984	9.84×10^{-7}	130.373
Hybrid-Newton	20, (60,2)	0.060	53/6(383)	5.58×10^{-7}	190.612	20, (60,3)	0.092	75/6(525)	9.95×10^{-11}	189.276
PG-ode15s		10.300	1730	9.66×10^{-7}	190.612		27.138	2318	9.42×10^{-7}	189.276
PG-ode45		6.916	2011	9.66×10^{-7}	190.612		9.380	2763	9.42×10^{-7}	189.276
PG-ode113		4.117	3494	9.66×10^{-7}	190.612		5.932	4936	9.42×10^{-7}	189.276

systems. Detailed descriptions of these solvers can be found in Shampine and Reichelt [21]. For all experiments, both the absolute and relative error tolerances are set to 10^{-12} in order to ensure high numerical accuracy in tracing the solution trajectories. Although such stringent tolerances are more conservative than those commonly used in practice, they are adopted here to facilitate precise performance comparisons.

Integration outputs are sampled every 10 time units, and the integration is terminated automatically once the relative decrease in the objective function between two consecutive samples falls below 10^{-6}, which provides a practical indication of near-convergence. This stopping rule is intentionally looser than the termination criterion used for the proposed `Hybrid-Newton` algorithm, following the guidelines in Takane et al. [22], Loisel and Takane [17].

In the numerical experiments, the number of subjects satisfies $m \leq 80$, while the number of stimuli satisfies $n \leq 200$. The embedding dimension is chosen as $r \in \{2, 3, 5\}$. Results are reported for the three dataset types NND, IND, and RAND. For each dataset and a common initial point, we run the proposed `Hybrid-Newton` algorithm and the PG method implemented with each of the MATLAB ODE solvers, yielding three variants denoted by `PG-ode45`, `PG-ode113`, and `PG-ode15s`. The corresponding numerical results are presented in Table 4 for different dimensional configurations, and the performance indicators are defined consistently with those in Table 1.

The results in Table 4 show that the proposed `Hybrid-Newton` algorithm consistently outperforms the `PG-ODE` variants in terms of computation time, while achieving essentially the same final objective values. This advantage becomes particularly pronounced in higher-dimensional settings. Notably, despite the use of a more relaxed stopping criterion for the `PG-ODE` methods, the hybrid approach delivers superior efficiency and more robust convergence behavior.

4.4 Comparison with Manopt-based Riemannian optimization methods

In a recent study, Trendafilov and Gallo [29] proposed a unified geometric framework for classical multivariate data analysis models by exploiting the structure of matrix manifolds. Within this framework, the O-INDSCAL problem is formulated as an optimization task over a product manifold and solved using the Manopt toolbox [7]. In this subsection, we benchmark the proposed `Hybrid-Newton` algorithm against several state-of-the-art Riemannian optimization algorithms implemented in Manopt. The compared methods are: Riemannian steepest descent (`Manopt-RSD`), Riemannian conjugate gradient (`Manopt-RCG`), Barzilai–Borwein (`Manopt-RBB`), trust-region (`Manopt-RTR`), limited-memory BFGS (`Manopt-RLBFGS`), and adaptive regularization by cubics (`Manopt-ARC`). All solvers are used with Manopt's default parameters and stopping criteria, and the maximum number of iterations is capped at 10,000.

To ensure a fair comparison, each algorithm is initialized with the same starting point as the `Hybrid-Newton` method. Moreover, we provide Manopt with the exact expressions for the Riemannian gradient and Hessian, as given in (12) and (18), respectively. While Manopt approximates the Hessian via finite differences when no explicit expression is supplied, our experiments consistently use the analytic Hessian to align the accuracy of curvature information across all solvers.

Table 5 reports the results for the three dataset types IND, NND, and RAND. The performance indicators are defined consistently with those in Table 1. Figure 5 displays the convergence trajectories of each algorithm, plotting $\log_{10} \| \operatorname{grad} f(Z^{(k)}) \|$ versus wall-clock time. The results show that the proposed `Hybrid-Newton` algorithm systematically outperforms the Manopt-based solvers in terms of computational efficiency. As illustrated in Figure 5, both `Manopt-RBB` and `Manopt-RSD` exhibit pronounced slowdown near critical points, reflecting the intrinsic limitations of first-order methods in achieving fast local convergence. Furthermore, Table 5 indicates that `Manopt-RLBFGS`, while quite effective in reducing the gradient norm, tends to incur higher computational costs. This observation is consistent with the Manopt documentation for the file "`rlbfgs.m`", which notes that the default memory parameter is set to 30 and that vector transport is performed via projection. Consequently, each iteration of `Manopt-RLBFGS` involves approximately 30 vector transport operations, leading to a substantial per-iteration overhead. Taken together, these findings confirm the advantages of our hybrid approach, particularly its ability to combine robust global behavior with the rapid local refinement afforded by second-order information.

4.5 Comparison with two state-of-the-art algorithms

In this final set of numerical experiments, we compare the proposed `Hybrid-Newton` algorithm with two established methods for solving the O-INDSCAL problem. The

Table 5 Comparison results for the proposed `Hybrid-Newton` algorithm with existing algorithms in toolbox Manopt

Algorithms	m, (n,r)	CPU(s)	IT	$\|\text{grad}f\|$	Fvalue	m, (n,r)	CPU(s)	IT	$\|\text{grad}f\|$	Fvalue
Partially structured with *nnd* weight matrices (NND)										
Hybrid Newton		0.013	3(237)	5.49×10^{-7}	0.0931		0.052	6(1017)	4.55×10^{-8}	0.2986
Manopt-RCG		32.408	202	5.31×10^{-7}	0.0931		188.565	1150	9.67×10^{-7}	0.2986
Manopt-RBB		44.505	366	9.70×10^{-7}	0.0931		393.061	3096	9.99×10^{-7}	0.2986
Manopt-RSD	50, (15,2)	155.551	2477	5.17×10^{-6}	0.0931	50, (15,5)	673.565	10000	3.58×10^{-5}	0.2986
Manopt-RTR		18.228	3	1.38×10^{-7}	0.0931		79.601	6	4.15×10^{-7}	0.2986
Manopt-RLBFGS		429.524	186	9.41×10^{-7}	0.0931		1140.127	466	7.75×10^{-7}	0.2986
Manopt-ARC		92.379	3	6.48×10^{-10}	0.0931		219.854	6	6.06×10^{-8}	0.2986
Hybrid Newton		0.031	3(240)	2.60×10^{-7}	0.0882		0.120	4(800)	8.61×10^{-8}	0.2967
Manopt-RCG		40.198	244	9.59×10^{-7}	0.0882		98.902	610	7.37×10^{-7}	0.2967
Manopt-RBB		44.699	356	8.78×10^{-7}	0.0882		181.165	1440	9.86×10^{-7}	0.2967
Manopt-RSD	50, (55,2)	258.346	3860	6.30×10^{-6}	0.0882	50, (55,5)	668.522	10000	7.79×10^{-6}	0.2967
Manopt-RTR		21.582	3	1.01×10^{-7}	0.0882		46.563	4	5.18×10^{-7}	0.2967
Manopt-RLBFGS		397.847	173	7.86×10^{-7}	0.0882		970.480	405	8.44×10^{-7}	0.2967
Manopt-ARC		79.635	3	3.35×10^{-9}	0.0882		192.372	4	3.67×10^{-8}	0.2967
Hybrid Newton		0.045	3(204)	1.71×10^{-10}	0.0917		0.282	4(827)	1.16×10^{-8}	0.2981
Manopt-RCG		12.493	74	8.38×10^{-7}	0.0917		138.822	831	8.54×10^{-7}	0.2981
Manopt-RBB		8.619	69	9.62×10^{-7}	0.0917		241.237	1878	9.97×10^{-7}	0.2981
Manopt-RSD	50, (90,2)	30.520	471	2.75×10^{-6}	0.0917	50, (90,5)	394.888	5865	1.41×10^{-5}	0.2981
Manopt-RTR		18.411	3	1.32×10^{-9}	0.0917		64.729	4	2.58×10^{-8}	0.2981
Manopt-RLBFGS		194.627	92	8.16×10^{-7}	0.0917		959.231	399	9.97×10^{-7}	0.2981
Manopt-ARC		58.625	2	2.44×10^{-9}	0.0917		121.298	3	9.49×10^{-7}	0.2981
Partially structured with indefinite weight matrices (IND)										
Hybrid Newton		0.036	4(228)	1.72×10^{-8}	10.4653		0.032	5(359)	1.88×10^{-7}	31.7836
Manopt-RCG		27.571	102	9.57×10^{-7}	10.4653		33.368	208	8.36×10^{-7}	31.7836
Manopt-RBB		24.483	135	7.07×10^{-7}	10.4653		38.485	321	9.58×10^{-7}	31.7836
Manopt-RSD	50, (15,2)	60.246	904	9.97×10^{-7}	10.4653	50, (15,5)	110.960	1889	8.94×10^{-7}	31.7836
Manopt-RTR		21.079	4	2.71×10^{-10}	10.4653		29.765	5	1.95×10^{-8}	31.7836
Manopt-RLBFGS		225.507	105	9.59×10^{-7}	10.4653		382.818	171	7.13×10^{-7}	31.7836
Manopt-ARC		48.257	3	1.26×10^{-8}	10.4653		69.586	4	2.03×10^{-8}	31.7836
Hybrid Newton		0.020	4(207)	3.46×10^{-10}	10.4796		0.037	4(267)	1.62×10^{-7}	31.9429
Manopt-RCG		19.846	122	8.25×10^{-7}	10.4796		29.688	183	9.75×10^{-7}	31.9429
Manopt-RBB		15.593	129	9.03×10^{-7}	10.4796		26.304	216	6.97×10^{-7}	31.9429
Manopt-RSD	50, (35,2)	45.873	774	9.53×10^{-7}	10.4796	50, (35,5)	106.164	1787	9.21×10^{-7}	31.9429
Manopt-RTR		14.675	4	9.92×10^{-9}	10.4796		18.279	4	2.53×10^{-7}	31.9429
Manopt-RLBFGS		226.805	106	9.64×10^{-7}	10.4796		316.432	143	9.66×10^{-7}	31.9429
Manopt-ARC		43.649	3	2.50×10^{-9}	10.4796		48.993	3	1.91×10^{-7}	31.9429
Hybrid Newton		0.042	4(225)	7.02×10^{-12}	10.4733		0.231	4(279)	2.05×10^{-9}	31.8808
Manopt-RCG		18.289	106	9.96×10^{-7}	10.4733		24.126	148	8.64×10^{-7}	31.8808
Manopt-RBB		12.890	104	9.76×10^{-7}	10.4733		19.032	155	9.14×10^{-7}	31.8808
Manopt-RSD	50, (85,2)	39.988	658	9.85×10^{-7}	10.4733	50, (85,5)	79.764	1297	9.78×10^{-7}	31.8808
Manopt-RTR		17.505	4	3.17×10^{-9}	10.4733		28.892	4	1.37×10^{-10}	31.8808
Manopt-RLBFGS		227.955	105	8.26×10^{-7}	10.4733		247.234	115	9.74×10^{-7}	31.8808
Manopt-ARC		58.007	3	2.78×10^{-10}	10.4733		57.078	3	1.31×10^{-8}	31.8808
Completely random data sets (RAND)										
Hybrid Newton		0.053	5/28(2261)	9.97×10^{-10}	6.5709		0.043	13/10(820)	1.39×10^{-7}	68.7091
Manopt-RCG		5.800	94	8.31×10^{-7}	6.3204		11.324	176	9.76×10^{-7}	68.7091
Manopt-RBB		6.540	141	9.89×10^{-7}	6.3204		10.226	222	9.39×10^{-7}	68.7091
Manopt-RSD	30, (10,2)	11.010	480	9.42×10^{-7}	6.3204	30, (30,3)	44.031	1808	9.75×10^{-7}	68.7091
Manopt-RTR		6.294	6	6.25×10^{-8}	6.3204		17.213	11	1.13×10^{-8}	68.7091
Manopt-RLBFGS		45.846	61	9.18×10^{-7}	6.3204		88.312	106	9.61×10^{-7}	68.7091
Manopt-ARC		14.183	8	1.88×10^{-9}	6.3204		22.637	11	4.90×10^{-7}	68.7091
Hybrid Newton		0.076	65/8(1010)	6.41×10^{-7}	116.3467		0.097	111/6(961)	8.68×10^{-7}	114.9462
Manopt-RCG		63.061	374	8.89×10^{-7}	116.3226		75.303	471	7.22×10^{-7}	114.9445
Manopt-RBB		61.901	491	9.36×10^{-7}	116.3226		174.732	664	9.98×10^{-7}	114.9445
Manopt-RSD	50, (30,2)	298.445	4789	9.90×10^{-7}	116.3226	50, (30,3)	657.785	4488	9.68×10^{-7}	114.9445
Manopt-RTR		47.298	27	5.98×10^{-7}	116.3226		264.229	40	3.25×10^{-10}	114.9445
Manopt-RLBFGS		837.902	215	8.37×10^{-7}	116.3226		1674.558	322	9.82×10^{-7}	114.9445
Manopt-ARC		232.617	22	5.89×10^{-7}	116.3226		295.294	52	2.47×10^{-9}	114.9445
Hybrid Newton		0.248	105/8(1129)	5.49×10^{-7}	536.1880		0.411	101/10(2635)	4.34×10^{-8}	534.0240
Manopt-RCG		111.296	283	9.51×10^{-7}	536.1880		281.036	687	9.32×10^{-7}	533.9966
Manopt-RBB		285.386	530	7.81×10^{-7}	536.1880		398.120	1284	8.48×10^{-7}	533.9966
Manopt-RSD	80, (50,2)	1585.000	4438	1.54×10^{-6}	536.1880	80, (50,3)	1509.881	10000	3.36×10^{-6}	533.9966
Manopt-RTR		299.818	14	3.30×10^{-7}	536.1880		318.586	19	1.58×10^{-7}	533.9966
Manopt-RLBFGS		2187.763	233	9.49×10^{-7}	536.1880		5623.949	467	7.49×10^{-7}	533.9966
Manopt-ARC		555.210	11	5.56×10^{-7}	536.1880		567.228	15	2.11×10^{-8}	533.9966

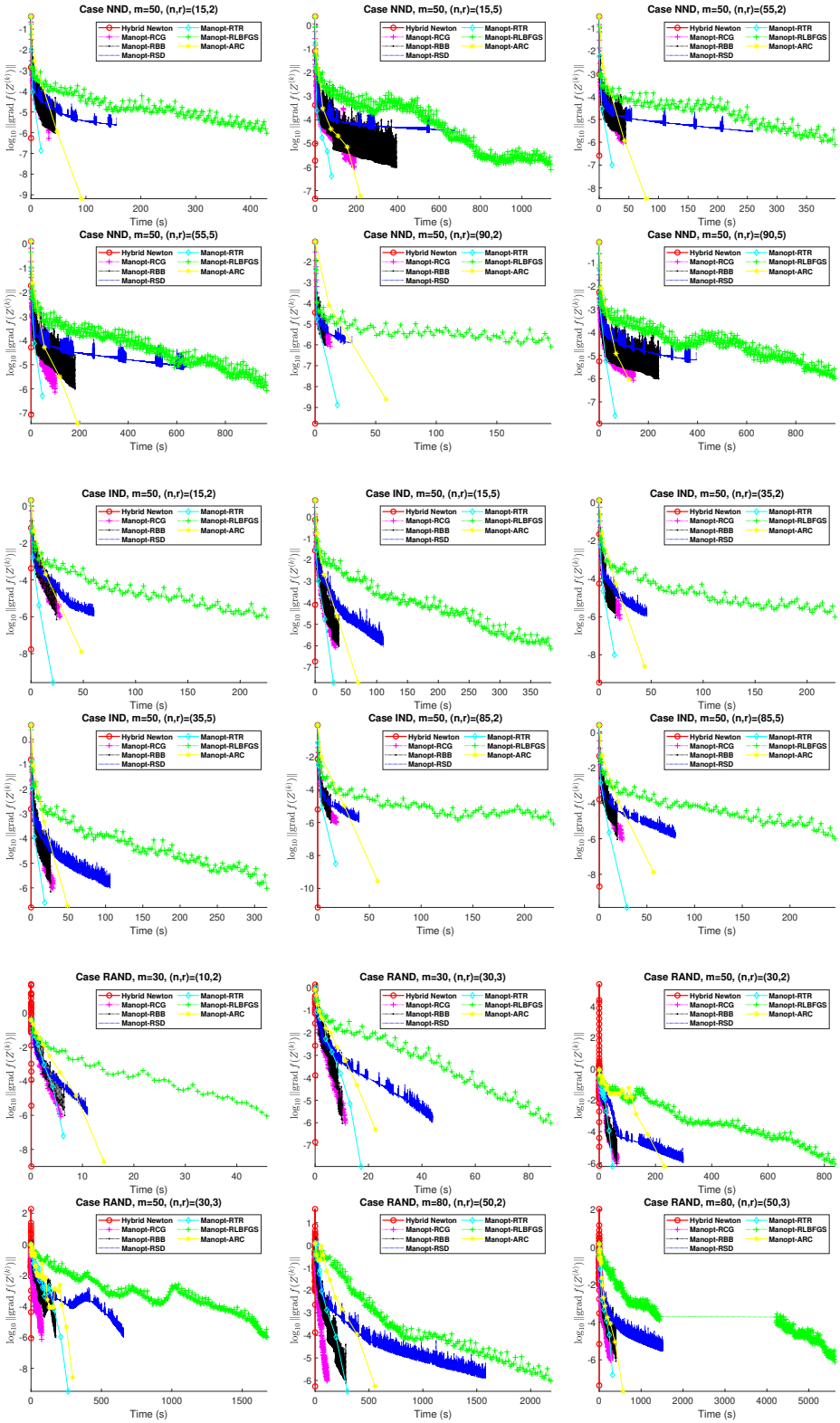


Fig. 5 Comparison among the performances of the proposed `Hybrid-Newton` algorithm and existing algorithms in Matlab toolbox Manopt for various problem dimensions and datasets.

benchmark algorithms are the original approximate alternating least-squares method (`ALS`) and its accelerated variant based on minimal polynomial extrapolation (`MPEALS`), both introduced in Takane et al. [22]. To assess the sensitivity of `MPEALS` to the extrapolation degree, we additionally consider three configurations with polynomial degrees $j = 3$, 4, and 5.

The experiments are conducted on synthetic O-INDSCAL instances with $n \times n$ data slices. The number of subjects is chosen as $m \in \{100, 500\}$, the number of stimuli satisfies $n \leq 200$, and the embedding dimension is set to $r \in \{2, 3, 5\}$. For consistency across all tested algorithms, we use the same initial point $Q^{(0)}$ and impose a common convergence criterion,

$$\| \operatorname{grad} f(Z^{(k)}) \| \leq 10^{-8},$$

where $Z^{(k)} = \left(Q^{(k)}, D_1^{(k)}, \ldots, D_m^{(k)}\right)$ denotes the k-th iterate.

Table 6 presents the comparative results for two dataset types, NND and RAND. The reported performance indicators–total computation time ("`CPU(s)`"), number of iterations ("`IT`"), final Riemannian gradient norm ("$\|\text{grad}f\|$"), and final objective value ("`Fvalue`")–are defined in the same manner as in Table 1. Convergence trajectories, plotted as $\log_{10} \| \operatorname{grad} f(Z^{(k)}) \|$ versus outer iterations of all compared algorithms, are shown in Figure 6. From Table 6, we observe that all algorithms perform well on the partially structured NND datasets, converging to accurate solutions under the prescribed stopping criterion. Benefiting from its quadratic local convergence, the proposed `Hybrid-Newton` algorithm typically attains the target precision in fewer iterations than `ALS` and `MPEALS`. However, this reduction in iteration count comes at the expense of a higher per-iteration computational cost, primarily due to the solution of the symmetric linear systems $H_\phi^{(k)} x = g^{(k)}$ via the conjugate residual (CR) method. As a result, `Hybrid-Newton` often exhibits slightly longer overall run times in these settings.

Despite its higher per-iteration cost, the proposed `Hybrid-Newton` algorithm remains competitive, especially for more challenging or less structured datasets, where its superior convergence rate can compensate for the additional overhead. Overall, the results confirm the robustness and effectiveness of the proposed method for solving the O-INDSCAL problem and indicate that it provides a compelling alternative to existing ALS-based techniques, particularly when high-accuracy solutions are required.

5 Conclusion

In this work, we have developed a Riemannian Newton-type algorithm for efficiently solving the orthonormality-constrained INDSCAL (O-INDSCAL) fitting problem in metric multidimensional scaling. By formulating the model as an optimization problem on the product manifold $\mathcal{O}(n, r) \times \mathcal{D}(r)^m$, we derived explicit expressions for both the Riemannian gradient and the Riemannian Hessian. To alleviate the computational burden associated with the Newton step, we further constructed a dimension-reduced matrix representation of the Riemannian Newton equation using vectorization and Kronecker-product techniques. This reformulation recasts the Newton step as the solution of a standard symmetric linear system whose size matches the intrinsic dimension of the manifold, thereby enabling the use of efficient direct or iterative Krylov-type linear solvers. On top of this core Newton framework, we designed a hybrid globalization strategy that couples a globally convergent Riemannian curvilinear search, equipped with Barzilai–Borwein step sizes, with a locally rapid Riemannian Newton refinement.

Table 6 Comparison results for the proposed `Hybrid-Newton` algorithm with `ALS` and `MPEALS`

Algorithms	m, (n,r)	CPU(s)	IT	$\|\text{grad} f\|$	Fvalue	m, (n,r)	CPU(s)	IT	$\|\text{grad} f\|$	Fvalue
Partially structured with *nnd* weight matrices (NND)										
Hybrid-Newton		1.830	7(2123)	5.90×10^{-9}	1.520		6.115	7(2848)	1.56×10^{-9}	2.717
TBALS		0.189	47	8.28×10^{-9}	1.520		0.222	52	9.05×10^{-9}	2.717
MPEALS($j=3$)	500, (25,3)	0.055	13	3.41×10^{-9}	1.520	500, (25,5)	0.072	16	8.58×10^{-9}	2.717
MPEALS($j=4$)		0.048	12	5.64×10^{-9}	1.520		0.068	16	1.51×10^{-9}	2.717
MPEALS($j=5$)		0.053	13	6.11×10^{-11}	1.520		0.064	14	7.96×10^{-9}	2.717
Hybrid-Newton		1.441	5(1510)	1.93×10^{-10}	1.469		9.935	9(3889)	2.64×10^{-11}	2.711
TBALS		0.608	46	9.92×10^{-9}	1.469		0.681	53	6.92×10^{-9}	2.711
MPEALS($j=3$)	500, (50,3)	0.171	13	7.77×10^{-10}	1.469	500, (50,5)	0.180	14	7.56×10^{-9}	2.711
MPEALS($j=4$)		0.166	12	7.67×10^{-9}	1.469		0.212	16	4.34×10^{-10}	2.711
MPEALS($j=5$)		0.169	13	2.69×10^{-11}	1.469		0.182	14	6.79×10^{-9}	2.711
Hybrid-Newton		1.020	3(495)	1.13×10^{-9}	1.468		5.281	4(1253)	5.77×10^{-10}	2.696
TBALS		1.170	41	9.50×10^{-9}	1.468		1.376	47	9.34×10^{-9}	2.696
MPEALS($j=3$)	500, (100,3)	0.374	13	1.40×10^{-9}	1.468	500, (100,5)	0.414	14	2.71×10^{-9}	2.696
MPEALS($j=4$)		0.318	11	7.97×10^{-10}	1.468		0.484	16	1.15×10^{-10}	2.696
MPEALS($j=5$)		0.378	13	2.86×10^{-11}	1.468		0.387	13	1.35×10^{-9}	2.696
Hybrid-Newton		4.182	4(1117)	1.03×10^{-12}	1.464		10.656	4(1217)	1.41×10^{-9}	2.673
TBALS		4.972	43	9.81×10^{-9}	1.464		5.737	49	8.40×10^{-9}	2.673
MPEALS($j=3$)	500, (200,3)	1.515	13	6.40×10^{-10}	1.464	500, (200,5)	1.513	13	4.96×10^{-9}	2.673
MPEALS($j=4$)		1.275	11	1.86×10^{-10}	1.464		1.869	16	8.07×10^{-11}	2.673
MPEALS($j=5$)		1.508	13	2.66×10^{-11}	1.464		1.500	13	1.80×10^{-9}	2.673
Completely random data sets (RAND)										
Hybrid-Newton		0.403	170/8(2108)	1.00×10^{-13}	669.266		0.283	110/7(1544)	1.98×10^{-13}	666.738
TBALS		1.075	614	9.68×10^{-9}	669.231		1.529	1004	9.88×10^{-9}	666.683
MPEALS($j=3$)	100, (50,2)	0.264	151	9.43×10^{-9}	669.231	100, (50,3)	0.381	247	9.49×10^{-9}	666.683
MPEALS($j=4$)		0.198	114	8.48×10^{-9}	669.231		0.284	183	9.65×10^{-9}	666.683
MPEALS($j=5$)		0.279	167	9.36×10^{-9}	669.231		0.287	189	9.44×10^{-9}	666.747
Hybrid-Newton		0.508	131/11(3236)	2.01×10^{-12}	1732.191		0.914	265/11(3070)	7.54×10^{-12}	1728.764
TBALS		2.762	1136	9.82×10^{-9}	1732.142		2.228	905	9.93×10^{-9}	1728.816
MPEALS($j=3$)	100, (80,2)	0.618	252	9.50×10^{-9}	1732.142	100, (80,3)	0.529	219	9.36×10^{-9}	1728.816
MPEALS($j=4$)		0.308	127	7.93×10^{-9}	1732.097		0.465	193	9.47×10^{-9}	1728.816
MPEALS($j=5$)		0.471	194	8.25×10^{-9}	1732.142		0.419	173	9.24×10^{-9}	1729.013
Hybrid-Newton		0.737	171/9(2890)	5.03×10^{-11}	2718.310		1.139	284/8(3500)	3.66×10^{-9}	2714.642
TBALS		2.171	624	9.94×10^{-9}	2718.247		2.259	632	9.94×10^{-9}	2714.526
MPEALS($j=3$)	100, (100,2)	0.632	179	8.57×10^{-9}	2718.247	100, (100,3)	0.660	190	9.90×10^{-9}	2714.526
MPEALS($j=4$)		0.458	129	9.24×10^{-9}	2718.247		0.485	138	9.38×10^{-9}	2714.526
MPEALS($j=5$)		0.780	225	9.22×10^{-9}	2718.247		0.454	130	9.81×10^{-9}	2714.526
Hybrid-Newton		0.785	114/11(2626)	6.54×10^{-11}	3919.159		1.347	153/14(5726)	8.97×10^{-12}	3915.119
TBALS		2.337	498	9.77×10^{-9}	3919.159		4.676	951	9.70×10^{-9}	3914.908
MPEALS($j=3$)	100, (120,2)	0.755	159	8.84×10^{-9}	3919.159	100, (120,3)	1.301	267	9.65×10^{-9}	3914.908
MPEALS($j=4$)		0.616	128	9.19×10^{-9}	3919.159		0.943	192	8.59×10^{-9}	3914.908
MPEALS($j=5$)		0.694	146	7.89×10^{-9}	3919.453		1.165	236	9.33×10^{-9}	3914.908

Consequently, the proposed method enjoys global convergence guarantees inherited from the first-order phase, while retaining the quadratic local convergence characteristic of Newton methods in a neighborhood of a solution. Extensive numerical experiments on both real and synthetic datasets indicate that the resulting `Hybrid-Newton` algorithm is competitive with the classical `ALS` and `MPEALS` schemes, and consistently outperforms projected gradient flows and several representative first- and second-order Riemannian optimization algorithms implemented in the Manopt toolbox, especially on medium- to large-scale instances.

We also remark that the O-INDSCAL model admits a natural tensorial interpretation. The collection of doubly centered dissimilarity matrices can be arranged into a third-order tensor, and the corresponding fitting problem is closely related to structured low-rank tensor approximation and simultaneous (joint) matrix diagonalization. This viewpoint provides a principled link between O-INDSCAL and the broader literature on multilinear algebra and tensor decompositions, and suggests alternative formulations as well as scalable algorithms tailored to large m and high-dimensional regimes. A systematic exploration of this tensor-based perspective, including the development and evaluation of tensor-oriented algorithms, is an important direction for future work. In the present manuscript, however, we have deliberately focused on the Riemannian manifold formulation and the associated algorithmic framework, leaving tensor-based methods and comparisons for subsequent studies.

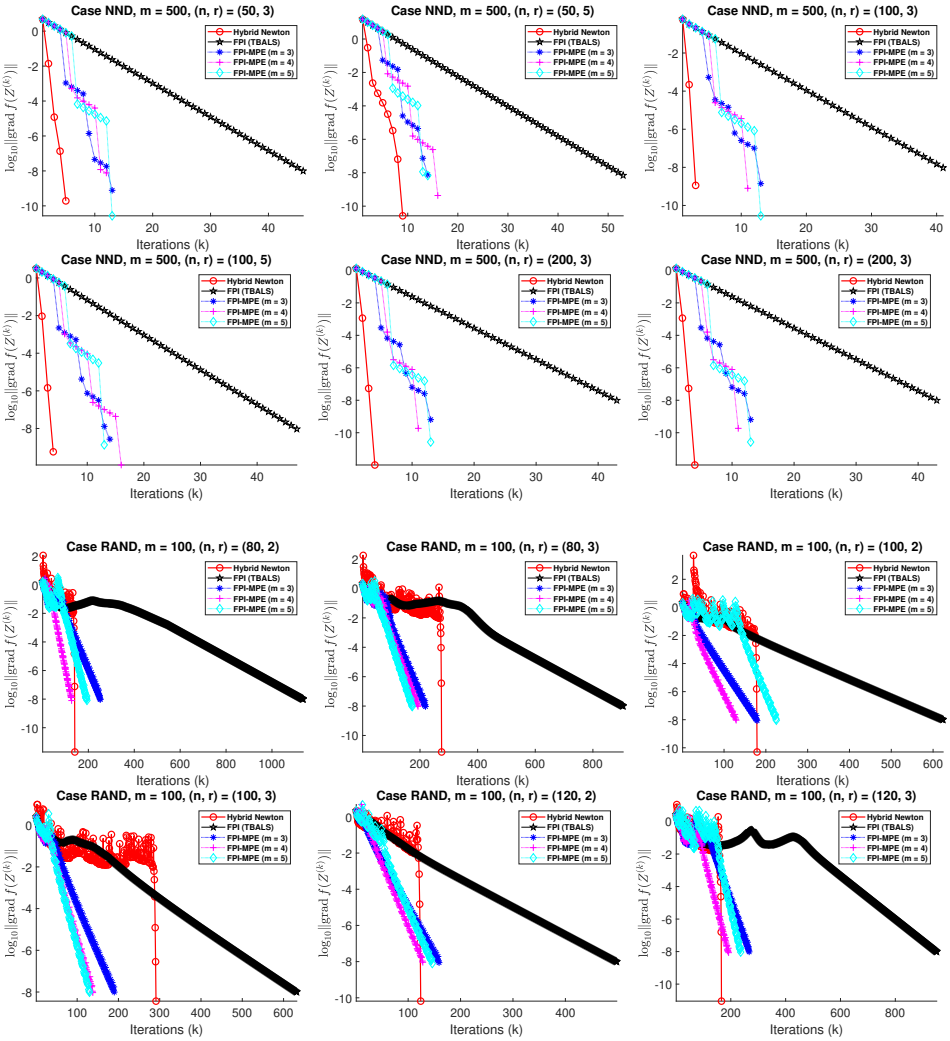


Fig. 6 Comparison among the performances of the proposed `Hybrid-Newton` algorithm and two state-of-the-art algorithms ALS and MPEALS for various problem dimensions and datasets.

Availability of supporting data

Data sharing is not applicable to this article, as no new datasets were generated or analyzed in this study.

Contributions

Jiao-fen Li, Rui-li Jiang, Rui-juan Jiao, Xue-lin Zhou and Ya-qiong Wen drafted the manuscript. All authors contributed to the numerical experiments and reviewed and approved the final version of the manuscript.

Corresponding author

Correspondence should be addressed to Xue-lin Zhou (zhouxuelin0309@163.com) and Ya-qiong Wen (pmxdgmcdsy@163.com).

Ethics declarations

Ethics approval

Not applicable

Conflict of interest

The authors declare no competing interests.

References

1. P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.
2. P.-A. Absil, R. Mahony, and J. Trumpf. An extrinsic look at the Riemannian Hessian. In *International Conference on Geometric Science of Information*, pages 361–368. Springer, 2013.
3. K. Aihara and H. Sato. A matrix-free implementation of Riemannian Newton's method on the Stiefel manifold. *Optimization Letters*, 11:1729–1741, 2017.
4. J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.
5. I. Borg and P. J. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.
6. I. Borg, P. J. Groenen, and P. Mair. *Applied Multidimensional Scaling*. Springer Science and Business Media, 2012.
7. N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a MATLAB toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(1):1455–1459, 2014.
8. M. Browne. The Young–Householder algorithm and the least squares multidimensional scaling of squared distances. *Journal of Classification*, 4(2):175–190, 1987.
9. J. D. Carroll and P. Arabie. Multidimensional scaling. *Measurement, Judgment and Decision Making*, pages 179–250, 1998.
10. J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart–Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.
11. M. T. Chu and N. T. Trendafilov. On a differential equation approach to the weighted orthogonal procrustes problem. *Statistics and Computing*, 8:125–133, 1998.
12. M. T. Chu and N. T. Trendafilov. Orthomax rotation problem: a differential equation approach. *Behaviormetrika*, 25(1):13–23, 1998.

13. H. V. Henderson and S. R. Searle. The vec-permutation matrix, the vec operator and Kronecker products: a review. *Linear and Multilinear Algebra*, 9(4):271–288, 1981.
14. J. Hu, X. Liu, Z.-W. Wen, and Y.-X. Yuan. A brief introduction to manifold optimization. *Journal of the Operations Research Society of China*, 8:199–248, 2020.
15. W. Krzanowski. *Principles of Multivariate Analysis*, volume 23. OUP Oxford, 2000.
16. J.-F. Li, W. Li, X.-F. Duan, and M. Xiao. Newton's method for the parameterized generalized eigenvalue problem with nonsquare matrix pencils. *Advances in Computational Mathematics*, 47:1–50, 2021.
17. S. Loisel and Y. Takane. Generalized GIPSCAL re-revisited: a fast convergent algorithm with acceleration by the minimal polynomial extrapolation. *Advances in Data Analysis and Classification*, 5(1):57–75, 2011.
18. Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.
19. H. Sato. Riemannian Newton-type methods for joint diagonalization on the Stiefel manifold with application to independent component analysis. *Optimization*, 66 (12):2211–2231, 2017.
20. S. S. Schiffman, M. L. Reynolds, F. W. Young, J. D. Carroll, and J. B. Kruskal. *Introduction to Multidimensional Scaling: Theory, Methods, and Applications*. Academic Press, 1981.
21. L. F. Shampine and M. W. Reichelt. The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.
22. Y. Takane, K. Jung, and H. Hwang. An acceleration method for Ten Berge et al.'s algorithm for orthogonal INDSCAL. *Computational Statistics*, 25(3):409–428, 2010.
23. N. T. Trendafilov. A continuous-time approach to the oblique procrustes problem. *Behaviormetrika*, 26:167–181, 1999.
24. N. T. Trendafilov. GIPSCAL revisited: a projected gradient approach. *Statistics and Computing*, 12(2):135–145, 2002.
25. N. T. Trendafilov. Dynamical system approach to factor analysis parameter estimation. *British Journal of Mathematical and Statistical Psychology*, 56(1):27–46, 2003.
26. N. T. Trendafilov. Orthonormality-constrained INDSCAL with nonnegative saliences. In *International Conference on Computational Science and Its Applications*, pages 952–960. Springer, 2004.
27. N. T. Trendafilov. The dynamical system approach to multivariate data analysis. *Journal of Computational and Graphical Statistics*, 15(3):628–650, 2006.
28. N. T. Trendafilov. Dindscal: direct INDSCAL. *Statistics and Computing*, 22(2): 445–454, 2012.
29. N. Trendafilov and M. Gallo. *Multivariate Data Analysis on Matrix Manifolds (with Manopt)*. Springer, 2021.
30. N. T. Trendafilov and I. T. Jolliffe. Projected gradient approach to the numerical solution of the SCoTLASS. *Computational Statistics and Data Analysis*, 50(1): 242–253, 2006.
31. Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1):397–434, 2013.
32. X.-L. Zhou and C.-Q. Li. A trust-region framework for iterative solution of the direct INDSCAL problem in metric multidimensional scaling. *Numerical Algorithms*,

pages 1–39, 2024.
33. X.-L. Zhou, C.-Q. Li, J.-F. Li, and X.-F. Duan. A Riemannian inexact Newton method for solving the orthogonal INDSCAL problem in multidimensional scaling. *IMA Journal of Numerical Analysis*, page draf047, 2025.