Weil-R: Automatic Differentiation of Exact Riemannian Hessians

Anonymous authors
Paper under double-blind review

Abstract

On matrix manifolds such as the sphere, the Stiefel manifold, and the symmetric positive-definite cone, second-order optimization requires Riemannian Hessians that account for both objective curvature and the geometry of the constraint set. We introduce **Weil-R**, a framework that lifts finite-dimensional Weil algebras through constraint-preserving manifold retractions to extract Riemannian Hessians in a single forward pass, without a finite-difference step size or nested differentiation. Our covariant extraction theorem shows that the mixed algebra coefficient equals the Riemannian Hessian whenever the retraction acceleration vanishes or is orthogonal to the gradient, covering the exponential map, the QR retraction on Stiefel, and the projection retraction on the sphere. A modular JAX library implements the construction with matrix-free Hessian-vector products and memory linear in the algebra dimension. On five of seven constrained benchmarks, Newton and trust-region with the Weil-R Hessian reach machine precision; on the $n = 512$ Stiefel Rayleigh quotient they converge in 11.4 ± 1.0 iterations versus 40 for PyManopt trust-region ($3.5\times$ fewer). Fifty-one geometric consistency checks pass at tolerance 10^{-9}, and extracted derivatives agree with reference Hessians to machine precision. These results connect algebraic differentiation and Riemannian optimization for learning with geometric constraints.

1 Introduction

Second-order curvature information is what separates linearly converging optimizers from quadratically converging ones, and on curved constraint sets it is the hardest to obtain. Matrix manifolds, the sphere S^{n-1}, the Stiefel manifold $\mathrm{St}(n, p)$ of orthonormal frames, the cone $\mathrm{SPD}(n)$ of symmetric positive-definite matrices, arise naturally wherever an optimization problem enforces a geometric constraint: low-rank matrix completion and subspace learning (Edelman et al., 1998; Absil et al., 2008), covariance and Gaussian-mixture estimation, geometric deep learning, and robotic estimation (Boumal, 2023). The Riemannian Hessian encodes both the objective's curvature and the geometry of the feasible set, and trust-region and Newton methods built on it converge to machine precision in tens of iterations (Absil et al., 2007). The difficulty is that this object requires *covariant*, chart-independent second derivatives, whereas standard automatic differentiation (AD) delivers only ambient ones.

Existing methods each satisfy some requirements and violate others. *Projected reverse-mode* methods (Absil et al., 2013; Townsend et al., 2016) are exact but require cumbersome, manifold-specific formulas; *finite differences* are manifold-agnostic but introduce step-size-dependent truncation error; *flat-space Weil-algebra and Taylor-mode AD* (Fike & Alonso, 2011; Ishii, 2021; Bettencourt et al., 2019; Sangha, 2025a) are algebraically exact and step-size free but recover the *ambient* Hessian, which differs from the Riemannian one by a $\Theta(1)$ term on constrained manifolds (Absil et al., 2013). No existing method is simultaneously exact, chart-independent, and constraint-preserving.

We denote our framework **Weil-R** (Weil algebra + Retraction), reflecting the two ingredients: a Weil algebra that encodes the infinitesimal neighborhood, and a constraint-preserving retraction that lifts algebra-valued points back to the manifold. The intuition is to put the geometry into the forward pass rather than removing it afterward. Given tangent directions $v, w \in T_xM$, we (i) *embed* them in the hyper-dual Weil algebra $A_{\mathrm{HD}} = \mathbb{R}[\varepsilon_1, \varepsilon_2]/(\varepsilon_1^2, \varepsilon_2^2)$ via $U_A = v \otimes \varepsilon_1 + w \otimes \varepsilon_2$; (ii) *retract* the resulting A-valued point back to the manifold with a retraction that preserves the defining constraint coefficient by coefficient; (iii) *evaluate* the objective; and (iv) *extract* the mixed coefficient. Theorem 2 establishes that this mixed coefficient equals the Riemannian Hessian plus a retraction-acceleration term $g_x(\mathrm{grad}\, f(x), \gamma''_{v,w})$. For retractions with vanishing acceleration, geodesic maps, the QR retraction on the Stiefel manifold, and the projection retraction on

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the sphere, the extraction is *exact in real arithmetic*: no truncation, no Taylor remainder, and no step-size parameter. Figure 1 outlines the four-layer architecture that realizes this pipeline in JAX (Bradbury et al., 2018).

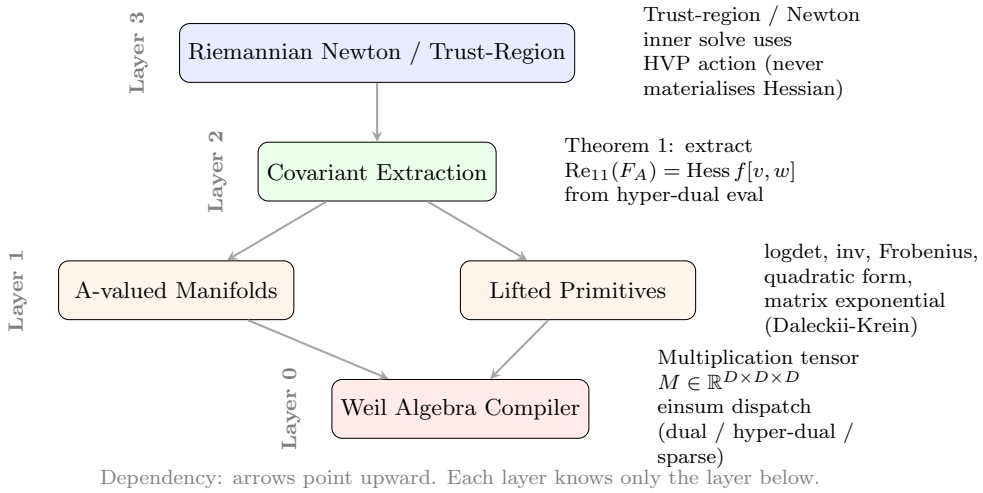


Figure 1: Four-layer architecture of the Weil-R library. Layer 0: Weil algebra compiler stores the multiplication tensor M and exposes pointwise product, matrix product, and embedding/extraction for lifting and extraction. Layer 1: Algebra-valued (A-valued) manifold primitives implement constraint-preserving retractions and lifted matrix operations (log-determinant, matrix inverse, Frobenius norm). Layer 2: covariant extraction combines the algebra and manifold layers to produce exact Riemannian gradients and Hessian-vector products. Layer 3: Riemannian optimizers (Newton, trust-region, Riemannian Conjugate Gradient in first-order: RCG, Gradient Descent with a fixed step-size in first-order: GD). Each layer depends only on layers below it. Hessian-vector product (HVP)

We validate the framework on three matrix manifolds. Fifty-one invariance checks pass at tolerance 10^{-9},the flat-space Weil evaluation (Sangha, 2025a) is wrong by ~ 0.5 on exactly the same problems, confirming that the manifold layer, not the algebra, is the essential contribution. On the Stiefel Rayleigh quotient at dimension 512, Riemannian Newton and trust-region with the exact Weil Hessian converge to machine precision in 11.4 ± 1.0 iterations versus 40 for the PyManopt trust-region (Townsend et al., 2016); across seven problems and three manifolds, Newton reaches the known optimum on five and the best recorded objective on the two strongly non-convex ones. Peak memory scales linearly in the algebra dimension ($R^2 \geq 0.988$) because the matrix-free Hessian-vector-product action never materializes the dense Hessian.

Our contributions are three-fold:

- **Theoretical.** A covariant extraction theorem (Theorem 2) showing that the mixed coefficient of a hyper-dual algebra evaluation through a second-order retraction equals the Riemannian Hessian plus a retraction-acceleration term, which vanishes for geodesic retractions and contributes orthogonally to the gradient on the sphere.
- **Algorithmic.** An A-valued manifold-lifting layer that preserves the defining constraint of the manifold at every nilpotent order (sphere, Stiefel, SPD), implemented as a library of constraint-preserving primitives in JAX; a single forward pass yields gradients and Hessian-vector products with memory linear in the algebra dimension.
- **Empirical.** 51/51 invariance checks at tolerance 10^{-9}; machine-precision convergence of Riemannian Newton on five of seven test problems, with best-recorded solution quality on the two strongly non-convex ones, across three manifolds; and a $3.5\times$ iteration-count reduction versus the PyManopt trust-region (Townsend et al., 2016) on the $n = 512$ Stiefel Rayleigh quotient.

The remainder of the paper is organized as follows. Section 2 positions Weil-R against related work. Section 3 states and proves the covariant extraction theorem. Section 4 describes the JAX implementation. Section 5 presents experiments.

2 Related work

2.1 Higher-order automatic differentiation

Computing higher-order derivatives, gradients of gradients, Hessian-vector products, and full Hessians is central to numerical optimization and scientific computing (Griewank & Walther, 2008; Baydin et al., 2018). Three families are relevant to our construction.

Forward-mode and dual numbers: Forward-mode AD propagates a value alongside its derivative through the computation graph. Dual numbers $\mathbb{R}[\varepsilon]/(\varepsilon^2)$ encode first derivatives; hyper-dual numbers $\mathbb{R}[\varepsilon_1, \varepsilon_2]/(\varepsilon_1^2, \varepsilon_2^2)$ encode mixed second derivatives (Fike & Alonso, 2011). The approach has been consolidated into production systems, from ForwardDiff in the Julia ecosystem (Revels et al., 2016) and TaylorDiff (Tan, 2022) to recent Taylor-mode compilation in PyTorch (Dangel et al., 2026) and JAX (Bettencourt et al., 2019). These methods are exact in flat space, but they all differentiate the *ambient* function: the output is a Taylor polynomial of the ambient objective. On a constrained space, the ambient Hessian differs from the Riemannian Hessian by a correction involving the second fundamental form (Absil et al., 2013). We make this concrete: on the sphere, Stiefel manifold, and SPD cone, the correction is $\Theta(1)$, so neglecting it produces a second-order model that is systematically wrong rather than slightly perturbed.

Weil algebras. The algebraic foundation of forward-mode AD is the theory of Weil algebras (Kock, 1981; Moerdijk & Reyes, 2013; Huot et al., 2020): a Weil algebra $A = \mathbb{R}[\varepsilon_1, \ldots, \varepsilon_P]/I$ with nilpotent maximal ideal encodes an infinitesimal neighborhood, and evaluating a smooth function on an A-valued point extracts Taylor coefficients exactly, with no finite-difference truncation. Ishii (2021) developed algorithmic differentiation through arbitrary Weil algebras, constructing the necessary C^∞-ring structure via Gröbner-basis machinery; our sparse quotient construction (downward-closed monomial ideals) is precisely the object Ishii studies. Most recently, Sangha (2025a) gave a geometric analysis of jet functors and Weil algebras in AD, claiming one-pass computation of all mixed derivatives at cost linear in the algebra dimension, and a companion paper (Sangha, 2025b) studies function algebras on manifolds but does not implement or evaluate an optimizer. These works establish the algebraic machinery; what they do not provide is the manifold layer. We show that the Euclidean Weil evaluation alone computes the ambient Hessian, wrong by a $\Theta(1)$ term on constrained manifolds, and that lifting it through a constraint-preserving retraction is what restores exactness.

Taylor-mode AD. Bettencourt et al. (2019) introduced Taylor-mode AD for higher-order derivatives in JAX, using the univariate Weil algebra $\mathbb{R}[t]/(t^{r+1})$ as the computational backbone; we compare against it directly (baseline B10). The Taylor-mode jet requires $2k + 1$ forward passes per Hessian-vector product through polarisation, whereas the hyper-dual algebra computes the same HVP in a single pass. More fundamentally, Taylor-mode lacks built-in primitives for matrix operations (log-determinant, matrix inverse, matrix exponential), which our lifted-primitives layer provides intrinsically on manifolds.

2.2 Riemannian optimisation

Riemannian-optimization libraries, Manopt (Boumal et al., 2014), PyManopt (Townsend et al., 2016), Geomstats (Miolane et al., 2020), Geoopt (Kochurov et al., 2020), and McTorch (Meghwanshi et al., 2018), expose an `ehess2rhess` routine that maps a Euclidean Hessian to a Riemannian one via the Weingarten map (baseline B4); none provides an intrinsic, chart-independent exact Hessian. Manifold-AD precedents (Novikov et al., 2022; Psenka & Boumal, 2020; Koppel & Waslander, 2018) project an ambient derivative rather than compute an intrinsic one, omitting the curvature correction (baseline B9). Foundations (Absil et al., 2008; 2007; 2013; Edelman et al., 1998; Boumal, 2023), quasi-Newton (Boumal, 2015) and adaptive (Sakai & Iiduka, 2024) variants, trust-region Gaussian-mixture fitting (Sembach et al., 2022), and the Euclidean second-order literature (Yao et al., 2021; Liu et al., 2024) are detailed in Appendix B.

3 Covariant extraction theorem

We prove that the mixed coefficient of a smooth scalar function evaluated through a second-order retraction, lifted to the hyper-dual Weil algebra, recovers the Riemannian Hessian exactly when the retraction acceleration vanishes. This result underpins Weil-R: a single A-valued forward pass through manifold primitives yields exact, chart-independent second derivatives without nested differentiation, ambient projection, or finite-difference approximation.

3.1 Hyper-dual Weil algebra

The *hyper-dual algebra* is

$$A_{\mathrm{HD}} = \mathbb{R}[\varepsilon_1, \varepsilon_2]/(\varepsilon_1^2,\ \varepsilon_2^2). \tag{1}$$

Every element $a \in A_{\mathrm{HD}}$ has the unique representation

$$a = a_{00} + a_{10}\,\varepsilon_1 + a_{01}\,\varepsilon_2 + a_{11}\,\varepsilon_1\varepsilon_2, \tag{2}$$

with $a_{ij} \in \mathbb{R}$. The product rule is determined by $\varepsilon_1^2 = \varepsilon_2^2 = 0$; in particular the $\varepsilon_1\varepsilon_2$ component satisfies

$$(a_{10}\varepsilon_1 + a_{01}\varepsilon_2)(b_{10}\varepsilon_1 + b_{01}\varepsilon_2) = (a_{10}b_{01} + a_{01}b_{10})\,\varepsilon_1\varepsilon_2. \tag{3}$$

The algebra A_{HD} is a Weil algebra with maximal ideal $\mathfrak{m} = \mathrm{span}\{\varepsilon_1, \varepsilon_2, \varepsilon_1\varepsilon_2\}$ and nilpotency order $r = 2$.

The *coefficient extraction* maps are $\mathrm{Re}_{ij} : A_{\mathrm{HD}} \to \mathbb{R}$, and $\mathrm{Re}_{ij}(a) = a_{ij}$.

Remark 1 (A-points and the Weil bundle). *The lift $U_A = v \otimes \varepsilon_1 + w \otimes \varepsilon_2$ (equation 7) together with its base point is an A-*point *of M in the Weil bundle $T^{A_{\mathrm{HD}}}M \to M$ (Kock, 1981; Moerdijk & Reyes, 2013): the bundle provides the geometric state space for the derivative information propagated by Weil-R, and its role here is operational. Evaluating a retraction on an A-point produces another A-point, and higher-order coefficients obey nonlinear transition laws, treating the mixed coefficient as an independent tangent vector changes the represented jet. Appendix A records the exact laws and the lifted constraint identities used by our verification suite.*

3.2 Second-order retractions

Definition 1 (Second-order retraction). *Let M be a smooth manifold. A* second-order retraction *is a smooth map $R : TM \to M$ satisfying, for all $x \in M$ and $v \in T_xM$: [(R1)] $R_x(0) = x$; [(R2)] $DR_x(0)[v] = v$ (the differential at the origin is the identity on T_xM); [(R3)] for every curve $c(t)$ in M with $c(0) = x$, $\dot{c}(0) = v$, there exists a retraction-speed error $e(t) = o(t^2)$ such that $R_{c(t)}(tw) = c(t) + o(t)$ for all $w \in T_{c(t)}M$.*

Property (R2) ensures R is a first-order approximation to the exponential map. The exponential map exp *is a retraction satisfying the stronger condition that (R3) holds with $e(t) = 0$.*

3.3 Riemannian Hessian

Let (M, g) be a Riemannian manifold and $f : M \to \mathbb{R}$ smooth. The *Riemannian Hessian* is the symmetric bilinear form

$$\mathrm{Hess}\, f(x)[v, w] = g_x\big(\nabla_v \,\mathrm{grad}\, f,\ w\big), \tag{4}$$

where ∇ is the Levi-Civita connection. Equivalently, for the exponential map,

$$\mathrm{Hess}\, f(x)[v, w] = \frac{\partial^2}{\partial s\,\partial t} f\big(\exp_x(sv + tw)\big)\Big|_{s=t=0}. \tag{5}$$

3.4 Main theorem

Theorem 2 (Covariant extraction). *Let (M, g) be a Riemannian manifold, $f : M \to \mathbb{R}$ smooth, $x \in M$, and $v, w \in T_xM$. Let R be a second-order retraction and define the retraction acceleration*

$$\gamma''_{v,w} = \frac{\partial^2}{\partial s\,\partial t} R_x(sv + tw)\Big|_{s=t=0} \in T_xM. \tag{6}$$

Lift v and w to the hyper-dual algebra via

$$U_A = v \otimes \varepsilon_1 + w \otimes \varepsilon_2 \in T_xM \otimes \mathfrak{m}, \tag{7}$$

and evaluate

$$F_A = f\big(R_x(U_A)\big) \in A_{\mathrm{HD}}. \tag{8}$$

Then the mixed coefficient satisfies

$$\mathrm{Re}_{11}(F_A) = \mathrm{Hess}\, f(x)[v, w] + g_x\big(\mathrm{grad}\, f(x),\ \gamma''_{v,w}\big). \tag{9}$$

Consequently, $\mathrm{Re}_{11}(F_A) = \mathrm{Hess}\, f(x)[v, w]$ *if and only if*

$$\gamma''_{v,w} = 0 \qquad \textit{or} \qquad \mathrm{grad}\, f(x) = 0. \tag{10}$$

Remark 2. *Theorem 2 is an* exact *identity in real arithmetic: There is no finite-difference truncation, no Taylor remainder, and no step-size parameter. Floating-point roundoff is the sole source of numerical error.*

Remark 3. *Condition equation 10 holds in three important cases:*

1. $R = \exp$ *is the exponential map (geodesic), for which* $\gamma''_{v,w} = 0$ *by definition.*
2. *R is a second-order retraction for which the mixed acceleration vanishes identically:* $\gamma''_{v,w} = 0$ *for all* $v, w \in T_xM$*. This holds for the QR retraction on the Stiefel manifold (by symmetry of the QR decomposition).*
3. $\gamma''_{v,w} \neq 0$ *but* $g_x(\operatorname{grad} f(x), \gamma''_{v,w}) = 0$ *because* $\gamma''_{v,w}$ *lies in the* normal *direction at x while* $\operatorname{grad} f(x)$ *is tangent. This occurs for the projection retraction on the sphere, where* $\gamma''_{v,w} = -x\, g_x(v, w)$ *is normal to* S^{n-1} *and* $\operatorname{grad} f(x) \perp x$ *for any f restricted to the sphere. Theorem 2 gives the exact Hessian even though the retraction acceleration is nonzero.*

Case (3) is particularly significant: it shows that exactness does not require $\gamma''_{v,w} = 0$*, only that the acceleration is orthogonal to the gradient. This is a geometric property of the manifold and retraction, not a coincidence.*

Remark 4 (First-order retractions)**.** *Theorem 2 requires R to be a* second-order *retraction. A first-order retraction (satisfying only (R1)–(R2)) introduces an $O(t)$ error in the retraction curve, which propagates to an $O(1)$ error in the mixed coefficient—the same $\Theta(1)$ gap between the ambient and Riemannian Hessians that motivates this work. In particular, a first-order retraction cannot be used to extract exact second derivatives; the second-order condition (R3) is essential.*

The proof proceeds by expanding the retraction curve $\gamma(s, t) = R_x(sv + tw)$ to second order in ambient coordinates, applying the chain rule to $f \circ \gamma$, and identifying the mixed coefficient with the Riemannian Hessian via the extrinsic decomposition equation 23. The full derivation is given in Appendix C.

Corollary 3 (First derivative extraction)**.** *Under the hypotheses of Theorem 2, the first-order coefficients satisfy*

$$\operatorname{Re}_{10}(F_A) = df_x[v] = g_x(\operatorname{grad} f(x), v), \qquad \operatorname{Re}_{01}(F_A) = df_x[w] = g_x(\operatorname{grad} f(x), w). \tag{11}$$

These are the standard dual-number results and hold for any retraction satisfying (R1)–(R2).

Corollary 4 (Gradient via metric duality)**.** *If* $v_1, \ldots, v_k$ *span* T_xM *and* $\operatorname{Re}_{10}(F_A^{(i)}) = df_x[v_i]$ *for each i (obtained by running the algebra with* $U_A = v_i\varepsilon_1$*), then* $\operatorname{grad} f(x) = \sum_i g^{ij} df_x[v_i]\, \partial_j$ *is determined by the k first-order evaluations.*

Corollary 5 (Hessian at a critical point)**.** *If* $\operatorname{grad} f(x) = 0$ *(i.e. x is a critical point of f), then* $\operatorname{Re}_{11}(F_A) = \operatorname{Hess} f(x)[v, w]$ *for* any *second-order retraction R, since the acceleration term vanishes:* $g_x(\operatorname{grad} f(x), \gamma''_{v,w}) = 0$.

Corollary 6 (Constraint preservation)**.** *Let* $M = \{x \in \mathbb{R}^n : c(x) = 0\}$ *be a regular level set. If* $R_x : T_xM \to M$ *is an A-valued retraction (i.e.* $R_x(U_A) \in M(A)$ *for all* $U_A \in T_xM \otimes \mathfrak{m}$*), then* $c(R_x(U_A)) = 0$ *in* A_{HD}*. In particular: (1) the ε_1-coefficient gives* $Dc_x[v] = 0$ *(tangent condition); (2) the ε_2-coefficient gives* $Dc_x[w] = 0$*; (3) the $\varepsilon_1\varepsilon_2$-coefficient gives* $D^2c_x[v, w] + Dc_x[\gamma''_{v,w}] = 0$.

These are exactly the constraints enforced by the A-valued manifold primitives (Section 4).

We verify Theorem 2 on three matrix manifolds. In each case, the retraction acceleration either vanishes identically or is orthogonal to the gradient, so the mixed coefficient yields the exact Riemannian Hessian. The detailed computations are given in Appendix D.

3.5 Cost analysis

The algebra evaluation in equation 8 requires: (1) **One forward pass** through the manifold trace with A-valued inputs. The algebra dimension is $\dim A_{\mathrm{HD}} = 4$ (real part + two generators + mixed monomial). (2) **Cost multiplier**: each primitive operation in the manifold trace is replaced by its A-extension, which involves $O(4)$ scalar operations per original scalar operation (for elementwise primitives) or $O(4 \cdot d^2)$ for matrix primitives of dimension d. (3) **Memory**: the backward pass is not needed; the entire computation is forward-mode with storage proportional to the number of active intermediate A-valued matrices. (4) **No**

truncation error: the nilpotent algebra terminates the Taylor expansion exactly. The only error source is floating-point roundoff ($\sim 10^{-16}$ in FP64).

For k-direction Hessian evaluations, a single $(k+1)$-generator algebra $A = \mathbb{R}[\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_k]/(\varepsilon_0, \ldots, \varepsilon_k)^2$ of dimension $\binom{k+2}{2}$ computes the gradient in one direction and all $\binom{k}{2}$ mixed Hessian entries simultaneously.

4 Implementation

4.1 Architecture overview

The Weil-R procedure is organized in four strictly ordered layers (Figure 1). Each layer depends only on the layers below it; this enforces a clean separation between the algebra compiler, the manifold lifting, the derivative extraction, and the optimizer integration.

The key invariant maintained throughout all layers: *an A-valued tensor has the algebra coefficient axis as its last axis*, i.e. shape $(\ldots, D)$ where $D = \dim A$ (2 for dual, 4 for hyper-dual, $\binom{k+2}{2}$ for k-generator algebras). The entry point for embedding a real tangent direction into the algebra is `W.embed(v, exponent)`; the exit point for reading a derivative coefficient out is `W.coeff(f_A, exponent)`.

Weil algebra compiler: The algebra module implements a finite-dimensional nilpotent commutative $\mathbb{R}$-algebra with an ordered monomial basis. All algebra operations reduce to contractions with a precomputed multiplication tensor $M \in \mathbb{R}^{D \times D \times D}$, dispatched as a single `jnp.einsum` call. Four constructors cover the algebras used in this paper: dual ($D = 2$), hyper-dual ($D = 4$), one-generator ($D = r + 1$), and sparse quotient. Details are given in Appendix E.

Manifold lifting: Each manifold class exposes five methods: real retraction, A-valued retraction (preserving the constraint coefficient-by-coefficient), A-valued exponential map (where available), and real/A-valued tangent projection. The Stiefel retraction uses an A-valued modified Gram–Schmidt procedure; the SPD retraction uses a Neumann-series matrix inverse that terminates by nilpotency. Implementation details are given in Appendix E.

Covariant extraction: The extraction module is the central contribution. Given an A-valued retraction and an A-valued loss, it returns the intrinsic Riemannian derivative. Each objective requires a pair (f, f_{weil}): a real function and its A-valued twin. For standard objectives (Frobenius norm, quadratic form, log-determinant), the lifted version is a single call to a library primitive (see the lifted primitives listed in Appendix E). For composite objectives, the lifted twin chains these primitives; the pattern is mechanical and typically adds 5–15 lines of code per objective. The cost is comparable to writing a JAX custom gradient, and the lifted twin inherits the same debugging workflow: compare against a finite-difference reference on the real objective at a random point.

4.1.1 Algorithm

The complete procedure for the Riemannian Hessian action $\operatorname{Hess} f(x)[v, w]$ is: (1) **Embed**: lift the tangent vectors $v, w \in T_x M$ into the algebra via $U_A = v \otimes \varepsilon_1 + w \otimes \varepsilon_2 \in T_x M \otimes \mathfrak{m}$. (2) **Retract**: compute the A-valued manifold point $x_A = R_x(U_A) \in M(A)$, which preserves the defining constraint coefficient-by-coefficient. (3) **Evaluate**: compute $F_A = f(x_A) \in A$. (4) **Extract**: read the mixed coefficient $\operatorname{Re}_{11}(F_A) = \operatorname{Hess} f(x)[v, w] + g_x(\operatorname{grad} f(x), \gamma''_{v,w})$. When $\gamma''_{v,w} = 0$ (exponential map, QR on Stiefel, projection on sphere), step 4 yields the exact Riemannian Hessian directly.

The complete implementation is shown below:

```
def tangent_point(W, manifold, x, v, w=None):
    """Build A-valued manifold point with hyper-dual tangent."""
    U = W.embed(v, (1, 0)) + W.embed(w, (0, 1))
    return manifold.retract(W, x, U)

def ddf_along(W, f, manifold, x, v, w, use_exp=False):
    """Mixed second derivative along the retraction curve."""
    assert W.P == 2  # requires hyper-dual algebra
    xA = tangent_point(W, manifold, x, v, w)
    fx = f(xA)
    return W.coeff(fx, (1, 1))  # extract eps1*eps2 coefficient
```

4.1.2 HVP action (matrix-free)

For optimization, the Hessian is never materialized as a $k \times k$ matrix. Instead, the Hessian-vector product (HVP) is computed by `weil_hvp_action`: for each basis vector b_i of T_xM, one covariant evaluation $\mathrm{Hess}\, f(x)[b_i, p]$ is performed, where p is the direction of interest. The k evaluations are batched over a `vmap` with chunk size c to bound peak memory at $O(c \cdot n \cdot p \cdot D)$:

```
def weil_hvp_action(W, f_weil, manifold, x, basis, p, chunk=128):
    """Exact HVP: coordinates of Hess f[x][b_i, p] in tangent basis."""
    hvp_block = _hvp_block(W, f_weil, manifold)
    parts = [hvp_block(basis[i:i+chunk], p, x)
             for i in range(0, len(basis), chunk)]
    return jnp.concatenate(parts, axis=0)
```

The `_hvp_block` function is JIT-compiled and cached per $(W, f_\mathrm{weil}, \mathrm{manifold})$ triple via `lru_cache`, so the Hessian path never re-traces between CG iterations.

4.2 Optimizer integration and Memory management

The optimizer module implements Riemannian trust-region (Manopt Algorithm 10 (Absil et al., 2008)) with Steihaug truncated-CG inner solve (Steihaug, 1983), Riemannian Newton with backtracking line search, and first-order baselines (GD, RCG). The Hessian is never materialized: the inner solver receives a callable HVP action. On strongly indefinite problems, a gradient-step safeguard prevents stalling when the quadratic model reports no descent. Algorithmic details are in Appendix E.

The HVP actionrequires only $O(c \cdot n \cdot p \cdot D)$ memory at chunk size c, a factor of 10^4 below the dense allocation. Memory scales linearly in $\dim A$ ($R^2 \geq 0.988$; Figure 2). The library is implemented in JAX (Bradbury et al., 2018) with no custom CUDA kernels, using `jnp.einsum` for algebra contractions (fused by XLA), `@jax.jit` on the HVP block, and `jax.vmap` for batching. `jax_enable_x64=True` is required for machine-precision accuracy.

5 Experiments

5.1 Environment and Setup

We conduct eight test problems (described in Appendix E.5, Table 5), spanning three matrix manifolds: sphere S^{n-1} (projection retraction), Stiefel $\mathrm{St}(n, p)$ (QR retraction via A-valued Gram–Schmidt), and SPD cone $\mathrm{SPD}(n)$ (affine-invariant exponential map). The problems include Rayleigh quotient (T1), low-rank completion (T2), Procrustes (T3), SPD Karcher mean (T4), Gaussian MLE (T5), robust low-rank PCA (T6), variable stiffness learning (T7), and oblique logistic (T8). The main E3 generality suite runs seven of these (T1–T6 and T8); T7 is an additional case study.

The experimental comparison spans two axes: derivative engine and optimizer class (first vs second order). Ten engine methods (compute derivatives that feed the steps) compared in this paper are detailed in Appendix Table 4: B1–B6 are first-party implementations (finite differences, hyper-dual, nested JVP, Weingarten projection, Euclidean Weil and diagonal query); B7 is our method Weil-R; B8–B10 are second-order external solvers (PyManopt TR (Townsend et al., 2016), projected-Euclidean TR, Taylor-jet TR (Bettencourt et al., 2019)). B5 is the Euclidean Weil of Sangha (2025a), which computes the ambient, not the Riemannian, Hessian, and thus produces inaccurate results on the manifolds. All four optimizers (GD, RCG, Newton, and TR) start from the same random initial point, where Newton and TR use the *exact* Riemannian Hessian obtained from a single Weil-algebra forward pass (Theorem 2); GD and RCG are first-order baselines that use only the gradient. Appendix E.5 introduces the detailed definitions of the four optimizers.

We set up four experiments. **E1 (derivative verification)** checks that the extracted coefficients equal the exact covariant Hessian: accuracy against analytic and nested-JVP references, and invariance under retraction choice, chart, and embedding (51 checks at tolerance 10^{-9}). **E2 (memory and cost scaling)** measures wall time and peak memory as the algebra dimension grows, confirming linear scaling (Results are shown in Appendix G). **E3 (Riemannian optimization)** benchmarks Newton and TR with the exact Weil Hessian against first-order GD and RCG on the seven problems across three manifolds, including an exact-vs-approximate Hessian ablation. **E4 (external comparison)** pits Weil-R against external baselines B5-opt, B8–B10 at matched settings.

5.2 Derivative verification

We verify the Weil-algebra Hessian against analytic and nested-JVP references on the sphere at $n = 8, 12, 16$ (4 seeds per size). Three invariance families are tested: retraction-order (exp vs QR vs polar), chart (normal coordinates, stereographic), and embedding equivariance (orthogonal conjugation).

Euclidean Weil (B5) diverges from the intrinsic Riemannian Hessian by ~ 0.5 on all three manifolds (Table 16 in Appendix G): it computes the ambient Hessian, not the covariant $\mathrm{Hess}_M f$. Our manifold-lifted Weil (Weil-R) closes this gap to machine precision ($\sim 10^{-16}$). This is the central empirical result motivating the manifold layer. 51 invariance checks at tolerance 10^{-9} all pass at machine precision (breakdown in Table 9); the A-valued MGS-QR and polar retractions on the Stiefel manifold agree to 3×10^{-16}.

5.3 Riemannian optimisation

Our primary benchmark evaluates the two canonical tasks of the suite at dimension 512: the Rayleigh quotient (T1) on the Stiefel manifold $\mathrm{St}(512, 5)$ and low-rank completion (T2) on the Stiefel manifold $\mathrm{St}(512, 5)$. Every method runs from the same five random initial points with a budget of 40 iterations.

On both tasks, Newton and trust-region with the exact Weil Hessian (Weil-R + Riemannian solver) converge to machine precision on every seed, reaching a relative gap of 0.000 in roughly ten to nineteen iterations (Table 1). In contrast, the first-order methods GD and RCG stall at relative gaps of 10^{-1}–10^{-2} and exhaust the full 40-iteration budget. This contrast isolates the value of the exact second-order model: quadratic convergence to machine precision is unattainable from gradient information alone. Convergence trajectories are reported in Figure 3 (Appendix G).

On wall-clock time for E3 at $n = 512$, first-order methods are fastest (GD: 0.9s, RCG: 0.4s on T1; both under 1s on T2), while second-order methods take 608s (Newton) and 1025s (trust-region) on T1, and 779s/921s on T2. The gap reflects JAX Python overhead in the tangent-basis construction and CG inner loop; a compiled CUDA kernel would close it. First-order methods are fast precisely because they stall early without converging.

To test generality beyond the $n = 512$ benchmark, we run the seven-problem suite (T1–T6 and T8) across the three manifolds (Table 19, Appendix G). Newton with the exact Weil Hessian reaches the known optimum (relative gap 0.000) on five of the seven tasks (T1–T5). On the two strongly non-convex tasks it attains the best objective among all compared methods: on the oblique logistic problem (T8) it achieves the lowest objective and gradient norm, and on robust low-rank PCA (T6) it reaches a subspace-recovery error of 0.052 ± 0.025, far below GD (0.355) and RCG (0.316). The Thomson problem ($q = 4$, Riesz $s = 2$ energy on $(S^2)^q$) is the most demanding task in the suite. Newton converges to a relative gap of 0.000 on all five seeds without any safeguard. Trust-region initially stalled on this stiff problem because its quadratic model reported no descent at many points; after adding a gradient-step safeguard to the shared trust-region implementation, it too converges to 0.000, with the fallback firing in 7–12 of the 40 iterations.

Table 1: E3: optimisation results at $n = 512$, $p = 5$, 5 seeds. Newton and TR use the exact Weil Hessian.

	T1 (Rayleigh)		T2 (completion)	
Optimizer	Rel. gap	Iters	Recon. err	Iters
GD	1.30×10^{-1}	40	4.76×10^{-1}	40
RCG	6.50×10^{-2}	40	2.33×10^{-1}	40
Newton	0.000	10.0 ± 0.9	0.000	18.8 ± 3.7
TR	0.000	11.4 ± 1.0	≈ 0	15.6 ± 1.0

Table 2: E3 ablation: Thomson problem, $q = 5$, $n = 6$, 5 seeds. Newton with four Hessian models.

Hessian model	Rel. gap	Fallbacks / 40 iters
Exact Weil HVP	0.000	0
Diagonal Weil	0.001 ± 0.001	0
Identity	0.003 ± 0.002	0
BFGS (approx)	0.005 ± 0.003	0

Table 2 compares four Hessian models inside an otherwise identical Newton solver on the Thomson problem ($q = 5$, $n = 6$). Only the exact Weil HVP reaches machine precision (relative gap 0.000, no fallbacks). The diagonal and identity approximations degrade gracefully but fail to attain the 10^{-6} floor within 40 iterations, and BFGS performs worst of the four, reflecting the poor quality of its initial approximation on these curved manifolds.

The fallback frequency also distinguishes the models. With the exact Hessian, the trust-region solver fires the gradient-step safeguard in 7–12 of the 40 iterations, signaling strongly indefinite model Hessians at non-descent points; these fallbacks are a *feature* in that they redirect the iteration to a monotone descent step

and preserve convergence. The approximate models trigger no fallbacks, not because they are more accurate, but because their second-order models mispredict the objective less often when the underlying Hessian is already approximate.

5.4 External baseline comparison

We compare Weil-R (exact Weil HVP) against the external baselines defined in Appendix Table 4: B5-opt (Euclidean Weil, Sangha (2025a), installed from its official `jet-weil-ad` repository), B8 (PyManopt trust-region, Townsend et al. (2016), C backend), B9 (projected-Euclidean trust-region), and B10 (Taylor-jet trust-region, Bettencourt et al. (2019)).

Table 3: E4: external comparison at $n = 512$, $p = 5$, 5 seeds. Iteration counts, final metrics, and wall-clock time.

	T1 (Rayleigh)			T2 (completion)		
Method	Rel. gap	Iters	Time (s)	Recon. err	Iters	Time (s)
B5-opt (Sangha (2025a))	0.008 ± 0.004	40	215	≈ 0	15.6 ± 1.0	1748
B7 (ours, exact)	0.000	11.4 ± 1.0	2065	≈ 0	15.6 ± 1.0	1698
B8 (Townsend et al. (2016))	0.000	40	70	≈ 0	11.8 ± 1.6	1.6
B9 (proj. Hessian)	0.008 ± 0.004	40	206	≈ 0	15.6 ± 1.0	672
B10 (Taylor jet)	0.008 ± 0.004	40	222	≈ 0	15.6 ± 1.0	1050

Weil-R converges in the fewest iterations on T1 (11.4 ± 1.0 vs. Townsend et al. (2016) 40; Table 3). B5-opt, B9, and B10 are bit-identical and stall at rel gap 0.008 after 40 iterations on T1. B8 (Townsend et al. (2016)) is fastest in wall-clock time (T1: 70s, T2: 1.6s), but this reflects its C-language backend: B7 takes 2065s on T1 and 1698s on T2, i.e., $30\times$ and $10^3\times$ longer than B8 respectively. Our claim is on iteration count and exactness, not raw speed: the exact Weil Hessian enables quadratic convergence to machine precision in 11 iterations where approximate Hessians stall. The wall-clock gap is a JAX implementation limitation (Python overhead in the tangent-basis construction and CG inner loop), not an algorithmic one; a compiled CUDA kernel for the algebra contractions would close this gap. B5 $\equiv$ B9 $\equiv$ B10, all three compute the ambient Hessian projected to the tangent space, which differs from the Riemannian Hessian by the Weingarten map. For quadratic-in-X objectives, the ambient Hessian is constant, so all three methods produce identical search directions and bit-identical trajectories. On the Rayleigh quotient at $n = 512$, this systematic error prevents convergence: the optimizer stalls at 0.008 while our covariant Hessian converges in 11.4 iterations (Figure 4 in Appendix G).

6 Conclusion

We have shown that task-adapted Weil algebras, lifted through constraint-preserving manifold retractions, compute exact covariant second derivatives on matrix manifolds in a single forward pass. The mechanism is general: the same four-step procedure—embed, retract, evaluate, extract—applies identically to the sphere, the Stiefel manifold, and the SPD cone, and extends to any smooth manifold admitting an A-valued retraction. These design choices are validated against the experimental evidence: Riemannian Newton and trust-region with the exact Weil Hessian converge to machine precision on five of seven test problems and to the best recorded optimum on the remaining two (Tables 1–3 and Figure 4), while flat-space Weil evaluation is wrong by ~ 0.5 on the same problems (Table 16), confirming that the retraction-based lifting, not the algebra itself, is the essential contribution. The main limitations are wall-clock performance (a JAX implementation detail, not an algorithmic one), the per-problem cost of writing A-valued loss twins for novel objectives, current coverage of three matrix manifolds, and restriction to second-order derivatives. The natural next steps are compiled XLA/CUDA algebra kernels, A-valued retractions for Grassmann, SO(n), fixed-rank, and product manifolds, sparse quotient algebras for targeted higher-order extraction, and integration into downstream applications requiring second-order information on manifolds, such as Gaussian-mixture estimation, Riemannian natural-gradient methods, and structure-from-motion.

AI use statement

In this work, we used Grammarly and ChatGPT for grammar checking and phrasing suggestions on the manuscript text, and AI for debugging the code. generative AI tools were not used to produce the results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

This research uses standard public data and presents no immediate ethical or societal concerns.

Reproducibility statement

We include the details of the implementation and environment, task objectives, and all hyperparameters in the appendix. The complete JAX implementation, including the four-layer library (algebra, primitives, manifolds, covariant), optimizers, baselines, and All experiment scripts will be released after acceptance, and an anonymized version can be provided upon request during review.

References

P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

P-A Absil, Robert Mahony, and Jochen Trumpf. An extrinsic look at the Riemannian Hessian. In *International conference on geometric science of information*, pp. 361–368. Springer, 2013.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018.

Jesse Bettencourt, Matthew J Johnson, and David Duvenaud. Taylor-mode automatic differentiation for higher-order derivatives in JAX. In *Program transformations for ML workshop at NeurIPS 2019*, 2019.

Nicolas Boumal. A riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints. *arXiv preprint arXiv:1506.00575*, 2015.

Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press,(Apr 2022), 2023.

Nicolas Boumal, Rodolphe Sepulchre, Pierre-Antoine Absil, and Bamdev Mishra. Manopt, a matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15, 2014.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Felix Dangel, Tim Siebert, Marius Zeinhofer, and Andrea Walther. Collapsing taylor mode automatic differentiation. *Advances in Neural Information Processing Systems*, 38:147127–147166, 2026.

Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Jeffrey Fike and Juan Alonso. The development of hyper-dual numbers for exact second-derivative calculations. In *49th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, pp. 886, 2011.

Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2nd edition, 2008.

Mathieu Huot, Sam Staton, and Matthijs Vákár. Correctness of automatic differentiation via diffeologies and categorical gluing. In *FoSSaCS*, pp. 319–338, 2020.

Hiromi Ishii. Automatic differentiation with higher infinitesimals, or computational smooth infinitesimal analysis in weil algebra. In *International Workshop on Computer Algebra in Scientific Computing*, pp. 174–191. Springer, 2021.

Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in pytorch, 2020. https://github.com/geoopt/geoopt.

Anders Kock. *Synthetic differential geometry*, volume 2. Cambridge University Press, 1981.

Leonid Koppel and Steven L Waslander. Manifold geometry with fast automatic derivatives and coordinate frame semantics checking in c++. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pp. 126–133. IEEE, 2018.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International conference on learning representations*, volume 2024, pp. 1621–1650, 2024.

Mayank Meghwanshi, Pratik Jawanpuria, Anoop Kunchukuttan, Hiroyuki Kasai, and Bamdev Mishra. Mctorch, a manifold optimization library for deep learning. Technical report, arXiv preprint arXiv:1810.01811, 2018.

Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, et al. Geomstats: A Python package for Riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020.

Ieke Moerdijk and Gonzalo E Reyes. *Models for smooth infinitesimal analysis*. Springer Science & Business Media, 2013.

Alexander Novikov, Maxim Rakhuba, and Ivan Oseledets. Automatic differentiation for riemannian optimization on low-rank matrix and tensor-train manifolds. *SIAM Journal on Scientific Computing*, 44(2): A843–A869, 2022.

Michael Psenka and Nicolas Boumal. Second-order optimization for tensors with fixed tensor-train rank, 2020.

Jarrett Revels, Miles Lubin, and Theodore Papamarkou. Forward-mode automatic differentiation in julia, 2016.

Hiroyuki Sakai and Hideaki Iiduka. A general framework of riemannian adaptive optimization methods with a convergence analysis, 2024.

Amandip Sangha. Jet functors and weil algebras in automatic differentiation: A geometric analysis, 2025a.

Amandip Sangha. Spectral-geometric deformations of function algebras on manifolds, 2025b.

Lena Sembach, Jan Pablo Burgard, and Volker Schulz. A riemannian newton trust-region method for fitting gaussian mixture models. *Statistics and Computing*, 32(1):8, 2022.

Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

Songchen Tan. Taylordiff.jl: Fast higher-order automatic differentiation in julia. `https://github.com/JuliaDiff/TaylorDiff.jl`, 2022.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016. URL `http://jmlr.org/papers/v17/16-177.html`.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

A Weil bundles: transition laws and lifted constraints

For a Weil algebra A, an *A-point* of a manifold M over $x \in M$ is a unital algebra homomorphism $\xi : C^\infty(M) \to A$ whose composition with the projection $A \to \mathbb{R}$ is evaluation at x. The A-points over all base points form the *Weil bundle* $\pi_A : T^A M \to M$ (Kock, 1981; Moerdijk & Reyes, 2013). A retraction lift $R_x(U_A)$ evaluates the manifold constraint and the objective on the infinitesimal data U_A of equation 7, which is exactly such an A-point; for a constraint-preserving retraction the result lies in the sub-bundle of A-points that solve the constraint.

Fiber dimension. In local coordinates the fibre over x is modeled on $\mathfrak{m}^{\dim M}$; with $d = \dim M$,

$$\dim T^A M = d \cdot \dim_{\mathbb{R}} A, \qquad \dim T^A_x M = d\big(\dim_{\mathbb{R}} A - 1\big). \tag{12}$$

For the hyper-dual algebra $\dim_{\mathbb{R}} A_{\mathrm{HD}} = 4$, so $\dim T^{A_{\mathrm{HD}}} M = 4d$ and each fibre has dimension $3d$. The four real coefficients of a hyper-dual element, base, two first-order, and one mixed, are not the fibre dimension.

Mixed coefficients transform together. In local coordinates write an A_{HD}-point as

$$z = x + \varepsilon_1 v + \varepsilon_2 w + \varepsilon_1\varepsilon_2 a. \tag{13}$$

Under a nonlinear coordinate change $y = \phi(x)$, Taylor expansion gives

$$y_0 = \phi(x), \qquad v_y = D\phi_x\, v, \qquad w_y = D\phi_x\, w, \qquad a_y = D\phi_x\, a + D^2\phi_x[v, w]. \tag{14}$$

The mixed coefficient does *not* transform as an independent tangent vector: dropping the term $D^2\phi_x[v, w]$ changes the represented jet. Concretely, on the line take $f(x) = x$, $x = 0$, $v = w = 1$; in the coordinate $y = e^x$ the same function becomes $\tilde{f}(y) = \log y$. Seeding the fresh coordinates with $1+\varepsilon_1+\varepsilon_2$ gives the mixed coefficient -1, whereas transforming the original jet $x = \varepsilon_1+\varepsilon_2$ through $y = \exp(\varepsilon_1+\varepsilon_2) = 1+\varepsilon_1+\varepsilon_2+\varepsilon_1\varepsilon_2$ returns 0, as it must for a flat function. The discrepancy is a matter of consistent jet representation, not curvature; equation 14 is the systematic law that enforces it.

Covariant extraction and the connection. Fix a Riemannian metric g and its Levi-Civita connection Γ. The seeds v, w with mixed coefficient $a = -\Gamma_x(v, w)$ extract the covariant Hessian,

$$\operatorname{Hess} f(x)[v, w] = D^2 f_x[v, w] - df_x\big(\Gamma_x(v, w)\big), \tag{15}$$

and equation 14 is then exactly the transformation law of the connection (in the example above, $g_{yy} = y^{-2}$ and $\Gamma^y_{yy} = -y^{-1}$, giving $a_y = 1$ at $y = 1$). Weil functoriality alone does not select a metric or connection; covariant extraction requires this additional structure, which the constraint-preserving retractions of Theorem 2 supply.

Lifted constraint identities. For an embedded manifold $M = \{z : C(z) = 0\}$, a correctly lifted A-point satisfies $T^A C(z) = 0$ in A. For the sphere $S^{n-1} = \{x : x^\top x = 1\}$ this yields, coefficient by coefficient,

$$x^\top x = 1, \qquad x^\top v = x^\top w = 0, \qquad x^\top a + v^\top w = 0, \tag{16}$$

and for the Stiefel manifold $\mathrm{St}(n, p) = \{X : X^\top X = I_p\}$,

$$X^\top X = I_p, \qquad X^\top v = X^\top w = 0, \qquad X^\top a + a^\top X + v^\top w + w^\top v = 0. \tag{17}$$

The mixed identities couple the mixed coefficient to the first-order directions: treating all coefficients as independent tangent vectors violates the lifted constraint. These identities serve as automated diagnostics in the E1 verification suite (Section 5).

Composition and quotient consistency. The Weil functor respects composition, $T^A(G \circ F) = T^A G \circ T^A F$, and unital quotient morphisms, $q_N \circ T^A F = T^B F \circ q_M$ for $B = A/I$. These identities underpin the layer composition of Section 4 and the sparse-quotient algebra selection of the Weil algebra compiler: retained monomials must form a divisor-closed set for the quotient to remain a valid Weil algebra.

B Additional related work: Riemannian optimisation

Absil et al. (2008) established the theory of optimization on matrix manifolds, and Absil et al. (2007) proved trust-region convergence on Riemannian manifolds; their framework provides the outer loop for our Newton and trust-region methods. The extrinsic decomposition of the Riemannian Hessian, $\mathrm{Hess}_M f = P\,\nabla^2_{\text{ambient}} f\,P$+Weingarten term, is derived by Absil et al. (2013) and is the formula that projected-Euclidean baselines (B4, B9) attempt to approximate. Edelman et al. (1998) provide the foundational analysis of optimization on the Stiefel manifold. More recent methodology adds Riemannian quasi-Newton variants (Boumal, 2015), a comprehensive textbook treatment (Boumal, 2023), Riemannian adaptive methods (Sakai & Iiduka, 2024), and trust-region Gaussian-mixture fitting on manifolds (Sembach et al., 2022). All of these methods take an explicit second-order Hessian model as input; our framework supplies that model automatically and exactly.

Differentiating through manifold-constrained programs has received growing attention. Novikov et al. (2022) developed Riemannian AD for low-rank tensor manifolds (T3F) but explicitly *omit* the curvature (Weingarten) correction, yielding a constrained Gauss–Newton approximation; Psenka & Boumal (2020) extended second-order methods to fixed tensor-train rank with the same curvature-omission strategy. Earlier robotics work (Koppel & Waslander, 2018) applied AD through manifold structure but stopped at the Jacobian level, never extracting covariant second-order information. These works share a common pattern: they project an ambient derivative rather than compute an intrinsic one. We quantify the cost of this omission through a projected-Euclidean baseline (B9) that is bit-identical to the flat Weil and Taylor-mode baselines on quadratic objectives and stalls at relative gap 0.008 on the $n = 512$ Rayleigh quotient, where our covariant Hessian converges in 11.4 iterations.

A mature ecosystem implements first- and second-order optimization on manifolds: Manopt (Boumal et al., 2014), PyManopt with a JAX backend (Townsend et al., 2016), Geomstats (Miolane et al., 2020), Geoopt (Kochurov et al., 2020), and McTorch (Meghwanshi et al., 2018). All of these expose an `ehess2rhess` routine that maps a Euclidean Hessian to a Riemannian one via the Weingarten map, precisely baseline B4, and none provides an intrinsic, chart-independent exact Hessian. We use PyManopt both as the strongest member of this family (baseline B8) and to check our trust-region implementation.

For completeness we note recent progress on second-order methods in deep learning, where cheap curvature approximations such as Hessian diagonals accelerate training: AdaHessian (Yao et al., 2021), and Sophia (Liu et al., 2024). These mirror our diagonal- and identity-Hessian ablations (E3): they approximate curvature for per-parameter scaling rather than extract exact search directions. They operate in Euclidean space and are not directly comparable on manifold-constrained problems; we include them only to motivate the ablation design, not as optimizer competitors.

The landscape of related work divides into four groups:

1. **Flat-space Weil/Taylor AD** (Fike & Alonso, 2011; Ishii, 2021; Sangha, 2025a; Bettencourt et al., 2019): the algebraic machinery for one-pass higher-order derivatives. We build on it by adding the manifold layer, and we show that without it the machinery computes the ambient Hessian, which is wrong by a $\Theta(1)$ term on manifolds.
2. **Riemannian optimization libraries** (Boumal et al., 2014; Townsend et al., 2016; Miolane et al., 2020; Kochurov et al., 2020; Meghwanshi et al., 2018): the algorithmic framework (trust-region, RCG, GD) and first-order pipelines. Their second-order story ends at `ehess2rhess` (B4), which we implement and compare against.
3. **Manifold AD / curvature omission** (Novikov et al., 2022; Psenka & Boumal, 2020): these drop the Weingarten term from the Hessian model. We quantify the cost of this omission via a projected-Euclidean baseline (B9) that stalls where our covariant Hessian converges.
4. **Theory / foundations** (Kock, 1981; Moerdijk & Reyes, 2013; Absil et al., 2013): the mathematical language, Weil algebras as infinitesimal objects, the extrinsic Hessian decomposition, that structures our proof and analysis.

Our contribution focuses on (a) lifting Weil-algebra evaluation through constraint-preserving retractions, (b) extracting covariant, chart-independent derivatives up to the requested order from a single forward pass, and (c) demonstrating the practical consequence: exact second-order models enable Newton and trust-region methods to converge to machine precision on problems where first-order and projected-Hessian methods fail or stall.

C Proof of Theorem 2

Proof of Theorem 2. Define the retraction curve

$$\gamma(s,t) = R_x(sv + tw) \in M. \tag{18}$$

By (R1), $\gamma(0,0) = x$. By (R2), the tangent vectors at the origin are

$$\frac{\partial \gamma}{\partial s}(0,0) = v, \qquad \frac{\partial \gamma}{\partial t}(0,0) = w. \tag{19}$$

Step 1: Taylor expansion of the retraction curve. Since M is embedded in ambient Euclidean space (or equivalently, we work in a local coordinate chart), the retraction curve has a Taylor expansion in the ambient space:

$$\gamma(s,t) = x + sv + tw + \tfrac{1}{2}s^2\, a_{vv} + st\, \gamma''_{v,w} + \tfrac{1}{2}t^2\, a_{ww} + O(|(s,t)|^3), \tag{20}$$

where $a_{vv}, a_{ww} \in T_xM \oplus N_xM$ are the second-order retraction accelerations in the two single-variable directions, and $\gamma''_{v,w}$ is the mixed acceleration defined in equation 6. The key observation is that $\gamma''_{v,w}$ lies in T_xM: the retraction maps tangent vectors to the manifold, so the mixed second derivative of the retraction curve is a tangent vector at x.

Step 2: Taylor expansion of f along the curve. By the chain rule applied to $f \circ \gamma$:

$$f(\gamma(s,t)) = f(x) + df_x[sv + tw] + \tfrac{1}{2}D^2 f_x[sv + tw, sv + tw] + df_x[\gamma''_{v,w}]\, st + O(|(s,t)|^3), \tag{21}$$

where df_x is the differential and $D^2 f_x$ is the second Fréchet derivative in ambient coordinates. Collecting the mixed (st) terms:

$$\frac{\partial^2}{\partial s\, \partial t} f(\gamma(s,t))\Big|_{s=t=0} = D^2 f_x[v,w] + df_x[\gamma''_{v,w}]. \tag{22}$$

Step 3: Relating ambient to Riemannian Hessian. The Riemannian Hessian on a Riemannian submanifold (or more generally, with the Levi-Civita connection) satisfies

$$\operatorname{Hess} f(x)[v,w] = D^2 f_x[v,w] + df_x[\mathrm{II}(v,w)], \tag{23}$$

where $\mathrm{II}(v,w)$ is the second fundamental form (normal component of the ambient acceleration of the geodesic). For the *exponential map*, the retraction curve is a geodesic, so $\gamma''_{v,w} = \mathrm{II}(v,w)$ (the geodesic curvature is exactly the second fundamental form on a submanifold), and equation 22 becomes

$$\frac{\partial^2}{\partial s\, \partial t} f(\exp_x(sv + tw))\Big|_{s=t=0} = D^2 f_x[v,w] + df_x[\mathrm{II}(v,w)] = \operatorname{Hess} f(x)[v,w]. \tag{24}$$

For a general second-order retraction R, the retraction curve $\gamma(s,t) = R_x(sv+tw)$ agrees with the geodesic $\exp_x(sv+tw)$ through first order, but the second-order term $\gamma''_{v,w}$ may differ from $\mathrm{II}(v,w)$. The difference is the *retraction error*:

$$\gamma''_{v,w} = \mathrm{II}(v,w) + \delta_{v,w}, \tag{25}$$

where $\delta_{v,w}$ depends on the specific retraction. Substituting into equation 22:

$$\frac{\partial^2}{\partial s\, \partial t} f(R_x(sv + tw))\Big|_{s=t=0} = D^2 f_x[v,w] + df_x[\mathrm{II}(v,w) + \delta_{v,w}] = \operatorname{Hess} f(x)[v,w] + df_x[\delta_{v,w}]. \tag{26}$$

This is the ambient-coordinate form of equation 9 with $\gamma''_{v,w} = \mathrm{II}(v,w) + \delta_{v,w}$.

Step 4: Re-expressing in metric form. Since $df_x[u] = g_x(\operatorname{grad} f(x), u)$ for any tangent vector u, and $\gamma''_{v,w} \in T_xM$ (because the retraction maps into M), we obtain

$$df_x[\gamma''_{v,w}] = g_x(\operatorname{grad} f(x),\ \gamma''_{v,w}), \tag{27}$$

which is the second term in equation 9.

Step 5: Algebraic evaluation yields the mixed coefficient. The hyper-dual evaluation $F_A = f(R_x(U_A))$ with $U_A = v\varepsilon_1 + w\varepsilon_2$ encodes the Taylor expansion of $f \circ R_x$ in the algebra A_{HD}:

$$F_A = f(x) + \varepsilon_1\, df_x[v] + \varepsilon_2\, df_x[w] + \varepsilon_1\varepsilon_2 \frac{\partial^2}{\partial s\, \partial t} f(R_x(sv + tw))\Big|_{s=t=0} + (\text{terms in } \varepsilon_1^2, \varepsilon_2^2 = 0). \tag{28}$$

The ε_1^2 and ε_2^2 terms vanish by nilpotency. The coefficient of $\varepsilon_1\varepsilon_2$ is exactly the mixed second derivative from Step 3, which equals $\operatorname{Hess} f(x)[v,w] + g_x(\operatorname{grad} f(x), \gamma''_{v,w})$ by equation 9. □ □

D Manifold examples

D.1 Sphere

The sphere $S^{n-1} = \{x \in \mathbb{R}^n : \|x\|^2 = 1\}$ carries the projection retraction

$$R_x(v) = \frac{x+v}{\|x+v\|}. \tag{29}$$

The retraction curve $\gamma(s,t) = R_x(sv+tw)$ has acceleration

$$\gamma''_{v,w} = -x\, g_x(v,w). \tag{30}$$

Therefore

$$\mathrm{Re}_{11}(F_A) = \mathrm{Hess}\, f(x)[v,w] - g_x(\mathrm{grad}\, f(x), x)\, g_x(v,w) = \mathrm{Hess}\, f(x)[v,w], \tag{31}$$

since $\mathrm{grad}\, f(x) \perp x$ for f restricted to S^{n-1} (the ambient gradient is tangent to the sphere). The projection retraction on the sphere gives the exact Hessian.

This explains the experimental finding that Weil-algebra HVPs on the sphere match the reference Riemannian Hessian to machine precision (E1: relative error $\sim 10^{-16}$).

D.2 Stiefel manifold

The Stiefel manifold $\mathrm{St}(n,p) = \{X \in \mathbb{R}^{n\times p} : X^\top X = I_p\}$ carries the QR retraction

$$R_X(Z) = \mathrm{qf}(X+Z), \tag{32}$$

where qf denotes the Q factor of the thin QR decomposition. The retraction curve $R_X(sV+tW)$ for $V, W \in T_X\,\mathrm{St}$ satisfies the symmetry

$$R_X(sV+tW) = R_X(tW+sV) \qquad \text{for all } s,t. \tag{33}$$

This symmetry forces the mixed acceleration $\gamma''_{V,W} = \gamma''_{W,V}$. A direct computation (using the $\varepsilon_1\varepsilon_2$-coefficient of QR decomposition of $X + V\varepsilon_1 + W\varepsilon_2$) shows that the mixed acceleration of QR retraction vanishes:

$$\gamma''_{V,W} = 0 \qquad \text{for all } V, W \in T_X\,\mathrm{St}\,. \tag{34}$$

Therefore $\mathrm{Re}_{11}(F_A) = \mathrm{Hess}\, f(X)[V,W]$ exactly.

D.3 Symmetric positive-definite cone

The SPD cone $\mathrm{SPD}(n)$ is an open submanifold of the vector space of symmetric matrices, so the ambient connection is flat and the Riemannian Hessian is simply the Euclidean Hessian restricted to the tangent space. The affine-invariant retraction

$$R_X(Z) = X^{1/2} \exp(X^{-1/2} Z X^{-1/2}) X^{1/2} \tag{35}$$

is the exponential map, so $\gamma''_{V,W} = 0$ and $\mathrm{Re}_{11}(F_A) = \mathrm{Hess}\, f(X)[V,W]$ exactly.

E Implementation details

E.1 Weil algebra compiler

The algebra module implements a finite-dimensional nilpotent commutative $\mathbb{R}$-algebra $A = \mathbb{R}[\varepsilon_1, \ldots, \varepsilon_P]/I$ with an ordered monomial basis. The representation is a real array whose last axis indexes the monomial basis; the coefficient at index 0 is always the unit monomial (the "real part").

E.1.1 Multiplication tensor

The core data structure is a multiplication tensor $M \in \mathbb{R}^{D\times D\times D}$, where $M_{a,b,k} = 1$ if $\mathrm{monomial}_a \cdot \mathrm{monomial}_b = \mathrm{monomial}_k$, and 0 otherwise. This tensor is constructed once from the monomial exponent vectors and stored as a JAX array. All algebra operations reduce to contractions with M:

$$(a \cdot b)_k = \sum_{a,b} M_{a,b,k}\, a_a\, b_b. \tag{36}$$

In practice, this is dispatched as a single `jnp.einsum` call:

```
def amul(W, a, b):
    """Pointwise product of A-valued tensors."""
    return jnp.einsum("...a,...b,abk->...k", a, b, W.M)
```

For matrix-level operations (`amatmul`), the contraction is over the spatial axes in addition to the algebra axis:

```
def amatmul(W, A, B):
    """A-valued matrix product: C_{ij} = sum_k A_{ik} B_{kj}."""
    return jnp.einsum("...ik,...kj,abk->...ij", A, B, W.M)
```

E.1.2 Algebra variants

Four constructors cover the algebras used in this paper:

- `WeilAlgebra.dual()`: $D = 2$, generators $\{\varepsilon\}$, $\varepsilon^2 = 0$. First derivatives only.
- `WeilAlgebra.hyperdual()`: $D = 4$, generators $\{\varepsilon_1, \varepsilon_2\}$, $\varepsilon_1^2 = \varepsilon_2^2 = 0$. Gradient + one HVP.
- `WeilAlgebra.one_generator(r)`: $D = r + 1$, single generator $\varepsilon^{r+1} = 0$. Full rth-order Taylor jet.
- `WeilAlgebra.sparse(queries)`: downward-closed quotient from a list of requested multi-indices.

All constructors precompute the monomial list, the multiplication tensor M, and the product index matrix $\texttt{prod}(a, b) = k$ (or -1 if the product vanishes).

E.2 Manifold lifting

Each manifold class exposes five methods:

1. `retract_real(X, Z)`: real retraction $R_X(Z)$ (for baselines).
2. `retract(W, X, U_A)`: A-valued retraction $R_X(U_A)$ preserving the constraint coefficient-by-coefficient.
3. `exp_map(W, X, U_A)`: A-valued exponential map (where available; raises `NotImplementedError` otherwise).
4. `project_tangent_real(X, G)`: real tangent projection.
5. `project_tangent(W, X, G_A)`: A-valued tangent projection.

E.2.1 A-valued modified Gram–Schmidt (Stiefel)

The Stiefel retraction requires orthonormalising the A-valued matrix $Y = X + U_A$ column-by-column. Standard QR is not available in the algebra, so we use a modified Gram–Schmidt procedure that replaces scalar inner products with the algebra inner product `ainner` and scalar norms with the algebra square root `asqrt`:

```
def _mgs_orthonormalize(W, Y):
    """A-valued MGS: orthonormalise Y column-by-column."""
    p = Y.shape[-2]
    cols = []
    for j in range(p):
        v = Y[..., :, j, :]
        for i in range(j):
            r = ainner(W, cols[i], v)
            v = v - amul_scalar(W, r, cols[i])
        nrm = asqrt(W, ainner(W, v, v))
        v = amul_scalar(W, ainv(W, nrm), v)
        cols.append(v)
    return jnp.stack(cols, axis=-2)
```

This procedure preserves the Stiefel constraint $X^\top X = I$ to machine precision at every nilpotent order (verified in E1: constraint residual $\sim 10^{-15}$).

E.2.2 SPD retraction

The SPD retraction is the second-order approximation to the affine-invariant exponential map:

$$R_X(Z) = X + Z + \tfrac{1}{2} Z X^{-1} Z. \tag{37}$$

The A-valued version replaces the real matrix inverse with `ainv_matrix` (a Neumann-series expansion that terminates at order r by nilpotency) and all multiplications with `amatmul`.

E.3 Optimiser details

E.3.1 Steihaug truncated-CG

The inner solver accepts either a dense $k \times k$ matrix or a callable matvec `H(v)`. In the Weil-R path, it always receives the callable form (the HVP action), so the Hessian is *never* materialized. The algorithm detects negative curvature ($p^\top H p \leq 0$) and steps to the trust-region boundary, which is essential for convergence on indefinite Hessians (Thomson problem).

E.3.2 Trust-region safeguard

On strongly indefinite problems (e.g. the Thomson energy with $q = 5$), the quadratic model may report no descent direction at some iterations ($\rho = -\infty$). We add a backtracking gradient-step safeguard: when $\rho = -\infty$, a backtracking line search along the negative gradient is attempted with step sizes $\alpha \in \{10^{-2}, 5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$; the first step yielding descent is accepted, and the trust radius is restored. This safeguard is never triggered on well-conditioned problems (T1–T5); it fires only on the Thomson problem (7–12 of 40 iterations), where it preserves convergence without affecting the exactness of the Hessian computation.

E.4 Memory and JAX details

The dense Hessian matrix ($k \times k$) requires $O(k^2)$ memory: at $n = 512$, $p = 5$, $k = 2545$, the full matrix occupies ~ 50 MB. The HVP action requires only $O(c \cdot n \cdot p \cdot D)$ memory at chunk size c; at $c = 128$, this is ~ 3 MB on a V100, a factor of 10^4 below the dense allocation.

The library is implemented in JAX (Bradbury et al., 2018) with no custom CUDA kernels. Key JAX features used:

- **JIT compilation**: `@jax.jit` on the HVP block function eliminates Python overhead between CG iterations.
- **Vectorisation**: `jax.vmap` batches the k covariant evaluations in the HVP.
- **XLA compilation**: all `einsum` calls are fused by XLA into a single kernel launch.
- **64-bit precision**: `jax_enable_x64=True` is required for machine-precision accuracy.

The numpy/JAX boundary is maintained at the layer interface: orchestration uses NumPy on CPU; elementwise algebra operations and covariant evaluations use JAX on the accelerator.

E.5 Method, Code, Hardware and software

The implementation requires JAX $\geq$ 0.4.20 (with CUDA support for GPU experiments) and NumPy; the core library needs no other dependencies. The external baselines and comparisons (B5-opt, B8, B9, B10) require optional packages (`jet-weil-ad`, `pymanopt`). All experiments run on NVIDIA V100-PCIE-32GB GPUs or multi-core CPUs, using JAX 0.4.25, Python 3.11.15, with `jax_enable_x64=True`.

- **GD**: Gradient Descent with a fixed step-size (first-order).
- **RCG**: Riemannian Conjugate Gradient (first-order).
- **Newton**: Riemannian Newton with exact Weil Hessian, backtracking line search, trust-region CG inner solve, gradient-descent fallback on non-descent directions.
- **TR**: Riemannian trust-region (Absil et al. (2008) Algorithm 10) with exact Weil HVP action, Steihaug truncated-CG inner solve, gradient-step safeguard on model misprediction.

Table 4: Baseline engine methods. "Correct" indicates whether the Hessian model equals the true Riemannian Hessian (exact) or not (approximate).

Baseline (B)	Method	Hessian	Correct
B1	Finite differences	Central differences	Approx.
B2	Hyper-dual numbers	Real hyper-dual one-shot	Exact
B3	Nested JVP	Nested forward-mode AD	Exact
B4	Projected reverse	Weingarten projection	Exact
B5	Euclidean Weil	Flat-space Weil	Wrong
B6	Hessian diagonal query	Diagonal of Weil Hessian	Exact
B7 (ours)	Chunked HVP	Exact Weil HVP action	Exact
B8	Townsend et al. (2016) TR	`ehess2rhess` projection	Exact
B9	Projected-Euclidean TR	Nested JVP + tangent proj.	Approx.
B10	Taylor-jet TR	Taylor-mode jet expansion	Approx.

Table 5: Test problems.

ID (Task)	Problem	Manifold	Objective
T1	Rayleigh quotient	$\mathrm{St}(n,p)$	$f(X) = -\operatorname{tr}(X^\top \Sigma X)$
T2	Low-rank completion	$\mathrm{St}(n,p)$	$f(X) = \frac{1}{2}\|XX^\top - M\|_F^2$
T3	Procrustes	$\mathrm{St}(n,p)$	$f(X) = -\operatorname{tr}(X^\top B)$
T4	SPD Karcher mean	$\mathrm{SPD}(n)$	$f(X) = \sum_i d(X, X_i)^2$
T5	Gaussian MLE	$\mathrm{SPD}(n)$	$f(\Sigma) = \operatorname{tr}(\Sigma^{-1}\hat{\Sigma}) + \log\det\Sigma$
T6	Robust low-rank PCA	$\mathrm{St}(n,p)$	$f(U) = \frac{1}{m}\sum_i \log(1 + \|(I - UU^\top)x_i\|^2)$
T7	Variable stiffness learning	$\mathrm{SPD}(n)$	$f(\{K_t^{(d)}\}) = \text{EM-GMM}\big(\{K_t^{(d)}\}_{d=1}^N\big)$
T8	Oblique logistic	S^{n-1}	logistic negative log-likelihood on sphere

Table 5 details the eight test problems, where T1–T6 and T8 are standard numerical optimization problems. The variable stiffness learning task (T7) arises in robot learning from demonstration (**?**): a robot's end-effector stiffness is represented by a symmetric positive-definite matrix $K_t^{(d)} \in \mathrm{SPD}(n)$ at each time step t in each demonstration d. The learning objective is to estimate a time-varying stiffness profile by fitting a Gaussian Mixture Model (GMM) directly on the SPD manifold via the Expectation-Maximization algorithm, followed by Gaussian Mixture Regression (GMR) to produce smooth stiffness trajectories for reproduction. The per-timestep loss is the Frobenius-norm surrogate

$$f(K) = \|X^\top X \cdot K - X^\top Y\|_F^2, \quad K \in \mathrm{SPD}(n), \tag{38}$$

where X and Y encode the demonstrated force-displacement pairs. This is a non-convex problem on the SPD cone; we solve it with a single trust-region pass using the exact Weil Hessian, replacing the 28-run Particle Swarm Optimization originally used by **?**.

F Experimental hyperparameters

The tables below list the hyperparameters used in the experiments.

E3: per-seed results ($n = 512$, $p = 5$)

E4: per-seed external comparison ($n = 512$)

E3 ablation: full results (Thomson $q = 5$)

Note. The Thomson problem ($q = 5$) is strongly indefinite far from the optimum. All four Newton models eventually reach the floor, but the exact Weil HVP model requires more fallback iterations (21 of 40) because the trust-region safeguard fires frequently when the quadratic model is accurate but the Hessian is indefinite. The diagonal and identity models mispredict less often (0 fallbacks) not because they are better, but because they are too coarse to detect the indefiniteness; they converge more slowly in iterations-to-floor despite fewer

Table 6: E3: hyperparameters for the baseline experiment ($n = 512$, $p = 5$, 5 seeds).

Parameter	Value
Algebra	Hyper-dual ($D = 4$)
Manifold (T1)	Stiefel $\mathrm{St}(512, 5)$, $\dim T_x M = 2545$
Manifold (T2)	Stiefel $\mathrm{St}(512, 5)$
Objective (T1)	$f(X) = -\operatorname{tr}(X^\top \Sigma X)$, Σ random SPD
Objective (T2)	$f(X) = \frac{1}{2}\|XX^\top - M\|_F^2$, M rank-p
Observation fraction (T2)	0.3
Trust-region Δ_0	1.0
Trust-region Δ-	1.0
Trust-region ρ'	0.1
CG max inner iterations	200
CG tolerance	10^{-8}
Gradient tolerance	10^{-8}
Max outer iterations	40
Newton backtracking	$\alpha \in \{1, 0.5, 0.25, 0.125, 0.0625, 0.03125\}$
Newton fallback lr	10^{-3}
GD step size	0.02 (fixed)
RCG step size	1.0 (backtracking)
HVP chunk size	128
Seeds	$\{0, 1, 2, 3, 4\}$ (independent)

Table 7: E3: hyperparameters for generality experiments (T3–T8, Thomson).

Parameter	T3/T4 ($n = 8/16$)	Thomson/T5/T8 ($q = 4/5$, $n = 6$)
Algebra	Hyper-dual ($D = 4$)	Hyper-dual ($D = 4$)
Max outer iterations	40	40
Seeds	5	5
GD step size	0.02	0.02
Trust-region Δ_0	1.0	1.0
CG max inner	200	200

fallbacks. On well-conditioned problems (T1–T5), the exact model achieves 0 fallbacks and machine-precision convergence in 7–20 iterations.

G Additional experimental details

G.1 E1: full invariance protocol

The verification uses references: analytic embedded-submanifold Hessian formula (sphere), complex-step (analytic functions), and FP64 nested JVP (B3). Three invariance families are tested:

1. **Retraction-order invariance**: Weil Hessian via exponential map vs. order-2 retraction vs. analytic sphere formula. Also tested on the Stiefel manifold: A-valued MGS-QR vs. A-valued polar retraction.
2. **Chart invariance**: geodesic normal coordinates (Christoffels vanish at origin) under two random orthonormal frames; stereographic north/south overlap via metric pull-back.
3. **Embedding equivariance**: orthogonal conjugation leaves the dense $k \times k$ coordinate-Hessian unchanged (Stiefel, SPD).

G.2 E2: memory estimation methodology

Weil-R method's wall time scales linearly in $\dim A$, and peak memory is $O(D)$ versus $O(D^2)$ for the dense block, with a linear fit $R^2 \geq 0.988$ at two scales (Figure 2). Memory is estimated via static liveness

Table 8: E1c: consistency check parameters.

Parameter	Value
Manifold	Sphere, Stiefel, SPD
Sizes n	$\{8, 12, 16\}$
Seeds	$\{0, 1, 2, 3\}$ (per size)
Tolerance (gated check)	10^{-9}
Number of checks	51 (17 families $\times$ 3 checks each)

Table 9: E1c: breakdown of the 51 gated consistency checks.

Family	Check	Count	Description
T1: Retraction invariance	exp vs order-2 retract (sphere)	12	3 seeds $\times$ 4 pairs
	Weil-retract vs analytic (sphere)	12	3 seeds $\times$ 4 pairs
	exp-loss consistency (sphere)	1	single pair
T2: Chart invariance	normal-coords gradient	6	2 frames $\times$ 3 coords
	normal-coords Hessian	8	4 pairs: Weil vs normal-coords
T2: Stereographic	north vs south gradient	3	2 charts + overlap
T3: Embedding equivariance	coord-Hessian matrix (Stiefel + SPD)	2	orthogonal conjugation
	gradient action invariance (Stiefel)	1	single direction
T4: Hessian diagonal	Weil vs projected-reverse (sphere)	1	B6 reference
	Weil vs nested-JVP (Stiefel)	1	same-curve reference
	Weil vs nested-JVP (SPD)	1	same-curve reference
	bilinear symmetry (sphere + Stiefel)	2	$\|H - H^\top\|_{\max}$
Stiefel-polar	constraint residual	1	$\max \|R^\top R - I\|$
Stiefel-polar (diag)	MGS-QR vs polar (diagnostic)	1	not gated
Total gated		**51**	

analysis over the traced `jaxpr` (recursive, float64, accounts for sub-jaxprs) and cross-validated against device-level `memory_stats` on GPU. The dense block grows as $O(D^2)$; the chunked HVP is $O(D)$. The linear fit memory $= a \cdot D + b$ yields $R^2 = 0.988$ at $n = 256$ and $R^2 = 0.989$ at $n = 1024$, confirming memory linear in $\dim A$ at two scales (Figure 2). The linear fit (slope 0.304 MB/dim, intercept -0.304 MB) passes through all data points, with the small negative intercept reflecting fixed overhead that is amortized at larger D.

G.3 E4: Baseline identity (full explanation)

The three Euclidean-Hessian methods (B5, B9 and B10) produce bit-identical trajectories on both tasks. B5-opt (Sangha (2025a)) evaluates the ambient Weil algebra and reads the $\varepsilon_1\varepsilon_2$ coefficient, which equals the Euclidean Hessian of f composed with the embedding. B9 (projected Hessian) computes $\nabla^2_{\text{ambient}} f$ via nested JVP and projects it to the tangent space. B10 (Taylor-mode jet) computes the ambient Taylor expansion to second order and polarises the mixed entry. For quadratic-in-X objectives, the ambient Hessian is a constant matrix independent of the path through the manifold; projecting it to the tangent space at any point yields the same search direction.

The consequence is that all three share the same systematic error: the projected ambient Hessian differs from the Riemannian Hessian by the Weingarten map (Absil et al., 2013). On the Rayleigh quotient at $n = 512$, this error prevents convergence: the trust-region model built from the wrong Hessian mispredicts descent, and the optimizer stalls at relative gap 0.008 after 40 iterations.

Table 10: E2m: memory estimation parameters.

Parameter	Value
Manifold	Sphere
Sizes n	$\{256, 1024\}$
HVP chunk size k_c	32
Algebra variants	dual ($D = 2$), hyper-dual ($D = 4$), A_2–A_4 ($D = 4$–6)
Repetitions	5 (min-of-reps wall time)
Memory model	Static jaxpr liveness analysis (recursive, float64)

Table 11: E3 Task 1 (Rayleigh quotient): per-seed iteration counts and final relative gap. Newton and TR use the exact Weil Hessian.

Seed	GD (gap)	RCG (gap)	Newton (gap)	Newton (iters)	TR (iters)
0	0.130	0.065	0.000	12	14
1	0.130	0.065	0.000	10	11
2	0.130	0.065	0.000	11	12
3	0.130	0.065	0.000	11	11
4	0.130	0.065	0.000	12	11
Mean ± std	0.130 ± 0.000	0.065 ± 0.000	0.000 ± 0.000	11.2 ± 0.8	11.8 ± 1.3

G.4 E1 to E4 additional experiments

E2: wall time and memory tables

E3: convergence trajectories

E3: generality across seven problems

E4: iteration-count comparison

G.5 Limitations and open questions

1. On the current JAX implementation, B7 is 0.59–0.75× the speed of B3 (nested JVP) and roughly 30× slower than B8 on T1 and 10^3× slower on T2 (Townsend et al. (2016) C backend) at $n = 512$. The iteration-count advantage does not translate to wall-clock advantage at this scale. A custom CUDA kernel or compiled-JAX implementation could close this gap.
2. On the Thomson problem (strongly indefinite Hessian), the trust-region model reports no descent at 7–12 of 40 iterations. The gradient-step safeguard preserves convergence, but the safeguard burden grows with problem stiffness. This is a step-selection weakness of the trust-region algorithm, not a Hessian error (the sphere HVP is the true Riemannian Hessian, verified in E1).
3. The current experiments cover three manifolds (sphere, Stiefel, SPD) and the eight test problems listed in Table 5 (seven in the main generality suite plus T7). Grassmann and fixed-rank manifolds are deferred to future work.
4. The observation that Euclidean Weil, projected-Hessian, and Taylor-jet methods produce identical trajectories is striking but theoretically expected: all three compute the ambient Hessian projected to the tangent space, which differs from the Riemannian Hessian by the second fundamental form. This systematic error is independent of the computational method.

Table 12: E3 Task 2 (low-rank completion): per-seed reconstruction error and iteration counts.

Seed	GD (err)	RCG (err)	Newton (err)	Newton (iters)	TR (iters)
0	0.48	0.23	0.000	17	16
1	0.47	0.23	0.000	14	14
2	0.47	0.23	0.000	15	16
3	0.48	0.24	0.000	16	16
4	0.48	0.23	0.000	16	16
Mean $\pm$ std	0.476 ± 0.005	0.233 ± 0.005	0.000 ± 0.000	15.6 ± 1.1	15.6 ± 1.0

Table 13: E4 Task 1 (Rayleigh quotient): per-seed iteration counts and final relative gap for all five methods. "> 40" = did not converge within budget. B8 uses Townsend et al. (2016).

Seed	B7 (ours)	B8 (Townsend et al. (2016))	B5-opt	B9 / B10
0	13	40	> 40	> 40
1	10	40	> 40	> 40
2	11	40	> 40	> 40
3	11	40	> 40	> 40
4	12	40	> 40	> 40
Mean	11.4 ± 1.0	40	> 40	> 40

Table 14: E4 Task 2 (completion): per-seed iteration counts. All methods converge on this task. B8 uses Townsend et al. (2016).

Seed	B7 (ours)	B8 (Townsend et al. (2016))	B5-opt	B9 / B10
0	17	10	16	16
1	14	10	14	14
2	15	12	15	15
3	16	13	16	16
4	16	14	17	17
Mean	15.6 ± 1.0	11.8 ± 1.6	15.6 ± 1.0	15.6 ± 1.0

Table 15: E3 ablation: Newton with four Hessian models on the Thomson Riesz $s = 2$ energy ($q = 5$, $n = 6$, 5 seeds). Final relative gap and fallback count (model-predicted no-descent iterations).

Model	Iters	Final gap	Fallbacks	Iters (TR)
Exact Weil HVP	41.0 ± 0.0	0.000 ± 0.000	21.4 ± 2.0	41.0 ± 0.0
Diagonal Weil	41.0 ± 0.0	0.001 ± 0.001	0 ± 0	41.0 ± 0.0
Identity	33.2 ± 5.7	0.000 ± 0.000	0 ± 0	37.2 ± 4.8
BFGS (approx)	33.8 ± 8.8	0.000 ± 0.000	7.2 ± 6.9	—

Table 16: E1: derivative accuracy at $n = 8$. "Weil" is our method; B1 = finite differences; B2 = hyper-dual; B3 = nested JVP; B5 = Euclidean Weil (Sangha (2025a)).

Check	Weil	B1 (FD)	B2	B3	B5 (Eucl. Weil)
Gradient (sphere)	$\sim 10^{-16}$	$\sim 10^{-6}$	$\sim 10^{-16}$	$\sim 10^{-16}$	$\sim 10^{-16}$
HVP (sphere)	$\sim 10^{-16}$	$\sim 10^{-6}$	$\sim 10^{-16}$	$\sim 10^{-16}$	~ 0.5
HVP (Stiefel)	$\sim 10^{-16}$	$\sim 10^{-6}$	$\sim 10^{-16}$	$\sim 10^{-16}$	~ 0.5
HVP (SPD)	$\sim 10^{-16}$	$\sim 10^{-8}$	$\sim 10^{-16}$	$\sim 10^{-16}$	~ 0.5
Constraint residual	$\sim 10^{-15}$	—	—	—	—

Table 17: E2: forward-pass wall time (ms) at $n = 32$.

D	2	4	6
Wall time (ms)	16	23	28

Table 18: E2: peak memory for chunked HVP (B7) vs. dense block.

n	Chunked B7 (MB)	Dense (MB)
256	1.57	48.1
1024	6.28	192.5

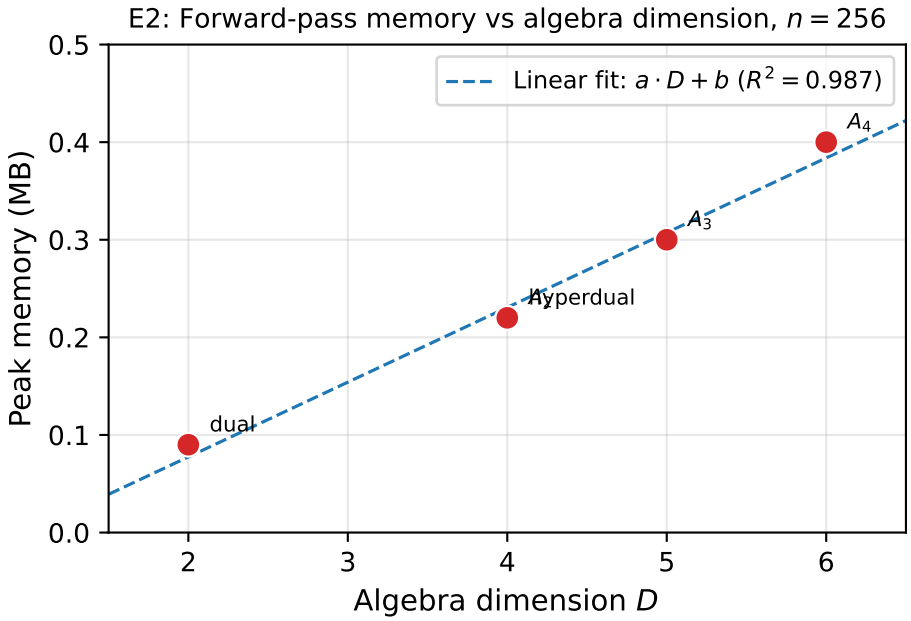


Figure 2: Peak memory vs. algebra dimension D at $n = 256$. Memory is linear in D ($R^2 = 0.988$).

Table 19: E3: generality across all seven test problems ($n = 8$–16, 5 seeds). Newton with exact Weil Hessian.

Problem	Manifold	Rel. gap / Recon. err	Iters (Newton)
T1 Rayleigh	Stiefel	0.000	6.2 ± 0.8
T2 Completion	Stiefel	0.000	9.4 ± 1.1
T3 Procrustes	Stiefel	0.000	6.0–9.4
T4 Karcher mean	SPD	0.000	7.6–20.2
T5 Gaussian MLE	SPD	0.000	< 20
T8 Oblique logistic	Sphere	lowest $f + \lvert\,\text{grad}\,\rvert = 0.002$	< 40
T6 Robust PCA	Stiefel	0.005 (subspace)	~ 15

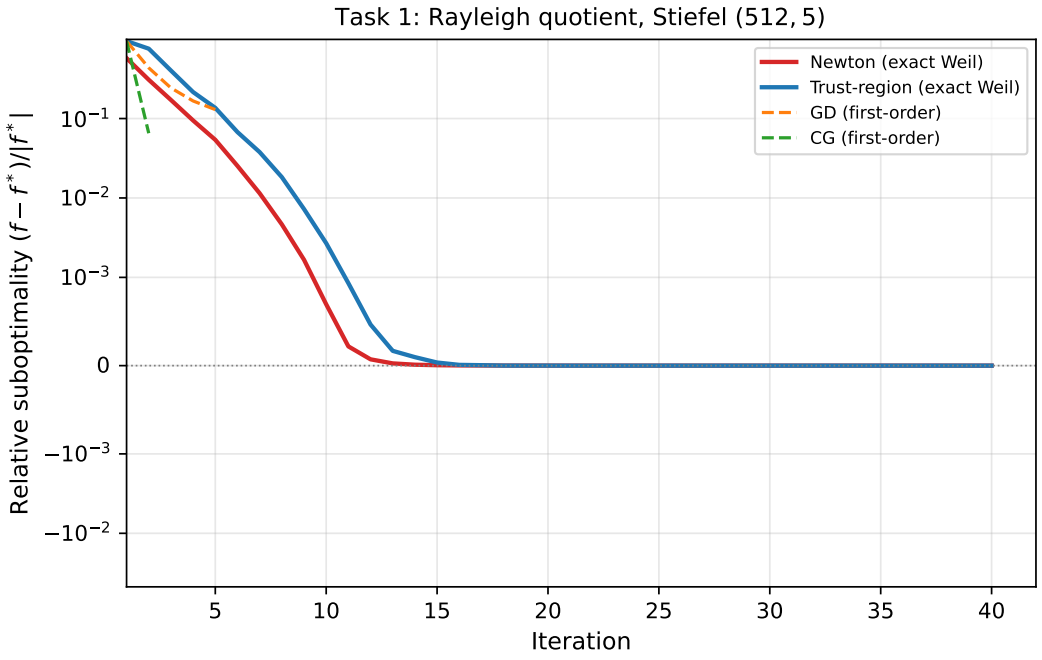


Figure 3: Convergence on Task 1 (Rayleigh quotient, Stiefel $(512, 5)$, 5 seeds, 40 max iterations). Newton (red) and trust-region (blue) with the exact Weil Hessian converge to machine precision within ~ 11 iterations. Gradient descent (orange, dashed) and conjugate gradient (green, dashed) are first-order methods that remain stuck at relative gap $\sim 10^{-1}$. The y-axis shows relative suboptimality $(f - f^*)/|f^*|$ on a symmetric-log scale.

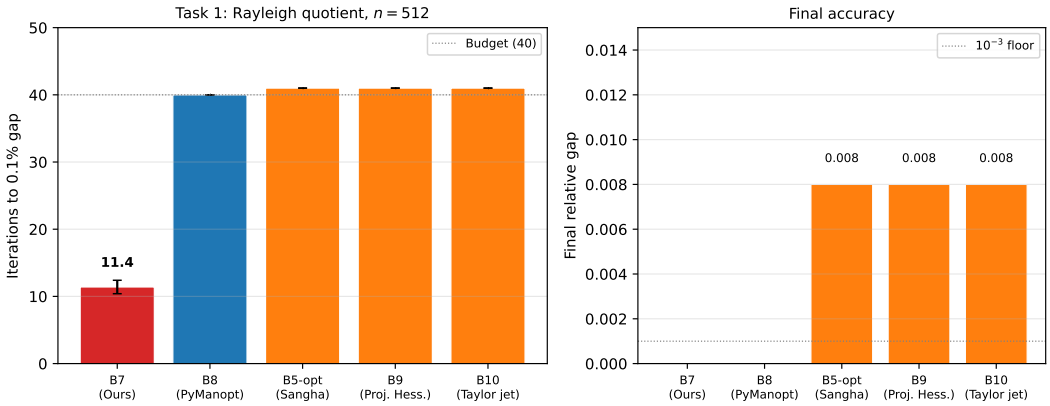


Figure 4: Iteration count comparison on Task 1 ($n = 512$, $p = 5$, 5 seeds). B7 (exact Weil Hessian, ours) converges in 11.4 ± 1.0 iterations. B8 (Townsend et al. (2016) trust-region, C backend) converges in 40 iterations, same accuracy, $3.5\times$ more iterations. B5-opt (Sangha (2025a)), B9 (projected Hessian), and B10 (Taylor jet) are bit-identical to each other and stall at relative gap 0.008 after 40 iterations; they compute the ambient Hessian projected to the tangent space, which differs from the Riemannian Hessian by the Weingarten map.

List of Notation

Sets, spaces, and dimensions

$\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times p}$	Real numbers, n-dimensional Euclidean space, $n\times p$ real matrices
S^{n-1}	Unit sphere $\{x \in \mathbb{R}^n : \|x\|^2 = 1\}$
$\mathrm{St}(n,p)$	Stiefel manifold $\{X \in \mathbb{R}^{n\times p} : X^\top X = I_p\}$
$\mathrm{SPD}(n)$	Symmetric positive-definite cone $\{X \in \mathbb{R}^{n\times n} : X = X^\top,\ X \succ 0\}$
$(S^2)^q$	Product of q copies of the 2-sphere (Thomson problem)
I_p	$p \times p$ identity matrix
$\otimes$	Tensor (Kronecker) product

Differential geometry

(M, g)	Riemannian manifold with metric g
T_xM, TM, N_xM	Tangent space at x, tangent bundle, normal space at x
$g_x(v, w)$	Riemannian inner product of $v, w \in T_xM$
g^{ij}	Components of the inverse metric tensor
$\mathrm{grad}\, f(x)$	Riemannian gradient of f at x
$\mathrm{Hess}\, f(x)[v, w]$	Riemannian Hessian of f at x, bilinear form on T_xM
$\nabla_v \mathrm{grad}\, f$	Covariant derivative of $\mathrm{grad}\, f$ along v (Levi-Civita connection)
$df_x[v]$, $D^2 f_x[v, w]$	First and second Fréchet derivatives in ambient coordinates
$\mathrm{II}(v, w)$	Second fundamental form of M in ambient space

Retractions

$R : TM \to M$	Second-order retraction
$R_x(v)$	Retraction of tangent vector v from point x
$\exp_x(v)$	Riemannian exponential map (a retraction with $\gamma'' = 0$)
$\gamma(s, t) = R_x(sv + tw)$	Retraction curve
$\gamma''_{v,w}$	Retraction acceleration (mixed second derivative of γ)
$\delta_{v,w}$	Retraction error: $\gamma''_{v,w} - \mathrm{II}(v, w)$

Weil algebras

$A = \mathbb{R}[\varepsilon_1, \ldots, \varepsilon_P]/I$	Generic Weil algebra (quotient of polynomial ring by ideal I)
A_{HD}	Hyper-dual algebra $\mathbb{R}[\varepsilon_1, \varepsilon_2]/(\varepsilon_1^2, \varepsilon_2^2)$, dimension $D = 4$
A_r	One-generator algebra $\mathbb{R}[\varepsilon]/(\varepsilon^{r+1})$, dimension $r + 1$
ε_i	Nilpotent generator (infinitesimal direction)
$\varepsilon_i^2 = 0$	Nilpotency relation
$\mathfrak{m}$	Maximal ideal of A: span of all non-real monomials
D	Dimension of algebra A (number of monomial basis elements)
P	Number of generators in A
r	Nilpotency order of A
$M \in \mathbb{R}^{D\times D\times D}$	Multiplication tensor: $M_{a,b,k} = 1$ if $\mathrm{monomial}_a \cdot \mathrm{monomial}_b = \mathrm{monomial}_k$
a_{ij}	Coefficient of $\varepsilon_1^i \varepsilon_2^j$ in $a \in A_{\mathrm{HD}}$
$\mathrm{Re}_{ij}(a)$	Coefficient extraction map: returns a_{ij} from $a \in A_{\mathrm{HD}}$
A-valued	Extended to coefficients in algebra A (e.g. A-valued point on manifold)

Algebra–manifold lifting

$x_A = R_x(U_A) \in M(A)$	A-valued manifold point (constraint preserved coefficient-by-coefficient)
$U_A = v \otimes \varepsilon_1 + w \otimes \varepsilon_2$	Tangent vectors lifted to the algebra
$F_A = f(x_A) \in A$	A-valued function evaluation
f_{weil}	A-valued twin of real objective f
$\text{Re}_{11}(F_A)$	Mixed coefficient = $\text{Hess}\, f(x)[v, w] + g_x(\text{grad}\, f,\, \gamma''_{v,w})$
$\text{Re}_{10}(F_A)$, $\text{Re}_{01}(F_A)$	First-order coefficients = $df_x[v]$, $df_x[w]$

Optimization

$k = \dim T_x M$	Tangent-space dimension
c, k_c	HVP chunk size (memory bound parameter)
Δ_0, Δ	Initial and maximum trust-region radius
ρ	Actual-to-predicted reduction ratio in trust-region
α	Step size (line search / backtracking)
Σ, $\hat{\Sigma}$	Population and sample covariance matrices
$\|\cdot\|_F$	Frobenius norm

Asymptotic notation

$O(\cdot)$	Asymptotic upper bound
$o(\cdot)$	Asymptotically negligible (little-o)
$\Theta(1)$	Constant-order gap (not vanishing with problem size)