Code Guide

Clifford Contractions and the First-Order Bosonic Action in Geometric Unity

R. Fisher [†]

November 17, 2026

1. Files and contents

The verification material accompanies the technical note *Clifford Contractions and the First-Order Bosonic Action in Geometric Unity*. It consists of the complete source in a Markdown computational supplement, an archive of source files and license terms, and this guide.

The text and figures in the note and companion documents are licensed under CC BY 4.0. The software and project configuration files, including source listings embedded in the computational supplement, are licensed under the MIT License. The archive's `README.md` specifies the scope of these grants, with full terms in `LICENSE-CC-BY-4.0.txt` and `LICENSE-MIT.txt`.

Material	Files and contents
Technical note	fisher-geometric-unity-2026-11-17.pdf and fisher-geometric-unity-2026-11-17.md. The mathematical paper in PDF and Markdown.
Computational supplement	fisher-geometric-unity-2026-11-17-computational-supplement.md. All forty Lean files, the Python program, the three project configuration files, the archive README, and both full license texts, with a linked file index and a summary for each listing.
Source archive	fisher-geometric-unity-2026-11-17-code.zip. The same files, ready to extract and run.
Code guide	fisher-geometric-unity-2026-11-17-code-guide.pdf and fisher-geometric-unity-2026-11-17-code-guide.md. This guide's file descriptions, execution instructions, result index, and benchmark tables.

Read the file descriptions to locate a proof, the execution instructions to run the archive, or the benchmark tables to compare numerical results. The computational supplement can also be read directly or supplied as a single text document for source review.

[†]Braid Dynamics Applied Research Group

2. Running the source archive

The Python program and Lean proofs can be run independently after extracting the archive.

Extract the project

Download and extract the source archive. It contains a directory named `gu-code`. All commands below run inside that directory, which contains `GU-test.py`, `lakefile.toml`, and the Lean files.

```
cd gu-code
```

The archive also contains `README.md`, both license texts, and `SHA256SUMS`, a manifest of the SHA-256 digests of every other archived file. Each archived file matches its listing in the computational supplement, using UTF-8 text and Unix line endings.

Run the Python checks

Install Python and NumPy, then run the verification program. The recorded benchmark used Python 3.13.1 and NumPy 2.3.3.

```
python -m pip install numpy==2.3.3
python GU-test.py
```

The program prints six groups of checks, labeled V1 through V6, and ends with `ALL 6 VERIFICATION GROUPS PASSED` when they pass. The benchmark took about three minutes and recorded a peak Python allocation of about 1.3 GiB. Runtime and floating-point residuals depend on the machine and numerical libraries. The exact checks and the measured residuals are identified in the benchmark tables.

Check the Lean proofs

Install Git and Elan, the Lean toolchain manager. The file `lean-toolchain` selects Lean 4.33.1. Mathlib v4.33.1 and its exact dependency revisions are recorded in `lakefile.toml` and `lake-manifest.json`. The project contains 39 Lean library targets and the separate audit module `Audit.lean`, for a total of 40 Lean files.

Run these commands in the extracted `gu-code` directory. The first downloads precompiled Mathlib files, the second checks the project with warnings treated as errors, and the third prints the axiom dependencies of the indexed results.

```
lake exe cache get
lake --wfail build
lake env lean -DwarningAsError=true Audit.lean
```

The first use needs an internet connection to obtain the toolchain and dependencies. A successful build ends with `Build completed successfully`. The audit declarations use `propext`, `Classical.choice`, and `Quot.sound`, the standard Lean axioms recorded in the checked project.

To check the real classification alone, run the named target. Lake builds its dependencies automatically.

```
lake --wfail build D7RealTheorem
```

To check one source file directly, run Lean from the project directory.

```
lake env lean -DwarningAsError=true D7RealTheorem.lean
```

Open the project in Visual Studio Code

Install the Lean 4 extension and open the extracted `gu-code` directory as the workspace folder. Open a Lean file, then use the Lean Infoview to inspect a theorem and its proof. Place the cursor inside a proof to see the local goals.

The terminal build and `Audit.lean` provide the project-wide check. An empty Infoview after the closing namespace reports the cursor's current location.

3. Mathematical coverage

The real classification is assembled in `D7RealTheorem.lean`. Its declaration `theorem_3_4` proves the real dimensions 200, 96, 56, and 48, the 96-dimensional Bianchi subspace, and equivalence with vanishing divergence on every closed coordinate two-form with coefficients in the modeled unitary real form.

The proof begins with complex exterior representations, establishes their intertwiner dimensions, identifies the real Spin action on the coefficient matrices, and transports the dimensions to that action. The real first-jet model then identifies closed inputs with the kernel of alternation. Restriction to the alternating, vector, and traceless-hook channels gives the classification in the note.

Theorem 5.2 uses a finite index set, coefficients in an associative real algebra, and an integrated cyclic trace. Its mixed-term boundary identity is supplied by Stokes' theorem in the geometric setting. Stationarity also requires separation by admissible variations. Witness W1 uses the Clifford algebra of signature $(7, 7)$ and the normalization $\mathrm{Tr}(I) = 128$.

The Noether formalization proves finite gauge invariance and uses dual numbers for the dressing differential. The commuting derivations are ordinary coordinate derivatives. The reference connection carries its own curvature. The paired identity assumes the integrated boundary identities. The off-shell coefficient equality also assumes a dressing Euler representative and separation by admissible gauge tests.

The Python program supplies the numerical comparisons, matrix checks, and integer tables described in Section 6. The Lean declarations provide the formal results listed in Section 5.

4. Source file descriptions

The filenames below are identical in the archive and in the computational supplement. Each filename links to its source listing. The groups follow the mathematical dependencies, with the main results listed first.

Main results and audit

File	Role
`Theorem52.lean`	Derives the first variation, stationarity criterion, and Einstein specialization under the stated boundary and separation hypotheses.
`WitnessW1.lean`	Constructs the exact nonzero torsion correction in the split Clifford algebra and checks the test blade and trace pairing.
`BianchiSymbol.lean`	Identifies closed first jets, realizes them by closed linear two-forms, and proves unique factorization through alternation.
`NoetherIdentity.lean`	Proves finite gauge invariance, the dressing differential, and the paired and off-shell Noether identities under explicit hypotheses.
`D7RealTheorem.lean`	Assembles real Theorem 3.4, including the dimensions 200, 96, 56, and 48 and the closed-field Bianchi criterion.
`Audit.lean`	Prints the axiom dependencies of the indexed mathematical results.

Complex exterior representations

File	Role
`D7Counts.lean`	Computes signed integer multiplicities and verifies the fifteen grade rows and channel totals.
`D7Classification.lean`	Provides general decomposition, intertwiner-dimension, block decomposition, and real-to-complex comparison lemmas.
`D7Exterior.lean`	Defines the split complex orthogonal action and checks exterior weights, highest-weight candidates, and Weyl dimensions.
`D7Hodge.lean`	Constructs the equivariant Hodge involution on the middle exterior power and its signed eigenspaces.

File	Role
`D7Irreducible.lean`	Proves irreducibility of the non-middle exterior powers using signed coordinate actions.
`D7VectorTensor.lean`	Constructs alternation, trace, and their sections and splits the vector times two-form representation into three channels.
`D7Hook.lean`	Proves irreducibility of the traceless hook representation.
`D7VectorGrades.lean`	Constructs wedge and metric-contraction maps and their equivariant sections in all exterior degrees.
`D7MiddleTensor.lean`	Constructs the two signed residual summands in the middle degree.
`D7MiddleIrreducible.lean`	Proves irreducibility of the signed middle residual summands.
`D7TensorColumns.lean`	Organizes repeated-index and distinct-index tensor coordinates used in the kernel arguments.
`D7TensorRotations.lean`	Calculates the plane rotations that connect the two tensor-coordinate families.
`D7GradeIrreducible.lean`	Proves irreducibility of the non-middle vector-exterior kernels.
`D7ComplementaryGrades.lean`	Transports exterior representations and their tensor kernels between complementary degrees by Hodge duality.
`D7SignedDistinction.lean`	Distinguishes the two signed middle constituents using a diagonal torus action.

Multiplicity counts and complex symbols

File	Role
`D7TripleInvariants.lean`	Classifies invariant trilinear exterior forms and bounds the vector and alternating multiplicities.
`D7TripleMultiplicity.lean`	Realizes the trilinear invariants and proves the vector and alternating totals of 56 and 96.
`D7QuadInvariants.lean`	Classifies the invariant forms with a vector factor and three exterior factors.
`D7QuadMultiplicity.lean`	Realizes the invariant four-factor forms needed for the unrestricted contraction count.

File	Role
`D7Unrestricted.lean`	Identifies unrestricted intertwiners with invariant forms and proves the total dimension 200.
`D7Final.lean`	Assembles the complex symbol channels, the Bianchi factorization, the hook count, and the general action-span comparison.
`D7BianchiFields.lean`	Matches the complex tensor symbols to coordinate first jets and proves the closed-field criterion in every grade.

Real Spin and Clifford actions

File	Role
`D7UnitaryRealForm.lean`	Models skew-adjoint coefficient matrices as a real form and identifies their complexification.
`D7SpinAction.lean`	Defines the real Spin vector and Clifford conjugation actions and their exterior-algebra identification.
`D7CliffordMatrix.lean`	Constructs the split Clifford algebra as matrices on the 128-dimensional exterior spinor module.
`D7CliffordAdjoint.lean`	Identifies the matrix adjoint, checks Hermitian signature (64,64), and intertwines coefficient conjugation.
`D7SpinNullFrame.lean`	Changes the real vector action into split null coordinates and maps its complexification into the orthogonal action.
`D7SpinRootLifts.lean`	Constructs real Spin lifts of the isotropic root transvections.
`D7SpinOrientation.lean`	Proves that the real Spin vector action has determinant one.
`D7RootInterpolation.lean`	Uses polynomial interpolation to place complex split-root tensor actions in the span of real Spin actions.
`D7RootGeneration.lean`	Proves generation of the complex split special orthogonal action by split roots and extends the action-span result.
`D7RealComparison.lean`	Identifies the complexified real coefficient, form, and covector tensor actions with the complex exterior-grade actions.

File	Role
`D7RealJets.lean`	Constructs real coordinate first jets, wedge and trace projections, and their equivariant sections.
`D7RealRanks.lean`	Transfers the complex intertwiner counts to the real operator and exterior-symbol spaces.

Program and project configuration

File	Role
`GU-test.py`	Runs the six groups of computational and numerical checks reported in the benchmark tables.
`lakefile.toml`	Declares the 39 Lean library targets and pins Mathlib to v4.33.1.
`lake-manifest.json`	Records the exact Mathlib and transitive dependency revisions for the portable project.
`lean-toolchain`	Pins Lean to leanprover/lean4:v4.33.1.

Publication and license files

File	Role
`README.md`	Records the publication DOI and citation, archive contents, and the scope of the text and software licenses.
`LICENSE-CC-BY-4.0.txt`	Contains the full Creative Commons Attribution 4.0 International terms for the text and figures.
`LICENSE-MIT.txt`	Contains the full MIT License and copyright notice for the software and project configuration files.

File names and declaration names

A Lean filename identifies a source module. A theorem is identified by its namespace and declaration name. The files `D7RealRanks.lean` and `D7RealTheorem.lean` both contribute to the namespace `GeometricUnity.D7RealTheorem`. The result index gives the declaration name together with the file where it is defined.

5. Main Lean results

Each row names the source file and the declaration to inspect. Add `GeometricUnity.`, the namespace, and the declaration name to obtain the full name. The namespace normally matches the filename without its extension. The two results defined in `D7RealRanks.lean` use the namespace `D7RealTheorem`. For example, the real classification has the full name

`GeometricUnity.D7RealTheorem.theorem_3_4`

Result	Source file	Declaration
Real Theorem 3.4	`D7RealTheorem.lean`	`theorem_3_4`
Closed-field equivalence	`D7RealTheorem.lean`	`mem_bianchi_iff_closed_fields`
Alternating factorization	`D7RealTheorem.lean`	`bianchiEquiv`
Traceless-hook projection	`D7RealTheorem.lean`	`mem_hook_iff_project`
Real unrestricted dimension	`D7RealRanks.lean`	`unrestricted_dimension`
Real alternating and vector dimensions	`D7RealRanks.lean`	`exterior_symbol_dimensions`
First variation	`Theorem52.lean`	`theorem_5_2`
Stationarity	`Theorem52.lean`	`stationary_iff`
Einstein specialization	`Theorem52.lean`	`einstein_euler`
Exact torsion correction	`WitnessW1.lean`	`correction_value`
Nonvanishing of the correction	`WitnessW1.lean`	`correction_ne_zero`
Closed linear realization	`BianchiSymbol.lean`	`closed_linear_realization`
Unique alternating factorization	`BianchiSymbol.lean`	`compatible_iff_unique_factor`
Finite gauge invariance	`NoetherIdentity.lean`	`dressedAction_gauge`
Paired Noether identity	`NoetherIdentity.lean`	`dressingDerivative_eq_divergence`
Off-shell Noether identity	`NoetherIdentity.lean`	`theorem_6_1`

6. Benchmark results

The benchmark run used Python 3.13.1 and NumPy 2.3.3 on Windows, with `OPENBLAS_NUM_THREADS=4` and `OMP_NUM_THREADS=4`. All six verification groups passed. The tables report relative residuals unless a row specifies another quantity.

Group	Check	Result
V1	Clifford relations and skew-adjointness under J'	0
V1	Eq. (3.27) checkerboard residual under J and under J'	4.29, 0
V1	Dimension of the compact derived algebra	8190
V1	Killing values on iI and iJ'	0, -32768
V2	Six-operator reductions to three-blade contractions, Einstein and commutator	1.3e-16, 1.3e-16
V2	Bianchi-compatible dimensions from SVD	2, 4
V2	Exact D_7 Bianchi, vector, and hook counts	96, 56, 48

Curvature, action, and gauge checks

Group	Check	Result
V3	Einstein reconstruction	4.79e-16
V3	Contracted differential Bianchi identity, absolute norm	9.41e-14
V3	Gauge covariance, bare and dressed	0.72, 3.05e-15
V4	Dynamical Bianchi identity $d_A F_A = 0$	1.11e-15
V4	Reference divergence, Einstein and commutator	2.43e-15, 2.40e-15
V4	Direct dynamical divergence formula	2.89e-15
V5	Hodge involution, degrees 0 through 14	0
V5	Uniform brackets in (9.3) on constant curvature, minimized over the target sign	5/6, 1/6
V5	Reconstructed Einstein operator on general adjoint input	3.99e-16
V5	Algebraic comparison for $\Delta E_0 = -\frac{8}{3}\gamma^{01234}$	0
V5	Sparse variation with the full Euler derivative	2.70e-10
V6	1D quadrature changes in action and Noether pairing, grids 8, 10, and 14	7.01e-16, 7.43e-15
V6	2D quadrature changes in action and Noether pairing, grids 8 and 10	1.35e-15, 1.49e-15
V6	Volume-normalized kinetic action and radial derivative	-128, -256
V6	Mixed kinetic integration by parts	7.37e-10
V6	Commutator-sector kinetic and cubic bulk variations	2.89e-10, 9.58e-10
V6	Periodic connection variation, $\lambda = 1/3,\ m = \kappa_1/2$	8.42e-10

Group	Check	Result
V6	Periodic connection variation, $\lambda = 2/3,\ m = \kappa_1$	9.41e-10
V6	Fixed-A dressing variation and Noether pairing	3.04e-10
	Wall time	180.02 s
	Peak `tracemalloc` allocation	1313.0 MiB

Directional derivatives

In the sparse blade test, the finite difference and full Euler expression both give 554.666667 to the displayed precision. Omitting ΔE gives 896.000000, a 61.5% discrepancy relative to the finite difference.

The periodic tests use $F_B = 0$, a two-variable field on the split-signature 14-torus, and volume-normalized quadrature. The script compares both coefficient choices with the same curvature-plus-$\kappa_1 T$ expression suggested by equation (9.7). Its discrepancy is 25.7% for $(\lambda, m) = (1/3, \kappa_1/2)$ and 62.7% for $(2/3, \kappa_1)$. The latter comparison also changes the cubic and mass normalizations. In the fixed-A dressing test, omitting ΔE changes the divergence pairing by 377.9% of the finite difference.

Additional checks

The grade-resolved counts in Theorem 3.4 were verified using the suite's integer D_7 routines. For each group of input grades K in the table, and each exterior power $r = 1, 2, 3$, the Weyl dimensions of the output representations, weighted by multiplicity, summed to $\binom{14}{r}\sum_{k\in K}\binom{14}{k}$.

Using the suite's Clifford matrices, the hatted commutator on the closed 01-jet with $M = \gamma^0$ was zero, and the jet with $M = \gamma^1$ gave $2\Gamma_*$. The hatted three-blade commutator on $F_{12} = \gamma^0$ gave $-2i\gamma^{12}\Gamma_*$. In the typed form calculation, $(-, +, +; 1/2)$ agreed with $-*\Theta_{\mathrm{Ein}}$ on constant Riemann curvature with zero matrix residual. On the closed 01-jet with $M = \gamma^0$ its one-form divergence was $-2\gamma^{01}$.

Publication DOI: 10.5281/zenodo.24325062.