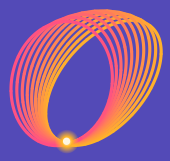

ODEROM

Oriented Differential Exterior calculus for Relativity, Operators and Manifolds

A symbolic engine for differential geometry · technical description

Rafael Camargo Rodrigues de Lima

Departamento de Física · Programa de Pós-Graduação em Física
Centro de Ciências Tecnológicas
Universidade do Estado de Santa Catarina (UDESC) — Joinville, SC, Brasil
rafael.lima@udesc.br · orcid.org/0000-0003-1718-3838

Technical note · version 0.1.0 · 21 August 2026
https://oderom.pages.dev

Abstract

ODEROM is a symbolic computer-algebra engine for differential geometry and general relativity. Given a manifold, a coordinate chart and a metric, it computes curvature in closed form — Christoffel symbols, the Riemann, Ricci, Einstein and Weyl tensors, the geodesic equations, and invariant scalars including Kretschmann — with no numerical approximation at any stage. A second and independent engine manipulates tensor expressions in *abstract* indices, deciding whether a sum vanishes under declared slot symmetries and declared identities such as the Bianchi identities. The program runs as a block notebook (a desktop application and, compiled to WebAssembly, a browser page that computes entirely on the reader's own machine), as a command-line tool, and as an interactive session, from a single binary with no Python, no external computer-algebra system and no LaTeX installation. This note describes what the program does, the representation choices behind it, and how its results are verified. It is not a source listing: the implementation is not reproduced here.

> **Availability.** The notebook runs at oderom.pages.dev with nothing to install. The software is licensed MIT OR Apache-2.0.

1. Why this exists

Curvature computations in general relativity are mechanical and error-prone in equal measure. A four-dimensional metric has forty independent Christoffel symbols and twenty independent Riemann components before any symmetry is applied; the Kretschmann scalar of a two-parameter metric is a rational function whose numerator, written out, occupies a full line. Doing this by hand is a rite of passage that teaches the structure once and then stops teaching anything.

Tools for it exist, and each carries a cost that this project set out to avoid. Some require a proprietary host system. Some require a Python scientific stack, which is a barrier in a teaching laboratory and an obstacle on a machine where nothing may be installed. Some return expressions the user must then simplify by hand, which reintroduces the error the tool was meant to remove. And some accept an input they cannot actually handle, returning a plausible-looking answer that is silently wrong — the failure mode this project treats as the most serious of all.

ODEROM is an attempt at a different set of trade-offs: one self-contained binary, no external system, results in a canonical closed form rather than an unreduced expression, and a strong preference for refusing loudly over answering quietly.

2. Design principles

Three commitments shape the program, and each has cost real features elsewhere.

2.1 A loud refusal beats a quiet wrong answer

Where the engine cannot do something correctly, it says so and names what it cannot do, rather than producing something that resembles an answer. The Weyl tensor is undefined in two dimensions because the standard formula carries a $1/(n-2)$ term; asked for it in two dimensions, the program says exactly that instead of returning a meaningless tensor. A scalar invariant that needs the inverse metric, asked of a bare connection, refuses and explains why.

The sharpest instance concerns function names. A name with no standard mathematical meaning — `f(r)`, `a(t)` — is accepted as an *indeterminate function*: an opaque leaf the engine never evaluates but differentiates correctly by the chain rule. That is how a generic scale factor enters a cosmological metric. But a name that *does* have a standard meaning and is not yet implemented is refused rather than absorbed into that same mechanism. Before this rule existed, `tan(theta)` was accepted as an opaque symbol and differentiated as an ordinary `tan´` — never the real $\sec^2$, yielding a plausible and wrong Christoffel symbol. A reserved-name list closed that hole; the functions have since been implemented, and the list still guards the ones that have not.

2.2 Nothing recomputes by itself

In the notebook, opening a file runs nothing, editing runs nothing, and no interface action triggers a computation. Only an explicit keystroke runs a block, and only that block. When an already-executed block is edited, its result is marked stale rather than recomputed or deleted — the old result stays readable and is labelled as belonging to text that no longer exists, and every executed block below it is marked too, since they may depend on what changed.

This is a pedagogical choice as much as a technical one. A notebook that recomputes on its own teaches the user that results appear; one that recomputes only when asked teaches that results are produced by a step the user took.

2.3 Axioms are declared, never inferred

In the abstract-index engine, multi-term identities are supplied by the user, never assumed. The first Bianchi identity holds for the Riemann tensor of a Levi-Civita connection; it does not hold for an arbitrary tensor with Riemann's slot symmetries, and the engine cannot tell which one the user means. So it is declared. The same applies to metric compatibility, $\nabla_c g_{ab} = 0$, which is a property of the connection rather than of the metric's shape.

An axiom is written as a declaration in the document, not as an option to a command:

```
head R : TM*, TM*, TM*, TM* symmetry (1 2)- (3 4)- (1 3)(2 4)+
axiom bianchi R
```

The choice of a declaration over a flag follows from what an axiom is. "The head R satisfies the first Bianchi identity" is a claim about the geometry, of the same kind as the slot symmetry the `head` line already carries, so it belongs where that claim belongs: scoped to

the document, saved with it, and identical across every interface, since all three read the same document. Command-line flags remain available and combine with declarations rather than replacing them.

The cost is visible: without the declaration, a cyclic Bianchi sum does not collapse, and the program returns three surviving terms for an expression a physicist knows to be zero. That is the correct behaviour for what was actually declared, and it is checked by a test that would fail if the sum ever collapsed unasked.

3. What the program computes

From a declared manifold, chart and metric, twelve queries are available. Each returns a closed-form result, listed by independent component where the object is a tensor.

Query	Object	From a bare connection?
`christoffel`	$\Gamma^{a}{}_{bc}$	yes
`riemann`	R_{abcd} from a metric; $R^{a}{}_{bcd}$ from a connection	yes
`ricci`	R_{ab}	yes
`scalar`	R	no
`kretschmann`	$R_{abcd}R^{abcd}$	no
`einstein`	$G_{ab} = R_{ab} - \tfrac{1}{2}g_{ab}R$	no
`weyl`, `weylsquare`	C_{abcd} and its invariant	no
`riccisquare`, `gaussbonnet`	$R_{ab}R^{ab}$; the Euler density	no
`geodesic`, `accel`	the geodesic equations, and the same solved for each $\ddot{x}^{a}$	yes

The distinction in the third column is not a limitation but a statement of what each object needs. A connection alone determines parallel transport and therefore curvature in mixed form; raising an index to contract requires a metric, and the queries that need one say so rather than inventing it.

An optional index-variance marker requests a variance other than the default — `riemann [up,down,down,down]` — and any query can be wrapped by an export directive that emits the result in SymPy or Mathematica syntax, with reserved-word collisions renamed so the output runs where it is pasted.

3.1 Beyond the fixed menu

The twelve above are a closed catalogue of curvature quantities, and a user's first question is often outside it. A general evaluator answers those: given components the user declared, it computes an arbitrary tensor expression written in abstract indices.

```
tensor p : TM on mink { [t] = E, [x] = p1 }

eval p[a] p[a]      ->   -E^2 + p1^2
```

No metric appears in that expression. It is inserted because both slots of `p` are contravariant, and summing two indices of the same variance requires one — abstract-index notation contracts without regard to variance, components cannot, and the rule that reconciles them decides pair by pair: dual variances are summed directly, two covariant slots draw in g^{ab}, two contravariant slots draw in g_{ab}. Where the chart declares no metric and a pair needs one, the evaluation is refused rather than completed with an assumed one: contracting two indices of the same variance *is* a claim about the geometry, and without a declared metric it has no value.

Indices may be written in the notation of a textbook, with the variance on the index itself — `g_{\mu \nu} V^\mu V^\nu` — or in a bracket form where variance comes from the declaration alone. The two produce the same monomial. Where the written variance disagrees with the declared one, the index is moved with the metric — the reading a physicist intends — by expanding the expression in the parser rather than by giving the evaluator a new rule, so that everything downstream sees an expression already honest about variance. Without a declared metric there is nothing to move it with, and the attempt is refused: moving an index is a claim about the geometry. Index groups may be symmetrised or antisymmetrised in place — `P_{(\mu\nu)}` and `P_{[\mu\nu]}` — with the usual $1/k!$ factor. The one exception to the variance rule is the metric itself, whose raised form names its inverse — a convention rather than a transformation, since raising the indices of g with g would return g.

The result carries whatever indices were not contracted, so it is a component grid whose rank is the number of free indices — rank zero for a scalar. The same expression handed to the abstract-index engine instead returns its canonical form and no numbers; the two engines differ in what they are asked, not in what they are given.

4. Architecture

The program is two engines that share a language and a window and share none of each other's machinery. One knows what R_{trtr} equals for a given metric; the other knows that $R_{abcd} + R_{acdb} + R_{adbc}$ vanishes for every metric. Keeping them separate is deliberate: they answer different questions and a shared representation would serve neither well.

4.1 The component engine

Representation. Scalar expressions are held in a canonical rational form with a *structured* denominator: rather than one opaque polynomial quotient, the denominator is kept factored over the generators the metric actually introduces. Curvature expressions

are dominated by repeated division by the same few factors — r, $(1 - 2M/r)$, Σ and Δ for a rotating black hole — and a form that recognises them keeps intermediate expressions from growing past what the next stage can handle. Rational arithmetic is exact throughout, on arbitrary-precision integers; the release build keeps integer-overflow checks enabled, because silent wraparound in a symbolic engine is a wrong answer rather than a crash.

Transcendental atoms. Five functions are atoms with known derivatives: sine, cosine, the exponential, and the hyperbolic sine and cosine. Two identities are applied during normalisation — $\sin^2 + \cos^2 = 1$ and $\cosh^2 - \sinh^2 = 1$ — and they are what allow a constant-curvature result to actually reduce to a constant rather than to an unsimplified expression that happens to equal one. The eight reciprocal and quotient functions (tangent, cotangent, secant, cosecant and their hyperbolic analogues) are represented as the quotients they are, so their derivatives follow from the existing rules with no new identity: $d(\sin/\cos) = (\cos^2 + \sin^2)/\cos^2$, which the first identity reduces to $\sec^2$.

Metric inversion. The inverse metric is obtained by first detecting the structure of g at three levels: diagonal, block-diagonal, and general. A diagonal metric inverts componentwise; a block-diagonal one inverts each decoupled block on its own, which is what makes the 2×2 $g_{t\varphi}$ block of a rotating black hole tractable; the general case uses cofactors. In every case the result is verified against $g^{ab}g_{bc} = \delta^a{}_c$ before being accepted.

Cancellation. A long computation can be interrupted, and the interruption is checked inside the normalisation loop rather than only between components — a single component of a difficult metric can take longer than a user is willing to wait, and a cancellation that only takes effect between components would not stop it.

4.2 The abstract-index engine

Canonicalisation. A monomial in abstract indices is brought to a canonical form under its head's declared slot-permutation group, so that two expressions that differ only by a symmetry become literally the same object and cancel by construction. The group is handled by a strong generating set, and the canonical form by the Butler–Portugal algorithm, which is the standard approach for this problem.

Why that is not enough. Canonicalisation cannot reach a multi-term identity, and the reason is structural rather than incidental. The first Bianchi identity relates three distinct monomials through a cyclic permutation that is not itself a symmetry of Riemann: the slot-symmetry group has order 8, the cyclic permutation has order 3, and 3 does not divide 8 — by Lagrange's theorem it cannot be a member of that group, not merely happens not to be. No amount of canonicalisation under the declared symmetries will ever collapse that sum.

The decision procedure. Declared identities are therefore handled by linear algebra, per stratum. Expressions are partitioned into strata by an invariant that any identity must preserve: the multiset of tensor heads (with derivative order), the count of free indices by variance, and the number of contractions. Within a stratum, the canonical basis is enumerated — for the Riemann tensor with four free indices it has exactly three elements, which are precisely the three terms of the Bianchi identity — the declared identities are written as rows over that basis, and the rows are reduced. Whether an expression vanishes

then becomes whether its coordinate vector lies in the row space, which is a decision rather than a heuristic; the reduced row echelon form is unique, so the result is a normal form and not an artefact of the order in which rules were applied.

Two measurements from this construction are worth recording. The first Bianchi identity has rank 1 in the three-element basis, reducing it to two independent elements. The second, differential identity, on a fifteen-element basis, has rank 7 — not the 5 that a naive extrapolation from the visible cases suggests, and confirmed by two independent methods before being accepted.

The fast path, and how it is kept honest. A term-rewriting engine built on an e-graph with equality saturation runs by default, because it is fast. It matches; it does not decide. The guarantee comes from being able to run both engines on the same input and fail if they disagree — a mode used across the manipulation catalogue, where it currently reports no divergence. A fast heuristic whose agreement with a decision procedure is never checked is a fast heuristic whose errors nobody will find.

5. The input language

A document is a sequence of declarations and queries. Four declarations describe the geometry; the parser decides what a block is from its first word alone.

```
manifold M dim 4
bundle TM on M dim 4
chart schw on M coords (t, r, theta, phi)
metric g on schw bundle TM {
  [t,t] = -(1 - 2*M/r),
  [r,r] = 1/(1 - 2*M/r),
  [theta,theta] = r^2,
  [phi,phi] = r^2 * sin(theta)^2
}
```

Components not written are zero. Expressions accept two spellings at every point that has a LaTeX equivalent — `r^2 * sin(theta)^2` and `r^2 \sin^2(\theta)` produce an identical tree — so a metric can be transcribed from a paper without translation. A named shorthand may be declared once and used throughout, which is what makes a rotating black hole's Σ and Δ readable, and it is pure substitution: after expansion nothing distinguishes it from the expression written out in full.

Where no metric is available, the connection may be declared directly, which is the entry point for curvature without a metric. The abstract-index engine uses a different declaration, giving a tensor head its slots and its symmetry group, with no components at all.

A further declaration gives components to any tensor, not only to the metric. Its syntax is the signature of an abstract head followed by the chart and component block of a metric — the two halves it is seamed from:

```
tensor F : TM*, TM* symmetry antisymmetric on mink {
  [t,x] = -Ex, [t,y] = -Ey, [x,y] = Bz
}
```

Components may also be computed rather than written. A tensor defined by an expression — `tensor T^{\mu\nu} := A^\mu B^\nu` — takes its name, index order and variances from the left-hand side and everything else from the right: the bundle and dimension of each index come from the slots of the factors used, and so does the chart. The evaluation happens at the declaration, which is what makes the result components like any other. A definition that stored the expression instead, to expand at each use, would also serve the abstract-index engine, but expanding requires renaming indices — the same definition appears as `T[a,b]` in one place and `T[c,d]` in another — and that is a separate piece of machinery.

A declared symmetry governs what is *not* written: the component `[x,t]` above exists and equals $+Ex$ because antisymmetry says so, and the diagonal vanishes for the same reason. Every read of a component resolves through the symmetry orbit rather than a raw table, which is what makes that a fact about the program rather than an intention — a reader that iterated the stored entries would see an antisymmetric tensor as half zeros, and the error would surface months later in one specific tensor, with no symptom before.

6. Interfaces

One engine, three surfaces.

The notebook is a stack of blocks with a typeset result under each. It exists as a desktop application and, compiled to WebAssembly, as a browser page where the computation runs on the reader's own machine — nothing is uploaded, there is no account, and the page works where no software may be installed. The interface is available in Portuguese, English, Italian and French; error messages are not translated, because they name what the user wrote wrong and are generated by the engine rather than the interface.

The command line takes one query per invocation, with guards on wall time and on expression size, and announces each stage of the computation as it begins, so that a timeout names the stage in progress rather than failing silently.

The interactive session keeps a loaded document and recomputes on request, marking entries stale when the file changes underneath them.

7. Verification

This is the part of the project that receives the most effort, and the reason is stated in section 2.1: a symbolic engine that is occasionally wrong is worse than one that is occasionally unavailable, because the user cannot tell.

7.1 Against known closed forms

Every metric in the built-in gallery is an acceptance test. Each was computed and checked against a known closed form before being admitted, and the expected invariant is

displayed beside the entry so that a user can confirm the engine before trusting it on a metric of their own. The following were measured at the version this note describes:

Spacetime	Query	Result
Schwarzschild	Ricci	all 10 independent components identically zero
Schwarzschild	Kretschmann	`48*M^2/r^6`
Reissner–Nordström	Kretschmann	`(-96*M*Q^2*r + 48*M^2*r^2 + 56*Q^4)/r^8`
Kerr	Ricci	all 10 independent components identically zero
de Sitter	Ricci scalar	`12*H^2`
anti-de Sitter	Ricci scalar	`-12*H^2`
FRW, generic a(t)	Ricci scalar	`(6*a(t)*a''(t) + 6*a'(t)^2)/a(t)^2`
hyperbolic plane H^2	Ricci scalar	`-2`

Three of these carry more weight than the rest. The Reissner–Nordström Kretschmann matches the textbook closed form $48M^2/r^6 - 96MQ^2/r^7 + 56Q^4/r^8$ term by term, and setting the charge to zero collapses it to the Schwarzschild value — a check the engine cannot pass by accident. Kerr is non-diagonal, and its vanishing Ricci tensor exercises the block-diagonal inversion path and the structured denominator together. The FRW entry uses an indeterminate function directly inside a metric component and differentiates correctly in terms of a(t), a′(t) and a″(t) without the user ever saying what a(t) is.

7.2 Differential testing

Where a fast heuristic and a decision procedure both exist, they are run against each other on the same inputs and disagreement is a failure. Where the same object can be written two ways, both are computed and required to agree byte for byte: a metric transcribed in ASCII against the same metric in LaTeX; a metric using a named shorthand against the same metric written out; a metric using the tangent against the same metric written as a quotient of sine and cosine. These pairs catch a class of error that no single-path test can, because they do not require knowing the right answer in advance — only that two roads lead to the same place.

7.3 Independent closed forms for new features

A feature is not accepted on the strength of its own output. When the reciprocal trigonometric functions were added, correctness was established against a formula derived independently of the implementation: for a metric of the form $ds^2 = dx^2 + f(x)^2 dy^2$ the Gaussian curvature is $K = -f''/f$, so taking $f = \tan x$ gives $R = 2K = -4\sec^2 x$. The engine returns exactly that. Each of the eight functions was also checked against its textbook derivative, written in terms of the five atomic functions so that the check could not compare the implementation with itself.

7.4 The suite

The test suite runs 731 tests, with 17 marked as known-limitation cases rather than deleted. It includes property-based tests, tests that drive the real desktop window and the real browser page rather than a reimplementation of them, and a permanent test that fails if integer-overflow checking is ever disabled in the release profile. Interface behaviour is verified in the browser that a reader actually uses, a discipline adopted after four consecutive fixes for one defect passed in the desktop harness while the browser version remained broken.

Every test added is checked in both directions: that it fails when the code is deliberately broken, and that it can pass at all. The second half matters more than it sounds — a probe that cannot succeed reports failure indistinguishable from a genuine defect.

8. Performance

Timings below were measured on an Intel Core 5 210H, release build, rustc 1.97.1, with no other load on the machine. They are reported with the machine named because a timing without one is not a measurement.

Metric	Query	Time
Kerr (non-diagonal, two parameters)	Ricci	1.18 s
Kerr	Ricci scalar	1.16 s
Kerr	Kretschmann	15.85 s

Kerr is the interesting case, and it was not always tractable. An earlier revision of the user manual recorded, correctly at the time, that its Christoffel and Riemann tensors did not finish in practical time against the metric's genuinely bivariate denominator, and that Kerr could therefore not join the gallery. Two changes removed that barrier, and neither was in the geometry.

The first was a quadratic accumulator in polynomial multiplication: like terms were being appended and only combined afterwards, so a product with many colliding monomials degraded sharply with degree. Grouping on construction removed it, with a measured improvement of roughly seventy-fold on the case that exposed it. The second was that the notebook session was not using the localised rational engine that had already been implemented and chosen for exactly this shape of problem.

The lesson is recorded here because it generalises: both fixes were found by measurement, and three plausible mechanisms proposed before the measurement — multivariate gcd, a threshold constant, denominator degree — were each refuted by it.

9. Limitations

Stated plainly, because a technical description that lists only capabilities is an advertisement.

- **Algebraic functions are not available.** Square roots, and every inverse function whose derivative contains one, are refused. This is not an omission but a boundary: the

engine is built on a rational normal form, and algebraic functions are not rational. Adding them is a change of foundation, not an extension. The logarithm and the arctangent, whose derivatives *are* rational, are the natural next additions.

- **One spacetime remains outside the gallery.** The Gödel metric computes a Ricci scalar that is algebraically correct but which the normaliser does not reduce to closed form. No entry joins the gallery without a verified curvature test, so it stays out, and the reason is documented rather than hidden.
- **Holonomy is not reachable from the language.** Numerical integration of geodesics and parallel transport exists in the codebase but is wired to no keyword, no notebook block and no command. The symbolic geodesic queries write and rearrange equations; they never integrate.
- **Cancelling in the browser discards computed results.** Stopping WebAssembly mid-computation is only possible by terminating the worker thread, and the worker holds every result with it. The user's text survives; the results do not. Measured, not assumed.

10. Availability

The notebook runs at oderom.pages.dev with nothing to install; the computation executes in the reader's own browser and no data leaves the machine. Desktop packages for Linux (`.deb`, `.rpm`, AppImage) and command-line binaries are built from the same source. The user manual and a shorter browser guide are published alongside the program.

ODEROM is written in Rust, in fourteen crates totalling roughly 29 000 lines of implementation across some 38 000 lines including tests. Its runtime dependencies are limited to arbitrary-precision arithmetic, serialisation, hashing and small-vector utilities; there is no computer algebra system underneath it, no Python, and no LaTeX installation.

The software is licensed under either the Apache License 2.0 or the MIT license, at the user's option.

11. How to cite

Please cite the archived record rather than the web address, so that the version you used remains identifiable:

```
de Lima, R. C. R. (2026). ODEROM: a symbolic engine for
differential geometry (version 0.1.0) [Computer software].
Zenodo. https://doi.org/10.5281/zenodo.22262764
```

A machine-readable `CITATION.cff` accompanies the source. Where a specific version matters, cite the version identifier; otherwise cite the concept identifier, `10.5281/zenodo.22262763`, which always resolves to the most recent release.

References

1. Butler, G. *Fundamental Algorithms for Permutation Groups*. Lecture Notes in Computer Science, Springer, 1991.
2. Sims, C. C. Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra*, Pergamon, 1970.
3. Portugal, R. Algorithmic simplification of tensor expressions. *Journal of Physics A: Mathematical and General*, 1999.
4. Manssur, L. R. U., Portugal, R., and Svaiter, B. F. Group-theoretic approach for symbolic tensor manipulation. *International Journal of Modern Physics C*, 2002.
5. Martín-García, J. M. xPerm: fast index canonicalization for tensor computer algebra. *Computer Physics Communications*, 2008.
6. Peeters, K. Introducing Cadabra: a symbolic computer algebra system for field theory problems. arXiv:hep-th/0701238, 2007.
7. Willsey, M., Nandi, C., Wang, Y. R., Flatt, O., Tatlock, Z., and Panchekha, P. egg: Fast and extensible equality saturation. *Proceedings of the ACM on Programming Languages* (POPL), 2021.
8. Meurer, A. et al. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 2017.
9. Misner, C. W., Thorne, K. S., and Wheeler, J. A. *Gravitation*. W. H. Freeman, 1973.
10. Wald, R. M. *General Relativity*. University of Chicago Press, 1984.
