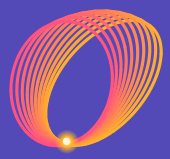

ODEROM

Oriented Differential Exterior calculus for Relativity, Operators and Manifolds

No navegador · um guia curto

Rafael Camargo Rodrigues de Lima

UDESC — Universidade do Estado de Santa Catarina
rafael.lima@udesc.br

oderom.pages.dev · nada a instalar · 3 de setembro de 2026

Sumário

1. O que isto é, e o que você não precisa ter

O ODEROM faz computação simbólica em geometria diferencial. Dê a ele uma variedade, uma carta de coordenadas e uma métrica, e ele calcula curvatura — símbolos de Christoffel, os tensores de Riemann, Ricci, Einstein e Weyl, as equações geodésicas e escalares invariantes como o de Kretschmann — em forma fechada, sem aproximação numérica em etapa alguma.

Você precisa de um navegador. É a lista inteira. Sem download, sem instalador, sem conta, sem direitos de administrador, sem Python, sem instalação de LaTeX. Funciona igual num laboratório da universidade, num notebook emprestado, e numa máquina onde você não tem permissão para instalar nada.

> **A conta roda na sua máquina.** O motor é compilado para WebAssembly e executa dentro da sua própria aba. Nada do que você digita é enviado a um servidor; não há conta, não há envio, e não há o que vazar. A consequência é que a velocidade depende da *sua* máquina, e que fechar a aba perde o que você não salvou.

2. Um minuto: a sua primeira curvatura

Abra a página. Ela chega com um exemplo já carregado — a métrica de Reissner–Nordström, um buraco negro carregado — em quatro blocos, nenhum deles executado:

```
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 + Q^2/r^2),
  [r,r] = 1/(1 - 2*M/r + Q^2/r^2),
  [theta,theta] = r^2,
  [phi,phi] = r^2 * sin(theta)^2
}

kretschmann
```

Clique no primeiro bloco e aperte `Shift`+`Enter` quatro vezes, uma por bloco. O último imprime o escalar de Kretschmann:

```
(-96*M*Q^2*r + 48*M^2*r^2 + 56*Q^4)/r^8
```

Na tela ele chega diagramado, como fração de verdade e com expoentes de verdade; a linha acima é o mesmo resultado escrito em linha.

É a forma fechada do livro-texto, calculada a partir da métrica que está à vista na tela. Faça $Q = 0$ na métrica, rode de novo, e ela colapsa em $48M^2/r^6$ — Schwarzschild.

3. O caderno

A página é uma pilha de blocos. Você edita um bloco e o executa; o resultado aparece embaixo, diagramado.

Tecla	O que faz
`Shift` + `Enter`	Executa este bloco e vai para o seguinte (criando um, se este era o último).
`Ctrl` + `Enter`	Executa este bloco e fica nele. É a tecla de quem está iterando.
`Alt` + `Enter`	Executa este bloco e insere um novo logo abaixo.

O número entre colchetes à esquerda é o contador de execução. `[ ]` quer dizer nunca executado; `[3]` quer dizer que este foi o terceiro bloco que você rodou. Ele segue a ordem em que você executou, não a ordem em que os blocos estão na página.

> **Nada recalcula sozinho.** Abrir um arquivo não executa nada. Editar não executa nada. Clicar não executa nada. Só as três teclas acima executam um bloco, e só o bloco em que você as apertou. Se há um resultado na tela, foi você que pediu.

Quando você edita um bloco que já rodou, ele fica *vencido*: contador âmbar, uma faixa, e um rótulo acima do resultado antigo. O resultado continua legível — ele apenas está marcado como pertencente a um texto que não existe mais. Todo bloco executado *abaixo* dele vence também, porque eles podem depender do que você mudou. Nada é recalculado e nada é apagado; execute o bloco de novo quando quiser o resultado fresco.

No fim da pilha há sempre um bloco vazio, de borda tracejada. Ele está ali para ser digitado.

4. Escolhendo o idioma

O menu à esquerda do cabeçalho troca a interface entre **Português**, **English**, **Italiano** e **Français**. A sua escolha é lembrada na próxima vez que você abrir a página.

Ela muda os botões, os painéis e a barra de estado — e também o documento que o botão **Manual** abre: em português você chega a este guia, e em inglês ao equivalente. As mensagens de erro saem em português, sempre: elas nomeiam o que você escreveu errado, e vêm do motor, não da interface.

5. Começando pela galeria

O botão **Galeria** abre uma lista de espaços-tempos conhecidos. Clicar em um cola as declarações dele como blocos novos e editáveis no fim do seu caderno — o mesmo texto

que você teria digitado, nunca uma métrica escondida atrás de um botão. Nada é executado; os blocos chegam como `[ ]`, prontos para você.

Entrada	O que é	O que você deve obter
Schwarzschild	Vácuo, estático, esfericamente simétrico	$R_{ab} = 0$; Kretschmann = $48M^2/r^6$
Reissner–Nordström	Carregado, esfericamente simétrico	$R = 0$, mas $R_{ab} \neq 0$
Kerr	Buraco negro em rotação, coordenadas de Boyer–Lindquist	$R_{ab} = 0$ nas dez componentes independentes
FRW	Cosmologia espacialmente plana, fator de escala genérico a(t)	$R = 6(a''/a + a'^2/a^2)$
De Sitter	Curvatura positiva constante	$R = 12H^2$; Weyl identicamente nulo
Anti-de Sitter	Curvatura negativa constante, coordenadas de Poincaré	$R = -12H^2$; $R_{ab} = -3H^2 g_{ab}$

Toda entrada carrega parâmetros simbólicos — *M*, *Q*, *a*, *H*. Para fixar um, edite o texto colado, ou escreva `M := 1` num bloco só dele antes da métrica.

Cada uma destas foi conferida contra uma forma fechada conhecida antes de entrar na lista. Elas são um bom jeito de confirmar que o motor está fazendo o que você espera, antes de confiar nele numa métrica sua.

6. Escrevendo a sua métrica — e os seus tensores

Quatro declarações, nesta ordem. Cada uma pode morar no seu próprio bloco ou dividir um; o caderno as lê de cima para baixo.

```
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
}
```

Linha	O que ela diz
`manifold M dim 4`	Uma variedade de dimensão 4 chamada `M`.
`bundle TM on M dim 4`	O fibrado que carrega os índices tensoriais.

`chart schw on M coords (...)`	As coordenadas. O número delas tem de bater com a dimensão da variedade.
`metric g on schw bundle TM { ... }`	As componentes. Só as que você escrever; todo o resto é zero.

Dentro das chaves, uma linha por componente: `[i,j] = expressão`, separadas por vírgula, sem vírgula depois da última. Os índices são nomes de coordenada da sua carta.

O que dá para escrever numa expressão

Você quer	Você escreve
aritmética	`a + b`, `a - b`, `a * b`, `a / b`; justapor também multiplica (`2M`)
potências	`r^2`, `r^-1`, `r^{-2}` — o expoente é sempre um inteiro
trigonometria	`sin(theta)`, `cos(theta)`, `tan`, `cot`, `sec`, `csc`
hiperbólicas	`sinh(chi)`, `cosh(chi)`, `tanh`, `coth`, `sech`, `csch`
exponencial	`exp(H*t)`
letras gregas	escreva o nome: `theta`, `phi`, `chi`. LaTeX (`\theta`, `\frac{a}{b}`) também é aceito.
uma abreviação	`f := 1 - 2*M/r` num bloco só dela, e depois `f` em qualquer lugar abaixo
uma função desconhecida	`f(r)`, `a(t)` — o motor a deriva como `f'(r)` sem você definir nada

Treze funções são conhecidas. Cinco são átomos do motor (`sin`, `cos`, `exp`, `sinh`, `cosh`); as outras oito são as recíprocas e os quocientes delas — `tan` é sin/cos, `sec` é 1/cos, e assim por diante para as quatro hiperbólicas. Uma consequência que vale esperar: um resultado que contenha uma tangente volta escrito como `sin(x)/cos(x)`, e não como `tan(x)`. É o mesmo valor, e a derivada dele é o sec² do livro — conferida contra a forma fechada.

Qualquer outra coisa é tratada de propósito, e não em silêncio:

- Um nome sem significado padrão (`f`, `a`, `h`) vira uma *função desconhecida*: opaca, mas derivada corretamente. É assim que a entrada de FRW escreve `a(t)`.
- Um nome que *é* uma função padrão mas ainda não está implementado (`sqrt`, `log`, `arcsin`, `atan`) é **recusado**, e não aceito como desconhecido. Isto importa: antes de a regra existir, `tan(theta)` era aceito e derivado como um `tan'` opaco, nunca o sec² — um resultado que parecia plausível e estava errado. Uma recusa em alto e bom som vale mais que uma resposta errada em silêncio. (O `tan` está implementado agora; a regra que o pegou continua guardando o resto.)

Para uma geometria sem métrica nenhuma — uma conexão nua — dá para declarar os símbolos de Christoffel diretamente:

```
connection Gamma on flat2 {
  [x,y,y] = x
}
```

`christoffel`, `riemann` e `ricci` funcionam a partir disso. `scalar` e os outros invariantes não, e dizem o porquê: eles precisam de uma métrica para subir um índice.

Dando componentes a algo que não é a métrica

A métrica não é a única coisa que tem componentes. Um quadrimomento, um campo eletromagnético, uma velocidade — qualquer coisa que você consiga escrever como uma lista de números na sua carta:

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

Leia assim: `p` tem um índice, esse índice está *em cima* (é o que `TM` sem asterisco quer dizer), as componentes dele vivem na carta `mink`, e aqui estão elas. O que você não escreve é zero, então `[y]` e `[z]` foram omitidos acima.

Em baixo é `TM*`, com o asterisco. A distinção não é enfeite — o capítulo 9 mostra o que ela lhe dá.

A simetria se declara do mesmo jeito, e daí em diante ela governa o que você não escreveu:

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

Você deu `[t,x]`; `[x,t]` vale $+Ex$ porque a antissimetria diz que sim, e a diagonal inteira é zero pelo mesmo motivo. Ninguém digitou nenhuma das duas.

Definindo um tensor a partir de outros

Você não precisa escrever cada componente à mão. Um tensor pode ser definido por uma expressão:

```
tensor T^{\mu\nu} := A^\mu B^\nu
```

O lado esquerdo dá o nome e diz quais são os índices; o lado direito diz quanto ele vale. Nada mais é necessário — a carta e os fibrados vêm dos tensores que você usou. Depois disso `T` se comporta como qualquer outro tensor declarado, então `eval g_{\mu\nu} T^{\mu\nu}` funciona, e dá o que a expressão escrita por extenso daria.

Uma definição pode se apoiar em outra acima dela:

```
tensor S^{\mu\nu} := T^{\mu\nu} + T^{\nu\mu}
```

Curvatura com nome

A terceira forma toma uma consulta de curvatura como lado direito:

```
tensor G  := einstein g
tensor Rc := ricci g
```

Sem isto, `ricci` e `einstein` eram perguntas que *imprimiam*: o resultado não tinha nome, e portanto não podia ser contraído, comparado nem posto num lado de uma equação. Agora pode:

```
eval G[a,b] G[a,b]        ->  0    (Schwarzschild é vácuo)
simplify Rc[a,b] - Rc[b,a] ->  0
```

A segunda linha merece um momento. O `simplify` nunca vê componente nenhuma — ele trabalha em índices abstratos. Ela colapsa porque a declaração *afirma* a simetria no head, e a afirmação é conferida: toda célula é escrita, inclusive as que a simetria já determinaria, e uma discordância seria reportada em vez de absorvida.

Serve para `ricci`, `einstein`, `riemann` e `weyl`. `christoffel` não, porque não é tensor; e os cinco escalares também não, porque `tensor` declara tensor — para um escalar com nome há a abreviação `NOME := ...`.

7. O que dá para perguntar

Um bloco cuja primeira palavra é uma destas é uma pergunta. Escreva-a sozinha, na própria linha, e execute.

Escreva	Você recebe
`christoffel`	Os símbolos de Christoffel $\Gamma^{a}{}_{bc}$.
`riemann`	O tensor de Riemann, totalmente covariante.
`ricci`	O tensor de Ricci R_{ab}.
`scalar`	O escalar de Ricci R — uma expressão.
`kretschmann`	$R_{abcd}R^{abcd}$ — o invariante que não se anula no vácuo.
`einstein`	O tensor de Einstein G_{ab}.
`weyl`, `weylsquare`	O tensor conforme e o invariante dele. Indefinido em 2 dimensões, e ele diz isso.
`riccisquare`, `gaussbonnet`	$R_{ab}R^{ab}$, e a densidade de Gauss–Bonnet.
`geodesic tau`	As equações geodésicas, uma por coordenada. A palavra no fim é o nome do seu parâmetro afim.

`accel tau`	As mesmas equações resolvidas para cada segunda derivada — a forma que um integrador numérico come.

Dois refinamentos que você vai querer em algum momento:

- **Variância dos índices.** `riemann [up,down,down,down]` sobe o primeiro índice. Um `up` ou `down` por índice, e a contagem tem de bater exatamente com o posto.
- **Qual métrica.** Se você declarou mais de uma, nomeie: `kretschmann g2`. Com uma só, você nunca precisa.

O parâmetro afim de `geodesic`/`accel` não pode colidir com uma coordenada nem com um símbolo que já esteja na sua métrica. `geodesic r` contra uma carta que tem *r* é recusado, porque `t(r)` significaria duas coisas ao mesmo tempo.

> **Resultados longos são truncados na tela.** O caderno mostra as primeiras 20 componentes independentes e depois uma linha dizendo quantas faltam — `... e mais 10 componentes independentes (truncado)` — além de quantas são identicamente nulas. A contagem é sempre honesta; só a exibição é encurtada.

8. Tirando os resultados de dentro

Clique em qualquer linha de um resultado e o LaTeX dela vai para a sua área de transferência — só aquela equação, limpa, pronta para colar num documento. Não a anotação ao lado, não a linha de resumo, não o bloco inteiro. Um selo "copiado" confirma.

O botão Exportar escreve uma linha de código para outro programa. Escolha um formato e uma consulta, e ele insere algo como

```
export sympy kretschmann
```

num bloco novo. Ele não executa — você executa, com `Shift`+`Enter`, como qualquer outra coisa. O painel mostra a linha exata antes de você clicar, e é esse o ponto: na segunda vez, você a escreve sozinho. Tanto Mathematica quanto SymPy estão disponíveis, e a exportação funciona em qualquer uma das consultas do capítulo 7.

O SymPy exportado vem com a linha `symbols(...)` dele e com as colisões de palavra reservada renomeadas, para rodar onde você colar.

9. A outra metade: índices abstratos — e a volta aos números

Tudo até aqui calcula *componentes*: você dá uma métrica, você recebe expressões em *r* e *θ*. O ODEROM tem uma segunda metade, e ela está disponível aqui também.

O `simplify` trabalha sobre expressões tensoriais em que os índices são apenas rótulos — sem métrica, sem coordenadas, sem componentes. Declare as simetrias de um tensor com `head`, e depois pergunte se uma expressão colapsa:

```
manifold M dim 4
bundle TM on M dim 4
head R : TM*, TM*, TM*, TM* symmetry (1 2)- (3 4)- (1 3)(2 4)+
```

e então, noutro bloco:

```
simplify R[a,b,c,d] + R[b,a,c,d]
```

```
0
```

É a antissimetria no primeiro par fazendo o seu trabalho. Mais alguns:

Você escreve	Você recebe	Por quê
`simplify R[a,b,c,d] - R[c,d,a,b]`	`0`	A simetria de troca de pares.
`simplify 3 R[a,b,c,d] + -1 R[a,b,c,d]`	`2 R[a,b,c,d]`	Termos semelhantes se juntam.
`simplify R[a,b,c,d] + R[a,c,b,d]`	os dois termos sobrevivem	Nada declarado os torna iguais — e um simplificador que devolvesse zero aqui seria pior que inútil.

Notação: tensores se escrevem lado a lado, não multiplicados com `*` — `g[a,b] R[b,c,d,e]`. Um índice repetido é uma contração. Um ponto e vírgula introduz uma derivada covariante: `T[a,b;c]`.

Declarando as identidades

Identidades de vários termos — as identidades de Bianchi, a compatibilidade métrica — são *declaradas*, nunca supostas. A primeira identidade de Bianchi vale para o tensor de Riemann de uma conexão de Levi-Civita e não para um tensor qualquer com as mesmas simetrias de slot, e o programa não tem como saber qual dos dois você quer dizer. Então você diz:

```
axiom bianchi R
```

num bloco só dele, e aí a soma cíclica colapsa:

```
simplify R[a,b,c,d] + R[a,c,d,b] + R[a,d,b,c]

0
```

Existem quatro espécies: `bianchi` (a identidade algébrica), `bianchi2` (a diferencial), `metric` (contrair este head sobe ou desce um índice) e `compatible` (a conexão é compatível com a métrica, $\nabla g = 0$). Um axioma é uma declaração como qualquer outra — o escopo dele é o caderno inteiro, e ele é salvo no arquivo junto com o resto.

> **Sem a declaração nada colapsa — e está correto.** Um simplificador que devolvesse zero para a soma cíclica sem ter sido avisado estaria adivinhando de que tensor você falava.

De volta aos números: `eval`

O `simplify` e o `eval` tomam a mesma expressão e fazem perguntas diferentes:

```
simplify p[a] p[a]   ->  p[a] p[a]      a forma canônica, sem números
eval     p[a] p[a]   ->  -E^2 + p1^2    porque as componentes existem
```

Aqui está a coisa inteira, cinco linhas e uma pergunta. Ponha as declarações num bloco:

```
manifold M dim 4
bundle TM on M dim 4
chart mink on M coords (t, x, y, z)
metric g on mink bundle TM { [t,t] = -1, [x,x] = 1, [y,y] = 1, [z,z] = 1 }
tensor p : TM on mink { [t] = E, [x] = p1 }
```

e a pergunta noutro:

```
eval p[a] p[a]
```

```
-E² + p1²
```

É a massa invariante, com o sinal onde ele deve estar.

> **Você não escreveu a métrica na expressão, e é esse o ponto.** Você declarou `p` como `TM` — em cima, como a física pede. Os dois slots de `p[a] p[a]` estão portanto em cima, então o ODEROM sabe que precisa de $g_{\mu\nu}$ para contraí-los e a insere sozinho. O sinal de menos vem de $g_{tt} = -1$, e não de outro lugar.
>
> Se você tivesse declarado `p` com um `*`, ele inseriria a métrica *inversa* — outra conta, e a certa para aquela declaração.

Se a carta não tem métrica declarada, o ODEROM recusa em vez de escolher uma. Contrair dois índices da mesma variância é uma afirmação sobre a geometria, e sem métrica ela não tem valor. Ele também recusa, nomeando o que falta, quando um tensor da sua expressão não tem componentes — devolver os símbolos seria responder a outra pergunta, e essa é a do `simplify`.

O resultado guarda os índices que não foram contraídos. Sobrando um índice livre, você recebe uma lista:

```
eval T[a,b] V[b]
```

```
[^t] = a*v0 + b*v1
[^x] = c*v0 + d*v1
```

Escrevendo do jeito do livro

Dá para pôr a variância no próprio índice, como num livro-texto:

```
eval g_{\mu \nu} V^\mu V^\nu
```

Mesma resposta que `eval V[a] V[a]` — três jeitos de escrever uma conta:

Você escreve	O que diz
`g_{\mu \nu} V^\mu V^\nu`	a notação do livro; toda contração casa um índice de cima com um de baixo
`g[a,b] V[a] V[b]`	a mesma coisa, colchetes no lugar das macros
`V[a] V[a]`	a abreviação: a métrica fica implícita

Um índice sozinho não precisa de chaves (`V^\mu`), e rótulos gregos e latinos são intercambiáveis — `\mu` é só um nome.

Escrever um índice do outro lado o move com a métrica, que é o que você quereria dizer no papel:

```
eval u_\mu     u declarado com índice em cima -> descido
eval A^\mu     A declarado com índice em baixo -> subido
```

Então `u_\mu A^\mu` e `g_{\mu\nu} u^\nu A^\mu` são a mesma conta, escrita de dois jeitos. Sem métrica declarada não há com o que mover um índice, e o ODEROM diz isso em vez de escolher uma.

Parênteses num grupo de índices o simetrizam; colchetes antissimetrizam:

```
eval P_{(\mu\nu)}     a parte simétrica
eval P_{[\mu\nu]}     a parte antissimétrica
```

Os dois carregam o fator 1/*k*! de sempre, então antissimetrizar um tensor simétrico dá zero, e simetrizar um antissimétrico também — um jeito rápido de conferir que um tensor é o que você pensa que ele é.

A métrica com índices em cima é a inversa dela, que é como se escreve a norma de um covetor:

```
eval g^{\mu\nu} A_\mu A_\nu
```

É o mesmo que `A_\mu A_\mu`, a abreviação — mas diz o que faz. `g^{ab}` não é *g* com os índices movidos (isso devolveria *g*); é a inversa, por definição. Só a métrica ganha isso: qualquer outro tensor escrito na variância oposta continua sendo erro.

Uma armadilha que vale conhecer: `g_{ab}` lê como um índice só, chamado `ab`. Rótulos latinos dentro de chaves precisam de separador — `g_{a b}` — enquanto `g_{\mu\nu}` está bem, porque uma macro se autodelimita.

É mesmo uma equação tensorial? `invariant`

Todo livro-texto diz que uma equação tensorial não depende das coordenadas que você escolheu. Nenhum sistema de álgebra computacional deixa você *perguntar* — Cadabra, xAct e SymPy deixam isso como coisa que você confere rodando a conta duas vezes e comparando com o olho. Aqui é uma linha.

Ela precisa de duas coisas. Primeiro, o mesmo objeto escrito em duas cartas — um `tensor` ou uma `metric` podem ser declarados mais de uma vez sob um nome, desde que cada declaração nomeie uma carta diferente:

```
tensor p : TM on cart  { [x] = x, [y] = y }
tensor p : TM on shear { [u] = u, [v] = v }
```

Segundo, a mudança de coordenadas entre elas:

```
transition cart to shear {
  u = 2*x,
  v = x + y
}
```

Cada linha dá uma coordenada do destino como função das da origem. Uma direção por declaração — derivar a inversa exigiria resolver um sistema simbólico. E então:

```
invariant g_{\mu\nu} p^\mu p^\nu
```

```
cart: x^2 + y^2
shear: -u*v + 1/2*u^2 + v^2
cart -> shear: concordam
invariante
```

As duas expressões não se parecem em nada, e é esse o ponto: elas são o mesmo número em dois sistemas de coordenadas, e o ODEROM conferiu em vez de acreditar na sua palavra.

Índices livres funcionam também. Cada um é levado pelo seu próprio fator — a Jacobiana para um índice em baixo, a *inversa* da Jacobiana para um em cima, que é o que as duas palavras significam:

```
invariant p^\mu        invariant p_\mu

cart:                  cart:
[^x] = x               [x] = x
[^y] = y               [y] = y
shear:                 shear:
```

```
[^u] = u                [u] = 1/2*u - 1/2*v
[^v] = v                [v] = -1/2*u + v
```

Os dois invariantes, e as componentes genuinamente diferentes entre si.

> **Quando o `eval` começa a recusar.** Uma vez que um tensor vive em duas cartas, o `eval` sobre uma expressão feita inteiramente de tensores assim não tem uma carta só para trabalhar, e diz isso em vez de escolher — avaliar na carta errada devolveria números plausíveis e errados. Um fator declarado numa carta só resolve; ou pergunte com `invariant`, que provavelmente era a pergunta que você queria.

A pergunta invertida: `equation` e `solve`

Todas as perguntas até aqui têm a forma "dado isto, calcule aquilo". Esta é a primeira que diz "isto vale aquilo, e agora me diga o que falta". A física pede a pergunta invertida o tempo todo, e a mais comum de todas é: dada uma geometria, *que matéria a produz?*

Uma equação recebe nome:

```
equation eq1 : x + y = 2
solve eq1 for y

y = 2 - x
```

E o caso que interessa. Declare a geometria, dê nome ao tensor de Einstein, declare a incógnita, e escreva as equações de campo:

```
tensor G := einstein g
head T : TM*, TM* symmetry (1 2)+
equation efe : G_{\mu\nu} = 8 T_{\mu\nu}
solve efe for T_{\mu\nu}
```

Sobre Schwarzschild:

```
T:
16 independent components identically zero
```

Schwarzschild é vácuo, e o programa acabou de dizer isso resolvendo, em vez de ter sido informado. Sobre Reissner–Nordström as mesmas quatro linhas devolvem o tensor de energia-momento eletromagnético:

```
T:
[t,t] = (-1/4*Mass*Q^2*r + 1/8*Q^2*r^2 + 1/8*Q^4)/r^6
[r,r] = 1/8*Q^2/(2*Mass*r^3 - Q^2*r^2 - r^4)
[theta,theta] = 1/8*Q^2/r^2
[phi,phi] = 1/8*Q^2*sin(theta)^2/r^2
12 independent components identically zero
```

> **Coeficiente se escreve justapondo, não com `*`.** Na gramática de monômios é `8 T_{\mu\nu}`; `8*T_{\mu\nu}` é erro de sintaxe. E um coeficiente *simbólico* tem de ser um head de posto zero (`head L : on M`), porque um nome não declarado vira "unknown tensor head". Isso não vale para uma equação escalar como `x + y = 2`, que usa a gramática escalar de sempre.

Quando a incógnita tem menos graus de liberdade que a equação, o que sobra vira conferência. Pergunte se a variedade é um espaço de Einstein — se o Ricci é proporcional à métrica:

```
tensor Rc := ricci g
head L : on M
equation esp : Rc_{\mu\nu} = L g_{\mu\nu}
solve esp for L
```

Métrica	Resposta
de Sitter	`L = 3*H^2`
Schwarzschild	`L = 0`
Reissner–Nordström	recusa: pede `L = -Q^2/r^4` *e* `L = Q^2/r^4`

Dezesseis equações, uma incógnita. Ou toda célula concorda — e o programa acabou de *provar* que a variedade é um espaço de Einstein, com o Λ calculado — ou não concordam, e o ansatz não fecha. As duas respostas são úteis, e a segunda é a que nenhuma outra ferramenta dá de graça. Os dois valores em conflito ali são exatamente a estrutura de sinais do Ricci eletromagnético.

Sem componentes, o que sobra é mover termos — o que se faz no papel:

```
equation e : T[a,b] + U[a,b] = S[a,b]
solve e for T[a,b]

T[a,b] = -1 U[a,b] + S[a,b]
```

Qual dos dois caminhos roda é decidido pelo *documento*: se algum head da equação tem componentes, a resposta são números; se nenhum tem, a resposta são símbolos rearranjados. É a mesma expressão com quantidades diferentes de informação — a relação que `simplify` e `eval` já têm.

O manual completo lista as onze recusas do `solve`, cada uma com o seu motivo. A que mais aparece: escrever o alvo na variância oposta à declarada deixa a incógnita atrás de fatores de métrica, e a mensagem diz em que variância ele foi declarado — que é o conserto.

10. Salvar e reabrir

Salvar baixa um arquivo. Digite um nome no campo de caminho antes — é esse nome que o download recebe, com `.od` acrescentado se você não puser. Uma página de navegador não pode escrever numa pasta que você escolha, então baixar é o que "salvar" significa aqui.

Abrir usa o seletor de arquivos do seu sistema. O campo de caminho é ignorado ao abrir, de propósito: uma página não pode ler um arquivo por caminho, só um que você entregou a ela, e fingir o contrário seria pior que a assimetria. Cancelar o seletor não muda nada e não é erro.

O arquivo é texto puro, com `%%` numa linha só dele entre os blocos. É o mesmo formato que o aplicativo de mesa lê e escreve, então um caderno passa de um para o outro intocado. Reabrir sempre volta com todos os blocos *não* executados — coerente com nada recalcular sozinho.

> **Fechar a aba perde tudo que não foi salvo.** Não há autossalvamento nem cópia no servidor — é a mesma propriedade que faz nada do que você digita sair da sua máquina. Salve antes de fechar.

11. Quando uma conta demora demais

Enquanto um bloco roda, a página continua funcionando: os outros blocos aceitam foco e edição, e o bloco em execução oferece **Cancelar**. Só um bloco roda por vez; apertar uma tecla de execução noutro é recusado, com um lampejo na barra de estado e no bloco que está segurando o motor.

Não há limite de tempo no navegador. Uma conta roda até terminar ou até você pará-la. O que a limita é um teto de tamanho de expressão (50 000 nós), que interrompe uma conta descontrolada antes que ela esgote a sua memória, e reporta em vez de travar para sempre.

> **CANCELAR CUSTA OS SEUS RESULTADOS**
>
> **Cancelar no navegador descarta todos os resultados que você já calculou** — não só o do bloco que você cancelou. O seu *texto* sobrevive intacto; os resultados não, e todos os blocos voltam a `[ ]`.
>
> Isto não é escolha de projeto, é a plataforma: parar WebAssembly no meio de uma conta só é possível terminando a thread de trabalho em que ela roda, e essa thread leva todos os resultados junto. Medido, não suposto: um teste dirige um navegador de verdade, conta quatro blocos com resultado antes do cancelamento e zero depois.
>
> Na prática: deixe a consulta cara por último, e se você espera cancelar, salve antes.

12. O que muda em relação à versão de mesa

A linguagem, o motor e os resultados são idênticos — o mesmo Rust, compilado de dois jeitos. Quatro diferenças, todas impostas pelo navegador:

Aqui	Na versão de mesa
Salvar baixa um arquivo	Salvar escreve no caminho que você digitou
Abrir usa o seletor de arquivos	Abrir lê o caminho que você digitou
Cancelar descarta todos os resultados	Cancelar para um bloco; os outros resultados ficam
Sem limite de tempo; um teto de tamanho no lugar	O mesmo teto de tamanho

Todo o resto — as teclas, os estados de bloco, o vencimento, a galeria, a Exportação, o clique para copiar, o seletor de idiomas — se comporta igual nos dois.

13. Para ir adiante

O manual completo (em inglês) cobre o que este guia deixa de fora: a gramática inteira, as ferramentas de linha de comando, a referência de mensagens de erro, os axiomas declarados para o trabalho em índices abstratos, a lista completa de palavras reservadas, e dez exemplos trabalhados. Leia este aqui para começar; leia aquele quando quiser saber exatamente o que o programa faz e o que ele não faz.

Todos os documentos — este guia, o manual e a nota técnica — estão listados aqui, para ler no navegador ou baixar em PDF. É também para onde o botão **Manual** do cabeçalho leva.

Dois hábitos que vale formar cedo, e em torno dos quais este programa foi construído:

- **Confira contra algo que você já sabe.** Rode uma métrica da galeria cujo invariante você consiga verificar antes de confiar numa métrica sua. É por isso que a galeria lista a resposta esperada ao lado de cada entrada.
- **Prefira um erro em alto e bom som a uma resposta silenciosa.** Quando o ODEROM recusa — um nome de função reservado, um parâmetro que colide, Weyl em duas dimensões — ele está recusando porque a alternativa era um resultado que parecia certo e não era.

Como citar

Se o ODEROM for útil num trabalho seu, cite o registro arquivado e não o endereço do site, para que a versão que você usou continue identificável:

```
de Lima, R. C. R. (2026). ODEROM: a symbolic engine for
differential geometry [Computer software].
Zenodo. https://doi.org/10.5281/zenodo.22262763
```

Esse identificador aponta sempre para a versão mais recente. Quando a versão importa, cite a dela — a 0.1.0 é `10.5281/zenodo.22262764`.

Um `CITATION.cff` legível por máquina acompanha o código-fonte, e a descrição técnica do programa — o que ele computa, como representa expressões, e como os resultados são verificados — é publicada ao lado deste manual como `ODEROM-paper.pdf`.

O ODEROM é distribuído sob a Licença Apache 2.0 ou a licença MIT, à sua escolha.

ODEROM · guia do navegador. Tradução do guia em inglês, que teve todo comportamento aqui descrito verificado contra o programa em execução em 20 de agosto de 2026 — inclusive o custo de cancelar, medido num navegador de verdade em vez de suposto. As seções do `invariant` e do `solve` vieram depois, com as saídas coladas do terminal em 2 e 3 de setembro de 2026.