

Elementary maths

This example is based on a similar example in the [Python collection](#). Its purpose is to show that Cadabra is fluent in Python – which is not surprisng since the Cadabra language is based on Python (and a subset of LaTeX).

```
from sympy import *
x, y, z = symbols('x y z')
a, b, c = symbols('a b c')
ans = expand((a+b)**3) # cdb (ans.101,ans)
ans = factor(-2*x+2*x+a*x-x**2+a*x**2-x**3) # cdb (ans.102,ans)
ans = solve(x**2-4, x) # cdb (ans.103,ans)
ans = solve([2*a-b - 3, a+b+c - 1,-b+c - 6],[a,b,c]) # cdb (ans.104,ans)
ans = N(pi,50) # cdb (ans.105,ans)
ans = apart(1/((1 + x)*(5 + x))) # cdb (ans.106,ans)
ans = together((1/(1 + x) - 1/(5 + x))/4) # cdb (ans.107,ans)
ans = simplify(tanh(log(x))) # cdb (rhs.108,ans)
ans = simplify(tanh(I*x)) # cdb (rhs.109,ans)
ans = simplify(sinh(3*x) - 3*sinh(x) - 4*(sinh(x))**3) # cdb (rhs.110,ans)
ans = tanh(log(x)) # cdb (lhs.108,ans)
ans = tanh(UnevaluatedExpr(I*x)) # cdb (lhs.109,ans)
ans = sinh(3*x) - 3*sinh(x) - 4*(sinh(x))**3 # cdb (lhs.110,ans)
```

```
\begin{align*}
&\&\backslash\mathrm{cdb}*{\mathrm{ans}.101}\\
&\&\backslash\mathrm{cdb}*{\mathrm{ans}.102}\\
&\&\backslash\mathrm{cdb}*{\mathrm{ans}.103}\\
&\&\backslash\mathrm{cdb}*{\mathrm{ans}.104}\\
&\&\backslash\mathrm{cdb}*{\mathrm{ans}.105}\\
&\&\backslash\mathrm{cdb}*{\mathrm{ans}.106}\\
&\&\backslash\mathrm{cdb}*{\mathrm{ans}.107}\\
&\backslash\mathrm{cdb}\{\mathrm{lhs}.108\} \&= \backslash\mathrm{Cdb}\{\mathrm{rhs}.108\}\\
&\backslash\mathrm{cdb}\{\mathrm{lhs}.109\} \&= \backslash\mathrm{Cdb}\{\mathrm{rhs}.109\}\\
&\backslash\mathrm{cdb}\{\mathrm{lhs}.110\} \&= \backslash\mathrm{Cdb}\{\mathrm{rhs}.110\}
\end{align*}
```

$$\mathrm{ans}.101 := a^3 + 3a^2b + 3ab^2 + b^3$$

$$\mathrm{ans}.102 := -x(-a+x)(x+1)$$

$$\mathrm{ans}.103 := [-2, 2]$$

$$\mathrm{ans}.104 := \left\{ a : \frac{1}{5}, b : -\frac{13}{5}, c : \frac{17}{5} \right\}$$

$$\mathrm{ans}.105 := 3.1415926535897932384626433832795028841971693993751$$

$$\mathrm{ans}.106 := -\frac{1}{4(x+5)} + \frac{1}{4(x+1)}$$

$$\mathrm{ans}.107 := \frac{1}{(x+1)(x+5)}$$

$$\tanh(\log(x)) = \tanh(\log(x)) \quad (\mathrm{rhs}.108)$$

$$\tanh(ix) = i \tan(x) \quad (\mathrm{rhs}.109)$$

$$-4 \sinh^3(x) - 3 \sinh(x) + \sinh(3x) = 0 \quad (\mathrm{rhs}.110)$$

Linear Algebra

```

from sympy import linsolve
lamda = Symbol('lamda')
mat = Matrix([[2,3], [5,4]])
eig1 = mat.eigenvecs()[0][0]
eig2 = mat.eigenvecs()[1][0]
v1 = mat.eigenvecs()[0][2][0]
v2 = mat.eigenvecs()[1][2][0]
eig = simplify(Matrix([eig1,eig2]))
vec = simplify(5*Matrix([]).col_insert(0,v1)
               .col_insert(1,v2))
det = expand((mat - lamda * eye(2)).det())
rhs = Matrix([[3], [7]])
ans = list(linsolve((mat,rhs),x,y))[0]

```

```

\begin{align*}
&\&\text{\texttt{ans.201}}\\
&\&\text{\texttt{ans.202}}\\
&\&\text{\texttt{ans.203}}\\
&\&\text{\texttt{ans.204}}\\
&\&\text{\texttt{ans.205}}\\
&\&\text{\texttt{ans.206}}
\end{align*}

```

$$\begin{aligned}
\text{ans.201} &:= \begin{bmatrix} 2 & 3 \\ 5 & 4 \end{bmatrix} \\
\text{ans.202} &:= \begin{bmatrix} -1 \\ 7 \end{bmatrix} \\
\text{ans.203} &:= \begin{bmatrix} -5 & 3 \\ 5 & 5 \end{bmatrix} \\
\text{ans.204} &:= \lambda^2 - 6\lambda - 7 \\
\text{ans.205} &:= \begin{bmatrix} 3 \\ 7 \end{bmatrix} \\
\text{ans.206} &:= \left(\frac{9}{7}, \frac{1}{7} \right)
\end{aligned}$$

Limits

```
n, dx = symbols('n dx')
ans = limit(sin(4*x)/x,x,0)           # cdb (ans.301,ans)
ans = limit(2**x/x,x,oo)              # cdb (ans.302,ans)
ans = limit(((x+dx)**2 - x**2)/dx, dx,0) # cdb (ans.303,ans)
ans = limit((4*n + 1)/(3*n - 1),n,oo)  # cdb (ans.304,ans)
ans = limit((1+(a/n))**n,n,oo)        # cdb (ans.305,ans)
```

```
\begin{align*}
&\&\backslash\text{cdb}\{ans.301\}\\
&\&\backslash\text{cdb}\{ans.302\}\\
&\&\backslash\text{cdb}\{ans.303\}\\
&\&\backslash\text{cdb}\{ans.304\}\\
&\&\backslash\text{cdb}\{ans.305\}
\end{align*}
```

```
ans.301 := 4
ans.302 := ∞
ans.303 := 2x
ans.304 :=  $\frac{4}{3}$ 
ans.305 :=  $e^a$ 
```

Series

```
ans = series((1 + x)**(-2), x, 1, 6)   # cdb (ans.401,ans)
ans = series(exp(x), x, 0, 6)          # cdb (ans.402,ans)
ans = Sum(1/n**2, (n,1,50)).doit()     # cdb (ans.403,ans)
ans = Sum(1/n**4, (n,1,oo)).doit()     # cdb (ans.404,ans)
```

```
\begin{align*}
&\&\backslash\text{cdb}\{ans.401\}\\
&\&\backslash\text{cdb}\{ans.402\}\\
&\&\backslash\text{cdb}\{ans.403\}\\
&\&\backslash\text{cdb}\{ans.404\}
\end{align*}
```

```
ans.401 :=  $\frac{1}{2} + \frac{3(x-1)^2}{16} - \frac{(x-1)^3}{8} + \frac{5(x-1)^4}{64} - \frac{3(x-1)^5}{64} - \frac{x}{4} + O((x-1)^6; x \rightarrow 1)$ 
ans.402 :=  $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + O(x^6)$ 
ans.403 :=  $\frac{3121579929551692678469635660835626209661709}{1920815367859463099600511526151929560192000}$ 
ans.404 :=  $\frac{\pi^4}{90}$ 
```

Calculus

This example shows how `\Cdb` can be used to set the equation tag on the far right hand side.

```
ans = diff(x*sin(x),x) # cdb (ans.501,ans)
ans = diff(x*sin(x),x).subs(x,pi/4) # cdb (ans.502,ans)
ans = integrate(2*sin(x)**2, (x,a,b)) # cdb (ans.503,ans)
ans = Integral(2*exp(-x**2), (x,0,oo)) # cdb (lhs.504,ans)
ans = ans.doit() # cdb (ans.504,ans)
ans = Integral(Integral(x**2 + y**2, (y,0,x)), (x,0,1)) # cdb (lhs.505,ans)
ans = ans.doit() # cdb (ans.505,ans)
```

```
\begin{align*}
&\&\backslash\cdb*{ans.501}\\
&\&\backslash\cdb*{ans.502}\\
&\&\backslash\cdb*{ans.503}\\
&\backslash\cdb{lhs.504}\&=\backslash\Cdb{ans.504}\\
&\backslash\cdb{lhs.505}\&=\backslash\Cdb{ans.505}
\end{align*}
```

$$\text{ans.501} := x \cos(x) + \sin(x)$$

$$\text{ans.502} := \frac{\sqrt{2}\pi}{8} + \frac{\sqrt{2}}{2}$$

$$\text{ans.503} := -a + b + \sin(a) \cos(a) - \sin(b) \cos(b)$$

$$\int_0^{\infty} 2e^{-x^2} dx = \sqrt{\pi} \quad (\text{ans.504})$$

$$\int_0^1 \int_0^x (x^2 + y^2) dy dx = \frac{1}{3} \quad (\text{ans.505})$$

Differential equations

```

y = Function('y')
C1, C2 = symbols('C1 C2')

ode = Eq(y(x).diff(x) + y(x), 2*a*sin(x))
sol = expand(dsolve(ode,y(x)).rhs) # cdb (ans.601,sol)
cst = solve([sol.subs(x,0)],dict=True)
sol = sol.subs(cst[0]) # cdb (ans.602,sol)

ode = Eq(y(x).diff(x,2) + y(x), 0)
sol = expand(dsolve(ode,y(x)).rhs) # cdb (ans.603,sol)
cst = solve([sol.subs(x,0),sol.diff(x).subs(x,0)-1],dict=True)
sol = sol.subs(cst[0]) # cdb (ans.604,sol)

ode = Eq(y(x).diff(x,2) + 5*y(x).diff(x) - 6*y(x), 0)
sol = expand(dsolve(ode,y(x)).rhs) # cdb (ans.605,sol)
sol = sol.subs({C1:2,C2:3}) # cdb (ans.606,sol)

```

[illegible]

$$\text{ans.601} := C_1 e^{-x} + a \sin(x) - a \cos(x)$$

$$\text{ans.602} := a \sin(x) - a \cos(x) + ae^{-x}$$

$$\text{ans.603} := C_1 \sin(x) + C_2 \cos(x)$$

$$\text{ans.604} := \sin(x)$$

$$\text{ans.605} := C_1 e^{-6x} + C_2 e^x$$

$$\text{ans.606} := 3e^x + 2e^{-6x}$$

The metric connection

This is a very standard computation that shows if

$$\Gamma_{bc}^a = \frac{1}{2} g^{ad} (\partial_b g_{dc} + \partial_c g_{bd} - \partial_d g_{bc}) \quad (1)$$

then

$$g_{ab;c} = 0. \quad (2)$$

This example might well be regarded as the Cadabra counterpart to the familiar *Hello World* program of undergraduate programming classes.

```
{a,b,c,d,e,f,h,i,j,k,l,m,n,o,p,q,r,s,t,u#}::Indices.

g_{a b}::Metric.
g_{a}^{b}::KroneckerDelta.

\partial_{#}::PartialDerivative.

cderiv:=\partial_{c}{g_{a b}} - g_{a d}\Gamma^{d}_{b c}
        - g_{d b}\Gamma^{d}_{a c}.          # cdb (term31,cderiv)

Gamma:=\Gamma^{a}_{b c} -> (1/2) g^{a d} ( \partial_{b}{g_{d c}}
        + \partial_{c}{g_{b d}}
        - \partial_{d}{g_{b c}} ) . # cdb (term32,Gamma)

substitute      (cderiv,Gamma);      # cdb (term33,cderiv)
distribute      (cderiv)              # cdb (term34,cderiv)
eliminate_metric (cderiv)              # cdb (term35,cderiv)
eliminate_kronecker (cderiv)          # cdb (term36,cderiv)
canonicalise    (cderiv)              # cdb (term37,cderiv)
```

The metric connection

$$\begin{aligned}
 \text{term31} &:= \partial_c g_{ab} - g_{ad} \Gamma_{bc}^d - g_{db} \Gamma_{ac}^d \\
 \text{term32} &:= \Gamma_{bc}^a \rightarrow \frac{1}{2} g^{ad} (\partial_b g_{dc} + \partial_c g_{bd} - \partial_d g_{bc}) \\
 \text{term33} &:= \partial_c g_{ab} - \frac{1}{2} g_{ad} g^{de} (\partial_b g_{ec} + \partial_c g_{be} - \partial_e g_{bc}) - \frac{1}{2} g_{db} g^{de} (\partial_a g_{ec} + \partial_c g_{ae} - \partial_e g_{ac}) \\
 \text{term34} &:= \partial_c g_{ab} - \frac{1}{2} g_{ad} g^{de} \partial_b g_{ec} - \frac{1}{2} g_{ad} g^{de} \partial_c g_{be} + \frac{1}{2} g_{ad} g^{de} \partial_e g_{bc} - \frac{1}{2} g_{db} g^{de} \partial_a g_{ec} - \frac{1}{2} g_{db} g^{de} \partial_c g_{ae} + \frac{1}{2} g_{db} g^{de} \partial_e g_{ac} \\
 \text{term35} &:= \partial_c g_{ab} - \frac{1}{2} g_a^e \partial_b g_{ec} - \frac{1}{2} g_a^e \partial_c g_{be} + \frac{1}{2} g_a^e \partial_e g_{bc} - \frac{1}{2} g_b^e \partial_a g_{ec} - \frac{1}{2} g_b^e \partial_c g_{ae} + \frac{1}{2} g_b^e \partial_e g_{ac} \\
 \text{term36} &:= \frac{1}{2} \partial_c g_{ab} - \frac{1}{2} \partial_c g_{ba} \\
 \text{term37} &:= 0
 \end{aligned}$$

```

\begin{align*}
&\&\text{cdb*{term31}}\\
&\&\text{cdb*{term32}}\\
&\&\text{cdb*{term33}}\\
&\&\text{cdb*{term34}}\\
&\&\text{cdb*{term35}}\\
&\&\text{cdb*{term36}}\\
&\&\text{cdb*{term37}}
\end{align*}

```

Curvature of a 2-sphere

This examples uses standard methods to compute the scalar curvature of a 2-sphere.

```
{\theta, \varphi}::Coordinate.
{\alpha, \beta, \gamma, \delta, \rho, \sigma, \mu, \nu, \lambda}::Indices(values={\varphi, \theta}, position=independent).

\partial{#}::PartialDerivative.

g_{\alpha\beta}::Metric.
g^{\alpha\beta}::InverseMetric.

Chr := \Gamma^{\alpha}_{\mu\nu} -> 1/2 g^{\alpha\beta} ( \partial_{\nu}\{g_{\beta\mu}\}
                                         + \partial_{\mu}\{g_{\beta\nu}\}
                                         - \partial_{\beta}\{g_{\mu\nu}\} ).

Rabcd := R^{\rho}_{\sigma\mu\nu} -> \partial_{\mu}\{\Gamma^{\rho}_{\sigma\nu}\}
- \partial_{\nu}\{\Gamma^{\rho}_{\sigma\mu}\}
+ \Gamma^{\rho}_{\beta\mu} \Gamma^{\beta}_{\sigma\nu} - \Gamma^{\rho}_{\beta\nu} \Gamma^{\beta}_{\sigma\mu}.

Rab := R_{\sigma\nu} -> R^{\rho}_{\sigma\rho\nu}.

R := R -> R_{\sigma\nu} g^{\sigma\nu}.

gab:={ g_{\theta\theta} = r**2,
       g_{\varphi\varphi} = r**2 \sin(\theta)**2 }. # cdb(gab,gab)

complete (gab, $g^{\alpha\beta}$) # cdb(iab,gab)

evaluate (Chr, gab, rhsonly=True) # cdb(Chr,Chr)

substitute (Rabcd, Chr)
evaluate (Rabcd, gab, rhsonly=True) # cdb(Rabcd,Rabcd)

substitute (Rab, Rabcd)
evaluate (Rab, gab, rhsonly=True) # cdb(Rab,Rab)
```


substitute (R, Rab)

evaluate (R, gab, rhsonly=True)

cdb(R,R)

$$\left[g_{\theta\theta} = r^2, \quad g_{\varphi\varphi} = r^2(\sin\theta)^2, \quad g^{\varphi\varphi} = (r^2(\sin\theta)^2)^{-1}, \quad g^{\theta\theta} = r^{-2} \right]$$

$$\Gamma^\alpha_{\mu\nu} \rightarrow \square_{\mu\nu}{}^\alpha \begin{cases} \square_{\varphi\theta}{}^\varphi = (\tan\theta)^{-1} \\ \square_{\theta\varphi}{}^\varphi = (\tan\theta)^{-1} \\ \square_{\varphi\varphi}{}^\theta = -\frac{1}{2}\sin(2\theta) \end{cases}$$

$$R^\rho_{\sigma\mu\nu} \rightarrow \square_{\sigma\nu}{}^\rho{}_\mu \begin{cases} \square_{\varphi\varphi}{}^\theta{}_\theta = \frac{1}{2}\sin(2\theta)(\tan\theta)^{-1} - \cos(2\theta) \\ \square_{\theta\varphi}{}^\varphi{}_\theta = -1 \\ \square_{\varphi\theta}{}^\theta{}_\varphi = -\frac{1}{2}\sin(2\theta)(\tan\theta)^{-1} + \cos(2\theta) \\ \square_{\theta\theta}{}^\varphi{}_\varphi = 1 \end{cases}$$

$$R_{\sigma\nu} \rightarrow \square_{\sigma\nu} \begin{cases} \square_{\varphi\varphi} = \frac{1}{2}\sin(2\theta)(\tan\theta)^{-1} - \cos(2\theta) \\ \square_{\theta\theta} = 1 \end{cases}$$

$$R \rightarrow 2r^{-2}$$

```
\begin{align*}
&\&\text{cdb}\{\text{iab}\}\&\&[10\text{pt}]
&\&\text{cdb}\{\text{Chr}\}\&\&[10\text{pt}]
&\&\text{cdb}\{\text{Rabcd}\}\&\&[10\text{pt}]
&\&\text{cdb}\{\text{Rab}\}\&\&[10\text{pt}]
&\&\text{cdb}\{\text{R}\}
\end{align*}
```

The Riemann curvature tensor

```
{a,b,c,d,e,f,i,j,k,l,m,n,o,p,q,r,s,t,u#}::Indices(position=independent).

\partial_{\#}::PartialDerivative.

\Gamma^{a}_{b c}::Depends(\partial{\#}).
\Gamma^{a}_{b c}::TableauSymmetry(shape={2}, indices={1,2});

;::Symbol; # Suggsted by Kasper as a way to (possibly) make use of ; legal
            # see https://cadabra.science/qa/473/is-this-legal-syntax?show=478
            # this code works with and without this trick

# generic rule for first two covariant derivs of a downstairs-vector

deriv1 := A?_{a ; b} -> \partial_{b}{A?_{a}} - \Gamma^{c}_{a b} A?_{c}.
deriv2 := A?_{a ; b ; c} -> \partial_{c}{A?_{a ; b}}
      - \Gamma^{d}_{a c} A?_{d ; b}
      - \Gamma^{d}_{b c} A?_{a ; d}.

substitute (deriv2,deriv1)      # cdb (ex01, deriv2)

Mabc := M_{a ; b ; c}.          # cdb (ex02, Mabc)

substitute (Mabc,deriv2)      # cdb (ex03, Mabc)

distribute (Mabc)             # cdb (ex04, Mabc)
product_rule (Mabc)           # cdb (ex05, Mabc)

Macb := M_{a ; c ; b}.         # cdb (ex06, Macb)

substitute (Macb,deriv2)      # cdb (ex07, Macb)

distribute (Macb)             # cdb (ex08, Macb)
product_rule (Macb)           # cdb (ex09, Macb)

diff := @(Mabc) - @(Macb).     # cdb (ex10, diff)
```

```

sort_product      (diff)          # cdb (ex11, diff)
rename_dummies    (diff)          # cdb (ex12, diff)
canonicalise      (diff)          # cdb (ex13, diff)
sort_sum          (diff)          # cdb (ex14, diff)
factor_out        (diff,$M_{a?}$) # cdb (ex15, diff)

```

$$\begin{aligned}
A^?_{a;b;c} &\rightarrow \partial_c (\partial_b (A^?_a) - \Gamma^d_{ab} A^?_d) - \Gamma^d_{ac} (\partial_b (A^?_d) - \Gamma^e_{db} A^?_e) - \Gamma^d_{bc} (\partial_d (A^?_a) - \Gamma^e_{ad} A^?_e) \\
M_{a;bc} &= M_{a;b;c} = \partial_c (\partial_b M_a - \Gamma^d_{ab} M_d) - \Gamma^d_{ac} (\partial_b M_d - \Gamma^e_{db} M_e) - \Gamma^d_{bc} (\partial_d M_a - \Gamma^e_{ad} M_e) \\
M_{a;cb} &= M_{a;c;b} = \partial_b (\partial_c M_a - \Gamma^d_{ac} M_d) - \Gamma^d_{ab} (\partial_c M_d - \Gamma^e_{dc} M_e) - \Gamma^d_{cb} (\partial_d M_a - \Gamma^e_{ad} M_e) \\
M_{a;bc} - M_{a;cb} &= M_d (-\partial_c \Gamma^d_{ab} + \partial_b \Gamma^d_{ac} - \Gamma^e_{ab} \Gamma^d_{ce} + \Gamma^e_{ac} \Gamma^d_{be})
\end{aligned}$$

```

\begin{gather*}
\backslash\text{cdb}\{\text{ex01}\}\backslash\backslash
M_{\{a;bc\}} = \backslash\text{cdb}\{\text{ex02}\} = \backslash\text{cdb}\{\text{ex03}\}\backslash\backslash
M_{\{a;cb\}} = \backslash\text{cdb}\{\text{ex06}\} = \backslash\text{cdb}\{\text{ex07}\}\backslash\backslash[10pt]
M_{\{a;bc\}} - M_{\{a;cb\}} = \backslash\text{cdb}\{\text{ex15}\}
\end{gather*}

```

Symmetry of the Ricci tensor

This simple example shows that, for the metric connection, the Ricci tensor is symmetric, that is $R_{ab} = R_{ba}$.

```
{a,b,c,d,e,f,i,j,k,l,m,n,o,p,q,r,s,t,u#}::Indices(position=independent).

\partial{#}::PartialDerivative;

g_{a b}::Symmetric;
g^{a b}::Symmetric;

g_{a b}::Depends(\partial{#});
g^{a b}::Depends(\partial{#});

dgab := \partial_{c}{g^{a b}} -> - g^{a e} g^{b f} \partial_{c}{g_{e f}}. # cdb (dgab,dgab)

Gamma := \Gamma^{a}_{b c} ->
(1/2) g^{a e} ( \partial_{b}{g_{e c}}
+ \partial_{c}{g_{b e}}
- \partial_{e}{g_{b c}}). # cdb (Chr,Gamma)

Rabcd := R^{a}_{b c d} ->
\partial_{c}{\Gamma^{a}_{b d}} + \Gamma^{a}_{e c} \Gamma^{e}_{b d}
- \partial_{d}{\Gamma^{a}_{b c}} - \Gamma^{a}_{e d} \Gamma^{e}_{b c}.
# cdb (Rabcd,Rabcd)

Rab := R_{a b} -> R^{c}_{c a b}. # cdb (Rab,Rab)

eqn := 2 (R_{a b} - R_{b a}).

substitute (eqn, Rab)
substitute (eqn, Rabcd)
substitute (eqn, Gamma)

distribute (eqn)
product_rule (eqn)
canonicalise (eqn) # cdb (final1,eqn)
```

```

substitute (eqn,dgab)
canonicalise (eqn)

```

```

# cdb (final2,eqn)

```

Symmetry of the Ricci tensor

$$g^{ab}{}_{,c} := \partial_c g^{ab} \rightarrow -g^{ae} g^{bf} \partial_c g_{ef}$$

$$\Gamma^a_{bc} := \Gamma^a_{bc} \rightarrow \frac{1}{2} g^{ae} (\partial_b g_{ec} + \partial_c g_{be} - \partial_e g_{bc})$$

$$R^a{}_{bcd} := R^a{}_{bcd} \rightarrow \partial_c \Gamma^a{}_{bd} + \Gamma^a{}_{ec} \Gamma^e{}_{bd} - \partial_d \Gamma^a{}_{bc} - \Gamma^a{}_{ed} \Gamma^e{}_{bc}$$

$$R_{ab} := R_{ab} \rightarrow R^c{}_{acb}$$

$$\begin{aligned}
2(R_{ab} - R_{ba}) &= -\partial_b g^{ce} \partial_a g_{ce} + \partial_a g^{ce} \partial_b g_{ce} \\
&= 0
\end{aligned}$$

```

\begin{align*}
g^{\{a\}_{b}\}_{c}} &:= \backslash cdb{dgab} \\
\backslash Gamma^{\{a\}_{bc}} &:= \backslash cdb{Chr} \\
R^{\{a\}_{bcd}} &:= \backslash cdb{Rabcd} \\
R_{\{ab\}} &:= \backslash cdb{Rab} \\
2(R_{\{ab\}} - R_{\{ba\}}) &= \backslash cdb{final1} \\
&= \backslash cdb{final2}
\end{align*}

```

The Gauss relation for the curvature of a hypersurface

```
{a,b,c,d,e,f,g,i,j,k,l,m,n,o,p,q,r,s,t,u#}::Indices.

\nabla_{#}::Derivative.

K_{a b}::Symmetric.
g^{a}_{b}::KroneckerDelta.

# Define the projection operator

hab:=h^{a}_{b} -> g^{a}_{b} - n^{a} n_{b}.

# 3-covariant derivative obtained by projection on 4-covariant derivative

vpq:=v_{p q} -> h^{a}_{p} h^{b}_{q} \nabla_{b}{v_{a}}.

# Compute 3-curvature by commutation of covariant derivatives

vpqr:= h^{a}_{p} h^{b}_{q} h^{c}_{r} ( \nabla_{c}{v_{a b}} - \nabla_{b}{v_{a c}} ).

substitute (vpq,hab)
substitute (vpqr,vpq)

distribute (vpqr)
product_rule (vpqr)
distribute (vpqr)
eliminate_kronecker(vpqr)

# Standard substitutions

substitute (vpqr,$h^{a}_{b} n^{b} -> 0$)
substitute (vpqr,$h^{a}_{b} n_{a} -> 0$)
substitute (vpqr,$\nabla_{a}{g^{b}_{c}} -> 0$)
substitute (vpqr,$n^{a} \nabla_{b}{v_{a}} -> -v_{a} \nabla_{b}{n^{a}}$)
substitute (vpqr,$v_{a} \nabla_{b}{n^{a}} -> v_{p} h^{p}_{a} \nabla_{b}{n^{a}}$)
substitute (vpqr,$h^{p}_{a} h^{q}_{b} \nabla_{p}{n_{q}} -> K_{a b}$)
substitute (vpqr,$h^{p}_{a} h^{q}_{b} \nabla_{p}{n^{b}} -> K_{a}^{q}$)
```

```

# Tidy up and display the results

{h^{a}_{b}, \nabla_{a}{v_{b}}>::SortOrder.

sort_product      (vpqr)
rename_dummies    (vpqr)
canonicalise      (vpqr)
factor_out        (vpqr, $h^{a?}_{b?}$)
factor_out        (vpqr, $v_{a?}$)      # cdb(gauss, vpqr)

```

The Gauss relation for the curvature of a hypersurface

$$D_r(D_q v_p) - D_q(D_r v_p) = h^a_p h^b_q h^c_r (\nabla_c (\nabla_b v_a) - \nabla_b (\nabla_c v_a)) + v_a (K_{pr} K_q^a - K_{pq} K_r^a)$$

```

\begin{align*}
D_{\{r\}}(D_{\{q\}}v_{\{p\}}) - D_{\{q\}}(D_{\{r\}}v_{\{p\}}) &= \backslash\text{cdb}\{\text{gauss}\} \\
\end{align*}

```

The metric connection in Riemann normal coordinates

In local Riemann normal coordinates, the metric components can always be expanded as a power series in the Riemann curvatures and its derivatives (provided the curvatures are finite at the expansion point). In particular

$$g_{ab}(x) = g_{ab} - \frac{1}{3}x^c x^d R_{acbd} - \frac{1}{6}x^c x^d x^e \nabla_c R_{adbe} + \dots$$
$$g^{ab}(x) = g^{ab} + \frac{1}{3}x^c x^d g^{ae} g^{bf} R_{cedf} + \frac{1}{6}x^c x^d x^e g^{af} g^{bg} \nabla_c R_{dfeg} + \dots$$

where g_{ab} and g^{ab} are independent of the coordinates x^a and where ∇ is the metric compatible derivative operator (i.e., $\nabla(g) = 0$). In applications in General Relativity the g_{ab} are often chosen to be $g_{ab} = \text{diag}(-1, 1, 1, 1)$.

Here we will use the standard metric compatible connection

$$\Gamma_{ab}^d(x) = \frac{1}{2}g^{dc}(g_{cb,a} + g_{ac,b} - g_{ab,c})$$

to compute $\Gamma_{ab}^d(x)$ to terms linear in R_{abcd} and $\nabla_e R_{abcd}$.

```
1 {a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w#}::Indices(position=independent).
2
3 D{#}::PartialDerivative.
4 \nabla{#}::Derivative.
5
6 g_{a b}::Metric.
7 g^{a b}::InverseMetric.
8
9 \delta{#}::KroneckerDelta.
10
11 R_{a b c d}::RiemannTensor.
12
13 x^{a}::Depends(D{#}).
14 x^{a}::Weight(label=num,value=1).
15
16 R_{a b c d}::Depends(\nabla{#}).
17
18 DxaDxb := D_{a}{x^{b}}->\delta^{b}_{a}.
19
```



```

20 # can chose lower order approximations by truncating the following pair
21
22 gab := g_{a b} - 1/3 x^{c} x^{d} R_{a c b d}
23         - 1/6 x^{c} x^{d} x^{e} \nabla_{c}\{R_{a d b e}\}. # cdb(gab.000,gab)
24
25 iab := g^{a b} + 1/3 x^{c} x^{d} g^{a e} g^{b f} R_{c e d f}
26         + 1/6 x^{c} x^{d} x^{e} g^{a f} g^{b g} \nabla_{c}\{R_{d f e g}\}. # cdb(iab.000,iab)
27
28 gab := g_{a b} -> @(gab).
29 iab := g^{a b} -> @(iab).
30
31 gam := 1/2 g^{d c} (D_{a}\{g_{c b}\} + D_{b}\{g_{a c}\} - D_{c}\{g_{a b}\}). # cdb(gam.001,gam)
32
33 substitute (gam,gab)
34 substitute (gam,iab)
35 distribute (gam) # cdb(gam.002,gam)
36 unwrap (gam) # cdb(gam.003,gam)
37 product_rule (gam) # cdb(gam.004,gam)
38 distribute (gam) # cdb(gam.005,gam)
39 substitute (gam,DxaDxb) # cdb(gam.006,gam)
40 eliminate_kronecker (gam) # cdb(gam.007,gam)
41 sort_product (gam) # cdb(gam.008,gam)
42 rename_dummies (gam) # cdb(gam.009,gam)
43 canonicalise (gam) # cdb(gam.010,gam)
44
45 def truncate (obj,n):
46
47     ans = Ex(0) # create a Cadabra object with value zero
48
49     for i in range (0,n+1):
50         foo := @(obj).
51         bah = Ex("num = " + str(i))
52         distribute (foo)
53         keep_weight (foo, bah)
54         ans = ans + foo
55
56     return ans
57

```

```

58 gam = truncate (gam,2)    # cdb (gam.101,gam) # allow up to 2nd order in x^a
59
60 # =====
61 # the remaining code is just for pretty printing
62
63 {x^{a},g^{a b},R_{a b c d},\nabla_{e}\{R_{a b c d}\}}::SortOrder.
64
65 def get_term (obj,n):
66
67     foo := @(obj).
68     bah  = Ex("num = " + str(n))
69     distribute    (foo)
70     keep_weight   (foo, bah)
71
72     return foo
73
74 def reformat (obj,scale):
75
76     foo  = Ex(str(scale))
77     bah := @(foo) @(obj).
78
79     distribute    (bah)
80     sort_product  (bah)
81     rename_dummies (bah)
82     canonicalise   (bah)
83     factor_out     (bah,$x^{a?},g^{b? c?}$)
84     ans := @(bah) / @(foo).
85
86     return ans
87
88 gam1 = get_term (gam,1)    # cdb (gam1.301,gam1)
89 gam2 = get_term (gam,2)    # cdb (gam2.301,gam2)
90
91 gam1 = reformat (gam1, 3)   # cdb (gam1.301,gam1)
92 gam2 = reformat (gam2, 12)  # cdb (gam2.301,gam2)
93
94 Gamma := @(gam1) + @(gam2). # cdb (Gamma.301,Gamma)
95 Scaled := 12 @(Gamma).      # cdb (Scaled.301,Scaled)

```

The metric connection in Riemann normal coordinates

$$\Gamma_{ab}^d = \frac{1}{3}x^c g^{de} (R_{aebc} + R_{acbe}) + \frac{1}{12}x^c x^e g^{df} (\nabla_a R_{bcef} + 2\nabla_c R_{afbe} + \nabla_b R_{acef} + 2\nabla_c R_{aebf} + \nabla_f R_{acbe})$$

$$12\Gamma_{ab}^d = 4x^c g^{de} (R_{aebc} + R_{acbe}) + x^c x^e g^{df} (\nabla_a R_{bcef} + 2\nabla_c R_{afbe} + \nabla_b R_{acef} + 2\nabla_c R_{aebf} + \nabla_f R_{acbe})$$

```
\begin{dgroup*}
  \begin{dmath*} \Gamma^{\mathrm{d}}_{\mathrm{a} \mathrm{b}} = \mathrm{cdb}\{Gamma.301\} \end{dmath*}
\end{dgroup*}
```

```
\begin{dgroup*}
  \begin{dmath*} 12 \Gamma^{\mathrm{d}}_{\mathrm{a} \mathrm{b}} = \mathrm{cdb}\{Scaled.301\} \end{dmath*}
\end{dgroup*}
```

Using tagged blocks

The following Python code block contains a matched `cdbBeg/cdbEnd` pair, with the tag name `info`, to capture the output from the formatted Python `print` statements.

```
import platform, datetime
# cdbBeg(info)
print("date :      &"+'{:a %d %b %Y %H:%M:%S}'.format(datetime.datetime.now())+"\\\\"")
print("python :    &"+str(platform.python_version())+"\\\\"")
print("sympy :      &"+str(sympy.__version__)+"\\\\"")
print("system :     &"+str(platform.system())+"\\\\"")
print("release :    &"+str(platform.release())+"\\\\"")
print("machine :    &"+str(platform.machine())+"\\\\"")
print("processor :  &"+str(platform.processor())+"\\\\"")
print("platform :   &"+str(platform.platform())
# cdbEnd(info)
```

```
\bgroup\tt
\begin{tabular}{rl}
\cdb{info}
\end{tabular}
\egroup
```

Here is the output caught from the above block.

```
date : Wed 17 Nov 2021 13:53:30
python : 3.9.7
sympy : 1.7.1
system : Darwin
release : 20.6.0
machine : x86_64
processor : i386
platform : macOS-11.6-x86_64-i386-64bit
```