xConf. Documentation

This is the doc file xConfDoc.nb of version 0.0.1 of xAct/xConf`. Last update on 04 April 2025.

Preamble

Author

Authors: Cyril Pitrou, Guillaume Faye

Intro

xAct`xConf` is a Mathematica Package which provides tools to compute formally the conformal transformations.

Loading the package

Loading the package is straightforward once it has been appropriately installed:

You just have to evaluate: '<< xAct`xConf.m;'

This loads all the definitions, but does not start any computation.

We can also check how much memory space it takes to store all the definitions, and the time it takes to load them

In[1]:=
```
MemBefore = MemoryInUse[];
Needs["xAct`xConf`"]; // Timing // First
```

```
------------------------------------------------------------

Package xAct`xPerm`  version 1.2.3, {2015, 8, 23}

CopyRight (C) 2003-2020, Jose M.
  Martin-Garcia, under the General Public License.

Connecting to external mac executable...

Connection established.

------------------------------------------------------------

Package xAct`xTensor`  version 1.2.0, {2021, 10, 17}

CopyRight (C) 2002-2021, Jose M.
  Martin-Garcia, under the General Public License.

------------------------------------------------------------

Package xAct`xConf`  version 0.0.1, {2025, 4, 4}

CopyRight (C) 2025-2026, Cyril
  Pitrou, Guillaume under the General Public License.

------------------------------------------------------------

These packages come with ABSOLUTELY NO WARRANTY; for details type
  Disclaimer[]. This is free software, and you are welcome to redistribute
  it under certain conditions. See the General Public License for details.

------------------------------------------------------------

** Variable $PrePrint assigned value ScreenDollarIndices

** Option AllowUpperDerivatives of ContractMetric changed from False to True

** Option UseMetricOnVBundle of ToCanonical changed from All to None
```

Out[2]=
```
0.248054
```

In[3]:=
```
MemoryInUse[] - MemBefore
```

Out[3]=
```
15 164 752
```

GPL

In[4]:=
```
Disclaimer[]
```

```
These are points 11 and 12 of the General Public License:

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
  FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT
  WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER
  PARTIES PROVIDE THE PROGRAM `AS IS´ WITHOUT WARRANTY OF ANY KIND,
  EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
  PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF
  THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU
  ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
  WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
  REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR
  DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL
  DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM
  (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED
  INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF
  THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER
  OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
```

Overview

Defining Manifold and fields

We define a Manifold and a base metric g

In[5]:=
```
DefManifold[M, 4, {α, β, γ, μ, ν, λ, σ}]
DefMetric[-1, g[-α, -β], CD, {";", "∇"}, PrintAs → "g"];
```
```
** DefManifold: Defining manifold M.

** DefVBundle: Defining vbundle TangentM.

** DefTensor: Defining symmetric metric tensor g[-α, -β].

** DefTensor: Defining antisymmetric tensor epsilong[-α, -β, -γ, -λ].

** DefTensor: Defining tetrametric Tetrag[-α, -β, -γ, -λ].

** DefTensor: Defining tetrametric Tetrag†[-α, -β, -γ, -λ].

** DefCovD: Defining covariant derivative CD[-α].

** DefTensor: Defining vanishing torsion tensor TorsionCD[α, -β, -γ].

** DefTensor: Defining symmetric Christoffel tensor ChristoffelCD[α, -β, -γ].

** DefTensor: Defining Riemann tensor RiemannCD[-α, -β, -γ, -λ].

** DefTensor: Defining symmetric Ricci tensor RicciCD[-α, -β].

** DefCovD:  Contractions of Riemann automatically replaced by Ricci.

** DefTensor: Defining Ricci scalar RicciScalarCD[].

** DefCovD:  Contractions of Ricci automatically replaced by RicciScalar.

** DefTensor: Defining symmetric Einstein tensor EinsteinCD[-α, -β].

** DefTensor: Defining Weyl tensor WeylCD[-α, -β, -γ, -λ].

** DefTensor: Defining symmetric TFRicci tensor TFRicciCD[-α, -β].

** DefTensor: Defining Kretschmann scalar KretschmannCD[].

** DefCovD:  Computing RiemannToWeylRules for dim 4

** DefCovD:  Computing RicciToTFRicci for dim 4

** DefCovD:  Computing RicciToEinsteinRules for dim 4

** DefTensor: Defining weight +2 density Detg[]. Determinant.
```

We define a a few tensors on that manifold

In[7]:=
```
DefTensor[k[-α], {M}, PrintAs → "k"]
DefTensor[MT2[-α, -β], {M}, PrintAs → "M"]
```
```
** DefTensor: Defining tensor k[-α].

** DefTensor: Defining tensor MT2[-α, -β].
```

Conformal Weights of tensors

Each tensor has a default conformal weight

In[9]:=
```
? ConformalWeight
```

Out[9]=

Symbol

ConformalWeight[tensor[inds]] specifies the power of the conformal factor applied in the conformal transformation of the tensor 'tensor'. It corresponds to the number of down indices, minus the number of up indices, plus the quantity ConformalWeight[tensor] (without the indices). The latter is set by default to zero.

For instance, with the setting ConformalWeight[tensor] = 0, we have: ConformalWeight[tensor[α, β]] = -2, and ConformalWeight[tensor[- α,- β]] = 2.

In[10]:=
```
ConformalWeight[k]
ConformalWeight[MT2]
```

Out[10]=
```
0
```

Out[11]=
```
0
```

It can be modified

In[12]:=
```
ConformalWeight[k] = -1;
ConformalWeight[k]
```

Out[13]=
```
-1
```

The weight of the tensor with indices then depends on the positioning of indices

In[14]:=
```
ConformalWeight[k[-μ]]
ConformalWeight[k[μ]]
ConformalWeight[MT2[-α, -β]]
ConformalWeight[MT2[α, β]]
ConformalWeight[MT2[α, -β]]
```

Out[14]=
```
0
```

Out[15]=
```
-2
```

Out[16]=
```
2
```

Out[17]=
```
-2
```

Out[18]=
```
0
```

Conformally related metrics

We can define metrics which are conformally related to previous metrics

In[19]:=
```
?DefConformalMetric
```

Out[19]=

Symbol

DefConformalMetric[g,S] defines a metric conformally related to 'g', with the conformal factor 'S'. If other metrics are conformally related to 'g', then 'gS2' is related to them by transitivity of the conformal transformation. If the scalar field S[] does not exist, it is defined.

Options are PrintAs->printasform, ConformalMetricName->MetricName, FrameColor->somecolor and SymbolOfCovD->{postfixsymbol,prefixsymbol}. If no ConformalMetricName is provided the name chosen is the concatenation 'metric+conformalfactor+2'. If no color is provided, it choses randomly a color. If no SymbolOfCovD is provided, the combination {:, ∇metricname} is chosen.

In[20]:=
```
DefConformalMetric[g, S, PrintAs -> "gS2",
 ConformalMetricName -> gS2, FrameColor -> Red]
DefConformalMetric[g, s, FrameColor -> Blue]
```

```
Choosing color ■ for this conformal frame.

** DefTensor: Defining tensor S[].

** DefTensor: Defining symmetric metric tensor gS2[-σ$8221, -σ$8222].

** DefTensor: Defining inverse metric tensor
 InvgS2[σ$8221, σ$8222]. Metric is frozen!
** DefTensor: Defining antisymmetric tensor epsilongS2[-α, -β, -γ, -λ].

** DefTensor: Defining tetrametric TetragS2[-α, -β, -γ, -λ].

** DefTensor: Defining tetrametric TetragS2†[-α, -β, -γ, -λ].

** DefCovD: Defining covariant derivative CDgS2[-σ$8221].

** DefTensor: Defining vanishing torsion tensor TorsionCDgS2[α, -β, -γ].

** DefTensor: Defining
 symmetric Christoffel tensor ChristoffelCDgS2[α, -β, -γ].
** DefTensor: Defining Riemann tensor RiemannDownCDgS2[-α, -β, -γ, -λ].

** DefTensor: Defining Riemann tensor
 RiemannCDgS2[-α, -β, -γ, λ]. Antisymmetric only in the first pair.
** DefTensor: Defining symmetric Ricci tensor RicciCDgS2[-α, -β].

** DefCovD:  Contractions of Riemann automatically replaced by Ricci.

** DefTensor: Defining Ricci scalar RicciScalarCDgS2[].

** DefCovD:  Contractions of Ricci automatically replaced by RicciScalar.

** DefTensor: Defining symmetric Einstein tensor EinsteinCDgS2[-α, -β].

** DefTensor: Defining Weyl tensor WeylCDgS2[-α, -β, -γ, -λ].

** DefTensor: Defining symmetric TFRicci tensor TFRicciCDgS2[-α, -β].

** DefTensor: Defining Kretschmann scalar KretschmannCDgS2[].

** DefCovD:  Computing RiemannToWeylRules for dim 4

** DefCovD:  Computing RicciToEinsteinRules for dim 4

** DefTensor: Defining weight +2 density DetgS2[]. Determinant.

** DefInertHead: Defining inert head ConformalFrame[gS2].

** DefInertHead: Defining inert head ConformalFrame[g].

Choosing color ■ for this conformal frame.

** DefTensor: Defining tensor s[].

** DefTensor: Defining symmetric metric tensor gs2[-σ$8605, -σ$8606].

** DefTensor: Defining inverse metric tensor
 Invgs2[σ$8605, σ$8606]. Metric is frozen!
** DefTensor: Defining antisymmetric tensor epsilongs2[-α, -β, -γ, -λ].

** DefTensor: Defining tetrametric Tetrags2[-α, -β, -γ, -λ].

** DefTensor: Defining tetrametric Tetrags2†[-α, -β, -γ, -λ].

** DefCovD: Defining covariant derivative CDgs2[-σ$8605].

** DefTensor: Defining vanishing torsion tensor TorsionCDgs2[α, -β, -γ].

** DefTensor: Defining
 symmetric Christoffel tensor ChristoffelCDgs2[α, -β, -γ].
** DefTensor: Defining Riemann tensor RiemannDownCDgs2[-α, -β, -γ, -λ].
```

```
** DefTensor: Defining Riemann tensor
 RiemannCDgs2[-α, -β, -γ, λ]. Antisymmetric only in the first pair.
** DefTensor: Defining symmetric Ricci tensor RicciCDgs2[-α, -β].
** DefCovD:  Contractions of Riemann automatically replaced by Ricci.
** DefTensor: Defining Ricci scalar RicciScalarCDgs2[].
** DefCovD:  Contractions of Ricci automatically replaced by RicciScalar.
** DefTensor: Defining symmetric Einstein tensor EinsteinCDgs2[-α, -β].
** DefTensor: Defining Weyl tensor WeylCDgs2[-α, -β, -γ, -λ].
** DefTensor: Defining symmetric TFRicci tensor TFRicciCDgs2[-α, -β].
** DefTensor: Defining Kretschmann scalar KretschmannCDgs2[].
** DefCovD:  Computing RiemannToWeylRules for dim 4
** DefCovD:  Computing RicciToEinsteinRules for dim 4
** DefTensor: Defining weight +2 density Detgs2[]. Determinant.
** DefInertHead: Defining inert head ConformalFrame[gs2].
```

The conformal relations have been defined among these 3 metrics

In[22]:=
```
ConformalRules[g, gS2]
ConformalRules[g, gs2]
ConformalRules[gS2, gs2]
```

Out[22]=
$$\left\{ g^{\alpha\beta}_{--} \to \frac{gS2^{\alpha\beta}}{S^2},\ g^{\alpha\beta}_{--} \to igS2^{\alpha\beta}\ S^2,\ \tilde{\tilde{g}} \to \frac{\tilde{gS2}}{S^8} \right\}$$

Out[23]=
$$\left\{ g^{\alpha\beta}_{--} \to \frac{[g\,s^2]^{\alpha\beta}}{s^2},\ g^{\alpha\beta}_{--} \to i[g\,s^2]^{\alpha\beta}\ s^2,\ \tilde{\tilde{g}} \to \frac{[\tilde{g\,s^2}]}{s^8} \right\}$$

Out[24]=
$$\left\{ gS2^{\alpha\beta}_{--} \to \frac{[g\,s^2]^{\alpha\beta}\ S^2}{s^2},\ igS2^{\alpha\beta}_{--} \to \frac{i[g\,s^2]^{\alpha\beta}\ s^2}{S^2},\ \tilde{gS2} \to \frac{[\tilde{g\,s^2}]\ S^8}{s^8} \right\}$$

ConformalFrame inert head

ConformalFrame[metric][tensor[indices]] represents the transformation of tensor[indices] in a given conformal frame.

In[25]:=
```
? ConformalFrame
```

Out[25]=

Symbol

Formal scalar head ConformalFrame[metric][expr] which indicates that expr (which must be of the form tens[inds]) is in fact the transformation of that quantity from the ambient frame to the frame associated with metric. Hence if metric is the first metric ConformalFrame[firstmetric][expr] = expr.

In[26]:=
```
ConformalFrame[gS2][MT2[-α, -β]]
```

Out[26]=

$$[M_{\alpha\beta}]$$

The formatting depends on the global option $FormatConformal which can either be “Default” or “Color”. The user choses what he/she finds most convenient to read.

In[27]:=
```
$FormatConformal = "Default";
```

In[28]:=
```
ConformalFrame[gS2][MT2[-α, -β]]
```

Out[28]=

$$^{gS2}[M_{\alpha\beta}]$$

In[29]:=
```
$FormatConformal = "Color";
```

In[30]:=
```
ConformalFrame[gS2][MT2[-α, -β]]
```

Out[30]=

$$[M_{\alpha\beta}]$$

ToMetric

In[31]:=
```
? ToMetric
```

Out[31]=

Symbol

ToMetric[metric][expr] formulates 'expr' in terms of conformal frame 'metric', and its associated covariant derivative and curvature tensors. It is a passive transformation.

ToMetric[metric][epxr] performs a passive transformation. It expressed a given expressions expr with tensors evaluated in a given frame metric.

In[32]:=
```
ToMetric[gS2][g[-μ, -ν]]
ToMetric[gS2][g[μ, ν]]
```

```
** DefTensor: Defining tensor ChristoffelCDCDgS2[α, -β, -γ].

** DefTensor: Defining tensor ChristoffelCDgs2CDgS2[α, -β, -γ].
```

Out[32]=

$$\frac{gS2_{\mu\nu}}{S^2}$$

Out[33]=

$$igS2^{\mu\nu}\ S^2$$

When applied on covariant derivatives it changes the derivative in which the expression is expressed

In[34]:=
```
ToMetric[gS2][CD[-μ][k[-ν]]] // NoScalar
```

Out[34]=

$$\nabla gS2_{\mu}[k_{\nu}] - \frac{gS2_{\mu\nu}\ igS2^{\alpha\beta}\ (\nabla gS2_{\beta}\ S)\ [k_{\alpha}]}{S} + \frac{(\nabla gS2_{\nu}\ S)\ [k_{\mu}]}{S} + \frac{(\nabla gS2_{\mu}\ S)\ [k_{\nu}]}{S}$$

When applied on curvature tensors it changes the curvature tensor in which the expression is expressed

In[35]:=
```
ToMetric[gS2][RicciCD[-μ, -ν]]
```

Out[35]=

$$R[\nabla gS2]_{\mu\nu} - \frac{3\ gS2_{\mu\nu}\ (igS2^{\alpha\beta}\ (\nabla gS2_{\alpha}\ S)\ (\nabla gS2_{\beta}\ S))}{S^2} + \frac{gS2_{\mu\nu}\ (igS2^{\alpha\beta}\ (\nabla gS2_{\beta}\ \nabla gS2_{\alpha}\ S))}{S} + \frac{2\ (\nabla gS2_{\nu}\ \nabla gS2_{\mu}\ S)}{S}$$

ConformalTransformation

In[36]:=
```
? ConformalTransformation
```

Out[36]=

Symbol

ConformalTransformation[metric1,metric2][expr]
performs a conformal transformation. More precisely it is a
Weyl transformation, but we call it conformal transformation
as in appendix D of Wald's book on General Relativity (1984)).

The expression 'expr' is transformed
actively from the frame 'metric1' to the frame 'metric2'.
The two metrics 'metric1' and 'metric2'
have to be conformally related by DefConformalMetric.

The Option FinalFrame is by default set to 'Automatic' and in that
case all quantities in the result are expressed in the metric1 frame.
The use can specifiy FinalFrame->targetframe,
where targetframe is a metric name, to express the results
in terms of quantities in the targetframe conformal frame.

It is an active transfromation of an expression going from frame metric1 to frame metric2.

It uses ToMetric[metric1] to first express everything in terms of quantities in frame metric1, then it applies ConformalFrame[metric2] on the expression to actively change the expression.

The rules built by RulesChangeConformalFrame[metric1,metric2] then replace all metric related quantites by their value in frame metric2 (it is automatic).

Finally ToMetric[finalframe] is applied to the result. By default finalframe = metric1.

In[37]:=
```
ConformalTransformation[g, gS2][CD[μ][k[ν]]]
```
```
** DefTensor: Defining tensor ChristoffelCDCDgs2[α, -β, -γ].
```

Out[37]=

$$\frac{g^{\mu\nu}\,(g^{\alpha\beta}\,k_\alpha\,\nabla_\beta S)}{S^5}+\frac{g^{\mu\beta}\,g^{\nu\alpha}\,\nabla_\beta k_\alpha}{S^4}-\frac{g^{\mu\beta}\,g^{\nu\alpha}\,k_\alpha\,\nabla_\beta S}{S^5}-\frac{g^{\mu\alpha}\,g^{\nu\beta}\,k_\alpha\,\nabla_\beta S}{S^5}$$

In[38]:=
```
ConformalTransformation[g, gS2][RicciCD[μ, ν]]
```

Out[38]=
$$\frac{g^{\mu\alpha}\ g^{\nu\beta}\ R[\nabla]_{\alpha\beta}}{S^4} - \frac{g^{\mu\nu}\ (g^{\alpha\beta}\ \nabla_\alpha S\ \nabla_\beta S)}{S^6} - \frac{g^{\mu\nu}\ (g^{\alpha\beta}\ \nabla_\beta\nabla_\alpha S)}{S^5} + \frac{4\ g^{\mu\alpha}\ g^{\nu\beta}\ \nabla_\alpha S\ \nabla_\beta S}{S^6} - \frac{2\ g^{\mu\alpha}\ g^{\nu\beta}\ \nabla_\beta\nabla_\alpha S}{S^5}$$

The result is then expressed by default in the frame metric1, but this can be modified by the option FinalFrame->metricfinal.

If the metricfinal=metric2, then formally we have replaced all expressions in metric1 by their counterpart in the frame metric2 and the transformation is trivial

In[39]:=
```
ConformalTransformation[g, gS2, FinalFrame -> gS2][CD[μ][k[ν]]]
```

Out[39]=
$$igS2^{\mu\beta}\ igS2^{\nu\alpha}\ \nabla gS2_\beta[k_\alpha]$$

In[40]:=
```
ConformalTransformation[g, gS2, FinalFrame -> gS2][RicciCD[μ, ν]]
```

Out[40]=
$$igS2^{\mu\alpha}\ igS2^{\nu\beta}\ R[\nabla gS2]_{\alpha\beta}$$

Simple examples

We can check that the Weyl tensor is invariant.

In[41]:=
```
ConformalTransformation[g, gS2][WeylCD[-μ, -ν, -α, β]]
% // RiemannToWeyl // ContractMetric // ToCanonical
```

Out[41]=
$$-\frac{1}{2}\ \delta^\beta{}_\nu\ R[\nabla]_{\alpha\mu} + \frac{1}{2}\ \delta^\beta{}_\mu\ R[\nabla]_{\alpha\nu} + \frac{1}{2}\ g_{\alpha\nu}\ g^{\beta\gamma}\ R[\nabla]_{\mu\gamma} - \frac{1}{2}\ g_{\alpha\mu}\ g^{\beta\gamma}\ R[\nabla]_{\nu\gamma} + \frac{1}{6}\ \delta^\beta{}_\nu\ g_{\alpha\mu}\ R[\nabla] - \frac{1}{6}\ \delta^\beta{}_\mu\ g_{\alpha\nu}\ R[\nabla] + g^{\beta\gamma}\ g_{\nu\lambda}\ R[\nabla]_{\alpha\gamma\mu}{}^\lambda$$

Out[42]=
$$W[\nabla]_\alpha{}^\beta{}_{\mu\nu}$$

Summary of public functions and variables

In[43]:=

```
? xAct`xConf` *
```

Out[43]=

xAct`xConf`

ConformalFrame	DefConformalMetric	RulesChangeConformalFrame	$Debug
ConformalMetricName	Disclaimer	SqrtConformalFactor	$FormatConformal
ConformalTransformation	FinalFrame	TensorNotMetricRelated	$Version
ConformalWeight	FrameColor	ToMetric	$xTensorVersionExpected