xAct`Spinors`

This is the doc file SpinorsDoc.nb of version 0.9.2 of `Spinors`. Last update on 9 September 2009.

■ Authors

© 2006–2009, under the GNU General Public License (GPL)

Alfonso García–Parrado Gómez–Lobo

Centro de Matematica

Universidade do Minho, Portugal

algar@mai.liu.se

and

José M. Martín–García

Institut d'Astrophysique de Paris & Laboratoire Univers et Théories, CNRS, France

garcia@iap.fr

http://metric.iem.csic.es/Martin-Garcia/

■ Intro

`Spinors`` is the `xAct`` package for computations with spinors in four dimensional Lorentzian manifolds. The conventions adopted in the standard formulae of the spinor calculus are those of Penrose & Rindler, *Spinors and Space–Time,* Cambridge University Press (Vols. 1, 2). We assume that the user is already familiar with spinor calculus and therefore we will not explain the terminology or symbols related to this subject unless necessary to clarify the workings of the package.

■ Load the package

This loads the package from the default directory, for example $Home/.*Mathematica*/Applications/xAct/ for a single–user installation under Linux.

```
In[1]:= MemoryInUse[]

Out[1]= 14 362 304
```

```
In[2]:= <<xAct`Spinors`

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

        Package xAct`xPerm`  version 1.0.3, {2009, 9, 9}

        CopyRight (C) 2003-2008, Jose M.
          Martin-Garcia, under the General Public License.

        Connecting to external linux executable...

        Connection established.

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

        Package xAct`xTensor`  version 0.9.9, {2009, 9, 9}

        CopyRight (C) 2002-2008, Jose M.
          Martin-Garcia, under the General Public License.

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

        Package xAct`Spinors`  version 0.9.2, {2009, 9, 9}

        CopyRight (C) 2006-2008, Alfonso Garcia-Parrado Gomez-Lobo
          and Jose M. Martin-Garcia, 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.

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

Comparing, we see that all packages take about 50 Mb in *Mathematica* 7.0:

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

Out[3]= 69 895 128

In[4]:= (Out[3] - Out[1]) / 2^20 // N

Out[4]= 52.9602
```

There are several contexts: `xAct`Spinors`, xAct`xTensor`, xAct`xPerm` and xAct`xCore` contain the respective reserved words. `System` contains *Mathematica*'s reserved words. The current context `Global` will contain your definitions and right now is empty.

```
In[5]:= $ContextPath

Out[5]= {xAct`Spinors`, xAct`xTensor`, xAct`xPerm`,
          xAct`xCore`, PacletManager`, WebServices`, System`, Global`}

In[6]:= Context[]

Out[6]= Global`
```

©2003–2004 José M. Martín–García

```
In[7]:= ?Global`*
```

```
Information::nomatch : No symbol matching Global`* found. >>
```

■ 1. Example session

Spinors are elements of a complex vector bundle (spin bundle) whose base space is a four dimensional Lorentzian manifold (spacetime). Therefore we need to define first a four dimensional differentiable manifold and a Lorentzian metric, which will be kept fixed in all this notebook. See `xTensor`` doc file for details about how to carry this out. Note also that all the tools and notation developed in `xAct`` to work with generic complex vector bundles apply in the particular case of a spin bundle, so the user will be assumed familiar with them. We refer again to the `xTensor`` doc file for detailed documentation about this point.

Definition of a four dimensional manifold `M4`:

```
In[8]:= DefManifold[M4, 4, {a, b, c, d, f, p, q, r, m, l, h, j, n, t, s}]
```

```
** DefManifold: Defining manifold M4.

** DefVBundle: Defining vbundle TangentM4.
```

©2003–2004 José M. Martín–García

Definition of a Lorentzian metric on the manifold:

```
In[9]:= DefMetric[{3, 1, 0}, g[-a, -b], CD]

        ** DefTensor: Defining symmetric metric tensor g[-a, -b].

        ** DefTensor: Defining antisymmetric tensor epsilong[a, b, c, d].

        ** DefTensor: Defining tensor Tetrag[-a, -b, -c, -d].

        ** DefTensor: Defining tensor Tetrag†[-a, -b, -c, -d].

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

        ** DefTensor: Defining vanishing torsion tensor TorsionCD[a, -b, -c].

        ** DefTensor: Defining symmetric Christoffel tensor ChristoffelCD[a, -b, -c].

        ** DefTensor: Defining Riemann tensor RiemannCD[-a, -b, -c, -d].

        ** DefTensor: Defining symmetric Ricci tensor RicciCD[-a, -b].

        ** 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[-a, -b].

        ** DefTensor: Defining Weyl tensor WeylCD[-a, -b, -c, -d].

        ** DefTensor: Defining symmetric TFRicci tensor TFRicciCD[-a, -b].

        ** 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.
```

The command `DefSpinStructure` defines a complex spin bundle named `Spin` comprising the spin metric ϵ`[-A,-B]`, the soldering form σ`[a,-A,-A†]` and the extension of the Levi–Civita covariant derivative `CD` to the unique covariant derivative `CDe` which is compatible with the soldering form and the spin metric (spin covariant derivative). The option `SpinorPrefix` is used to introduce a tag which will be prepended automatically to any spinor related to a spacetime tensor.

```
In[10]:= DefSpinStructure[g, Spin, {A, B, C, D, F, H, L, P, Q}, ϵ, σ, CDe, SpinorPrefix -> SP]

         ** DefVBundle: Defining vbundle Spin.

       ValidateSymbol::capital :
        System name C is overloaded as an abstract index.

       ValidateSymbol::capital :
        System name D is overloaded as an abstract index.
```

©2003–2004 José M. Martín–García

```
** DefVBundle: Defining conjugated vbundle Spin†
 . Assuming fixed anti-isomorphism between Spin and Spin†

** DefTensor: Defining soldering form σ[a, -C, -D†].

** DefTensor: Defining soldering form σ †[a, -C†, -D].

** DefTensor: Defining spin metric ϵ[-A, -B].

** DefTensor: Defining spin metric ϵ †[-A†, -B†].

** DefTensor: Defining tensor Sigmaσ[a, b, -A, -B].

** DefTensor: Defining tensor Sigmaσ †[a, b, -A†, -B†].

** DefTensor: Defining spinor SP␣g[A, C†, B, D†]. Equivalent of tensor g

** DefTensor: Defining spinor SP␣g†[A†, C, B†, D]. Equivalent of tensor g

** DefTensor: Defining spinor
 SP␣epsilong[A, A†, B, B†, C, C†, D, D†]. Equivalent of tensor epsilong

** DefTensor: Defining spinor
 SP␣epsilong†[A†, A, B†, B, C†, C, D†, D]. Equivalent of tensor epsilong

** DefCovD: Defining covariant derivative CDe[-a].

** DefTensor: Defining
 nonsymmetric AChristoffel tensor AChristoffelCDe[A, -b, -C].

** DefTensor: Defining
 nonsymmetric AChristoffel tensor AChristoffelCDe†[A†, -b, -C†].

** DefTensor: Defining FRiemann tensor
 FRiemannCDe[-a, -b, -C, D]. Antisymmetric only in the first pair.

** DefTensor: Defining FRiemann tensor
 FRiemannCDe†[-a, -b, -C†, D†]. Antisymmetric only in the first pair.

** DefTensor: Defining spinor
 SP␣FRiemannCDe[-A, -A†, -B, -B†, -C, -D]. Equivalent of tensor FRiemannCDe

** DefTensor: Defining spinor
 SP␣FRiemannCDe†[-A†, -A, -B†, -B, -C†, -D†]. Equivalent of tensor FRiemannCDe†

** DefTensor: Defining curvature spinor ChiCDe[-A, -B, -C, -D].

** DefTensor: Defining curvature spinor ChiCDe†[-A†, -B†, -C†, -D†].

** DefTensor: Defining curvature spinor PhiCDe[-A, -B, -C†, -D†].

** DefTensor: Defining curvature spinor PhiCDe†[-A†, -B†, -C, -D].

** DefTensor: Defining spinor scalar curvature LambdaCDe[].

** DefTensor: Defining Weyl spinor PsiCDe[-A, -B, -C, -D].

** DefTensor: Defining Weyl spinor PsiCDe†[-A†, -B†, -C†, -D†].

** DefTensor: Defining spinor SP␣RiemannCD[-A, -A†, -B, -B†, -C, -C†, -D, -D†]
 . Equivalent of tensor RiemannCD
```

©2003–2004 José M. Martín–García

```
** DefTensor: Defining spinor SP$RiemannCD†[-A†, -A, -B†, -B, -C†, -C, -D†, -D]
 . Equivalent of tensor RiemannCD

** DefTensor: Defining spinor
 SP$RicciCD[-A, -A†, -B, -B†]. Equivalent of tensor RicciCD

** DefTensor: Defining spinor
 SP$RicciCD†[-A†, -A, -B†, -B]. Equivalent of tensor RicciCD

** DefTensor: Defining spinor
 SP$WeylCD[-A, -A†, -B, -B†, -C, -C†, -D, -D†]. Equivalent of tensor WeylCD

** DefTensor: Defining spinor
 SP$WeylCD†[-A†, -A, -B†, -B, -C†, -C, -D†, -D]. Equivalent of tensor WeylCD

** DefTensor: Defining spinor
 SP$TFRicciCD[-A, -A†, -B, -B†]. Equivalent of tensor TFRicciCD

** DefTensor: Defining spinor
 SP$TFRicciCD†[-A†, -A, -B†, -B]. Equivalent of tensor TFRicciCD
```

When a spin structure is defined, several new objects are defined automatically. These are the spinor equivalents of the Riemann, the Ricci, the traceless Ricci and the Weyl tensors. Note the character σ prepended to each head in the output . This is the default output form of the tag mentioned above. This output form can be changed with the option `SpinorMark` in the command `DefSpinStructure`.

```
In[11]:= {SP$RiemannCD[-A, -A†, -B, -B†, -C, -C†, -F, -F†], SP$RicciCD[-A, -A†, -B, -B†],
          SP$TFRicciCD[-A, -A†, -B, -B†], SP$WeylCD[-A, -A†, -B, -B†, -C, -C†, -F, -F†]}

Out[11]= {σ R[∇]AA† BB† CC† FF†, σ R[∇]AA† BB†, σ S[∇]AA† BB†, σ W[∇]AA† BB† CC† FF†}
```

The curvature spinors are also automatically defined.

```
In[12]:= {ChiCDe[-A, -B, -C, -D], PhiCDe[-A, -B, -A†, -B†]}

Out[12]= {X[∇]ABCD, Φ[∇]ABA† B†}
```

Each of the above spinors can be decomposed into irreducible parts by means of the command `Decomposition`. Let us see some examples.

```
In[13]:= Decomposition[SP$RiemannCD[-A, -A†, -B, -B†, -C, -C†, -D, -D†]]

Out[13]= Ψ[∇] ϵA† B† D† C† ϵAB ϵDC − Φ[∇]ABD† C† ϵDC ϵ A† B† −
          2 Λ[∇] (ϵAD ϵBC ϵ A† D† ϵ B† C† − ϵAC ϵBD ϵ A† C† ϵ B† D†) −
          Φ[∇]DCA† B† ϵAB ϵ D† C† + Ψ[∇]ABDC ϵ A† B† ϵ D† C†
```

The decomposition of the spinor equivalent of the Ricci tensor is

```
In[14]:= Decomposition[SP$RicciCD[-A, -A†, -B, -B†]]

Out[14]= −2 Φ[∇]ABA† B† + 6 Λ[∇] ϵAB ϵ A† B†
```

The decomposition of the spinor equivalent of the Weyl tensor is

```
In[15]:= Decomposition[SP$WeylCD[-A, -A†, -B, -B†, -C, -C†, -D, -D†]]

Out[15]= Ψ[∇] ϵA† B† D† C† ϵAB ϵDC + Ψ[∇]ABDC ϵ A† B† ϵ D† C†
```

©2003–2004 José M. Martín–García

The inner curvature of the spin bundle can be also decomposed according to standard rules .

```
In[16]:= Decomposition[SP`FRiemannCDe[-A, -A†, -B, -B†, -C, -D]]

Out[16]= Φ[∇]CDA† B† ϵAB + X[∇]CDAB ∈ ⨯A† B†

In[17]:= Decomposition[%, Chi]

Out[17]= Φ[∇]CDA† B† ϵAB + (Ψ[∇]CDAB + Λ[∇] (ϵCB ϵDA + ϵCA ϵDB)) ∈ ⨯A† B†

In[18]:= ChiCDe[-A, -B, -C, -D]

Out[18]= X[∇]ABCD

In[19]:= Decomposition[%, Chi]

Out[19]= Ψ[∇]ABCD + Λ[∇] (ϵAD ϵBC + ϵAC ϵBD)
```

`Spinors`' is able to work with the 2–index derivatives used in spinor calculus.

```
In[20]:= CDe[-F, -F†]@PsiCDe[-A, -B, -C, -D]

Out[20]= ∇FF† Ψ[∇]ABCD
```

As is well–known this is nothing but a shorthand for

```
In[21]:= SeparateSolderingForm[][%, CDe]

Out[21]= σ^a FF† (∇a Ψ[∇]ABCD)
```

We can also transform one–index covariant derivatives back into their two index form by means of the soldering form.

```
In[22]:= CDe[-b]@PsiCDe[-A, -B, -C, -D]

Out[22]= ∇b Ψ[∇]ABCD

In[23]:= PutSolderingForm@%

Out[23]= σ^a FF† (∇a Ψ[∇]ABCD)

In[24]:= ContractSolderingForm@%

Out[24]= ∇FF† Ψ[∇]ABCD
```

Spinors are defined using the command `DefSpinor`, which is nothing but a renaming of standard `xAct`' command `DefTensor` (by default the option `Dagger->Complex` is assumed).

```
In[25]:= DefSpinor[κ[-A, -A†], M4]

** DefTensor: Defining tensor κ[-A, -A†].

** DefTensor: Defining tensor κ⨯[-A†, -A].
```

©2003–2004 José M. Martín–García

In this case the spinor just defined can be related to a tensor in the spacetime tangent bundle. This vector is given by

```
In[26]:= PutSolderingForm[κ[-A, -A†]]

Out[26]= κ_{AA†} σ_a^{AA†}
```

The soldering form can be absorbed into the spinor κ. In this case a new tensor is automatically defined.

```
In[27]:= ContractSolderingForm[%]

      TensorOfSpinor::name : Tensor of κ not defined. Prepending SP.

         ** DefTensor: Defining tensor SP_κ[-a]. Equivalent of spinor κ

         ** DefTensor: Defining tensor SP_κ†[-a]. Equivalent of spinor κ†

Out[27]= σκ_a
```

The original spinor may be also recovered from $\sigma\kappa_a$.

```
In[28]:= PutSolderingForm[%]

Out[28]= σκ_a σ^a_{AA†}

In[29]:= ContractSolderingForm[%]

Out[29]= κ_{AA†}
```

An interesting feature is the possibility of working with Hermitian spinors. These are defined by supplying the option `Dagger->Hermitian` to `DefSpinor`.

```
In[30]:= DefSpinor[μ[-A, -A†, -B, -B†], M4, Dagger -> Hermitian]

         ** DefTensor: Defining tensor μ[-A, -A†, -B, -B†].

         ** DefTensor: Defining tensor μ†[-A†, -A, -B†, -B].
```

We check that the new spinor μ is hermitian.

```
In[31]:= μ[-A, -A†, -B, -B†] // Dagger

Out[31]= μ_{AA† BB†}
```

Any Hermitian spinor has a tensor counterpart which is real. We compute the tensor counterpart of μ and check that the result is a real rank−2 spacetime tensor.

```
In[32]:= % // PutSolderingForm // ContractSolderingForm

      TensorOfSpinor::name : Tensor of μ not defined. Prepending SP.

         ** DefTensor: Defining tensor SP_μ[-a, -b]. Equivalent of spinor μ

Out[32]= σμ_{ab}
```

©2003−2004 José M. Martín−García

In[33]:= **% // Dagger**

Out[33]= $\sigma\mu_{ab}$

The `xAct`` command `ContractMetric` is aware of the standard index raising and lowering conventions for the spinor indices and therefore it can act on spinor expressions without further options. We present some examples:

In[34]:= **DefSpinor[ξ[A], M4]**

```
** DefTensor: Defining tensor ξ[A].

** DefTensor: Defining tensor ξ†[A†].
```

We consider the following list of quantities.

In[35]:= **{ξ[-B] ϵ[A, B], ξ[B] ϵ[-B, -A], ϵ†[C†, A†] CDe[-A, -A†]@ξ[-B]}**

Out[35]= $\left\{\epsilon^{AB}\,\xi_B,\ \epsilon_{BA}\,\xi^B,\ \epsilon^{\dagger\,A\dagger}\,(\nabla_{AA\dagger}\,\xi_B)\right\}$

In[36]:= **ContractMetric /@ %**

Out[36]= $\left\{\xi^A,\ \xi_A,\ \nabla_A{}^{C\dagger}\,\xi_B\right\}$

The same goes for the remaining `xAct`` commands. For example `ToCanonical` takes into account the "see–saw" rule when canonicalising spinor expressions and adds the suitable signs. Also it works with any kind of spinor expression without any further option. To see this let us canonicalise the following list of spinor expressions.

In[37]:= **{ξ[A] ξ[-A], κ[-A, -B†] ξ†[B†], CDe[-F, -F†]@CDe[-A, F†]@PsiCDe[-B, -C, -D, -P] + CDe[-F, F†]@CDe[-A, -F†]@PsiCDe[-B, -C, -D, -P]}**

Out[37]= $\left\{\xi_A\,\xi^A,\ \kappa_{AB\dagger}\,\xi^{\dagger}{}^{B\dagger},\ \nabla_{FF\dagger}\,\nabla_A{}^{F\dagger}\,\Psi[\nabla]_{BCDP} + \nabla_F{}^{F\dagger}\,\nabla_{AF\dagger}\,\Psi[\nabla]_{BCDP}\right\}$

In[38]:= **ToCanonical /@ %**

Out[38]= $\left\{0,\ -\kappa_A{}^{B\dagger}\,\xi^{\dagger}_{B\dagger},\ 0\right\}$

©2003–2004 José M. Martín–García

A spin structure can be undefined and all its dependent symbols are thus removed from the session. To accomplish this we remove first all the visitors of the soldering form σ:

```
In[39]:= Undef /@ VisitorsOf[σ];

          ** UndefTensor: Undefined spinor SP`g†

          ** UndefTensor: Undefined spinor SP`g

          ** UndefTensor: Undefined spinor SP`epsilong†

          ** UndefTensor: Undefined spinor SP`epsilong

          ** UndefTensor: Undefined spinor SP`FRiemannCDe†

          ** UndefTensor: Undefined spinor SP`FRiemannCDe

          ** UndefTensor: Undefined spinor SP`RiemannCD†

          ** UndefTensor: Undefined spinor SP`RiemannCD

          ** UndefTensor: Undefined spinor SP`RicciCD†

          ** UndefTensor: Undefined spinor SP`RicciCD

          ** UndefTensor: Undefined spinor SP`WeylCD†

          ** UndefTensor: Undefined spinor SP`WeylCD

          ** UndefTensor: Undefined spinor SP`TFRicciCD†

          ** UndefTensor: Undefined spinor SP`TFRicciCD

          ** UndefTensor: Undefined tensor SP`κ †

          ** UndefTensor: Undefined tensor SP`κ

          ** UndefTensor: Undefined tensor SP`μ
```

and all the objects defined on the `Spin` vbundle :

```
In[40]:= VisitorsOf@Spin

Out[40]= {κ, κ †, μ, μ †, ξ}

In[41]:= Undef /@ {κ, ξ, μ};

          ** UndefTensor: Undefined tensor κ †

          ** UndefTensor: Undefined tensor κ

          ** UndefTensor: Undefined tensor ξ †

          ** UndefTensor: Undefined tensor ξ

          ** UndefTensor: Undefined tensor μ †

          ** UndefTensor: Undefined tensor μ
```

©2003–2004 José M. Martín–García

Finally the spin structure and all its dependent objects are removed.

```
In[42]:= UndefSpinStructure[σ]

          ** UndefTensor: Undefined nonsymmetric AChristoffel tensor AChristoffelCDe†

          ** UndefTensor: Undefined nonsymmetric AChristoffel tensor AChristoffelCDe

          ** UndefTensor: Undefined curvature spinor ChiCDe†

          ** UndefTensor: Undefined curvature spinor ChiCDe

          ** UndefTensor: Undefined FRiemann tensor FRiemannCDe†

          ** UndefTensor: Undefined FRiemann tensor FRiemannCDe

          ** UndefTensor: Undefined spinor scalar curvature LambdaCDe

          ** UndefTensor: Undefined curvature spinor PhiCDe†

          ** UndefTensor: Undefined curvature spinor PhiCDe

          ** UndefTensor: Undefined Weyl spinor PsiCDe†

          ** UndefTensor: Undefined Weyl spinor PsiCDe

          ** UndefCovD: Undefined covariant derivative CDe

          ** UndefTensor: Undefined tensor Sigmaσ †

          ** UndefTensor: Undefined tensor Sigmaσ

          ** UndefTensor: Undefined spin metric ϵ †

          ** UndefTensor: Undefined spin metric ϵ

          ** UndefTensor: Undefined soldering form σ †

          ** UndefTensor: Undefined soldering form σ

          ** UndefVBundle: Undefined conjugated vbundle Spin†

          ** UndefVBundle: Undefined vbundle Spin
```

Now, we are only left with spacetime quantities.

```
In[43]:= $Tensors

Out[43]= {g, epsilong, Tetrag, Tetrag†, TorsionCD, ChristoffelCD, RiemannCD, RicciCD,
          RicciScalarCD, EinsteinCD, WeylCD, TFRicciCD, KretschmannCD, Detg}
```

©2003–2004 José M. Martín–García

■ 2. Definition of a spin structure

In our present context we need the following ingredients to introduce a spin structure on a Lorentzian manifold: a 2–dimensional complex vector bundle (the spin bundle), the antisymmetric spin metric and the *soldering form.* This last quantity is a mixed object having indices in the spin bundle and the tangent bundle and serves to relate space–time quantities to spinor quantities and back (see e. g. appendix A of A. Ashtekar *Lectures on non–perturbative canonical gravity* World Scientific, Singapore, 1991). The explicit components of the soldering form in a basis are known as the *Infeld–Van der Waerden symbols.* It can be shown that there exists a unique covariant derivative (spin covariant derivative) which is compatible with both the spin metric and the soldering form. The restriction of the spin covariant derivative to tensors defined on the tangent bundle coincides with the Levi–Civita covariant derivative.

It is possible to introduce a spin structure both in a spacetime with or without torsion. We separate those two cases:

2.1. Spin structures on spacetimes without torsion

To define a spin structure we need first a four dimensional Lorentzian manifold

```
In[44]:= {$Manifolds, $Metrics, DimOfManifold /@ $Manifolds}

Out[44]= {{M4}, {g}, {4}}
```

DefSpinStructure	Define a spin structure on a four dimensional Lorentzian manifold.
UndefSpinStructure	Undefine a spin structure.
$SolderingForms	List of soldering forms (a.k.a. spin structures).

We define a spin structure comprising the complex vector bundle `Spin` whose abstract indices are `{A, B, C, D, F, H, L, P, Q}`. We need to supply as arguments the name of the metric tensor of the background Lorentzian manifold `g[-a,-b]`, the name of the antisymmetric spin metric `ϵ[-A,-B]`, the name of the soldering form `σ[a,-A,-A†]` and the name of the spin covariant derivative `CDe[-a]`. The option `SpinorPrefix` is used to introduce a tag which will be prepended automatically to any spinor related to a spacetime tensor and the option `SpinorMark` introduces a formatting of the tag `SP` different to the default one. The long output of `DefSpinStructure` can be suppressed by adding the option `Info->False` which deactivates the information of the definition commands .

```
In[45]:= DefSpinStructure[g, Spin, {A, B, C, D, F, H, L, P, Q},
           ϵ, σ, CDe, SpinorPrefix -> SP, SpinorMark -> "S", Info -> False]

      ValidateSymbol::capital :
       System name C is overloaded as an abstract index.

      ValidateSymbol::capital :
       System name D is overloaded as an abstract index.
```

We choose a formatting for the "primed" spinor indices according to the standard conventions used in the literature.

```
In[46]:= {PrintAs[A†] ^= "A'", PrintAs[B†] ^= "B'", PrintAs[C†] ^= "C'", PrintAs[D†] ^= "D'",
           PrintAs[F†] ^= "F'", PrintAs[H†] ^= "H'", PrintAs[L†] ^= "L'", PrintAs[M†] ^= "M'",
           PrintAs[P†] ^= "P'", PrintAs[Q†] ^= "Q'", PrintAs[ϵ ] ^= "ϵ"};
```

By convention we choose the symbol σ as the representative of all the elements which comprise the spin structure. Hence it will sometimes be referred to as "the spin structure". In particular this means that it is the host of the spinor

©2003–2004 José M. Martín–García

counterparts of the metric tensor, the volume element and all the curvature spinors associated to the covariant derivative `CDe` .

```
In[47]:= VisitorsOf@σ

Out[47]= {SP_g, SP_epsilong, SP_FRiemannCDe,
          SP_RiemannCD, SP_RicciCD, SP_WeylCD, SP_TFRicciCD}
```

The set of soldering forms is kept in the global variable `$SolderingForms`.

```
In[48]:= $SolderingForms

Out[48]= {σ}
```

All the essential algebraic properties of the soldering form are included as automatic rules. Example:

```
In[49]:= {σ[a, -A, -A†] σ[-a, -B, -B†], σ[-a, A, A†] σ[-b, -A, -A†]}

Out[49]= {ϵ_AB ϵ̄_A′B′, g_ba}
```

It is possible to define a second spin structure on the manifold `M4`.

```
In[50]:= DefSpinStructure[g, Spin2, {α, β, γ, δ, μ, ν}, ϵ2, σ2,
          CDe2, SpinorPrefix -> SP2, SpinorMark -> "S2", Info -> False]

In[51]:= $SolderingForms

Out[51]= {σ, σ2}
```

If we have more than one spin structure in the session then we need to specify which one are we working with in some of the `Spinors`` commands. Otherwise it is automatically assumed that we work with the default spin structure which is the first element of the list `$SolderingForms`.Examples:

```
In[52]:= RiemannCD[-a, -b, -c, -d]

Out[52]= R[∇]_abcd

In[53]:= PutSolderingForm[%, IndicesOf[], σ2]

Out[53]= R[∇]_abcd σ2^a_αα† σ2^b_ββ† σ2^c_γγ† σ2^d_δδ†

In[54]:= ContractSolderingForm@%

Out[54]= S2R[∇]_αα†ββ†γγ†

In[55]:= Decomposition[%]

Out[55]= Ψ[∇] ϵ†_α†β†δ† ϵ2_αβ ϵ2_δγ − Φ[∇]_αβδ†γ†δγ ϵ2_γ†α ϵ2_α†β†
         2 Λ[∇] (ϵ2_αδ ϵ2_βγ ϵ2†_α†δ† ϵ2†_β†γ† + ϵ2_αγ ϵ2_βδ ϵ2†_α†γ† ϵ2†_β†δ†) −
         Φ[∇]_δγα†β†αβ ϵ2_αβ ϵ2†_δ†γ† + Ψ[∇]_αβδγ ϵ2†_α†β† ϵ2†_δ†γ†

In[56]:= RiemannCD[-a, -b, -c, -d]

Out[56]= R[∇]_abcd
```

©2003–2004 José M. Martín–García

```
In[57]:= SeparateSolderingForm[σ2][%]

Out[57]= S2R[∇]αα† ββ† γγ† σ2^αα† σ2^ββ† σ2^γγ† σ2^δδ
```

```
In[58]:= PutSolderingForm[%, IndicesOf[], σ2]

Out[58]= S2R[∇]αα† ββ† γγ
```

We remove the spin structure $\sigma 2$. As usual, we start with the visitors of $\sigma 2$ and then we proceed to erase the spin structure itself.

```
In[59]:= Undef /@ VisitorsOf@σ2

** UndefTensor: Undefined SP2_g†

** UndefTensor: Undefined SP2_g

** UndefTensor: Undefined SP2_epsilong†

** UndefTensor: Undefined SP2_epsilong

** UndefTensor: Undefined SP2_FRiemannCDe2†

** UndefTensor: Undefined SP2_FRiemannCDe2

** UndefTensor: Undefined SP2_RiemannCD†

** UndefTensor: Undefined SP2_RiemannCD

** UndefTensor: Undefined SP2_RicciCD†

** UndefTensor: Undefined SP2_RicciCD

** UndefTensor: Undefined SP2_WeylCD†

** UndefTensor: Undefined SP2_WeylCD

** UndefTensor: Undefined SP2_TFRicciCD†

** UndefTensor: Undefined SP2_TFRicciCD
```

©2003–2004 José M. Martín–García

```
In[60]:= UndefSpinStructure@σ 2

** UndefTensor: Undefined AChristoffelCDe2†

** UndefTensor: Undefined AChristoffelCDe2

** UndefTensor: Undefined ChiCDe2†

** UndefTensor: Undefined ChiCDe2

** UndefTensor: Undefined FRiemannCDe2†

** UndefTensor: Undefined FRiemannCDe2

** UndefTensor: Undefined LambdaCDe2

** UndefTensor: Undefined PhiCDe2†

** UndefTensor: Undefined PhiCDe2

** UndefTensor: Undefined PsiCDe2†

** UndefTensor: Undefined PsiCDe2

** UndefCovD: Undefined CDe2

** UndefTensor: Undefined Sigmaσ 2†

** UndefTensor: Undefined Sigmaσ 2

** UndefTensor: Undefined ∈ 2†

** UndefTensor: Undefined ∈ 2

** UndefTensor: Undefined σ 2†

** UndefTensor: Undefined σ 2

** UndefVBundle: Undefined Spin2†

** UndefVBundle: Undefined Spin2
```

2.2. Spin structures on spacetimes with torsion

It is possible to define a spin structure on a spacetime in which the connection compatible with the metric tensor is assumed to have torsion. In this section we illustrate this.

We need to define a new 4−dimensional manifold and a new metric. Note the non−standard symmetry properties of the curvature tensors:

```
In[61]:= DefManifold[MT, 4, {α, β, γ, δ, μ, ν, ρ, λ}]

** DefManifold: Defining manifold MT.

** DefVBundle: Defining vbundle TangentMT.
```

©2003−2004 José M. Martín−García

```
In[62]:= DefMetric[{3, 1, 0}, G[-α, -β], cd, Torsion -> True]

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

** DefTensor: Defining antisymmetric tensor epsilonG[α, β, γ, δ].

** DefTensor: Defining tensor TetraG[-α, -β, -γ, -δ].

** DefTensor: Defining tensor TetraG†[-α, -β, -γ, -δ].

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

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

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

** DefTensor: Defining Riemann tensor
 Riemanncd[-α, -β, -γ, -δ]. Antisymmetric pairs cannot be exchanged.

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

** DefTensor: Defining Weyl tensor
 Weylcd[-α, -β, -γ, -δ]. Antisymmetric pairs cannot be exchanged.

** DefTensor: Defining non-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.
```

Definition of the spin structure

```
In[63]:= DefSpinStructure[G, SpinT, { , , , , , , , }, , Σ, cde, SpinorPrefix -> S]

** DefVBundle: Defining vbundle SpinT.

** DefVBundle: Defining conjugated vbundle SpinT†.
 Assuming fixed anti-isomorphism between SpinT and SpinT†

** DefTensor: Defining soldering form Σ[α, - , -  †].

** DefTensor: Defining soldering form Σ †[α, -  †, - ].

** DefTensor: Defining spin metric  [- , - ].

** DefTensor: Defining spin metric  †[-  †, -  †].
```

©2003–2004 José M. Martín–García

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

** DefTensor: Defining tensor SigmaΣ ↑[α, β, -  ↓ -  ↑].

** DefTensor: Defining spinor S⎵G[ ,  ↓ ,  ↑]. Equivalent of tensor G

** DefTensor: Defining spinor S⎵G†[  ↓ ,  ↓ ]. Equivalent of tensor G

** DefTensor: Defining spinor
 S⎵epsilonG[ ,  ↓ ,  ↓ ,  ↓ ,  ↑]. Equivalent of tensor epsilonG

** DefTensor: Defining spinor
 S⎵epsilonG†[  ↓ ,  ↓ ,  ↓ ,  ↓ ]. Equivalent of tensor epsilonG

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

** DefTensor: Defining
 nonsymmetric AChristoffel tensor AChristoffelcde[ , -β, - ].

** DefTensor: Defining
 nonsymmetric AChristoffel tensor AChristoffelcde†[  ↓ -β, -  ↑].

** DefTensor: Defining FRiemann tensor
 FRiemanncde[-α, -β, - , ]. Antisymmetric only in the first pair.

** DefTensor: Defining FRiemann tensor
 FRiemanncde†[-α, -β, -  ↓  ↑]. Antisymmetric only in the first pair.

** DefTensor: Defining spinor
 S⎵FRiemanncde[- , -  ↓ - , -  ↓ - , - ]. Equivalent of tensor FRiemanncde

** DefTensor: Defining spinor
 S⎵FRiemanncde†[-  ↓ - , -  ↓ - , -  ↓ -  ↑]. Equivalent of tensor FRiemanncde†

** DefTensor: Defining curvature spinor Chicde[- , - , - , - ].

** DefTensor: Defining curvature spinor Chicde†[-  ↓ -  ↓ -  ↓ -  ↑].

** DefTensor: Defining curvature spinor Phicde[- , - , -  ↓ -  ↑].

** DefTensor: Defining curvature spinor Phicde†[-  ↓ -  ↓ - , - ].

** DefTensor: Defining spinor
 S⎵Torsioncd[- , -  ↓ - , -  ↓ - , -  ↑]. Equivalent of tensor Torsioncd

** DefTensor: Defining spinor
 S⎵Torsioncd†[-  ↓ - , -  ↓ - , -  ↓ - ]. Equivalent of tensor Torsioncd

** DefTensor: Defining torsion spinor Omegacde[-  ↓ - , - , - ].

** DefTensor: Defining torsion spinor Omegacde†[- , -  ↓ -  ↓ -  ↑].

** DefTensor: Defining spinor S⎵Riemanncd[- , -  ↓ - , -  ↓ - , -  ↓ - , -  ↑
 . Equivalent of tensor Riemanncd

** DefTensor: Defining spinor S⎵Riemanncd†[-  ↓ - , -  ↓ - , -  ↓ - , -  ↓ - ]
 . Equivalent of tensor Riemanncd

** DefTensor: Defining spinor
 S⎵Riccicd[- , -  ↓ - , -  ↑]. Equivalent of tensor Riccicd

** DefTensor: Defining spinor
 S⎵Riccicd†[-  ↓ - , -  ↓ - ]. Equivalent of tensor Riccicd
```

©2003–2004 José M. Martín–García

This is the torsion tensor.

```
In[64]:= Torsioncd[α, -β, -γ]

Out[64]= T[∇]^α_βγ
```

We compute next the spinor equivalent of the torsion.

```
In[65]:= PutSolderingForm[%, IndexList[α, -β, -γ], Σ]

Out[65]= T[∇]^α_βγ Σ_α  †Σ^β  †Σ^γ
```

```
In[66]:= ContractSolderingForm@%

Out[66]= Σ T[∇]  †  †
```

This can be decomposed as follows.

```
In[67]:= Decomposition[%]

Out[67]=  †  †  (Ω[∇] †  †  †  † + Ω[∇]  †  †  †)†
```

```
In[68]:= ContractMetric@%

Out[68]= Ω[∇] †  †  †  + Ω[∇]  †  †  †
```

The spinor $\Omega[\nabla]^{\dagger}$ is called the "torsion spinor" of the covariant derivative. It is symmetric in the last pair of indices.

```
In[69]:= SymmetryGroupOfTensor@Omegacde

Out[69]= StrongGenSet[{3, 4}, GenSet[Cycles[{3, 4}]]]
```

That spinor appears in all standard expressions involving the torsion. Let us see an example:

```
In[70]:= DefSpinor[T[ ], MT]

** DefTensor: Defining tensor T[ ].

** DefTensor: Defining tensor T†[ †].
```

This is the spinor form of the Ricci identity associated to the covariant derivative `cde`. As is well known this spinor form contains the "box operator" whose expansion contains explicitly the torsion spinor.

```
In[71]:= Boxcde[- , - ]@T[ ]

Out[71]= □[∇]  T
```

```
In[72]:= BoxToCurvature[%, Boxcde]

Out[72]= -X[∇]  T - Ω[∇]  †  (∇  †T)
```

©2003–2004 José M. Martín–García

We remove all the symbols used in this subsection.

```
In[73]:= Undef /@ VisitorsOf@Σ;

          ** UndefTensor: Undefined spinor S_G†

          ** UndefTensor: Undefined spinor S_G

          ** UndefTensor: Undefined spinor S_epsilonG†

          ** UndefTensor: Undefined spinor S_epsilonG

          ** UndefTensor: Undefined spinor S_FRiemanncde†

          ** UndefTensor: Undefined spinor S_FRiemanncde

          ** UndefTensor: Undefined spinor S_Torsioncd†

          ** UndefTensor: Undefined spinor S_Torsioncd

          ** UndefTensor: Undefined spinor S_Riemanncd†

          ** UndefTensor: Undefined spinor S_Riemanncd

          ** UndefTensor: Undefined spinor S_Riccicd†

          ** UndefTensor: Undefined spinor S_Riccicd

In[74]:= Undef /@ VisitorsOf@SpinT;

          ** UndefTensor: Undefined tensor T†

          ** UndefTensor: Undefined tensor T
```

©2003–2004 José M. Martín–García

In[75]:= **UndefSpinStructure@Σ**

```
** UndefTensor: Undefined nonsymmetric AChristoffel tensor AChristoffelcde†

** UndefTensor: Undefined nonsymmetric AChristoffel tensor AChristoffelcde

** UndefTensor: Undefined curvature spinor Chicde†

** UndefTensor: Undefined curvature spinor Chicde

** UndefTensor: Undefined FRiemann tensor FRiemanncde†

** UndefTensor: Undefined FRiemann tensor FRiemanncde

** UndefTensor: Undefined torsion spinor Omegacde†

** UndefTensor: Undefined torsion spinor Omegacde

** UndefTensor: Undefined curvature spinor Phicde†

** UndefTensor: Undefined curvature spinor Phicde

** UndefCovD: Undefined covariant derivative cde

** UndefTensor: Undefined tensor SigmaΣ †

** UndefTensor: Undefined tensor SigmaΣ

** UndefTensor: Undefined spin metric   †

** UndefTensor: Undefined spin metric

** UndefTensor: Undefined soldering form Σ †

** UndefTensor: Undefined soldering form Σ

** UndefVBundle: Undefined conjugated vbundle SpinT†

** UndefVBundle: Undefined vbundle SpinT
```

©2003–2004 José M. Martín–García

```
In[76]:= UndefMetric@G

          ** UndefTensor: Undefined weight +2 density DetG

          ** UndefTensor: Undefined non-symmetric Christoffel tensor Christoffelcd

          ** UndefTensor: Undefined non-symmetric Einstein tensor Einsteincd

          ** UndefTensor: Undefined Kretschmann scalar Kretschmanncd

          ** UndefTensor: Undefined non-symmetric Ricci tensor Riccicd

          ** UndefTensor: Undefined Ricci scalar RicciScalarcd

          ** UndefTensor: Undefined Riemann tensor Riemanncd

          ** UndefTensor: Undefined non-symmetric TFRicci tensor TFRiccicd

          ** UndefTensor: Undefined torsion tensor Torsioncd

          ** UndefTensor: Undefined Weyl tensor Weylcd

          ** UndefCovD: Undefined covariant derivative cd

          ** UndefTensor: Undefined tensor TetraG†

          ** UndefTensor: Undefined tensor TetraG

          ** UndefTensor: Undefined antisymmetric tensor epsilonG

          ** UndefTensor: Undefined symmetric metric tensor G

In[77]:= UndefManifold@MT

          ** UndefVBundle: Undefined vbundle TangentMT

          ** UndefManifold: Undefined manifold MT

In[78]:= $Manifolds

Out[78]= {M4}
```

■ 3. Transformation between spinor and tensor expressions

Being able to transform spinor expressions into tensor ones and back is an important issue. Sometimes this is a trivial task but there are cases in which more efforts are required. `Spinors`` has a complete suite of commands which enable us to handle any case as we show below.

3.1. Insertion and removal of soldering forms in an expression

When transforming a spinor expression into a tensor one (or vice−versa) we may need to insert or eliminate a number of soldering forms. `Spinors`` provides commands to perform this task in an efficient fashion.

©2003–2004 José M. Martín–García

PutSolderingForm	Contraction of spinor/tensor indices with a suitable number of soldering forms
ContractSolderingForm	Transformation of contracted products of soldering forms into tensor/spinor expressions
SeparateSolderingForm	Extraction of spinor/tensor indices by means of soldering forms

Simple application: we start with the Riemann tensor and finish with its spinor counterpart

```
In[79]:= RiemannCD[-a, -b, -c, -d]

Out[79]= R[∇]abcd

In[80]:= PutSolderingForm@%

Out[80]= R[∇]abcd σ^a_AA, σ^b_BB, σ^c_CC, σ^d_DD,

In[81]:= ContractSolderingForm@%

Out[81]= SR[∇]AA, BB, CC, DD,
```

We can now follow previous procedure backwards and end up in the spacetime Riemann tensor.

```
In[82]:= SeparateSolderingForm[σ][%]

Out[82]= R[∇]abcd σ^a_AA, σ^b_BB, σ^c_CC, σ^d_DD,

In[83]:= PutSolderingForm@%

Out[83]= R[∇]abcd
```

Sometimes intermediate objects need to be created in a computation like the one just shown. This is done by the system in an automatic way.

```
In[84]:= DefTensor[M[-a, -b], M4]

          ** DefTensor: Defining tensor M[-a, -b].

In[85]:= PutSolderingForm@M[-a, -b]

Out[85]= Mab σ^a_AA, σ^b_BB,
```

Here, the contraction of the soldering forms results in the spinor counterpart of the tensor `M[-a, -b]`. The former has not been defined previously and thus the system must introduce this new object (a message informing about this is issued). The definition of spinor counterparts of tensor objects can be performed by the user independently (see next section).

```
In[86]:= ContractSolderingForm@%

      SpinorOfTensor::name : Spinor of M not defined. Prepending SP.

          ** DefTensor: Defining spinor SP__M[-A, -A†, -B, -B†]. Equivalent of tensor M

          ** DefTensor: Defining spinor SP__M†[-A†, -A, -B†, -B]. Equivalent of tensor M

Out[86]= SMAA, BB,
```

©2003–2004 José M. Martín–García

The separation of the soldering form gets us back to the original tensor M_{ab}.

```
In[87]:= SeparateSolderingForm[]@%
```

$$\text{Out[87]= } M_{ab}\, \sigma^{a}{}_{AA'}\, \sigma^{b}{}_{BB'}$$

```
In[88]:= PutSolderingForm@%
```

$$\text{Out[88]= } M_{ab}$$

The spinors SP$_$M and SP$_$M† are visitors of M. Therefore if we wish to remove M, we need to remove first its visitors.

```
In[89]:= Undef /@ VisitorsOf@M;

          ** UndefTensor: Undefined spinor SP$_$M†

          ** UndefTensor: Undefined spinor SP$_$M
```

```
In[90]:= UndefTensor@M

          ** UndefTensor: Undefined tensor M
```

There are cases in which we need to tell explicitly how the indices of the soldering forms should be contracted when they act on an expression. This is achieved by means of extra options in the commands just ilustrated.

Consider an expression containing spacetime and spinor indices.

```
In[91]:= expr = CDe[-A, -A†]@FRiemannCDe[-a, -b, -C, -D]
```

$$\text{Out[91]= } \nabla_{AA'}\, \text{FCDe}_{abCD}$$

By default SeparateSolderingForm will extract soldering forms from all the tensor indices and all the complex conjugated pairs of spinor indices.

```
In[92]:= SeparateSolderingForm[σ][expr]
```

$$\text{Out[92]= } \sigma_{a}{}^{BB'}\, \sigma_{b}{}^{FC'}\, \sigma^{c}{}_{AA'}\, (\nabla_{c}\, \text{SFCDe}_{BB'\, FC'\, CD})$$

We may control with extra options the vector bundles whose indices will be pulled out by SeparateSolderingForm.

```
In[93]:= SeparateSolderingForm[σ][expr, Spin]
```

$$\text{Out[93]= } \sigma^{c}{}_{AA'}\, \left(\nabla_{c}\, \text{FCDe}_{abCD}\right)$$

```
In[94]:= SeparateSolderingForm[σ][expr, TangentM4]
```

$$\text{Out[94]= } \sigma_{a}{}^{BB'}\, \sigma_{b}{}^{FC'}\, (\nabla_{AA'}\, \text{SFCDe}_{BB'\, FC'\, CD})$$

©2003–2004 José M. Martín–García

A similar index–selection is also available for PutSolderingForm.

```
In[95]:= PutSolderingForm[expr, Spin]

Out[95]= σ_c^AA' (∇_AA, FCDe_abCD)

In[96]:= PutSolderingForm[expr, TangentM4]

Out[96]= σ^a_BB, σ^b_FF, (∇_AA, FCDe_abCD)
```

The following expression is the spinor form of a tensor (Bel–Robinson tensor). If we wish to use PutSolderingForm to find its tensor equivalent we need to pass an argument stating how the spinor indices are grouped into tensor ones as there is not a unique way of doing it. This is achieved by means of the head Pair which represents conjugated pairs of spinor indices.

```
In[97]:= PsiCDe[-A, -B, -C, -D] Dagger[PsiCDe[-A, -B, -C, -D]]

Out[97]= Ψ[∇]_ABCD Ψ[∇] †_A', B', C', D',

In[98]:= PutSolderingForm[%, {Pair[-A, -A†], Pair[-B, -B†], Pair[-C, -C†], Pair[-D, -D†]}]

Out[98]= Ψ[∇]_ABCD Ψ[∇] †_A', B', C', D', σ_a^AA' σ_b^BB' σ_c^CC' σ_d^DD'
```

3.2. Relations between spinors and tensors

There are spinors which can be put into relation with tensors and tensors which can be put into relation with spinors. In both cases the relation is carried out by means of the soldering form. Whenever one of these relations apply, Spinors` enables the user to define the spinor (resp. tensor) counterpart of a tensor (resp. spinor).

DefSpinorOfTensor	Defines the spinor counterpart of a tensor (when applicable)
DefTensorOfSpinor	Defines the tensor counterpart of a spinor (when applicable)

Let us define a spinor.

```
In[99]:= DefSpinor[κ[-A, -A†], M4]

** DefTensor: Defining tensor κ[-A, -A†].

** DefTensor: Defining tensor κ ǂ[-A†, -A].
```

This spinor has a tensor counterpart. To introduce it into the session we use the command DefTensorOfSpinor in the following way:

```
In[100]:=
DefTensorOfSpinor[KT[a], κ[A, A†], σ]

** DefTensor: Defining tensor KT[a]. Equivalent of spinor κ

** DefTensor: Defining tensor KT†[a]. Equivalent of spinor κ †
```

©2003–2004 José M. Martín–García

The output form of the tensor is actually constructed from that of the spinor:

In[101]:=
```
KT[a]
```

Out[101]=
$S\kappa^{a}$

The symbol `KT` represents the tensor counterpart of the spinor κ. This must be a symbol which does not correspond to any object defined previously in the session. From this point on, the system stores `KT` as the tensor counterpart of κ. Any of the commands which relate spinors to tensors are now aware of this binding between κ and `KT`. Let us see some examples:

In[102]:=
```
TensorOfSpinor[κ, σ]
```

Out[102]=
KT

We start from the spinor $\kappa_{AA'}$ transform it into a tensor and then back to the same spinor all by using `PutSolderingForm`/`ContractSolderingForm`.

In[103]:=
```
κ[-A, -A†]
```

Out[103]=
$\kappa_{AA'}$

In[104]:=
```
PutSolderingForm[%]
```

Out[104]=
$\kappa_{AA'}\, \sigma_{a}^{\ AA'}$

In[105]:=
```
ContractSolderingForm[%]
```

Out[105]=
$S\kappa_{a}$

In[106]:=
```
PutSolderingForm[%]
```

Out[106]=
$S\kappa_{a}\, \sigma^{a}_{\ AA'}$

In[107]:=
```
ContractSolderingForm[%]
```

Out[107]=
$\kappa_{AA'}$

©2003–2004 José M. Martín–García

Similarly we can introduce the spinor counterpart of any tensor already defined in the sesssion. For example:

```
In[108]:=
DefTensor[T[-a, -b], M4, Dagger → Complex]

    ** DefTensor: Defining tensor T[-a, -b].

    ** DefTensor: Defining tensor T†[-a, -b].

In[109]:=
DefSpinorOfTensor[ST[-A, -A†, -B, -B†], T[-a, -b], σ]

    ** DefTensor: Defining spinor ST[-A, -A†, -B, -B†]. Equivalent of tensor T

    ** DefTensor: Defining spinor ST†[-A†, -A, -B†, -B]. Equivalent of tensor T†
```

As above the symbol `ST` should not correspond to any of the symbols defined previously in the session. Note carefully the index arrangement in the spinor `ST`. This is fixed by convention and cannot be altered by the user. We will come back to this important point below. For now we present some manipulation examples.

```
In[110]:=
T[-a, -b] // PutSolderingForm

Out[110]=
T_ab σ^a_AA' σ^b_BB'

In[111]:=
ContractSolderingForm[%]

Out[111]=
ST_AA' BB'

In[112]:=
T†[-a, -b] // PutSolderingForm

Out[112]=
T†_ab σ^a_AA' σ^b_BB'

In[113]:=
ContractSolderingForm[%]

Out[113]=
ST†_A' AB' B

In[114]:=
PutSolderingForm[%]

Out[114]=
ST†_A' AB' B σ_a^AA' σ_b^BB'

In[115]:=
ContractSolderingForm[%]

Out[115]=
T†_ab
```

The above examples can be generalized to tensors and spinors with a higher number of indices with the following provision: in the commands `DefSpinorOfTensor` and `DefTensorOfSpinor` spinor indices must be always arranged

©2003–2004 José M. Martín–García

according to the pattern index–Dagger[index]. Otherwise an error message is thrown.

```
In[116]:=
DefSpinorOfTensor[TST[-A†, -A, -B†, -B], T[-a, -b], σ]

DefSpinorOfTensor::incomp :
 Indices {-A†, -A, -B†, -B} and {-a, -b} are incompatible.

In[117]:=
DefSpinor[Φ[-A, -B, -A†, -B†], M4]

   ** DefTensor: Defining tensor Φ[-A, -B, -A†, -B†].

   ** DefTensor: Defining tensor Φ†[-A†, -B†, -A, -B].

In[118]:=
DefTensorOfSpinor[Φ T[-a, -b], Φ[-A, -B, -A†, -B†], σ]

DefTensorOfSpinor::incomp :
 Indices {-a, -b} and {-A, -B, -A†, -B†} are incompatible.
```

We undefine the objects used in this subsection. Note that whenever we wish to undefine a spinor (tensor) quantity with a tensor (spinor) counterpart we must erase first the latter, which is a visitor of the former.

```
In[119]:=
UndefSpinor[Φ]

   ** UndefTensor: Undefined tensor Φ†

   ** UndefTensor: Undefined tensor Φ

In[120]:=
Undef /@ VisitorsOf@T;

   ** UndefTensor: Undefined spinor ST†

   ** UndefTensor: Undefined spinor ST

In[121]:=
Undef[T]

   ** UndefTensor: Undefined tensor T†

   ** UndefTensor: Undefined tensor T

In[122]:=
Undef /@ VisitorsOf@κ;

   ** UndefTensor: Undefined tensor KT†

   ** UndefTensor: Undefined tensor KT

In[123]:=
Undef[κ]

   ** UndefTensor: Undefined tensor κ†

   ** UndefTensor: Undefined tensor κ
```

©2003–2004 José M. Martín–García

3.3. Products of soldering forms

Contracted products of soldering forms appear in a natural way when transforming spinor expressions into tensor ones and back. Therefore it is important to be able to work with such products. In this subsection we show how `Spinors`` deals with such quantities.

The simplest case is the product of 2 soldering forms which gives rise to an auxiliary quantity which is needed when going from spinor expressions to tensor expressions.

In[124]:=

```
σ[a, -A, -A†] σ[b, -B, A†] // ContractSolderingForm
```

Out[124]=

$\Sigma\sigma^{ab}{}_{AB}$

The algebraic properties of $\Sigma\sigma^{ab}{}_{AB}$ are stored as automatic rules.

In[125]:=

```
Sigma[σ]
```

Out[125]=

Sigmaσ

In[126]:=

```
? Sigmaσ
```

```
Global`Sigmaσ

Dagger[Sigmaσ] ^:= Sigmaσ †

DependenciesOfTensor[Sigmaσ] ^:= {M4}

HostsOf[Sigmaσ] ^:= {M4, TangentM4}

Info[Sigmaσ] ^:= False

MasterOf[Sigmaσ] ^:= σ

PrintAs[Sigmaσ] ^:= Σσ

ServantsOf[Sigmaσ] ^:= {Sigmaσ †}

SlotsOfTensor[Sigmaσ] ^:= {TangentM4, TangentM4, -Spin, -Spin}

SymmetryGroupOfTensor[Sigmaσ] ^:=
 StrongGenSet[{1}, GenSet[-Cycles[{1, 2}, {3, 4}]]]

TensorID[Sigmaσ] ^:= {}

xTensorQ[Sigmaσ] ^:= True
```

$\Sigma\sigma^{\underline{a}\,\underline{b}\,\underline{A}\,\underline{B}}\,\Sigma\sigma^{\underline{c}\,\underline{d}\,\underline{C}}{}_{\underline{B}}$ ^:= Module$\left[\{s\$2724\},\ \Sigma\sigma^{s\$2724cAC}\,Gg^{bd}{}_{s\$2724}{}^{a}\right]$

©2003–2004 José M. Martín–García

```
Σσ[-d,-c,-B,-C] Σσ[-a,-b,-A,-B] ^:= Module[{s$2724}, -Σσ[s$2724cAC] Gg[bd, s$2724, a]]

Σσ[-b,-a,-B,-A] Σσ[-c,-d,-C,-B] ^:= Module[{s$2724}, -Σσ[s$2724cAC] Gg[bd, s$2724, a]]

Σσ[-d,-c,-B,-C] Σσ[-b,-a,-B,-A] ^:= Module[{s$2724}, Σσ[s$2724cAC] Gg[bd, s$2724, a]]

Sigmaσ /: Σσ[-f,-h,-C,-D] Gg[-a,-b,-c,-d] :=
  -g[ab] ϵ[CD] /; (PairQ[c, f] && PairQ[d, h]) || (PairQ[c, h] && PairQ[d, f])

Sigmaσ /: Σσ[-f,-h,-C,-D] Gg[-c,-d,-a,-b] :=
  -g[ab] ϵ[CD] /; (PairQ[c, f] && PairQ[d, h]) || (PairQ[c, h] && PairQ[d, f])

Σσ[xAct`Spinors`Private`a$, xAct`Spinors`Private`b$, C$, -C$] := -g[xAct`Spinors`Private`a$, xAct`Spinors`Private`b$]

Σσ[xAct`Spinors`Private`a$, xAct`Spinors`Private`b$, -C$, C$] := g[xAct`Spinors`Private`a$, xAct`Spinors`Private`b$]

Σσ[xAct`Spinors`Private`a$, xAct`Spinors`Private`a$, -C$, -D$] := -2 ϵ[C$D$]

Σσ[xAct`Spinors`Private`a$, xAct`Spinors`Private`a$, C$, D$] := -2 ϵ[C$D$]
```

One of the previous properties is that the "square" of $\Sigma\sigma^{ab}{}_{AB}$ gives rise to the "tetra–metric". The latter is a four rank spacetime tensor which is defined in eq. (3.4.57) of vol. 1 of Penrose & Rindler (note that they use the name `U` for it).

In[127]:=

```
Sigmaσ[-a, -b, -A, -B] Sigmaσ[-c, -d, A, B] // InputForm
```

Out[127]//InputForm=

```
-Tetrag[-d, -b, -a, -c]
```

This is the explicit definition of the tetra–metric (note that there is a sign difference between formula (3.4.57) of Vol. 1of Penrose & Rindler and our formula).

In[128]:=

```
Tetrag[-a, -b, -c, -d] /. TetraRule[g]
```

Out[128]=

$$\frac{i\,\epsilon_{abcd}}{2} - \frac{1}{2}\,g_{ad}\,g_{bc} + \frac{1}{2}\,g_{ac}\,g_{bd} - \frac{1}{2}\,g_{ab}\,g_{cd}$$

©2003–2004 José M. Martín–García

As we can see the tetra–metric only depends on spacetime quantities. Therefore the tetra–metric is always automatically defined whenever a Lorentzian four dimensional metric is introduced regardless to whether a spin structure has been defined. Also automatic rules implementing the algebraic properties of the tetra–metric are stored.

In[129]:=
```
? Tetrag
```

```
Global`Tetrag

Dagger[Tetrag] ^= Tetrag†

DependenciesOfTensor[Tetrag] ^= {M4}

HostsOf[Tetrag] ^= {M4, TangentM4}

Info[Tetrag] ^= {tensor, }

MasterOf[Tetrag] ^= g

PrintAs[Tetrag] ^= Gg

ServantsOf[Tetrag] ^= {Tetrag†}

SlotsOfTensor[Tetrag] ^= {-TangentM4, -TangentM4, -TangentM4, -TangentM4}

SymmetryGroupOfTensor[Tetrag] ^=
 StrongGenSet[{1}, GenSet[Cycles[{1, 2}, {3, 4}], Cycles[{1, 3}, {2, 4}]]]

TensorID[Tetrag] ^= {}

xTensorQ[Tetrag] ^= True
```

A typical problem is the computation of a product of soldering forms with no free spinor indices. These always give rise to contracted products of tetra metrics. Let us see some examples.

In[130]:=
```
σ[a, -A, -Bt] σ[b, -C, Bt] σ[c, C, -Dt] σ[d, A, Dt]
```

Out[130]=

$$\sigma^{a}{}_{AB'}\,\sigma^{b}{}_{C}{}^{B'}\,\sigma^{cC}{}_{D}\,\sigma^{dAD'}$$

In[131]:=
```
% // ContractSolderingForm // ContractMetric
```

Out[131]=

$$Gg^{dabc}$$

In[132]:=
```
σ[a, -A, -Bt] σ[b, -C, Bt] σ[c, C, -Dt] σ[d, -P, Dt] σ[f, P, -Qt] σ[p, A, Qt]
```

Out[132]=

$$\sigma^{a}{}_{AB'}\,\sigma^{b}{}_{C}{}^{B'}\,\sigma^{cC}{}_{D}\,\sigma^{dD'}{}_{P}\,\sigma^{fP}{}_{Q}\,\sigma^{pAQ'}$$

©2003–2004 José M. Martín–García

In[133]:=
```
% // ContractSolderingForm // ContractMetric
```

Out[133]=
$$Gg^{bc\ a}_{\ \ h}\ Gg^{phdf}$$

Expressions of the sort just found are frequent in computations involving the transformation of spinor expressions into tensor ones. Let us see an example

---

Transformation of the Weyl tensor into its spinor counterpart

In[134]:=
```
weyl = WeylCD[-a, -b, -c, -d]
```

Out[134]=
$$W[\nabla]_{abcd}$$

In[135]:=
```
SeparateSolderingForm[][%]
```

Out[135]=
$$SW[\nabla]_{AA',BB',CC',DD'}\ \sigma_a^{\ AA'}\ \sigma_b^{\ BB'}\ \sigma_c^{\ CC'}\ \sigma_d^{\ DD'}$$

In[136]:=
```
Decomposition[%, Weyl]
```

Out[136]=
$$\left(\Psi[\nabla]\ \dagger_{A',B',D',C'}\ \epsilon_{AB}\ \epsilon_{DC} + \Psi[\nabla]_{ABDC}\ \overline{\epsilon}_{A',B'}\ \overline{\epsilon}_{D',C'}\right)\ \sigma_a^{\ AA'}\ \sigma_b^{\ BB'}\ \sigma_c^{\ CC'}\ \sigma_d^{\ DD'}$$

This spinor expression can be transformed back into the Weyl tensor

In[137]:=
```
weyl == ToCanonical[ContractMetric[%]]
```

Out[137]=
$$W[\nabla]_{abcd} == -\Psi[\nabla]\ \dagger_{A',B',C',D'}\ \sigma_a^{\ AA'}\ \sigma_{bA}^{\ \ B'}\ \sigma_c^{\ BC'}\ \sigma_{dB}^{\ \ D'} - \Psi[\nabla]_{ABCD}\ \sigma_a^{\ AA'}\ \sigma_b^{\ B}{}_{A'}\ \sigma_c^{\ CB'}\ \sigma_d^{\ D}{}_{B'}$$

We use this expression to write the Weyl spinor $\Psi[\nabla]_{ABCD}$ in terms of the Weyl tensor $W[\nabla]_{abcd}$.

In[138]:=
```
Expand /@ PutSolderingForm /@ %
```

Out[138]=
$$W[\nabla]_{abcd}\ \sigma^a_{\ AA'}\ \sigma^b_{\ BB'}\ \sigma^c_{\ CC'}\ \sigma^d_{\ DD'} == -\Psi[\nabla]\ \dagger_{A',B',C',D'}\ \epsilon_{BA}\ \epsilon_{DC} - \Psi[\nabla]_{ABCD}\ \overline{\epsilon}_{B',A'}\ \overline{\epsilon}_{D',C'}$$

In[139]:=
```
% /. -B† -> A† /. -D† -> C†
```

Out[139]=
$$W[\nabla]_{abcd}\ \sigma^a_{\ AA'}\ \sigma^b_{\ B}{}^{A'}\ \sigma^c_{\ CC'}\ \sigma^d_{\ D}{}^{C'} == -4\ \Psi[\nabla]_{ABCD} - \Psi[\nabla]\ \dagger_{A'}{}^{A'}{}_{C'}{}^{C'}\ \epsilon_{BA}\ \epsilon_{DC}$$

In[140]:=
```
ToCanonical /@ %
```

Out[140]=
$$W[\nabla]_{abcd}\ \sigma^a_{\ A}{}^{A'}\ \sigma^b_{\ BA'}\ \sigma^c_{\ C}{}^{C'}\ \sigma^d_{\ DC'} == -4\ \Psi[\nabla]_{ABCD}$$

©2003–2004 José M. Martín–García

In[141]:=
```
WeylSpinorToWeylTensor = IndexSolve[%, Ψ[∇]ABCD]
```

Out[141]=
$$\left\{\text{HoldPattern}\left[\Psi[\nabla]^{\underline{A}\,\underline{B}\,\underline{C}\,\underline{D}}\right] :\mapsto \text{Module}\left[\{a, A\dagger, b, c, B\dagger, d\}, -\frac{1}{4}\, W[\nabla]_{abcd}\, \sigma^{aAA'}\, \sigma^{bB}{}_{A'}\, \sigma^{cCB'}\, \sigma^{dD}{}_{B'}\right]\right\}$$

Now we can use previous rule to find the tensor expression of any invariant written in terms of the Weyl spinor. For example consider

In[142]:=
```
expr = PsiCDe[-A, -B, -C, -D] PsiCDe[A, B, C, D] /. WeylSpinorToWeylTensor
```

Out[142]=
$$\frac{1}{16}\, W[\nabla]_{acfj}\, W[\nabla]_{bdhl}\, \sigma^{a}{}_{A}{}^{A'}\, \sigma^{bAB'}\, \sigma^{c}{}_{BA'}\, \sigma^{dB}{}_{B'}\, \sigma^{f}{}_{C}{}^{C'}\, \sigma^{hCD'}\, \sigma^{j}{}_{DC'}\, \sigma^{lD}{}_{D'}$$

A direct use of `ContractSolderingForm` would contract the soldering forms with the Weyl tensor, yielding an undesired result:

In[143]:=
```
ContractSolderingForm@%
```

Out[143]=
$$\frac{1}{16}\, SW[\nabla]_{A}{}^{A'}{}_{BA'}{}^{C'}{}_{C'DC'}\, SW[\nabla]^{AB'}{}_{B}{}^{B}{}_{B'}{}^{CD'}{}_{D}{}^{D}{}_{D'}$$

Instead we specify contraction of the indices on the soldering form which are dummies, generating a product of "tetra–metrics".

In[144]:=
```
ContractSolderingForm[expr, IndicesOf[Dummy][expr]]
```

Out[144]=
$$\frac{1}{16}\, g^{af}\, g^{bm}\, Gg^{dc}{}_{a}{}^{h}\, Gg^{lj}{}_{b}{}^{n}\, W[\nabla]_{cfjm}\, W[\nabla]_{dhln}$$

In[145]:=
```
ToCanonical@ContractMetric@%
```

Out[145]=
$$\frac{1}{16}\, Gg^{abcd}\, Gg^{fhjl}\, W[\nabla]_{adfl}\, W[\nabla]_{bchj}$$

Finally we expand the tetra–metrics into ordinary metrics and volume elements. The simplification of the resulting expression provides the final answer.

In[146]:=
```
% /. TetraRule[g]
```

Out[146]=
$$\frac{1}{16}\left(\frac{i \in g^{abcd}}{2} - \frac{1}{2}\, g^{ad}\, g^{bc} + \frac{1}{2}\, g^{ac}\, g^{bd} - \frac{1}{2}\, g^{ab}\, g^{cd}\right)$$
$$\left(\frac{i \in g^{fhjl}}{2} - \frac{1}{2}\, g^{fl}\, g^{hj} + \frac{1}{2}\, g^{fj}\, g^{hl} - \frac{1}{2}\, g^{fh}\, g^{jl}\right) W[\nabla]_{adfl}\, W[\nabla]_{bchj}$$

©2003–2004 José M. Martín–García

```
In[147]:=
ToCanonical@ContractMetric@%
```

```
Out[147]=
1/8 W[∇]abcd W[∇]abcd - 1/16 i ϵ gcdfh W[∇]ab fh W[∇]abcd
```

Clean up :

```
In[148]:=
Remove[expr]
```

■ 4. Decomposition of a spinor into irreducible parts

A classical result of the spinor algebra is the posibility of decomposing any spinor into a sum which consists of a totally symmetric spinor of the same rank as the given spinor plus other terms which are outer products of the antisymmetric metric with totally symmetric spinors of lower rank (see proposition 3.3.54 of Penrose & Rindler vol. I for full details). In `Spinors`` there is an automated procedure to find such a decomposition as we explain below.

`IrreducibleDecomposition`	Finds the decomposition of any spinor into irreducible parts

We show how `IrreducibleDecomposition` works by means of some simple examples. Let us start by defining a spinor

```
In[149]:=
DefSpinor[Kh[-A, -B, -C, -D], M4,
 GenSet[Cycles[{-A, -B}], Cycles[{-C, -D}]], PrintAs → "K"]

   ** DefTensor: Defining tensor Kh[-A, -B, -C, -D].

   ** DefTensor: Defining tensor Kh†[-A†, -B†, -C†, -D†].
```

The decomposition of the spinor just introduced into irreducible parts is given by

```
In[150]:=
exprA = IrreducibleDecomposition[Kh[-A, B, -C, -D]]

   ** DefTensor: Defining tensor
    TFKh[LI[TF], AnyIndices[Spin], AnyIndices[Spin†]].

   ** DefTensor: Defining tensor
    TFKh†[LI[TF], AnyIndices[Spin†], AnyIndices[Spin]].
```

```
Out[150]=
δA B TFKh{{1, 2}} CD + δB C TFKh{{2, 3}} AD + δB D TFKh{{2, 4}} AC + TFKh{} A B CD +
 δB D TFKh{{1, 3}, {2, 4}} ϵAC + TFKh{{1, 3}} D B ϵAC + δB C TFKh{{1, 4}, {2, 3}} ϵAD +
 TFKh{{1, 4}} C B ϵAD + δA B TFKh{{1, 2}, {3, 4}} ϵCD + TFKh{{3, 4}} A B ϵCD
```

The command `IrreducibleDecomposition` defines first the totally symmetric spinors intervening in the decompo sition (irreducible spinors). The actual decomposition is found in a second step and displayed in the output. In order to represent the irreducible spinors, we introduce an indexed quantity with a variable number of slots (see the documenta–tion of the package `xTensor`` for further details about this). Note the labels on each of the totally symmetric spinors: these are label indices which are used to distinguish each of the irreducible spinors. The pairs of numbers which appear inside the list indicate which slots of the tensor being decomposed have been picked to "pull out" factors of the antisym–metric metric.

©2003–2004 José M. Martín–García

This is the list of all the irreducible spinors resulting from the decomposition of `Kh[-A, -B, -C, -D]`.

In[151]:=
```
Cases[exprA, _TFKh, Infinity]
```

Out[151]=
$$\left\{\text{TFKh}^{\{\{1,\,2\}\}}_{CD},\ \text{TFKh}^{\{\{2,\,3\}\}}_{AD},\ \text{TFKh}^{\{\{2,\,4\}\}}_{AC},\ \text{TFKh}^{\{\}\,B}_{A\,CD},\ \text{TFKh}^{\{\{1,\,3\},\,\{2,\,4\}\}},\right.$$
$$\left.\text{TFKh}^{\{\{1,\,3\}\}\,B}_{D},\ \text{TFKh}^{\{\{1,\,4\},\,\{2,\,3\}\}},\ \text{TFKh}^{\{\{1,\,4\}\}\,B}_{C},\ \text{TFKh}^{\{\{1,\,2\},\,\{3,\,4\}\}},\ \text{TFKh}^{\{\{3,\,4\}\}\,B}_{A}\right\}$$

The symbol `TF` (trace free) is prepended to the symbol `Kh`. All the quantities of the previous list which are not scalars are totally symmetric.

In[152]:=
```
SymmetryGroupOfTensor /@ %
```

Out[152]=
```
{StrongGenSet[{2, 3}, GenSet[Cycles[{2, 3}]]],
 StrongGenSet[{2, 3}, GenSet[Cycles[{2, 3}]]],
 StrongGenSet[{2, 3}, GenSet[Cycles[{2, 3}]]],
 StrongGenSet[{2, 3, 4, 5}, GenSet[Cycles[{2, 3}], Cycles[{3, 4}], Cycles[{4, 5}]]],
 StrongGenSet[{}, GenSet[]], StrongGenSet[{2, 3}, GenSet[Cycles[{2, 3}]]],
 StrongGenSet[{}, GenSet[]], StrongGenSet[{2, 3}, GenSet[Cycles[{2, 3}]]],
 StrongGenSet[{}, GenSet[]], StrongGenSet[{2, 3}, GenSet[Cycles[{2, 3}]]]}
```

The irreducible spinors are servants of the spinor they decompose. Therefore they are removed if the latter is erased.

In[153]:=
```
UndefSpinor@Kh
```

```
** UndefTensor: Undefined tensor TFKh†

** UndefTensor: Undefined tensor TFKh

** UndefTensor: Undefined tensor Kh†

** UndefTensor: Undefined tensor Kh
```

Example of the decomposition of a mixed spinor.

In[154]:=
```
IrreducibleDecomposition[SP_RicciCD[A, -A†, -B, -B†]]
```

```
** DefTensor: Defining tensor
 TFSP_RicciCD[LI[TF], AnyIndices[Spin], AnyIndices[Spin†]].

** DefTensor: Defining tensor
 TFSP_RicciCD†[LI[TF], AnyIndices[Spin†], AnyIndices[Spin]].
```

Out[154]=
$$\delta^{A}_{B}\ \text{TFSP\_RicciCD}^{\{\{1,\,3\}\}}_{A',B'} + \text{TFSP\_RicciCD}^{\{\}\,A}_{A',BB'} +$$
$$\delta^{A}_{B}\ \text{TFSP\_RicciCD}^{\{\{1,\,3\},\,\{2,\,4\}\}}\ \overline{\epsilon}_{A',B'} + \text{TFSP\_RicciCD}^{\{\{2,\,4\}\}\,A}_{B}\ \overline{\epsilon}_{A',B'}$$

©2003–2004 José M. Martín–García

■ 5. Spin–compatible covariant derivatives

`Spinors`` includes the concept of spin–compatible covariant derivative. A covariant derivative is spin–compatible if it yields zero when acting upon a soldering form. The spin covariant derivative which extends the Levi–Civita covariant derivative is an example of it.

DefSpinCovD	Defines a spin–compatible covariant derivative
UndefSpinCovD	Undefines a spin–compatible covariant derivative

We define a covariant derivative which is spin–compatible with the soldering form σ. Note that the syntax of `DefSpinCovD` is similar to the `xAct`` command `DefCovD` used to define covariant derivatives. We use non–standard symbols to denote the derivative:

In[155]:=

```
DefSpinCovD[nb[-a], σ, SymbolOfCovD -> {"|", "D"}, Torsion -> True]
```

```
** DefCovD: Defining covariant derivative nb[-a].

** DefTensor: Defining torsion tensor Torsionnb[a, -b, -c].

** DefTensor: Defining
 non-symmetric Christoffel tensor Christoffelnb[a, -b, -c].

** DefTensor: Defining Riemann tensor
 Riemannnb[-a, -b, -c, d]. Antisymmetric only in the first pair.

** DefTensor: Defining non-symmetric Ricci tensor Riccinb[-a, -b].

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

** DefTensor: Defining
 nonsymmetric AChristoffel tensor AChristoffelnb[A, -b, -C].

** DefTensor: Defining
 nonsymmetric AChristoffel tensor AChristoffelnb†[A†, -b, -C†].

** DefTensor: Defining FRiemann tensor
 FRiemannnb[-a, -b, -C, D]. Antisymmetric only in the first pair.

** DefTensor: Defining FRiemann tensor
 FRiemannnb†[-a, -b, -C†, D†]. Antisymmetric only in the first pair.

** DefTensor: Defining spinor
 SP␣FRiemannnb[-A, -A†, -B, -B†, -C, -D]. Equivalent of tensor FRiemannnb

** DefTensor: Defining spinor
 SP␣FRiemannnb†[-A†, -A, -B†, -B, -C†, -D†]. Equivalent of tensor FRiemannnb†

** DefTensor: Defining curvature spinor Chinb[-A, -B, -C, -D].

** DefTensor: Defining curvature spinor Chinb†[-A†, -B†, -C†, -D†].

** DefTensor: Defining curvature spinor Phinb[-A, -B, -C†, -D†].

** DefTensor: Defining curvature spinor Phinb†[-A†, -B†, -C, -D].

** DefTensor: Defining spinor
 SP␣Torsionnb[-A, -A†, -B, -B†, -C, -C†]. Equivalent of tensor Torsionnb
```

©2003–2004 José M. Martín–García

```
** DefTensor: Defining spinor
 SP␣Torsionnb†[-A†, -A, -B†, -B, -C†, -C]. Equivalent of tensor Torsionnb

** DefTensor: Defining torsion spinor Omeganb[-A†, -A, -B, -C].

** DefTensor: Defining torsion spinor Omeganb†[-A, -A†, -B†, -C†].

** DefTensor: Defining spinor SP␣Riemannnb[-A, -A†, -B, -B†, -C, -C†, -D, -D†]
 . Equivalent of tensor Riemannnb

** DefTensor: Defining spinor SP␣Riemannnb†[-A†, -A, -B†, -B, -C†, -C, -D†, -D]
 . Equivalent of tensor Riemannnb

** DefTensor: Defining spinor
 SP␣Riccinb[-A, -A†, -B, -B†]. Equivalent of tensor Riccinb

** DefTensor: Defining spinor
 SP␣Riccinb†[-A†, -A, -B†, -B]. Equivalent of tensor Riccinb
```

Several new objects are defined along with the covariant derivative. These are the same which are introduced by `DefCovD` plus the spinor equivalents of all the curvature quantities. By definition the compatibility of the covariant derivative `nb` with the spin structure associated to σ means that

In[156]:=
```
nb[-a]@σ[b, -A, -A†]
```

Out[156]=
```
0
```

Rules to decompose the curvature spinors into irreducible parts are also available. For example, let us compute the full decomposition of the spinor counterpart of the Riemann tensor of the covariant derivative `nb` in irreducible parts.

In[157]:=
```
PutSolderingForm@Riemannnb[-a, -b, -c, -d]
```

Out[157]=
$$\text{R[D]}_{\text{abcd}}\ \sigma^{\text{a}}{}_{\text{AA'}}\ \sigma^{\text{b}}{}_{\text{BB'}}\ \sigma^{\text{c}}{}_{\text{CC'}}\ \sigma^{\text{d}}{}_{\text{DD'}}$$

In[158]:=
```
ContractSolderingForm@%
```

Out[158]=
$$\text{SR[D]}_{\text{AA' BB' CC' DD'}}$$

In[159]:=
```
Decomposition[SP␣Riemannnb[-A, -A†, -B, -B†, -C, -C†, -D, -D†]]
```

Out[159]=
$$\text{SFnb}\dagger_{\text{A' AB' BD' C'}}\ \epsilon_{\text{CD}} + \text{SFnb}_{\text{AA' BB' DC}}\ \overline{\epsilon}_{\text{C' D'}}$$

In[160]:=
```
Decomposition[%]
```

Out[160]=
$$\epsilon_{\text{CD}}\ \left(\text{X[D]}\ \dagger_{\text{D' C' A' B'}}\ \epsilon_{\text{AB}} + \Phi\text{[D]}\ \dagger_{\text{D' C' AB}}\ \overline{\epsilon}_{\text{A' B'}}\right) + \left(\Phi\text{[D]}_{\text{DCA' B'}}\ \epsilon_{\text{AB}} + \text{X[D]}_{\text{DCAB}}\ \overline{\epsilon}_{\text{A' B'}}\right)\ \overline{\epsilon}_{\text{C' D'}}$$

©2003–2004 José M. Martín–García

In[161]:=
```
ToCanonical@%
```

Out[161]=
$$X[D]\ \dagger_{D',C',A',B'}\ \epsilon_{AB}\ \epsilon_{CD} + \Phi[D]\ \dagger_{D',C',AB}\ \epsilon_{CD}\ \overline{\epsilon}_{A',B'} + \Phi[D]_{DCA',B'}\ \epsilon_{AB}\ \overline{\epsilon}_{C',D'} + X[D]_{DCAB}\ \overline{\epsilon}_{A',B'}\ \overline{\epsilon}_{C',D'}$$

Note that the curvature spinors associated to a generic spin–compatible covariant derivative do not have the same symmetries as those of the spin covariant derivative.

In[162]:=
```
{{SymmetryGroupOfTensor@Chinb, SymmetryGroupOfTensor@ChiCDe},
 {SymmetryGroupOfTensor@Phinb, SymmetryGroupOfTensor@PhiCDe}}
```

Out[162]=
```
{{StrongGenSet[{3, 4}, GenSet[Cycles[{3, 4}]]], StrongGenSet[{1, 3},
    GenSet[Cycles[{1, 2}]], Cycles[{3, 4}], Cycles[{1, 3}, {2, 4}]]]},
 {StrongGenSet[{3, 4}, GenSet[Cycles[{3, 4}]]],
  StrongGenSet[{1, 3}, GenSet[Cycles[{1, 2}], Cycles[{3, 4}]]]]}}
```

There are also decomposition rules for the torsion spinor.

In[163]:=
```
Torsionnb[a, -b, -c] == SeparateSolderingForm[]@Torsionnb[a, -b, -c]
```

Out[163]=
$$T[D]^{a}{}_{bc} == ST[D]^{AA'}{}_{BB',CC'}\ \sigma^{a}{}_{AA'}\ \sigma_{b}{}^{BB'}\ \sigma_{c}{}^{CC'}$$

In[164]:=
```
PutSolderingForm /@ %
```

Out[164]=
$$T[D]^{a}{}_{bc}\ \sigma_{a}{}^{AA'}\ \sigma^{b}{}_{BB'}\ \sigma^{c}{}_{CC'} == ST[D]^{AA'}{}_{BB',CC'}$$

In[165]:=
```
Decomposition /@ %
```

Out[165]=
$$T[D]^{a}{}_{bc}\ \sigma_{a}{}^{AA'}\ \sigma^{b}{}_{BB'}\ \sigma^{c}{}_{CC'} == \epsilon^{AD}\ \overline{\epsilon}^{A',D'}\ \left(\Omega[D]\ \dagger_{DD',B',C'}\ \epsilon_{BC} + \Omega[D]_{D',DBC}\ \overline{\epsilon}_{B',C'}\right)$$

In[166]:=
```
ContractMetric /@ %
```

Out[166]=
$$T[D]^{a}{}_{bc}\ \sigma_{a}{}^{AA'}\ \sigma^{b}{}_{BB'}\ \sigma^{c}{}_{CC'} == \Omega[D]\ \dagger^{AA'}{}_{B',C'}\ \epsilon_{BC} + \Omega[D]^{A'\,A}{}_{BC}\ \overline{\epsilon}_{B',C'}$$

The covariant derivative `nb` can be manipulated in 2–index or in single index notation.

In[167]:=
```
nb[-A, -A†]@Chinb[-C, -D, -F, -P]
```

Out[167]=
$$D_{AA'}\ X[D]_{CDFP}$$

In[168]:=
```
SeparateSolderingForm[][%, nb]
```

Out[168]=
$$\sigma^{a}{}_{AA'}\ (D_{a}\ X[D]_{CDFP})$$

©2003–2004 José M. Martín–García

```
In[169]:=
PutSolderingForm@%

Out[169]=
D_a X[D]_CDFP
```

In postfix notation :

```
In[170]:=
$CovDFormat = "Postfix";

In[171]:=
nb[-A, -A†]@Chinb[-C, -D, -F, -P]

Out[171]=
X[D]_CDFP|AA'

In[172]:=
$CovDFormat = "Prefix";
```

The spinor expression of the Ricci identities of any spin–compatible covariant derivative is neatly rendered in terms of the *box operator*. This is a linear differential operator and `Spinors`` is able to work with it. The box operator is represented by the head `BoxCovDname` where the symbol `CovDname` represents the spin–compatible covariant derivative.

Action of the box operator on a spinor and some simple manipulations involving it.

```
In[173]:=
DefSpinor[ξ[-A], M4]

    ** DefTensor: Defining tensor ξ[-A].

    ** DefTensor: Defining tensor ξ†[-A†].

In[174]:=
BoxCDe[-A, -B]@ξ[-C]

Out[174]=
□[∇]_AB ξ_C

In[175]:=
ϵ[A, B] BoxCDe[-A, -B]@ξ[-C]

Out[175]=
ϵ^AB (□[∇]_AB ξ_C)

In[176]:=
ContractMetric@%

Out[176]=
□[∇]_A^A ξ_C

In[177]:=
ToCanonical@%

Out[177]=
0
```

The box operator can be expanded into spin covariant derivatives or into curvature spinors plus the torsion, according to

©2003–2004 José M. Martín–García

known formulae (see eqs. (4.9.8)–(4.9.17) of Penrose & Rindler Vol. 2). *Spinors* has built–in rules to perform these expansions.

BoxToCovD	Expand the box operator into covariant derivatives
BoxToCurvature	Expand the box operator into curvature spinors

Examples of the expansion of the Box operator.

```
In[178]:=
BoxCDe[-A, -B]@ξ[-C]
```

```
Out[178]=
□[∇]AB ξC
```

```
In[179]:=
BoxToCovD[%, BoxCDe]
```

```
Out[179]=
1/2 (∇AA', ∇B A' ξC + ∇BA', ∇A A' ξC)
```

```
In[180]:=
BoxToCurvature[%%, BoxCDe]
```

```
Out[180]=
-X[∇]DCAB ξD
```

Similar computations for the box operator arising from the covariant derivative `nb`. Note the presence of an extra term due to the fact that the covariant derivative `nb` has torsion.

```
In[181]:=
Boxnb[-A, -B]@ξ[-C]
```

```
Out[181]=
□[D]AB ξC
```

```
In[182]:=
BoxToCovD[%, Boxnb]
```

```
Out[182]=
1/2 (DAA', DB A' ξC + DBA', DA A' ξC)
```

```
In[183]:=
BoxToCurvature[%%, Boxnb]
```

```
Out[183]=
-X[D]DCAB ξD - Ω[D]A' D AB (DDA', ξC)
```

Other computations involving the box operator.

```
In[184]:=
BoxCDe[-A†, -B†]@ξ[-C]
```

```
Out[184]=
□[∇]A', B', ξC
```

©2003–2004 José M. Martín–García

In[185]:=
```
BoxToCovD[%, BoxCDe]
```

Out[185]=
$$\frac{1}{2}\left(\nabla_{AA'}\,\nabla^{A}{}_{B'}\,\xi_{C}+\nabla_{AB'}\,\nabla^{A}{}_{A'}\,\xi_{C}\right)$$

In[186]:=
```
BoxToCurvature[%%, BoxCDe]
```

Out[186]=
$$-\Phi[\nabla]_{ACA'B'}\,\xi^{A}$$

In[187]:=
```
BoxCDe[-A, -B]@PhiCDe[-C, -D, -C†, -D†]
```

Out[187]=
$$\Box[\nabla]_{AB}\,\Phi[\nabla]_{CDC'D'}$$

In[188]:=
```
BoxToCovD[%, BoxCDe]
```

Out[188]=
$$\frac{1}{2}\left(\nabla_{AA'}\,\nabla_{B}{}^{A'}\,\Phi[\nabla]_{CDC'D'}+\nabla_{BA'}\,\nabla_{A}{}^{A'}\,\Phi[\nabla]_{CDC'D'}\right)$$

In[189]:=
```
BoxToCurvature[%%, BoxCDe]
```

Out[189]=
$$\Phi[\nabla]_{ABA'C'}\,\Phi[\nabla]_{CD}{}^{A'}{}_{D'}+\Phi[\nabla]_{ABA'D'}\,\Phi[\nabla]_{CDC'}{}^{A'}-X[\nabla]_{FDAB}\,\Phi[\nabla]_{C}{}^{F}{}_{C'D'}-X[\nabla]_{FCAB}\,\Phi[\nabla]^{F}{}_{DC'D'}$$

©2003–2004 José M. Martín–García

We no longer need the spin−compatible covariant derivative nb :

In[190]:=
```
UndefSpinCovD[nb]
```

```
** UndefTensor: Undefined spinor SP`Torsionnb†

** UndefTensor: Undefined spinor SP`Torsionnb

** UndefTensor: Undefined spinor SP`Riemannnb†

** UndefTensor: Undefined spinor SP`Riemannnb

** UndefTensor: Undefined spinor SP`Riccinb†

** UndefTensor: Undefined spinor SP`Riccinb

** UndefTensor: Undefined spinor SP`FRiemannnb†

** UndefTensor: Undefined spinor SP`FRiemannnb

** UndefTensor: Undefined nonsymmetric AChristoffel tensor AChristoffelnb†

** UndefTensor: Undefined nonsymmetric AChristoffel tensor AChristoffelnb

** UndefTensor: Undefined curvature spinor Chinb†

** UndefTensor: Undefined curvature spinor Chinb

** UndefTensor: Undefined non-symmetric Christoffel tensor Christoffelnb

** UndefTensor: Undefined FRiemann tensor FRiemannnb†

** UndefTensor: Undefined FRiemann tensor FRiemannnb

** UndefTensor: Undefined torsion spinor Omeganb†

** UndefTensor: Undefined torsion spinor Omeganb

** UndefTensor: Undefined curvature spinor Phinb†

** UndefTensor: Undefined curvature spinor Phinb

** UndefTensor: Undefined non-symmetric Ricci tensor Riccinb

** UndefTensor: Undefined Riemann tensor Riemannnb

** UndefTensor: Undefined torsion tensor Torsionnb

** UndefCovD: Undefined covariant derivative nb
```

©2003−2004 José M. Martín−García

■ 6. Example: Killing spinors

Let us introduce a symmetric valence−2 spinor:

```
In[191]:=
DefSpinor[κ[-A, -B], M4, Symmetric[{-A, -B}]]; PrintAs@κ ↾^= "κ̄";

   ** DefTensor: Defining tensor κ[-A, -B].

   ** DefTensor: Defining tensor κ̄[-A↾, -B↾].
```

We will say that the spinor κ_{AB} is a Killing spinor if the following differential condition is fulfilled.

```
In[192]:=
ToCanonical@Symmetrize[3 CDe[-C, -C↾]@κ[-A, -B], {-A, -B, -C}] == 0
```

$$\nabla_{AC}\,\kappa_{BC} + \nabla_{BC}\,\kappa_{AC} + \nabla_{CC}\,\kappa_{AB} == 0$$

```
In[193]:=
KillingSpinorCondition = %;
```

Killing spinors have a number of applications. For example whenever a Killing spinor exists then automatically one can construct a conformal Killing−Yano tensor and a conformal Killing spinor in the spacetime which in turn lead to the existence of integration constants for the geodesic equations. These results were presented by M. Walker and R. Penrose in "On Quadratic First Integrals of the Geodesic Equations in {2,2} Spacetimes" *Comm. Math. Phys.* **18** 265−274 (1970) and we will re−derive them with the aid of `Spinors`.

6.2. Existence of a Killing spinor

The existence of a Killing spinor imposes restrictions on the Weyl spinor. To find them, we start by taking the derivative of the Killing spinor condition.

```
In[194]:=
CDe[-F, C↾] /@ KillingSpinorCondition
```

$$\nabla_F{}^{C'}\,\nabla_{AC}\,\kappa_{BC} + \nabla_F{}^{C'}\,\nabla_{BC}\,\kappa_{AC} + \nabla_F{}^{C'}\,\nabla_{CC}\,\kappa_{AB} == 0$$

We decompose each term of this sum into irreducible parts by making use of the "box operator".

```
In[195]:=
% /. CDe[-F_, C↾_]@CDe[-A_, -C↾_]@κ[-B_, -C_] ↦
  -BoxCDe[-F, -A]@κ[-B, -C] - 1 / 2 ϵ[-F, -A] CDe[-P, -C↾]@CDe[P, C↾]@κ[-B, -C]
```

$$-\left(\Box[\nabla]_{FA}\,\kappa_{BC}\right) - \Box[\nabla]_{FB}\,\kappa_{AC} - \Box[\nabla]_{FC}\,\kappa_{AB} - \frac{1}{2}\epsilon_{FC}\left(\nabla_{DA'}\,\nabla^{DA'}\,\kappa_{AB}\right) - \frac{1}{2}\epsilon_{FB}\left(\nabla_{DA'}\,\nabla^{DA'}\,\kappa_{AC}\right) - \frac{1}{2}\epsilon_{FA}\left(\nabla_{DA'}\,\nabla^{DA'}\,\kappa_{BC}\right) == 0$$

©2003−2004 José M. Martín−García

The action of the box operator is computed next.

In[196]:=
```
BoxToCurvature[#, BoxCDe] & /@ %
```

Out[196]=

$$X[\nabla]_{DBFC}\,\kappa_A{}^D + X[\nabla]_{DCFB}\,\kappa_A{}^D + X[\nabla]_{DCFA}\,\kappa_B{}^D + X[\nabla]_{DAFC}\,\kappa^D{}_B + X[\nabla]_{DAFB}\,\kappa^D{}_C + X[\nabla]_{DBFA}\,\kappa^D{}_C - \frac{1}{2}\epsilon_{FC}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{AB}\right) - \frac{1}{2}\epsilon_{FB}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{AC}\right) - \frac{1}{2}\epsilon_{FA}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{BC}\right) == 0$$

We decompose the curvature spinor into irreducible parts.

In[197]:=
```
Decomposition[%, Chi]
```

Out[197]=

$$(\Psi[\nabla]_{DBFC} + \Lambda[\nabla]\ (\epsilon_{BF}\,\epsilon_{DC} + \epsilon_{BC}\,\epsilon_{DF}))\,\kappa_A{}^D + (\Psi[\nabla]_{DCFB} + \Lambda[\nabla]\ (\epsilon_{CF}\,\epsilon_{DB} + \epsilon_{CB}\,\epsilon_{DF}))\,\kappa_A{}^D + (\Psi[\nabla]_{DCFA} + \Lambda[\nabla]\ (\epsilon_{CF}\,\epsilon_{DA} + \epsilon_{CA}\,\epsilon_{DF}))\,\kappa_B{}^D + (\Psi[\nabla]_{DAFC} + \Lambda[\nabla]\ (\epsilon_{AF}\,\epsilon_{DC} + \epsilon_{AC}\,\epsilon_{DF}))\,\kappa^D{}_B + (\Psi[\nabla]_{DAFB} + \Lambda[\nabla]\ (\epsilon_{AF}\,\epsilon_{DB} + \epsilon_{AB}\,\epsilon_{DF}))\,\kappa^D{}_C + (\Psi[\nabla]_{DBFA} + \Lambda[\nabla]\ (\epsilon_{BF}\,\epsilon_{DA} + \epsilon_{BA}\,\epsilon_{DF}))\,\kappa^D{}_C - \frac{1}{2}\epsilon_{FC}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{AB}\right) - \frac{1}{2}\epsilon_{FB}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{AC}\right) - \frac{1}{2}\epsilon_{FA}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{BC}\right) == 0$$

In[198]:=
```
ToCanonical /@ (ContractMetric /@ %)
```

Out[198]=

$$2\,\Lambda[\nabla]\ \epsilon_{CF}\,\kappa_{AB} + 2\,\Lambda[\nabla]\ \epsilon_{BF}\,\kappa_{AC} + 2\,\Psi[\nabla]_{BCFD}\,\kappa_A{}^D + 2\,\Lambda[\nabla]\ \epsilon_{AF}\,\kappa_{BC} + 2\,\Psi[\nabla]_{ACFD}\,\kappa_B{}^D + 2\,\Psi[\nabla]_{ABFD}\,\kappa_C{}^D + \frac{1}{2}\epsilon_{CF}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{AB}\right) + \frac{1}{2}\epsilon_{BF}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{AC}\right) + \frac{1}{2}\epsilon_{AF}\left(\nabla_{DA}, \nabla^{DA'}\,\kappa_{BC}\right) == 0$$

We are only interested in the symmetric part of this expression.

In[199]:=
```
Symmetrize[#, {-B, -F, -A, -C}] & /@ %;
```

In[200]:=
```
ToCanonical /@ (Times[#, 2 / 3] & /@ %)
```

Out[200]=

$$\Psi[\nabla]_{BCFD}\,\kappa_A{}^D + \Psi[\nabla]_{ACFD}\,\kappa_B{}^D + \Psi[\nabla]_{ABFD}\,\kappa_C{}^D + \Psi[\nabla]_{ABCD}\,\kappa_F{}^D == 0$$

We will refer to this condition as the Killing spinor integrability condition.

In[201]:=
```
KSIntegrability = %;
```

©2003–2004 José M. Martín–García

This algebraic condition implies that the algebraic type of the Weyl spinor can be only D or N.

6.3. Conformal Killing–Yano tensors

We are going to find the tensor representation of the Killing spinor condition. To that end we define an antisymmetric tensor and its spinor counterpart.

```
In[202]:=
DefTensor[DF[-a, -b], M4, Antisymmetric[{-a, -b}], PrintAs → "D"]

   ** DefTensor: Defining tensor DF[-a, -b].

In[203]:=
DefSpinorOfTensor[DFs[-A, -A†, -B, -B†], DF[-a, -b], σ, PrintAs -> "D"]

   ** DefTensor: Defining spinor DFs[-A, -A†, -B, -B†]. Equivalent of tensor DF

   ** DefTensor: Defining spinor DFs†[-A†, -A, -B†, -B]. Equivalent of tensor DF
```

The spinor $D_{AA', BB'}$ is defined by the condition:

```
In[204]:=
DFs[-A, -A†, -B, -B†] == κ[-A, -B] ϵ ϵ[-A†, -B†] + Dagger[κ[-A, -B] ϵ ϵ[-A†, -B†]]

Out[204]=
D_{AA', BB'} == \overline{ϵ}_{A', B'} κ_{AB} + ϵ_{AB} \overline{κ}_{A', B'}
```

This entails:

```
In[205]:=
SeparateSolderingForm[σ] /@ %

Out[205]=
D_{ab} σ^{a}_{AA'} σ^{b}_{BB'} == \overline{ϵ}_{A', B'} κ_{AB} + ϵ_{AB} \overline{κ}_{A', B'}

In[206]:=
% /. -A† -> B†

Out[206]=
D_{ab} σ^{a}_{A}^{B'} σ^{b}_{BB'} == -2 κ_{AB} + ϵ_{AB} \overline{κ}^{B'}_{B'}

In[207]:=
ToCanonical /@ %

Out[207]=
D_{ab} σ^{a}_{A}^{B'} σ^{b}_{BB'} == -2 κ_{AB}

In[208]:=
IndexSolve[%, κ[-A, -B]]

Out[208]=
{HoldPattern[\overline{κ}^{A B}] :→ Module[{a, A†, b}, -1/2 D_{ab} σ^{aAA'} σ^{bB}_{A'}]}

In[209]:=
KillingSpinorToTensor = %;
```

©2003–2004 José M. Martín–García

We need to compute the complex conjugate of this rule

In[210]:=
```
Dagger /@ %%%
```

Out[210]=

$$D_{ab}\,\sigma^{aB}{}_{A'},\,\sigma^{b}{}_{BB'} == -2\,\overline{\kappa}_{A'\,B'}$$

In[211]:=
```
IndexSolve[%, κ ↾[-A↾, -B↾]]
```

Out[211]=

$$\left\{\text{HoldPattern}\left[\overline{\kappa}^{A'\,B'}\right] :\to \text{Module}\left[\{a, A, b\}, -\frac{1}{2}\,D_{ab}\,\sigma^{aAA'}\,\sigma^{b}{}_{A}{}^{B'}\right]\right\}$$

In[212]:=
```
KillingSpinorToTensor↾ = %;
```

We can now proceed to the computation of the spinor form of the Killing spinor condition. First of all, we need to have as many primed indices as unprimed ones and we achieve this by multiplying by $\overline{\epsilon}_{A'\,B'}$ on both sides of the Killing spinor condition.

In[213]:=
```
Times[#, ϵ ↾[-A↾, -B↾]] & /@ KillingSpinorCondition
```

Out[213]=

$$\overline{\epsilon}_{A'\,B'}\,\left(\nabla_{AC}\,\kappa_{BC} + \nabla_{BC}\,\kappa_{AC} + \nabla_{CC'}\,\kappa_{AB}\right) == 0$$

This can be now transformed into a tensor expression.

In[214]:=
```
PutSolderingForm[#, {Pair[-A, -A↾], Pair[-B, -B↾], Pair[-C, -C↾]}, σ] & /@ %
```

Out[214]=

$$\overline{\epsilon}_{A'\,B'}\,\sigma_{a}{}^{AA'}\,\sigma_{b}{}^{BB'}\,\sigma_{c}{}^{CC'}\,\left(\nabla_{AC}\,\kappa_{BC} + \nabla_{BC}\,\kappa_{AC} + \nabla_{CC'}\,\kappa_{AB}\right) == 0$$

In[215]:=
```
ToCanonical /@ (ContractMetric /@ %)
```

Out[215]=

$$-\sigma_{a}{}^{AA'}\,\sigma_{b}{}^{B}{}_{A},\,\sigma_{c}{}^{CB'}\,\left(\nabla_{AB},\,\kappa_{BC}\right) - \sigma_{a}{}^{AA'}\,\sigma_{b}{}^{B}{}_{A},\,\sigma_{c}{}^{CB'}\,\left(\nabla_{BB},\,\kappa_{AC}\right) - \sigma_{a}{}^{AA'}\,\sigma_{b}{}^{B}{}_{A},\,\sigma_{c}{}^{CB'}\,\left(\nabla_{CB},\,\kappa_{AB}\right) == 0$$

We replace κ_{AB} by the expression in terms of D_{df} found above.

In[216]:=
```
% /. KillingSpinorToTensor
```

Out[216]=

$$\frac{1}{2}\,\sigma_{a}{}^{AB'}\,\sigma_{b}{}^{B}{}_{B},\,\sigma_{c}{}^{CC'}\,\sigma^{d}{}_{B}{}^{A'}\,\sigma^{f}{}_{CA},\,\left(\nabla_{AC},\,D_{df}\right) + \frac{1}{2}\,\sigma_{a}{}^{AB'}\,\sigma_{b}{}^{B}{}_{B},\,\sigma_{c}{}^{CC'}\,\sigma^{d}{}_{A}{}^{A'}\,\sigma^{f}{}_{CA},\,\left(\nabla_{BC},\,D_{df}\right) + \frac{1}{2}\,\sigma_{a}{}^{AB'}\,\sigma_{b}{}^{B}{}_{B},\,\sigma_{c}{}^{CC'}\,\sigma^{d}{}_{A}{}^{A'}\,\sigma^{f}{}_{BA},\,\left(\nabla_{CC'},\,D_{df}\right) == 0$$

©2003–2004 José M. Martín–García

We get rid of spinor indices in the covariant derivatives.

```
In[217]:=
% /. CDe[-A_, -A† _]@DF[-a_, -b_] ↦
  SeparateSolderingForm[σ][CDe[-A, -A†]@DF[-a, -b], IndexList[-A, -A†]]
```

```
Out[217]=
1/2 σa^AB' σb^B_B, σ^d_A^A' σ^f_BA', (∇c Ddf) +
  1/2 σa^AB' σb^B_B, σc^CC' σ^d_B^A' σ^f_CA', σ^h_AC', (∇h Ddf) + 1/2 σa^AB' σb^B_B, σc^CC' σ^d_A^A' σ^f_CA', σ^h_BC', (∇h Ddf) == 0
```

Finally, we replace the product of soldering form by products of "tetra–metrics". This eliminates all the spinor indices.

```
In[218]:=
MapAt[ContractSolderingForm[#, Spin] &, %, 1]
```

```
Out[218]=
-1/2 g^dh Gg_a^f_db (∇c Dfh) - 1/2 Gg_a^j_db Gg^fd_c^h (∇j Dfh) + 1/2 g^fd Gg_a^h_db Gg_c^j_f^l (∇l Dhj) == 0
```

All the quantities appearing in previous expression are tensorial, so we may replace the spin covariant derivative by the Levi–Civita covariant derivative.

```
In[219]:=
% /. CDe → CD
```

```
Out[219]=
-1/2 g^dh Gg_a^f_db (∇c Dfh) - 1/2 Gg_a^j_db Gg^fd_c^h (∇j Dfh) + 1/2 g^fd Gg_a^h_db Gg_c^j_f^l (∇l Dhj) == 0
```

```
In[220]:=
% /. TetraRule@g
```

```
Out[220]=
-1/2 (i ∈ g^f_db/2 - 1/2 δd^f gab + 1/2 δb^f gad - 1/2 δa^f gdb) g^dh (∇c Dfh) -
  1/2 (i ∈ g^j_db/2 - 1/2 δd^j gab + 1/2 δb^j gad - 1/2 δa^j gdb) (i ∈ g^fd_c^h/2 + 1/2 δc^f g^dh - 1/2 δc^h g^fd - 1/2 δc^d g^fh)
    (∇j Dfh) + 1/2 (i ∈ g^h_db/2 - 1/2 δd^h gab + 1/2 δb^h gad - 1/2 δa^h gdb) g^fd
    (-1/2 δc^l δf^j - 1/2 δc^j δf^l + i ∈ g_c^j_f^l/2 + 1/2 gcf g^jl) (∇l Dhj) == 0
```

©2003–2004 José M. Martín–García

In[221]:=
```
ToCanonical /@ (ContractMetric /@ %)
```

Out[221]=

$$-\frac{1}{2}\left(\nabla_a D_{bc}\right)+\frac{1}{8}\,i\in g_{bcdf}\left(\nabla_a D^{df}\right)+\frac{1}{2}\left(\nabla_b D_{ac}\right)-\frac{1}{8}\,i\in g_{acdf}\left(\nabla_b D^{df}\right)+$$
$$\nabla_c D_{ab}-\frac{3}{8}\,i\in g_{abdf}\left(\nabla_c D^{df}\right)-\frac{1}{2}\,g_{bc}\left(\nabla_d D_a{}^d\right)+\frac{1}{2}\,g_{ac}\left(\nabla_d D_b{}^d\right)+\frac{1}{8}\,i\in g_{abcf}\left(\nabla_d D^{df}\right)-$$
$$\frac{1}{8}\,i\in g_{bcdf}\left(\nabla^f D_a{}^d\right)+\frac{1}{8}\,i\in g_{acdf}\left(\nabla^f D_b{}^d\right)+\frac{3}{8}\,i\in g_{abdf}\left(\nabla^f D_c{}^d\right)==0$$

In[222]:=
```
Simplification /@ %
```

Out[222]=

$$\frac{1}{8}\left(-4\left(\nabla_a D_{bc}\right)+i\left(-4\,i\left(\nabla_b D_{ac}\right)-\in g_{acdf}\left(\nabla_b D^{df}\right)-8\,i\left(\nabla_c D_{ab}\right)-\right.\right.$$
$$3\in g_{abdf}\left(\nabla_c D^{df}\right)+4\,i\,g_{bc}\left(\nabla_d D_a{}^d\right)-4\,i\,g_{ac}\left(\nabla_d D_b{}^d\right)+\in g_{abcf}\left(\nabla_d D^{df}\right)+$$
$$\left.\left.\in g_{bcdf}\left(\nabla_a D^{df}-\nabla^f D_a{}^d\right)+\in g_{acdf}\left(\nabla^f D_b{}^d\right)+3\in g_{abdf}\left(\nabla^f D_c{}^d\right)\right)\right)==0$$

In[223]:=
```
ScreenDollarIndices /@ %
```

Out[223]=

$$\frac{1}{8}\left(-4\left(\nabla_a D_{bc}\right)+i\left(-4\,i\left(\nabla_b D_{ac}\right)-\in g_{acdf}\left(\nabla_b D^{df}\right)-8\,i\left(\nabla_c D_{ab}\right)-\right.\right.$$
$$3\in g_{abdf}\left(\nabla_c D^{df}\right)+4\,i\,g_{bc}\left(\nabla_d D_a{}^d\right)-4\,i\,g_{ac}\left(\nabla_d D_b{}^d\right)+\in g_{abcf}\left(\nabla_d D^{df}\right)+$$
$$\left.\left.\in g_{bcdf}\left(\nabla_a D^{df}-\nabla^f D_a{}^d\right)+\in g_{acdf}\left(\nabla^f D_b{}^d\right)+3\in g_{abdf}\left(\nabla^f D_c{}^d\right)\right)\right)==0$$

We split this expression into real and imaginary parts.

In[224]:=
```
Re /@ %
```

Out[224]=

$$\frac{1}{8}\left(-\mathrm{Im}\left[-\in g_{acdf}\left(\nabla_b D^{df}\right)-3\in g_{abdf}\left(\nabla_c D^{df}\right)+\in g_{abcf}\left(\nabla_d D^{df}\right)+\right.\right.$$
$$\left.\in g_{bcdf}\left(\nabla_a D^{df}-\nabla^f D_a{}^d\right)+\in g_{acdf}\left(\nabla^f D_b{}^d\right)+3\in g_{abdf}\left(\nabla^f D_c{}^d\right)\right]-4\,\mathrm{Re}\left[\nabla_a D_{bc}\right]+$$
$$\left.4\,\mathrm{Re}\left[\nabla_b D_{ac}\right]+8\,\mathrm{Re}\left[\nabla_c D_{ab}\right]-4\,\mathrm{Re}\left[g_{bc}\left(\nabla_d D_a{}^d\right)\right]+4\,\mathrm{Re}\left[g_{ac}\left(\nabla_d D_b{}^d\right)\right]\right)==0$$

In[225]:=
```
ComplexExpand /@ %
```

Out[225]=

$$-\frac{1}{8}\in g_{bcdf}\,\mathrm{Im}\left[\nabla_a D^{df}\right]+\frac{1}{8}\in g_{acdf}\,\mathrm{Im}\left[\nabla_b D^{df}\right]+\frac{3}{8}\in g_{abdf}\,\mathrm{Im}\left[\nabla_c D^{df}\right]-$$
$$\frac{1}{8}\in g_{abcf}\,\mathrm{Im}\left[\nabla_d D^{df}\right]+\frac{1}{8}\in g_{bcdf}\,\mathrm{Im}\left[\nabla^f D_a{}^d\right]-\frac{1}{8}\in g_{acdf}\,\mathrm{Im}\left[\nabla^f D_b{}^d\right]-\frac{3}{8}\in g_{abdf}\,\mathrm{Im}\left[\nabla^f D_c{}^d\right]-$$
$$\frac{1}{2}\,\mathrm{Re}\left[\nabla_a D_{bc}\right]+\frac{1}{2}\,\mathrm{Re}\left[\nabla_b D_{ac}\right]+\mathrm{Re}\left[\nabla_c D_{ab}\right]-\frac{1}{2}\,g_{bc}\,\mathrm{Re}\left[\nabla_d D_a{}^d\right]+\frac{1}{2}\,g_{ac}\,\mathrm{Re}\left[\nabla_d D_b{}^d\right]==0$$

©2003–2004 José M. Martín–García

In[226]:=
```
% /. Im[_] → 0 /. Re → Identity
```

Out[226]=
$$-\frac{1}{2}\left(\nabla_a D_{bc}\right)+\frac{1}{2}\left(\nabla_b D_{ac}\right)+\nabla_c D_{ab}-\frac{1}{2}g_{bc}\left(\nabla_d D_a{}^d\right)+\frac{1}{2}g_{ac}\left(\nabla_d D_b{}^d\right)==0$$

In[227]:=
```
ToCanonical /@ %
```

Out[227]=
$$-\frac{1}{2}\left(\nabla_a D_{bc}\right)+\frac{1}{2}\left(\nabla_b D_{ac}\right)+\nabla_c D_{ab}-\frac{1}{2}g_{bc}\left(\nabla_d D_a{}^d\right)+\frac{1}{2}g_{ac}\left(\nabla_d D_b{}^d\right)==0$$

In[228]:=
```
Symmetrize[#, {-c, -a}] & /@ %
```

Out[228]=
$$\frac{1}{2}\left(-\frac{1}{2}\left(\nabla_a D_{bc}\right)+\nabla_a D_{cb}+\frac{1}{2}\left(\nabla_b D_{ac}\right)+\frac{1}{2}\left(\nabla_b D_{ca}\right)+\nabla_c D_{ab}-\frac{1}{2}\left(\nabla_c D_{ba}\right)-\frac{1}{2}g_{bc}\left(\nabla_d D_a{}^d\right)+\frac{1}{2}g_{ac}\left(\nabla_d D_b{}^d\right)+\frac{1}{2}g_{ca}\left(\nabla_d D_b{}^d\right)-\frac{1}{2}g_{ba}\left(\nabla_d D_c{}^d\right)\right)==0$$

In[229]:=
```
ToCanonical /@ %
```

Out[229]=
$$-\frac{3}{4}\left(\nabla_a D_{bc}\right)+\frac{3}{4}\left(\nabla_c D_{ab}\right)-\frac{1}{4}g_{bc}\left(\nabla_d D_a{}^d\right)+\frac{1}{2}g_{ac}\left(\nabla_d D_b{}^d\right)-\frac{1}{4}g_{ab}\left(\nabla_d D_c{}^d\right)==0$$

In[230]:=
```
Times[#, 2 / 3] & /@ %
```

Out[230]=
$$\frac{2}{3}\left(-\frac{3}{4}\left(\nabla_a D_{bc}\right)+\frac{3}{4}\left(\nabla_c D_{ab}\right)-\frac{1}{4}g_{bc}\left(\nabla_d D_a{}^d\right)+\frac{1}{2}g_{ac}\left(\nabla_d D_b{}^d\right)-\frac{1}{4}g_{ab}\left(\nabla_d D_c{}^d\right)\right)==0$$

In[231]:=
```
ToCanonical /@ %
```

Out[231]=
$$-\frac{1}{2}\left(\nabla_a D_{bc}\right)+\frac{1}{2}\left(\nabla_c D_{ab}\right)-\frac{1}{6}g_{bc}\left(\nabla_d D_a{}^d\right)+\frac{1}{3}g_{ac}\left(\nabla_d D_b{}^d\right)-\frac{1}{6}g_{ab}\left(\nabla_d D_c{}^d\right)==0$$

In[232]:=
```
ConformalKillingYano = %;
```

In[233]:=
```
CDDF = IndexSolve[ConformalKillingYano, CD[-c]@DF[-a, -b]] // Flatten
```

Out[233]=
$$\left\{\text{HoldPattern}\left[\nabla^{\overset{c}{=}}D^{\overset{a\,b}{==}}\right]:\to\text{Module}\left[\{d\},\ \nabla^a D^{bc}+\frac{1}{3}g^{cb}\left(\nabla_d D^{ad}\right)-\frac{2}{3}g^{ca}\left(\nabla_d D^{bd}\right)+\frac{1}{3}g^{ba}\left(\nabla_d D^{cd}\right)\right]\right\}$$

©2003–2004 José M. Martín–García

6.3. Conformal Killing tensors

```
In[234]:=
DefTensor[PK[-a, -b], M4, Symmetric[{-a, -b}], PrintAs → "P"]

    ** DefTensor: Defining tensor PK[-a, -b].

In[235]:=
DefSpinorOfTensor[PKS[-A, -A†, -B, -B†], PK[-a, -b], σ, PrintAs → "P"]

    ** DefTensor: Defining spinor PKS[-A, -A†, -B, -B†]. Equivalent of tensor PK

    ** DefTensor: Defining spinor PKS†[-A†, -A, -B†, -B]. Equivalent of tensor PK
```

Consider the following quantity

```
In[236]:=
PKS[-A, -A†, -B, -B†] == 2 κ[-A, -B] Dagger@κ[-A, -B]

Out[236]=
P_{AA′ BB′} == 2 κ_{AB} κ̄_{A′ B′}

In[237]:=
SeparateSolderingForm[σ] /@ %

Out[237]=
P_{ab} σ^{a}_{AA′} σ^{b}_{BB′} == 2 κ_{AB} κ̄_{A′ B′}
```

We compute the tensor equivalent of this

```
In[238]:=
PutSolderingForm[#, {Pair[-A, -A†], Pair[-B, -B†]}, σ] & /@ %

Out[238]=
P_{ab} == 2 κ_{AB} κ̄_{A′ B′} σ_a^{AA′} σ_b^{BB′}

In[239]:=
% /. KillingSpinorToTensor

Out[239]=
P_{ab} == -D_{cd} κ̄_{B′ C′} σ_a^{AB′} σ_b^{BC′} σ^{c}_{A}^{A′} σ^{d}_{BA′}

In[240]:=
% /. KillingSpinorToTensor†

Out[240]=
P_{ab} == 1/2 D_{cd} D_{fh} σ_a^{BB′} σ_b^{CC′} σ^{c}_{B}^{A′} σ^{d}_{CA′} σ^{fA}_{B′} σ^{h}_{AC′}

In[241]:=
ContractSolderingForm[#, σ] & /@ %

Out[241]=
P_{ab} == 1/2 D_{fh} D_{jl} g^{dc} Gg_{a}^{f}_{c}^{j} Gg_{b}^{h}_{d}^{l}
```

©2003–2004 José M. Martín–García

In[242]:=
```
% /. TetraRule@g
```

Out[242]=

$$P_{ab} == \frac{1}{2} D_{fh} D_{jl} g^{dc} \left(-\frac{1}{2} \delta_a{}^j \delta_c{}^f - \frac{1}{2} \delta_a{}^f \delta_c{}^j + \frac{i \in g^{f\,j}_{\;\;c}}{2} + \frac{1}{2} g_{ac} g^{fj} \right)$$
$$\left(-\frac{1}{2} \delta_b{}^l \delta_d{}^h - \frac{1}{2} \delta_b{}^h \delta_d{}^l + \frac{i \in g^{h\,l}_{\;\;d}}{2} + \frac{1}{2} g_{bd} g^{hl} \right)$$

In[243]:=
```
ToCanonical /@ (ContractMetric /@ %)
```

Out[243]=

$$P_{ab} == -D_a{}^c D_{bc} + \frac{1}{4} D_{cd} D^{cd} g_{ab}$$

In[244]:=
```
ExpandKillingTensor = %;
```

We check next that P_{ab} is indeed a conformal Killing tensor.

In[245]:=
```
CD[-f] /@ ExpandKillingTensor
```

Out[245]=

$$\nabla_f P_{ab} == -D_{bc} \left(\nabla_f D_a{}^c\right) - D_a{}^c \left(\nabla_f D_{bc}\right) + \frac{1}{4} \left(D^{cd} g_{ab} \left(\nabla_f D_{cd}\right) + D_{cd} g_{ab} \left(\nabla_f D^{cd}\right) \right)$$

In[246]:=
```
ToCanonical /@ (Symmetrize[#, {-a, -b, -f}] & /@ %)
```

Out[246]=

$$\frac{1}{3} \left(\nabla_a P_{bf}\right) + \frac{1}{3} \left(\nabla_b P_{af}\right) + \frac{1}{3} \left(\nabla_f P_{ab}\right) ==$$
$$-\frac{1}{3} D_f{}^c \left(\nabla_a D_{bc}\right) + \frac{1}{6} D^{cd} g_{bf} \left(\nabla_a D_{cd}\right) - \frac{1}{3} D_b{}^c \left(\nabla_a D_{fc}\right) - \frac{1}{3} D_f{}^c \left(\nabla_b D_{ac}\right) +$$
$$\frac{1}{6} D^{cd} g_{af} \left(\nabla_b D_{cd}\right) - \frac{1}{3} D_a{}^c \left(\nabla_b D_{fc}\right) - \frac{1}{3} D_b{}^c \left(\nabla_f D_{ac}\right) - \frac{1}{3} D_a{}^c \left(\nabla_f D_{bc}\right) + \frac{1}{6} D^{cd} g_{ab} \left(\nabla_f D_{cd}\right)$$

In[247]:=
```
Simplification /@ %
```

Out[247]=

$$\frac{1}{3} \left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) ==$$
$$\frac{1}{6} \left(D^{cd} g_{bf} \left(\nabla_a D_{cd}\right) - 2 D_f{}^c \left(\nabla_a D_{bc} + \nabla_b D_{ac}\right) + D^{cd} g_{af} \left(\nabla_b D_{cd}\right) - 2 D_a{}^c \left(\nabla_b D_{fc}\right) - 2 D_b{}^c \left(\nabla_a D_{fc} + \nabla_f D_{ac}\right) - 2 D_a{}^c \left(\nabla_f D_{bc}\right) + D^{cd} g_{ab} \left(\nabla_f D_{cd}\right) \right)$$

©2003–2004 José M. Martín–García

In[248]:=
```
FullSimplify /@ %
```

Out[248]=

$$\frac{1}{3}\left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) == \frac{1}{6}\left(-2 D_f{}^c \left(\nabla_a D_{bc} + \nabla_b D_{ac}\right) - 2 D_b{}^c \left(\nabla_a D_{fc} + \nabla_f D_{ac}\right) - 2 D_a{}^c \left(\nabla_b D_{fc} + \nabla_f D_{bc}\right) + D^{cd}\left(g_{bf}\left(\nabla_a D_{cd}\right) + g_{af}\left(\nabla_b D_{cd}\right) + g_{ab}\left(\nabla_f D_{cd}\right)\right)\right)$$

In[249]:=
```
ScreenDollarIndices /@ %
```

Out[249]=

$$\frac{1}{3}\left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) == \frac{1}{6}\left(-2 D_f{}^c \left(\nabla_a D_{bc} + \nabla_b D_{ac}\right) - 2 D_b{}^c \left(\nabla_a D_{fc} + \nabla_f D_{ac}\right) - 2 D_a{}^c \left(\nabla_b D_{fc} + \nabla_f D_{bc}\right) + D^{cd}\left(g_{bf}\left(\nabla_a D_{cd}\right) + g_{af}\left(\nabla_b D_{cd}\right) + g_{ab}\left(\nabla_f D_{cd}\right)\right)\right)$$

In[250]:=
```
% /. CD[-a]@DF[-b, -c] → (CD[-a]@DF[-b, -c] /. CDDF)
```

Out[250]=

$$\frac{1}{3}\left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) == \frac{1}{6}\left(\left(-2 D_b{}^c \left(\nabla_a D_{fc} + \nabla_f D_{ac}\right) - 2 D_a{}^c \left(\nabla_b D_{fc} + \nabla_f D_{bc}\right) + D^{cd}\left(g_{bf}\left(\nabla_a D_{cd}\right) + g_{af}\left(\nabla_b D_{cd}\right) + g_{ab}\left(\nabla_f D_{cd}\right)\right) - 2 D_f{}^c \left(\nabla_b D_{ac} + \nabla_b D_{ca} + \frac{1}{3} g_{cb}\left(\nabla_h D_a{}^h\right) + \frac{1}{3} g_{ac}\left(\nabla_h D_b{}^h\right) - \frac{2}{3} g_{ab}\left(\nabla_h D_c{}^h\right)\right)\right)$$

In[251]:=
```
% /. CD[-a]@DF[-f, -c] → (CD[-a]@DF[-f, -c] /. CDDF)
```

Out[251]=

$$\frac{1}{3}\left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) == \frac{1}{6}\left(-2 D_a{}^c \left(\nabla_b D_{fc} + \nabla_f D_{bc}\right) + D^{cd}\left(g_{bf}\left(\nabla_a D_{cd}\right) + g_{af}\left(\nabla_b D_{cd}\right) + g_{ab}\left(\nabla_f D_{cd}\right)\right) - 2 D_f{}^c \left(\nabla_b D_{ac} + \nabla_b D_{ca} + \frac{1}{3} g_{cb}\left(\nabla_h D_a{}^h\right) + \frac{1}{3} g_{ac}\left(\nabla_h D_b{}^h\right) - \frac{2}{3} g_{ab}\left(\nabla_h D_c{}^h\right)\right) - 2 D_b{}^c \left(\nabla_f D_{ac} + \nabla_f D_{ca} + \frac{1}{3} g_{cf}\left(\nabla_j D_a{}^j\right) - \frac{2}{3} g_{af}\left(\nabla_j D_c{}^j\right) + \frac{1}{3} g_{ac}\left(\nabla_j D_f{}^j\right)\right)\right)$$

©2003–2004 José M. Martín–García

In[252]:=
```
% /. CD[-b]@DF[-f, -c] → (CD[-b]@DF[-f, -c] /. CDDF)
```

Out[252]=
$$\frac{1}{3}\left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) == \frac{1}{6}\left(D^{cd}\left(g_{bf}\left(\nabla_a D_{cd}\right) + g_{af}\left(\nabla_b D_{cd}\right) + g_{ab}\left(\nabla_f D_{cd}\right)\right) - 2\,D_f{}^c\left(\nabla_b D_{ac} + \nabla_b D_{ca} + \frac{1}{3}g_{cb}\left(\nabla_h D_a{}^h\right) + \frac{1}{3}g_{ac}\left(\nabla_h D_b{}^h\right) - \frac{2}{3}g_{ab}\left(\nabla_h D_c{}^h\right)\right) - 2\,D_b{}^c\left(\nabla_f D_{ac} + \nabla_f D_{ca} + \frac{1}{3}g_{cf}\left(\nabla_j D_a{}^j\right) - \frac{2}{3}g_{af}\left(\nabla_j D_c{}^j\right) + \frac{1}{3}g_{ac}\left(\nabla_j D_f{}^j\right)\right) - 2\,D_a{}^c\left(\nabla_f D_{bc} + \nabla_f D_{cb} + \frac{1}{3}g_{cf}\left(\nabla_l D_b{}^l\right) - \frac{2}{3}g_{bf}\left(\nabla_l D_c{}^l\right) + \frac{1}{3}g_{bc}\left(\nabla_l D_f{}^l\right)\right)\right)$$

In[253]:=
```
ToCanonical /@ (ContractMetric /@ %)
```

Out[253]=
$$\frac{1}{3}\left(\nabla_a P_{bf}\right) + \frac{1}{3}\left(\nabla_b P_{af}\right) + \frac{1}{3}\left(\nabla_f P_{ab}\right) == \frac{1}{6}D^{cd}\,g_{bf}\left(\nabla_a D_{cd}\right) + \frac{1}{6}D^{cd}\,g_{af}\left(\nabla_b D_{cd}\right) + \frac{2}{9}D_f{}^c\,g_{ab}\left(\nabla_d D_c{}^d\right) + \frac{2}{9}D_b{}^c\,g_{af}\left(\nabla_d D_c{}^d\right) + \frac{2}{9}D_a{}^c\,g_{bf}\left(\nabla_d D_c{}^d\right) + \frac{1}{6}D^{cd}\,g_{ab}\left(\nabla_f D_{cd}\right)$$

In[254]:=
```
Simplification /@ %
```

Out[254]=
$$\frac{1}{3}\left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) == \frac{1}{18}\left(4\left(D_f{}^c\,g_{ab} + D_b{}^c\,g_{af} + D_a{}^c\,g_{bf}\right)\left(\nabla_d D_c{}^d\right) + 3\,D^{cd}\left(g_{bf}\left(\nabla_a D_{cd}\right) + g_{af}\left(\nabla_b D_{cd}\right) + g_{ab}\left(\nabla_f D_{cd}\right)\right)\right)$$

In[255]:=
```
Collect[#, g[-a_, -b_], Simplify] & /@ %
```

Out[255]=
$$\frac{1}{3}\left(\nabla_a P_{bf} + \nabla_b P_{af} + \nabla_f P_{ab}\right) == g_{bf}\left(\frac{1}{6}D^{cd}\left(\nabla_a D_{cd}\right) + \frac{2}{9}D_a{}^c\left(\nabla_d D_c{}^d\right)\right) + g_{af}\left(\frac{1}{6}D^{cd}\left(\nabla_b D_{cd}\right) + \frac{2}{9}D_b{}^c\left(\nabla_d D_c{}^d\right)\right) + g_{ab}\left(\frac{2}{9}D_f{}^c\left(\nabla_d D_c{}^d\right) + \frac{1}{6}D^{cd}\left(\nabla_f D_{cd}\right)\right)$$

©2003–2004 José M. Martín–García

By looking at this expression we realize that $X_{AB}\,\overline{X}_{A'B'}$ is the trace–free part of a Killing tensor if and only if

7. Example: the Dirac equation

7.1. The Dirac Bundle

We define the vector bundle which will contain the Dirac spinors. This is a direct sum of the vector bundles Spin and Spin†.

```
In[256]:=
DefVBundle[SpinD, M4, {Spin, Spin†}, {α, β, δ, ρ, λ, μ, ν}, Dagger -> Complex]

    ** DefVBundle: Defining vbundle SpinD.

    ** DefVBundle: Defining conjugated vbundle SpinD†.
     Assuming fixed anti-isomorphism between SpinD and SpinD†
```

Extension of the Levi–Civita covariant derivative to the bundle **SpinD**.

```
In[257]:=
DefCovD[CDd[-a], SpinD, ExtendedFrom → CD]

    ** DefCovD: Defining covariant derivative CDd[-a].

    ** DefTensor: Defining
     nonsymmetric AChristoffel tensor AChristoffelCDd[α, -b, -δ].

    ** DefTensor: Defining
     nonsymmetric AChristoffel tensor AChristoffelCDd†[α ↓, -b, -δ ↑].

    ** DefTensor: Defining FRiemann tensor
     FRiemannCDd[-a, -b, -δ, λ]. Antisymmetric only in the first pair.

    ** DefTensor: Defining FRiemann tensor
     FRiemannCDd†[-a, -b, -δ ↓, λ ↑]. Antisymmetric only in the first pair.
```

Definition of the Clifford algebra elements (gamma matrices)

```
In[258]:=
DefTensor[γ[-a, -α, β], M4, Dagger -> Complex]

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

    ** DefTensor: Defining tensor γ†[-a, -α ↓, β ↑].
```

The Clifford relation is

```
In[259]:=
CliffordRelation =
 γ[-a, -ρ, λ] γ[-b, -λ, δ] + γ[-b, -ρ, λ] γ[-a, -λ, δ] == -2 g[-a, -b] delta[-ρ, δ]

Out[259]=
γ_aρ^λ γ_bλ^δ + γ_aλ^δ γ_bρ^λ == -2 δ_ρ^δ g_ab
```

©2003–2004 José M. Martín–García

The representation of of the Clifford algebra on SpinD is reducible, Spin and Spin† being invariant subspaces. This entails the relations

```
In[260]:=
AutomaticRules[γ, MakeRule[{γ[-a, -A, B], 0}]]

        Rules {1} have been declared as DownValues for γ.

In[261]:=
AutomaticRules[γ, MakeRule[{γ[-a, -A†, B†], 0}]]

        Rules {1} have been declared as DownValues for γ.

In[262]:=
Unprotect@delta;

In[263]:=
AutomaticRules[delta, MakeRule[{delta[-A, B†], 0}, MetricOn → None]]

        Rules {1} have been declared as DownValues for delta.

In[264]:=
AutomaticRules[delta, MakeRule[{delta[-A†, B], 0}, MetricOn → None]]

        Rules {1} have been declared as DownValues for delta.

In[265]:=
Protect@delta;
```

Using previous relations, the Clifford relation can be split in two equations in the following way

```
In[266]:=
TraceProductDummy /@ CliffordRelation
```

Out[266]=

$$\gamma_{a\rho}{}^{A}\,\gamma_{bA}{}^{\delta}+\gamma_{a\rho}{}^{A'}\,\gamma_{bA'}{}^{\delta}+\gamma_{aA}{}^{\delta}\,\gamma_{b\rho}{}^{A}+\gamma_{aA'}{}^{\delta}\,\gamma_{b\rho}{}^{A'}==-2\,\delta_{\rho}{}^{\delta}\,g_{ab}$$

```
In[267]:=
Flatten@SplitIndex[%, -ρ → IndexList[-C, -C†]]
```

Out[267]=

$$\left\{\gamma_{aC}{}^{A'}\,\gamma_{bA'}{}^{\delta}+\gamma_{aA'}{}^{\delta}\,\gamma_{bC}{}^{A'}==-2\,\delta_{C}{}^{\delta}\,g_{ab},\ \gamma_{aC'}{}^{A}\,\gamma_{bA}{}^{\delta}+\gamma_{aA}{}^{\delta}\,\gamma_{bC'}{}^{A}==-2\,\delta_{C'}{}^{\delta}\,g_{ab}\right\}$$

```
In[268]:=
Union@Flatten@SplitIndex[%, δ → IndexList[D, D†]]
```

Out[268]=

$$\left\{\text{True},\ \gamma_{aC'}{}^{A}\,\gamma_{bA}{}^{D'}+\gamma_{aA}{}^{D'}\,\gamma_{bC'}{}^{A}==-2\,\delta_{C'}{}^{D'}\,g_{ab},\ \gamma_{aC}{}^{A'}\,\gamma_{bA'}{}^{D}+\gamma_{aA'}{}^{D}\,\gamma_{bC}{}^{A'}==-2\,\delta_{C}{}^{D}\,g_{ab}\right\}$$

```
In[269]:=
CliffordSplit = Rest[%]
```

Out[269]=

$$\left\{\gamma_{aC'}{}^{A}\,\gamma_{bA}{}^{D'}+\gamma_{aA}{}^{D'}\,\gamma_{bC'}{}^{A}==-2\,\delta_{C'}{}^{D'}\,g_{ab},\ \gamma_{aC}{}^{A'}\,\gamma_{bA'}{}^{D}+\gamma_{aA'}{}^{D}\,\gamma_{bC}{}^{A'}==-2\,\delta_{C}{}^{D}\,g_{ab}\right\}$$

©2003–2004 José M. Martín–García

We compare these equations with the following algebraic properties fulfilled by the soldering form

```
In[270]:=
{σ[-b, -A, D†] σ ŧ[-a, -C†, A] + σ[-a, -A, D†] σ ŧ[-b, -C†, A] ==
  -delta[-C†, D†] g[-a, -b],
 Dagger /@ (σ[-b, -A, D†] σ ŧ[-a, -C†, A] + σ[-a, -A, D†] σ ŧ[-b, -C†, A] ==
    -delta[-C†, D†] g[-a, -b])}
```

```
Out[270]=
{σa A C' σbA D' + σaA D' σb A C' == -δC' D' gab, σa D A' σbC A' + σaC A' σb D A' == -δC D gab}
```

Therefore, we conclude the rules

```
In[271]:=
{γ[-a, -C†, A] → Sqrt@2 σ ŧ[-a, -C†, A], γ[-b, -A, D†] → Sqrt@2 σ[-b, -A, D†]}
```

```
Out[271]=
{γaC' A → √2 σa A C', γbA D' → √2 σbA D'}
```

We turn these relations into automatic rules.

```
In[272]:=
AutomaticRules[γ, MakeRule[{γ[-a, -C†, A], Sqrt@2 σ ŧ[-a, -C†, A]}]]
```

```
Rules {1} have been declared as DownValues for γ.
```

```
In[273]:=
AutomaticRules[γ, MakeRule[{γ[-b, -A, D†], Sqrt@2 σ[-b, -A, D†]}]]
```

```
Rules {1} have been declared as DownValues for γ.
```

7.2. The Dirac equation

We define the Dirac spinor (fermion).

```
In[274]:=
DefTensor[Ψ[-α], M4, Dagger -> Complex]
```

```
** DefTensor: Defining tensor Ψ[-α].

** DefTensor: Defining tensor Ψ ŧ[-α ŧ].
```

We define the fermion mass

```
In[275]:=
DefConstantSymbol[M]
```

```
** DefConstantSymbol: Defining constant symbol M.
```

©2003–2004 José M. Martín–García

This is the Dirac equation as it is usually presented in textbooks.

```
In[276]:=
DiracEquation = I γ[a, -α, β] CDd[-a]@Ψ[-β] == -Sqrt@2 M Ψ[-α]
```

```
Out[276]=
i γ^a_α^β (∇_a Ψ_β) == -√2 M Ψ_α
```

We wish to write this equation in terms of spinors defined in `Spin` and `Spin†`.

```
In[277]:=
TraceProductDummy /@ DiracEquation
```

```
Out[277]=
i γ^a_α^A (∇_a Ψ_A) + i γ^a_α^A' (∇_a Ψ_A') == -√2 M Ψ_α
```

```
In[278]:=
Flatten@SplitIndex[%, IndexList[-α] → IndexList[-C, -C†]]
```

```
Out[278]=
{i √2 σ^a_C^A' (∇_a Ψ_A') == -√2 M Ψ_C, i √2 σ^aA_C' (∇_a Ψ_A) == -√2 M Ψ_C'}
```

The covariant derivative `CDd` acts now on quantities belonging to `Spin` and `Spin†`. Therefore it must be replaced by `CDe`.

```
In[279]:=
% /. CDd → CDe
```

```
Out[279]=
{i √2 σ^a_C^A' (∇_a Ψ_A') == -√2 M Ψ_C, i √2 σ^aA_C' (∇_a Ψ_A) == -√2 M Ψ_C'}
```

```
In[280]:=
Map[ContractSolderingForm, %, {2}]
```

```
Out[280]=
{i √2 (∇_C^A' Ψ_A') == -√2 M Ψ_C, i √2 (∇^A_C' Ψ_A) == -√2 M Ψ_C'}
```

We cancel the factor $\sqrt{2}$ on each side.

```
In[281]:=
Map[Times[#, 1 / Sqrt[2]] &, %, {2}]
```

```
Out[281]=
{i (∇_C^A' Ψ_A') == -M Ψ_C, i (∇^A_C' Ψ_A) == -M Ψ_C'}
```

```
In[282]:=
SpinDiracEquation = %;
```

It is convenient to introduce two new spinors to represent Ψ_C and $\Psi_{C'}$

```
In[283]:=
DefSpinor[ϕ[-A], M4]
```

```
** DefTensor: Defining tensor ϕ[-A].

** DefTensor: Defining tensor ϕ†[-A†].
```

©2003–2004 José M. Martín–García

In[284]:=
```
PrintAs[ϕ ¶] ^= "$\overline{\phi}$";
```

In[285]:=
```
DefSpinor[χ[-A], M4]
```

```
** DefTensor: Defining tensor χ[-A].

** DefTensor: Defining tensor χ ¶[-A†].
```

In[286]:=
```
PrintAs[χ ¶] ^= "$\overline{\chi}$";
```

By definition

In[287]:=
```
{Ψ[-A] == ϕ[-A], Ψ[-A†] == χ ¶[-A†]}
```

Out[287]=
$$\left\{\Psi_{A} == \phi_{A},\ \Psi_{A'} == \overline{\chi}_{A'}\right\}$$

We turn these definitions into automatic rules.

In[288]:=
```
AutomaticRules[Ψ, MakeRule[{Ψ[-A], ϕ[-A]}]]
```

```
Rules {1} have been declared as DownValues for Ψ.
```

In[289]:=
```
AutomaticRules[Ψ, MakeRule[{Ψ[-A†], χ ¶[-A†]}]]
```

```
Rules {1} have been declared as DownValues for Ψ.
```

Hence

In[290]:=
```
SpinDiracEquation
```

Out[290]=
$$\left\{i\ \left(\nabla_{C}{}^{A'}\ \overline{\chi}_{A'}\right) == -M\,\phi_{C},\ i\ \left(\nabla^{A}{}_{C'}\ \phi_{A}\right) == -M\,\overline{\chi}_{C'}\right\}$$

Compare this with eq. (4.4.66) of Penrose & Rindler Vol. 1.

7.3. Traces of products of gamma matrices

Traces of an odd number of gamma's are zero:

In[291]:=
```
γ[a, -α, β] γ[b, -β, δ] γ[c, -δ, α]
```

Out[291]=
$$\gamma^{a}{}_{\alpha}{}^{\beta}\ \gamma^{b}{}_{\beta}{}^{\delta}\ \gamma^{c}{}_{\delta}{}^{\alpha}$$

©2003–2004 José M. Martín–García

In[292]:=
```
TraceProductDummy@%
```

Out[292]=
```
0
```

But not traces of an even number :

In[293]:=
```
γ[a, -α, β] γ[b, -β, δ] γ[c, -δ, ρ] γ[d, -ρ, α]
```

Out[293]=
$$\gamma^{a}{}_{\alpha}{}^{\beta}\,\gamma^{b}{}_{\beta}{}^{\delta}\,\gamma^{c}{}_{\delta}{}^{\rho}\,\gamma^{d}{}_{\rho}{}^{\alpha}$$

In[294]:=
```
TraceProductDummy@%
```

Out[294]=
$$4\,\sigma^{a}{}_{A}{}^{A'}\,\sigma^{bB}{}_{A},\,\sigma^{c}{}_{B}{}^{B'}\,\sigma^{dA}{}_{B},+4\,\sigma^{aA}{}_{A'},\,\sigma^{b}{}_{A}{}^{B'}\,\sigma^{cB}{}_{B'},\,\sigma^{d}{}_{B}{}^{A'}$$

In[295]:=
```
ContractSolderingForm@%
```

Out[295]=
$$-4\,g^{fc}\,Gg^{ab}{}_{f}{}^{d}-4\,g^{fc}\,Gg^{ad}{}_{f}{}^{b}$$

In[296]:=
```
% /. TetraRule[g]
```

Out[296]=
$$-4\left(\frac{i\in g^{ab}{}_{f}{}^{d}}{2}-\frac{1}{2}\delta_{f}{}^{d}\,g^{ab}-\frac{1}{2}\delta_{f}{}^{b}\,g^{ad}+\frac{1}{2}\delta_{f}{}^{a}\,g^{bd}\right)g^{fc}-$$
$$4\left(\frac{i\in g^{ad}{}_{f}{}^{b}}{2}-\frac{1}{2}\delta_{f}{}^{d}\,g^{ab}-\frac{1}{2}\delta_{f}{}^{b}\,g^{ad}+\frac{1}{2}\delta_{f}{}^{a}\,g^{db}\right)g^{fc}$$

In[297]:=
```
ToCanonical@ContractMetric@%
```

Out[297]=
$$4\,g^{ad}\,g^{bc}-4\,g^{ac}\,g^{bd}+4\,g^{ab}\,g^{cd}$$

■ Notes

In[298]:=
```
MaxMemoryUsed[]
```

Out[298]=
```
77 124 896
```

In[299]:=
```
TimeUsed[]
```

Out[299]=
```
11.1393
```

Note: For further information about `Spinors`, and to be kept informed about new releases, you may contact the

©2003–2004 José M. Martín–García

authors electronically at algar@mai.liu.se, garcia@iap.fr. This is `SpinorsDoc.nb`, the docfile of `Spinors`, currently in version 0.9.2.

In[300]:=
```
?xAct`Spinors`*
```

▼ **xAct`Spinors`**

BoxToCovD	Omega	TangentBundleOfSolderingForm
BoxToCurvature	Phi	TensorOfSpinor
Chi	Psi	UndefSpinCovD
ContractSolderingForm	PutSolderingForm	UndefSpinor
Decomposition	SeparateSolderingForm	UndefSpinStructure
DefSpinCovD	Sigma	VBundleOfSolderingForm
DefSpinor	SolderingFormOfSpinCovD	$ChiSign
DefSpinorOfTensor	SolderingFormOfVBundle	$LambdaSign
DefSpinStructure	SpinCovDsOfSolderingForm	$OmegaSign
DefTensorOfSpinor	SpinMetricOfSolderingForm	$PhiSign
Disclaimer	SpinorMark	$PsiSign
IrreducibleDecomposition	SpinorOfTensor	$SolderingForms
Lambda	SpinorPrefix	$Version

©2003–2004 José M. Martín–García