xActを用いたテンソル解析の基礎

Mathematicaの数式演算の能力を人間能力では大変重労働になるテンソルの計算に活かしたいと思う。
これをアシストするパッケージは古くからMathtensorがあり、本サイトでもすでに紹介している。
これより、安価で新しいMathematicaに対応したものにTensorialがあり2014年初でそのバージョンはVer5である。
これはMathematicaVer6を対象にしていて最新のMathematicaで利用するためには少々の書き換えが必要になる。

Tensorialの古いバージョンはウルフラムサイトに無償でアーカイブされている。
最新のTensorial情報や購入は下記のサイトを見るとよい。

http : // www.jfgouyet.fr / tcm / tcm.html

無償のオープンウェアであるxActもテンソルを扱える。最新のものはMathmatica9やWindows8.1にも対応し、相性もいい。MathTensorに近い操作性があり、関数も多いが多様体から定義し、バンドルやテンソルを構成していくなど数学的な体系を用いている。しかし、コマンド体系が複雑で、知識が必要になる。

今回はこのxActを紹介する。詳しくは以下のサイトを参考にするとよい。本稿は未完成である、今後修正、加筆される必要がある。

http : // www.xact.es /

xActのインストール

インストールするにはサイトからダウンロードしたファイルを隠しフォルダににになっているDocument and Settingのショートカットから

ユーザー\Application\Mathematicaのフォルダの中のApplicationsフォルダの中に解凍し、ダウンロードファイルをそのまま入れる。
ロードするにはMathematicaを起動し、ノートブックを開き、MathematicaVer9では次のようにコマンドをいれる。
まず順に実行しながら基本的なコマンドを学ぼう。

```
<< xAct`xTensor`
```

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

Package xAct`xPerm`  version 1.2.0, {2013, 1, 27}

CopyRight (C) 2003-2013, Jose M. Martin-Garcia, under the General Public License.

Connecting to external cygwin executable...

Connection established.

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

Package xAct`xTensor`  version 1.0.5, {2013, 1, 30}

CopyRight (C) 2002-2013, 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.

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

さらに基底を具体的に操作して演算する場合は拡張パッケージのxCobaをロードしておく必要がある。

Tensorialとことなり、このパッケージをロードしておかないとDefBasis[]、DefVBundle[]などのコマンドが使えない。

xCobaのロードは次のようにおこなう。xCobaだけでもxActは自動ロードされるのでTensor解析をする場合はメモリが十分であれば
これをはじめにロードしておけばいい。以下のようにタイプすると各パッケージの内容とバージョンが返される。

```
<< xAct`xCoba`
```

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

Package xAct`xPerm`  version 1.2.0, {2013, 1, 27}

CopyRight (C) 2003-2013, Jose M. Martin-Garcia, under the General Public License.

Connecting to external cygwin executable...

Connection established.

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

Package xAct`xTensor`  version 1.0.5, {2013, 1, 30}

CopyRight (C) 2002-2013, Jose M. Martin-Garcia, under the General Public License.

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

Package xAct`xCoba`  version 0.8.0, {2013, 1, 30}

CopyRight (C) 2005-2013, David Yllanes 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.

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

１．xActの基本

多様体の定義

ｘActでははじめに多様体を定義し、この上にテンソルを展開することが特徴である。

次のコマンドで多様体を次元と共に定義する。この次元がテンソルにも引き継がれる。

DefManifold[多様体名**,**次元***{List}]***

次のコマンドで多様体を定義する。例えば3次元のM3多様体を定義しよう。

```
DefManifold[M3, 3, {a, b, c, d, e, f}]
```

```
** DefManifold: Defining manifold M3.

** DefVBundle: Defining vbundle TangentM3.
```

テンソルの定義

DefManifoldで多様体を定義したら次にDefTensorでテンソルを定義する。消去する場合はUndefTensorを使う。

DefTensor[,テンソル**,(**対称性***)]***

UndeffTensor[,テンソル記号***]***

例えば次で対称テンソルｖと反対称テンソルFをM3上に定義する。

ｘActでは下付添え字にはーをつけて表わす。

```
DefTensor[v[a], M3]
DefTensor[F[-a, -b], M3, Antisymmetric[{-a, -b}]]
```

```
** DefTensor: Defining tensor v[a].

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

これだけでテンソルの演算が可能である。

Simplification[式***]***、***InputForm[***式***]***

次のようにテンソル演算を満たすことがわかる。Simplification[式]で簡約でき、InputForm[式]で入力形式を確認できる。

```
F[-a, -b] v[b]
F[a, b] v[-a] v[-b]
Simplification[%]
InputForm[%%]
```

$$F_{ab}\ v^{b}$$

$$F^{ab}\ v_{a}\ v_{b}$$

$$0$$

```
F[a, b]*v[-a]*v[-b]
```

ToCanonical[式]

この場合Fは反対称テンソルと定義しているのでToCanonical[]でこれを確認することができる。

```
F[-b, -a]
ToCanonical[%]
```

$$F_{ba}$$

$$-F_{ab}$$

テンソルの演算

定義したテンソルに代入する方法を見てみよう。まず新たに次のテンソルを定義する。

```
DefTensor[w[a], M3]
DefTensor[T[a, b], M3]
```

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

次のように引数を指定し、代入する。

```
w[a_] = T[a, b] v[-b]
```

$$T^{ab}\ v_{b}$$

以後のこのwが次のように使える。

```
w[b]
w[a] - w[a] // Simplification
w[a] w[-a]
ToCanonical[%]
```

$$T^{bb}\ v_{b}$$

$$0$$

$$T_{a}^{\ b}\ T^{ab}\ v_{b}^{\ 2}$$

$$(v_{b})^{2}\ T_{a}^{\ b}\ T^{ab}$$

Mathematica の定義式を利用すると次のように内部変数が表示されてしまう。

これを解決にするにはScreenDollarIndices[]を使う。

```
w[a_] := Module[{b}, T[a, b] v[-b]]
```

```
w[b]
w[a] - w[a] // Simplification
w[a] w[-a]
ToCanonical[%]
% // ScreenDollarIndices
```

$$T^{bb\$1942}\ v_{b\$1942}$$

$$0$$

$$T_a{}^{b\$1946}\ T^{ab\$1945}\ v_{b\$1945}\ v_{b\$1946}$$

$$T_a{}^{b\$1945}\ T^{ab\$1946}\ v_{b\$1945}\ v_{b\$1946}$$

$$T_a{}^{b}\ T^{ac}\ v_b\ v_c$$

Dir[テンソル]

上付、下付の指定は次のようにDir[]を使って直接引数に入れ、行うことができる。

```
dirvup = Dir[v[d]];
dirvdown = Dir[v[-d]];

w[a_] := T[a]

{w[a], w[-a], w[dirvdown], w[dirvup], w[A], w[-A]}
```

$$\left\{T^a,\ T_a,\ T^v,\ T_v,\ T^A,\ T_A\right\}$$

?DownIndexQ

?UpIndexQ

これは次のように添え字により分類することに利用できる。

比較に利用できるQコマンドが次のようにたくさん用意されている。

AIndexQ	***Detect abstract indices***
BIndexQ	***Detect basis indices***
CIndexQ	***Detect component indices***
DIndexQ	***Detect directional indices***
LIndexQ	***Detect label indices***
GIndexQ	***Detect all generalized indices, but not patterns***
ABIndexQ	***Detect contractible indices (abstract or basis indices)***
BCIndexQ	***Detect indices associated to a basis or chart (basis or component indices)***
CDIndexQ	***Detect indices representing a direction (component or directional indices)***
PIndexQ	***Detect pattern indices***

以前の定義を消す場合は次のようにピリオドを代入する。

```
w[a_] =.
w[a_?DownIndexQ] := T[a]
{w[a], w[-a], w[dirvdown], w[dirvup], w[A], w[-A]}
```

$$\left\{w^a,\ T_a,\ w^v,\ T_v,\ w^A,\ T_A\right\}$$

下付だけが置き換わっている。

指定したシンボルだけを置き換えたい時は次のようにする。

```
w[a_?DownIndexQ] =.
w[a_?AIndexQ] := T[a]
{w[a], w[-a], w[dirvup], w[dirvdown], w[A], w[-A]}
```

$$\left\{T^{a},\ T_{a},\ w_{v},\ w^{v},\ w^{A},\ w_{A}\right\}$$

大文字、小文字を区別させない場合は次のようにする。

```
w[a_?AIndexQ] =.
w[-a_Symbol] := T[-a]
{w[a], w[-a], w[dirvup], w[dirvdown], w[A], w[-A]}
```

$$\left\{w^{a},\ T_{a},\ w_{v},\ w^{v},\ w^{A},\ T_{A}\right\}$$

添え字a、Aを持つ下付のみが置き換わった。

座標系、基底、バンドルの定義

複素バンドルを多様体上に定義できるのはxCobaの特徴である。次のように多様体Mの内部Cに2次元複素バンドルを定義する。
はじめにDefManifold[]を実行しておく必要がある。

```
DefVBundle[InnerC, M3, 2, {A, B, G}, Dagger → Complex]
```

```
** DefVBundle: Defining vbundle InnerC.

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

*DefBasis[*座標名,対象,リスト,*option]*

xCobaがロードされていれば次のように座標系を定義できる。次の例では極座標と、カーテシアン座標、複素座標をM3上に定義する。
区別するために極座標には赤、カーテシアンには青色、複素座標に緑色をつける。はじめにDefManifold[]を実行しておく必要がある。

```
DefBasis[polar, TangentM3, {0, 1, 2}]
DefBasis[cartesian, TangentM3, {0, 1, 2}, BasisColor → RGBColor[0, 0, 1]]
DefBasis[complex, InnerC, {4, 5}, Dagger → Complex, BasisColor → RGBColor[0, 1, 0]]
```

```
** DefBasis: Defining basis polar.

** DefCovD: Defining parallel derivative PDpolar[-a].

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

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

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

** DefTensor: Defining vanishing Ricci tensor RicciPDpolar[-a, -b].

** DefTensor: Defining antisymmetric +1 density etaUppolar[a, b, c].

** DefTensor: Defining antisymmetric -1 density etaDownpolar[-a, -b, -c].

** DefBasis: Defining basis cartesian.

** DefCovD: Defining parallel derivative PDcartesian[-a].

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

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

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

** DefTensor: Defining vanishing Ricci tensor RicciPDcartesian[-a, -b].

** DefTensor: Defining antisymmetric +1 density etaUpcartesian[a, b, c].

** DefTensor: Defining antisymmetric -1 density etaDowncartesian[-a, -b, -c].

** DefBasis: Defining basis complex.

** DefCovD: Defining parallel derivative PDcomplex[-a].

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

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

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

** DefTensor: Defining vanishing Ricci tensor RicciPDcomplex[-a, -b].

** DefTensor: Defining
 nonsymmetric AChristoffel tensor AChristoffelPDcomplex[A, -b, -G].

** DefTensor: Defining nonsymmetric AChristoffel tensor
 AChristoffelPDcomplex†[A†, -b, -G†].

** DefTensor: Defining vanishing FRiemann tensor FRiemannPDcomplex[-a, -b, -G, G1].

** DefTensor: Defining vanishing FRiemann tensor FRiemannPDcomplex†[-a, -b, -G†, G1†].

** DefBasis: Defining basis complex†.

** DefTensor: Defining antisymmetric +1 density etaUpcomplex[A, B].

** DefTensor: Defining antisymmetric +1 density etaUpcomplex†[A†, B†].

** DefTensor: Defining antisymmetric -1 density etaDowncomplex[-A, -B].

** DefTensor: Defining antisymmetric -1 density etaDowncomplex†[-A†, -B†].
```

実行すると定義リストが大量に表示される。表示されたテンソルやフォームは利用できるのでチェックしておく。

計量テンソルの定義

*DefMetric[*対称性*Flag,*テンソル*,*微分*,*表記*]*

ではテンソルで重要な計量テンソルgの定義を見よう。この計量テンソルの定義の仕方により、テンソル解析ソフトの特徴がよくわかる。

xActでは次のようにおこなう。Tensorialは単純なマトリックスとしての定義であるがxActでははじめに多様体を定義している

のでその計量と付属してくるリーマン曲率やリッチテンソルなども同時に定義される。SymbolOfCovD->でMathTensorのように記号をどう表現するか指定ができる。

```
DefMetric[-1, g[-a, -b], Cd, SymbolOfCovD -> {"|", "∇"}]
```

```
** DefTensor: Defining symmetric metric tensor g[-a, -b].
** DefTensor: Defining antisymmetric tensor epsilong[-a, -b, -c].
** 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 vanishing 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 3
** DefCovD:  Computing RicciToTFRicci for dim 3
** DefCovD:  Computing RicciToEinsteinRules for dim 3
** DefTensor: Defining weight +2 density Detg[]. Determinant.
```

出力結果を見ると非常に多くのテンソルや微分が定義されたことがわかる。

```
RiemannCd[-a, b, -b, -c]
```

$$-\mathrm{R}[\nabla]_{ac}$$

添え字の上げ下げも次のように確認できる。Simplification[]で縮約される。

計量として定義すれば次のように直交関係が満たされる。δがすぐに利用できるのはxActの利点である。

```
v[-a, -b] g[b, c]
% // Simplification
g[-a, -c] g[c, d]
```

$$g^{bc}\ v_{ab}$$

$$v_a{}^c$$

$$\delta_a{}^d$$

定義の取り消し

定義したものの取り消しは次のようにUndef..を使う。

UndefMetric[計量]

UndefTensor[テンソル]

UndefManifold[多様体]

UndefBasis[基底]

例えば先のテンソルを消去するには次のように1つづつおこなう。

```
UndefTensor[v]
UndefTensor[F]
```

```
** UndefTensor: Undefined tensor v

** UndefTensor: Undefined tensor F
```

成分の表示と代入

MetricInBasis[計量テンソル,座標系,代入リスト]

Tensorialに比べると多様な定義ができる反面、成分の指定はやや面倒である。MetricInBasis[]を使い、定義したgにここではTableコマンドを使って代入する。次元が一致していれば行列をそのまま代入することもできる。座標系も指定しないといけない。

```
MetricInBasis[g, -cartesian, Table[i + j, {i, 3}, {j, 3}]]
% // MatrixForm
```

```
Added independent rule g[0, 0] → 2 for tensor g

Added independent rule g[0, 1] → 3 for tensor g

Added independent rule g[0, 2] → 4 for tensor g

Added dependent rule g[1, 0] → g[0, 1] for tensor g

Added independent rule g[1, 1] → 4 for tensor g

Added independent rule g[1, 2] → 5 for tensor g

Added dependent rule g[2, 0] → g[0, 2] for tensor g

Added dependent rule g[2, 1] → g[1, 2] for tensor g

Added independent rule g[2, 2] → 6 for tensor g
```

$$\{\{g_{00} \to 2,\ g_{01} \to 3,\ g_{02} \to 4\},\ \{g_{10} \to 3,\ g_{11} \to 4,\ g_{12} \to 5\},\ \{g_{20} \to 4,\ g_{21} \to 5,\ g_{22} \to 6\}\}$$

$$\begin{pmatrix} g_{00} \to 2 & g_{01} \to 3 & g_{02} \to 4 \\ g_{10} \to 3 & g_{11} \to 4 & g_{12} \to 5 \\ g_{20} \to 4 & g_{21} \to 5 & g_{22} \to 6 \end{pmatrix}$$

ただし、このように置換リストが作られるだけであるので注意する。

DefTensor[テンソル[{引数、座標系}],多様体,対称性]

新たに次の対称、反対称のテンソル、ベクトルを定義しよう。

```
UndefTensor[Ts];
UndefTensor[Ta];
DefTensor[Ta[-a, -b], M3, Antisymmetric[{-a, -b}]]
DefTensor[Ts[-a, -b], M3]
DefTensor[v[-a], M3]
```

```
UndefTensor: Unknown tensor Ts.

UndefTensor: Unknown tensor Ta.
```

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

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

** DefTensor: Defining tensor v[-a].
```

ComponentArray[*テンソル*[{*引数、座標系*}]

先に定義した座標系を利用し、成分表示させる。

成分、行列表示をする。座標系 -cartesianオプションは青字になる。

```
ComponentArray[Ta[{-a, -cartesian}, {-b, -cartesian}]]
Simplification[%] // MatrixForm
ComponentArray[Ts[{-a, -cartesian}, {-b, -cartesian}]]
Simplification[%] // MatrixForm
```

$$\{\{Ta_{00}, Ta_{01}, Ta_{02}\}, \{Ta_{10}, Ta_{11}, Ta_{12}\}, \{Ta_{20}, Ta_{21}, Ta_{22}\}\}$$

$$\begin{pmatrix} 0 & Ta_{01} & Ta_{02} \\ -Ta_{01} & 0 & Ta_{12} \\ -Ta_{02} & -Ta_{12} & 0 \end{pmatrix}$$

$$\{\{Ts_{00}, Ts_{01}, Ts_{02}\}, \{Ts_{10}, Ts_{11}, Ts_{12}\}, \{Ts_{20}, Ts_{21}, Ts_{22}\}\}$$

$$\begin{pmatrix} Ts_{00} & Ts_{01} & Ts_{02} \\ Ts_{10} & Ts_{11} & Ts_{12} \\ Ts_{20} & Ts_{21} & Ts_{22} \end{pmatrix}$$

さて、これら成分に任意の関数が代入できなければ応用がきかない。まず、次の行列を定義して、これをテンソルの成分にする方法を紹介しよう。入力には次のコマンドが用意されている。

ComponentValue[*テンソル*[{*引数、座標系*},*代入配列*]

AllComponentValues[*テンソル*[{*引数、座標系*},*代入配列*]

例えば先に定義したTsの[1,1]成分に代入する。ただし、-座標系で下付で指定される。

xActは座標系とセットで組み合わせないといけない。座標系の符号で上付、下付を区別するのである。

このあたりやや煩雑で上げ下げをどちらでやっているか混乱する。

```
ComponentValue[Ts[{1, -cartesian}, {1, -cartesian}], 2 t]
ComponentValue[Ts[{1, cartesian}, {1, cartesian}], 6 t]
ComponentValue[Ts[{1, -cartesian}, {1, cartesian}], 3]
```

```
Added independent rule Ts_11 → 2 t for tensor Ts
```

$$Ts_{11} \to 2\,t$$

```
Added independent rule Ts^11 → 6 t for tensor Ts
```

$$Ts^{11} \to 6\,t$$

```
Added independent rule Ts_1^1 → 3 for tensor Ts
```

$$Ts_1^{\ 1} \to 3$$

このあたりはTensorialの方が軽快である。もう少し簡単にできるいいと思う。さらに注意すべきは結果に表示されるように代入というより置き換えリストFoldRulesを作成しているだけである。よって結果を表示する時もこの置き換えリストを指定しないといけない。

そのためには次のように入力する。実際の行列をTmとして代入させている。

```
ComponentArray[Ts[{-a, -cartesian}, {-b, -cartesian}]];
Tm = Simplification[%] /. TensorValues[Ts];
% // MatrixForm
ComponentArray[Ts[{-a, -cartesian}, {b, -cartesian}]];
Tm = Simplification[%] /. TensorValues[Ts];
% // MatrixForm
```

$$\begin{pmatrix} Ts_{00} & Ts_{01} & Ts_{02} \\ Ts_{10} & 2\,t & Ts_{12} \\ Ts_{20} & Ts_{21} & Ts_{22} \end{pmatrix}$$

$$\begin{pmatrix} Ts_0{}^0 & Ts_0{}^1 & Ts_0{}^2 \\ Ts_1{}^0 & 3 & Ts_1{}^2 \\ Ts_2{}^0 & Ts_2{}^1 & Ts_2{}^2 \end{pmatrix}$$

たしかに[1,1]成分だけが置き換わったのがわかる。

リストからそのまま成分をつくれれば便利である。この方法を紹介しよう。

反対称なテンソルTaにも次の行列を定義して代入し、その結果を確認してみよう。

その前に先に利用した成分は次のコマンドで消しておく。

ComponentArray[]と組み合わせると行列をそのまま代入できる。

AllComponentValues[]を使うと全ての成分が決められる。

DeleteTensorValues[テンソル]

```
DeleteTensorValues[Ts];
values = {{1, 20 t, 30 t}, {-2 t, 2, 40 t}, {-3 t, -4 t, 3}};
ComponentValue[ComponentArray[Ts[{a, -cartesian}, {b, -cartesian}]], values]
AllComponentValues[Ta[{a, -cartesian}, {-b, -cartesian}], values]
```

```
Deleted values for tensor Ts, derivatives {} and bases {{-cartesian, cartesian}}.
Deleted values for tensor Ts, derivatives {} and bases {{cartesian, cartesian}}.
Deleted values for tensor Ts, derivatives {} and bases {{-cartesian, -cartesian}}.
Added independent rule Ts^00 → 1 for tensor Ts
Added independent rule Ts^01 → 20 t for tensor Ts
Added independent rule Ts^02 → 30 t for tensor Ts
Added independent rule Ts^10 → -2 t for tensor Ts
Added independent rule Ts^11 → 2 for tensor Ts
Added independent rule Ts^12 → 40 t for tensor Ts
Added independent rule Ts^20 → -3 t for tensor Ts
Added independent rule Ts^21 → -4 t for tensor Ts
Added independent rule Ts^22 → 3 for tensor Ts
{{Ts^00 → 1, Ts^01 → 20 t, Ts^02 → 30 t},
 {Ts^10 → -2 t, Ts^11 → 2, Ts^12 → 40 t}, {Ts^20 → -3 t, Ts^21 → -4 t, Ts^22 → 3}}
```

```
Added independent rule Ta^0_0 → 1 for tensor Ta

Added independent rule Ta^0_1 → 20 t for tensor Ta

Added independent rule Ta^0_2 → 30 t for tensor Ta

Added dependent rule Ta^1_0 → - Ta_0^1 for tensor Ta

Added independent rule Ta_0^1 → 2 t for tensor Ta

Added independent rule Ta^1_1 → 2 for tensor Ta

Added independent rule Ta^1_2 → 40 t for tensor Ta

Added dependent rule Ta^2_0 → - Ta_0^2 for tensor Ta

Added independent rule Ta_0^2 → 3 t for tensor Ta

Added dependent rule Ta^2_1 → - Ta_1^2 for tensor Ta

Added independent rule Ta_1^2 → 4 t for tensor Ta

Added independent rule Ta^2_2 → 3 for tensor Ta

Added dependent rule Ta_00 → 0 for tensor Ta

Added independent rule Ta_01 → Ta_01 for tensor Ta

Added independent rule Ta_02 → Ta_02 for tensor Ta

Added dependent rule Ta_10 → - Ta_01 for tensor Ta

Added dependent rule Ta_11 → 0 for tensor Ta

Added independent rule Ta_12 → Ta_12 for tensor Ta

Added dependent rule Ta_20 → - Ta_02 for tensor Ta

Added dependent rule Ta_21 → - Ta_12 for tensor Ta

Added dependent rule Ta_22 → 0 for tensor Ta

  FoldedRule[
   {Ta_00 → 0, Ta_10 → - Ta_01, Ta_11 → 0, Ta_20 → - Ta_02, Ta_21 → - Ta_12, Ta_22 → 0},
   {Ta_01 → Ta_01, Ta_02 → Ta_02, Ta_12 → Ta_12}]
```

結果を確認するとComponentArray[]では代入した成分以外は代入されないがAllComponentValues[]では全て表現することができる。
少々面倒ではあるがこうした細かさはxActの特徴である。

```
ComponentArray[Ts[{a, -cartesian}, {b, -cartesian}]]
Tm = Simplification[%] /. TensorValues[Ts];
% // MatrixForm
ComponentArray[Ts[{a, -cartesian}, {-b, -cartesian}]];
Tm = Simplification[%] /. TensorValues[Ts];
% // MatrixForm
ComponentArray[Ta[{-a, -cartesian}, {-b, -cartesian}]]
Tm = Simplification[%] /. TensorValues[Ta];
% // MatrixForm
ComponentArray[Ta[{-a, -cartesian}, {b, -cartesian}]]
Tm = Simplification[%] /. TensorValues[Ta];
% // MatrixForm
```

$$\{\{Ts^{00}, Ts^{01}, Ts^{02}\}, \{Ts^{10}, Ts^{11}, Ts^{12}\}, \{Ts^{20}, Ts^{21}, Ts^{22}\}\}$$

$$\begin{pmatrix} 1 & 20\,t & 30\,t \\ -2\,t & 2 & 40\,t \\ -3\,t & -4\,t & 3 \end{pmatrix}$$

$$\begin{pmatrix} Ts^0{}_0 & Ts^0{}_1 & Ts^0{}_2 \\ Ts^1{}_0 & Ts^1{}_1 & Ts^1{}_2 \\ Ts^2{}_0 & Ts^2{}_1 & Ts^2{}_2 \end{pmatrix}$$

$$\{\{Ta_{00}, Ta_{01}, Ta_{02}\}, \{Ta_{10}, Ta_{11}, Ta_{12}\}, \{Ta_{20}, Ta_{21}, Ta_{22}\}\}$$

$$\begin{pmatrix} 0 & Ta_{01} & Ta_{02} \\ -Ta_{01} & 0 & Ta_{12} \\ -Ta_{02} & -Ta_{12} & 0 \end{pmatrix}$$

$$\{\{Ta_0{}^0, Ta_0{}^1, Ta_0{}^2\}, \{Ta_1{}^0, Ta_1{}^1, Ta_1{}^2\}, \{Ta_2{}^0, Ta_2{}^1, Ta_2{}^2\}\}$$

$$\begin{pmatrix} -1 & 2\,t & 3\,t \\ -20\,t & -2 & 4\,t \\ -30\,t & -40\,t & -3 \end{pmatrix}$$

xActは数式処理をするためにそのまま値を代入することを避け上のように置換リストであるFoldRulesをつくる。
この置き換えリストで先のAllComponentValues[]を置換して表示すればいい。次では結果をTmに代入させている。

$$\begin{pmatrix} 0 & -5\,t & -5\,t \\ 5\,t & 0 & -15\,t \\ 5\,t & 15\,t & 0 \end{pmatrix}$$

xActでは次のように直接Tsを成分演算できない。置き換えたTmであれば手軽に演算したり、参照できる。

```
PD[-a][Ta[-b, -c]] /. TensorValues[Ta]
Inverse[Tm];
Simplify[%] // MatrixForm
Tm[[1, 2]]
```

$$\partial_{a} Ta_{bc}$$

$$\begin{pmatrix} \frac{3+80\,t^2}{-3-230\,t^2+1080\,t^3} & \frac{3\,(1-20\,t)\,t}{-3-230\,t^2+1080\,t^3} & \frac{t\,(3+4\,t)}{-3-230\,t^2+1080\,t^3} \\ -\frac{30\,t\,(1+2\,t)}{-3-230\,t^2+1080\,t^3} & \frac{3+90\,t^2}{-6-460\,t^2+2160\,t^3} & \frac{2\,(1-15\,t)\,t}{-3-230\,t^2+1080\,t^3} \\ \frac{10\,t\,(-3+40\,t)}{-3-230\,t^2+1080\,t^3} & -\frac{10\,t\,(2+3\,t)}{-3-230\,t^2+1080\,t^3} & \frac{1+20\,t^2}{-3-230\,t^2+1080\,t^3} \end{pmatrix}$$

$$2\,t$$

しかしこれでは単に行列演算である。

xActには豊富な演算群がある。その分少々面倒なところもあるがその方法を紹介する。

2．テンソル解析

この章からは実際にテンソル解析にxActを利用するための基礎を学ぼう<<xAct`xCoba`が実行され、xCobaがロードした直後であるとして再び多様体の定義からおこなう。続いてバンドル、テンソル、基底を次のように定義する。

```
DefManifold[M3, 3, {a, b, c, d, e, f}]
DefVBundle[InnerC, M3, 2, {A, B, C, D, F, G}, Dagger → Complex]
DefTensor[v[a], M3]
DefTensor[T[-a, -b], M3, Antisymmetric[{-a, -b}]]
DefTensor[R[-A, -B, -C, -D], M3, RiemannSymmetric[{1, 2, 3, 4}], Dagger → Complex]
DefTensor[u[-A], M3, Dagger → Complex]
DefBasis[polar, TangentM3, {0, 1, 2}]
DefBasis[cartesian, TangentM3, {0, 1, 2}, BasisColor → RGBColor[0, 0, 1]]
DefBasis[complex, InnerC, {4, 5}, Dagger → Complex, BasisColor → RGBColor[0, 1, 0]]
```

```
** DefManifold: Defining manifold M3.

** DefVBundle: Defining vbundle TangentM3.

** DefVBundle: Defining vbundle InnerC.

Validate Scyamp oi:tSaylstemnameCisoverloaudaendabstraicntd e

Validate Scyamp oi:tSaylstemnameDisoverloaudaendabstraicntd e

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

** DefTensor: Defining tensor v[a].

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

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

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

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

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

** DefBasis: Defining basis polar.

** DefCovD: Defining parallel derivative PDpolar[-a].

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

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

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

** DefTensor: Defining vanishing Ricci tensor RicciPDpolar[-a, -b].

** DefTensor: Defining antisymmetric +1 density etaUppolar[a, b, c].

** DefTensor: Defining antisymmetric -1 density etaDownpolar[-a, -b, -c].

** DefBasis: Defining basis cartesian.

** DefCovD: Defining parallel derivative PDcartesian[-a].

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

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

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

** DefTensor: Defining vanishing Ricci tensor RicciPDcartesian[-a, -b].

** DefTensor: Defining antisymmetric +1 density etaUpcartesian[a, b, c].

** DefTensor: Defining antisymmetric -1 density etaDowncartesian[-a, -b, -c].

** DefBasis: Defining basis complex.

** DefCovD: Defining parallel derivative PDcomplex[-a].

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

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

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

** DefTensor: Defining vanishing Ricci tensor RicciPDcomplex[-a, -b].

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

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

** DefTensor: Defining vanishing FRiemann tensor FRiemannPDcomplex[-a, -b, -C, D].

** DefTensor: Defining vanishing FRiemann tensor FRiemannPDcomplex†[-a, -b, -C†, D†].

** DefBasis: Defining basis complex†.

** DefTensor: Defining antisymmetric +1 density etaUpcomplex[A, B].

** DefTensor: Defining antisymmetric +1 density etaUpcomplex†[A†, B†].

** DefTensor: Defining antisymmetric -1 density etaDowncomplex[-A, -B].

** DefTensor: Defining antisymmetric -1 density etaDowncomplex†[-A†, -B†].
```

大量のメッセージが出るが多くのよく用いるテンソルや曲率が同時に定義されている。複素表示ができるのもxActの特徴である。

ここまで見てきたようにxActは座標系と共に定義されるのでその分細かな取扱いが可能になる。

例えば平行微分は次のようになる。

```
PDpolar[-a][T[-b, -c]]
```

$\mathcal{D}_a T_{bc}$

色が付けることで次のように極座標が赤、直交座標が青で表現される。

```
T[{-a, -polar}, b] v[{1, cartesian}]
```

$$T_{a}^{\ b}\ v^{1}$$

テンソルに代入した時のように座標系とセットにして、数字を指定する場合は座標系の符号で上げ下げする。

```
T[{-a, -polar}, b] v[{1, -cartesian}]
```

$$T_{a}^{\ b}\ v_{1}$$

基底の定義、取り扱い

基底の操作はxActでは重要である。現在使用している計量や基底は次のようにして確認できる。空白になっている場合は未定義である。

```
$Metrics
$Bases
```

```
{}
```

```
{polar, cartesian, complex, complex†}
```

Basis[{添え字,座標系}]

次のようにBasis[]で基底を表すことができる。座標系とセットにし、その前の符号で上げ下げをおこなう。

```
Basis[{4, -complex}, A]
{Basis[{1, -polar}, a], Basis[-a, b], Basis[{-a, -cartesian}, {b, polar}],
 Basis[{-a, -cartesian}, {b, cartesian}], Basis[{1, -polar}, {1, polar}],
 Basis[{1, -polar}, {2, polar}]}
```

$$e_{4}^{\ A}$$

$$\left\{e_{1}^{\ a}, \delta_{a}^{\ b}, e_{a}^{\ b}, e_{a}^{\ b}, 1, 0\right\}$$

複素座標が緑、極座標が赤、直交座標が青で表現され、直交関係が確認できる。

これまでに使用している基底は次に新しく定義を変える前にUndefBasis[]で消去しておく。

ContractBasis[式]

例えば次のような基底とベクトルの演算を考える。

```
expr = Basis[{1, -polar}, a] Basis[-c, {1, cartesian}] v[-a, -b, c, {d, polar}]
  Basis[{-d, -polar}, e]
```

$$e_{c}^{\ 1}\ e_{1}^{\ a}\ e_{d}^{\ e}\ v_{ab}^{\ \ cd}$$

基底は単位ベクトルであるから縮約できる。

次のように座標系毎に縮約ができる。

```
expr1 = ContractBasis[expr]
ContractBasis[expr, polar]
ContractBasis[expr, cartesian]
```

$$v_{1b}^{\ \ 1e}$$

$$e_{c}^{\ 1}\ e_{1}^{\ a}\ v_{ab}^{\ \ ce}$$

$$e_{c}^{\ 1}\ e_{1}^{\ a}\ e_{d}^{\ e}\ v_{ab}^{\ \ cd}$$

逆に縮約したものを分ける場合は次のようにする。内添え字が内部変数で出てしまった場合はScreenDollarIndices[]を試すとよい。

```
SeparateBasis[AIndex][expr1]
% // ScreenDollarIndices
```

$$e_{f\$9669}{}^{1}\ e_{1}{}^{f\$9668}\ v_{f\$9668b}{}^{f\$9669e}$$

次のようにトレースをとる関数も用意されている。

```
TraceBasisDummy[v[{a, cartesian}] T[{-a, -cartesian}, -b]]
```

$$T_{0b}\ v^{0} + T_{1b}\ v^{1} + T_{2b}\ v^{2}$$

*BasisArray[*座標系*][*添え字*]*

次のようにBasisArray[]で基底を成分で表すことができる。添え字とセットにし、その前の符号で上げ下げをおこなう。

```
BasisArray[polar][a]
BasisArray[cartesian][-a]
BasisArray[polar, cartesian, cartesian][a, b, c] // MatrixForm
```

$$\{e_{0}{}^{a},\ e_{1}{}^{a},\ e_{2}{}^{a}\}$$

$$\{e_{a}{}^{0},\ e_{a}{}^{1},\ e_{a}{}^{2}\}$$

$$\begin{pmatrix}
\begin{pmatrix} e_0{}^b\ e_0{}^c\ e_0{}^a \\ e_0{}^b\ e_0{}^a\ e_1{}^c \\ e_0{}^b\ e_0{}^a\ e_2{}^c \end{pmatrix} &
\begin{pmatrix} e_0{}^c\ e_0{}^a\ e_1{}^b \\ e_0{}^a\ e_1{}^b\ e_1{}^c \\ e_0{}^a\ e_1{}^b\ e_2{}^c \end{pmatrix} &
\begin{pmatrix} e_0{}^c\ e_0{}^a\ e_2{}^b \\ e_0{}^a\ e_1{}^c\ e_2{}^b \\ e_0{}^a\ e_2{}^b\ e_2{}^c \end{pmatrix} \\
\begin{pmatrix} e_0{}^b\ e_0{}^c\ e_1{}^a \\ e_0{}^b\ e_1{}^c\ e_1{}^a \\ e_0{}^b\ e_1{}^a\ e_2{}^c \end{pmatrix} &
\begin{pmatrix} e_0{}^c\ e_1{}^b\ e_1{}^a \\ e_1{}^b\ e_1{}^c\ e_1{}^a \\ e_1{}^b\ e_1{}^a\ e_2{}^c \end{pmatrix} &
\begin{pmatrix} e_0{}^c\ e_1{}^a\ e_2{}^b \\ e_1{}^c\ e_1{}^a\ e_2{}^b \\ e_1{}^a\ e_2{}^b\ e_2{}^c \end{pmatrix} \\
\begin{pmatrix} e_0{}^b\ e_0{}^c\ e_2{}^a \\ e_0{}^b\ e_1{}^c\ e_2{}^a \\ e_0{}^b\ e_2{}^c\ e_2{}^a \end{pmatrix} &
\begin{pmatrix} e_0{}^c\ e_1{}^b\ e_2{}^a \\ e_1{}^b\ e_1{}^c\ e_2{}^a \\ e_1{}^b\ e_2{}^c\ e_2{}^a \end{pmatrix} &
\begin{pmatrix} e_0{}^c\ e_2{}^b\ e_2{}^a \\ e_1{}^c\ e_2{}^b\ e_2{}^a \\ e_2{}^b\ e_2{}^c\ e_2{}^a \end{pmatrix}
\end{pmatrix}$$

微分

xACTには豊富な微分関数が用意されている。しかし、これらを利用するには次のように多様体の定義から出発して、パラメタやテンソルを定義しないといけない。

これまでのテンソルに加えて次のテンソルを多様体M3上に定義する。

```
DefTensor[TT[-a, -b, c, d, -e], M3]
DefTensor[U[-a, -b, c, d], M3]
DefTensor[S[-A, B], M3, Dagger → Complex]
```

```
** DefTensor: Defining tensor TT[-a, -b, c, d, -e].
** DefTensor: Defining tensor U[-a, -b, c, d].
** DefTensor: Defining tensor S[-A, B].
** DefTensor: Defining tensor S†[-A†, B†].
```

さらに計量を次のように定義する。

```
DefMetric[-1, metricg[-a, -b], CD, {";", "∇"}, PrintAs -> "g"]
```

```
** DefTensor: Defining symmetric metric tensor metricg[-a, -b].
** DefTensor: Defining antisymmetric tensor epsilonmetricg[-a, -b, -c].
** 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 vanishing 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 3
** DefCovD:  Computing RicciToTFRicci for dim 3
** DefCovD:  Computing RicciToEinsteinRules for dim 3
** DefTensor: Defining weight +2 density Detmetricg[]. Determinant.
```

よって計量が定義された後は次のようにgが使えるようになる。

```
{Basis[-a, -b], Basis[{a, cartesian}, {b, cartesian}],
 Basis[{a, cartesian}, {b, polar}]}
{Basis[a, -b], Basis[{a, polar}, {-b, -cartesian}]}
```

$$\{ g_{ab} , g^{ab} , g^{ab} \}$$

$$\{ \delta_{b}^{a} , e_{b}^{a} \}$$

```
$Metrics
$Bases
SymmetryGroupOfTensor[Basis[{1, polar}, {-2, cartesian}]]
{metricg}
{polar, cartesian, complex, complex†}
StrongGenSet[{1, 2}, GenSet[Cycles[{1, 2}]]]
```

座標系を極座標に変えて、次のようなベクトルとテンソルの演算をつくる。

```
IndicesOfVBundle[TangentM3]
{{a, b, c, d, e, f}, {f1, f2, f3}}
```

```
T[{-a, -polar}, {-b, -polar}] v[{a, polar}] v[{b, polar}]
Simplification[%]
```

$$T_{ab} \; v^{a} \; v^{b}$$

```
0
```

しかし、異なる座標系である場合次の演算は0にならない。

```
v[-a] v[a] - v[{-a, -polar}] v[{a, polar}];
Simplification[%]
```

$$v_a \; v^a - v_a \; v^a$$

次のように基底を統一すれば0になる。

```
v[-a] v[a] - v[{-a, -polar}] v[{a, polar}]
```

$$v_a \; v^a - v_a \; v^a$$

```
Torsion[cartesian]
Bracket[a][Basis[{1, -polar}, b], Basis[{2, -polar}, c]]
```

```
TorsionPDcartesian
```

$$-T^a{}_{12}$$

反対称のテンソルであればこの結果は0になる。これはSimplification[]を実行して確認できた。

```
expr = TT[{1, -polar}, -b, {2, cartesian}, c, -a] Basis[{1, -polar}, a] +
  Basis[{1, -polar}, a] Basis[-d, {2, cartesian}] U[-a, -b, d, {e, polar}]
   Basis[{-e, -polar}, c] S[{-A, -comp}, B] S[-B, {A, comp}]
```

$$e_1{}^a \; TT_{1b}{}^{2c}{}_a + e_d{}^2 \; e_1{}^a \; e_e{}^c \; S_B{}^A \; S_A{}^B \; U_{ab}{}^{de}$$

```
ChangeComponents[Ts[{-a, -cartesian}, {-b, -cartesian}],
 Ts[{a, cartesian}, {b, cartesian}]]
```

Added independent rule $Ts^{00} \to Ts^{00}$ for tensor Ts

Added independent rule $Ts^{01} \to Ts^{01}$ for tensor Ts

Added independent rule $Ts^{02} \to Ts^{02}$ for tensor Ts

Added independent rule $Ts^{10} \to Ts^{10}$ for tensor Ts

Added independent rule $Ts^{11} \to Ts^{11}$ for tensor Ts

Added independent rule $Ts^{12} \to Ts^{12}$ for tensor Ts

Added independent rule $Ts^{20} \to Ts^{20}$ for tensor Ts

Added independent rule $Ts^{21} \to Ts^{21}$ for tensor Ts

Added independent rule $Ts^{22} \to Ts^{22}$ for tensor Ts

Added independent rule $Ts^{0}_{0} \to g_{00}\ Ts^{00} + g_{10}\ Ts^{01} + g_{20}\ Ts^{02}$ for tensor Ts

Added independent rule $Ts^{1}_{0} \to g_{00}\ Ts^{10} + g_{10}\ Ts^{11} + g_{20}\ Ts^{12}$ for tensor Ts

Added independent rule $Ts^{2}_{0} \to g_{00}\ Ts^{20} + g_{10}\ Ts^{21} + g_{20}\ Ts^{22}$ for tensor Ts

Added independent rule $Ts^{0}_{1} \to g_{01}\ Ts^{00} + g_{11}\ Ts^{01} + g_{21}\ Ts^{02}$ for tensor Ts

Added independent rule $Ts^{1}_{1} \to g_{01}\ Ts^{10} + g_{11}\ Ts^{11} + g_{21}\ Ts^{12}$ for tensor Ts

Added independent rule $Ts^{2}_{1} \to g_{01}\ Ts^{20} + g_{11}\ Ts^{21} + g_{21}\ Ts^{22}$ for tensor Ts

Added independent rule $Ts^{0}_{2} \to g_{02}\ Ts^{00} + g_{12}\ Ts^{01} + g_{22}\ Ts^{02}$ for tensor Ts

Added independent rule $Ts^{1}_{2} \to g_{02}\ Ts^{10} + g_{12}\ Ts^{11} + g_{22}\ Ts^{12}$ for tensor Ts

Added independent rule $Ts^{2}_{2} \to g_{02}\ Ts^{20} + g_{12}\ Ts^{21} + g_{22}\ Ts^{22}$ for tensor Ts

Computed $Ts^{a}_{b} \to g_{cb}\ Ts^{ac}$ in 0.0500007 Seconds

Added independent rule $Ts_{00} \to g_{00}\ Ts^{0}_{0} + g_{10}\ Ts^{1}_{0} + g_{20}\ Ts^{2}_{0}$ for tensor Ts

Added independent rule $Ts_{01} \to g_{00}\ Ts^{0}_{1} + g_{10}\ Ts^{1}_{1} + g_{20}\ Ts^{2}_{1}$ for tensor Ts

Added independent rule $Ts_{02} \to g_{00}\ Ts^{0}_{2} + g_{10}\ Ts^{1}_{2} + g_{20}\ Ts^{2}_{2}$ for tensor Ts

Added independent rule $Ts_{10} \to g_{01}\ Ts^{0}_{0} + g_{11}\ Ts^{1}_{0} + g_{21}\ Ts^{2}_{0}$ for tensor Ts

Added independent rule $Ts_{11} \to g_{01}\ Ts^{0}_{1} + g_{11}\ Ts^{1}_{1} + g_{21}\ Ts^{2}_{1}$ for tensor Ts

Added independent rule $Ts_{12} \to g_{01}\ Ts^{0}_{2} + g_{11}\ Ts^{1}_{2} + g_{21}\ Ts^{2}_{2}$ for tensor Ts

Added independent rule $Ts_{20} \to g_{02}\ Ts^{0}_{0} + g_{12}\ Ts^{1}_{0} + g_{22}\ Ts^{2}_{0}$ for tensor Ts

Added independent rule $Ts_{21} \to g_{02}\ Ts^{0}_{1} + g_{12}\ Ts^{1}_{1} + g_{22}\ Ts^{2}_{1}$ for tensor Ts

Added independent rule $Ts_{22} \to g_{02}\ Ts^{0}_{2} + g_{12}\ Ts^{1}_{2} + g_{22}\ Ts^{2}_{2}$ for tensor Ts

Computed $Ts_{ab} \to g_{ca}\ Ts^{c}_{b}$ in 0.0500020 Seconds

```
FoldedRule[{}, {Ts00 → g00 Ts0 0 + g10 Ts1 0 + g20 Ts2 0,
   Ts01 → g00 Ts0 1 + g10 Ts1 1 + g20 Ts2 1, Ts02 → g00 Ts0 2 + g10 Ts1 2 + g20 Ts2 2,
   Ts10 → g01 Ts0 0 + g11 Ts1 0 + g21 Ts2 0, Ts11 → g01 Ts0 1 + g11 Ts1 1 + g21 Ts2 1,
   Ts12 → g01 Ts0 2 + g11 Ts1 2 + g21 Ts2 2, Ts20 → g02 Ts0 0 + g12 Ts1 0 + g22 Ts2 0,
   Ts21 → g02 Ts0 1 + g12 Ts1 1 + g22 Ts2 1, Ts22 → g02 Ts0 2 + g12 Ts1 2 + g22 Ts2 2},
 {}, {Ts0 0 → g00 Ts00 + g10 Ts01 + g20 Ts02,
   Ts1 0 → g00 Ts10 + g10 Ts11 + g20 Ts12, Ts2 0 → g00 Ts20 + g10 Ts21 + g20 Ts22,
   Ts0 1 → g01 Ts00 + g11 Ts01 + g21 Ts02, Ts1 1 → g01 Ts10 + g11 Ts11 + g21 Ts12,
   Ts2 1 → g01 Ts20 + g11 Ts21 + g21 Ts22, Ts0 2 → g02 Ts00 + g12 Ts01 + g22 Ts02,
   Ts1 2 → g02 Ts10 + g12 Ts11 + g22 Ts12, Ts2 2 → g02 Ts20 + g12 Ts21 + g22 Ts22}]
```

```
ComponentArray[Ts[{-a, -cartesian}, {-b, -cartesian}]];
% /. TensorValues[Ts] // Simplify;
% // MatrixForm
```

$$\begin{pmatrix} g_{00}^{\,2}\ Ts^{00} + g_{10}^{\,2}\ Ts^{11} + g_{00}\left(g_{10}\left(Ts^{01} + Ts^{10}\right) + g_{20}\left(Ts^{02} + Ts^{20}\right)\right) + g_{10}\ g_{20}\ (\\ g_{01}\left(g_{00}\ Ts^{00} + g_{10}\ Ts^{01} + g_{20}\ Ts^{02}\right) + g_{11}\left(g_{00}\ Ts^{10} + g_{10}\ Ts^{11} + g_{20}\ Ts^{12}\right) + g_{21}\left(g \\ g_{02}\left(g_{00}\ Ts^{00} + g_{10}\ Ts^{01} + g_{20}\ Ts^{02}\right) + g_{12}\left(g_{00}\ Ts^{10} + g_{10}\ Ts^{11} + g_{20}\ Ts^{12}\right) + g_{22}\left(g \end{pmatrix}$$

この章ではxCobaがロードされた直後として改めて次のように定義する。すでに定義していればUndefで消去しておく。

```
DefManifold[M3, 3, {a, b, c, d, e}]
DefManifold[S2, 2, {A, B, C, D, G, H}]
DefManifold[M5, {M3, S2}, {μ, ν, λ, σ, η, ρ}]
```

```
ValidateSymbol::used: Symbol M3 is already used as a manif…
ValidateSymbol::used: Symbol A is already used as an abstract ind…
DimOfManifold::unknown: Unknown manifold S2.
Throw::nocatch: Uncaught Throw[Null] をキャッチしないまま上層レベルに戻しま…
  Hold[Throw[Null]]
```

次のようにテンソルとパラメタを定義する。

4行目のr[]ように[]の中を空白として定義すると、これを変数として微積分に利用できる。

```
DefTensor[T[a, b, -c], M3]
DefTensor[S[a, b], M3]
DefTensor[v[a], M3]
DefTensor[r[], M3]
DefParameter[time]
DefTensor[U[a, b, -C], {M3, S2, time}]
DefTensor[F[-a, -b], M3, Antisymmetric[{-a, -b}]]
DefTensor[w[a], M3]
```

```
** DefTensor: Defining tensor T[a, b, -c].
ValidateSymbol::used: Symbol S is already used as a tensor.
ValidateSymbol::used: Symbol v is already used as a tensor.
** DefTensor: Defining tensor r[].
** DefParameter: Defining parameter time.
ValidateSymbol::used: Symbol U is already used as a tensor.
```

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

** DefTensor: Defining tensor w[a].
```

*DefCovD[*表現*[*添え字*]]*

共変微分を定義する。Cd[]で簡単に表すことにしよう。

これを実行すると次のようにリーマンテンソルや、クリストッフェルの微分まで定義される。

```
DefCovD[Cd[-a]]
```

```
** 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]. Antisymmetric only in the first pair.

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

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

実際に実行するには次のように左から作用させると次のように表現される。ただしCd[添え字][対称]のように[]を2つ書くというのは間違えやすいので注意がいる。

```
Cd[-a][v[c]]
Cd[-a][F[-b, -c] v[c]] v[b]
```

$$\nabla_a v^c$$

$$v^b \left(v^c \left(\nabla_a F_{bc} \right) + F_{bc} \left(\nabla_a v^c \right) \right)$$

2行目にでは積の微分則が使われていることがわかる。簡約表現を使うと次のようになる。

```
Cd[-a][F[-b, -c] v[c]] v[b] // Simplification
```

$$F_{bc} \; v^b \left(\nabla_a v^c \right)$$

リーマンテンソルは次のようになる。

```
RiemannCd[-a, -b, -c, d] + RiemannCd[-b, -a, -c, d]
% // Simplification
```

$$R[\nabla]_{abc}{}^{d} + R[\nabla]_{bac}{}^{d}$$

$$0$$

添え字を正しく表示させるためには$PrePrint=ScreenDollarIndices;を宣言しておく

これをしておかないと内部変数がそのまま表示さることがある。

```
$PrePrint = ScreenDollarIndices;
RiemannCd[-a, -b, -c, d] // RiemannToChristoffel
```

$$\Gamma[\nabla]^{d}{}_{be} \; \Gamma[\nabla]^{e}{}_{ac} - \Gamma[\nabla]^{d}{}_{ae} \; \Gamma[\nabla]^{e}{}_{bc} - \partial_a \Gamma[\nabla]^{d}{}_{bc} + \partial_b \Gamma[\nabla]^{d}{}_{ac}$$

*RiemannToChristoffel[*式*]*

*CovDToChristoffel[*式*]*

RiemannToChristoffel[]を使うとリーマンテンソルの表示をクリストッフェルの接続記号で表現できる。

また、CovDToChristoffel[]を使うと共変微分形式からクリストッフェルの接続記号に変えることができる。

対称性があるので次の例ではSimulify かToCanonicalをつかうと0になる。

またFが反対称、Sが対象テンソルで定義したので接続記号で表記させると符号の相違が確認できる。

```
RiemannCd[-a, -b, -c, d] + RiemannCd[-b, -c, -a, d] + RiemannCd[-c, -a, -b, d] //
 RiemannToChristoffel
% // ToCanonical
Cd[-c][F[-a, -b]]
% // CovDToChristoffel // ToCanonical
Cd[-c][S[-a, -b]]
% // CovDToChristoffel // ToCanonical
```

$$-\Gamma[\nabla]^{d}{}_{ce}\ \Gamma[\nabla]^{e}{}_{ab} + \Gamma[\nabla]^{d}{}_{be}\ \Gamma[\nabla]^{e}{}_{ac} + \Gamma[\nabla]^{d}{}_{ce}\ \Gamma[\nabla]^{e}{}_{ba} - \Gamma[\nabla]^{d}{}_{ae}\ \Gamma[\nabla]^{e}{}_{bc} - \Gamma[\nabla]^{d}{}_{be}\ \Gamma[\nabla]^{e}{}_{ca} + \Gamma[\nabla]^{d}{}_{ae}\ \Gamma[\nabla]^{e}{}_{cb} - \partial_a\Gamma[\nabla]^{d}{}_{bc} + \partial_a\Gamma[\nabla]^{d}{}_{cb} + \partial_b\Gamma[\nabla]^{d}{}_{ac} - \partial_b\Gamma[\nabla]^{d}{}_{ca} - \partial_c\Gamma[\nabla]^{d}{}_{ab} + \partial_c\Gamma[\nabla]^{d}{}_{ba}$$

$$0$$

$$\nabla_c F_{ab}$$

$$-\Gamma[\nabla]^{d}{}_{bc}\ F_{ad} + \Gamma[\nabla]^{d}{}_{ac}\ F_{bd} + \partial_c F_{ab}$$

$$\nabla_c S_{ab}$$

$$-\Gamma[\nabla]^{d}{}_{bc}\ S_{ad} - \Gamma[\nabla]^{d}{}_{ac}\ S_{db} + \partial_c S_{ab}$$

*Antisymmetrize[*式*,{*添え字リスト*}]*

Antisymmetrize[]を使うと各添え字の反対称和がつくれる便利なコマンドである。

これでビアンキの第2恒等式が簡単に次のように確認できる。

```
Antisymmetrize[Cd[-e][RiemannCd[-c, -d, -b, a]], {-c, -d, -e}]
% // ToCanonical
% // Simplify
% // CovDToChristoffel
```

$$\frac{1}{6}\left(\nabla_c R[\nabla]_{deb}{}^{a} - \nabla_c R[\nabla]_{edb}{}^{a} - \nabla_d R[\nabla]_{ceb}{}^{a} + \nabla_d R[\nabla]_{ecb}{}^{a} + \nabla_e R[\nabla]_{cdb}{}^{a} - \nabla_e R[\nabla]_{dcb}{}^{a}\right)$$

$$\frac{1}{3}\left(\nabla_c R[\nabla]_{deb}{}^{a}\right) - \frac{1}{3}\left(\nabla_d R[\nabla]_{ceb}{}^{a}\right) + \frac{1}{3}\left(\nabla_e R[\nabla]_{cdb}{}^{a}\right)$$

$$\frac{1}{3}\left(\nabla_c R[\nabla]_{deb}{}^{a} - \nabla_d R[\nabla]_{ceb}{}^{a} + \nabla_e R[\nabla]_{cdb}{}^{a}\right)$$

$$\frac{1}{3}\left(\Gamma[\nabla]^{a}{}_{ee9}\ R[\nabla]_{cdb}{}^{e9} - \Gamma[\nabla]^{e10}{}_{eb}\ R[\nabla]_{cde10}{}^{a} - \Gamma[\nabla]^{a}{}_{de5}\ R[\nabla]_{ceb}{}^{e5} + \Gamma[\nabla]^{e6}{}_{db}\ R[\nabla]_{cee6}{}^{a} - \Gamma[\nabla]^{e12}{}_{ed}\ R[\nabla]_{ce12b}{}^{a} + \Gamma[\nabla]^{e8}{}_{de}\ R[\nabla]_{ce8b}{}^{a} + \Gamma[\nabla]^{a}{}_{ce1}\ R[\nabla]_{deb}{}^{e1} - \Gamma[\nabla]^{e2}{}_{cb}\ R[\nabla]_{dee2}{}^{a} - \Gamma[\nabla]^{e4}{}_{ce}\ R[\nabla]_{de4b}{}^{a} - \Gamma[\nabla]^{e11}{}_{ec}\ R[\nabla]_{e11db}{}^{a} - \Gamma[\nabla]^{e3}{}_{cd}\ R[\nabla]_{e3eb}{}^{a} + \Gamma[\nabla]^{e7}{}_{dc}\ R[\nabla]_{e7eb}{}^{a} + \partial_c R[\nabla]_{deb}{}^{a} - \partial_d R[\nabla]_{ceb}{}^{a} + \partial_e R[\nabla]_{cdb}{}^{a}\right)$$

共変微分のCd[][]は繰り返しさようさせると次のように入れ後状になって醜い。

そこでMathematica の@コマンドを利用すると次のようにすっきりする。

```
Cd[-a][Cd[-b][v[c]]]
Cd[-a]@Cd[-b]@v[c]
```

$$\nabla_a \nabla_b v^{c}$$

$$\nabla_a \nabla_b v^{c}$$

*CommuteCovDs[*式,演算子,*{*添え字リスト*}]*

ある演算子の交換関係がxActではコマンドがある。例えば先の結果をCdで交換積をとればこれはリーマンテンソルの定義である。

```
CommuteCovDs[%, Cd, {-b, -a}]
CommuteCovDs[Cd[-a]@Cd[-b]@r[], Cd, {-b, -a}]
```

$$-F^{ab}\ R[\nabla]_{ab} + F^{ab}\ R[\nabla]_{ba} + \nabla_b \nabla_a F^{ab}$$

$$\nabla_b \nabla_a r$$