MathTensorの利用

H.Yanase

はじめに

Mathematicaは2012年現在Ver9になり、その計算能力だけではなく、インターフェスやWEBアプリケーションへの対応も充実し、グラフ作成、数式処理のソフトの枠を越えて数学の膨大なデータベースも取り込み発展してきている。しかし、物理や数学でよく利用するTensorの計算については「The Mathematica Packages CARTAN and MathTensor」のパッケージが出ているが、十分ではない。そこでMathTensor,　Inc社から出ている「MathTensor」が充実している。しかし、有償であり、MathematicaがDOSの時代からTensor面を補うパッケージとしてつくられたもので現段階のバージョンが2.2になる。ほとんど大きなバーションアップはなく開発者個人の意思に委ねられているようである。この点はMathematicaも未だに株式を公開することなく開発されているので共通しているが巨大な企業となったMathematicaから見ると不便なことも多い。マニュアルも兼ねた「MathTensor」という本が日本でも手に入る。ここでの引用も多いので原文を参考にしてみるといいだろう。しかし、日本語の解説本はほとんどない。サポートも英語のメールのみである。そこでここではMathTensorのインストールから基本的な活用方法を紹介し、Tensorを利用した研究に少しでも役立つことを期待したい。下記のサイトがMathTensorの公式サイトである。オンラインマニュアルもあるので便利である。あまり更新されないがMathematicaのバージョンアップに伴い、情報が掲載される。

【参考URL:Mathtensor】http://smc.vnet.net/MathTensor.html

【著者のサイト：教育を考える会http://www.ne.jp/asahi/buturiwa/ai/EducationM1.htm】

【参考文献】

また相対論、微分幾何については基本的なものを一読しておくといいだろう(例えば内山氏の岩波から出ている「相対性理論 (物理テキストシリーズ 8)」、同じく、岩波からの微分・位相幾何 (理工系の基礎数学 10) 和達 三樹、またここでも引用させていただいている名著、高橋康氏の「量子場を学ぶための場の解析力学入門」講談社サイエンティフィックなどがある。

Mathtensorのインストール

＊MathematicaVer6以上の場合のインストール

　ファイルのフォルダ名は一般的な場合なので自分の環境に合わせ、ドライブ名やAdministrator名は変更すること。

１．次のフォルダを作り**Mathtensor**をコピーする。

バージョンアップの場合・・下記の例ではVer6.0のフォルダが例になっているがVer8.0の場合は8.0等に修正する。

2011年現在MathematicaVer8に対応するためVer2.2.2にVerUPされ、Mathematica　も新しくし、旧いバージョンは削除した場合もこのフォルダは削除されずに残るのでVerUP際のPath指定の修正の必要はない。ただ、旧フォルダを削除した時は新しいフォルダ用にパスを書き換えないといけない。）

```
"C:\\Program Files\\Wolfram Research\\Mathematica\\6.0\\AddOns\\Applications\\mathtensor"]
```

２．次の場所（隠しフォルダ）の**init.m**に次の内容を書き加える。

　フォルダオプションで隠しファイルも表示できるようにしておく。

C:\Documents and Settings\Administrator\Application Data\Mathematica\Kernel

```
$Path = Join[$Path,
  {"C:\\Program Files\\Wolfram Research\\Mathematica\\6.0\\AddOns\\Applications\\mathtensor"}];
  Off[Syntax::"com"]
```

次のコマンドでパスが追加されたか確認する。最後にMathtensorのPathが書かれていればよい。

以下の例ではMathematicaのVer7でインストールしたMathtensorをそのままVer8で利用している。

```
$Path
```

```
{E:\Program Files\Wolfram Research\Mathematica\8.0\SystemFiles\Links,
 E:\Documents and Settings\Yanase\Application Data\Mathematica\Kernel,
 E:\Documents and Settings\Yanase\Application Data\Mathematica\Autoload,
 E:\Documents and Settings\Yanase\Application Data\Mathematica\Applications,
 E:\Documents and Settings\All Users\Application Data\Mathematica\Kernel,
 E:\Documents and Settings\All Users\Application Data\Mathematica\Autoload,
 E:\Documents and Settings\All Users\Application Data\Mathematica\Applications,
 ., E:\Documents and Settings\Yanase,
 E:\Program Files\Wolfram Research\Mathematica\8.0\AddOns\Packages,
 E:\Program Files\Wolfram Research\Mathematica\8.0\AddOns\LegacyPackages,
 E:\Program Files\Wolfram Research\Mathematica\8.0\SystemFiles\Autoload,
 E:\Program Files\Wolfram Research\Mathematica\8.0\AddOns\Autoload,
 E:\Program Files\Wolfram Research\Mathematica\8.0\AddOns\Applications,
 E:\Program Files\Wolfram Research\Mathematica\8.0\AddOns\ExtraPackages,
 E:\Program Files\Wolfram Research\Mathematica\8.0\SystemFiles\Kernel\Packages,
 E:\Program Files\Wolfram Research\Mathematica\8.0\Documentation\English\System,
 E:\Program Files\Wolfram Research\Mathematica\7.0\AddOns\Applications\mathtensor}
```

＊はじめての場合は次のようにパスワードを入力
2013年以降はMachineIDを伝えてはじめからパスワードが.mファイルに組み込まれる場合があるので
その場合は下記は必要ない。

```
$MachineID
```

(パスワード数値列)

初期設定は以上である。
次はMathematicaを起動し、Mathtensorをロードしよう。

MathTensorのロード

Mathtensorを読み込む。　MathematicaVer6以降を利用している場合はTensorQのプロテクトをはずす。
さらにVer7以降はRiemannRのプロテクトをはずす。さらにVer9以降は次のように入力する。

```
Unprotect[TensorQ]
Unprotect[RiemannR]
Remove[RiemannR]
Unprotect[Symmetrize]
Remove[Symmetrize]
<< Mathtens.m
```

```
{TensorQ}
```

```
{RiemannR}
```

```
{Symmetrize}
```

```
Get::noopen: init.m を開くことができません。
```

```
$Failed
```

```
Loading MathTensor for DOS/Windows . . .



======================================================================

MathTensor (TM) 2.2.2 (Windows/Linux/Unix/MacOS X (R)) (July 25, 2011)

by Leonard Parker and Steven M. Christensen

Copyright (c) 1991-2011 MathTensor, Inc.

Runs with Mathematica (R) Versions 2.x-8.x

Licensed to machine soraduo.

======================================================================

No unit system is chosen. If you want one,

you must edit the file called Conventions.m,

or enter a command to interactively set units.

Units: {}

Sign conventions: Rmsign = 1 Rcsign = 1

MetricgSign = 1 DetgSign = -1

TensorForm turned on,
ShowTime turned off,
MetricgFlag = True.

========================================
```

上記のようなメッセージが出てくればOKである。

Tensorの定義と基本コマンド

■ ●テンソルの定義

ここではMathTensorの基本を習得しよう。Mathematicaの基本操作は理解していることを前提とする。
まだMathematicaに慣れていない場合は本サイトからMathematiaの基本をダウンロードして学んでおこう。

以下ではすでにMathtensorをロードしてあるとする。

■ **DefineTnsor[記号, {対称性の定義}]　【Tensorの定義】**

次のようにテンソルｔとベクトルｖを定義する。
{}内の数値は{次元、階数}対称性である。ベクトルは全て１階、対称の場合は1、反対称は-1である。

```
Clear[va, vb, ta, tb, ant];
DefineTensor[va, {{1, 1}, 1}]
DefineTensor[vb, {{1, 1}, 1}]
DefineTensor[ta, {{2, 1}, 1}]
DefineTensor[tb, "t", {{2, 1}, 1}]
DefineTensor[ant, "a", {{2, 1}, -1}]
DefineTensor[{t1, t2, t3}, {{2, 1}, -1}]
```

最後2行の書式は”a”で表記をさせるオプションが追加されている。
このオプションをつけると省略形で表記できる。
上付き添え字はua,ubなどで表し、下付添え字はla,lbなどで表す。
残念ながらMathtensorはスタンダードの表示ではスクリプトを表示してしまうので例えば
次のように後ろに　//OutputForm で出力形式を切り替えるとMathematicaの環境設定は変更の必要はない。
画面にテンソル表示をさせたい時は最後に//OutputFormをつけるようにしよう。
これをつけないとScriptがそのまま表示されてしまう。
だだし、結果を利用したい時にはつけてはいけない。（後述）

また最後の行のように複数をリストで一度に定義することもできる。

```
ta[ua, lb] // OutputForm
tb[ua, ub] // OutputForm
```

```
   a
ta
  b

  ab
t
```

では(2,1)型のテンソルTsを定義しよう。

```
DefineTensor[Ts, {{2, 1}, 1}]
```

```
PermWeightsbytm: Symmetryof Tsassigned
PermWeightdelete fObject Tsdefi
```

テンソルの下付の順番は次のように関係ない

```
Ts[la, lb] + Ts[lb, la] // OutputForm
```

```
2 Ts
    ab
```

■ **Tsimplify[式]　【Tensorの簡約】**

テンソルを簡約する場合はTsimplifyをつかう。
先に定義した反対称antと対称taのテンソルをかけてみよう。

ここでよく使用するf1,f2,f3はここでは適当な記号である。
式の部分に//OutputFormをつけて代入すると正しく演算しないので次のようこの記号を一時的に利用する。
スクリプトを表示させたくないので1行目は ; で画面出力をさせないようにしている。

```
f1 = ant[la, lb] ta[ua, ub];
f1 // OutputForm
Tsimplify[f1] // OutputForm
f2 = tb[la, lb] ta[ua, ub];
f2 // OutputForm
```

```
      ab
a   ta
 ab

0

   ab
ta   t
      ab
```

対称、反対称の積が縮約された。

■ **MaxwellF[添え字、添え字]**　電磁場テンソル

MathTensorにはMaxwell、Ricci、Riemannなどのテンソルは定義されている。
例えばMaxwellの電磁場テンソルを共変微分してみよう。

■ **CD[テンソル,添え字]**　テンソルを添え字で共変微分する

■ **CDtoOD[式]**　共変微分を普通微分で表す。

■ **AffineToMetric[式]**　接続係数を計量テンソルで表す。
AffineG[{添え字}]　**Christoffel**の接続係数

```
AffineToMetric[f4] // OutputForm
```

$$\frac{F^{pq}\,g^{rs}\,g_{rs,p,q}}{2} - \frac{F^{pq}\,g^{rs}\,g_{pr,qs}}{2} + \frac{F^{pq}\,g^{rs}\,g_{ps,qr}}{2} + \frac{F^{pq}\,g^{rs}\,g_{rs,pq}}{2}$$

ついでに計量テンソルを用いたアフィン係数の公式をAffineToMetric[]で表してみよう。

```
AffineToMetric[AffineG[ui, lj, lk]] // OutputForm
```

$$\frac{g^{pi}\ g_{pj,k}}{2} + \frac{g^{pi}\ g_{pk,j}}{2} - \frac{g^{pi}\ g_{jk,p}}{2}$$

次にリッチテンソルについても見てみよう。

■ **RicciR[添え字、添え字]**　リッチテンソル

```
Clear[f1, f2, f3];
f1 = RicciR[la, uc] RicciR[lc, lb] + RicciR[le, la] RicciR[lb, ue];
f1 // OutputForm
```

$$R_a^{\ c}\ R_{bc} + R_{ae}\ R_b^{\ e}$$

これを簡約してみよう。

```
Tsimplify[f1] // OutputForm
```

$$2\ R_a^{\ c}\ R_{bc}$$

と簡単になった。さらに次のような式をつくる。

```
f2 = Expand[(RicciR[la, lb] + MaxwellF[la, lb]) (RicciR[ua, ub] + MaxwellF[ua, ub])];
f2 // OutputForm
```

$$F_{ab}\ F^{ab} + F^{ab}\ R_{ab} + F_{ab}\ R^{ab} + R_{ab}\ R^{ab}$$

```
f3 = Tsimplify[f2];
f3 // OutputForm
```

$$F_{ab}\ F^{ab} + R_{ab}\ R^{ab}$$

次の項が0になっていることがわかる。

```
Tsimplify[MaxwellF[la, lb] RicciR[ua, ub]]
```

```
0
```

■ **Downuserlist** 下付添え字一覧　　**Upuserlist** 上付き添え字一覧

```
Downuserlist // OutputForm
Upuserlist // OutputForm
```

$$\{a', b', c', d', e', f', g', h', i', j', k', l', m', n', o'\}$$

$$\{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o\}$$

反対称テンソルは表示オプションに従い次のように表示される。

```
ant[la, lb] // OutputForm
ant[lb, la] // OutputForm
```

$$a_{ab}$$

$$-a_{ab}$$

これらの定義は次のコマンドで確認できる。

■ **Symmetries[** 記号**, {**添字のリスト**}]**　対称性の確認

```
Symmetries[tb[la, lb]]
Symmetries[MaxwellF[la, lb]]
Symmetries[RicciR[la, lb]]
Symmetries[RiemannR[la, lb, lc, ld]]
Symmetries[AffineG[ui, lj, lk]]
```

```
{{2, 1}, 1}
```

```
{{2, 1}, -1}
```

```
{{2, 1}, 1}
```

```
{{2, 1, 3, 4}, -1, {1, 2, 4, 3}, -1, {3, 4, 1, 2}, 1}
```

```
{{1, 3, 2}, 1}
```

完全な表現として次の方法もある。

■ **AllSymmetries[** 記号**, {**添字のリスト**}]**　完全な対称性の確認

```
AllSymmetries[ant[la, lb]]
AllSymmetries[MaxwellF[la, lb]]
AllSymmetries[AffineG[ui, lj, lk]]
```

```
{{1, 2}, 1, {2, 1}, -1}
```

```
{{1, 2}, 1, {2, 1}, -1}
```

```
{{1, 2, 3}, 1, {1, 3, 2}, 1}
```

■ ●微分形式

Mathtensorを利用すると微分形式の取り扱いがMathematcaでできるようになる。
この節で利用するテンソルの定義からはじめよう。
MathTensorでは次のようにはじめに次元を定義する。
初期化コマンドのClear[ ]はMathematicaで変数を利用していないなら必要ない。

■ **Dimension** 次元の定義

```
Clear[x]; Clear[y]; Clear[z];
Dimension = 3;
x[1] = x; x[2] = y; x[3] = z;
```

次に3形式までを次のように定義する。

*s1,s2,s3*を１形式として次のように定義する。
*w1,w2,w3*で２形式を次のように定義する。
*t1,t2,t3*で*3*形式を次のように定義する。
以下のように多くの警告が出るが無視してよい。

■ **DefineForm[{**記号リスト**}, {**形式数のリスト**}]**　【微分形式の定義】

ここでは次元を3ととしてユークリッドの座標を用いる
Mathtensorではx[ ]は座標構成として予約されている。
必要なら極座標などを用いることもできる。

次のようにここで利用する微分形式を定義しておこう。

Mahthematicaでは ; を最後に追加すると画面に結果をかえさなくなるがMathtensorの定義は

；　をつけてもメッセージが返ってくる。

```
Clear[w1, w2, w3]; Clear[s1, s2, s3]; Clear[t1, t2, t3];
DefineForm[{s1, s2, s3}, {1, 1, 1}];
DefineForm[{w1, w2, w3}, {2, 2, 2}];
DefineForm[{t1, t2, t3}, {3, 3, 3}];
```

■ **XD[{記号}]**　外微分をつくる。

XD[ ]は外微分をつくる。
次元が３なので３形式の外微分は０となることが確かめられる。

```
XD[s1]
XD[w1]
XD[t1]
```

```
ds1

dw1

0
```

これでは結果だけなので
次のコマンドで詳細に中身を見てみよう。

単位ベクトルも含くめ、指定された座標表示で次のように表すことができる。

■ **CoorRep[{記号}]**　座標成分表示させる
CollectForm[{記号}]　次数順に整理する。

```
dg1 = CoordRep[XD[s1]];
CollectForm[%] // OutputForm
dg2 = CoordRep[XD[w1]];
CollectForm[%] // OutputForm
dg3 = CoordRep[XD[t1]];
CollectForm[%] // OutputForm
```

```
(s1    - s1    ) dx ^ dy + (s1    - s1    ) dx ^ dz + (s1    - s1    ) dy ^ dz
   2;1     1;2                3;1     1;3                3;2     2;3

(-w1     + w1     - w1     ) dx ^ dy ^ dz
    32;1     31;2     21;3

0
```

見慣れた表現になった。添え字*1,2,3*はここでは*x,y,z*に対応する。
具体的に*s1*や*w1*の中身は次のように参照することができる。

```
Table[s1[i, j], {i, 1, 3}] // MatrixForm // OutputForm
Table[w1[i, j], {i, 1, 3}, {j, 1, 3}] // MatrixForm // OutputForm
```

```
s1 1
   j
s1 2
   j
s1 3
   j
```

```
0        w1 12    w1 13
-w1 12   0        w1 23
-w1 13   -w1 23   0
```

*s1*は*1*形式なので縦ベクトルになっている。
*w1*は*2*形式なので対角項は全て０である。
注意点は*MathTensor*では配列*[]*の中の数字は正が上付き、負が下付を切り替える。
例えば先の*w1*は次のように下付表現できる。

```
Table[w1[-i, -j], {i, 1, 3}, {j, 1, 3}] // MatrixForm // OutputForm
```

```
0        -w1     -w1
            21      31
w1          0       -w1
   21                   32
w1          w1
   31       32      0
```

また数字の１，２，３ではわかりにくい場合は次のように表現を変えることができる。

*li*のｌは下付、*u*で上付きを表す。ただし、指定できる添え字は先ほどの一覧リストで決まっていて*x*ｙなどは指定できない。

■ **FtoC[{記号},{添え字リスト}]**　フォーム表示から成分表示に切り替える。

```
FtoC[w2, {li, lj}] // OutputForm
FtoC[w2, {ui, uj}] // OutputForm
```

```
w2
  ij

   ij
w2
```

もちろん任意に各成分を指定できなかったら*Mathematia*のパワーを利用できない、これは次のようにおこなう。
要素を代入するときは次のように－をつけて代入することに注意する。
これは計量テンソルを作成するときも同様である。

```
w1[-1, -2] = 3 x^2 + 2 y - z;
w1[-1, -3] = 3 x^2 - 2 y + z;
w1[-2, -3] = -3 x^2 + 2 y + z;
w1[-2, -1] = -w1[-1, -2];
w1[-3, -1] = -w1[-1, -3];
w1[-3, -2] = -w1[-2, -3];
Table[w1[-i, -j], {i, 1, 3}, {j, 1, 3}] // MatrixForm
```

$$\begin{pmatrix} 0 & 3x^2+2y-z & 3x^2-2y+z \\ -3x^2-2y+z & 0 & -3x^2+2y+z \\ -3x^2+2y-z & 3x^2-2y-z & 0 \end{pmatrix}$$

こちらから入力した結果が反映されるか座標表示させてみよう。

```
dg1 = CoordRep[XD[w1]];
CollectForm[%] // OutputForm
```

```
(1 - 6 x) dx ^ dy ^ dz
```

正しく表示された。
ここで注意すべきは各成分の参照も次のように負にしておかないと参照しない。

```
Table[w1[-i, -j], {i, 1, 3}, {j, 1, 3}] // MatrixForm   // OutputForm
Table[w1[i, j], {i, 1, 3}, {j, 1, 3}] // MatrixForm   // OutputForm
```

```
0                 3 x^2 + 2 y - z    3 x^2 - 2 y + z
-3 x^2 - 2 y + z  0                  -3 x^2 + 2 y + z
-3 x^2 + 2 y - z  3 x^2 - 2 y - z    0

0       w1^12     w1^13
-w1^12  0         w1^23
-w1^13  -w1^23    0
```

*Mathtensor*はいくつかの表現方法を持ち、これらを切り替えることができる。
そこで成分を与えてない*1*形式の*s1,s2,s3*を用いて外微分の表現を共変微分を用いて表してみよう。
通常の微分は , で共変微分はセミコロン ; で表す。

■ **On[XDtoCDFlag]** 外微分から共変微分表現に切り替える。
Off[XDtoCDFlag] 外微分から共変微分表現に切り替えをやめる。

*1*形式から外微分で*2*形式をつくる。
表現の*On,Off*は次のように*CoordRep[]*を使う前に宣言しないといけない。

```
On[XDtoCDFlag]
df1 = CoordRep[XD[s1]];
df2 = CoordRep[XD[s2]];
df3 = CoordRep[XD[s3]];
df = df1 + df2 + df3;
CollectForm[df] // OutputForm
Off[XDtoCDFlag]
df1 = CoordRep[XD[s1]];
df2 = CoordRep[XD[s2]];
df3 = CoordRep[XD[s3]];
df = df1 + df2 + df3;
CollectForm[df] // OutputForm
```

$$(s1_{2,1} - s1_{1,2} + s2_{2,1} - s2_{1,2} + s3_{2,1} - s3_{1,2})\ dx \wedge dy +$$
$$(s1_{3,1} - s1_{1,3} + s2_{3,1} - s2_{1,3} + s3_{3,1} - s3_{1,3})\ dx \wedge dz +$$
$$(s1_{3,2} - s1_{2,3} + s2_{3,2} - s2_{2,3} + s3_{3,2} - s3_{2,3})\ dy \wedge dz$$

$$(s1_{2,1} - s1_{1,2} + s2_{2,1} - s2_{1,2} + s3_{2,1} - s3_{1,2})\ dx \wedge dy +$$
$$(s1_{3,1} - s1_{1,3} + s2_{3,1} - s2_{1,3} + s3_{3,1} - s3_{1,3})\ dx \wedge dz +$$
$$(s1_{3,2} - s1_{2,3} + s2_{3,2} - s2_{2,3} + s3_{3,2} - s3_{2,3})\ dy \wedge dz$$

■ XP[A,B]【AとＢの外積】

ここでこれまでの変数を初期化し、
改めてベクトルv1,v2,v3と(2,1)型のテンソルt1,t2,t3さらに
1形式s1,s2,s3, 2形式w1,w2,w3を定義する。
外積は次元に依存するのではじめに次元を決める。これまでは3次元だったので4次元にしてみよう。

```
Clear[v1, v2, v3, s1, s2, s3, w1, w2, w3, t1, t2, t3, f1, f2, f3];
Dimension = 4;
DefineForm[{s1, s2, s3}, {1, 1, 1}];
DefineForm[{w1, w2, w3}, {2, 2, 2}];
DefineTensor[{v1, v2, v3}, {{1, 1}, 1}];
DefineTensor[{t1, t2, t3}, {{2, 1}, 1}];
```

では次に外積はどうなるか確かめてみよう。

```
fw = XP[w1, w1]
fs = XP[s1, s2, s3]
fv = XP[v1, v2, v3]
ft = XP[t1, t2]
```

```
w1 ^ w1
```

```
s1 ^ s2 ^ s3
```

```
v1 v2 v3
```

```
t1 t2
```

微分形式で定義したものにはWedge作用素^が表現される。
2形式どうしの外積でつくったfwを成分であらわそう。
4形式になるので添え字を4つ指定する。

```
f1 = FtoC[fw, {la, lb, lc, ld}];
f2 = f1 // OutputForm
```

$$2\ w1_{ad}\ w1_{bc} - 2\ w1_{ac}\ w1_{bd} + 2\ w1_{ab}\ w1_{cd}$$

ではこのfwを外微分してみよう。
fwが4形式だから結果は5形式になってしまう。でも次元は4なので結果は次のようになる。

```
f2 = XD[fw]
```

```
0
```

2形式のw1であれば外微分し、成分で表すと3形式だから添え字を3つ指定して、

```
f3 = XD[w1]
FtoC[%, {la, lb, lc}] // OutputForm
```

```
dw1

w1      - w1      + w1
  ab;c      ac;b      bc;a
```

よってこの余次元は1である。CoXD[]を使うと外微分を作れる。

```
CoXD[f3]
FtoC[%, {la, lb}] // OutputForm
CoXD[w1]
FtoC[%, {la}] // OutputForm
```

```
delta(dw1)

-w1     p  + w1     p  - w1     p
   pa;b       pb;a       ab;p

delta(w1)

-w1     p
   pa;
```

■ **CoXD[{記号}]** 余外微分をつくる。

余外微分としてCoXD[]を使うこともできる。
4次元なので2形式の外積の外微分は0である。
また、テンソルやベクトルは通常の微分になる。
余外微分では

```
dw = XD[fw]
ds = XD[fs]
dv = XD[fv]
dt = XD[ft]
gw = CoXD[fw]
FtoC[%, {la, lb, lc}] // OutputForm
gs = CoXD[fs]
FtoC[%, {la, lb}] // OutputForm
```

```
0

s1 ^ s2 ^ (ds3) - s1 ^ s3 ^ (ds2) + s2 ^ s3 ^ (ds1)

v2 v3 (dv1) + v1 v3 (dv2) + v1 v2 (dv3)

t2 (dt1) + t1 (dt2)

delta(w1 ^ w1)

-2 w1     p  w1     + 2 w1     p  w1     - 2 w1     p  w1     - 2 w1     p  w1     + 2 w1     p  w1     - 2 w1
      bc;     pa          ac;     pb          ab;     pc          pc;     ab          pb;     ac          pa;

delta(s1 ^ s2 ^ s3)

s3     p  s1     s2     - s3     p  s1     s2     - s3     p  s1     s2     + s3     p  s1     s2     + s3     p  s1     s2     - s3     p  s1
  b;     a     p      a;     b     p      b;     p     a      p;     b     a      a;     p     b      p;

  s2     p  s1     s3     + s2     p  s1     s3     + s1     p  s2     s3     - s1     p  s2     s3     + s2     p  s1     s3     - s2     p
    b;     a     p      a;     b     p      b;     a     p      a;     b     p      b;     p     a      p;

  s1     p  s2     s3     + s1     p  s2     s3     - s2     p  s1     s3     + s2     p  s1     s3     + s1     p  s2     s3     - s1     p
    b;     p     a      p;     b     a      a;     p     b      p;     a     b      a;     p     b      p;
```

双対なHodge作用素も利用できる。
微分形式でも表現させてみよう。レビシビタテンソルが登場する。

```
Clear[f1, f2, f3, f4, g1, g2, g3];
f1 = HodgeStar[s1]
f2 = HodgeStar[w2]
f3 = HodgeStar[gw]
f4 = HodgeStar[gs]
g1 = FtoC[f1, {la, lb, lc}] // OutputForm
g2 = FtoC[f2, {la, lb}] // OutputForm
g4 = FtoC[f4, {li, lj}] // OutputForm
g5 = FtoC[f5, {li, lj}] // OutputForm
```

```
*(s1)

*(w2)

*(delta(w1 ^ w2))

*(delta(s1 ^ s2 ^ s3))

-(Epsilon    p s1 )
        abc    p

Epsilon    pq w2
        ab     pq
-------------------
         2
```

$$
\begin{aligned}
&-(s3_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{r}\,s2_{q}) + \frac{s3_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{r}\,s2_{q}}{2} + s3_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{q}\,s2_{r} - \\
&\frac{s3_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{q}\,s2_{r}}{2} + s2_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{r}\,s3_{q} - \frac{s2_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{r}\,s3_{q}}{2} - \\
&s1_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s2_{r}\,s3_{q} + \frac{s1_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s2_{r}\,s3_{q}}{2} - s2_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{q}\,s3_{r} + \\
&\frac{s2_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{q}\,s3_{r}}{2} + s1_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s2_{q}\,s3_{r} - \frac{s1_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s2_{q}\,s3_{r}}{2}
\end{aligned}
$$

$$
\begin{aligned}
&-(s3_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{r}\,s2_{q}) + \frac{s3_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{r}\,s2_{q}}{2} + s3_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{q}\,s2_{r} - \\
&\frac{s3_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{q}\,s2_{r}}{2} + s2_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{r}\,s3_{q} - \frac{s2_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{r}\,s3_{q}}{2} - \\
&s1_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s2_{r}\,s3_{q} + \frac{s1_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s2_{r}\,s3_{q}}{2} - s2_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s1_{q}\,s3_{r} + \\
&\frac{s2_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s1_{q}\,s3_{r}}{2} + s1_{p;}^{\ q}\,\text{Epsilon}_{ij}^{\ pr}\,s2_{q}\,s3_{r} - \frac{s1_{p;}^{\ p}\,\text{Epsilon}_{ij}^{\ qr}\,s2_{q}\,s3_{r}}{2}
\end{aligned}
$$

```
Tb = Table[g2, {la, 0, 3}, {lb, 0, 3}] // MatrixForm // OutputForm
```

$$\begin{pmatrix} 0 & \frac{\text{EpsilonTensorForm}[0]^{1pq}\ w2_{pq}}{2} & \frac{\text{EpsilonTensorForm}[0]}{2} & \frac{\text{EpsilonTensorForm}[0]^{3pq}\ w2_{pq}}{2} \\ -\frac{(\text{EpsilonTensorForm}[0]^{1pq}\ w2_{pq})}{2} & 0 & \frac{\text{Epsilon}^{12pq}\ w2_{pq}}{2} & \frac{\text{Epsilon}^{13pq}\ w2_{pq}}{2} \\ -\frac{(\text{EpsilonTensorForm}[0]^{2pq}\ w2_{pq})}{2} & -\frac{(\text{Epsilon}^{12pq}\ w2_{pq})}{2} & 0 & \frac{\text{Epsilon}^{23pq}\ w2_{pq}}{2} \\ -\frac{(\text{EpsilonTensorForm}[0]^{3pq}\ w2_{pq})}{2} & -\frac{(\text{Epsilon}^{13pq}\ w2_{pq})}{2} & -\frac{(\text{Epsilon}^{23pq}\ w2_{pq})}{2} & 0 \end{pmatrix}$$

```
s1 /: s1[-1, -1] = a1;
s1 /: s1[-2, 2] = a2;
s2 /: s2[1, 1] = b1;
s2 /: s2[2, 2] = b2;
XP[s1, s2]
FtoC[%, {la, lb}] // OutputForm
```

```
s1 ^ s2

-(s1  s2 ) + s1  s2
   b   a      a   b
```

次に４次元で定義してみよう。４次元の外微分は0になる。

```
Clear[f1]; Clear[f2]; Clear[f3]; Clear[f4]
f1 = XD[w1]
f2 = XD[w2]
f3 = XD[w3]
f4 = XD[w4]
```

```
dw1

dw2

dw3

dw4
```

次のように表現を共変微分形式で表示することができる。
外微分にしたがい符号が変化している。
最後の行では共変微分を通常微分で表現している。この場合はアフェイン接続項が０であることがわかる。

```
g1 = FtoC[f1, {la, lb}]
g2 = FtoC[f2, {la, lb, lc}]
g3 = FtoC[f3, {li, lj, lk, ll}]
g4 = FtoC[f4, {li, lj, lk, ll, lm}]
CDtoOD[g3]
```

$$-(w1_{a;b}) + w1_{b;a}$$

$$w2_{ab;c} - w2_{ac;b} + w2_{bc;a}$$

$$-\left(w3_{ijk;l}\right) + w3_{ijl;k} - w3_{ikl;j} + w3_{jkl;i}$$

$$0$$

$$-\left(w3_{ijk,l}\right) + w3_{ijl,k} - w3_{ikl,j} + w3_{jkl,i}$$

■ **XD[**記号 **]**【外微分】

外微分は次のようになる。４次元で定義しているので４次元の外微分は０になる。

```
f1 = XD[w1]
f2 = XD[w2]
f3 = XD[w3]
f4 = XD[w4]
```

$$dw1$$

$$dw2$$

$$dw3$$

$$0$$

すっきりとした表現でかえって来るがこれだと中身が見えない。そこで
次のように微分形式の表現を共変微分形式で表示することができる。
外微分にしたがい符号が変化していることがわかる。
さらにMathTensorは多様な表現を持っている。
最後の行では共変微分を通常微分で表現している。この場合はアフェイン接続項が０であることがわかる。

計量の作成と利用

■ ●計量の作成

ここまでは計量MetricG[]は一般的なgとして表現されてきたが、実際に相対論的な世界を研究する上では計量テンソルは自由に決めたいだろう。MathTensorを有用に利用するためにはよくつかうコマンド群や特に計量を作成しておいて、必要な時に読み込むのが便利である。そのために下記のコマンドを実行して、パッケージを作成、保存できるようにする。しかし、ここで重要な注意がいる。通常の **Mathtensor** のロードコマンド　**<<Mathtens.m**　を実行してはいけない。
Mathtensorの読み込み前に次のコンポーネントを作成し、実行する。そうしないとコマンドの定義が上書きされてしまうので注意がいる。
後はデフォルトの保存領域に結果が出力されるので以後はそのパッケージファイルを読み込めばよい。

ではまず計量パッケージを作成しよう。これにはMathtensorの読み込み前に次のコンポーネントを読み込む

```
<< Componen.m
```

```
Loading Components for DOS/Windows . . .



==================================================
MathTensor (TM) 2.2 (DOS/Windows(R)) (June 1, 1994)
             Components Package
by Leonard Parker and Steven M. Christensen
Copyright (c) 1991-1994 MathSolutions, Inc.
Runs with Mathematica (R) Versions 2.X.
Licensed to one machine only, copying prohibited.
==================================================
```

相対論では計量が重要な役割をはたす。
Mathtenorではやや煩雑だがこれを自由に定義し、利用できる。
その例を紹介しよう。

■ シュバルツシルドの時空

シュバルツシルドの時空の計量は次で与えられた。

$$ds^2 = \left(1 - 2G\frac{M}{r}\right)^{-1} dr^2 + r^2 d\theta^2 + r^2 \sin[\theta]^2 d\phi^2 - \left(1 - 2G\frac{M}{r}\right) dt^2$$

これをMathTensorの計量に記録させる。
シュバルツシルドの時空の計量を定義する。0でない成分は対角成分のみである。
Metricgの代入以後はアインシュタイン、リーマン、ワイルの曲率などを計算するかどうかのフラグになるので計算する場合は1としておこう。
次の内容をまとめて実行する。次元が上がると計量の定義が大変だが一度やれば後は上書きすればよい。

```
Dimension = 4;
x /: x[1] = r;
x /: x[2] = θ;
x /: x[3] = ϕ;
x /: x[4] = t;
Metricg /: Metricg[-1, -1] = (1 - 2 G M/r)^-1;
Metricg /: Metricg[-2, -1] = 0;
Metricg /: Metricg[-3, -1] = 0;
Metricg /: Metricg[-4, -1] = 0;
Metricg /: Metricg[-2, -2] = r^2;
Metricg /: Metricg[-3, -2] = 0;
Metricg /: Metricg[-4, -2] = 0;
Metricg /: Metricg[-3, -3] = r^2 Sin[θ]^2;
Metricg /: Metricg[-4, -3] = 0;
Metricg /: Metricg[-4, -4] = -(1 - 2 G M/r);
Rmsign = 1;
Rcsign = 1;
CalcEinstein = 1;
CalcRiemann = 1;
CalcWeyl = 1;
```

さらにフルパス指定で次のパッケージをデータフォルダに作成する。
名前は半角で覚え易いものにする。拡張子は.mを2つ、OutPut用に.outファイルを" ",で区切り指定する。

シュバルツシルドの時空計量を作成。1行目のエラーは無視してよい。

拡張して使えるようになったコマンドが示される。
うまくいかない時はMathTnsorのフォルダ内のComponen.mを直接実行させる。

```
Components["CompInSchw.m", "CompSchw.m", "CompSchw.out"]
```

```
Ｇｅ:trroｏｐｅｎComｐＩｎＳ cmを開くことができませんん
```

```
 The following tensors have been calculated and stored

 in the file CompSchw.m in InputForm, and

 in the file CompSchw.out in OutputForm:

 Metricg

MatrixMetricgLower

MatrixMetricgUpper

Detg

AffineG[ua,lb,lc]

RicciR[la,lb]

ScalarR

 EinsteinG[la,lb]

 RiemannR[la,lb,lc,ld]

 WeylC[la,lb,lc,ld]

 You can edit CompSchw.out to print a record of the results.
```

これで完了である。この計量のリーマンやワイルテンソルなどが利用できるわけだ。
読み込みの際には　<<Componen.mとする。

■ ドジッター時空

2つめの例として時間をｔ、空間を双曲線関数で定義するde Sitter時空を定義する。
これも場の理論でも登場する有用な時空である。
de Sitter時空をMathematicaで見るために次のような3次元の双曲線の関数を定義する。

```
Clear[Xd]
Xd[t_, r_, θ_, ϕ_] := {t, Cosh[r] Cos[θ] Cos[ϕ], Cosh[r] Cos[θ] Sin[ϕ], Cosh[r] Sin[θ]}
```

さらに光速ｃとして次の関係を付け加える。

```
r = c t
```

c=1として上の第一、から第四成分までをx1～x4に代入し、各座標成分による微分をd1～d4に代入する。

```
x1 = Part[Xd[t, r, θ, ϕ], 1] /. r → t
x2 = Part[Xd[t, r, θ, ϕ], 2] /. r → t
x3 = Part[Xd[t, r, θ, ϕ], 3] /. r → t
x4 = Part[Xd[t, r, θ, ϕ], 4] /. r → t
d1 = D[Xd[t, r, θ, ϕ], t] /. r → t
d2 = D[Xd[t, r, θ, ϕ], r] /. r → t
d3 = D[Xd[t, r, θ, ϕ], θ] /. r → t
d4 = D[Xd[t, r, θ, ϕ], ϕ] /. r → t
```

```
t
```

```
Cos[θ] Cos[ϕ] Cosh[t]
```

```
Cos[θ] Cosh[t] Sin[ϕ]
```

```
Cosh[t] Sin[θ]
```

```
{1, 0, 0, 0}
```

```
{0, Cos[θ] Cos[ϕ] Sinh[t], Cos[θ] Sin[ϕ] Sinh[t], Sin[θ] Sinh[t]}
```

```
{0, -Cos[ϕ] Cosh[t] Sin[θ], -Cosh[t] Sin[θ] Sin[ϕ], Cos[θ] Cosh[t]}
```

```
{0, -Cos[θ] Cosh[t] Sin[ϕ], Cos[θ] Cos[ϕ] Cosh[t], 0}
```

didjを行列表示すると次のようになる。

```
Gij = Table[Simplify[d[i].d[j]], {i, 1, 4}, {j, 1, 4}] // MatrixForm
```

$$\begin{pmatrix} -\text{Cosh}[t]^2 & 0 & 0 & 0 \\ 0 & \text{Sinh}[t]^2 & 0 & 0 \\ 0 & 0 & \text{Cosh}[t]^2 & 0 \\ 0 & 0 & 0 & \text{Cos}[\theta]^2\,\text{Cosh}[t]^2 \end{pmatrix}$$

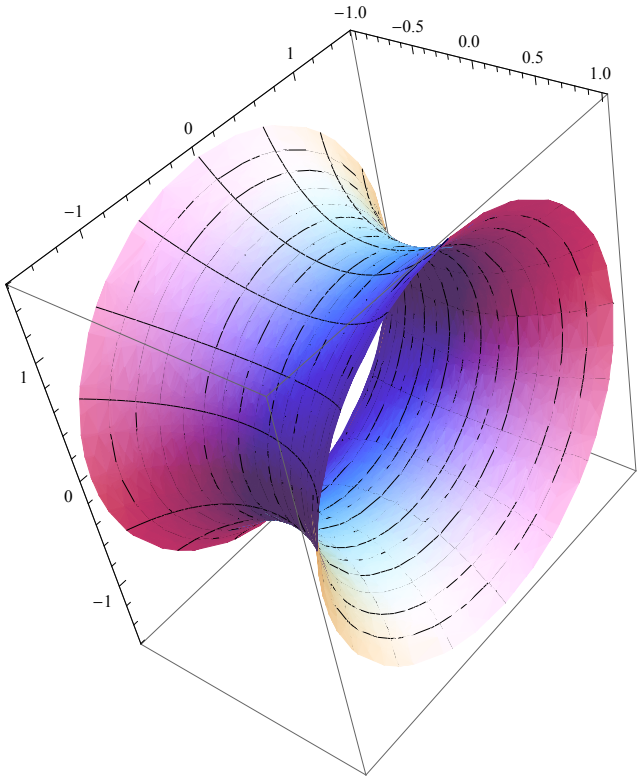
この条件のもとで次の関係が成り立つ

```
Nx = Limit[Simplify[√(x1^2 + x2^2 + x3^2 + x4^2)], t → 0]
```

```
1
```

ϕ=0として t軸(x1)とx2,x4のグラフを描かせる。

```
x2 = x2 /. ϕ → 0;
x4 = x4 /. ϕ → 0;
ParametricPlot3D[{t, x2, x4}, {t, -1, 1}, {θ, -π, π}]
```



ドジッター時空の3次元の様子が描かれた。
次にMathTensorにこの計量を定義する。先のdidjを利用しよう。
既にd1.d1等が得られていれば次を一気に実行させよう。

```
Dimension = 4;
x /: x[1] = r;
x /: x[2] = θ;
x /: x[2] = ϕ;
x /: x[3] = t;
Metricg /: Metricg[-1, -1] = FullSimplify[d1.d1]
Metricg /: Metricg[-2, -1] = 0;
Metricg /: Metricg[-3, -1] = 0;
Metricg /: Metricg[-4, -1] = 0;
Metricg /: Metricg[-2, -2] = FullSimplify[d2.d2]
Metricg /: Metricg[-3, -2] = 0;
Metricg /: Metricg[-4, -2] = 0;
Metricg /: Metricg[-3, -3] = FullSimplify[d3.d3]
Metricg /: Metricg[-4, -3] = 0;
Metricg /: Metricg[-4, -4] = FullSimplify[d4.d4]
Rmsign = 1;
Rcsign = 1;
CalcEinstein = 1;
CalcRiemann = 1;
CalcWeyl = 1;
```

```
1
```

```
Sinh[t]^2
```

```
Cosh[t]^2
```

```
Cos[θ]^2 Cosh[t]^2
```

次のコマンドでデータフォルダに適当な名称をつけ先と同様にこの計量をパッケージとして保存する。
その他、各自で時空を作成した場合は拡張子は変えずに同じように保存する。

```
Components["CompInDeSit.m", "CompDeSit.m", "CompDeSit.out"]
```

次のメッセージが最後に出ればパッケージの作成が成功している。
いったんMathematicaを終了し、再び起動し、Mathtensorをロードする

- ●パッケージの読み込みと利用例
- シュバルツシルドの時空

Mathtensorの初期設定についてはMathTensorインストールを参照

Mathtensorの読み込みがされてない場合はMahtensorのロードをする。

シュバルツシルドの時空の計量を定義する。0でない成分は対角成分のみである。

この結果と比較するために前章で既にカレントフォルダに作成してあるシュバルツシルド時空の定義パッケージを読み込む。
各自で作成した場合はファイル名を変えて、フルパス指定で読み込む。

```
<< CompSchw.m
```

```
MetricgFlag has been turned off.
```

計量テンソルGを確認してみよう。

- **Metricg[{成分}]** 計量テンソルの利用

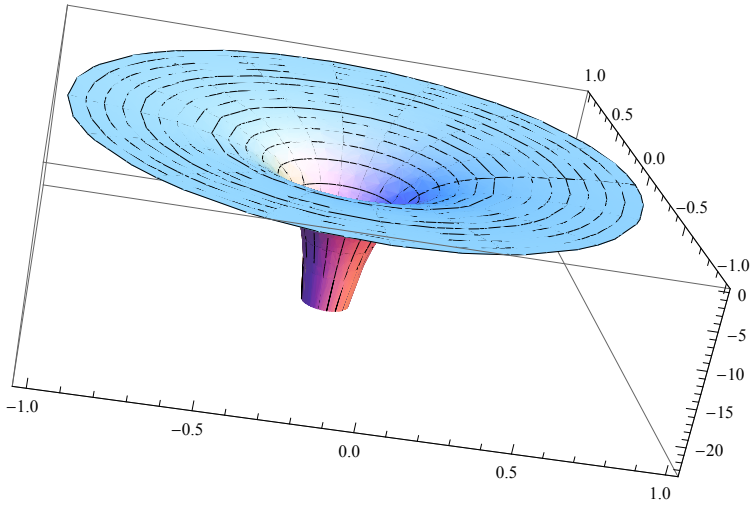
```
MG = Table[Metricg[-i, -j], {i, 4}, {j, 4}] // MatrixForm
```

$$\begin{pmatrix} \frac{1}{1-\frac{2GM}{r}} & 0 & 0 & 0 \\ 0 & r^2 & 0 & 0 \\ 0 & 0 & r^2\,\text{Sin}[\theta]^2 & 0 \\ 0 & 0 & 0 & -1+\frac{2GM}{r} \end{pmatrix}$$

球対照な重力分布を持ち、原点に質量Mが集中しているとしてシュバツルツシルドの時空計量が読み込めた。

x,y,z面でこの時空を部分的に眺めてみよう。

```
f = -pc[r] /. G → 1 /. M → 1 /. θ → π / 2;
RevolutionPlot3D[Part[f, 3], {r, 0.00001, 1}]
```



- **AffineG[{添え字}]** **Christoffel**の接続係数

まずChristoffelの係数を定義からたしかめて見よう。次のように計量テンソルから定義する。
次が成り立つから

$\Gamma_{ik,h} = \Gamma^{j}_{ik}\, g_{jh}$

$\Gamma^{i}_{jk} = \Gamma_{jk,h}\, g^{ih}$

次のように計量テンソルを用いて表すことができるがこれはテンソルではない。

$$(\Gamma^{a})_{bc} = \frac{1}{2} g^{ab} \left(g_{db,c} + g_{dc,b} - g_{bc,d}\right)$$

これからMathematiaで次のように定義する。

```
Af[i_, j_, k_] := Simplify[
  1/2 Sum[(D[Metricg[-j, -h], x[k]] + D[Metricg[-k, -h], x[j]] - D[Metricg[-j, -k], x[h]])
    Metricg[i, h], {h, 1, 4}]]
```

具体的な成分は次のようになる。

```
Af[1, 1, 1] // OutputForm
Af[1, 4, 4] // OutputForm
```

```
      G M
-----------------
2 G M r - r^2

G M (-2 G M + r)
----------------
       3
      r
```

この定義を使わなくてもMathTensorで作成した計量が使えるか確かめる。
次のように定義したファイルが読め込めていればAffineGで参照できる。マイナスは下付を表す。

```
AffineG[1, -1, -1]
AffineG[1, -4, -4]
```

```
      G M
-----------------
(2 G M - r) r

-2 G^2 M^2 + G M r
------------------
        3
       r
```

ただしく表示された。

■ RiemannR[{成分}]　リーマンテンソル

次にRiemannテンソルも確かめよう。
定義はアフィン係数を使って次のようになる。

```
Clear[Rm];
Rm[i_, j_, k_, l_] := Simplify[D[Af[i, j, l], x[k]] -
   D[Af[i, j, k], x[l]] + Sum[Af[p, j, l] Af[i, p, k], {p, 1, 4}] - Sum[Af[p, j, k] Af[i, p, l], {p, 1, 4}]]
```

MathTensorのコマンドで確かめる。

```
Rm[1, 3, 1, 3]
RiemannR[1, -3, -1, -3]
```

```
-(G M Sin[θ]^2 / r)

-(G M Sin[θ]^2 / r)
```

一致した。行列表示すると

```
Rim = Table[RiemannR[-i, -j, -i, -j], {i, 4}, {j, 4}] // MatrixForm
```

$$\begin{pmatrix} 0 & \frac{GM}{2GM-r} & \frac{GM\,\mathrm{Sin}[\theta]^2}{2GM-r} \\ \frac{GM}{2GM-r} & 0 & r^2 - r^2\,\mathrm{Cos}[\theta]^2 + 2GMr\,\mathrm{Sin}[\theta]^2 - r^2\,\mathrm{Sin}[\theta]^2 \\ \frac{GM\,\mathrm{Sin}[\theta]^2}{2GM-r} & r^2 - r^2\,\mathrm{Cos}[\theta]^2 + 2GMr\,\mathrm{Sin}[\theta]^2 - r^2\,\mathrm{Sin}[\theta]^2 & 0 \\ -\frac{2GM}{r^3} & \frac{-2G^2M^2+GMr}{r^2} & -\frac{GM(2GM-r)\,\mathrm{Sin}[\theta]^2}{r^2} \end{pmatrix}$$

リッチテンソルやスカラー曲率も求まる。

■ **RiemannR[{成分}]** リーマンテンソル
RicciR[{成分}] リーマンテンソル
ScalarR[{成分}] スカラー曲率

リッチテンソルは次のような成分になる。

```
Ric = Table[RicciR[-i, -j], {i, 4}, {j, 4}] // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 - \mathrm{Cot}[\theta]^2 + \mathrm{Csc}[\theta]^2 & 0 & 0 \\ 0 & 0 & 1 - \mathrm{Cos}[\theta]^2 - \mathrm{Sin}[\theta]^2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

この対角和がスカラー曲率だからシュバルツシルド時空では0になる。

```
Simplify[Sum[RicciR[-i, -i], {i, 1, 4}]]
```

```
0
```

Mathtensorで確かめると

```
ScalarR
Simplify[%]
```

$$-\frac{2\left(1 + \mathrm{Cot}[\theta]^2 - \mathrm{Csc}[\theta]^2\right)}{r^2}$$

```
0
```

と一致する。次にドジッターの時空を読み込む場合は一度Mathematicaを終了させ、起動した方がいいだろう。

■ ドジッターの時空

Mathtensorの初期設定についてはMathTensorインストールを参照

Mathtensorの読み込みがされてない場合はMahtensorのロードをする。

この結果と比較するために前章で既にカレントフォルダに作成してあるドジッター時空の定義パッケージを読み込む。
各自で作成した場合はファイル名を変えて、フルパス指定で読み込む。

```
<< CompDeSit.m
```

```
MetricgFlag has been turned off.
```

同様に計量テンソルGを確認してみよう。

```
MG = Table[Metricg[-i, -j], {i, 4}, {j, 4}] // MatrixForm
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{Sinh}[t]^2 & 0 & 0 \\ 0 & 0 & \mathrm{Cosh}[t]^2 & 0 \\ 0 & 0 & 0 & \mathrm{Cos}[\theta]^2\,\mathrm{Cosh}[t]^2 \end{pmatrix}$$

先のシュバツルツシルドの時空計量と異なりスカラー曲率は0にならない。

x,y,z面でこの時空を部分的に眺めてみよう。

```
Clear[f];
ScalarR
```

この条件のもとで次の関係が成り立つ

```
g4 = Sqrt[Metricg[-4, -4]]
```

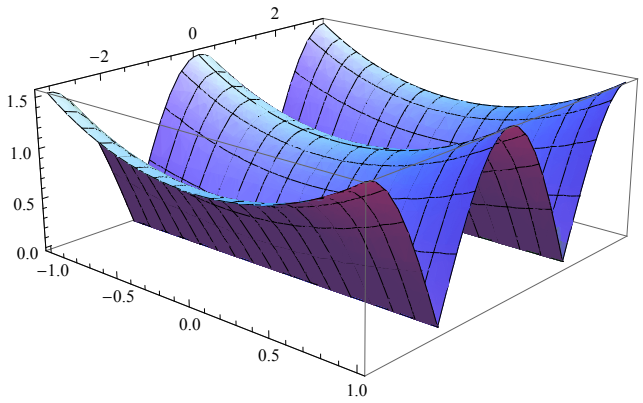
$$\sqrt{\text{Cos}[\theta]^2\,\text{Cosh}[t]^2}$$

```
Nx = Limit[Simplify[Sqrt[x1^2 + x2^2 + x3^2 + x4^2], t → 0]]
```

```
1
```

ϕ=0として t軸(x1)とx2,x4のグラフを描かせる。

```
Plot3D[g4, {t, -1, 1}, {θ, -π, π}]
```



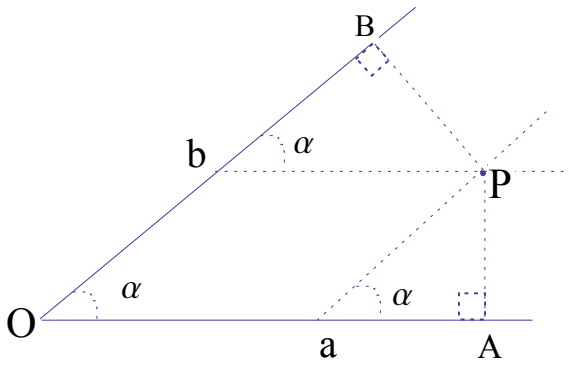
Tensor解析入門

■ 反変と共変

相対論の世界に入ると反変ベクトル、共変ベクトルが出てくる。さらにこれらを両方含むテンソルが登場するのである。
さて、初心者がはじめにぶつかる反変と共変の壁は他人と自分との間に共有できるものは何かということを考えると理解し易い？

数学の世界でベクトルは様々な座標系によりその姿を変える。何か他人や宇宙人？とコミュニケーションを考えた時、
どんな座標系からみても、いうならば誰から見ても同じに見えるものを基準にすべきだろう。
数学の世界ではそれは「数」で物理の世界ならそれは「光の速さ」というところだ。
そこであるベクトルと別なベクトルがあってそれらを決まった規則で演算させた時、この「数」が出てくれば
このベクトルの関係が「共変」「反変」となる。
物理でも磁場と電場の伝達から光りの速さが導かれる。
ある「数」をつくることのできる2つの関係は双対と呼ばれる。また、この数をスカラーという。

下図のように2次元の斜交座標を用いて点Pを特定する方法を考えよう。
これには平行線の交点であるa,bを指定する方法とPからの垂線であるA,Bを指定する方法がある。
前者を反変、後者を共変のやり方とここで命名しよう。変な？命名だが気にしないで進める。



このとき幾何的に各線分の長さについて次の関係が成り立つ。

$$\begin{aligned} OA &= Oa + ObCos\alpha \\ OB &= Ob + OaCos\alpha \end{aligned} \tag{1}$$

また、逆にこの2式から

$$\begin{aligned} Oa &= (OA - OBCos\alpha) \,/\, Sin^2\,\alpha \\ Ob &= (OB - OACos\alpha) \,/\, Sin^2\,\alpha \end{aligned} \tag{2}$$

また、逆にこの2式から反変のやり方でOPを表すと

$$OP = \sqrt{Oa^2 + Ob^2 + 2\,Oa\,Ob\,Cos\alpha} \tag{3}$$

共変のやり方でOPを表すと

$$OP = \sqrt{\left(OA^2 + OB^2 - 2\,OA\,OB\,Cos\alpha\right) \,/\, Sin^2\,\alpha} \tag{4}$$

ところが式 1 から共変のやり方と反変のやり方を混合して表すと

$$OP = \sqrt{OaOA + ObOB} \tag{5}$$

と簡単になる。混合して表すのもまんざらではない。

さてここで改めて上付き添え字で反変成分を

$$Oa = x^1,\ Ob = x^2 \tag{6}$$

また、下付き添え字で共変成分を

$$OA = x_1,\ OB = x_2 \tag{7}$$

と表すことにする。さらに変換を表す行列を計量と命名し、添え字i jを用いて(i=1,2 j=1,2)

$$g_{ij} = \begin{pmatrix} 1 & Cos\alpha \\ Cos\alpha & 1 \end{pmatrix} \tag{8}$$

またこの逆行列を上付き表現で

$$g^{ij} = \begin{pmatrix} 1 & Cos\alpha \\ -Cos\alpha & 1 \end{pmatrix} \frac{1}{Sin^2\,\alpha} \tag{9}$$

と定義しておくと便利である。式2はこの行列で次のように表現できる。

$$x^i = \sum_{i,j=1}^{2} g_{ij}\, x^j \tag{10}$$

$$x_i = \sum_{i,j=1}^{2} g^{ij}\, x_j \tag{11}$$

さらに OP^2 は式5から次のように表すことができる。

$$OP^2 = \sum_{i=1}^{2} x_i\, x^i = \sum_{i,j=1}^{2} g_{ij}\, x^i\, x^j = \sum_{i,j=1}^{2} g^{ij}\, x_i\, x_j \tag{12}$$

以後アインシュタインの既約に従い同じ添え字が複数出てきた場合はその添え字について和をとることにする。

これによりΣ記号を省略できるので少し、楽になる。

OPの長さに注目したが図においてPからの垂線であるPA,PBに注目すると幾何的に次の関係があることがわかる。

$$Oa\ PA = Ob\ PB = |\, a \wedge b\, | \tag{13}$$

ただし∧は外積を表すとする。

■ 平行移動

ベクトル v^a の平行移動をどうするかを考える。
相対論では座標移動に伴いベクトルの向きが変化することが考えられるがこれらは座標変化
Δx に比例するはずだから
a方向に移動させた場合、bc方向についての変化も考えないといけないので平行移動のベクトルを $(v^a)_{par}(x)$ で表すと

$$(v^a)_{par} = v^a(x + \Delta x) + (\Gamma^a)_{bc}(x)\, v^b(x)\, \Delta x^c$$

ここに登場した Γ がアフィン接続係数でテンソルにはならない。座標変換に依存する。
これから共変微分が次ぎのように定義できる。

$$(v^a)_{;c} = \lim_{\Delta x^c \to 0} \frac{(v^a)_{par} - v^a}{\Delta x^c} = (v^a)_{,c} + (\Gamma^a)_{bc}\, v^b$$

また、反変の場合は

$$(v_a)_{;c} = (v_a)_{,c} - (\Gamma^a)_{bc}\, v_b$$

と表すとこれは次が成り立てば長さが変化しない。

$$g_{ab;c} = 0$$

これからアフィン係数は次のように定義される。

$$(\Gamma^a)_{bc} = \frac{1}{2} g^{ad}\,(g_{db,c} + g_{dc,b} - g_{bc,d})$$

MathTensor ではアフィン係数は $G^a{}_{bc}$ で表現される。

■ 座標変換

先にコンポーネントを利用して計量の作成と利用を紹介したが
座標の変換規則を満たす任意のテンソルを作成することができる。
まずMathtenorがロードされていない場合はMathematicaバージョンに合わせてロードする。
Ver9以降では次のように入力する。

```
Unprotect[TensorQ]
Unprotect[RiemannR]
Remove[RiemannR]
Unprotect[Symmetrize]
Remove[Symmetrize]
<< Mathtens.m
```

```
{TensorQ}

{RiemannR}

{Symmetrize}

Loading MathTensor for DOS/Windows . . .
```

```
======================================================================

MathTensor (TM) 2.2.2 (Windows/Linux/Unix/MacOS X (R)) (July 25, 2011)

by Leonard Parker and Steven M. Christensen

Copyright (c) 1991-2011 MathTensor, Inc.

Runs with Mathematica (R) Versions 2.x-8.x

Licensed to machine soraduo.

======================================================================

No unit system is chosen. If you want one,

you must edit the file called Conventions.m,

or enter a command to interactively set units.

Units: {}

Sign conventions: Rmsign = 1 Rcsign = 1

MetricgSign = 1 DetgSign = -1

TensorForm turned on,
ShowTime turned off,
MetricgFlag = True.

========================================
```

まず平坦な4次元ユークリッド空間を考えよう。
MathTensorでは次のようにまず、次元を指定する。

```
Dimension = 4;
```

次に座標系を設定する。他に極座標などを選べる。

```
coord = {x, y, z, t}
```

```
{x, y, z, t}
```

次に変換規則を例えば次で決める。
ここでは簡単な1次変換にする。

```
trans = {x + K y, y, z, t}
```

```
{x + K y, y, z, t}
```

テンソルとして(2,1)型のテンソルTaを定義しよう。

```
DefineTensor[Ta, {{2, 1}, 1}]
```

```
PermWeightSymtm: Symmetry of Ta assigned
```

```
PermWeightdefObjectTadefined
```

このテンソルを与えられた4次元ユークリッド空間上で与えられた変換則を満たすように決める。
そのためにクロネッカーδを用いよう。

■ **Transform[テンソル,計量,座標系,変換式,対称性]**

```
Ttransform[Ta, Kdelta[la, lb], coord, trans, -1]
```

```
Components assigned to Ta
```

結果を行列表示で見ることにする

```
Ma = MatrixForm[Table[Ta[-i, -j], {i, 4}, {j, 4}]]
```

$$\begin{pmatrix} 1 & K & 0 & 0 \\ K & 1+K^2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

ちなみにKdeta[]は組み込み関数で次のように表される。

```
MatrixForm[Table[Kdelta[-i, -j], {i, 4}, {j, 4}]]
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

■ **Lie微分**

時空の座標をx^a で表すことにし、速度を v^a で表し、
時間は普通成分をもたないのでパラメタのように微少時間 ϵ で表すと

```
x'a = xa - ϵva
```

と時間変化 ϵ の間に変化することになる。
これをテンソルTに拡張すると次元の数だけ成分があるので次のような変換になる。, pはpによる微分を表す。

```
T'ab (x) = Tab (x) + ϵ (-va,p Tpb - vb,p Tap + Tab,p vp)
```

これからv方向へのLie微分が次のように定義できる。

```
LieDv (tab) = Limitϵ→0 (t'ab - tab)/ϵ = -va,p Tpb - vb,p Tap + Tab,p vp
```

同様にして下付成分は正の符号をつける。

```
LieDv (tab) = Limitϵ→0 (t'ab - tab□)/ϵ = vp,a Tpb + vp,a T ap + Tab,p vp
```

MathTensor でもLie微分を扱うことができる。
これを見るために次のように (2, 1) 型のテンソルtとベクトルvを定義しておこう。

```
DefineTensor[t, {{1, 2}, 1}]
DefineTensor[v, {{1}, 1}]
```

```
PermWeightdefObjecttdefined
```

```
PermWeightdefObjectvdefined
```

■ **LieD[テンソル,ベクトル]**　　ベクトル方向へのテンソルの共変微分

MathTensor でLie微分は次のように実行する。

```
f1 = LieD[t[ua, ub], v] ;
f1 // OutputForm
```

```
LieD  (t[a, b])
   v
```

このリー微分を共変微分と通常微分で表してみる。

■ **LieDtoCD[式]**　　**Lie**微分の式を共変微分で表す。
LieDtoOD[式]　　**Lie**微分の式を共変微分で表す。

```
f2 = LieDtoCD[f1];
f2 // OutputForm
f3 = LieDtoOD[f1];
f3 // OutputForm
```

```
-(v[a]   t[p, b]) - v[b]   t[a, p] + t[a, b]   v[p]
     ;p                ;p                ;p

-(v[a]   t[p, b]) - v[b]   t[a, p] + t[a, b]   v[p]
     ,p                ,p                ,p
```

さらに共変微分の表示をアフィン係数を使って表現する場合は次のようにする。

```
f4 = CDtoOD[f2];
f4 // OutputForm
```

```
-(v[a]   t[p, b]) - v[b]   t[a, p] + t[a, b]   v[p] - G[a]   t[q, b] v[p] -
     ,p                ,p                ,p        pq

  G[b]   t[a, q] v[p] + G[a]   t[p, b] v[q] + G[b]   t[a, p] v[q]
    pq             pq             pq
```

共形変換

次のような計量を作成するのでMathtensorをロードする前に以下を実行する。

ではまず計量パッケージを作成しよう。Mathtensorの読み込み前に次のコンポーネントを読み込む

```
<< Componen.m
```

```
============================================================
MathTensor (TM) 2.2.1 (UNIX/Linux/Mac OS X (R))
            Components Package
by Leonard Parker and Steven M. Christensen
Copyright (c) 1991-2009 MathTensor, Inc.
Runs with Mathematica (R) Version 2.x to 7.x
Licensed to machine soraduo.
============================================================
```

3 次元リーマン計量を例えば次のように決めてみる。

```
Clear[x, y, z];
Dimension = 3;
x /: x[1] = x;
x /: x[2] = y;
x /: x[3] = z;

f1 = 4 (x^2 + y^2 + z^2) / ((x^2 + y^2 + z^2) - x^2);
f2 = 4 (x^2 + y^2 + z^2) / ((x^2 + y^2 + z^2) - y^2);;
f3 = 4 (x^2 + y^2 + z^2) / ((x^2 + y^2 + z^2) - z^2);;
Metricg /: Metricg[-1, -1] = f1;
Metricg /: Metricg[-2, -1] = 0;
Metricg /: Metricg[-3, -1] = 0;

Metricg /: Metricg[-2, -2] = f2;
Metricg /: Metricg[-3, -2] = 0;

Metricg /: Metricg[-3, -3] = f3;
Metricg /: Metricg[-4, -3] = 0;

Rmsign = 1;
Rcsign = 1;
CalcEinstein = 1;
CalcRiemann = 1;
CalcWeyl = 1;

Components["CompInRiem.m", "CompRiem.m", "CompRiem.out"]
```

Get::noopen: Cannot open CompInRiem.m を開くことができません

```
 The following tensors have been calculated and stored

 in the file CompRiem.m in InputForm, and

 in the file CompRiem.out in OutputForm:

 Metricg

MatrixMetricgLower

MatrixMetricgUpper

Detg

AffineG[ua,lb,lc]

RicciR[la,lb]

ScalarR

 EinsteinG[la,lb]

 RiemannR[la,lb,lc,ld]

 WeylC[la,lb,lc,ld]

 You can edit CompRiem.out to print a record of the results.
```

■ 定曲率の空間

まず、ユークリッド空間 R^{n+1} 内の半径Rの球面を考える。

次のような計量を作成するのでMathtensorをロードする前に以下を実行する。

ではまず計量パッケージを作成しよう。Mathtensorの読み込み前に次のコンポーネントを読み込む

```
<< Componen.m
```

```
============================================================
MathTensor (TM) 2.2.1 (UNIX/Linux/Mac OS X (R))
            Components Package
by Leonard Parker and Steven M. Christensen
Copyright (c) 1991-2009 MathTensor, Inc.
Runs with Mathematica (R) Version 2.x to 7.x
Licensed to machine soraduo.
============================================================
```

```
Dimension = 4;
x /: x[1] = x;
x /: x[2] = y;
x /: x[3] = z;
x /: x[4] = t;
Metricg /: Metricg[-1, -1] = 4 R^2 / (R^2 + (x[1]^2));
Metricg /: Metricg[-2, -1] = 0;
Metricg /: Metricg[-3, -1] = 0;
Metricg /: Metricg[-4, -1] = 0;
Metricg /: Metricg[-2, -2] = 4 R^2 / (R^2 + (x[2]^2));
Metricg /: Metricg[-3, -2] = 0;
Metricg /: Metricg[-4, -2] = 0;
Metricg /: Metricg[-3, -3] = 4 R^2 / (R^2 + (x[3]^2));
Metricg /: Metricg[-4, -3] = 0;
Metricg /: Metricg[-4, -4] = 4 R^2 / (R^2 + (x[4]^2));
Rmsign = 1;
Rcsign = 1;
CalcEinstein = 1;
CalcRiemann = 1;
CalcWeyl = 1;

Components["CompInRiem1.m", "CompRiem1.m", "CompRiwm1.out"]
```

Get::noopen: CompInRiem1.m を開くことができません。

```
 The following tensors have been calculated and stored

 in the file CompRiem1.m in InputForm, and

 in the file CompRiwm1.out in OutputForm:

 Metricg

MatrixMetricgLower

MatrixMetricgUpper

Detg

AffineG[ua,lb,lc]

RicciR[la,lb]

ScalarR

 EinsteinG[la,lb]

 RiemannR[la,lb,lc,ld]

 WeylC[la,lb,lc,ld]

 You can edit CompRiwm1.out to print a record of the results.
```

これでこの計量が読み込まれたのでMathematicaを終了し、再起動させる。
Mathtensorの読み込みがされてない場合は次のようにMahtensorのロードをする。(Ver9以降の場合)

```
Unprotect[TensorQ]
Unprotect[RiemannR]
Remove[RiemannR]
Unprotect[Symmetrize]
Remove[Symmetrize]
<< Mathtens.m
```

```
{TensorQ}

{RiemannR}

{Symmetrize}
```

その後定義した計量を次で読み込む。

```
<< CompRiem.m


RicciR[la, lb] // OutputForm
Simplify[ScalarR] // OutputForm
RiemannR[la, lb, lc] // OutputForm
```

$$R_{ab}$$

$$R_{abc}$$

```
MG = Table[Metricg[i, j], {i, 3}, {j, 3}] // MatrixForm // OutputForm
AG = Table[AffineG[1, -j, -k], {j, 3}, {k, 3}] // MatrixForm // OutputForm
Rie = Table[Simplify[RiemannR[-i, -j, -i, -j]], {i, 3}, {j, 3}] // OutputForm
Ric = Table[Simplify[RicciR[-i, -j]], {i, 3}, {j, 3}] // OutputForm
```