測地線・曲率テンソルとSageMath

伊藤 誠[†]

1　はじめに

物理学を志す学生にとって、力学や電磁気学だけではなく、やはり量子力学や相対性理論を習得したいと思うものであろう。特に一般相対性理論は理解するのは難しいと言われるため返って挑戦したくなる場合もあるかも知れない。

ところで、一般相対性理論では空間が曲がっていると良く言われる。しかし3次元空間に住んでいて4次元空間を認識することが出来ない我々「3次元人」には、空間が曲がっているかどうかを外部の4次元の世界から見ることは出来ない。したがって、我々が住んでいる空間の曲率を認識することは難しい。地球を3次元的に俯瞰出来なかった時代に、人類が地球を丸いと認識出来なかったのと同様である。また曲線と曲面の教科書で曲率を学んだ後であっても、一般相対性理論で初めてリーマンの曲率テンソルに出会うと、それが4本の足(添字)を持つテンソルであるため、数学的な曲率テンソルをイメージすることも難しい。

一般相対性理論は時間軸も含めた4次元時空を考察するわけだが、4次元であるから曲率テンソルに4本の足が現れるわけではない。3次元ユークリッド空間内の2次元曲面であっても4本足の曲率テンソルは定義される。したがって、曲率テンソルを考察するのに、いきなり4次元時空を考える必要はない。最初は、3次元ユークリッド空間内の、2次元曲面から始めると良いであろう。3次元空間内の2次元曲面を、3次元の世界に住む「3次元人」から見ると、2次元曲面がどの様に曲がっているのかを認識するのは容易である。翻るに2次元曲面内に住む「2次元人」から2次元曲面はどの様に見えるのであろう。「2次元人」はもう一つの3次元方向を認識することは出来ないので、「2次元人」にとっては自身が住んでいる世界はあくまでも2次元(リーマン)平面としか認識できないであろう。本稿では「2次元人」がどの様に空間を認識するのか、3次元空間とどの様に関係しているのかを考察してみることにする。

以下の議論は文献[1]の第5章を基にしたが、文献[1]はあくまでも(抽象的な)数学のテキストであるため、内容を理解するための具体的な説明は乏しいものがある。そこで本稿では文献[1]には書かれていない事柄を、より具体的に考察することにより測地線や曲率をより良く理解することが目的である。なお式の導出等は文献[1]を参照して欲しい。また具体的な計算および作図(可視化)に関して、本稿ではフリーソフトウェアのSageMathを用いることにする。

[†]大阪産業大学 経済学部 経済学科 教授
草 稿 提 出 日 10月14日
最終原稿提出日 11月16日

一般相対性理論では物質やエネルギーの分布 (エネルギー運動量テンソル) と計量の整合性が取れるように、4 次元時空の計量がアインシュタイン方程式から決定される。しかし曲率だけを考察するのであれば物質やエネルギーの分布を考えることなく、純粋に微分幾何学の問題として捉えることが出来る。本稿では計量が与えられた空間の測地線と曲率を考察する。

2　SageMath

SageMath[1]は Mathematica や Maple のような数式処理システムである。有償の Mathematica や Maple とは異なり、SageMath はフリーソフトウェアであり、プログラミング言語 Python[2]で記述されている。SageMath ではブラウザを用いたユーザーインターフェイス (jupyter notebook や jupyter-lab) が利用可能である。図 1 は jupyter-lab の Launcher の画面である。

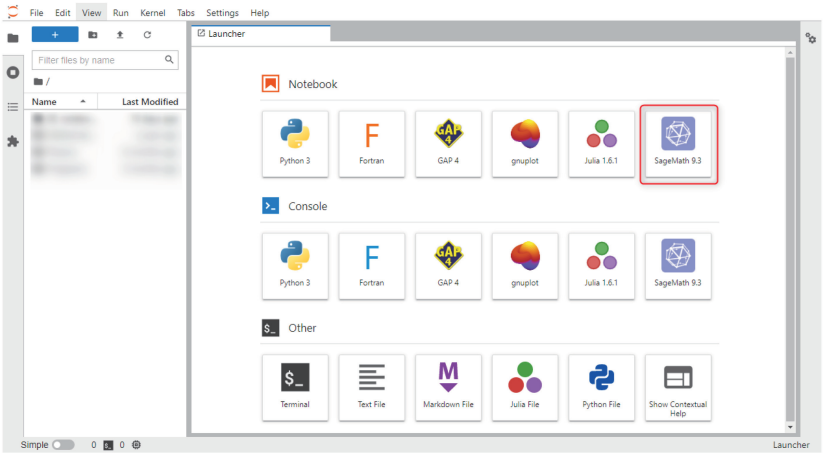

図 1: jupyter-lab の Launcher 画面

図 1 の四角で囲まれたアイコンをクリックすると、SageMath のノートブックが起動する (図 2)。ノートブック内のセルと呼ばれる部分に処理内容を記述することにより、対話的に数式処理を行うことが出来る。もちろん解析的に解けない場合は、数値的に処理することも可能である。SageMath の利用方法についてはやや内容は古いが文献 [2] を、jupyter については文献 [3] を見て欲しい。

[1]本稿の執筆に利用した SageMath は、ubuntu 20.04 上の SageMath9.4 である。
[2]正確には python3。

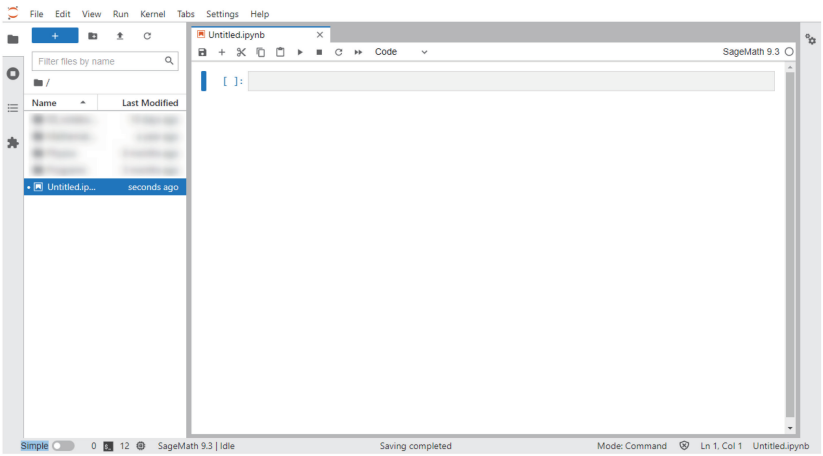

図 2: SageMath ノートブック

3　リーマン空間

文献 [1] によるとリーマン空間の定義は、

> **Definition 5.1.2.** A *Riemannian space* is a domain $U \subset \mathbf{R}^n$ equipped with a Riemannian metric g, and is denoted by (U, g).

である。また計量の構成方法として以下の命題がある [1]。

> **Proposition 5.2.3.** Let (U, g) be a Riemannian space, where $U \subset \mathbf{R}^n$ is a domain. Suppose that for a domain $V \subset \mathbf{R}^k (1 \leq k \leq n)$, we are given a smooth, one-to-one function $\phi : V \to U$ that is regular in the sense that for all $p \in V$,$(\phi_*)_p$ is one-to-one. Then the pullback $g_\phi = \phi^* g$ of g by ϕ is a Riemannian metric defined on V.

つまり、$V \subset \mathbf{R}^k$ は、既知のリーマン空間 (U, g) の計量 g を、関数 ϕ によって引き戻した (pullback) 計量 g_ϕ を持つリーマン空間 (V, g_ϕ) となる。

では次に本稿で考察する 2 次元曲面を見てみよう。

4　2 次元曲面と 2 次元リーマン平面

2 次元リーマン平面 $V \subset \mathbf{R}^2$ から、3 次元ユークリッド空間 $U \subset \mathbf{R}^3$ への関数 $\phi : V \to U$

$$\phi(u, v) = (u, v, u^2 + v^2) \tag{1}$$

を考える[3]。つまり 3 次元ユークリッド空間内で、この関数の像は回転放物面 (2 次元曲面) となる。3 次元ユークリッド空間の計量は

$$g_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

であるから、$V \subset \mathbf{R}^2$ の計量は、g_0 を ϕ で引き戻した

$$g_1 = \phi^* g_0 = \begin{bmatrix} 1+4u^2 & 4uv \\ 4uv & 1+4v^2 \end{bmatrix} \tag{3}$$

となる ([1] Example 5.2.4)。

ここで再び「2 次元人」に登場してもらう。3 次元ユークリッド空間内に住んでいる「3 次元人」から見た場合、「2 次元人」は計量 g_0 の 3 次元ユークリッド空間内の回転放物面の中に住んでいることになる。しかし「2 次元人」が認識出来る世界はあくまでも 2 次元であり、「2 次元人」自身は、式 (3) で与えられる計量 g_1 を持つ 2 次元リーマン平面 V 内に住んでいると認識することになるであろう。

5　接続係数

計量から導かれる様々な量を具体的に計算するには接続係数が必要となる。計量 g_1 の接続係数のうち 0 でないものは

$$\Gamma^1_{11} = \frac{4u}{1+4u^2+4v^2} \quad \Gamma^2_{11} = \frac{4v}{1+4u^2+4v^2}$$
$$\Gamma^1_{22} = \frac{4u}{1+4u^2+4v^2} \quad \Gamma^2_{22} = \frac{4v}{1+4u^2+4v^2}$$

であり、その他の接続係数は全て 0 である。接続係数の計算方法は種々あるが、詳細は文献 [1] を参照して欲しい。

6　測地線

まず計量 g_1 を用いて測地線を求めてみる。測地線 $c : I \to \mathbf{R}^2$ $(I \subset \mathbf{R})$ を、

$$c(t) = (c_1(t), c_2(t)),\ t \in I \tag{4}$$

[3]当然 3 次元ユークリッド空間 U は、(U, g_0) のリーマン空間でもある。

とすると、測地線の方程式は

$$\frac{d^2c_i}{dt^2} + \sum_{i,j} \Gamma^i_{jk} \frac{dc_j}{dt} \frac{dc_k}{dt} = 0 \tag{5}$$

である。計量 g_1 の場合、具体的には、

$$\ddot{c_1} + \left(\frac{4c_1}{1 + 4c_1^2 + 4c_2^2} \right) (\dot{c_1}^2 + \dot{c_2}^2) = 0$$
$$\ddot{c_2} + \left(\frac{4c_2}{1 + 4c_1^2 + 4c_2^2} \right) (\dot{c_1}^2 + \dot{c_2}^2) = 0$$

となる。ここで $\dot{c}_i = \dfrac{dc_i}{dt}$ 等である。この常微分方程式を解析的に解くのは難しい。文献 [1] には 3D-XplorMath-J を用いて作成した解の図 (Fig.5.8) が掲載されている[4]。ここでは実際に SageMath を用いて解を求め、作図してみることにする。SageMath で数値的に常微分方程式を解くには、関数 `desolve_system_rk4` を用いる [2]。ここでは数値解を求めるのと同時に、2 次元リーマン平面内の測地線を描いた (図 3)。文献 [1] の図 (Fig.5.8) と同様の図を描こうと試みたが、初期条件が不明であるため文献 [1] の図とは似て非なるものになっている。なおここで用いた初期条件は $c(0) = (-1, 0)$、$\dot{c}(0) = (0, -0.5)$ である。

```
var('t c1 c2 c1dot c2dot')
eqs = [c1dot, c2dot, -4 * c1 / (1 + 4 * c1^2 + 4 * c2^2) * (c1dot^2 + c2dot^2),
                     -4 * c2 / (1 + 4 * c1^2 + 4 * c2^2) * (c1dot^2 + c2dot^2)]
sol = desolve_system_rk4(eqs, [c1, c2, c1dot, c2dot], ics=[0, -1, 0, 0, -0.5], ivar=t, end_points=100)

# Plot
pos = [(c1, c2) for t, c1, c2, c1dot, c2dot in sol]
P1 = point(pos, size=20, color='red')
show(P1)
```

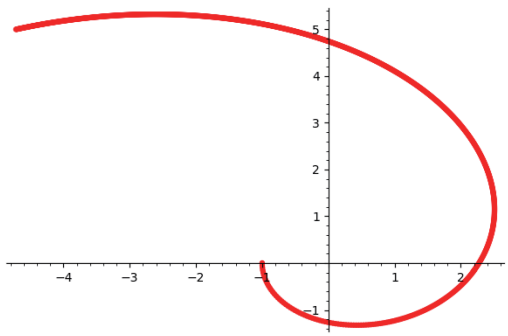

図 3: 2 次元リーマン平面

図 3 内で計算および作図に用いたプログラムは以下の通りである。

[4]文献 [1] の解説はここまでである。作図方法等についての言及はない。

```
var('t c1 c2 c1dot c2dot')
eqs = [c1dot, c2dot, -4 * c1 / (1 + 4 * c1^2 + 4 * c2^2) * (c1dot^2 + c2dot^2),
                     -4 * c2 / (1 + 4 * c1^2 + 4 * c2^2) * (c1dot^2 + c2dot^2)]
sol = desolve_system_rk4(eqs, [c1, c2, c1dot, c2dot], ics=[0, -1, 0, 0, -0.5],
                                                      ivar=t, end_points=100)

# Plot
pos = [(c1, c2) for t, c1, c2, c1dot, c2dot in sol]
P1 = point(pos, size=20, color='red')
show(P1)
```

図 3 の実線で表された測地線は、「2 次元人」が車 (もし存在すればばだが) に乗り、ハンドルを切ることなく走り続けた場合の、車の軌跡に相当する。「2 次元人」はひたすら真っすぐ走り続けているつもりなのだが、実際は自身が住んでいると認識している 2 次元平面内を、反時計回りに回転しながら進むことになる。

ではこの様子を 3 次元ユークリッド空間内の「3 次元人」が見たらどの様に見えるのであろうか。上で求めた測地線を 3 次元ユークリッド空間内に描いたのが図 4 である。

```
var('t c1 c2 c1dot c2dot')
eqs = [c1dot, c2dot, -4 * c1 / (1 + 4 * c1^2 + 4 * c2^2) * (c1dot^2 + c2dot^2),
                     -4 * c2 / (1 + 4 * c1^2 + 4 * c2^2) * (c1dot^2 + c2dot^2)]
sol = desolve_system_rk4(eqs, [c1, c2, c1dot, c2dot], ics=[0, -1, 0, 0, -0.5], ivar=t, end_points=100)

# Plot
pos3 = [(c1, c2, c1^2 + c2^2) for t, c1, c2, c1dot, c2dot in sol]
P1 = point(pos3, size=20, color='red')
P2 = plot3d(c1^2 + c2^2, (c1, -3, 3), (c2, -3, 3), alpha=0.3)
show(P1 + P2)
```

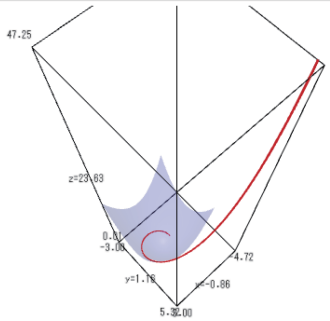

図 4: 3 次元ユークリッド空間

これは上述のプログラムの描画部分を以下のように修正し、3 次元ユークリッド空間内に測地線と同時に回転放物面を描いたものである。

```
# Plot
pos3 = [(c1, c2, c1^2 + c2^2) for t, c1, c2, c1dot, c2dot in sol]
P1 = point(pos3, size=20, color='red')
P2 = plot3d(c1^2 + c2^2, (c1, -3, 3), (c2, -3, 3), alpha=0.3)
show(P1 + P2)
```

図 4 では測地線と回転放物面との関係が分かり難いので、拡大した図が図 5 である。図 5 から明らかなように、「3 次元人」から見れば、回転放物面内に住んでいる「2 次元人」は、単に 3 次元ユークリッド空間内を、回転放物面の測地線 (直線に相当) に沿って、車で真っ直ぐ移動しているに過ぎないことが良く分かる。

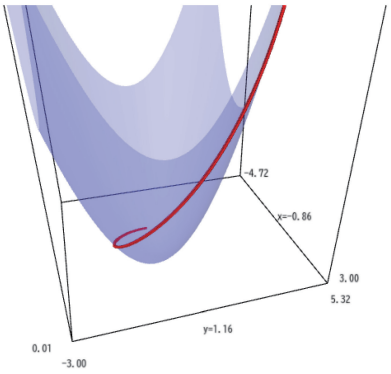

図 5: 測地線と回転放物面

ちなみに SageMath で描画した 3 次元の図はマウスを用いて自由に回転・拡大・移動することが出来るので、いろいろな角度から図を確認することが可能である。

7　曲率テンソル

では次に曲率について考察する。文献 [1] によれば、曲率の定義は

Definition 5.5.1. Let (U, g) be a Riemannian space with associated Riemannian connection ∇. Let $\chi(U)$ denote the set of smooth vector fields on U. The *curvature* of (U, g) is a map R that associates to each pair of vector fields $X, Y \in \chi(U)$ be map

$$R(X, Y) : \chi(U) \to \chi(U)$$

defined by

$$R(X, Y)(Z) = \nabla_X(\nabla_Y Z) - \nabla_Y(\nabla_X Z) - \nabla_{[X,Y]}Z.$$

である。この定義によれば、曲率はベクトル空間の間の写像で与えられる。またリーマンの曲率テンソルの定義は、

Definition 5.5.3. The *Riemannian curvature tensor* R is the (0,4)-tensor defined by

$$R(X, Y, Z, W) = g(R(X, Y)(Z), W)$$

for all vector fields X, Y, Z, W on U.

である。基底ベクトルを $\partial_i = \dfrac{\partial}{\partial x_i}$ とし、$X = \sum X^i \partial_i$、$Y = \sum Y^j \partial_j$、$Z = \sum Z^k \partial_k$ とした場合、定義 5.5.1 の曲率 R は

$$R(X,Y)(Z) = \sum_{i,j,k} X^i Y^j Z^k R(\partial_i, \partial_j)(\partial_k) = \sum_l \left(\sum_{i,j,k} X^i Y^j Z^k R^l_{ijk} \right) (\partial_l) \qquad (6)$$

と書ける [1]。ここで R^l_{ijk} は、変換後のベクトル場 $R(\partial_i, \partial_j)(\partial_k)$ の第 l 成分である。**Definition 5.5.3** との関係は

$$R_{ijkl} = R(\partial_i, \partial_j, \partial_k, \partial_l) = \sum_{t=1}^{n} R^t_{ijk} g_{tl} \qquad (7)$$

である [1]。物理学の教科書ではこの R^l_{ijk} をリーマンの曲率テンソルと呼ぶ [4]。

それでは計量が g_1 の場合の曲率を考察しよう。定義に従って点 (u, v) での曲率を計算すると、

$$R(\partial_1, \partial_2)(\partial_1) = R^1_{121}\partial_1 + R^2_{121}\partial_2 = \frac{16uv}{(1+4u^2+4v^2)^2}\partial_1 - \frac{4(1+4u^2)}{(1+4u^2+4v^2)^2}\partial_2 \qquad (8)$$

となる [1]。ここで $\partial_1 = \dfrac{\partial}{\partial u}$、$\partial_2 = \dfrac{\partial}{\partial v}$ である。引数としての基底ベクトルの組み合わせは他にも考えられるが、2 次元の場合独立なものは式 (8) だけである。

式 (8) は、写像 $R(\partial_1, \partial_2)$ によって、u 軸方向の基底ベクトル ∂_1 が右辺のベクトルへと変換されることを示している。右辺のベクトルを図示 (文献 [1](Fig.5.9)) するのは容易であるが、この写像の意味するところは何であろう。微分幾何学の教科書には、曲線と曲面の教科書にあるガウス曲率との関係を示しているものが多いようである[5]。ガウス曲率は 3 次元空間内の 2 次元曲面の曲がり具合を定量的に示したものであるので理解しやすいが、式 (8) にあるように、曲率を『ベクトル場をベクトル場へ変換する写像』であると考えると、意味が良く分からなくなってしまう。

では物理学の教科書には、曲率テンソルに関してどの様な説明がなされているのだろうか。文献 [4](P85) には、

> リーマンの曲率テンソルとは、ベクトルを微小面積要素 $dx^k dx^l$ の周りを一周させたとき、出発前のベクトルと帰ってきたベクトルがどれだけずれるかを示す量である。

と書かれている。任意の方法で一周させればそのずれは当然不定であるが、ここで一周させるとは、ベクトルを平行移動させて一周させるという意味である。また最後の「ずれるかを示す量」とは単位面積あたりの変化量のことである。この物理学の定義を考慮すると、式 (8) は『点 (u, v) における曲率 $R(\partial_1, \partial_2)$ は、ベクト

[5] 文献 [1] にもガウス曲率との関係が述べられているが、解説はそこまでである。

ル ∂_1 と ∂_2 が張る平面上の微小 (無限小) 面積要素の周りを、ベクトルを平行移動させながら一周させたとき、そのベクトルがどれだけずれるかを表す単位面積当たりのベクトルを与える写像である。』と解釈出来る。

では実際にベクトルを微小面積要素の周りを一周させて、上述の写像 $R(\partial_1, \partial_2)$ に対する解釈が正しいかどうかを確かめてみよう。その前にベクトルの平行移動の方程式を確認しておく。ベクトル V を曲線 c に沿って平行移動させるとその各成分は、方程式

$$\frac{dV^k}{dt} + \sum_{ij} \Gamma^k_{ij} \frac{dc_i}{dt} V^j = 0 \tag{9}$$

に従って変化する [1]。ちなみに $V^k = \dfrac{dc_k}{dt}$ としたものが測地線の方程式である。曲線を $c = (c_1, c_2)$ とすると、計量が g_1 である平面上での平行移動の方程式は

$$\dot{V^1} + \frac{4c_1}{1 + 4c_1^2 + 4c_2^2}\dot{c_1}V^1 + \frac{4c_1}{1 + 4c_1^2 + 4c_1^2}\dot{c_2}V^2 = 0$$
$$\dot{V^2} + \frac{4c_2}{1 + 4c_1^2 + 4c_2^2}\dot{c_1}V^1 + \frac{4c_2}{1 + 4c_1^2 + 4c_2^2}\dot{c_2}V^2 = 0$$

となる。まずはじめに、原点 $(u, v) = (0, 0)$ を始点として、ベクトルを微小面積要素の周りを平行移動しながら一周させる。ここでは、初期条件を $V = (V^1, V^2) = \partial_1 = (1, 0)$ とし、この初期ベクトルを図 6 のような半径 a の円の周りを、原点 (0,0) を出発点として反時計回りに一周させる[6]。微小面積要素の形状は問題ではないの

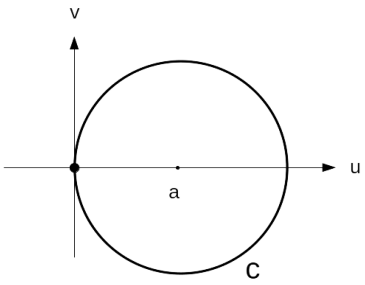

図 6: 微小面積要素

で、計算を簡略化するために円周を用いることにする。円周の式は、

$$c = (a - a\cos t, -a\sin t) \quad t \in [0, 2\pi], \tag{10}$$

であり、その導関数は

$$\dot{c} = (a\sin t, -a\cos t), \tag{11}$$

[6] ちなみに $R(\partial_2, \partial_1)(\partial_1)$ を求めるのであれば、∂_1 を時計回りに一周させれば良い。

である。a をだんだん小さくした時、円周を一周した後、初期ベクトル ∂_1 が最終的にどの様なベクトルになるのかを調べることにする。先程と同様に平行移動の微分方程式は SageMath を利用して数値的に解く。a の値を $1 \sim 0.001$ とした時、円周の周りを一周させた後の最終的なベクトルを求めるプログラムを以下に示す。

```
var('t V1 V2')
import numpy as np

Vfinal = []
for i in range(4):
    a = 1 / 10^i
    c1 = a - a * cos(t)
    c2 = - a * sin(t)
    c1dot = diff(c1, t)
    c2dot = diff(c2, t)
    G111 = 4 * c1 / (1 + 4 * c1^2 + 4 * c2^2)
    G122 = 4 * c1 / (1 + 4 * c1^2 + 4 * c2^2)
    G211 = 4 * c2 / (1 + 4 * c1^2 + 4 * c2^2)
    G222 = 4 * c2 / (1 + 4 * c1^2 + 4 * c2^2)
    eqs = [ -G111 * c1dot * V1 - G122 * c2dot * V2,
            -G211 * c1dot * V1 - G222 * c2dot * V2]
    sol = desolve_system_rk4(eqs, [V1, V2], ics=[0, 1, 0],
                                ivar=t, end_points=2*np.pi)
    Vfinal.append((sol[-1][1], sol[-1][2]))

for i in range(4):
    print(vector(Vfinal[i]))
```

実行結果は

```
(-0.196638524970895, 0.9804783732284775)
(0.9933402517502478, 0.1152178119109067)
(0.9999992118511006, 0.001255507018680614)
(0.9999999999210484, 1.25662575660254e-05)
```

となる。上から順に半径 a を 1、0.1、0.01、0.001 とした時の最終ベクトルである。最終ベクトルが次第に初期ベクトル $\partial_1 = (1, 0)$ に近づいているのが分かる。では初期ベクトルと最終ベクトルとの差を、単位面積当たりに換算した場合、どの様に変化するのであろうか。ベクトルの差を円の面積で割った時の変化を見てみよう。プログラムとしては最後の `for` 文の部分を

```
for i in range(4):
    a = 1 / 10^i
    print((vector((1,0))-vector(Vfinal[i])) / (np.pi * a^2))
```

と修正すればよい。結果は

```
(0.3809018726866248, -0.3120959593880249)
(0.21198637073913346, -3.6674968595706114)
(0.0025087558645327038, -3.996402962191766)
(2.5131082136591057e-05, -3.999964015597744)
```

となる。明らかに初期ベクトル $(1,0)$ との差が、ベクトル $(0,-4)$ に近づくことが分かる。式 (8) で $(u,v)=(0,0)$ とすると

$$R(\partial_1,\partial_2)(\partial_1)=R^1_{121}\partial_1+R^2_{121}\partial_2=-4\partial_2=(0,-4) \tag{12}$$

の様に ∂_1 は写像 $R(\partial_1,\partial_2)$ により $-4\partial_2$ に変換される。つまり計算結果は式 (8) と一致していることが分かる。このことにより上で述へた曲率 $R(\partial_1,\partial_2)$ の解釈が正しいことが確認出来、また物理学の教科書 [4] の説明と、**Definition 5.5.1** で定義される曲率が同等であることが明らかとなった。

初学者が、物理学の教科書で言うところの曲率テンソル R^l_{ijk} に初めて出会うと、その添字の多さに驚き、添字の意味を理解できずに困惑する。また、添字の意味を、**Definition 5.5.1** から導かれる式 (6) を見たとても、理解するのは難しい。しかし、本稿のようにベクトルを実際に微小面積要素の周りを一周させてみれば、その添字"$ijkl$" の意味は明確となる。つまりの i - j 平面内の微小面積要素の周りを k 軸方向の基底ベクトル $\partial_k=\dfrac{\partial}{\partial x^k}$ を平行移動させながら一周させた時、このベクトルと最終ベクトルとの差を表すベクトルの第 l 成分が R^l_{ijk} となる。このように、実際に計算を行うことにより、曲率テンソルの添字の意味を体感し、確認することが可能となる。またこの様な計算をを SageMath を利用して容易に実行することが出来る。

では任意の点 (u_0,v_0) での曲率を数値的に求めるにはどうしたらよいであろうか。これは円周の式 (10) を

$$c=(u_0+a-a\cos t,v_0-a\sin t) \tag{13}$$

とすれば良い。例えば $(u_0,v_0)=(1,1)$ での曲率を求めるには、プログラムを

```
c1 = 1 + a - a * cos(t)
c2 = 1 - a * sin(t)
```

と変更するだけである。プログラムをこの様に変更してベクトルの差を計算すると、$a=0.001$ のとき、ベクトルの差は、

```
(0.19726773652749433, -0.24658469406712896)
```

となる。式 (8) で $(u,v)=(1,1)$ とすると、

$$R(\partial_1,\partial_2)(\partial_1)=R^1_{121}\partial_1+R^2_{121}\partial_2=\frac{16}{81}\partial_1-\frac{20}{81}\partial_2\simeq(0.1975,-0.2469) \tag{14}$$

であるから、やはり計算結果は式 (8) と一致していることが分かる。

それでは次に、**Definition 5.5.3.** で定義される曲率テンソルはどのようになるのであろう。式 (7) は $R^i_{121}\,(i=1,2)$ に関して、

$$\begin{bmatrix}R_{1211} & R_{1212}\end{bmatrix}=\begin{bmatrix}R^1_{121} & R^2_{121}\end{bmatrix}\begin{bmatrix}1+4u^2 & 4uv\\ 4uv & 1+4v^2\end{bmatrix} \tag{15}$$

と書けるので、$(u,v)=(0,0)$ では、

$$\begin{bmatrix} R_{1211} & R_{1212} \end{bmatrix} = \begin{bmatrix} 0 & -4 \end{bmatrix} \tag{16}$$

であり、$(u,v)=(1,1)$ では、

$$\begin{bmatrix} R_{1211} & R_{1212} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{4}{9} \end{bmatrix} \tag{17}$$

となる。曲率テンソル R_{ijkl} は後ろの添字 kl に関して反対称

$$R_{ijkl} = -R_{ijlk} \tag{18}$$

なので [1]、R_{1211} は当然 0 になる。このことは SageMath を用いて、図 7 の様に簡単に確認することが出来る。教科書には対称性に関して、単に式 (18) が書かれているだけのことが多いが、SageMath を用いて実際に計算し、曲率テンソルの反対称性を容易に確認することが出来る。

```
%display latex
var('u v')
g1 = matrix([[1 + 4*u^2, 4*u*v], [4*u*v, 1 + 4*v^2]])
vector([16*u*v / (1 + 4*u^2 + 4*v^2)^2,
           -4*(1 + 4*u^2) / (1 + 4*u^2 + 4*v^2)^2]) * g1
```

$$\left(0,\ \frac{64\,u^2v^2}{\left(4\,u^2+4\,v^2+1\right)^2}-\frac{4\left(4\,u^2+1\right)\left(4\,v^2+1\right)}{\left(4\,u^2+4\,v^2+1\right)^2}\right)$$

図 7: 曲率テンソル

2 次元の場合、曲率テンソルの独立な成分は、R_{1212} だけである。曲率テンソルとガウス曲率との関係は、計量を

$$g = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \tag{19}$$

としたとき、

$$K = \frac{-R_{1212}}{g_{11}g_{22} - (g_{12})^2} \tag{20}$$

で与えられる [1]。したがって、今の場合、ガウス曲率は

$$K(u,v) = \frac{4}{(1+4u^2+4v^2)^2} \tag{21}$$

となり、よく知られた回転放物面のガウス曲率となる。

8　平行移動

最後に平行移動に関してもう少し考察をしてみる。例えば、初期ベクトル $\partial_1 = (1,0)$ を、原点を始点とした半径 1 の比較的大きな円周に沿って平行移動させた場合、最終ベクトルが、

```
(-0.196638524970895, 0.9804783732284775)
```

であることから、この最終ベクトルは初期ベクトルの方向とは全く異なる方向を向いていることが分かる。つまり最初 u 軸に沿っていたベクトルが、最終的に u-v 平面の第 2 象限内にあるのである。この現象を 3 次元ユークリッド空間で見た場合どの様に見えるのであろうか。

このことを調べる前に、2 次元リーマン平面内の点 $p = (u, v)$ におけるベクトル X と、3 次元ユークリッド空間内の回転放物面上のベクトル Y との関係を述べておく。ベクトル Y は、回転放物面上の点 $\phi(p)$ における接空間 $T_{\phi(p)}(S)(S = \phi(V))$ 内にあって、2 次元リーマン平面のベクトル X を関数 ϕ によって押し出した (pushforward) ベクトル

$$Y = \phi_* X = J(\phi)X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2u & 2v \end{bmatrix} \begin{bmatrix} X^1 \\ X^2 \end{bmatrix} = \begin{bmatrix} X^1 \\ X^2 \\ 2uX^1 + 2vX^2 \end{bmatrix} \tag{22}$$

となる。ここで $J(\phi)$ は ϕ のヤコビ行列である。

では円周上の $t = \pi/2$、$t = \pi$、$t = 3\pi/2$、$t = 2\pi$ における 3 次元ユークリッド空間内でのベクトルの変化を見てみよう。2 次元リーマン面上では順に

```
(0.736674968809306, 0.5212836818962254)
(0.153714883546341, 0.7735114788028638)
(-0.2560720282732448, 0.6248503783048281)
(-0.196638524970895, 0.9804783732284775)
```

であり、3 次元ユークリッド空間内では

```
(0.736674968809306, 0.521283681896225, 0.430782573826161)
(0.153714883546341, 0.773511478802864, 0.614859534185364)
(-0.256072028273245, 0.624850378304828, 0.737556700063167)
(-0.196638524970895, 0.980478373228478, 0.000000000000000)
```

となる。これを図 8 に示す。紙面上の図では分かり難いかも知れないが、初期ベクトルが各点で回転放物面上に接するように長さを変えずに移動していることが分かる (図 8 の太い矢印)。つまり 3 次元ユークリッド空間内ではベクトルが回転放物面に接したまま、曲線に沿って単に平行移動しているだけである。地球儀上に置かれた鉛筆を地球儀に接したまま、向きを保ちながら移動させることを思い浮かべると良いであろう。また、2 次元リーマン平面では、初期ベクトルが、2 次

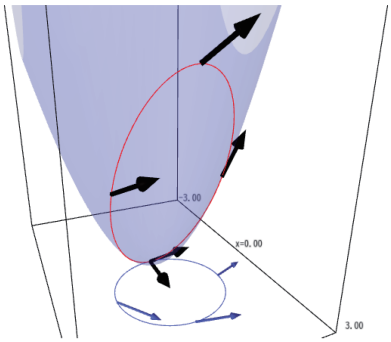

図 8: 平行移動

元ユークリッド平面上の平行移動とは異なる動きをしながら、円の周りを移動していることが分かる (図 8 の細い矢印)。つまり 2 次元ユークリッド平面内でベクトルを平行移動させると、ベクトルは長さと向きを変えずに平面内を移動するだけであるが、2 次元リーマン平面内のベクトルは図 8 の細い矢印のような動きをするのである。「2 次元人」は平行移動しているベクトルが、2 次元平面上でこの様な不思議な動きをするのを観測することになる。しかし 3 次元ユークリッド空間内の「3 次元人」が見たら、ベクトルが単に回転放物面上を平行移動しているのを見るに過ぎない。

9　まとめ

本稿の今までの議論で、3 次元ユークリッド空間内に住む「3 次元人」から見ると、測地線や平行移動と言った「2 次元人」にとっては理解に苦しむ現象であっても、容易にその本質を理解することが出来ることが分かった。残念ながら一般的な微分幾何学の教科書にも、一般相対性理論の教科書にも、本稿の様に具体的に測地線や平行移動を実際に計算し、2 次元リーマン平面内及び 3 次元ユークリッド空間内で視覚化したり、曲率を実際にベクトルを平行移動して計算しているものはない。本稿では、これらの点を容易に実現可能であることを示すことが出来、理解を深めるための手助けになることを示すことが出来た。

また本稿では計算および可視化するツールとしてフリーソフトウェアである SageMath を用いた。プログラミング・作図・数式処理システムを兼ね備えたものを、あるいはそれぞれが独立したものであっても、それらを利用することによって、教科書には説明されていないことを自らの手で理解を深める手助けとすることが出来る。SageMath はフリーソフトウェアであり、ブラウザを用いたユーザーインターフェイスが利用可能であるため、本稿で取り扱ったような内容は、誰でも自身のパソコンを使って自由に行うことが出来る。本稿で利用した SageMath の機能は、

そのほんの一部分だけであるが、有償の Mathematica や Maple と同様の処理を、フリーソフトウェアの SageMath で可能であることを示すことが出来た。

SageMath の特徴として、SageMath 自体がプログラミング言語 Python で記述されており、数式処理と Python プログラミングを同時に記述出来る点が挙げられる。例えば、本稿で示したプログラムでは、SageMath の機能を利用して

```
c1 = a - a * cos(t)
c2 = - a * sin(t)
c1dot = diff(c1, t)
c2dot = diff(c2, t)
```

の様に、曲線の導関数を解析的に求めて変数に代入している。当然 `c1dot` には $a\sin t$ が代入される。そしてそれらをまとめて、接続係数を計算したり、そのまま連立常微分方程式を数値的に求める関数に引数として渡している。それと同時に a の値を順に変えながら計算するために、Python の `for` ループを使用したり、計算結果を Python のリストに代入したりしている。この様に本稿に例示したプログラムは、SageMath と Python の機能を利用して記述してある (ただし SageMath と Python の文法の微妙な違いには注意が必要である。例えば Python では `10^3=9` であって、`10^3=1000` ではない[7])。Mathematica や Maple にも、それぞれのシステム内で利用可能なプログラミング言語があるのだが、それらの言語はその処理システム内だけでしか利用出来ない。その点 Python は汎用のプログラミング言語であるため、Python プログラミングの知識は様々な場面で利用可能である。

では最後に、3 次元の世界を 4 次元空間から眺めたらどの様に見えるのであろう。残念ながらこればかりは想像するしか方法はない。ただ本稿で触れたように「2 次元人」から見た測地線や、平行移動の不思議な振る舞いはその空間内にいたとしても認識することが出来る。つまり「2 次元人」は 3 次元空間を知らずして 2 次元空間の曲率を知ることが出来る[8]。「3 次元人」も同様に 3 次元空間内にいながらにして自身の空間の曲がり具合を知ることが出来る。ところで、人類が地球が丸いことを認識することが容易に出来たのは、そもそも我々が 3 次元空間に住んでいるからである。しかし、「3 次元人」の住む 3 次元空間が、例えば 4 次元ユークリッド空間内の 3 次元球面であったとしても、「3 次元人」が 4 次元方向を認識する術はないので、3 次元球面が余りにも巨大なものであれば、残念ながら、「3 次元人」が大局的な空間の曲がり具合を知るのは困難なことかも知れない。

本稿では、1 つの座標系だけで覆うことの出来る 2 次曲面 (回転放物面) だけを考察したが、これらの考察が、例えば複数の座標系でなければ覆えない球面のようなリーマン多様体 [7] や、一般相対性理論に現れる 4 次元時空 (擬リーマン空間)[8] に応用できるどうかを検証するのも興味深いことであろう。

[7] 演算子"^" は、SageMath ではべき乗だが、Python ではビット単位の排他的論理和である [5]。
[8] この事実は「ガウスの驚異の定理 (Theorema egregium)」として知られている [6]。

参考文献

[1] Andrew McInerney (2013) First Steps in Differential Geometry (Springer)

[2] Craig Finch (2011) Sage (Packt Publishing)

[3] 掌田津耶乃 (2018) データ解析ツール Jupyter 入門 (秀和システム)

[4] 佐藤勝彦 (1996) 相対性理論 (岩波書店)

[5] 柴田望洋 (2019) 新・明解 Python 入門 (SB クリエイティブ)

[6] 小林昭七 (1995) 曲線と曲面の微分幾何 (改訂版) (裳華房)

[7] 松本幸夫 (1988) 多様体の基礎 (東京大学出版会)

[8] M.P.Hobson, G.P.Efstathiou and A.N.Lasenby (2006) General Relativity An Introduction for Physicists (Cambridge University Press)

Understanding Geodesics and Curvature Tensor using SageMath

ITOH Makoto

Key Words: Riemannian Geometry, Differential Geometry, Geodesics, Curvature Tensor, Computer Algebra System

Abstract

The curvature tensor of 4-dimensional spacetime, which appears in general relativity, is difficult to understand because it is a complex tensor with four subscripts. Since it is impossible to visualize geodesics and curvatures in 4-dimensional spacetime, we visualized geodesics and calculate curvature tensors of 2-dimensional surfaces, which corresponds to a 2-dimensional Riemann plane in 3-dimensional Euclidean space in order to understand the meanings of geodesics and curvature tensors. For the visualization and the calculation, we used SageMath, which is a free computer algebra system software. We found that the visualization of these geodesics and the calculation of curvatures on a 2-dimensional surface in 3-dimensional Euclidean space is useful for understanding their meanings. The usefulness of SageMath is also clarified.