

# 4次元を見る

WITH MATHEMATICA by H.Y

予備知識：三角関数、虚数、(積分)対象：中3以上

4次元の世界をMathematicaのグラフィックの表現力を借りて見ていこう。

4次元を想像することは難しいのではじめに3次元の空間図形が2次元にどう映るかを見て、それを4次元に利用することを考える。

## 世界地図で遊ぶ

はじめに例として地球全体の地図を考えよう。3次元の中に地球があり、地表面の位置は緯度と経度で表すことができる。

これは地表面であれば2次元の座標の組で場所を表現できることを意味し、3次元実数空間  $R^3$  の中に2次元の球面  $S^2$  があると表す。まず、地球儀を地図にすることを考える。

### 【Mathematica入力】

次の2つのコマンドをshift+enterで入力してみよう。

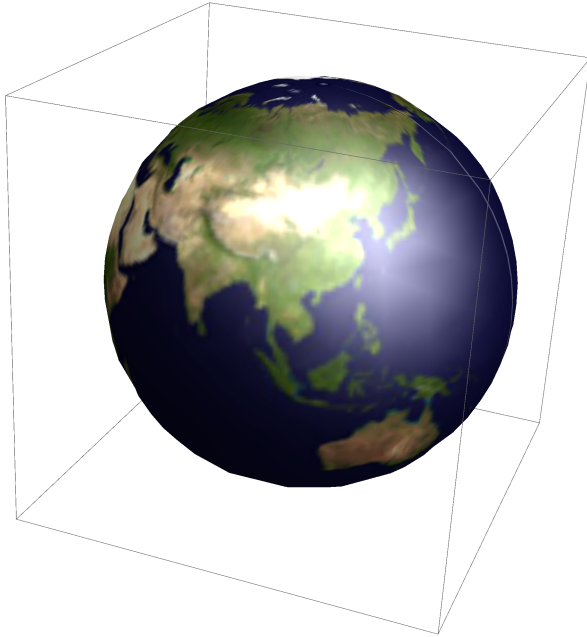
3次元の図形の表面にTextureを指定することで任意の絵や写真を張り付けることができる。3次元の図形は光源の位置や、視点を自由に変えることができる。ここではearthで指定した衛星四角形の衛星写真を球に張り付ける。

```
earth =  ;
```

```

ParametricPlot3D[{Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ]}, {θ, 0, π},
  {φ, 0, 2 π},
  PlotStyle → Directive[Specularity[White, 10], Texture[earth]],
  TextureCoordinateFunction → ({#5, 1 - #4} &), Lighting → "Neutral",
  Mesh → None, Axes → False, RotationAction → "Clip"]

```



これらを平面つまり $R^2$ 上の地図にしようとするときに、角度、長さを両方ともに正確に表すことができない。

よく知られているように、はじめのユークリッド図法では北極、南極側の面積はとて大きくなってしまふ。

#### 【Mathematica入力】 ☞

Mathematicaのデータベースには数学だけではなく、天文や地理のデータが多くある。次のコマンドでこれを利用することができる。

以下では世界地図を引用し、2つの投影法を用いているが、国別にしたり、別の投影図にしたり、国にいろをつけたり、人口などの情報を得ることもできる。以下のコマンドはインターネットにアクセスできないと使えない。また、回線によっては時間がかかる。

```
GeoGraphics[{}], GeoRange -> "World", GeoProjection -> "Equirectangular",
GeoGridLines -> Automatic]
GeoGraphics[{}], GeoRange -> "World", GeoProjection -> "Albers",
GeoGridLines -> Automatic]
```



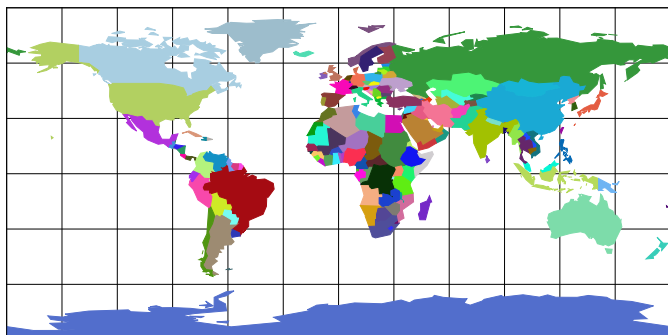
余談になるが地理のパッケージの読み込みとその利用例を少し紹介しておこう。

【Mathematica入力】 ☞

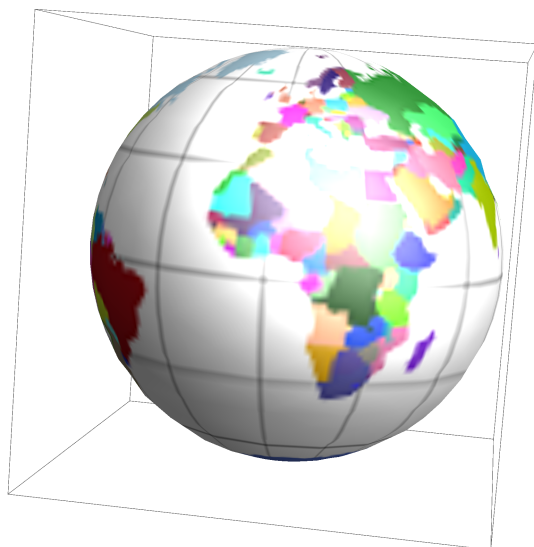
下のれ一例ではWorldPlotのパッケージを読み込み、その中の世界地図をgwに入れて、先と同じようにParametricPlot3Dに張り付ける方法である。

```
<< WorldPlot`
```

```
gw = WorldPlot[{World, RandomColors}]
```



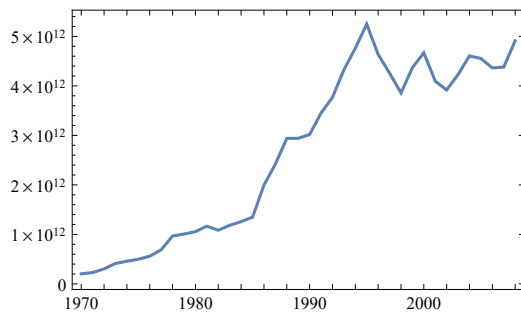
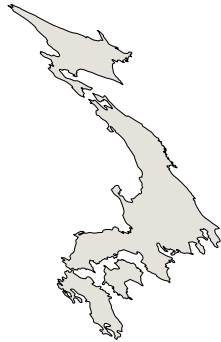
```
ParametricPlot3D[{Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ]}, {θ, 0, π},
  {φ, 0, 2 π}, PlotStyle → Directive[Specularity[White, 10], Texture[gw]],
  TextureCoordinateFunction → ({#5, 1 - #4} &), Lighting → "Neutral",
  Mesh → None, Axes → False, RotationAction → "Clip"]
```



### 【Mathematica入力】 ☞

パッケージを読み込まなくても次のCountryData[]コマンドは標準で使える。(Ver10)  
以下の例では日本の輪郭(Shape),とGDPの1970年から2010年までの推移を表示させた。

```
CountryData["Japan", {"Shape", "Mollweide"}]  
DateListPlot[CountryData["Japan", {"GDP"}, {1970, 2010}]]
```



#### 【Mathematica入力】

詳しくはHelpを参照するとよい。またMathematicaのサイトには豊富なデモプログラムがあり、利用例を学習できる。

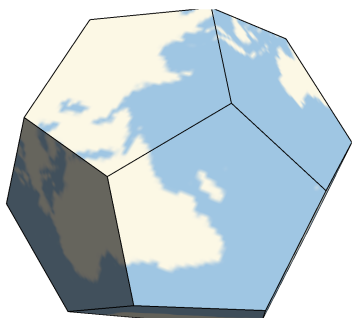
以下の例はその1つで多面体に世界地図を貼りこむことができる。

```

PolyhedronProjection[polyhedron_] :=
Module[{pts3D, center, pts2D, proj, pts2Dprojected, geographics,
  plotrange, pts2Dscaled, rescale},
  rescale[{x_, y_}, {xs_, ys_}] := {Rescale[x, xs], Rescale[y, ys]};
  Graphics3D[{
    pts3D = First[#];
    center = Mean[pts3D];
    center = GeoPosition[GeoPositionXYZ[center, Norm[center]]];
    pts2D = GeoPosition[GeoPositionXYZ[pts3D, Norm[pts3D[[1]]]]];
    proj = {"Gnomonic", "Centering" → center};
    pts2Dprojected = Most /@ GeoGridPosition[pts2D, proj][[1]];
    geographics = GeoGraphics[
      {Opacity[0], center, GeoPath[pts2D[[1]], CurveClosed → True]},
      GeoProjection → proj, GeoZoomLevel → 1];
    plotrange = PlotRange /. AbsoluteOptions[geographics, PlotRange];
    pts2Dscaled = rescale[#, plotrange] & /@ pts2Dprojected;
    {Texture[ImageData[Rasterize[geographics[[1]], "Image"]]},
      Polygon[pts3D, VertexTextureCoordinates → pts2Dscaled]} & /@
    N@Normal[PolyhedronData[polyhedron, "Faces"]][[1]],
    Lighting → "Neutral", Boxed → False, Method → {"ShrinkWrap" → True},
    ImageSize → Small]]

```

```
PolyhedronProjection["Dodecahedron"]
```



上の例では1面の5角形では平面になった地図をみることができが、全地球の地図を1面で見ることができない。

もちろん展開図にしてみれば見ることができるとぎれとぎれになりみにくいであろう。そこで3次元の中にある球の表面全てをできるだけ、スムーズに平坦な平面に写す方法はあるだろうか。

鉛筆片手にいろいろ試してほしい。次の節でこの問題を取りあげよう。

## 立体射影

3次元の立体を2次元にしてみるには影をみるようにその立体をある方向から見たものを平面に写す「射影」という操作をする。次の多面体の射影のデモプログラムを見ればすぐに理解できるだろう。

## 【Mathematica入力】

WolframのデモのサイトにIzidor Hafner氏の「Three Orthogonal Projections of Polyhedra」というプログラムを以下に引用する。プログラムの中身の理解は後にして、射影が何なのか実際に実行することで確認してほしい。

若干著者によってプログラムを変更してある。

```

Manipulate[
Module[{cc, cc1, py1, pyss},
cc1 = With[{cu = PolyhedronData[p, "Faces"],
cu1 = PolyhedronData[p, "Edges"]},
{cu[[2, 1]], cu[[1]] // N, cu1[[2, 1]]}];
cc = translate23[{xx, yy, zz}][cc1];
pyss = pys[cc];
py1 = visib2[cc, {0, 100, 0}, pyss];
Column[
{Graphics3D[
{{GrayLevel[0.6],
Line[{{0, 0, 0}, {5+xx, 0, 0}, {5+xx, 0, 5+zz},
{0, 0, 5+zz}, {0, 0, 0}}]}],
{GrayLevel[0.6],
Line[{{0, 0, 0}, {0, yy+5, 0}, {0, yy+5, 5+zz},
{0, 0, 5+zz}, {0, 0, 0}}]}],
{GrayLevel[0.6],
Line[{{0, 0, 0}, {5+xx, 0, 0}, {5+xx, 5+yy, 0},
{0, 5+yy, 0}, {0, 0, 0}}]}], visib2[cc, {0, 0, 100}, pzs[cc]],
visib2[cc, {0, 100, 0}, pys[cc]], visib2[cc, {100, 0, 0}, pxs[cc]],
If[shs, show[If[ef == 1, Polygon, ClosedLine]][cc], {}]},
ViewPoint -> {3, 3, 1}, ViewAngle -> 15 Degree, Boxed -> False,
ImageSize -> {400, 400}, Lighting -> "Neutral",
SphericalRegion -> True,
PlotRange -> {{-.5, 4.7}, {-.5, 4.7}, {-2, 4.7}}]}],
{{p, "SquashedDodecahedron", "polyhedron"},
Intersection[PolyhedronData["Convex"], PolyhedronData[ ; 14]]},
Row[{
"translate ",
Column[{
Control@{{xx, 2, Style["x", Italic]}, 1, 3, .001, ImageSize -> Tiny,
Appearance -> "Labeled"},
Control@{{yy, 2, Style["y", Italic]}, 1, 3, .001,
ImageSize -> Tiny, Appearance -> "Labeled"},
Control@{{zz, 2, Style["z", Italic]}, 1, 3, .001,
ImageSize -> Tiny, Appearance -> "Labeled"}
}
}

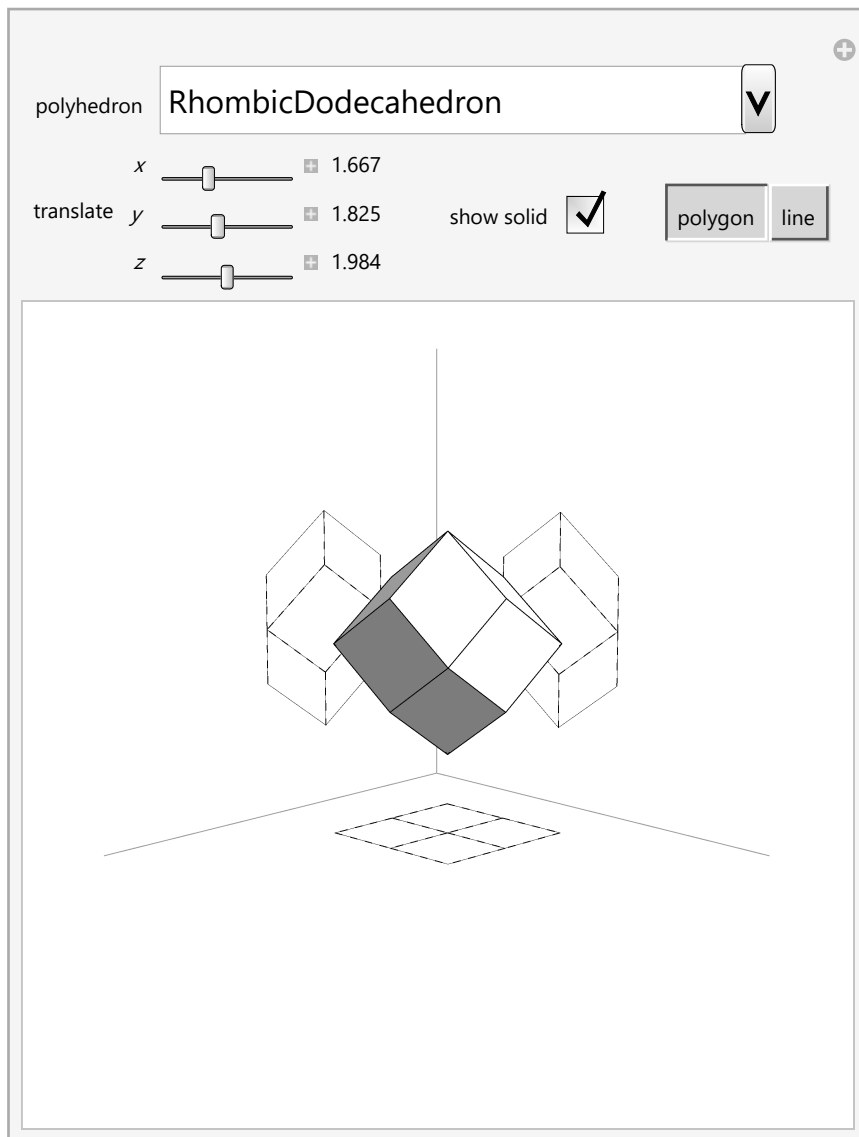
```

```

    ]],
  Spacer[20],
  Control@{{shs, True, "show solid"}, {True, False}},
  Spacer[20],
  Control@{{ef, 1, ""}, {1 → "polygon", 2 → "line"}, Enabled → shs}
}], SaveDefinitions → True, Initialization ⇒ {
  px[r_] := {0, r[[2]], r[[3]]};
  py[r_] := {r[[1]], 0, r[[3]]};
  pz[r_] := {r[[1]], r[[2]], 0};
  pxs[solid_] := {solid[[1]], Map[px, solid[[2]]], solid[[3]]};
  pys[solid_] := {solid[[1]], Map[py, solid[[2]]], solid[[3]]};
  pzs[solid_] := {solid[[1]], Map[pz, solid[[2]]], solid[[3]]};
  show[poly_][solid_] := Map[poly, Map[solid[[2, #]] &, solid[[1]]];
  trans[vec_][r_] := r + vec;
  translate3[vec_, solid_] :=
    {solid[[1]], Map[trans[vec], solid[[2]]], solid[[3]]};
  translate23[vec_][solid_] := translate3[vec, solid];
  ClosedLine[a_] := Line[Append[a, First[a]]];
  visible[solid_, view_][edge_] :=
    Module[{edges = solid[[3]], vert = solid[[2]], faces = solid[[1]],
      f = Length[solid[[1]]], fac, normals},
      fac = Select[Range[f],
        Length[Intersection[faces[[#]], edge]] == 2 &];
      normals =
        Table[Normalize[
          Cross[vert[[faces[[fac[[i]], 2]]]] -
            vert[[faces[[fac[[i]], 1]]]],
          vert[[faces[[fac[[i]], 3]]]] - vert[[faces[[fac[[i]], 1]]]]],
          {i, 1, 2}];
      view.normals[[1]] > 0.001 || view.normals[[2]] > 0.001];
  visib2[solid_, viewp_, prsolid_] :=
    Table[{If[! visible[solid, viewp][solid[[3, j]]], Dashed, Thin],
      Line[Map[prsolid[[2, #]] &, prsolid[[3, j]]]],
      {j, 1, Length[solid[[3]]]}}];
}]]

```





☞単純なプリズムから多面体まで多くの3つの平面への射影図を見ただろう。各壁に写った3つの射影図はベクトルの3つの成分のように独立していてこの3つがあれば、3Dプリンターなどで実体化できるわけだ。

しかしここでは、実際の物体は1つなので3枚の設計図ではなく、1枚で表す方法を考えたのである。

...アイデアは浮かんだだろうか？

☞球面上の点を平坦な平面に写す方法の1つとして立体射影を紹介する。まずは実物をMathematicaで実感してほしい。







#### 【Mathematica入力】

WolframのデモのサイトにErik Mahieu氏の「Inverse Stereographic Projection of Simple Geometric Shapes」

というプログラムを以下に引用する。ここでもプログラムの中身の理解は後にして、立体

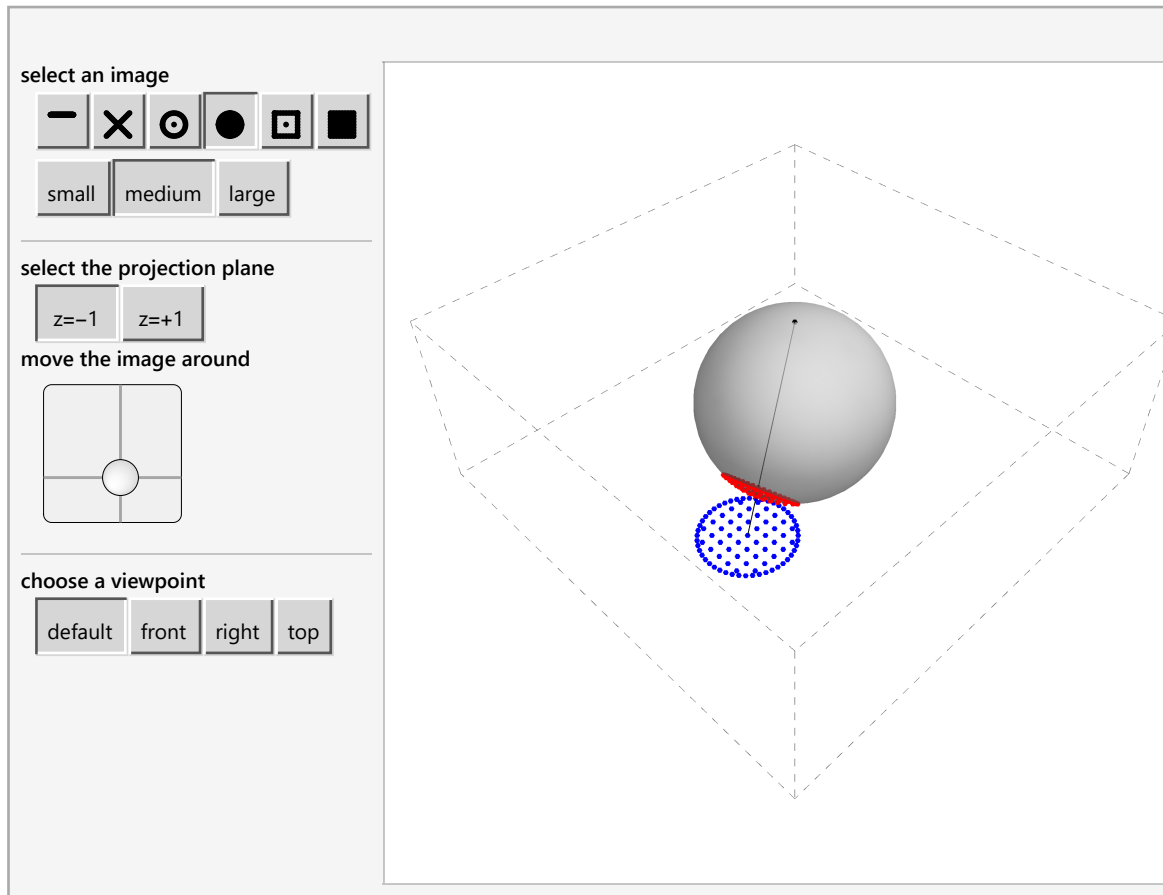
射影が何なのか実際に実行することで確認してほしい。

```

Manipulate[
  Graphics3D[{{GrayLevel[.75], Opacity[.65], Sphere[]},
    {Point[{0, 0, -z]}}, {Line[{{0, 0, -z}, {pos[[1], pos[[2], z]}]}},
    {Blue, Point/@(image[r, pos] /. {x_, y_} → {x, y, z})},
    {Red, Point/@(inverseStereo[#1, z] &) /@image[r, pos]}},
  PlotRange → {{-2 - r, 2 + r}, {-2 - r, 2 + r}, {-1.01, 1.01}},
  BoxRatios → {2 + r, 2 + r, 1}, BoxStyle → Dashed, Lighting → "Neutral",
  ViewPoint → Dynamic[vp], ImageSize → {400, 400}],
Style["select an image", Bold],
{{image, circle, ""}, {line → , cross → , circle → ,
  filledCircle → , square → , filledSquare → }, SetterBar},
{{r, .5, ""}, {.2 → "small", .5 → "medium", 1 → "large"}},
Delimiter,
Style["select the projection plane", Bold],
{{z, -1, ""}, {-1 → " z=-1 ", 1 → " z=+1 "}},
Style["move the image around", Bold],
{{pos, {1.35, -1.5, ""}, {-2, -2}, {2, 2}},
  Delimiter,
Style["choose a viewpoint", Bold],
{{vp, {1, -1, 1.25}, ""},
  {{1, -1, 1.25} → "default", Front → "front", Right → "right",
  Top → "top"}},
ControlPlacement → Left,
AutorunSequencing → {{4, 10}, {1, 2}, {2, 2}, {3, 2}, {5, 2}},
Initialization ⇒ (
  inverseStereo[{x_, y_}, z_] := {4 x, 4 y, z (4 - #)} / (# + 4) & [x^2 + y^2];
  line[r_, pos_] := (pos + #1 &) /@ Table[{i, 0}, {i, -50, 50, 0.25}];
  cross[r_, pos_] :=
    (pos + #1 &) /@ (r #1.RotationMatrix[π / 4] &) /@
      Flatten[{Table[{i, 0}, {i, -1, 1, .1}],
        Table[{0, i}, {i, -1, 1, .1}], 1];
  circle[r_, pos_] :=
    (pos + #1 &) /@Join[r Table[{Cos[γ], Sin[γ]}, {γ, -π, π, π / 24}],
      {{0, 0}}];
  filledCircle[r_, pos_] :=
    Join[circle[r, pos], Select[filledSquare[r, pos],
      Norm[pos - #1] ≤ r &]];
  square[r_, pos_] :=
    (pos + #1 &) /@
      (r Flatten[Table[#, {i, -1, 1, .25}] & /@
        {{i, 1}, {i, -1}, {-1, i}, {1, -i}, {0, 0}}, 1]);

```

```
filledSquare[r_, pos_] :=
  (pos + #1 &) /@
  (r Flatten[Table[Table[{{i, j}}, {i, -1, 1, .25}], {j, -1, 1, .25}],
    1]))]
```



実行してみると  $z=-1$  の北極点、 $z=1$  の南極点から球に接している平面に向けて直線が伸びている。

この直線の球面との交点が赤、接平面との交点が青で示される。

球面上の点が円を描けば面白いことに平面上の点も円を描く。

selectボタンを直線にすると接平面に直線を描く、特別な円が球面上にあることが確認できるだろう。

またxボタンを選ぶと球面上で直交する直線は接平面でも直交していることがわかる。

このプログラムでは描画する領域を限っているが無限大まで接平面を広げれば球面上のあらゆる点を接平面に写すことができる？

おっと、1点だけ不可能な点が球面に存在する。わかるだろうか？

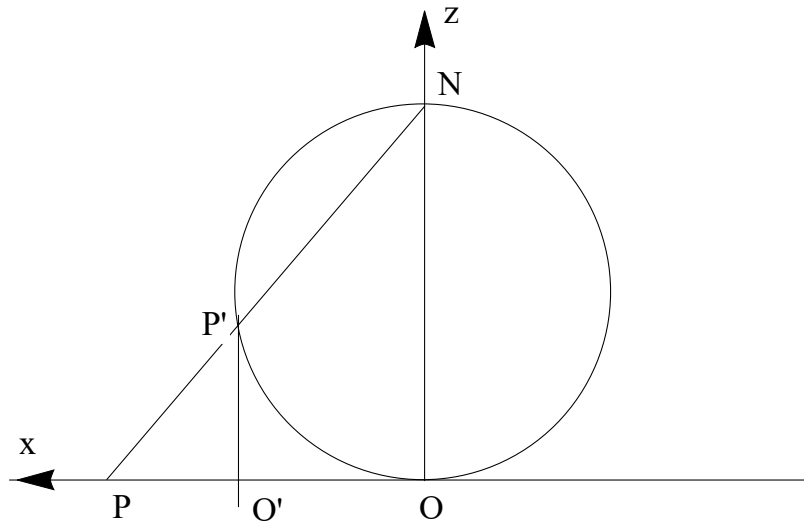
それが直線を始めている極だ。しかし、この北極(南極)を除外すれば接平面に写すことができる。

この立体射影の方法は1点の例外をつくるが今回の4次元を見る方法に使いそうなので少し詳しくみていみよう。

図のように半径1の円で  $z$  軸上の北極を  $N$ 、 $x$  軸上の交点  $P$ 、円周上の交点  $P'$  その垂線の足を

O'とする。

原点はz軸と円の接点Oにとる。



【Q1】ここで問題である。Pのx座標をx,P'の座標を(x',z')としてxを円周上の座標x',z'で表してみよ。

ON:O'P'=2:z'だから三角形の相似から

$$\begin{aligned}
 2 : z' &= x : (x - x') \\
 x z' &= 2x - 2x' \\
 x(2 - z') &= 2x' \\
 x &= \frac{2x'}{2 - z'}
 \end{aligned} \tag{1}$$

となる。

さらにP'(x',z')が円周上の点であるから

$$(x')^2 + (z')^2 = 1 \tag{2}$$

が成り立つことに注意する。紙面裏から表に図のO点にy軸を作れば3次元の空間に拡張できる。

P'を(x',y',z')とすると球面上の極を除いたあらゆる点はx-y平面上のP(x,y)写される。これが立体射影である。

P'は球面上にあるので

$$x'^2 + y'^2 + z'^2 = 1 \tag{3}$$

が成り立つ。従って3次元の空間の中で球面上の点はこの式3に従わなくては行けないから結局独立した変数は2つしかない。

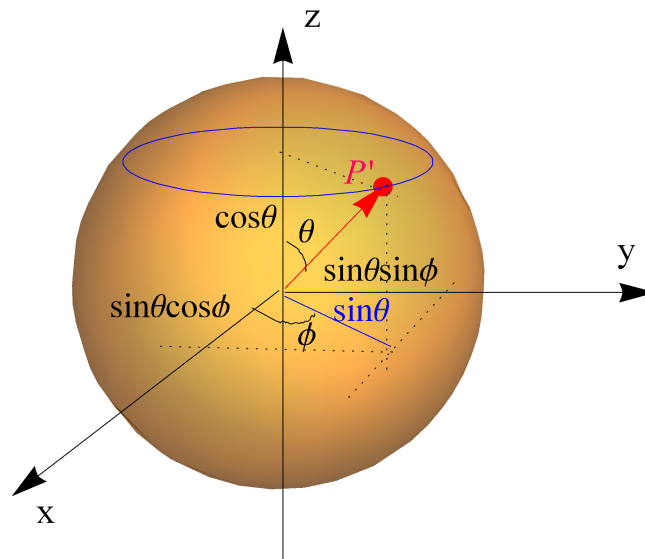
その2つの変数が新たに「面」を表すわけだ。式3という関係があるのでこれを満たす座

標表示を $\theta, \phi$ の2つで表すことを考えればよい。球面上の点 $x', y', z'$ から立体射影された平面が

$$\frac{1}{1-z'} \{x', y'\} \quad (4)$$

のように書ける。これは4次元の場合に応用できそうである。

【Q2】 半径1の球として、次の図のように $\theta, \phi$ を決めると球上の点 $P'$ はどう表されるか考えてみよ。



結果は次のようになる。これを $P'$ としよう。

$$x' = \text{Sin}[\theta] \text{Cos}[\phi], y' = \text{Sin}[\theta] \text{Sin}[\phi], z = \text{Cos}[\theta] \quad (5)$$

$$P' = \{\text{Sin}[\theta] \text{Cos}[\phi], \text{Sin}[\theta] \text{Sin}[\phi], \text{Cos}[\theta]\} \quad (6)$$

自分で各成分を2乗して足してみよ。三角関数の性質をつかうと確かに1になる。

【Mathematica入力】

Mathematicaでも簡単に確かめることができる。

```
Clear[θ, φ];
FullSimplify[(Sin[θ] Cos[φ])2 + (Sin[θ] Sin[φ])2 + (Cos[θ])2]
```

1

次にこの点をMathematicaで表して $\theta, \phi$ を自由に变化できるようにしよう。

はじめに最初の単純な射影は球面上の点が動いたとき、3つの面上にどういう動きになるかイメージを作っておこう。

【Mathematica入力】

最初のデモプログラムはやや複雑なので次にシンプルなもの順番に理解していこう。

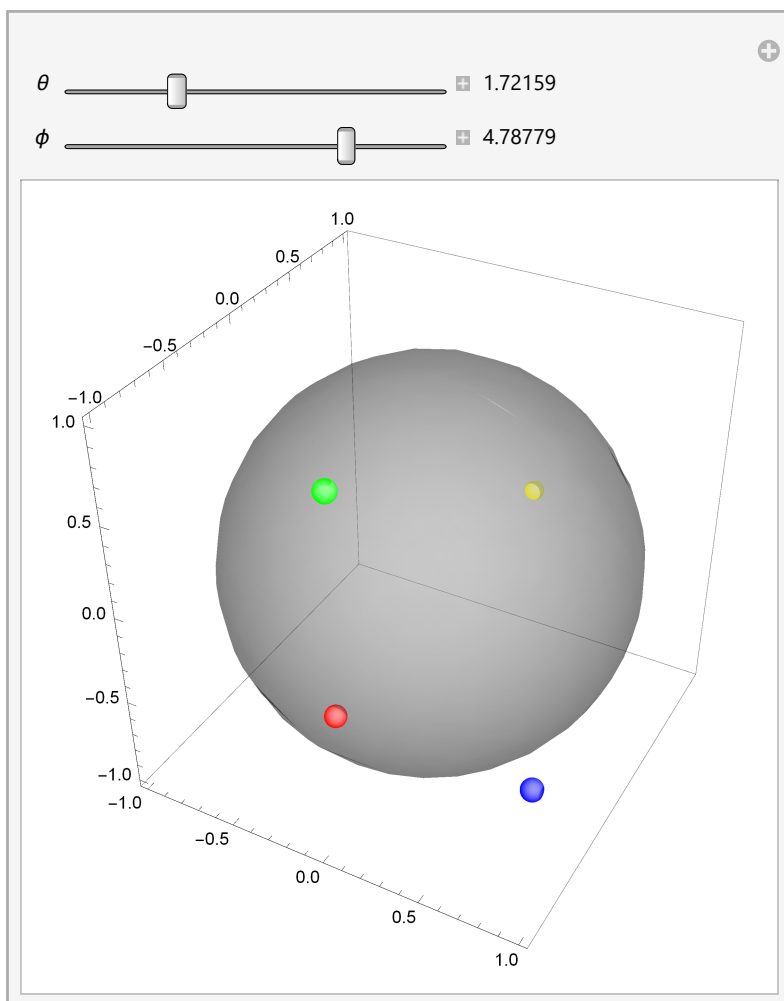
次の例では半径 1 の球を薄く表示させてある。P'はptに置き換える。

赤い小球が球面上を自由に動くと、 $xy, yz, zx$ 平面に射影された青、黄、緑の点が動く。いろいろな角度を変えて観測してみよう。

```

Manipulate[
  pt = {Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ]};
  gp1 = ParametricPlot3D[{Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ]},
    {θ, 0, π}, {φ, 0, 2 π}, PlotStyle → Opacity[0.3], Mesh → None];
  gp2 = Graphics3D[{Red, Sphere[pt, 0.05]},
    PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}, {-1.2, 1.2}}];
  gp3 =
    Graphics3D[{Green, Sphere[{Sin[θ] Cos[φ], Sin[θ] Sin[φ], 1}, 0.05]},
    PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}, {-1.2, 1.2}}];
  gpy = Graphics3D[{Blue, Sphere[{1, Sin[θ] Sin[φ], Cos[θ]}, 0.05]},
    PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}, {-1.2, 1.2}}];
  gpz = Graphics3D[{Yellow, Sphere[{Sin[θ] Cos[φ], 1, Cos[θ]}, 0.05]},
    PlotRange → {{-1.2, 1.2}, {-1.2, 1.2}, {-1.2, 1.2}}];
  Show[gp1, gp2, gp3, gpy, gpz],
  {{θ, π/3, "θ"}, 0, 2 π, Appearance → "Labeled"},
  {{φ, π/3, "φ"}, 0, 2 π, Appearance → "Labeled"}]

```



このように普通の射影では球面上を動くP'の影をxy,yz,zx面からみた様子である。

では次に立体射影を同じように見てみよう。

### 【Mathematica入力】

原点は中心ではないので z 成分のみ + 1 されることに注意する。

0で割るとエラーになるのでその補正で微小値入れてある。

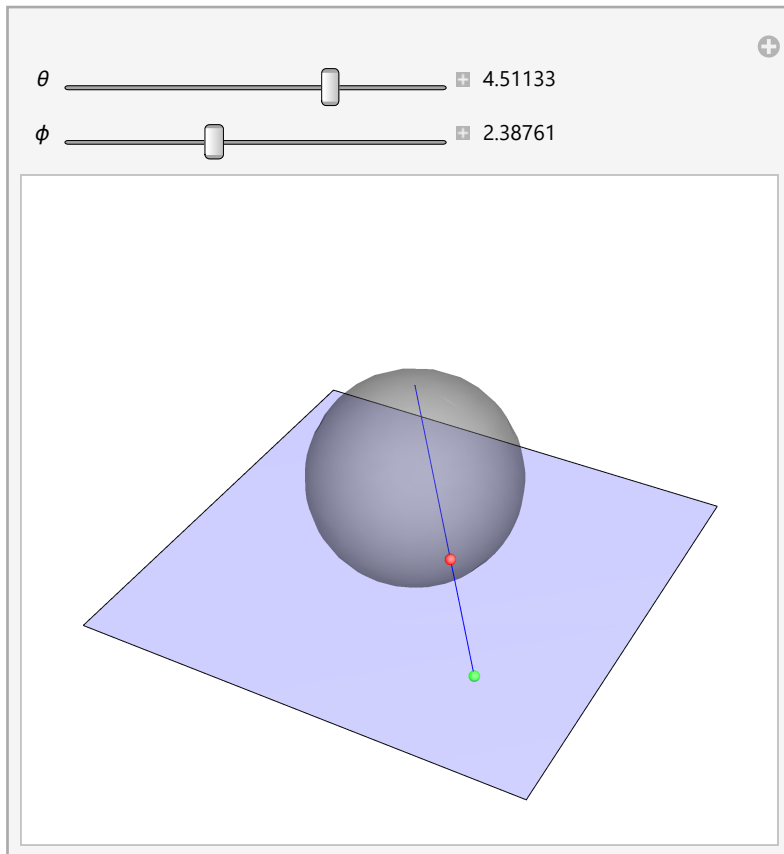
原点中心のxy平面を紫の透明色を使うためにHue[ ]を利用している。

```

Clear[ps, gss, gs1, gs2]
Manipulate[
  po = {0, 0, 2};
  ps = {Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ] + 1};
  k = 2 / (1.00001 - Cos[θ]);
  pl = {k Sin[θ] Cos[φ], k Sin[θ] Sin[φ], 0};
  rga = 1.2;
  rgb = 2.2;
  gs1 = ParametricPlot3D[{Sin[θ] Cos[φ], Sin[θ] Sin[φ], Cos[θ] + 1},
    {θ, 0, π}, {φ, 0, 2 π}, PlotStyle → Opacity[0.3], Boxed → False,
    Mesh → None, Axes → None,
    PlotRange → {{-rgb, rgb}, {-rgb, rgb}, {-0.2, rgb}}];
  gs2 = Graphics3D[{Red, Sphere[ps, 0.05]},
    PlotRange → {{-rgb, rgb}, {-rgb, rgb}, {-0.2, rgb}}];
  gss = Graphics3D[{Green, Sphere[pl, 0.05]},
    PlotRange → {{-rgb, rgb}, {-rgb, rgb}, {-0.2, rgb}}];
  gs1 = Graphics3D[{Blue, Line[{po, pl}]},
    PlotRange → {{-rgb, rgb}, {-rgb, rgb}, {-0.2, rgb}}];
  gsp =
  Graphics3D[{Hue[2 / 3, 1, 1, .3],
    Polygon[{{-rgb, rgb, 0}, {rgb, rgb, 0}, {rgb, -rgb, 0},
      {-rgb, -rgb, 0}}]},
    PlotRange → {{-rgb, rgb}, {-rgb, rgb}, {-0.2, rgb}}];
  Show[gs1, gs2, gss, gs1, gsp],
  {{θ, 2 π / 3, "θ"}, 0, 2 π, Appearance → "Labeled"},
  {{φ, 4 π / 3, "φ"}, 0, 2 π, Appearance → "Labeled"}]

```





☞球面上の赤点の動きが立体射影によってxy平面の緑の動きになることを確かめてほしい。

球面上の一点を除く全ての点を平面に写すことができる。

球面上での点の動きを線として残せば最初に示したプログラムのようなになるわけだ。

☞立体射影がイメージできたらいよいよ次に4次元の世界に行く。

とその前にせっかく球上の点を角度をつかって表すことをしたので寄り道をしよう。

## 4次元球の体積

☞この節は積分を用いる。寄り道なので積分を習ってない場合、次の節に飛んでもよい。

体積という言葉を広げ、円板の面積  $\pi r^2$  は2次元の体積  $V_2$  とし、 $V_3 = \frac{4}{3} \pi r^3$  とする。

そこで、ここでは4次元の体積  $V_4$  を求めてみる。

☞まず  $V_2$  から始めよう。学校で学んだやり方は

$$x = r \cos\theta, y = r \sin\theta \quad (7)$$

とすると小さな円輪の1部分の面積が  $r d\theta \times dr$  となるので次を計算する。

$$\int_0^R \int_0^{2\pi} r \, dr \, d\theta \quad (8)$$

【Mathematica入力】

積分記号はESC int ESC、ESC dd ESCで積分変数を次にかく。

文字式で積分する時は次のようなコマンドを直接使った方がよい。

```
Clear[R, r, θ];
Integrate[r, {r, 0, R}, {θ, 0, 2π}]
π R2
```

☞ここでは一般化するために断面半径Dを次のように定義する。2次元の円は

$$x^2 + y^2 = r^2 \quad (9)$$

だからこの時の断面半径はDはx座標になる。

$$x = D_2 = \pm \sqrt{r^2 - y^2} \quad (10)$$

xをこの $\sqrt{r^2 - y^2}$ で置き換えると次のように1つ目の積分が簡単にDに置き換わる。

この場合は{}の中が断面を表し、ちょうどxの2倍の長さになることに注意して次を得る。

$$\begin{aligned} V_2 &= \int_{-R}^R \int_{D_2} dx dy \\ &= \int_{-R}^R \left\{ \int_{-\sqrt{r^2 - y^2}}^{\sqrt{r^2 - y^2}} dx \right\} dy \\ &= \int_{-R}^R 2 \sqrt{r^2 - y^2} dy \end{aligned} \quad (11)$$

### 【Mathematica入力】

最後の積分を手計算したらMathematicaで確かめると次の結果が得られる。

```
Clear[R, y];
Integrate[2√R2 - y2, {y, -R, R}];
Simplify[%, R > 0]
π R2
```

積分とは簡単にはある区間を細かく刻んで足し合わせることである。断面を刻む数を増やして足し合わせると円板ができる様子をMathematicaで描いてみよう。

### 【Mathematica入力】

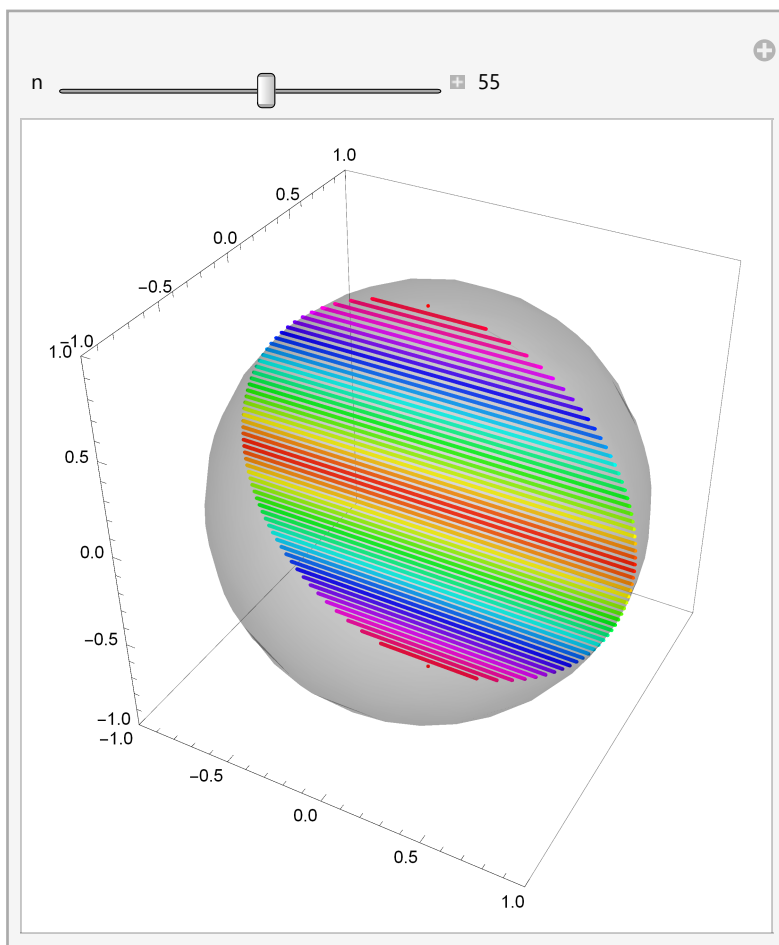
nを増やすと断面の直線の足し合わせで円板ができる。参考のために半径1の球を薄く表示してある。

少しづつ変化させたデータを作る時にはTable[]コマンドを利用する。

```

Clear[z, gs, ψ1, φ1, θ1];
Manipulate[
  gs1 = ParametricPlot3D[{Cos[φ1] Sin[θ1], Sin[φ1] Sin[θ1], Cos[θ1]},
    {φ1, 0, 2 π}, {θ1, 0, π}, PlotStyle → Opacity[0.2], Mesh → None,
    PlotRange → {{-1, 1}, {-1, 1}, {-1, 1}}];
  gs2 =
  Table[
    Graphics3D[{{Hue[Abs[z], 1, 1], Thick, Line[{{√(1 - z^2), 0, z}, {-√(1 - z^2), 0, z}]}}],
    {z, -1, 1, 2/n}];
  Show[gs1, gs2]
  , {{n, 20, "n"}, 1, 100, 1, Appearance → "Labeled"}]

```



同じことを3次元でやろう。

$$x^2 + y^2 = r^2 - z^2 \quad (12)$$

よって断面半径を

$$D_3 = \pm \sqrt{r^2 - z^2} \quad (13)$$

とおけば今度は{}の中の断面が円板の面積になることに注意する。この  $D_3$  を  $V_2 = \pi R^2$  の  $R$  の中に入れると

$$\begin{aligned}
 V_3 &= \int_{-R}^R \left\{ \int_{D_3} dx dy \right\} dz \\
 &= \int_{-R}^R \left\{ \pi(r^2 - z^2) \right\} dz
 \end{aligned}
 \tag{14}$$

となる。このように断面半径と断面をうまくつかうと{}の外を[-R,R]までの積分に置き換えることができる。ただし、断面は線になったり面積になったり、体積になったりするわけだ。

### 【Mathematica入力】

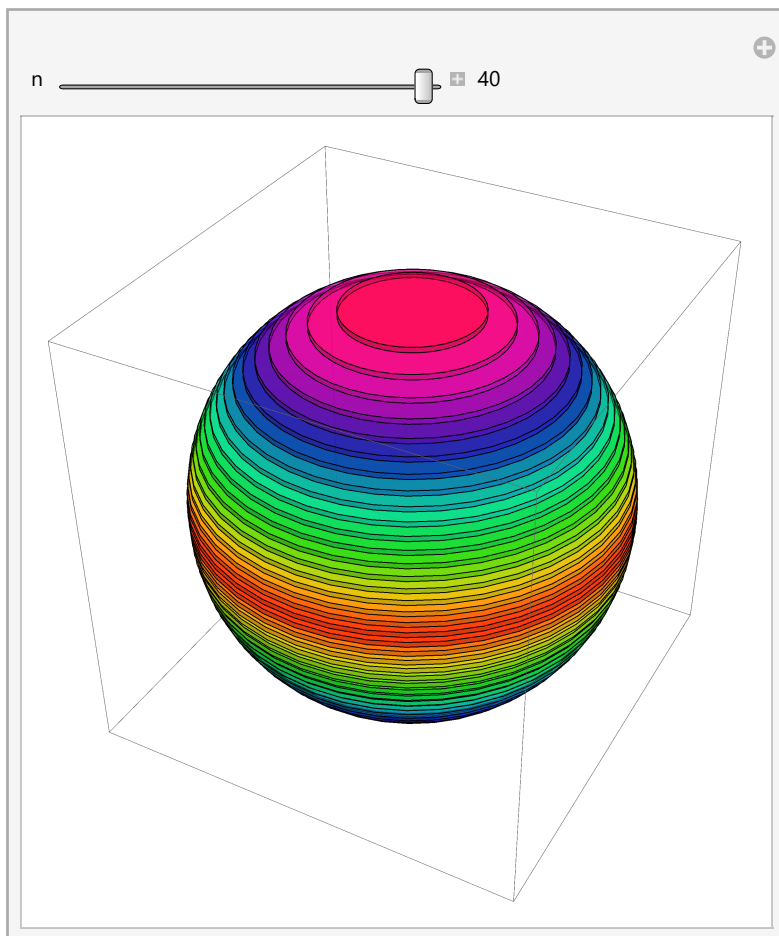
先と同じようにこの様子を断面である円板の足し合わせで表してみよう。

nを大きくすると球ができることがわかるだろう。

```

Clear[gt, z, ψ1, φ1, θ1];
Manipulate[
  gt =
  Table[
    Graphics3D[{{Hue[Abs[z], 1, 1, 0.3], Cylinder[{{0, 0, z}, {0, 0, z + 1/n}], Sqrt[1 - z^2]},
      Mesh -> None}}, {z, -1, 1, 2/n}];
  Show[gt]
  , {{n, 10, "n"}, 4, 40, 1, Appearance -> "Labeled"}]

```



計算と共に積分の操作がイメージできただろうか。

【Mathematica入力】

最後の積分を手計算したらMathematicaで確かめると次の結果が得られる。

```
Clear[R, z];
Integrate[π (R2 - z2), {z, -R, R}];
Simplify[%, R > 0]

$$\frac{4 \pi R^3}{3}$$

```

では同じように  $V_4$  を求めよう。

$$\begin{aligned} x^2 + y^2 + z^2 + w^2 &= r^2 \\ x^2 + y^2 + z^2 &= r^2 - w^2 \\ D_4 &= \pm \sqrt{r^2 - w^2} \end{aligned} \quad (15)$$

とおけば今度は  $\{ \}$  の中の断面が体積になることに注意する。この  $D_4$  を  $V_3 = \frac{4\pi R^3}{3}$  の  $R$  の中に入れると

$$\begin{aligned} V_3 &= \int_{-R}^R \left\{ \int_{D_4} dx dy dz \right\} dw \\ &= \int_{-R}^R \left\{ \frac{4\pi}{3} \left( \sqrt{r^2 - w^2} \right)^3 \right\} dw \end{aligned} \quad (16)$$

【Mathematica入力】

最後の積分を手計算したらMathematicaで確かめると次の結果が得られる。

```
Clear[R, w];
Integrate[ $\frac{4\pi}{3} \left( \sqrt{R^2 - w^2} \right)^3$ , {w, -R, R}];
Simplify[%, R > 0]

$$\frac{\pi^2 R^4}{2}$$

```

4次元球の体積が出た！しかもこの調子で、この結果を次々に代入していけば  $n$ 次元球の体積も公式化できそうだ。是非チャレンジしてみたい。

では次の節で本題に戻ろう。

## 複素数の組

これまでの学習で3次元の球が式3を満たしたように4次元の空間においても4次元球が存在し、次を満たす。

$$x'^2 + y'^2 + z'^2 + w'^2 = 1 \quad (17)$$

この球の表面は  $S^3$  で表され、3つの変数をもつことが予想される。

この条件をうまく表すものをみつけるのに複素数を知っておく必要がある。

そこでこの節では簡単に複素数の基礎を学ぶ。複素数は非常に面白く、奥が深い。物理とも密接に関係しているが、今回はその基礎だけに留めておこう。

複素数とは  $z$  を複素数、 $x, y$  を実数とするととすると

$$z = x + iy \quad (18)$$

で表され、 $i$  を虚数単位といい

$$i^2 = -1 \quad (19)$$

のように2乗してマイナスになるものまで数の世界を広げるのである。実数は2乗してマイナスになることはない。

また、複素数には「共役」と呼ばれる相棒がいて、この相棒をかけると必ず実数になる。共役とは  $i$  の前の符号を  $-$  にしたものだ。

数式では次のようにバー  $\bar{z}$  で表す。

$$\bar{z} = x - iy \quad (20)$$

$$z\bar{z} = x^2 + y^2 \quad (21)$$

これはMathematicaでも簡単に確かめることができる。

#### 【Mathematica入力】

虚数単位の入力は ESC ii ESCで行い通常の  $i$  とは区別する。共役は `Conjugate[]` というコマンドを用いる。

ただし3つ目の例で示すように簡素化する時に、 $x, y$  が実数であることを明示しておく。

簡単には4番目に例のようにしてもよい。

```
z1 = x + i y; z2 = x - i y;
Simplify[z1 z2]
Simplify[z1 Conjugate[z1], {x ∈ Reals, y ∈ Reals}]
Simplify[z1 Conjugate[z1], {x > 0, y > 0}]
x2 + y2
x2 + y2
x2 + y2
```

さて、気が付いた人もいるだろう。複素数の大きさをとると式17の内2つが出てくるので2組の複素数をうまくつかうとよさそうである。

そして2組の複素数には独立した4つの実数が入っている。これは4次元  $R^4$  を表せそうである。

複数をイメージしやすいように再びMathematicaのデモプログラムをいくつか見てみよう。

#### 【Mathematica入力】

☞ Wolframのデモのサイトに Štefan Porubský氏の「Complex Number」

というプログラムを以下に引用する。ここでもプログラムの中身の理解は後にして、立体射影が何なのか実際に実行することで確認してほしい。

著者により若干のプログラムの修正がある。

これは単純に縦軸を虚数、横軸を実数にして複素数  $z=x+iy$  を表している。これを今後複素平面と呼ぶ。

複素平面上で共役がどういう関係になるか、考えてみよう。

共役関係はちょうど $180^\circ$ の関係であることがわかったらどうか。

また、この表示は複素数が円や回転と関係していそうなこともわかる。

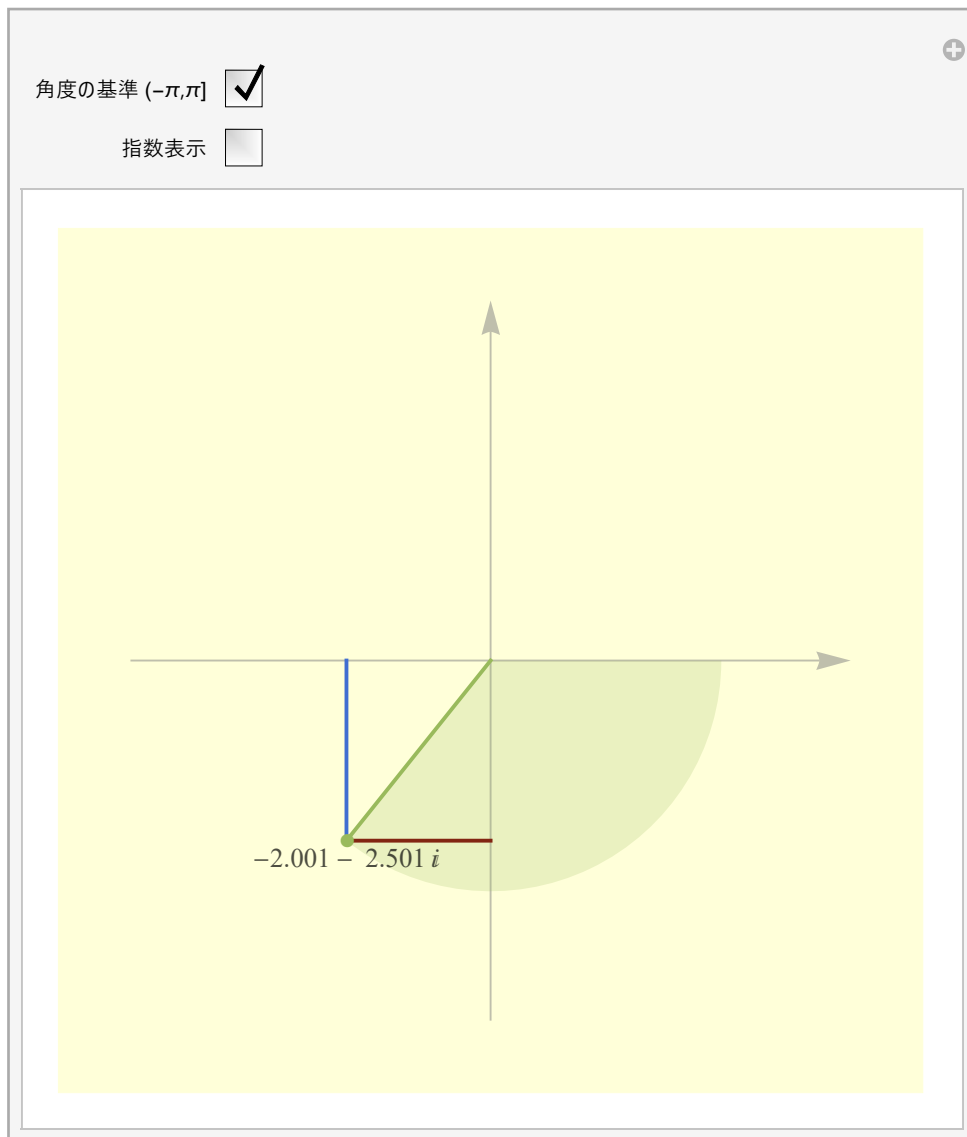
```
Manipulate[
  r = Norm[xy];
  If[r == 0, r = .1];
  If[showAngle,  $\theta = \text{Arg}[xy[[1]] + xy[[2]] i]$ ,
     $\theta = \text{If}[\text{Arg}[xy[[1]] + xy[[2]] i] \geq 0, \text{Arg}[xy[[1]] + xy[[2]] i],$ 
       $\text{Arg}[xy[[1]] + xy[[2]] i] + 2\pi]$ ];
  Graphics[{{White, Rectangle[{-6, 6}, {6, 8}]},
    {Lighter[Yellow, 0.85], Rectangle[{-6, -6}, {6, 6}]},
    {Lighter[Black, 0.5], Opacity[0.5], AbsoluteThickness[1],
      Arrow[{{0, -5}, {0, 5}}], Arrow[{{-5, 0}, {5, 0}}]},
    {RGBColor[.6, .73, .36], Opacity[1], AbsoluteThickness[2],
      Line[{{0, 0}, xy]}], {RGBColor[.25, .43, .82], Opacity[1],
      AbsoluteThickness[2], Line[{{xy[[1]], 0}, xy]}],
    {RGBColor[.49, 0, 0], Opacity[1], AbsoluteThickness[2],
      Line[{{0, xy[[2]]}, xy]}],
    {Black, Opacity[1], AbsolutePointSize[5], Point[xy]},
    {RGBColor[.6, .73, .36], Opacity[0.2],
      Disk[{{0, 0}, r, If[showAngle &&  $\theta < 0$ , { $\theta$ , 0}, {0,  $\theta$ }]},
      {Opacity[0.7]},
    If[showGon,
      Text[
        Style[TraditionalForm[
          ToString[PaddedForm[N@Abs[xy[[1]] + I xy[[2]]], {4, 3}]] *
          Power[e, ToString[PaddedForm[N@ $\theta$ , {4, 3}]] <> ToString["i"]]],
          RGBColor[.25, .43, .82], 14], xy, If[ $0 \leq \theta \leq \pi$ , {0, -1}, {0, 1}]],
        Text[
          Style[TraditionalForm[PaddedForm[Chop[N@xy[[1]] + i xy[[2]]],
            {4, 3}]], Black, 14], xy, If[ $0 \leq \theta \leq \pi$ , {0, -1}, {0, 1}]]],
      {Opacity[0.7]},
      Text[
        Style[Row[{"Re(", Style["z", Italic], ") = ",
          ToString[PaddedForm[N@Chop[Re[xy[[1]] + i xy[[2]]]], {4, 3}]]}],
          RGBColor[.49, 0, 0], 12], {-5.5, 7.4}, {-1, 0}]],
```

```

{Opacity[0.7],
  Text[
    Style[Row[{"Im(", Style["z", Italic], ") = ",
      ToString[PaddedForm[N@Chop[Im[xy[[1]] + i xy[[2]]]], {4, 3}]]],
      RGBColor[.25, .43, .82], 12], {-5.5, 6.6}, {-1, 0}]],
{
  Text[
    Style[Row[{"abs(", Style["z", Italic], ") = ",
      ToString[PaddedForm[N@Chop[Abs[xy[[1]] + i xy[[2]]]],
        {4, 3}]]], RGBColor[.6, .73, .36], 12], {2, 7.4}, {-1, 0}]],
{
  Text[
    Style[Row[{"arg(", Style["z", Italic], ") = ",
      ToString[PaddedForm[N@Chop[θ], {4, 3}]]],
      RGBColor[.6, .73, .36], 12], {2, 6.6}, {-1, 0}]]
}, PlotRange → {{-6, 6}, {-6, 6}}, AspectRatio → 1, ImageSize → 450],
{{showAngle, True, "角度の基準(-π,π)"}, {True, False}},
{{showGon, False, "指数表示"}, {True, False}},
{{xy, {-2.001, -2.501}}, {-4, -4}, {4, 4}, ControlType → Locator,
  Appearance → Style["●", 12, RGBColor[.6, .73, .36]]},
TrackedSymbols → Manipulate]

```





このように複素数は実は原点の周りを回転するベクトルのように表すことができる。

2乗して-1になるとはちょうど実軸の1から出発して2回の90°の回転で実軸上の-1に写ること、これは180°=πラジアン回転である。

これから複素数の別の表現として半径 $r$ 、角度 $\theta$ [rad]の回転で表すことができ

$$z = r e^{i\theta} \quad (22)$$

で表すことができる。例えば大きさが1で3回で-1になれば $\theta = \pi/3$ である。これから $n$ 乗根が簡単にわかることになる、。

#### 【Mathematica入力】

さっそくMathematicaで確認しておこう。-1の3乗根と、複素平面での60°の回転をNコマンドで計算させる。

$$\mathbf{N}\left[\sqrt[3]{-1}\right]$$

$$\mathbf{N}\left[\mathbf{Exp}\left[\mathbf{i}\frac{\pi}{3}\right]\right]$$

$$0.5 + 0.866025 \mathbf{i}$$

$$0.5 + 0.866025 \mathbf{i}$$

同じ複素数を表している。さらにn乗根を大きくしていくと、これはn角形と関係してることが複素平面の図形からわかるだろう。

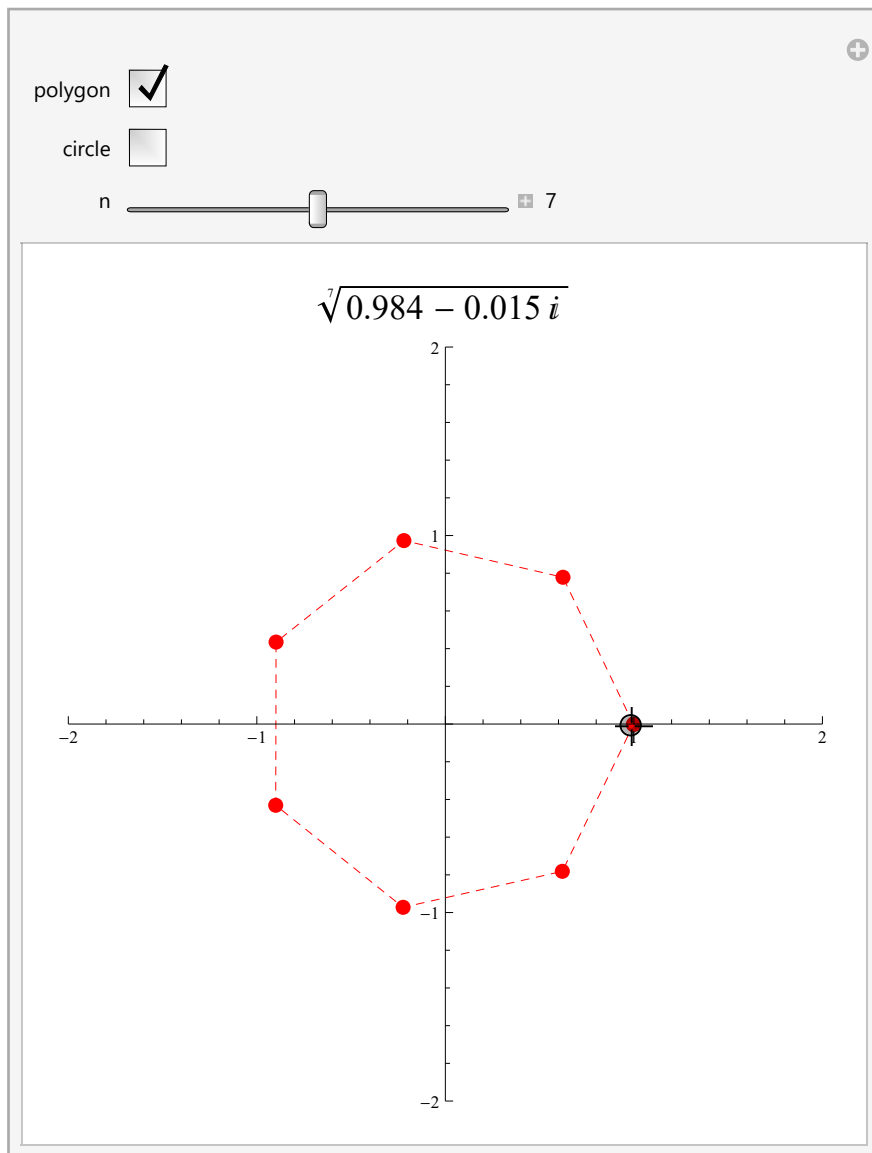
これもMathematicaのデモに簡単な例があるので確かめてみよう。

### 【Mathematica入力】

Wolframのデモのサイトに Germán Alvarado Jiménez 氏の「Roots of a Complex Number」というプログラムを以下に引用する。ここでもプログラムの中身の理解は後にして、立体射影が何なのか実際に実行することで確認してほしい。

このプログラムではLocatorを使っているのでマウスでダイレクトに座標を変更できる。

```
Manipulate[z = aa[[1]] + I aa[[2]];
raices =
Table[{{ $\sqrt[n]{\text{Abs}[z]}$  Cos[(Arg[z] + 2 π k) / n],
 $\sqrt[n]{\text{Abs}[z]}$  Sin[(Arg[z] + 2 π k) / n]}, {k, 0, n}];
Column[{
Text@Style[RadicalBox[z, n] // DisplayForm, 18],
Graphics[{
If[circle, Circle[{0, 0},  $\sqrt[n]{\text{Abs}[z]}$ ], {}],
If[polygon, {Dashed, Red, Line[Append[raices, First@raices]]},
{}],
Red, PointSize[.02], Point[raices]},
PlotRange → 2, AspectRatio → Automatic, Axes → True,
ImageSize → {400, 400}]
}, Alignment → Center],
{polygon, {False, True}},
{circle, {False, True}},
{{n, 2, "n"}, 2, 12, 1, Appearance → "Labeled"},
{{aa, {1, 1}}, {-2, -2}, {2, 2}, Locator},
TrackedSymbols :> {polygon, circle, n, aa}]
```



複素数は  $z=x+iy$  のように1つの数に2つの実数の組を持っているので2成分をもつベクトル  $z(x,y)$  と同じように扱うことができる。

Mathematicaは複素数の実成分と虚成分を取り出すコマンドが用意されていて  $\text{Re}[]$ ,  $\text{Im}[]$  がそうだ

#### 【Mathematica入力】

実数部分は  $\text{Re}[]$ 、虚数部分は  $\text{Im}[]$  で取り出す。いろいろ試してみよう。

```
Clear[z, x, y];
z = 3 + i 4;
Re[z]
Im[z]

3
4
```

#### 【Mathematica入力】

$\theta$ を実数指定すれば正しく、表示する。

```
Clear[z,  $\theta$ ];
z = Cos[ $\theta$ ] + i Sin[ $\theta$ ];
Re[z]
Simplify[%,  $\theta > 0$ ]
Im[z]
Simplify[%,  $\theta > 0$ ]
-Im[Sin[ $\theta$ ]] + Re[Cos[ $\theta$ ]]
Cos[ $\theta$ ]
Im[Cos[ $\theta$ ]] + Re[Sin[ $\theta$ ]]
Sin[ $\theta$ ]
```

### 【Mathematica入力】

さて、以下の結果をよーく考えてほしい。

FullSimplifyを用いると多少時間がかかるが、簡単な関数になるものは簡単にしてくれる。

```
Clear[z,  $\theta$ ];
z = Exp[i  $\theta$ ];
Re[z];
FullSimplify[%,  $\theta > 0$ ]
Im[z];
FullSimplify[%,  $\theta > 0$ ]
Cos[ $\theta$ ]
Sin[ $\theta$ ]
```

この結果は

$$e^{i\theta} = \cos\theta + i \sin\theta \quad (23)$$

であることを表している！これは数学上最も美しい式の1つといわれるEulerの公式である。

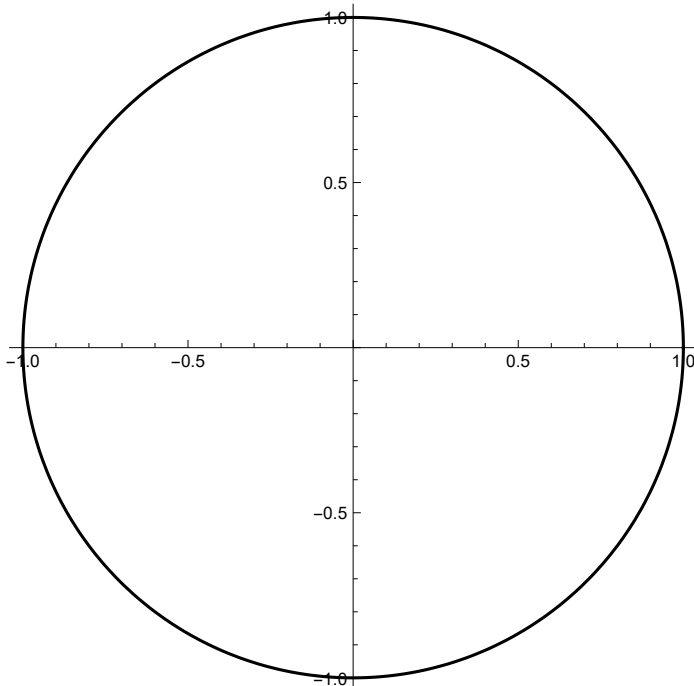
つまり、複素数の世界で指数関数は三角関数になる。右辺の実成分、虚成分をベクトルの成分とみなせば、

このベクトルはぐるぐる回転する長さ1のベクトルである。

### 【Mathematica入力】

ParametricPlotで確かめてみよう。

```
Clear[z, θ];
z = Exp[i θ];
ParametricPlot[{Re[z], Im[z]}, {θ, 0, 2 π}]
```



確かに円になった。改めてこの節の最初のデモに戻って、複素数を指数表示にしているいろいろ試してみるといいだろう。

複素数と立体射影、この2つの道具が準備できたところで、いよいよ次節で本題に戻る。

## Hopf-fibration

問題を確認すると4次元のなかの3次元球面を表すために式17

$$x'^2 + y'^2 + z'^2 + w'^2 = 1$$

満たすような、3つの変数を探すことであった。

さらに立体射影の式4からこれを拡張すれば

$$\frac{1}{1-w'} \{x', y', z'\} \quad (24)$$

を考えればよいだろう。そこでこれを満たすものの1つとして、これまで考えてきた、複素数を用いて次のような2つの組を考えよう。

まず、式17を満たすものの1つとして、これまで考えてきた、複素数を用いて次のような2つの組を考えよう。

【Mathematica入力】

$\psi$ はESC psi ESC で入力する。複素数 $z$ と $w$ は3つの変数  $\theta, \phi, \psi$  を持っている。

```

Clear[z, w, φ, θ];
z = Cos[θ] Exp[-i (ψ + φ)];
w = Sin[θ] Exp[-i (ψ - φ)];

```

この2つの複素数の実成分と虚成分を考えれば4つの成分をもつことになる。これが式17を満たすかどうか確かめてみよう。

### 【Mathematica入力】

$\theta, \phi, \psi$  を実数として簡単にする。

```

zR = FullSimplify[Re[z]^2 + Re[w]^2, {φ > 0, ψ > 0, θ > 0}]
zI = FullSimplify[Im[z]^2 + Im[w]^2, {φ > 0, ψ > 0, θ > 0}]
Simplify[zI + zR]

```

$$\text{Cos}[\theta]^2 \text{Cos}[\phi + \psi]^2 + \text{Cos}[\phi - \psi]^2 \text{Sin}[\theta]^2$$

$$\text{Sin}[\theta]^2 \text{Sin}[\phi - \psi]^2 + \text{Cos}[\theta]^2 \text{Sin}[\phi + \psi]^2$$

1

実成分、虚成分の大きさの2乗は $\theta, \phi, \psi$  の関数だがこれを足すと1になる。

この2組の複素数の4つの成分が4次元の空間をつくる。

はじめに地図を射影してみたように3次元に射影してみよう。

ただし、最初は単純に1つの成分をカットし、見えてない部分があることを承知して3次元の空間に表す。

次のプログラムではkという変数を追加して片方の角度の変化を遅らせるようにしている。

はじめこのkは1として $\psi$ の方を変化させ見てみるとよい。

### 【Mathematica入力】

4つの成分から3つを選んでみる。他にも選び方があるので

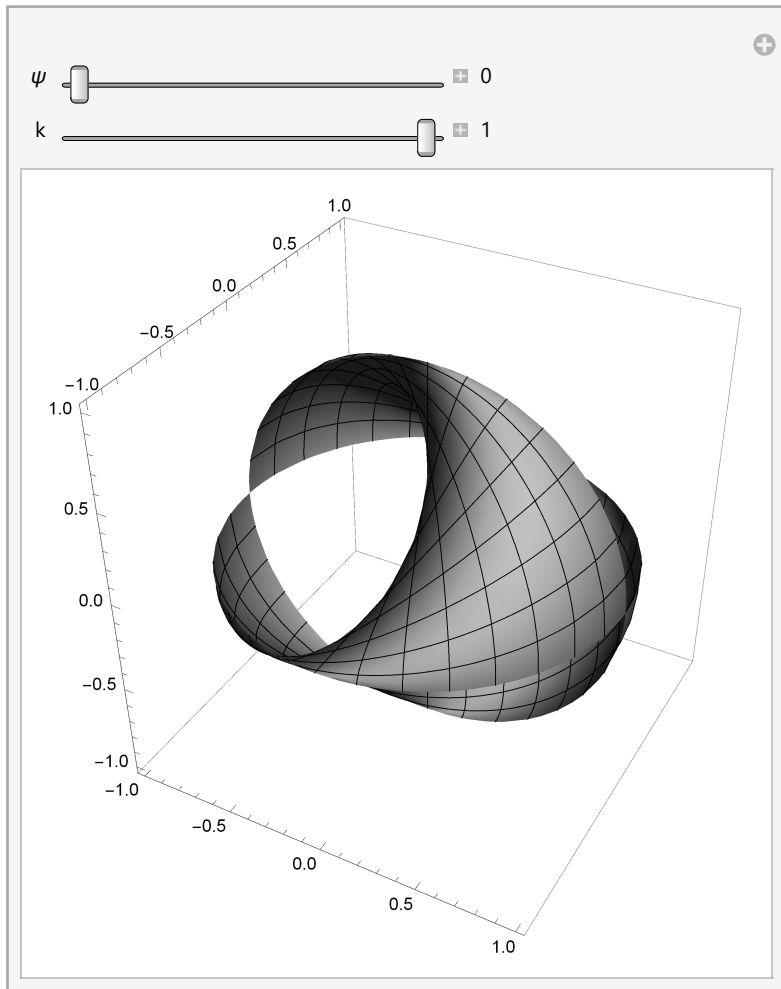
自分で変更して試し見るとよい。角度も $2\pi$ にしてみたりして見てみよ。

観測する向きが変わるが他におおきな相違はない。

```

Clear[θa, za, wa, ψa, φa, gal, k];
Manipulate[
  za = Cos[θa] Exp[-i (ψa + φa)];
  wa = Sin[θa] Exp[-k i (ψa - φa)];
  gal = ParametricPlot3D[{Re[za], Im[za], Re[wa]}, {θa, 0, π}, {φa, 0, π}];
  Show[gal]
  , {{ψa, 0, "ψ"}, 0, 2 π, Appearance → "Labeled"},
  {{k, 1, "k"}, 0, 1, Appearance → "Labeled"}]

```



最初に実行すると $\theta$ や $\varphi$ の範囲が $\pi$ までにとっているため切り口があり、中が見える。

次にこれを $2\pi$ に変えると閉じた図形になるが、きれいな球には見えない。

凸凹が2ヶ所あるのがわかるだろう。4次元球で見てたでこぼこは3次元球になると見えない。

これは3次元で1周することが4次元では半周していることに相当するからである。

次にこのプログラムの $k$ を変化させてみよう。

1から0まで連続的に変化させると4次元球をつくっていた1つの変数の影響が少なくなり、やがて

3次元に移行する。

ところがこれでは4次元の球の全体像を見ているわけではない。

全体像を1点だけ除き、3次元の空間に表すためには前節の立体射影をつかう。

式24を用いて4次元を3次元に立体射影してみる。

2乗した和が1になる4つの変数から3次元の座標を作り出すわけだ。これをHopf写像という。

#### 【Mathematica入力】

ここでは立体射影の定義を最後のInitializationの中にしている。

Mnipulateを使い動的な絵を得る時の関数の定義はこうしておく他と競合しない。

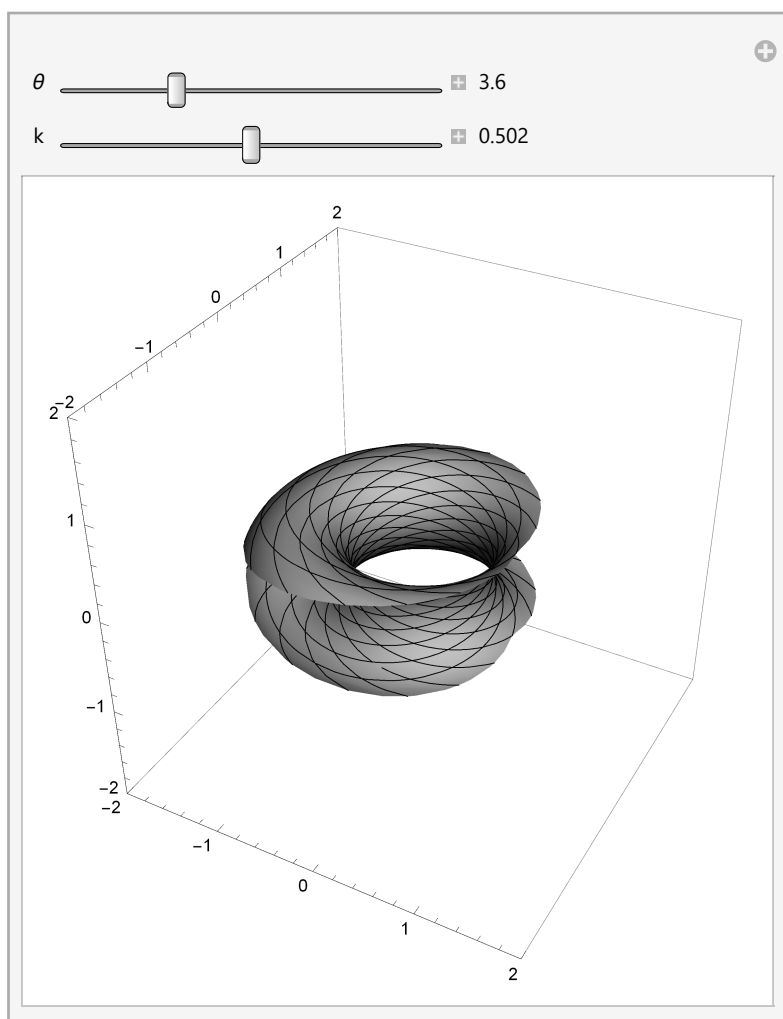
このプログラムにも先と同じ役割をするkを入れてみた。はじめは1に固定してためすとよい。



```

Clear[θ, k, zb, wb, ψ, φ, x, y, z];
Manipulate[
  zb = Cos[θ] Exp[-i (ψ + φ)];
  wb = Sin[θ] Exp[-i k (ψ - φ)];
  ParametricPlot3D[Evaluate[Pj[zb, wb]], {ψ, 0, 2 π}, {φ, 0, 2 π},
    PlotRange → {{-2, 2}, {-2, 2}, {-2, 2}}
  , {{θ, 3.6, "θ"}, 0, 4 π, Appearance → "Labeled"},
  {{k, 1, "k"}, 0, 1, Appearance → "Labeled"},
  Initialization → {Pj[x_, y_] :=  $\frac{1}{1 - \text{Re}[y]}$  {Re[x], Im[x], Im[y]}}]

```



☞ ドーナツができた！この穴が何に由来しているかわかるだろうか。立体射影が表すことができない1点が

あったことを思い出してほしい。

$\psi, \phi$ の範囲を  $\pi$  までにして  $\theta$  を 3.6 程度にすると最初の図と同じような断面図が得られる。

$\psi, \phi$ の範囲を  $2\pi$  までとると  $\theta$  の変化で大きさが変わるが、中心軸が一定なドーナツ (トーラスという) が描ける。

次に  $k$  を変化させると今度は図形の一部が消えていくことがわかるだろう。Hopf写像が

#### 4次元の全体像を

ほぼ表しているのので、1つの変数の影響を小さくするとその面が消えていくわけである。

この絵だけではなかなかこの立体写像をイメージするのは難しい。

そこで3次元の場合の最初のデモでは3次元の中の球の表面を曲線でたどると、

立体射影が同じように曲線で得られた。同じように4次元の球面上を曲線に沿って進んでいくと立体射影された3次元内にどんな曲線を描くか見てみよう。

そのためには変数  $\psi$  を固定してみればよい。

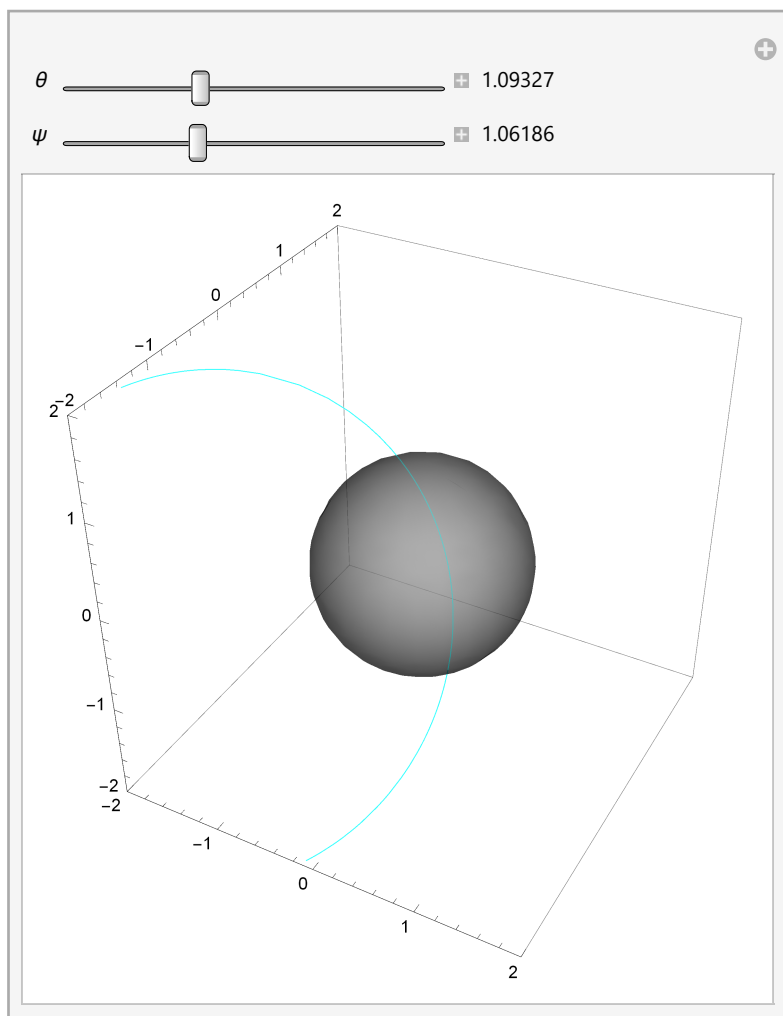
#### 【Mathematica入力】

$\psi$  を選べるように修正した。また、線には色をつけ、参考のために半径1の球面を表示させた。

```

Clear[θ, zc, wc, ψ, φ, x, y, z];
Manipulate[
  zc = Cos[θ] Exp[-i (ψ + φ)];
  wc = Sin[θ] Exp[-i (ψ - φ)];
  gc1 = ParametricPlot3D[ {Cos[φ1] Sin[θ1], Sin[φ1] Sin[θ1], Cos[θ1]},
    {φ1, 0, 2 π}, {θ1, 0, π}, PlotStyle → Opacity[0.5], Mesh → None,
    PlotRange → {{-2, 2}, {-2, 2}, {-2, 2}}];
  gc2 = ParametricPlot3D[Evaluate[Pj[zc, wc]], {φ, 0, 2 π},
    PlotRange → {{-2, 2}, {-2, 2}, {-2, 2}}, PlotStyle → Hue[0.5, 0.8, 1]];
  Show[gc1, gc2]
  , {{θ, π/6, "θ"}, 0, π, Appearance → "Labeled"}
  , {{ψ, π/2, "ψ"}, 0, π, Appearance → "Labeled"}
  Initialization → {Pj[x_, y_] :=  $\frac{1}{1 - \text{Re}[y]}$  {Re[x], Im[x], Im[y]}}]

```



真中の球に絡みながら円輪が回転していく様子からトーラスの表面が想像できる。さらに $\psi, \phi$ の範囲を上プログラムは $\pi$ としているがこれを $2\pi$ に変えてみるといい。 $2\pi$ までの回転で球のまわりを2回、周っていることがわかるだろう。物理学で全体像を見る時には少し変化させて比べて見るという方法をよくとる。

そこでここで $\psi$ を固定して、固定した $\psi$ を少しずつ変化させて見てみよう。

【Mathematica入力】

次のプログラムでは $\psi$ を少しずつ変化させ、 $\pi$ だけ全部で変化するようにしてある。

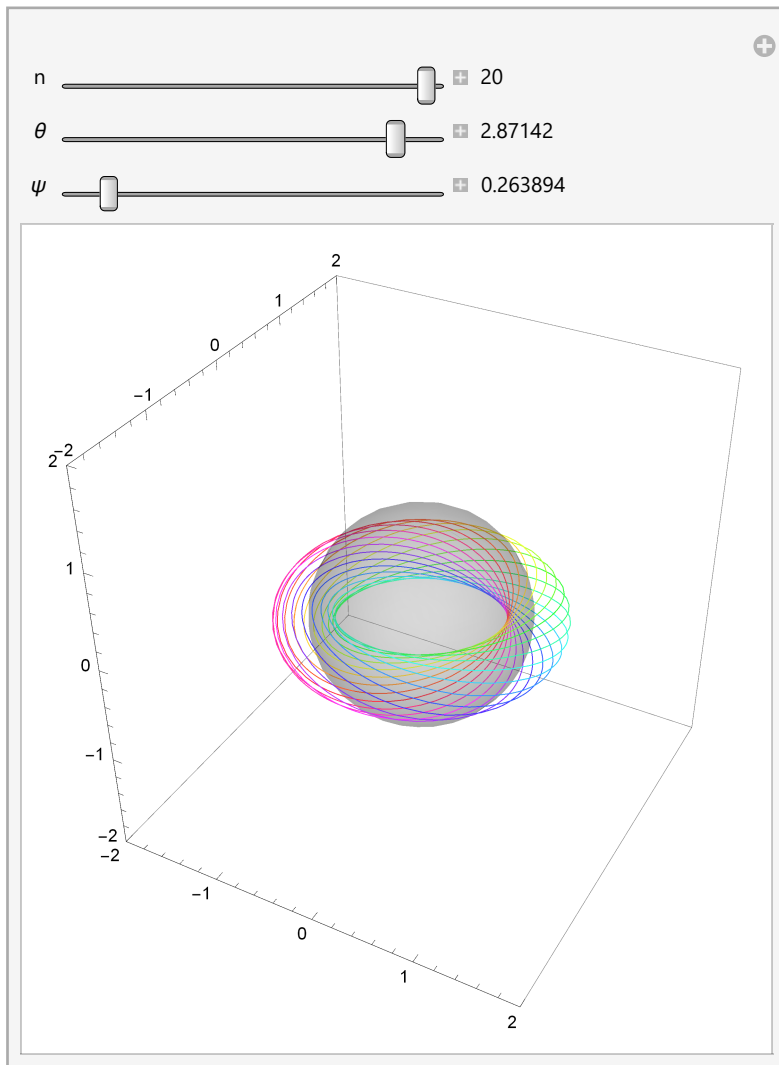
$n$ を多きすると刻みが小さくなり、全体像が見えてくるが、環境によっては重くなるかもしれない。

色も変化するように修正した。

```

Clear[θ, zz, wc, ψ1, φ, x, y, z];
Manipulate[
  gg1 = ParametricPlot3D[ {Cos[φ1] Sin[θ1], Sin[φ1] Sin[θ1], Cos[θ1]},
    {φ1, 0, 2 π}, {θ1, 0, π}, PlotStyle → Opacity[0.2], Mesh → None,
    PlotRange → {{-2, 2}, {-2, 2}, {-2, 2}}];
  gg2 = Table[ParametricPlot3D[Evaluate[Pj[Cos[θ] zz[ψ1, φ], Sin[θ] zz[ψ1, -φ]]],
    {φ, 0, 2 π}, PlotRange → {{-2, 2}, {-2, 2}, {-2, 2}},
    PlotStyle → Hue[ψ1 / π, 0.8, 1]], {ψ1, ψ, ψ + π, π / n}];
  Show[gg1, gg2]
  , {{n, 4, "n"}, 1, 20, 1, Appearance → "Labeled"}
  , {{θ, π / 6, "θ"}, 0, π, Appearance → "Labeled"}
  , {{ψ, π / 2, "ψ"}, 0, π, Appearance → "Labeled"}
  Initialization ⇒ {Pj[x_, y_] :=  $\frac{1}{1 - \text{Re}[y]}$  {Re[x], Im[x], Im[y]},
    zz[ψ_, φ_] := Exp[-i (ψ + φ)]};]

```



$n$ を大きくすると、はじめのトーラスの形が浮かびあがる。しかし、その曲線はねじれていることがわかる。

さて、4次元の世界の動きを3次元の世界で見ることに少しでも興味をもてただろうか。最後に重要な発想の逆転をしてほしい。

上の絵には我々にとって身近な3次元の球面がある。どの色の線も4次元にある球面上を1周する。

逆にこの曲線と3次元球の交点が必ず2つあることに注目してほしい。

3次元のこの「点」は4次元でみるとこの点を突き抜けている円周なのだ。

この円周上のどの座標も3次元の球面ではこの2点を表す。

4次元の世界では3次元の決まった点を表す座標が無数にあるともいえる。

この考えは射影の反対でHopf-fibrationという。一般的にファイバーと呼ばれるものだ。