Althlooti法と３次元形空間

Althlooti法によ回転同一視

Althlooti法では、形が$k \times 3$ 行列で表されているときに

「O_2と回転を無視して形を比較するためにO_1の配置を最適化する３次元空間回転行列$(V_1|V_2)」を

$$\hat{R}(V1|V2) = argmin_R||RV_1 - V_2||^2$$

と定め、$\hat{R}(V1|V2)$を線形代数的に求める。

ただしViは$k \times 3$行列であり、特定の形の特定の回転置かれ方に対応する。 また、$||V||^2 = 1$に標準化されているものとする。

回転同一視空間はどんな様子をしているか？

あるオブジェクトの特定の置かれ方と比較する

$||V||^2 = 1$を満たしている$k \times 3$行列の集合は、$k \times 3$次元空間に置かれた 単位球面を構成している。自由度$k \times 3 - 1$の多様体である。

今、ある$V0$を基準として定めることにする。 これはある特定のオブジェクトがある特定の置かれ方をしているときに、 すべてのオブジェクトを、このオブジェクトの置かれ方$V0$と最適な形比較をするように、 すべてのオブジェクトを回転して、 オブジェクトごとに特定の置かれ方のみを考えることに相当する。

各オブジェクトには3D回転分の無限の置かれ方があるので、それを1点にまとめるという作業である。

3D回転の自由度は3なので、 このまとめられた形置かれ方集合は $k \times 3 - 1 - 3 = k \times 3 - 4$の自由度の単位超球面の連続な部分となる。

自由度$k \times 3 - 4$の超球面は、$k 3 -3 $次元空間に置けるから、結局、

$k \times 3 - 3$ 次元空間の自由度$k \times 3 - 4$の単位球面 $S_{k \times 3 - 4}$ の連続な部分である。

k = 2 の例を図示しておく

$k = 2$のとき、$k \times 3 - 3 = 3$次元空間に置かれた、いわゆる普通の単位球面S_2の一部となることを以下に図示す。

緑が$V0$に相当する点。 赤がランダムに発生させた色々なオブジェクトの置かれ方を最適化したときの形置かれ方に対応する点。

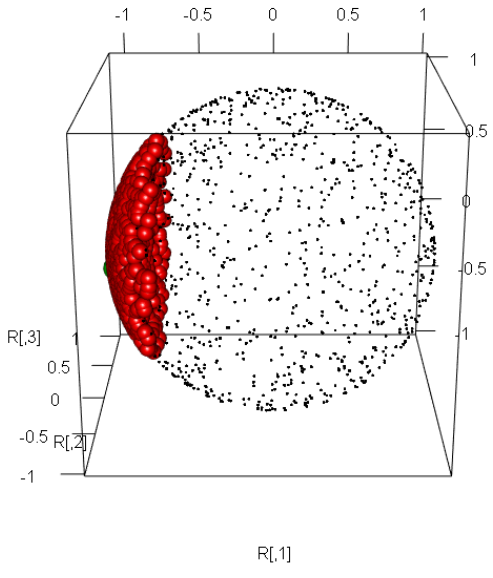
赤の点はS_2の部分領域を閉め、その内部に緑の点がある。

これが、$k=2のときの、形空間の様子である。

$k > 2$の場合はこれを幾何的に一般化したものと考えて想起すればよい。

```
n.pt <- 1000
k <- 2
d <- 3
V0 <- my.runit.matrix(k,d) # 基準の形の、特定の置かれ方

Vs <- matrix(0,n.pt,k*d)
Vs.rot <- Vs

for(i in 1:n.pt){
  tmp <- my.runit.matrix(k,d)
  al.out <- my.althlooti(tmp,V0)
  Vs[i,] <- c(tmp)
  Vs.rot[i,] <- c(al.out$RotX1)
}
```

```
# 座標を与える
Vsall <- rbind(c(V0),Vs.rot)
H <- Vsall %*% t(Vsall)
eigen.out <- eigen(H)
X <- eigen.out[[2]][,1:3] %*% diag(sqrt(eigen.out[[1]][1:3]))

R <- matrix(rnorm(1000*d),ncol=d)
R <- R/sqrt(apply(R^2,1,sum))
plot3d(R)
spheres3d(X[2:(n.pt+1),],col=2,radius=0.05)
spheres3d(X[1,],col=3,radius=0.08)
```



回転同一視の部分球面は$V0$の取り方で形が変わる

オブジェクトが同じなら置かれ方は形空間を変えない

あるオブジェクトの、ある特定の置かれ方を決めると、上述のように、球面の部分集合が、形空間として得られる。

今、同じオブジェクトを取り上げ、その置かれ方を変えたとする。

形空間は変わらない。$k \times 3$行列の値は変わるが、それらの回転同一視したときの距離は変わらないからである。

オブジェクトを変えると形空間は変わる

オブジェクトが変え、異なるオブジェクトのある置かれ方を基準として採用すると、$S_{k\times 3-4}$の連続部分が形空間となるが、

形空間は変わる。

この形空間が変わるというのは２つの意味がある。

- 連続部分の輪郭が変わる
- 連続部分に並びあう形の相互位置関係が変わる

形空間は大円が張る部分空間である

形空間が$S_{k\times 3-4}$の連続部分であるとわかったが、その輪郭はどういう意味があるのだろうか？

球面の部分である、形空間に、基準点$V0$に対応する点を通る「大円」を引く。

「大円」は$S_{k\times 3-4}$の上にあるS_1(普通の円周)であって、$V0$の対蹠点 $-V0$を通るものである。

今、$V0$に対して、あるオブジェクトの最適配置を見つけたところ、$V1$だったとする。

$$U = pV0 + qV1, ||U||^2 = 1$$

のように線形和で表される単位球面上の点は、あるオブジェクトのある置かれ方を表しているが、 この置かれ方は、$V0$に対してすでに最適置かれ方(または特定の置かれ方)になっていることが示せる。

そして、このオブジェクトは$V0$オブジェクトの形とも$V1$オブジェクトの形とも異なる。

したがって、$V0$と$V1$とを通る大円上の点は、「$V0$から少しずつ変形し$V1$になり、どんどん変形を続けると$-V0$に相当する形になり、最後には$V0$に戻る」ようなオブジェクトのシリーズであり、その置かれ方は、$V0$と$V1$とを含むある範囲の弧では、$V0$に対する最適置かれ方にているようなものである。

また、$V0$,$V1$を含む弧でない部分では、$V0$に対する最適置かれ方になっていないが、ある3D次元回転軸を選んで、πだけ回すと、$V0$に対する最適置かれ方になるような置かれ方であることも示せる。

したがって、形空間「$S_{k\times 3-4}$球面の部分」とは、 この大円のうち、$V0$に対する最適置かれ方になっている弧を全方向に集めたものであり、 その周辺境界は、 この大円が持つ「最適置かれ方の範囲と、最適置かれ方にするにはπ回転する必要のある範囲との境界」に対応する。

形空間を張る

最適置かれ方になっている形とその置かれ方は、$V0$と、最適置かれ方された別のオブジェクトの$V1$との線形和で表せることを上で書いた。

今、$V0$のほかに、$k \times 3 - 4$個の線形独立な、最適置かれ方$V1, \ldots$が与えられると、任意の$V0$に対する最適置かれ方オブジェクトは、$V0, V1, \ldots, V_{k\times 3-4}$の線形和で表せる。

言い換えると、形空間は、基準$V0$と、$k \times 3 - 4$個の独立な最適な置かれ方行列が張る部分超球面であると言える。

どうして線形結合になるのか？

Althlooti法では、 最適回転を次のようにして求めている。

$V0, V1$が与えられたとき

$$M = V1^T V0$$

なる3×3行列Mを計算する。

ついで、Mの成分のある線形計算によって、4×4 行列Nを作り、その固有値分解をして、 最大固有値を与える、固有ベクトルを取り出す。

この固有ベクトルは3D回転を表すクオータニオンになっている。 このクオータニオンの第１成分は実部であり、それは、回転角θに対して$\cos \theta/2$になっており、虚部は回転軸を表している。

ここで、得られる最適回転によって$V1$を3D回転し、回転した後の$V1'$について、再度、最適回転計算をすると、上述の4×4行列Nは

$$N = \begin{pmatrix} \alpha, 0, 0, 0 \\ 0, \beta_{2,2}, \beta_{2,3}, \beta_{2,4} \\ 0, \beta_{3,2}, \beta_{3,3}, \beta_{2,4} \\ 0, \beta_{4,2}, \beta_{4,3}, \beta_{4,4} \end{pmatrix}$$

という形をしている。

このNの固有値分解では、必ず、$(1, 0, 0, 0)$なる固有ベクトルと、$(0, x, y, z)$なる３つの固有ベクトルが得られる。

このようなNをもたらすVの線形和は、やはり同じ構成のNを生じるので、その固有ベクトルは
$(1, 0, 0, 0)$か、$(0, x, y, z)$かになる。

$(1, 0, 0, 0)$は無回転に相当するが、それは「最適配置になっている」ことを意味するし、

$(0, x, y, z)$に対応する固有値が最大固有値であれば、その回転角θは$\cos \theta/2 = 0$を満足するから、$\theta = \pi$となる。

以下は上記の考察のためのごちゃごちゃとした試行錯誤であるので見るに値しない

回転同一視での形集合の形

結論から先に言う。

$k \times 3$ 次元空間に置かれた、$k \times 3 - 4$ 自由度の超球面の連続な一部である。

特定の$V0$とそれに対応する形置かれ方$Vx_{(V0)}$

あるオブジェクトの特定の回転置かれ方$V0$を取り上げると、すべての形には、$V0$に対応した $\hat{R}(Vx|V0)$が決まり、 それに対応して その形の特定の置かれ方

$$\hat{Vx}_{(V0)} = \hat{R}(Vx|V0)Vx$$

が決まる。

形置かれ方の線形関係

形置かれ方が定める大円

$V0$を定め、ある形について、対応する置かれ方$Vx_{(V0)}$が得られたとする。

そのとき、 ある形置かれ方の集まり

$$U = a_0 V0 + a_x V x_{(V0)}; ||U||^2 = 1$$

を考える。

ちなみに、

$$U = a_0 V0 + a_x V x_{(V0)}; ||U||^2 = 1$$

で表されるUは、$V0$と$Vx_{(X0)}$とを通る$k \times 3$次元空間に置かれた単位超球表面の「大円」になっている。 この「大円」は２次元平面に置かれた普通の単位円である。

そして、この「大円」は$V0$の対蹠点である$-V0$($V0$のすべての成分の符号を反転した行列)を通る。

このUの要素は、次のいずれかであることが示せる。

- ちょうど$V0$に対して最適化された置かれ方になっている
- Uをある軸について３次元回転してちょうどpi回した形置かれ方、U_πが$V0$に対して最適化された置かれ方になっている

そして * U自体が最適な置かれ方になっている場合は、大円のうち、$V0$をはさむ前後の弧をなし、U_πが最適な置かれ方になっている場合は、残りの部分(対蹠点を挟む弧)となる

この表現についての解釈上の補足

形とその置かれ方は$k \times 3$次元空間の単位球面上の点である。

ある形を選び、その置かれ方も選ぶ($V0$)。

そうすると、すべての形には“$V0$に対応する”最適な置かれ方が定まる。

それぞれの形置かれ方には、「特定の形」の「無数の置かれ方」行列が対応する。

一方、２つの形を選び、$V0$とVxとしたとき、$Vx_{(X0)}$が決まるが、今、$V0$からあるルールで変形して$Vx_{(V0)}$まで変えていくとする。

その変形はうまくできていて、その変形をする限り、$V0$に対する最適な置かれ方になっているものとする。

その変形を続けて行くと、突然、$V0$に対する最適な置かれ方ではなくなることになる。

それを無視して、そのルールで変形を続けると、$-V0$という行列に達し、さらに変形を続けていくと、また突然に$V0$に対する最適な置かれ方になる。

そのまま変形を続けていくと$V0$に戻る。

この大円上の形置かれ方のうち、$V0$に対する最適な置かれ方になっていないものは、ある軸に関するπ回転が最適な置かれ方になっている。

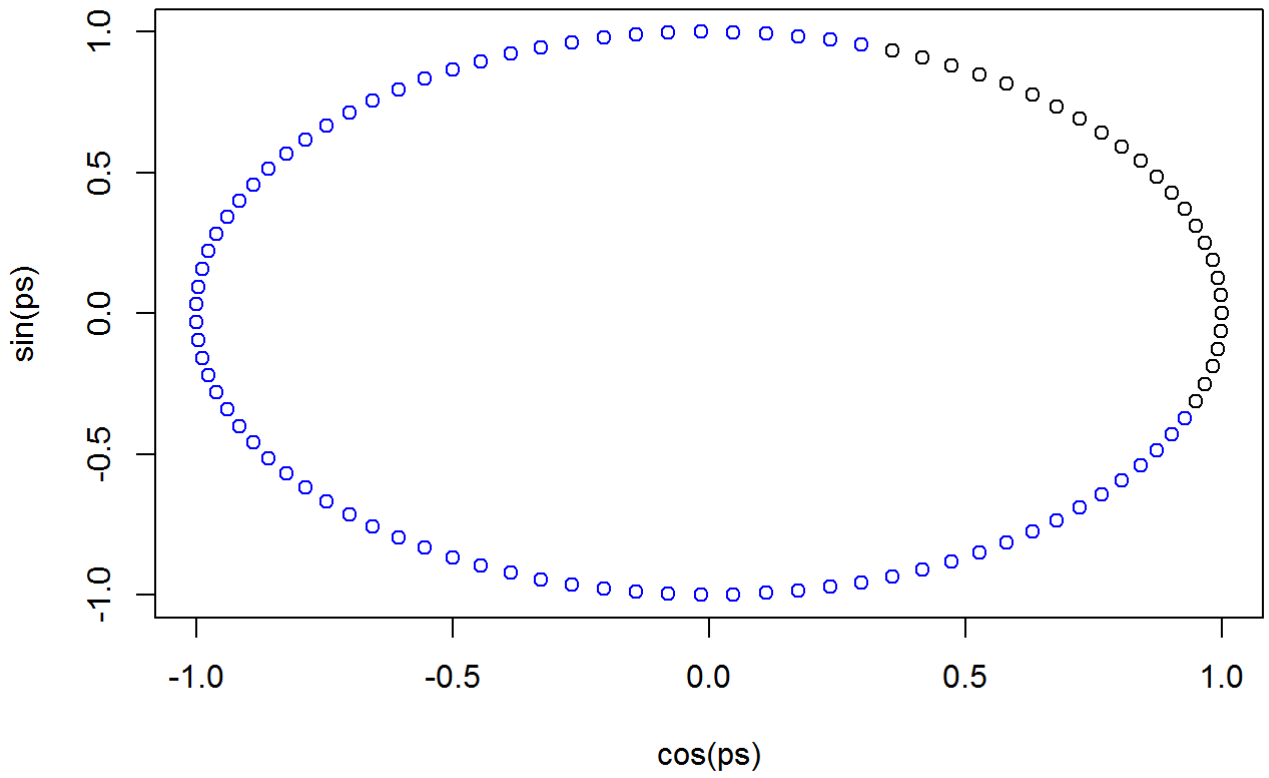
また、$V0$に対する最適な置かれ方になっていないものを、$V0$の最適置かれ方にしてやり、それをUとすると、$V0$からUを通る「大円変形」が別途、定まる。

この変形も、始めのうちは最適置かれ方になっているが、どこか(Uを通り越したどこか)で球に最適置かれ方でなくなり、$-V0$を通り、また、いつしか最適置かれ方になり、$V0$に戻る。

```
k <- 2
d <- 3
V0 <- my.runit.matrix(k,d)
V1 <- my.runit.matrix(k,d)
V1_v0 <- my.althlooti(V1,V0)$RotX1

ps <- seq(from=0,to=2*pi,length=100)
Vs <- matrix(0,length(ps),k*d)
Vs2 <- Vs
angles <- rep(0,length(ps))
for(i in 1:length(ps)){
  tmp <- my.vector.sum.sp2(V0,V1_v0,ps[i])$z
  Vs[i,] <- c(tmp)
  al.out <- my.althlooti(tmp,V0)
  Vs2[i,] <- c(al.out$RotX1)
  angles[i] <- acos(Re(al.out$q))*2
}
```

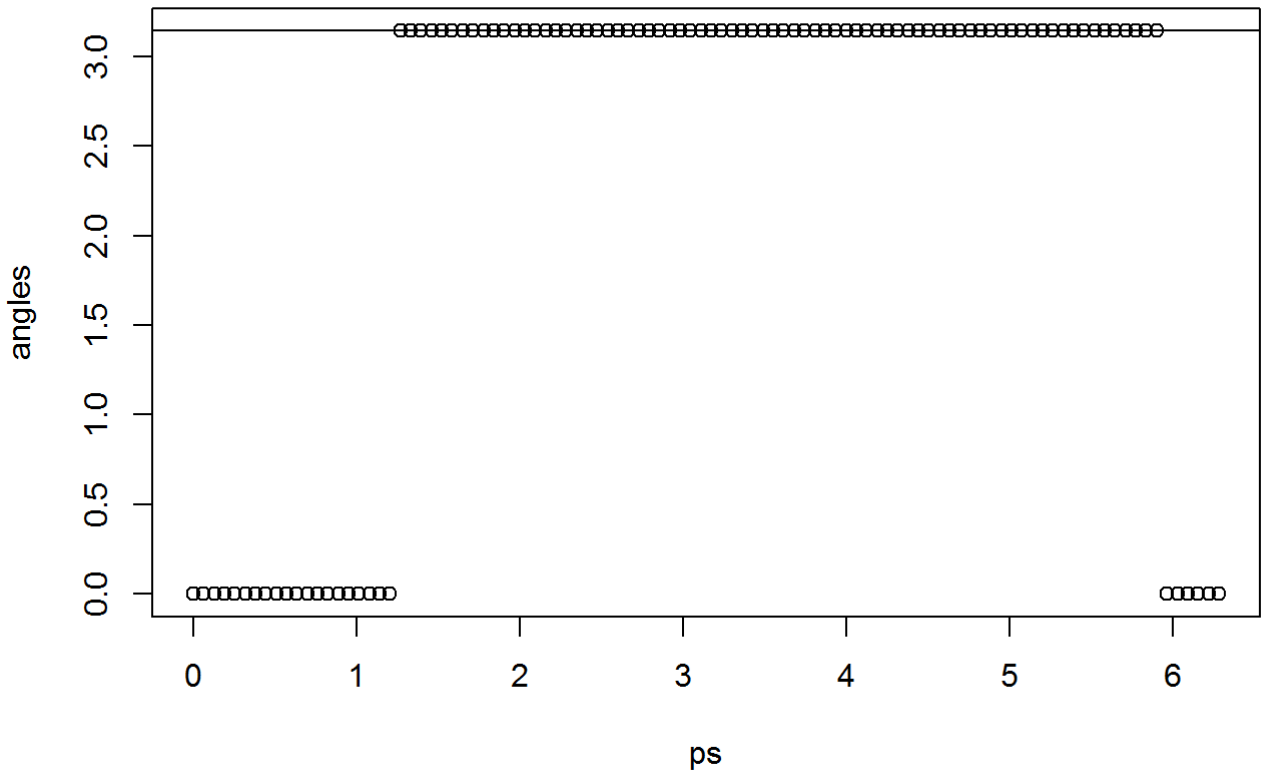
大円をぐるりと回すと、最適回転されている(黒)か、そうでないか(位置)が解る。

```
plot(cos(ps),sin(ps),col = angles+1)
```



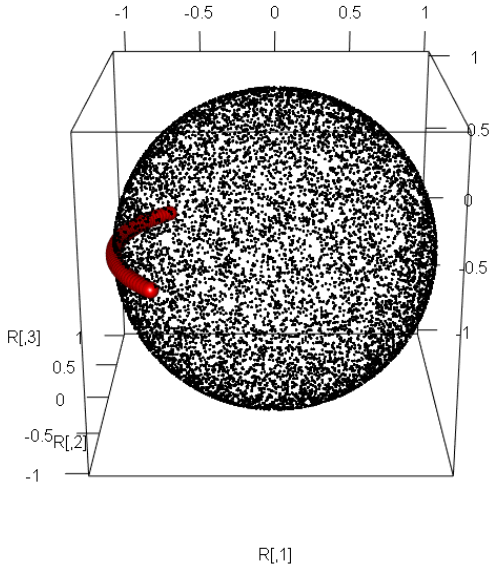
最適回転になっていない場合はどのくらい回転するかを示すとπになっていることも解る

```
plot(ps,angles)
abline(h=pi)
```



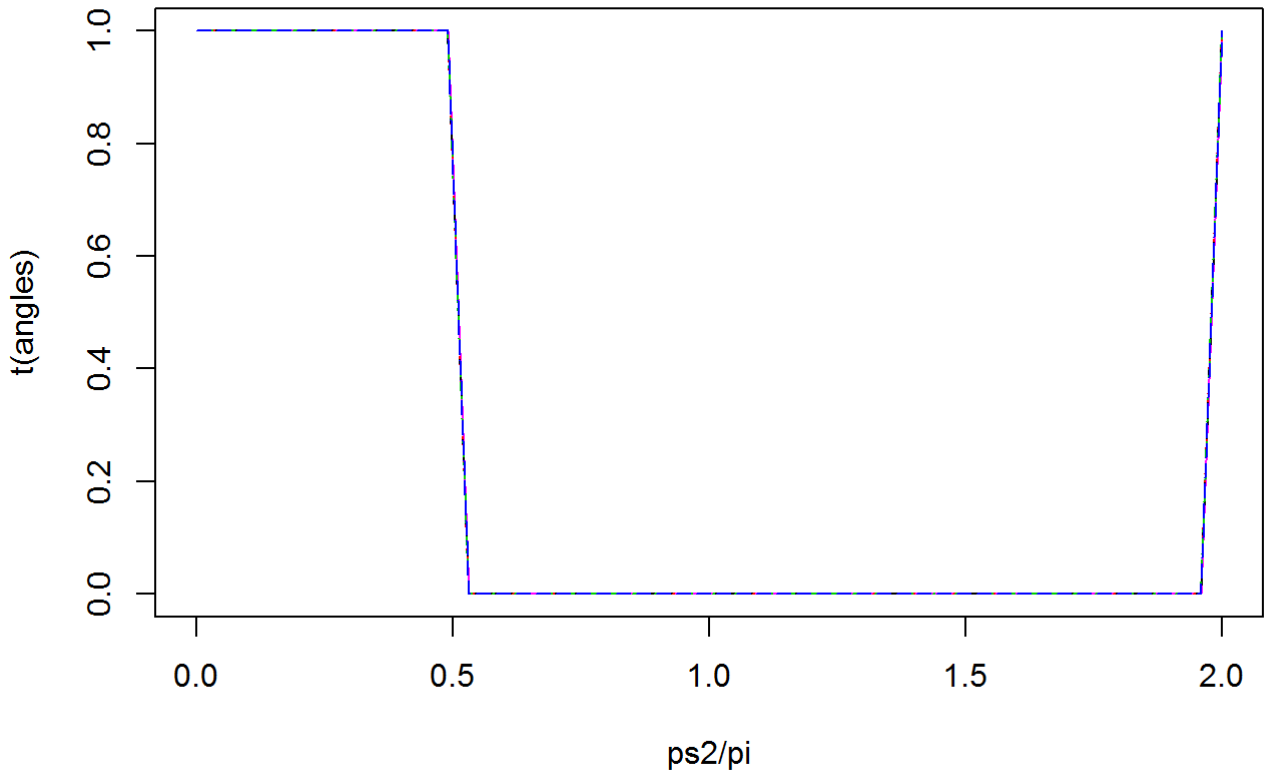
$V0$を通る大円の最適回転後の内積から、低次元写像を作ると...

```
tmp <- Vs2 %*% t(Vs2)
eigen.out <- eigen(tmp)
ev <- eigen.out[[1]]
ev[which(ev<0)] <- 0
H <- eigen.out[[2]]  %*% diag(sqrt(ev))
R <- matrix(rnorm(10000*3),ncol=3)
R <- R/sqrt(apply(R^2,1,sum))
plot3d(R)
spheres3d(H[,1:3],col=2,radius=0.05)
```

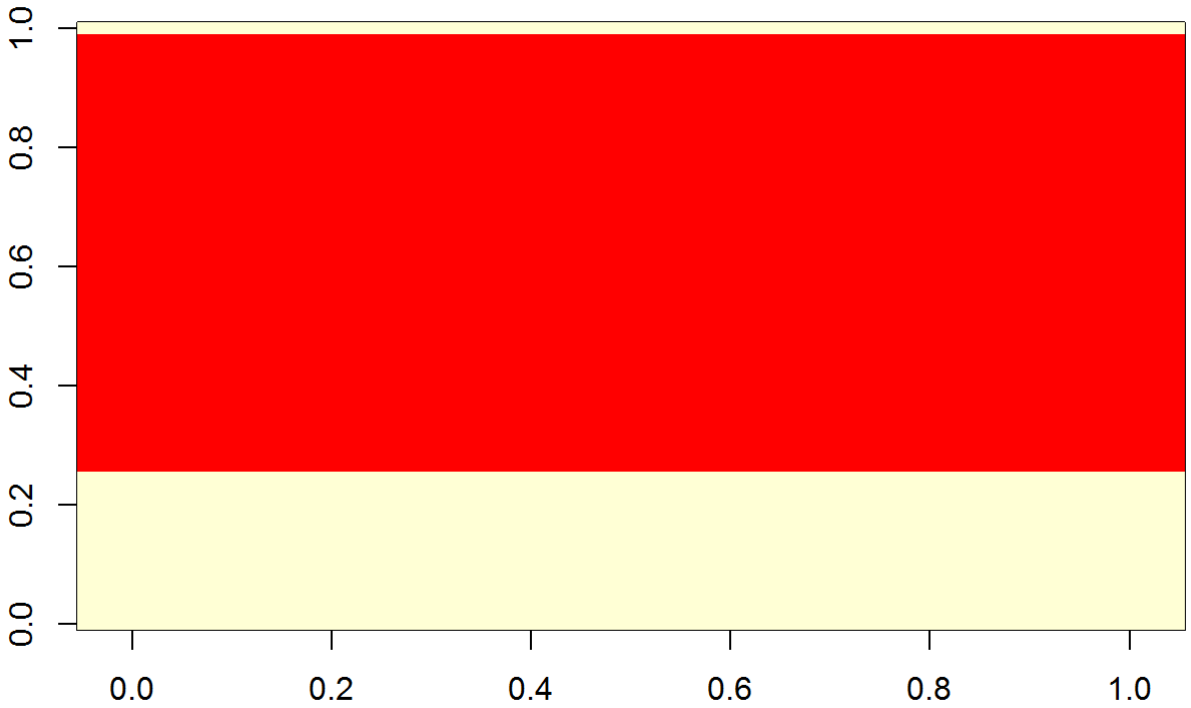

-1
-0.5
0
0.5
1
1
0.5
0
-0.5
-1
R[,3]
1
0.5
0
-0.5
-1
R[,2]
R[,1]

```
k <- 20
d <- 3
X <- matrix(rnorm(k*d),ncol=d)
X <- matrix(c(1,0,0,0,0,0),k,d)
X <- matrix(rep(1/sqrt(6),6),k,d)
X <- X + rnorm(6)*0.001
X <- X/sqrt(sum(X^2))
Y1 <- matrix(rnorm(k*d),ncol=d)
Y2 <- matrix(rnorm(k*d),ncol=d)

X <- X/sqrt(sum(X^2))
Y1 <- Y1/sqrt(sum(Y1^2))
Y2 <- Y2/sqrt(sum(Y2^2))

al.out1 <- my.althlooti(Y1,X)
Y1.rot <- al.out1$RotX1
al.out2 <- my.althlooti(Y2,X)
Y2.rot <- al.out2$RotX1


tmp.out1 <- my.vector.sum.sp2(c(X),c(Y1.rot),pi/2)
tmp.out2 <- my.vector.sum.sp2(c(X),c(Y2.rot),pi/2)

Y1.rot.90 <- matrix(tmp.out1$z,ncol=d)
Y2.rot.90 <- matrix(tmp.out2$z,ncol=d)

Y.series <- list()
ps <- seq(from=0,to=1,length=10)
for(i in 1:length(ps)){
  tmp <- my.vector.sum.sp(c(Y1.rot.90),c(Y2.rot.90),ps[i])
  Y.series[[i]] <- matrix(tmp$z,ncol=d)
}
ps2 <- seq(from=0,to=2*pi,length=50)
angles <- matrix(0,length(ps),length(ps2))
for(i in 1:length(Y.series)){
  #angles[[i]] <- rep(0,length(ps2))
  for(j in 1:length(ps2)){
    tmp.out <- my.vector.sum.sp2(c(X),c(Y.series[[i]]),ps2[j])
    Z <- matrix(tmp.out$z,ncol=d)
    al.out <- my.althlooti(Z,X)
    tmp.q <- al.out$q
    angles[i,j] <- Re(tmp.q)
  }

}
```

```
matplot(ps2/pi,t(angles),type="l")
```




```
image(angles)
```



```
table(round(angles,15))
```

```
##
## -1.97e-13 -1.18e-13 -1.11e-13  -9.7e-14  -8.3e-14  -6.6e-14  -6.1e-14
##         1         1         1         1         2         1         1
##  -5.9e-14  -5.8e-14  -5.5e-14  -5.3e-14  -5.2e-14  -5.1e-14    -5e-14
##         1         1         1         1         1         1         1
##  -4.9e-14    -4e-14  -3.9e-14  -3.6e-14  -3.1e-14    -3e-14  -2.7e-14
##         1         1         1         1         1         1         2
##  -2.6e-14  -1.9e-14  -1.7e-14  -1.5e-14    -8e-15    -6e-15         0
##         2         1         1         2         1         1       320
##     1e-15     5e-14   5.1e-14   5.6e-14   7.4e-14   8.1e-14         1
##         4         1         1         1         1         1       140
```

```
ps <- seq(from=0,to=4*pi,by=0.01)
n.iter <- length(ps)
angles <- newangles <- ips <- rep(0,n.iter)
qmat1 <- qmat2 <- matrix(0,n.iter,9)
qh1 <- qh2 <- rep(0+Hi,n.iter)
diffs <- rep(0,n.iter)
k <- 2
d <- 3
X <- matrix(rnorm(k*d),ncol=d)
Y <- matrix(rnorm(k*d),ncol=d)
X <- X/sqrt(sum(X^2))
Y <- Y/sqrt(sum(Y^2))
al.out <- my.althlooti(Y,X)
Y.rot <- al.out$RotX1
for(i in 1:n.iter){

#p <- runif(1)*10
p <- ps[i]
#p<-1
tmp.out <- my.vector.sum.sp2(c(X),c(Y.rot),p)
Z <- matrix(tmp.out$z,ncol=d)
al.out2 <- my.althlooti(Z,X)
#Matrix::rankMatrix(rbind(X,Y.rot,Z))
#my.althlooti(Y.rot,X)$qmat
#my.althlooti(Y.rot,Z)$qmat
angles[i] <- tmp.out$angle
newangles[i] <- tmp.out$newangle
ips[i] <- tmp.out$ip
al1 <- my.althlooti(Y.rot,X)
al2 <- my.althlooti(Y.rot,Z)
qmat1[i,] <- al1$qmat
qmat2[i,] <- al2$qmat
qh1[i] <- al1$q
qh2[i] <- al.out2$q
#Rotmat.X2Y <- al.out$qmat
#Z1 <- t(Rotmat.X2Y %*% t(X))
#Z2 <- t(Rotmat.X2Y %*% t(Z))
#al.out2 <- my.althlooti(X,Z2)
diffs[i] <- sum((tmp.out$V1 * tmp.out$comp1 + tmp.out$V2 * tmp.out$comp2 -Z)^2)
}
```

複数の大円が作る形置かれ方集合

$X0$に対して、複数のオブジェクトの形置かれ方最適化を行うと、 複数の$Vi_{(X0)}$が得られる。

このとき

$$U = a_0 V0 + \sum_{i=1} a_i Vi_{(V0)} = \sum_{i=0} a_i Vi_{(V0)}; ||U||^2 = 1$$

のような形置かれ方も、やはり

- ちょうど$V0$に対して最適化された置かれ方になっている
- Uをある軸について３次元回転してちょうどpi回した形置かれ方、U_πが$V0$に対して最適化された置かれ方になっている

であることが示せる。

今、$k \times 3$行列の自由度はたかだか$k \times 3$であるから、$Vi_{(X0)}$をいくつか集めてくれば、それ以外の形置かれ方はそれらの線形和として書き表せることも意味する。

大円が張る最適置かれ方集合

結局$V0$を定めると、その周りに、大円の線形和で定まる$k \times 3$行列の部分集合があって、その線形和には「ここまでは最適置かれ方になっている」という「限り」がある。

この限りの外の形置かれ方は、 ３次元回転することにより、別の線形和表現にできて、それは「限り」の内側に対応する。

$$U = \sum_{i=0} a_i Vi_{(X0)}$$

$$U' = RU = \sum_{i=0} a_i RVi_{(X0)}$$

３次元閉曲面を表す$k \times 3$ 行列

３次元閉曲面オブジェクトの形がk個の３次元ベクトル

$$V = (v_1, \ldots, v_k,); v_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

にて特徴づけられているとする。

この３次元ベクトルのセットには２つの取り方がある。

- 1. ３次元オブジェクトの表面にk個のメルクマール点がある場合。個々のメルクマール点の３次元座標をそれとする
- 2. ３次元オブジェクトを単位球面にマップし、単位球面上にオリジナルオブジェクトのx,y,z座標の場を作り、それぞれを球面調和関数分解し、k個の係数を取り出す。この$k \times 3$個の係数を、対応するx,y,zのトリオとしたものを個々の３次元ベクトルとする

ただし、オブジェクトの位置と大きさを無視し、形のみに着目するため

$$||V||^2 = \sum_{i=1}^{k} ||v_i||^2 = 1$$

と標準化する。

回転の影響を無視して形を比較するための回転

３次元空間での置かれ方を無視し、形の異同を次のようにする。

２つのオブジェクト0_iが$V_i = \left(v_{i,1}, \ldots, v_{i,k} \right), i = 1, 2$と表されているとしたとき、

「O_2と回転を無視して形を比較するためにO_1の配置を最適化する３次元空間回転行列$(V_1|V_2)」を

$$\hat{R}(V1|V2) = argmin_R ||RV_1 - V_2||^2$$

とする。

$\hat{R}(V_1|V_2)$はAlthlootiの方法により、線形代数的に求まることが知られている。

回転同一視をしたときの次元

集合$V = \{v_i\}$は$k \times 3$次元空間に置かれた単位超球を成している。

今、あるオブジェクトのある置かれ方に対応する V_0を決めたとき

すべてのv_iには、最適な回転

$$\hat{R}(v_i|V_0)$$

が決まる。

v_iを

$$\hat{R}(v_i|V_0)v_i$$

に移して考えてよい、とも言える。

今、v_iと形としては同じだが、配置のされ方が違うオブジェクト v_i'があったとすると、 v_i'には別の回転$\hat{R}(v_i'|V_0)$が定まり、

$$\hat{R}(v_i'|V_0)v_i'$$

に移して考えてよい。 これは$\hat{R}(v_i'|V_0)v_i' = \hat{R}(v_i|V_0)v_i$ な関係にある。

逆に言うと、「同じ形」とは、任意のV_0に対して

$$\hat{R}(v_i'|V_0)v_i' = \hat{R}(v_i|V_0)v_i$$

であるようにv_i, v_i'のことである。

に * 形は$k \times 3$の値で表されている。この時点で自由度は$k \times 3$

- $||V||^2 = 1$ という制約がある。自由度が1減って $k \times 3 - 1$
- 回転による違いを無視しているのでその分の自由度3が減って、$k \times 3 - 4$

となる。

これがどういうことかを図解も含めて以下に示す。

k=2の場合

自由度は$k \times 3 - 4 = 2 \times 3 - 4 = 2$。

``` ### 多数のオブジェクトは自由度2のS2上の点として表せる
```

```
n.obj <- 500
k <- 2
d <- 3
V0 <- my.runit.matrix(k,d)
#V0 <- matrix(c(1,0,0,0,0,0),k,d)
#V0 <- matrix(rep(1/sqrt(6),6),k,d)+rnorm(6)*0.001
#V0 <- V0/sqrt(sum(V0^2))

Vs <- list()
V.rots <- V.rots.inv <- matrix(0,n.obj,k*d)
library(MCMCpack)
```

```
## Warning: package 'MCMCpack' was built under R version 3.4.3
```

```
## Loading required package: coda
```

```
## Loading required package: MASS
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2018 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

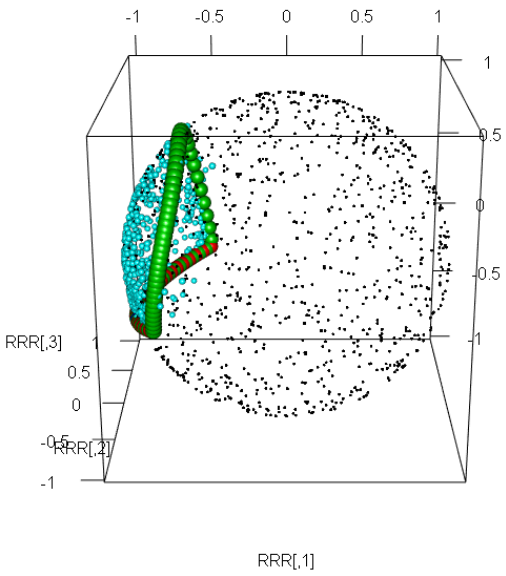
```
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
for(i in 1:n.obj){
  #Vs[[i]] <- my.runit.matrix(k,d)
  tmp <- rcauchy(k*d,0,10) * sample(c(-1,1),k*d,replace=TRUE)
  tmp <- tmp/sqrt(sum(tmp^2))
  Vs[[i]] <- matrix(tmp,k,d)
  al.out <- my.althlooti(Vs[[i]],V0)
  al.out.inv <- my.althlooti(Vs[[i]],-V0)
  V.rots[i,] <- c(al.out$RotX1)
  V.rots.inv[i,] <- c(al.out.inv$RotX1)
}

V0.inv <- -V0 + rnorm(6)*0.001
V0.inv <- V0.inv/sqrt(sum(V0.inv^2))

Y <- matrix(V.rots[1,],ncol=d)
ps <- seq(from=0,to=2*pi,length=100)
Yseries <- matrix(0,length(ps),k*d)
Yseries.check <- rep(0,length(ps))
for(i in 1:length(ps)){
  tmp.out <- my.vector.sum.sp2(c(V0),c(Y),ps[i])$z
  al.out <-  my.althlooti(matrix(tmp.out,ncol=d),V0)
  Yseries[i,] <- c(al.out$RotX1)
  if(Re(al.out$q)==1){
    Yseries.check[i] <- 1
  }
}
```

```
al.out.. <- my.althlooti(V0.inv,V0)
V0.inv.RotX <- al.out..$RotX1
V0Vrots <- rbind(c(V0),Yseries,V.rots,c(V0.inv.RotX))
ddd <- V0Vrots %*% t(V0Vrots)
eigen.out <- eigen(ddd)
```

```
eigen.out.1 <- eigen.out[[1]]
eigen.out.1[4:length(eigen.out.1)] <- 0
newv <- eigen.out[[2]] %*% diag(sqrt(eigen.out.1))

RRR <- matrix(rnorm(1000*3),ncol=3)
RRR <- RRR/sqrt(apply(RRR^2,1,sum))
plot3d(RRR)

spheres3d(newv[(1+1+length(ps)):(length(newv[,1])-1),1:3],radius=0.02,col=5)
spheres3d(newv[1,1:3],radius=0.05,col=4)
spheres3d(newv[length(newv[,1]),1:3],radius=0.05,col=6)

check0 <- which(Yseries.check==0)
check1 <- which(Yseries.check==1)
spheres3d(newv[(2:(1+length(ps)))[check0],1:3],radius=0.05,col=3)
spheres3d(newv[(2:(1+length(ps)))[check1],1:3],radius=0.05,col=2)
```




```
apply(newv[,1:3]^2,1,sum)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [246] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [316] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [351] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [386] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [421] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [456] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [491] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [526] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [561] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [596] 1 1 1 1 1 1 1
```

```
Matrix::rankMatrix(V.rots)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 1.110223e-13
```

```
Matrix::rankMatrix(V.rots.inv)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 1.110223e-13
```

```
V.rots.better <- matrix(0,n.obj,k*d)
for(i in 1:n.obj){
  tmp1 <- sum(c(V0)*V.rots[i,])
  tmp2 <- sum(c(V0)*V.rots.inv[i,])
  if(tmp1 <= tmp2){
    V.rots.better[i,] = V.rots.inv[i,]
  }else{
    V.rots.better[i,] = V.rots[i,]
  }
}
Matrix::rankMatrix(V.rots.better)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 1.110223e-13
```

```
ddd <- V.rots.better %*% t(V.rots.better)
eigen.out <- eigen(ddd)
```

```
eigen.out.1 <- eigen.out[[1]]
eigen.out.1[4:length(eigen.out.1)] <- 0
newv <- eigen.out[[2]] %*% diag(sqrt(eigen.out.1))
plot3d(newv[,1:3])
```

```
plot(apply(qmat2,1,mean),newangles)
plot(diffs,newangles)
plot(ps,Re(qh2))
abline(v=pi*(-4:4)*0.5)
library(rgl)
rot.v <- cbind(i(qh2),j(qh2),k(qh2))
#plot3d(rot.v[which(Re(qh2)==0,)])
plot(rot.v[which(Re(qh2)==0,1)])
```

```
tmp.out$V1 * tmp.out$comp1 + tmp.out$V2 * tmp.out$comp2 -Z
```

```
library(GPArotation)
library(onion)
k <- 5
d <- 3
X <- matrix(rnorm(k*d),ncol=d)
Y <- matrix(rnorm(k*d),ncol=d)
X <- X/sqrt(sum(X^2))
Y <- Y/sqrt(sum(Y^2))

al.out <- my.althlooti(Y,X)
Y.rot <- al.out$RotX1

rot.vec <- rnorm(d)
rot.vec <- rot.vec/sqrt(sum(rot.vec^2))
# theta <- runif(1) * 2*pi # 任意の角度で対応が取れることは確認済み
theta <- pi
rot.q <- my.rotq(rot.vec,theta)
R <- my.q2rotmat(rot.q)

X. <- t(R %*% t(X))
Y.rot. <- t(R %*% t(Y.rot))
Y.rot.2 <- my.althlooti(Y.rot.,X.)$RotX1
Y.rot.22 <- my.althlooti(Y,X.)$RotX1

range(Y.rot.-Y.rot.2)
```

```
## [1] -8.881784e-16  1.332268e-15
```

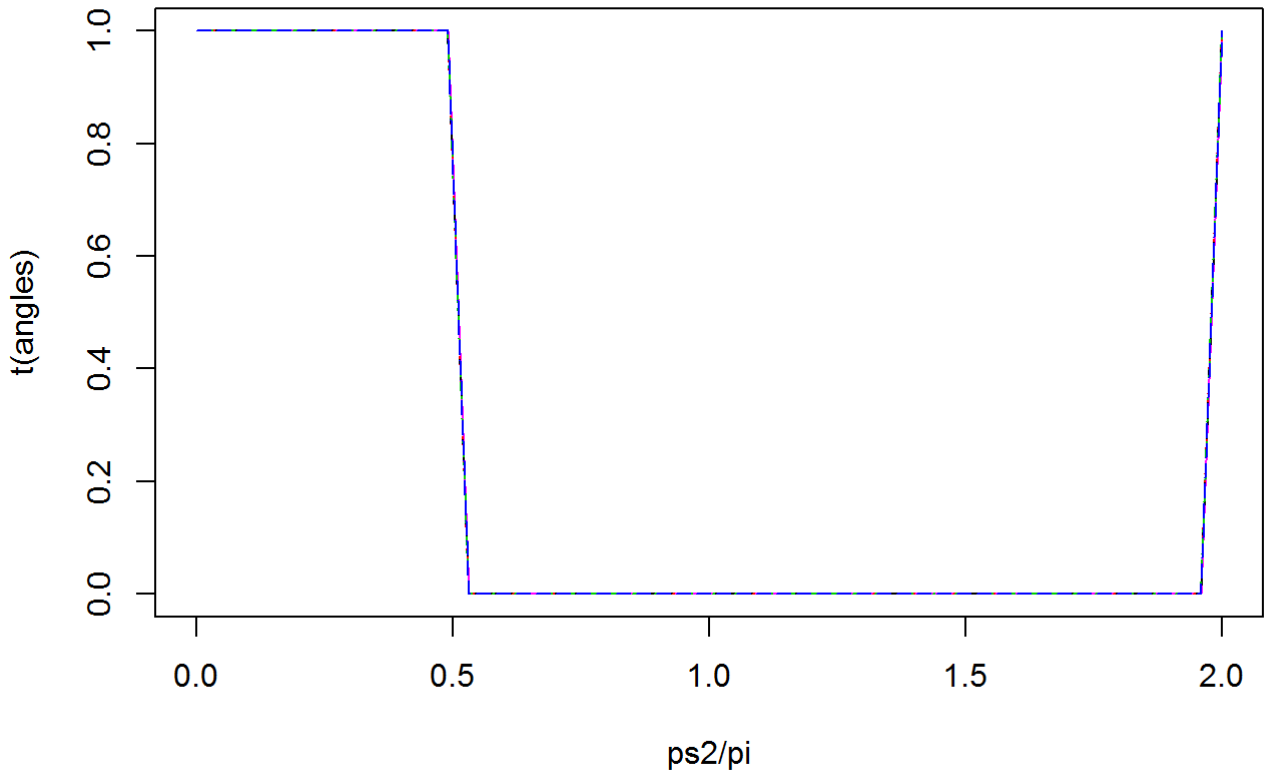
```
range(Y.rot.-Y.rot.22)
```

```
## [1] -8.881784e-16  1.523087e-15
```

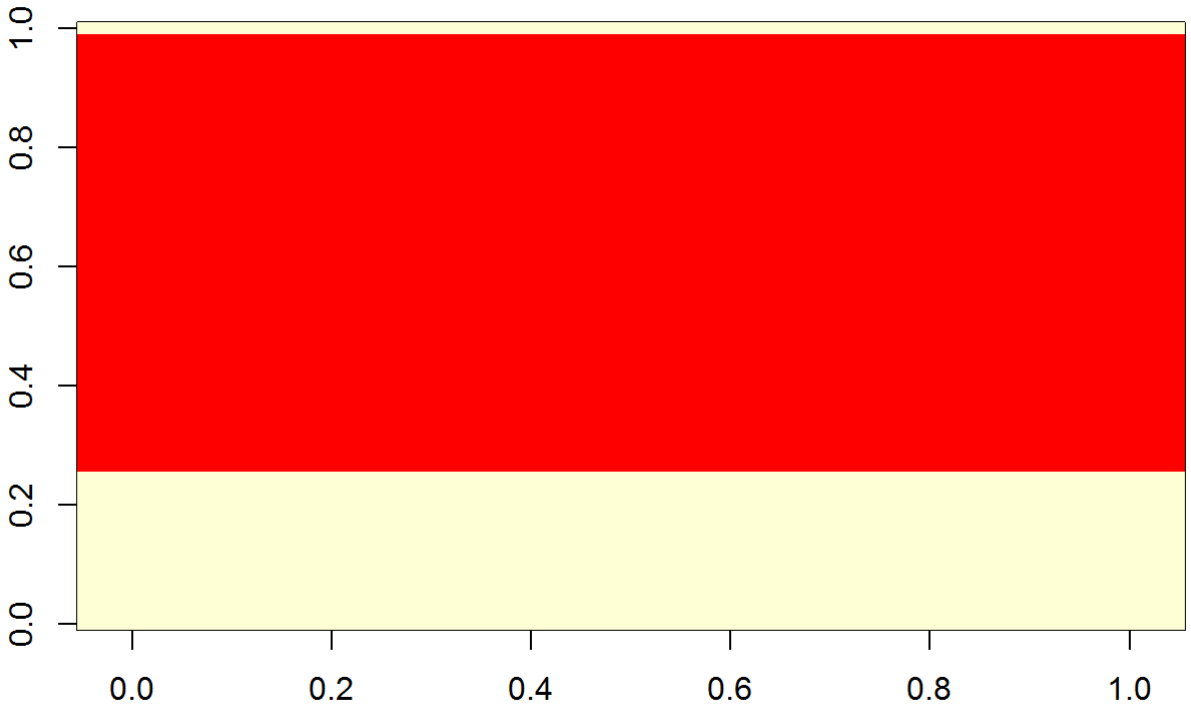
```
# 3d回転では大円は作ったらだめ(反転させることだから)
# X.. <- my.vector.sum.sp2(X,X.,pi)$z
# Y.rot.. <- my.vector.sum.sp2(Y.rot,Y.rot.,pi)$z
# Y.rot..2 <- my.althlooti(Y.rot..,X..)$RotX1
# Y.rot..22 <- my.althlooti(Y,X..)$RotX1

# range(Y.rot..-Y.rot..2)
# range(Y.rot..-Y.rot..22)
```

```
k <- 20
d <- 3
X <- matrix(rnorm(k*d),ncol=d)
X <- matrix(c(1,0,0,0,0,0),k,d)
X <- matrix(rep(1/sqrt(6),6),k,d)
X <- X + rnorm(6)*0.001
X <- X/sqrt(sum(X^2))
Y1 <- matrix(rnorm(k*d),ncol=d)
Y2 <- matrix(rnorm(k*d),ncol=d)

X <- X/sqrt(sum(X^2))
Y1 <- Y1/sqrt(sum(Y1^2))
Y2 <- Y2/sqrt(sum(Y2^2))

al.out1 <- my.althlooti(Y1,X)
Y1.rot <- al.out1$RotX1
al.out2 <- my.althlooti(Y2,X)
Y2.rot <- al.out2$RotX1



tmp.out1 <- my.vector.sum.sp2(c(X),c(Y1.rot),pi/2)
tmp.out2 <- my.vector.sum.sp2(c(X),c(Y2.rot),pi/2)

Y1.rot.90 <- matrix(tmp.out1$z,ncol=d)
Y2.rot.90 <- matrix(tmp.out2$z,ncol=d)

Y.series <- list()
ps <- seq(from=0,to=1,length=10)
for(i in 1:length(ps)){
  tmp <- my.vector.sum.sp(c(Y1.rot.90),c(Y2.rot.90),ps[i])
  Y.series[[i]] <- matrix(tmp$z,ncol=d)
}
ps2 <- seq(from=0,to=2*pi,length=50)
angles <- matrix(0,length(ps),length(ps2))
for(i in 1:length(Y.series)){
  #angles[[i]] <- rep(0,length(ps2))
  for(j in 1:length(ps2)){
    tmp.out <- my.vector.sum.sp2(c(X),c(Y.series[[i]]),ps2[j])
    Z <- matrix(tmp.out$z,ncol=d)
    al.out <- my.althlooti(Z,X)
    tmp.q <- al.out$q
    angles[i,j] <- Re(tmp.q)
  }

}
```

```
matplot(ps2/pi,t(angles),type="l")
```




```
image(angles)
```



```
table(round(angles,15))
```

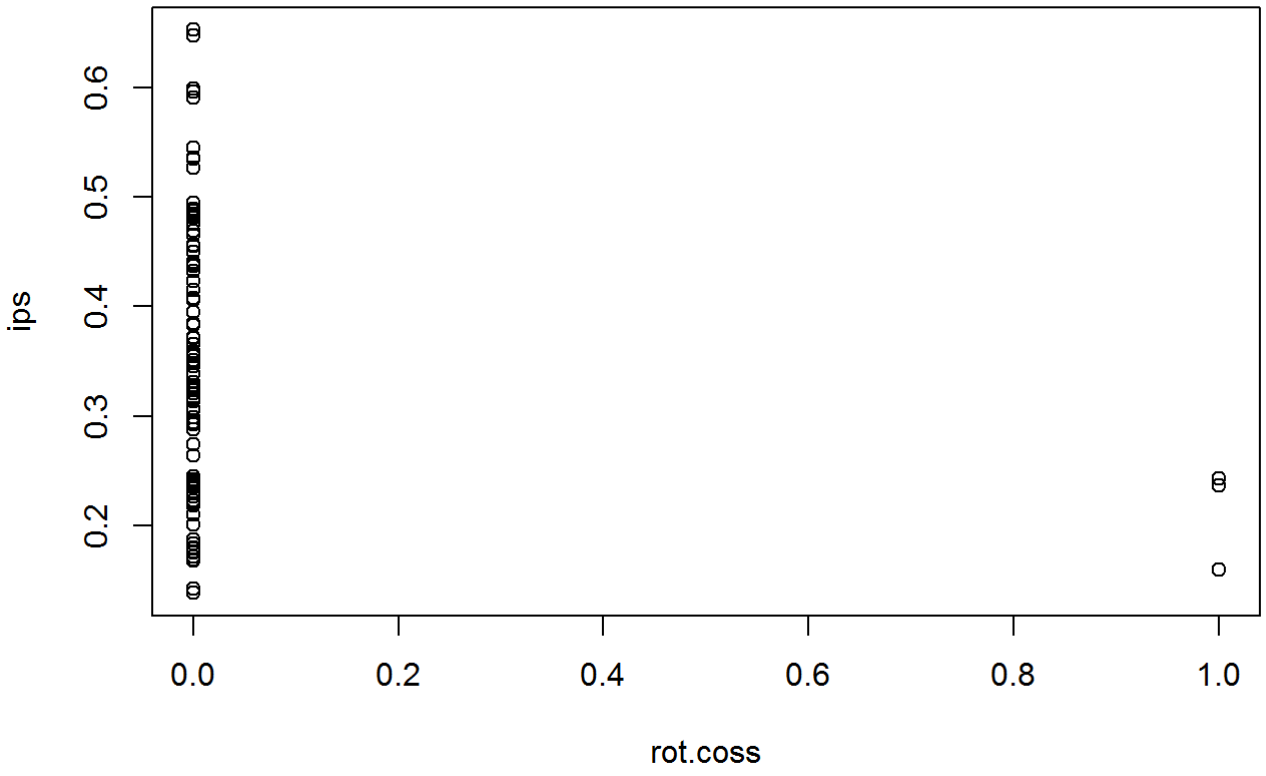
```
##
## -3.8e-14 -3.7e-14 -2.7e-14 -1.9e-14 -1.8e-14 -1.6e-14 -1.5e-14   -8e-15
##        1        2        1        1        1        1        2        1
##   -4e-15        0  1.4e-14  2.9e-14        1
##        1      347        1        1      140
```

対蹠点ペアにとって、回転同一な点集合

```
library(MCMCpack)
n.sites <- 100
k <- 6
d <- 3
X <- my.runit.matrix(k,d)
Ys <- list()
for(i in 1:n.sites){
  Ys[[i]] <- my.runit.matrix(k,d)
}
Y.rot.90s <- list()
Y.rot.90s.mat <- matrix(0,n.sites,k*d)
for(i in 1:n.sites){
  al.out <- my.althlooti(Ys[[i]],X)
  Y.rot <- al.out$RotX1
  tmp <- my.vector.sum.sp2(c(X),c(Y.rot),pi/2)
  Y.rot.90s[[i]] <- matrix(tmp$z,ncol=d)
  Y.rot.90s.mat[i,] <- tmp$z
}

n.trial <- 100
rot.coss <- rep(0,n.trial)
ips <- rep(0,n.trial)
for(i in 1:n.trial){
  #r <- rdirichlet(1,rep(0.1,n.sites+1))
  r <- rnorm(n.sites+1)*10
  r <- r/sum(r)
  tmp <- r[n.sites+1] * X
  for(j in 1:n.sites){
    tmp <- tmp + r[j] * Y.rot.90s[[j]]
  }
  tmp <- tmp/sqrt(sum(tmp^2))
  al.out <- my.althlooti(tmp,X)
  rot.coss[i] <- Re(al.out$q)
  ips[i] <- sum(X*al.out$RotX1)
}
table(round(rot.coss,15))
```

```
##
## -6.6e-14 -1.4e-14   -3e-15   -2e-15   -1e-15        0    1e-15    2e-15
##        1        1        2        8       11       42       23        7
##    3e-15        1
##        2        3
```

```
plot(rot.coss,ips)
```




```
Matrix::rankMatrix(Y.rot.90s.mat)
```

```
## [1] 14
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 2.220446e-14
```