正三角形メッシュに近づける

ryamada

2019年12月16日

問題設定の確認

おおよそ、均等な三角形でできた３次元に埋め込まれた閉多面体があるとする。

形をある程度、犠牲にしてでも、正三角形メッシュに変換したい。

正三角形メッシュグラフには、複数の３次元埋め込みが対応しうるが、そのうちの一つに変換できればよいとする。

非正三角形メッシュの埋め込み状態と似たような正三角形埋め込みが得られれば、それにこしたことはない。

正二十面体様の多面体。その複数の３次元埋め込み

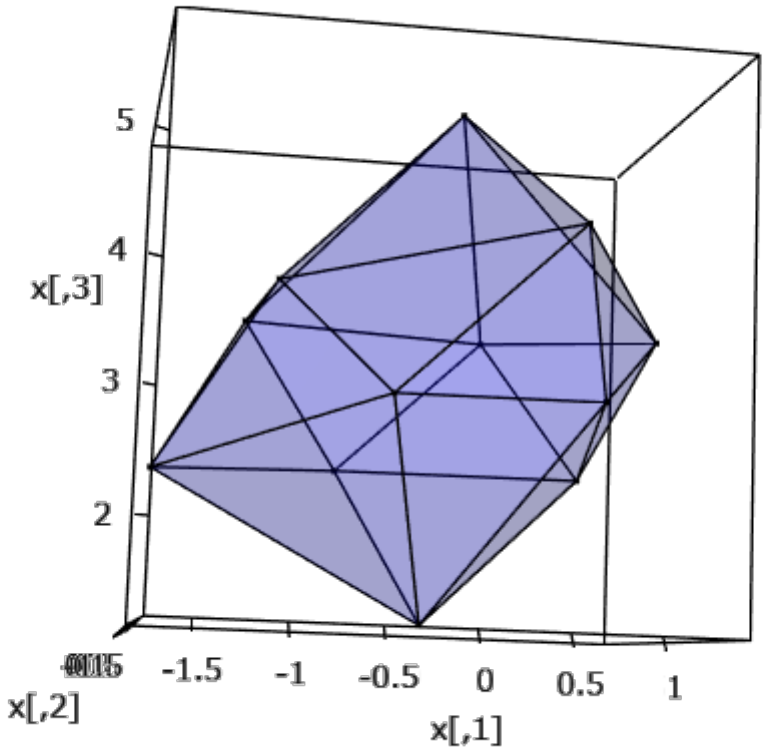
正二十面体と同じ構成の３画メッシュ情報を作成し、 ２通りの埋め込み(正二十面体に近いものと、１頂点が凹んでいるもの)座標を作成する。

```
icosa <- rbind(c(1,2,3),c(1,3,4),c(1,4,5),c(1,5,6),c(1,6,2),c(3,2,7),c(4,3,8),c(5,4,9),c(6,5,10),c(2,6,11),c(3,7,8),c(4,8,9),c(5,9,10),c(6,10,11),c(2,11,7),c(12,8,7),c(12,9,8),c(12,10,9),c(12,11,10),c(12,7,11))



theta1 <- (1:5)/5 * 2 * pi
theta2 <- theta1 + theta1[1]/2
# 正二十面体に近い埋め込み座標
x1 <- rbind(c(0,0,4),cbind(cos(theta1),sin(theta1),rep(3,5)),cbind(cos(theta2),sin(theta2),rep(2,5)),c(0,0,1)) * sqrt(2)

x1 <- x1 + rnorm(length(x1),0,0.2)

# 凹みのある埋め込み座標
x2 <- rbind(c(0,0,2.5),cbind(cos(theta1),sin(theta1),rep(3,5)),cbind(cos(theta2),sin(theta2),rep(2,5)),c(0,0,1)) * sqrt(2)
# 乱雑項を加えて、形をいびつにする
x2 <- x2 + rnorm(length(x2),0,0.2)
```

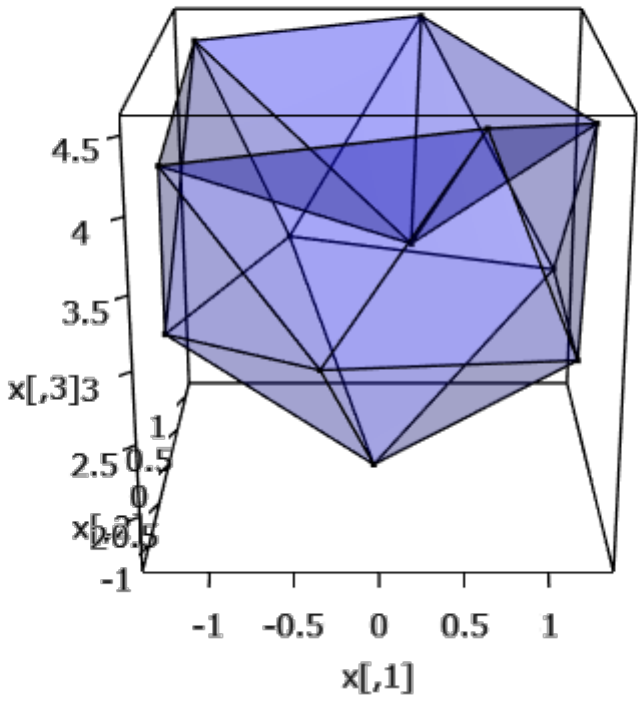
三角メッシュを描図する関数を作っておく。

```
my.plot.polygon <- function(x,el,f,color=4){
  plot3d(x)
  segments3d(x[t(el),])
  triangles3d(x[t(f),],alpha=0.2,color=color)
}
```

```
f <- icosa

el <- rbind(f[,1:2],f[,2:3],f[,c(3,1)])

# 正二十面体に近い埋め込み
my.plot.polygon(x1,el,f)
```




```
# 凹みのある埋め込み
my.plot.polygon(x2,el,f)
```



関数を作る

すべてのエッジの自身との内積は1である。

三角形を構成するエッジ２本同士の内積は± 0.5である。

エッジは２つの三角形の構成要素になっているので、上記を満足するとき、三角形は全体として閉多面体となる。

したがって、エッジの内積行列(対称)に上記の制約を要求しつつ、その内積行列が、ランク3となるようなものが得られれば、それが求める、エッジの３次元ベクトルである。

すべてのエッジの３次元ベクトルが得られれば、三角形メッシュのすべての頂点は連結なので、ある頂点を起点として、全頂点の相対座標は算出できる。

以下では、三角形メッシュの頂点座標を計算するにあたり、別の方法を採用した。

その方法は以下の通りである。

頂点数次元空間の頂点数-正単体が、３次元に埋め込まれたとみなす。

頂点数次元空間に対応するエッジベクトルが得られるので、これを、上で求めた３次元空間ベクトルに移す線形変換を算出し、その線形変換によって頂点数次元空間の正単体の頂点座標を３次元空間に写像する。

内積行列が条件を満たすように繰り返し処理をするにあたり、制約のある内積値に徐々に近づける方法もあると思われるが、以下では、いきなり、制約そのものを課して、randomized svd分解により近似３次元ベクトルを取り出すことを繰り返している。

この部分には工夫の余地があるように思われる。

```
# 順逆両方向のエッジの始点・終点を持つ２列行列elから、エッジペア情報を取り出す
my.Epair <- function(el){
    s.el <- apply(el,1,sort)
    mv <- max(el)+1
    V <- s.el[1,] * mv + s.el[2,]
    tmp <- outer(V,V,"-")
    diag(tmp) <- 1
    pairs <- which(tmp==0,arr.ind=TRUE)
    return(pairs)
}
# 向きつけられた三角形の頂点座標行列fから
# エッジに関する各種情報を取り出す
my.Einfo <- function(f){
    el <- rbind(f[,1:2],f[,2:3],f[,c(3,1)])
    E.V <- matrix(0,length(el[,1]),max(f))
    for(i in 1:length(el[,1])){
        E.V[i,el[i,1]] <- -1
        E.V[i,el[i,2]] <- 1
    }

    ne <- length(el[,1])
    edge.trio <- matrix(1:length(el[,1]),ncol=3)

    e.pair <- my.Epair(el)
    # 順方向か逆方向か
    e.dir <- rep(1,ne)
    e.dir[which(el[,1] < el[,2])] <- -1
    # エッジのシークエンスID
    e.id <- rep(0,ne)
    e.id[which(el[,1] > el[,2])] <- 1:(ne/2)
    for(i in 1:ne){
        if(e.id[i]==0){
            e.id[i] <- e.id[e.pair[which(e.pair[,1]==i),2]]
        }
    }
    # eid.dir
    # エッジID(第一カラム)とその順逆情報(第二カラム)
    eid.dir <- cbind(e.id,e.dir)

    # eidの始点・終点情報
    eid.stend <- matrix(0,ne/2,2)
    for(i in 1:length(eid.stend[,1])){
        tmp <- which(eid.dir[,1] * eid.dir[,2] == i)
        eid.stend[i,] <- el[tmp,]
    }
    eid.E.V <- matrix(0,length(eid.stend[,1]),max(f))
    for(i in 1:length(eid.E.V[,1])){
        eid.E.V[i,eid.stend[i,1]] <- -1
        eid.E.V[i,eid.stend[i,2]] <- 1
    }
    regM <- diag(rep(1,ne/2))
    regM.v <- diag(rep(1,ne/2))
    for(i in 1:length(edge.trio[,1])){
        tmp <- edge.trio[i,]
        tmp <- c(tmp,tmp[1])
        for(j in 1:3){
            e1 <- tmp[j]
```

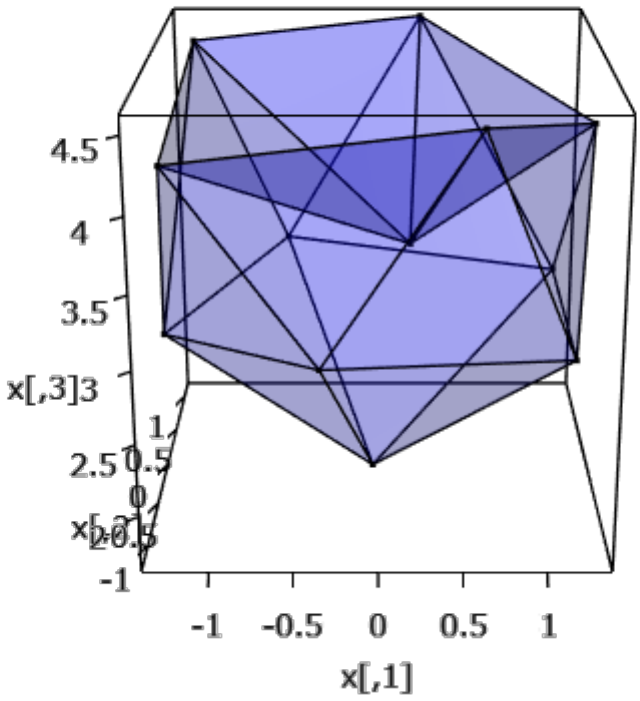
```
            e2 <- tmp[j+1]

            eid1 <- eid.dir[e1,1]
            eid2 <- eid.dir[e2,1]

            edir1 <- eid.dir[e1,2]
            edir2 <- eid.dir[e2,2]

            regM[eid1,eid2] <- regM[eid2,eid1] <- 1
            regM.v[eid1,eid2] <- regM.v[eid2,eid1] <- -0.5
            if(edir1 * edir2 == -1){
                regM.v[eid1,eid2] <- regM.v[eid2,eid1] <- 0.5
            }
        }
    }
    # 三角形を１周するエッジ３本を、その向きに気を付けて |F| * |E| 行列(成分は{0,-1,1})で作る
    FEmat <- matrix(0,length(f[,1]),ne/2)
    for(i in 1:length(f[,1])){
      tmp.e <- edge.trio[i,]
      tmp.e.id <- eid.dir[tmp.e,1]
      tmp.e.dir <- eid.dir[tmp.e,2]
      FEmat[i,tmp.e.id] <- tmp.e.dir
    }
    # eid は3k本のエッジ、e2は6k本のエッジ
    return(list(eid.stend = eid.stend,eid.EV = eid.E.V,regM = regM, regM.v = regM.v, e2.stend
 = el,e2.EV = E.V,e.pair = e.pair,e2.iddir = eid.dir,e2.trio = edge.trio,FEmat=FEmat,ne=ne))
}

#####
# 引数
#####
# 頂点座標行列X3
# 三角形頂点トリオ情報 f
# 分解ランク k=3
# 収束閾値 eps : 内積が1,-0.5,+0.5に収束したと判定する誤差
# 最大処理回数

#####
# 出力
#####
# edges: ３次元エッジベクトルの最終推定結果
# IPmat: エッジ内積行列の最終推定結果
# X3.est: ３次元頂点座標の最終推定結果
# n.iter: 計算繰り返し数
# X3.ori.st: 与えた頂点座標情報を適当な大きさ・位置に標準化した座標
# scaler: 与えた頂点座標の拡縮割合
# e.info$eid.stend: エッジの始点終点頂点情報
# e.info$eid.EV: エッジを始点・終点座標から算出するための行列
#                頂点座標行列Xに対して、x %*% e.info$eid.EV により
#                エッジベクトルが算出される
# e.info$regM: エッジ内積行列のうち、1,-0.5,+0.5の制約のあるセルに1を立てた行列
# e.info$regM.v: 制約のあるセルに制約値(1,-0.5,+0.5)の値を持った行列
my.rsvd.triCycle <- function(X3,f,k=3,eps=10^(-10),maxiter=10000,n.pinv = 100){
    e.info <- my.Einfo(f)
    ne <- e.info$ne
    regM <- e.info$regM
    regM.v <- e.info$regM.v
    add <- which(regM==1)
```

```
ipv <- regM.v[add]
FEmat <- e.info$FEmat
# 三角形が閉じる制約を表したFEmat行列と

tmpn <- length(FEmat[,1])
tmpm <- length(FEmat[1,])

# 三角形・エッジ関連行列を(x,y,z)の３要素分に合わせて３倍する
FEtriple <- matrix(0,tmpn*3,tmpm*3)

FEtriple[1:tmpn,1:tmpm] <- FEmat
FEtriple[(1:tmpn)+tmpn,(1:tmpm)+tmpm] <-FEmat
FEtriple[(1:tmpn)+tmpn*2,(1:tmpm)+tmpm*2] <-FEmat

# 与えられたエッジ座標を維持することを要請する正方行列とを連結する
FEmatPlus <- rbind(FEtriple,diag(rep(1,ne/2*3)))

e.vec <- matrix(0,length(e.info$eid.stend[,1]),3)
for(i in 1:length(e.vec[,1])){
    e.vec[i,] <- X3[e.info$eid.stend[i,2],] - X3[e.info$eid.stend[i,1],]
}
M <- e.vec %*% t(e.vec)
# 辺の長さの平均を1にそろえる
scaler <- mean(diag(M))
X3. <- X3/sqrt(scaler)
X3. <- t(t(X3.) - apply(X3.,2,mean))
M. <- M/scaler

K <- M.
n <- length(M.[,1])
ret <- list()

tmp.rsvd <- rsvd(K,k=k)
edges <- diag(sqrt(tmp.rsvd$d)) %*% t(tmp.rsvd$u)
K <- t(edges) %*% edges
ret[[1]] <- K
retX <- list()
retX[[1]] <- edges
iter.cnt <- 1
while(iter.cnt < maxiter){
    # 内積制約
  check.ip <- max(abs(K * regM - regM.v))
  #check.rowsum <- max(abs(apply(K,1,sum)))
  # cycle制約

  check.cycle <- max(abs((FEmatPlus %*% c(edges))[1:length(f)]))
  if(check.ip < eps & check.cycle < eps){
    return(list(edges = retX, IPmat = ret))
    break
  }
  # 内積制約の調整
  K[add] <- ipv
  # rsvd
  tmp.rsvd <- rsvd(K,k=k)
  K <- tmp.rsvd$v %*% diag(tmp.rsvd$d) %*% t(tmp.rsvd$u)
    ret[[iter.cnt+1]] <- K
  edges <- diag(sqrt(tmp.rsvd$d)) %*% t(tmp.rsvd$u)
  # サイクル制約の調整
```

```
    edge.vec <- c(t(edges))

    b.vec <- c(rep(0,length(FEtriple[,1])), edge.vec)

    # FEmatPlus %*% edge.vec ~ b.vec にフィットする edge.vec' を推定する

    est.edge.vec <- solve(t(FEmatPlus)%*%FEmatPlus)%*%t(FEmatPlus) %*% b.vec
    est.edges <- matrix(est.edge.vec,byrow=TRUE,nrow=3)
    retX[[iter.cnt+1]] <- est.edges
      K <- t(est.edges) %*% est.edges
      iter.cnt <- iter.cnt + 1
  }
  # 内積制約の調整
    K[add] <- ipv
    # rsvd
    tmp.rsvd <- rsvd(K,k=k)
    K <- tmp.rsvd$v %*% diag(tmp.rsvd$d) %*% t(tmp.rsvd$u)
      ret[[iter.cnt+1]] <- K
    edges <- diag(sqrt(tmp.rsvd$d)) %*% t(tmp.rsvd$u)
    ret[[iter.cnt+1]] <- K
    retX[[iter.cnt+1]] <- edges
  #edges <- diag(sqrt(tmp$d)) %*% t(tmp$u)
  final.K <- ret[[iter.cnt+1]]
  final.edges <- retX[[iter.cnt+1]]

  nv <- max(f)
  Xn <- diag(rep(1,nv))
  En <- t(e.info$eid.EV)

  final.edges. <- matrix(0,ne/2,ne/2)
  final.edges.[1:3,] <- final.edges
  R <- final.edges. %*% Ginv(En)

  X3.est <- R %*% Xn
  X3.est. <- t(X3.est[1:k,])
  return(list(edges = final.edges,IPmat=final.K,X3.est = X3.est., X3.estmat = X3.est,R=R,edg
es.hx = retX, IPmat.hx = ret,e.info=e.info,niter=iter.cnt,X3.ori.st=X3.,f=f,scaler=scaler))
  #return(list(edges = final.edges,IPmat=final.K,X3.est = X3.est.mean,R=R,edges.hx = retX, IP
mat.hx = ret,e.info=e.info,niter=iter.cnt,X3.ori.st=X3.,scaler=scaler))
}
```

関数を使う

```
out1 <- my.rsvd.triCycle(x1,f)
out2 <- my.rsvd.triCycle(x2,f)
el <- out1$e.info$e2.stend
```

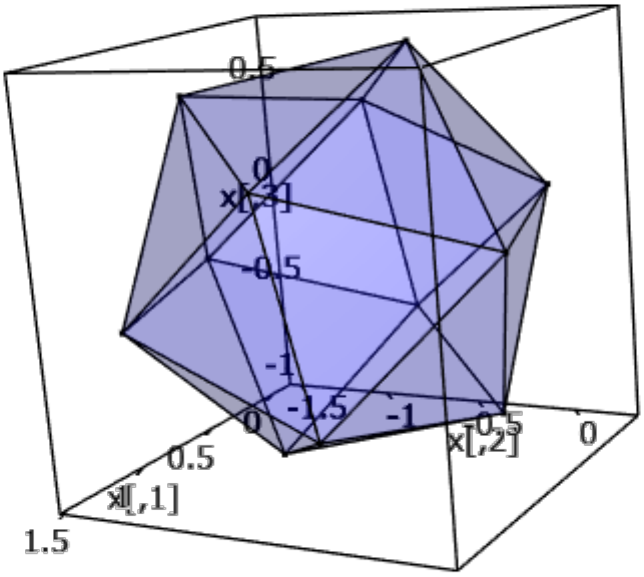


見てくれの確認

推定閉多面体の様子を図示し、正三角形でおおわれているように見えることを確認する。

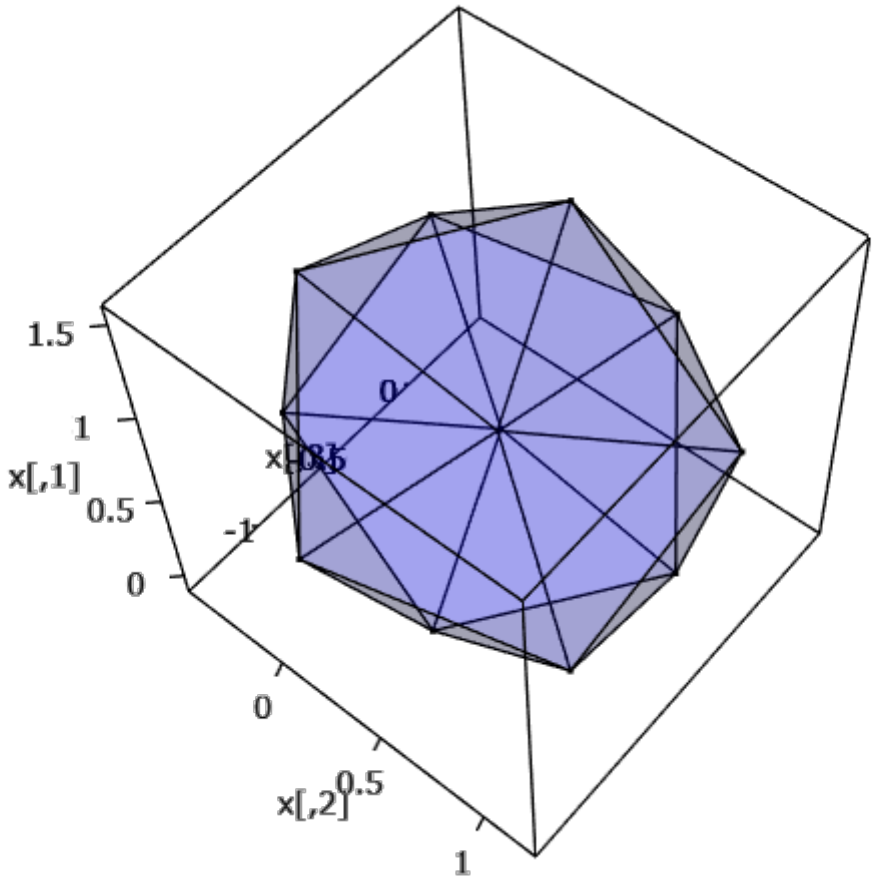
いわゆる正二十面体になっている。

```
my.plot.polygon(out1$X3.est,el,f)
```



凹みのある立体になっているが、多面体の構成面は正三角形に見える。

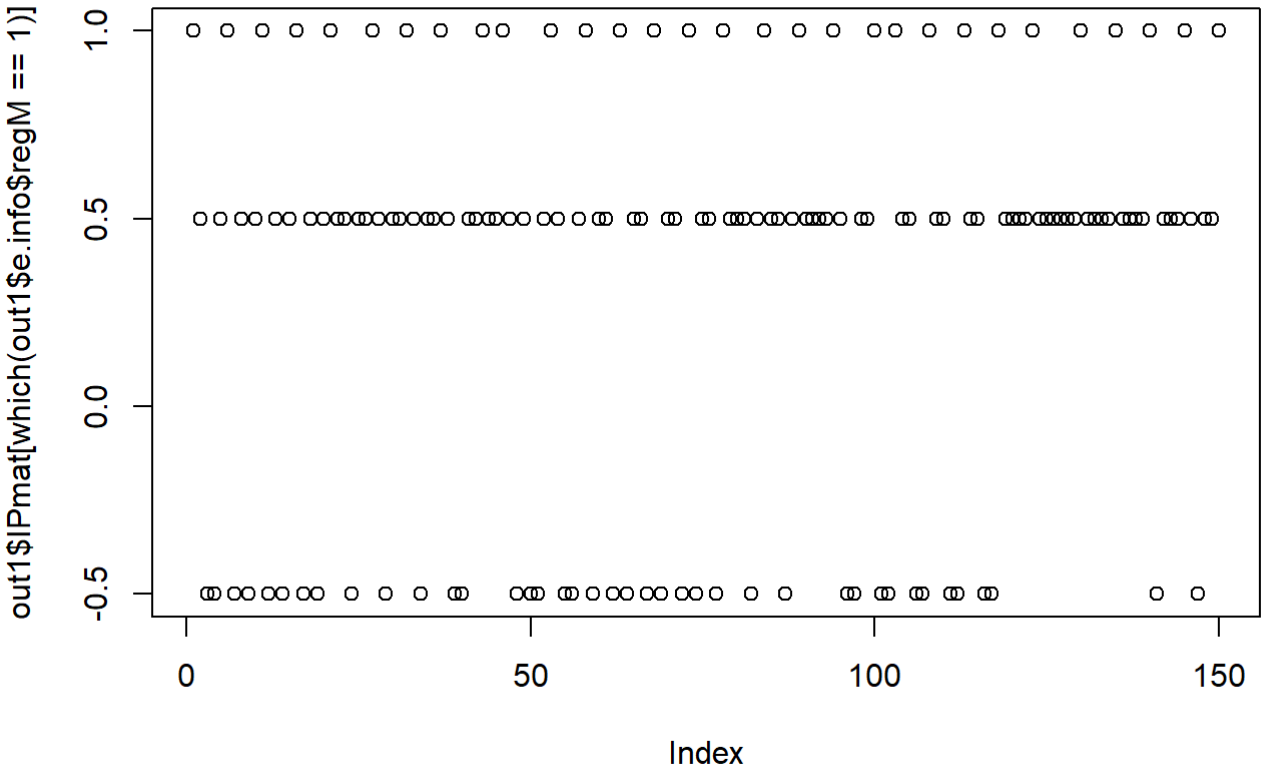
```
my.plot.polygon(out2$X3.est,el,f)
```



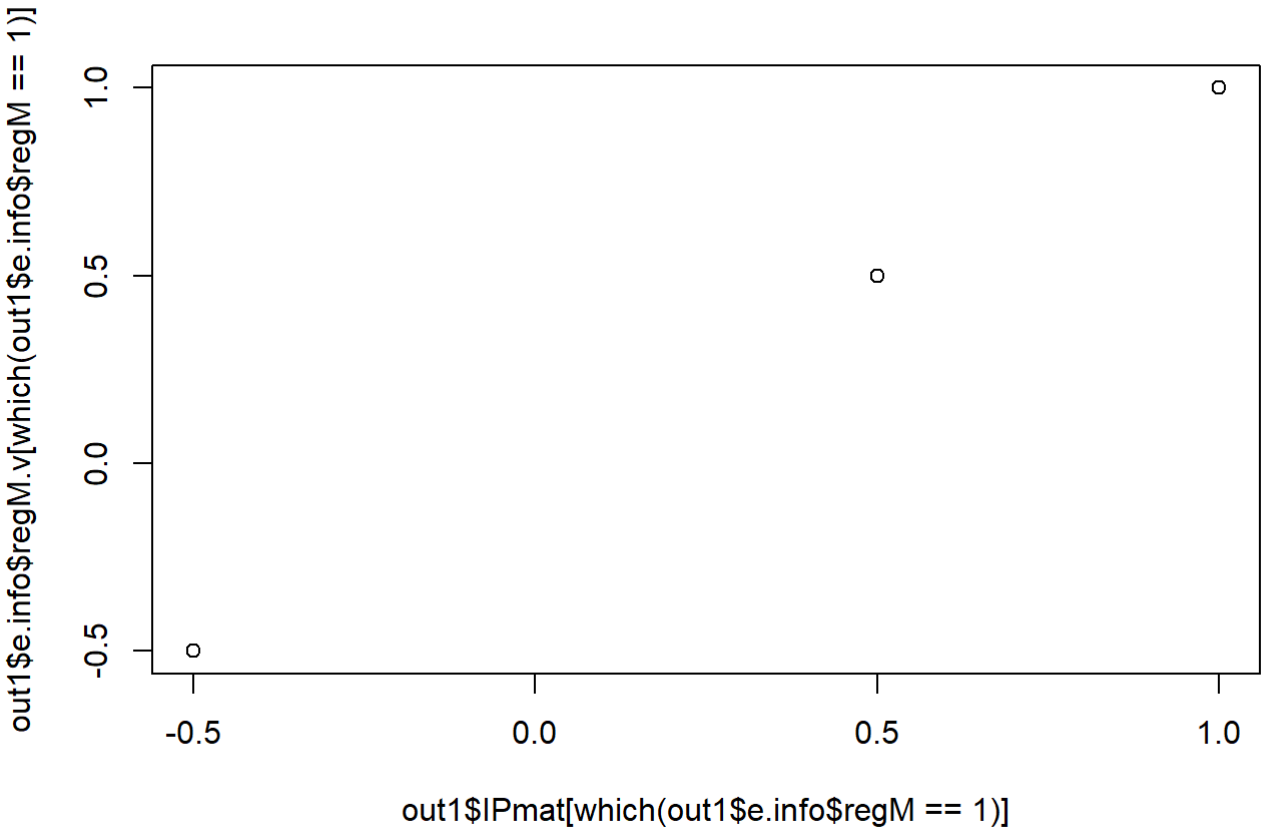
内積制約の確認

制御したい内積(辺の長さなら１、三角形構成２辺なら± 0.5) がほぼ達成できていることを確認

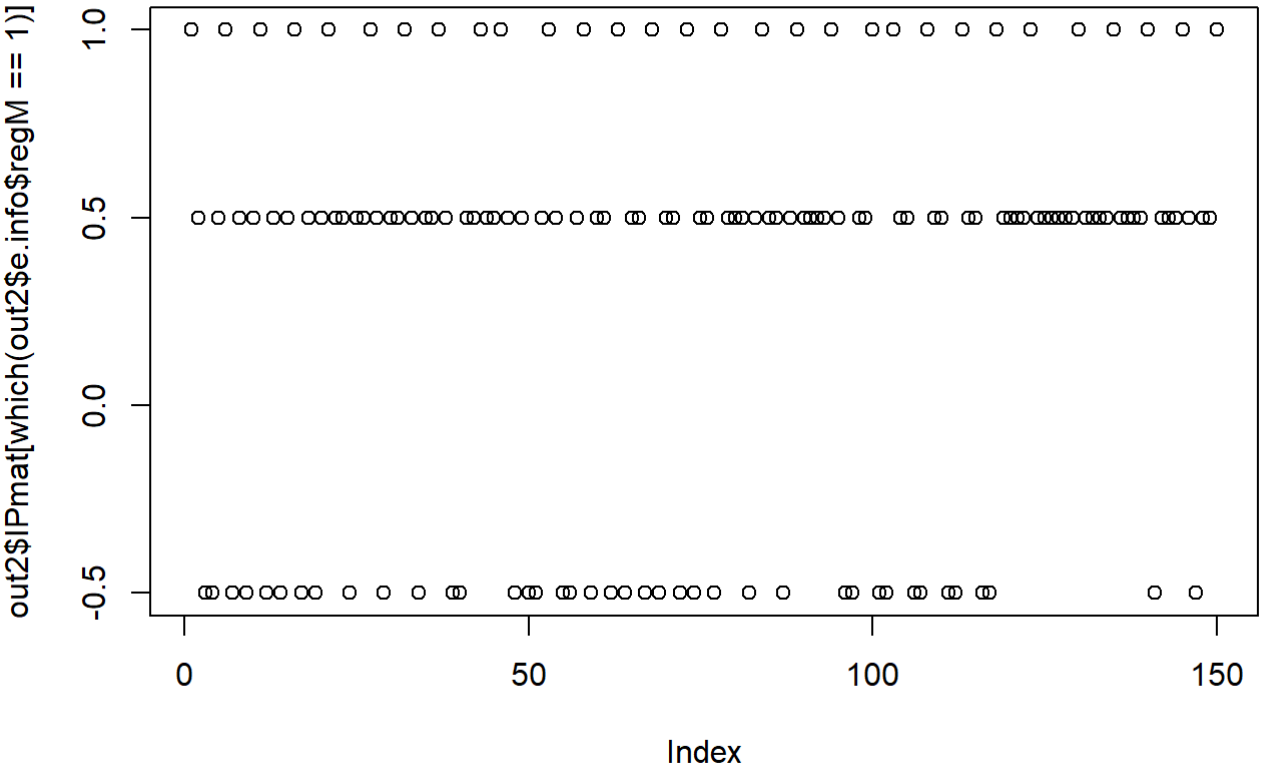
```
# すべての制約内積が1, -0.5, +0.5のいずれかであることがわかる
plot(out1$IPmat[which(out1$e.info$regM==1)])
```



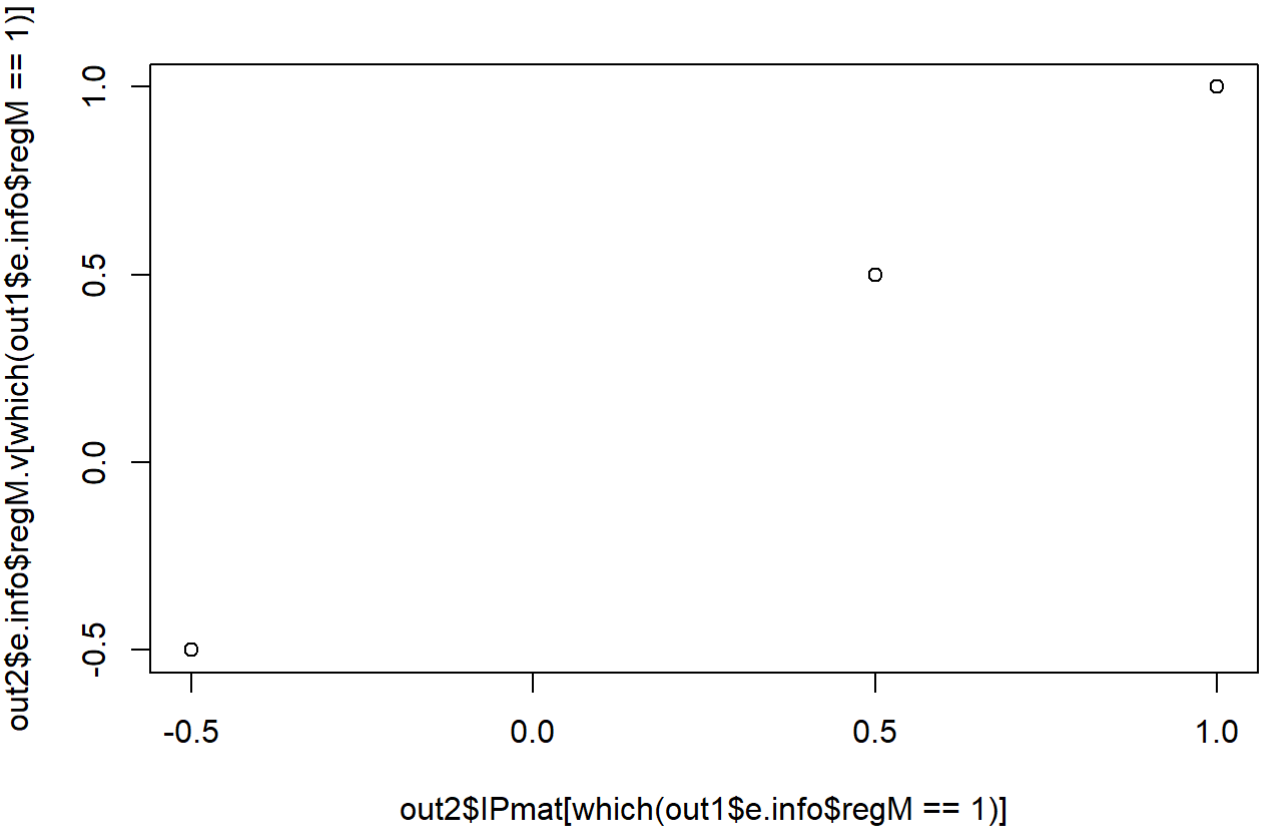

```
# また、1であるべきもの、-0.5,+0.5であるべきものが、その通り、1,-0.5,+0.5であることもわかる
plot(out1$IPmat[which(out1$e.info$regM==1)],out1$e.info$regM.v[which(out1$e.info$regM==1)])
```



```
plot(out2$IPmat[which(out2$e.info$regM==1)])
```




```
plot(out2$IPmat[which(out1$e.info$regM==1)],out2$e.info$regM.v[which(out1$e.info$regM==1)])
```



三角形が閉じていることの確認

それぞれの三角形の周囲のベクトルを向きを考慮して加算して得られる値はそのx,y,z座標のすべてを通じて、ほぼゼロになっている。

```
range(out1$e.info$FEmat %*% t(out1$edges))
```

```
## [1] -6.661338e-16  6.106227e-16
```

```
range(out2$e.info$FEmat %*% t(out2$edges))
```

```
## [1] -7.216450e-16  8.326673e-16
```

正三角形メッシュのグラフスペクトル考察

正三角形メッシュの埋め込み近似ができた。

オブジェクトの３次元空間配置は、頂点の３次元座標を確定することによって決まるが、 オブジェクトの回転不変的性質は、全エッジ間の相対的配置(内積)によって、ある程度(かなりの程度)、決まるので、エッジ内積行列の性質を考察したい。

正二十面体と凹みのあるバリアントとの、それぞれのエッジ内積行列の固有値を調べよう。

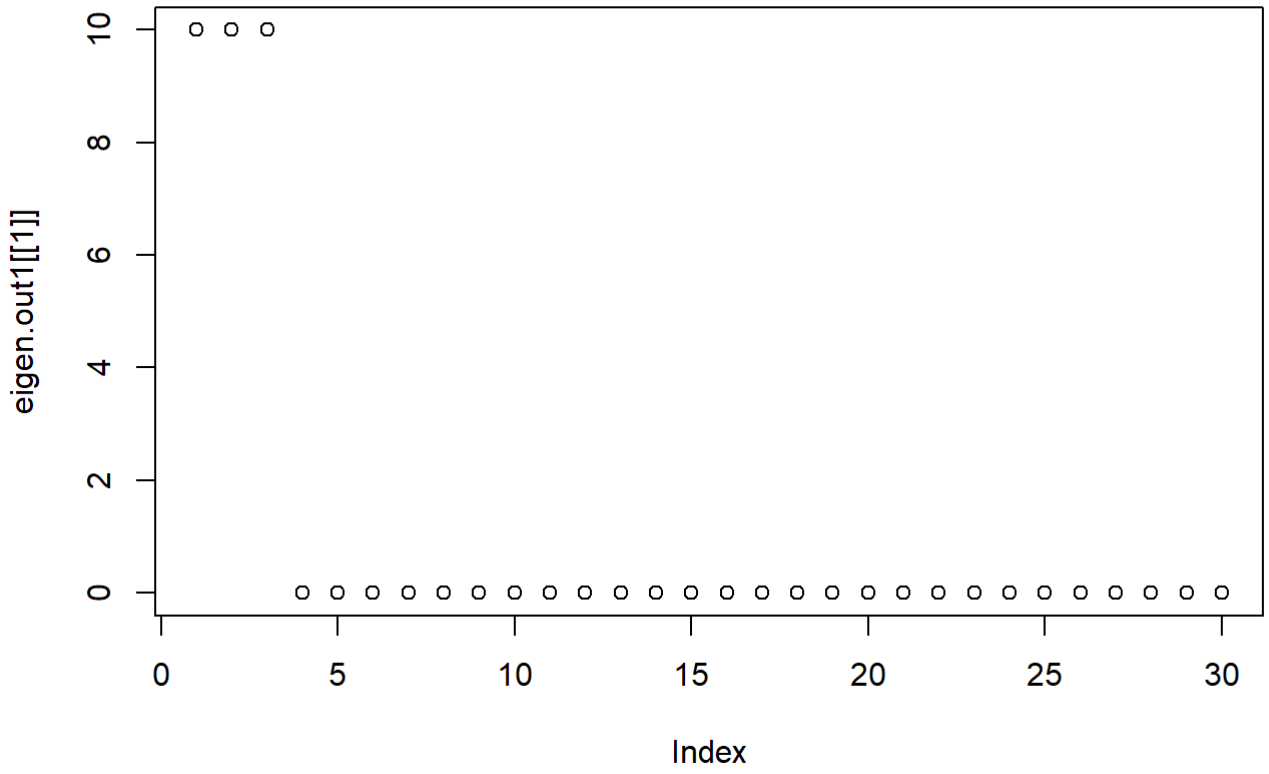
```
IPmat1 <- out1$IPmat
IPmat2 <- out2$IPmat
```

```
eigen.out1 <- eigen(IPmat1)
eigen.out2 <- eigen(IPmat2)
```

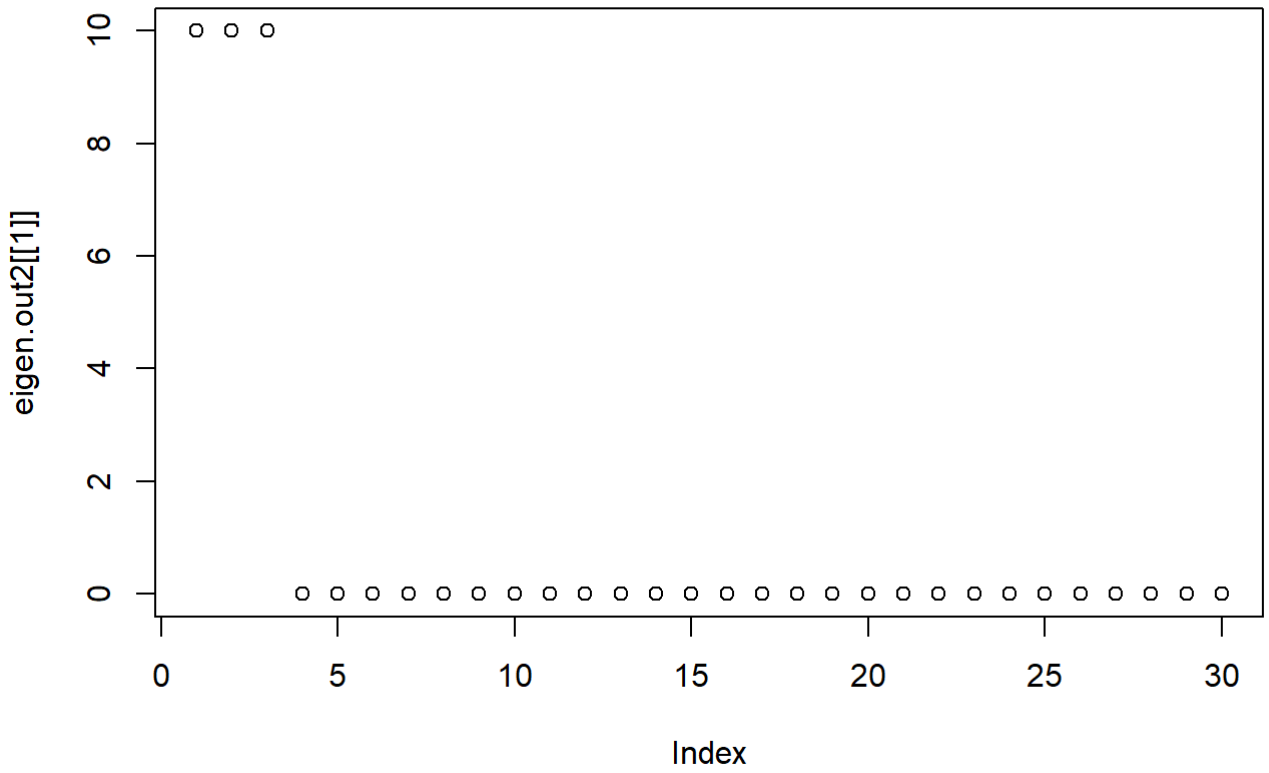
非ゼロ固有値の個数は3個である。

これは、エッジ数ｘ３の行列からこの内積行列ができていることからもわかるように、この内積行列には３次元に収まるという性質があり、それがランク３として表れている。

```
plot(eigen.out1[[1]])
```




```
plot(eigen.out2[[1]])
```



また、３つの非ゼロ固有値の和は30になっている。

この内積行列の対角成分は、各辺の二乗ノルムであり、すべて１である。 辺の数が30なので、対角成分の和～トレースは30である。

固有値の和はトレースなので、この値になることの説明がつく。

```
sum(eigen.out1[[1]])
```

```
## [1] 30
```

```
sum(eigen.out2[[1]])
```

```
## [1] 30
```

最後に、非ゼロ固有値は、すべて等しく、10,10,10となっている。 正二十面体であっても、凹みのあるバリアントであっても、それは変わらないことがわかる。

逆に言うと、平面グラフのレベルでこの２つの閉多面体は同じであり、埋め込みが違おうとも、内積行列の固有値スペクトルには違いがないのだということなのだろう。

さらに言うと、この固有値スペクトルでは、形の凹凸情報は取れないこともわかった。

最後に。非ゼロ固有値がすべて等しかったことは、おそらく、この閉多面体が対称的平面グラフに対応していることを意味するものと思われる。

正三角形メッシュが作る幾何空間

任意の正三角形メッシュグラフの埋め込みが与えるエッジ内積行列の非ゼロ固有値の個数は３で、その和はエッジ数に一致することが分かった。

したがって、エッジ本数が$|E|$の正三角形メッシュのエッジ長を、$\frac{1}{|E|}$となるようにサイズ調整をすれば、任意の正三角形メッシュの非ゼロ固有値は3個で、その和が1であるような座標に対応付けられる。

これは、$X_1 + X_2 + X_3 = 1$という平面が正三角形メッシュのスペクトル空間であることがわかる($X_1 \geq X_2 \geq X_3$という制約を加えれば、この平面の部分になることもわかる)。

箙変異による正三角形メッシュ埋め込みの変形

三角形メッシュの箙変異は、

隣接する２つの正三角形をとり、その共有エッジを取り去り、向きを90度変えて置きなおす、という変化である。

以下の関数は、三角形メッシュの向き付き三角形頂点情報行列を入力として取り、 １つの三角形をランダムに選び、その選ばれた三角形の３辺のうちの１つをランダムに選び、そのエッジの向きを変えることで得られる、三角形メッシュの頂点情報行列を返すものである。

```
my.random.tri.mut <- function(f){
    nf <- length(f[,1])
    f1 <- sample(1:nf,1)
    e1 <- sample(1:3,1)
    vs <- c(f[f1,],f[f1,1])[c(e1,e1+1)]

    f12 <- which(apply(t(f)-vs[1],2,prod)==0 & apply(t(f)-vs[2],2,prod)==0)

    f2 <- f12[which(f12 != f1)]

    e2 <- which(f[f2,] == vs[1])

    f1. <- c(f[f1,],f[f1,])
    f2. <- c(f[f2,],f[f2,])

    newf1 <- c(f1.[e1],f2.[e2+1],f1.[e1+2])
    newf2 <- c(f1.[e1+1],f1.[e1+2],f2.[e2+1])

    new.f <- f
    new.f[f1,] <- newf1
    new.f[f2,] <- newf2

    return(new.f)
}
```

この関数を用いて、メッシュ構造を変え、それに対する３次元埋め込み座標を算出することとする。

なお、算出開始時点の頂点座標は、箙変異をする前の座標そのものとする。

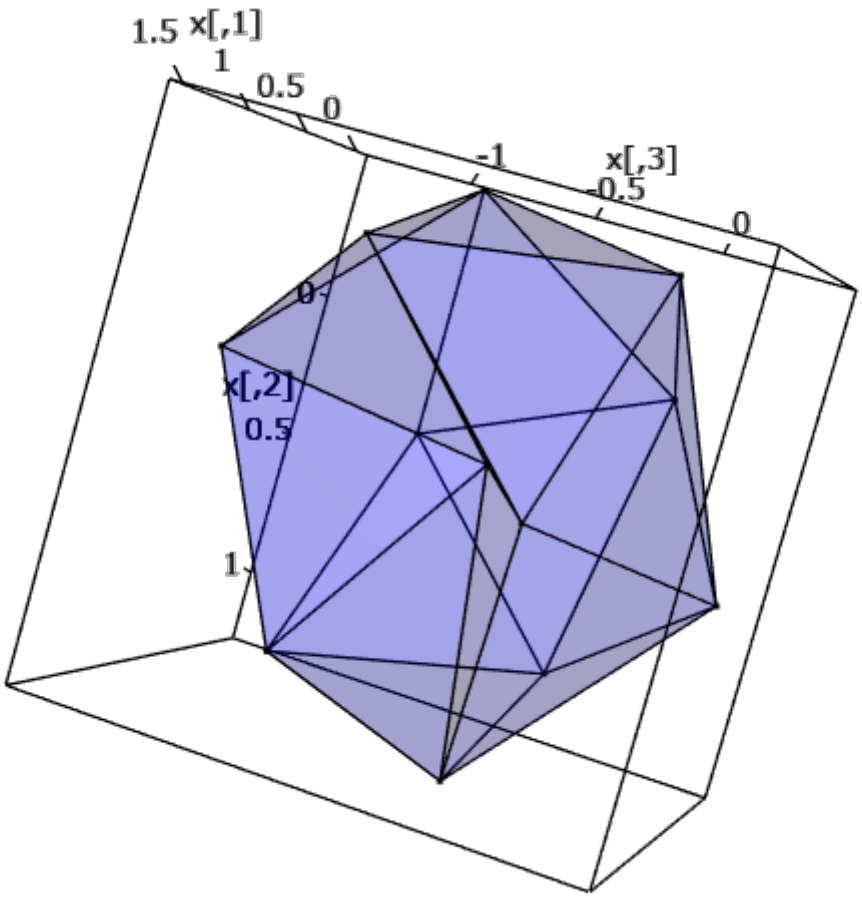
```
# 正二十面体
f <- rbind(c(1,2,3),c(1,3,4),c(1,4,5),c(1,5,6),c(1,6,2),c(3,2,7),c(4,3,8),c(5,4,9),c(6,5,10),c(2,6,11),c(3,7,8),c(4,8,9),c(5,9,10),c(6,10,11),c(2,11,7),c(12,8,7),c(12,9,8),c(12,10,9),c(12,11,10),c(12,7,11))

# 正二十面体の埋め込み座標は
x.ori <- out1$X3.est

# １回の腹変異でできた、三角メッシュ情報
quiver1 <- my.random.tri.mut(f)
```

```
out.q1 <- my.rsvd.triCycle(x.ori,quiver1)
```

埋め込み推定結果をプロットする関数。

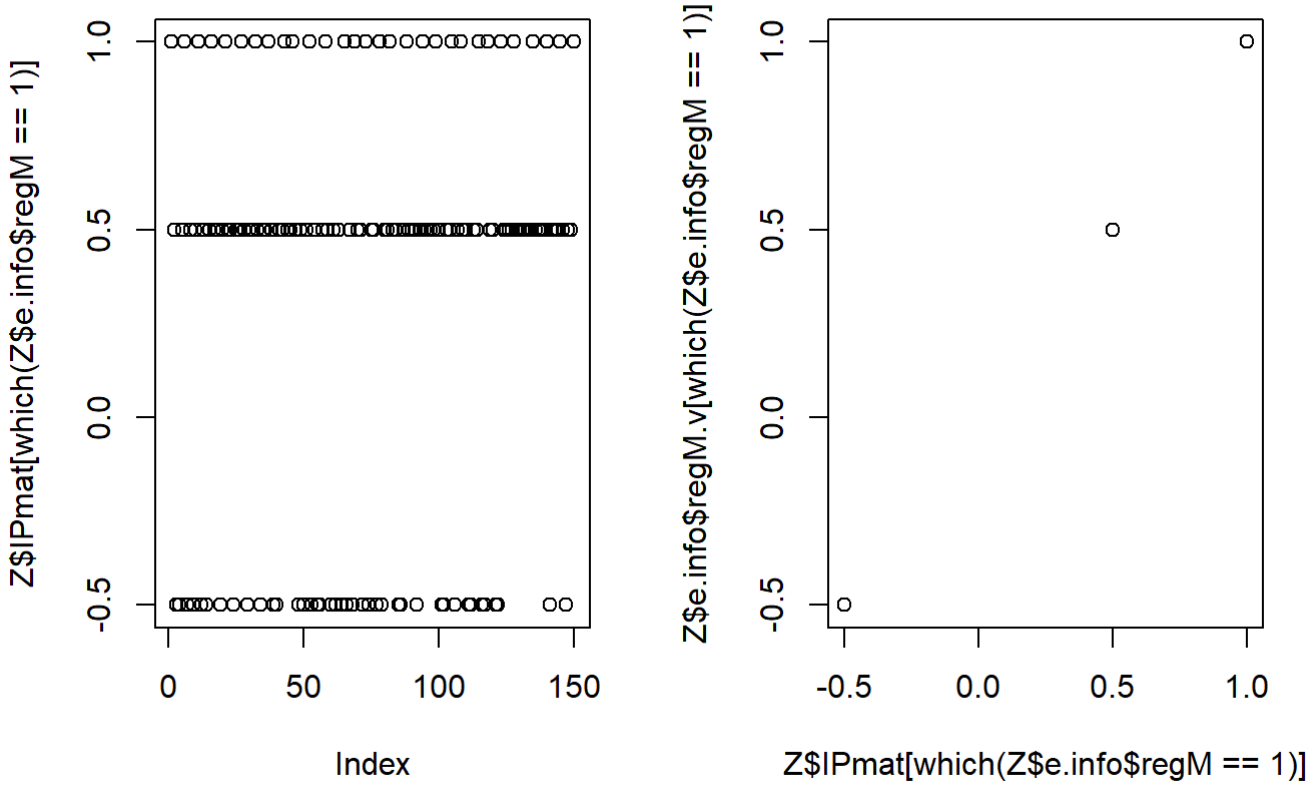
埋め込み３次元図を描くとともに、内積の収束、三角形が閉じる収束の情報を表示する。

```
# X : my.rsvd.triCycle()の出力
my.equilateral.plot <- function(Z){
  my.plot.polygon(Z$X3.est,Z$e.info$e2.stend,Z$f)
  par(mfcol=c(1,2))
  print("inner product conditions")
  plot(Z$IPmat[which(Z$e.info$regM==1)])
  plot(Z$IPmat[which(Z$e.info$regM==1)],Z$e.info$regM.v[which(Z$e.info$regM==1)])
  par(mfcol=c(1,1))
  print("triangle-closing conditions")
  print(range(Z$e.info$FEmat %*% t(Z$edges)))
}
```

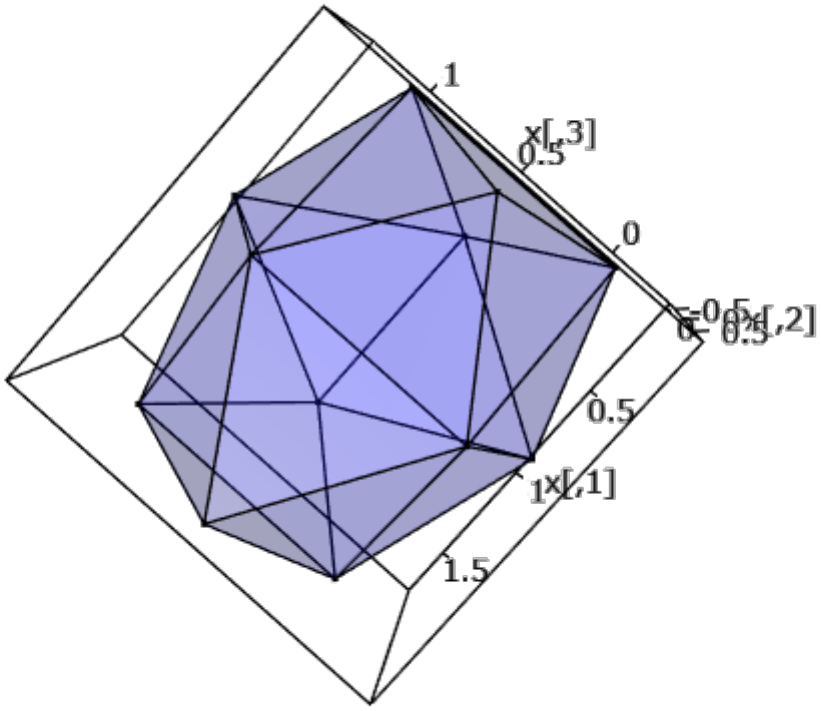


```
my.equilateral.plot(out.q1)
```

```
## [1] "inner product conditions"
```

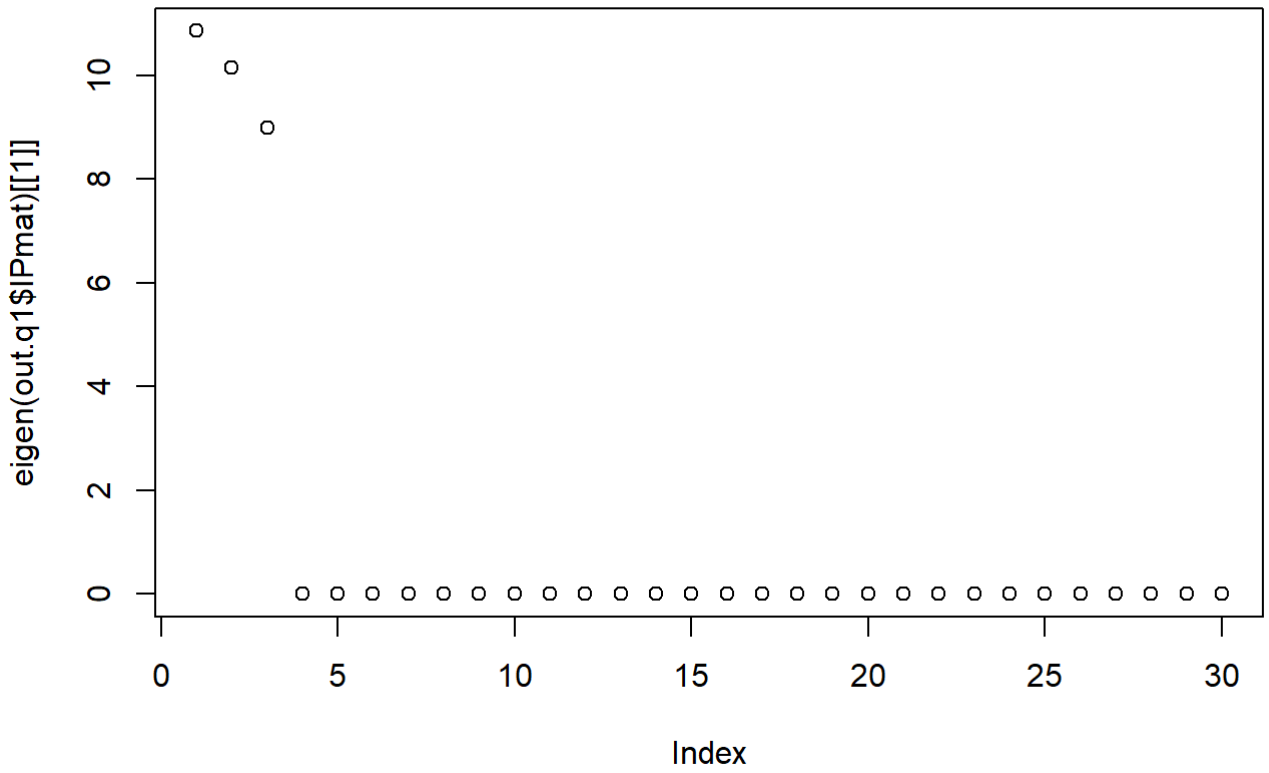



```
## [1] "triangle-closing conditions"
## [1] -7.771561e-16  5.551115e-16
```



エッジ内積行列の固有値はどうなっているか？

```
plot(eigen(out.q1$IPmat)[[1]])
```



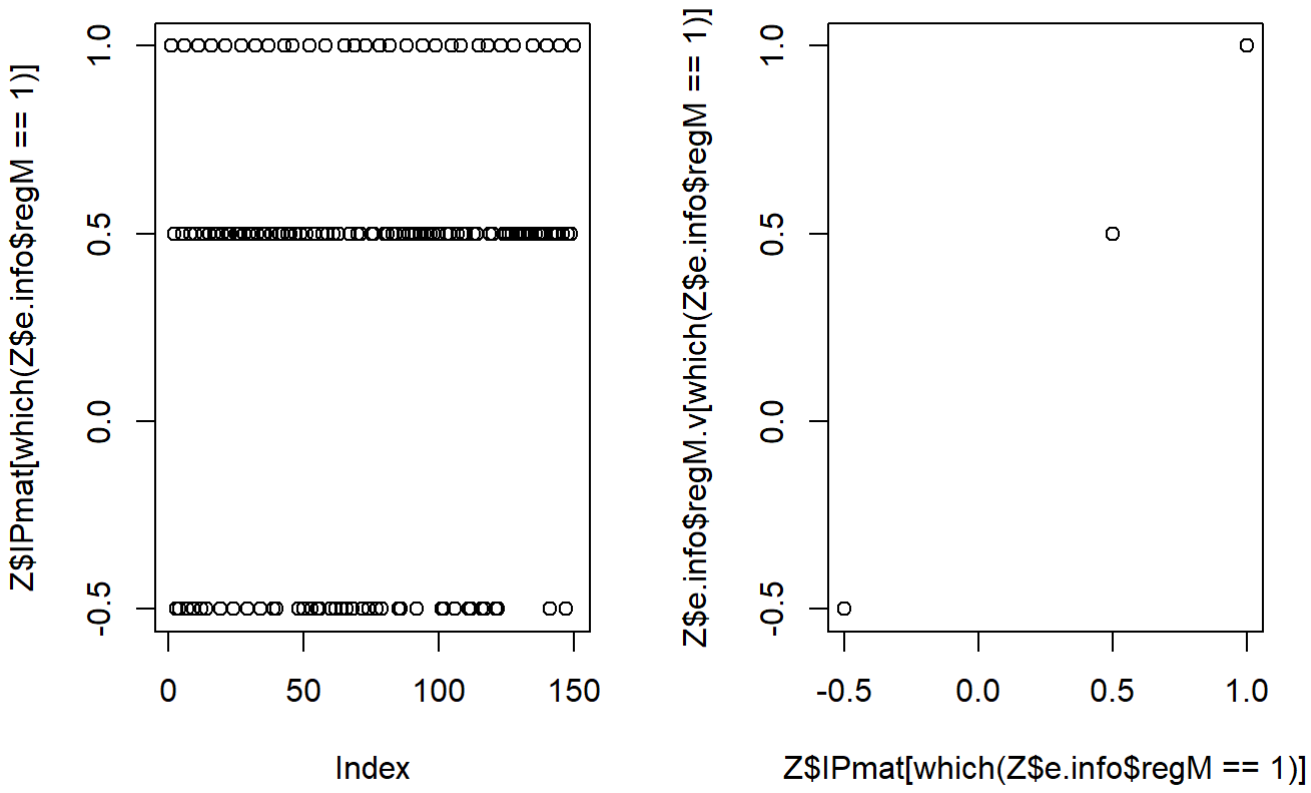

```
sum(eigen(out.q1$IPmat)[[1]])
```

```
## [1] 30
```

さらに変異させてみる。

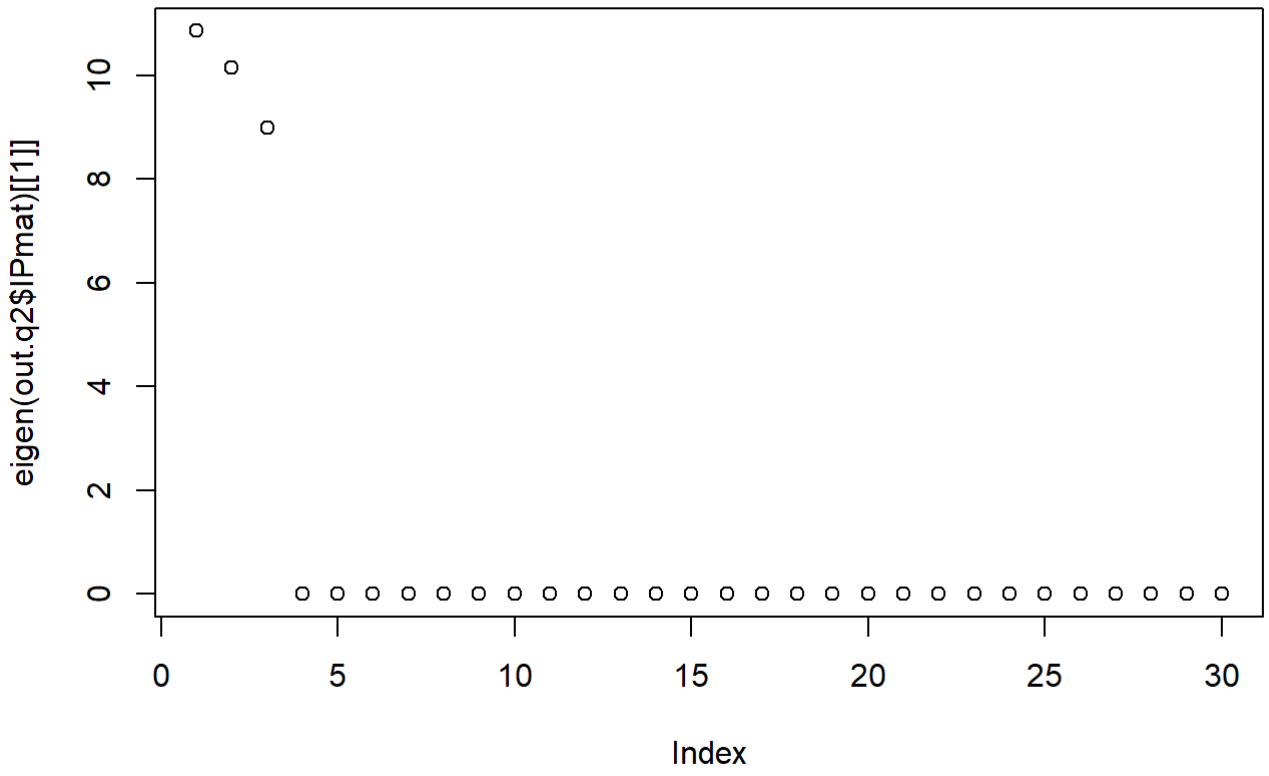
```
quiver2 <- my.random.tri.mut(out.q1$f)
out.q2 <- my.rsvd.triCycle(out.q1$X3.est,quiver1)
my.equilateral.plot(out.q2)
```

```
## [1] "inner product conditions"
```



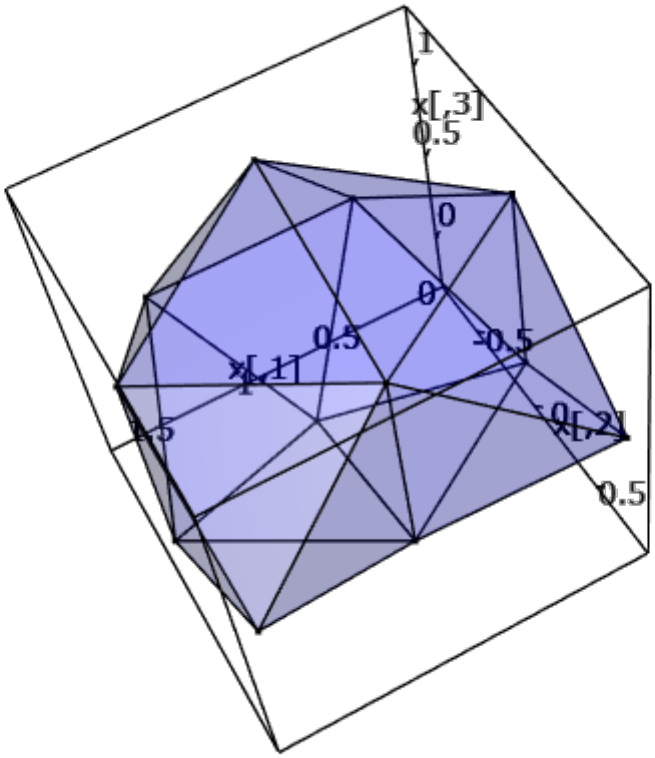

```
## [1] "triangle-closing conditions"
## [1] -7.771561e-16  9.992007e-16
```

```
plot(eigen(out.q2$IPmat)[[1]])
```




```
sum(eigen(out.q2$IPmat)[[1]])
```

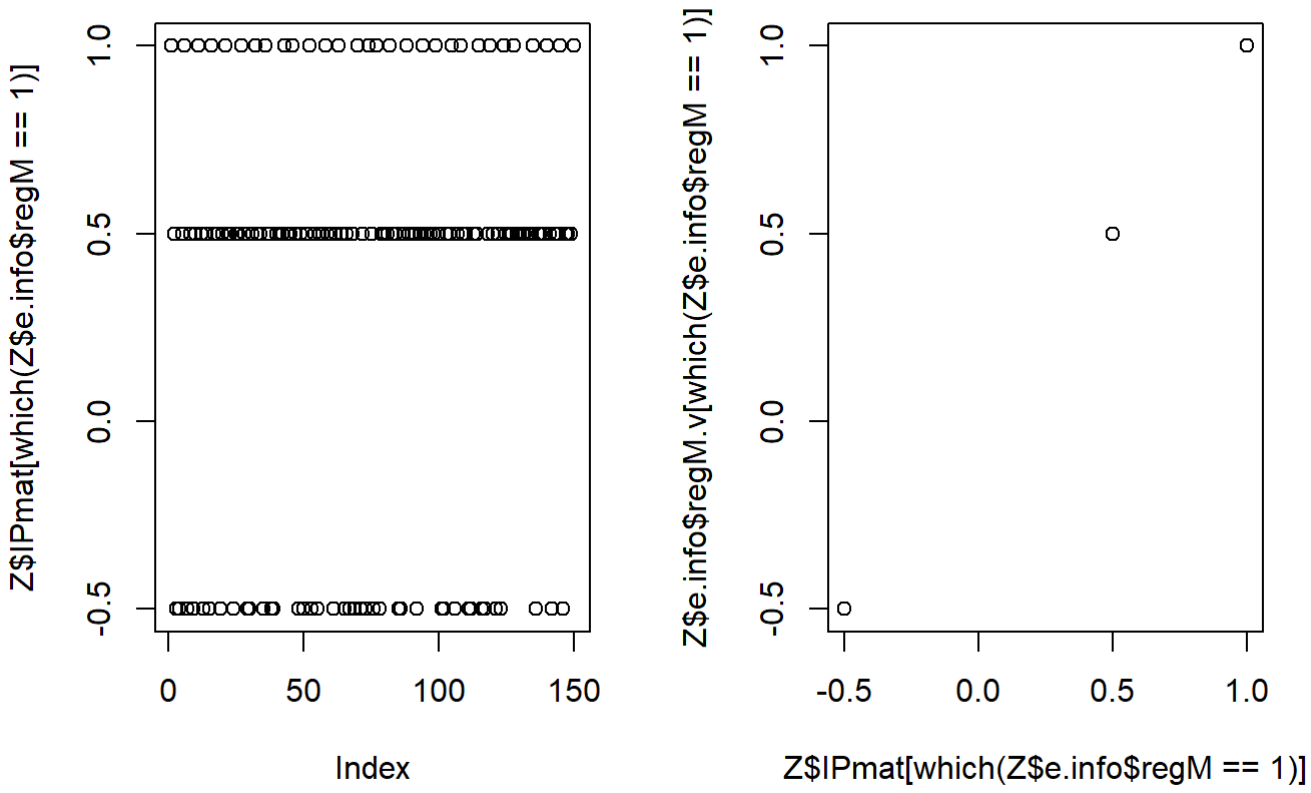
```
## [1] 30
```



さらに変異させてみる。

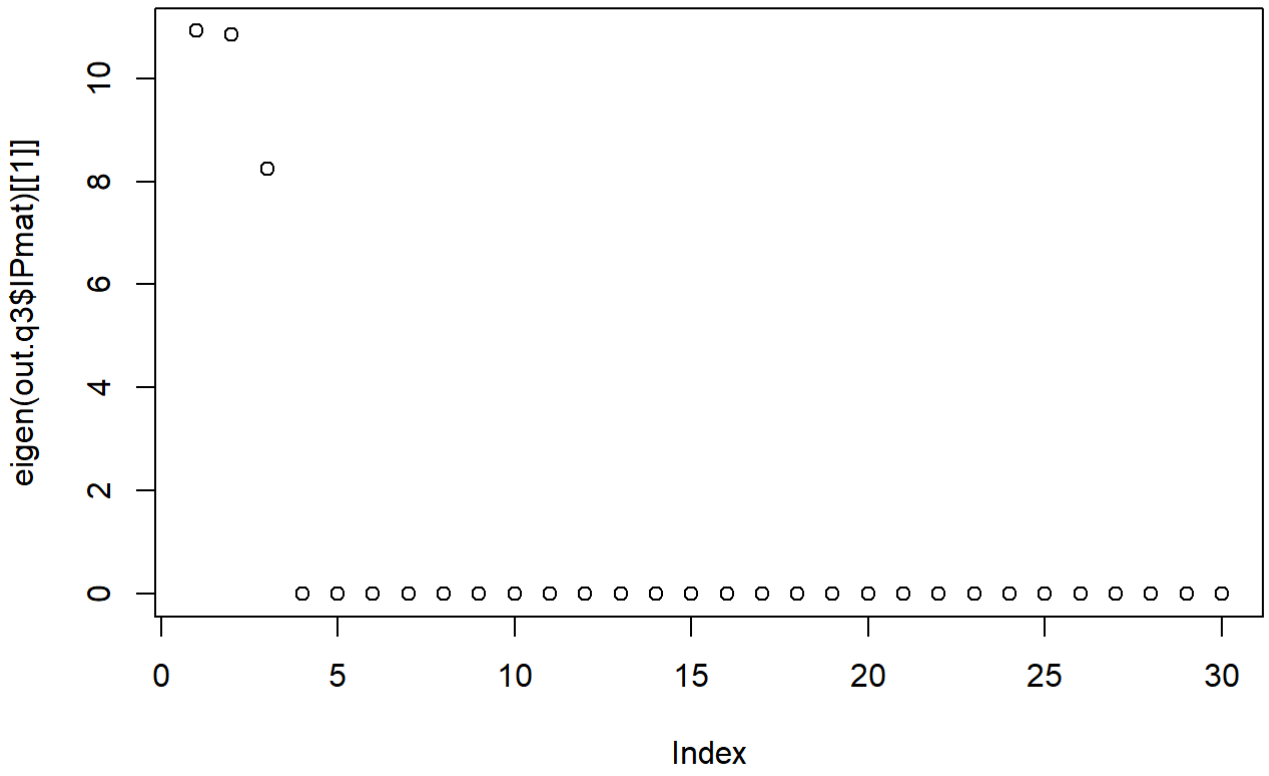
```
quiver3 <- my.random.tri.mut(out.q2$f)
out.q3 <- my.rsvd.triCycle(out.q2$X3.est,quiver2)
my.equilateral.plot(out.q3)
```

```
## [1] "inner product conditions"
```



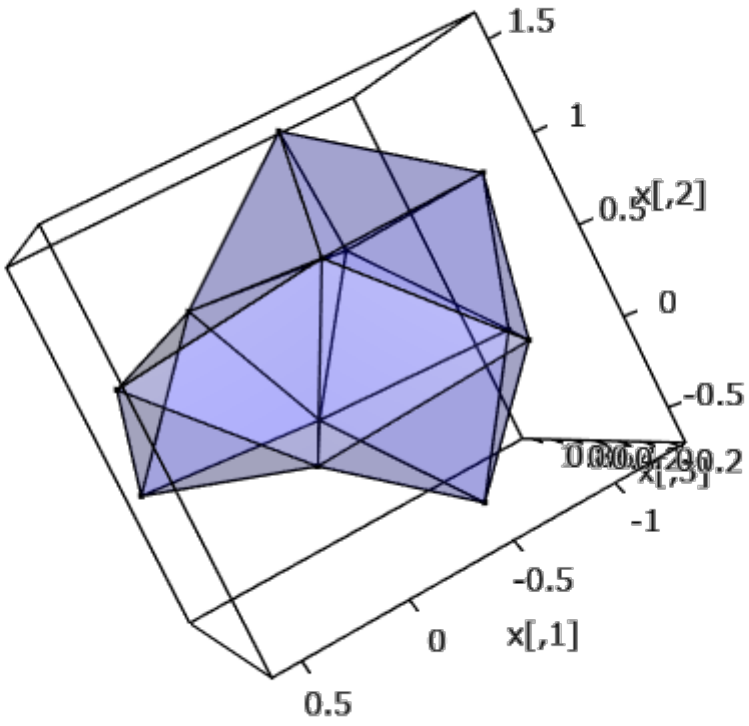

```
## [1] "triangle-closing conditions"
## [1] -7.771561e-16  8.326673e-16
```

```
plot(eigen(out.q3$IPmat)[[1]])
```



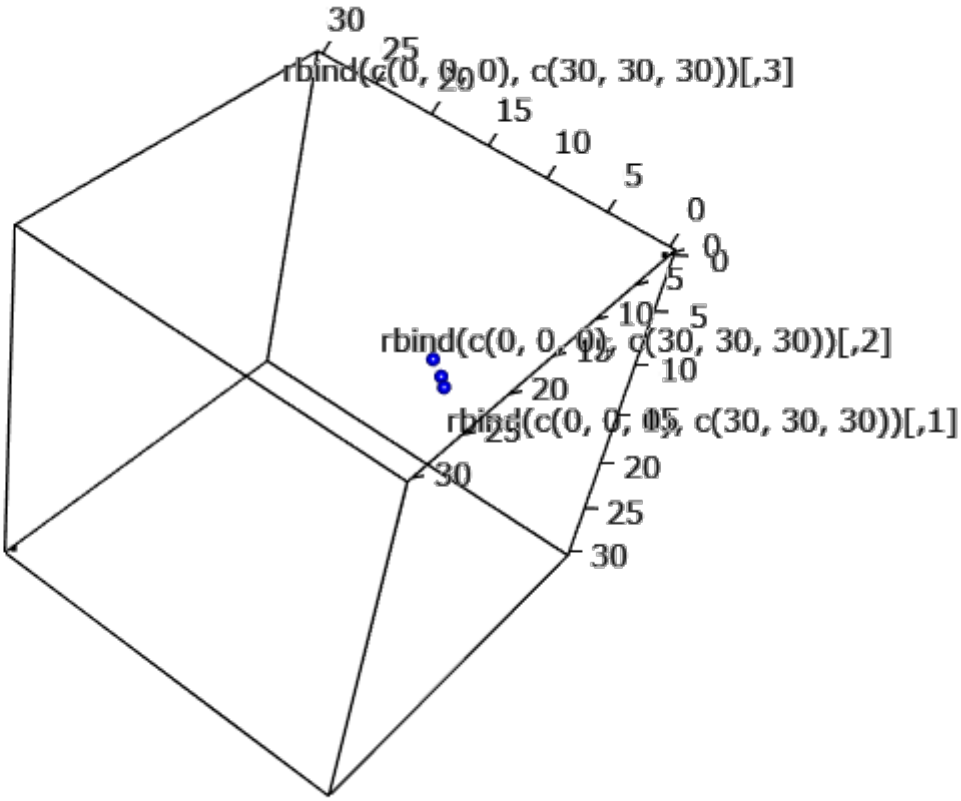

```
sum(eigen(out.q3$IPmat)[[1]])
```

```
## [1] 30
```



3つの固有値が表す点を打点してみる

```
e.xyz1 <- eigen(out.q1$IPmat)[[1]]
e.xyz2 <- eigen(out.q2$IPmat)[[1]]
e.xyz3 <- eigen(out.q3$IPmat)[[1]]

e.vecs <- rbind(eigen.out1[[1]],eigen.out2[[1]],e.xyz1,e.xyz2,e.xyz3)
plot3d(rbind(c(0,0,0),c(30,30,30)))
spheres3d(e.vecs[,1:3],radius=0.5,col=4)
```


rbind(c(0, 0, 0), c(30, 30, 30))[,3]
rbind(c(0, 0, 0), c(30, 30, 30))[,2]
rbind(c(0, 0, 0), c(30, 30, 30))[,1]