非正定値内積行列で座標化する

ryamada

*2018*年*5*月*26*日

グラフ距離行列問題を、線形独立ベクトルの組み合わせ問題と考える

グラフ距離行列がある。

今、グラフの頂点をすべて、m次元単位球面にあるものと仮定する。

グラフ距離の最大値でうまく補正して、最大距離がπ以下になるようにする。

今、球面上の２点間には円弧が引けるから、グラフ距離が円弧の長さになるように、各点の座標を取ったとする。

すると、2点のベクトルの内積は$\cos\theta$であり、今、円弧の長さがθである。

そうすると、各点の原点からの距離は1と仮定しているから、内積行列が、グラフ距離行列から作れる。

さて。

グラフ距離的に、「パス」が最短距離を表している、とは、点のトリオ(i,j,k)があったときに、
$d_{ij}+d_{jk}=d_{ik}$という関係にある、ということだが、

これは、上述の球面座標系では、3つの単位ベクトルが非線形独立であることと同じ。

じゃあ、このような内積行列があったとき、それを分解して、各点に座標を与えてやったら(内積行列が正定置であってほしい)、その中に、非線形独立なトリオたちが、どういう部分集合帰属関係になっているかを探す問題になるようである。

こういう問題は解けるのだろうか・・・

グラフ距離行列が、そんな都合のよい正定置内積行列はあるのか？

答えは不明。

でも、どうも、うまく行くっぽいことは、5月27日の処理を少し書き換えたコードでは、負の固有値が返ってこなさそうだから。

以下は、グラフ距離から、内積を計算する処理だけ、少し変更して、距離が内角であると考えて、内積を$\cos distance$として内積行列を作らせている。

１次元グラフ

直線状のグラフを考える。

ノード間距離を求めることができる。

距離と内積には内積を定める対称行列Mが与えられると、以下の関係を満足する。

$$(x_1-x_2)^TM(x_1-x_2)=x_1^TMx_1+x_2^TMX_2-2x_1^TMx_2$$

今、距離$(x_1 - x_2)^T M (x_1 - x_2)$は与えられるが、$x^T M x$は不明である。

ここですべての点で$x^T M x = 1$であると仮定した上で、座標を与えることとする。

これにより点間距離さえわかれば、内積行列を定めることができる。

その内積行列Pを

$$P = X^T S X = V^T \Sigma V = V^T |\Sigma| S V = (V \sqrt{|\Sigma|})^T S (V \sqrt{|\Sigma|})$$

と固有値分解することで、

Xをグラフの頂点の座標とすることにする。

ただし、Σは固有値を対角成分とする行列、 $|\Sigma|$ は固有値の絶対値を対角成分とする行列、 Sは固有値の具法を対角成分とする行列、 $\sqrt{|\Sigma}$ は固有値の絶対値の平方根を対角成分とする行列とする。

Rでやってみる

まずは、内積計算の関数を準備する。

```
my.ip <- function(v1,v2,M){
    matrix(v1,nrow=1) %*% M %*% matrix(v2,ncol=1)
}
my.ip.mat <- function(x,M){
    n <- length(x[,1])
    ret <- matrix(0,n,n)
    for(i in 1:n){
        for(j in 1:n){
            ret[i,j] <- my.ip(x[i,],x[j,],M)
        }
    }
    return(ret)
}
my.D2IP <- function(D,L=rep(1,length(D[,1]))){
  ret <- -D^2
  ret <- ret + L
  ret <- t(t(ret)+L)
  ret <- ret/2
  diag(ret) <- L
  return(ret)
}
# 球面上の弧長で考える
my.D2IP2 <- function(D,L=rep(1,length(D[,1]))){
  max.D <- max(D)
  r <- max.D/pi
  D. <- D/max.D * pi
  ret <- cos(D.)
  #ret <- ret + L
  #ret <- t(t(ret)+L)
  #ret <- ret/2
  diag(ret) <- 1
  return(list(P=ret,r=r))
}
my.IPcoords <- function(g,e.len){
  # 距離行列
  D <- distances(g,weights=e.len)
  # 内積行列
  P <- my.D2IP(D)
  # 固有値分解
  eout <- eigen(P)
  # 固有値の符号
  s <- sign(eout[[1]])
  # 内積行列(対角成分が±1)
  M <- diag(s)
  # 固有値絶対値の平方根を対角成分とする行列
  sq.ev <- sqrt(eout[[1]] * s)
  # 頂点座標
  V <- eout[[2]] %*% diag(sq.ev)
  return(list(X=V,M=M,eout=eout,D=D,P=P,s=s))
}

my.IPcoords2 <- function(g,e.len){
  # 距離行列
  D <- distances(g,weights=e.len)

  # 内積行列
  tmp <- my.D2IP2(D)
```

```
  P <- tmp$P
  r <- tmp$r
  # 固有値分解
  eout <- eigen(P)
  # 固有値の符号
  s <- sign(eout[[1]])
  # 内積行列(対角成分が±1)
  M <- diag(s)
  # 固有値絶対値の平方根を対角成分とする行列
  sq.ev <- sqrt(eout[[1]] * s)
  # 頂点座標
  V <- eout[[2]] %*% diag(sq.ev)
  return(list(X=V,M=M,eout=eout,D=D,P=P,s=s,r=r))
}
```

エッジ長がすべて等しい単純な一本鎖グラフ

グラフを作り、エッジ長を与え、 頂点間距離行列を作成し、 内積行列を作る。

```
# 1次元グラフ
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 3.4.4
```

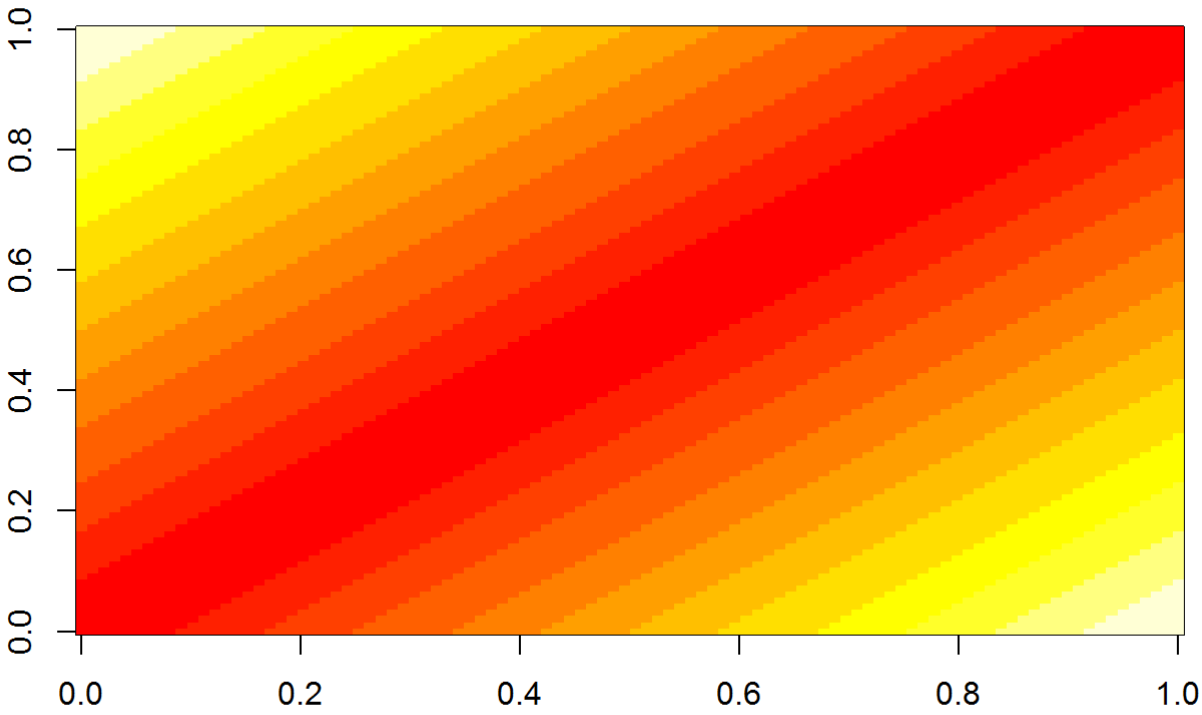
```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

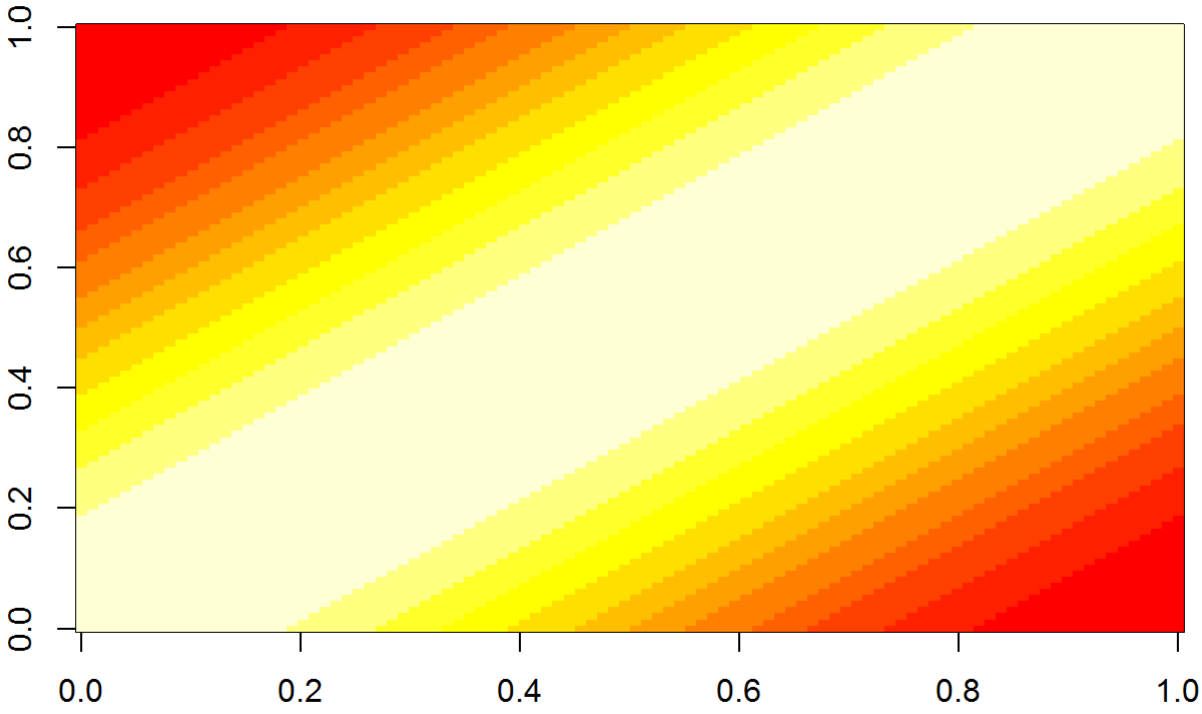
```
## The following object is masked from 'package:base':
##
##     union
```

```
# 頂点数
n1 <- 100
el1 <- cbind(1:(n1-1),2:n1)
# グラフオブジェクト
g1 <- graph.edgelist(el1,directed=FALSE)
# エッジ長
e.len1 <- rep(1,length(el1[,1]))
```

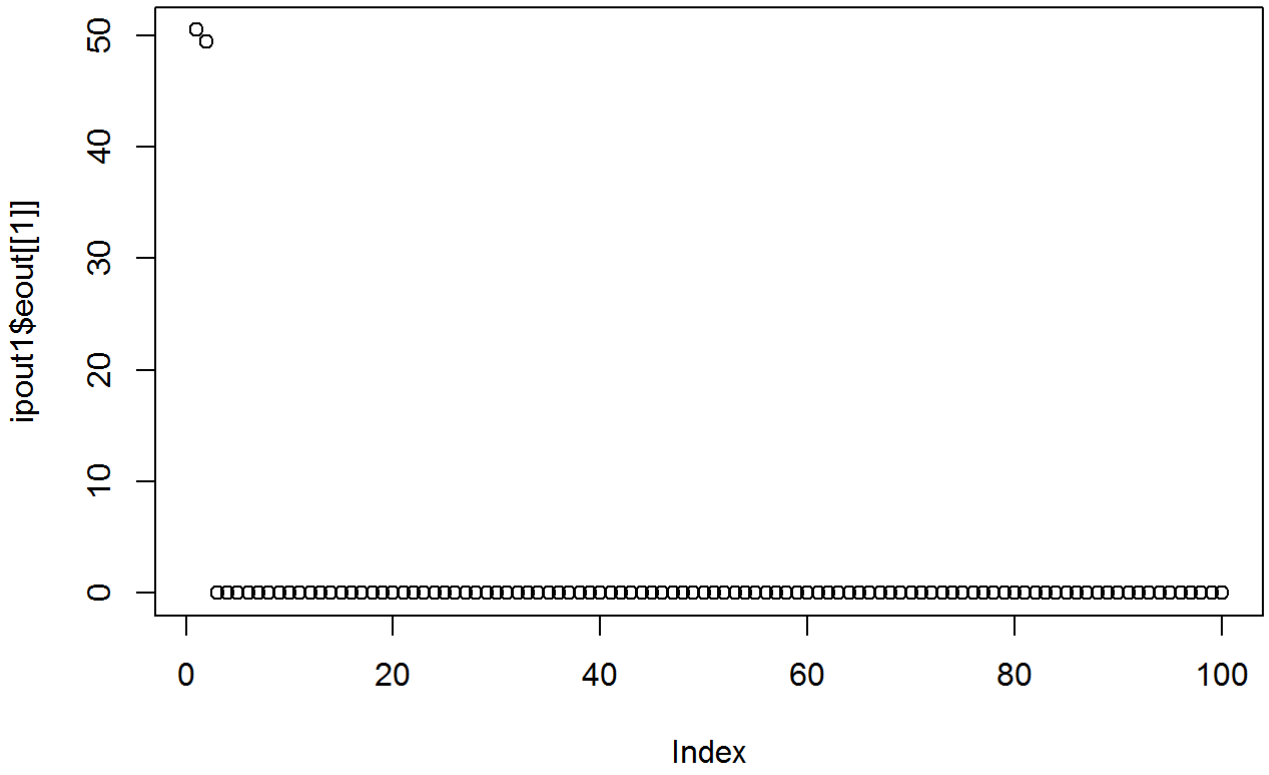
```
ipout1 <- my.IPcoords2(g1,e.len1)
# 距離行列
image(ipout1$D)
```



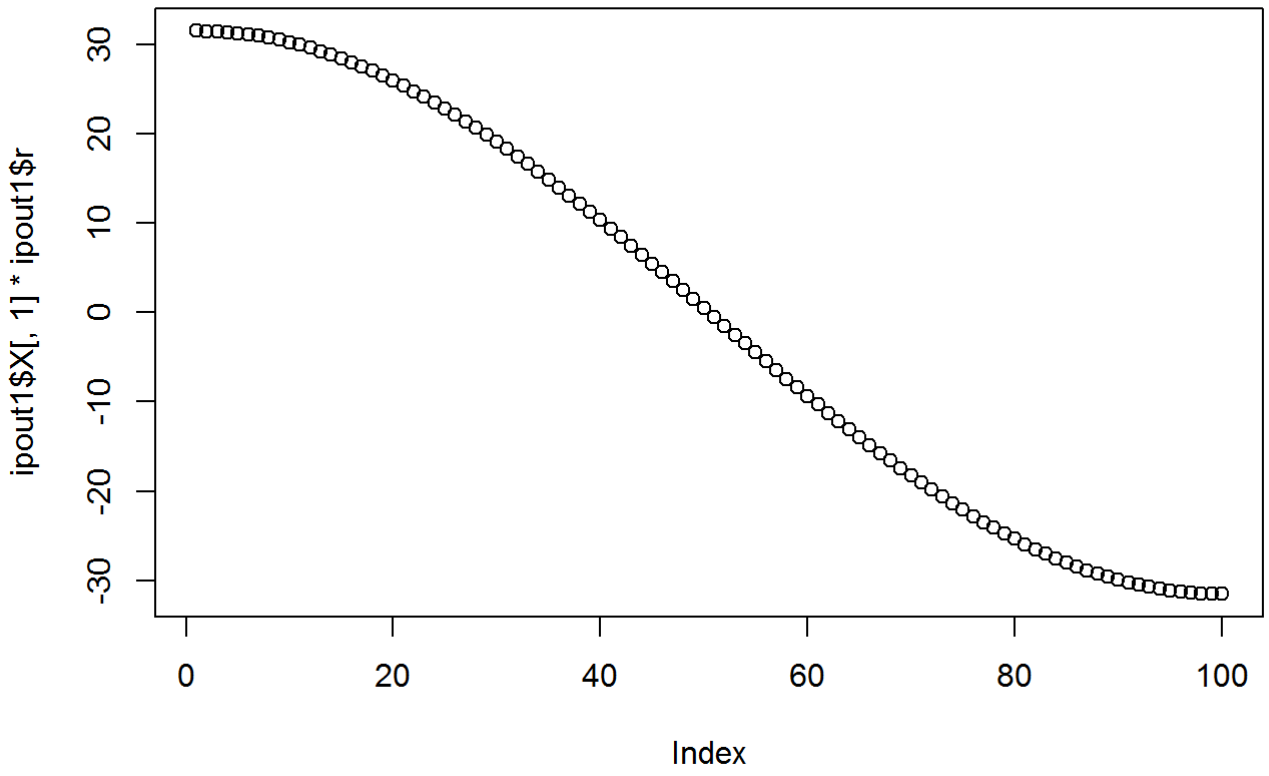

```
# IP行列
image(ipout1$P)
```



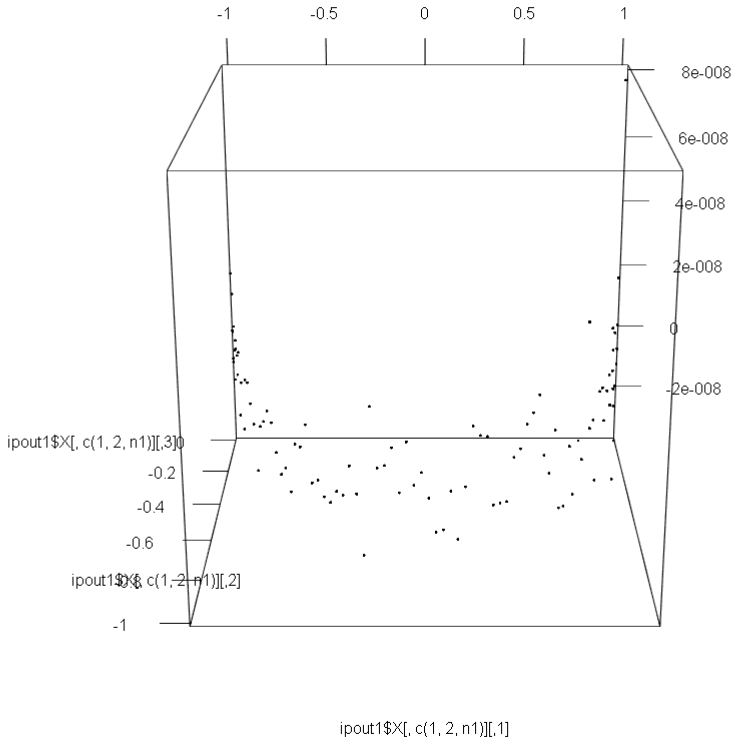
```
# 固有値
plot(ipout1$eout[[1]])
```




```
plot(ipout1$X[,1]*ipout1$r)
```



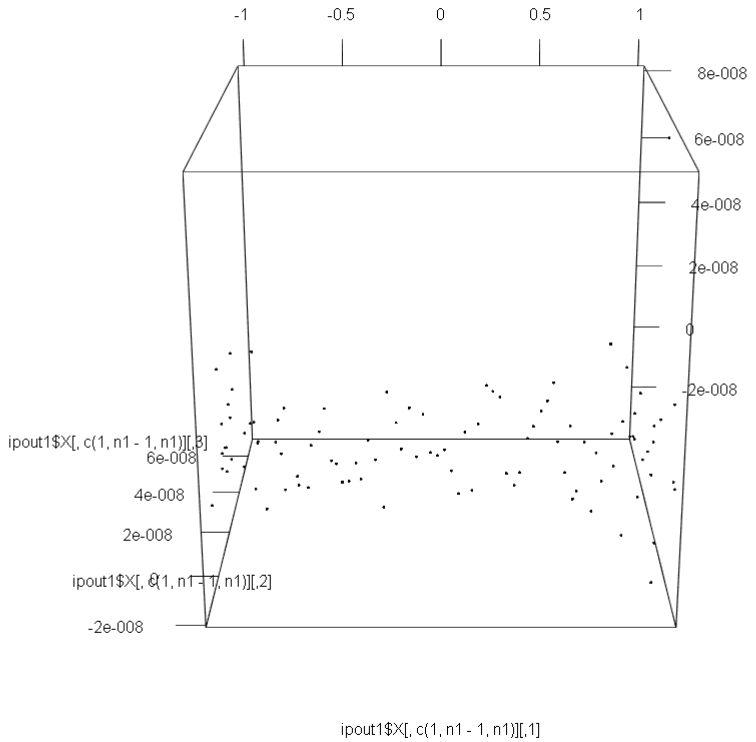

```
plot3d(ipout1$X[,c(1,2,n1)])
#spheres3d(ipout1$X[,c(1,2,n1)],radius=1)
```



```
ipout1$X[,1]^2 - ipout1$X[,n1]^2
```

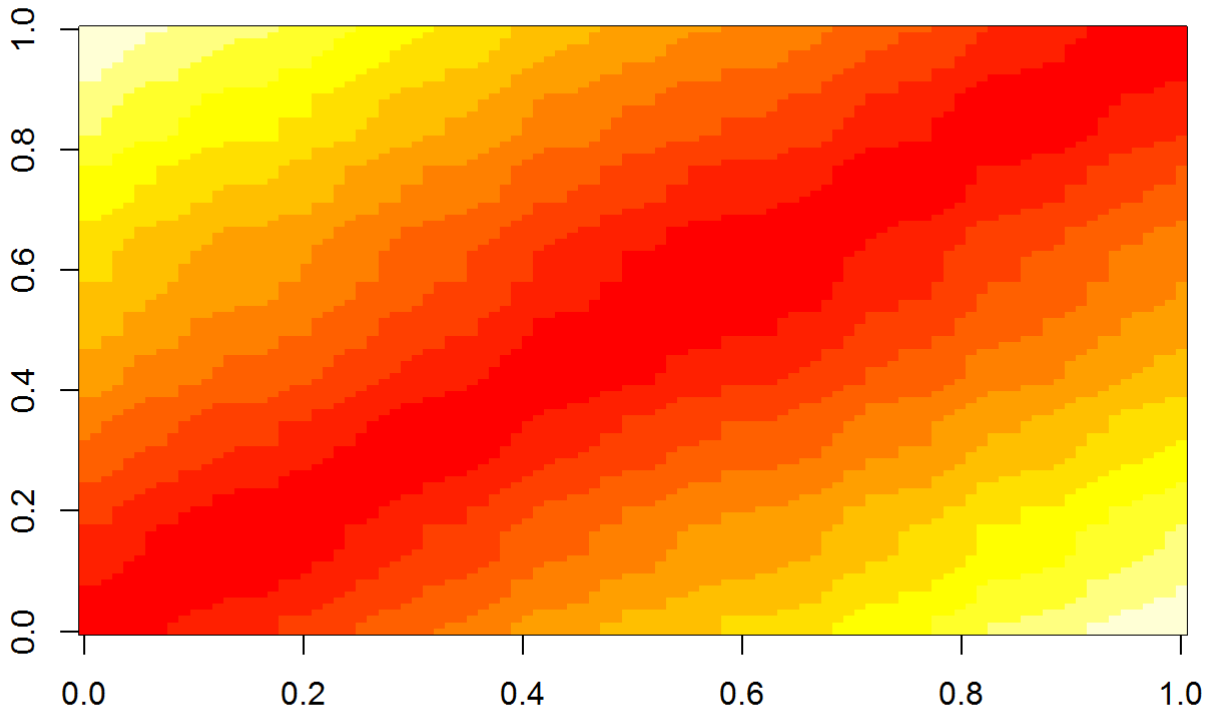
```
##   [1] 1.0000000000 0.9989933382 0.9959774064 0.9909643486 0.9839743507
##   [6] 0.9750355589 0.9641839665 0.9514632691 0.9369246885 0.9206267664
##  [11] 0.9026351288 0.8830222216 0.8618670191 0.8392547058 0.8152763335
##  [16] 0.7900284548 0.7636127338 0.7361355374 0.7077075065 0.6784431108
##  [21] 0.6484601877 0.6178794678 0.5868240888 0.5554191000 0.5237909579
##  [26] 0.4920670181 0.4603750216 0.4288425809 0.3975966660 0.3667630932
##  [31] 0.3364660183 0.3068274372 0.2779666937 0.2500000000 0.2230399681
##  [36] 0.1971951564 0.1725696330 0.1492625561 0.1273677752 0.1069734526
##  [41] 0.0881617093 0.0710082934 0.0555822757 0.0419457713 0.0301536896
##  [46] 0.0202535132 0.0122851066 0.0062805557 0.0022640387 0.0002517288
##  [51] 0.0002517288 0.0022640387 0.0062805557 0.0122851066 0.0202535132
##  [56] 0.0301536896 0.0419457713 0.0555822757 0.0710082934 0.0881617093
##  [61] 0.1069734526 0.1273677752 0.1492625561 0.1725696330 0.1971951564
##  [66] 0.2230399681 0.2500000000 0.2779666937 0.3068274372 0.3364660183
##  [71] 0.3667630932 0.3975966660 0.4288425809 0.4603750216 0.4920670181
##  [76] 0.5237909579 0.5554191000 0.5868240888 0.6178794678 0.6484601877
##  [81] 0.6784431108 0.7077075065 0.7361355374 0.7636127338 0.7900284548
##  [86] 0.8152763335 0.8392547058 0.8618670191 0.8830222216 0.9026351288
##  [91] 0.9206267664 0.9369246885 0.9514632691 0.9641839665 0.9750355589
##  [96] 0.9839743507 0.9909643486 0.9959774064 0.9989933382 1.0000000000
```

```
plot3d(ipout1$X[,c(1,n1-1,n1)])
spheres3d(ipout1$X[,c(1,n1-1,n1)],radius=0.001)
```

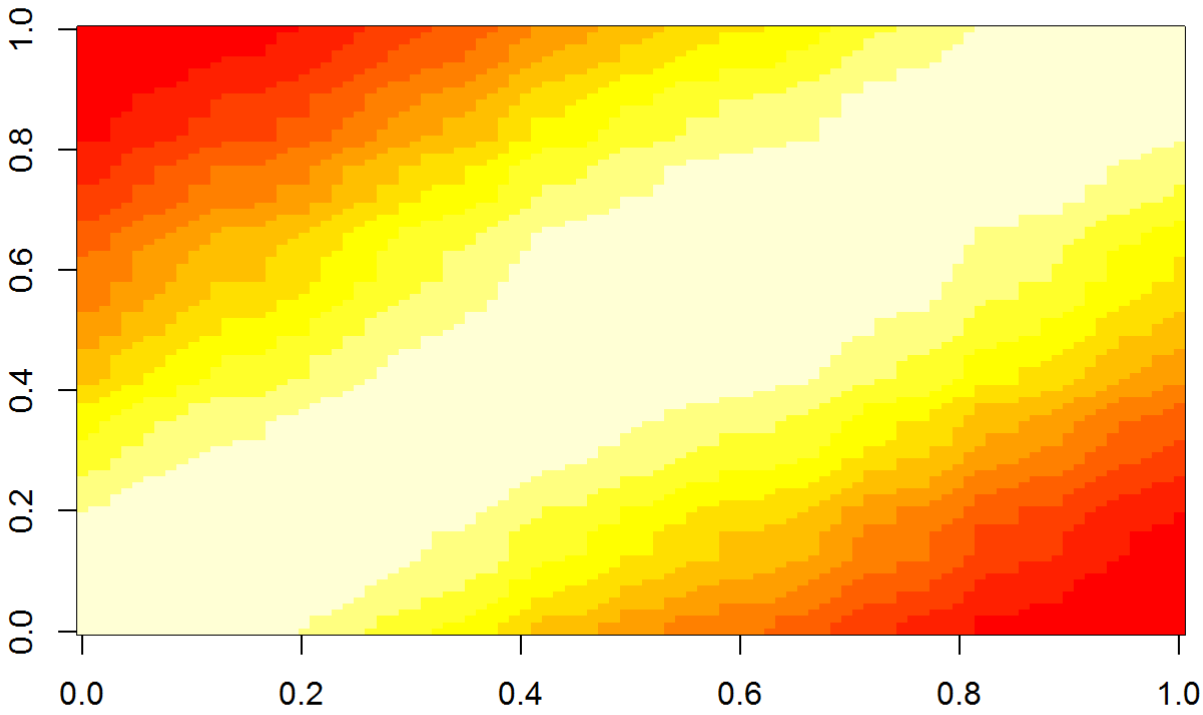


エッジ長がばらばら

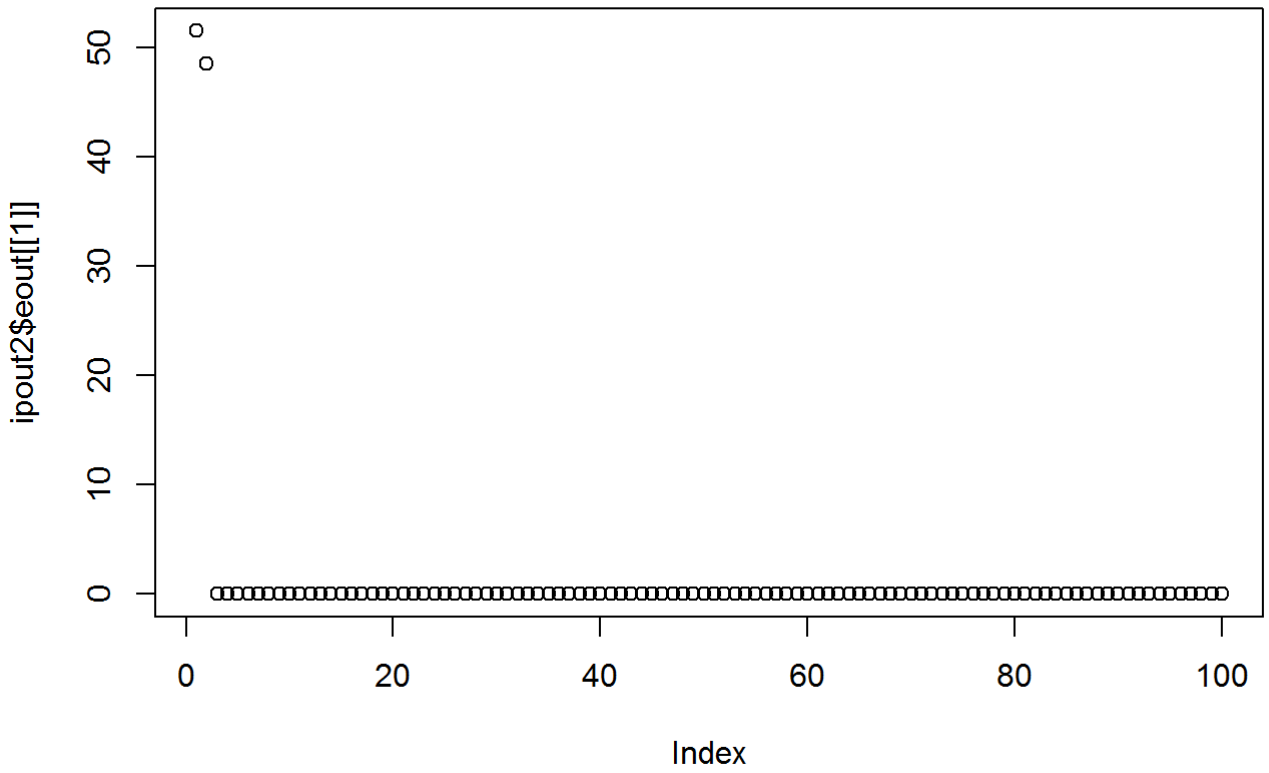
```
# 1次元グラフ
library(igraph)
# 頂点数
n2 <- 100
el2 <- cbind(1:(n2-1),2:n2)
# グラフオブジェクト
g2 <- graph.edgelist(el2,directed=FALSE)
# エッジ長をばらばらにする
e.len2 <- runif(length(el2[,1]))

ipout2 <- my.IPcoords2(g2,e.len2)
# 距離行列
image(ipout2$D)
```




```
# IP行列
image(ipout2$P)
```

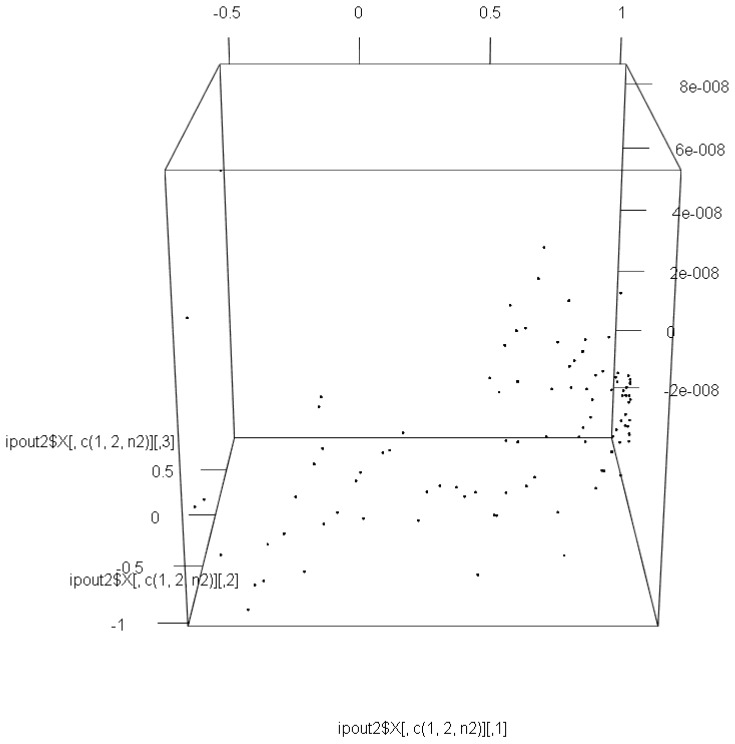



```
# 固有値
plot(ipout2$eout[[1]])
```



現れるカーブは同じだが、打点の間隔が異なっている。

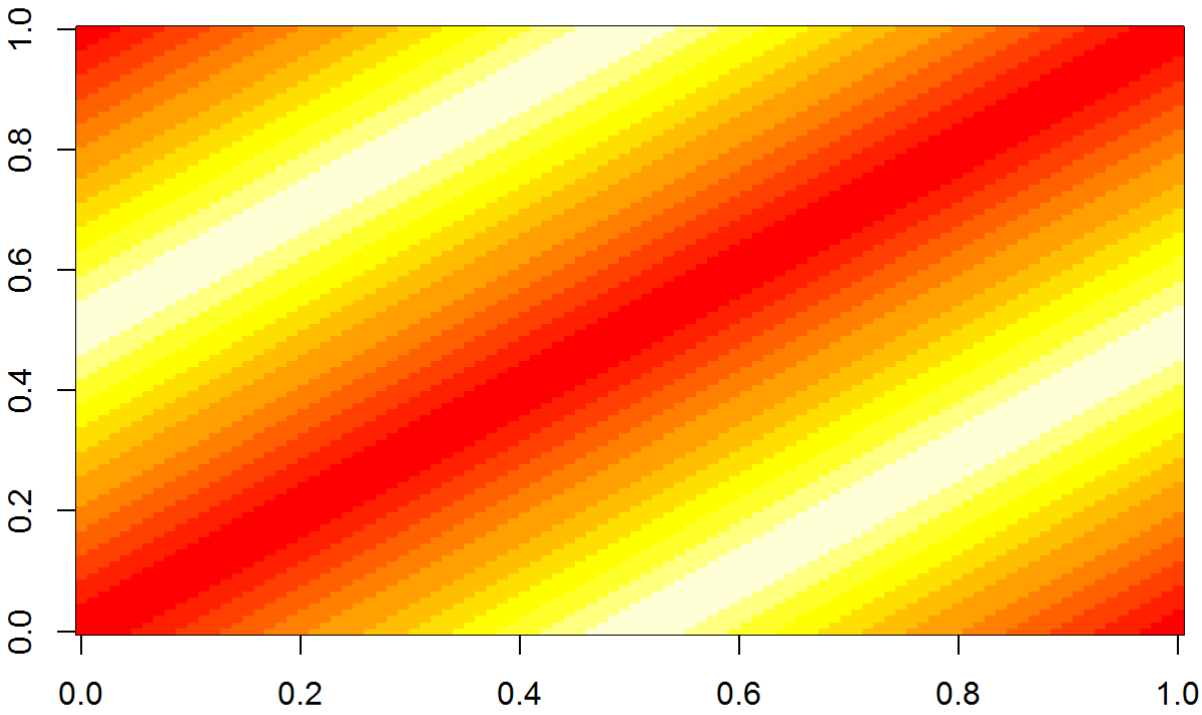
```
plot3d(ipout2$X[,c(1,2,n2)])
#spheres3d(ipout2$X[,c(1,2,n2)],radius=1)
```



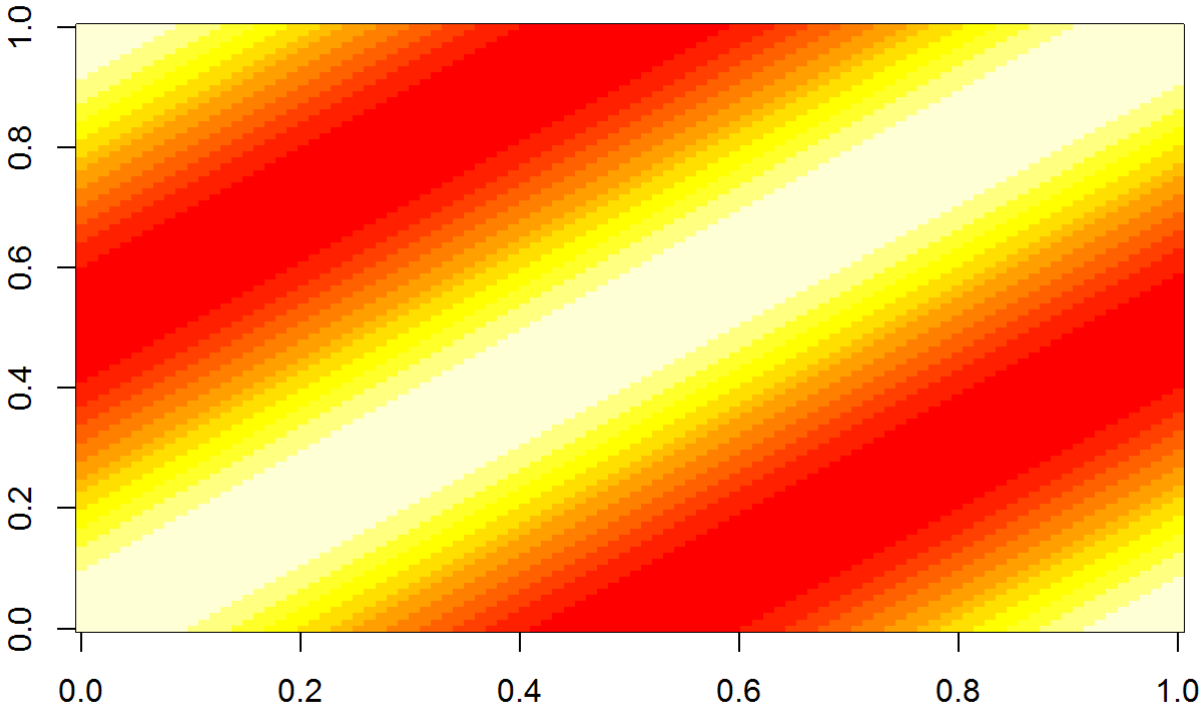
エッジ長がすべて等しい単純な周回グラフ

グラフを作り、エッジ長を与え、 頂点間距離行列を作成し、 内積行列を作る。

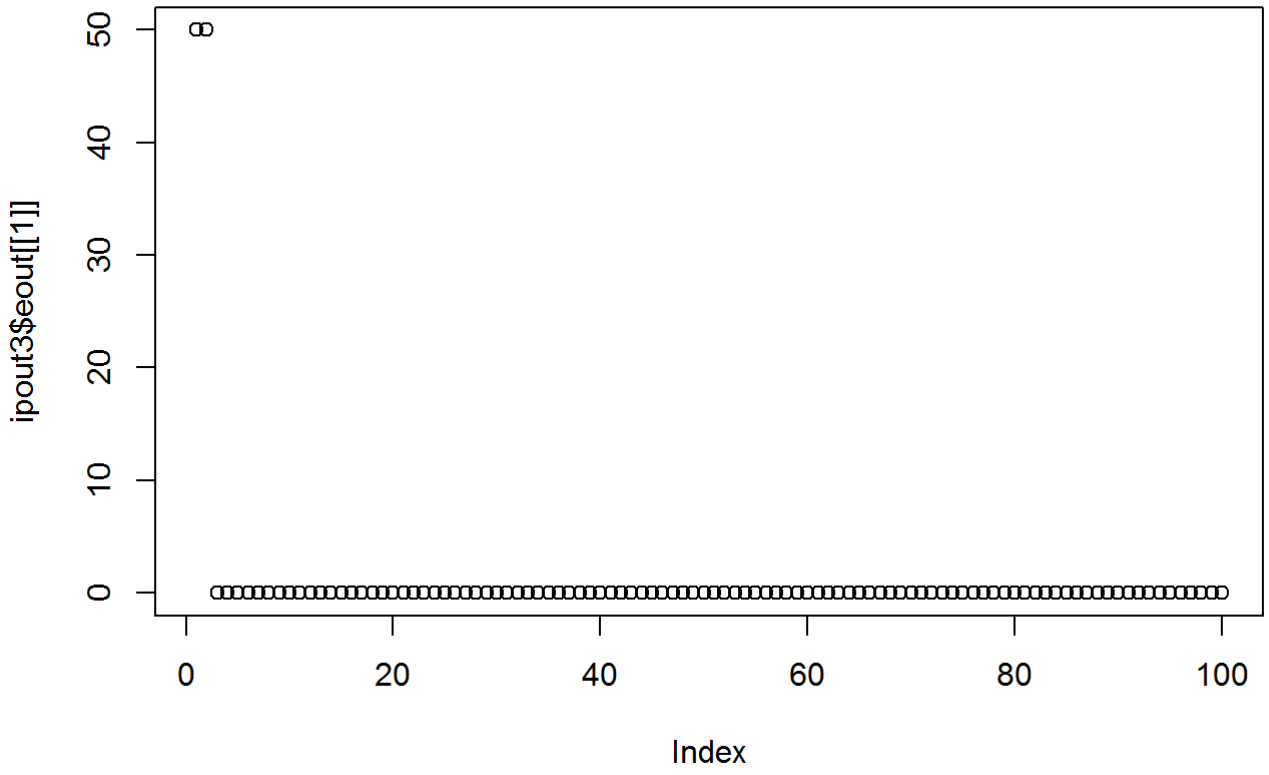
```
# 円周グラフ
# 頂点数
n3 <- 100
el3 <- cbind(1:(n3-1),2:n3)
el3 <- rbind(el3,c(n3,1))
# グラフオブジェクト
g3 <- graph.edgelist(el3,directed=FALSE)
# エッジ長
e.len3 <- rep(1,length(el3[,1]))
ipout3 <- my.IPcoords2(g3,e.len3)
# 距離行列
image(ipout3$D)
```



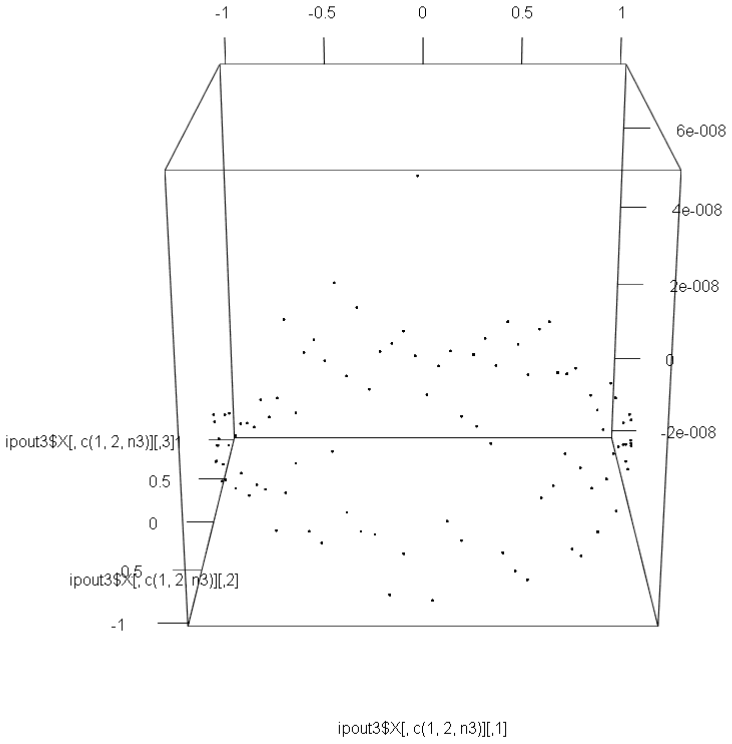

```
# IP行列
image(ipout3$P)
```



```
# 固有値
plot(ipout3$eout[[1]])
```




```
plot3d(ipout3$X[,c(1,2,n3)])
#spheres3d(ipout3$X[,c(1,2,n3)],radius=1)
```



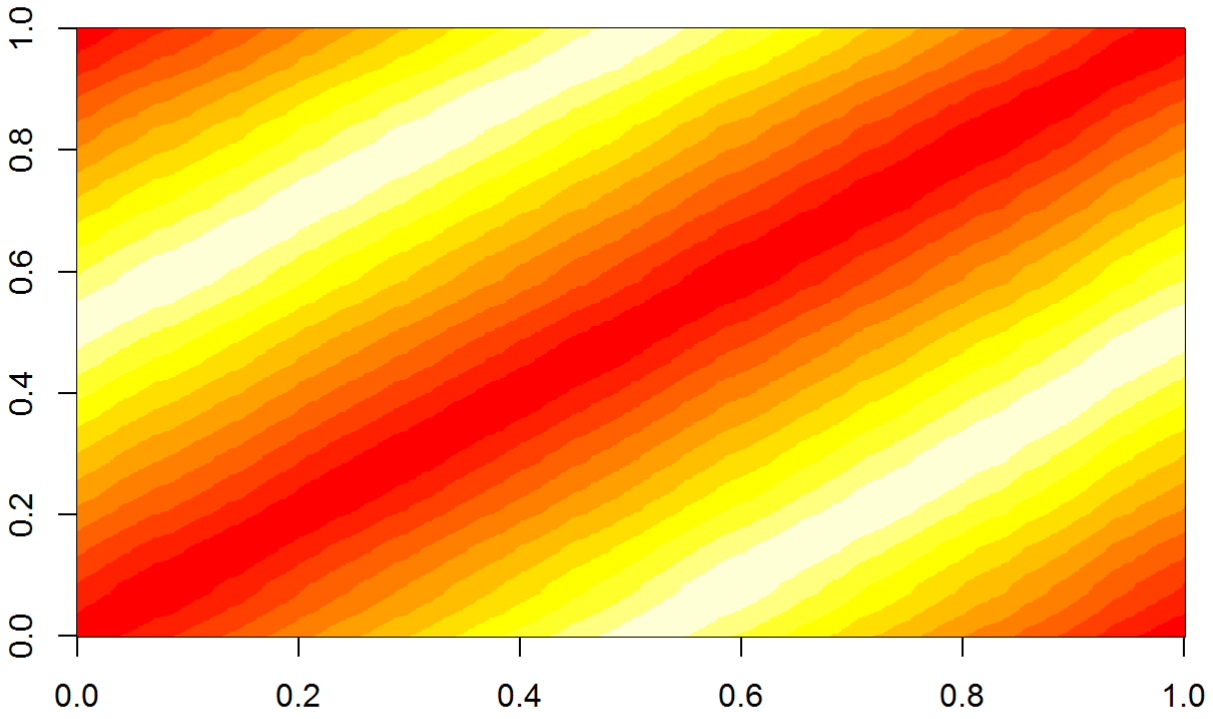

```
ipout3$X[1,n3]
```

```
## [1] -1.669439e-08
```

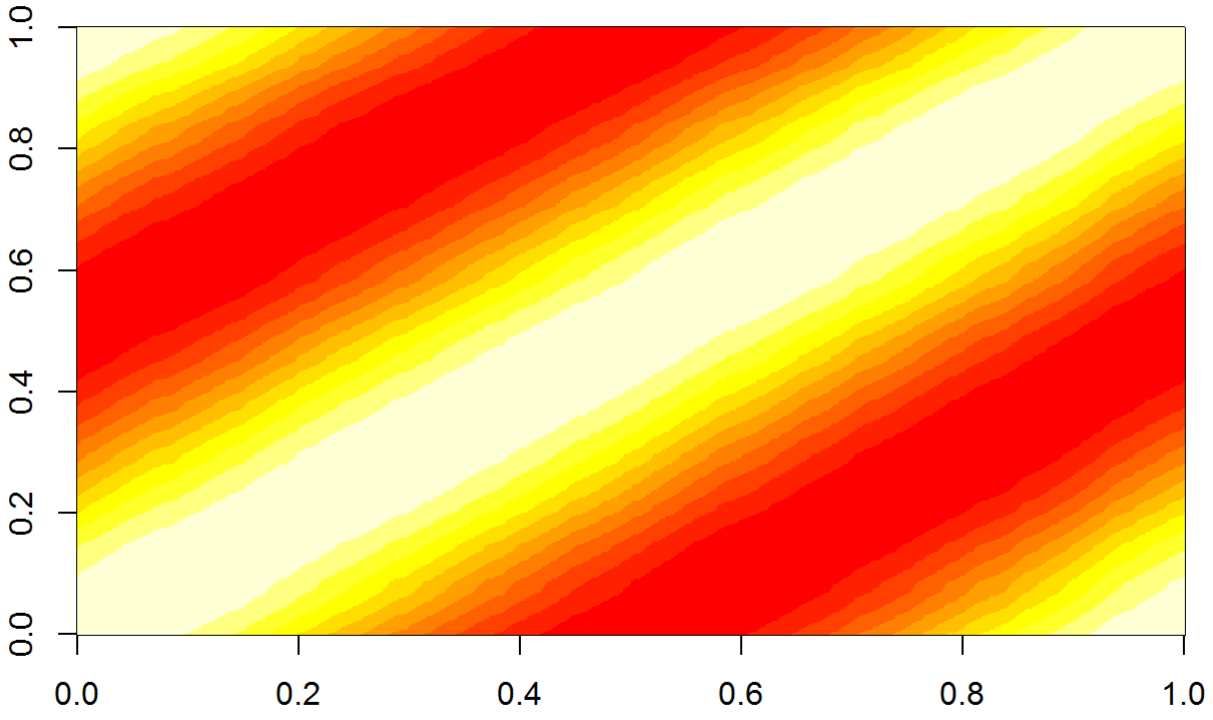
エッジ長をバラバラにする

エッジのばらばら加減が座標に反映する

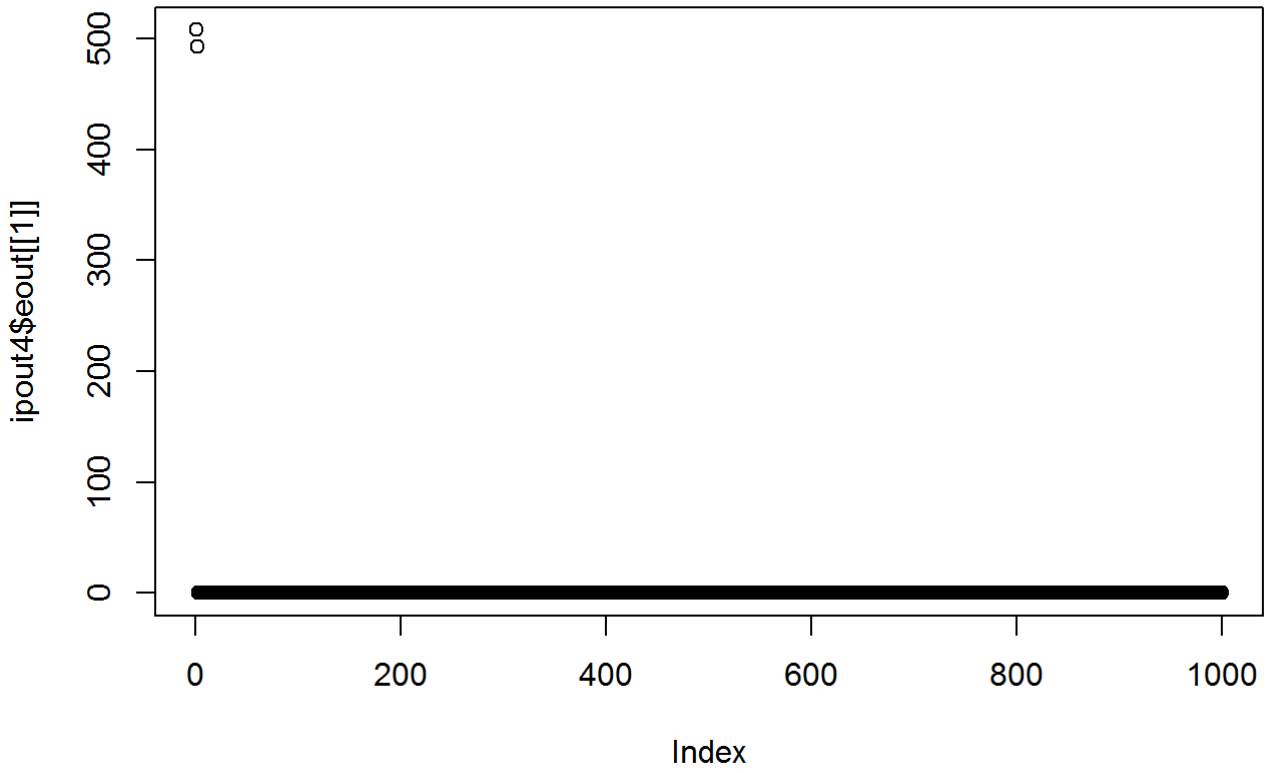
```
# 円周グラフ
# 頂点数
n4 <- 1000
el4 <- cbind(1:(n4-1),2:n4)
el4 <- rbind(el4,c(n4,1))
# グラフオブジェクト
g4 <- graph.edgelist(el4,directed=FALSE)
# エッジ長
e.len4 <- runif(length(el4[,1]))
ipout4 <- my.IPcoords2(g4,e.len4)
# 距離行列
image(ipout4$D)
```




```
# IP行列
image(ipout4$P)
```

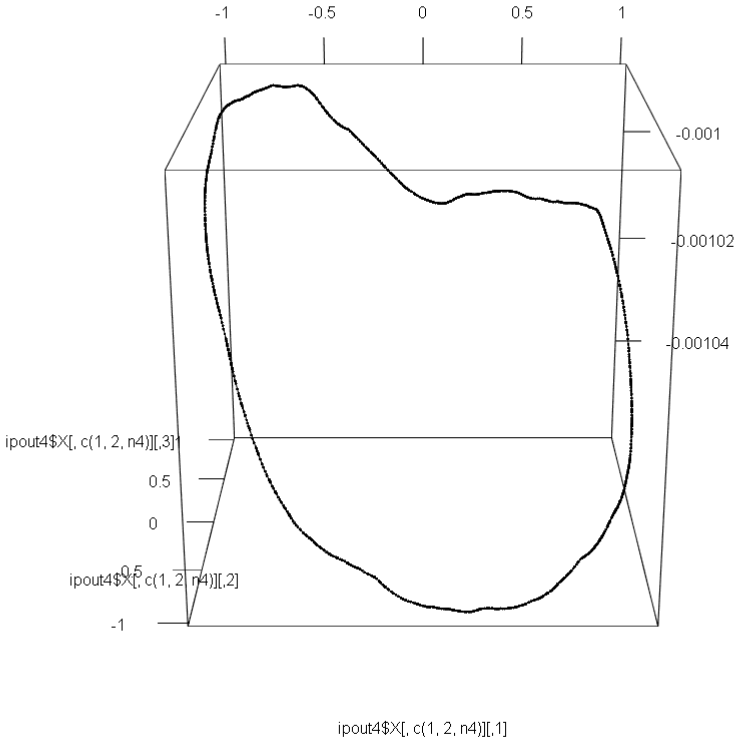


```
# 固有値
plot(ipout4$eout[[1]])
```



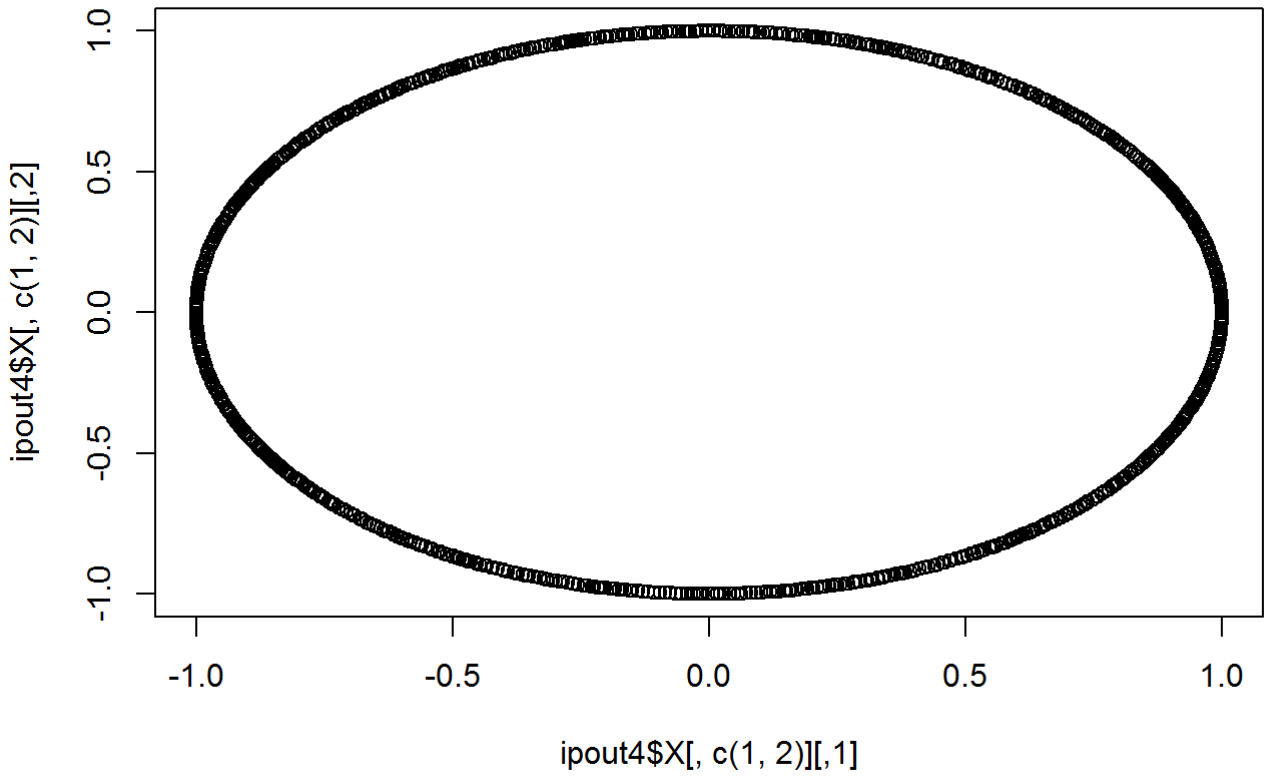
打点間隔がばらつくとともに、実現空間の次元も上がっている。

```
plot3d(ipout4$X[,c(1,2,n4)])
#spheres3d(ipout4$X[,c(1,2,n4)],radius=1)
el4 <- get.edgelist(g4)
segments3d(ipout4$X[t(el4),c(1,2,n4)])
```




```
plot3d(ipout4$X[,c(1,2,3)])
```

```
plot(ipout4$X[,c(1,2)])
```

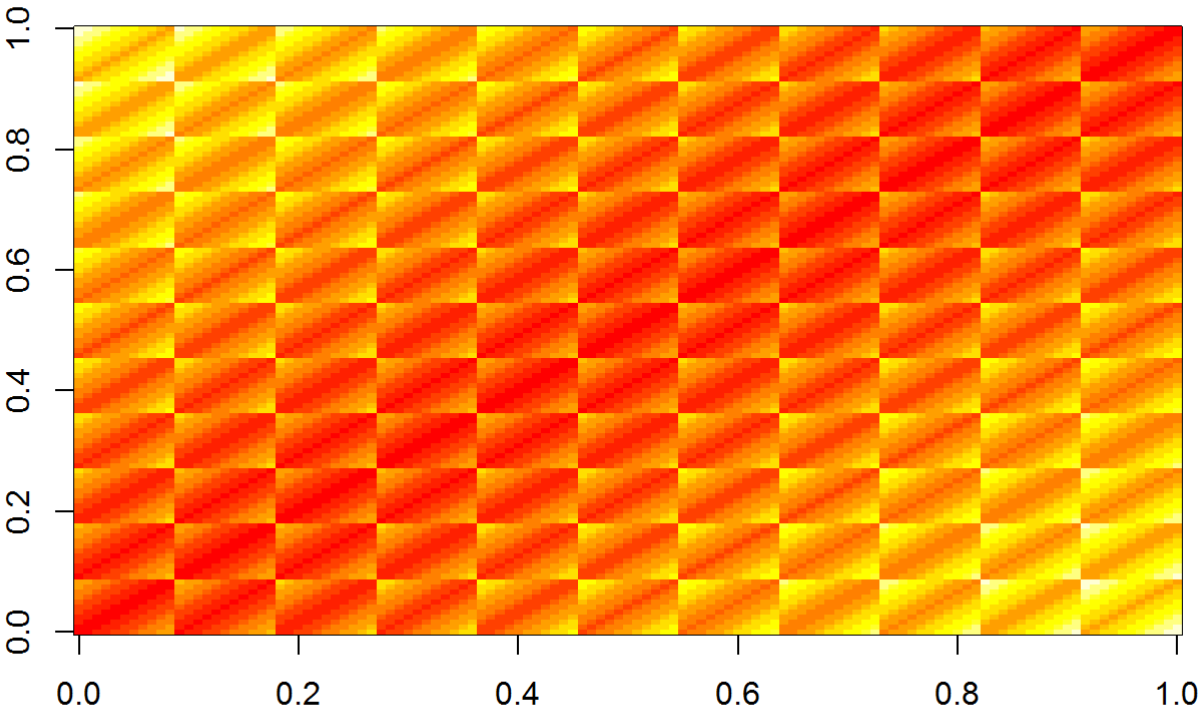


#

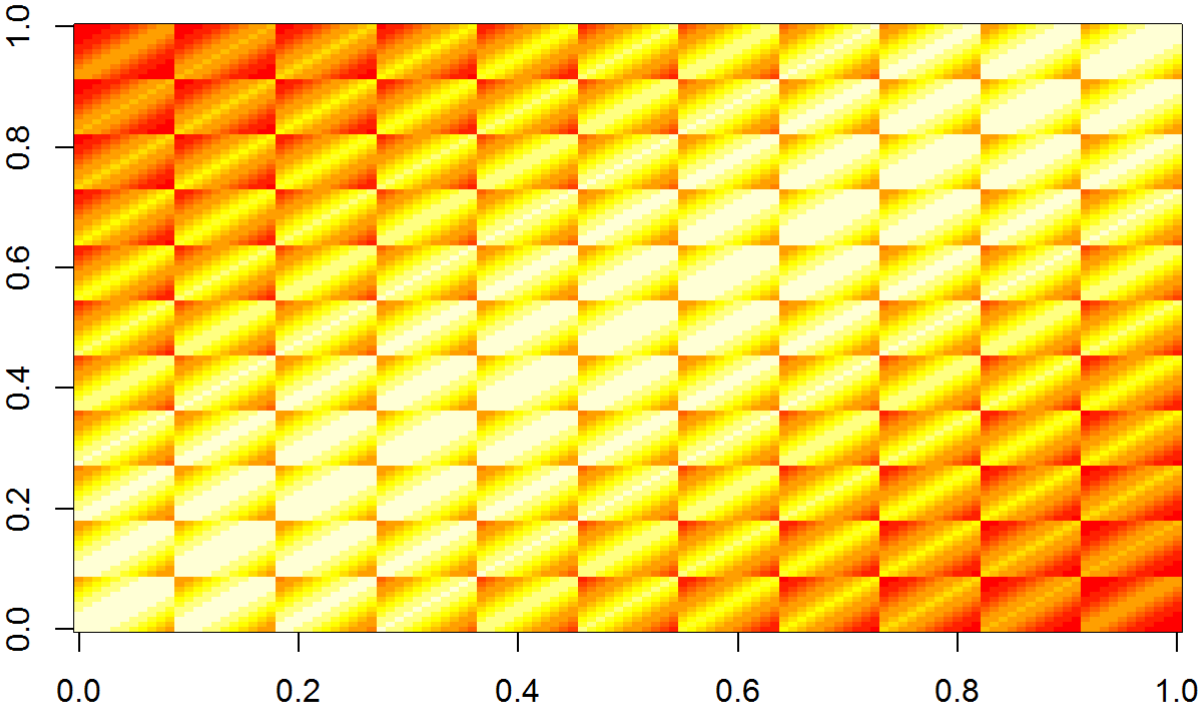
2次元

平面方眼紙グラフの場合

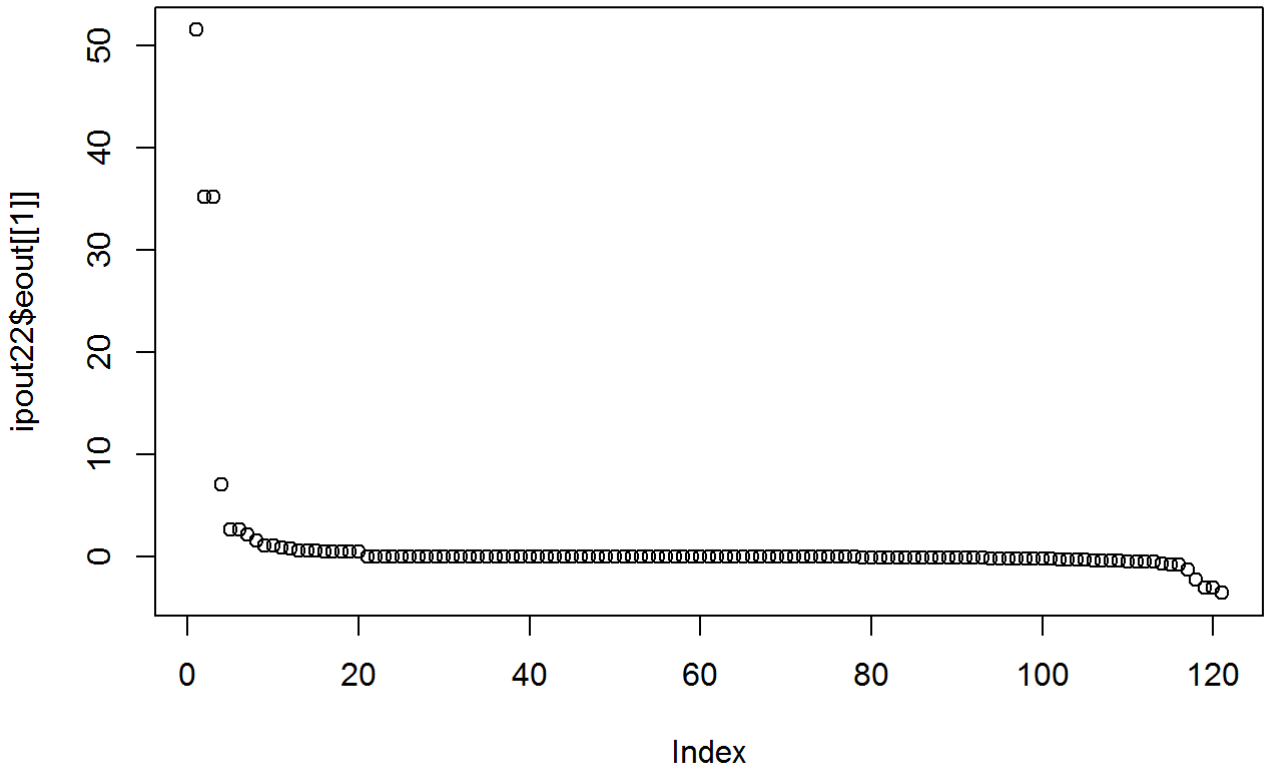
```
xy <- as.matrix(expand.grid(0:10,0:10))
d <- as.matrix(dist(xy))
d <- d==1 +0
g22 <- graph.adjacency(d,mode="undirected")
# エッジ長
e.len22 <- rep(1,length(E(g22)))
ipout22 <- my.IPcoords2(g22,e.len22)
# 距離行列
image(ipout22$D)
```



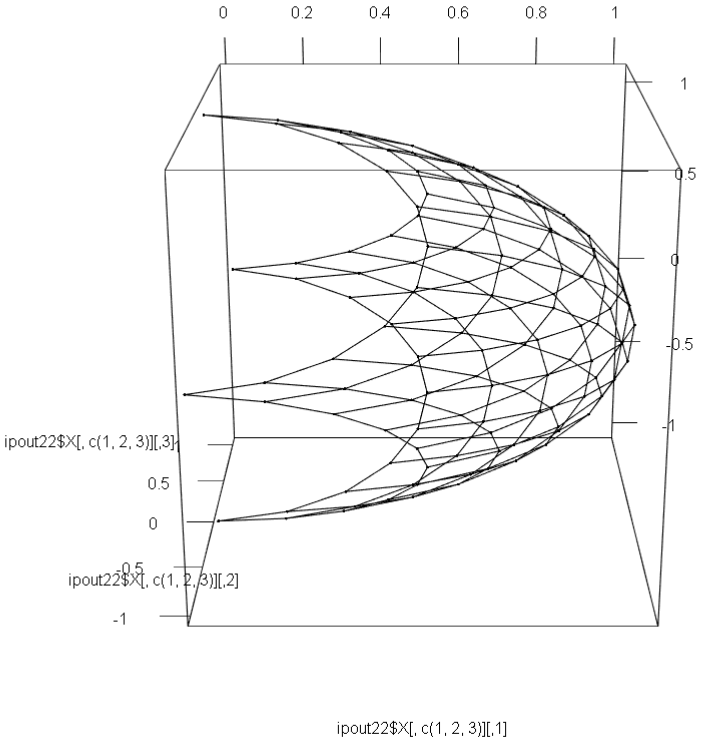

```
# IP行列
image(ipout22$P)
```



```
# 固有値
plot(ipout22$eout[[1]])
```

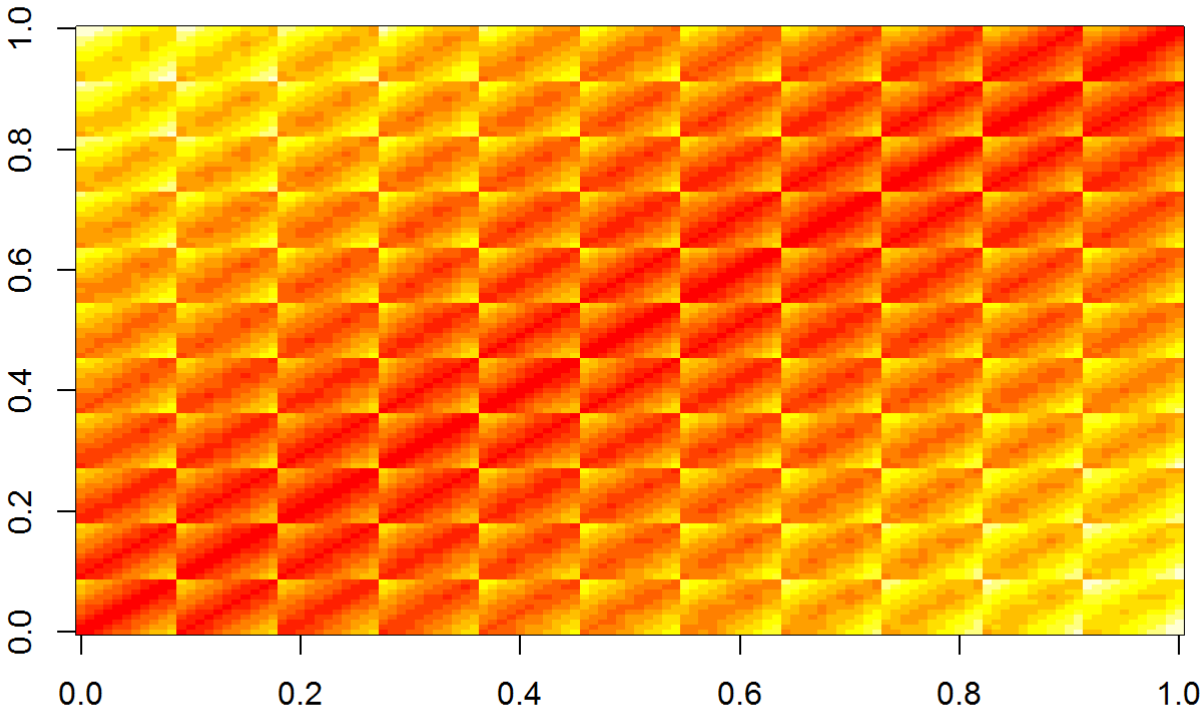



```
npt <- length(xy[,1])
plot3d(ipout22$X[,c(1,2,3)])
#spheres3d(ipout22$X[,c(1,2,npt)],radius=0.1)
el22 <- matrix(as.numeric(get.edgelist(g22)),ncol=2)
segments3d(ipout22$X[t(el22),c(1,2,3)])
```

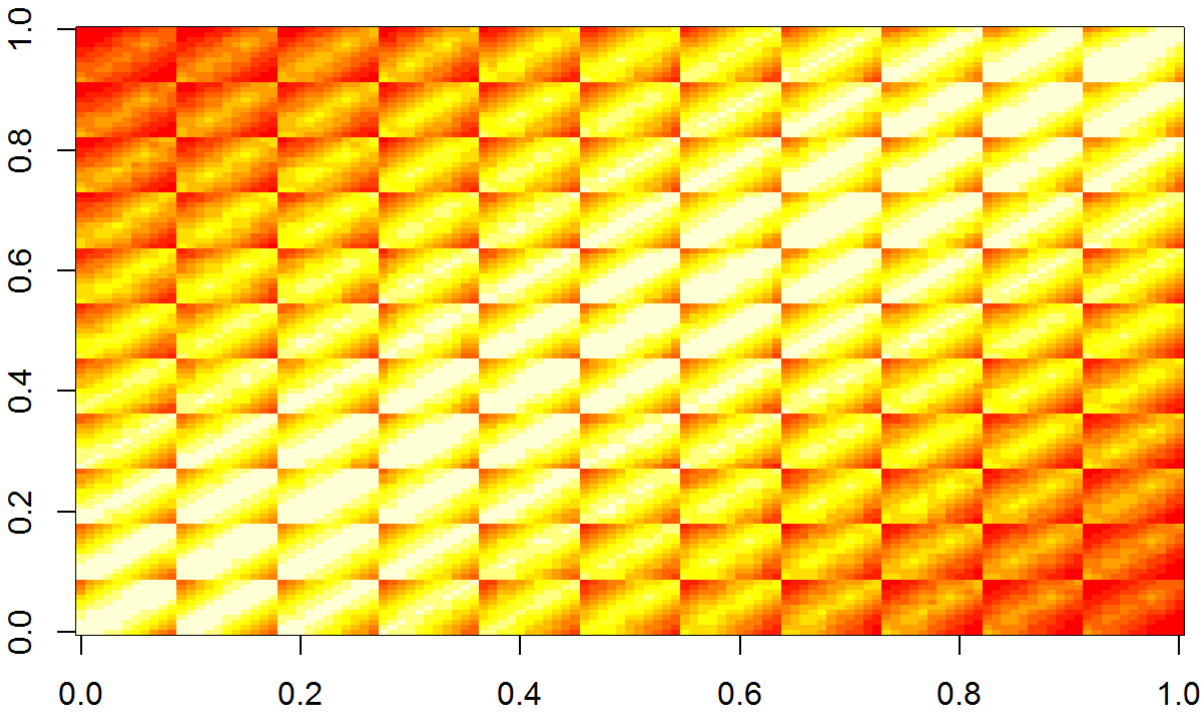


ばらつかせる。

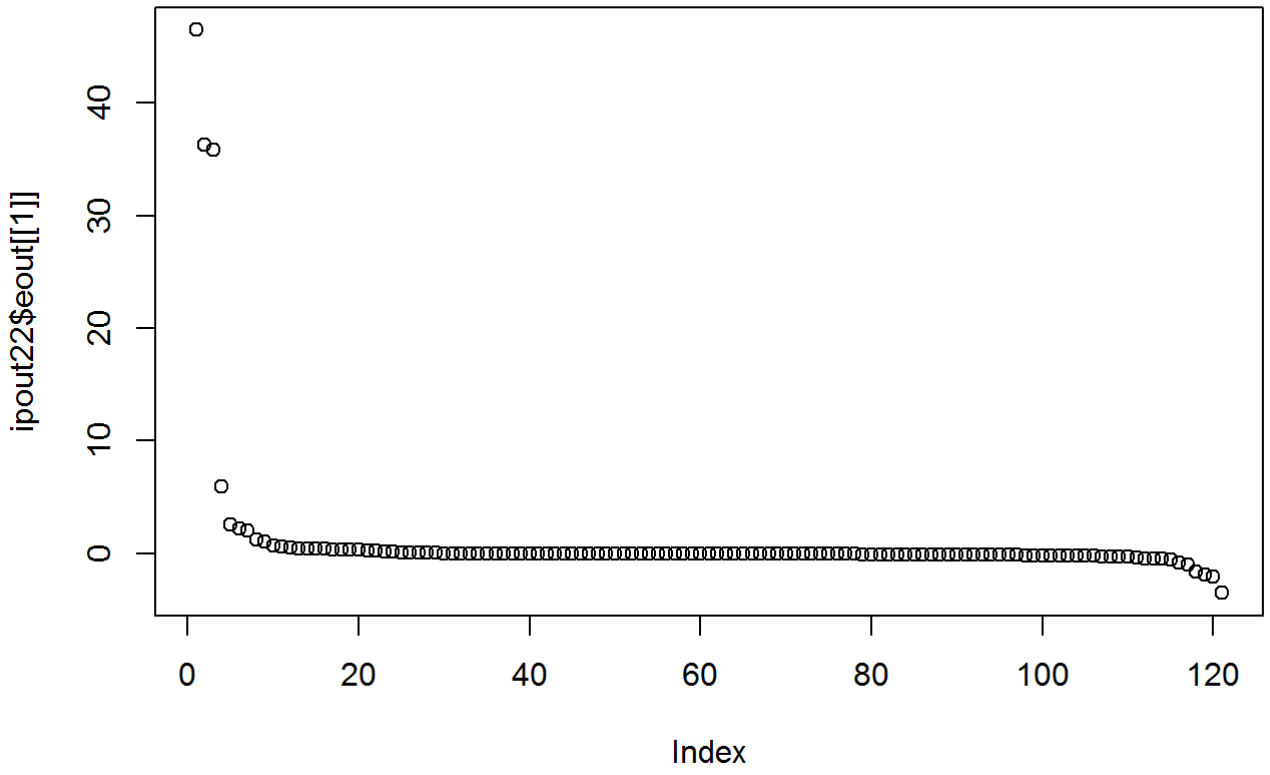
```
xy <- as.matrix(expand.grid(0:10,0:10))
d <- as.matrix(dist(xy))
d <- d==1 +0
g22 <- graph.adjacency(d,mode="undirected")
# エッジ長
e.len22 <- rep(1,length(E(g22))) + runif(length(E(g22)))
ipout22 <- my.IPcoords2(g22,e.len22)
# 距離行列
image(ipout22$D)
```



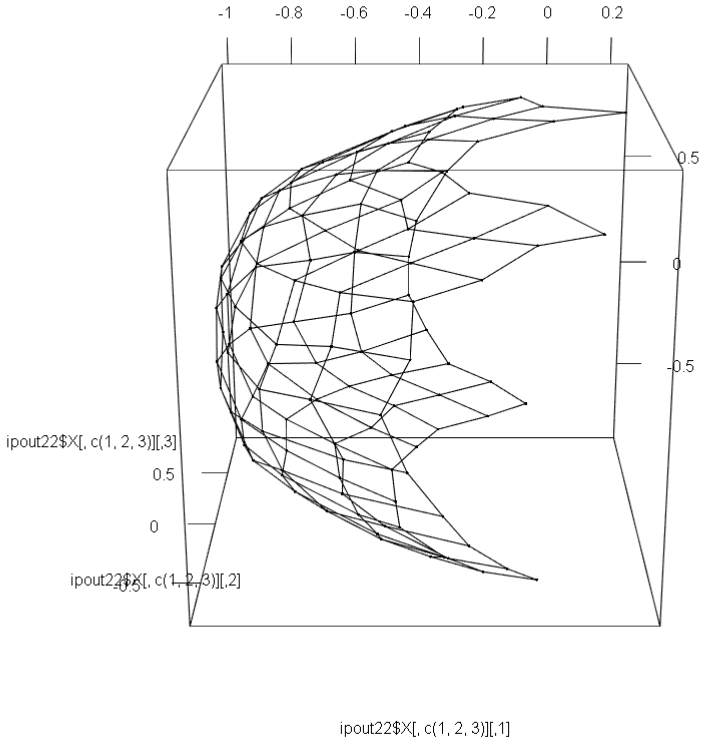

```
# IP行列
image(ipout22$P)
```



```
# 固有値
plot(ipout22$eout[[1]])
```




```
npt <- length(xy[,1])
plot3d(ipout22$X[,c(1,2,3)])
#spheres3d(ipout22$X[,c(1,2,npt)],radius=0.1)
el22 <- matrix(as.numeric(get.edgelist(g22)),ncol=2)
segments3d(ipout22$X[t(el22),c(1,2,3)])
```




```
newip <- ipout22$X %*% ipout22$M %*% t(ipout22$X)
newip[which(newip > 1)] <- 1
nn <- length(newip[,1])
newdsq <- matrix(0,nn,nn)
for(i in 1:nn){
  for(j in 1:nn){
    newdsq[i,j] <- acos(newip[i,j])
  }
}
range(newdsq^2*ipout22$r^2 - ipout22$D^2)
```

```
## [1] -5.402048e-05  5.616130e-11
```

```
#newip - ipout22$P
```

閉曲面メッシュグラフを作ってやってみる。

それなりのメッシュグラフを作る

```
library(devtools)
```

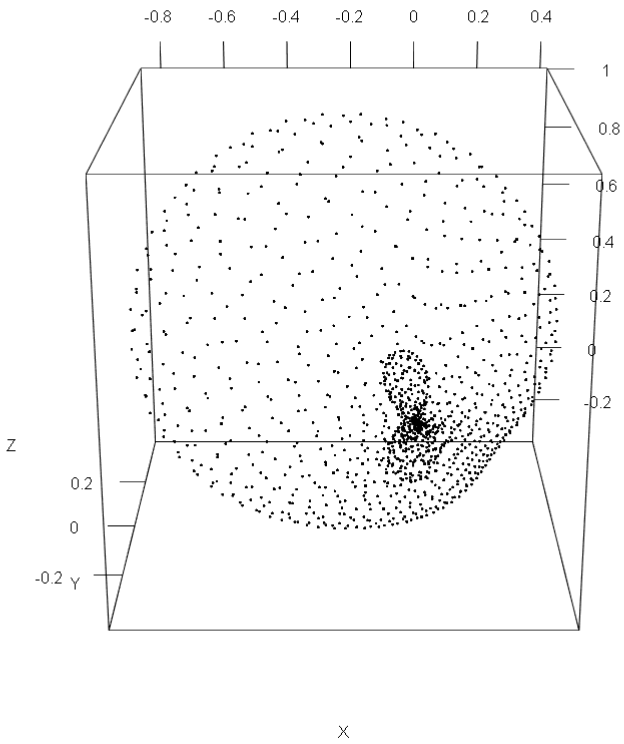
```
## Warning: package 'devtools' was built under R version 3.4.3
```

```
# install_github("ryamada22/Ronlyryamada") 初回はインストールする
library(Ronlyryamada)
library(RFOC)
```

```
## Warning: package 'RFOC' was built under R version 3.4.4
```

```
n <- 3 # メッシュの複雑さを指定(大きいと凹凸の周期が細かくなる)
k <- 1 # メッシュの複雑さを指定(大きいと真球に近くなる)
n.mesh <- 32 # メッシュの細かさを指定
A. <- matrix(runif(n^2), n, n)
A.[1, 1] <- k
A. <- A. + rnorm(n^2, 0, 0.05)
xxx <- my.spherical.harm.mesh(A = A., n = n.mesh)
```

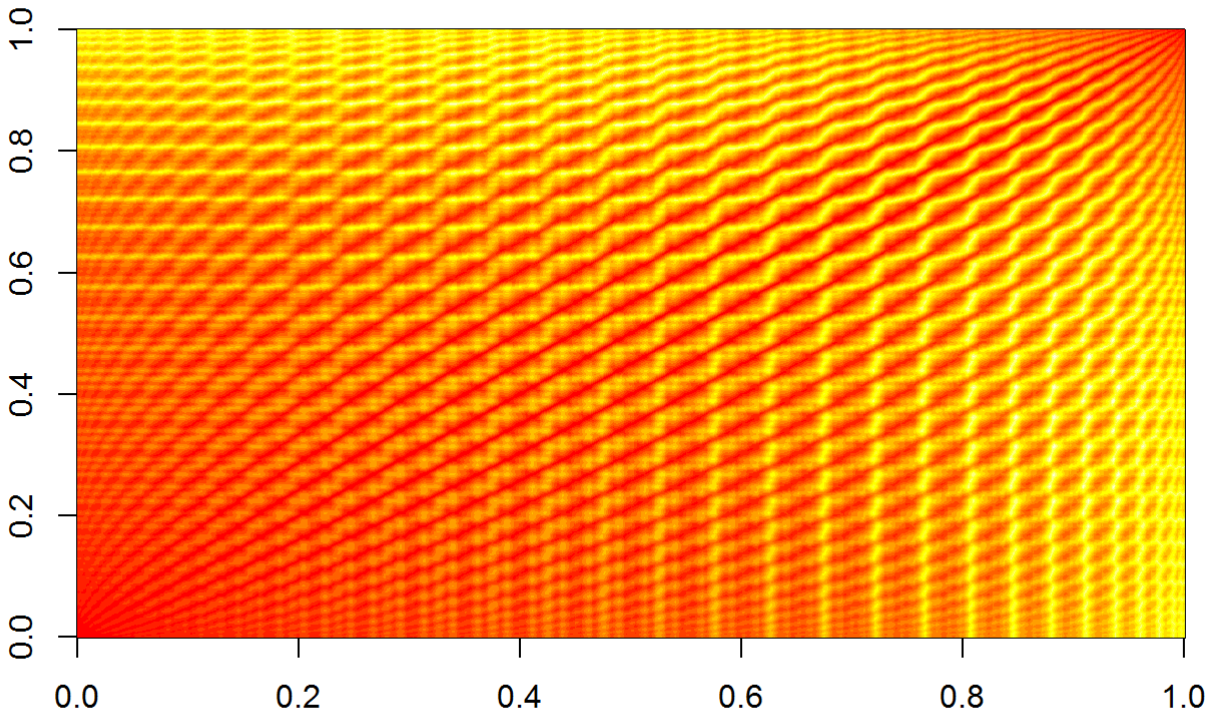
```
X <- xxx[[1]]
plot3d(X)
```



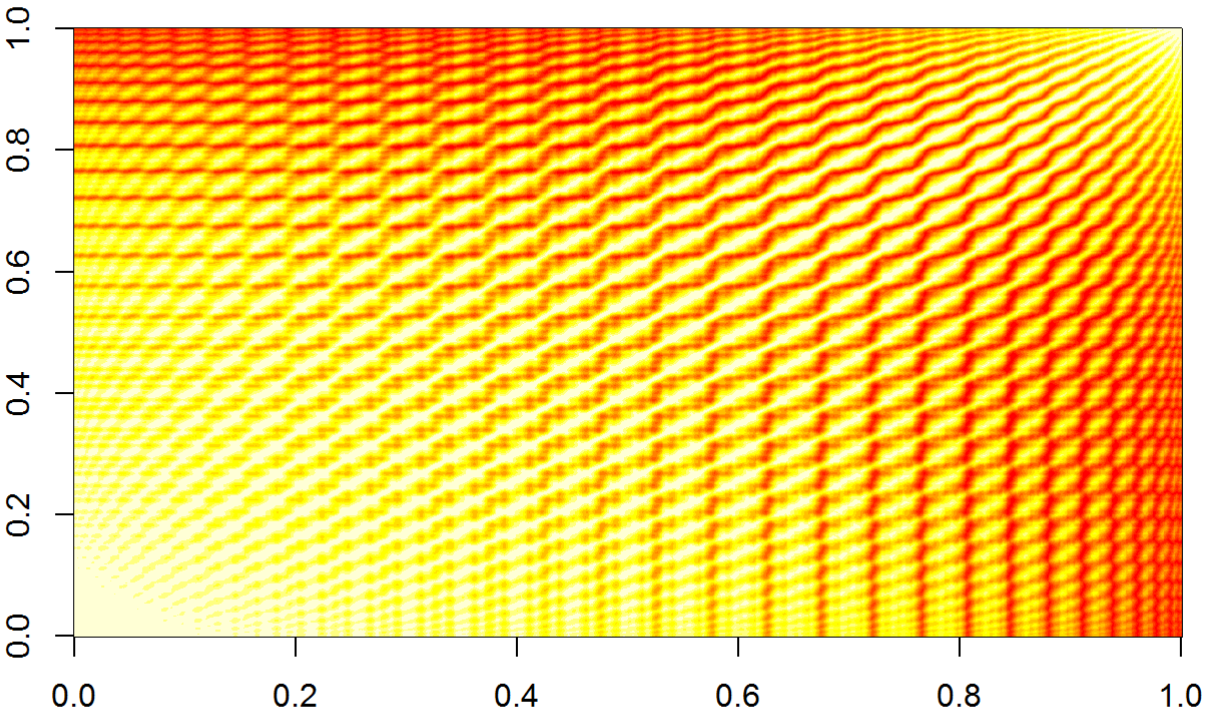

```
g5 <- graph.edgelist(xxx$edge,directed=FALSE)
#plot(g)

e.len5 <- rep(0,length(xxx$edge[,1]))
for(i in 1:length(e.len5)){
    e.len5[i] <- sqrt(sum((X[xxx$edge[i,1],]-X[xxx$edge[i,2],])^2))
}
```

```
ipout5 <- my.IPcoords2(g5,e.len5)
```

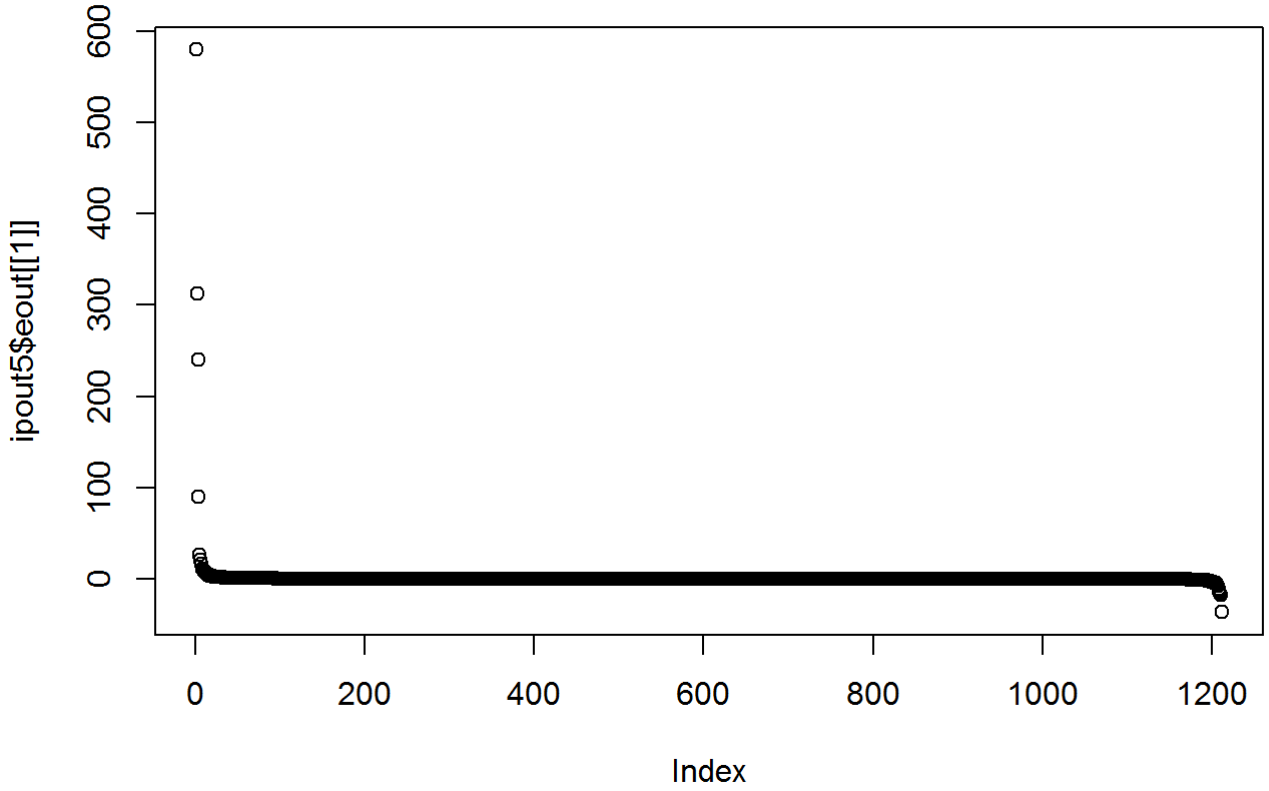
```
# 距離行列
image(ipout5$D)
```



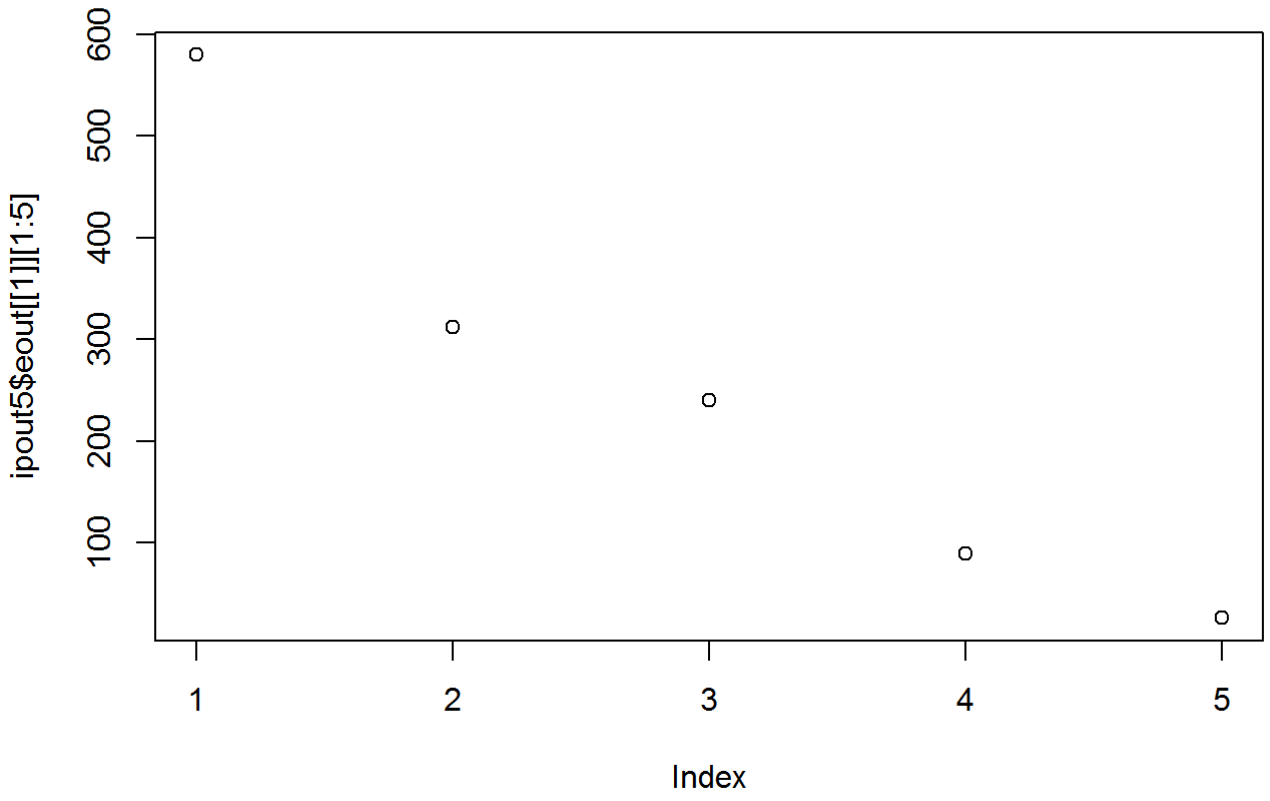

```
# IP行列
image(ipout5$P)
```



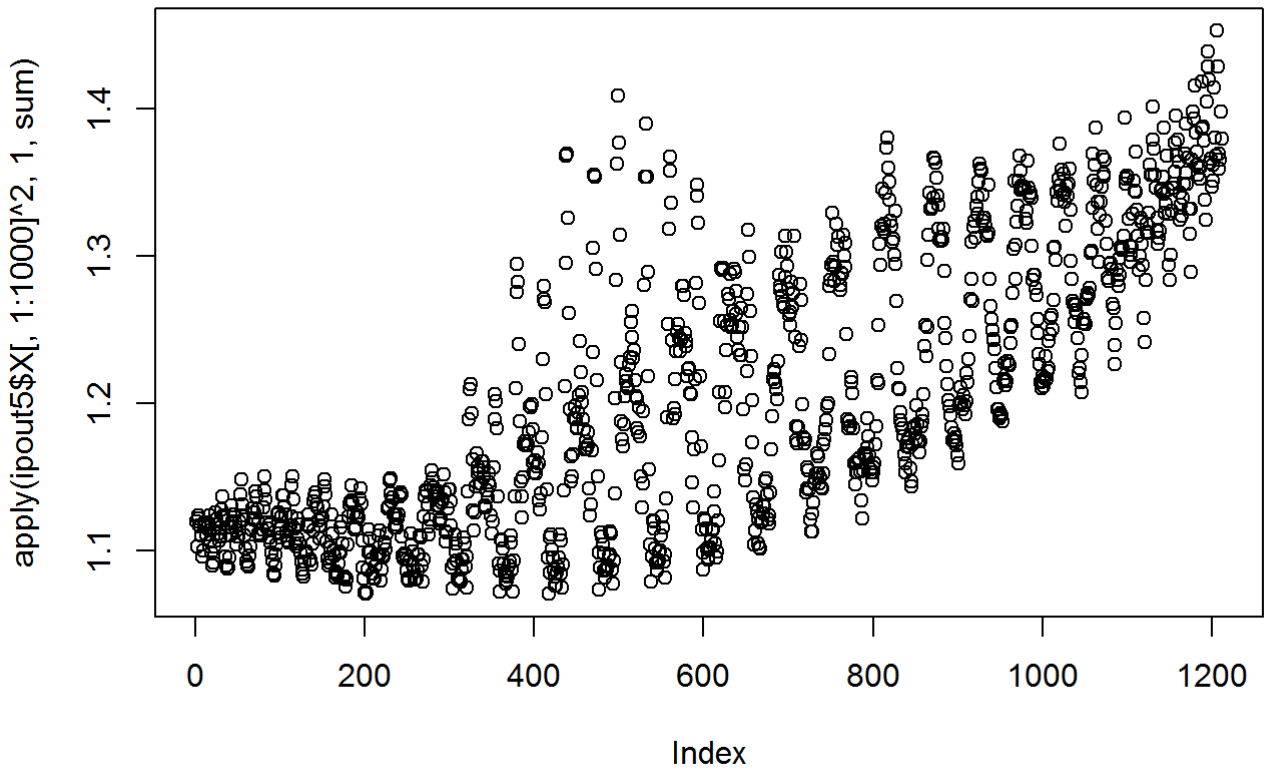

```
# 固有値
plot(ipout5$eout[[1]])
```



```
plot(ipout5$eout[[1]][1:5])
```

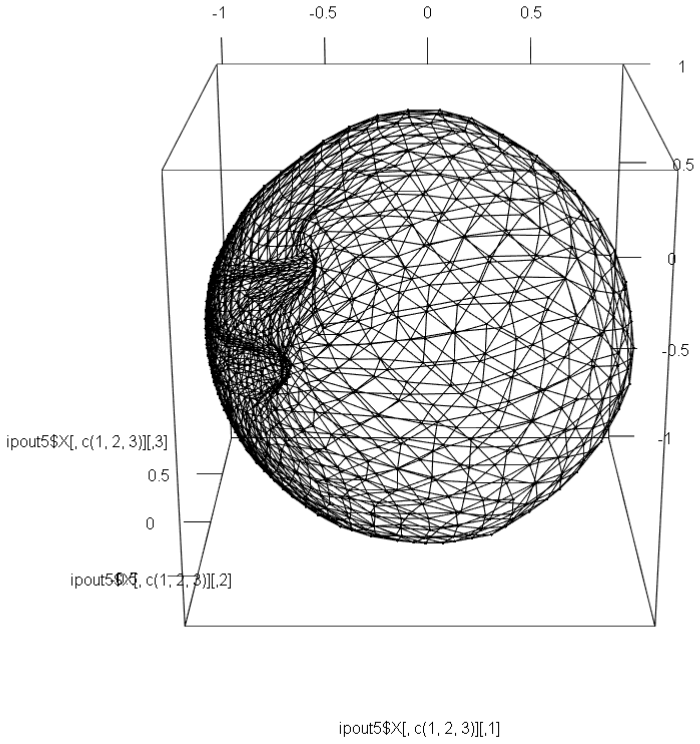



```
plot(apply(ipout5$X[,1:1000]^2,1,sum))
```

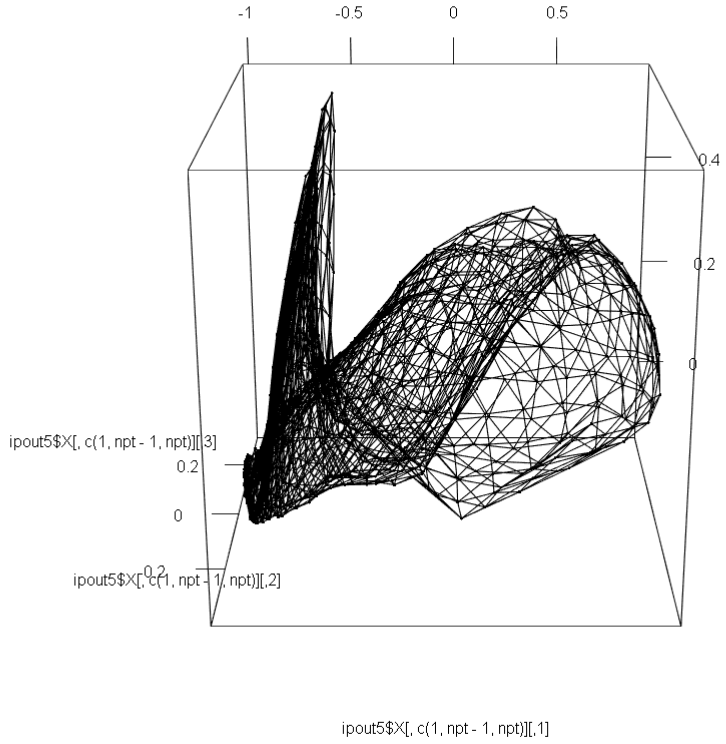


次の3dプロットは、一部の意味のある固有値成分しか使っていないことに注意。

```
npt <- length(ipout5$eout[[1]])
plot3d(ipout5$X[,c(1,2,3)])
#spheres3d(ipout5$X[,c(1,2,3)],radius=0.05)


el5 <- get.edgelist(g5)
segments3d(ipout5$X[t(el5),c(1,2,3)])
```




```
npt <- length(ipout5$eout[[1]])
plot3d(ipout5$X[,c(1,npt-1,npt)])
#spheres3d(ipout5$X[,c(1,npt-1,npt)],radius=0.05)


el5 <- get.edgelist(g5)
segments3d(ipout5$X[t(el5),c(1,npt-1,npt)])
```



内積行列Mで表されている空間での点がばらつくということは、負寄与方向の座標を使って、 局所の長さの測り方(内積・曲率の指標)が異なることを示していることになる。

したがって、今、行っている方法では、 現実空間での形は同じでも、そこへのメッシュの張り方を変えると、内積行列Mの空間での表現が変わることになる。 言い換えると、現実空間での点の取り方は等距離で取るのが良いのでは...。

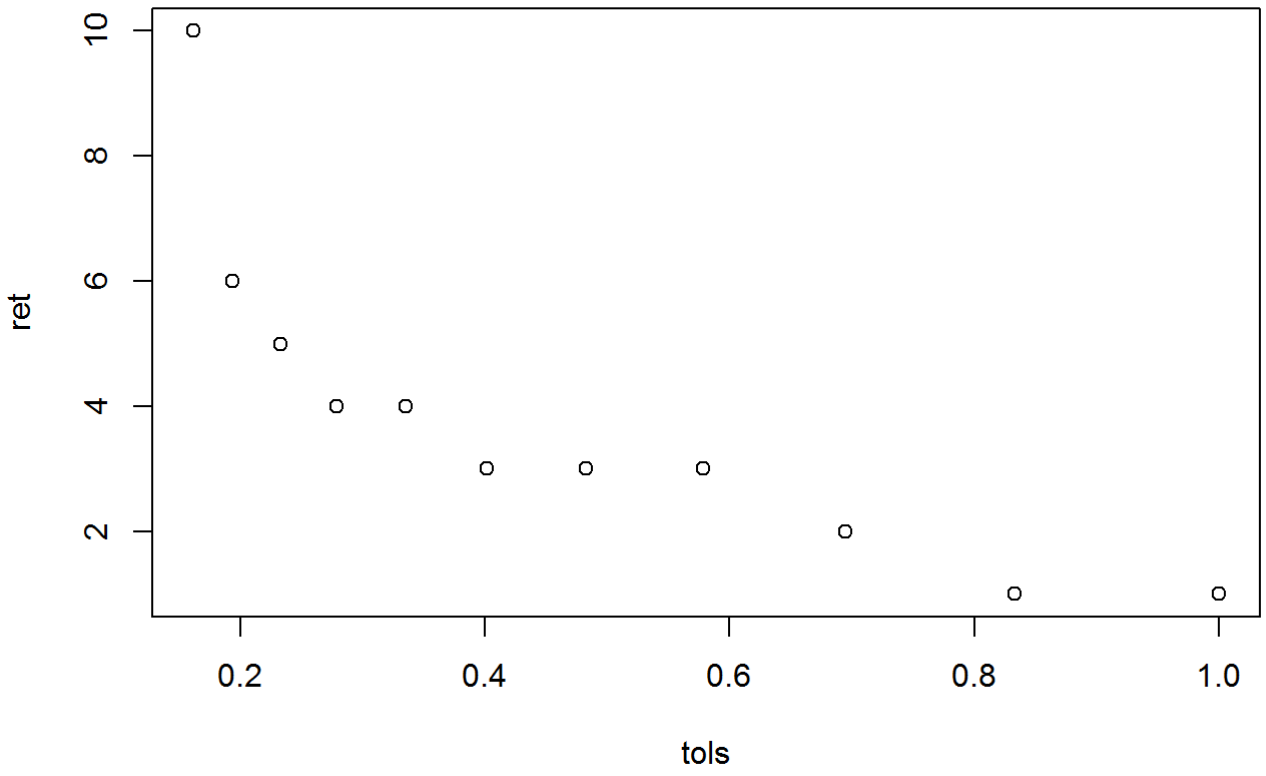
```
library(Matrix)
```

```
rankMatrix(ipout5$X,tol=0.5)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 0.5
```

```
tols <- 1.2^(0:(-10))
ret <- rep(0,length(tols))
for(i in 1:length(tols)){
  ret[i] <- rankMatrix(ipout5$X,tol=tols[i])
}
```

```
plot(tols,ret)
```




```
for(i in 1:(npt-2)){
  for(j in (i+1):(npt-1)){
    for(k in (j+1):npt){
      tmp <- ipout5$X[,c(i,j,k)]
      rk <- rankMatrix(tmp)
      if(rk<3){
        print(rk)
      }

    }
  }
}
```

現実空間で、エッジ距離が等長であるような場合

ボクセル集合の周囲を四角化グラフにしたものは、すべてのエッジの長さが等しい。

諸関数を作る

適当にボクセルリストを作って、使ってみる

```
n.step <- 1000
#xx0x <- matrix(rep(0,3),ncol=3)
xxx <- as.matrix(expand.grid(-1:1,-1:1,-1:1))
for(i in 1:n.step){
  pr <- apply(xxx^2,1,sum)
    r <- sample(1:length(xxx[,1]),1,prob=pr+0.1)
    #p <- sample(1:3,1)
    tmp <- xxx[r,]
    p <- sample(1:3,1)
    if(runif(1)<0.5){
      p <- order(tmp)[2]
    }

    tmp[p] <- tmp[p] + 1
    xxx <- rbind(xxx,tmp)
    xxx <- unique(xxx)
}
Vox.list <- unique(xxx)
quad <- my.vox2quad(Vox.list)
rootid <- 10
tr <- my.quad2tree(quad,rootid)
```

```
my.draw.surface.tree(tr)
```

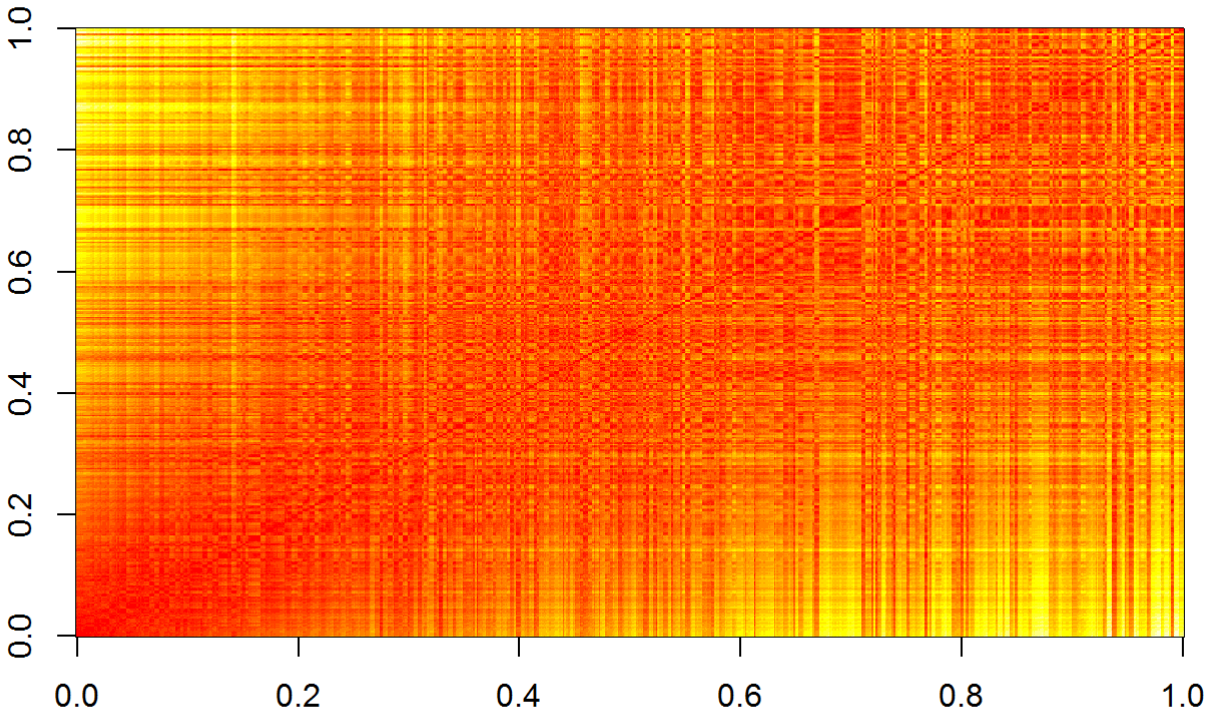
ルートノードからの距離に応じてノードに色を付けてみる。

```
for(i in 1:length(tr$quad$nodes[,1])){
  # グラフ距離を適当倍して大雑把にカラースケールが現れるようにする
  d <- floor(tr$rootdist[i] * 0.3) +1
  #print(d)
  spheres3d(tr$quad$nodes[i,],col=d,radius=0.1)
}
```

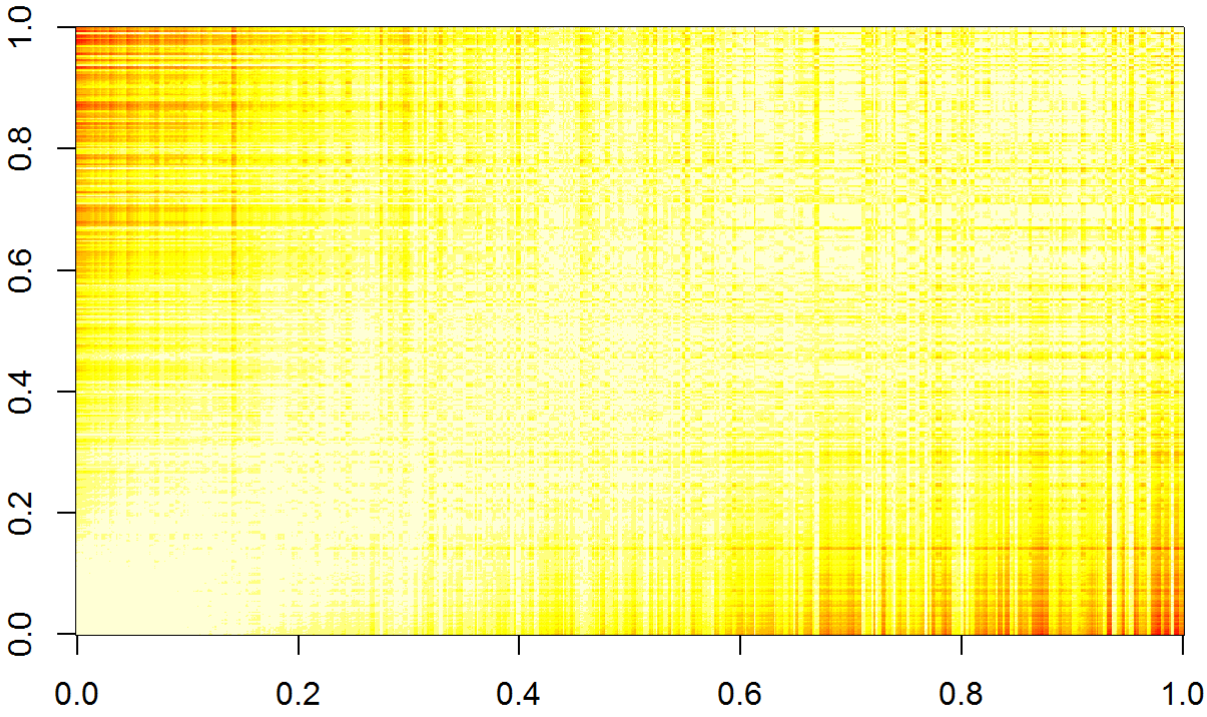
```
g6 <- quad$g
e.len6 <- rep(1,length(E(g6)))
```

```
ipout6 <- my.IPcoords(g6,e.len6)
```

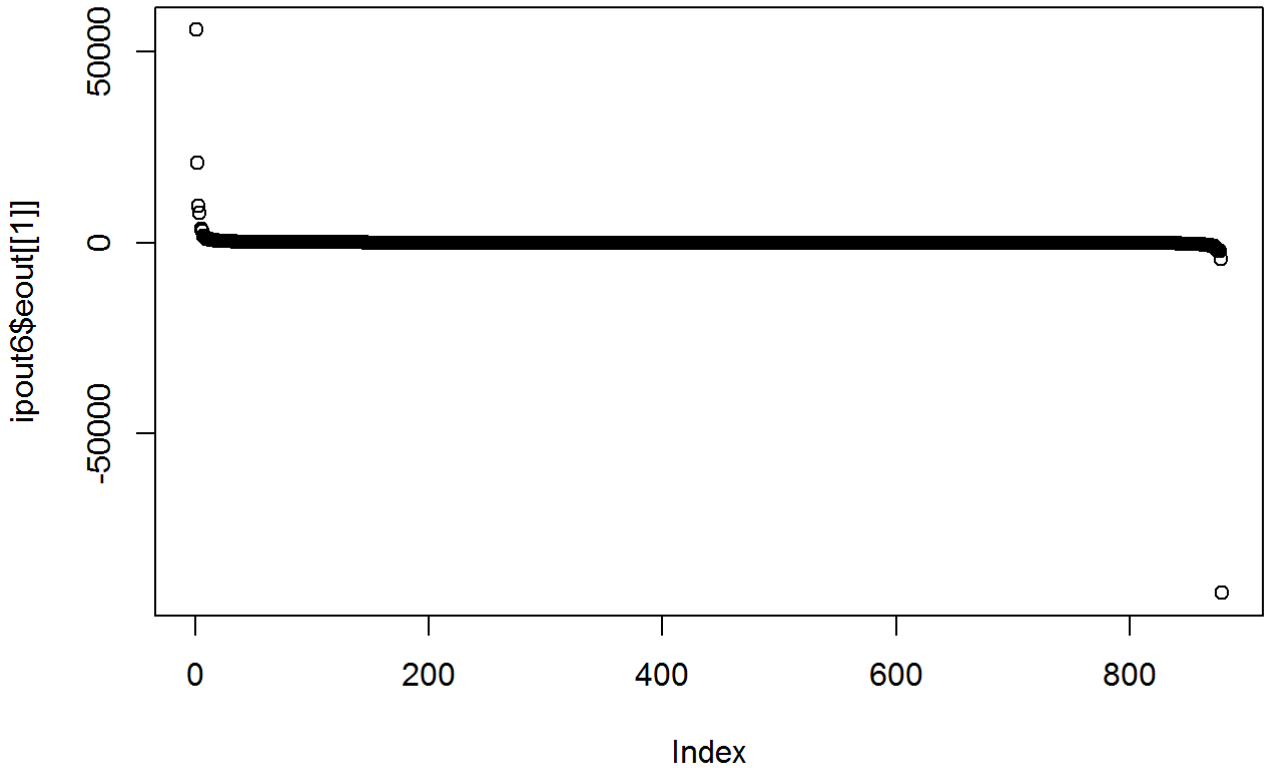
```
# 距離行列
image(ipout6$D)
```



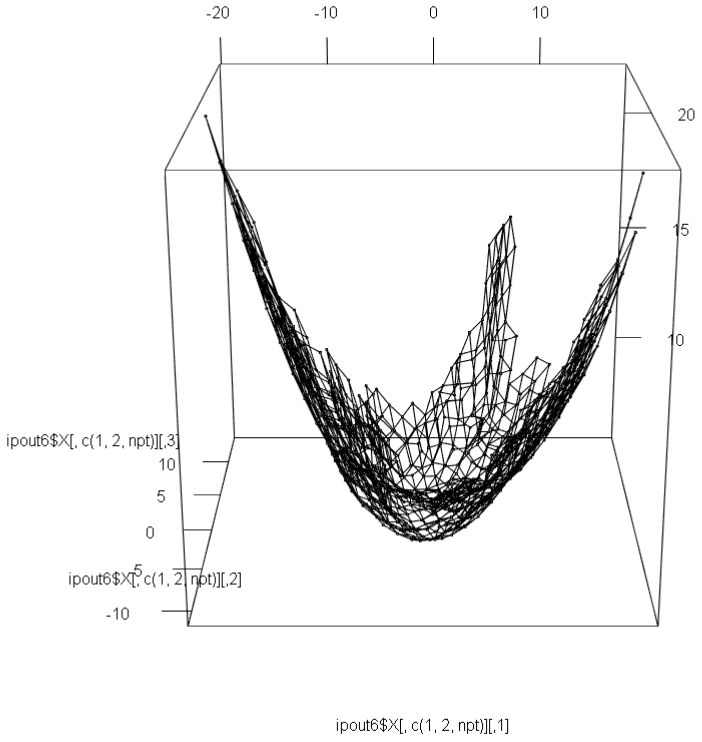

```
# IP行列
image(ipout6$P)
```



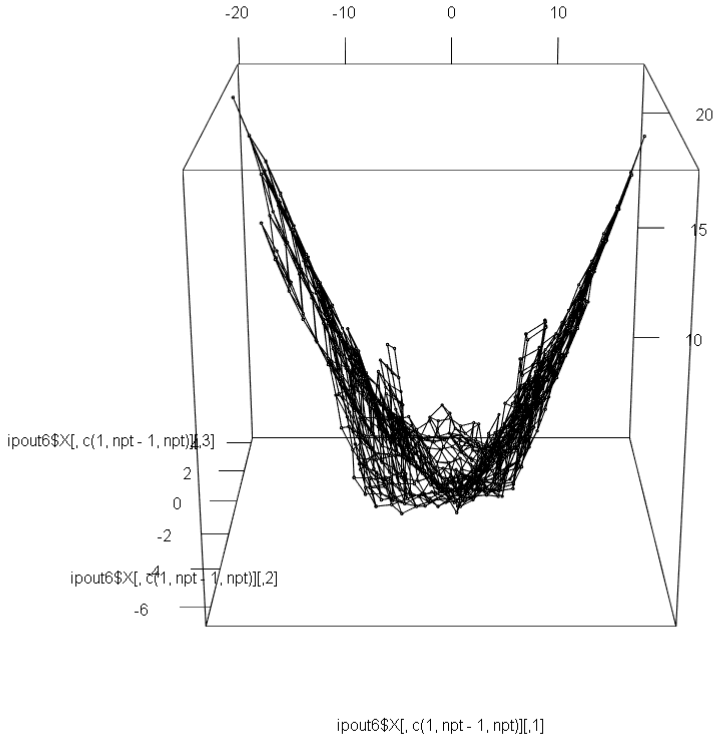
```
# 固有値
plot(ipout6$eout[[1]])
```




```
npt <- length(ipout6$eout[[1]])
plot3d(ipout6$X[,c(1,2,npt)])
spheres3d(ipout6$X[,c(1,2,npt)],radius=0.1)
el6 <- get.edgelist(g6)
segments3d(ipout6$X[t(el6),c(1,2,npt)])
```




```
npt <- length(ipout6$eout[[1]])
plot3d(ipout6$X[,c(1,npt-1,npt)])
spheres3d(ipout6$X[,c(1,npt-1,npt)],radius=0.1)
el6 <- get.edgelist(g6)
segments3d(ipout6$X[t(el6),c(1,npt-1,npt)])
```


-20
-10
0
10
20
15
10
ipout6$X[, c(1, npt - 1, npt)][,3]
2
0
-2
-4
-6
ipout6$X[, c(1, npt - 1, npt)][,2]
ipout6$X[, c(1, npt - 1, npt)][,1]