非正定値内積行列で座標化する

ryamada

*2018*年*5*月*26*日

内積行列

複数の分布を観察したとき、それらのペアワイズ内積を推定することができる。

ペアワイズ内積は内積行列Pとして表せる。

と固有値分解できて

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

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は固有値の符号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。

今、Pが正定値であるとき、固有値の符号行列$S= I $(単位行列)であるので

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

と簡単になっているから、観測された各分布に

$$(V\sqrt{\Sigma})$$

という座標を与えれば、分布を空間の点として与えることができて、

分布間の距離は、分布の位置座標の差ベクトルの２乗ノルムを使って計算できる。

$$(x_1 - x_2)^T (x_1 - x_2)$$

一方、Pが非正定値行列であるとき、符号行列Sには-1も含まれるので、

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

各分布には

$$(V\sqrt{|\Sigma|})$$

なる座標を与えることとなり、

この座標空間では、内積を

$$(V_1\sqrt{|\Sigma|})^T S (V_2\sqrt{|\Sigma|})$$

のように、単位行列ではない対角行列(S, ただし対角成分は± 1)を用いて定義してあることになっている。

ただし、内積の定義がSではあるが、各分布にはきちんと座標があるし、

点間距離は、相変わらず、この内積定義行列を使って

$$(x_1 - x_2)^T S (x_1 - x_2) = x_1^T S x_1 + x_2^T S X_2 - 2 x_1^T S x_2$$

と定まっている。

実際、この計算を指数型分布族表現にあてはめると、各分布のポテンシャル関数の値は、内積定義行列Sの下での、「原点」からの距離になるのですね。

そして、２分布間の距離は、分布の位置座標から作った差ベクトルのポテンシャル関数の値が、分布間の「距離」になってもいるようです。

何が嬉しい処理なのか？

普通に平な空間では、座標を決めて、単位行列が定める内積で、すべての点の間の距離が定まっている。

逆に言うと、単位行列が距離を定めているのが、「普通の空間」である。

他方、分布は、「普通ではない空間」に広がっている。

ただし、分布がk個のパラメタで表せるとき、分布はk次元多様体として広がっている。

k次元多様体は曲がっているので、曲がった空間のまま、k次元多様体の上を辿って距離を測ろうとすると、曲がった空間に巻尺を持ち出して、距離を測る(局所の曲がり具合を考慮しながら積分する)必要があるのがふつうである。

ただし、-1を含む内積行列Sを持ち出すことで、分布に座標を与えると、その曲がった空間上で積分して初めて求まるはずの「多様体に沿った距離」が、座標を使った計算

$$(x_1 - x_2)^T S (x_1 - x_2) = x_1^T S x_1 + x_2^T S X_2 - 2 x_1^T S x_2$$

だけでできることになる。

どうやら、これが、非正定値内積を用いて座標化することの嬉しさ、のようである。

内積行列を推定してみたところ、正定値であることがわかれば、対象は、「普通に平坦な」広がりを持つk次元多様体であることがわかるし、

非正定値であることがわかれば、意味のある絶対値を有する固有値の数だけ複雑にのたくった多様体を、その分布たちが構成していることを意味しているらしい。

以下は、この非正定内積行列を、「形表面の点たちの間のグラフ距離(表面に沿って辿った最短距離の行列)」に適用すると、形の複雑さと行列Sとの間に関連があるらしい、という、形プロジェクト系のディスカッションのメモです。

参考まで。

本題～形と非正定値内積行列

曲線・曲面上の点間距離が知りたいが、そのためには、「最短経路を割り出し」て、最短経路にそって巻尺を這わせて測定する必要がある。

グラフになっていれば、グラフ上の最短距離を求めるアルゴリズムを回すことがそれに相当する。

いずれにしろ、点の並び具合と、隣り合う点の間の距離を「積分」することが必要。

『普通の平面』ではそんな面倒なことはしなくてよい。

点に座標を与えて、

$$(x_1 - x_2)^T (x_1 - x_2) = (x_1 - x_2)^T I (x_1 - x_2)$$

とすれば距離の２乗が求められる。

なぜなら、平面ではいたるところで内積が単位行列 Iとなっているから。

曲線・曲面では場所ごとに内積を定める行列が異なっているので、うまく行かない。

『非正定値内積行列で座標化する』は、$(x_1 - x_2)^T M(x_1 - x_2)$ によって、いたるところの点同士の距離が座標から直接求められるようにする作戦である。

積分しなくてよいようにする作戦である。

逆に言うと、曲がっているので、積分することが必要な情報だったもの(グラフそのもの)に、自由度を与えて、積分しないで、すべての点間関係(多様体全体の様子)を簡単に取り出せる表現に改めること、そして、そのような表現が持つ特徴を使って、多様体の様子を記述する(比較・分類したりする)作戦である。

やってみよう。

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。

$$(x_1 - x_2)^T M(x_1 - x_2) = x_1^T M x_1 + x_2^T M X_2 - 2x_1^T M x_2$$

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

ここですべての点で$x^T Mx = 1$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

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

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)
}
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))
}
```

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
```

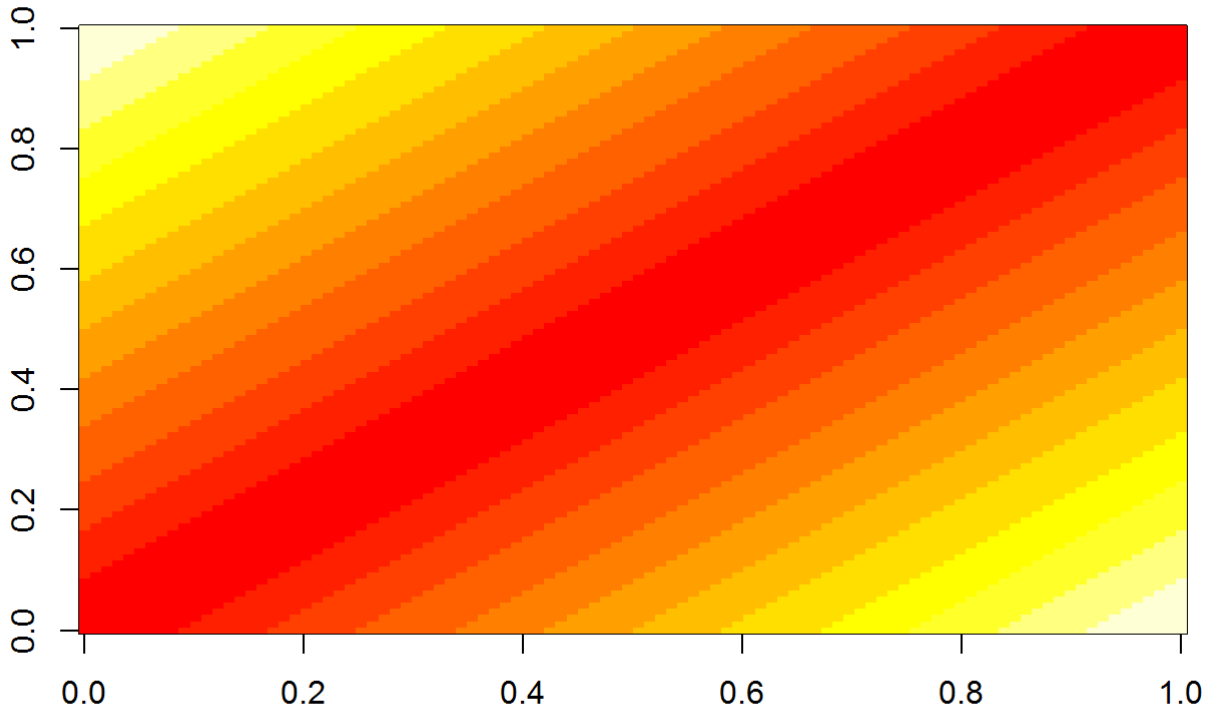
```
##
## Attaching package: 'igraph'
```

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

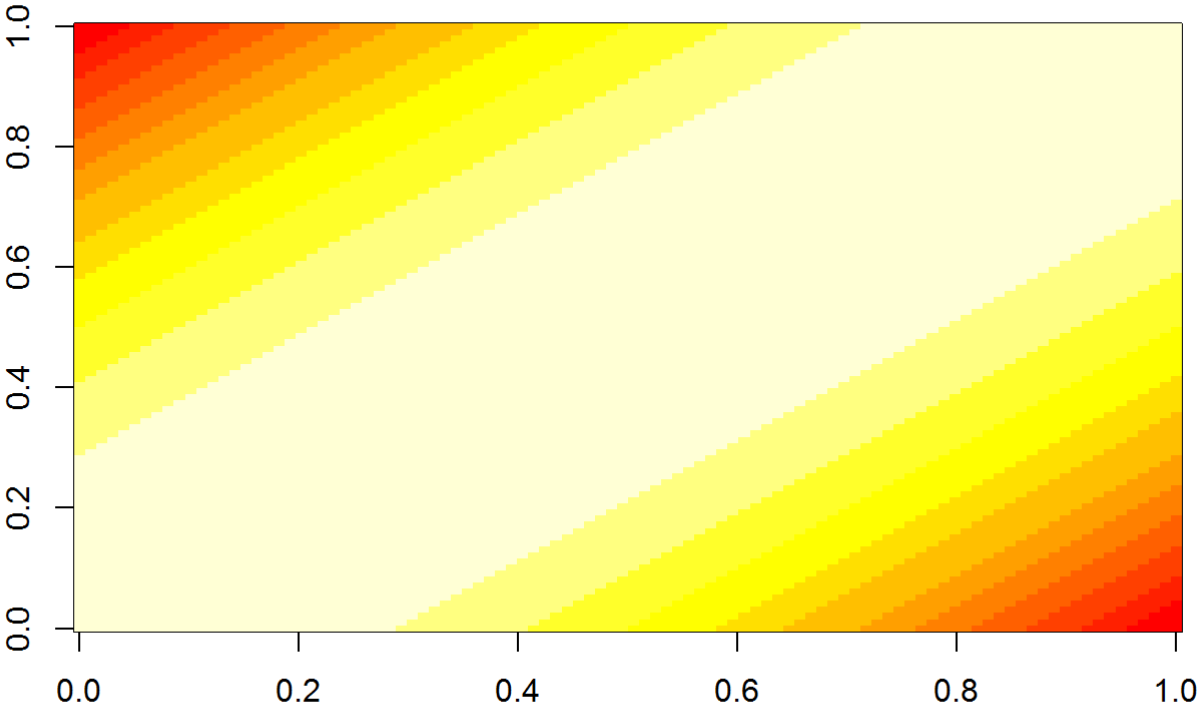
```
## The following object is masked from 'package:base':
##
##     union
```

```
# 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]))
```

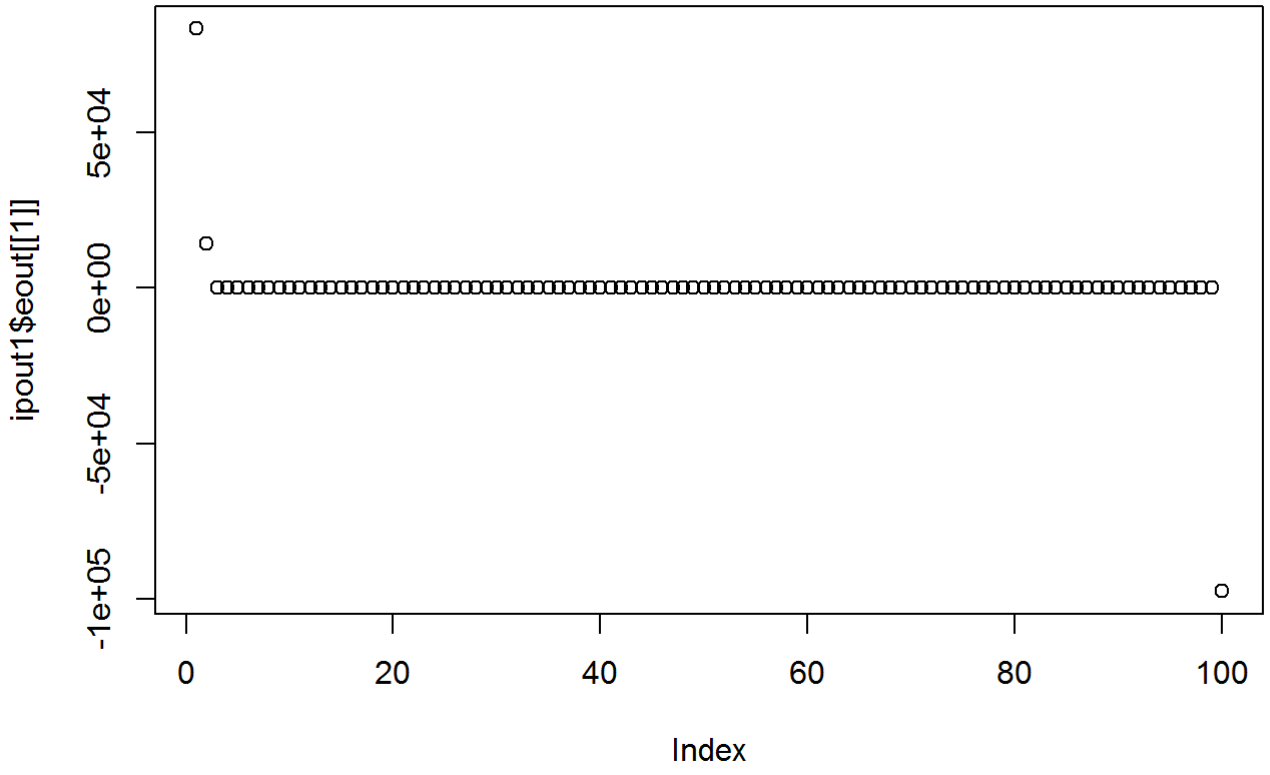
```
ipout1 <- my.IPcoords(g1,e.len1)
# 距離行列
image(ipout1$D)
```



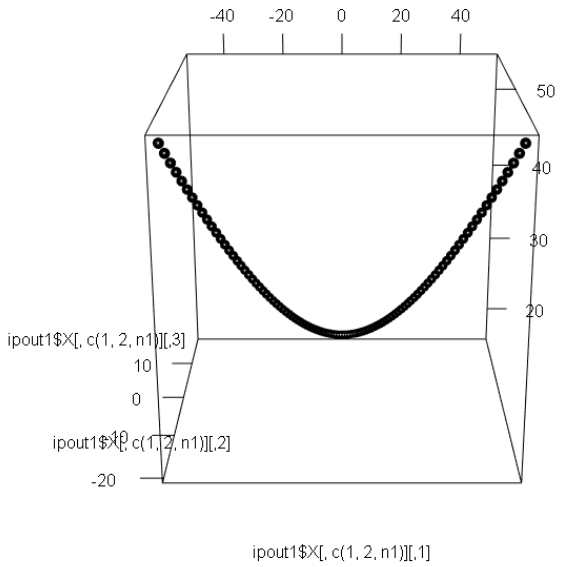

```
# IP行列
image(ipout1$P)
```



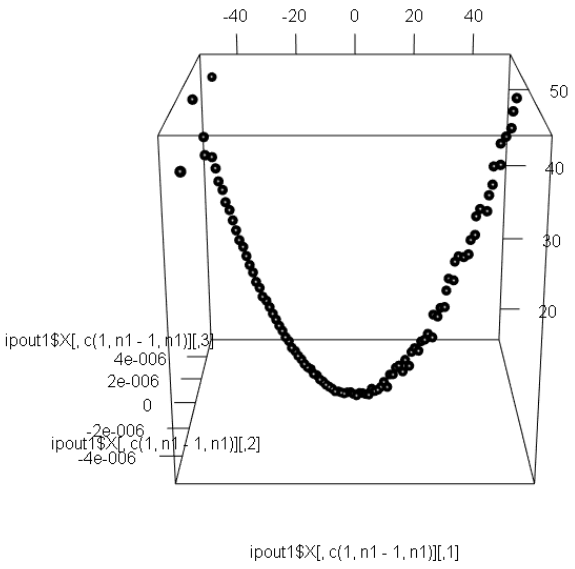

```
# 固有値
plot(ipout1$eout[[1]])
```



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

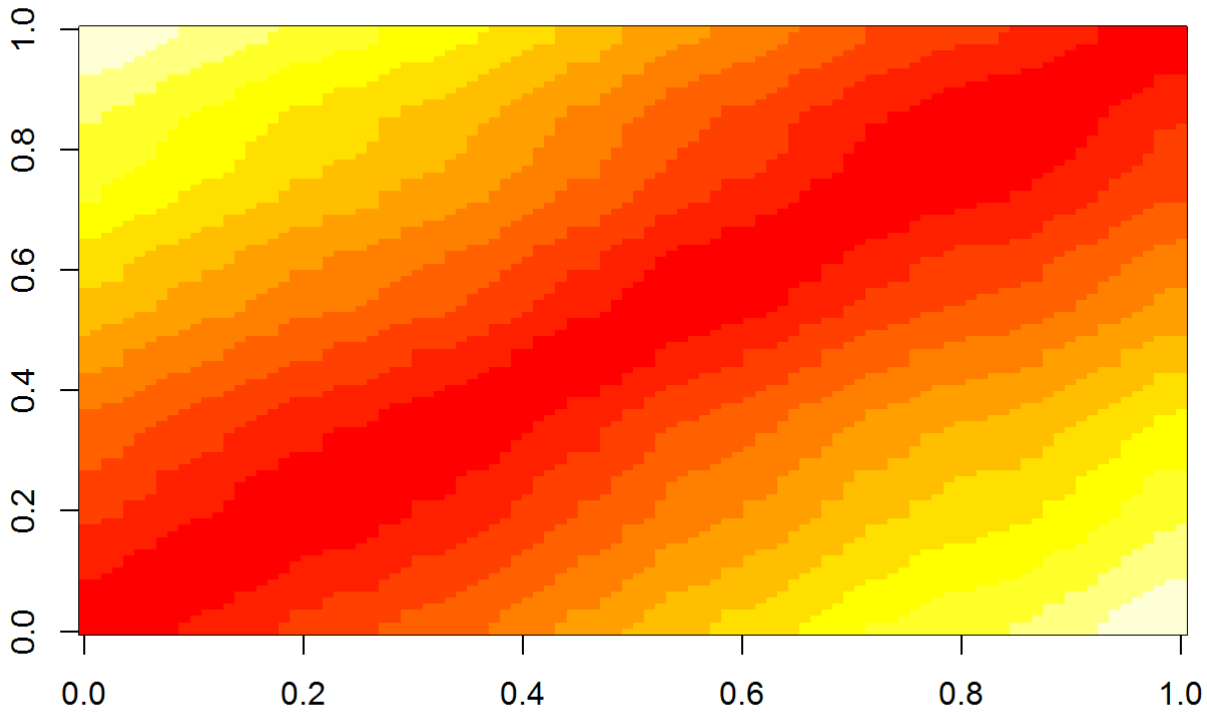



```
plot3d(ipout1$X[,c(1,n1-1,n1)])
spheres3d(ipout1$X[,c(1,n1-1,n1)],radius=1)
```

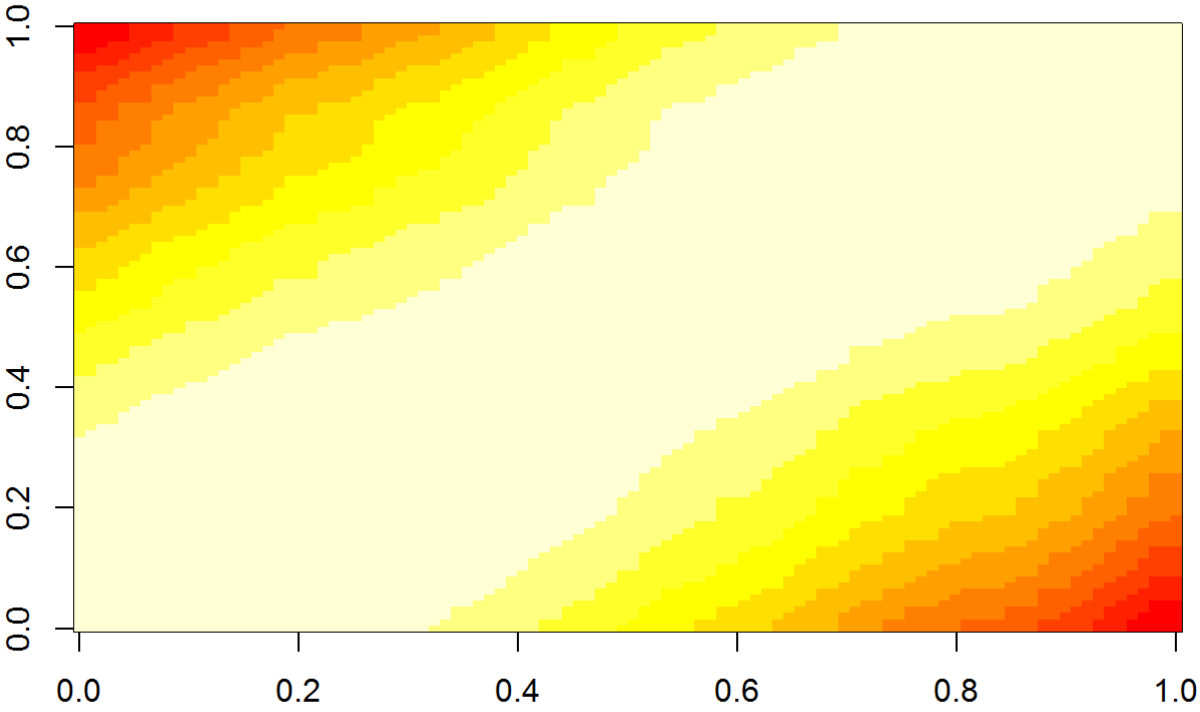


エッジ長がばらばら

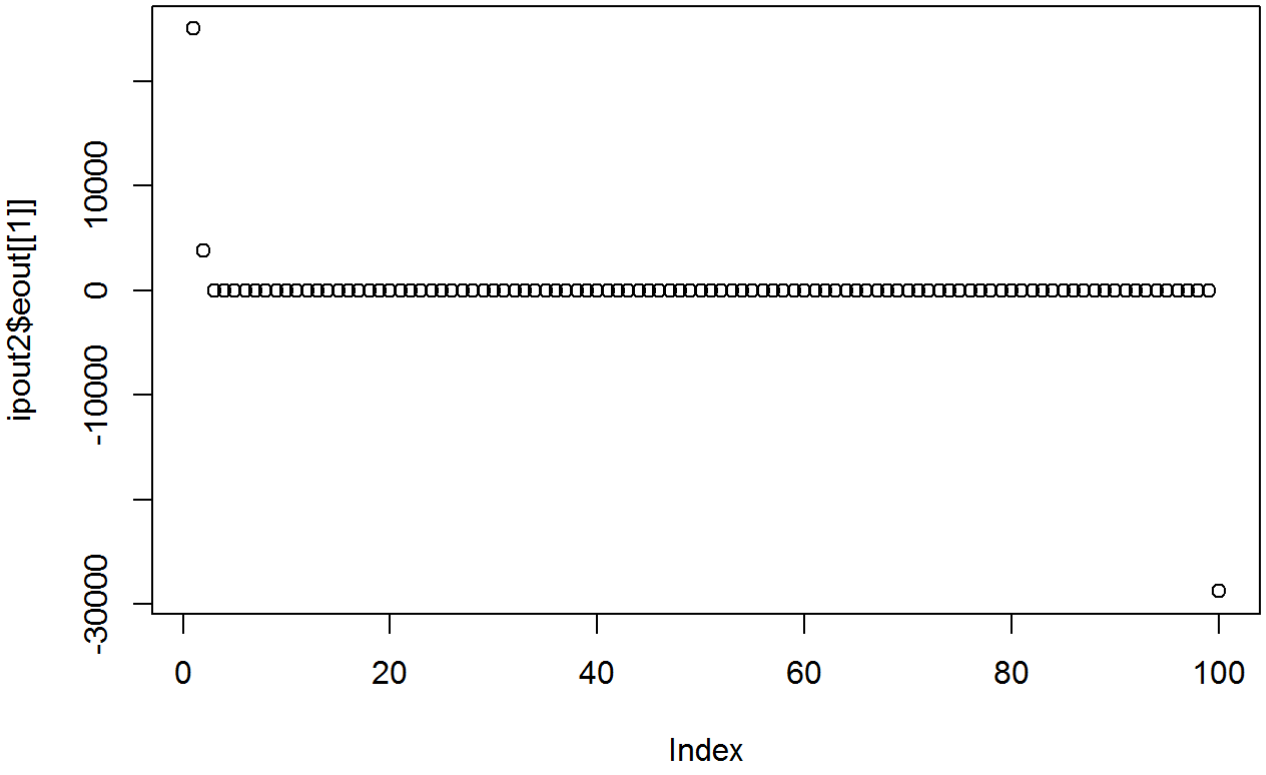
```
# 1次元グラフ
library(igraph)
# 頂点数
n2 <- 100
el2 <- cbind(1:(n2-1),2:n2)
# グラフオブジェクト
g2 <- graph.edgelist(el2,directed=FALSE)
# エッジ長をばらばらにする
e.len2 <- runif(length(el2[,1]))

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




```
# IP行列
image(ipout2$P)
```

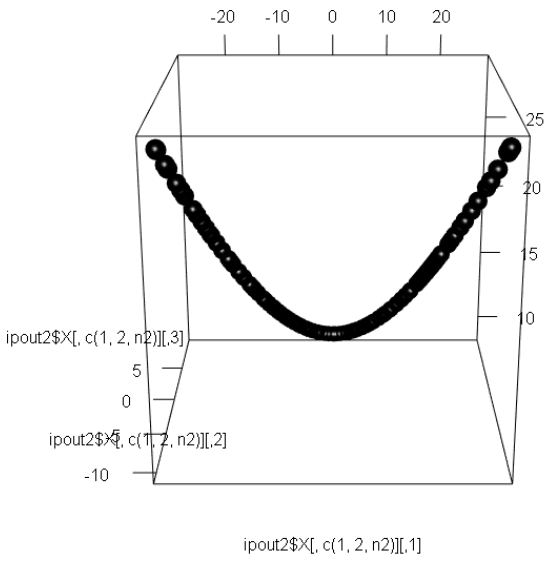



```
# 固有値
plot(ipout2$eout[[1]])
```



現れるカーブは同じだが、打点の間隔が異なっている。

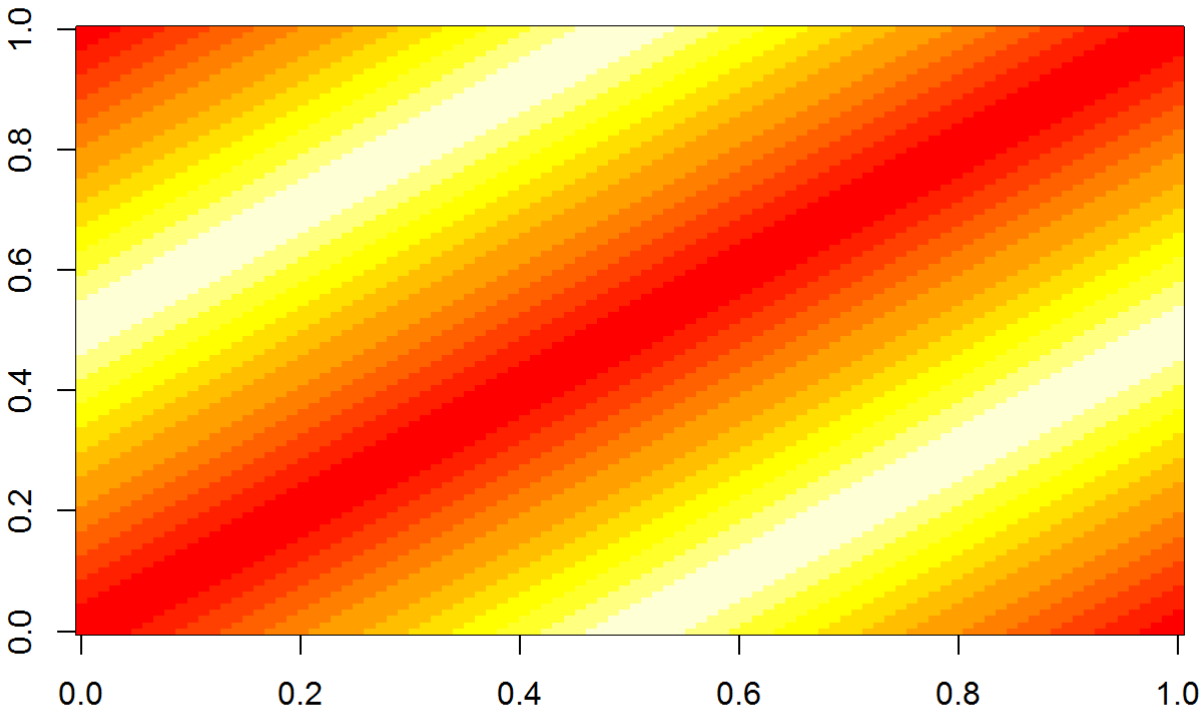
```
plot3d(ipout2$X[,c(1,2,n2)])
spheres3d(ipout2$X[,c(1,2,n2)],radius=1)
```



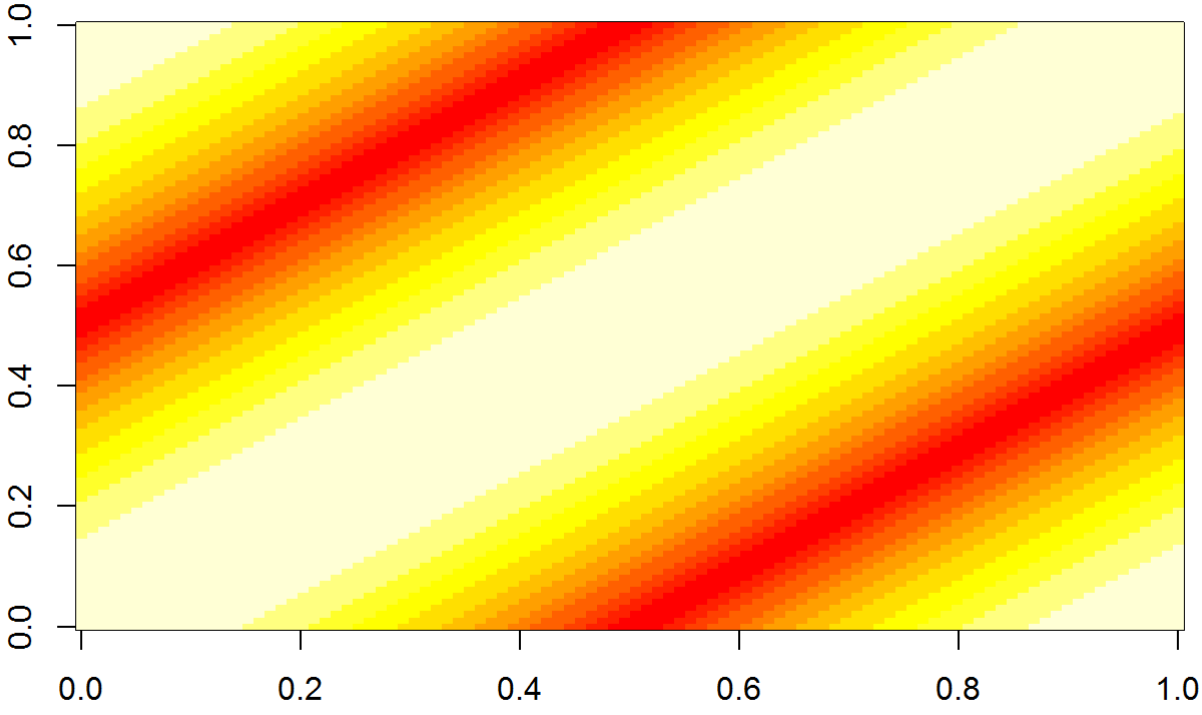
エッジ長がすべて等しい単純な周回グラフ

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。

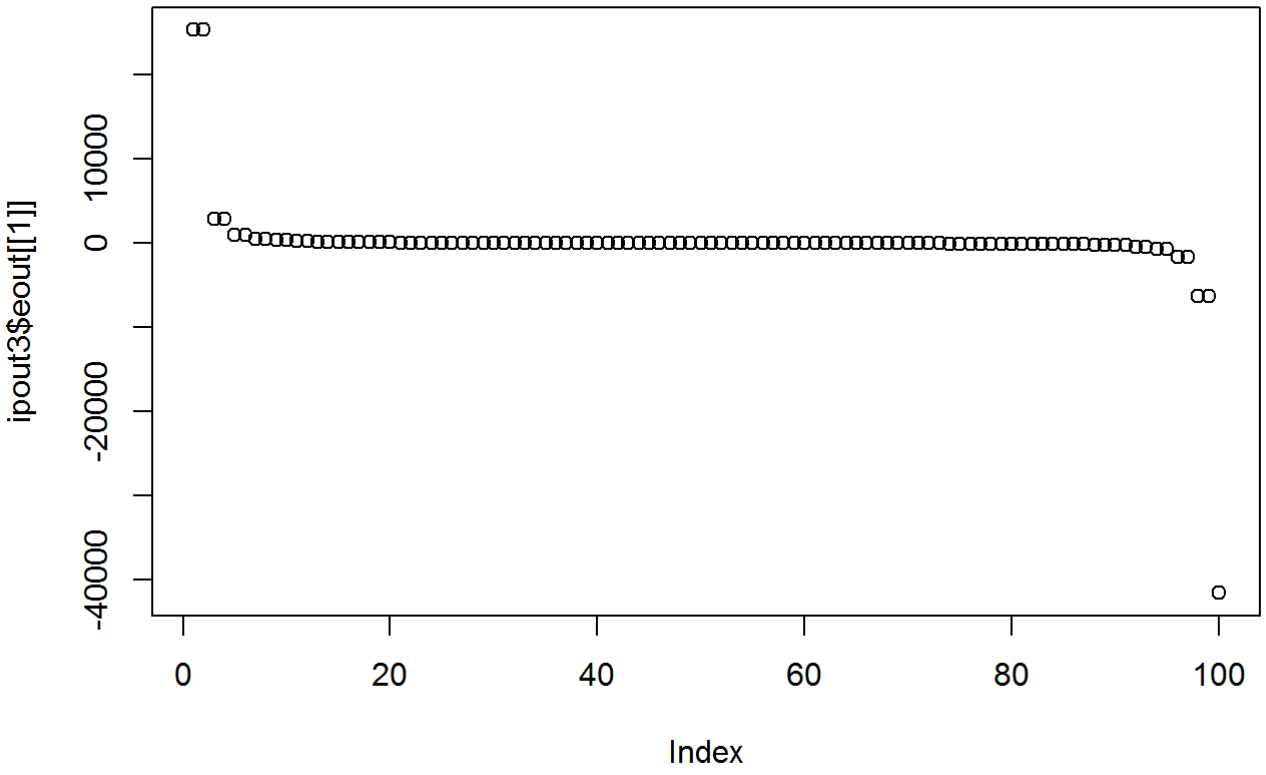
```
# 円周グラフ
# 頂点数
n3 <- 100
el3 <- cbind(1:(n3-1),2:n3)
el3 <- rbind(el3,c(n3,1))
# グラフオブジェクト
g3 <- graph.edgelist(el3,directed=FALSE)
# エッジ長
e.len3 <- rep(1,length(el3[,1]))
ipout3 <- my.IPcoords(g3,e.len3)
# 距離行列
image(ipout3$D)
```



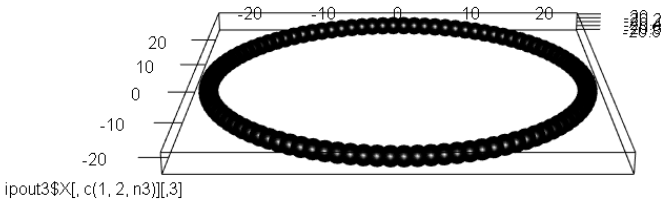

```
# IP行列
image(ipout3$P)
```

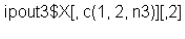


```
# 固有値
plot(ipout3$eout[[1]])
```




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

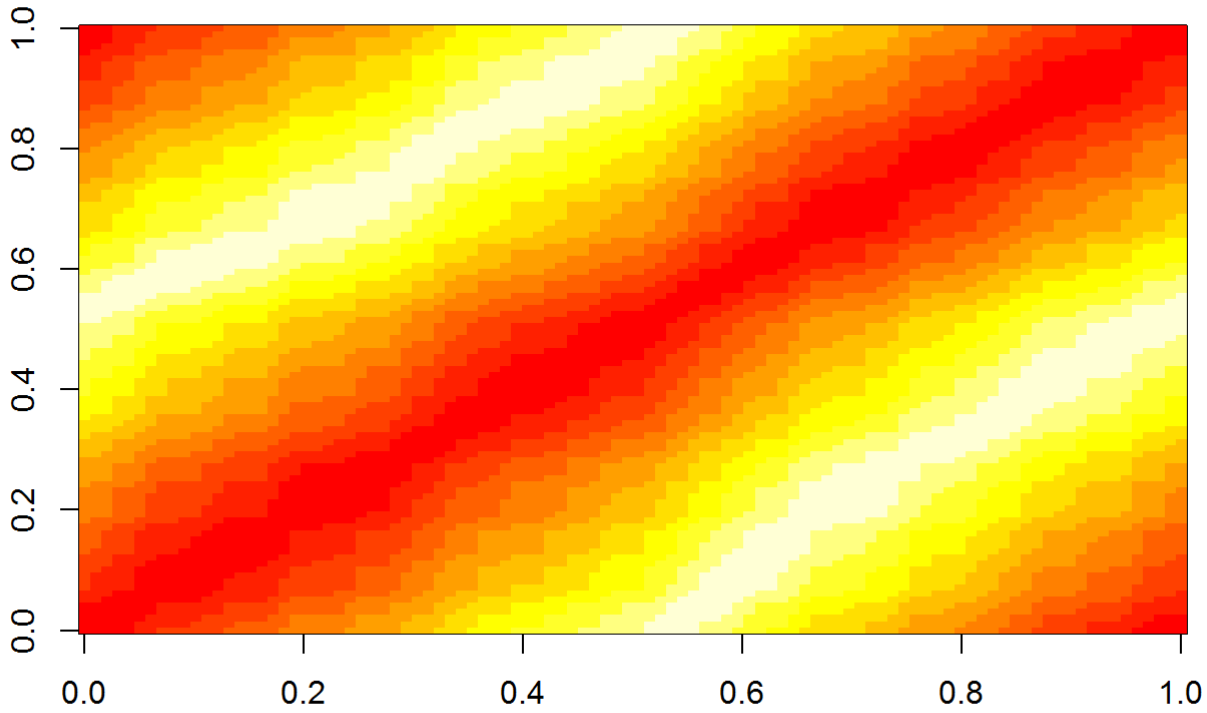




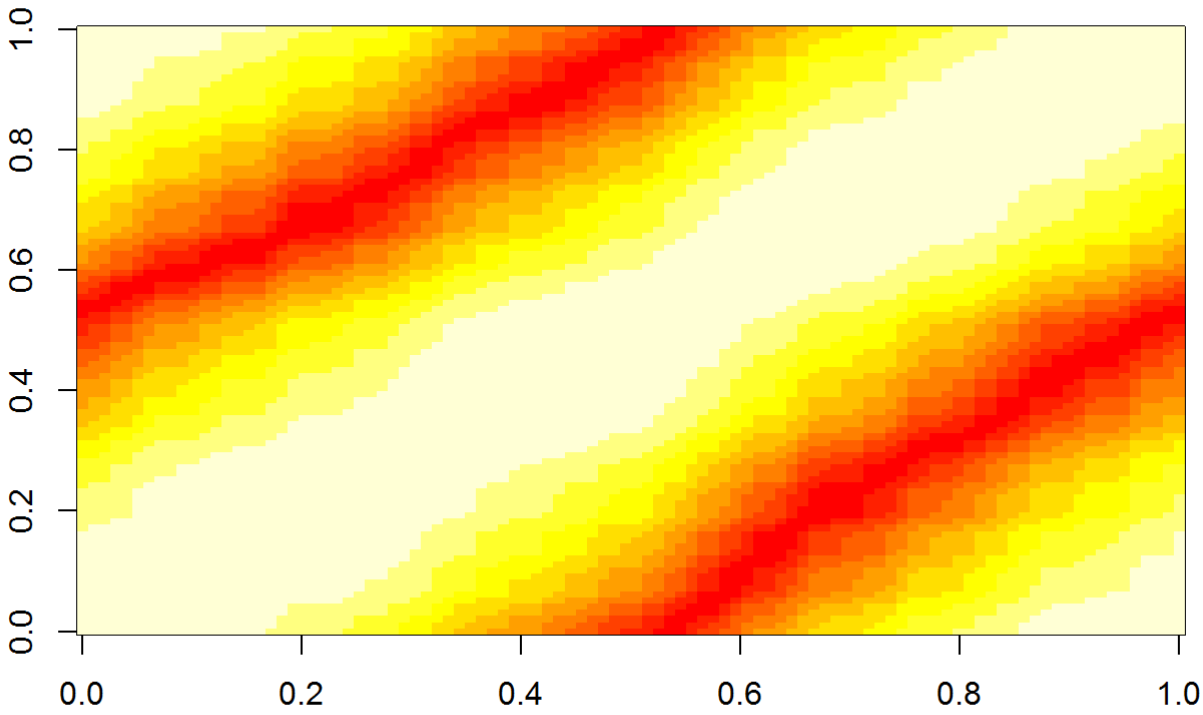
エッジ長をバラバラにする

エッジのばらばら加減が座標に反映する

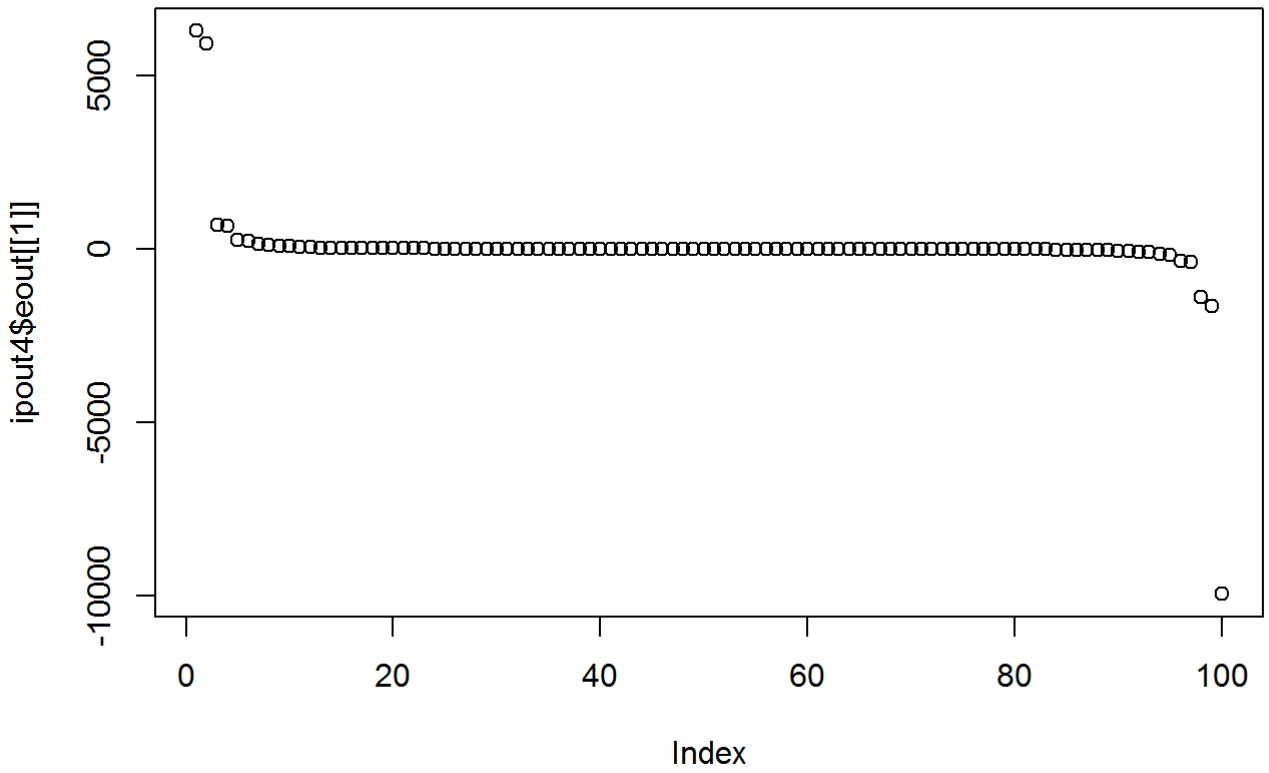
```
# 円周グラフ
# 頂点数
n4 <- 100
el4 <- cbind(1:(n4-1),2:n4)
el4 <- rbind(el4,c(n4,1))
# グラフオブジェクト
g4 <- graph.edgelist(el4,directed=FALSE)
# エッジ長
e.len4 <- runif(length(el4[,1]))
ipout4 <- my.IPcoords(g4,e.len4)
# 距離行列
image(ipout4$D)
```




```
# IP行列
image(ipout4$P)
```

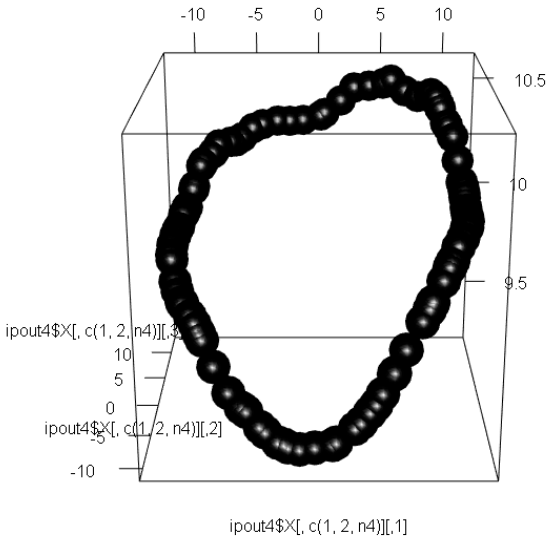



```
# 固有値
plot(ipout4$eout[[1]])
```



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。

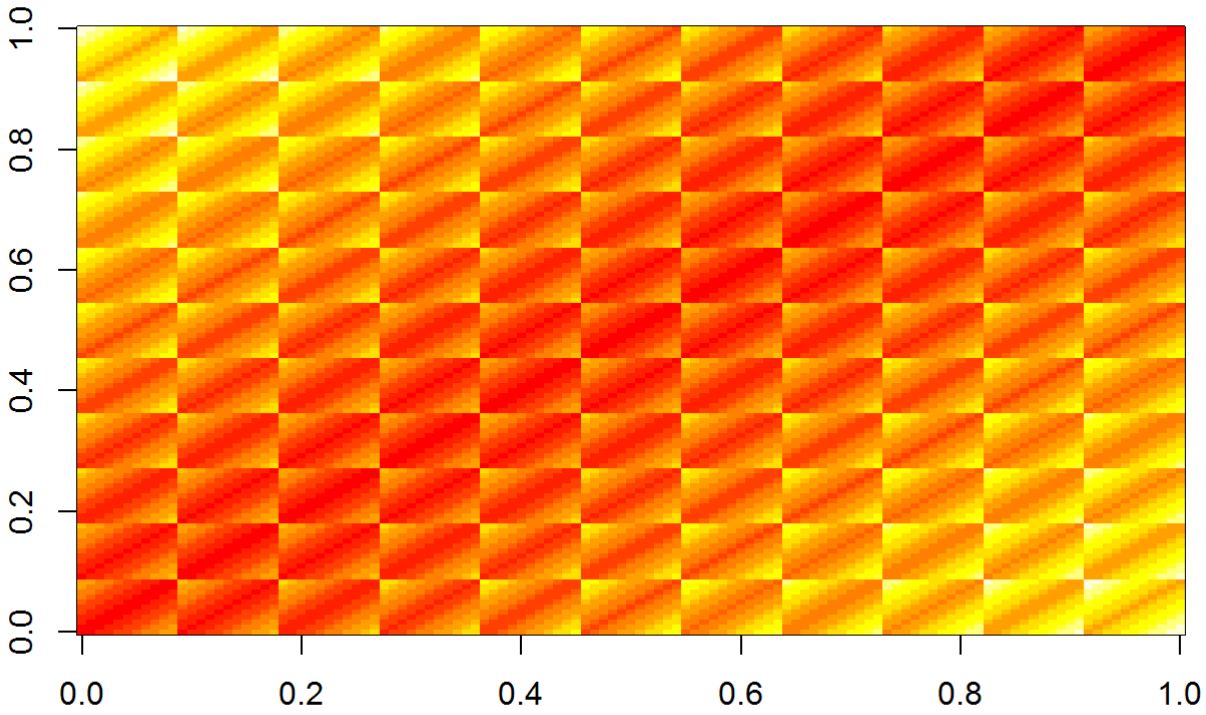
```
plot3d(ipout4$X[,c(1,2,n4)])
spheres3d(ipout4$X[,c(1,2,n4)],radius=1)
el4 <- get.edgelist(g4)
segments3d(ipout4$X[t(el4),c(1,2,n4)])
```



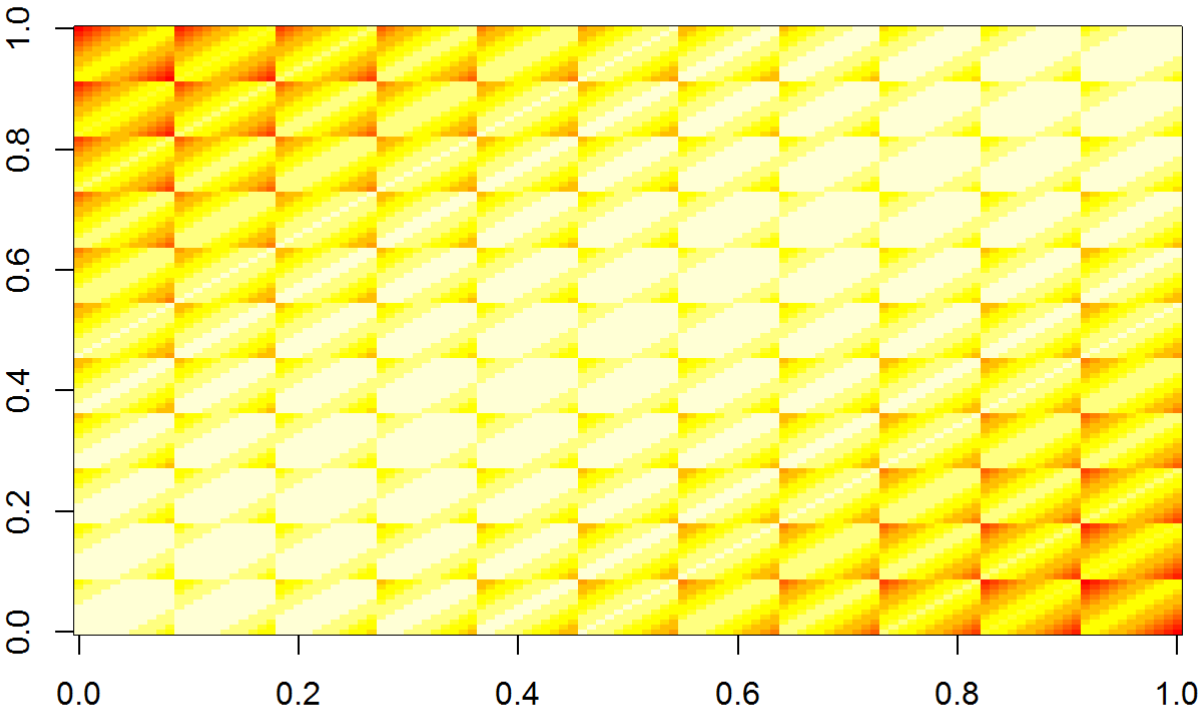
2次元

平面方眼紙グラフの場合

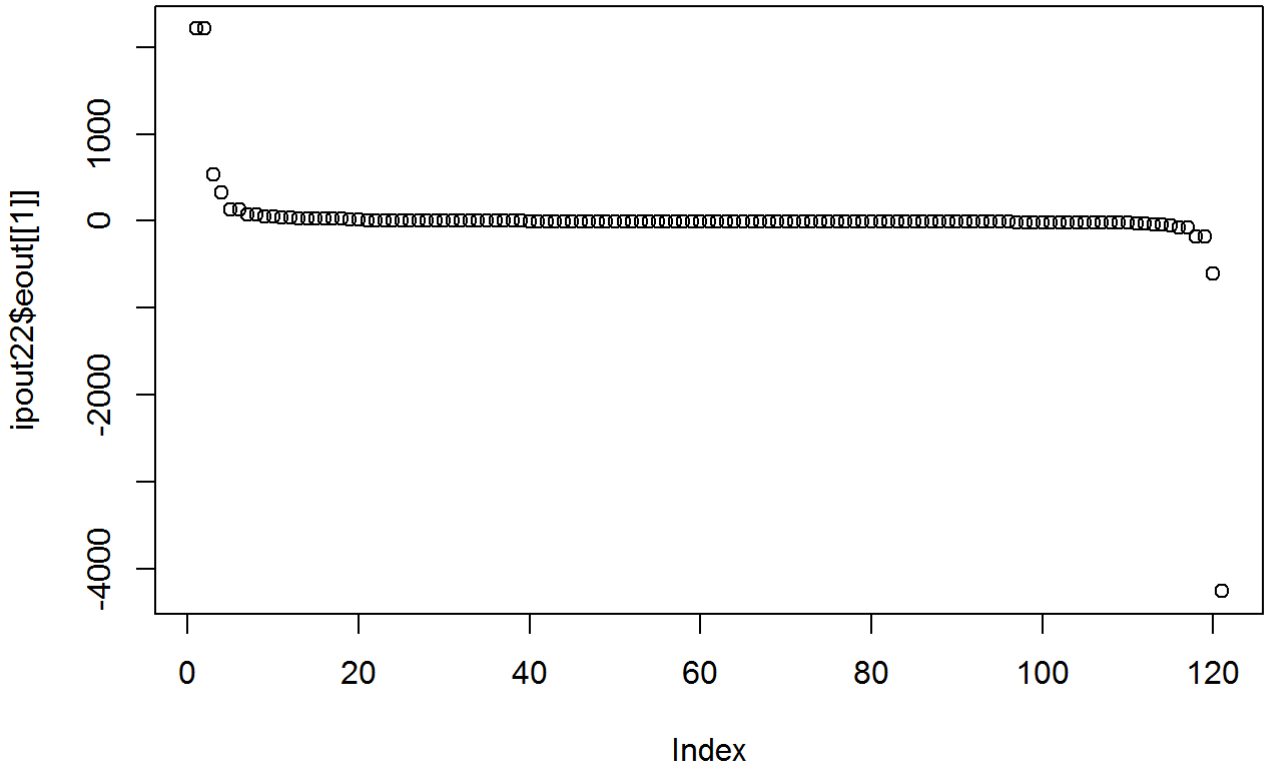
```
xy <- as.matrix(expand.grid(0:10,0:10))
d <- as.matrix(dist(xy))
d <- d==1 +0
g22 <- graph.adjacency(d,mode="undirected")
# エッジ長
e.len22 <- rep(1,length(E(g22)))
ipout22 <- my.IPcoords(g22,e.len22)
# 距離行列
image(ipout22$D)
```



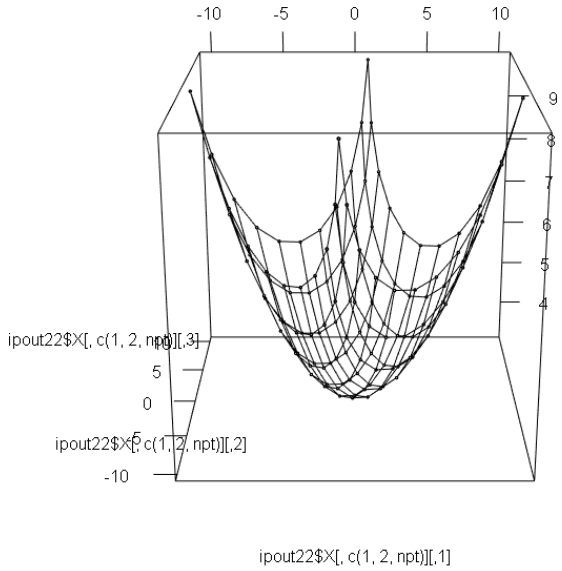

```
# IP行列
image(ipout22$P)
```




```
# 固有値
plot(ipout22$eout[[1]])
```

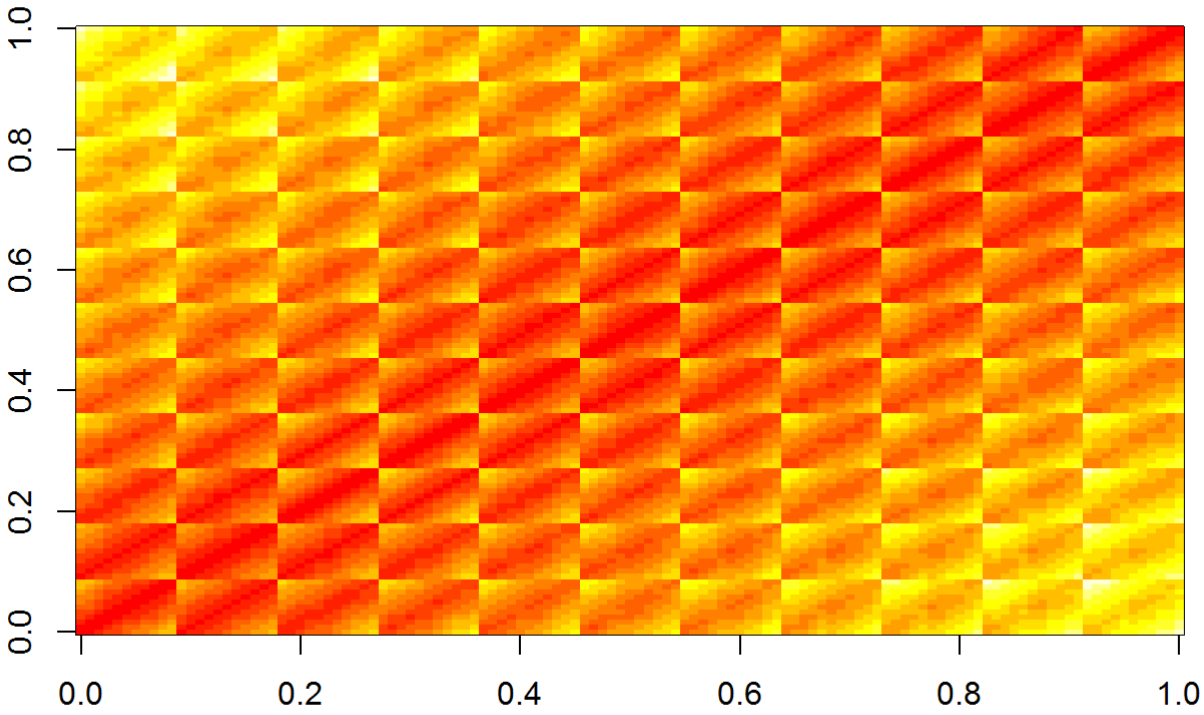


```
npt <- length(xy[,1])
plot3d(ipout22$X[,c(1,2,npt)])
spheres3d(ipout22$X[,c(1,2,npt)],radius=0.1)
el22 <- matrix(as.numeric(get.edgelist(g22)),ncol=2)
segments3d(ipout22$X[t(el22),c(1,2,npt)])
```

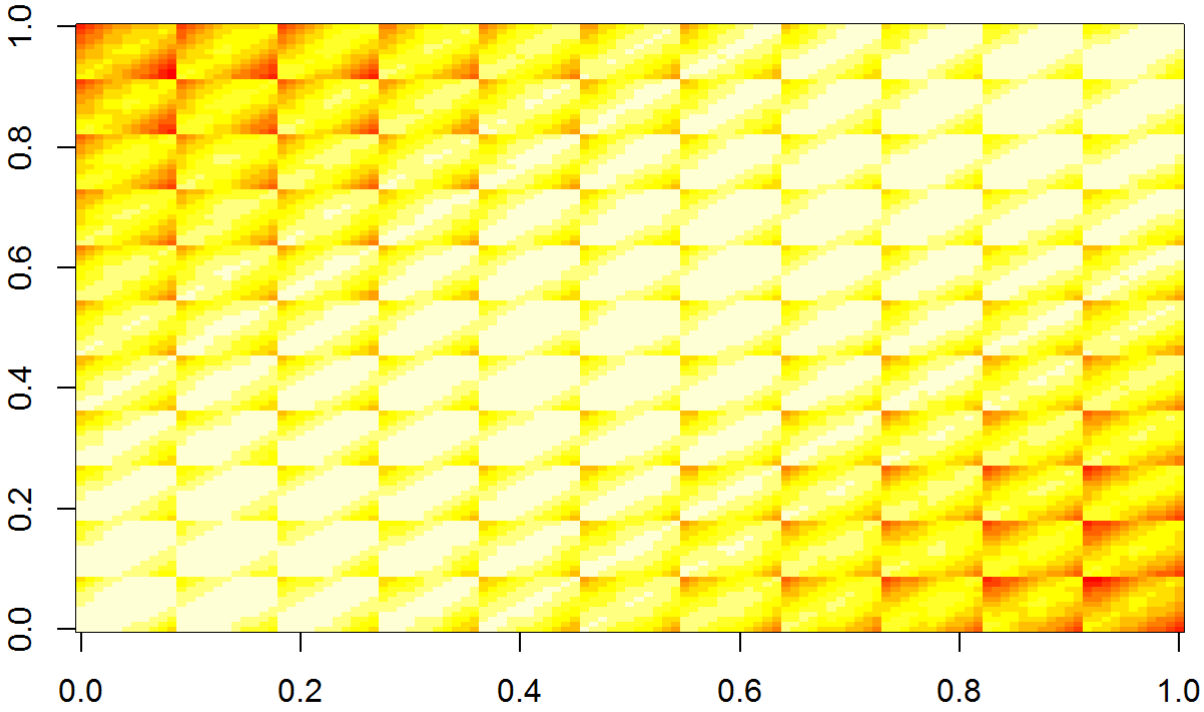


ばらつかせる。

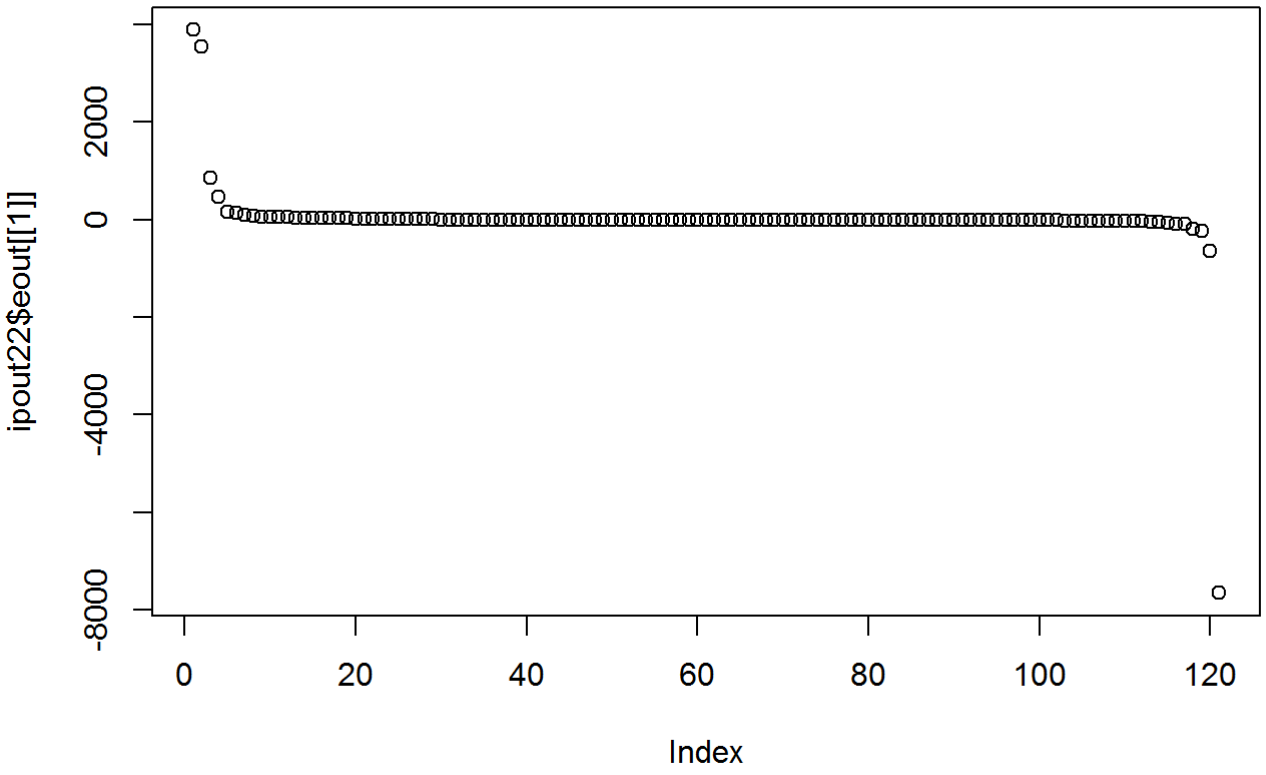
```
xy <- as.matrix(expand.grid(0:10,0:10))
d <- as.matrix(dist(xy))
d <- d==1 +0
g22 <- graph.adjacency(d,mode="undirected")
# エッジ長
e.len22 <- rep(1,length(E(g22))) + runif(length(E(g22)))
ipout22 <- my.IPcoords(g22,e.len22)
# 距離行列
image(ipout22$D)
```



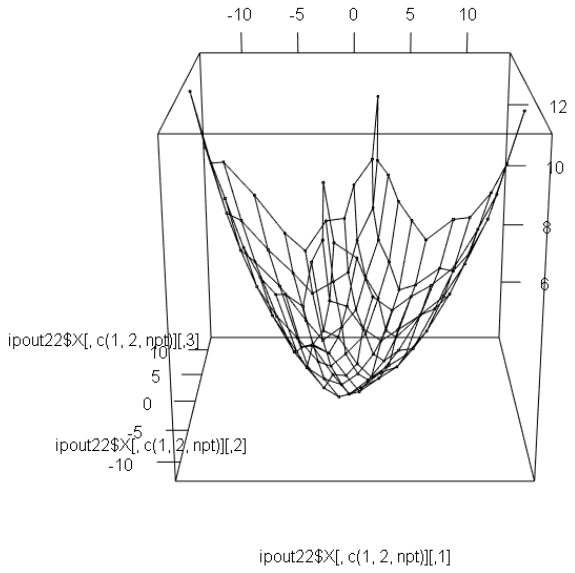

```
# IP行列
image(ipout22$P)
```



```
# 固有値
plot(ipout22$eout[[1]])
```




```
npt <- length(xy[,1])
plot3d(ipout22$X[,c(1,2,npt)])
spheres3d(ipout22$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,npt)])
```



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

```
## [1] -3.439027e-12  5.798029e-12
```

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

閉曲面メッシュグラフを作ってやってみる。

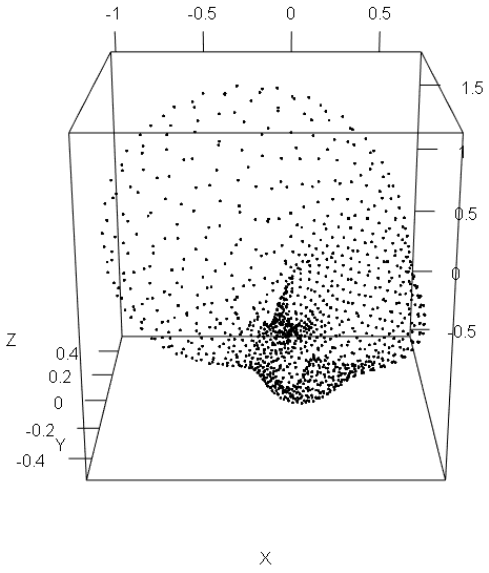
それなりのメッシュグラフを作る

```
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 3.4.2
```

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

n <- 5 # 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 <- 2 # 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)
```

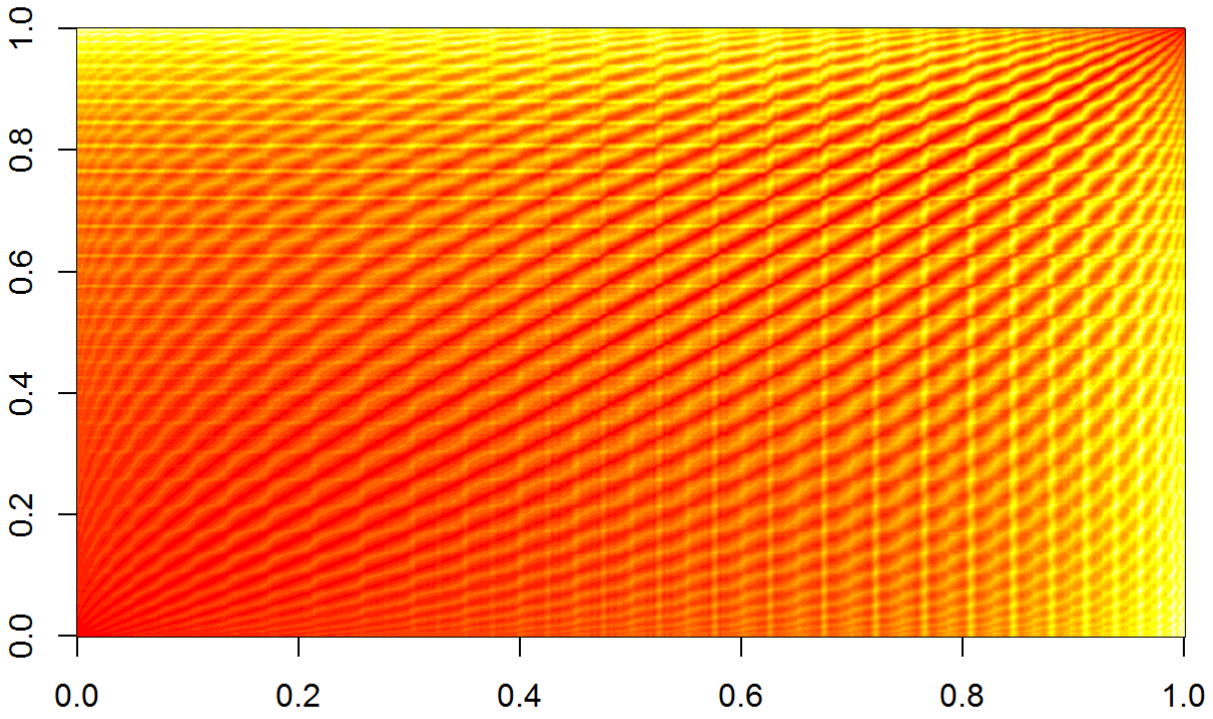
```
X <- xxx[[1]]
plot3d(X)
```



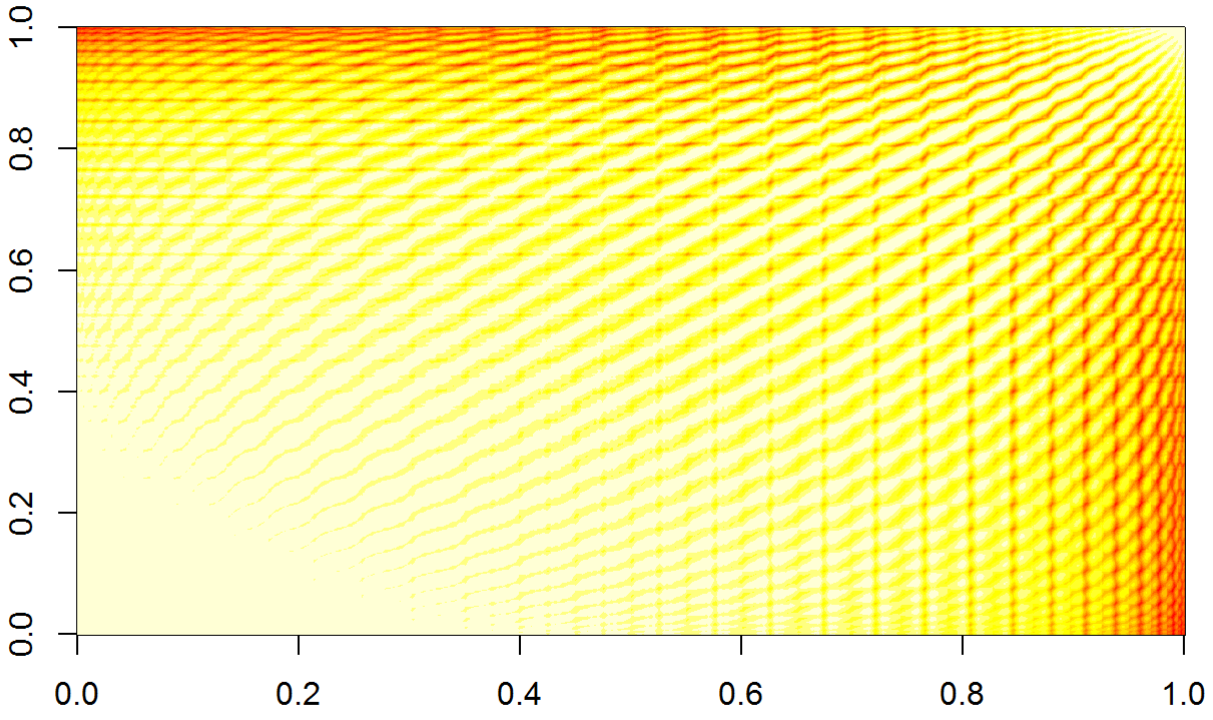

```
g5 <- graph.edgelist(xxx$edge,directed=FALSE)
#plot(g)

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.IPcoords(g5,e.len5)
```

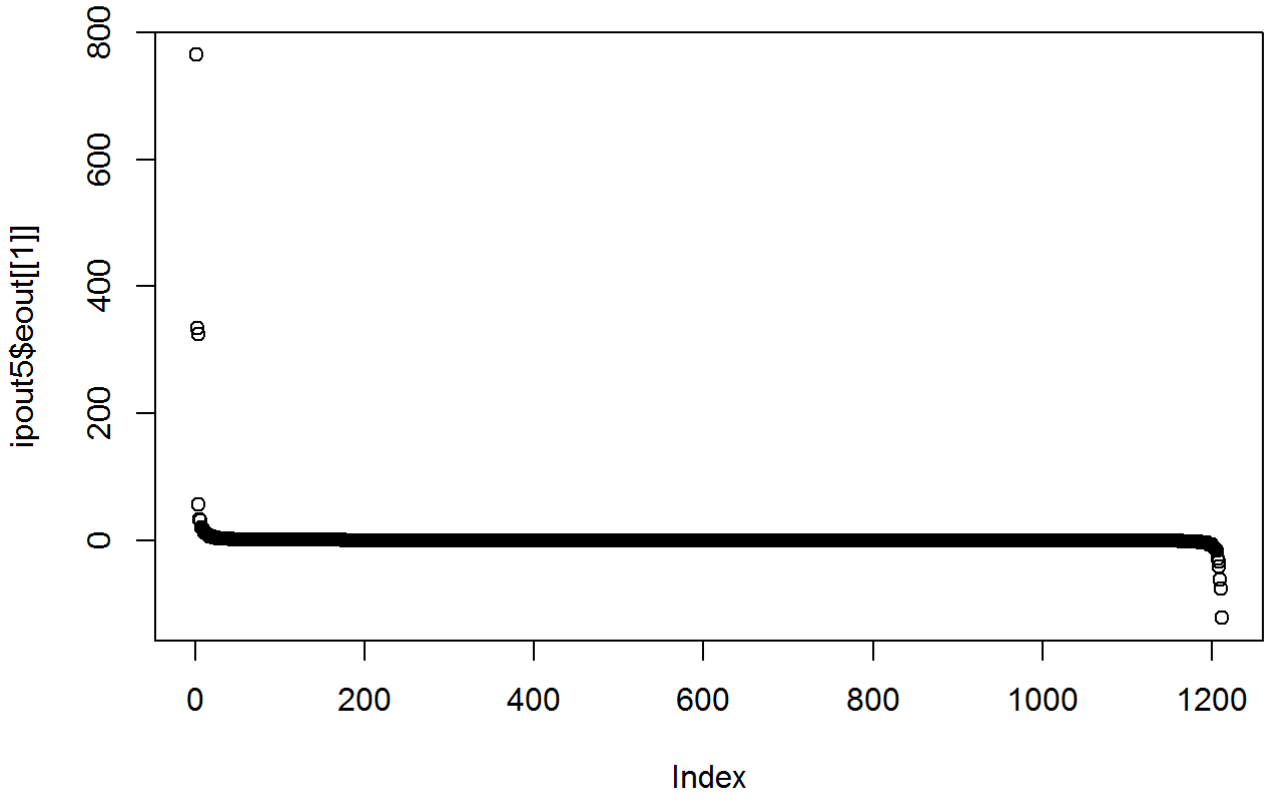
```
# 距離行列
image(ipout5$D)
```




```
# IP行列
image(ipout5$P)
```

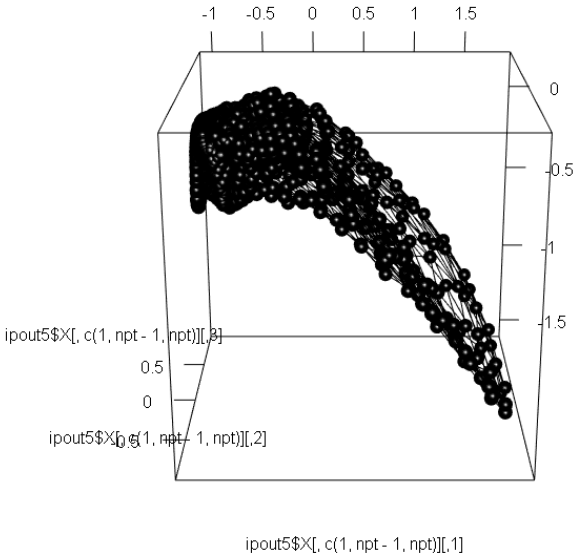


```
# 固有値
plot(ipout5$eout[[1]])
```



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,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…。

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)
```

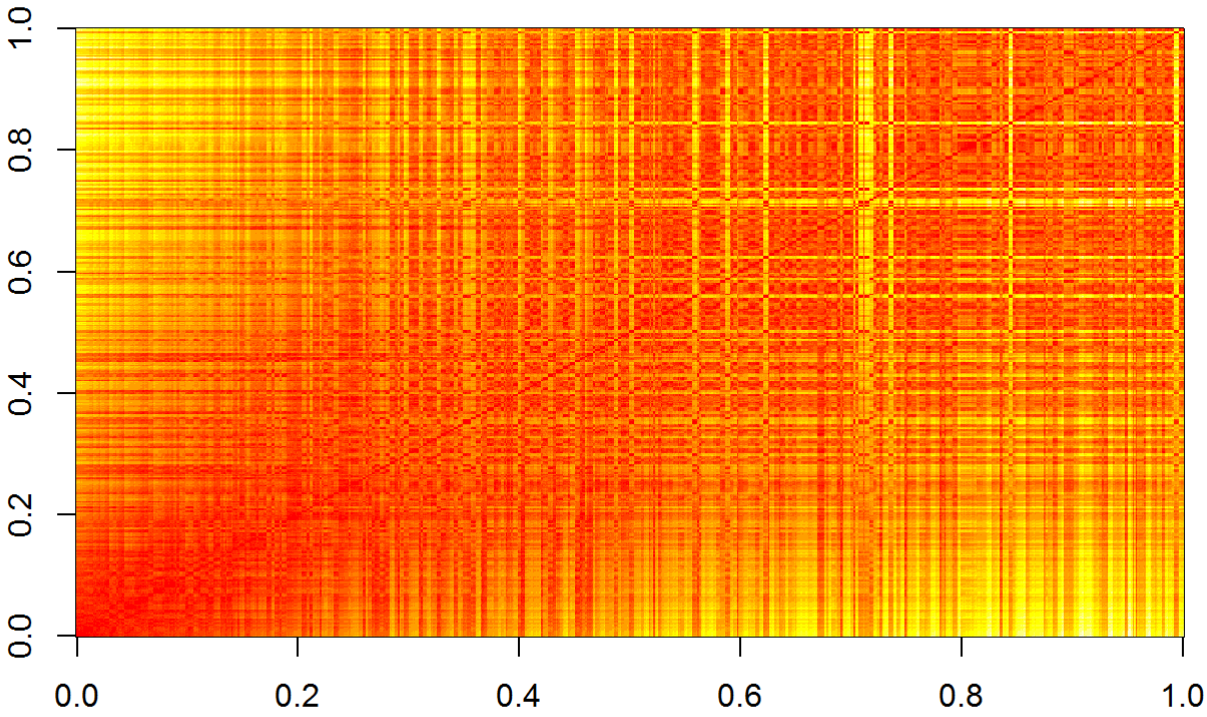
ルートノードからの距離に応じてノードに色を付けてみる。

```
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)
}
```

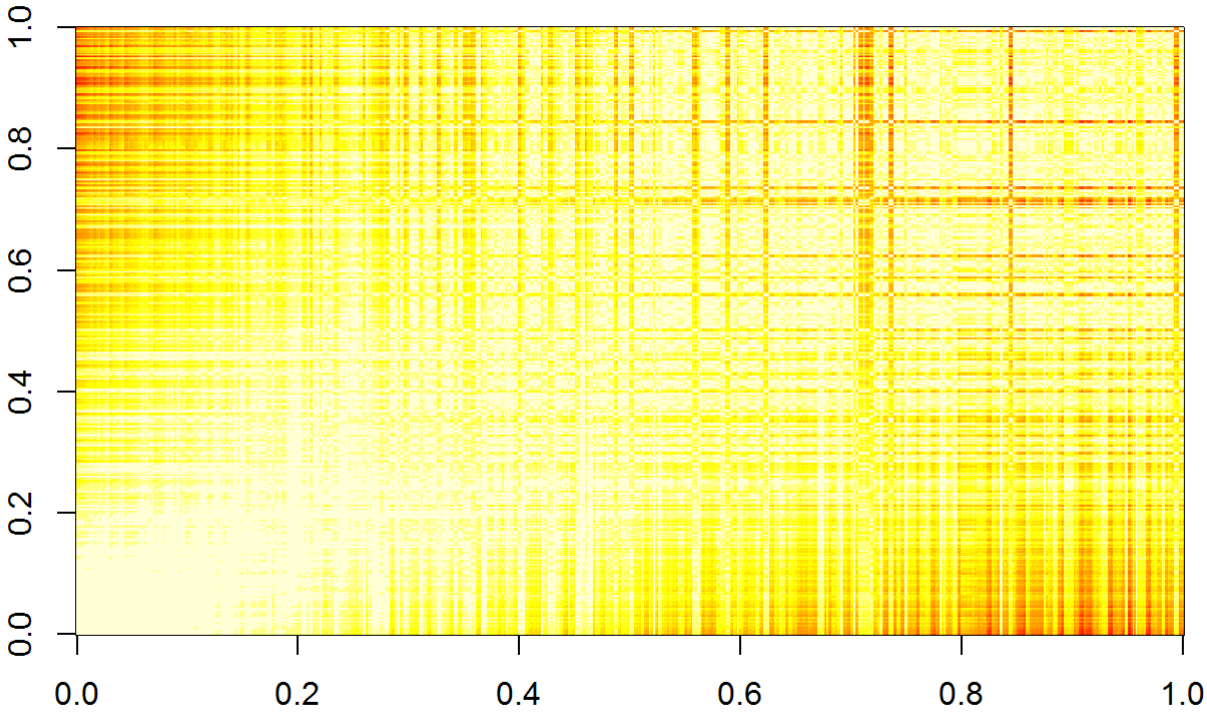
```
g6 <- quad$g
e.len6 <- rep(1,length(E(g6)))
```

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

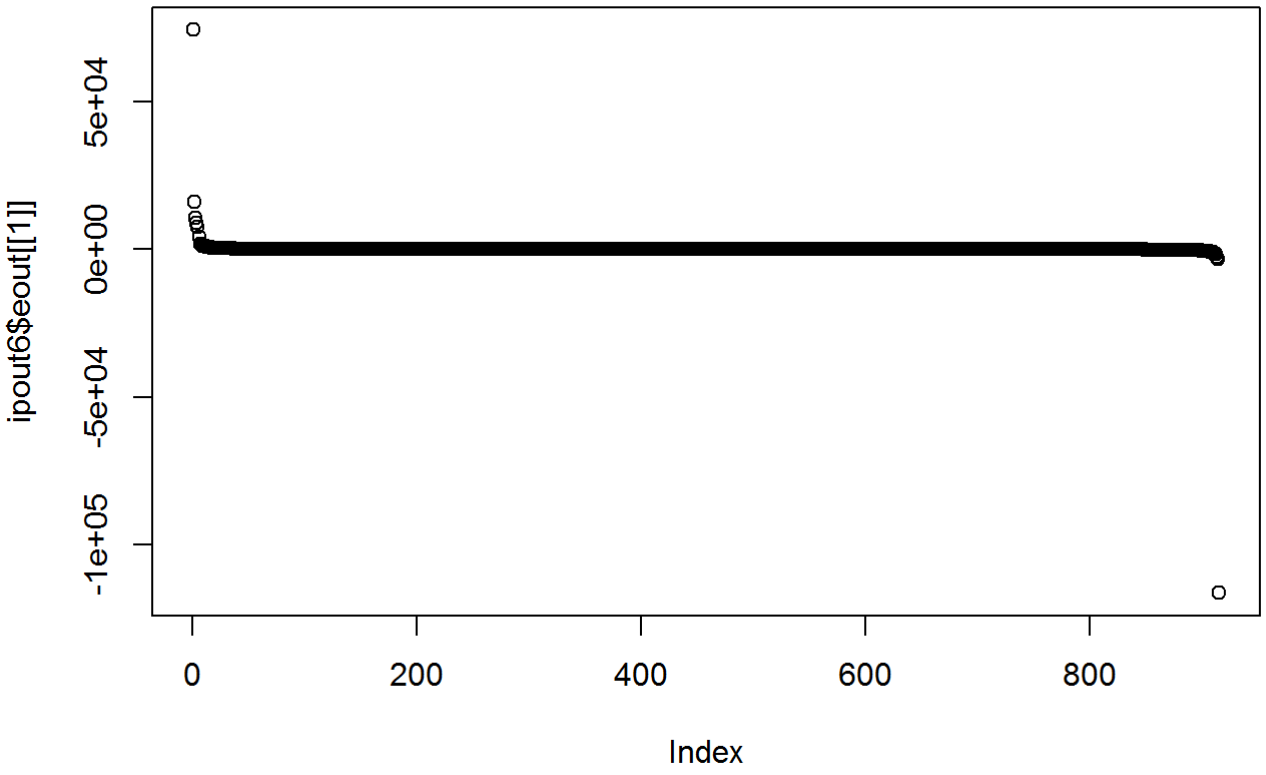
```
# 距離行列
image(ipout6$D)
```



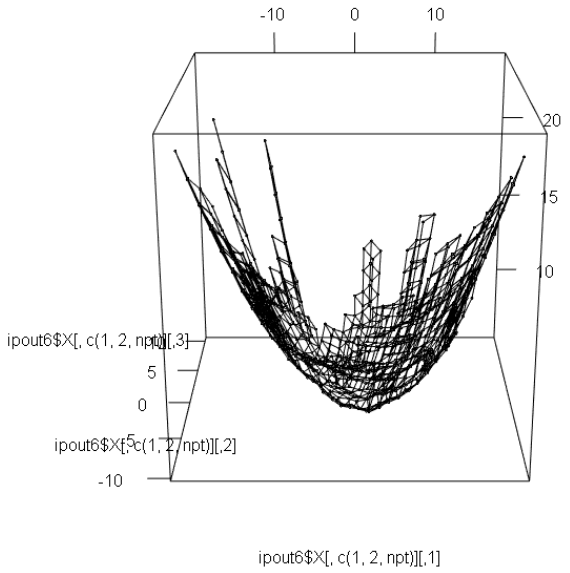

```
# IP行列
image(ipout6$P)
```



```
# 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)])
```