Varimax rotation for principal components in the two-dimensional case

The procedure described here is an extension of the method described in 13 of Harman (1976). It can be shown that the necessary formulae are similar to those given there. In our case the loadings are 2-vectors, and to get the correct results we replace the squared loading values x_j^2 and y_j^2 in equation (13.24) by the squared norm of the loading 2-vectors, and cross-products $x_j y_j$ of values by inner products between 2-vectors.

The first step is to write a function that finds the optimal rotation of a pair of loadings. In what follows, suppose that the first half of the rows of the loadings matrix are the x loadings, and the second half are the y loadings. The routine produces the optimally rotated loadings matrix in a quadrimax sense.

```
rotatetwoloadings <- function(loadings) {
#
#  perform quadrimax rotation of the loadings matrix xl
#   where the first half of the rows of xl correspond to
#   x-loadings and the second half to corresponding y loadings
#
#  Sort out correct dimensions, and separate the x and y
#          loadings
#
     nl <- dim(loadings)[1]/2
     xl <- loadings[1:nl, ]
     yl <- loadings[nl+ (1:nl), ]
#
#  Find coefficients of cos( 4 phi) and sin( 4 phi) in
#   expression for rotated quadrimax criterion
#
     cp <-  xl[,1]*xl[,2] + yl[,1]*yl[,2]
     nd <-  xl[,1]^2 - xl[,2]^2 + yl[,1]^2 - yl[,2]^2
     a  <-  4* sum( cp*nd )
     b  <-  sum( nd^2 - 4*cp^2 )
#
#  sin (4 phi) and a have to have the same sign, so
#   this will find the correct phi
#
     phi <- 0.25 * atan(a, b)
#
#  Calculate and apply the rotation
#
     cphi <- cos(phi)
     sphi  <- sin(phi)
     rmat <- matrix( c( cphi, sphi, -sphi, cphi), nrow=2)
     newload <- loadings %*% rmat
     return(newload)
}
```

Now we can use the function to do an iteration of the quadrimax procedure on every pair of columns in a loadings matrix.

```
rotateallloadings <- function(loadings) {
#
#  loadings is now a matrix of m loadings.
#  The top half is x loadings and the bottom half is y
#          loadings
#  This rotates each pair of columns in turn to increase
#     the vector quadrimax criterion
#
      m <- dim(loadings)[2]
      for (j in (2:m)) { for (i in (1:(j-1))) {
            ij_ c(i, j)
            loadings[, ij]_ rotate2loadings(loadings[, ij])
      }     }
      return(loadings)
}
```

Finally, we can write an iteration that performs this process to convergence.

```
vectorquadrimax <-  function(loads, tol=1.e-6){
#
#  Carries out vector quadrimax rotation for a matrix loads
#   where the top half is x loadings and the bottom half
#   is y loadings.
#  Iterates until increase in score is less than tol
#
  nl <- dim(loads)[1]/2
  qs <- sum((loads[(1:nl),]^2 + loads[nl + (1:nl),]^2)^2)
for(iter in (1:100)) {
      qold <- qs
      loads <- rotateallloadings(loads)
      qs <- sum((loads[(1:nl),]^2 + loads[nl + (1:nl),]^2)^2)
      if((qs - qold) < tol) break
      }
  return(loads)
}
```