Package ‘gyro’

August 20, 2022

Type Package

Title Hyperbolic Geometry

Version 1.1.1

Author Stéphane Laurent

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Hyperbolic geometry in the Minkowski model and the Poincaré model. The methods are based on the gyrovector space theory developed by A. A. Ungar that can be found in the book 'Analytic Hyperbolic Geometry: Mathematical Foundations And Applications' <doi:10.1142/5914>. The package provides functions to plot three-dimensional hyperbolic polyhedra and to plot hyperbolic tilings of the Poincaré disk.

License GPL-3

URL https://github.com/stla/gyro

BugReports https://github.com/stla/gyro/issues

Imports clipr, cxhull (>= 0.3.0), graphics, grDevices, Morpho, plotrix, purrr, Rcpp, rgl, rstudioapi, Rvcg, RCDT, randomcoloR

Suggests arrangements, knitr, rmarkdown, trekcolors, uniformly

LinkingTo Rcpp

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.1

SystemRequirements C++ 11

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-08-20 08:00:02 UTC

R topics documented:

`changesOfSign` *Changes of sign*

Description

Sometimes, the coordinates of the vertices of a polyhedron are given with changes of sign (with a symbol **+/-**). This function performs the changes of sign.

Usage

```
changesOfSign(M, changes = "all")
```

Arguments

`M`	a numeric matrix of coordinates of some points (one point per row)
`changes`	either the indices of the columns of `M` where the changes of sign must be done, or `"all"` to select all the indices

Value

A numeric matrix, `M` transformed by the changes of sign.

Examples

```
library(gyro)
library(rgl)
## ~~ rhombicosidodecahedron ~~##
phi <- (1 + sqrt(5)) / 2
vs1 <- rbind(
  c(1, 1, phi^3),
  c(phi^2, phi, 2 * phi),
  c(2 + phi, 0, phi^2)
)
vs2 <- rbind(vs1, vs1[, c(2, 3, 1)], vs1[, c(3, 1, 2)]) # even permutations
vs <- changesOfSign(vs2)
open3d(windowRect = c(50, 50, 562, 562), zoom = 0.65)
plotGyrohull3d(vs)
```

---

gyroABt *Point on a gyroline*

Description

Point of coordinate `t` on the gyroline passing through two given points `A` and `B`. This is `A` for `t=0` and this is `B` for `t=1`. For `t=1/2` this is the gyromidpoint of the gyrosegment joining `A` and `B`.

Usage

```
gyroABt(A, B, t, s = 1, model = "U")
```

Arguments

`A, B`	two distinct points
`t`	a number
`s`	positive number, the radius of the Poincaré ball if `model="M"`, otherwise, if `model="U"`, this number defines the hyperbolic curvature
`model`	the hyperbolic model, either `"M"` (Möbius model, i.e. Poincaré model) or `"U"` (Ungar model, i.e. hyperboloid model)

Value

A point.

`gyrocentroid` *Gyrocentroid*

Description

Gyrocenroid of a triangle.

Usage

```
gyrocentroid(A, B, C, s = 1, model = "U")
```

Arguments

`A, B, C`	three distinct points
`s`	positive number, the radius of the Poincaré ball if `model="M"`, otherwise, if `model="U"`, this number defines the hyperbolic curvature (the smaller, the more curved)
`model`	the hyperbolic model, either `"M"` (Möbius model, i.e. Poincaré model) or `"U"` (Ungar model, i.e. hyperboloid model)

Value

A point, the gyrocentroid of the triangle `ABC`.

`gyrodemos` *Examples of the 'gyro' package*

Description

Some examples of hyperbolic polyhedra realized with the 'gyro' package.

Usage

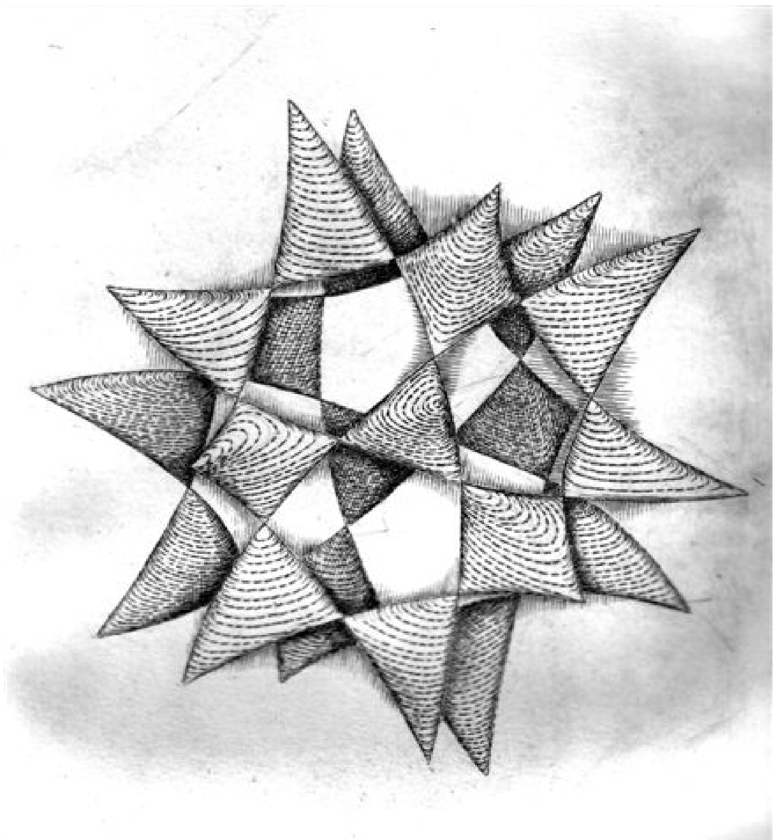
```
gyrodemos()
```

Value

No value. The function firstly copies the demo files in a temporary directory. If you use RStudio, the function opens these files. Otherwise it prints a message giving the instructions to access to these files.

Note

The *BarthLike* file has this name because the figure it generates looks like the Barth sextic (drawing by Patrice Jeener):



`gyromidpoint` *Gyromidpoint*

Description

The gyromidpoint of a `gyrosegment`.

Usage

```
gyromidpoint(A, B, s = 1, model = "U")
```

Arguments

`A, B`	two distinct points (of the same dimension)
`s`	positive number, the radius of the Poincaré ball if `model="M"`, otherwise, if `model="U"`, this number defines the hyperbolic curvature
`model`	the hyperbolic model, either `"M"` (Möbius model, i.e. Poincaré model) or `"U"` (Ungar model, i.e. hyperboloid model)

Value

A point, the gyromidpoint of a the `gyrosegment` joining `A` and `B`.

Note

This is the same as `gyroABt(A, B, 1/2, s)` but the calculation is more efficient.

`gyrosegment` *Gyrosegment*

Description

Gyrosegment joining two given points.

Usage

```
gyrosegment(A, B, s = 1, model = "U", n = 100)
```

Arguments

`A, B`	two distinct points (of the same dimension)
`s`	positive number, the radius of the Poincaré ball if `model="M"`, otherwise, if `model="U"`, this number defines the hyperbolic curvature
`model`	the hyperbolic model, either `"M"` (Möbius model, i.e. Poincaré model) or `"U"` (Ungar model, i.e. hyperboloid model)
`n`	number of points forming the gyrosegment from `A` to `B`

Value

A numeric matrix with `n` rows. Each row is a point of the gyrosegment from `A` (the first row) to `B` (the last row).

Note

The gyrosegment is obtained from `gyroABt` by varying `t` from `0` to `1`.

Examples

```
library(gyro)
# a 2D example ####
A <- c(1, 2); B <- c(1, 1)
opar <- par(mfrow = c(1, 2), mar = c(2, 2, 2, 0.5))
plot(rbind(A, B), type = "p", pch = 19, xlab = NA, ylab = NA,
     xlim = c(0, 2), ylim = c(0, 2), main = "s = 0.2")
s <- 0.2
AB <- gyrosegment(A, B, s)
lines(AB, col = "blue", lwd = 2)
text(t(A), expression(italic(A)), pos = 2)
text(t(B), expression(italic(B)), pos = 3)
# this is an hyperbola whose asymptotes meet at the origin
# approximate asymptotes
lines(rbind(c(0, 0), gyroABt(A, B, t = -20, s)), lty = "dashed")
lines(rbind(c(0, 0), gyroABt(A, B, t = 20, s)), lty = "dashed")
# plot the gyromidoint
points(
 rbind(gyromidpoint(A, B, s)),
 type = "p", pch = 19, col = "red"
)
# another one, with a different `s`
plot(rbind(A, B), type = "p", pch = 19, xlab = NA, ylab = NA,
     xlim = c(0, 2), ylim = c(0, 2), main = "s = 0.1")
s <- 0.1
AB <- gyrosegment(A, B, s)
lines(AB, col = "blue", lwd = 2)
text(t(A), expression(italic(A)), pos = 2)
text(t(B), expression(italic(B)), pos = 3)
# approximate asymptotes
lines(rbind(c(0, 0), gyroABt(A, B, t = -20, s)), lty = "dashed")
lines(rbind(c(0, 0), gyroABt(A, B, t = 20, s)), lty = "dashed")
# plot the gyromidoint
points(
 rbind(gyromidpoint(A, B, s)),
 type = "p", pch = 19, col = "red"
)

# a 3D hyperbolic triangle ####
library(rgl)
A <- c(1, 0, 0); B <- c(0, 1, 0); C <- c(0, 0, 1)
s <- 0.3
AB <- gyrosegment(A, B, s)
AC <- gyrosegment(A, C, s)
BC <- gyrosegment(B, C, s)
view3d(30, 30, zoom = 0.75)
lines3d(AB, lwd = 3); lines3d(AC, lwd = 3); lines3d(BC, lwd = 3)
```

---

gyrotriangle *Gyrotriangle in 3D space*

Description

3D gyrotriangle as a mesh.

Usage

```
gyrotriangle(
  A,
  B,
  C,
  s = 1,
  model = "U",
  iterations = 5,
  palette = NULL,
  bias = 1,
  interpolate = "linear",
  g = identity
)
```

Arguments

`A, B, C`	three distinct 3D points
`s`	positive number, the radius of the Poincaré ball if `model="M"`, otherwise, if `model="U"`, this number defines the hyperbolic curvature (the smaller, the more curved)
`model`	the hyperbolic model, either `"M"` (Möbius model, i.e. Poincaré model) or `"U"` (Ungar model, i.e. hyperboloid model)
`iterations`	the gyrotriangle is constructed by iterated subdivisions, this argument is the number of iterations
`palette`	a vector of colors to decorate the triangle, or `NULL` if you don't want to use a color palette
`bias, interpolate`	if `palette` is not `NULL`, these arguments are passed to `colorRamp`
`g`	a function from [0,1] to [0,1]; if `palette` is not `NULL`, this function is applied to the scalars defining the colors (the normalized gyrodistances to the gyrocentroid of the gyrotriangle)

Value

A `mesh3d` object.

Examples

```
library(gyro)
library(rgl)
A <- c(1, 0, 0); B <- c(0, 1, 0); C <- c(0, 0, 1)
ABC <- gyrotriangle(A, B, C, s = 0.3)
open3d(windowRect = c(50, 50, 562, 562))
view3d(30, 30, zoom = 0.75)
```

```
shade3d(ABC, color = "navy", specular = "cyan")

# using a color palette ####
library(trekcolors)
ABC <- gyrotriangle(
  A, B, C, s = 0.5,
  palette = trek_pal("klingon"), bias = 1.5, interpolate = "spline"
)
open3d(windowRect = c(50, 50, 562, 562))
view3d(zoom = 0.75)
shade3d(ABC)

# hyperbolic icosahedron ####
library(rgl)
library(Rvcg) # to get the edges with the `vcgGetEdge` function
icosahedron <- icosahedron3d() # mesh with 12 vertices, 20 triangles
vertices <- t(icosahedron$vb[-4, ])
triangles <- t(icosahedron$it)
edges <- as.matrix(vcgGetEdge(icosahedron)[, c("vert1", "vert2")])
s <- 0.3
open3d(windowRect = c(50, 50, 562, 562))
view3d(zoom = 0.75)
for(i in 1:nrow(triangles)){
  triangle <- triangles[i, ]
  A <- vertices[triangle[1], ]
  B <- vertices[triangle[2], ]
  C <- vertices[triangle[3], ]
  gtriangle <- gyrotriangle(A, B, C, s)
  shade3d(gtriangle, color = "midnightblue")
}
for(i in 1:nrow(edges)){
  edge <- edges[i, ]
  A <- vertices[edge[1], ]
  B <- vertices[edge[2], ]
  gtube <- gyrotube(A, B, s, radius = 0.03)
  shade3d(gtube, color = "lemonchiffon")
}
spheres3d(vertices, radius = 0.05, color = "lemonchiffon")
```

`gyrotube` *Gyrotube (tubular gyrosegment)*

Description

Tubular gyrosegment joining two given 3D points.

Usage

```
gyrotube(A, B, s = 1, model = "U", n = 100, radius, sides = 90, caps = FALSE)
```

Arguments

A, B	distinct 3D points
s	positive number, the radius of the Poincaré ball if model="M", otherwise, if model="U", this number defines the hyperbolic curvature (higher value, less curved)
model	the hyperbolic model, either "M" (Möbius model, i.e. Poincaré model) or "U" (Ungar model, i.e. hyperboloid model)
n	number of points forming the gyrosegment
radius	radius of the tube around the gyrosegment
sides	number of sides in the polygon cross section
caps	Boolean, whether to put caps on the ends of the tube

Value

A mesh3d object.

Examples

```
library(gyro)
library(rgl)
A <- c(1, 2, 0); B <- c(1, 1, 0)
tube <- gyrotube(A, B, s = 0.2, radius = 0.02)
shade3d(tube, color = "orangered")

# a 3D hyperbolic triangle ####
library(rgl)
A <- c(1, 0, 0); B <- c(0, 1, 0); C <- c(0, 0, 1)
s <- 0.3
r <- 0.03
AB <- gyrotube(A, B, s, radius = r)
AC <- gyrotube(A, C, s, radius = r)
BC <- gyrotube(B, C, s, radius = r)
view3d(30, 30, zoom = 0.75)
shade3d(AB, color = "gold")
shade3d(AC, color = "gold")
shade3d(BC, color = "gold")
spheres3d(rbind(A, B, C), radius = 0.04, color = "gold")
```

---

hdelaunay *Hyperbolic Delaunay triangulation*

Description

Computes the hyperbolic Delaunay triangulation of a set of points.

Usage

```
hdelaunay(points, model = "M")
```

Arguments

`points`	points in the unit disk given as a numeric matrix with two columns
`model`	the hyperbolic model, either `"M"` (Möbius model, i.e. Poincaré model) or `"U"` (Ungar model, i.e. hyperboloid model)

Value

A list with five fields `vertices`, `edges`, `triangles`, `ntriangles`, and `centroids`, a matrix giving the gyrocentroids of the triangles. The input `points` matrix and the output `vertices` matrix are the same up to the order of the rows if `model="M"`, and if `model="U"`, the points in the output `vertices` matrix are obtained by isomorphism.

See Also

`plotHdelaunay`

Examples

```
library(gyro)
library(uniformly)
set.seed(666)
points <- runif_in_sphere(10L, d = 2)
hdelaunay(points)
```

`hreflection` *Hyperbolic reflection*

Description

Hyperbolic reflection in the Poincaré disk.

Usage

```
hreflection(A, B, M)
```

Arguments

`A, B`	two points in the Poincaré disk defining the reflection line
`M`	a point in the Poincaré disk to be reflected

Value

A point in the Poincaré disk, the image of `M` by the hyperbolic reflection with respect to the line passing through `A` and `B`.

Examples

```
library(gyro)
library(plotrix)
A <- c(0.45, 0.78)
B <- c(0.1, -0.5)
M <- c(0.7, 0)
opar <- par(mar = c(0, 0, 0, 0))
plot(NULL, type = "n", xlim = c(-1, 1), ylim = c(-1, 1), asp = 1,
     axes = FALSE, xlab = NA, ylab = NA)
draw.circle(0, 0, radius = 1, lwd = 2)
lines(gyrosegment(A, B, model = "M"))
points(rbind(A, B), pch = 19)
points(rbind(M), pch = 19, col = "blue")
P <- hreflection(A, B, M)
points(rbind(P), pch = 19, col = "red")
par(opar)
```

PhiMU — *Isomorphism from Ungar gyrovector space to Möbius gyrovector space*

Description

Isomorphism from the Ungar gyrovector space to the Möbius gyrovector space.

Usage

```
PhiMU(A, s = 1)
```

Arguments

A	a point in the Ungar vector space with curvature s
s	a positive number, the hyperbolic curvature of the Ungar vector space

Value

The point of the Poincaré ball of radius s corresponding to A by isomorphism.

`PhiUM` *Isomorphism from Möbius gyrovector space to Ungar gyrovector space*

Description

Isomorphism from the Möbius gyrovector space to the Ungar gyrovector space.

Usage

```
PhiUM(A, s = 1)
```

Arguments

`A`	a point whose norm is lower than `s`
`s`	positive number, the radius of the Poincaré ball

Value

The point of the Ungar gyrovector space corresponding to `A` by isomorphism.

`plotGyrohull3d` *Hyperbolic convex hull*

Description

Plot the hyperbolic convex hull of a set of 3D points.

Usage

```
plotGyrohull3d(
  points,
  s = 1,
  model = "U",
  iterations = 5,
  n = 100,
  edgesAsTubes = TRUE,
  verticesAsSpheres = edgesAsTubes,
  edgesColor = "yellow",
  spheresColor = edgesColor,
  tubesRadius = 0.03,
  spheresRadius = 0.05,
  facesColor = "navy",
  bias = 1,
  interpolate = "linear",
  g = identity
)
```

Arguments

- `points` — matrix of 3D points, one point per row
- `s` — positive number, the radius of the Poincaré ball if `model="M"`, otherwise, if `model="U"`, this number defines the hyperbolic curvature (the smaller, the more curved)
- `model` — the hyperbolic model, either `"M"` (Möbius model, i.e. Poincaré model) or `"U"` (Ungar model, i.e. hyperboloid model)
- `iterations` — argument passed to `gyrotriangle`
- `n` — argument passed to `gyrotube` or `gyrosegment`, the number of points for each edge
- `edgesAsTubes` — Boolean, whether to represent tubular edges
- `verticesAsSpheres` — Boolean, whether to represent the vertices as spheres
- `edgesColor` — a color for the edges
- `spheresColor` — a color for the spheres, if `verticesAsSpheres = TRUE`
- `tubesRadius` — radius of the tubes, if `edgesAsTubes = TRUE`
- `spheresRadius` — radius of the spheres, if `verticesAsSpheres = TRUE`
- `facesColor` — this argument sets the color of the faces; it can be either a single color or a color palette, i.e. a vector of colors; if it is a color palette, it will be passed to the argument `palette` of `gyrotriangle`
- `bias, interpolate, g` — these arguments are passed to `gyrotriangle` in the case when `facesColor` is a color palette

Value

No value, called for plotting.

Examples

```
library(gyro)
library(rgl)
# Triangular orthobicopula ####
points <- rbind(
  c(1, -1/sqrt(3), sqrt(8/3)),
  c(1, -1/sqrt(3), -sqrt(8/3)),
  c(-1, -1/sqrt(3), sqrt(8/3)),
  c(-1, -1/sqrt(3), -sqrt(8/3)),
  c(0, 2/sqrt(3), sqrt(8/3)),
  c(0, 2/sqrt(3), -sqrt(8/3)),
  c(1, sqrt(3), 0),
  c(1, -sqrt(3), 0),
  c(-1, sqrt(3), 0),
  c(-1, -sqrt(3), 0),
  c(2, 0, 0),
  c(-2, 0, 0)
```

```
)
open3d(windowRect = c(50, 50, 562, 562))
view3d(zoom = 0.7)
plotGyrohull3d(points, s = 0.4)

# a non-convex polyhedron with triangular faces ####
vertices <- rbind(
  c(-2.1806973249, -2.1806973249, -2.1806973249),
  c(-3.5617820682, 0.00000000000, 0.00000000000),
  c(0.00000000000, -3.5617820682, 0.00000000000),
  c(0.00000000000, 0.00000000000, -3.5617820682),
  c(-2.1806973249, -2.1806973249, 2.18069732490),
  c(0.00000000000, 0.00000000000, 3.56178206820),
  c(-2.1806973249, 2.18069732490, -2.1806973249),
  c(0.00000000000, 3.56178206820, 0.00000000000),
  c(-2.1806973249, 2.18069732490, 2.18069732490),
  c(2.18069732490, -2.1806973249, -2.1806973249),
  c(3.56178206820, 0.00000000000, 0.00000000000),
  c(2.18069732490, -2.1806973249, 2.18069732490),
  c(2.18069732490, 2.18069732490, -2.1806973249),
  c(2.18069732490, 2.18069732490, 2.18069732490))
triangles <- 1 + rbind(
  c(3, 2, 0),
  c(0, 1, 3),
  c(2, 1, 0),
  c(4, 2, 5),
  c(5, 1, 4),
  c(4, 1, 2),
  c(6, 7, 3),
  c(3, 1, 6),
  c(6, 1, 7),
  c(5, 7, 8),
  c(8, 1, 5),
  c(7, 1, 8),
  c(9, 2, 3),
  c(3, 10, 9),
  c(9, 10, 2),
  c(5, 2, 11),
  c(11, 10, 5),
  c(2, 10, 11),
  c(3, 7, 12),
  c(12, 10, 3),
  c(7, 10, 12),
  c(13, 7, 5),
  c(5, 10, 13),
  c(13, 10, 7))
edges0 <- do.call(c, lapply(1:nrow(triangles), function(i){
  face <- triangles[i, ]
  list(
    sort(c(face[1], face[2])),
    sort(c(face[1], face[3])),
    sort(c(face[2], face[3]))
  )
```

```
}))
edges <- do.call(rbind, edges0)
edges <- edges[!duplicated(edges), ]
s <- 2
library(rgl)
open3d(windowRect = c(50, 50, 1074, 562))
mfrow3d(1, 2)
view3d(zoom = 0.65)
for(i in 1:nrow(triangles)){
  triangle <- triangles[i, ]
  A <- vertices[triangle[1], ]
  B <- vertices[triangle[2], ]
  C <- vertices[triangle[3], ]
  gtriangle <- gyrotriangle(A, B, C, s)
  shade3d(gtriangle, color = "violetred")
}
for(i in 1:nrow(edges)){
  edge <- edges[i, ]
  A <- vertices[edge[1], ]
  B <- vertices[edge[2], ]
  gtube <- gyrotube(A, B, s, radius = 0.06)
  shade3d(gtube, color = "darkviolet")
}
spheres3d(vertices, radius = 0.09, color = "deeppink")
# now plot the hyperbolic convex hull
next3d()
view3d(zoom = 0.65)
plotGyrohull3d(vertices, s)

# an example of color palette ####
library(trekcolors)
library(uniformly)
set.seed(666)
points <- runif_on_sphere(50, d = 3)
open3d(windowRect = c(50, 50, 562, 562))
plotGyrohull3d(
  points, edgesColor = "brown",
  facesColor = trek_pal("lcars_series"), g = function(u) 1-u^2
)
```

---

plotHdelaunay *Plot hyperbolic Delaunay triangulation*

Description

Plot a hyperbolic Delaunay triangulation obtained with `hdelaunay`.

Usage

```
plotHdelaunay(
```

```
  hdel,
  vertices = TRUE,
  edges = TRUE,
  circle = TRUE,
  color = "distinct",
  hue = "random",
  luminosity = "random"
)
```

Arguments

hdel	an output of hdelaunay
vertices	Boolean, whether to plot the vertices
edges	Boolean, whether to plot the edges
circle	Boolean, whether to plot the unit circle; ignored for the Ungar model
color	this argument controls the colors of the triangles; it can be NA for no color, "random" for random colors generated with randomColor, "distinct" for distinct colors generated with distinctColorPalette, a single color, a vector of colors (color i attributed to the i-th triangle), or a vectorized function mapping each point in the unit interval to a color
hue, luminosity	passed to randomColor if color="random"

Value

No returned value, just generates a plot.

Examples

```
library(gyro)
library(uniformly)
set.seed(666)

points <- runif_in_sphere(35L, d = 2)
hdel <- hdelaunay(points, model = "M")
plotHdelaunay(hdel)

points <- runif_in_sphere(35L, d = 2, r = 0.7)
hdel <- hdelaunay(points, model = "U")
plotHdelaunay(hdel)

# example with colors given by a function ####
library(gyro)
library(trekcolors)

phi <- (1 + sqrt(5)) / 2
theta <- head(seq(0, pi/2, length.out = 11), -1L)
a <- phi^((2*theta/pi)^0.8 - 1)
u <- a * cos(theta)
```

```
v <- a * sin(theta)
x <- c(0, u, -v, -u, v)
y <- c(0, v, u, -v, -u)
pts <- cbind(x, y) / 1.03

hdel <- hdelaunay(pts, model = "M")

fcolor <- function(t){
  RGB <- colorRamp(trek_pal("klingon"))(t)
  rgb(RGB[, 1L], RGB[, 2L], RGB[, 3L], maxColorValue = 255)
}

plotHdelaunay(
  hdel, vertices = FALSE, circle = FALSE, color = fcolor
)
```

`tiling` *Hyperbolic tiling*

Description

Draw a hyperbolic tiling of the Poincaré disk.

Usage

```
tiling(n, p, depth = 4, colors = c("navy", "yellow"), circle = TRUE, ...)
```

Arguments

`n, p`	two positive integers satisfying $1/n + 1/p < 1/2$
`depth`	positive integer, the number of recursions
`colors`	two colors to fill the hyperbolic tiling
`circle`	Boolean, whether to draw the unit circle
`...`	additional arguments passed to `draw.circle`

Value

No returned value, just draws the hyperbolic tiling.

Note

The higher value of `n`, the slower. And of course increasing `depth` slows down the rendering. The value of `p` has no influence on the speed.

Examples

```
library(gyro)
tiling(3, 7, border = "orange")
```

Index