def show\_round():  
 for u in range(0, params.width \* params.wall):  
 a = 2.0 \* u / (params.width \* params.wall) - 1.0  
 for v in range(0, params.height \* params.wall):  
 b = 2.0 \* v / (params.height \* params.wall) - 1.0  
 y = np.sqrt(np.square(a) + np.square(b))  
 if y <= 1.0:  
 x = 0.25 + 0.5 \* (np.arctan(b / (a + 1e-8)) / np.pi

if a < 0.0:

x += 0.5   
 k = maze[int(params.width \* x)][int(params.height \* y)]  
 im.putpixel((u, v), k \* params.color)

def show\_round\_matrix():  
 maze2 = np.array(maze)  
 a = np.arange(-1, 1 + 1 / params.width, 1 / params.width).reshape(-1, 1)  
 b = np.arange(-1, 1 + 1 / params.height, 1 / params.height).reshape(-1, 1).T  
  
 a2 = a \* a  
 b2 = b \* b  
  
 x = (b @ (1 / (a + 1e-8)).T).T / np.pi + (1 - np.sign(a)) / 2  
 x = (x \* params.width).astype(int)  
 y = np.tile(a2, [1, b.shape[0]]) + np.tile(b2, [1, a.shape[0]]).T  
 y = y \*\* 0.5  
 ix = y <= 1  
 y = (y \* params.height).astype(int)  
  
 x = x[ix]  
 y = y[ix]  
  
 v = maze2[x, y] \* params.color  
 res = np.zeros\_like(maze2)  
 res[ix] = v

numpy polárkoordináta

kep = np.array(width, height)

i in (0, width\*wall)

j in (0, width\*wall

i 🡪 2 \* i / (width\*wall) – 1 -1 és 1 közé 2/(width\*wall) lépéssel width\*wall db

j 🡪 2 \* j / (height\*wall) – 1 -1 és 1 közé 2/(width\*height) lépéssel width\*height db