1 Orthogonal Polynomials

1.1 Overview

Python functions and a script to generate, evaluate, and visualize Zernike polynomials, a family of orthogonal polynomials over the unit disk, $D = \{(\rho, \theta) | 0 \leq \rho \leq 1, 0 \leq \theta \leq 2\pi\}$ and discussions and proofs of some of their properties. Zernike polynomials are used to describe aberrations in a lens (e.g., the cornea).

```
import math
import numpy as np
import scipy as sp
import scipy.misc as spm
import matplotlib.pyplot as plt
import pyradi.ryplot as ryplot
from matplotlib import cm
%matplotlib inline
```

1.2 Description

Zernike polynomials are either even or odd.

$$Z_n^m(\rho, \varphi) = R_n^m(\rho)\, \cos(m\, \varphi) \qquad \text{(Even Zernike Polynomials)}$$
$$Z_n^{-m}(\rho, \varphi) = R_n^m(\rho)\, \sin(m\, \varphi), \qquad \text{(Odd Zernike Polynomials)}$$

where m and n are nonnegative integers with $n \geq m$, ϕ is the azimuthal angle, ρ is the radial distance $0 \leq \rho \leq 1$, and R_n^m are the radial polynomials defined below. Zernike polynomials have the property of being limited to a range of -1 to +1, i.e. $|Z_n^m(\rho, \varphi)| \leq 1$. The radial polynomials are defined as

$$R_n^m(\rho) = \sum_{k=0}^{\frac{n-m}{2}} \frac{(-1)^k\, (n-k)!}{k!\, \left(\frac{n+m}{2} - k\right)!\, \left(\frac{n-m}{2} - k\right)!}\, \rho^{n-2\,k}$$

for $n - m$ even, and are identically 0 for $n - m$ odd.

1.3 Calculate a few

$$
\begin{aligned}
R_0^0(\rho) &= 1 \\
R_1^1(\rho) &= \rho \\
R_2^0(\rho) &= 2\rho^2 - 1 \\
R_2^2(\rho) &= \rho^2 \\
R_3^1(\rho) &= 3\rho^3 - 2\rho \\
R_3^3(\rho) &= \rho^3 \\
R_4^0(\rho) &= 6\rho^4 - 6\rho^2 + 1 \\
R_4^2(\rho) &= 4\rho^4 - 3\rho^2 \\
R_4^4(\rho) &= \rho^4
\end{aligned}
$$

Zernike Polynomials: Even and odd Zernike Polynomials are defined respectively as

$$
Z_n^m(\rho, \theta) = R_m^n(\rho)\cos(m\theta) \quad \text{and} \quad Z_n^{-m}(\rho, \theta) = R_m^n(\rho)\sin(m\theta)
$$

where $n, m \in \mathbb{Z}, n \geq m \geq 0$, θ is the *azimuthal angle*, ρ the radial distance, and $R_n^m(\rho)$, the radial polynomials given by

$$
R_n^m(\rho) = \sum_{k=0}^{\frac{n-m}{2}} \frac{(-1)^k\,(n-k)!}{k!\left(\frac{n+m}{2}-k\right)!\left(\frac{n-m}{2}-k\right)!}\,\rho^{n-2\,k}
$$

Rewriting the ratios of factorials in the radial part as products of Binomial coefficient shows that the coefficients are integer numbers:

$$
R_n^m(\rho) = \sum_{k=0}^{\frac{n-m}{2}} (-1)^k \binom{n-k}{k} \binom{n-2k}{\frac{n-m}{2}-k} \rho^{n-2k}
$$

They satisfy $|Z_n^m(\rho, \theta)| \leq 1|$ for all $\rho \in (0, 1)$ and all θ.

1.4 Toward a Recursive Method for Generating the Polynomials

We can find a recursive formula to compute Zernike radial polynomials[1].

The following recurrence relationship has been previously proposed[2]

$$R_n^m(\rho) = \frac{1}{K_1}\Big[(K_2\rho^2 + K_3)R_{n\ 2}^m(\rho) + K_4 R_{n\ 4}^m(\rho)\Big]$$

for

$$n = m + 4, m + 6, \ldots$$

where the coefficients are given by[3]

$$\begin{aligned}
K_1 &= \frac{(n+m)(n-m)(n-2)}{2} \\
K_2 &= 2n(n-1)(n-2) \\
K_3 &= -m^2(n-1) - n(n-1)(n-2) \\
K_4 &= -\frac{n(n+m-2)(n-m-2)}{2}
\end{aligned}$$

To initialize the method we specify cases in which $n = m$ and $n - m = 2$[4]:

$$\begin{aligned}
R_m^m(\rho) &= \rho^m \\
R_{m+2}^m(\rho) &= (m+2)\rho^{m+2} - (m+1)r^m
\end{aligned}$$

Taking a three-term recurrence relation previously derived[5]

$$R_n^m(\rho) = \rho L_1 R_{n-1}^{|m-1|}(\rho) + L_2 R_{n-2}^m(\rho)$$

where the coefficients L_1 and L_2 are

[1] Honarvar Shakibaei, Barmak, and Raveendran Paramesran. "Recursive formula to compute Zernike radial polynomials." Optics letters 38.14 (2013): 2487-2489.

[2] E. C. Kintner, On the Mathematical Properties of the Zemike Polynomials. Opt. Acta 23, 679 (1976)

[3] Janssen, Augustus JEM, and Peter Dirksen. "Computing Zernike polynomials of arbitrary degree using the discrete Fourier transform." Journal of the European Optical Society-Rapid publications 2 (2007).

[4] Chong, Chee-Way, P. Raveendran, and R. Mukundan. "A comparative analysis of algorithms for fast computation of Zernike moments." Pattern Recognition 36.3 (2003): 731-742.

[5] Prata Jr, Aluizio, and W. V. T. Rusch. "Algorithm for computation of Zernike polynomials expansion coefficients." Applied Optics 28.4 (1989): 749-754.

$$L_1 = \frac{2n}{m+n}$$
$$L_2 = \frac{m-n}{m+n}$$

We then start with $R_0^0 = 1$ and can generate all the polynomials.

1.5 Problem 1

Show that the radial polynomials satisfy the recurrence relation

$$R_n^m(\rho) + R_{n-2}^m(\rho) = \rho\left[R_{n-1}^{|m-1|}(\rho) + R_{n-1}^{m+1}(\rho)\right]$$

Janssen and Dirksen have shown that[6]

$$R_n^m(\rho) = \frac{1}{2\pi}\int_{k=0}^{2\pi} U_n \rho\cos\theta\cos\theta \qquad \text{(integral representation of radial polynomials)}$$

where U_n is the Chebyshev polynomial of the second kind and of degree n that satisfies the following recurrence relation for $n \geq 2$[7]

$$U_n(x) = 2xU_{n-1}(x) - U_{n-2}(x)$$

with $U_0(x) = 1$ and $U_1(x) = 2x$.

Clearly, the degree of the radial polynomials is equal to the degree of the Chebshev polynomials of the second kind.

Furthermore, the azimuthal order equals the frequency of the cosine function.

Therefore, simply set $x = \rho\cos\theta$ and multiply by $\cos(m\theta)$ to get

$$U_n(\rho\cos\theta)\cos(m\theta) = 2\rho\cos\theta U_{n-1}(\rho\cos\theta)\cos(m\theta) - U_{n-2}(\rho\cos\theta)\cos(m\theta)$$
$$U_n(\rho\cos\theta)\cos(m\theta) = \rho\Big[\cos(|m-1|\theta) + \cos((m+1)\theta)\Big]U_{n-1}(\rho\cos\theta)cos(m\theta) - U_{n-2}(\rho\cos\theta)\cos(m\theta)$$
$$\int_0^{2\pi} U_n(\rho\cos\theta)\cos(m\theta)\ d\theta = \int_0^{2\pi}\left(\rho\Big[\cos(|m-1|\theta) + \cos((m+1)\theta)\Big]U_{n-1}(\rho\cos\theta)cos(m\theta) - U_{n-2}(\rho\cos\theta)\cos(m\theta)\right)\ d\theta$$

[6] Janssen, Augustus JEM, and Peter Dirksen. "Computing Zernike polynomials of arbitrary degree using the discrete Fourier transform." Journal of the European Optical Society-Rapid publications 2 (2007).

[7] Honarvar Shakibaei, Barmak, and Raveendran Paramesran. "Recursive formula to compute Zernike radial polynomials." Optics letters 38.14 (2013): 2487-2489.

Using the integral representation of radial polynomials, we have

$$R_n^m(\rho) = \rho\Big[R_{n-1}^{|m-1|}(\rho) + R_{n-1}^{m+1}(\rho)\Big] - R_{n-2}^m(\rho)$$

or

$$R_n^m(\rho) + R_{n-2}^m(\rho) = \rho\left[R_{n-1}^{|m-1|}(\rho) + R_{n-1}^{m+1}(\rho)\right]$$

Problem 2. Show that the radial polynomials are orthogonal with respect to the inner product
Two functions F_1 and F_2 are orthogonal over a unit cirle if:

$$\int_0^1 \int_0^{2\pi} F_1 F_2 \rho d\theta d\rho = 0$$

For axisymmetric functions with no θ variance, this reduces to

$$2\pi \int_0^1 F_1 F_2 \rho d\rho = 0$$

Consider $F_1 = \rho^2$ and $F_2 = \rho^4$.

$$2\pi \int_0^1 \rho^2 \rho^4 \rho d\rho = \frac{\pi}{4} \neq 0$$

Consider $F_3 = 2\rho^2 - 1$ and $F_4 = 6\rho^4 - 6\rho^2 + 1$.

$$2\pi \int_0^1 (2\rho^2 - 2)(6\rho^4 - 6\rho^2 + 1)\rho d\rho = 0$$

These are Zernike polynomials. In fact, the Zernike polynomials are created by subtracting lower order polynomials to create this orthogonality relationship.

Orthogonal Functions on the Unit Circle

We seek:

$$\int_0^1 \int_0^{2\pi} R_i(\rho)\Theta_i(\theta) R_j(\rho)\Theta_j(\theta)\rho d\theta d\rho = \delta_j = \begin{cases} C_j & i = j \\ 0 & i \neq j \end{cases}$$

$$\int_0^1 R_i R_j \rho d\rho \int_0^{2\pi} \Theta_i \Theta_j d\theta = \delta_j$$

For a 2π-periodic function, we can chose an orthogonal basis (in fact, orthonormal) in the trigonometric functions $\sin n\theta$ and $\cos n\theta$. Therefore, our problem is reduced to finding an orthogonal radial basis.

$$\int_0^1 R_i R_j \rho d\rho = \delta_j$$

Problem 3. Write a function named `zernike_Rcoeffs` that takes as input n and m and returns the coefficients of $R_n^m(\rho)$ as a 1d array of size $n + 1$.

```
def zernike_Rcoeffs(n,m):
    '''given an n and m returns an numpy array of n+1 elements which are
    the coefficients of their indicial polynomial'''
    coeffs = np.zeros(n+1)

    for k in range((n-m)/2+1):
        coeffs[n-2*k] = ((-1.)**k*math.factorial(n-k))/
          (math.factorial(k)*math.factorial((n+m)/2.-k)*math.factorial((n-m)/2.-k))

    return coeffs


print zernike_Rcoeffs(0,0)
print zernike_Rcoeffs(1,1)
print zernike_Rcoeffs(2,0)
print zernike_Rcoeffs(2,2)
print zernike_Rcoeffs(3,1)
print zernike_Rcoeffs(3,3)
print zernike_Rcoeffs(4,0)
print zernike_Rcoeffs(4,2)
print zernike_Rcoeffs(4,4)
```

```
[ 1.]
[ 0.  1.]
[-1.  0.  2.]
[ 0.  0.  1.]
[ 0. -2.  0.  3.]
[ 0.  0.  0.  1.]
[ 1.  0. -6.  0.  6.]
[ 0.  0. -3.  0.  4.]
[ 0.  0.  0.  0.  1.]
```

Problem 4. Write two functions named `zernike_even` and `zernike_odd` that take as input n, m, ρ, and θ, where $[\rho, \theta]$ represents the mesh grid $[0,1] \times [0, 2\pi]$, and return, respectively, $Z_n^m(\rho, \theta)$ and $Z_n^{-m}(\rho, \theta)$.

This functions should call `zernike_Rcoeffs` and use its output to evaluate the polynomials over $[0,1] \times [0, 2\pi]$.

```
def zernicke_even(n,m,Rho,Theta,density):
    coeffs = zernike_Rcoeffs(n,m)
    Z = np.zeros((density,density))
    for k in range(coeffs.size):
        Z = Z + coeffs[k]*Rho**k*np.cos(m*Theta)
    return Z

def zernicke_odd(n,m,Rho,Theta,density):
    coeffs = zernike_Rcoeffs(n,m)
    Z = np.zeros((density,density))
    for k in range(coeffs.size):
        Z = Z + coeffs[k]*Rho**k*np.sin(m*Theta)
    return Z
```

Problem 5. Write a script that for given n and m:

1. generates 1d arrays $0 \leq \rho \leq 1$ and $0 \leq \theta \leq 2\pi$,
2. generates the mesh grid $[0,1] \times [0, 2\pi]$ from the 1d arrays in 1.,
3. calls `zernike_even` and `zernike_odd`,
4. produces polar plots of $Z_n^m(\rho, \theta)$ and $Z_n^{-m}(\rho, \theta)$ over the unit disk.

Problem 6. Use the script and functions of problems 3 - 5 to generate the plots of the first 15 Zernike polynomials, corresponding to $n = 0$, 1, 2, 3, 4, and the corresponding admissible values of m for each n.

```
density = 100
rho = np.linspace(-1,1,density)
theta = np.linspace(-math.pi,math.pi,density)
Rho, Theta = np.meshgrid(rho,theta)



fig, (ax1) = plt.subplots(figsize=(3,3),ncols=1,subplot_kw=dict(projection='polar'))
ax1.pcolormesh(Theta, Rho, zernicke_even(0,0,Rho,Theta,density))
ax1.get_xaxis().set_visible(False)
ax1.get_yaxis().set_visible(False)
fig, (ax1,ax2) = plt.subplots(figsize=(6,3),ncols=2,subplot_kw=dict(projection='polar'))
ax1.pcolormesh(Theta, Rho, zernicke_odd(1,1,Rho,Theta,density))
ax1.get_xaxis().set_visible(False)
ax1.get_yaxis().set_visible(False)
ax2.pcolormesh(Theta, Rho, zernicke_even(1,1,Rho,Theta,density))
ax2.get_xaxis().set_visible(False)
ax2.get_yaxis().set_visible(False)
```

```
fig, (ax1,ax2,ax3) = plt.subplots(figsize=(9,3),ncols=3,subplot_kw=dict(projection='polar'))
ax1.pcolormesh(Theta, Rho, zernicke_odd(2,2,Rho,Theta,density))
ax1.get_xaxis().set_visible(False)
ax1.get_yaxis().set_visible(False)
ax2.pcolormesh(Theta, Rho, zernicke_even(2,0,Rho,Theta,density))
ax2.get_xaxis().set_visible(False)
ax2.get_yaxis().set_visible(False)
ax3.pcolormesh(Theta, Rho, zernicke_even(2,2,Rho,Theta,density))
ax3.get_xaxis().set_visible(False)
ax3.get_yaxis().set_visible(False)
fig, (ax1,ax2,ax3,ax4) = plt.subplots(figsize=(12,3),ncols=4,subplot_kw=dict(projection='polar'))
ax1.pcolormesh(Theta, Rho, zernicke_odd(3,3,Rho,Theta,density))
ax1.get_xaxis().set_visible(False)
ax1.get_yaxis().set_visible(False)
ax2.pcolormesh(Theta, Rho, zernicke_odd(3,1,Rho,Theta,density))
ax2.get_xaxis().set_visible(False)
ax2.get_yaxis().set_visible(False)
ax3.pcolormesh(Theta, Rho, zernicke_even(3,1,Rho,Theta,density))
ax3.get_xaxis().set_visible(False)
ax3.get_yaxis().set_visible(False)
ax4.pcolormesh(Theta, Rho, zernicke_even(3,3,Rho,Theta,density))
ax4.get_xaxis().set_visible(False)
ax4.get_yaxis().set_visible(False)
fig, (ax1,ax2,ax3,ax4,ax5) = plt.subplots(figsize=(15,3),ncols=5,subplot_kw=dict(projection='polar'))
ax1.pcolormesh(Theta, Rho, zernicke_odd(4,4,Rho,Theta,density))
ax1.get_xaxis().set_visible(False)
ax1.get_yaxis().set_visible(False)
ax2.pcolormesh(Theta, Rho, zernicke_odd(4,2,Rho,Theta,density))
ax2.get_xaxis().set_visible(False)
ax2.get_yaxis().set_visible(False)
ax3.pcolormesh(Theta, Rho, zernicke_even(4,0,Rho,Theta,density))
ax3.get_xaxis().set_visible(False)
ax3.get_yaxis().set_visible(False)
ax4.pcolormesh(Theta, Rho, zernicke_even(4,2,Rho,Theta,density))
ax4.get_xaxis().set_visible(False)
ax4.get_yaxis().set_visible(False)
ax5.pcolormesh(Theta, Rho, zernicke_even(4,4,Rho,Theta,density))
ax5.get_xaxis().set_visible(False)
ax5.get_yaxis().set_visible(False)
```

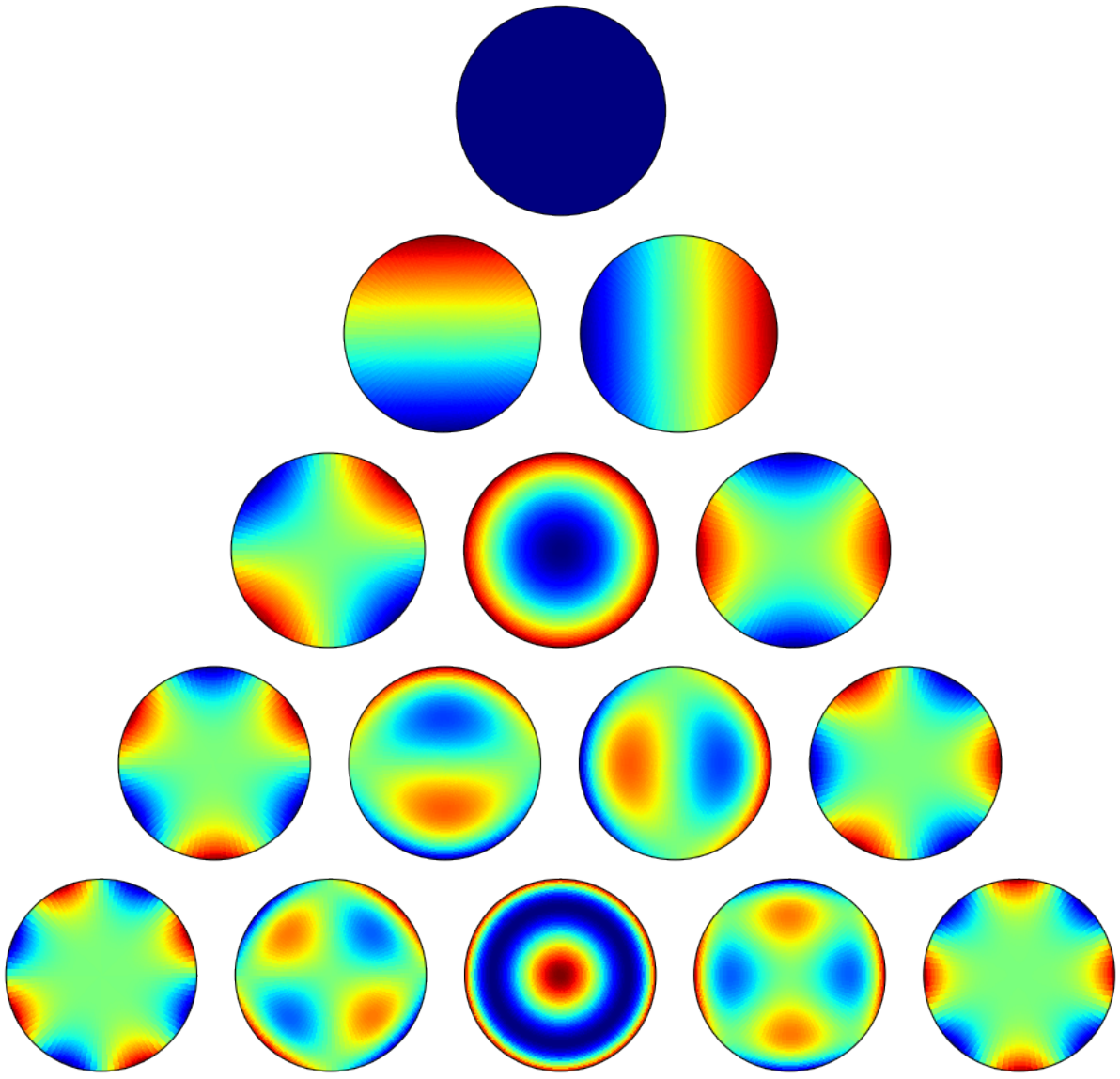

Figure 1: the first 15 Zernike Polynomials