```
from scipy.special import pro_cv, pro_cv_seq, pro_ang1, pro_rad1, pro_rad2
import math
from math import pi
import numpy as np
from numpy import *
import scipy
from scipy.linalg import eigvals

def calculate_alphar(c, m, r):
    a = (((2 * m + r + 2) * (2 * m + r + 1)) / ((2 * m + 2 * r + 5) * (2 * m + 2
* r + 3))) * (c ** 2)
    return(a)

def calculate_betar(c, m, r):
    b = (m + r) * (m + r + 1) + ((2 * (m + r) * (m + r + 1) - 2 * (m ** 2) -
1) / ((2 * m + 2 * r - 1) * (2 * m + 2 * r + 3))) * (c ** 2)
    return(b)

def calculate_gammar(c, m, r):
    g = ((r * (r - 1)) / ((2 * m + 2 * r - 3) * (2 * m + 2 * r - 1))) * (c ** 2)
    return(g)

def lambdamn(c, m, n):
    N = m + n + 50
    #N = m + n + 200
    A = np.zeros((N, N))
    if (mod(n - m, 2) == 0):
        r = 0
    else:
        r = 1
    for i in range(1,N):
        if (i == 1):
            A[1, 1] = calculate_betar(c, m, r)
            A[1, 2] = calculate_alphar(c, m, r)
        else:
            if (i == N):
                A[N, N - 1] = calculate_gammar(c, m, r)
                A[N, N] = calculate_betar(c, m, r)
            else:
                if i+1 < N:
                    A[i, i - 1] = calculate_gammar(c, m, r)
                    A[i, i] = calculate_betar(c, m, r)
                    A[i, i + 1] = calculate_alphar(c, m, r)
        r = r+2
    d = sorted(scipy.linalg.eigvals(A))
    if (mod(n - m, 2) == 0):
        lambdapprox = d[(n - m + 2) // 2]
    else:
        lambdapprox = d[(n - m + 1) // 2]
    return(lambdapprox)

m = 0
c = (2**0.5)/2
print('c = ',c,' m = ',m)
for n in range(-11,11):
    print('n= ',n,' --> ',lambdamn(c, m, n),' procv = ',pro_cv(0,n-2,c))
```