```python
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 21 00:20:16 2024

@author: dnick
"""

#
# Utility functions for CL scheme.
#

import numpy as np
from scipy.linalg import toeplitz
from sympy import Symbol, Poly, expand
import matplotlib.pyplot as plt
import matplotlib
from math import pi


#
#
def _get_min_pos_root(coefs):
    roots = np.roots(coefs)
    opt = roots[np.isreal(roots)]
    opt = opt[opt>0]
    if opt.size>1:
       return opt.min()

    elif opt.size==0:
        return 0

    else:
        return opt.item()

#
#
def calc_beta_opt(s_2,n,g,r):
    b = Symbol('b')
    coefs = Poly(expand(b**(2*n) \
                        - (n+(1+s_2)*n*g*r)*(b**2) \
                        + n-1,b)).all_coeffs()

    return _get_min_pos_root(coefs).real

#
#
def calc_gamma_opt(s_2,r,n):
    a = s_2
    b = r*(1+s_2)

    if n < (1 + 1/b):
        return 0

    g = Symbol('g')
    coefs = Poly(expand(a*((1+b*g)**n) - n*b*(1-g) + b+1,g)).all_coeffs()

    return _get_min_pos_root(coefs).real
```

```
#
#
def calc_q(b,n):
    return np.sqrt((1-b**2)/(1-b**(2*n)))*np.array([b**nn for nn in range(n)]).reshape(-1,1)

#
#
def calc_F(s_2, b, n):
    lead_coef = -(1-b**2)/(b*(1+s_2))
    vec = [0]
    vec += [b**nn for nn in range(n-1)]
    return lead_coef*toeplitz(vec,np.zeros(n))

#
#
def calc_recv_snr(s_2,n,g,r,b):
    numer = (1+s_2) * n * (1-g) * r
    denom = s_2 + b**(2*(n-1))
    return numer / denom

#
#
def MPSK(data, num_constellation_syms=2, symbol_precision=8, tx_pwr=1):
    syms = np.exp(1j * data * 2* pi / num_constellation_syms).round(symbol_precision)

    return np.sqrt(tx_pwr)*syms


if __name__=='__main__':
    RHO = 3
    N = 2
    sigma_2 = 1
    gamma_opt = calc_gamma_opt(sigma_2,RHO,N).real
    beta_opt = np.sqrt((N - 1) / (N+(1+sigma_2)*N*gamma_opt*RHO)).real
    q = calc_q(beta_opt,N)
    F = calc_F(sigma_2, beta_opt, N)
    print(gamma_opt)
    print(beta_opt)
    print(q)
    print(F.round(3))

    font = {'family' : 'normal',
            'weight' : 'normal',
            'size' : 15}
    matplotlib.rc('font',**font)
    gamma_opt = [[],[],[]]
    rhos = np.linspace(.2,10,100)
    for r in rhos:
        gamma_opt[0].append(calc_gamma_opt(1,r,3))
        gamma_opt[1].append(calc_gamma_opt(1,r,10))
        gamma_opt[2].append(calc_gamma_opt(.1,r,10))


    plt.plot(rhos,gamma_opt[0],label=r'$N=3,\sigma^{2}=1$')
    plt.plot(rhos,gamma_opt[1],label=r'$N=10,\sigma^{2}=1$')
    plt.plot(rhos,gamma_opt[2],label=r'$N=10,\sigma^{2}=0.1$')
```

```
plt.xlabel(r'$\rho$')
plt.ylabel(r'$\gamma_{opt}$')
plt.grid()
plt.legend()


sigmas = 10**(np.linspace(-20,5,100)/10)
r = 1
Ns = [3,7,10]
snrs = [[] for _ in range(len(Ns))]
for idx,n in enumerate(Ns):
    for sig in sigmas:
        gamma_opt = calc_gamma_opt(sig,r,n).real
        beta_opt = calc_beta_opt(sig, n, gamma_opt, r)
        snrs[idx].append(calc_recv_snr(sig,n,gamma_opt,r,beta_opt))

plt.figure()
for i,s in enumerate(snrs):
    plt.plot(10*np.log10(sigmas),10*np.log10(s),label=f'N={Ns[i]}')
plt.xlabel(r'$\sigma^{2}$ (dB)')
plt.ylabel('Received SNR (dB)')
plt.grid()
plt.legend()
```