Table of Contents

Simulation parameters

```
clear all
close all

sim_times = 1e5; % number of simulation trials

N = 10; % number of reflecting elements
% Note: bigger N, longer running time

snr_dB = 0:2:20; % average transmit SNR in dB
R_th = 1; % SE threshold b/s/Hz
snr_th = 2^R_th - 1;

d_Sr = 1+rand; % random distance S->RIS
d_rD = 1+rand; % random distance RIS->D
PLE = 2.7; % path-loss exponent

% Nakagami-m parameters
m_Sr = 2.5 + rand; % random shape, S->RIS
m_rD = 2.5 + rand; % random shape, RIS->D
Omega_Sr = d_Sr^(-PLE); % random spread, S->RIS
Omega_rD = d_rD^(-PLE); % random spread, RIS->D

% Inverse Gammar (IG)'s parameters
alpha_Sr = 3.0+rand; % random shape, S->RIS
alpha_rD = 3.0+rand; % random shape, RIS->D
beta_Sr = 1; % random spread, S->RIS
beta_rD = 1; % random spread, RIS->D

Z_sim = 0;
kappa = 1; % for RIS
```

Channel modeling

```
% Nakagami-m fading channel
G_Sr = random('Naka', m_Sr, Omega_Sr, [N, sim_times]);
G_rD = random('Naka', m_rD, Omega_rD, [N, sim_times]);

% Inverse Gamma shadowing
L_Sr = 1./random('Gamma', alpha_Sr, 1/beta_Sr, [N, sim_times]);
L_rD = 1./random('Gamma', alpha_rD, 1/beta_rD, [N, sim_times]);
```

```
% Here, beta is the "rate"

Gr = G_Sr.*G_rD; % e2e fading w.r.t. one element
Lr = L_Sr.*L_rD; % e2e shadowing w.r.t. one element

W_sim = Gr.*Lr; % e2e channel w.r.t. one element
Z_sim = sum(W_sim,1); % e2e chanel w.r.t. the whole RIS
% end

Z2_sim = Z_sim.^2; % e2e squared magnitude
```

new

```
% phase of channels
phase_Sr = 2*pi*rand(N, sim_times); % domain [0,2pi)
phase_rD = 2*pi*rand(N, sim_times); % domain [0,2pi)

% Channel modeling
G_Sr_complex_fading = G_Sr .* exp(1i*phase_Sr);
G_rD_complex_fading = G_rD .* exp(1i*phase_rD);

% Phase-shift configuration
Z_sim_optimal_phase_shift = zeros(1,sim_times);
for ss = 1:sim_times % loop over simulation trials
    for ll = 1:N % loop over each elements of the RIS
        % unknown domain phase-shift
        phase_shift_element_temp(ll,ss) = - phase_Sr(ll,ss) -
 phase_rD(ll,ss);

        % convert to domain of [0, 2pi)
        phase_shift_element(ll,ss) =
 wrapTo2Pi(phase_shift_element_temp(ll,ss));

        Gr_optimal_phase_shift(ll,ss) = abs(G_Sr_complex_fading(ll,ss)
 * ...
            exp(1i*phase_shift_element(ll,ss)) *
 G_rD_complex_fading(ll,ss));
        W_optimal_phase_shift(ll,ss) =
 Gr_optimal_phase_shift(ll,ss)*Lr(ll,ss);
    end
end
%
% Z_sim = sum(abs(W_e2e),1); % Magnitude of the e2e channel
Z_sim_optimal_phase_shift = sum(W_optimal_phase_shift,1);
% Magnitude of the e2e channel

Z2_sim_optimal_phase_shift = Z_sim_optimal_phase_shift.^2;
```

Analysis

```
% STEP-1: MOMENT MATCHING Gr -> Gamma
Upsilon_G = m_Sr*m_rD/Omega_Sr/Omega_rD;
Omega_G = gamma(m_Sr + 1/2) * gamma(m_rD+1/2) / ...
```

```
    gamma(m_Sr) / gamma(m_rD) * Upsilon_G^(-1/2);
m_G = Omega_G^2/ (Omega_Sr*Omega_rD - Omega_G^2);

% Check the analytical parameters by fitting
distParam = fitdist(Gr(:), 'gamma'); % Gr(:) is a column vector
m_G_fit = distParam.a;
Omega_G_fit = distParam.b*distParam.a;

% STEP-2: MOMENT MATCHING 1./sqrt(Lr) -> Gamma
Omega_L = gamma(alpha_Sr+1/2)*gamma(alpha_rD+1/2) / ...
    sqrt(beta_Sr*beta_rD) / gamma(alpha_Sr) / gamma(alpha_rD);
m_L = Omega_L^2 / (alpha_Sr*alpha_rD/beta_Sr/beta_rD - Omega_L^2);

% Check the analytical parameters by fitting
distParam = fitdist(1./sqrt(Lr(:)), 'gamma');
m_L_fit = distParam.a;
Omega_L_fit = distParam.b*distParam.a;

% STEP-3: APPROXIMATE THE PDF of \tilde{R}_{r} USING MG
K = 5; % number of terms in G-L quadrature
[ z_W, w_W, ~ ] = gengausslegquadrule(K); % G-L abscissas and weights
f_W_n = @(r) 0;

for kk = 1:K
    %
    zeta_W = @(x) m_G/Omega_G*(z_W(x)*Omega_L/m_L).^2;
    theta_W = @(x) w_W(x)/gamma(m_L)*zeta_W(x)^m_G*z_W(x)^(m_L-1);%->
 psi_k
    alpha_W = @(x) 0; %-> xi_k
    %
    for ii = 1:K
        alpha_W = @(x) alpha_W(x)...
            + theta_W(ii)*zeta_W(ii)^(-m_G);
    end
    %
    alpha_W = @(x) theta_W(x)/alpha_W(x);
    %
    f_W_n = @(r) f_W_n(r)...
        + alpha_W(kk)/gamma(m_G).*r.^(m_G-1).*exp(-zeta_W(kk)*r);
end

% STEP-4: LAPLACE TEST
L_W_fit = @(s) ( integral(@(r) exp(-s*r).*f_W_n(r), 0, Inf) )^N;
L_W_exact= @(s) mean( exp(-s*Z_sim) );

zz = linspace(0, max(max(Z_sim)), 100);
for tt = 1:length(zz)
    point_L_W_exact(tt) = L_W_exact(zz(tt));
    point_L_W_fit(tt) = L_W_fit(zz(tt));
end

figure(1);
plot(zz, point_L_W_exact); hold on;
plot(zz, point_L_W_fit, '+'); hold on;
```

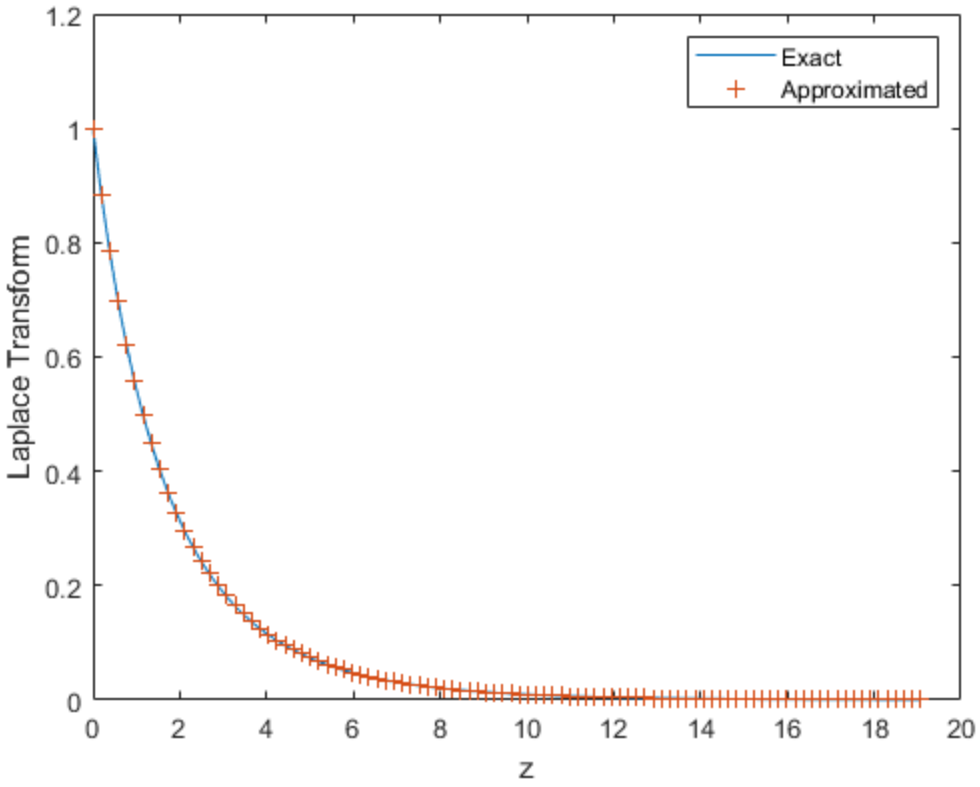
```
ylabel('Laplace Transform');
xlabel('z');
legend('Exact', 'Approximated');

% STEP-5: DERIVE CDF and PDF pf \sum_{r=1}^N( \tilde{W}_{r} )
[~, setInd] = nsumk(K, N);

F_Z_fit = @(x) 0;

for caseIndex = 1:size(setInd, 1)
    indices= setInd(caseIndex, :);
    %
    prodAlp = 1;
    for kk = 1:K
        vk = indices(kk);
        %
        prodAlp = prodAlp * alpha_W(kk)^vk;
        %
    end
    %
    F_Z_fit = @(x) F_Z_fit(x)...
        + factorial(N)/prod(factorial(indices))...
        * prodAlp * Phi2(indices.*m_G, 1+m_G*N, -zeta_W(1:K)*x, 50);
    % Phi2 is Humbert function
end

F_Z_fit = @(x) F_Z_fit(x).*x.^(m_G*N)/gamma(1+m_G*N);
F_Z2_fit= @(y) F_Z_fit(sqrt(y)); % F_Y (y) = F_X (sqrt(y))
```



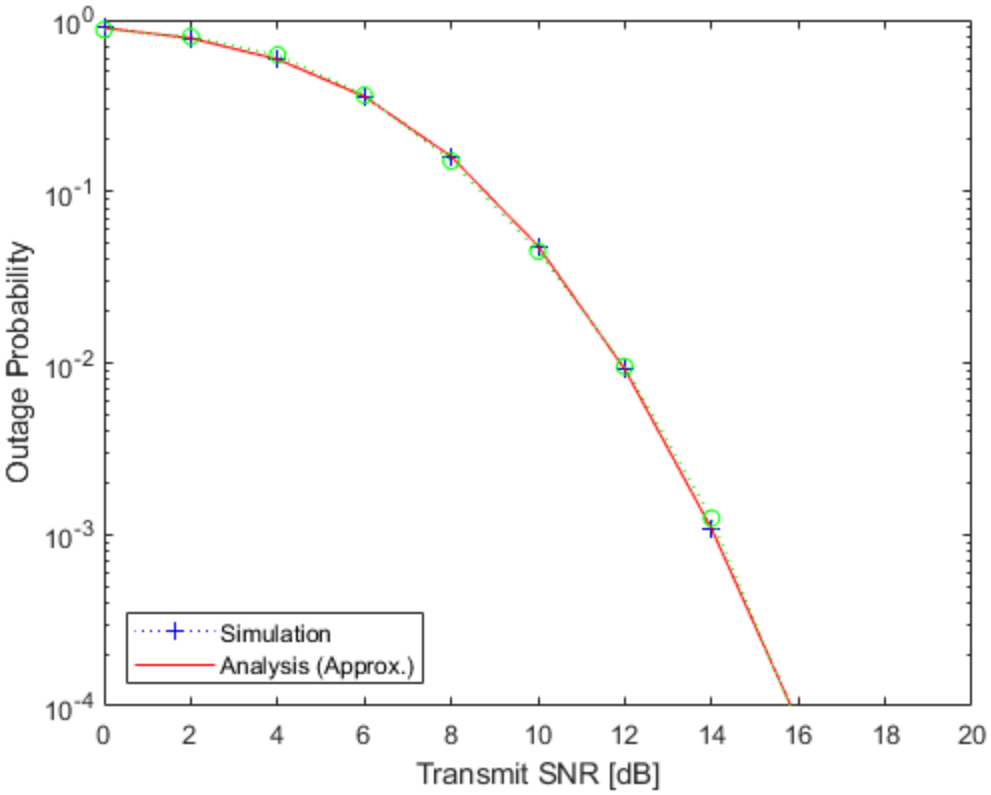
Outage probability

```
for ii = 1:length(snr_dB)
    snr = 10^(snr_dB(ii)/10);
    %
    OP_sim(ii) = mean( Z2_sim < snr_th/snr );
    OP_sim_new(ii) = mean( Z2_sim_optimal_phase_shift < snr_th/snr );
    OP_ana(ii) = F_Z2_fit( snr_th/snr );
end
```

Plotting results

```
figure(2);
semilogy(snr_dB, OP_sim, 'b+:'); hold on;
semilogy(snr_dB, OP_sim_new, 'r-'); hold on;

semilogy(snr_dB, OP_ana, 'g:o'); hold on;
xlabel('Transmit SNR [dB]');
ylabel('Outage Probability');
ldg = legend('Simulation', 'Analysis
 (Approx.)', 'location', 'southwest');
axis([-Inf Inf 10^(-4) 1]);
```



Published with MATLAB® R2020b