Table of Contents

PAPER

```
% Title: Multi-RIS-aided Wireless Systems: Statistical
 Characterization and Performance Analysis
% Authors : Tri Nhu Do, Georges Kaddoum, Thanh Luan Nguyen, Daniel
 Benevides da Costa, Zygmunt J. Haas
% Online: https://github.com/trinhudo/Multi-RIS
% Version: 12-Sep-2021

% Multiple RISs with detailed phase-shift configuration
% --ERA scheme: all RISs participate
% --ORA scheme: only the best RIS participates
% --Analysis is based on Log-Normal distribution

tic
% rng('default');
```

SETTING

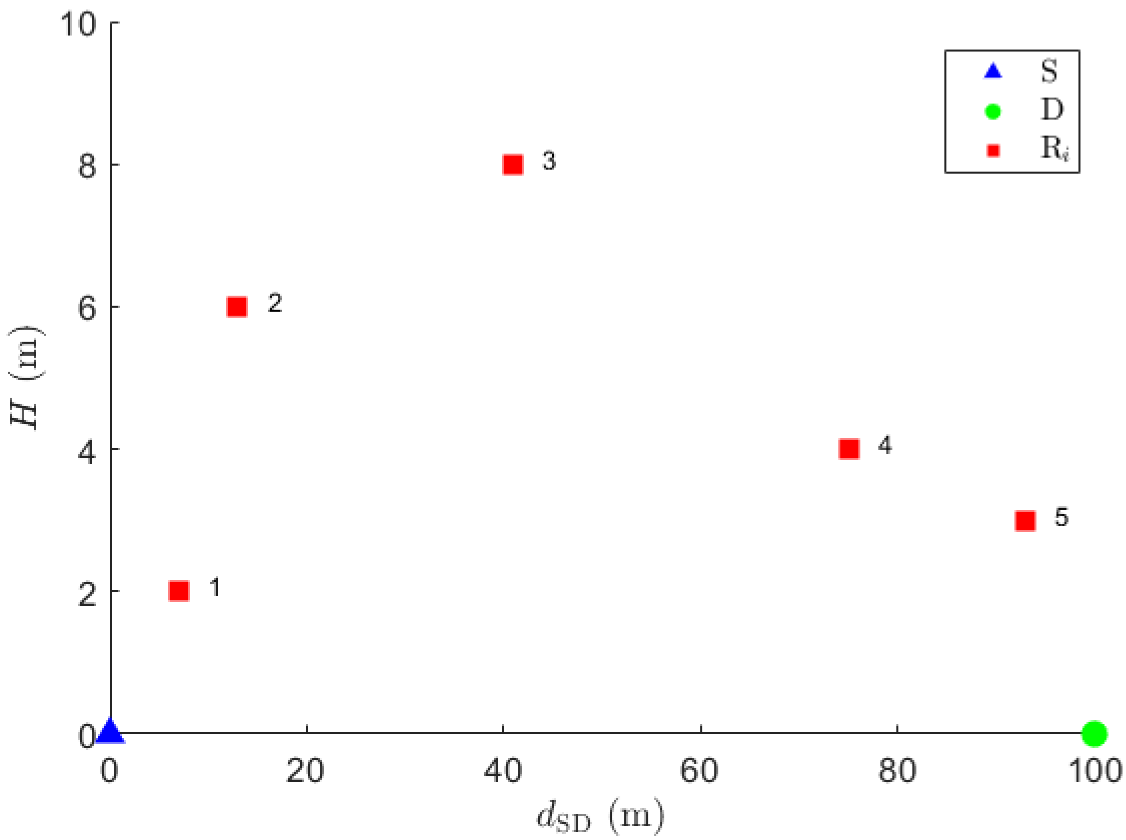
```
clear all
close all

sim_times = 1e5; % Number of simulation trails

R_th = 1; % Predefined target spectral efficiency [b/s/Hz]

SNR_th = 2^R_th-1; % Predefined SNR threshold

N_RIS = 5; % Number of distributed RISs

L_single = 25; % Number of elements at each RIS

L = L_single*ones(1,N_RIS); % all RISs

kappa_nl = 1; % Amplitude reflection coefficient
```

```
% Network area
x_area_min = 0;
x_area_max = 100; % in meters
y_area_min = 0;
y_area_max = 10;

% Source location
x_source = x_area_min;
y_source = y_area_min;

% Destination location
x_des = x_area_max;
y_des = y_area_min;

% Random location setting
% x_RIS = x_area_min + (x_area_max-x_area_min)*rand(N_RIS, 1); %
 [num_RIS x 1] vector
% y_RIS = y_area_min + (y_area_max-y_area_min)*rand(N_RIS, 1);

%Location setting D1
x_RIS = [7; 13; 41; 75; 93];
y_RIS = [2; 6; 8; 4; 3];

% Compute location of nodes
pos_source = [x_source, y_source];

pos_des = [x_des, y_des];
pos_RIS = [x_RIS, y_RIS]; % [num_RIS x 2] matrix

% Compute distances
d_SR = sqrt(sum((pos_source - pos_RIS).^2 , 2)); % [num_RIS x 1]
 vector
d_RD = sqrt(sum((pos_RIS - pos_des).^2 , 2));
d_SD = sqrt(sum((pos_source - pos_des).^2 , 2));
```

NETWORK TOPOLOGY

```
figure;

scatter(x_source, y_source, 100, 'b^', 'filled'); hold on
scatter(x_des, y_des, 100, 'go', 'filled'); hold on
scatter(x_RIS, y_RIS, 100, 'rs', 'filled'); hold on

for kk = 1:N_RIS
    text(x_RIS(kk)+3, y_RIS(kk)+0.1, num2str(kk));
    hold on
end

xlabel('$d_{\rm SD}$ (m)', 'Interpreter', 'Latex')
ylabel('$H$ (m)', 'Interpreter', 'Latex')
axis([x_area_min x_area_max y_area_min y_area_max])
legend('$\rm S$', '$\rm D$', '$\mathrm{R}_i$',...
```

```
    'Interpreter', 'Latex',...
    'Location', 'best')

set(gca, 'LooseInset', get(gca, 'TightInset')) % remove plot padding
set(gca,'fontsize',13);
hold off

% Path-loss model
% ---------------

% Carrier frequency (in GHz)
fc = 3; % GHz

% 3GPP Urban Micro in 3GPP TS 36.814
% NLoS path-loss component based on distance, x is in meter
pathloss_NLOS = @(x) db2pow(-22.7 - 26*log10(fc) - 36.7*log10(x));

antenna_gain_S = db2pow(5); % Source antenna gain, dBi
antenna_gain_RIS = db2pow(5); % Gain of each element of a RIS, dBi
antenna_gain_D = db2pow(0); % Destination antenna gain, dBi

% Noise power and Transmit power P_S
% ----------------------------------

% Bandwidth
BW = 10e6; % 10 MHz

% Noise figure (in dB)
noiseFiguredB = 10;

% Compute the noise power in dBm
sigma2dBm = -174 + 10*log10(BW) + noiseFiguredB; % -94 dBm
sigma2 = db2pow(sigma2dBm);

P_S_dB = -5:25; % Transmit power of the source, dBm, e.g., 200mW =
 23dBm

SNRdB = P_S_dB - sigma2dBm; % Average transmit SNR, dB = dBm - dBm,
 bar{rho} = P_S / sigma2
```



SIMULATION | ERA SCHEME

```
% Nakagami scale parameter
m_0 = 2.5 + rand; % S -> D, scale parameter, heuristic setting

m_h = 2.5 + rand(N_RIS, 1); % S -> RIS

m_g = 2.5 + rand(N_RIS, 1); % RIS -> D

% Nakagami spread parameter

Omega_0 = 1; % Normalized spread parameter of S->D link
Omega_h = 1; % Normalized spread parameter of S->RIS link
Omega_g = 1; % Normalized spread parameter of RIS->D link

% Path-loss

path_loss_0 = pathloss_NLOS(d_SD)*antenna_gain_S; % S -> D link

path_loss_h = pathloss_NLOS(d_SR) * ...
    antenna_gain_S*antenna_gain_RIS*L_single; % S -> RIS

path_loss_g = pathloss_NLOS(d_RD) * ...
    antenna_gain_RIS*L_single*antenna_gain_D; % RIS -> D
```

```
% Phase of channels

phase_h_SD = 2*pi*rand(1, sim_times); % domain [0,2pi)
phase_h_SR = 2*pi*rand(N_RIS, L_single, sim_times); % domain [0,2pi)
phase_g_RD = 2*pi*rand(N_RIS, L_single, sim_times); % domain [0,2pi)

phase_h_SR_eachRIS = zeros(L_single, sim_times);
phase_g_RD_eachRIS = zeros(L_single, sim_times);

% Channel modeling

h_SD = sqrt(path_loss_0) * ... % need sqrt because path-loss is
 outside of random()
    random('Naka', m_0, Omega_0, [1, sim_times]) .* ...
    exp(1i*phase_h_SD);

h_SR = zeros(N_RIS,L_single,sim_times); % S to RIS channel
g_RD = zeros(N_RIS,L_single,sim_times); % RIS to D channel

for nn = 1:N_RIS
    phase_h_SR_eachRIS = squeeze(phase_h_SR(nn,:,:)); % random() just
 uses 2D
    phase_g_RD_eachRIS = squeeze(phase_g_RD(nn,:,:)); % random() just
 uses 2D

    for kk=1:L(nn)
        h_SR(nn,kk,:) = sqrt(path_loss_h(nn)) .* ... % need sqrt
 because path-loss is outside of random()
            random('Naka', m_h(nn), Omega_h, [1, sim_times]) .* ...
            exp(1i*phase_h_SR_eachRIS(kk,:));

        g_RD(nn,kk,:) = sqrt(path_loss_g(nn)) .* ... % need sqrt
 because path-loss is outside of random()
            random('Naka', m_g(nn), Omega_g, [1, sim_times]) .* ...
            exp(1i*phase_g_RD_eachRIS(kk,:));
    end
end

% Phase-shift Configuration for ERA scheme
%-----------------------------------------

h_ERA_cascade = zeros(N_RIS, sim_times); % matrix of cascade channel
 S-via-RIS-to-D

for ss = 1:sim_times % loop over simulation trials
    phase_shift_config_ideal = zeros(L_single,1);
    phase_shift_config_ideal_normalized = zeros(L_single,1);
    phase_shift_complex_vector = zeros(L_single,1);

    for nn=1:N_RIS % loop over each RIS
        for ll = 1:L_single % loop over each elements of one RIS
            % Unknown domain phase-shift
            phase_shift_config_ideal(ll) = ...
```

```
                 phase_h_SD(ss) - phase_h_SR(nn,ll,ss) -
 phase_g_RD(nn,ll,ss);
            % Convert to domain of [0, 2pi)
            phase_shift_config_ideal_normalized(ll) =
 wrapTo2Pi(phase_shift_config_ideal(ll));
            phase_shift_complex_vector(ll) =
 exp(1i*phase_shift_config_ideal_normalized(ll));
        end

        phase_shift_matrix = kappa_nl .*
 diag(phase_shift_complex_vector);

        % cascade channel (complex, not magnitude)
        h_ERA_cascade(nn,ss) =  h_SR(nn,:,ss) * phase_shift_matrix *
 g_RD(nn,:,ss).'; % returns a number
    end
end

h_ERA_e2e_magnitude = abs(h_SD + sum(h_ERA_cascade,1)); % direct +
 cascade channels

Z_ERA = h_ERA_e2e_magnitude; % RV Z in the analysis

Z2_ERA = Z_ERA.^2; % RV Z^2
```

SIMULATION | ORA SCHEME (BEST RIS SELECTION)

```
% Simple simulation
%------------------

% V_M_ORA = max(h_e2e_RIS_path, [], 1); %V_M for the best RIS
% R_ORA   = abs(h_SD + V_M_ORA); %Magnitude of the e2e channel
% R2_ORA  = R_ORA.^2; %Squared magnitude of the e2e channel

% Detailed simulation
%--------------------

h_ORA_cascade = zeros(1, sim_times);

[~,idx] = max(h_ERA_cascade,[],1);

for ss = 1:sim_times
    phase_shift_config_ideal = zeros(L_single,1);
    phase_shift_config_ideal_normalized = zeros(L_single,1);
    phase_shift_complex_vector = zeros(L_single,1);
    for ll = 1:L_single % loop over each elements of one RIS
        phase_shift_config_ideal(ll) = phase_h_SD(ss) -
 phase_h_SR(idx(ss),ll,ss) - phase_g_RD(idx(ss),ll,ss);
        phase_shift_config_ideal_normalized(ll) =
 wrapTo2Pi(phase_shift_config_ideal(ll));
```

```
        phase_shift_complex_vector(ll) =
 exp(1i*phase_shift_config_ideal_normalized(ll));
    end

    phase_shift_matrix = kappa_nl .* diag(phase_shift_complex_vector);

    % e2e channel coefficient (complex number, not magnitude)
    h_ERA_cascade(idx(ss),ss) =  h_SR(idx(ss),:,ss) *
 phase_shift_matrix * g_RD(idx(ss),:,ss).';

    h_ORA_cascade(ss) = h_ERA_cascade(idx(ss),ss);
end

h_ORA_e2e_magnitude = abs(h_SD + h_ORA_cascade);

R_ORA = h_ORA_e2e_magnitude; % RV R in the analysis

R2_ORA = h_ORA_e2e_magnitude.^2; % RV R^2
```

ANALYSIS | ERA SCHEME | LOG-NORMAL DISTRIBUTION

```
Omg_0 = Omega_0*path_loss_0;
Omg_h = Omega_h*path_loss_h;
Omg_g = Omega_g*path_loss_g;

lambda = sqrt(m_h./Omg_h .* m_g./Omg_g) ./ kappa_nl; % lambda_nl

% Working on h0
%--------------

% Working on h0
% The k-th moment of h0
E_h0_k = @(k) gamma(m_0+k/2)/gamma(m_0)*(m_0/Omg_0)^(-k/2);

F_h0 = @(x) gammainc(m_0*x.^2/Omg_0, m_0, 'lower');

% Working on U_nl
%----------------

% The k-moment of U_nl
E_U_nl_k = @(k, n) lambda(n)^(-k)*gamma(m_h(n)+0.5*k)...
    * gamma(m_g(n)+0.5*k) / gamma(m_h(n)) / gamma(m_g(n));

% Parameter of the approximate Gamma distribution of U_nl
alpha_U = @(n) E_U_nl_k(1, n)^2/(E_U_nl_k(2, n)-E_U_nl_k(1, n)^2);
beta_U = @(n) E_U_nl_k(1, n)/(E_U_nl_k(2, n)-E_U_nl_k(1, n)^2);

% Working on V_n
%---------------

% The k-moment of V_n
```

```
E_V_n_k = @(k, n) gamma(L(n) * alpha_U(n)+k) ...
    / gamma(L(n) * alpha_U(n)) * beta_U(n)^(-k);

% Working on T
%-------------

%The 1st moment of T
E_T1 = 0;
for n = 1:N_RIS
    for l = 1:L(n)
        E_T1 = E_T1 + E_U_nl_k(1, n);
    end
end

%The 2nd  moment of T
E_T2 = 0;
for n = 1:N_RIS
    tmpA = 0;
    for l = 1:L(n)
        tmpA = tmpA + E_U_nl_k(1, n);
    end
    for ii = n+1:N_RIS
        tmpB = 0;
        for l = 1:L(ii)
            tmpB = tmpB + E_U_nl_k(1, ii);
        end
        E_T2 = E_T2 + 2 * tmpA * tmpB;
    end
end

for n = 1:N_RIS
    tmpC = 0;
    for l = 1:L(n)
        tmpC = tmpC + E_U_nl_k(2, n);
    end
    tmpD = 0;
    for l = 1:L(n)
        for v = (l+1):L(n)
            tmpD = tmpD + 2 * E_U_nl_k(1, n) * E_U_nl_k(1, n);
        end
    end
    E_T2 = E_T2 + tmpC + tmpD;
end

E_Z = E_h0_k(1) + E_T1; % 1st moment

E_Z2 = E_h0_k(2) + E_T2 + 2 * E_h0_k(1) * E_T1; % 2nd moment

% Fit Z_ERA to Log-Normal
%-------------------------

E_Z4 = 0; % the 2nd moment of Z^2, i.e., E[(Z^2)^2], 4th moment of Z
for k = 0:4
    E_Tk = 0; % E[T^k] % the k-th momemt of T >> CAN BE USE IN ERA ???
```

```
    [cases_T, indT_mat] = nsumk(N_RIS, k);
    for icaseT = 1:cases_T
        indT_arr = indT_mat(icaseT, :);
        tmpT = 1;
        for t = 1:N_RIS
            tmpT = tmpT * E_V_n_k(indT_arr(t), t);
        end
        E_Tk = E_Tk + factorial(k)/prod( factorial(indT_arr) ) * tmpT;
    end
    E_Z4 = E_Z4 + nchoosek(4, k)*E_h0_k(4-k)*E_Tk;
end

nu_Z2_ERA_LN = log( E_Z2^2/sqrt(E_Z4) ); % for Z2 in ERA, used in EC
zeta2_Z2_ERA_LN = log( E_Z4/E_Z2^2 ); % for Z2 in ERA, used in EC

nu_Z_ERA_LN = log(E_Z^2 / sqrt(E_Z2) ); % for Z in ERA
zeta2_Z_ERA_LN = log( E_Z2 / (E_Z^2) ); % for Z in ERA

% CDF of Z
F_Z_ERA_new = ...
    @(x) 1/2 + 1/2*erf( (log(x)-nu_Z_ERA_LN)/sqrt(2*zeta2_Z_ERA_LN) );

% CDF of Z^2
F_Z2_ERA_LN = ...
    @(x) 1/2 + 1/2*erf( (log(x)-nu_Z2_ERA_LN)/
sqrt(2*zeta2_Z2_ERA_LN) );
```

ANALYSIS | ORA SCHEME | LOG-NORMAL DISTRIBUTION

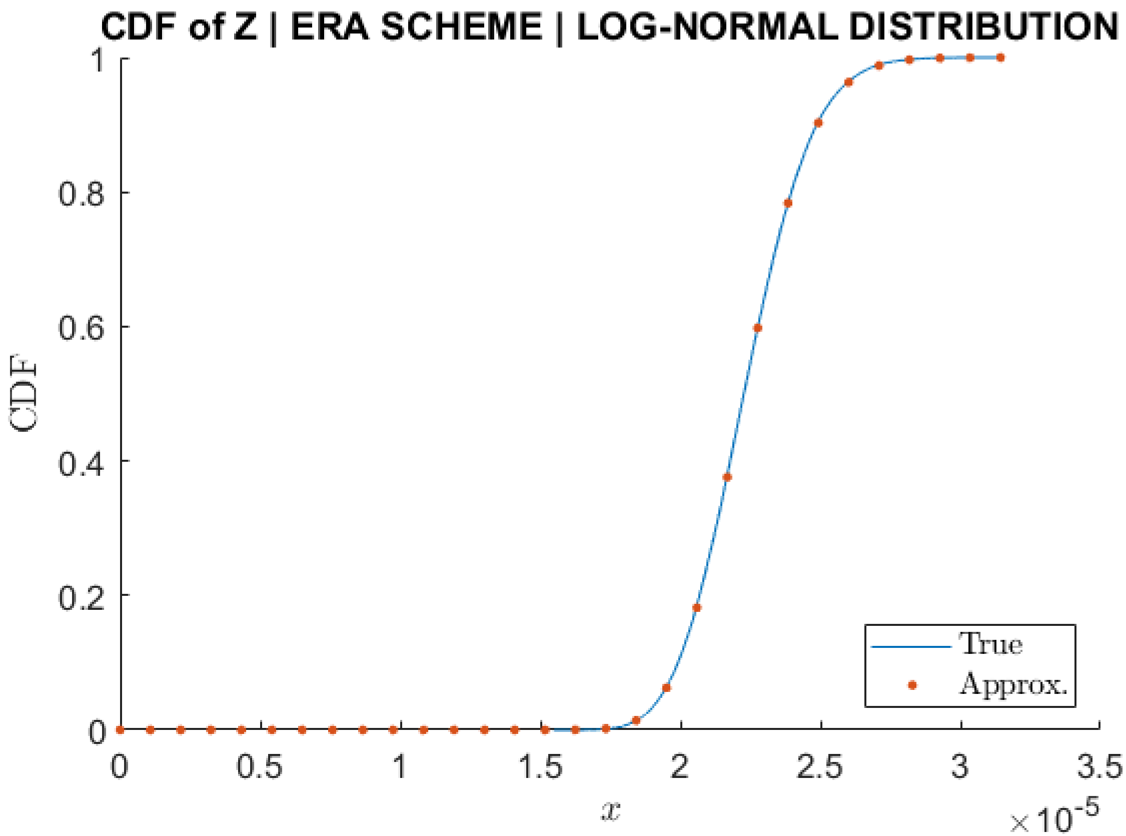
```
alpha_U_arr = zeros(1, N_RIS);
beta_U_arr = zeros(1, N_RIS);

for n = 1:N_RIS
    alpha_U_arr(n) = alpha_U(n);
    beta_U_arr(n) = beta_U(n);
end
%
chi_t= @(t) beta_U_arr(t) ./ sum(beta_U_arr);

% Approxiate result, using self-built F_A() function
%---------------------------------------------------

alpha_U_arr= zeros(1, N_RIS);
beta_U_arr = zeros(1, N_RIS);
for n = 1:N_RIS
    alpha_U_arr(n) = alpha_U(n);
    beta_U_arr(n) = beta_U(n);
end

f_V_n = @(v, n) beta_U(n)^(L(n)*alpha_U(n))/gamma(L(n)*alpha_U(n))...
    * v.^(L(n)*alpha_U(n)-1) .* exp( -beta_U(n)*v );
```

```
F_V_n = @(v, n) gammainc(beta_U(n)*v, L(n)*alpha_U(n), 'lower');

f_M_V = @(x) 0;

for n = 1:N_RIS
    func_tmp = @(x) 1;
    for t = 1:N_RIS
        if (n ~= t)
            func_tmp = @(x) func_tmp(x) .* F_V_n(x, t);
        end
    end
    f_M_V = @(x) f_M_V(x) + f_V_n(x, n) .* func_tmp(x);
end

% mu_M_V = zeros(1, 4); % the k-th moment of M_V (k = 1, 2, 3, 4)
% for k = 1:4
%     mu_M_V(k) = integral(@(x) x.^k .* f_M_V(x), 0, 250);
% end

X = sym('X', [1, N_RIS]);
mu_M_V = sym(zeros(1, 4)); % the k-th moment of M_V (k = 1, 2, 3, 4)
for k = 1:4
    for n = 1:N_RIS
        %
        Sn = setdiff(1:N_RIS, n);
        %
        tmp = Lauricella_FA(sum(L.*alpha_U_arr)+k, ones(1, N_RIS-1),
 L(Sn).*alpha_U_arr(Sn)+1, chi_t(Sn));
        %
        if (tmp > 0)
            mu_M_V(k) = mu_M_V(k) + gamma( sum(X)+k ) / gamma(X(n))...
                / prod( gamma(X(Sn)+1) ) * sym(tmp);
        end
    end
    mu_M_V(k) = vpa(subs(sum(beta_U_arr)^(-k) *
 prod( chi_t(1:N_RIS).^(X) ) * mu_M_V(k), X, L.*alpha_U_arr));
end
mu_M_V = double(mu_M_V);


E_R = 0; % E[R]
for k = 0:1
    if k >= 1
        E_R = E_R + nchoosek(1, k) * E_h0_k(1-k) * mu_M_V(k);
    else
        E_R = E_R + E_h0_k(1);
    end
end

E_R_2 = 0; % E[R^2] by using R^2 expressions, not R
for k = 0:2
    if k >= 1
        E_R_2 = E_R_2 + nchoosek(2, k) * E_h0_k(2-k) * mu_M_V(k);
    else
```

```
        E_R_2 = E_R_2 + E_h0_k(2);
    end
end

E_R_4 = 0; % E[(R^2)^2] by using R^2 expressions, not R
for k = 0:4
    if k >= 1
        E_R_4 = E_R_4 + nchoosek(4, k) * E_h0_k(4-k) * mu_M_V(k);
    else
        E_R_4 = E_R_4 + E_h0_k(4);
    end
end

nu_R_ORA_LN = log( E_R^2/sqrt(E_R_2) ); % for R in ORA
zeta2_R_ORA_LN = log( E_R_2/E_R^2 ); % for R in ORA

nu_R2_ORA_LN = log( E_R_2^2/sqrt(E_R_4) ); % for R^2 in ORA
zeta2_R2_ORA_LN = log( E_R_4/E_R_2^2 ); % for R^2 in ORA

F_R_ORA_LN = @(x) 1/2 + ...
    1/2*erf( (log(x)-nu_R_ORA_LN)/sqrt(2*zeta2_R_ORA_LN) ); % CDF of R

F_R2_ORA_LN = @(x) 1/2 + ...
    1/2*erf( (log(x)-nu_R2_ORA_LN)/sqrt(2*zeta2_R2_ORA_LN) ); % CDF of
 R^2
```

CDF of Z | ERA SCHEME | LOG-NORMAL DISTRIBUTION

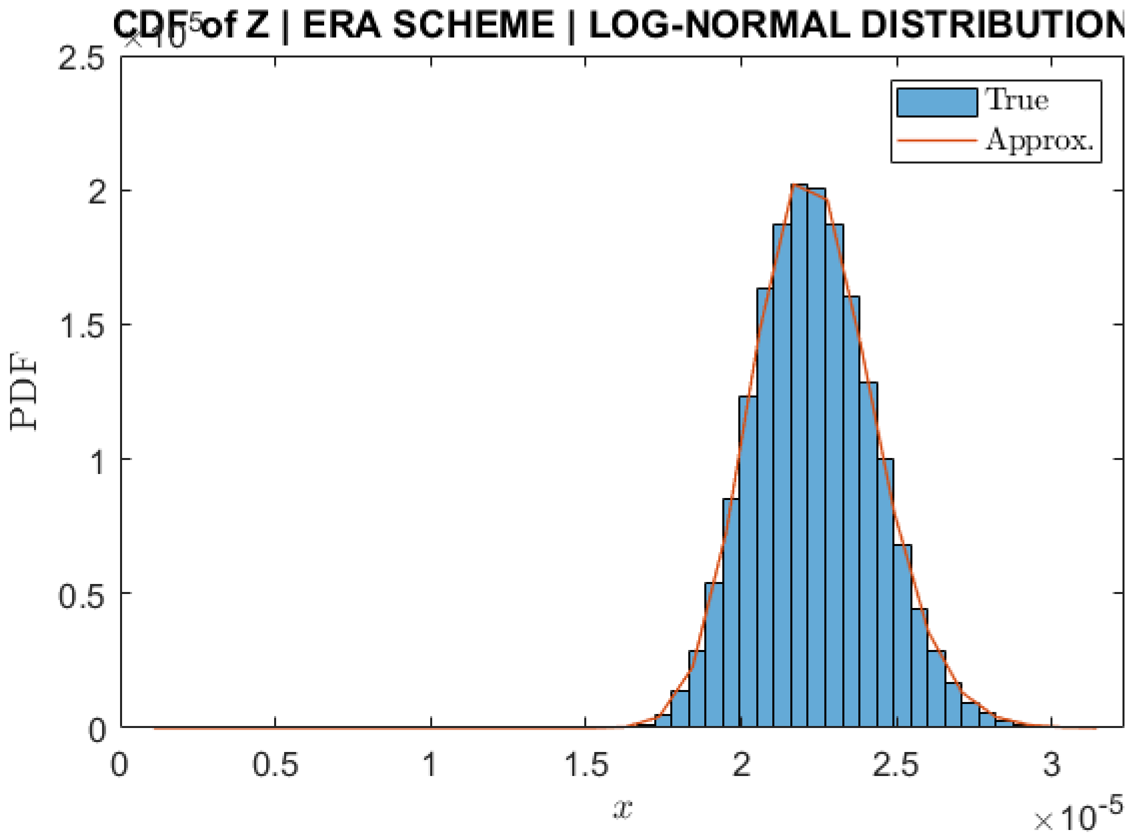
```
figure;

[y, x] = ecdf(Z_ERA); hold on;
domain_Z = linspace(0, max(x), 30);

plot(x, y); hold on;
plot(domain_Z, F_Z_ERA_new(domain_Z), '.', 'markersize', 10); hold on;

title('CDF of Z | ERA SCHEME | LOG-NORMAL DISTRIBUTION')
xlabel('$x$', 'Interpreter', 'Latex')
ylabel('CDF','Interpreter', 'Latex')
legend('True',...
    'Approx.',...
    'location', 'se',...
    'Interpreter', 'Latex');

set(gca, 'LooseInset', get(gca, 'TightInset')) % remove plot padding
set(gca,'fontsize',13);
```



PDF of Z | ERA SCHEME | LOG-NORMAL DISTRIBUTION

```
figure;

f_Z_ERA_new = @(x) 1./(x .* sqrt(2*pi*zeta2_Z_ERA_LN) )...
    .* exp( -(log(x) - nu_Z_ERA_LN).^2./(2*zeta2_Z_ERA_LN) ); % PDF of
 Z

f_Z2_ERA_LN = @(x) 1./(x .* sqrt(2*pi*zeta2_Z2_ERA_LN) )...
    .* exp( -(log(x) - nu_Z2_ERA_LN).^2./(2*zeta2_Z2_ERA_LN) ); % PDF
 of Z^2

number_of_bins = 30;
histogram(Z_ERA, number_of_bins, 'normalization', 'pdf'); hold on;

plot(domain_Z, double(vpa(f_Z_ERA_new(sym(domain_Z)))), 'linewidth',
 1); hold on;

title('CDF of Z | ERA SCHEME | LOG-NORMAL DISTRIBUTION')
xlabel('$x$', 'Interpreter', 'Latex')
ylabel('PDF', 'Interpreter', 'Latex')
legend('True',...
    'Approx.',...
```

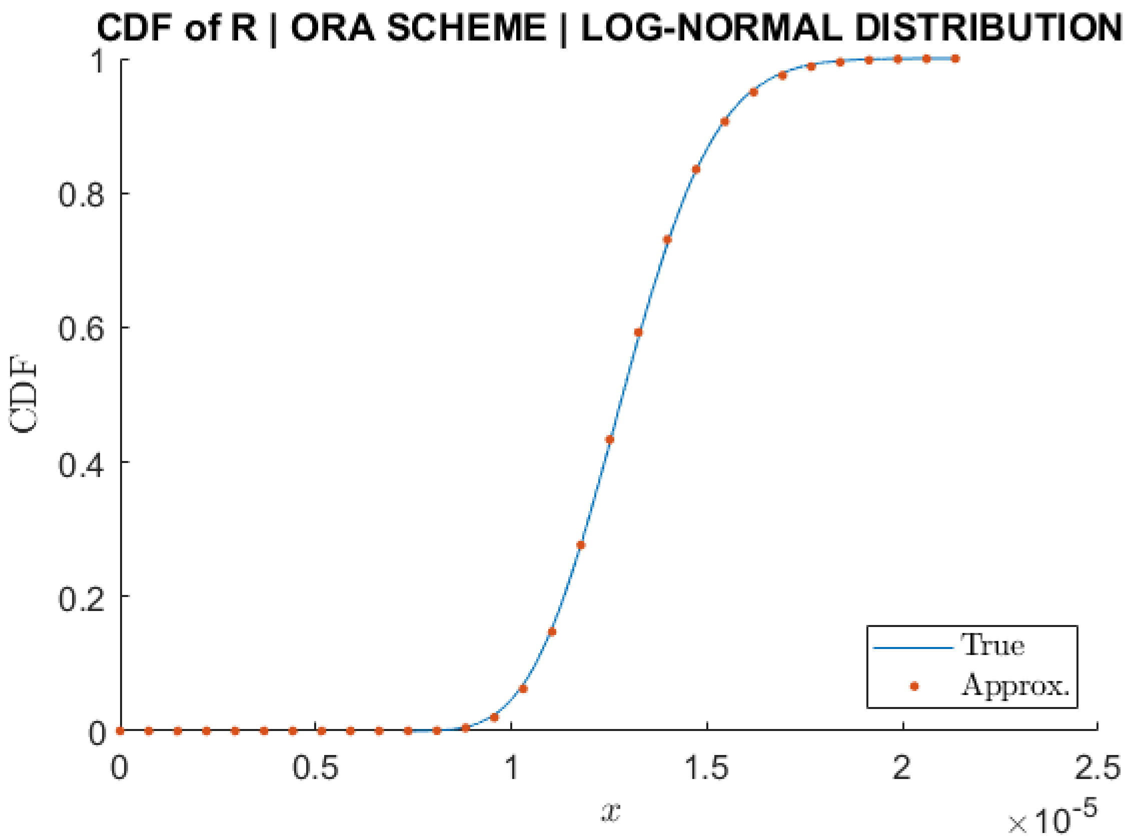
```
    'location', 'ne',...
    'Interpreter', 'Latex');

set(gca, 'LooseInset', get(gca, 'TightInset')) %remove plot padding
set(gca,'fontsize',13);
```



CDF of R | ORA SCHEME | LOG-NORMAL DISTRIBUTION

```
figure;

[y, x] = ecdf(R_ORA); hold on;
domain_R = linspace(0, max(x), 30);
plot(x, y); hold on;
plot(domain_R, F_R_ORA_LN(domain_R), '.', 'markersize', 10); hold on;

title('CDF of R | ORA SCHEME | LOG-NORMAL DISTRIBUTION')
xlabel('$x$', 'Interpreter', 'Latex')
ylabel('CDF', 'Interpreter', 'Latex')

legend('True',...
    'Approx.',...
    'location', 'se',...
    'Interpreter', 'Latex');
```

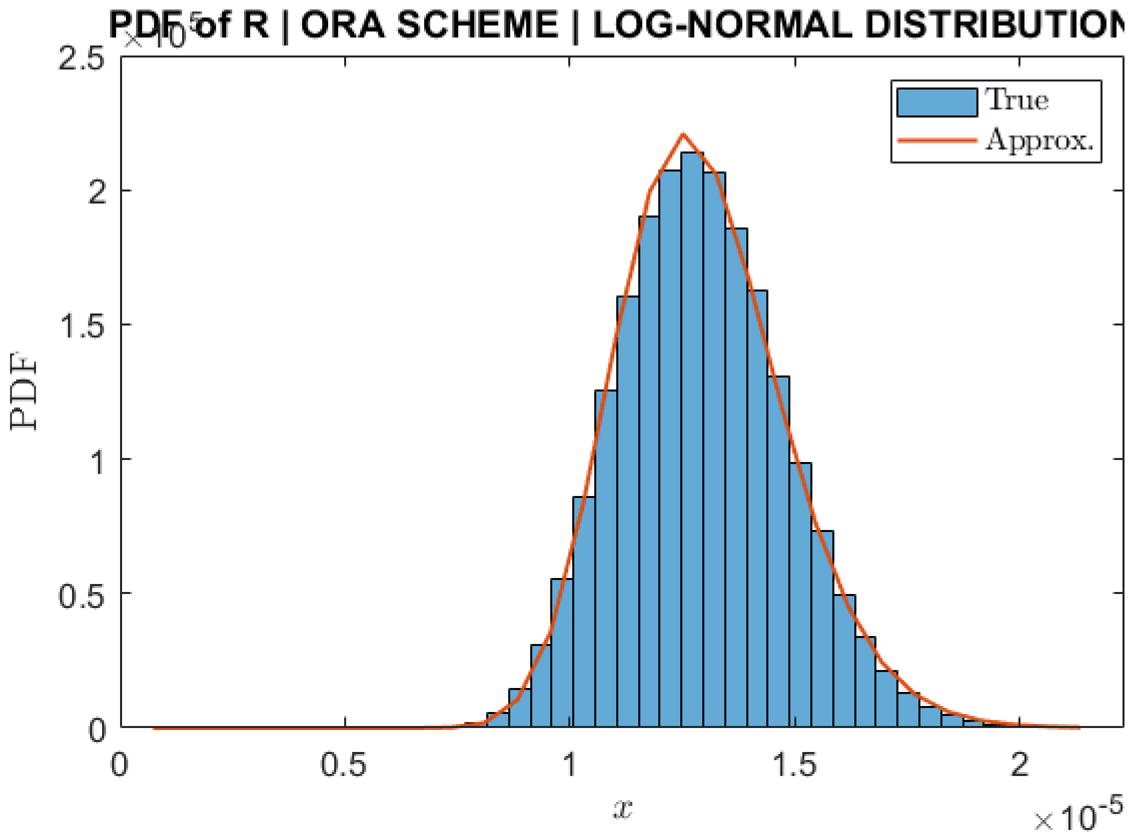
```
set(gca, 'LooseInset', get(gca, 'TightInset')) %remove plot padding
set(gca,'fontsize',13);
```



PDF of R | ORA SCHEME | LOG-NORMAL DISTRIBUTION

```
figure;

f_R_ORA_LN = @(x) 1./(x .* sqrt(2*pi*zeta2_R_ORA_LN) )...
    .* exp( -(log(x) - nu_R_ORA_LN).^2./(2*zeta2_R_ORA_LN) ); % CDF of
 R

%PDF of R^2
f_R2_ORA_LN = @(x) 1./(x .* sqrt(2*pi*zeta2_ORA_LN) )...
    .* exp( -(log(x) - nu_ORA_LN).^2./(2*zeta2_ORA_LN) );

number_of_bins = 30;

histogram(R_ORA, number_of_bins, 'normalization', 'pdf'); hold on;
plot(domain_R, double(vpa(f_R_ORA_LN(sym(domain_R)))), 'linewidth',
 1.5); hold on;

title('PDF of R | ORA SCHEME | LOG-NORMAL DISTRIBUTION')
xlabel('$x$', 'Interpreter', 'Latex')
ylabel('PDF', 'Interpreter', 'Latex')
```

```
legend('True ',...
    'Approx.',...
    'location', 'ne',...
    'Interpreter', 'Latex');

set(gca, 'LooseInset', get(gca, 'TightInset')) %remove plot padding
set(gca,'fontsize',13);
```



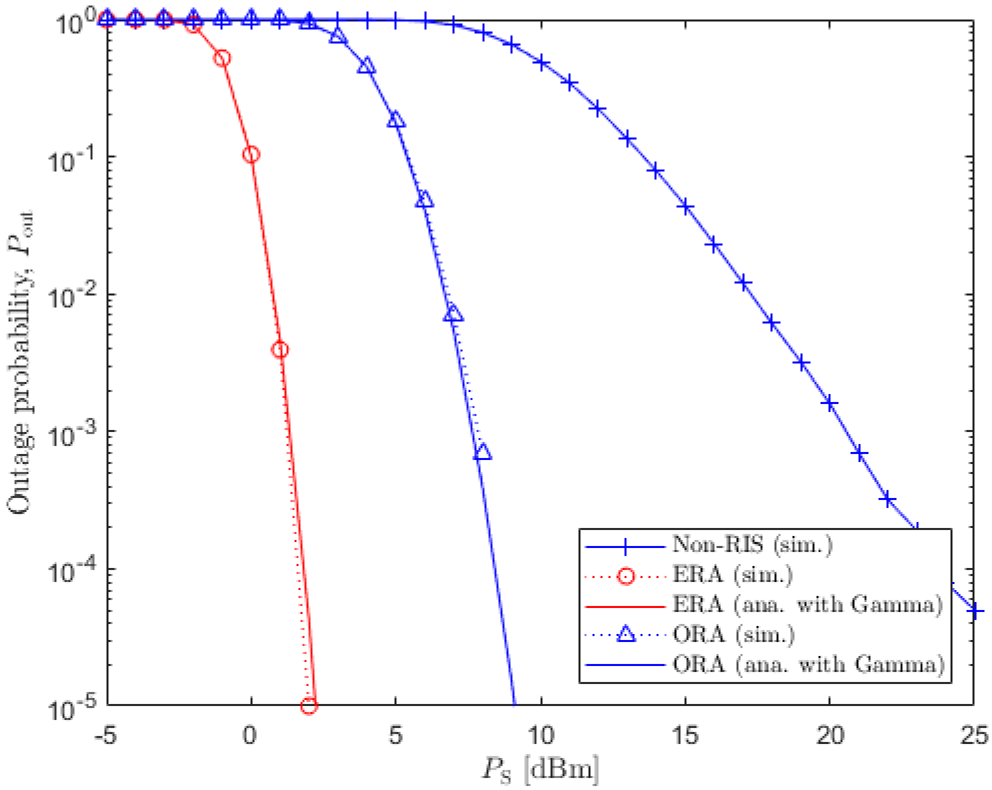
OUTAGE PROBABILITY | LOG-NORMAL DISTRIBUTION

```
OP_non_RIS_sim = zeros(length(SNRdB),1); % should be column vector
OP_ERA_sim = zeros(length(SNRdB),1);
OP_ERA_ana = zeros(length(SNRdB),1);
OP_ORA_sim = zeros(length(SNRdB),1);
OP_ORA_ana = zeros(length(SNRdB),1);

SNR_h0 = abs(h_SD).^2;

for idx = 1:length(SNRdB)
    avgSNR = db2pow(SNRdB(idx)); % i.e., 10^(SNRdB/10)

    OP_non_RIS_sim(idx) = mean(avgSNR*SNR_h0 < SNR_th);

    % ERA scheme
```

```
    OP_ERA_sim(idx) = mean(avgSNR*Z2_ERA < SNR_th);

    OP_ERA_ana(idx) = F_Z_ERA_new(sqrt(SNR_th/avgSNR)); % F_Z
 (sqrt(x))

    %ORA scheme

    OP_ORA_sim(idx) = mean(avgSNR*R2_ORA < SNR_th);

    OP_ORA_ana(idx) = F_R_ORA_LN(sqrt(SNR_th/avgSNR));

    fprintf('Outage probability, SNR = % d \n', round(SNRdB(idx)));
end

figure;

semilogy(P_S_dB, OP_non_RIS_sim, 'b+-'); hold on;
semilogy(P_S_dB, OP_ERA_sim, 'ro:'); hold on;
semilogy(P_S_dB, OP_ERA_ana, 'r-'); hold on;
semilogy(P_S_dB, OP_ORA_sim, 'b^:'); hold on;
semilogy(P_S_dB, OP_ORA_ana, 'b-'); hold on;

xlabel('$P_{\rm S}$ [dBm]', 'Interpreter', 'Latex');
ylabel('Outage probability, $P_{\rm out}$', 'Interpreter', 'Latex');
legend('Non-RIS (sim.)',...
    'ERA (sim.)', ...
    'ERA (ana. with Gamma)',...
    'ORA (sim.)', ...
    'ORA (ana. with Gamma)',...
    'Location','se',...
    'Interpreter', 'Latex');
axis([-Inf Inf 10^(-5) 10^(0)]);
```

```
Outage probability, SNR =  89
Outage probability, SNR =  90
Outage probability, SNR =  91
Outage probability, SNR =  92
Outage probability, SNR =  93
Outage probability, SNR =  94
Outage probability, SNR =  95
Outage probability, SNR =  96
Outage probability, SNR =  97
Outage probability, SNR =  98
Outage probability, SNR =  99
Outage probability, SNR =  100
Outage probability, SNR =  101
Outage probability, SNR =  102
Outage probability, SNR =  103
Outage probability, SNR =  104
Outage probability, SNR =  105
Outage probability, SNR =  106
Outage probability, SNR =  107
Outage probability, SNR =  108
```

```
Outage probability, SNR =  109
Outage probability, SNR =  110
Outage probability, SNR =  111
Outage probability, SNR =  112
Outage probability, SNR =  113
Outage probability, SNR =  114
Outage probability, SNR =  115
Outage probability, SNR =  116
Outage probability, SNR =  117
Outage probability, SNR =  118
Outage probability, SNR =  119
```



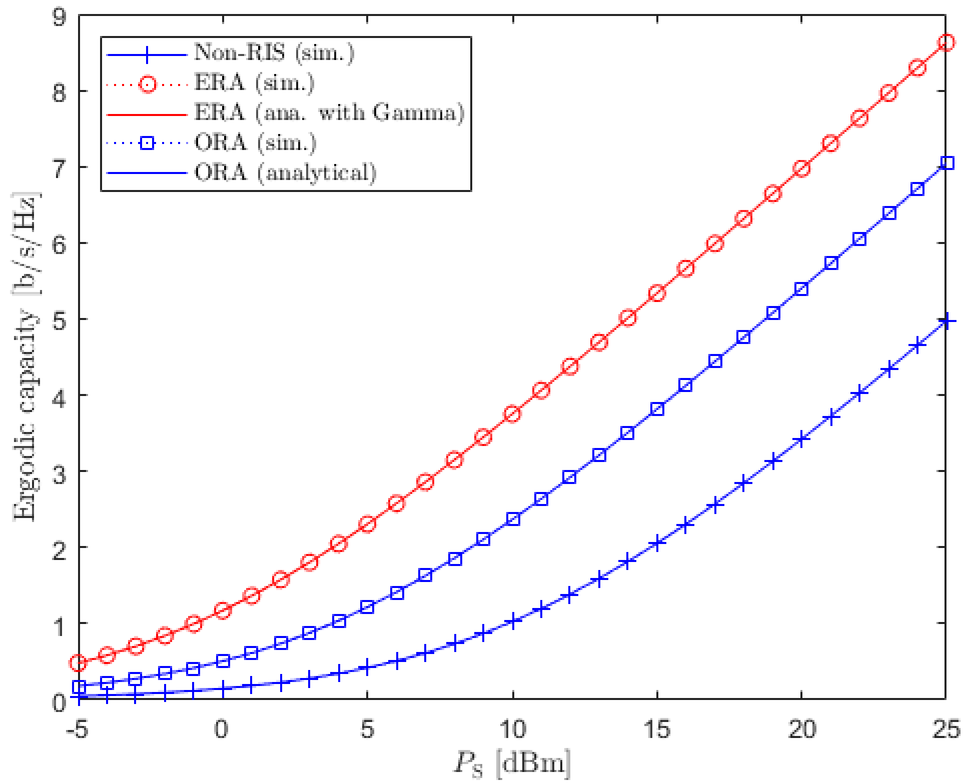
ERGODIC CAPACITY | LOG-NORMAL DISTRIBUTION

```
EC_non_RIS_sim = zeros(length(SNRdB),1); % should be column vector
EC_ERA_sim = zeros(length(SNRdB),1);
EC_ERA_ana = zeros(length(SNRdB),1);
EC_ORA_sim = zeros(length(SNRdB),1);
EC_ORA_ana = zeros(length(SNRdB),1);

a1 =  0.9999964239;
a2 = -0.4998741238;
a3 =  0.3317990258;
a4 = -0.2407338084;
```

```
a5 =  0.1676540711;
a6 = -0.0953293897;
a7 =  0.0360884937;
a8 = -0.0064535442;
a_arr = [a1, a2, a3, a4, a5, a6, a7, a8];

syms aXi bXi Xi(aXi, bXi)
Xi(aXi, bXi) = 0;

for k = 1:8
    Xi(aXi, bXi) = Xi(aXi, bXi) + exp(-bXi^2)/2 * a_arr(k)...
        * exp((k/(2*aXi) + bXi)^2) * erfc(k/(2*aXi) + bXi);
end

syms zeta2 nu
EC_LN(zeta2, nu) = ...
    Xi(1/sqrt(2*zeta2),  nu/sqrt(2*zeta2))...
    + Xi(1/sqrt(2*zeta2), -nu/sqrt(2*zeta2))...
    + sqrt(zeta2/2/pi) * exp(-nu^2/(2*zeta2))...
    + nu/2*erfc(-nu/sqrt(2*zeta2));

for idx = 1:length(SNRdB)
    avgSNR = db2pow(SNRdB(idx)); % 10^(SNRdB(idx)/10)

    EC_non_RIS_sim(idx) = mean(log2(1 + avgSNR*SNR_h0));

    EC_ERA_sim(idx) = mean(log2(1+avgSNR*Z2_ERA));

    EC_ERA_ana(idx) = double(vpa(EC_LN(zeta2_Z2_ERA_LN, log(avgSNR) +
 nu_Z2_ERA_LN)))/log(2);

    EC_ORA_sim(idx) = mean(log2(1+avgSNR*R2_ORA));

    EC_ORA_ana(idx) = double(vpa(EC_LN(zeta2_R2_ORA_LN, log(avgSNR) +
 nu_R2_ORA_LN)))/log(2);

    fprintf('Ergodic capacity, SNR = % d \n', round(SNRdB(idx)));
end

figure;

plot(P_S_dB, EC_non_RIS_sim, 'b+-'); hold on;
plot(P_S_dB, EC_ERA_sim, 'ro:'); hold on;
plot(P_S_dB, EC_ERA_ana, 'r-'); hold on;
plot(P_S_dB, EC_ORA_sim, 'bs:'); hold on;
plot(P_S_dB, EC_ORA_ana, 'b-'); hold on;

xlabel('$P_{\rm S}$ [dBm]', 'Interpreter', 'Latex');
ylabel('Ergodic capacity [b/s/Hz]', 'Interpreter', 'Latex');
legend('Non-RIS (sim.)',...
    'ERA (sim.)',...
    'ERA (ana. with Gamma)',...
    'ORA (sim.)',...
    'ORA (analytical)',...
```

```
    'Interpreter', 'Latex',...
    'Location','NW');

toc
```

```
Ergodic capacity, SNR =  89
Ergodic capacity, SNR =  90
Ergodic capacity, SNR =  91
Ergodic capacity, SNR =  92
Ergodic capacity, SNR =  93
Ergodic capacity, SNR =  94
Ergodic capacity, SNR =  95
Ergodic capacity, SNR =  96
Ergodic capacity, SNR =  97
Ergodic capacity, SNR =  98
Ergodic capacity, SNR =  99
Ergodic capacity, SNR =  100
Ergodic capacity, SNR =  101
Ergodic capacity, SNR =  102
Ergodic capacity, SNR =  103
Ergodic capacity, SNR =  104
Ergodic capacity, SNR =  105
Ergodic capacity, SNR =  106
Ergodic capacity, SNR =  107
Ergodic capacity, SNR =  108
Ergodic capacity, SNR =  109
Ergodic capacity, SNR =  110
Ergodic capacity, SNR =  111
Ergodic capacity, SNR =  112
Ergodic capacity, SNR =  113
Ergodic capacity, SNR =  114
Ergodic capacity, SNR =  115
Ergodic capacity, SNR =  116
Ergodic capacity, SNR =  117
Ergodic capacity, SNR =  118
Ergodic capacity, SNR =  119
Elapsed time is 458.392968 seconds.
```



Published with MATLAB® R2020b