Table of Contents

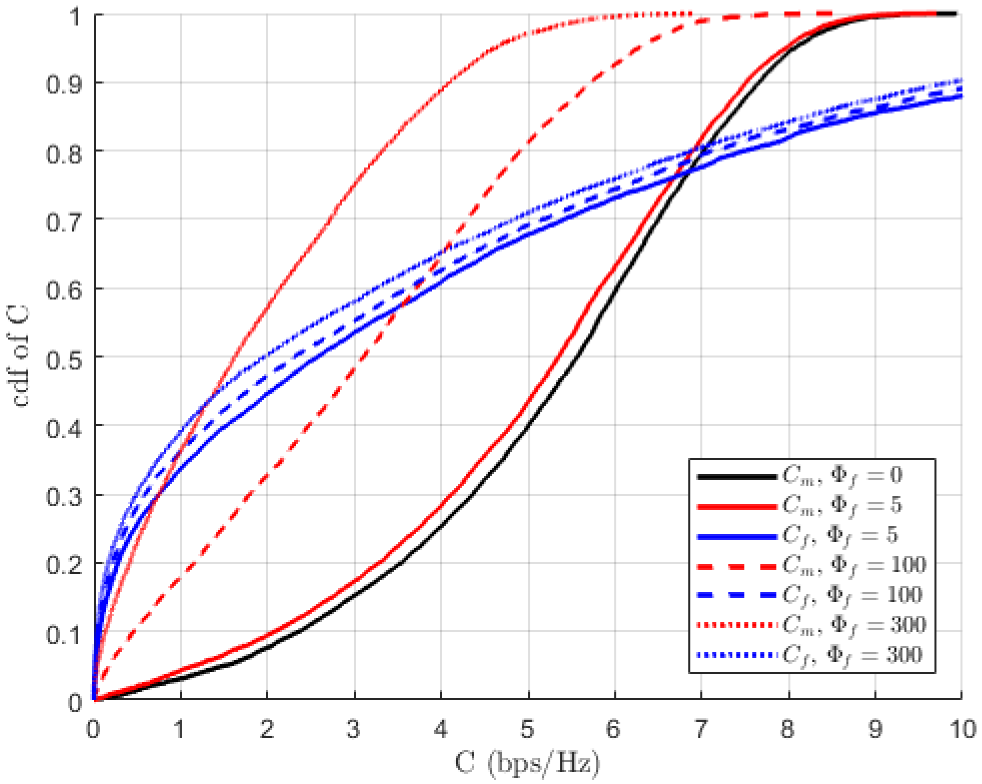
ECEN 310 Project

David Dobbie - 300340161

```
clc
clear all
set(0,'defaultTextInterpreter','latex');

r_f = 20;
R = 1000;
R_o = 2e3;
p_act = 0.25;
p_ind = 0.5;
pathloss_m = 3;
pathloss_f = 3;
sigma_sfdB =8;
SNR_targetdB = 10;
alpha = 0.95;
SNR_maxdB = 20;
WdB = 10;
NodB = 130;



%-----------------------------------------------------

capacity_macro = generateCM(2000, 0, R, R_o, r_f, sigma_sfdB,
 SNR_targetdB, SNR_maxdB,  pathloss_m, NodB, WdB);
capacityM_femto5 =  generateCM(2000, 5, R, R_o, r_f, sigma_sfdB,
 SNR_targetdB, SNR_maxdB,  pathloss_m, NodB, WdB);
capacityM_femto100 =  generateCM(2000, 100, R, R_o, r_f, sigma_sfdB,
 SNR_targetdB, SNR_maxdB,  pathloss_m, NodB, WdB);
capacityM_femto300 =  generateCM(2000, 300, R, R_o, r_f, sigma_sfdB,
 SNR_targetdB, SNR_maxdB,  pathloss_m, NodB, WdB);


capacityF_femto5 =  generateFM(500, 5, R, R_o, r_f, sigma_sfdB,
 SNR_targetdB, ...
         SNR_maxdB,  pathloss_m, NodB, WdB, p_act, p_ind);
```

```
capacityF_femto100 =  generateFM(500, 100, R, R_o, r_f, sigma_sfdB,
 SNR_targetdB, ...
         SNR_maxdB,  pathloss_m, NodB, WdB, p_act, p_ind);

capacityF_femto300 =  generateFM(500, 300, R, R_o, r_f, sigma_sfdB,
 SNR_targetdB, ...
 SNR_maxdB,  pathloss_m, NodB, WdB, p_act, p_ind);

figure(2)
clf
hold on

c = cdfplot(capacity_macro);
c.Color = 'k';
c.LineWidth = 2;

c = cdfplot(capacityM_femto5);
c.Color = 'r';
c.LineWidth = 2;


c = cdfplot(capacityF_femto5);
c.Color = 'b';
c.LineStyle = '-';
c.LineWidth = 2;


c = cdfplot(capacityM_femto100);
c.Color = 'r';
c.LineStyle = '--';
c.LineWidth = 2;

c = cdfplot(capacityF_femto100);
c.Color = 'b';
c.LineStyle = '--';
c.LineWidth = 2;


c = cdfplot(capacityM_femto300);
c.Color = 'r';
c.LineStyle = ':';
c.LineWidth = 2;


c = cdfplot(capacityF_femto300);
c.Color = 'b';
c.LineStyle = ':';
c.LineWidth = 2;

hold off
xlabel('C (bps/Hz)');
ylabel('cdf of C');
xlim([0 10]);
title('')
```

```
lgnd = legend ('$C_m$, $\Phi_f = 0$', '$C_m$, $\Phi_f = 5$', '$C_f$, $
\Phi_f = 5$', ...
'$C_m$, $\Phi_f = 100$', '$C_f$, $\Phi_f = 100$', '$C_m$, $\Phi_f =
 300$', '$C_f$, $\Phi_f = 300$');

set(lgnd,'Interpreter','latex');
set(lgnd,'Location','southeast');
```



```
femto_density_change = [0 100 200 300 400 500];

cap_mean_loss = zeros(6,4);

indx = 1;
for phi = femto_density_change
    phi

    capFemto1st =  generateCM(500, phi, R, R_o, r_f, 8, SNR_targetdB,
 SNR_maxdB, 4, NodB, WdB);
    cap_mean_loss(indx,1) = mean(capFemto1st);

    capFemto2nd =  generateCM(500, phi, R, R_o, r_f, 8, SNR_targetdB,
 SNR_maxdB,  3, NodB, WdB);
```

```
    cap_mean_loss(indx,2) = mean(capFemto2nd);

    capFemto3rd =  generateCM(500, phi, R, R_o, r_f, 10, SNR_targetdB,
 SNR_maxdB,  4, NodB, WdB);
    cap_mean_loss(indx,3) = mean(capFemto3rd);

    capFemto4th =  generateCM(500, phi, R, R_o, r_f, 10, SNR_targetdB,
 SNR_maxdB,  3, NodB, WdB);
    cap_mean_loss(indx,4) = mean(capFemto4th);


    indx = indx +1;
end

baseline_loss = cap_mean_loss(1,:);


cap_mean_loss = 100*(baseline_loss - cap_mean_loss)./baseline_loss;



figure(3)
clf
hold on
c = plot(femto_density_change, cap_mean_loss(:,1));
c.Color = 'k';
c.LineStyle = '-';
c.LineWidth = 2;

c = plot(femto_density_change, cap_mean_loss(:,2));
c.Color = 'k';
c.LineStyle = '--';
c.LineWidth = 2;

c = plot(femto_density_change, cap_mean_loss(:,3));
c.Color = 'b';
c.LineStyle = '-';
c.LineWidth = 2;

c = plot(femto_density_change, cap_mean_loss(:,4));
c.Color = 'b';
c.LineStyle = '--';
c.LineWidth = 2;



xlabel('$\Phi_{F}$');
ylabel('percentage mean $C_m$ loss');
xlim([0 500]);
title('')


lgnd = legend ('$\sigma_{sf} = 8$ dB, $\gamma_m = 4$', ...
    '$\sigma_{sf} = 8$ dB, $\gamma_m = 3$', ...
    '$\sigma_{sf} = 10$ dB, $\gamma_m = 4$', ...
    '$\sigma_{sf} = 10$ dB, $\gamma_m = 3$');

set(lgnd,'Interpreter','latex');
set(lgnd,'Location','northwest');

grid on
hold off
snapnow
```

```
phi =

     0
```

functions

inputs: results: we find the capacity of each macro user due to macro and femto inteference

MACRO USER CAPACITY

```
function [capacity] = generateCM(macro_userDens, femto_Dens, R ,
 R_outRange, r_femto, ...
    sigma_sfdB, SNR_targetdB, SNR_maxdB, pathloss_m ,NodB, wallLossdB)


    lambda_macro = macro_userDens * pi * (R*1e-3)^2; % convert to km
    N = poissrnd(lambda_macro); % number users to simulate

    lambda_femto = femto_Dens * pi * (R*1e-3)^2; % convert to km
    N_femto = 0.25*poissrnd(lambda_femto); % number users to simulate
    SINR_macro_all = [];

    for el = 1:10
    %required power at d0= 1m is capped at 20dB SNR_mac_maxdB
    % generate macro users
    % randomly intialised polar coordinates
    magnitude = sqrt(abs(rand(N,1)*R^2));
    %magnitude = (rand(N,1))*R;
    bearing = 2*pi*(rand(N,1));
    pos = magnitude .* exp(1i*bearing);

    lambda_femto = femto_Dens * pi * (R_outRange*1e-3)^2; % convert to
 km
    N_femto = poissrnd(lambda_femto); % number users to simulate


    % generate femtos range
    magnitude_femto = sqrt(abs(rand(N_femto,1)*R_outRange^2));
    %magnitude_femto = rand(N_femto,1)*R_outRange;
    bearing_femto = 2*pi*(rand(N_femto,1));
    pos_femto = magnitude_femto .* exp(1i*bearing_femto);
```

```
   dm= magnitude; %randomly created d - macro user
   L = db2pow((sigma_sfdB)*randn(N,1)); %lognormal shadowing

   %------------ SETS TRANSMIT POWER ACCORDING TO eq 2
   %sets transmit power such that mean macro SNR  meets threshold (eq
2)
   P_tx = db2pow(+SNR_targetdB    -    sigma_sfdB*qfuncinv(0.95)  +
10*pathloss_m*log10(mean(dm)) + NodB); %for macro signal strength


   %P_tx = reset_transmit(0.95, sigma_sfdB, SNR_targetdB, pathloss_m,
NodB, R, wallLossdB ,false);
   %P_tx = offset + P_tx;

   P_rx = (P_tx .* dm.^(-pathloss_m) .* L);


   for idx = 1:length(P_rx)
       if pow2db(P_rx(idx)) - NodB > SNR_maxdB
           P_rx(idx) = db2pow(SNR_maxdB + NodB) ;
       end
   end
   %{
   figure(66)
   hold on
   cdfplot(pow2db(P_rx) - NodB)
   hold off
   %}
   I = 0; %femto macro interference


   %pow2db(P_tx)
   % add in femto interference to the macro user
   P_tx_femto = db2pow(+SNR_targetdB    -
sigma_sfdB*qfuncinv(0.95) ...
               + 10*pathloss_m*log10((2/3)*r_femto) + NodB);
   %P_tx_femto = reset_transmit(0.95, sigma_sfdB, SNR_targetdB,
pathloss_m, NodB, r_femto, wallLossdB ,true);

   if N_femto > 0
     for idx = 1:N_femto
         dist_from_femto = zeros(N,1);
         for user = 1:N %distance between each macro user and the
current femto
           dist_from_femto(user) = abs( pos_femto(idx) - pos(user));
         end
           % setting femto transmit power to attempt 95% connectivity
within its
           % range for received users
           %P_tx_femto = db2pow(+SNR_targetdB    -
sigma_sfdB*qfuncinv(0.95) ...
           %    + 10*pathloss_m*log10((2/3)*r_femto) + NodB);
%compute mean dist as 2/3 radius
```

```
            L_femto = db2pow((sigma_sfdB)*randn(N,1)); %lognormal
 shadowing
            P_rx_femto = (P_tx_femto .* dist_from_femto.^(-
pathloss_m) .* L_femto) / db2pow(wallLossdB);
            rayleigh = (  abs(    sqrt(1/2) * ( randn(N,1) +
 1j*randn(N,1) )    )  ).^2; %unit variance, zero mean
            % creates interference term with the received power from
 the
            % femto
            interfere_femto = (P_rx_femto .* rayleigh);
            I = I + interfere_femto;
        end
    end


    %rayleigh distribution on receiver
    rayleigh = (  abs(    sqrt(1/2) * ( randn(N,1) + 1j*randn(N,1) )
  )  ).^2; %unit variance, zero mean
    SINR_macro_dB = pow2db(P_rx) + pow2db(rayleigh) -
 pow2db((db2pow(NodB) + I));
    SINR_macro = db2pow(SINR_macro_dB);

    SINR_macro_all = [SINR_macro_all SINR_macro];
    end
    capacity = log2(1 + SINR_macro);
end
```

MACRO USER CAPACITY OPEN ACCESS

```
function [capacity] = generateCMOA(macro_userDens, femto_Dens, R ,
 R_outRange, r_femto, ...
    sigma_sfdB, SNR_targetdB, SNR_maxdB, pathloss_m ,NodB, wallLossdB)
    lambda_macro = macro_userDens * pi * (R*1e-3)^2; % convert to km
    N = poissrnd(lambda_macro); % number users to simulate

    lambda_femto = femto_Dens * pi * (R*1e-3)^2; % convert to km
    N_femto = poissrnd(lambda_femto); % number users to simulate

    %required power at d0= 1m is capped at 20dB SNR_mac_maxdB
    % generate macro users
    % randomly intialised polar coordinates
    magnitude = sqrt(abs(rand(N,1)*R^2));
    bearing = 2*pi*(rand(N,1));
    pos = magnitude .* exp(1i*bearing);

    lambda_femto = femto_Dens * pi * (R_outRange*1e-3)^2; % convert to
 km
    N_femto = poissrnd(lambda_femto); % number users to simulate


    % generate femtos range
    magnitude_femto = sqrt(abs(rand(N_femto,1)*R_outRange^2));
    bearing_femto = 2*pi*(rand(N_femto,1));
    pos_femto = magnitude_femto .* exp(1i*bearing_femto);

    dm= magnitude; %randomly created d - macro user
    L = db2pow((sigma_sfdB)*randn(N,1)); %lognormal shadowing

    %------------ SETS TRANSMIT POWER ACCORDING TO eq 2
    %sets transmit power such that mean macro SNR  meets threshold (eq
 2)
    P_tx = db2pow(+SNR_targetdB    -   sigma_sfdB*qfuncinv(0.95)  +
 10*pathloss_m*log10(mean(dm)) + NodB) %for macro signal strength
    P_rx = (P_tx .* dm.^(-pathloss_m) .* L);
    for idx = 1:length(P_rx)
        if pow2db(P_rx(idx)) - NodB > SNR_maxdB
            P_rx(idx) = db2pow(SNR_maxdB + NodB) ;
        end
    end
    I = 0; %femto macro interference
    %pow2db(P_tx)
    % add in femto interference to the macro user
    if N_femto > 0
      for idx = 1:N_femto
          dist_from_femto = zeros(N,1);
          for user = 1:N %distance between each macro user and the
 current femto
            dist_from_femto(user) = abs( pos_femto(idx) - pos(user));
          end
            % setting femto transmit power to attempt 95% connectivity
 within its
            % range for received users
            P_tx_femto = db2pow(+SNR_targetdB    -
 sigma_sfdB*qfuncinv(0.95) ...
                + 10*pathloss_m*log10((2/3)*r_femto) + NodB); %compute
 mean dist as 2/3 radius
            L_femto = db2pow((sigma_sfdB)*randn(N,1)); %lognormal
 shadowing
            P_rx_femto = (P_tx_femto .* dist_from_femto.^(-
pathloss_m) .* L_femto) / db2pow(wallLossdB);
             %pow2db(P_tx_femto)
            % maxes out received power from the femto for each user to
 20dB
            for idx = 1:length(P_rx_femto)
                if pow2db(P_rx_femto(idx)) - NodB > SNR_maxdB
                    P_rx_femto(idx) = db2pow(SNR_maxdB + NodB) ;
                end
            end
            rayleigh = (  abs(    sqrt(1/2) * ( randn(N,1) +
 1j*randn(N,1) )    )  ).^2; %unit variance, zero mean
            % creates interference term with the received power from
 the
            % femto
            interfere_femto = (P_rx_femto .* rayleigh);
            I = I + interfere_femto;
        end
    end
    %rayleigh distribution on receiver
    rayleigh = (  abs(    sqrt(1/2) * ( randn(N,1) + 1j*randn(N,1) )
  )  ).^2; %unit variance, zero mean
    SINR_macro_dB = pow2db(P_rx) + pow2db(rayleigh) -
 pow2db((db2pow(NodB) + I));
    SINR_macro = db2pow(SINR_macro_dB);

    capacity = log2(1 + SINR_macro);
end
```

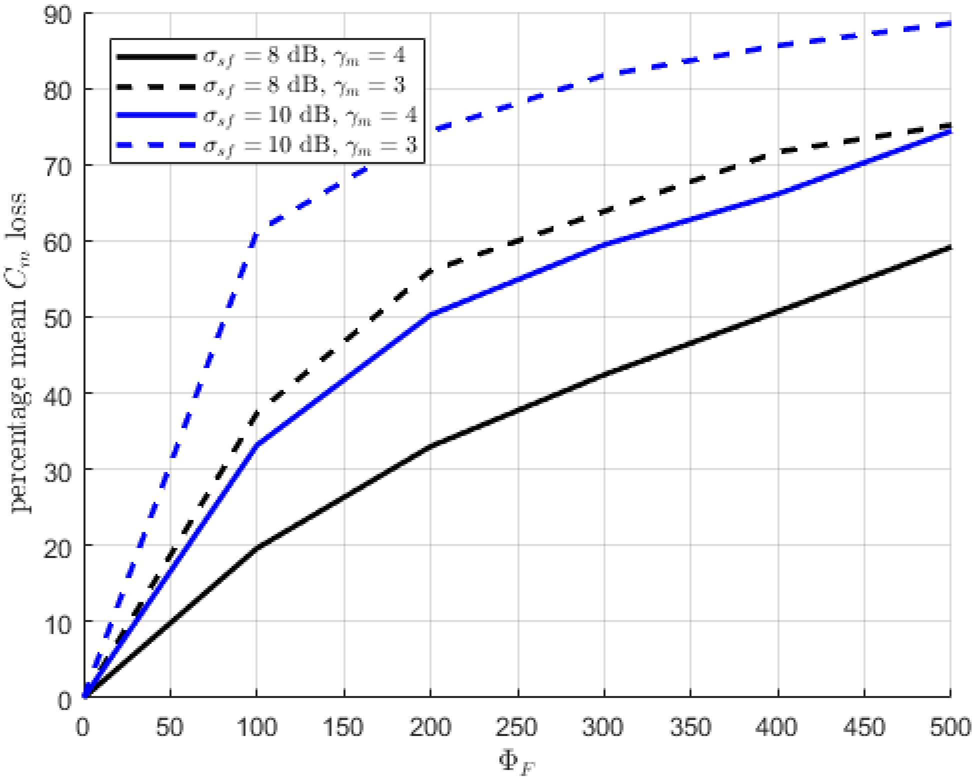
FEMTO USER CAPACITY

```
function [capacity] = generateFM(macro_userDens, femto_Dens, R ,
 R_outRange, r_femto, ...
    sigma_sfdB, SNR_targetdB, SNR_maxdB, pathloss_m ,NodB, wallLossdB,
 p_activity , p_indoor)

    %generate SINR for each femto cell's femto users
    SINR_femto_users = [];

    for overallIter = 1:4
    lambda_femto = femto_Dens * pi * (R_outRange*1e-3)^2; % convert to
 km
    N_femto = poissrnd(lambda_femto); % number users to simulate

    %N_femto = cast(N_femto * p_activity, 'uint8');
    Nusr = cast(1000/(N_femto*p_activity), 'uint32');

    % generate femtos
    magnitude_femto = rand(N_femto,1)*R_outRange;
    bearing_femto = 2*pi*(rand(N_femto,1));
    pos_femto = magnitude_femto .* exp(1i*bearing_femto);

    % generate femto users - this number of users under each femto
 cell
    magnitude_femto_user = abs(rand(Nusr,1))*r_femto;
    bearing_femto_user = 2*pi*(rand(Nusr,1));
    pos_femto_user = magnitude_femto_user .*
 exp(1i*bearing_femto_user);
    femto_user_inside = randsrc(Nusr,1,([1 0]));

    % ------------- generate each femto cell's transmit power
    P_tx_femto_cell = db2pow(+SNR_targetdB    -
 sigma_sfdB*qfuncinv(0.95) ...
        + 10*pathloss_m*log10(0.67*r_femto) + NodB); %compute mean
 dist as 2/3 radius
    % ------------- generate the macro transmit power
```

```
   P_tx_macro = reset_transmit(0.95, sigma_sfdB, SNR_targetdB,
pathloss_m, NodB, R, wallLossdB ,false);
   %P_tx_macro = db2pow(+SNR_targetdB - sigma_sfdB*qfuncinv(0.95)  +
10*pathloss_m*log10(0.67*R) + NodB); %for macro signal strength

   for cellIdx = 1:N_femto
       % generate femto users - this number of users under each femto
cell
       magnitude_femto_user = rand(Nusr,1)*r_femto;
       bearing_femto_user = 2*pi*(rand(Nusr,1));
       pos_femto_user = magnitude_femto_user .*
exp(1i*bearing_femto_user);
       femto_user_inside = randsrc(Nusr,1,([1 0]));

       % init users for single femto cell
       users = pos_femto(cellIdx) + pos_femto_user;

       for usrIdx = 1:Nusr

           if (abs(users(usrIdx)) < R) %user wtihin femto cell placed
within R
               %-------- COMPUTE FEMTO-to-FEMTO interference
               % removes the user's own femto cell from compared
femtos
               other_femto_pos = pos_femto;
               other_femto_pos(cellIdx) = [];

               other_femto_dists = abs(users(usrIdx) -
other_femto_pos);

               N_active = cast(N_femto * p_activity, 'uint32');
               %N_active = N_femto-1;
               other_femto_dists =
datasample(other_femto_dists,N_active); %only p_active femtos
considered to cause interference


               L = db2pow((sigma_sfdB)*randn(N_active,1));
               h = (  abs(    sqrt(1/2) * ( randn(N_active,1) +
1j*randn(N_active,1) )    )  ).^2; %unit variance, zero mean
               %h =1;
               I_femto_individ = P_tx_femto_cell .*
other_femto_dists .^(-3) ...
                     .* (L) ./ ((1+
femto_user_inside(usrIdx))*(db2pow(wallLossdB)));
               I_femto_individ = I_femto_individ.* h;
               I_femto_inteference = sum(I_femto_individ);

               %-------- COMPUTE MACRO interference

               user_dist_from_macro = abs(users(usrIdx) - [0]); %
since macro is 0,0
```

```
                L = db2pow((sigma_sfdB)*randn(1,1));
                h = (  abs(    sqrt(1/2) * ( randn(1,1) +
 1j*randn(1,1) )    )  ).^2;
                I_macro_inteference_longterm = P_tx_macro .*
 user_dist_from_macro .^(-pathloss_m) ...
                     .* (L) ./ (db2pow(wallLossdB *
 femto_user_inside(usrIdx)));


                % caps transmit power of the macro tx
                if pow2db(I_macro_inteference_longterm) - NodB >
 SNR_maxdB
                    I_macro_inteference_longterm = db2pow(SNR_maxdB +
 NodB);
                end

                I_macro_interference = I_macro_inteference_longterm *
 h;
                %-------- COMPUTE SOURCE FEMTO RX POWER

                L = db2pow((sigma_sfdB)*randn(1,1));
                h = (  abs(    sqrt(1/2) * ( randn(1,1) +
 1j*randn(1,1) )    )  ).^2;
                dist_user = abs(pos_femto_user(usrIdx));
                P_rx_femto_user_longterm = P_tx_femto_cell *
 dist_user^(-3) * L ...
                    / (db2pow(wallLossdB * (1 -
 femto_user_inside(usrIdx)))) ;

                P_rx_femto_user_instant = P_rx_femto_user_longterm *
 h;

                %-------- add SINR to total results
                SINR_single_femto_user = P_rx_femto_user_instant /
 ( db2pow(NodB) +  I_femto_inteference ...
                    + I_macro_interference );

                SINR_femto_users = [SINR_femto_users
 SINR_single_femto_user];
            end

        end





    end
    end

    capacity = log2(1 + SINR_femto_users);
end

function newP_tx = reset_transmit(reliability, sigma_shadow,
 SNR_targetdB, pathloss, NodB, range, wallLossdB ,isFemtoUser)
```

```
    P_tx = db2pow(+SNR_targetdB -  sigma_shadow*qfuncinv(reliability)
 + 10*pathloss*log10((2/3)*range) + NodB); %for signal strength


    Nusr = 5e4;
    magnitude_user = abs(rand(Nusr,1))*range;
    bearing_user = 2*pi*(rand(Nusr,1));
    pos_user = magnitude_user .* exp(1i*bearing_user);
    user_inside = randsrc(Nusr,1,([1 0]));

    dist = abs(pos_user);
    L = db2pow((sigma_shadow)*randn(Nusr,1));


    P_rx = (P_tx .* dist.^(-pathloss) .* L);



    if(isFemtoUser)
        P_rx = (P_tx .* dist.^(-pathloss) .* L)./(db2pow(wallLossdB .*
 user_inside));
    end
    if(~isFemtoUser) %macro user
        P_rx = (P_tx .* dist.^(-pathloss) .* L);
    end
    h = cdfplot(pow2db(P_rx) - NodB);
    SNRdB = h.XData;
    prob = h.YData;

    [ d, idxTargetProb ] = min( abs( (1-reliability)-prob ) );

    adjust = SNR_targetdB - SNRdB(idxTargetProb);

    newP_tx = db2pow(pow2db(P_tx) + adjust);
end
```



Published with MATLAB® R2017b