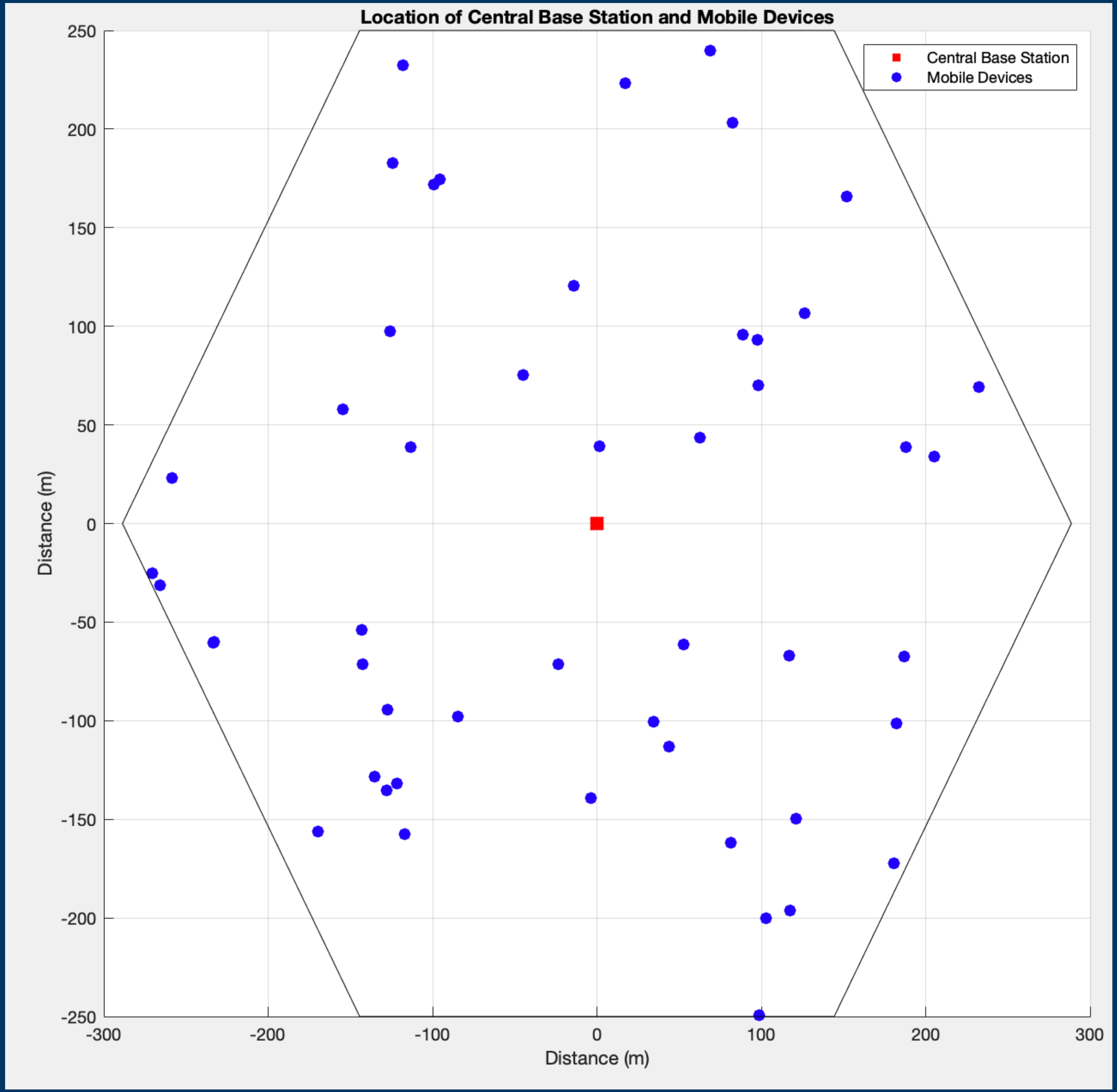
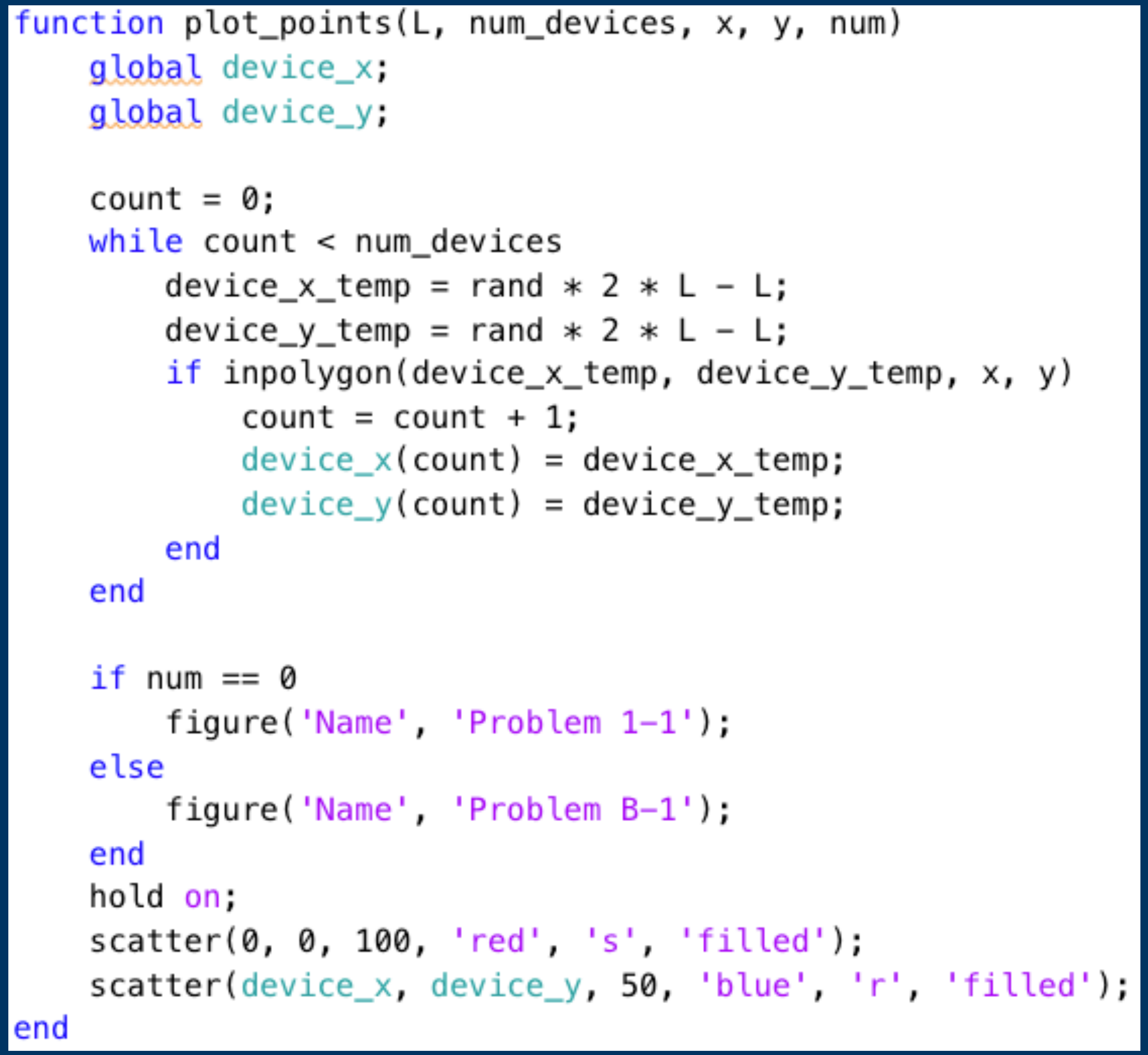
Introduction to Wireless and Mobile Network HW4

B09602017 電機三 白宗民

2023/5/18

1-1. Arrangement of the devices

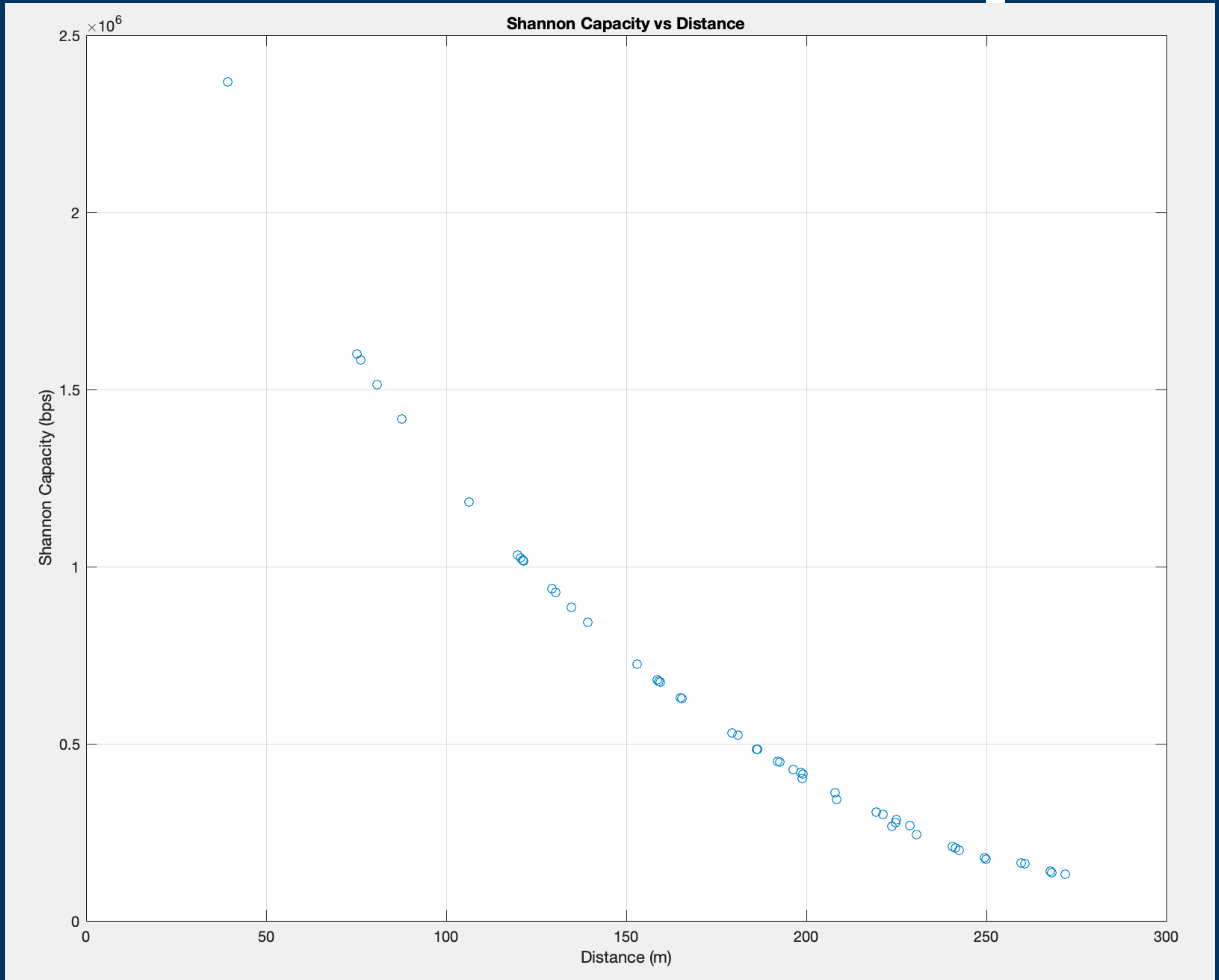


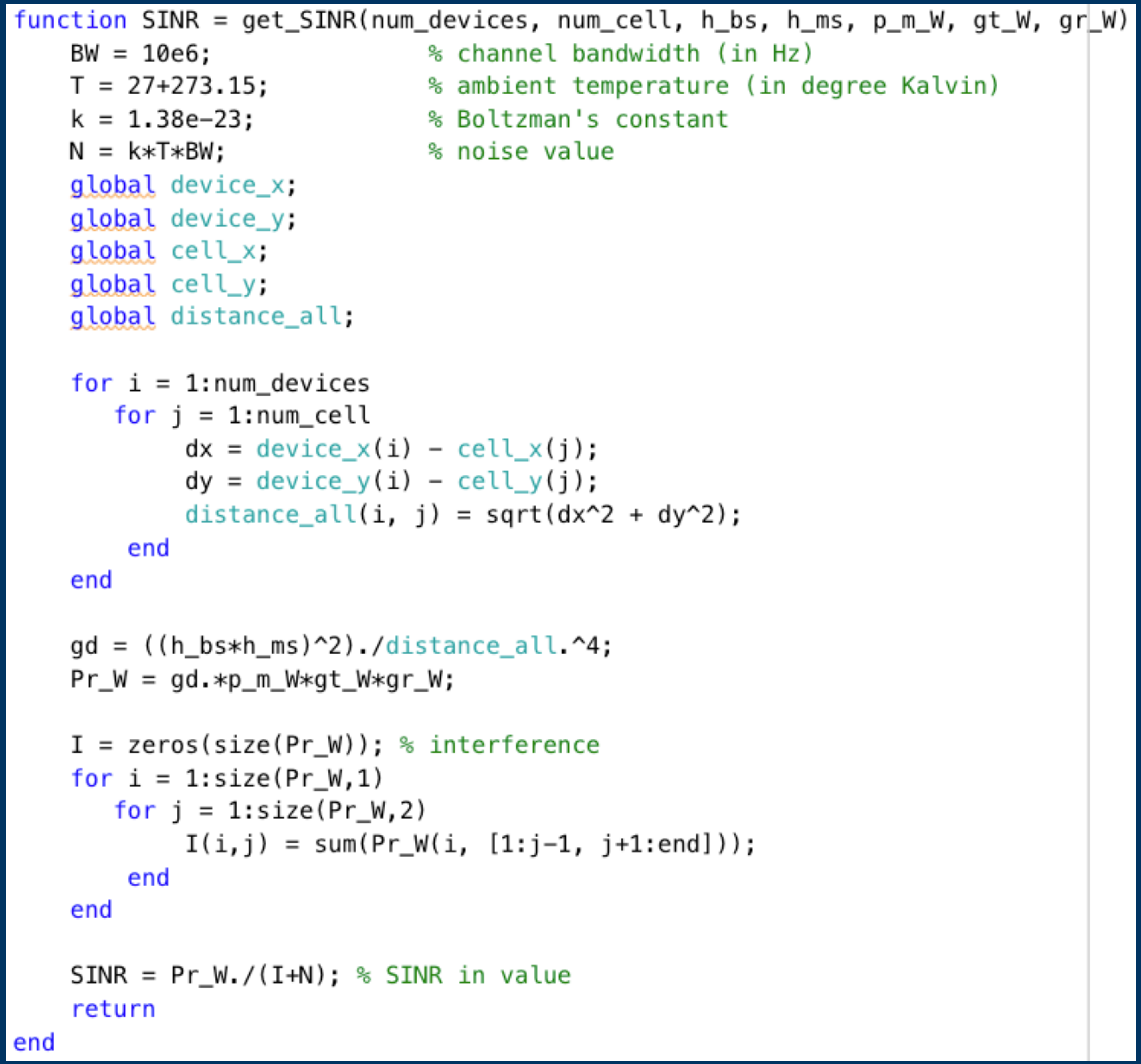


```
function plot_points(L, num_devices, x, y, num)
    global device_x;
    global device_y;

    count = 0;
    while count < num_devices
        device_x_temp = rand * 2 * L - L;
        device_y_temp = rand * 2 * L - L;
        if inpolygon(device_x_temp, device_y_temp, x, y)
            count = count + 1;
            device_x(count) = device_x_temp;
            device_y(count) = device_y_temp;
        end
    end

    if num == 0
        figure('Name', 'Problem 1-1');
    else
        figure('Name', 'Problem B-1');
    end
    hold on;
    scatter(0, 0, 100, 'red', 's', 'filled');
    scatter(device_x, device_y, 50, 'blue', 'r', 'filled');
end
```

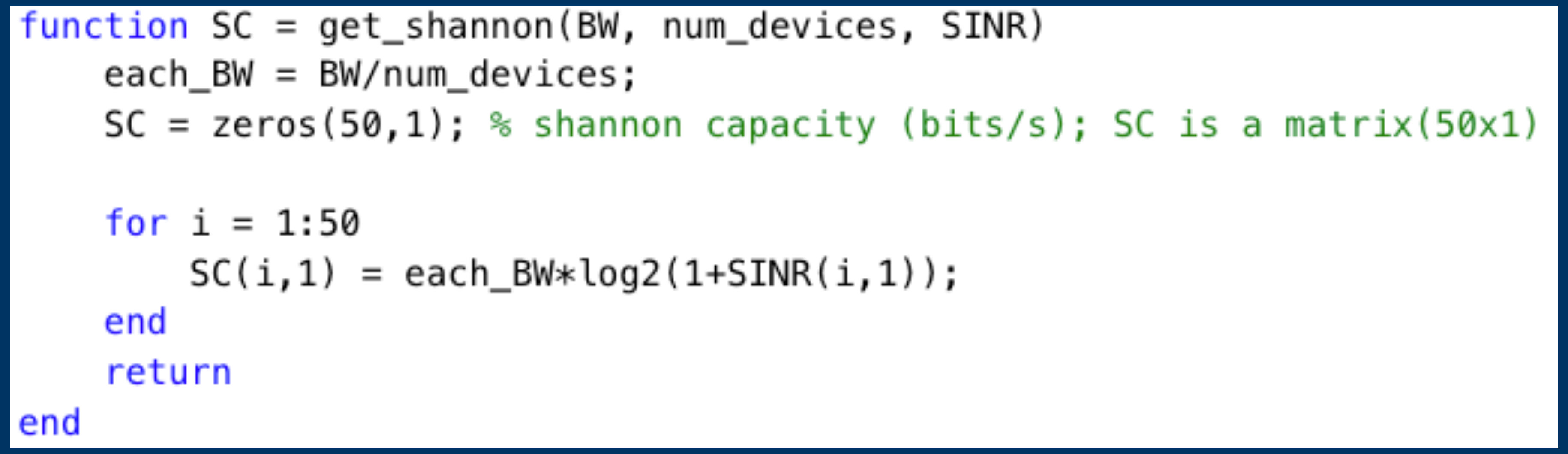
This is how I plot the 50 points in the cell

1-2. Shannon capacity vs distance





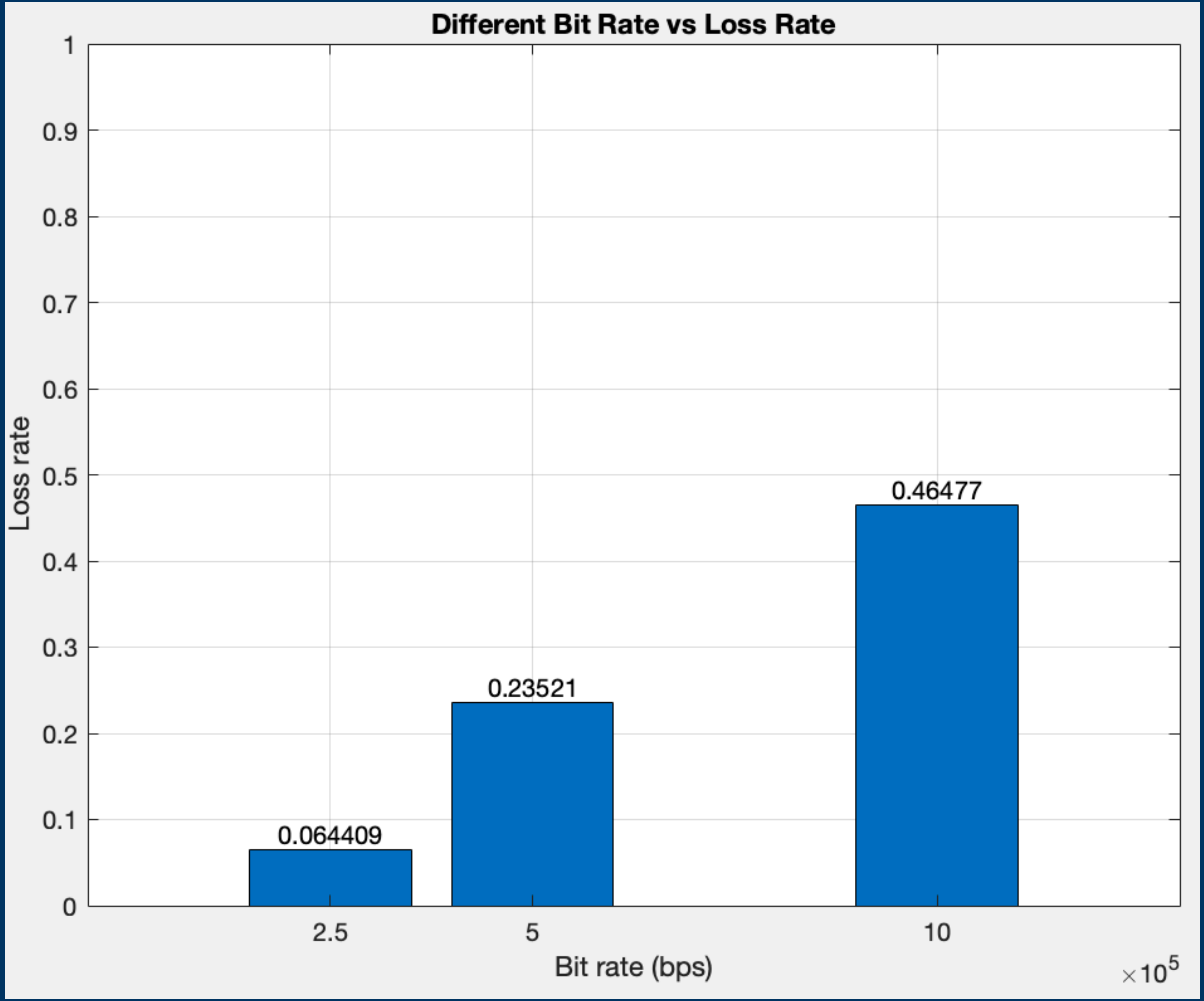
```
function SINR = get_SINR(num_devices, num_cell, h_bs, h_ms, p_m_W, gt_W, gr_W)
    BW = 10e6;               % channel bandwidth (in Hz)
    T = 27+273.15;           % ambient temperature (in degree Kalvin)
    k = 1.38e-23;            % Boltzman's constant
    N = k*T*BW;              % noise value
    global device_x;
    global device_y;
    global cell_x;
    global cell_y;
    global distance_all;

    for i = 1:num_devices
       for j = 1:num_cell
            dx = device_x(i) - cell_x(j);
            dy = device_y(i) - cell_y(j);
            distance_all(i, j) = sqrt(dx^2 + dy^2);
        end
    end

    gd = ((h_bs*h_ms)^2)./distance_all.^4;
    Pr_W = gd.*p_m_W*gt_W*gr_W;

    I = zeros(size(Pr_W)); % interference
    for i = 1:size(Pr_W,1)
       for j = 1:size(Pr_W,2)
            I(i,j) = sum(Pr_W(i, [1:j-1, j+1:end]));
        end
    end

    SINR = Pr_W./(I+N); % SINR in value
    return
end
```



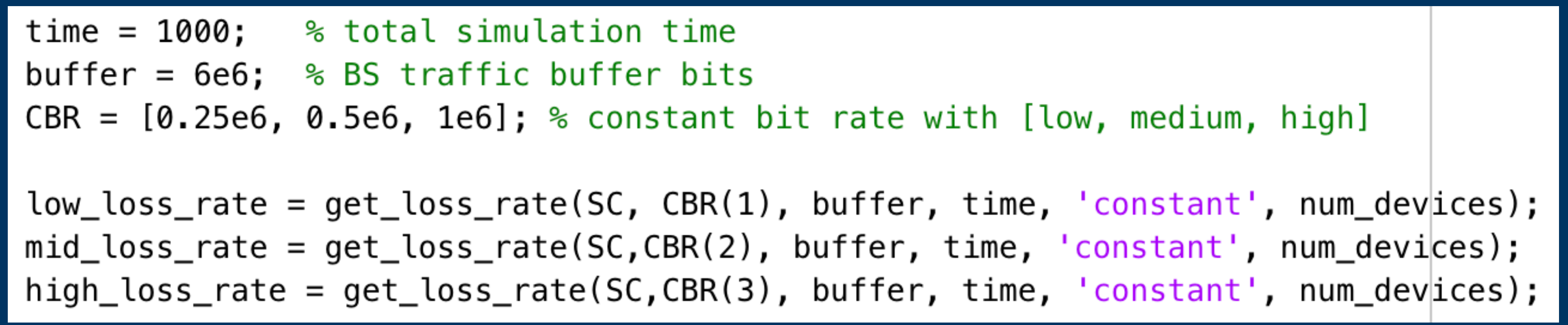
```
function SC = get_shannon(BW, num_devices, SINR)
    each_BW = BW/num_devices;
    SC = zeros(50,1); % shannon capacity (bits/s); SC is a matrix(50x1)

    for i = 1:50
        SC(i,1) = each_BW*log2(1+SINR(i,1));
    end
    return
end
```

I use 2 functions to get the Shannon Capacity.
First to get the SINR, then use the second one to get Shannon capacity.
Then plot the Shannon capacity.

1-3. Loss rate vs traffic load

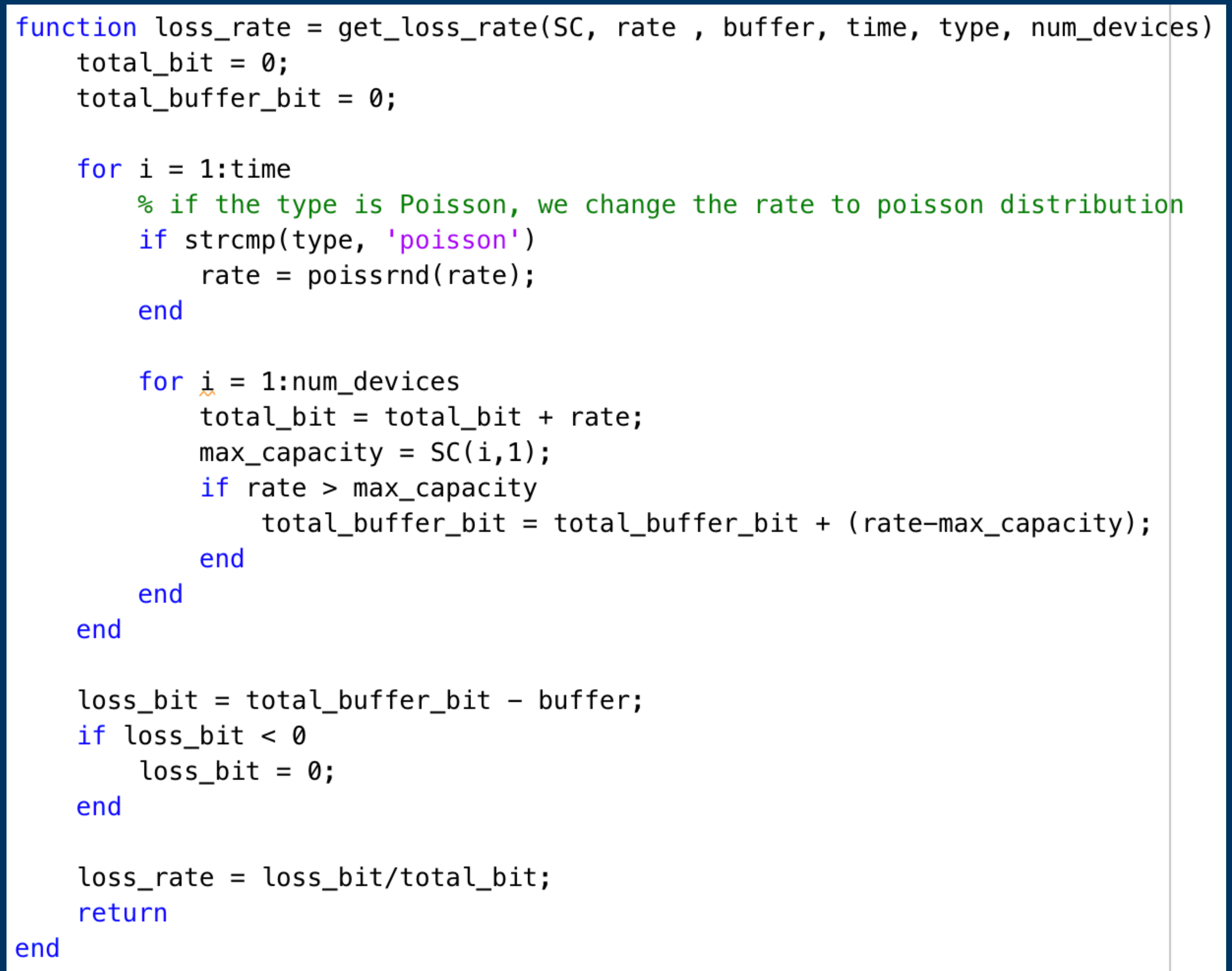


Parameters I used in this question



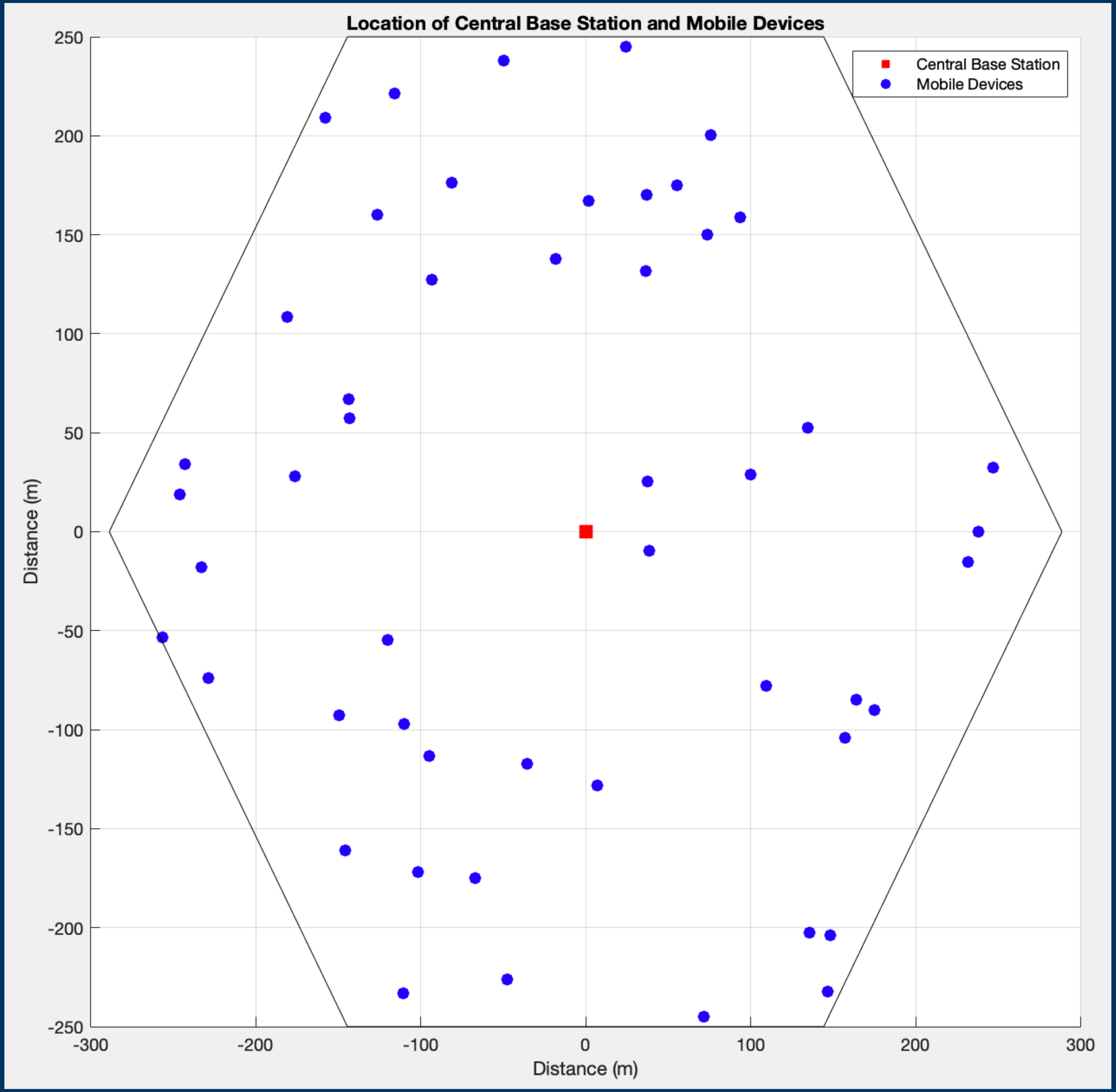
```
time = 1000;   % total simulation time
buffer = 6e6;  % BS traffic buffer bits
CBR = [0.25e6, 0.5e6, 1e6]; % constant bit rate with [low, medium, high]

low_loss_rate = get_loss_rate(SC, CBR(1), buffer, time, 'constant', num_devices);
mid_loss_rate = get_loss_rate(SC,CBR(2), buffer, time, 'constant', num_devices);
high_loss_rate = get_loss_rate(SC,CBR(3), buffer, time, 'constant', num_devices);
```

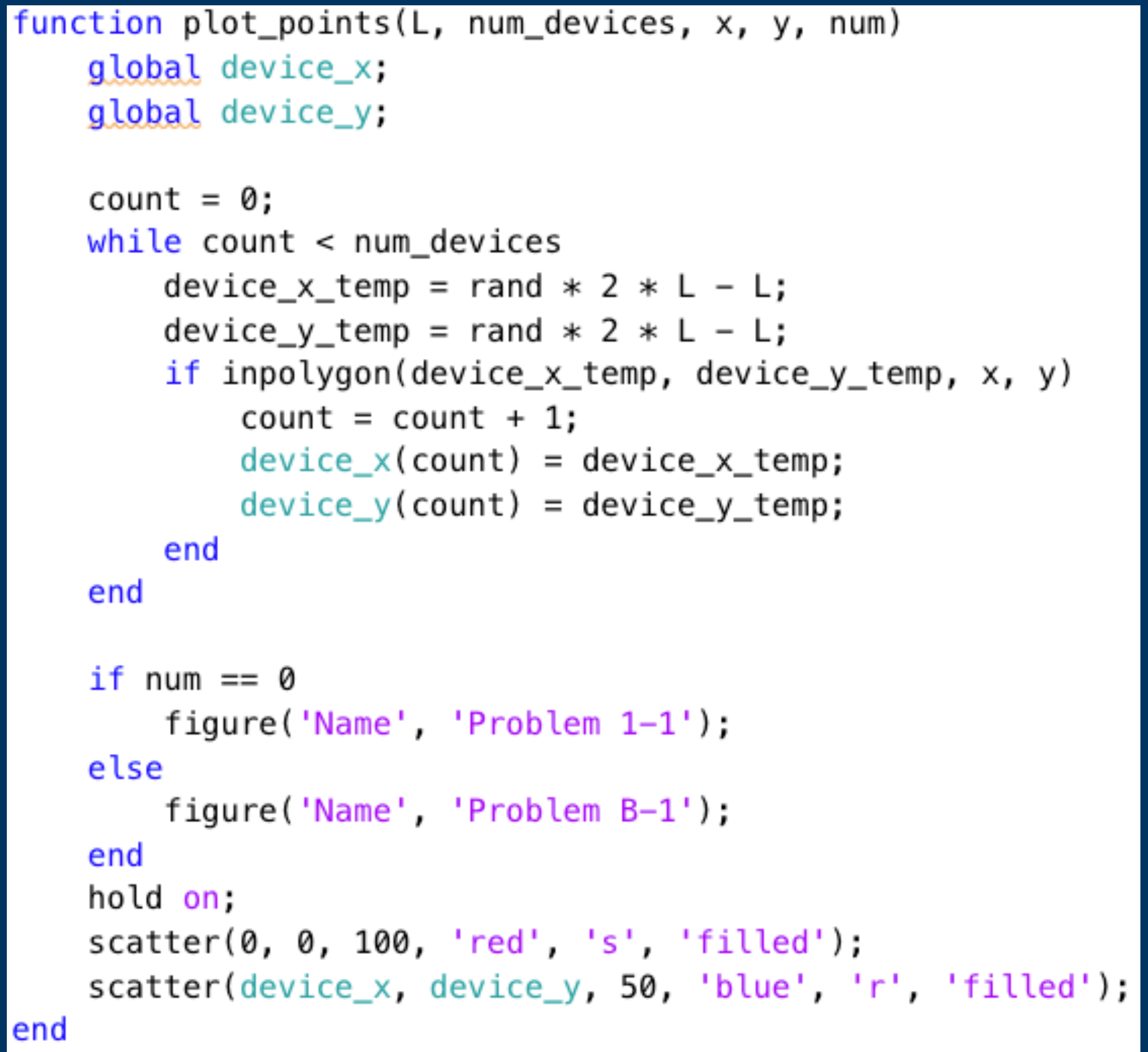
Function that calculates the loss rate



```
function loss_rate = get_loss_rate(SC, rate , buffer, time, type, num_devices)
    total_bit = 0;
    total_buffer_bit = 0;

    for i = 1:time
        % if the type is Poisson, we change the rate to poisson distribution
        if strcmp(type, 'poisson')
            rate = poissrnd(rate);
        end

        for i = 1:num_devices
            total_bit = total_bit + rate;
            max_capacity = SC(i,1);
            if rate > max_capacity
                total_buffer_bit = total_buffer_bit + (rate-max_capacity);
            end
        end
    end

    loss_bit = total_buffer_bit - buffer;
    if loss_bit < 0
        loss_bit = 0;
    end

    loss_rate = loss_bit/total_bit;
    return
end
```

B-1. Arrangement of the devices

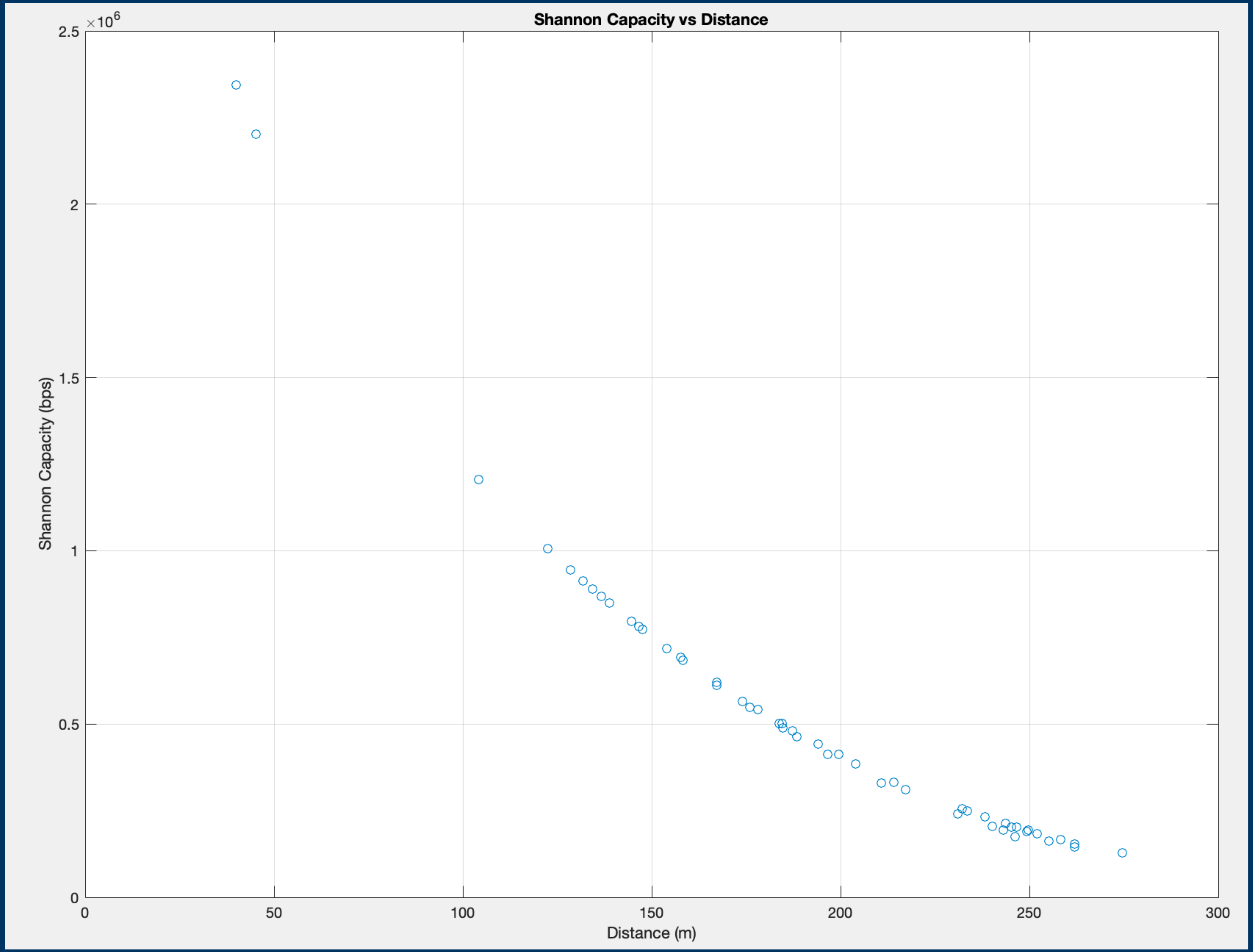


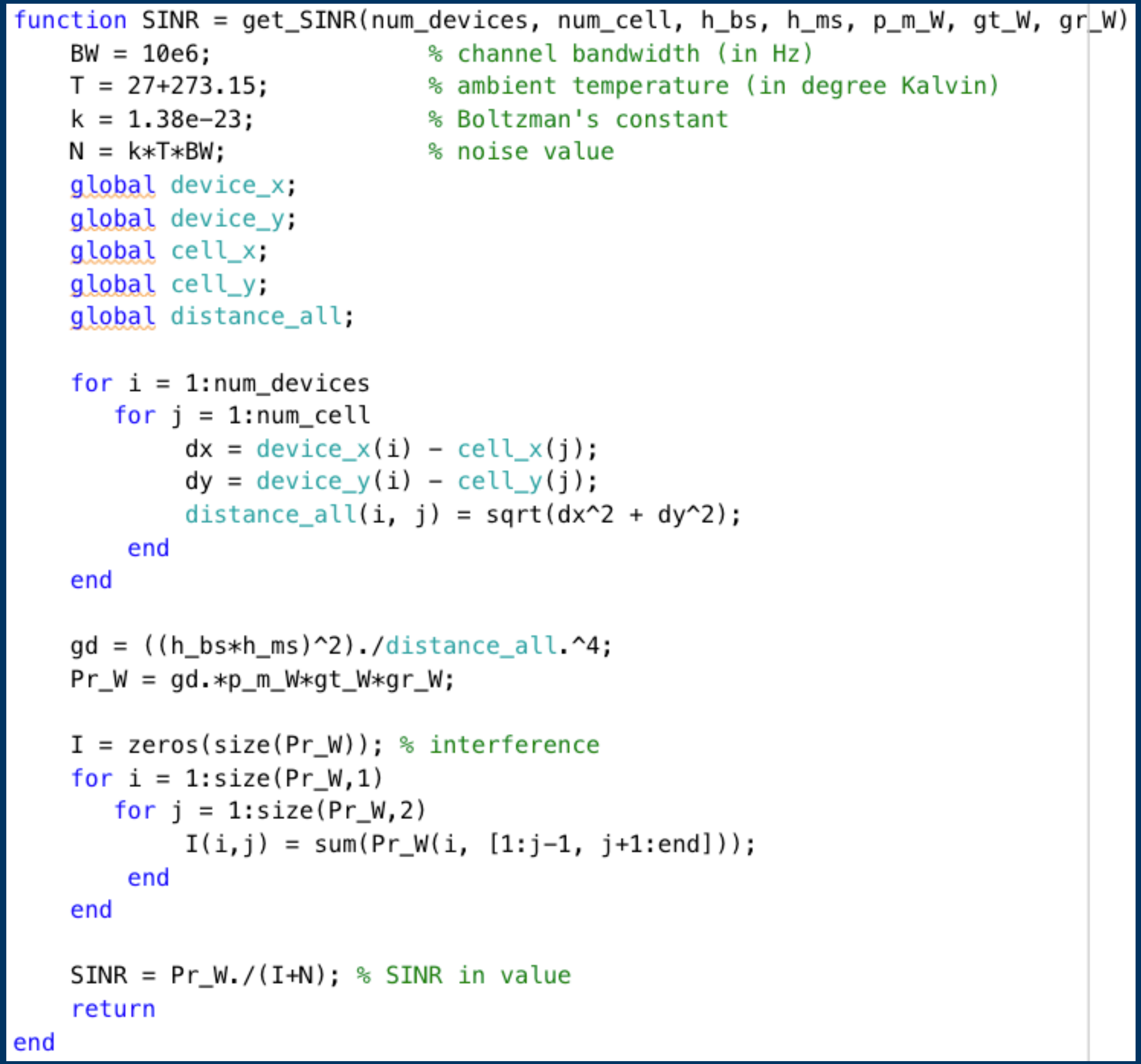


```
function plot_points(L, num_devices, x, y, num)
    global device_x;
    global device_y;

    count = 0;
    while count < num_devices
        device_x_temp = rand ∗ 2 ∗ L − L;
        device_y_temp = rand ∗ 2 ∗ L − L;
        if inpolygon(device_x_temp, device_y_temp, x, y)
            count = count + 1;
            device_x(count) = device_x_temp;
            device_y(count) = device_y_temp;
        end
    end

    if num == 0
        figure('Name', 'Problem 1−1');
    else
        figure('Name', 'Problem B−1');
    end
    hold on;
    scatter(0, 0, 100, 'red', 's', 'filled');
    scatter(device_x, device_y, 50, 'blue', 'r', 'filled');
end
```

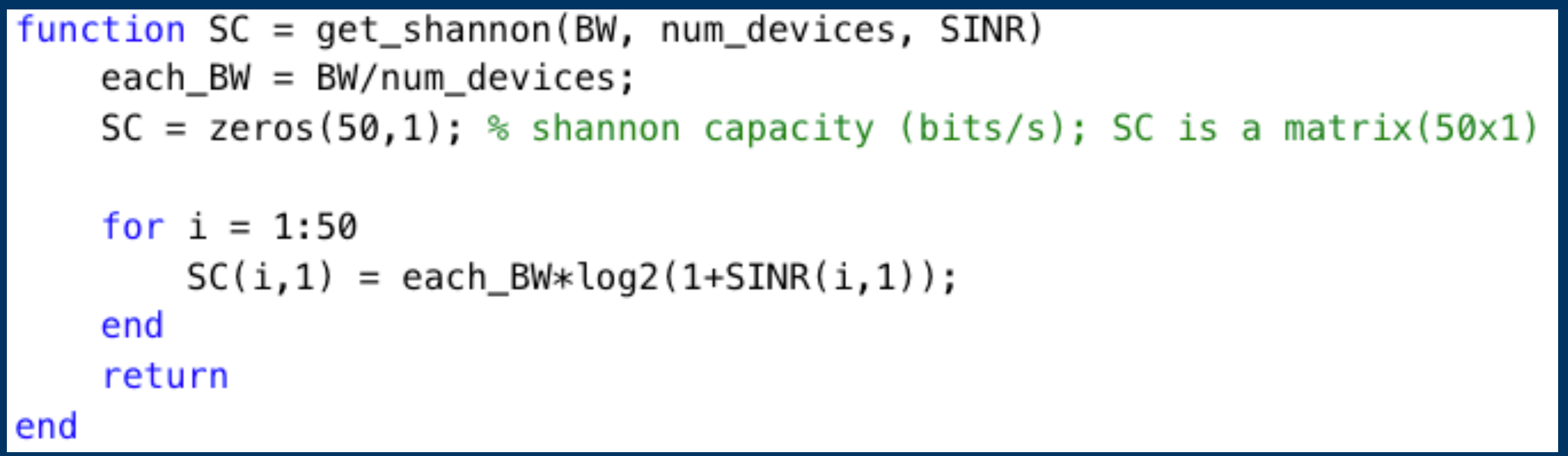
This is how I plot the 50 points in the cell

B-2. Shannon capacity vs distance





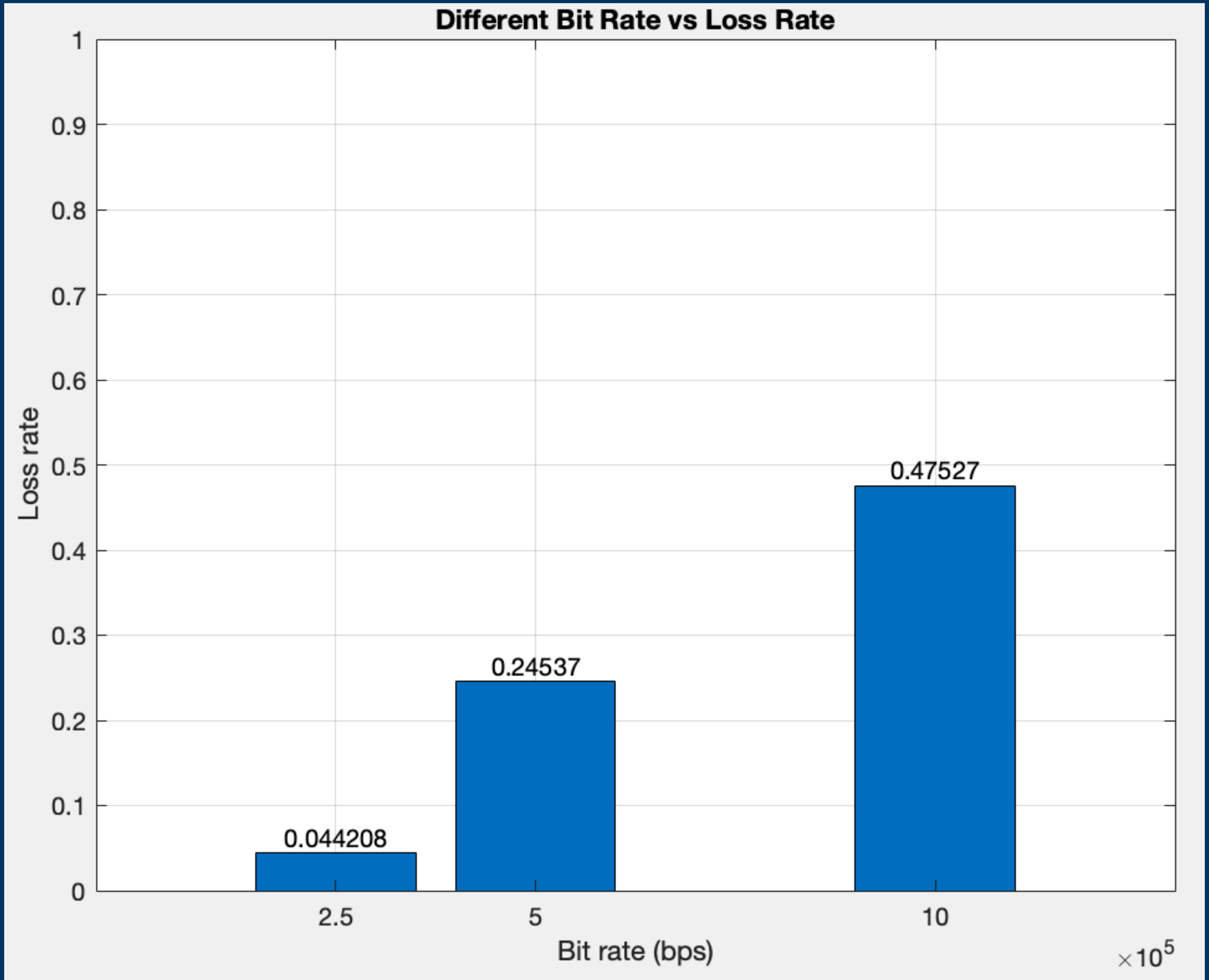
```
function SINR = get_SINR(num_devices, num_cell, h_bs, h_ms, p_m_W, gt_W, gr_W)
    BW = 10e6;                  % channel bandwidth (in Hz)
    T = 27+273.15;              % ambient temperature (in degree Kalvin)
    k = 1.38e-23;               % Boltzman's constant
    N = k*T*BW;                 % noise value
    global device_x;
    global device_y;
    global cell_x;
    global cell_y;
    global distance_all;

    for i = 1:num_devices
       for j = 1:num_cell
            dx = device_x(i) - cell_x(j);
            dy = device_y(i) - cell_y(j);
            distance_all(i, j) = sqrt(dx^2 + dy^2);
        end
    end

    gd = ((h_bs*h_ms)^2)./distance_all.^4;
    Pr_W = gd.*p_m_W*gt_W*gr_W;

    I = zeros(size(Pr_W)); % interference
    for i = 1:size(Pr_W,1)
       for j = 1:size(Pr_W,2)
            I(i,j) = sum(Pr_W(i, [1:j-1, j+1:end]));
        end
    end

    SINR = Pr_W./(I+N); % SINR in value
    return
end
```



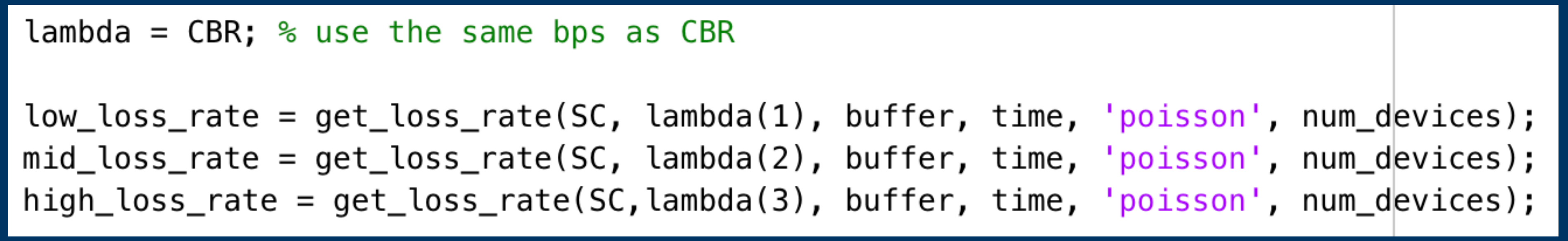
```
function SC = get_shannon(BW, num_devices, SINR)
    each_BW = BW/num_devices;
    SC = zeros(50,1); % shannon capacity (bits/s); SC is a matrix(50x1)

    for i = 1:50
        SC(i,1) = each_BW*log2(1+SINR(i,1));
    end
    return
end
```

I use 2 functions to get the Shannon Capacity.
First to get the SINR, then use the second one to get Shannon capacity.
Then plot the Shannon capacity.

B-3. Loss rate vs traffic load

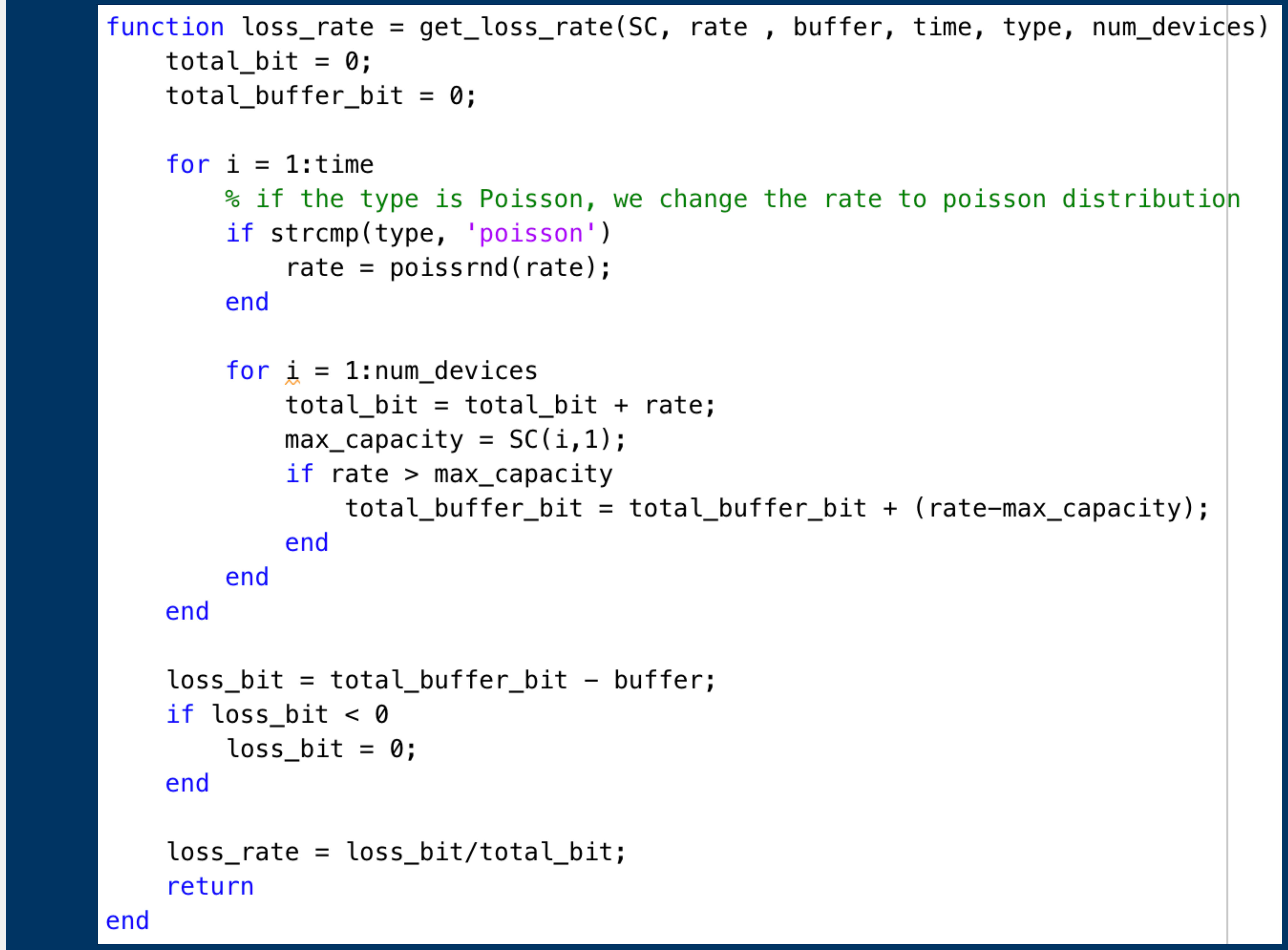


Parameters I used in this question



```
lambda = CBR; % use the same bps as CBR

low_loss_rate = get_loss_rate(SC, lambda(1), buffer, time, 'poisson', num_devices);
mid_loss_rate = get_loss_rate(SC, lambda(2), buffer, time, 'poisson', num_devices);
high_loss_rate = get_loss_rate(SC,lambda(3), buffer, time, 'poisson', num_devices);
```

Function that calculates the loss rate



```
function loss_rate = get_loss_rate(SC, rate , buffer, time, type, num_devices)
    total_bit = 0;
    total_buffer_bit = 0;

    for i = 1:time
        % if the type is Poisson, we change the rate to poisson distribution
        if strcmp(type, 'poisson')
            rate = poissrnd(rate);
        end

        for i = 1:num_devices
            total_bit = total_bit + rate;
            max_capacity = SC(i,1);
            if rate > max_capacity
                total_buffer_bit = total_buffer_bit + (rate-max_capacity);
            end
        end
    end

    loss_bit = total_buffer_bit - buffer;
    if loss_bit < 0
        loss_bit = 0;
    end

    loss_rate = loss_bit/total_bit;
    return
end
```

Reference

https://chat.openai.com/chat
I use chatGPT to help me understand the usage of function and global variable. Also the plot cell function is helped by it as well.
It help me with some definition and regulation about Matlab.