```matlab
%Copy the code in MATLAB and run the program
% 5G Network Simulation in MATLAB with User Input
clc;
clear;
close all;

%% User Input Using Dialog Box
prompt = {'Enter the number of cells in the network:', ...
          'Enter the number of users per cell:', ...
          'Enter the total simulation time in seconds:', ...
          'Enter the radius of each cell in meters:', ...
          'Enter the average user speed in m/s:', ...
          'Enter the handover RSSI threshold in dBm:', ...
          'Enter the transmit power of base stations in dBm:'};
dlgtitle = '5G Network Simulation Parameters';
dims = [1 50];
definput = {'3', '20', '100', '500', '5', '-80', '40'};
answer = inputdlg(prompt, dlgtitle, dims, definput);

% Check if the user cancelled the input dialog
if isempty(answer)
  error('Simulation cancelled by the user.');
end

% Convert inputs to numerical values and validate
numCells = str2double(answer{1});
numUsersPerCell = str2double(answer{2});
simulationTime = str2double(answer{3});
cellRadius = str2double(answer{4});
speed = str2double(answer{5});
```

```
handoverThreshold = str2double(answer{6});
baseStationPower = str2double(answer{7});

% Validate the inputs
if isnan(numCells) || isnan(numUsersPerCell) || isnan(simulationTime) || ...
   isnan(cellRadius) || isnan(speed) || isnan(handoverThreshold) || isnan(baseStationPower)
    error('Invalid input detected. Please enter numeric values only.');
end

%% Initialize Cell Locations (Assuming Hexagonal Grid)
cellCenters = [0, 0; 2*cellRadius, 0; cellRadius, sqrt(3)*cellRadius];
figure;
hold on;
for i = 1:numCells
    plot(cellCenters(i,1), cellCenters(i,2), 'b^', 'MarkerSize', 10, 'LineWidth', 2);
    viscircles(cellCenters(i,:), cellRadius, 'LineStyle', '--');
end
title('5G Network Topology');
xlabel('X (meters)');
ylabel('Y (meters)');
grid on;

%% User Initialization
numUsers = numCells * numUsersPerCell;
users = struct('position', zeros(numUsers, 2), 'velocity', zeros(numUsers, 2), 'cellID',
zeros(numUsers, 1), 'RSSI', zeros(numUsers, 1));

for i = 1:numUsers
    % Assign initial position randomly within a cell
    cellID = ceil(i / numUsersPerCell);
```

```
    angle = 2 * pi * rand();
    radius = cellRadius * sqrt(rand());
    users(i).position = cellCenters(cellID,:) + [radius * cos(angle), radius * sin(angle)];
    users(i).cellID = cellID;
    % Assign random velocity direction
    users(i).velocity = speed * [cos(2 * pi * rand()), sin(2 * pi * rand())];
end

%% Simulation Loop
throughput = zeros(numUsers, simulationTime);
latency = zeros(numUsers, simulationTime);
handoverCount = zeros(1, simulationTime);

for t = 1:simulationTime
    for i = 1:numUsers
        % Update user position
        users(i).position = users(i).position + users(i).velocity;

        % Wrap around if the user goes out of the simulation area
        if norm(users(i).position) > 2 * cellRadius
            angle = 2 * pi * rand();
            users(i).velocity = speed * [cos(angle), sin(angle)];
        end

        % Calculate RSSI for each user from all base stations
        RSSI = zeros(1, numCells);
        for j = 1:numCells
            distance = norm(users(i).position - cellCenters(j,:));
            RSSI(j) = baseStationPower - (20 * log10(distance + 1)); % Simple path loss model
        end
```

```
        % Associate user with the cell having maximum RSSI
        [maxRSSI, maxCellID] = max(RSSI);

        % Check for handover
        if maxCellID ~= users(i).cellID && maxRSSI > handoverThreshold
            users(i).cellID = maxCellID;
            handoverCount(t) = handoverCount(t) + 1;
        end

        % Store the RSSI for the associated cell
        users(i).RSSI = maxRSSI;

        % Calculate throughput and latency based on RSSI (simplified)
        if maxRSSI > 0
            throughput(i, t) = max(0, 100 * log2(1 + maxRSSI/20)); % Simplified Shannon capacity
formula
            latency(i, t) = 100 / throughput(i, t); % Inverse relationship for simplicity
        else
            throughput(i, t) = 0;
            latency(i, t) = inf;
        end
    end

    % Plot user positions
    cla;
    hold on;
    for j = 1:numCells
        plot(cellCenters(j,1), cellCenters(j,2), 'b^', 'MarkerSize', 10, 'LineWidth', 2);
        viscircles(cellCenters(j,:), cellRadius, 'LineStyle', '--');
```

```
    end
    plot([users.position]', 'ro', 'MarkerSize', 5);
    title(['5G Network Simulation - Time: ', num2str(t), ' seconds']);
    xlabel('X (meters)');
    ylabel('Y (meters)');
    grid on;
    pause(0.05);
end

%% Post-Simulation Analysis
averageThroughput = mean(throughput, 2);
averageLatency = mean(latency(~isinf(latency)), 2); % Exclude inf values for average
calculation
totalHandovers = sum(handoverCount);

fprintf('Average Throughput per User: %.2f Mbps\n', mean(averageThroughput));
fprintf('Average Latency per User: %.2f ms\n', mean(averageLatency));
fprintf('Total Handovers during Simulation: %d\n', totalHandovers);

%% Plotting the Throughput Distribution Using bar
figure;
throughputBins = linspace(min(averageThroughput), max(averageThroughput), 10);
throughputCounts = histcounts(averageThroughput, throughputBins);
bar(throughputBins(1:end-1), throughputCounts, 'histc');
title('Throughput Distribution');
xlabel('Throughput (Mbps)');
ylabel('Number of Users');
grid on;

%% Plotting the Latency Distribution Using bar
```

```
figure;
latencyBins = linspace(min(averageLatency), max(averageLatency), 10);
latencyCounts = histcounts(averageLatency, latencyBins);
bar(latencyBins(1:end-1), latencyCounts, 'histc');
title('Latency Distribution');
xlabel('Latency (ms)');
ylabel('Number of Users');
grid on;
```