Enhancing Cellular Network Performance Using UAV-Assisted Coverage

Under the guidance of **Prof. Abhishek K Gupta** (Mentor)
DEPARTMENT OF ELECTRICAL ENGINEERING
IIT Kanpur

SUBMITTED BY
Pratham Maan
B.E. (ECE), BITS Pilani (Pilani Campus)

FOR THE COMPLETION OF
Students-Undergraduate Research Graduate Excellence (SURGE-2025)

Indian Institute of Technology, Kanpur
Kanpur, Uttar Pradesh - 208016

ACKNOWLEDGEMENT

First and foremost, I would like to extend my profound appreciation to my supervisor, Prof. Abhishek K Gupta, for his guidance and expertise throughout this project. His valuable insights, constructive feedback, and advice were instrumental in shaping the direction and outcomes of this research. I got to learn a lot from him, including academic as well as other professional and disciplinary aspects.

I would also like to thank the Office of Outreach Activities (OOA) at IIT Kanpur and the entire SURGE team for providing me with such a fantastic opportunity to work on industry-level projects, which has helped me gain practical knowledge and research experience. Lastly, I would like to thank my parents who have been a constant support during my internship period.

In conclusion, this project was only possible with the combined efforts and support of all those mentioned above. Each individual has played a vital role in shaping this internship experience and the successful completion of this research endeavour. I am genuinely grateful for their contributions and look forward to applying the knowledge and skills gained from this internship in my future endeavours.

ABSTRACT

This report presents an in-depth exploration of methods to enhance the coverage and reliability of cellular networks by leveraging drone-mounted Unmanned Aerial Vehicles (UAVs) as aerial base stations using MATLAB-based simulations. The study is structured around a large-scale 19-cell hexagonal network, incorporating realistic user distributions and clusters by leveraging advanced optimisation techniques for UAVs' dynamic deployments in each hexagonal cell. The project encompasses the design and simulation of cellular networks, the dynamic allocation of UAVs to maximize user connectivity—particularly in challenging or clustered environments and their comprehensive performance analysis and comparison. The simulation scenario simulates users with realistic spatial distributions, including hotspots, commercial districts, residential districts, and sparse clusters, and therefore includes the complexities of modern urban networks. UAV deployment is dynamically optimised based on real-time user density and cluster importance, maximising overall network performance metrics such as signal-to-interference-plus-noise ratio (SINR), throughput and user coverage. The results demonstrate significant improvements in network performance and capabilities, offering insights into the creation of adaptive and resilient communication infrastructures capable of responding to real-world demands and emergencies. The outcome of this research will lead to the development of resilient, adaptive cellular networks that can adapt to coverage challenges in variable or densely populated environments, as well as be adaptive to the dynamic needs of disaster recovery operations.

Keywords: Cellular Network Modelling, UAV-Assisted Networks, Realistic User Distributions

Project Areas: Communication Networks, Communication Systems, Optimization

TABLE OF CONTENTS

INTRODUCTION

The rapid evolution of wireless communication technologies has revolutionised the way we communicate and interact with one another, work, and with information. Everywhere throughout urban and rural communities, everybody desires to have good mobile coverage at all times. Although cellular networks have improved a great deal, however, continuous and high-quality coverage remains a serious issue to attain.

Standard ground-based cell towers are the bulk of cellular networks. They are typically deployed in a hexagonal configuration to provide as wide an area of coverage as possible and minimise interference. This technique is effective in ideal situations, but in real-world situations, where user quantities fluctuate, buildings interfere, and demand spikes suddenly, it tends to fall short. Large events such as huge gatherings, natural disasters, or sudden population movement can drive the fixed systems over capacity, resulting in spots with no service and lower quality.

This work investigates the deployment of drones as aerial base stations in a massive, 19-cell hexagonal network. Drones possess a singular benefit: they are mobile and can be easily redeployed to other locations. This allows them to be perfectly suited for augmenting ground infrastructure during times of normalcy and catastrophe. By sending drones to areas of greatest need, networks can rapidly adapt to shifting user demand, environmental factors, and unplanned incidents.

BACKGROUND AND MOTIVATION

Traditionally, cellular networks are organised in a hexagonal grid layout—a design choice that efficiently enhances coverage areas and facilitates systematic frequency reuse. This geometric structuring is fundamental to maximising spectral efficiency and managing interference between adjacent cells.

However, the practical urban environments and modern user behaviour present significant challenges to this idealised model. In a real scenario, user distribution is rarely uniform; dense clusters form in hotspots such as stadiums, shopping malls, or business districts, while other areas may remain sparsely populated. Physical obstacles like buildings, terrain variations further complicate signal propagation, leading to shadowing and coverage holes. Additionally, unpredictable surges in user demand—such as those during large events or emergencies—can quickly overwhelm static terrestrial infrastructure, resulting in degraded service quality and connectivity gaps.

The Role of UAVs in Modern Networks

Unmanned Aerial Vehicles (UAVs) have emerged as a promising solution to these challenges. Their inherent mobility and rapid deployability allow them to act as aerial base stations, supplementing ground infrastructure precisely where and when additional coverage or capacity is needed. UAVs can be dynamically positioned to serve high-demand clusters, bypass obstacles, and restore connectivity in disaster-affected regions, making them invaluable for both routine network optimisation and emergency response scenarios.

Foundational Work Enabling UAV-Assisted Coverage

Before delving into UAV integration, a series of foundational studies were conducted to build a robust simulation and analysis platform:

Hexagonal Network Design

- Developed and simulated a large-scale 19-cell hexagonal network, incorporating realistic parameters such as cell radius, frequency reuse patterns, and inter-cell interference.
- Modelled user assignment and load distribution using stochastic processes to reflect real-world user behaviour.
- Evaluated baseline network performance using metrics like signal-to-interference-plus-noise ratio (SINR) and user throughput, establishing a benchmark for subsequent enhancements.

Antenna Diversity and Alamouti Coding

- Implemented Alamouti space-time block coding at the link level to address the detrimental effects of multipath fading—a common issue in wireless channels, especially in urban settings.
- Conducted simulations comparing the performance of Quadrature Phase Shift Keying (QPSK) modulation over additive white Gaussian noise (AWGN) and Rayleigh fading channels.
- Demonstrated that antenna diversity, particularly the Alamouti scheme, significantly reduces bit error rates (BER) in fading environments, thereby improving link reliability and quality.

Diversity Gain Analysis

- Compared traditional single-input single-output (SISO) systems with diversity-enabled configurations.
- Highlighted the practical benefits of diversity gain, such as improved coverage and service continuity in challenging environments.

Building Towards UAV-Based Enhancements

This comprehensive groundwork—spanning network topology, advanced modulation, and diversity techniques—created a solid analytical and simulation framework. With these elements in place, the project was well-positioned to explore the integration of UAVs as dynamic, adaptive network nodes. The ability to accurately model user behaviour, channel conditions, and baseline network performance ensured that the impact of UAV-assisted coverage could be rigorously assessed, paving the way for innovative solutions to the persistent challenges of modern cellular network.

SYSTEM MODEL AND SIMULATION FRAMEWORK

The section below gives the details of the model's parameters that have been used in the proposed system:

Network Parameters

- **Cells:** 19 hexagonal cells, each with a circumradius (R) of 1000 metres (1 km).
- **Base Stations (BS):** Located at the centre of each cell with a transmission power (P_t) of 46dBm (a typical value of transmitting base station antennas).
- **Frequency Reuse Factor:** Factor of 7 to manage interference.
- **Bandwidth:** 10MHz per cell.
- **Path Loss Exponent (η):** 4, to consider the shadowing effect for urban propagation conditions.

UAV Parameters

- **Altitude Range:** 10 meters to 120 meters (typical height of UAVs for networking applications).
- **Transmit Power (P_t):** Up to 23 dBm.
- **Carrier Frequency (f_c):** 2GHz

User Modelling

- **Density:** An average of 50 users per km^2, adjusted for clustering.
- **Distribution:** Users are placed in cells according to a Poisson process, with clustering patterns reflecting real-world scenarios –
  - **Hotspots:** High-density, typically near cell centres, i.e. near the BS.
  - **Commercial:** Medium-density, with 2-3 clusters per cell.
  - **Residential:** Moderate density, 3-5 clusters dispersed within the cell.
  - **Sparse:** Low density, with users more uniformly spread

  Each cell is randomly assigned a cluster type, and user density is scaled accordingly. This approach enables the model to capture the effects of urban heterogeneity on network performance.

Simulation Flow

1. **Cell and Base Station Placement:** Hexagonal grid generation and BS deployment.
2. **User Distribution:** Assignment of users to cells with non-uniform clustering.
3. **UAV Deployment:** Optimisation of UAV positions based on user density and cluster type.
4. **Performance Metrics:** Calculation of SINR, throughput and user assignment (BS vs UAV).

UAV Deployment and Optimisation

Optimising the placement of UAVs in a multi-cellular network is crucial to enhance coverage and improve the Signal-to-Interference-plus-Noise ratio (SINR) for users. The strategy involved user clustering analysis, mathematical optimisation, and adaptive altitude selection. Below is a detailed breakdown of the approach:

Weighted Centroid Calculation

To ensure UAVs are deployed where they are most needed, the algorithm first computes a **weighted centroid** for users within each cell. Here, users are assigned weights based on their cluster type, reflecting their relative importance –

- **Hotspot users:** Highest weight (here, 3.0)
- **Commercial users:** Moderate weight (here, 2.0)
- **Residential users:** Lower weight (here, 1.2)
- **Sparse users:** Lowest weight (here, 0.3)

The weighted centroid $\mathbf{C_w}$ for a set of users is calculated as:

$$C_w = \frac{\sum_{i=1}^{N} w_i u_i}{\sum_{i=1}^{N} w_i}$$

Fig. 1: Formula for calculating the weighted centroid in a cell

Where:

- N = number of users in the cells
- w_i = weight assigned to user i based on cluster type
- U_i = position vector of user i

This centroid serves as an initial anchor point for UAV placement, ensuring that UAVs are biased toward regions of higher demand.

Genetic Algorithm Optimisation

To further refine UAV positioning, a **genetic algorithm (GA)** is employed. The goal is to find the optimal (x, y, h) coordinates for the UAV within each cell that maximise a custom-defined score function, which encapsulates both SINR and proximity to cluster users.

Optimisation Variables:

- x, y: UAV's horizontal coordinates (within cell bounds)
- H: UAV altitude (constrained between minimum and maximum allowable heights)

Objective Function:

The score for a candidate UAV position is calculated as:

$$\text{Score} = \frac{1}{N} \sum_{i=1}^{N} w_i \cdot \text{SINR}_i - \lambda \cdot d_{\text{centroid}}$$

where:

> w_i = weight for user i (as above)
> SINR_i = SINR at user i from the candidate UAV position
> d_{centroid} = distance from UAV to the weighted centroid
> λ = penalty factor to discourage UAVs from staying too far from the weighted centroid

Fig. 2: Formula for calculating the Score of a Candidate UAV, i.e. quantifies how well a UAV, if positioned at a certain location and height, can serve the users in its assigned cell

SINR Calculation:

For a user at position $\mathbf{u}_i$ and a UAV at (x, y, h):

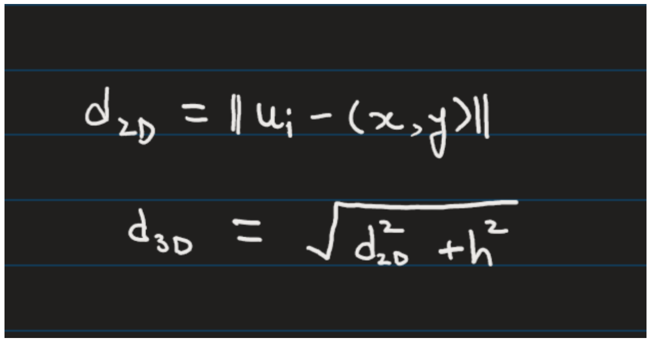

Fig. 3: Formulas for calculating the 2D Euclidean and 3D distance between a user and a UAV in the horizontal and 3D plane

The path loss and SINR are computed using:

$$\textbf{Path Loss}_{\text{LoS}} = 28.0 + 22\log_{10}(d_{3D}) + 20\log_{10}(f_c)$$

$$\textbf{Signal Power} = P_{tx,UAV} - \textbf{Path Loss}$$

$$\textbf{SINR}_i = \textbf{Signal Power} - \textbf{Noise Power}$$

where f_c is the carrier frequency (in GHz), and $P_{tx,UAV}$ is the UAV transmit power (in dBm).

Genetic Algorithm Workflow

- **Initialization:** Randomly generate a population of UAV positions within the cell.
- **Evaluation:** Calculate the score for each candidate using the above objective function.
- **Selection:** Retain the best-performing candidates.
- **Crossover and Mutation:** Generate new candidates by combining and slightly altering the best solutions.
- **Iteration:** Repeat evaluation and selection over multiple generations.
- **Termination:** Stop after a set number of generations or when improvement plateaus; select the candidate with the highest score as the optimal UAV position.

Height Selection

The UAV's altitude is not static but is dynamically optimized as part of the GA. This allows the algorithm to balance two competing factors:

- <u>Higher altitude</u>: Increases coverage area but may reduce received signal strength due to greater path loss.
- <u>Lower altitude</u>: Improves signal strength but limits coverage footprint.

The GA searches for the altitude that yields the best trade-off for the current user distribution.

PERFORMANCE EVALUATION AND RESULTS

The ordinary cellular network model consisting of only Base Stations (BS) has been compared with a network with both BS and UAVs on the following metrics:

- **SINR (Signal-to-Interference-plus-Noise Ratio):** Calculated for each user from both the BS and the UAV.
- **Throughput:** Derived from SINR using Shannon's formula:

$$C = B \cdot \log_2(1 + SINR)$$

  where:
  - ❖ C = Channel capacity or throughput (in bits per second, bps)
  - ❖ B = Channel bandwidth (in Hz)
  - ❖ SINR = Signal-to-Interference-plus-Noise Ratio (linear, not in dB)

- **User Assignment:** Each user is assigned to either the BS or UAV based on an SINR threshold (10dB).
- **Coverage Improvement:** Percentage of users served by UAVs, highlighting the enhancement over traditional networks.

Results:

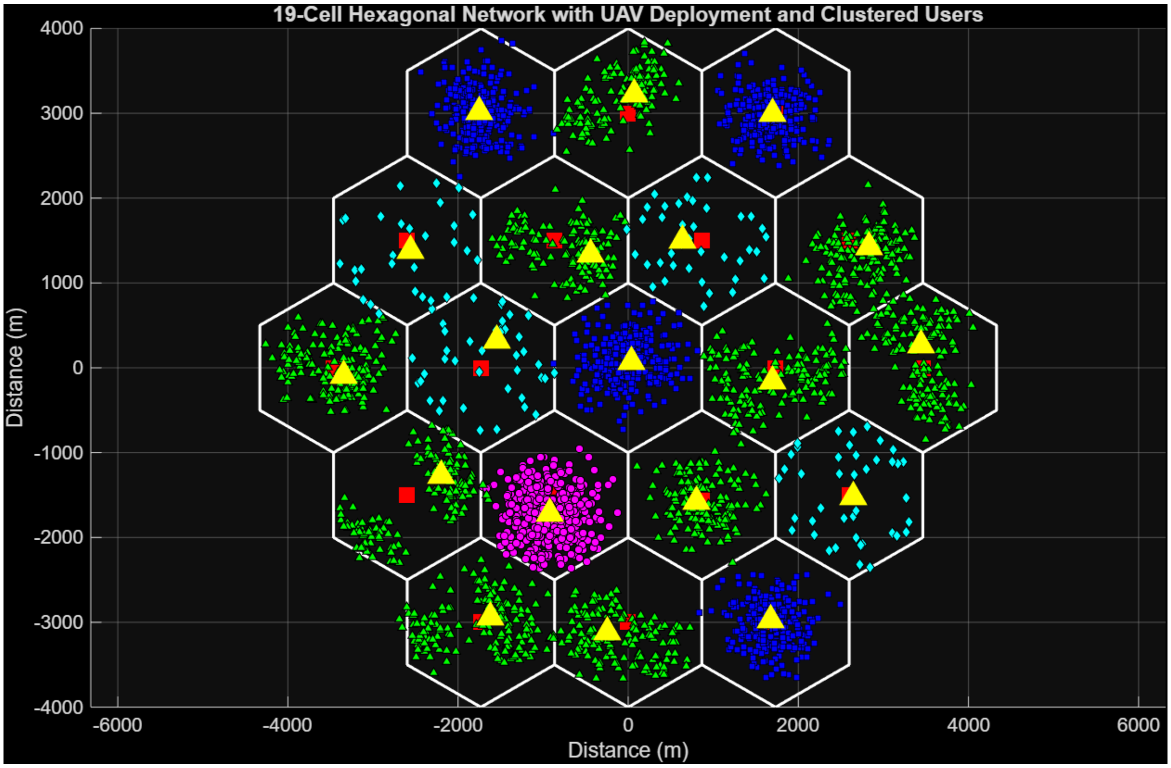

Fig. 4: The 19-cell hexagonal network, with base stations and UAVs,
is visualised alongside user distributions by cluster type.

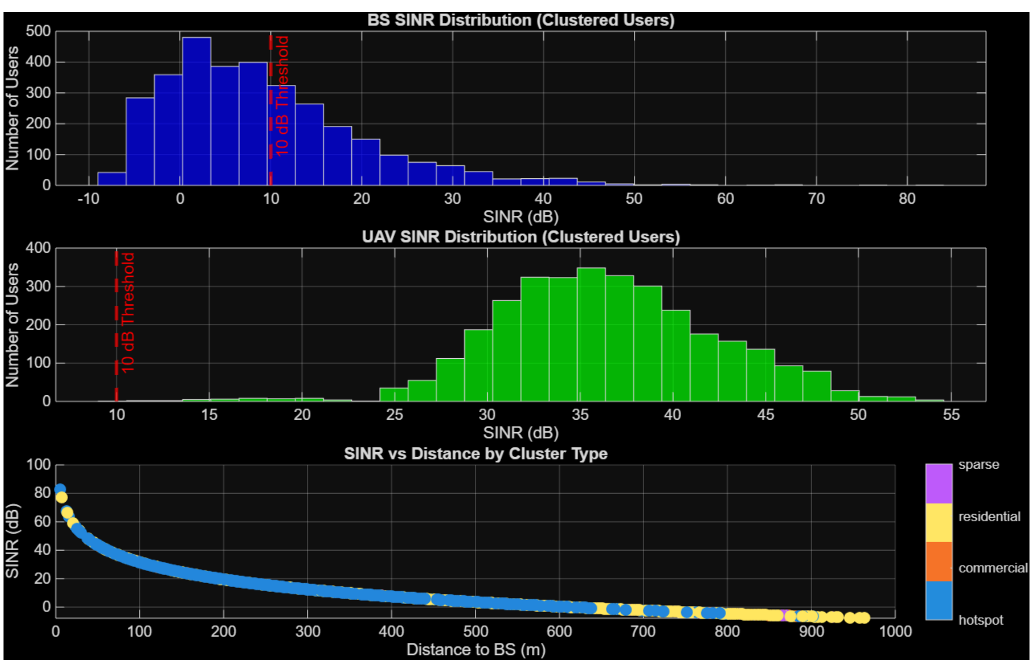

Fig. 5: Shows Number of Users vs SINR distribution for both BS and UAVs, and SINR variation with distance to BS

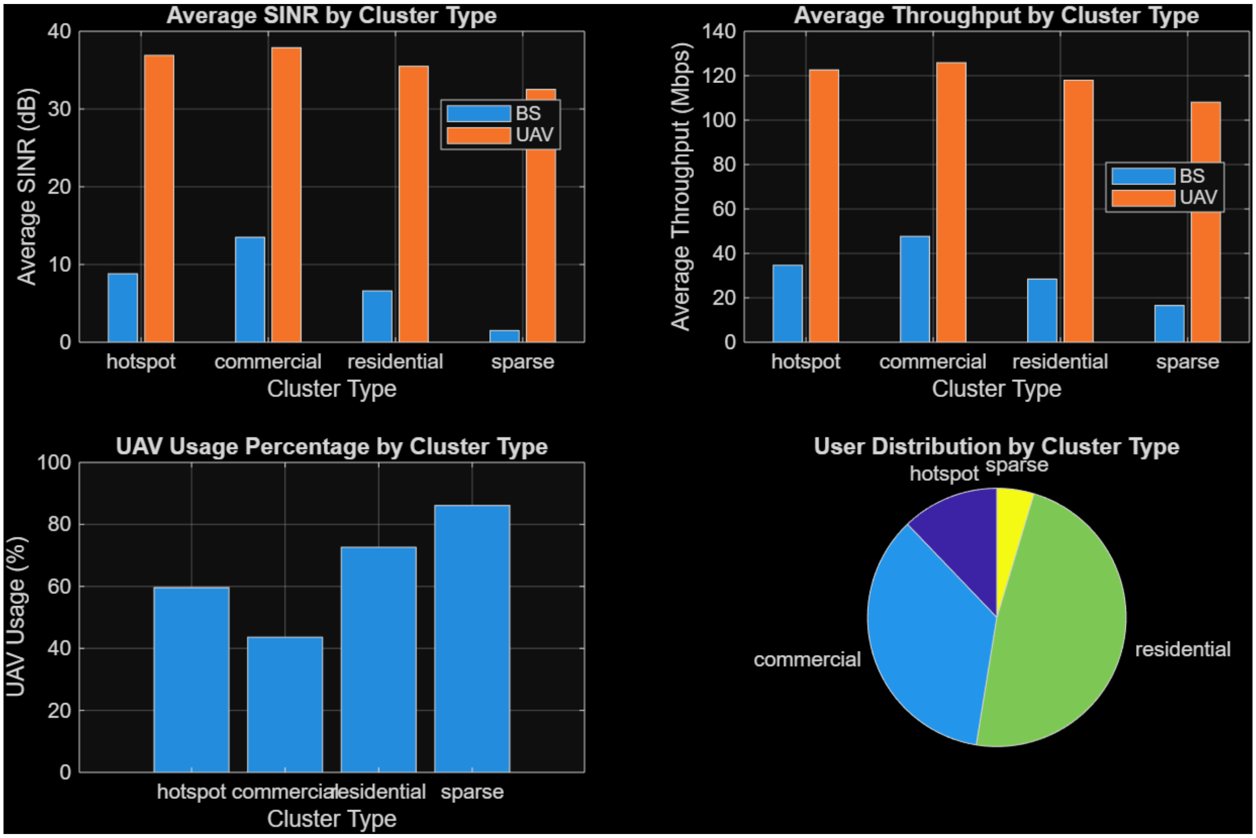

Fig. 6: Shows the SINR and Throughput relationship with the Type of Clusters and UAV Usage by Cluster Type

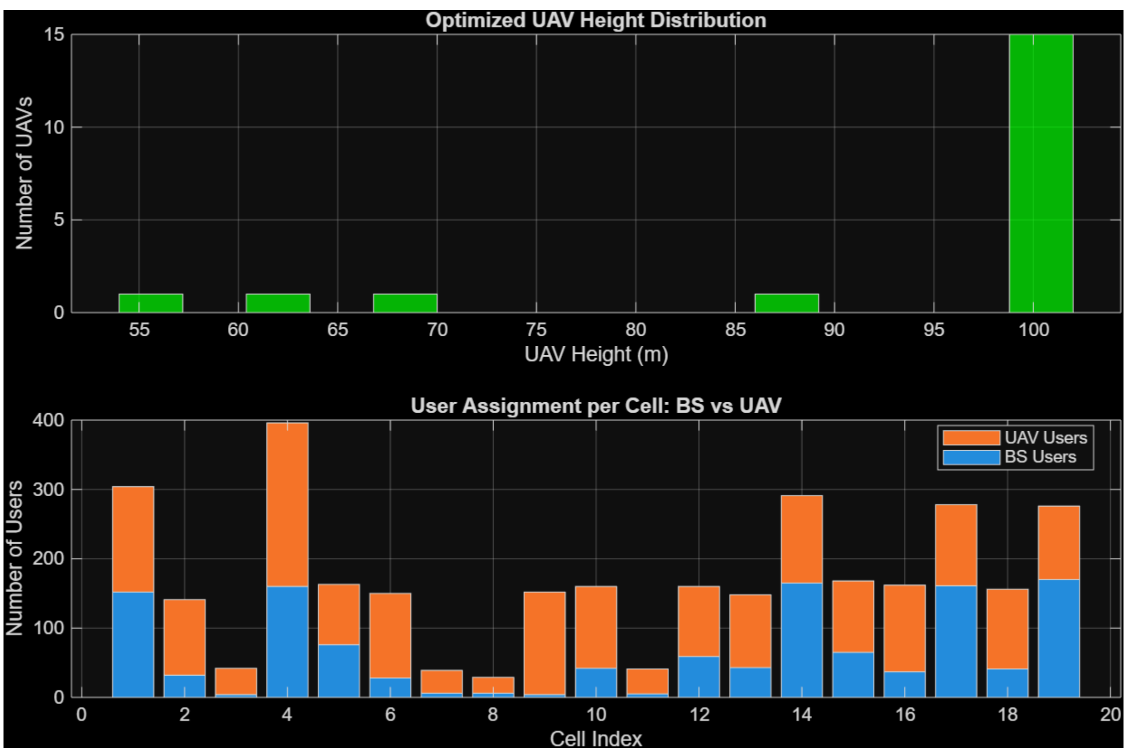

Fig. 7: Presents Optimised UAV Heights and per cell User Assignment - BS vs UAV

Inference

Network Visualisation: UAVs are typically positioned closer to high-density clusters, especially in hotspots and commercial areas.

User Assignment

- Total Users: Varies per simulation, typically several hundred.
- Users Served by BS: Around 36%. The majority are in well-covered areas.
- Users Served by UAV: Around 64%.Concentrated in clusters where BS SINR falls below the threshold.
- Coverage Improvement: UAVs consistently serve 60% of users, depending on clustering and network load.

SINR and Throughput

- Average BS SINR: Typically above 10 dB for most users, but drops in dense clusters or at cell edges.
- Average UAV SINR: Higher for users in coverage holes, validating UAV deployment.
- Throughput Gains: Notable for users reassigned to UAVs, especially in hotspot and commercial clusters.

Cluster Type Insights

- Hotspot Clusters: See the greatest benefit from UAVs, with significant improvements in both SINR and throughput.
- Commercial and Residential: Moderate gains, depending on the density and spread of users.
- Sparse Areas: Minimal UAV usage, as BS coverage is generally sufficient.

UAV Optimisation

- Altitude Distribution: Optimised UAV heights typically range from 30 to 100 meters, balancing coverage and signal strength.
- Per-Cell Analysis: Some cells require more UAV intervention than others, depending on user clustering and density.

CONCLUSION

This project report documents my learning experience with UAVs and how they can be optimised, emphasising the revolutionary potential of UAV-assisted cellular networks in addressing the longstanding challenges of modern wireless communication systems. With the dynamic deployment of UAVs as aerial base stations, the network achieves an unparalleled degree of flexibility, which makes it capable of effectively mitigating real-time variations in user demand, spatial user clustering, and environmental limitations. The use of UAVs in the cellular system fills gaps in coverage that would otherwise occur in dense urban environments where traditional infrastructure can be disrupted. UAVs are quickly deployed into regions of poor terrestrial coverage, keeping users connected even in the most challenging environments. By taking advantage of real-time knowledge of user locations and clustering patterns, UAVs are best positioned to push the SINR for poor-coverage users to their maximum values.

Lastly, this research demonstrates how the integration of UAV technology into sophisticated network modelling and optimisation can produce cellular networks that are not merely more efficient and resilient but also natively adaptive to the dynamics of the real-world communication needs. The findings developed here are the foundation of the future generation of resilient, user-centric wireless networks—networks that can deliver tomorrow's dynamic and diverse environments the connectivity they require.

APPENDIX

My source code for the above proposed model is shown below:

```
clear;
close all;
clc;
% n/w param
R = 1000; % Cell circumradius(m)
N_cells = 19;% No of hexagonal cells
lambda_users = 50;% Avg user density per cell (users per km^2)
P_tx_BS = 46;% BS P_t in dBm
P_tx_UAV = 23; % UAV P_t in dBm
noise_power = -174;% Noise PSD in dBm/Hz
bandwidth = 10e6;% System bandwidth in Hz
path_loss_exp = 4; % Path loss exp
freq_reuse_factor = 7; % Freq reuse factor
SINR_threshold = 10; % SINR threshold in dB for BS/UAV selection
%UAV Parameters
UAV_height_min = 10;  % Min UAV height (m)
UAV_height_max = 120; % Max UAV height (m)
UAV_power_max = 23;% Max UAV power (dBm)
fc_GHz = 2;% Carrier freq in GHz
% User Distribution Parameters for Non-Uniform Clustering
cluster_types = {'hotspot', 'commercial', 'residential', 'sparse'};
cluster_probabilities = [0.2, 0.3, 0.4, 0.1]; % Probability of each cluster type
hotspot_intensity = 3.0; % Multiplier for hotspot areas
commercial_intensity = 2.0;% Multiplier for commercial areas
residential_intensity = 1.2;% Multiplier for residential areas
sparse_intensity = 0.3;% Multiplier for sparse areas
%Generate Hexagonal Cell Structure
ISD = sqrt(3) * R;
cell_centers = generate_hex_centers_rotated(R);
% Plot network structure
figure(1);
plot_hexagonal_cells_rotated(cell_centers, R);
title('19-Cell Hexagonal Network with UAV Deployment and Clustered Users');
xlabel('Distance (m)'); ylabel('Distance (m)');
grid on; axis equal;
% Deploy Base Stations
BS_positions = cell_centers;
N_BS = size(BS_positions, 1);
hold on;
plot(BS_positions(:,1), BS_positions(:,2), 'rs', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
%Generate Users with Non-Uniform Clustering
fprintf('Generating users with non-uniform clustering patterns...\n');
all_users = [];
user_cell_assignment = [];
user_cluster_types = {};
for cell_idx = 1:N_cells
   cell_area = 3 * sqrt(3) * R^2 / 2;
   cell_area_km2 = cell_area / 1e6;

   % Determine cluster type for this cell
   cluster_type = cluster_types{randsample(length(cluster_types), 1, true,
cluster_probabilities)};
```

```
   % Adjust user density based on cluster type
   switch cluster_type
       case 'hotspot'
           effective_lambda = lambda_users * hotspot_intensity;
       case 'commercial'
           effective_lambda = lambda_users * commercial_intensity;
       case 'residential'
           effective_lambda = lambda_users * residential_intensity;
       case 'sparse'
           effective_lambda = lambda_users * sparse_intensity;
   end

   N_users_cell = poissrnd(effective_lambda * cell_area_km2);

   %Generate clustered user positions
   users_in_cell = generate_clustered_users_in_hex(cell_centers(cell_idx, :), R, N_users_cell,
cluster_type);

   % Store users
   all_users = [all_users; users_in_cell];
   user_cell_assignment = [user_cell_assignment; cell_idx * ones(N_users_cell, 1)];
   user_cluster_types = [user_cluster_types; repmat({cluster_type}, N_users_cell, 1)];

   fprintf('Cell %d: %s cluster with %d users\n', cell_idx, cluster_type, N_users_cell);
end
N_total_users = size(all_users, 1);
% Plot users based on cluster type
plot_clustered_users(all_users, user_cluster_types);
%Optimize UAV Positions
fprintf('Optimizing UAV positions for %d cells...\n', N_cells);
UAV_positions = optimize_UAV_positions_enhanced(cell_centers, all_users, user_cell_assignment, R,
user_cluster_types);
% Plot optimized UAV positions
plot(UAV_positions(:,1), UAV_positions(:,2), 'y^', 'MarkerSize', 12, 'MarkerFaceColor', 'y');
legend('Cell Boundaries', 'Base Stations', 'Hotspot Users', 'Commercial Users', ...
      'Residential Users', 'Sparse Users', 'UAVs', 'Location', 'best');
% Calculate SINR and Throughput for All Users
fprintf('Calculating SINR and throughput for all users...\n');
% Initialize arrays
SINR_BS = zeros(N_total_users, 1);
SINR_UAV = zeros(N_total_users, 1);
throughput_BS = zeros(N_total_users, 1);
throughput_UAV = zeros(N_total_users, 1);
user_assignment = zeros(N_total_users, 1); % 0: BS, 1: UAV
user_distances_BS = zeros(N_total_users, 1);
user_distances_UAV = zeros(N_total_users, 1);
for user_idx = 1:N_total_users
   user_pos = all_users(user_idx, :);
   serving_cell = user_cell_assignment(user_idx);
   serving_BS = BS_positions(serving_cell, :);
   serving_UAV = UAV_positions(serving_cell, :);

   %distance to BS and UAV
   d_BS = norm(user_pos - serving_BS);
   d_UAV_2D = norm(user_pos - serving_UAV(1:2));
   d_UAV_3D = sqrt(d_UAV_2D^2 + serving_UAV(3)^2);

   user_distances_BS(user_idx) = d_BS;
   user_distances_UAV(user_idx) = d_UAV_3D;

   % SINR from BS
   SINR_BS(user_idx) = calculate_SINR_BS(user_pos, serving_BS, BS_positions, ...
```

```
                                        user_cell_assignment, P_tx_BS, path_loss_exp);

    %SINR from UAV
    SINR_UAV(user_idx) = calculate_SINR_UAV(user_pos, serving_UAV, UAV_positions, ...
                                        user_cell_assignment, P_tx_UAV, fc_GHz);

    % throughput
    throughput_BS(user_idx) = bandwidth * log2(1 + 10^(SINR_BS(user_idx)/10)) / 1e6;
    throughput_UAV(user_idx) = bandwidth * log2(1 + 10^(SINR_UAV(user_idx)/10)) / 1e6;

    % User assignment based on threshold
    if SINR_BS(user_idx) >= SINR_threshold
        user_assignment(user_idx) = 0; % Served by BS
    else
        user_assignment(user_idx) = 1; % Served by UAV
    end
end
%Performance Analysis
users_served_by_BS = sum(user_assignment == 0);
users_served_by_UAV = sum(user_assignment == 1);
coverage_improvement = users_served_by_UAV / N_total_users * 100;
fprintf('\n=== UAV-Assisted Network Performance with Clustered Users ===\n');
fprintf('Total users: %d\n', N_total_users);
fprintf('Users served by BS: %d (%.1f%%)\n', users_served_by_BS,
users_served_by_BS/N_total_users*100);
fprintf('Users served by UAV: %d (%.1f%%)\n', users_served_by_UAV,
users_served_by_UAV/N_total_users*100);
fprintf('Coverage improvement: %.1f%%\n', coverage_improvement);
fprintf('Average BS SINR: %.2f dB\n', mean(SINR_BS));
fprintf('Average UAV SINR: %.2f dB\n', mean(SINR_UAV));
fprintf('Average BS throughput: %.2f Mbps\n', mean(throughput_BS));
fprintf('Average UAV throughput: %.2f Mbps\n', mean(throughput_UAV));
% Analyze performance by cluster type
analyze_performance_by_cluster(user_cluster_types, SINR_BS, SINR_UAV, throughput_BS,
throughput_UAV, user_assignment);
% Plot 2: Cluster Distribution Analysis
figure(2);
subplot(3,1,1);
histogram(SINR_BS, 30, 'FaceColor', 'blue', 'FaceAlpha', 0.7);
xlabel('SINR (dB)'); ylabel('Number of Users');
title('BS SINR Distribution (Clustered Users)');
xline(SINR_threshold, 'r--', 'LineWidth', 2, 'Label', '10 dB Threshold');
grid on;
subplot(3,1,2);
histogram(SINR_UAV, 30, 'FaceColor', 'green', 'FaceAlpha', 0.7);
xlabel('SINR (dB)'); ylabel('Number of Users');
title('UAV SINR Distribution (Clustered Users)');
xline(SINR_threshold, 'r--', 'LineWidth', 2, 'Label', '10 dB Threshold');
grid on;
subplot(3,1,3);
scatter(user_distances_BS, SINR_BS, 50, categorical(user_cluster_types), 'filled');
colormap(lines(4));
colorbar('Ticks', 1:4, 'TickLabels', cluster_types);
xlabel('Distance to BS (m)'); ylabel('SINR (dB)');
title('SINR vs Distance by Cluster Type');
grid on;
%Performance Comparison by Cluster Type
figure(3);
plot_cluster_performance_comparison(user_cluster_types, SINR_BS, SINR_UAV, throughput_BS,
throughput_UAV, user_assignment);
%UAV Optimization Results
figure(4);
```

```
subplot(2,1,1);
histogram(UAV_positions(:,3), 15, 'FaceColor', 'green', 'FaceAlpha', 0.7);
xlabel('UAV Height (m)'); ylabel('Number of UAVs');
title('Optimized UAV Height Distribution');
grid on;
%Calc per-cell user assignment data
users_per_cell_BS = zeros(N_cells, 1);
users_per_cell_UAV = zeros(N_cells, 1);
for cell_idx = 1:N_cells
   cell_users = find(user_cell_assignment == cell_idx);
   users_per_cell_BS(cell_idx) = sum(user_assignment(cell_users) == 0);
   users_per_cell_UAV(cell_idx) = sum(user_assignment(cell_users) == 1);
end
subplot(2,1,2);
bar_data = [users_per_cell_BS, users_per_cell_UAV];
bar(1:N_cells, bar_data, 'stacked');
xlabel('Cell Index'); ylabel('Number of Users');
title('User Assignment per Cell: BS vs UAV');
legend('BS Users', 'UAV Users', 'Location', 'best');
grid on;
%Fns for Enhanced User Distribution
function users = generate_clustered_users_in_hex(center, R, N_users, cluster_type)
   % Generate clustered users within hexagonal cell based on cluster type
   users = zeros(N_users, 2);

   switch cluster_type
       case 'hotspot'
           % Single high-density cluster near cell center
           cluster_centers = [center + [R*0.2*randn(), R*0.2*randn()]];
           cluster_weights = [1.0];
           cluster_spreads = [R*0.3];

       case 'commercial'
           % 2-3 medium-density clusters
           n_clusters = randi([2, 3]);
           cluster_centers = zeros(n_clusters, 2);
           for i = 1:n_clusters
               angle = 2*pi*rand();
               distance = R * 0.4 * rand();
               cluster_centers(i, :) = center + distance * [cos(angle), sin(angle)];
           end
           cluster_weights = ones(1, n_clusters) / n_clusters;
           cluster_spreads = R * 0.25 * ones(1, n_clusters);

       case 'residential'
           % 3-5 smaller clusters distributed across cell
           n_clusters = randi([3, 5]);
           cluster_centers = zeros(n_clusters, 2);
           for i = 1:n_clusters
               angle = 2*pi*rand();
               distance = R * (0.2 + 0.5*rand());
               cluster_centers(i, :) = center + distance * [cos(angle), sin(angle)];
           end
           cluster_weights = ones(1, n_clusters) / n_clusters;
           cluster_spreads = R * 0.2 * ones(1, n_clusters);

       case 'sparse'
           % Uniform distribution with slight clustering
           cluster_centers = [center];
           cluster_weights = [1.0];
           cluster_spreads = [R*0.8];
   end
```

```matlab
    % Generate users based on cluster parameters
    for i = 1:N_users
        valid = false;
        attempts = 0;
        max_attempts = 100;

        while ~valid && attempts < max_attempts
            attempts = attempts + 1;

            % Select cluster based on weights
            cluster_idx = randsample(length(cluster_weights), 1, true, cluster_weights);
            cluster_center = cluster_centers(cluster_idx, :);
            cluster_spread = cluster_spreads(cluster_idx);

            % Generate user position around cluster center
            if strcmp(cluster_type, 'sparse')
                % More uniform distribution for sparse areas
                r = R * sqrt(rand());
                theta = 2 * pi * rand();
                x = center(1) + r * cos(theta);
                y = center(2) + r * sin(theta);
            else
                % Gaussian clustering around cluster center
                x = cluster_center(1) + cluster_spread * randn();
                y = cluster_center(2) + cluster_spread * randn();
            end

            % Check if point is inside hexagon
            if is_inside_hexagon_rotated([x, y], center, R, 120 * pi / 180)
                users(i, :) = [x, y];
                valid = true;
            end
        end

        % Fallback to uniform distribution if clustering fails
        if ~valid
            users(i, :) = generate_uniform_user_in_hex(center, R);
        end
    end
end
function user = generate_uniform_user_in_hex(center, R)
    % Fallback uniform user generation
    valid = false;
    while ~valid
        r = R * sqrt(rand());
        theta = 2 * pi * rand();
        x = center(1) + r * cos(theta);
        y = center(2) + r * sin(theta);

        if is_inside_hexagon_rotated([x, y], center, R, 120 * pi / 180)
            user = [x, y];
            valid = true;
        end
    end
end
function plot_clustered_users(all_users, user_cluster_types)
    % Plot users with different colors based on cluster type
    cluster_colors = containers.Map({'hotspot', 'commercial', 'residential', 'sparse'}, ...
                                    {'magenta', 'blue', 'green', 'cyan'});
    cluster_markers = containers.Map({'hotspot', 'commercial', 'residential', 'sparse'}, ...
                                     {'o', 's', '^', 'd'});
```

```matlab
    hold on;
    for cluster_type = {'hotspot', 'commercial', 'residential', 'sparse'}
        cluster_type = cluster_type{1};
        cluster_users = all_users(strcmp(user_cluster_types, cluster_type), :);

        if ~isempty(cluster_users)
            plot(cluster_users(:,1), cluster_users(:,2), ...
                 cluster_markers(cluster_type), 'MarkerSize', 4, ...
                 'MarkerFaceColor', cluster_colors(cluster_type), ...
                 'MarkerEdgeColor', 'black', 'LineWidth', 0.5);
        end
    end
end
function UAV_positions = optimize_UAV_positions_enhanced(cell_centers, all_users,
user_cell_assignment, R, user_cluster_types)
    % Enhanced UAV position optimization considering user clustering
    N_cells = size(cell_centers, 1);
    UAV_positions = zeros(N_cells, 3);

    for cell_idx = 1:N_cells
        fprintf('Optimizing UAV position for cell %d...\n', cell_idx);

        cell_users_idx = find(user_cell_assignment == cell_idx);
        if isempty(cell_users_idx)
            UAV_positions(cell_idx, :) = [cell_centers(cell_idx, :), 100];
            continue;
        end

        cell_users = all_users(cell_users_idx, :);
        cell_clusters = user_cluster_types(cell_users_idx);

        % Calculate weighted centroid based on cluster importance
        cluster_weights = containers.Map({'hotspot', 'commercial', 'residential', 'sparse'}, ...
                                        {3.0, 2.0, 1.2, 0.5});

        weighted_center = [0, 0];
        total_weight = 0;

        for i = 1:length(cell_users_idx)
            weight = cluster_weights(cell_clusters{i});
            weighted_center = weighted_center + weight * cell_users(i, :);
            total_weight = total_weight + weight;
        end

        if total_weight > 0
            weighted_center = weighted_center / total_weight;
        else
            weighted_center = cell_centers(cell_idx, :);
        end

        % Optimize around weighted centroid
        cell_center = cell_centers(cell_idx, :);
        x_bounds = [cell_center(1) - R*0.8, cell_center(1) + R*0.8];
        y_bounds = [cell_center(2) - R*0.8, cell_center(2) + R*0.8];
        h_bounds = [10, 120];

        options = optimoptions('ga', 'Display', 'off', 'MaxGenerations', 50);
        lb = [x_bounds(1), y_bounds(1), h_bounds(1)];
        ub = [x_bounds(2), y_bounds(2), h_bounds(2)];
```

```
        objective = @(pos) -evaluate_UAV_position_enhanced(pos, cell_users, cell_clusters,
weighted_center);

        [optimal_pos, ~] = ga(objective, 3, [], [], [], [], lb, ub, [], options);
        UAV_positions(cell_idx, :) = optimal_pos;
    end
end
function score = evaluate_UAV_position_enhanced(UAV_pos, users, clusters, weighted_center)
    % Enhanced UAV position evaluation considering clustering
    if isempty(users)
        score = 0;
        return;
    end

    cluster_weights = containers.Map({'hotspot', 'commercial', 'residential', 'sparse'}, ...
                                    {3.0, 2.0, 1.2, 0.5});

    total_score = 0;
    P_tx_UAV = 23;
    fc_GHz = 2;

    % Distance penalty from weighted centroid
    centroid_distance = norm(UAV_pos(1:2) - weighted_center);
    centroid_penalty = centroid_distance / 1000; % Normalize

    for i = 1:size(users, 1)
        user_pos = users(i, :);
        cluster_type = clusters{i};
        cluster_weight = cluster_weights(cluster_type);

        d_2D = norm(user_pos - UAV_pos(1:2));
        d_3D = sqrt(d_2D^2 + UAV_pos(3)^2);

        [path_loss, ~] = calculate_UAV_path_loss(d_2D, UAV_pos(3), fc_GHz);
        signal_power = P_tx_UAV - path_loss;

        noise_power_dBm = -174 + 10*log10(10e6);
        SINR = signal_power - noise_power_dBm;

        if SINR > 0
            total_score = total_score + cluster_weight * SINR;
        end
    end

    score = (total_score / size(users, 1)) - centroid_penalty;
end
function analyze_performance_by_cluster(user_cluster_types, SINR_BS, SINR_UAV, throughput_BS,
throughput_UAV, user_assignment)
    % Analyze network performance by cluster type
    fprintf('\n=== Performance Analysis by Cluster Type ===\n');

    cluster_types = unique(user_cluster_types);

    for i = 1:length(cluster_types)
        cluster_type = cluster_types{i};
        cluster_indices = strcmp(user_cluster_types, cluster_type);

        cluster_SINR_BS = SINR_BS(cluster_indices);
        cluster_SINR_UAV = SINR_UAV(cluster_indices);
        cluster_throughput_BS = throughput_BS(cluster_indices);
        cluster_throughput_UAV = throughput_UAV(cluster_indices);
        cluster_assignment = user_assignment(cluster_indices);
```

```
        bs_users = sum(cluster_assignment == 0);
        uav_users = sum(cluster_assignment == 1);
        total_users = length(cluster_assignment);

        fprintf('\n%s Cluster:\n', upper(cluster_type));
        fprintf('  Total users: %d\n', total_users);
        fprintf('  BS users: %d (%.1f%%)\n', bs_users, bs_users/total_users*100);
        fprintf('  UAV users: %d (%.1f%%)\n', uav_users, uav_users/total_users*100);
        fprintf('  Avg BS SINR: %.2f dB\n', mean(cluster_SINR_BS));
        fprintf('  Avg UAV SINR: %.2f dB\n', mean(cluster_SINR_UAV));
        fprintf('  Avg BS throughput: %.2f Mbps\n', mean(cluster_throughput_BS));
        fprintf('  Avg UAV throughput: %.2f Mbps\n', mean(cluster_throughput_UAV));
    end
end
function plot_cluster_performance_comparison(user_cluster_types, SINR_BS, SINR_UAV,
throughput_BS, throughput_UAV, user_assignment)
    % Create comprehensive performance comparison plots
    cluster_types = {'hotspot', 'commercial', 'residential', 'sparse'};
    n_clusters = length(cluster_types);

    % Prepare data for plotting
    avg_sinr_bs = zeros(1, n_clusters);
    avg_sinr_uav = zeros(1, n_clusters);
    avg_throughput_bs = zeros(1, n_clusters);
    avg_throughput_uav = zeros(1, n_clusters);
    uav_usage_percent = zeros(1, n_clusters);

    for i = 1:n_clusters
        cluster_type = cluster_types{i};
        cluster_indices = strcmp(user_cluster_types, cluster_type);

        if sum(cluster_indices) > 0
            avg_sinr_bs(i) = mean(SINR_BS(cluster_indices));
            avg_sinr_uav(i) = mean(SINR_UAV(cluster_indices));
            avg_throughput_bs(i) = mean(throughput_BS(cluster_indices));
            avg_throughput_uav(i) = mean(throughput_UAV(cluster_indices));
            uav_usage_percent(i) = sum(user_assignment(cluster_indices) == 1) /
sum(cluster_indices) * 100;
        end
    end

    subplot(2,2,1);
    bar(1:n_clusters, [avg_sinr_bs; avg_sinr_uav]', 'grouped');
    set(gca, 'XTickLabel', cluster_types);
    xlabel('Cluster Type'); ylabel('Average SINR (dB)');
    title('Average SINR by Cluster Type');
    legend('BS', 'UAV', 'Location', 'best');
    grid on;

    subplot(2,2,2);
    bar(1:n_clusters, [avg_throughput_bs; avg_throughput_uav]', 'grouped');
    set(gca, 'XTickLabel', cluster_types);
    xlabel('Cluster Type'); ylabel('Average Throughput (Mbps)');
    title('Average Throughput by Cluster Type');
    legend('BS', 'UAV', 'Location', 'best');
    grid on;

    subplot(2,2,3);
    bar(1:n_clusters, uav_usage_percent);
    set(gca, 'XTickLabel', cluster_types);
    xlabel('Cluster Type'); ylabel('UAV Usage (%)');
```

```
    title('UAV Usage Percentage by Cluster Type');
    grid on;

    subplot(2,2,4);
    user_counts = zeros(1, n_clusters);
    for i = 1:n_clusters
        user_counts(i) = sum(strcmp(user_cluster_types, cluster_types{i}));
    end
    pie(user_counts, cluster_types);
    title('User Distribution by Cluster Type');
end
function SINR_dB = calculate_SINR_BS(user_pos, serving_BS, all_BS, user_cell_assignment, P_tx,
path_loss_exp)
    d_serving = norm(user_pos - serving_BS);
    path_loss_serving = calculate_path_loss(d_serving, path_loss_exp);
    signal_power = P_tx - path_loss_serving;

    interference_power = 0;
    noise_power_dBm = -174 + 10*log10(10e6);

    signal_power_linear = 10^((signal_power - 30)/10);
    noise_power_linear = 10^((noise_power_dBm - 30)/10);

    SINR_linear = signal_power_linear / (interference_power + noise_power_linear);
    SINR_dB = 10 * log10(SINR_linear);
end
function SINR_dB = calculate_SINR_UAV(user_pos, serving_UAV, all_UAV, user_cell_assignment, P_tx,
fc_GHz)
    d_2D = norm(user_pos - serving_UAV(1:2));
    %d_3D = sqrt(d_2D^2 + serving_UAV(3)^2);

    [path_loss, ~] = calculate_UAV_path_loss(d_2D, serving_UAV(3), fc_GHz);
    signal_power = P_tx - path_loss;

    noise_power_dBm = -174 + 10*log10(10e6);

    signal_power_linear = 10^((signal_power - 30)/10);
    noise_power_linear = 10^((noise_power_dBm - 30)/10);

    SINR_linear = signal_power_linear / noise_power_linear;
    SINR_dB = 10 * log10(SINR_linear);
end
function [path_loss_dB, is_LoS] = calculate_UAV_path_loss(d_2D, height, fc_GHz)
    d_3D = sqrt(d_2D^2 + height^2);

    if height > 100
        P_LoS = 1;
    elseif d_2D <= max(460*log10(height) - 700, 18)
        P_LoS = 1;
    else
        d1 = max(460*log10(height) - 700, 18);
        p1 = 4300*log10(height) - 3800;
        P_LoS = d1/d_2D + exp(-d_2D/p1) * (1 - d1/d_2D);
    end

    is_LoS = rand() < P_LoS;

    if is_LoS
        path_loss_dB = 28.0 + 22*log10(d_3D) + 20*log10(fc_GHz);
    else
        path_loss_dB = -17.5 + (46 - 7*log10(height))*log10(d_3D) + 20*log10(40*pi*fc_GHz/3);
    end
```

```matlab
end
function path_loss_dB = calculate_path_loss(distance, path_loss_exp)
   if distance == 0
       path_loss_dB = 0;
   else
       f_GHz = 2;
       PL_1m = 32.44 + 20 * log10(f_GHz);
       path_loss_dB = PL_1m + 10 * path_loss_exp * log10(distance);
   end
end
function centers = generate_hex_centers_rotated(R)
   ISD = sqrt(3) * R;
   rotation_angle = 120 * pi / 180;
   rotation_matrix = [cos(rotation_angle) -sin(rotation_angle);
                      sin(rotation_angle)  cos(rotation_angle)];

   centers = zeros(19, 2);
   centers(1, :) = [0, 0];

   thetaRing1 = 0:pi/3:(5*pi/3);
   coordinatesRing1 = [cos(thetaRing1); sin(thetaRing1)]';
   centers(2:7, :) = ISD * coordinatesRing1;

   thetaRing2 = pi/6 + (0:pi/3:(5*pi/3));
   coordinatesRing2 = [cos(thetaRing2); sin(thetaRing2)]';
   centers(8:13, :) = ISD * sqrt(3) * coordinatesRing2;
   centers(14:19, :) = 2 * ISD * coordinatesRing1;

   for i = 1:19
       rotated_center = rotation_matrix * centers(i, :)';
       centers(i, :) = rotated_center';
   end
end
function plot_hexagonal_cells_rotated(centers, R)
   rotation_angle = 90 * pi / 180;
   hold on;
   for i = 1:size(centers, 1)
       hex_x = []; hex_y = [];
       for angle = 0:60:300
           angle_rad = deg2rad(angle) + rotation_angle;
           hex_x = [hex_x, centers(i,1) + R * cos(angle_rad)];
           hex_y = [hex_y, centers(i,2) + R * sin(angle_rad)];
       end
       hex_x = [hex_x, hex_x(1)]; hex_y = [hex_y, hex_y(1)];
       plot(hex_x, hex_y, 'w-', 'LineWidth', 1.5);
   end
end
function inside = is_inside_hexagon_rotated(point, center, R, rotation_angle)
   p = point - center;
   inside = true;
   apothem = R * cos(pi/6);

   for angle = 0:60:300
       angle_rad = deg2rad(angle + 30) + rotation_angle;
       normal = [cos(angle_rad), sin(angle_rad)];

       if dot(p, normal) > apothem
           inside = false;
           break;
       end
   end
end
```