

```

clear all
close all
clc

rng("default");

% --- System Parameters ---
h0 = 550; iK = 53; Nsat = 22; Norbits = 72; U = 50; dt = 10; Tmax = 3600;
deltaPhi = 30; M = Nsat * Norbits; T = Tmax / dt + 1;
user_latitudes = 45 + (50 - 45) * rand(U, 1);
user_longitudes = 55 + (60 - 55) * rand(U, 1);

elevation_ts_all = zeros(M, T, U); distance_ts_all = zeros(M, T, U);
for u = 1:U
    for ind = 1:Norbits
        base_idx = (ind - 1) * Nsat + 1;
        [elev_ts, dist_ts, ~] = generate_passage_singleorbit(h0, iK, ...
            (ind - 1) * deltaPhi, -(ind - 1) * 360 / Nsat / Norbits, ...
            Nsat, user_latitudes(u), user_longitudes(u), dt, Tmax);
        elevation_ts_all(base_idx:base_idx + Nsat - 1, :, u) = elev_ts;
        distance_ts_all(base_idx:base_idx + Nsat - 1, :, u) = dist_ts;
    end
end

% --- Elevation Quantization ---
allowed_elevs = [30, 45, 60, 70];
partition = [25, 37.5, 52.5, 65, inf];
elevation_quantized_all = NaN(size(elevation_ts_all));
for u = 1:U
    quantized_indices = discretize(elevation_ts_all(:, :, u), partition);
    temp = NaN(size(elevation_ts_all(:, :, u)));
    valid_idx = ~isnan(quantized_indices);
    temp(valid_idx) = allowed_elevs(quantized_indices(valid_idx));
    temp(elevation_ts_all(:, :, u) < 5) = NaN;
    elevation_quantized_all(:, :, u) = temp;
end

% --- Channel Fading Traces ---
f_c = 2e9; envi = 'Village'; Tch = 5000;
channel_trace_map = containers.Map('KeyType', 'double', 'ValueType',
    'any');
for k = 1:length(allowed_elevs)
    elev = allowed_elevs(k);
    try
        [y, ~, ~, ~, ~, ~] = coef_time_series_long_pedestrian(f_c, envi,
            elev, Tch);
        channel_trace_map(elev) = abs(y).^2; % Only average used
    catch
        channel_trace_map(elev) = zeros(1, Tch);
    end
end

% --- Transmit power (fixed) ---
Ps = 1; % Watts

% --- Weight Matrices ---
[W_SNR_ideal, W_Rate_ideal, W_SNR_forecast, W_Rate_forecast] = ...

```

```

    build_weight_matrices_AT(Ps, channel_trace_map,
elevation_quantized_all, distance_ts_all, allowed_elevs, f_c);

% --- Run Ideal Simulation (Algo 1) ---
[snr_ideal, rate_ideal, snr_ideal_primary, snr_ideal_dual,
rate_ideal_primary, rate_ideal_dual] = ...
    simulate_ideal_mode_algo_1(W_SNR_ideal, W_Rate_ideal,
elevation_quantized_all, distance_ts_all, channel_trace_map,
allowed_elevs, f_c, Ps);

% --- Run Forecast Simulation (Algo 1) ---
[snr_forecast, rate_forecast, snr_forecast_primary, snr_forecast_dual,
rate_forecast_primary, rate_forecast_dual] = ...
    simulate_forecast_mode_algo_1(W_SNR_forecast, W_Rate_forecast,
W_SNR_ideal, W_Rate_ideal, elevation_quantized_all, distance_ts_all,
channel_trace_map, allowed_elevs, f_c, Ps);

% --- Store Results ---
results_ideal = struct('snr', snr_ideal, 'rate', rate_ideal, ...
    'snr_primary', snr_ideal_primary, 'snr_dual', snr_ideal_dual, ...
    'rate_primary', rate_ideal_primary, 'rate_dual', rate_ideal_dual);

results_forecast = struct('snr', snr_forecast, 'rate', rate_forecast, ...
    'snr_primary', snr_forecast_primary, 'snr_dual', snr_forecast_dual,
...
    'rate_primary', rate_forecast_primary, 'rate_dual',
rate_forecast_dual);

% --- Save Results Folder ---
output_folder = fullfile(pwd, 'SISO_algo1_Plots_Ps1');
if ~exist(output_folder, 'dir')
    mkdir(output_folder);
end
save_plot = @(name) saveas(gcf, fullfile(output_folder, name));

% --- Per-User Average Metrics ---
user_rate_mean_ideal = mean(rate_ideal, 2, 'omitnan');
user_rate_mean_forecast = mean(rate_forecast, 2, 'omitnan');
user_snr_mean_ideal = mean(snr_ideal, 2, 'omitnan');
user_snr_mean_forecast = mean(snr_forecast, 2, 'omitnan');

% --- Overall Means ---
overall_rate_ideal = mean(user_rate_mean_ideal, 'omitnan');
overall_rate_forecast = mean(user_rate_mean_forecast, 'omitnan');
overall_snr_ideal = mean(user_snr_mean_ideal, 'omitnan');
overall_snr_forecast = mean(user_snr_mean_forecast, 'omitnan');

fprintf('Overall Avg Rate (Ideal):    %.4f bps/Hz\n',
overall_rate_ideal);
fprintf('Overall Avg Rate (Forecast): %.4f bps/Hz\n',
overall_rate_forecast);
fprintf('Overall Avg SNR (Ideal):    %.4f dB\n', overall_snr_ideal);
fprintf('Overall Avg SNR (Forecast): %.4f dB\n', overall_snr_forecast);

% --- Save .mat Results ---
save(fullfile(output_folder, 'results_avg_Ps1_algo1.mat'), ...
    'user_rate_mean_ideal', 'user_rate_mean_forecast', ...
    'user_snr_mean_ideal', 'user_snr_mean_forecast', ...

```

```

    'overall_rate_ideal', 'overall_rate_forecast', ...
    'overall_snr_ideal', 'overall_snr_forecast');

% ===== PLOTTING (simple + consistent axes) =====
dt_label = sprintf('Time Step (%d s)', dt); % For any time-based plots
you add later
meta_line = sprintf('U = %d users, T = %d time steps (%d s each), %s', U,
T, dt, envi);

% --- CDF of average rate per user ---
figure;
cdfplot(user_rate_mean_ideal); hold on;
cdfplot(user_rate_mean_forecast);
legend('Ideal', 'Forecast', 'Location', 'best');
xlabel('Average Rate per User (bps/Hz)', 'FontWeight', 'bold');
ylabel('Cumulative Probability', 'FontWeight', 'bold');
title({'CDF of Per-User Average Rate (Ps = 1)', meta_line}, 'FontWeight',
'bold');
grid on;
axis tight; xlim padded; ylim padded;
save_plot('CDF_User_Avg_Rate_Village.png');

% --- Bar plot of average rate per user ---
figure;
bar([user_rate_mean_ideal, user_rate_mean_forecast]);
xlabel('User Index', 'FontWeight', 'bold');
ylabel('Average Rate (bps/Hz)', 'FontWeight', 'bold');
legend('Ideal', 'Forecast', 'Location', 'best');
title({'Per-User Average Rate Comparison (Ps = 1)', meta_line},
'FontWeight', 'bold');
grid on;
axis tight; ylim padded;
save_plot('Bar_User_Avg_Rate_Village.png');

% --- CDF of average SNR per user ---
figure;
cdfplot(user_snr_mean_ideal); hold on;
cdfplot(user_snr_mean_forecast);
legend('Ideal', 'Forecast', 'Location', 'best');
xlabel('Average SNR per User (dB)', 'FontWeight', 'bold');
ylabel('Cumulative Probability', 'FontWeight', 'bold');
title({'CDF of Per-User Average SNR (Ps = 1)', meta_line}, 'FontWeight',
'bold');
grid on;
axis tight; xlim padded; ylim padded;
save_plot('CDF_User_Avg_SNR_Village.png');

% --- Bar plot of average SNR per user ---
figure;
bar([user_snr_mean_ideal, user_snr_mean_forecast]);
xlabel('User Index', 'FontWeight', 'bold');
ylabel('Average SNR (dB)', 'FontWeight', 'bold');
legend('Ideal', 'Forecast', 'Location', 'best');
title({'Per-User Average SNR Comparison (Ps = 1)', meta_line},
'FontWeight', 'bold');
grid on;
axis tight; ylim padded;
save_plot('Bar_User_Avg_SNR_Village.png');

```