```
clear;
clc;
close all;

startup();

%% Parameters
p = default_params();

%% Initial relative state (LVLH)
x0 = [p.R_init_m; 0; 0; 0; 0; 0]; % State x = [x y z xdot↙
ydot zdot]'

%% Initial attitude and wheel state

q0 = [1; 0; 0; 0]; % quat
w0 = [0; 0; 0]; % body rates rad per second
wh0 = [0; 0; 0]; % wheel speeds rad per second

%% Pre allocate

N = floor(p.t_final_s / p.dt_s) + 1;
t = (0:N-1)' * p.dt_s;

x_hist  = zeros(N, 6);
u_hist  = zeros(N, 3);
r_hist  = zeros(N, 1);

q_hist  = zeros(N, 4);
w_hist  = zeros(N, 3);
tau_w_hist = zeros(N, 3);
wh_hist = zeros(N, 3);
mode_hist = zeros(N, 1);
h_w = zeros(3,1);

in_band_time   = 0;
```

```
outside_time   = 0;
in_band_time_hist = zeros(N,1);
outside_time_hist = zeros(N,1);

los_time = 0;
los_hist = zeros(N, 1);

%% LQR gain for translation

[A, B] = cw_state_space(p.n_radps);
[K, ~, ~] = lqr(A, B, p.Q_lqr, p.R_lqr);

%% Sim loop

x = x0;
q = q0;
w = w0;
wh = wh0;
r_rel = x(1:3);
r = norm(r_rel);

for k = 1:N
    tk = t(k);

    % Reference fly around path
    ref = inspection_reference(tk, p);

    % Translational guidance and KOZ handling
    [u_cmd, mode] = translational_guidance_lqr(x, ref, K,↙
p);

    % Attitude to target
    r_rel = x(1:3);

    if norm(r_rel) < 1e-6
```

```
        r_rel = [1; 0; 0];

    end

    R_des = desired_attitude_pointing(r_rel, p);
    q_des = dcm_to_quat(R_des);


    % Turn on reaction wheels when inside inspection↙
range
    if norm(r_rel) < 90


        tau_body_cmd = attitude_pd_torque(q, w, q_des,↙
h_w,p);




    else
        tau_body_cmd = [0; 0; 0];
    end

    % Reaction wheel actuation and attitude dynamics↙
integration
    [q_next, w_next, wh_next, tau_w] =↙
step_attitude_wheels(q, w, wh, tau_body_cmd, p, p.dt_s);

    % Orbit integration
    x_next = rk4_step_translation(x, u_cmd, p, p.dt_s);

    % Logs
    x_hist(k,:) = x';
    u_hist(k,:) = u_cmd';
    r_hist(k)   = norm(x(1:3));

    % Line of sigh check
```

```
    has_los = check_los(q, r_rel, p);

    if has_los

        los_time = los_time + p.dt_s;

    end

    los_hist(k) = has_los;

    % Dwell tracking inside range band

    if r_hist(k) >= p.R_min_m && r_hist(k) <= p.R_max_m

        in_band_time = in_band_time + p.dt_s;
        outside_time = 0;

    else

        outside_time = outside_time + p.dt_s;

    end

    dwell_hist(k) = in_band_time;

    q_hist(k,:) = q';
    w_hist(k,:) = w';
    tau_w_hist(k,:) = tau_w';
    wh_hist(k,:) = wh';
    mode_hist(k) = mode;

    x = x_next;
    q = q_next;
    w = w_next;
    wh = wh_next;
```

```
    h_w = h_w + (-tau_body_cmd) * p.dt_s;

end

%% Plots
make_plots(t, x_hist, u_hist, r_hist, tau_w_hist,↙
wh_hist, mode_hist, q_hist, los_hist, p)


%% Coverage analysis
[coverage_pct, unique_views, covered_patches] =↙
compute_coverage(x_hist, los_hist, p);

total_patches = 36 * 18;

fprintf('\n=== MISSION PERFORMANCE ===\n');

fprintf('Surface Coverage: %.1f%% / %.0f%% required (%d↙
of %d patches)\n', ...
    coverage_pct, p.required_coverage_pct,↙
covered_patches, total_patches);

fprintf('Unique Viewing Angles: %d\n', unique_views);

fprintf('Dwell Time in Band: %.1f s / %.0f s required\n',↙
...
    in_band_time, p.required_dwell_s);

fprintf('LOS Time within %.0f m: %.1f s\n', ...
    p.los_max_range_m, los_time);
```