```
function [u_cmd, mode] = translational_guidance_lqr(x,↙
ref, K, p)

% State error
x_ref = [ref.r; ref.v];
e = x - x_ref;

% Feedforward acceleration
n = p.n_radps;
x_r = ref.r(1); y_r = ref.r(2); z_r = ref.r(3);
xd_r = ref.v(1); yd_r = ref.v(2); zd_r = ref.v(3);

ax_ff = ref.a(1) - (3*n^2*x_r + 2*n*yd_r);
ay_ff = ref.a(2) - (-2*n*xd_r);
az_ff = ref.a(3) - (-n^2*z_r);

a_ff = [ax_ff; ay_ff; az_ff];

% LQR feedback
u_cmd = -K * e + a_ff;

% Axis saturation
u_cmd = max(min(u_cmd, p.a_max_mps2), -p.a_max_mps2);

% Range band constraint
rvec = x(1:3);
vvec = x(4:6);
r = norm(rvec);

mode = 0;

if r > 1e-9
    rhat = rvec / r;
    vr = dot(vvec, rhat);

    inner_soft = p.R_min_m + p.band_margin_m;
```

```
outer_soft = p.R_max_m - p.band_margin_m;

a_cons = [0;0;0];

% Soft enforcement near the inner boundary
if r < inner_soft
    mode = 1;
    dr = inner_soft - r;
    a_rep = p.band_k_rep * dr * rhat;
    a_damp = -p.band_k_damp * vr * rhat;
    a_cons = a_cons + a_rep + a_damp;
end

% Soft enforcement near the outer boundary
if r > outer_soft
    mode = 1;
    dr = r - outer_soft;
    a_rep = -p.band_k_rep * dr * rhat;
    a_damp = -p.band_k_damp * vr * rhat;
    a_cons = a_cons + a_rep + a_damp;
end

% Hard return if outside the allowed band
if r < p.R_min_m
    mode = 2;
    a_cons = p.a_max_mps2 * rhat;

elseif r > p.R_max_m
    mode = 2;
    a_cons = -p.a_max_mps2 * rhat;
end

u_cmd = u_cmd + a_cons;

% Saturation again
u_cmd = max(min(u_cmd, p.a_max_mps2), -p.a_max_mps2);
```

```
end

end
```