```
clear
clc
close all

%% Time
dt = 0.01;
tf = 60;
t  = (0:dt:tf).';
nt = numel(t);

%% Constants and parameters
p.g = 9.80665;

% Missile properties
p.mM   = 150;          % kg
p.CdM  = 0.6;
p.Aref = 0.05;         % m^2
p.aMax = 35 * p.g;     % m/s^2
p.tauA = 0.15;         % s

% Simple atmosphere
p.rho0 = 1.225;
p.H    = 8500;         % m scale height
p.zMin = 0;

% Seeker
p.Ts        = 0.05;                % s sample period
p.sigmaAz   = 0.7e-3;              % rad
p.sigmaEl   = 0.7e-3;              % rad
p.sigmaR    = 5.0;                 % m
p.biasAz    = 0.2e-3;              % rad
p.biasEl    = -0.15e-3;            % rad
p.biasR     = 2.0;                 % m
p.fovAz     = deg2rad(35);         % rad
p.fovEl     = deg2rad(25);         % rad

% EKF tuning (target is modeled as constant velocity with accel process noise)
p.qAcc  = 12^2;                    % (m/s^2)^2 spectral density proxy
p.Rmeas = diag([p.sigmaAz^2, p.sigmaEl^2, p.sigmaR^2]);

% APN acceleration estimate filter
p.tauAccEst = 0.5;                 % s low pass on accel estimate

% Intercept
p.Rkill = 5.0;

rng(7);

%% Truth initial conditions (inertial, z up)
rM0 = [0; 0; 0];
vM0 = [600; 0; 0];

rT0 = [8500; 2500; 1200];
vT0 = [-250; -20; 0];

aM0 = [0; 0; 0];

xTruth = [rM0; vM0; rT0; vT0; aM0];
```

```
%% EKF initial guess
xHat = [rT0 + [80; -60; 40]; vT0 + [8; -6; 4]];
P    = diag([150^2, 150^2, 150^2, 30^2, 30^2, 30^2]);

vTHatPrev = xHat(4:6);
aTHatLP   = [0; 0; 0];

%% Logs
xTruthHist = zeros(nt, 15);
xTruthHist(1,:) = xTruth.';

xHatHist   = nan(nt, 6);
xHatHist(1,:) = xHat.';

PHistTrace = nan(nt,1);

zHist      = nan(nt, 3);
seekerOK   = false(nt,1);

aCmdHist   = nan(nt, 3);
aSatHist   = nan(nt, 3);
aAchHist   = nan(nt, 3);

modeHist   = zeros(nt,1);
NHisto     = nan(nt,1);
KapnHisto  = nan(nt,1);

rangeHist  = nan(nt,1);
VcHist     = nan(nt,1);
missHist   = nan(nt,1);

hitIndex = nt;

nextMeasTime = 0.0;

%% Main loop
for k = 1:nt-1
    tk = t(k);
    rM = xTruth(1:3);
    vM = xTruth(4:6);
    rT = xTruth(7:9);
    vT = xTruth(10:12);
    aM = xTruth(13:15);

    rRel = rT - rM;
    vRel = vT - vM;
    R = norm(rRel);
    rangeHist(k) = R;
    missHist(k)  = R;

    if R < p.Rkill
        hitIndex = k;
        break
    end

    %% EKF prediction each dt
    [xHat, P] = ekf_predict_target_cv(xHat, P, dt, p);
    %% Seeker measurement update at sample rate
    doMeas = (tk + 1e-12) >= nextMeasTime;
    if doMeas
        [z, ok] = seeker_measurement(rRel, p);
        seekerOK(k) = ok;
        zHist(k,:)  = z.';
        if ok
            [xHat, P] = ekf_update_target(xHat, P, z, rM, p);
        end
        nextMeasTime = nextMeasTime + p.Ts;
    end

    xHatHist(k,:) = xHat.';
    PHistTrace(k) = trace(P);

    %% Acceleration estimate for APN from EKF target velocity
    if doMeas
        aTHat = (xHat(4:6) - vTHatPrev) / p.Ts;
        vTHatPrev = xHat(4:6);
        alpha = min(p.Ts / p.tauAccEst, 1.0);
        aTHatLP = aTHatLP + alpha * (aTHat - aTHatLP);
    end

    %% Guidance using estimated target state
    rRelHat = xHat(1:3) - rM;
    vRelHat = xHat(4:6) - vM;

    if norm(rRelHat) < 1e-6
        aCmd = [0; 0; 0];
        Vc   = 0;
        mode = 0;
        Nnav = 0;
        Kapn = 0;
    else
        [aCmd, Vc, mode, Nnav, Kapn] = guidance_modes(rRelHat, vRelHat, aTHatLP, p);
    end

    aCmdHist(k,:) = aCmd.';
    VcHist(k) = Vc;
    modeHist(k) = mode;
    NHisto(k) = Nnav;
    KapnHisto(k) = Kapn;

    aSat = saturate_vector(aCmd, p.aMax);
    aSatHist(k,:) = aSat.';
    aAchHist(k,:) = aM.';

    %% Truth propagation one step with RK4 holding aSat constant over dt
    k1 = dyn_truth(tk, xTruth, aSat, p);
    k2 = dyn_truth(tk + 0.5*dt, xTruth + 0.5*dt*k1, aSat, p);
    k3 = dyn_truth(tk + 0.5*dt, xTruth + 0.5*dt*k2, aSat, p);
    k4 = dyn_truth(tk + dt, xTruth + dt*k3, aSat, p);

    xTruth = xTruth + (dt/6) * (k1 + 2*k2 + 2*k3 + k4);

    xTruthHist(k+1,:) = xTruth.';
end
```

```
%% Final logs
tUse = t(1:hitIndex);
xTuse = xTruthHist(1:hitIndex,:);

rM = xTuse(:,1:3);
vM = xTuse(:,4:6);
rT = xTuse(:,7:9);
vT = xTuse(:,10:12);
aM = xTuse(:,13:15);

rRel = rT - rM;
rangeHist(hitIndex) = norm(rRel(end,:).');
missHist(hitIndex)  = rangeHist(hitIndex);

%% Plots
figure
plot3(rM(:,1), rM(:,2), rM(:,3), "LineWidth", 1.5)
hold on
plot3(rT(:,1), rT(:,2), rT(:,3), "LineWidth", 1.5)
grid on
axis equal
xlabel("x m")
ylabel("y m")
zlabel("z m")
legend("Missile", "Target", "Location", "best")
title("3D trajectories")

figure
plot(tUse, rangeHist(1:hitIndex), "LineWidth", 1.5)
grid on
xlabel("t s")
ylabel("Range m")
title("Range versus time")

figure
plot(tUse, vecnorm(aCmdHist(1:hitIndex,:),2,2), "LineWidth", 1.5)
hold on
plot(tUse, vecnorm(aSatHist(1:hitIndex,:),2,2), "LineWidth", 1.5)
grid on
xlabel("t s")
ylabel("Accel magnitude m per s squared")
legend("Commanded", "Saturated", "Location", "best")
title("Guidance command and saturation")

figure
stairs(tUse, modeHist(1:hitIndex), "LineWidth", 1.5)
grid on
xlabel("t s")
ylabel("Mode index")
title("Guidance mode switching")

figure
plot(tUse, vecnorm((xHatHist(1:hitIndex,1:3) - rT(1:hitIndex,:)),2,2), "LineWidth", 1.5)
grid on
xlabel("t s")
ylabel("Target position estimation error m")
title("EKF target position error norm")
```

```
figure
plot(tUse, VcHist(1:hitIndex), "LineWidth", 1.5)
grid on
xlabel("t s")
ylabel("Closing speed m per s")
title("Closing speed")

disp("Final range m")
disp(rangeHist(hitIndex))

if rangeHist(hitIndex) < p.Rkill
    disp("Intercept achieved")
else
    disp("No intercept within final time")
end

%%
%% 3D animated view (interactive)
fig = figure;
grid on
axis equal
xlabel("x m"); ylabel("y m"); zlabel("z m")
title("3D engagement animation")
view(3)
rotate3d on

hold on
trailM = animatedline("LineWidth", 1.5);
trailT = animatedline("LineWidth", 1.5);

hM = plot3(rM(1,1), rM(1,2), rM(1,3), "o", "MarkerSize", 7, "LineWidth", 1.5);
hT = plot3(rT(1,1), rT(1,2), rT(1,3), "o", "MarkerSize", 7, "LineWidth", 1.5);

hLOS = plot3([rM(1,1) rT(1,1)], [rM(1,2) rT(1,2)], [rM(1,3) rT(1,3)], "LineWidth", 1.0);

% Optional: keep camera centered near the missile
followMissile = true;

% Optional: save to MP4
saveVideo = true;
if saveVideo
    v = VideoWriter("intercept3d.mp4", "MPEG-4");
    v.FrameRate = 30;
    open(v);
end

step = 5; % increase for faster playback

for k = 1:step:numel(tUse)
    addpoints(trailM, rM(k,1), rM(k,2), rM(k,3));
    addpoints(trailT, rT(k,1), rT(k,2), rT(k,3));

    set(hM, "XData", rM(k,1), "YData", rM(k,2), "ZData", rM(k,3));
    set(hT, "XData", rT(k,1), "YData", rT(k,2), "ZData", rT(k,3));

    set(hLOS, "XData", [rM(k,1) rT(k,1)], "YData", [rM(k,2) rT(k,2)], "ZData", [rM(k,3)↙
rT(k,3)]);
```

```
    if followMissile
        cx = rM(k,1); cy = rM(k,2); cz = rM(k,3);
        xlim([cx-2000 cx+2000])
        ylim([cy-2000 cy+2000])
        zlim([max(cz-2000,0) cz+2000])
    end

    drawnow

    if saveVideo
        writeVideo(v, getframe(fig));
    end
end

if saveVideo
    close(v);
    disp("Saved intercept3d.mp4")
end

%% ======== Functions ========

function xdot = dyn_truth(t, x, aSat, p)
    rM = x(1:3);
    vM = x(4:6);
    rT = x(7:9);
    vT = x(10:12);
    aM = x(13:15);

    % Target maneuver (aircraft style, lateral to velocity)
    aT = target_accel(t);
    vTn = norm(vT);
    if vTn > 1e-9
        vThat = vT / vTn;
        aT = aT - dot(aT, vThat) * vThat;
    end

    % Missile gravity and drag
    aG = [0; 0; -p.g];

    z = max(rM(3), p.zMin);
    rho = p.rho0 * exp(-z / p.H);

    V = norm(vM);
    if V > 1e-9
        vHat = vM / V;
        Dmag = 0.5 * rho * V^2 * p.CdM * p.Aref;
        aD   = -(Dmag / p.mM) * vHat;
    else
        aD = [0; 0; 0];
    end

    rMdot = vM;
    vMdot = aM + aG + aD;

    rTdot = vT;
    vTdot = aT;
    aMdot = (aSat - aM) / p.tauA;

    xdot = [rMdot; vMdot; rTdot; vTdot; aMdot];
end

function aT = target_accel(t)
    t0 = 5.0;
    aMag = 8 * 9.80665;
    Tflip = 2.0;

    if t < t0
        aT = [0; 0; 0];
        return
    end

    s = sign(sin(2*pi*(t - t0)/Tflip));
    aT = [0; aMag*s; 0];
end

function [aCmd, Vc, mode, Nnav, Kapn] = guidance_modes(rRel, vRel, aTHat, p)
    R = norm(rRel);
    rHat = rRel / max(R, 1e-9);

    Vc = -dot(rHat, vRel);
    VcUse = max(Vc, 1.0);
    tgo = R / VcUse;

    if tgo > 8
        mode = 1;
        Nnav = 3.0;
        Kapn = 0.0;
        aCmd = guidance_true_pn(rRel, vRel, Nnav);
    elseif tgo > 3
        mode = 2;
        Nnav = 4.0;
        Kapn = 0.5 * Nnav;
        aCmd = guidance_apn(rRel, vRel, aTHat, Nnav, Kapn);
    else
        mode = 3;
        Nnav = 5.5;
        Kapn = 0.5 * Nnav;
        aCmd = guidance_apn(rRel, vRel, aTHat, Nnav, Kapn);
    end

    if ~all(isfinite(aCmd))
        aCmd = [0; 0; 0];
    end
end

function aCmd = guidance_true_pn(rRel, vRel, Nnav)
    R = norm(rRel);
    Ruse = max(R, 1e-9);

    rHat = rRel / Ruse;
    Vc = -dot(rHat, vRel);

    omegaLos = cross(rRel, vRel) / (Ruse^2);
    aCmd = Nnav * Vc * cross(omegaLos, rHat);
end

function aCmd = guidance_apn(rRel, vRel, aT, Nnav, Kapn)
    aPN = guidance_true_pn(rRel, vRel, Nnav);

    R = norm(rRel);
    rHat = rRel / max(R, 1e-9);

    aTperp = aT - dot(aT, rHat) * rHat;

    aCmd = aPN + Kapn * aTperp;
end

function vOut = saturate_vector(vIn, vmax)
    n = norm(vIn);
    if n < 1e-12
        vOut = vIn;
        return
    end
    if n <= vmax
        vOut = vIn;
        return
    end
    vOut = (vmax / n) * vIn;
end

function [z, ok] = seeker_measurement(rRel, p)
    x = rRel(1);
    y = rRel(2);
    zrel = rRel(3);

    Rxy = sqrt(x^2 + y^2);
    R   = sqrt(x^2 + y^2 + zrel^2);

    az = atan2(y, x);
    el = atan2(zrel, max(Rxy, 1e-12));

    ok = (abs(az) <= p.fovAz) && (abs(el) <= p.fovEl) && (R > 1);

    azm = az + p.biasAz + p.sigmaAz * randn;
    elm = el + p.biasEl + p.sigmaEl * randn;
    Rm  = R  + p.biasR  + p.sigmaR  * randn;

    z = [azm; elm; Rm];
end

function [xHat, P] = ekf_predict_target_cv(xHat, P, dt, p)
    F = [eye(3), dt*eye(3); zeros(3), eye(3)];

    q = p.qAcc;
    Q11 = (dt^3/3) * q * eye(3);
    Q12 = (dt^2/2) * q * eye(3);
    Q22 = dt * q * eye(3);
    Q = [Q11, Q12; Q12, Q22];

    xHat = F * xHat;
    P = F * P * F.' + Q;
```

```
end

function [xHat, P] = ekf_update_target(xHat, P, z, rM, p)
    rT = xHat(1:3);
    vT = xHat(4:6);

    rRel = rT - rM;
    x = rRel(1);
    y = rRel(2);
    zrel = rRel(3);

    Rxy2 = x^2 + y^2;
    Rxy  = sqrt(max(Rxy2, 1e-12));
    R2   = x^2 + y^2 + zrel^2;
    R    = sqrt(max(R2, 1e-12));

    az = atan2(y, x);
    el = atan2(zrel, Rxy);

    h = [az; el; R];

    Hpos = zeros(3,3);

    Hpos(1,1) = -y / max(Rxy2, 1e-12);
    Hpos(1,2) =  x / max(Rxy2, 1e-12);
    Hpos(1,3) =  0;

    denomEl = max(R2, 1e-12);
    Hpos(2,1) = -(x*zrel) / (denomEl*Rxy);
    Hpos(2,2) = -(y*zrel) / (denomEl*Rxy);
    Hpos(2,3) =  Rxy / denomEl;

    Hpos(3,1) = x / R;
    Hpos(3,2) = y / R;
    Hpos(3,3) = zrel / R;

    H = [Hpos, zeros(3,3)];

    ytil = z - h;
    ytil(1) = atan2(sin(ytil(1)), cos(ytil(1)));

    S = H * P * H.' + p.Rmeas;
    K = (P * H.') / S;

    xHat = [rT; vT] + K * ytil;
    P = (eye(6) - K * H) * P;
end
```