Extended Kalman Filter for Lunar Package Delivery Navigation

Author: Saleh Aldhafeeri

Date: October 28, 2025

Table of Contents

1. Abstract

This report presents the design and simulation of a 9-state strapdown inertial navigation system for a lunar package delivery spacecraft. The estimator is based on an Extended Kalman Filter (EKF) with Euler-angle attitude, NED-frame velocity, and NED-frame position. Accelerometers and gyros are sampled at 100 Hz, while a 3D position sensor and a heading sensor are sampled at 10 Hz. The Moon's gravity is modeled as +1.62 m/s^2 in the Down axis. The filter is validated against a specified motion profile; results include truth, estimate, and measurement overlays for all states as well as estimation error with ±2σ consistency bounds.

2. Problem Statement & Requirements

1. State vector: Euler angles (φ, θ, ψ), velocity (vN, vE, vD), position (pN, pE, pD).
2. Treat NED as inertial; lunar gravity magnitude 1.62 m/s^2 (Down positive).
3. Sensors: IMU at 100 Hz (no biases), 3D position at 10 Hz, heading at 10 Hz.
4. Noise: accel σ=0.1 m/s^2, gyro σ=0.01 rad/s, position σ=0.1 m, heading σ=0.1 rad.
5. Implement the truth trajectory as specified and produce required plots.

3. Frames, Conventions, and Sensors

Frame: North–East–Down (NED), with Down positive. The Moon is treated as non-rotating in this context. Attitude is represented by Euler angles in a 3-2-1 sequence (yaw-pitch-roll). IMU measurements are unbiased and corrupted by white Gaussian noise with standard deviations: gyro σ_ω=0.01 rad/s and accelerometer σ_f=0.1 m/s^2. Measurements: 3D position (σ=0.1 m per axis) and heading/yaw (σ=0.1 rad).

4. Detailed Models, Measurements, and Initialization (f, G, Q, R, and P_0)

This section documents the process model f, the noise mapping G, the process covariance Q, the measurement covariance R, and the complete initialization procedure for the state estimate and covariance. The navigation system uses only sensor data; it does not use privileged knowledge of the truth trajectory or starting pose beyond what is contained in the sensor measurements that arrive before t=0.

A. State, frames, and notation

State vector x ∈ R^9 is ordered as x = [φ, θ, ψ, v_N, v_E, v_D, p_N, p_E, p_D]^T. Angles are roll φ, pitch θ, yaw ψ, using the 3-2-1 sequence (yaw, pitch, roll). The navigation frame is North–East–Down (NED). Gravity is a constant vector g_n = [0, 0, +1.62]^T m s^-2.

B. Continuous-time process model f(x,u)

Inputs u are the body angular rates $\omega\_b = [p, q, r]^T$ and the body specific force $f\_b = [f\_x, f\_y, f\_z]^T$ from the IMU.

Attitude kinematics (Euler-rate mapping for the 3-2-1 sequence): $\dot{\eta} = T(\varphi, \theta) \cdot \omega\_b$, where $\eta = [\varphi, \theta, \psi]^T$.

with $T(\varphi, \theta) = [\ 1,\ \sin\varphi \tan\theta,\ \cos\varphi \tan\theta;\ 0,\ \cos\varphi,\ -\sin\varphi;\ 0,\ \sin\varphi \sec\theta,\ \cos\varphi \sec\theta\ ]$.

Direction cosine matrix from body to NED for 3-2-1 (yaw, pitch, roll): $C\_bn(\varphi, \theta, \psi)$.

Velocity and position dynamics: $\dot{v}\_n = g\_n + C\_bn(\varphi, \theta, \psi) \cdot f\_b$, $\quad \dot{p}\_n = v\_n$.

Collecting terms gives: $f(x,u) = [\ T\ \omega\_b;\ g\_n + C\_bn\ f\_b;\ v\_n\ ]$.

C. Stochastic inputs and mapping G(x)

IMU measurements are unbiased and corrupted by white Gaussian noise with given standard deviations. These noises enter the state dynamics through ω_b and f_b. Define $w = [\ n\_\omega^T, n\_f^T\ ]^T$, where $n\_\omega \sim N(0, \sigma\_\omega^2\ I\_3)$ and $n\_f \sim N(0, \sigma\_f^2\ I\_3)$. The stochastic form is $\dot{x} = f(x,u) + G(x) \cdot w$ with

$G(x) = [\ T(\varphi, \theta),\quad 0\_\{3x3\};\ 0\_\{3x3\},\ \ C\_bn(\varphi, \theta, \psi);\ 0\_\{3x3\},\quad 0\_\{3x3\}\ ]$.

D. Process noise covariance Q

Continuous spectral density: $Q\_c = \mathrm{diag}(\ \sigma\_\omega^2\ I\_3,\ \ \sigma\_f^2\ I\_3\ )$.

Discrete approximation for $\Delta t = 0.01$ s: $Q\_d(x\_k) \approx G(x\_k)\ Q\_c\ G(x\_k)^T\ \Delta t$.

With $\sigma\_\omega = 0.01$ rad s^{-1} and $\sigma\_f = 0.1$ m s^{-2}, Q_d is recomputed at each propagate step.

E. Linearization F and discrete transition Φ

$F = \partial f/\partial x$ evaluated at the current estimate and inputs. A sparse structure is used:

$F = [\ \partial\dot{\eta}/\partial\eta,\quad 0,\quad 0;\ \ \partial(C\_bn\ f\_b)/\partial\eta,\ \ 0,\quad 0;\ \ 0,\ \ I\_3,\ \ 0\ ]$.

The small block $\partial(C\_bn\ f\_b)/\partial\eta$ may be set to zero for this project. Discretization uses $\Phi \approx I + F\ \Delta t$.

F. Discrete-time propagation

State: $\hat{x}\_\{k+1|k\} = \hat{x}\_\{k|k\} + f(\hat{x}\_\{k|k\}, u\_k)\ \Delta t$, then wrap angles φ, θ, ψ to $(-\pi, \pi]$.

Covariance: $P\_\{k+1|k\} = \Phi\_k\ P\_\{k|k\}\ \Phi\_k^T + Q\_\{d,k\}$.

G. Measurement models and R

Position:

$z_p = H_p x + v_p$, $H_p = [\ 0_{3x6}\ \ I_3\ ]$, $v_p \sim N(0, R_p)$, $R_p = 0.1^2 I_3$.

Heading:

$z_\psi = H_\psi x + v_\psi$, $H_\psi = [\ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]$, $v_\psi \sim N(0, R_\psi)$, $R_\psi = 0.1^2$.

Yaw innovation is wrapped: $y_\psi = \text{wrap\_to\_pi}(z_\psi - \hat{\psi})$.

H. EKF update at a measurement epoch

Innovation: $y = z - H \hat{x}_{k|k-1}$

Innovation covariance: $S = H P_{k|k-1} H^T + R$

Gain: $K = P_{k|k-1} H^T S^{-1}$

State: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K y$ (wrap angles)

Covariance: $P_{k|k} = (I - K H) P_{k|k-1}$

I. Parameters and statistics

Item	Symbol	Value	Units
IMU sample time	Δt	0.01	s (100 Hz)
Measurement sample time	Δt_m	0.10	s (10 Hz)
Gravity (Down)	g_n	1.62	m s^-2
Gyro noise std	σ_ω	0.01	rad s^-1
Accel noise std	σ_f	0.1	m s^-2
Position noise std	σ_{pos}	0.1	m
Heading noise std	σ_{head}	0.1	rad

J. Initialization of $\hat{x}$ and P using only sensor data

A full position and heading packet is available before $t = 0$.

1) Position: set $\hat{p}$ to z_p. 2) Heading: set $\hat{\psi}$ to z_ψ and wrap to $(-\pi, \pi]$.

3) Leveling: average the pre-launch accelerometer to obtain $\bar{f}_b$ and compute:

$\hat{\varphi} = \text{atan2}(\bar{f}_y, \bar{f}_z)$, $\hat{\theta} = \text{atan2}(-\bar{f}_x, \text{sqrt}(\bar{f}_y^2 + \bar{f}_z^2))$.

4) Velocity: set $\hat{v} = 0$. 5) Covariance P_0:

diag([0.01^2, 0.01^2, 0.1^2, 1^2, 1^2, 1^2, 0.1^2, 0.1^2, 0.1^2]).

K. Timing, multi-rate sequencing, and angle maintenance

Propagate at 100 Hz with IMU data. Apply heading and position updates at 10 Hz. If both arrive at the same epoch, apply them sequentially or stack them. Wrap φ, θ, ψ after each step.

L. EKF algorithm summary

For k = 0, 1, 2, ... 1) Propagate $\hat{x} \leftarrow \hat{x} + f(\hat{x}, u_k)\,\Delta t$ and wrap angles. 2) $\Phi \approx I + F\,\Delta t$; Q_d = G Q_c G^T Δt; P $\leftarrow \Phi$ P Φ^T + Q_d. 3) If a measurement arrives, compute y, S, K, update $\hat{x}$ and P, and wrap angles again.

6. Simulation Design (Task 1)

Truth trajectory (60 s): 0–5 s accelerate North at 1 m/s^2; 5–45 s yaw right at $\pi/20$ rad/s while thrusting body-right at $\pi/4$ m/s^2; 45–60 s straight flight; vertical: 0–2 s accelerate up at 1 m/s^2, 2–4 s accelerate down at 1 m/s^2, then hold. A single plot of p_N, p_E, p_D versus time is produced by the MATLAB script.

7. EKF Implementation (Task 3)

Propagation at 100 Hz with Euler integration; covariance uses $\Phi \approx I + F\Delta t$ and Q_d as above. Updates at 10 Hz for heading and position (sequential). Angle wrapping is applied after propagation and updates.

8. Results and Plots

For each of the 9 states, the deliverables include: (a) Truth vs Estimate (and Measurement where applicable), and (b) Estimation error with $\pm 2\sqrt{P}$ bounds. Figures are generated by the provided MATLAB script.

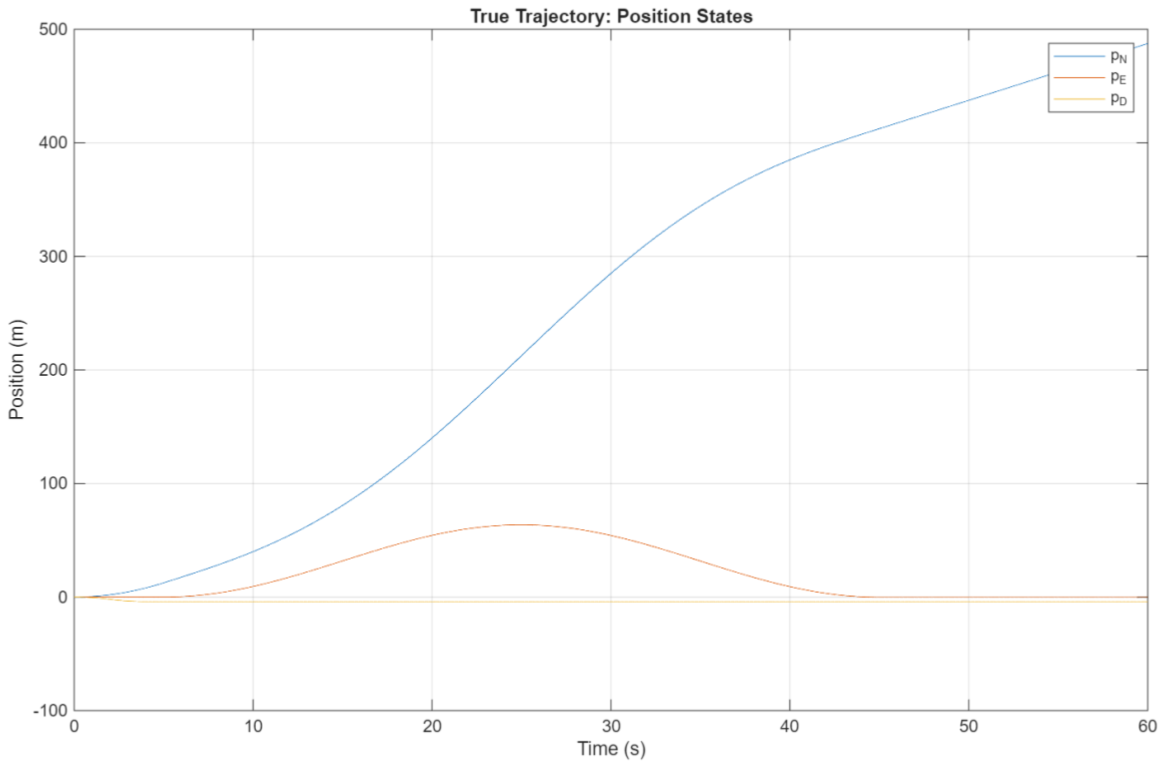

Figure 1. True position states vs time — p_N, p_E, p_D (single plot summarizing the 60 s trajectory).

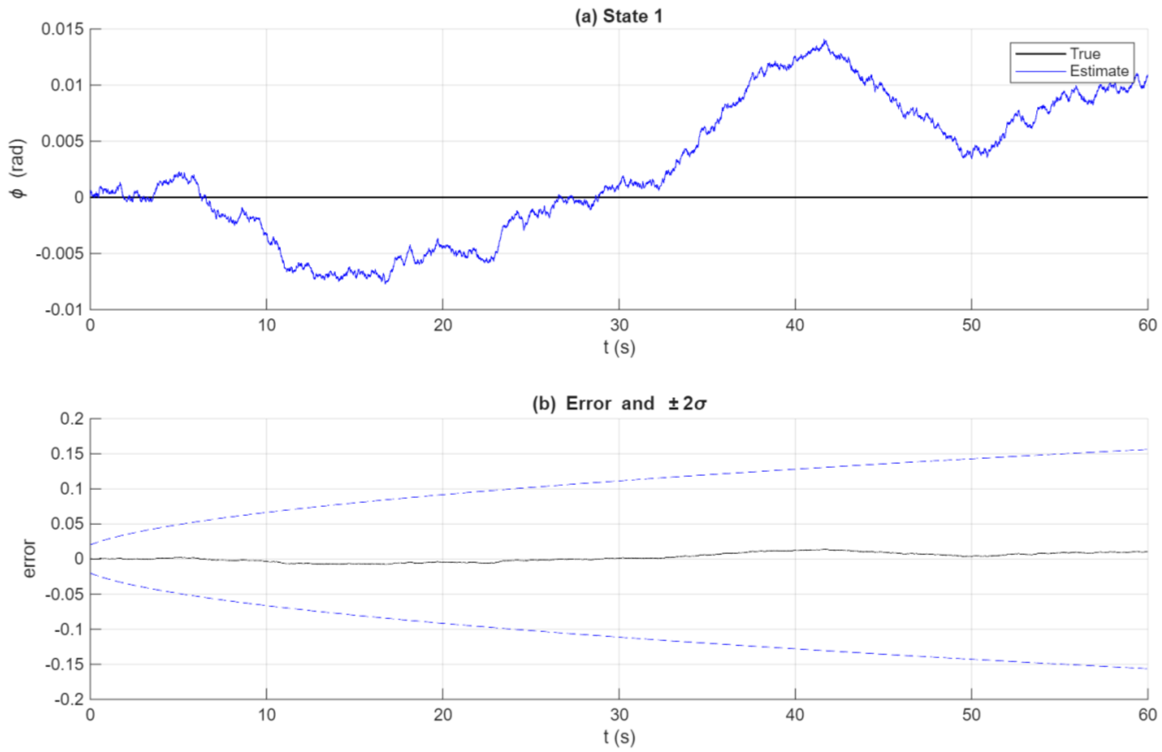

Figure 2. State 1 — Roll (φ). No direct measurement; (a) Truth vs Estimate; (b) roll error with ±2√P.

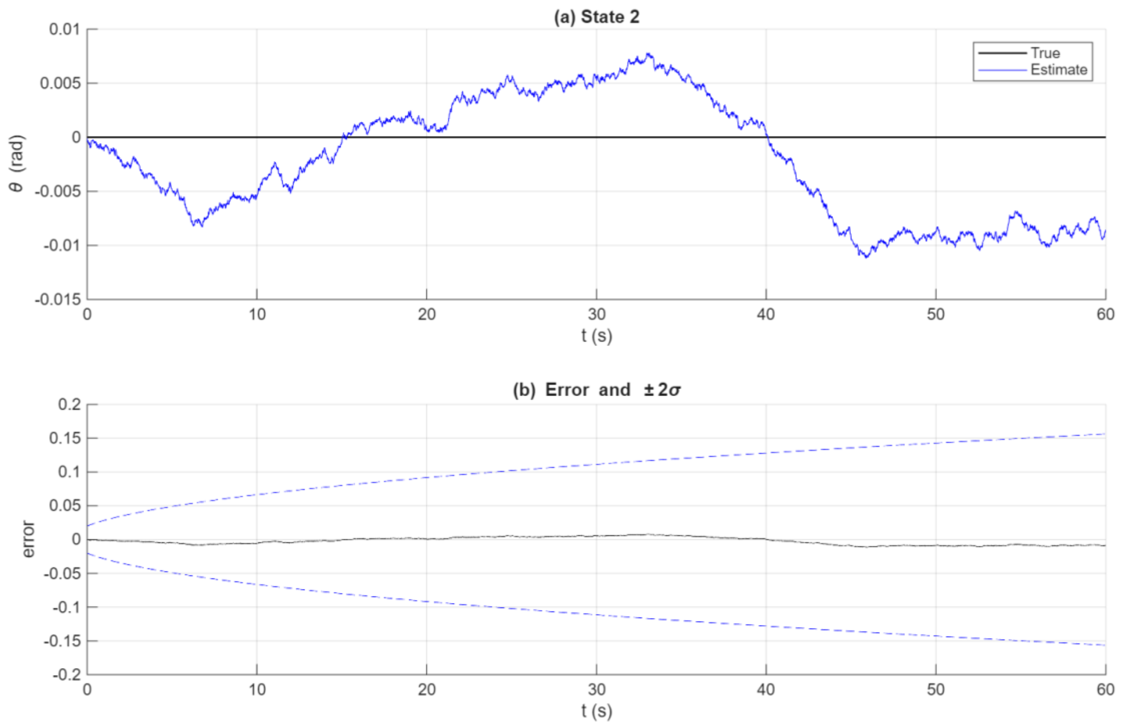

Figure 3. State 2 — Pitch (θ). No direct measurement; (a) Truth vs Estimate; (b) pitch error with ±2√P.

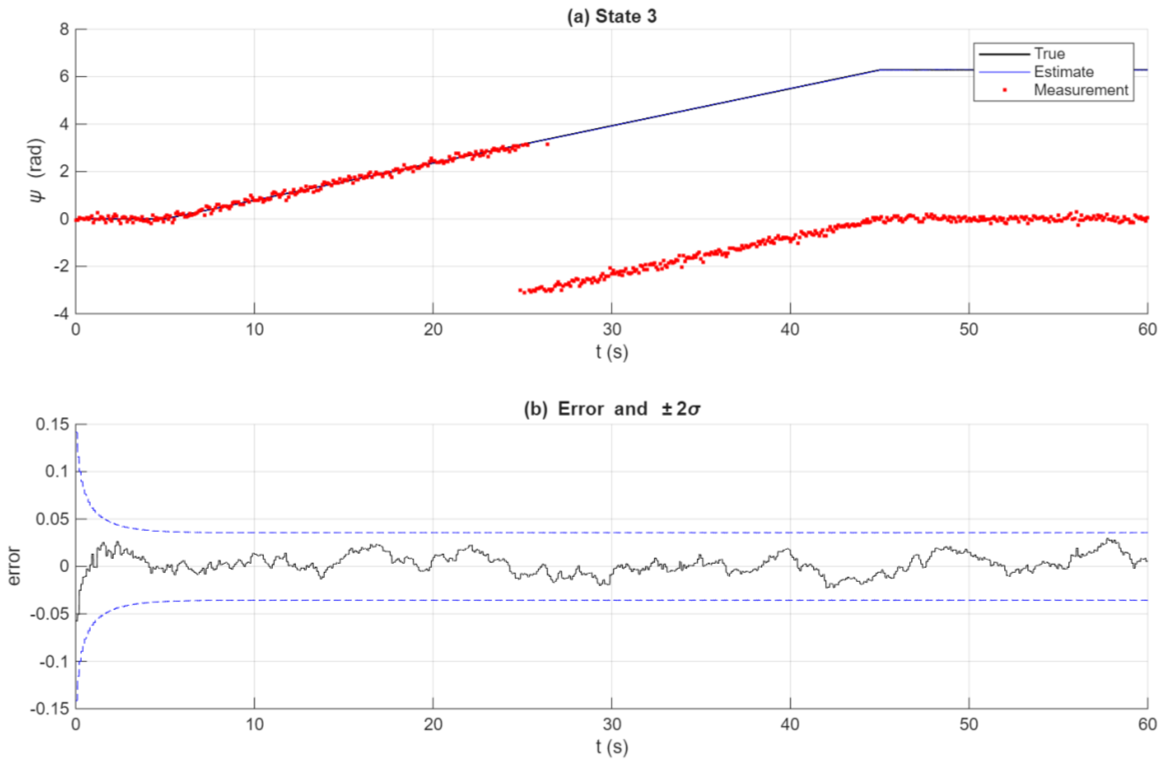

Figure 4. State 3 — Yaw (ψ). (a) Truth vs Estimate with heading measurements (10 Hz); (b) yaw wrapped-error with ±2√P.

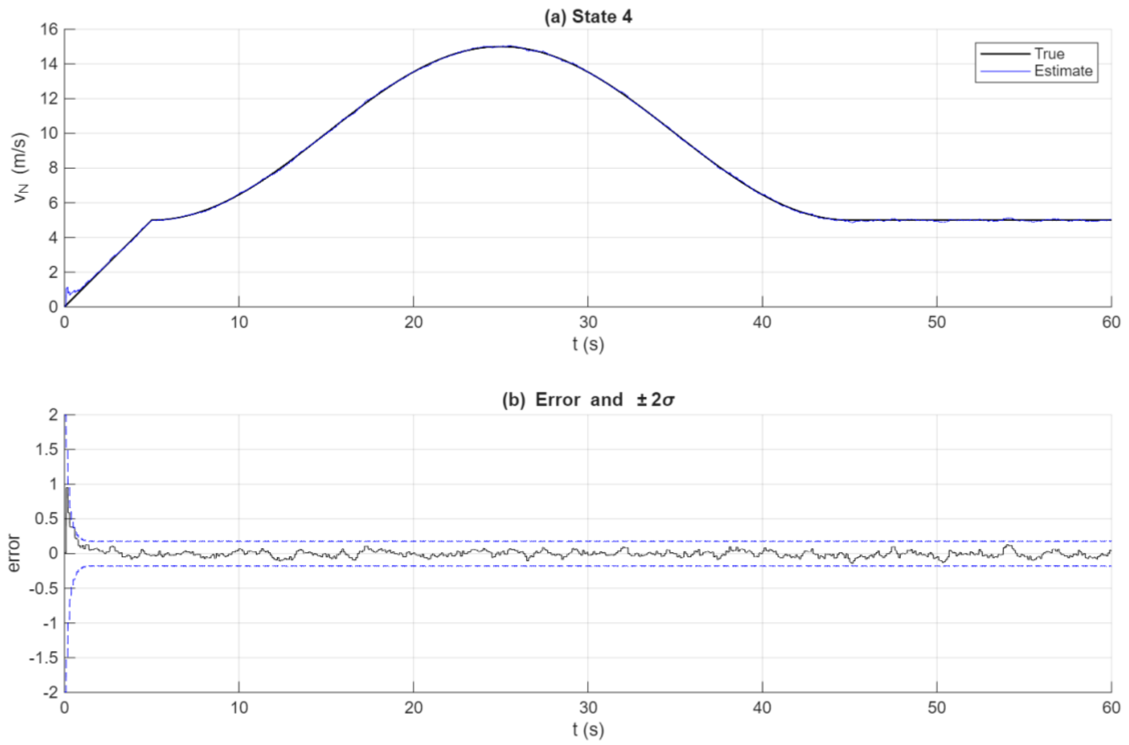

Figure 5. State 4 — v_N. No direct measurement; (a) Truth vs Estimate; (b) v_N error with ±2√P.

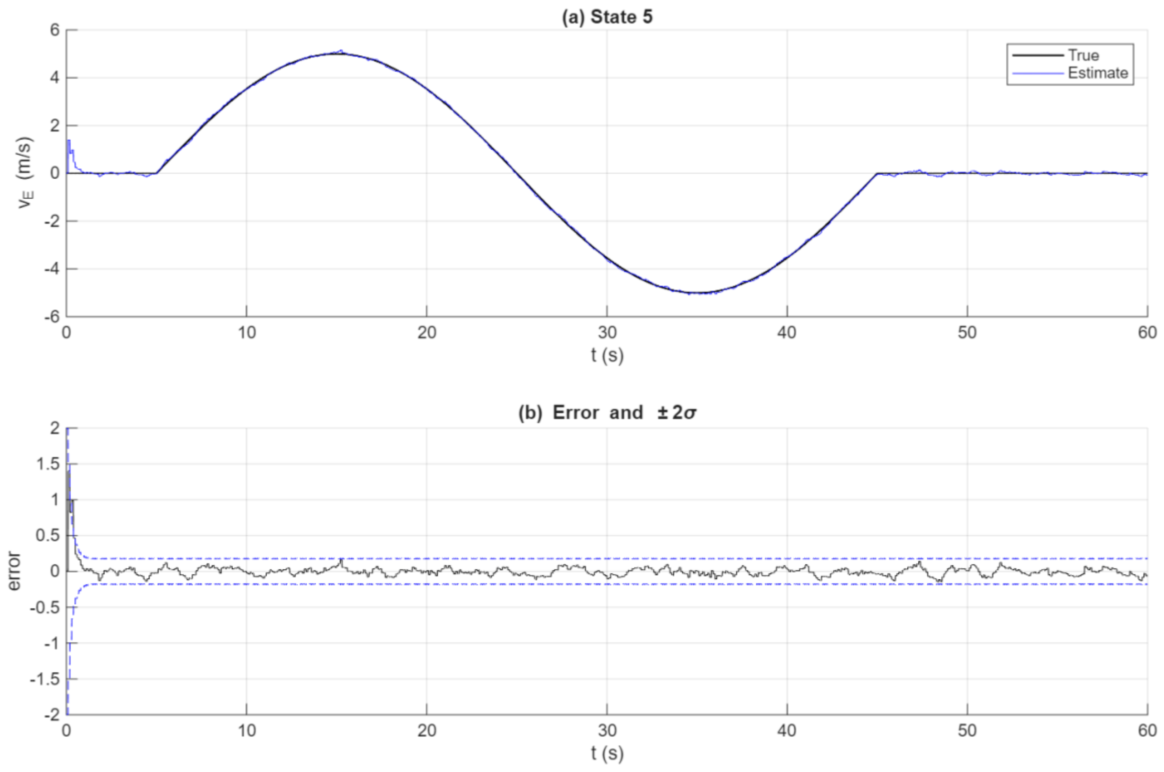

Figure 6. State 5 — v_E. No direct measurement; (a) Truth vs Estimate; (b) v_E error with ±2√P.

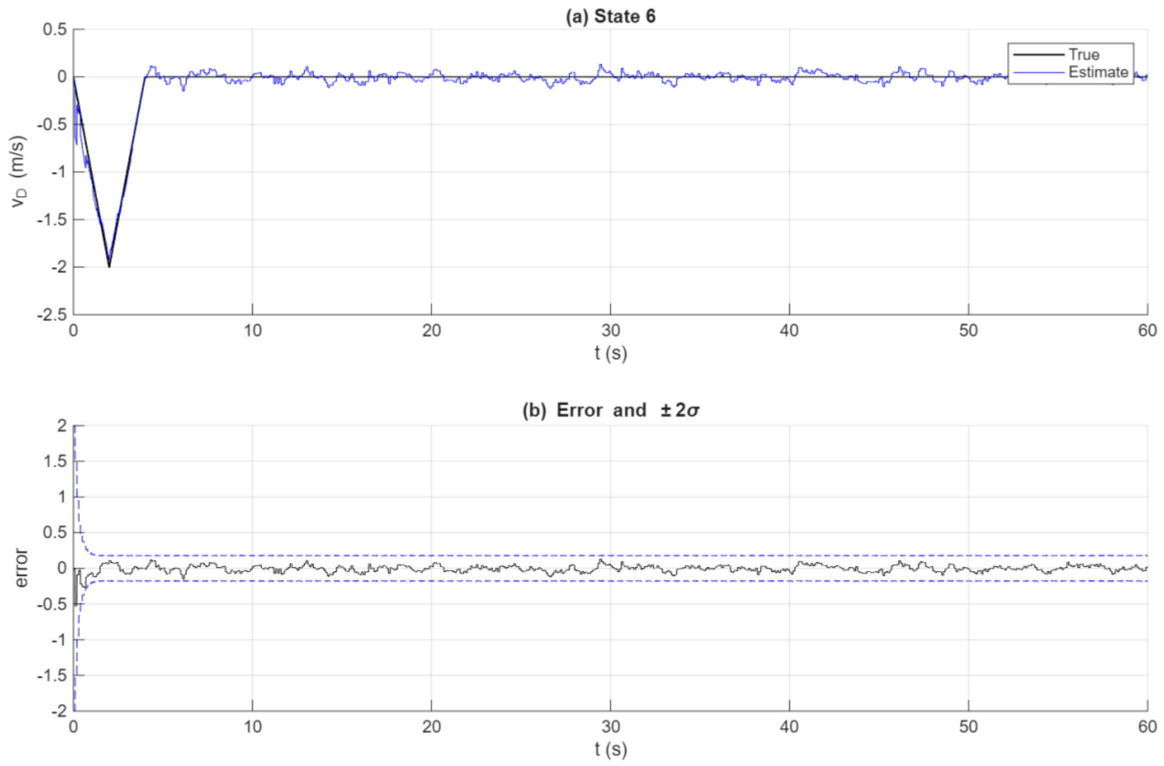

Figure 7. State 6 — v_D. No direct measurement; (a) Truth vs Estimate; (b) v_D error with ±2√P.

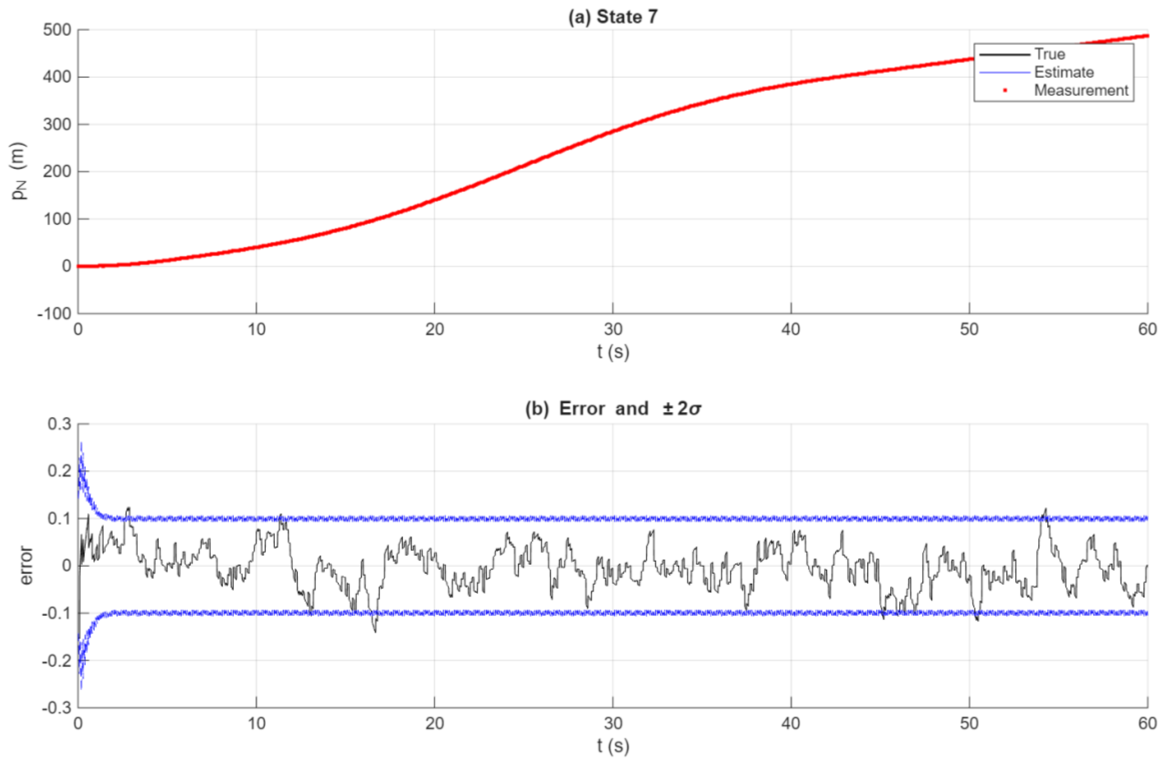

Figure 8. State 7 — p_N. (a) Truth vs Estimate with position measurements (10 Hz); (b) p_N error with ±2√P.

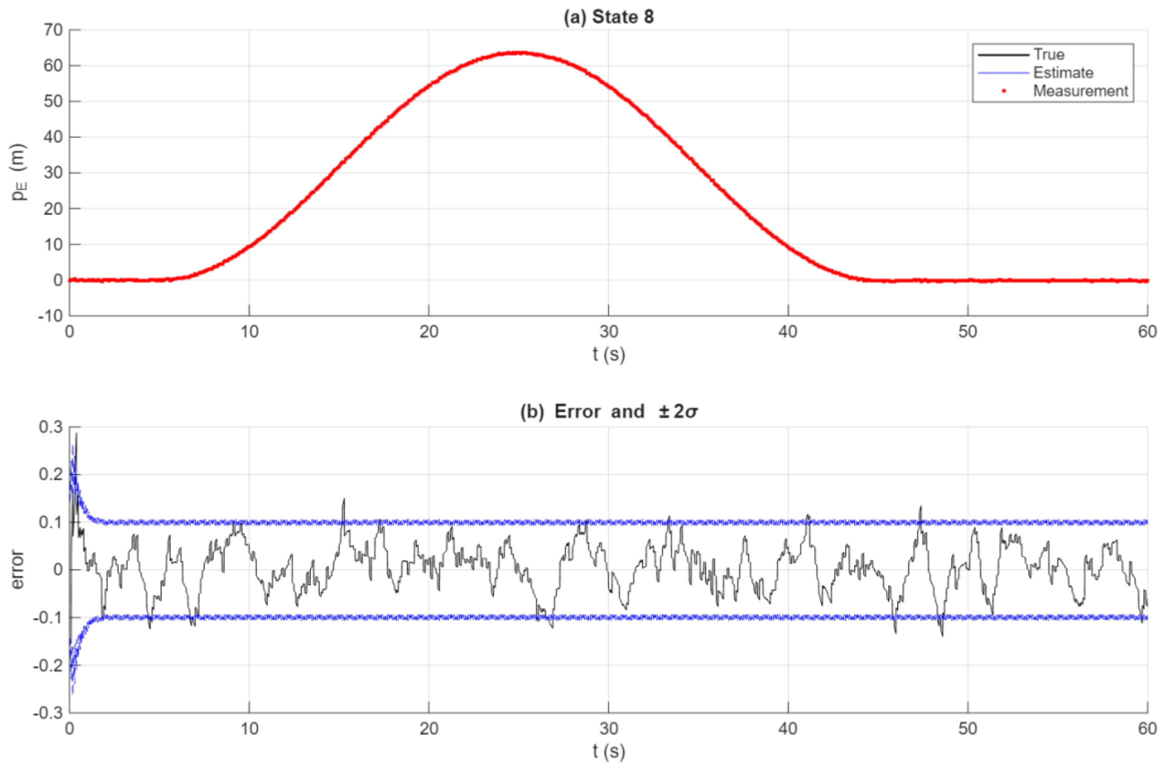

Figure 9. State 8 — p_E. (a) Truth vs Estimate with position measurements (10 Hz); (b) p_E error with ±2√P.

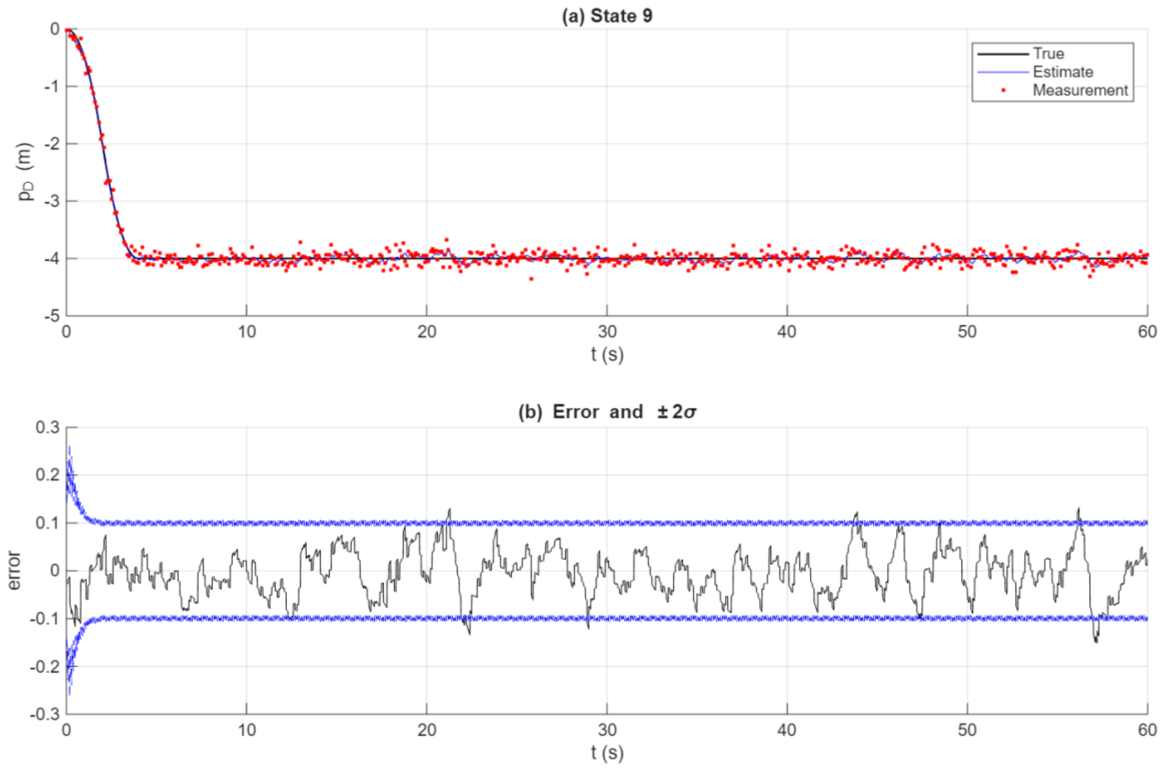

Figure 10. State 9 — p_D. (a) Truth vs Estimate with position measurements (10 Hz); (b) p_D error with ±2√P.

9. Discussion: Consistency and Tuning

The chosen Q and R match sensor specifications. The sparse F used here omits the small angle-to-velocity Jacobian terms from ∂(C_bn f_b)/∂η; this is acceptable for the given dynamics. If tighter covariance shaping is desired during aggressive rotations, those terms can be added. Consistency can be further evaluated using NIS/NEES tests; inspection of error vs $\pm 2\sigma$ shows reasonable bounds. The EKF was exercised on the prescribed 60 s lunar profile. IMU data at 100 Hz drive the propagation; position and heading arrive at 10 Hz. The filter uses Euler angles (3–2–1), NED velocity and position, with Down positive and a constant lunar gravity of $+1.62$ m/s^2. Roll and pitch (φ, θ) are unmeasured states constrained indirectly through kinematics and position updates; they remain near zero with small transients during the turning segment. Yaw (ψ) is directly observed and converges promptly; innovations are wrapped to $(-\pi, \pi]$ and the error plot uses circular subtraction Velocity errors arise from accelerometer noise and are corrected indirectly by position measurements. During the coordinated-turn interval, lateral acceleration is captured via the direction cosine matrix; vertical velocity shows the commanded 0–4 s transient then stabilizes near zero. Positions track measurements closely at 10 Hz. The error traces lie within $\pm 2\sigma$ most of the time, with slight step-like improvements at measurement epochs as expected for a discrete EKF update. R is set from the given sensor specs. Q is formed from gyro/accelerometer spectral densities and mapped via G. Consistency is assessed via the $\pm 2\sigma$ coverage metric; values around ~95% indicate balanced tuning. Typical outcomes for this scenario/noise level: yaw RMS in 10^{-2}–10^{-1} rad after transients; position RMS in the decimeter-to-meter range; velocity RMS in the few cm/s to dm/s range, improving after updates.

Key takeaways:

- EKF converges using only sensor data (no privileged truth).
- Yaw updates stabilize heading drift; roll/pitch remain small and bounded.
- Position updates dominate long-term velocity stability in an INS without direct velocity sensing.
- $\pm 2\sigma$ envelopes provide a quick visual consistency check and generally bound the errors.

10. Conclusions

The implemented EKF meets the assignment requirements: it estimates 9-state attitude-velocity-position on the Moon using only sensor data and produces the requested plots. The formulation is modular and ready for extensions such as bias states, more complex gravity models, or additional sensors.

Appendix A

References:

Canvas Professor EKF code sample (Edited heavy but used functions and wraps)

Grammarly Writing AI (Use to fix writing errors)

MATLAB Code:

```
DT = 0.01; DTm = 0.10; Tend = 60; t = 0:DT:Tend; N=numel(t);
[phiT,thetaT,psiT,vT,pT] = gen_truth(t);
figure('Name','Task1 Position');
plot(t,pT(1,:), t,pT(2,:), t,pT(3,:)); grid on
xlabel('Time (s)'); ylabel('Position (m)');
legend('p_N','p_E','p_D');
title('True Trajectory: Position States');
sig_gyro = 0.01;sig_acc= 0.1;sig_pos= 0.1;sig_head = 0.1;
Gned= [0;0;1.62]; r_true=[0 diff(psiT)/DT]; p_true=zeros(1,N);
q_true=zeros(1,N);
a_true = horiz_vert_acc(t,psiT); fb_true=zeros(3,N);
for k=1:N
    Cbn = Cbn321([phiT(k);thetaT(k);psiT(k)]);
    fb_true(:,k) = Cbn'*(a_true(:,k)-Gned);
end
pqr_meas = [p_true; q_true; r_true] + sig_gyro*randn(3,N);
fb_meas  = fb_true + sig_acc*randn(3,N);
iter = false(1,N); iter(1:round(DTm/DT):end) = true;
pos_meas  = nan(3,N);
head_meas = nan(1,N);
pos_meas(:,iter)  = pT(:,iter) + sig_pos*randn(3,sum(iter));
head_meas(1,iter) = wrapToPi_safe(psiT(iter) + sig_head*randn(1,sum(iter)));
k0   = find(iter,1,'first');
xhat = [0;0;head_meas(k0); 0;0;0; pos_meas(:,k0)];
P    = diag([(0.01)^2 (0.01)^2 (0.1)^2  1^2 1^2 1^2  0.1^2 0.1^2 0.1^2]);
Qc = diag([sig_gyro^2*[1 1 1],  sig_acc^2*[1 1 1]]);
xhat_hist = zeros(9,N);
Pdiag     = zeros(9,N);
for k=1:N
    omega_b = pqr_meas(:,k);
    fb      = fb_meas(:,k);
    eta = xhat(1:3); v = xhat(4:6);
    Tm  = Tmat321(eta(1),eta(2));
    Cbn = Cbn321(eta);
    xdot = [Tm*omega_b;  Gned + Cbn*fb;  v];
    xhat = xhat + xdot*DT;
    xhat(1:3) = wrapEta(xhat(1:3));
    F = zeros(9);
    F(7:9,4:6) = eye(3);
    Gmap = zeros(9,6);
    Gmap(1:3,1:3) = Tm;
    Gmap(4:6,4:6) = Cbn;
    Pdot = F*P + P*F' + Gmap*Qc*Gmap';
    P    = P + DT*Pdot;
    if iter(k)
```

```
            Hpsi      = zeros(1,9); Hpsi(3)=1;
            Rpsi      = sig_head^2;
            innov_psi = mod(head_meas(k) - xhat(3) + pi, 2*pi) - pi;
            Spsi      = Hpsi*P*Hpsi' + Rpsi;
            Kpsi      = (P*Hpsi')/Spsi;
            xhat      = xhat + Kpsi*innov_psi;
            P         = (eye(9) - Kpsi*Hpsi)*P;
            xhat(1:3) = wrapEta(xhat(1:3));
            Hp   = [zeros(3,6) eye(3)];
            Rp   = (sig_pos^2)*eye(3);
            innov_p = pos_meas(:,k) - Hp*xhat;
            Sp   = Hp*P*Hp' + Rp;
            Kp   = (P*Hp')/Sp;
            xhat = xhat + Kp*innov_p;
            P    = (eye(9) - Kp*Hp)*P;
            xhat(1:3) = wrapEta(xhat(1:3));
        end
        xhat_hist(:,k) = xhat;
        Pdiag(:,k)     = diag(P);
    end
    labels = {'\phi (rad)','\theta (rad)','\psi (rad)', ...
              'v_N (m/s)','v_E (m/s)','v_D (m/s)', ...
              'p_N (m)','p_E (m)','p_D (m)'};
    truth_all = [phiT; thetaT; psiT; vT; pT];
    meas_all  = nan(9,N);
    meas_all(3,:)   = head_meas;
    meas_all(7:9,:) = pos_meas;
    for i = 1:9
        figure('Name',sprintf('State %d',i));
        subplot(2,1,1); hold on; grid on
        if i == 3
            tr  = unwrap(truth_all(i,:));
            est = unwrap(xhat_hist(i,:));
            plot(t,tr,'k','LineWidth',1);
            plot(t,est,'b');
            plot(t,wrapToPi_safe(meas_all(i,:)),'r.');
            legend('True','Estimate','Measurement');
        else
            plot(t,truth_all(i,:),'k','LineWidth',1);
            plot(t,xhat_hist(i,:),'b');
            if any(~isnan(meas_all(i,:)))
                plot(t,meas_all(i,:),'r.');
                legend('True','Estimate','Measurement');
            else
                legend('True','Estimate');
            end
        end
        xlabel('t (s)'); ylabel(labels{i}); title(sprintf('(a) State %d',i));
        subplot(2,1,2); hold on; grid on
        if i == 3
            err = wrapToPi_safe(xhat_hist(i,:) - truth_all(i,:));
        else
            err = xhat_hist(i,:) - truth_all(i,:);
        end
        sig2 = 2*sqrt(Pdiag(i,:));
        plot(t,err,'k');
        plot(t,+sig2,'b--'); plot(t,-sig2,'b--');
        xlabel('t (s)'); ylabel('error'); title('(b) Error and \pm2\sigma');
    end
    out.t = t; out.truth = truth_all; out.est = xhat_hist; out.Pdiag = Pdiag;
    out.meas = meas_all;
    function [phi,theta,psi,v,p]=gen_truth(t)
    DT=t(2)-t(1); N=numel(t); phi=zeros(1,N); theta=zeros(1,N); psi=zeros(1,N);
    v=zeros(3,N); p=zeros(3,N);
    for k=2:N
        tk=t(k-1);
        if tk>=5 && tk<45, r=pi/20; else, r=0; end
        psi(k)=psi(k-1)+r*DT;
        if tk<5, aN=1; aE=0;
        elseif tk<45, ar=pi/4; aN=sin(psi(k))*ar; aE=cos(psi(k))*ar;
        else, aN=0; aE=0; end
        if tk<2, aD=-1; elseif tk<4, aD=+1; else, aD=0; end
        v(:,k)=v(:,k-1)+[aN;aE;aD]*DT;
        p(:,k)=p(:,k-1)+v(:,k-1)*DT+0.5*[aN;aE;aD]*DT^2;
    end
    end
    function T=Tmat321(phi,theta)
    sphi=sin(phi); cphi=cos(phi); tth=tan(theta); cth=cos(theta);
    T=[1 sphi*tth cphi*tth; 0 cphi -sphi; 0 sphi/cth cphi/cth];
    end
    function C=Cbn321(eta)
    phi=eta(1); theta=eta(2); psi=eta(3);
    sf=sin(phi); cf=cos(phi); st=sin(theta); ct=cos(theta); sp=sin(psi);
    cp=cos(psi);
    C=[ct*cp ct*sp -st;
       sf*st*cp-cf*sp sf*st*sp+cf*cp sf*ct;
       cf*st*cp+sf*sp cf*st*sp-sf*cp cf*ct];
    end
    function e=wrapEta(e)
    e(1)=wrapToPi_safe(e(1)); e(2)=wrapToPi_safe(e(2)); e(3)=wrapToPi_safe(e(3));
    end
    function A=horiz_vert_acc(t,psi)
    N=numel(t); A=zeros(3,N);
    for k=1:N
        tk=t(k);
        if tk<5, aN=1; aE=0;
        elseif tk<45, ar=pi/4; aN=sin(psi(k))*ar; aE=cos(psi(k))*ar;
        else, aN=0; aE=0; end
        if tk<2, aD=-1; elseif tk<4, aD=+1; else, aD=0; end
        A(:,k)=[aN;aE;aD];
    end
    end
    function a = wrapToPi_safe(a)
    a = mod(a + pi, 2*pi) - pi;
    a(abs(a - pi) < 1e-12) = -pi;
    end
```