# Theory and Architecture of a Modular Radar Simulation Framework

bboyg

January 14, 2026

## Contents

# 1 Introduction

This document describes the physical models, coordinate systems, signal processing, and software architecture used in a modular monostatic radar simulation. The simulation models:

- Radar geometry and antenna beam patterns,
- Target kinematics and attitude,
- Radar cross section (RCS) variation with aspect,
- Detection via the radar equation,
- Tracking via a Kalman filter.

Each theoretical concept is mapped directly to its implementation in the code.

# 2 Coordinate Systems

Three coordinate frames are used:

## 2.1 Earth-Centered Earth-Fixed (ECEF)

A Cartesian frame fixed to the Earth, used for absolute positions:

$$\mathbf{r}_{\text{ECEF}} = (x, y, z).$$

Conversion from latitude $\phi$, longitude $\lambda$, altitude $h$ uses the WGS-84 ellipsoid.

## 2.2 Local East-North-Up (ENU)

A tangent plane at the radar location:

$$\mathbf{v}_{\text{ENU}} = [e, n, u]^T.$$

This is the radar's working frame.

## 2.3 Body Frame

Attached to the target, rotating with it. Orientation is represented by a rotation matrix:

$$\mathbf{R}_{\text{body}\to\text{ENU}}.$$

# 3 Trajectory and Attitude Propagation

This chapter describes how target motion and orientation are modeled in the simulation. The trajectory model separates *translational motion* (position vs time) from *rotational motion* (attitude vs time). Translational motion is prescribed directly by the scenario, while rotational motion is propagated by integrating angular rates.

### 3.1 Trajectory Representation

A trajectory is defined as discrete time samples:

$$\{t_k,\ \mathbf{p}_k,\ \boldsymbol{\omega}_k\}_{k=0}^{N-1},$$

where:

- $t_k$ is time in seconds,
- $\mathbf{p}_k = (\text{lat}_k,\ \text{lon}_k,\ h_k)$ is position in latitude, longitude, altitude (km),
- $\boldsymbol{\omega}_k = (p_k, q_k, r_k)$ are body-frame angular rates in rad/s.

The code stores:

- `self.t` as a 1D array of times,
- `self.pos` as an $N \times 3$ array of LLA samples,
- `self.roll` as an $N \times 3$ array of angular rates.

### 3.2 Translational Motion

The simulation does not integrate translational dynamics (no forces or accelerations). Instead, $\mathbf{p}_k$ is defined explicitly by the scenario generator (e.g. ballistic arc, linear motion, circular arc).

Thus:

$$\mathbf{p}(t) = \text{user-defined}.$$

This allows complete freedom in specifying motion (ballistic, orbital, straight-line, etc.) without coupling to the attitude model.

### 3.3 Attitude Representation

Orientation is represented by a rotation matrix:

$$\mathbf{R}_{b\to e} \in SO(3),$$

mapping vectors from the body frame to the ENU frame.

The matrix satisfies:

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}, \quad \det(\mathbf{R}) = +1.$$

### 3.4 Angular Rate Kinematics

The body-frame angular rate vector is:

$$\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}.$$

The continuous-time kinematic equation for rotation is:

$$\dot{\mathbf{R}} = \mathbf{R}[\boldsymbol{\omega}]_\times,$$

where $[\boldsymbol{\omega}]_\times$ is the skew-symmetric cross-product matrix:

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}.$$

### 3.5 Discrete-Time Integration

Over a timestep $\Delta t = t_{k+1} - t_k$, assuming $\boldsymbol{\omega}$ is constant over the interval, the exact solution is:

$$\mathbf{R}_{k+1} = \mathbf{R}_k \exp\left([\boldsymbol{\omega}_k]_\times \Delta t\right).$$

This is implemented using Rodrigues' formula.

### 3.6 Rodrigues' Rotation Formula

Let $\theta = \|\boldsymbol{\omega}\|\Delta t$ and $\hat{\mathbf{u}} = \boldsymbol{\omega}/\|\boldsymbol{\omega}\|$. Then:

$$\exp([\boldsymbol{\omega}]_\times \Delta t) = \mathbf{I} + \sin\theta[\hat{\mathbf{u}}]_\times + (1 - \cos\theta)[\hat{\mathbf{u}}]_\times^2.$$

This yields a proper rotation matrix for any rotation axis and angle.
In code:

- $\theta$ is computed from $\boldsymbol{\omega}$ and $\Delta t$,
- the skew matrix is formed,
- Rodrigues' formula produces the incremental rotation,
- it is post-multiplied with the current orientation.

### 3.7 Initial Orientation

The user supplies an initial orientation:

$$\mathbf{R}_0 = \mathbf{R}_{b\to e}(t_0).$$

Often this is chosen as identity:

$$\mathbf{R}_0 = \mathbf{I},$$

meaning body axes are initially aligned with ENU.

### 3.8 Numerical Properties

#### 3.8.1 Orthogonality Preservation

Rodrigues' formula preserves orthogonality analytically. However, floating-point error can accumulate.

The code optionally normalizes or re-orthogonalizes matrices if needed (or this can be added as a post-step).

#### 3.8.2 Small-Angle Behavior

When $\|\boldsymbol{\omega}\|\Delta t$ is very small, the formula reduces smoothly to:

$$\exp([\boldsymbol{\omega}]_\times \Delta t) \approx \mathbf{I} + [\boldsymbol{\omega}]_\times \Delta t.$$

This avoids numerical instability.

### 3.9 Relationship to RCS Modeling

The body-to-ENU rotation $\mathbf{R}_{b\to e}$ is used to express the radar line-of-sight in the body frame:

$$\hat{\mathbf{u}}_b = \mathbf{R}_{b\to e}^T \hat{\mathbf{u}}_e.$$

From $\hat{\mathbf{u}}_b$, the aspect angles $(\phi, \theta)$ are computed and passed into the RCS model:

$$\sigma = \sigma(\phi, \theta, t).$$

Thus:

$$\text{trajectory} \to \text{attitude} \to \text{aspect} \to \text{RCS}.$$

### 3.10 Special Case: Planar "Flipping" Target

For the flipping-disc test:

$$\boldsymbol{\omega} = (\omega, 0, 0), \quad \omega = \frac{\pi}{2} \text{ rad/s}.$$

This yields a 90° rotation per second about the body $x$-axis, cycling the target through:

$$\text{front} \to \text{edge} \to \text{back} \to \text{edge} \to \text{front}.$$

This directly drives the observed RCS modulation.

### 3.11 Assumptions and Limitations

- No translational dynamics (no gravity, drag, thrust).
- Angular rates are prescribed, not dynamically computed.
- No coupling between translation and rotation.
- No modeling of nutation, precession, or torque dynamics.

### 3.12 Possible Extensions

- Add rigid-body dynamics with inertia and applied torques.
- Integrate attitude using quaternions for improved numerical robustness.
- Include aerodynamic or gravitational torques.
- Couple translation and rotation (e.g., lift-induced rotation).

### 3.13 Summary

The trajectory module provides:

- Arbitrary translational motion,
- Physically correct rotational kinematics,
- Clean separation between motion and sensing,
- A direct pipeline into RCS and detection physics.

It is the foundation on which the entire radar observability chain is built.

# 4 Scan Patterns and Beam Pointing Control

This chapter describes how the radar beam is steered in azimuth and elevation as a function of time. Scan patterns determine where the radar looks, how long it dwells, and how often a given direction is revisited. They are a critical link between sensing physics and track quality.

## 4.1 Conceptual Role of a Scan Pattern

A scan pattern defines a time-parametrized sequence:

$$(\alpha_c(t),\ \epsilon_c(t),\ \Delta t(t)),$$

where:

- $\alpha_c$ is the commanded azimuth,
- $\epsilon_c$ is the commanded elevation,
- $\Delta t$ is the dwell time at that pointing.

The radar processes one dwell per scan step, after which the scan advances to the next pointing.

## 4.2 Software Interface

All scan patterns implement a common interface:

$$\texttt{next\_pointing}(t) \to (\alpha, \epsilon, \text{dwell}).$$

This makes scan patterns interchangeable and decouples radar physics from scheduling logic.

## 4.3 Fixed Pointing (Stare)

### 4.3.1 Definition

A fixed pointing scan maintains a constant beam direction:

$$\alpha_c(t) = \alpha_0, \quad \epsilon_c(t) = \epsilon_0.$$

### 4.3.2 Purpose

Used for:

- track refinement,
- stationary experiments,
- controlled tests (e.g., flipping target).

### 4.3.3 Implementation

The code simply returns the same angles each call.

## 4.4 Circular Scan

### 4.4.1 Definition

The beam rotates continuously in azimuth at a fixed elevation:

$$\alpha_c(t) = \alpha_0 + \omega t, \quad \epsilon_c(t) = \epsilon_0.$$

### 4.4.2 Purpose

Used for:

- horizon surveillance,
- detecting targets uniformly around the radar.

## 4.5 Sector Scan

### 4.5.1 Definition

The beam sweeps back and forth between limits:

$$\alpha_{\min} \leq \alpha_c(t) \leq \alpha_{\max},$$

with a triangular-wave temporal profile.

### 4.5.2 Purpose

Used when interest is confined to a particular angular region.

## 4.6 Raster Scan

### 4.6.1 Definition

Raster scanning covers a 2D region by sweeping azimuth lines at successive elevations:

$$\alpha \in [\alpha_{\min}, \alpha_{\max}], \quad \epsilon \in [\epsilon_{\min}, \epsilon_{\max}].$$

Procedure:

1. Sweep azimuth from $\alpha_{\min}$ to $\alpha_{\max}$,
2. Step elevation by $\Delta\epsilon$,
3. Repeat.

### 4.6.2 Mathematical Form

Let $i$ be the azimuth index and $j$ the elevation index:

$$\alpha_i = \alpha_{\min} + i\Delta\alpha, \quad \epsilon_j = \epsilon_{\min} + j\Delta\epsilon.$$

## 4.7 Temporal Sampling and Revisit Time

The revisit time for a given direction is:

$$T_{\text{rev}} = N_{\text{points}} \cdot \Delta t_{\text{dwell}},$$

where $N_{\text{points}}$ is the number of pointings in the scan.

Revisit time directly impacts track quality: long revisit times reduce update rate and increase state uncertainty.

### 4.8 Relationship to Tracking

Track update rate depends on:

- whether the beam intersects the target direction,
- how often that intersection occurs.

Thus:

$$\text{scan pattern} \rightarrow \text{measurement times} \rightarrow \text{filter stability}.$$

### 4.9 Adaptive and Track-Driven Scanning

The code architecture supports adaptive scanning:

- confirmed tracks can request service,
- a round-robin queue can schedule track revisits.

While not fully implemented yet, the scan patterns can be extended to:

- prioritize confirmed tracks,
- dynamically modify $\alpha_c(t)$ based on predicted track position,
- switch between search and track modes.

### 4.10 Assumptions and Limitations

- No beam slewing dynamics (instantaneous pointing change),
- No mechanical constraints or acceleration limits,
- No scan-to-scan coupling with track quality yet.

### 4.11 Possible Extensions

- Add beam slew rate limits,
- Implement track-while-scan scheduling,
- Optimize scan patterns for detection probability,
- Introduce stochastic scan perturbations.

### 4.12 Summary

Scan patterns govern the radar's attention. They define where the radar looks, how often it revisits each direction, and how effectively it supports tracking.

They form the operational layer between sensor physics and estimator logic.

## 5 Radar Sensor and Detection Physics

This chapter describes the physical radar model, including geometry, antenna behavior, signal-to-noise computation, and detection logic. It explains how raw target geometry and radar parameters are transformed into detections.

### 5.1 Radar Conceptual Model

The simulation models a monostatic pulsed radar with:

- A steerable antenna beam,
- A single coherent processing interval per dwell,
- Noise-limited detection,
- No explicit clutter or interference.

Each dwell produces at most one measurement per target.

### 5.2 Radar Parameters

The radar is parameterized by:

- $P_t$ : transmit power [W],
- $f$ : carrier frequency [Hz],
- $\lambda = c/f$ : wavelength,
- $G$ : peak antenna gain (linear),
- $\mathrm{BW}_\alpha, \mathrm{BW}_\epsilon$ : 3 dB beamwidths,
- $L$ : system loss,
- $N_F$ : noise figure,
- $B$ : receiver bandwidth,
- $T_0$ : system noise temperature,
- $\mathrm{SNR}_{\min}$ : detection threshold.

These parameters define the sensitivity and resolution of the radar.

### 5.3 Line-of-Sight Geometry

The radar computes the line-of-sight vector in ENU:

$$\ell = \mathbf{r}_{\text{target}} - \mathbf{r}_{\text{radar}}.$$

Range:

$$R = \|\ell\|.$$

Azimuth and elevation are:

$$\text{az} = \text{atan2}(\ell_E, \ell_N), \quad \text{el} = \text{atan2}(\ell_U, \sqrt{\ell_E^2 + \ell_N^2}).$$

### 5.4 Antenna Gain Model

The antenna mainlobe is approximated by a separable Gaussian:

$$G(\Delta\alpha, \Delta\epsilon) = G_0 \exp\left(-4 \ln 2 \left(\frac{\Delta\alpha^2}{\mathrm{BW}_\alpha^2} + \frac{\Delta\epsilon^2}{\mathrm{BW}_\epsilon^2}\right)\right).$$

This model:

- Has unity at boresight,
- Falls to $-3$ dB at $\Delta\alpha = \mathrm{BW}_\alpha/2$.

### 5.5 Radar Equation

The SNR is computed using the monostatic radar equation:

$$\mathrm{SNR} = \frac{P_t G^2 \lambda^2 \sigma T_{\mathrm{dwell}}}{(4\pi)^3 R^4 k T_0 B L N_F}.$$

This expression assumes:

- Free-space propagation,
- Isotropic thermal noise,
- No multipath or fading.

### 5.6 Dwell Time

Longer dwell increases coherent integration time:

$$\mathrm{SNR} \propto T_{\mathrm{dwell}}.$$

Thus scan speed trades coverage for sensitivity.

### 5.7 Detection Decision

Detection is binary:

$$\mathrm{detected} = \begin{cases} 1 & \text{if } 10\log_{10}(\mathrm{SNR}) \geq \mathrm{SNR}_{\min}, \\ 0 & \text{otherwise.} \end{cases}$$

### 5.8 Field of View Gating

A RadarFace constrains detections to:

$$\alpha_{\min} \leq \mathrm{az} \leq \alpha_{\max}, \quad \epsilon_{\min} \leq \mathrm{el} \leq \epsilon_{\max}, \quad R \leq R_{\max}.$$

Outside this region, detections are suppressed.

### 5.9 Aspect-Dependent RCS Coupling

The radar uses the body-frame line-of-sight vector to query the RCS model:

$$\sigma = \sigma(\phi, \theta, t),$$

linking orientation and detection probability.

### 5.10 Assumptions and Limitations

- No Doppler or range-rate modeling,
- No clutter or false alarms,
- No sidelobe modeling,
- No multipath or terrain effects.

### 5.11 Possible Extensions

- Add Doppler processing,
- Model sidelobes and grating lobes,
- Include clutter and CFAR detection,
- Add bistatic or multistatic configurations.

### 5.12 Summary

The radar module maps geometry and orientation into measurable quantities via physically interpretable models. It is the sensing front-end of the simulation.

## 6 Radar Cross Section Modeling

This chapter describes how the target's electromagnetic scattering properties are modeled and how they are coupled to target orientation and radar line-of-sight.

### 6.1 Physical Meaning of Radar Cross Section

Radar Cross Section (RCS) is defined as:

$$\sigma = \lim_{R \to \infty} 4\pi R^2 \frac{P_{\text{scattered}}}{P_{\text{incident}}},$$

and represents the effective area of a target as seen by the radar.

RCS depends on:

- Target shape and size,
- Material properties,
- Radar wavelength,
- Aspect angles (viewing direction relative to the body),
- Polarization (not modeled here).

### 6.2 Aspect-Dependent RCS

The simulation models RCS as a function of body-frame aspect angles:

$$\sigma = \sigma(\phi, \theta, t),$$

where:

- $\phi$ is the azimuth in the body frame,
- $\theta$ is the elevation in the body frame,
- $t$ allows for time-varying RCS.

### 6.3 Mapping from LOS to Body Aspect

Given the unit line-of-sight vector in ENU, $\hat{\mathbf{u}}_e$, and body-to-ENU rotation $\mathbf{R}_{b\to e}$:

$$\hat{\mathbf{u}}_b = \mathbf{R}_{b\to e}^T \hat{\mathbf{u}}_e.$$

Aspect angles are computed as:

$$\phi = \text{atan2}(u_{b,x}, u_{b,y}), \quad \theta = \text{atan2}\left(u_{b,z}, \sqrt{u_{b,x}^2 + u_{b,y}^2}\right).$$

### 6.4 General RCS Grid Model

The general RCS model stores a sampled RCS map:

$$\sigma_{ij} = \sigma(\phi_i, \theta_j),$$

on a regular grid of angles.

Interpolation is performed using bilinear interpolation in $(\phi, \theta)$ space.

### 6.5 Simple Parametric Flip Model

For controlled experiments, a simplified model is used:

- $\sigma_{\text{front}}$ when target faces radar,
- $\sigma_{\text{back}}$ when target faces away,
- $\sigma_{\text{edge}}$ when target is sideways.

The model classifies the aspect angle into one of these regions based on $\theta$ and $\phi$.

### 6.6 Apparent RCS

Apparent RCS includes antenna gain:

$$\sigma_{\text{app}} = \sigma \frac{G(\Delta\alpha, \Delta\epsilon)}{G_0}.$$

This accounts for reduced illumination when off-boresight.

### 6.7 Temporal Variation

Time-varying RCS is supported through explicit dependence on $t$ or via time-varying orientation.

### 6.8 Assumptions and Limitations

- Monostatic RCS only,
- No polarization dependence,
- No frequency dependence,
- No speckle or scintillation,
- No multipath or shadowing.

### 6.9 Possible Extensions

- Polarization modeling,
- Frequency-dependent RCS,
- Stochastic RCS fluctuations (Swerling models),
- Bistatic RCS,
- High-fidelity EM solvers.

### 6.10 Summary

The RCS model provides the link between target geometry, orientation, and detectability. It is the mechanism by which physical shape influences radar observability.

## 7 Radar Face and Field-of-View Gating

This chapter describes how geometric visibility constraints are applied to radar detections. The radar face defines where the radar is physically capable of observing and imposes hard constraints on detections.

### 7.1 Motivation for Field-of-View Modeling

Real radars are mechanically and electronically constrained:

- Antennas can only steer within certain azimuth/elevation limits,
- Terrain, mounting, and platform geometry restrict visibility,
- Sensors may be sector-limited (e.g., shipboard or vehicle-mounted radars).

The radar face models these constraints explicitly.

### 7.2 Definition of the Radar Face

A radar face is defined by:

$$\alpha_{\min}, \alpha_{\max}, \epsilon_{\min}, \epsilon_{\max}, R_{\max}.$$

These define a 3D angular sector:

- Azimuth bounds: $[\alpha_{\min}, \alpha_{\max}]$,
- Elevation bounds: $[\epsilon_{\min}, \epsilon_{\max}]$,
- Maximum detection range $R_{\max}$.

Only detections within this volume are considered valid.

### 7.3 Wrap-Aware Angular Gating

Azimuth is circular modulo $2\pi$. The interval $[\alpha_{\min}, \alpha_{\max}]$ may wrap around zero (e.g., 300° to 60°).

To handle this robustly:

$$\text{span} = (\alpha_{\max} - \alpha_{\min}) \bmod 2\pi.$$

If span $\approx 2\pi$, the face covers all azimuths.

Otherwise:

$$\text{az\_ok} = \begin{cases} \alpha_{\min} \le \alpha \le \alpha_{\max}, & \alpha_{\min} \le \alpha_{\max}, \\ (\alpha \ge \alpha_{\min}) \vee (\alpha \le \alpha_{\max}), & \alpha_{\min} > \alpha_{\max}. \end{cases}$$

### 7.4 Elevation Gating

Elevation is linear and bounded:

$$\epsilon_{\min} \leq \epsilon \leq \epsilon_{\max}.$$

### 7.5 Range Gating

Targets beyond $R_{\max}$ are suppressed:

$$R \leq R_{\max}.$$

### 7.6 Combined Visibility Test

A detection is visible if and only if:

$$\text{visible} = (\text{az\_ok}) \wedge (\text{el\_ok}) \wedge (R \leq R_{\max}).$$

### 7.7 Interaction with Detection

The radar face operates after raw detection:

1. Radar computes SNR and provisional detection,
2. RadarFace checks geometric visibility,
3. If not visible, detection is forced false.

This preserves physical realism by preventing detections outside mechanical limits.

### 7.8 Implementation Details

The gating logic is implemented in `RadarFace.is_within_fov()`.

Important considerations:

- Azimuth values are wrapped to $[-\pi, \pi)$ before comparison,
- Full-circle faces are detected and treated as always visible,
- Small numerical tolerances avoid wrap-edge artifacts.

### 7.9 Practical Examples

**Forward-facing radar:**

$$\alpha_{\min} = -90^\circ,\ \alpha_{\max} = 90^\circ.$$

**Shipboard starboard face:**

$$\alpha_{\min} = 0^\circ,\ \alpha_{\max} = 180^\circ.$$

**Full surveillance radar:**

$$\alpha_{\min} = 0^\circ,\ \alpha_{\max} = 360^\circ.$$

### 7.10 Assumptions and Limitations

- No terrain masking or shadowing,
- No occlusion by the host platform,
- No diffraction or sidelobe visibility,
- No time-varying mechanical limits.

### 7.11 Possible Extensions

- Add terrain and horizon masking,
- Include platform attitude constraints,
- Support dynamic FOV (e.g., gimbal limits),
- Add per-face sidelobe detection models.

### 7.12 Summary

The radar face defines what is physically observable. It acts as a geometric filter that enforces realism on the detection process and scan logic.

## 8 Tracking: Estimation, Association, and Track Management

This chapter describes in detail how tracking is implemented in the simulation. The code uses a *constant-velocity* (CV) *Extended Kalman Filter* (EKF), with measurements in spherical coordinates:

$$\mathbf{z} = [R,\ \text{az},\ \text{el}]^T,$$

where $R$ is slant range, az is azimuth, and el is elevation in the radar ENU frame. Track-to-measurement association is performed with nearest-neighbor (NN) gating, and track life-cycle management includes hit/miss counting and confirmation logic.

### 8.1 State Definition and Kinematic Model

Each track maintains a 6D Cartesian state in the radar local ENU frame:

$$\mathbf{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^T,$$

where $(x, y, z)$ are position components (East, North, Up) in meters, and $(v_x, v_y, v_z)$ are velocities in meters per second.

#### 8.1.1 Constant-Velocity Discrete-Time Dynamics

The constant-velocity model assumes velocity is approximately constant between updates:

$$\mathbf{x}_{k+1} = \mathbf{F}(\Delta t)\mathbf{x}_k + \mathbf{w}_k,$$

with state transition matrix

$$\mathbf{F}(\Delta t) = \begin{bmatrix} \mathbf{I}_3 & \Delta t\, \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}.$$

This matches the implementation (where $F_{0,3} = F_{1,4} = F_{2,5} = \Delta t$).
The process noise $\mathbf{w}_k$ captures unmodeled accelerations.

### 8.2 Process Noise Model: White-Acceleration (Integrated) Noise

The code uses a white-acceleration process noise model. Let the continuous-time acceleration be white noise with spectral density parameter $q$ (code parameter `q_acc`). The corresponding discrete-time process noise covariance for one axis is the standard integrated form:

$$\mathbf{Q}_{1D}(\Delta t) = q \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix}.$$

For 3D with independent axes, $\mathbf{Q}$ is block-diagonal across axes, yielding:

$$\mathbf{Q}(\Delta t) = \begin{bmatrix} q\frac{\Delta t^4}{4}\mathbf{I}_3 & q\frac{\Delta t^3}{2}\mathbf{I}_3 \\ q\frac{\Delta t^3}{2}\mathbf{I}_3 & q\Delta t^2\mathbf{I}_3 \end{bmatrix}.$$

This is exactly how `CVKalman.Q(dt)` is filled:

$$Q_{pp} = q\Delta t^4/4, \quad Q_{pv} = q\Delta t^3/2, \quad Q_{vv} = q\Delta t^2.$$

The parameter $q$ is a tuning knob: higher $q$ allows the filter to follow more maneuvering targets at the cost of noisier estimates.

## 8.3 Kalman Prediction Step

Given the current estimate $(\mathbf{x}, \mathbf{P})$, prediction is:

$$\mathbf{x}_{k+1}^- = \mathbf{F}(\Delta t)\mathbf{x}_k, \qquad \mathbf{P}_{k+1}^- = \mathbf{F}(\Delta t)\mathbf{P}_k\mathbf{F}(\Delta t)^T + \mathbf{Q}(\Delta t).$$

In code, this is `CVKalman.predict(dt)`.

## 8.4 Measurement Model in Spherical Coordinates

Radar detections are treated as measurements of range, azimuth, and elevation:

$$\mathbf{z} = \begin{bmatrix} R \\ \text{az} \\ \text{el} \end{bmatrix}.$$

The EKF needs a nonlinear measurement function $\mathbf{h}(\mathbf{x})$.

### 8.4.1 Line-of-Sight Geometry

Let $\mathbf{r} = [x, y, z]^T$ be the target position in ENU and $\mathbf{r}_r$ the radar position in ENU (often zero in your simulation). Define the line-of-sight vector:

$$\ell = \mathbf{r} - \mathbf{r}_r = \begin{bmatrix} \ell_E \\ \ell_N \\ \ell_U \end{bmatrix}.$$

Then

$$R = \|\ell\| = \sqrt{\ell_E^2 + \ell_N^2 + \ell_U^2}.$$

Azimuth is measured from North toward East (consistent with $\arctan 2(E, N)$):

$$\text{az} = \text{atan2}(\ell_E,\ \ell_N).$$

Elevation is:

$$\text{el} = \text{atan2}\left(\ell_U,\ \sqrt{\ell_E^2 + \ell_N^2}\right).$$

This is implemented by `CVKalman.h(x, radar_pos_enu)`. The code also maps azimuth into $[0, 2\pi)$ by adding $2\pi$ if negative.

## 8.5 Extended Kalman Filter Update

The EKF linearizes $\mathbf{h}$ around the predicted state $\mathbf{x}^-$:

$$\mathbf{z} \approx \mathbf{h}(\mathbf{x}^-) + \mathbf{H}(\mathbf{x}^-)(\mathbf{x} - \mathbf{x}^-) + \mathbf{v},$$

where $\mathbf{v} \sim \mathcal{N}(0, \mathbf{R})$ and $\mathbf{H}$ is the Jacobian:

$$\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}.$$

### 8.5.1 Numeric Jacobian Used in Code

Rather than implementing closed-form derivatives, the code computes $\mathbf{H}$ numerically:

$$H_{:,i} \approx \frac{\mathbf{h}(\mathbf{x} + \epsilon_i \mathbf{e}_i) - \mathbf{h}(\mathbf{x})}{\epsilon_i},$$

with a scale-aware perturbation vector:

$$\epsilon = [1, 1, 1, 0.1, 0.1, 0.1],$$

meaning 1 meter perturbations in position and 0.1 m/s in velocity. This is implemented in `CVKalman.H_numeric`.

**Angle wrapping in the Jacobian.** Because azimuth is angular and wraps at $2\pi$, the code wraps the azimuth difference during Jacobian calculation:

$$\Delta\text{az} \leftarrow \text{wrap}(\Delta\text{az}),$$

ensuring that derivatives near $0/2\pi$ are not corrupted by discontinuities.

### 8.5.2 Innovation (Residual) and Wrapping

Define the predicted measurement:

$$\hat{\mathbf{z}} = \mathbf{h}(\mathbf{x}^-).$$

Innovation:

$$\mathbf{y} = \mathbf{z} - \hat{\mathbf{z}}.$$

Azimuth residual must be wrapped:

$$y_{\text{az}} \leftarrow \text{wrap}(y_{\text{az}}),$$

which the code does via `angle_wrap`.

### 8.5.3 Kalman Gain and Covariance Update

Innovation covariance:

$$\mathbf{S} = \mathbf{H}\mathbf{P}^-\mathbf{H}^T + \mathbf{R}.$$

Kalman gain:

$$\mathbf{K} = \mathbf{P}^-\mathbf{H}^T\mathbf{S}^{-1}.$$

State update:

$$\mathbf{x}^+ = \mathbf{x}^- + \mathbf{K}\mathbf{y}.$$

Covariance update (as implemented):

$$\mathbf{P}^+ = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}^-.$$

These correspond exactly to `CVKalman.update`.

**Note on Joseph form.** A numerically robust alternative is the Joseph stabilized update:

$$\mathbf{P}^+ = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}^-(\mathbf{I} - \mathbf{K}\mathbf{H})^T + \mathbf{K}\mathbf{R}\mathbf{K}^T.$$

Your current code uses the simpler form; for most benign scenarios it is adequate, but Joseph form can help maintain symmetry/PSD under numerical stress.

## 8.6 Measurement Noise Model R

Measurement covariance $\mathbf{R}$ is a 3x3 matrix for $[R, \text{az}, \text{el}]$. If not provided, the code uses defaults:

$$\mathbf{R} = \text{diag}\left(25^2,\ (\text{deg2rad}(0.2))^2,\ (\text{deg2rad}(0.2))^2\right).$$

Interpreting these:

- Range standard deviation: 25 m,
- Azimuth standard deviation: 0.2 degrees,
- Elevation standard deviation: 0.2 degrees.

This is a convenient starting point; in higher fidelity modeling these could be functions of SNR, beamwidth, and waveform.

## 8.7 Track Initialization from a Single Detection

When a detection $(R, \text{az}, \text{el})$ is not associated to any existing track, a new track is spawned.

The code converts the spherical measurement into ENU Cartesian position:

$$\ell_E = R\cos(\text{el})\sin(\text{az}), \quad \ell_N = R\cos(\text{el})\cos(\text{az}), \quad \ell_U = R\sin(\text{el}).$$

Then:

$$\mathbf{r} = \mathbf{r}_r + [\ell_E, \ell_N, \ell_U]^T.$$

Velocities are initialized to zero:

$$[v_x, v_y, v_z] = [0, 0, 0].$$

This is implemented in `TrackManager._init_state_from_meas`.

### 8.7.1 Initial Covariance

The code assigns a relatively large initial covariance:

$$\mathbf{P}_0 = \text{diag}(500^2,\ 500^2,\ 200^2,\ 50^2,\ 50^2,\ 20^2).$$

This encodes significant uncertainty in initial position and velocity, letting the filter converge as more detections arrive.

### 8.7.2 Process Noise Tuning for New Tracks

New tracks use `q_acc=5.0`, which sets the process noise power in `CVKalman.Q(dt)`. This value governs how quickly the track can adapt to changes in motion.

## 8.8 Association and Gating (Nearest Neighbor)

Given a set of detections $\{\mathbf{z}_i\}$ and existing tracks $\{T_j\}$, the code tries to assign at most one detection to each track.

### 8.8.1 Predicting Each Track's Measurement

For each track, a predicted measurement is computed from the predicted state:

$$\hat{\mathbf{z}}_j = \mathbf{h}(\mathbf{x}_j, \mathbf{r}_r).$$

This is done by `_meas_from_track` using `CVKalman.h`.

### 8.8.2 Rectangular Gates

A detection $\mathbf{z}$ is eligible for a track if:

$$|R - \hat{R}| \le g_R, \quad |\mathrm{wrap}(\mathrm{az} - \widehat{\mathrm{az}})| \le g_{\mathrm{az}}, \quad |\mathrm{el} - \widehat{\mathrm{el}}| \le g_{\mathrm{el}},$$

where $g_R$, $g_{\mathrm{az}}$, $g_{\mathrm{el}}$ are gate thresholds (set from `gate_range_m`, `gate_az_deg`, `gate_el_deg`). This is a simple and fast gating strategy.

### 8.8.3 Nearest-Neighbor Scoring

Among detections that pass the gate, the code selects the one with smallest score:

$$\mathrm{score}(\mathbf{z}, \hat{\mathbf{z}}) = \frac{|R - \hat{R}|}{g_R} + \frac{|\mathrm{wrap}(\mathrm{az} - \widehat{\mathrm{az}})|}{g_{\mathrm{az}}} + \frac{|\mathrm{el} - \widehat{\mathrm{el}}|}{g_{\mathrm{el}}}.$$

This is a normalized Manhattan-like residual metric.

**Why normalize by gate sizes?** It makes the score dimensionless and ensures that range and angles contribute comparably relative to their acceptance thresholds.

### 8.8.4 Update and Bookkeeping

If a track is assigned a detection:

- EKF update is applied,
- `hits` increments,
- `misses` resets to 0,
- `last_update_t` updated.

If no detection is assigned:

- `misses` increments,
- if misses exceed `drop_misses`, the track is deleted.

## 8.9 Track Confirmation Logic

Tracks begin unconfirmed. A track becomes confirmed when:

$$\texttt{hits} \ge \texttt{promote\_hits}.$$

Once confirmed, the track is eligible for scheduling service in the round-robin queue.

**Interpretation.** This implements a basic $M/N$-like confirmation rule (though here it is simply $M$ hits total, not necessarily consecutive). The intention is to prevent single spurious detections from creating persistent tracks.

## 8.10 Track Deletion Logic

Tracks are removed when:

$$\texttt{misses} \ge \texttt{drop\_misses}.$$

Deletion removes them from both the active track map and the round-robin service queue.

### 8.11 Round-Robin Track Service Scheduling

The code maintains a deque of confirmed track IDs. When the radar needs to decide which confirmed track to service next, it uses:

$$\texttt{tid} \leftarrow \texttt{popleft()}, \quad \texttt{append}(tid).$$

This produces a simple round-robin schedule, implemented in `next_track_for_service()`.

**Connection to radar control.** In a full Track-While-Scan system, this method supports allocating additional dwell time to confirmed tracks (track updates) while still scanning the broader volume. Your current simulation includes the mechanism but may not yet drive scan-pointing based on this queue (depending on how `main.py` uses it).

### 8.12 Assumptions, Limitations, and Extensions

#### 8.12.1 Assumptions

- Single-radar monostatic geometry, ENU local frame.
- Measurement vector is $(R, \text{az}, \text{el})$ with fixed covariance $\mathbf{R}$.
- Targets follow nearly constant velocity with random accelerations.
- Association uses nearest neighbor with rectangular gates.

#### 8.12.2 Limitations

- No explicit false-alarm model or clutter; detections are assumed target-originated.
- No track splitting/merging logic; multi-target scenarios can cause association errors.
- Jacobian computed numerically (robust, but slower than analytic form).
- Covariance update uses simple form rather than Joseph stabilized form.

#### 8.12.3 Suggested Extensions

- **Statistical gating:** Replace rectangular gates with Mahalanobis gating $\mathbf{y}^T\mathbf{S}^{-1}\mathbf{y} \leq \gamma$.
- **Probabilistic association:** Implement PDAF or JPDA for dense target environments.
- **SNR-dependent measurement noise:** Make $\mathbf{R}$ depend on SNR and beamwidth (e.g., angle variance proportional to $\text{BW}^2/\text{SNR}$).
- **Track management:** Use an $M$-of-$N$ consecutive hits rule and separate tentative/confirmed states.
- **Joseph form covariance:** Improve numerical stability, especially with long runs.

## 9 Simulation Flow

At each timestep:

1. Update scan pointing,
2. Compute LOS and angles,

3. Compute antenna gain,
4. Compute RCS from orientation,
5. Compute SNR and detection,
6. Apply FOV gating,
7. Feed detections into tracker,
8. Save output.

## 10 Visualization

The output is post-processed to show:

- Azimuth/elevation tracks,
- RCS vs time,
- Beam footprint and target position,
- Target orientation animation.

## 11 Conclusion

This framework cleanly separates physics, geometry, signal processing, and tracking into modular components. It allows experimentation with scan strategies, RCS models, and tracker parameters while remaining physically interpretable.