Trajectory function

Goal

Generate a five-minute, physically plausible 3D drone trajectory sampled at 100 Hz that:

- Stays inside a bounded airspace,
- Is smooth (continuous position, velocity, and acceleration),
- Respects motion limits on speed, acceleration, and jerk (rate of change of acceleration),
- Is randomized every run to provide diverse test cases for filters and intercept algorithms.

Output: traj ∈ $\mathbb{R}^{N\times3}$ with columns [x y z] in meters, where N ≈ 5 min × 100 Hz = 30,000.

Setup & Key Parameters

- Sampling: dt = 0.01 s (100 Hz); duration = 5 min → N = round(duration/dt).
- Bounds: x,y ∈ [−1200, 1200] m, z ∈ [40, 700] m (keeps the path inside a safe "box").
- Kinematic limits: v_max = 25 m/s, a_max = 8 m/s^2, j_max = 50 m/s^3.
- Randomness: rng('shuffle') seeds the RNG from system time so every run produces a new path.

Algorithm (high level)

1. Waypoint sampling & spline path
   - Draw 10–20 random waypoints over the 5-minute horizon (including start and end times).
   - For each axis (x, y, z), form a shape-preserving cubic spline through those waypoints using interp1(...,'pchip').

- Result: a smooth reference position path raw(t) without any dynamics limits yet.

2. Velocity, acceleration, and jerk limiting (online profile shaping)
   - For each time step k:
     - Compute desired velocity from the reference path: $v\_des = (raw(k) - raw(k-1)) / dt$.
     - Clip speed: if $\|v\_des\| > v\_max$, scale it back along its direction.
     - Compute desired acceleration to move current velocity toward v_des:
       $a\_des = (v\_des - v(k-1)) / dt$, then clip to a_max.
     - Enforce jerk limit:
       $j = (a\_des - a(k-1)) / dt$; if $\|j\| > j\_max$, scale j and recompute $a\_des = a(k-1) + j \cdot dt$.
     - Integrate to update dynamics:
       - $a(k) = a\_des$
       - $v(k) = v(k-1) + a(k) \cdot dt$

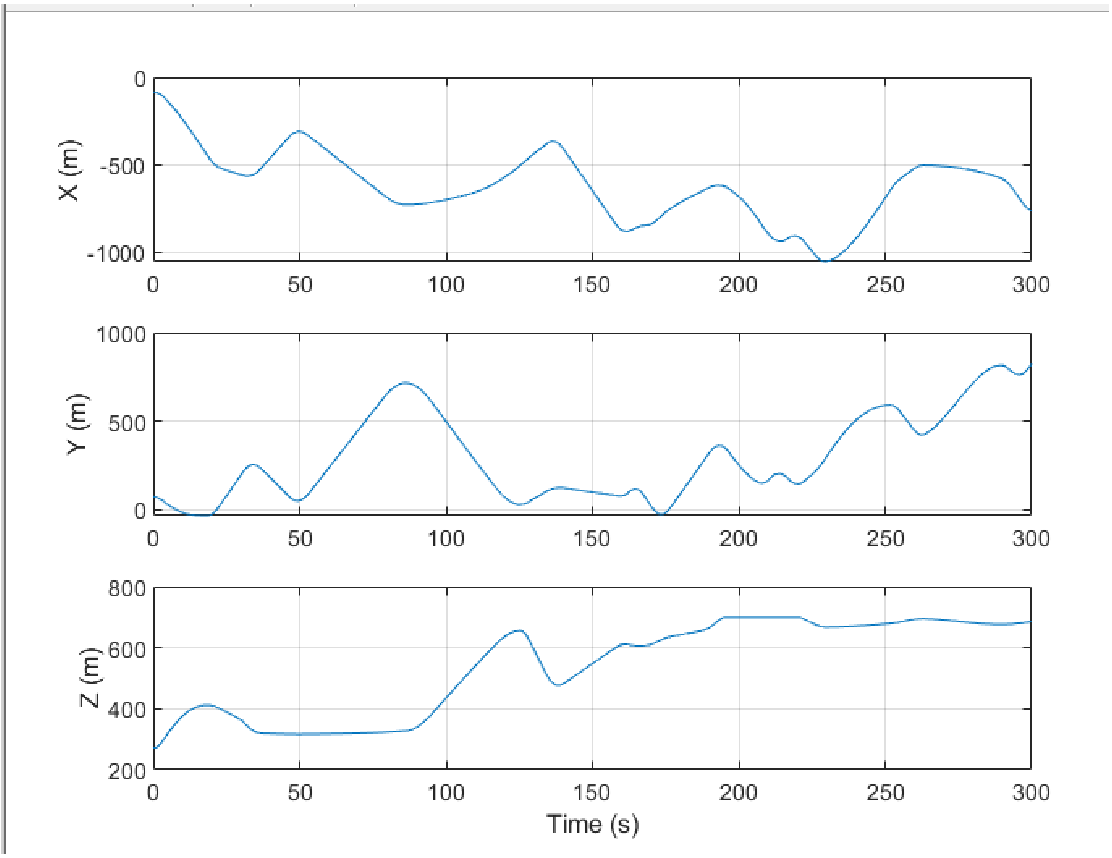
This loop converts an arbitrary spline into a feasible motion profile that respects v/a/jerk constraints—i.e., smooth and flyable.

3. Position recomputation by integration
   - Starting from the initial spline point, integrate the limited velocity:
     - $traj(k) = traj(k-1) + v(k) \cdot dt$
   - This guarantees position is consistent with the limited velocity/acceleration.
4. Clamping to domain
   - After integration, clamp each axis to the allowed bounds to avoid numerical drift outside the airspace.
5. Metrics & plots

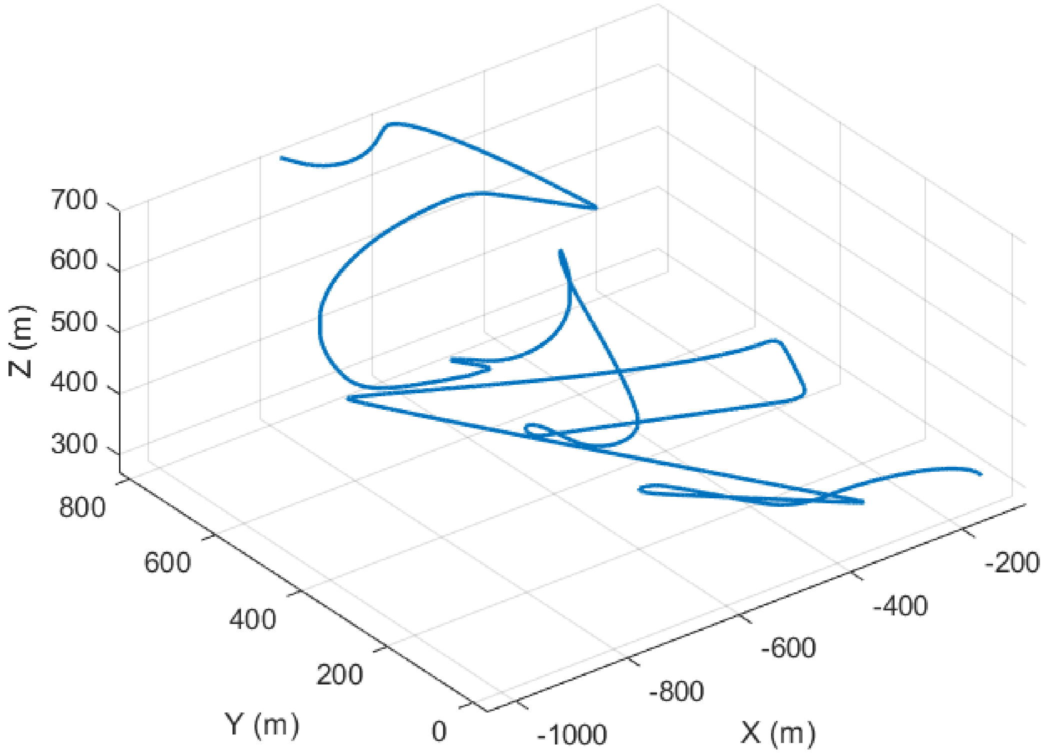
  - Compute ‖v‖ and ‖a‖ over time with vecnorm and print averages.
  - Produce quick-look plots: 3D path, positions vs. time, and velocity/acceleration profiles.

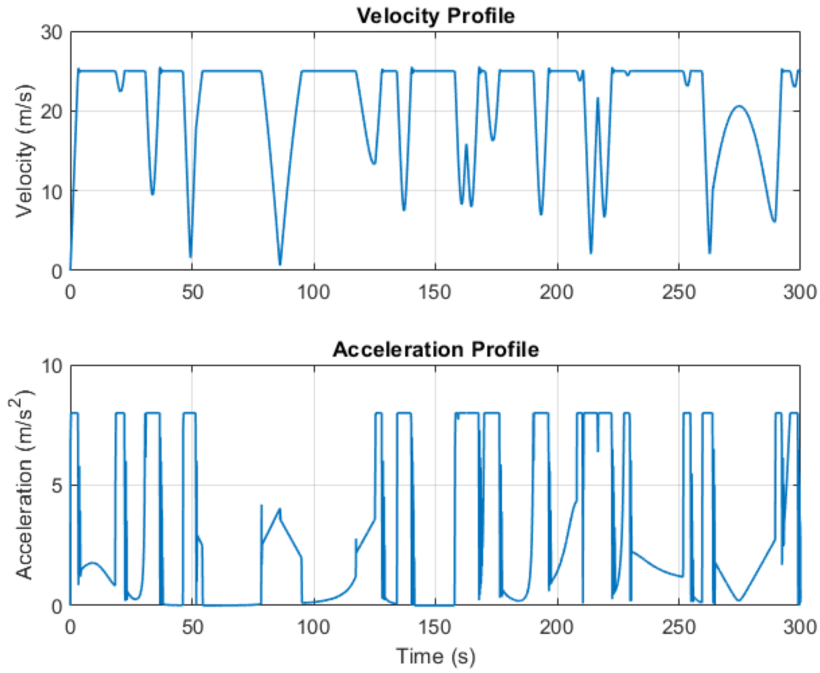
Why these design choices?

- PCHIP ('pchip'): shape-preserving cubic interpolation avoids overshoot common in standard cubic splines, producing realistic paths between random waypoints.
- Sequential limiting (speed → accel → jerk): mirrors real flight controllers where commands are saturated hierarchically for flyability and to reduce excitation of the estimator.
- Jerk limiting: ensures continuous acceleration, reducing sharp changes that can destabilize filters or cause unrealistic actuator demands.
- Integrate velocity to position: keeps states physically consistent after limiting.
- Random waypoints + shuffled RNG: creates diverse trajectories for robust testing of filters and learning-based tuning.



Randomized Smooth Trajectory





Velocity Profile
Velocity (m/s)
Acceleration Profile
Acceleration (m/s^2)
Time (s)