Synchronized Digital Twin Prediction and Decision Support for Six-Degree-of-Freedom Nonlinear Aircraft Landing

Rick Zhang

A thesis
submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS & ASTRONAUTICS

University of Washington

2026

Committee:

Mehran Mesbahi

Amir Taghvaei

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Synchronized Digital Twin Prediction and Decision Support for Six-Degree-of-Freedom Nonlinear Aircraft Landing

Rick Zhang

Chair of the Supervisory Committee:
Mehran Mesbahi
Aeronautics and Astronautics

A synchronized digital-twin is used to provide decision support for nonlinear six-degree-of-freedom aircraft landing under wind-induced plant–twin mismatch. A nominal landing trajectory is first generated offline by solving a constrained nonlinear trajectory-optimization problem with an imperfect physics-based aircraft model. The reference is then tracked on a higher-fidelity plant proxy using a closed loop tracking controller. During execution, wind disturbances, actuator lag, and initial-condition errors cause the plant trajectory to deviate from the nominal prediction. An Extended Kalman Filter is used as the data assimilation method to estimate a reduced local pose-and-wind state from the measured aircraft response. At each update time, the digital twin uses the current synchronized state to initialize a set of candidate terminal control modes that modify the tracking controller near touchdown, and evaluates each one on how it improves landing metrics. Simulation results shows that digital twin support can provide real time prediction of future aircraft states by propagating the nonlinear aircraft dynamics forward, and intervene ahead of time to reduce harsh terminal conditions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author would like to thank Dr. Mehran Mesbahi for their guidance on the direction of this thesis, as well as Dr. Amir Taghvaei for their instruction on nonlinear control systems and state estimation. Shoutout to the members of the Robotics, Aerospace, and Information Networks (RAIN) Lab for their input throughout the development of this work.

Chapter 1

INTRODUCTION

Aircraft landing remains a challenging controls problem due to the nonlinear nature of flight dynamics, the presence of safety-critical state and input constraints, and the mismatch between mathematical models and the true vehicle dynamics. Even when a nominal landing trajectory is generated using a high-quality model, the actual aircraft may deviate from that prediction because of unmodeled disturbances. These effects are especially important near the runway, where lateral alignment, glide-path behavior, attitude limits, actuator usage, and touchdown conditions all matter over a short time horizon.

A digital twin is a dynamical model that can be used to represent, predict, and support decision-making for a physical system. In general, the term usually refers to an asset-specific model that evolves with incoming data [1]. In aerospace, much of the existing digital-twin literature has focused on operations, maintenance, structural integrity, condition monitoring, and lifecycle support, rather than on uses inside flight-control laws [2, 3]. Recent work has explored cloud-based digital twins, digital-twin frameworks for Unmaned aerial systems operations, and physics-based digital twins for autonomous UAV landing [4, 5, 6]. In space systems, digital twin concepts have been investigated for on-board spacecraft software development and testing, mission simulation, thermal management in satellites, and aid in the development and operation of electric propulsion systems [7, 8, 9].

For automatic airplane landing, controllers are typically designed using classical feedback, optimal control, adaptive control, nonlinear control, intelligent control, robust H_2/H_∞ methods, LQR/LQG, QFT, sliding mode control, μ-synthesis, and fuzzy techniques [10]. During landing, the aircraft must track the desired glide path, regulate attitude, avoid excessive sink rate or floating during flare, and maintain actuator and flight-envelope limits while operating close to the runway. Robust landing-control studies have therefore placed particular emphasis on wind turbulence, wind shear, sensor errors, actuator faults, and model uncertainty [11, 12, 13]. More recent UAV and carrier-landing work has extended this tracking viewpoint with collision-avoidance logic, nonlinear backstepping and dynamic-inversion controllers [14, 15].

Model Predictive Control (MPC) is a common approach for constrained landing-control problems be-

cause it uses an internal model to predict future motion and repeatedly solves a finite-horizon optimization problem for the control input. This makes MPC attractive when state limits, actuator limits, touchdown requirements, or landing-risk measures need to be included directly in the control problem. At the same time, MPC depends on having a model that is accurate enough for prediction and simple enough to evaluate online, so aircraft applications often rely on linearization, reduced-order models, robust formulations, or layered control architectures to make the method practical in real time [16, 17]. For example, Latif et al. used an H_∞-based MPC formulation for fixed-wing UAV landing under wind-shear disturbances, where the controller tracks predefined glide and flare references while enforcing input constraints [16]. Wang et al. developed an MPC-based automatic carrier-landing guidance law with an additional landing-risk term based on trajectory and touchdown-point prediction [17].

McClellan et al. developed a physics-based digital twin for autonomous UAV landing . Their architecture uses a digital-twin instance for MPC wrapped around a real-time digital-twin prototype for fluid-structure interaction and flight dynamics. The inner twin predicts the aircraft state and aerodynamic forces and moments during flight, while the outer MPC layer uses those predictions to compute constrained control inputs [6]. Because the high-fidelity fluid-structure model is too expensive to use directly online, the authors linearize the model about a pre-designed glideslope and project it onto a low-dimensional subspace to obtain a real-time reduced-order model. This makes the method computationally feasible, but it also highlights that in MPC, prediction quality depends on the validity of the model reduction, linearization, and disturbance assumptions. In fact, their preliminary flight results showed steady-state offsets associated with disturbances such as wind that were not included in the model used by the MPC, motivating the use of either integral correction, a database of precomputed reduced-order twins, or online model synchronization.

For a fixed reference trajectory, finite-horizon LQT/LQR and TVLQR are attractive methods as most of the expensive computational work is done offline. A nominal path is generated, the dynamics are linearized along that path, and the Riccati equations are solved ahead of time. Online implementation only involves applying a time-varying gain around the reference. In some works, a State-Dependent Ricatti Equation based controller was modified to better handle parametric uncertainty [18]. The ricatti equation may also be updated directly using uncertainty estimates [19].

Riccati-based tracking controllers provide another way to use model information for landing and flight-control problems without solving a full constrained optimal-control problem online. For a fixed reference trajectory, finite-horizon LQT and LQR are computationally inexpensive during online deployment. In the

offline setting, a nominal trajectory is generated, the dynamics are linearized along that trajectory, and the Riccati equations are solved backward in time to obtain a time-varying feedback schedule. Online deployment only involves evaluating the reference, computing the tracking error, and applying the corresponding gain. The main limitation is that the resulting controller is local to the nominal trajectory and linearization, so model mismatch, disturbances, or poor phase synchronization can degrade performance.

State-Dependent Riccati Equation (SDRE) methods provide a nonlinear extension of Riccati-based control by rewriting the dynamics in a state-dependent coefficient form and solving Riccati equations at selected states or time instants [20]. This has been applied to unmanned aircraft tracking using dual-loop SDRE structures, where separate Riccati-based tracking problems are solved for outer-loop kinematics and inner-loop flight dynamics [21]. Robust Riccati methods modify this structure to account for uncertainty, either by tuning SDRE weights or by updating a time-varying Riccati equation using estimated model uncertainty [18, 19]. These approaches generally rely on solving the Riccati equations for uncertainty-dependent gain updates online. In contrast, this work keeps the TVLQR gain schedule offline and uses EKF-based digital-twin synchronization to update the wind-aware internal model used for reference timing, feedforward correction, and short-horizon prediction.

This thesis studies nonlinear aircraft landing using an imperfect digital twin. The twin is first used offline to generate a nominal landing trajectory using a nonlinear trajectory optimization method. This nominal trajectory provides the planned state and input history for the maneuver, but it is not assumed to be a complete representation of the higher-fidelity plant. Instead, the plant includes effects that are simplified or omitted in the planning twin, including wind disturbances, actuator lag, and initial perturbations. This creates a controlled plant-twin mismatch to evaluate the online synchronization.

The main contribution of this work is to use the airplane's measured response to synchronize a physics-based digital twin during the landing maneuver and then use that synchronized twin for tracking support and terminal decision-making. Plant measurements are processed through an Extended Kalman Filter (EKF) that estimates wind-related mismatch. These estimates are used to update the synchronized digital twin, whose environmental state is corrected using information inferred from the plant response. This approach differs from digital-twin-informed landing methods that primarily use high-fidelity model reduction to make a detailed aerodynamic model suitable for MPC [6]. Here, the imperfect flight-dynamics model is updated online so that it becomes more useful as a wind-aware internal model for tracking support, short-horizon rollout, and terminal policy evaluation under plant–twin mismatch.

The synchronized twin is used in two main roles. First, it supports the closed-loop tracking controller by providing wind-aware information for reference timing, feedforward correction, and command shaping while retaining an offline TVLQR gain schedule. Second, the synchronized twin is used as a short-horizon consequence prediction model for terminal decision support. During the final phase of the landing maneuver, the current synchronized state and EKF wind estimate are used to roll the digital twin forward under a set of candidate terminal policies. These candidate policies are scored using predicted touchdown-quality metrics such as main-gear clearance, clearance rate, sink rate, pitch angle, penetration, touchdown timing, and touchdown location. The selected policy is then applied to the plant.

Segment-wise residual mismatch analysis is used to evaluate how much local plant–twin discrepancy remains after EKF synchronization. This diagnostic comparison complements the forecast and terminal-policy results by showing whether the synchronized twin provides a tighter local description of the plant than the nominal or unsynchronized twin. Additionally, each terminal policy-selection event produces a record of the synchronized state, EKF wind estimate, candidate-policy predictions, selected policy, and plant outcomes. These stored results form a digital-twin database that can be used for forecast reliability, evaluate policy effectiveness, and inform future cost tuning or residual-learning extensions.

The objective of this thesis is therefore to demonstrate how a physics-based digital twin can be updated from measured aircraft responses and used for online tracking support, predictive terminal decision-making, and data-driven post-flight analysis.

The following questions are addressed in this thesis:

a. **Can an EKF estimate a local pose-and-wind state accurately enough to synchronize a nonlinear aircraft landing digital twin without direct plant-state access?**

b. **Can EKF-based twin synchronization improve closed-loop tracking support through progress synchronization and wind-aware feedforward correction?**

c. **Can the synchronized digital twin serve as a short-horizon prediction and consequence model for terminal policy selection?**

d. **Can forecast-selected terminal policies improve touchdown quality of the aircraft?**

The remainder of this thesis is organized as follows.

Section 2 defines the landing problem, the plant–twin modeling objects, the closed-loop execution setting, and the role of online synchronization and terminal prediction. Section 3 introduces the RCAM aircraft model, the digital-twin modeling framework, and the main sources of plant–twin mismatch considered in this work. Sections 4, 5, and 6 then describe the supporting control architecture. Section 5 presents the nominal landing trajectory generation problem using the penalized trust region method, Section 5 develops the TVLQR tracking controller used to follow the nominal trajectory, and Section 6 introduces the EKF-based wind synchronization method used to update the digital twin during closed-loop execution.

Section 7 presents the proposed digital-twin formulation for landing decision support, including short-horizon forecasting, forecast-selected terminal policy supervision, and prediction–decision–outcome data logging. Section 8 reports the resulting forecast, landing-quality, and stored decision-data results. Finally, Section 9 discusses the main findings, limitations of the current implementation, and possible directions for future work. Readers primarily interested in the digital-twin use case may skip the PTR, TVLQR, and EKF developments in Sections 4–6 and begin with the decision-support formulation in Section 7.

Chapter 2

PROBLEM FORMULATION

Table 2.1: Summary of notation used in the problem formulation.

Symbol	Meaning
x_k^p	Plant aircraft state. The superscript p denotes plant.
a_k^p	Plant actuator state.
x_k^t	Imperfect digital-twin state used for planning or propagation.
$\hat{x}_k^s$	Synchronized digital-twin state during online execution.
$u_{c,k}$	Commanded control input sent to the plant actuator model.
$u_{a,k}$	Realized actuator state.
$z_k = [x_k^\top, u_{a,k}^\top]^\top$	Augmented tracking state used by the TVLQR controller.
s_k	Reference progress variable used to evaluate the scheduled trajectory.
$\chi_k = [r_N^\top, \eta^\top, w_N^\top]^\top$	Reduced pose-and-wind EKF synchronization state.
$\hat{\chi}_k$	EKF estimate of the reduced synchronization state.
$\hat{w}_k$	EKF wind estimate used by the synchronized digital twin.
$\Delta u_{\mathrm{sync},k}$	Synchronization-based command support.
$\Delta u_{\mathrm{terminal},k}$	Terminal-policy command modification near touchdown.
Π	Finite set of candidate terminal policies.
$m_i(\pi)$	Predicted terminal metric vector for policy π at selector update time t_i.
$\pi_i^\star$	Terminal policy selected at selector update time t_i.

This chapter defines the landing problem solved in this thesis. The goal is to perform a nonlinear aircraft landing maneuver when the aircraft model that is used for planning does not fully represent the model used for closed-loop execution. The nominal landing trajectory is generated offline with an imperfect physics-based digital twin. During execution, the aircraft is represented by a higher-fidelity plant proxy that includes effects that are simplified or left out of the planning model, including wind disturbances, actuator lag, and

initial-condition perturbations. The main components are the plant proxy, the imperfect digital twin, the nominal landing reference, the tracking controller, the EKF-based synchronization method, and the use of the synchronized twin for short-horizon terminal prediction. More detailed aircraft dynamics, trajectory optimization, TVLQR design, EKF implementation, and terminal-policy scoring are more fully discussed in later chapters.

2.1 Problem Setting and Modeling Objects

Plant proxy. The plant proxy represents the higher-fidelity closed-loop simulation model. The superscript p denotes plant quantities. In discrete time, the plant is written as

$$x_{k+1}^p = f_p\big(x_k^p, a_k^p, u_{c,k}, w_k\big)\,, \qquad a_{k+1}^p = g_p\big(a_k^p, u_{c,k}\big)$$

Here, x_k^p is the aircraft state, a_k^p represents actuator states, $u_{c,k}$ is the commanded control input, and w_k represents external disturbances and unmodeled effects. In this thesis, w_k is mainly associated with wind-induced plant–twin mismatch. The plant produces the measurement stream

$$y_k = h_p(x_k^p, a_k^p) + \nu_k$$

where ν_k represents measurement noise or sensor uncertainty. These measurements are used by the EKF synchronization layer, not by directly assigning the true plant state to the digital twin.

Imperfect digital twin. The digital twin is a physics-based aircraft model constructed from the same general RCAM modeling structure as the plant, but with deliberate simplifications. The superscript t denotes twin quantities. The planning twin is written as

$$x_{k+1}^t = f_t\big(x_k^t, u_{c,k}; \theta_t\big)$$

where x_k^t is the twin state and θ_t denotes fixed model parameters. The planning twin omits or simplifies effects that are present in the plant proxy, including wind disturbance and actuator lag. This creates the controlled plant–twin mismatch used to evaluate online synchronization.

Nominal landing reference. The nominal landing trajectory is generated offline using the imperfect digital twin. The resulting reference is denoted by

$$\mathcal{T}^\star = \{x_k^\star, u_k^\star\}_{k=0}^N$$

where $x_k^\star$ is the planned aircraft state and $u_k^\star$ is the planned input. At a high level, the planner solves

$$\begin{aligned} \min_{\{x_k,u_k\}_{k=0}^N,\,T} \quad & J_{\text{plan}}\big(\{x_k,u_k\}_{k=0}^N, T\big) \\ \text{s.t.} \quad & x_{k+1} = f_t(x_k,u_k;\theta_t), \qquad k=0,\dots,N-1 \\ & x_k \in \mathcal{X}_k^{\text{plan}}, \qquad u_k \in \mathcal{U}_k^{\text{plan}} \\ & x_0 = x_{\text{init}}, \qquad x_N \in \mathcal{X}_{\text{terminal}} \end{aligned}$$

The objective J_{plan} represents the trajectory-generation cost, including terms used to shape the landing path and input history. The dynamic constraint enforces propagation through the imperfect digital-twin model, while $\mathcal{X}_k^{\text{plan}}$ and $\mathcal{U}_k^{\text{plan}}$ represent the planning-stage state and input limits. The terminal set $\mathcal{X}_{\text{term}}$ defines the desired near-runway endpoint used to generate the airborne landing reference. The detailed transcription and solution method are developed in Chapter 4. The resulting nominal trajectory provides the reference path and feedforward command history that can be used by the tracking controller.

2.2 Closed-Loop Execution Problem

During execution, the plant tracks the nominal landing reference using a TVLQR gain schedule computed offline about $\mathcal{T}^\star$. The controller is evaluated on a denser controller grid than the planner grid. The reference state, command target, actuator reference, and gain schedule are evaluated using the reference progress variable s_k.

Let z_k denote the augmented tracking state used by the TVLQR controller,

$$z_k = \begin{bmatrix} x_k \\ u_{a,k} \end{bmatrix}$$

where $u_{a,k}$ contains the realized actuator states. The TVLQR command is organized as

$$u_{c,k} = u_{\text{ref}}(s_k) - K(s_k)e_k + \Delta u_{\text{sync},k} + \Delta u_{\text{terminal},k}$$

where $u_{\text{ref}}(s_k)$ is the nominal command target, $K(s_k)$ is the stored TVLQR gain, and e_k is the tracking error relative to the selected reference state. The term $\Delta u_{\text{sync},k}$ represents synchronization-based support to keep the airplane aligned with the nominal trajectory using the estimated wind disturbance. The term $\Delta u_{\text{terminal},k}$ represents terminal-policy modifications that the digital twin setup uses near touchdown.

This equation shows how the later synchronization and terminal-policy additions enter the same command structure used by the tracking controller. The baseline case corresponds to $\Delta u_{\mathrm{sync},k} = 0$ and $\Delta u_{\mathrm{term},k} = 0$. The synchronized tracking case includes $\Delta u_{\mathrm{sync},k}$, while the forecast-selected terminal-policy case also includes $\Delta u_{\mathrm{terminal},k}$. The closed-loop objective is to keep the plant close to the nominal landing reference while respecting operational constraints,

$$x_k^p \in \mathcal{X}_k, \qquad u_{c,k} \in \mathcal{U}_k$$

The set $\mathcal{X}_k$ represents the flight-envelope, runway-alignment, attitude, clearance, and terminal landing requirements. The set $\mathcal{U}_k$ represents actuator magnitude and rate limits.

2.3 Online Digital-Twin Synchronization

The synchronized digital twin is the online version of the imperfect twin after it has been updated using plant measurements. Its state is denoted by $\hat{x}_k^s$. The EKF does not estimate the full aircraft state. Instead, it estimates the reduced synchronization state

$$\chi_k = \begin{bmatrix} r_{N,k}^\top & \eta_k^\top & w_{N,k}^\top \end{bmatrix}^\top$$

where $r_N = [N, E, D]^\top$ is local NED position, $\eta = [\phi, \theta, \psi]^\top$ is Euler attitude, and $w_N = [w_N, w_E, w_D]^\top$ is the local NED wind vector. The wind estimate w_k is obtained using an Extended Kalman Filter that interprets the discrepancy between measured ground-referenced motion and twin-predicted air-relative motion as wind-related mismatch. The purpose of synchronization is to keep the online twin close enough to the plant trajectory that it can support tracking corrections and short-horizon prediction under wind-induced mismatch.

2.4 Terminal Prediction and Policy Selection

Near the terminal phase of the landing maneuver, tracking the nominal trajectory alone may not be sufficient to ensure desirable touchdown behavior. A trajectory can reach the runway region while still producing poor terminal quality, including having a high sink rate, inadequate clearance, undesirable pitch angle, or delayed touchdown. Therefore, the synchronized twin is also used as a short-horizon prediction model for future aircraft states. In this setup, the physics based synchronized digital twin evaluates a set of candidate terminal policies to predict future landing consequences before choosing the control behavior. It should

be noted that unlike [6], which uses a digital twin instance for MPC, the digital twin in this thesis uses a receding-horizon policy selection method. In another work, a simulation-based manufacturing DT is used to evaluate disruption mitigation strategies, where real-time system updates change the preferred recovery action and simulation outputs are used to train predictive models for decision optimization [22]. Thus in the present landing problem, the same decision-support principle is adapted to evaluate candidate terminal policies under the current EKF-synchronized state, and the selected policy is updated repeatedly as touchdown conditions evolve. At selector update time t_i, the synchronized augmented twin state $\hat{z}_i$ and EKF wind estimate $\hat{w}_i$ initialize a set of candidate terminal deployments. Let

$$\Pi = \{\pi_{\text{nominal}}, \pi_{\text{closure}}, \pi_{\text{sink}}, \pi_{\text{flare}}, \pi_{\text{float}}, \pi_{\text{contact}}\}$$

denote the finite set of candidate terminal policies. These policies represent different ways of modifying the terminal reference, command target, feedback authority, command limits, or additive command overlays while retaining the same underlying TVLQR tracking structure. For each candidate policy, the synchronized twin is propagated over a short prediction horizon,

$$\hat{z}^{\pi}_{i,\ell+1} = F_{\Delta t}\left(\hat{z}^{\pi}_{i,\ell}, u^{\pi}_{i,\ell}, \hat{w}_{i,\ell}; \theta_t\right) \qquad \ell = 0, \ldots, N_h - 1$$

where $F_{\Delta t}$ denotes one step of nonlinear synchronized-twin propagation, and $u^{\pi}_{i,\ell}$ is the command generated under candidate policy π. Each candidate starts from the same synchronized state, so differences in predicted terminal behavior come from the policy-dependent command behavior rather than from different initial conditions. The synchronized twin therefore evaluates each candidate policy by propagating the same current aircraft condition forward under policy-dependent command behavior. The resulting rollout is summarized by a terminal metric vector $m_i(\pi)$, which collects the predicted landing quantities used for scoring, including main-gear clearance, clearance rate, sink rate, pitch attitude, touchdown timing, touchdown location, and safety flags. The selector assigns each candidate a touchdown-quality score,

$$J_i(\pi) = \ell_{\text{td}}(m_i(\pi))$$

where ℓ_{td} is the terminal scoring function defined in the digital-twin decision-support chapter. The selected terminal policy is

$$\pi^{\star}_i = \arg\min_{\pi \in \Pi_{\text{safe},i}} J_i(\pi),$$

where $\Pi_{\text{safe},i} \subseteq \Pi$ is the set of candidate policies that pass the terminal safety checks at update time t_i. The selected policy is then mapped to the corresponding terminal modification $\Delta u_{\text{terminal},k}$ in the closed-loop command equation. This keeps the online decision problem lightweight. The expensive trajectory optimization and TVLQR gain construction are done offline, while the synchronized twin is used online to compare a small set of interpretable terminal behaviors.

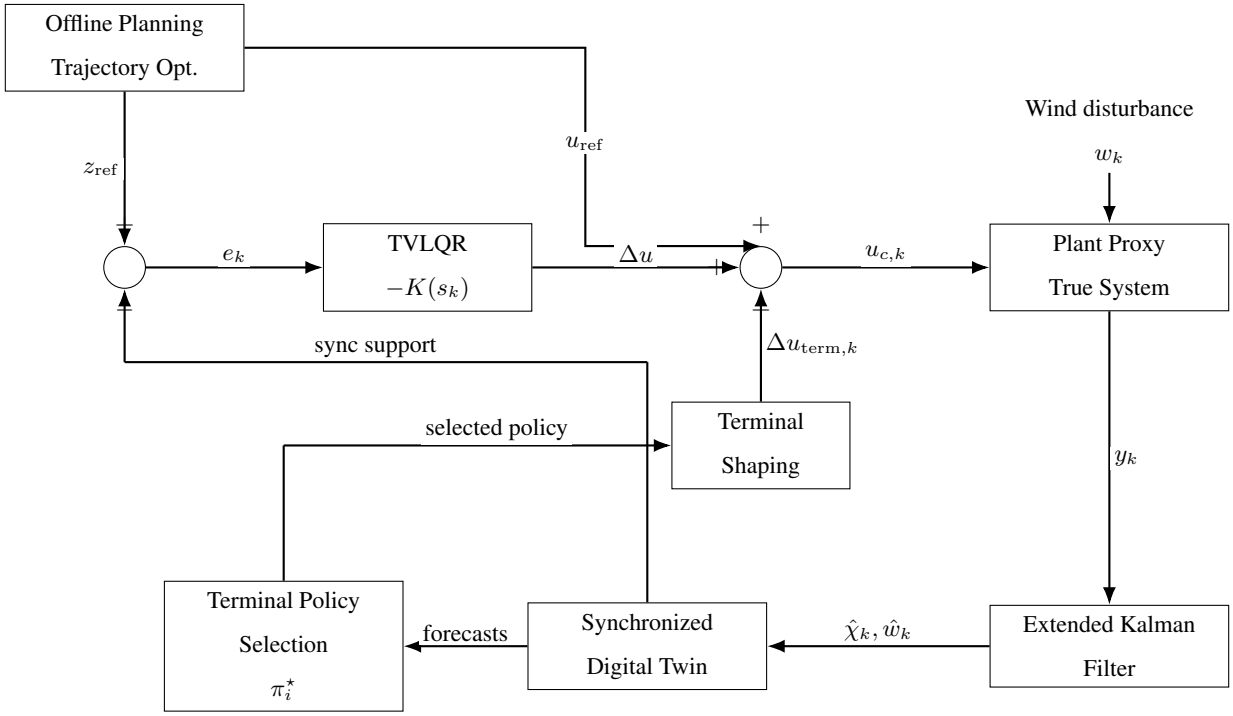


Figure 2.1: Block-diagram of the landing-control architecture. The offline planner provides the nominal state and command references, the TVLQR controller applies feedback around the scheduled reference, the EKF synchronizes the digital twin using plant measurements, and the synchronized twin supports terminal forecasting and policy selection.

Chapter 3

AIRCRAFT DYNAMICS AND DIGITAL TWIN MODELING FRAMEWORK

3.1 The Research Civil Aircraft Model (RCAM)

The physical system in this paper is the Research Civil Aircraft Model (RCAM) [23] implemented using the PSim-RCAM simulator [24]. This project uses a six-degree-of-freedom, twin-engine aircraft model developed by the Group for Aeronautical Research and Technology in Europe (GARTEUR), which serves as a proxy for a full-scale commercial transport aircraft. The Research Civil Aircraft Model specifies a standard set of parameters for a commercial transport aircraft that is similar in size, configuration, and flight regime to a Boeing 757-200. The RCAM documentation [23] includes a standardized plant model, actuators, sensors, and control architecture, providing a high-fidelity digital representation of the physical system.

The RCAM is formulated as a six-degree-of-freedom nonlinear aircraft model in a landing configuration, and it is structured to support robust control design and evaluation. The RCAM model architecture separates the aircraft rigid-body dynamics from actuator dynamics and wind/disturbance models, where actuator nonlinearities (e.g., saturations and rate limits) and wind inputs are treated as part of the simulation environment. The RCAM model is commonly represented using a block structure consisting of (i) actuator models, (ii) wind/disturbance models, and (iii) the aircraft dynamic model, with measured variables provided as outputs for control and analysis.

3.2 System Boundaries and Assumptions

The physical system is considered to be represented by the PSim-RCAM plant proxy, which includes (i) rigid-body 6-DoF aircraft dynamics (expressed in body rates, Euler angles, and body-axis velocity), (ii) propulsion effects for a twin-engine configuration, (iii) actuator dynamics mapping commanded controls to realized surface/throttle states, (iv) configuration-dependent aerodynamic increments (e.g., flap/gear/spoiler effects), and (v) a navigation/kinematics model that integrates geodetic position and altitude using a ground-referenced velocity.

System boundary. The plant proxy system boundary includes all physics that generate the logged data stream:

- **Rigid-body dynamics:** body-axis translational/rotational motion and Euler kinematics.
- **Propulsion:** engine thrust and logged propulsion states.
- **Actuation:** first-order actuator dynamics and saturation logic applied to commanded inputs.
- **Configuration logic:** high-lift and gear drag increments and ground-mode effects.
- **Navigation:** integration of $(\mathrm{lat}, \mathrm{lon}, h)$ using a ground-relative NED velocity.

Reference frames and sign conventions. All translational kinematics are expressed relative to a local Earth-fixed navigation frame $\mathcal{F}_N$ using North-East-Down (NED) coordinates. The origin of this frame is fixed at the runway reference point, with N positive downrange along the runway centerline, E positive to the right of the runway, and D positive downward. Therefore, altitude above the runway is represented as $h = -D$. The aircraft body-fixed frame $\mathcal{F}_B$ is attached to the aircraft center of mass, with x_B positive forward, y_B positive out the right wing, and z_B positive downward. The body-axis translational velocities are denoted by (u, v, w), and the body-axis angular rates are denoted by (p, q, r). The Euler angles (ϕ, θ, ψ) define the attitude of $\mathcal{F}_B$ with respect to $\mathcal{F}_N$ using the standard $3-2-1$ yaw–pitch–roll sequence, where ψ is heading/yaw, θ is pitch, and ϕ is roll.

Atmosphere and gravity assumptions. The simulation uses an atmosphere model through an altitude-dependent density $\rho(h)$ (using an ISA-style routine), rather than a fixed sea-level density. Gravity is modeled as constant magnitude $g = 9.81\ \mathrm{m/s^2}$ and is not varied with altitude. These assumptions are valid for short-duration, approach/landing flight trajectories.

Earth model assumptions. Earth rotation and curvature effects are neglected for the rigid-body dynamics. Navigation states are represented as geodetic latitude/longitude and altitude, but the propagation is driven by local NED velocities and does not attempt high-fidelity long-range geodesy. This is appropriate for terminal-area flight segments but would not be valid for long-duration navigation.

Disturbances and wind modeling. In the plant proxy, wind is applied in the navigation propagation by adding a wind vector in NED coordinates to the air-relative NED velocity before integrating $(\mathrm{lat}, \mathrm{lon}, h)$. That is, the ground velocity used in navigation is

$$\mathbf{V}_{g,\mathrm{NED}} = \mathbf{V}_{a,\mathrm{NED}} + \mathbf{V}_{w,\mathrm{NED}} \tag{3.1}$$

where $\mathbf{V}_{w,\mathrm{NED}}$ may include a constant bias plus small stochastic variations.

Sensors and measurement assumptions. The simulator produces logged signals that are treated as a time-stamped measurement stream. Because the plant proxy is simulated, these values are essentially noise-free unless noise is explicitly added in post-processing or within the synchronization algorithm. This report treats sensor noise a modeling layer that can be added to avoid unrealistic synchronization performance.

3.3 State Variables

PSim-RCAM evolves the aircraft using a split structure: a rigid-body (airframe) state integrated by the flight dynamics model and a separate navigation state integrated using ground-referenced NED velocity.

The Rigid-body aircraft state integrated by the flight dynamics integrator is

$$x_{\mathrm{rb}} = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta & \psi \end{bmatrix}^\top \tag{3.2}$$

where u, v, w are the body-axis velocity components (m/s), p, q, r are the roll/pitch/yaw body rates (rad/s), and ϕ, θ, ψ are Euler roll/pitch/yaw angles (rad). This 9-state vector captures the translational and rotational motion of the rigid body that is typically used in standard 6-DoF nonlinear flight dynamics models.

A separate navigation integrator propagates geodetic position and altitude:

$$x_{\mathrm{nav}} = \begin{bmatrix} \mathrm{lat} & \mathrm{lon} & h \end{bmatrix}^\top \tag{3.3}$$

where $\mathrm{lat}, \mathrm{lon}$ are latitude and longitude (radians in the logs), and h is altitude (m). The navigation propagation is driven by the ground-referenced NED velocity $\mathbf{V}_{g,\mathrm{NED}}$, which is formed from the air-relative NED velocity computed from $(u, v, w, \phi, \theta, \psi)$ plus wind as in (3.1).

For brevity, it is convenient to refer to the combined state as

$$x = \begin{bmatrix} x_{\mathrm{rb}} \\ x_{\mathrm{nav}} \end{bmatrix} \in \mathbb{R}^{12} \tag{3.4}$$

even though the simulator integrates (3.2) and (3.3) using separate numerical integrators.

3.4 Imperfect Digital Twin and Sources of Plant-Twin Mismatch

The digital model is defined as a physics-based, nonlinear state-space digital twin constructed from the same RCAM modeling structure as the plant, but with simplifications and assumptions that introduces a virtual-to-physical gap which would require calibration from the plant to remain accurate. The digital twin is considered to be a reduced-complexity model that runs in parallel with the plant and is to be calibrated using the measured (logged) data from the plant. This structure provides an accurate re-creation of a real Commercial Transport Airplane Plant-Twin Synchronization method, despite not having access to actual flight trajectory data from a real-world physical system. Through this setup, the digital twin remains computationally tractable while retaining enough physics to be interpretable and useful for prediction and control.

The digital twin introduces three primary sources of mismatch relative to the plant model. First, the twin assumes **zero wind**, so disturbance-driven deviations in air-relative quantities such as airspeed and aerodynamic angles appear as systematic prediction error. Second, the twin assumes **ideal actuators** (no lag), so command-to-surface/throttle dynamics present in the plant appear as transient mismatch. Finally, modeling **sensor noise** is added in the form of Gaussian noise on the measured plant outputs.

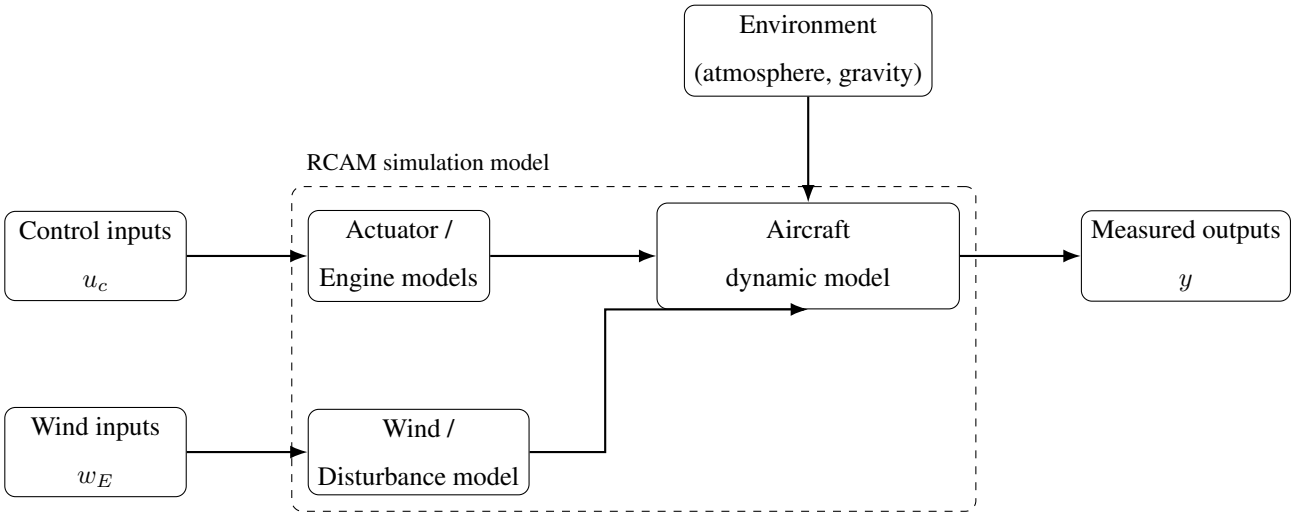


Figure 3.1: RCAM block structure: actuator/engine and wind/disturbance models feed the aircraft dynamics model under fixed environmental assumptions, producing measured outputs for control and analysis.

3.5 Mathematical Formulation

Three reference frames are used to define the 6-DoF Nonlinear Aircraft system for the digital twin model. An Earth-fixed inertial frame (E-fr), a body-fixed frame attached to the aircraft (B-fr), and an air-velocity (wind) frame aligned with the relative wind (A-fr). Gravity is assumed constant in magnitude and direction over the operating region, and Earth curvature and rotation effects are neglected (local-flight assumption). The digital twin also assumes a stationary wind field with zero wind, which is the first source of discrepancy with the plant.

$$\hat{v}_{w,E}(t) = 0 \qquad \Rightarrow \qquad \hat{v}_{a,B}(t) = \hat{v}_{g,B}(t)$$

so the air-relative velocity equals the ground-relative velocity.

Let the digital twin state be defined by the matrix:

$$\hat{x} = \begin{bmatrix} \hat{v}_B^\top & \hat{\omega}_B^\top & \hat{\eta}^\top & \hat{r}_E^\top \end{bmatrix}^\top = \begin{bmatrix} \hat{u}_B & \hat{v}_B & \hat{w}_B & \hat{p} & \hat{q} & \hat{r} & \hat{\phi} & \hat{\theta} & \hat{\psi} & \hat{x}_E & \hat{y}_E & \hat{z}_E \end{bmatrix}^\top$$

where $\hat{v}_B = [\hat{u}_B\, \hat{v}_B\, \hat{w}_B]^\top$ is the body-frame velocity, $\hat{\omega}_B = [\hat{p}\, \hat{q}\, \hat{r}]^\top$ are body rates, $\hat{\eta} = [\hat{\phi}\, \hat{\theta}\, \hat{\psi}]^\top$ are Euler angles, and $\hat{r}_E = [\hat{x}_E\, \hat{y}_E\, \hat{z}_E]^\top$ is Earth-frame position. The commanded control input is the full RCAM command vector:

$$u_c(t) \;=\; \begin{bmatrix} \delta_a(t) & \delta_e(t) & \delta_r(t) & \delta_{T1}(t) & \delta_{T2}(t) & \delta_f(t) & \delta_g(t) & \delta_{sp}(t) & \delta_b(t) \end{bmatrix}^\top$$

Since the baseline twin assumes ideal actuation (no actuator lag), the realized actuator state is approximated by a static saturation map,

$$\widehat{u}_a(t) \;=\; \mathrm{sat}\big(u_c(t)\big)$$

where $\mathrm{sat}(\cdot)$ enforces the known magnitude limits on each channel.

For the digital twin, wind is assumed to be negligible, $\hat{v}_{w,E} = 0$. Therefore, define the air-relative velocity in the body frame as $\hat{v}_{a,B} = \hat{v}_B$ and the airspeed as $\hat{V} = \|\hat{v}_{a,B}\|$. Assuming $\hat{V} > 0$ and forward flight, the digital twin computes aerodynamic angles using

$$\hat{\alpha} = \tan^{-1}\left(\frac{\hat{w}_B}{\hat{u}_B}\right), \qquad \hat{\beta} = \sin^{-1}\left(\frac{\hat{v}_B}{\hat{V}}\right)$$

These expressions define the wind (air) frame A-fr through a rotation parameterized by $(\hat{\alpha}, \hat{\beta})$, which is used to map aerodynamic forces defined in A-fr into B-fr.

Aerodynamic forces are defined in the air frame as

$$\hat{f}_{a,A} = \begin{bmatrix} -\hat{D} \\ -\hat{C} \\ -\hat{L} \end{bmatrix}, \qquad \hat{L} = \hat{q}S\,\hat{C}_L,\ \hat{D} = \hat{q}S\,\hat{C}_D,\ \hat{C} = \hat{q}S\,\hat{C}_Y$$

where S is a reference area and $\hat{C}_L, \hat{C}_D, \hat{C}_Y$ are dimensionless coefficients.

Aerodynamic moments are defined in the body frame as

$$\hat{M}_{a,B} = \begin{bmatrix} \hat{L}_m \\ \hat{M} \\ \hat{N} \end{bmatrix}, \qquad \hat{L}_m = \hat{q}Sb\,\hat{C}_l,\ \hat{M} = \hat{q}Sc\,\hat{C}_m,\ \hat{N} = \hat{q}Sb\,\hat{C}_n$$

where b and c are reference lengths and $\hat{C}_l, \hat{C}_m, \hat{C}_n$ are dimensionless moment coefficients

$$\hat{C}_m = k_{Cm}\,C_m^0(\hat{\alpha}, \hat{\beta}, \hat{u}_a)$$

For a twin-engine aircraft, thrust forces are modeled in the body frame as

$$\hat{f}_{t,B} = \sum_{k=1}^{2} \hat{f}_{t,k}, \qquad \hat{f}_{t,k} = \hat{T}_k \begin{bmatrix} \cos\epsilon_k \\ 0 \\ \sin\epsilon_k \end{bmatrix}$$

where $\hat{T}_k$ is the commanded thrust magnitude for engine k and ϵ_k is the fixed thrust inclination angle in the body frame. The net thrust moment about the center of mass is

$$\hat{M}_{t,B} = \sum_{k=1}^{2} \hat{r}_{t,k} \times \hat{f}_{t,k}$$

where $\hat{r}_{t,k}$ is the body-frame position vector of engine k relative to the aircraft center of mass.

Equations of motion (6-DoF dynamics): Let m be the aircraft mass and J the inertia matrix about the center of mass expressed in B-fr. The translational dynamics in the body frame are written as

$$\dot{\hat{v}}_B = \hat{v}_B \times \hat{\omega}_B + R_{BE}(\hat{\eta})\, g_E + \frac{1}{m}\left(R_{BA}(\hat{\alpha}, \hat{\beta})\, \hat{f}_{a,A} + \hat{f}_{t,B}\right) \tag{3.5}$$

where g_E is the constant gravity vector in E-fr (with the local sign convention), $R_{BE}(\hat{\eta})$ maps vectors from E-fr to B-fr, and $R_{BA}(\hat{\alpha}, \hat{\beta})$ maps vectors from A-fr to B-fr. The rotational dynamics in the body frame are given by

$$J\,\dot{\hat{\omega}}_B + \hat{\omega}_B \times (J\hat{\omega}_B) = \hat{M}_{a,B} + \hat{M}_{t,B} \tag{3.6}$$

Kinematics: The Earth-frame position evolves as

$$\dot{\hat{r}}_E = R_{EB}(\hat{\eta})\, \hat{v}_B \tag{3.7}$$

and the Euler angle rates are related to body rates through the expression

$$\dot{\hat{\eta}} = \mathcal{E}(\hat{\eta})\, \hat{\omega}_B \tag{3.8}$$

where $\mathcal{E}(\hat{\eta})$ is the Euler-rate transformation matrix for a 3-2-1 (yaw-pitch-roll) parameterization.

Nonlinear state-space form: By enforcing ideal actuation, the intermediate actuator dynamics are removed entirely, directly mapping commanded inputs to the control surfaces via $\hat{u}_a(t) = \mathrm{sat}(u_c(t))$. Furthermore, a zero-wind assumption is used by setting $\hat{\mathbf{v}}_{w,E}(t) \equiv \mathbf{0}$, which removes wind-driven disturbances from the navigation kinematics. Finally, to evaluate the system under realistic sensor uncertainties, an additive Gaussian noise is added into the discrete-time measurement model. Equations (3.5)–(3.8) define the twin dynamics in the compact form

$$\dot{\hat{x}}(t) = f\Big(\hat{x}(t),\, \mathrm{sat}(u_c(t)),\, \hat{\theta}\Big) \tag{3.9a}$$

$$y_k = h(\hat{x}(t_k)) + \nu_k, \qquad \nu_k \sim \mathcal{N}(0, R) \tag{3.9b}$$

where $\hat{\theta}$ collects the low-dimensional aerodynamic bias parameters and constant wind bias terms. y_k represents the sampled measurement vector at time t_k, $h(\cdot)$ is the nonlinear observation function mapping the state to the measured channels, and ν_k represents the injected Gaussian observation noise with covariance matrix R.

3.6 Digital Twin Inputs

The digital twin is controlled by the same commanded input stream $u_c(t)$ that is used by the plant proxy. This ensures that plant–twin discrepancies reflect modeling mismatch rather than input mismatch. At each sample time t_k, the digital twin receives a commanded input vector

$$u_{c,k} \;=\; \Big[\delta_{a,k} \quad \delta_{e,k} \quad \delta_{r,k} \quad \delta_{T1,k} \quad \delta_{T2,k} \quad \delta_{f,k} \quad \delta_{g,k} \quad \delta_{sp,k} \quad \delta_{b,k}\Big]^{\top} \tag{3.10}$$

where the entries correspond to the logged command channels:

$$(dA, dE, dR, dT1, dT2, \text{flap\_pos}, \text{gear\_pos}, dgsp, \text{brake})$$

In the plant proxy, commanded inputs are mapped to realized actuator states through actuator dynamics:

$$u_a(t) \;=\; \mathcal{A}(u_c(t))$$

where $\mathcal{A}(\cdot)$ includes saturation and first-order actuator response. In contrast, the digital twin used in this report assumes an Ideal Actuator which can be represented mathematically as:

$$\widehat{u}_a(t) = \mathrm{sat}(u_c(t))$$

The plant proxy includes wind in the navigation propagation using $\mathbf{V}_{g,\mathrm{NED}} = \mathbf{V}_{a,\mathrm{NED}} + \mathbf{V}_{w,\mathrm{NED}}$. The digital twin on the other hand, assumes:

$$\widehat{\mathbf{V}}_{w,\mathrm{NED}}(t) \equiv \mathbf{0} \tag{3.11}$$

so no explicit disturbance input is provided to the twin. Environmental quantities such as gravity and atmospheric density are treated as known functions/parameters (with density computed from altitude using the same atmospheric routine when the twin matches the plant environment model).

3.7 Tunable Parameters

(A) Disturbance and navigation parameters (implemented mismatch: wind): The aircraft kinematics in the Earth-fixed navigation frame are driven by the ground-relative velocity, which is the sum of air-relative velocity and wind:

$${}^{E}\dot{\mathbf{p}}(t) = \mathbf{v}_g(t) = \mathbf{v}_a(t) + \mathbf{v}_w(t) \tag{3.12}$$

consistent with standard 6-DoF nonlinear aircraft dynamics. The digital twin assumes no wind perturbations $\widehat{\mathbf{v}}_{w,\mathrm{NED}}(t) \equiv \mathbf{0}$. A tunable parameterization for synchronization is therefore a constant wind-bias vector

$$\theta_w \;=\; \mathbf{b}_w \in \mathbb{R}^3, \qquad \mathbf{v}_{g,\mathrm{NED}} = \mathbf{v}_{a,\mathrm{NED}} + \mathbf{b}_w \tag{3.13}$$

which directly targets the dominant systematic drift observed in ground-referenced channels (V_N, V_E, V_D, lat, lon, h).

(B) Actuator dynamics parameters: The simulator updates the actual surface/throttle states from commanded inputs using actuator dynamics. A tunable form is a diagonal first-order lag

$$\tau_i \dot{\delta}_i = \delta_{c,i} - \delta_i, \qquad \theta_{\mathrm{act}} = \begin{bmatrix} \tau_a & \tau_e & \tau_r & \tau_{th1} & \tau_{th2} \end{bmatrix}^\top \tag{3.14}$$

which includes magnitude limits and rate limits. In the digital twin, actuator mismatch was implemented by setting $\tau_i = 0$ to represent ideal actuation.

(C) Propulsion (engine) parameters: PSim-RCAM logs engine states such as net thrust F_n, fuel flow, spool speeds (N_1, N_2), and related deck quantities. Tunable parameters for these metrics include spool time constants, thrust efficiency scalings, as well as thrust-bias per engine:

$$\theta_{\mathrm{eng}} = \begin{bmatrix} k_{T1} & k_{T2} \end{bmatrix}^\top, \qquad \widehat{T}_k = k_{Tk}\, T_k^0 \tag{3.15}$$

(D) Observation Noise parameters: To replicate realistic sensor characteristics, the digital twin adds zero-mean Gaussian noise into the logged measurement stream. Specifically, noise is added to the geodetic position and ground-referenced velocity channels to simulate a level of sensor uncertainty. The discrete measurement vector y_k at sample step k is formed as

$$y_k = h(\hat{x}_k) + \nu_k, \qquad \nu_k \sim \mathcal{N}(0, R) \tag{3.16}$$

where the observation noise $\nu_k \in \mathbb{R}^6$ corresponds to the measured channels $[\mathrm{lat}, \mathrm{lon}, h, V_N, V_E, V_D]^\top$. The noise covariance matrix R is assumed diagonal and is parameterized by a set of tunable standard deviations targeting horizontal position, altitude, and velocity:

$$\theta_{\mathrm{obs}} = \begin{bmatrix} \sigma_{\mathrm{pos}} & \sigma_h & \sigma_v \end{bmatrix}^\top \tag{3.17}$$

For the geodetic channels, the horizontal noise parameter σ_{pos} (in meters) is converted into angular latitude and longitude deviations using a small-angle approximation scaled by the Earth's radius R_E:

$$\sigma_{\mathrm{lat}} = \frac{\sigma_{\mathrm{pos}}}{R_E}, \qquad \sigma_{\mathrm{lon}} = \frac{\sigma_{\mathrm{pos}}}{R_E \cos(\mathrm{lat})} \tag{3.18}$$

Thus, the aggregate measurement covariance is defined as $R = \mathrm{diag}(\sigma_{\mathrm{lat}}^2, \sigma_{\mathrm{lon}}^2, \sigma_h^2, \sigma_v^2, \sigma_v^2, \sigma_v^2)$. By tuning θ_{obs}, the digital twin can simulate varying levels of sensor uncertainty.

3.8 Digital Twin Outputs

The PSim-RCAM implementation produces outputs at two levels: (i) Core States integrated by the rigid-body and navigation integrators, and (ii) Derived/Diagnostic channels computed by the observation and propulsion modules. The simulator integrates the rigid-body states

$$x_{\mathrm{rb}} = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta & \psi \end{bmatrix}^\top$$

and separately integrates navigation states $x_{\mathrm{nav}} = \begin{bmatrix} \mathrm{lat} & \mathrm{lon} & h \end{bmatrix}^\top$. The mapping between body rates and Euler angle rates follows the standard 3-2-1 kinematics:

$$\dot{\phi} = p + q \sin\phi \tan\theta + r \cos\phi \tan\theta, \quad \dot{\theta} = q \cos\phi - r \sin\phi, \quad \dot{\psi} = q \frac{\sin\phi}{\cos\theta} + r \frac{\cos\phi}{\cos\theta}$$

which is the conventional relation, typically used in aircraft dynamics. Besides the airframe states, the simulator also logs ground-relative NED velocity: $\mathbf{V}_{g,\mathrm{NED}} = \begin{bmatrix} V_N & V_E & V_D \end{bmatrix}^\top$

These channels are the most sensitive to the wind/no-wind modeling choice because navigation kinematics depend on $\mathbf{v}_g = \mathbf{v}_a + \mathbf{v}_w$. The data stream also includes the commanded control inputs (and configuration states) used by the simulator:

$$u_c = \begin{bmatrix} \delta_a & \delta_e & \delta_r & \delta_{T1} & \delta_{T2} & \delta_f & \delta_g & \delta_{sp} & \delta_b \end{bmatrix}^\top$$

corresponding to aileron, elevator, rudder, engine 1/2 throttle commands, flap position, gear position, ground-spoiler command, and brake command.

Chapter 4

NOMINAL LANDING TRAJECTORY GENERATION

Generating a nominal landing trajectory for a nonlinear aircraft model can be posed as a constrained optimal control problem. The objective is to obtain a state and control trajectory that satisfies the full nonlinear aircraft dynamics, enforces operational constraints such as attitude, control limits, and path constraints, while achieving the desired terminal condition corresponding to a safe touchdown. However, due to the nonlinear and non-convex nature of the aircraft dynamics and constraints, this problem cannot generally be solved directly using standard convex optimization techniques. To address this challenge, a sequential convex programming (SCP) framework is considered, which allows for the solution of non-convex optimal control problems through a sequence of convex approximations [25, 26]. Rather than attempting to solve the original nonlinear problem directly, SCP iteratively constructs and solves locally valid convex subproblems, each of which approximates the original dynamics and constraints about a reference trajectory. This approach is computationally efficient and reliable through the use of convex optimization while at the same time being able to handle nonlinear system behavior [26, 27].

4.1 Nonlinear Optimal Control Problem and Sequential Convex Programming

Start by formulating the aircraft landing problem as a continuous-time nonlinear optimal control problem of the form

$$\begin{aligned} \min_{u(\cdot),\,p} \quad & \phi\big(x(t_f), p\big) \\ \text{s.t.} \quad & \dot{x}(t) = f\big(x(t), u(t), p\big) \\ & x(t_0) = x_0, \quad x(t_f) \in \mathcal{X}_f \\ & g_i\big(x(t), u(t), p\big) \leq 0, \quad i = 1, \ldots, m \end{aligned} \tag{4.1}$$

where $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_u}$ is the control input, and $p \in \mathbb{R}^{n_p}$ represents additional decision variables such as final time. The function $f(\cdot)$ represents the nonlinear aircraft dynamics, while the constraint functions $g_i(\cdot)$ captures path and control limitations. The terminal cost $\phi(\cdot)$ enforces the desired landing condition. Problem (4.1) is non-convex due to the nonlinear airplane dynamics discussed in previ-

ous sections and due to the presence of non-convex constraints. As a result, classical convex optimization methods are not directly applicable. Sequential convex programming addresses this challenge by replacing the original problem with a sequence of convex subproblems constructed around a reference trajectory. At iteration k, given a reference solution $(x^{(k)}(t), u^{(k)}(t))$, the nonlinear dynamics and non-convex constraints are approximated using locally valid convex models. The dynamics are linearized to obtain a time-varying affine system, and non-convex constraint functions are replaced by convex approximations, which can typically be done using first-order Taylor expansions [27]. This results in a convex optimization problem of the form

$$\begin{aligned} \min_{x(\cdot),\,u(\cdot)} \quad & \tilde{\phi}^{(k)} \\ \text{s.t.} \quad & \dot{x}(t) = A^{(k)}(t)x(t) + B^{(k)}(t)u(t) + c^{(k)}(t) \\ & \tilde{g}_i^{(k)}\big(x(t), u(t)\big) \leq 0 \\ & x(t) \in \mathcal{T}^{(k)} \end{aligned} \tag{4.2}$$

where $\mathcal{T}^{(k)}$ denotes a trust region that restricts the solution to remain close to the reference trajectory. The use of a trust region in this scenario is important as the convex approximations are only locally accurate. Without this restriction, the solution to the convex subproblem may deviate significantly from the region where the linearization is valid, potentially leading to divergence or infeasibility [28]. The SCP then proceeds iteratively where each convex subproblem is solved to obtain an updated trajectory, which becomes the reference for the next iteration. Under the right conditions, this process converges to a locally optimal solution of the original nonlinear problem [29]. However, the implementation usually requires careful handling of feasibility and convergence, particularly in the presence of nonlinear dynamics and tight constraints. These considerations motivate the use of penalized trust region methods.

4.2 The Penalized Trust Region Method

While sequential convex programming provides a systematic framework for solving nonlinear optimal control problems, its practical implementation presents several challenges. Specifically, the convex subproblems constructed at each iteration may become artificially infeasible or exhibit poor convergence behavior due to inaccuracies in the local linearization of nonlinear dynamics and constraints. These issues arise because the convex approximations are only locally valid and may not admit a feasible solution within the imposed

trust region, even when the original nonlinear problem is feasible [26]. To address these challenges, Penalized Trust Region (PTR) methods augment the SCP subproblem with slack variables and penalty terms that ensure feasibility of each convex subproblem while guiding convergence toward a solution of the original nonlinear problem [30]. Rather than enforcing the linearized dynamics and constraints exactly, PTR introduces virtual control inputs and constraint relaxation variables, which are penalized in the objective function. Specifically, the linearized dynamics can be modified as

$$\dot{x}(t) = A^{(k)}(t)x(t) + B^{(k)}(t)u(t) + c^{(k)}(t) + v(t) \tag{4.3}$$

where $v(t)$ represents a virtual control input that absorbs linearization error. Similarly, non-convex constraints can be relaxed using slack variables $\sigma_i(t) \geq 0$, giving

$$\tilde{g}_i^{(k)}\big(x(t), u(t)\big) \leq \sigma_i(t) \tag{4.4}$$

These additional variables are incorporated into the objective function through a penalty term, resulting in an augmented cost of the form

$$J = \tilde{\phi}^{(k)} + \lambda_v \int \|v(t)\|_2^2 \, dt + \lambda_\sigma \sum_i \int \sigma_i(t)^2 \, dt \tag{4.5}$$

where λ_v and λ_σ are large positive weights that discourage the use of virtual control and constraint violation.

The inclusion of virtual control provides guarantees that the linearized dynamics remain feasible at each iteration, while the penalty terms helps to ensure that these artificial control variables shrink towards zero as an assessment of whether the solution converges to one that satisfies the original nonlinear dynamics and constraints [31]. This formulation transforms the SCP subproblem into a convex optimization problem that is more feasible and better conditioned for numerical solution [25, 26]. An additional challenge in SCP methods is the selection and update of the trust region. Classical trust region methods enforce hard bounds on the deviation from the reference trajectory, which can lead to slow progress or the so-called *crawling phenomenon*, where iterates make only marginal improvements despite being far from optimality [26]. Penalized trust region methods address this issue by incorporating the trust region as a soft constraint within the objective function, allowing the optimization to adaptively balance model fidelity and progress toward optimality [30].

Building on this framework, Kim et al. introduce an extended penalized trust region method (xPTR) that is well-suited for six-degree-of-freedom nonlinear aircraft landing problems [31]. The xPTR formulation

builds upon the standard PTR approach by incorporating phase-based trajectory structure, improved scaling, and tailored penalty strategies to better handle the strong nonlinearities present in aircraft dynamics. xPTR enables robust convergence to dynamically feasible landing trajectories while maintaining computational tractability. In this work, the xPTR framework is adopted to generate a nominal landing trajectory for the nonlinear aircraft model. The method balances robustness and computational efficiency, making it well-suited for generating dynamically consistent trajectories that can then be tracked by a feedback controller.

4.3 Discretization, Convex Transcription, and PTR Formulation

The continuous-time landing trajectory optimization problem is transcribed into a finite-dimensional optimization problem using a direct shooting formulation with first-order hold (FOH) discretization. The time horizon is divided into N intervals with variable durations $\{S_k\}_{k=0}^{N-1}$, allowing for free-final-time optimization. Within each interval, control inputs are linearly interpolated, and the nonlinear aircraft dynamics are integrated using an FOH scheme to propagate the state. To allow for tractable optimization, the nonlinear 6-DoF airplane dynamics are linearized about a reference trajectory $(x_k^{(i)}, u_k^{(i)}, S_k^{(i)})$ at each iteration i. This produces affine, time-varying discrete dynamics of the form

$$x_{k+1} \approx A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + F_k S_k + c_k \tag{4.6}$$

where the matrices $(A_k, B_k^-, B_k^+, F_k, c_k)$ are obtained using numerical linearization of the FOH-integrated dynamics. This linearization allows for the construction of a convex subproblem at each iteration of the Sequential Convex Programming procedure. The resulting subproblem consists of affine equality constraints from the linearized dynamics, convex path and terminal constraints (including bounds, cone constraints, and rate limits), and a convex objective composed of quadratic and weighted norm penalties. The stage cost is formulated as a weighted quadratic penalty on particular chosen quantities, including control deviation from trim, angular rates, glide-slope deviation, airspeed tracking, and control rate usage. The quadratic penalties of the following form,

$$\|x_k - x_{\text{ref}}\|_Q^2, \qquad \|u_k - u_{\text{ref}}\|_R^2$$

are implemented using scaled least-squares terms. To maintain feasibility of the linearized subproblems, virtual control variables are introduced to relax the dynamics constraints. These slack variables are penalized

in the objective with a large weight w_{vc}, which helpts to ensure that the solution converges to a dynamically feasible trajectory as the SCP iterations progress. Additionally, trust-region constraints are imposed on the state, control, and time variables to restrict deviations from the reference trajectory, stabilizing the linearization process. The use of variable time intervals $\{S_k\}$ introduces free-final-time optimization into the framework. Convex constraints enforce bounds on individual interval durations as well as the total trajectory time, while additional structure is imposed to maintain phase-wise time consistency across the descent, approach, and flare segments. Each SCP iteration solves a convex optimization problem, which in this thesis, is solved using the python package, `CVXPY` [32]. The reference trajectory is then updated using a line search over candidate step sizes, and the process is repeated until convergence in both the objective and constraint violation metrics.

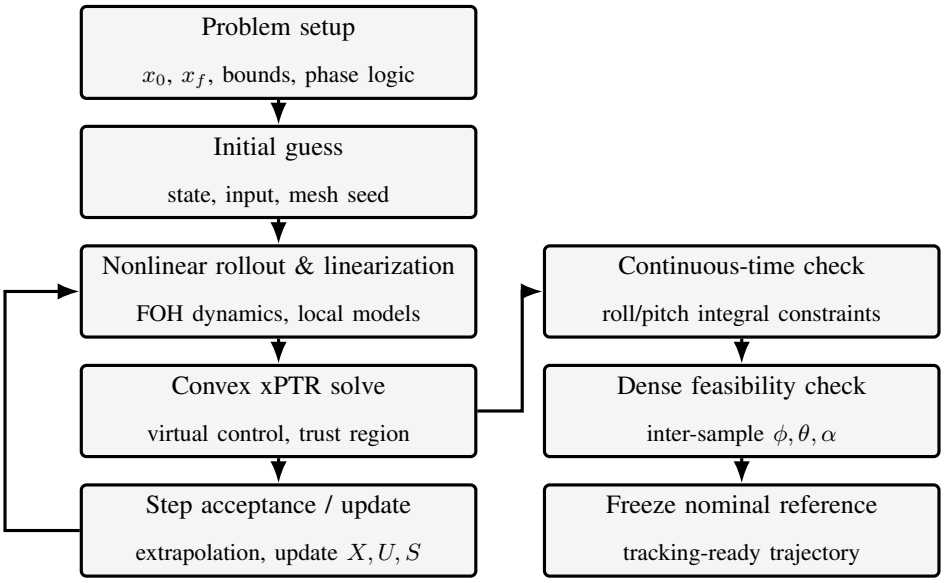


Figure 4.1: Overview of the nominal landing trajectory generation and verification. Start by defining initial and final conditions as well as constraints, generate an initial reference trajectory then at each iteration of the convex subproblem, linearize the nonlinear 6-DoF dynamics around that trajectory, build and solve a convex subproblem using the local approximations, evaluate whether the candidate solution improves the nonlinear problem, and repeat over the total number of iterations.

4.4 Initial Conditions and State-Input Description

The nominal landing planner is posed on a reduced state-input representation extracted from the full RCAM model. At each node k, the planner state is defined as

$$x_k = \begin{bmatrix} N_k & E_k & D_k & u_k & v_k & w_k & p_k & q_k & r_k & \phi_k & \theta_k & \psi_k \end{bmatrix}^\top$$

and the planner input is represented by

$$u_k^c = \begin{bmatrix} \delta_{a,k} & \delta_{e,k} & \delta_{r,k} & T_k \end{bmatrix}^\top$$

Recall that (N, E, D) denote North-East-Down position, (u, v, w) are body-axis translational velocities, (p, q, r) are body angular rates, and (ϕ, θ, ψ) are Euler angles. The reduced planner input contains aileron, elevator, rudder, and a single per-engine thrust command. Internally, this reduced input is embedded into the full RCAM control vector by applying the same thrust command to both engines while holding flap and landing-gear settings fixed during planning: $\delta_f = 0,\ \delta_g = 0$ and wind is initialized as

$$w_{\text{NED}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \ \text{m/s}$$

The landing problem is discretized over $N = 90$ intervals, so the optimization contains $N + 1 = 91$ state nodes and $N = 90$ time-interval variables,

$$X = \{x_k\}_{k=0}^{N}, \qquad U = \{u_k^c\}_{k=0}^{N}, \qquad S = \{S_k\}_{k=0}^{N-1}$$

The runway threshold is placed at the origin of the NED frame and the nominal touchdown heading is initialized as $\psi_{\text{TD}} = 0$. The initial state is imposed as a hard boundary condition so that each PTR update begin from the same prescribed approach condition. The final state x_f, however, is used both as a reference in the terminal cost and as the center of a terminal admissible set. Thus, the optimizer is encouraged to approach x_f, but it is not required to match every terminal component exactly. The initial and final boundary conditions specified for the optimizer are

$$x_0 = \begin{bmatrix} -4500.0 & 0.0 & -235.8 & 95.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -0.0524 & 0.0 \end{bmatrix}^\top$$

and

$$x_f = \begin{bmatrix} 0.0 & 0.0 & -0.10 & 85.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}^\top$$

where angles are represented in radians. These initial conditions were chosen based on a standard 3° glide slope from the initial point. From a downrange distance of $4500\ \mathrm{m}$ from the runway landing location, the initial altitude that best aligns on the 3° descent line to the runway is hence found using $4500\tan(3^\circ) \approx 235.8$. The initial and final forward speed was set to $95\ \mathrm{m/s}$ and $85\ \mathrm{m/s}$, respectively, which follows from the implementation in [31]. At the terminal node, the desired down-position was set to $D_f = -0.10\ \mathrm{m}$, corresponding to a reference-point altitude of $0.10\ \mathrm{m}$ above the runway in the NED frame. This value should not be interpreted as a main-gear touchdown height. The PTR planner does not include landing-gear contact, gear compression, or runway-contact dynamics, and the landing gear state is held fixed during nominal planning. Therefore, the terminal condition is best interpreted as a low-altitude mathematical target used to generate an airborne reference trajectory near the runway, while physically meaningful touchdown and main-gear clearance are handled later in the closed-loop tracking and digital-twin terminal-policy layers.

Trim-informed seed generation and first reference rollout. Before solving the first convex subproblem, the algorithm constructs a trim-informed initial guess for the state, input, and interval histories. The purpose of this initialization is to provide a dynamically feasible starting trajectory so that the first SCP iteration begins near a reasonable steady-descent condition rather.

A steady descent trim is first computed from the RCAM trim routine using nominal straight-in approach settings, $\gamma_{\mathrm{trim}} = -3^\circ$, $V_{A,\mathrm{trim}} = 85\ \mathrm{m/s}$, $\beta_{\mathrm{trim}} = 0$, $\phi_{\mathrm{trim}} = 0$, and $\psi_{\mathrm{trim}} = 0$ with trim altitude taken from the initialized approach condition. This produces a trimmed body-state seed x_{trim} and a corresponding trim control u^c_{trim}. These quantities provide the baseline longitudinal attitude, body-velocity components, and control quantities used to initialize the optimizer. The control seed $U^{(0)}$ is then constructed from this trim solution. Aileron and rudder are initialized at zero, while elevator and thrust are seeded from trim-based profiles,

$$u_k^{c,(0)} = \begin{bmatrix} 0 & \delta_{e,k}^{(0)} & 0 & T_k^{(0)} \end{bmatrix}^\top$$

The elevator seed is slightly relaxed near the beginning of the horizon and then returned toward the trim value over an initial portion of the mesh. This avoids beginning the optimization from an overly aggressive pitch-control while still biasing the terminal portion of the seed toward flare-like behavior. Similarly, the thrust seed is initialized from the trim thrust level so that the first input history remains smooth and physically reasonable. A phase-structured state seed $X_{\mathrm{seed}}^{(0)}$ is constructed next. Rather than interpolating directly

from the initial condition to the terminal state in one step, the initializer introduces an auxiliary flare-entry point $(N_{\mathrm{flare}}, D_{\mathrm{flare}})$, chosen from the nominal glide geometry and flare altitude. The index k_s denotes the beginning of the final-approach portion of the seed, while k_f denotes the beginning of the flare portion. These two indices partition the horizon into three conceptual regions: an early approach segment, a final-approach shaping segment, and a terminal flare segment.

For nodes prior to flare, the north position, altitude, and forward speed are interpolated from the initial condition toward the nominal flare-entry point. For nodes after k_f, the seed transitions from the flare-entry point to the terminal condition using a smoothstep map $\sigma(a) = a^2(3 - 2a)$ to avoid erratic trajectories in the terminal portion of the initial guess. The remaining lateral states and angular rates are initialized near trimmed straight-flight values, with lateral quantities set to zero. Pitch is seeded so that it remains near the trimmed descent attitude early in the horizon, is gradually adjusted through the middle portion, and is then smoothly transitioned toward the terminal pitch target in the flare portion.

For the interval history $S^{(0)}$, a geometry-based estimate of travel time is first formed by approximating the path length from the initial state to the flare-entry point and from the flare-entry point to the terminal state, using representative approach and flare speeds. This produces an initial final-time guess $t_f^{(0)}$, which is then clipped to the admissible final-time bounds. The total time is distributed across the horizon using the same phase partition $(0, k_s, k_f, N)$, with a deliberately finer mesh near touchdown.

It is important to distinguish between the *seed trajectory* returned by the initializer and the *first reference trajectory* used by SCP. The initializer returns $(X^{(0)}_{\mathrm{seed}}, U^{(0)}, S^{(0)})$, but the solver does not linearize directly about this interpolated state history. Instead, the input seed is first saturated, the interval seed is projected onto the admissible set, and the nonlinear dynamics are rolled out from the exact initial condition:

$$U^{(0)}_{\mathrm{ref}} = \mathrm{sat}\Big(U^{(0)}\Big), \qquad S^{(0)}_{\mathrm{ref}} = \Pi_{\mathcal{S}}\Big(S^{(0)}\Big), \qquad X^{(0)}_{\mathrm{ref}} = \mathrm{Rollout}\Big(x_0, U^{(0)}_{\mathrm{ref}}, S^{(0)}_{\mathrm{ref}}\Big)$$

Thus, the first convexification is performed about a dynamically propagated reference rather than about the raw interpolated seed. This distinction is important because the seed trajectory provides structural guidance, whereas the first reference trajectory provides the dynamically consistent baseline about which the initial linearization is formed.

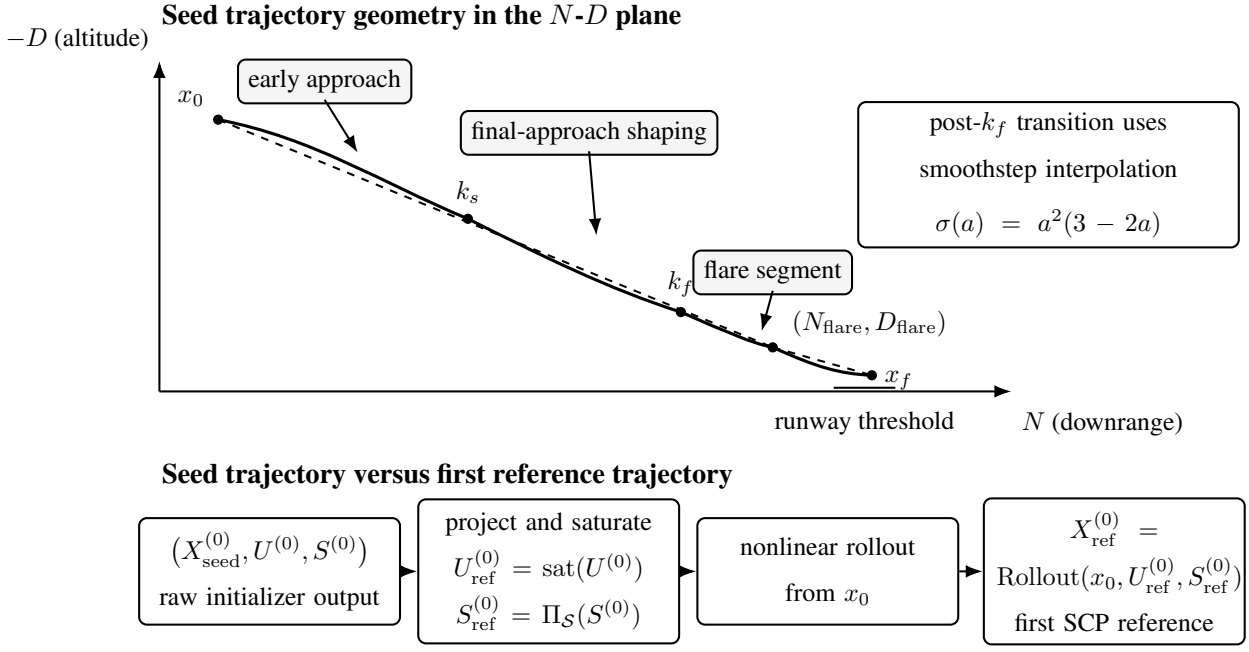


Figure 4.2: Schematic of the trim-informed initialization procedure. The top panel illustrates the phase-structured seed trajectory in the N-D plane, including the flare-entry point and the phase boundaries indexed by k_s and k_f. The bottom panel distinguishes the raw initialized seed from the first dynamically propagated reference trajectory, $(X_{\text{ref}}^{(0)})$ used for the initial SCP linearization.

4.5 Multi-Phase Landing Geometry and Constraint Design

The nominal landing planner is organized into three phases. The first phase is a pre-alignment descent segment, the second phase is a final-approach segment where runway-corridor constraints become active, and the third phase is a flare segment where tighter attitude, lateral-motion, and sink-rate restrictions are imposed. This phase structure keeps the early part of the maneuver from being over-constrained while still forcing the solution to behave like a controlled runway approach near the end of the horizon.

Let k_s denote the start of the final-approach phase and k_f denote the start of the flare phase. The corresponding phase index sets are

$$\mathcal{K}_{\text{pre}} = \{0, \ldots, k_s - 1\}, \qquad \mathcal{K}_{\text{app}} = \{k_s, \ldots, k_f - 1\}, \qquad \mathcal{K}_{\text{flare}} = \{k_f, \ldots, N\}$$

The final-approach phase is selected using altitude and downrange triggers so that the runway-alignment corridor is not imposed too early in the maneuver. This was useful because enforcing the runway corridor from the first node made the problem unnecessarily rigid and reduced the optimizer's ability to shape a

smooth descent before final approach. The flare phase is defined from the nominal glide geometry and a minimum flare range. Although the code can compute k_f from this geometry, the flare index was fixed in the present implementation so that the optimizer had more horizon length to reshape the descent and pitch behavior before reaching the terminal region.

The constraints are applied in layers. The first layer is active over the full horizon and includes bounds on body-axis velocities, angular rates, Euler angles, angle of attack, control deflections, thrust, and control-rate changes. The angle-of-attack condition is enforced using the body-axis approximation

$$\alpha_k \approx \tan^{-1}\left(\frac{w_k}{u_k}\right)$$

which is implemented through affine bounds on w_k relative to u_k. The planner also prevents the nominal trajectory from passing below the runway plane. Since the NED altitude convention gives $h = -D$, the condition $D_k \leq D_f$ keeps the reference point at or above the selected terminal height. In the current setup, $D_f = -0.10\ \mathrm{m}$, so this constraint corresponds to keeping the planner reference point at least $0.10\ \mathrm{m}$ above the runway.

Once the trajectory enters the final-approach phase, the runway-corridor constraints are enabled. In the local NED runway frame, the runway threshold is located at the origin and the aircraft approaches from negative N. The lateral and vertical corridors are represented by affine cone inequalities about the runway centerline and nominal descent path. These constraints guide the trajectory into the runway region without forcing the early descent to remain inside a narrow approach cone.

In the flare phase, the planner imposes a tighter local operating envelope. The flare constraints reduce the allowable bank angle, pitch-rate motion, lateral velocity, and vertical body-axis velocity. These restrictions are meant to prevent aggressive motion near the runway and to produce a terminal state that is suitable for closed-loop tracking. The terminal node is also constrained to lie inside a low-altitude box near the runway threshold, with bounds on terminal position, speed, attitude, and angular rates. These terminal conditions should be interpreted as a near-runway airborne reference condition, not as a main-gear touchdown model.

Table 4.1: Summary of the main constraint groups used in the nominal landing planner.

Constraint group	Representative bounds	Purpose
Full-horizon flight envelope	$75 \le u_k \le 110$ m/s, $\lvert v_k\rvert, \lvert w_k\rvert \le 12$ m/s, $\lvert\phi_k\rvert \le 25^\circ$, $\lvert\theta_k\rvert \le 15^\circ$, $-5^\circ \le \alpha_k \le 15^\circ$	Keeps the nominal trajectory inside a moderate airborne operating envelope.
Input and rate limits	$-20^\circ \le \delta_{a,k} \le 20^\circ$, $-20^\circ \le \delta_{e,k} \le 10^\circ$, $-25^\circ \le \delta_{r,k} \le 25^\circ$, $0 \le T_k \le 50{,}000$ N, with inter-node rate limits scaled by S_k	Prevents unrealistic control magnitudes and sharp command changes.
Runway corridor	Final-approach lateral corridor $\theta_{\mathrm{lat}} \in [-1.5^\circ, 2.0^\circ]$, vertical corridor $\theta_{\mathrm{ver}} \in [2.5^\circ, 3.5^\circ]$	Guides the trajectory toward the runway centerline and descent corridor after final approach begins.
Flare-phase envelope	$\lvert\phi_k\rvert \le 8^\circ$, $\lvert q_k\rvert \le 6^\circ/$s, $\lvert v_k\rvert \le 2$ m/s, $w_k \le 2.5$ m/s	Tightens the local operating envelope near the runway.
Terminal box	$\lvert N_N - N_f\rvert \le 9$ m, $\lvert E_N - E_f\rvert \le 4$ m, $D_f - 2.0 \le D_N \le D_f$, $82 \le u_N \le 87$ m/s, with small terminal attitude and rate bounds	Defines a low-altitude, aligned handoff condition rather than a literal gear-contact state.
Time mesh	$0.30 \le S_k \le 1.25$ s, $42 \le \sum_{k=0}^{N-1} S_k \le 72$ s, with finer phase-level spacing near flare	Allows free-final-time optimization while keeping more resolution near the terminal region.

The time mesh also follows the phase structure. Within each phase, the interval duration is held uniform, and the phase-level time steps are constrained so that the mesh becomes no coarser as the trajectory approaches the runway. This gives the flare and terminal portions more temporal resolution, where the dynamics and constraints are more sensitive. The planner does not include landing-gear contact, gear compression, runway reaction forces, or ground-contact dynamics. Therefore, the terminal value $D_f = -0.10$ m is not a main-gear clearance target. It is a reference-point altitude used to pull the nominal trajectory toward the runway region while keeping the planning problem airborne. Physically meaningful touchdown behavior is handled later during closed-loop tracking and digital-twin terminal evaluation, where main-gear clearance, clearance rate, sink rate, and contact geometry are evaluated directly.

4.6 Objective Function and Weighting Strategy

The objective function combines trajectory-shaping terms with algorithmic terms used by the xPTR solver. The trajectory-shaping terms define the desired nominal landing behavior, while the trust-region and virtual-

control terms stabilize the convexification process. To keep the different state and input channels comparable, the implementation uses scaled quadratic penalties of the form

$$\|e\|_\Sigma^2 := \|\Sigma^{-1}e\|_2^2 = \sum_i \left(\frac{e_i}{\sigma_i}\right)^2$$

where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$. Following the scaling approach used in [31], the scale values are generated from the current reference trajectory and lower-bounded by fixed floor values. This prevents small-magnitude channels from becoming artificially dominant in the least-squares penalties.

The performance portion of the objective is written as

$$J_{\mathrm{perf}} = J_{\mathrm{time}} + J_{\mathrm{stage}} + J_{\mathrm{terminal}}$$

where the free-final-time term is

$$J_{\mathrm{time}} = w_{\mathrm{time}} \sum_{k=0}^{N-1} S_k$$

This term mildly encourages shorter maneuver duration, but its weight is kept small so that time minimization does not dominate the landing geometry or terminal behavior. The stage cost is made up of control, rate, attitude-rate, lateral-centering, speed, glide-slope, and flare-shaping terms. Let

$$\omega_k = \begin{bmatrix} p_k & q_k & r_k \end{bmatrix}^\top, \qquad u_k^c = \begin{bmatrix} \delta_{a,k} & \delta_{e,k} & \delta_{r,k} & T_k \end{bmatrix}^\top$$

and let $u_{k,\mathrm{ref}}^c$ denote the trim-informed reference control profile. The control-deviation penalty is

$$J_{\mathrm{ctrl}} = \sum_{k=0}^{N} \left[w_{\mathrm{ctrl,ref}} \left\| \begin{bmatrix} \delta_{a,k} \\ \delta_{e,k} \\ \delta_{r,k} \end{bmatrix} - \begin{bmatrix} \delta_{a,k}^{\mathrm{ref}} \\ \delta_{e,k}^{\mathrm{ref}} \\ \delta_{r,k}^{\mathrm{ref}} \end{bmatrix} \right\|_{\Sigma_{\mathrm{surf},k}}^2 + w_{\mathrm{thr}} \left(\frac{T_k - T_k^{\mathrm{ref}}}{\sigma_{T,k}} \right)^2 \right]$$

The surface deflections are penalized together, while thrust is weighted separately. During tuning, the thrust-deviation weight was kept relatively small because heavier thrust penalties made the optimizer preserve trim thrust too aggressively and reduced its ability to manage energy during the terminal part of the maneuver.

The other stage penalties are

$$J_\omega = \sum_{k=0}^{N} w_\omega \, \|\omega_k\|^2_{\Sigma_{\omega,k}} \tag{4.7}$$

$$J_{\text{lat}} = \sum_{k=0}^{N} w_{\text{lateral}} \left(\frac{E_k}{\sigma_E}\right)^2 \tag{4.8}$$

$$J_{\text{speed}} = \sum_{k=0}^{N} w_{\text{speed}} \left(\frac{u_k - u_k^{\text{ref}}}{\sigma_u}\right)^2 \tag{4.9}$$

$$J_{\text{glide}} = \sum_{k \in \mathcal{K}_{\text{pre}} \cup \mathcal{K}_{\text{app}}} w_{\text{glide}} \left(\frac{e_{g,k}}{\sigma_D}\right)^2 \tag{4.10}$$

$$J_{\text{flare}} = \sum_{k \in \mathcal{K}_{\text{flare}}} \left[w_{\text{flare,sink}} \left(\frac{w_k - w_{\text{flare,ref}}}{\sigma_w^{\text{flare}}}\right)^2 + w_{\text{flare,pitch}} \left(\frac{\theta_k - \theta_f}{\sigma_\theta^{\text{flare}}}\right)^2 \right] \tag{4.11}$$

$$J_{\text{rate}} = \sum_{k=0}^{N-1} w_{\text{rate}} \left\| u_{k+1}^c - u_k^c \right\|^2_{\Sigma_{\Delta u,k}} \tag{4.12}$$

Here J_ω suppresses unnecessary angular-rate motion, J_{lat} keeps the trajectory near the runway centerline, and J_{speed} shapes the longitudinal energy state. The speed reference is $u_k^{\text{ref}} = 85\ \text{m/s}$ outside the flare phase and $u_k^{\text{ref}} = 82\ \text{m/s}$ during flare. The glide-slope error is defined as

$$e_{g,k} = D_k - \big(\tan(\gamma_g) N_k + b_g\big), \qquad b_g = D_0 - \tan(\gamma_g) N_0, \qquad \gamma_g = 3^\circ$$

This term is a soft path-shaping penalty rather than a hard constraint. It biases the solution toward a conventional descent path before the final-approach corridor and terminal constraints become dominant. During flare, the glide penalty is replaced by the sink-rate and pitch-shaping penalties in J_{flare}. These terms encourage a shallower terminal state and reduce the descent rate near the runway, but they do not represent a full touchdown-contact model. The rate penalty is normalized by a scale vector based on actuator rate limits and the local reference interval length,

$$\Sigma_{\Delta u,k} = \text{diag}\Big(\dot{\delta}_{a,\max} S_k^{\text{ref}}, \dot{\delta}_{e,\max} S_k^{\text{ref}}, \dot{\delta}_{r,\max} S_k^{\text{ref}}, \dot{T}_{\max} S_k^{\text{ref}}\Big)$$

This makes the penalty relative to the amount of command change that is reasonable over the current time step. The terminal cost is applied at the last node and shapes the trajectory toward a controlled low-altitude terminal condition:

$$J_{\text{terminal}} = w_{\text{terminal,pos}}\|x_{N,0:3} - x_{f,0:3}\|^2_{\Sigma_{\text{pos}}} + w_{\text{terminal,vel}}\|x_{N,3:6} - x_{f,3:6}\|^2_{\Sigma_{\text{vel}}} + w_{\text{terminal,att}}\|x_{N,9:12} - x_{f,9:12}\|^2_{\Sigma_{\text{att}}} + 0.25\, w_{\text{terminal,att}}\|x_{N,6:9} - x_{f,6:9}\|^2_{\Sigma_{\text{rate}}} \tag{4.13}$$

The scale vectors are taken from the terminal admissible box used in the constraints:

$$\Sigma_{\text{pos}} = \text{diag}(9,\ 4,\ 2)\ \text{m}, \qquad \Sigma_{\text{vel}} = \text{diag}(2,\ 1,\ 1.5)\ \text{m/s},$$

$$\Sigma_{\text{att}} = \text{diag}(3^\circ,\ 2.5^\circ,\ 1^\circ), \qquad \Sigma_{\text{rate}} = \text{diag}(2.5^\circ/\text{s},\ 2.5^\circ/\text{s},\ 2.5^\circ/\text{s}).$$

Using the same scales as the terminal box makes the terminal penalty consistent with how terminal accuracy is judged. The position term carries the largest weight because the nominal should end near the runway threshold and within the selected low-altitude band. The velocity, attitude, and angular-rate terms keep the endpoint suitable for the later tracking phase.

Landing Performance Cost. The performance objective minimized inside the convex subproblem is therefore

$$J_{\text{perf}} = J_{\text{time}} + J_{\text{ctrl}} + J_{\omega} + J_{\text{lat}} + J_{\text{speed}} + J_{\text{glide}} + J_{\text{flare}} + J_{\text{rate}} + J_{\text{terminal}}$$

This cost defines the desired nominal trajectory behavior. It penalizes excessive maneuver time, control deviation, angular-rate motion, lateral offset, speed error, glide-slope error, aggressive terminal descent, and terminal-state error. It should be interpreted as a shaping objective for an airborne nominal reference, not as a complete landing-contact objective.

Convex Subproblem Optimization. At each xPTR iteration, the solver minimizes an augmented surrogate objective around the current reference trajectory. The optimization variables include the state, input, interval durations, discrete-time virtual-control variables, and continuous-time relaxation variables. The convex subproblem objective is

$$J_{\text{sub}} = J_{\text{perf}} + J_{\text{TR}} + J_{\text{VC}}$$

where the trust-region penalty is

$$J_{\text{TR}} = \sum_{k=0}^{N} \left[w_{\text{tr},x}\, \|x_k - x_k^{\text{ref}}\|^2_{\Sigma_{x,k}} + w_{\text{tr},u}\, \|u_k^c - u_{k,\text{ref}}^c\|^2_{\Sigma_{u,k}} \right] + \sum_{k=0}^{N-1} w_{\text{tr},s} \left(\frac{S_k - S_k^{\text{ref}}}{\sigma_s} \right)^2$$

and the virtual-control penalty is

$$J_{\mathrm{VC}} = \sum_{k=0}^{N-1} w_{\mathrm{vc}} \left\| v_k \right\|_{1,\Sigma_{x,k}} + \sum_{k=0}^{N-1} w_{\mathrm{vc}} \left\| v_k^{\mathrm{ct}} \right\|_{1,\Sigma_{\mathrm{ct},k}}$$

The variable v_k relaxes the discrete-time linearized dynamics, while v_k^{ct} relaxes the continuous-time roll and pitch integral constraints. The performance cost gives the optimizer its trajectory objective, the trust-region term keeps the update near the current linearization, and the virtual-control term keeps the convex subproblem feasible while still penalizing deviations from the nonlinear dynamics and continuous-time constraints.

Nonlinear Merit Function. The convex subproblem is only a local approximation, so a lower value of J_{sub} does not automatically mean that the nonlinear trajectory has improved. After each candidate update is generated, the solver rolls the candidate through the nonlinear model and evaluates a separate nonlinear merit function:

$$M(X,U,S) = J_{\mathrm{time}} + J_{\mathrm{terminal}} + J_{\omega} + J_{\mathrm{speed}} + J_{\mathrm{ctrl}} + J_{\mathrm{rate}} + w_{\mathrm{merit,path}} \Big(V_{\mathrm{path,node}} + w_{\mathrm{ct,path}} V_{\mathrm{path,ct}} \Big) \tag{4.14}$$

This merit function is used to decide whether a candidate step should be accepted. It is important to note that the merit function is not identical to the landing performance cost J_{perf}. The lateral-centering, glide-slope, and flare-shaping terms are left out so that acceptance is based mainly on terminal quality, major state and input behavior, and path feasibility. This prevents the line search from rejecting otherwise useful progress simply because one soft shaping term did not improve.

Thus, the convex subproblem uses J_{perf} to aim the candidate toward a useful nominal landing trajectory. The trust-region and virtual-control terms keep the local approximation numerically manageable. The candidate is then propagated through the nonlinear model, and the merit function decides whether the step should be accepted.

The final weights used in this thesis are listed in Table 4.2. These values were obtained through repeated tuning of the coupled cost terms. The most consequential choices were the reduced thrust-deviation penalty, the stronger glide and flare-shaping penalties, and the relatively large terminal position and attitude weights. The solution remains sensitive to weight selection, which is a limitation of the current formulation. This

sensitivity is not surprising because the planner is solving a tightly constrained nonlinear landing problem while intentionally avoiding explicit ground-contact modeling.

Table 4.2: Selected objective and merit weights used in the nominal landing planner.

Quantity	Value	Quantity	Value	Quantity	Value
w_{time}	0.02	w_{lateral}	2.0	$w_{\text{terminal,pos}}$	350.0
$w_{\text{ctrl,ref}}$	10.0	w_{glide}	75.0	$w_{\text{terminal,vel}}$	180.0
w_{thr}	0.20	w_{speed}	10.0	$w_{\text{terminal,att}}$	200.0
w_{rate}	40.0	$w_{\text{flare,sink}}$	65.0	$w_{\text{merit,path}}$	2.0×10^3
w_{ω}	15.0	$w_{\text{flare,pitch}}$	35.0	$w_{\text{ct,path}}$	25.0

4.7 Acceptance Logic, Merit Function, and Adaptive Penalties

After each convex subproblem is solved, the solver performs a post-solution line search on the candidate update. Let $(X^{\text{ref}}, U^{\text{ref}}, S^{\text{ref}})$ denote the current reference trajectory and let $(\hat{X}, \hat{U}, \hat{S})$ denote the solution of the convexified subproblem. The algorithm then forms trial updates of the form

$$U^{\text{trial}} = U^{\text{ref}} + \alpha\,\gamma_{\text{eff}}\left(\hat{U} - U^{\text{ref}}\right), \qquad S^{\text{trial}} = S^{\text{ref}} + \alpha\,\gamma_{\text{eff}}\left(\hat{S} - S^{\text{ref}}\right)$$

and obtains the corresponding state history from a nonlinear rollout,

$$X^{\text{trial}} = \text{Rollout}\left(x_0, U^{\text{trial}}, S^{\text{trial}}\right)$$

The step size α is selected from a discrete candidate set:

$$\alpha \in \{1.00,\ 0.75,\ 0.50,\ 0.25,\ 0.10,\ 0.05,\ 0.02,\ 0.01\}$$

and each candidate is evaluated using the nonlinear merit function (Eq. (4.14)) described above rather than the convex subproblem objective. Because the convexified subproblem is only a local approximation, a decrease in J_{sub} does not guarantee improvement of the underlying nonlinear landing trajectory. Thus, each candidate update is re-propagated through the nonlinear model and scored using the nonlinear merit function M. A trial step is accepted when it produces a sufficient decrease in merit relative to the current reference.

The α values reported in the convergence history should be interpreted as accepted line-search step sizes, rather than as trust-region radii.

The implementation also includes a *soft acceptance* mechanism for cases in which strict merit decrease is not obtained but the candidate still improves the terminal metrics used to judge the nominal solution. If no trial step satisfies the nominal decrease criteria, the best available small step may still be accepted provided that its merit is nearly unchanged and it improves at least one main metric, such as terminal position error, altitude violation, or virtual-control magnitude. This helps prevent the solver from rejecting useful late-iteration progress when the problem becomes numerically flat even though the near-runway terminal condition is still being refined.

The penalty weights associated with virtual control and trust-region regularization were updated after each iteration. Let w_{vc}, $w_{\mathrm{tr},x}$, $w_{\mathrm{tr},u}$, and $w_{\mathrm{tr},s}$ denote the effective penalties on virtual control, state deviation, input deviation, and time-interval deviation, respectively. If an iteration is rejected entirely, the penalty values are increased to push the next convex subproblem to remain closer to the current reference. If a step is accepted but exhibits poor progress, meaning that the accepted step size is small or the virtual-control norm fails to decrease, the same penalty increase is applied. On the other hand, when an accepted step shows good progress, the penalties are relaxed.

The extrapolation factor is updated in parallel with these penalties. The nominal value is $\gamma_{\mathrm{eff}} = 1$. When an accepted step has both a reasonably large α and a substantial reduction in virtual-control magnitude, the next iteration is allowed to use a slightly larger extrapolation factor, capped at $\gamma_{\mathrm{eff}} = 1.20$.

This allows the next iteration to take a slightly more aggressive step when the local convex model appears trustworthy. If progress is poor or a step is rejected, the extrapolation is reset to $\gamma_{\mathrm{eff}} = 1$.Finally, the algorithm terminates when repeated rejections indicate that the solve has plateaued near an acceptable terminal condition. In this case, the merit improvement is negligible, the trust-region step is already very small, the virtual-control terms are nearly zero, and the terminal state lies inside the practical near-runway envelope. This stopping logic is useful because, in the landing problem considered here, late iterations often yield very small numerical improvements even after the trajectory is already operationally adequate.

4.8 Continuous-Time Constraint Satisfaction and Inter-Sample Verification

A nodewise transcription alone does not guarantee that path constraints remain satisfied between adjacent grid points. This is especially important near the terminal portion of the approach, where the aircraft dynamics are more sensitive to nonlinear effects, the time mesh is locally refined, and the allowable attitude envelope becomes tighter. A trajectory can therefore appear feasible at the nodes while still violating roll or pitch limits between them. To address this issue, the present implementation incorporates the continuous-time integral reformulation framework developed in the xPTR framework of [31] and then performs a separate dense inter-sample verification on the final accepted nominal trajectory.

For each interval $[t_k, t_{k+1}]$, let the continuous-time attitude bound vector be written as

$$b_k^{\mathrm{ct}} = \begin{bmatrix} \phi_{\lim}^{(k)} & \phi_{\lim}^{(k)} & \theta_{\lim} & \theta_{\lim} \end{bmatrix}^\top$$

where

$$\phi_{\lim}^{(k)} = \begin{cases} \phi_{\max} = 25^\circ, & k < k_f \\ \min(\phi_{\max}, \phi_{\mathrm{flare}}) = 8^\circ, & k \geq k_f \end{cases}, \qquad \theta_{\lim} = \theta_{\max} = 15^\circ$$

Thus, the continuous-time pitch bound is the same global pitch bound used at the nodes, while the roll bound is automatically tightened in the flare phase to match the phase-specific bank-angle restriction. The corresponding continuous-time violation function is

$$g_{\mathrm{ct}}(x(t), k) = \begin{bmatrix} \phi(t) - \phi_{\lim}^{(k)} \\ -\phi(t) - \phi_{\lim}^{(k)} \\ \theta(t) - \theta_{\lim} \\ -\theta(t) - \theta_{\lim} \end{bmatrix}$$

The interval is feasible in continuous time iff all four components remain nonpositive for the entire interval. Following [31], this condition is enforced through the nonconvex constraint

$$I_k := \int_{t_k}^{t_{k+1}} \max\big(g_{\mathrm{ct}}(x(t), k), 0\big)^2 \, dt = 0$$

This integral is approximated numerically using the nonlinear model and a first-order-hold (FOH) control interpolation between adjacent nodes. Let

$$u(t) \approx (1-\tau)u_k + \tau u_{k+1}, \qquad \tau = \frac{t-t_k}{S_k}$$

over the interval of duration $S_k = t_{k+1} - t_k$. Each interval is subdivided into $n_{\mathrm{ct}} = 5$ substeps during optimization, and the state is propagated sequentially with the nonlinear FOH step operator. Writing x_j and x_{j+1} for the propagated states at the ends of the j-th substep, the integral gets approximated using a trapezoidal accumulation:

$$I_k \approx \sum_{j=0}^{n_{\mathrm{ct}}-1} \frac{\Delta t_k}{2n_{\mathrm{ct}}} \left[\max\big(g_{\mathrm{ct}}(x_j,k),0\big)^2 + \max\big(g_{\mathrm{ct}}(x_{j+1},k),0\big)^2 \right]$$

This construction is meant to replicate the continuous time constraint formulated in [31]. It evaluates the actual nonlinear trajectory inside each interval rather than only relying on a nodal approximation. The resulting interval-integral map:

$$I_k = I_k(x_k, u_k, u_{k+1}, S_k)$$

is itself nonlinear, so it is linearized inside each SCP iteration about the current reference trajectory. Specifically, finite-difference Jacobians are computed with respect to the interval initial state, the left and right endpoint controls, and the interval duration, producing an affine approximation of the form

$$I_k(x_k, u_k, u_{k+1}, S_k) \approx A_k^{\mathrm{ct}} x_k + B_{m,k}^{\mathrm{ct}} u_k + B_{p,k}^{\mathrm{ct}} u_{k+1} + z_k^{\mathrm{ct}} S_k + c_k^{\mathrm{ct}}$$

The convex subproblem then enforces

$$A_k^{\mathrm{ct}} x_k + B_{m,k}^{\mathrm{ct}} u_k + B_{p,k}^{\mathrm{ct}} u_{k+1} + z_k^{\mathrm{ct}} S_k + c_k^{\mathrm{ct}} + v_k^{\mathrm{ct}} = 0$$

where v_k^{ct} is a continuous-time virtual-control term. As with the discrete dynamic virtual control, this term is not part of the desired physical solution and is used to preserve feasibility of the convexified subproblem while the linearization is still imperfect. The continuous-time virtual-control term is penalized in the augmented objective using the same large penalty weight w_{vc} used for the dynamic virtual control, so the optimization drives it toward zero. To keep the continuous-time penalty well scaled across intervals of different lengths and across attitude bounds of different magnitudes, each interval-integral component is normalized using

$$\Sigma_{I,k} = \max\Big(\big(b_k^{\mathrm{ct}}\big)^{\circ 2}\, S_k^{\mathrm{ref}},\ 10^{-8} \Big)$$

where S_k^{ref} is the current reference interval duration. This prevents the continuous-time integral terms from becoming numerically dominant just because a particular interval is long or because its allowable bound is small. The same normalized interval-integral quantity is also included in the nonlinear merit evaluation through the continuous-time path-violation term

$$V_{\mathrm{ct}}(X,U,S) = \sum_{k=0}^{N-1} \left\| \frac{I_k(X,U,S)}{\Sigma_{I,k}} \right\|_2^2$$

The continuous-time constraints influence the solution through the affine equality constraints with virtual control inside each convex subproblem, and also through the nonlinear merit function used to accept or reject candidate updates.

Dense Inter-Sample Verification. The continuous-time integral constraints influence the optimization, but the final accepted trajectory is still checked separately after convergence. This post-solution check is used to confirm that the nonlinear trajectory does not develop inter-sample roll, pitch, or angle-of-attack violations before it is passed to the tracking controller. To address this, the continuous-time integral reformulation from the xPTR framework for roll and pitch is implemented, and then a separate dense nonlinear inter-sample verification is performed on the final accepted nominal trajectory [31].

After the xPTR solver terminates, the final accepted nominal trajectory is subjected to a separate dense inter-sample feasibility check before being used as the reference for closed-loop tracking. Each interval is re-propagated through the nonlinear model using first-order-hold control interpolation and $n_{\mathrm{dense}} = 20$ subsamples per interval, resulting in a dense continuous-time approximation of the nominal over the full horizon. At each dense sample, the verification evaluates roll, pitch, and angle of attack using the same phase-dependent roll limits and global pitch and angle-of-attack bounds imposed during planning. Unlike the continuous-time integral terms that enter the xPTR subproblems and affect the computed solution, the dense verification step evaluates the accepted nominal trajectory on a finer grid. The dense verification checks

$$|\phi(t)| \le \begin{cases} 25^\circ, & t \in \text{pre-flare}, \\ 8^\circ, & t \in \text{flare}, \end{cases} \qquad |\theta(t)| \le 15^\circ, \qquad -5^\circ \le \alpha(t) \le 15^\circ$$

where the angle of attack is reconstructed from the propagated body-axis velocity as

$$\alpha(t) = \tan^{-1}\left(\frac{w(t)}{u(t)}\right)$$

This dense inter-sample verification step serves as the final check before the nominal landing trajectory is passed to the tracking controller. In this thesis, the nominal trajectory is treated as tracking-ready only after this dense check confirms that roll, pitch, and angle of attack remain within their prescribed bounds.

Algorithm 1: Nominal landing trajectory generation using xPTR

Input: x_0, x_f, geometry bounds, initial weights, iteration limits
Output: Nominal trajectory $(X^\star, U^\star, S^\star)$
Initialize trim-informed guess $(X^{(0)}, U^{(0)}, S^{(0)})$ and penalty weights
for $i = 0, 1, \ldots, i_{\max}$ **do**
  `// 1. Convexification`
  Linearize FOH dynamics about $(X^{(i)}, U^{(i)}, S^{(i)})$ (Eq. 4.6)
  `// 2. xPTR Subproblem`
  Solve convex subproblem for candidate $(\bar{X}, \bar{U}, \bar{S})$ using virtual control V
  `// 3. Line Search & Extrapolation`
  accepted $\leftarrow$ false
  foreach $\alpha \in \mathcal{A}$ **do**
    $(X_\alpha, U_\alpha, S_\alpha) \leftarrow (X^{(i)}, U^{(i)}, S^{(i)}) + \alpha\gamma\big((\bar{X}, \bar{U}, \bar{S}) - (X^{(i)}, U^{(i)}, S^{(i)})\big)$
    Project S_α onto admissible mesh and evaluate nonlinear merit $M(X_\alpha, U_\alpha, S_\alpha)$
    if *sufficient merit decrease* **then**
      $(X^{(i+1)}, U^{(i+1)}, S^{(i+1)}) \leftarrow (X_\alpha, U_\alpha, S_\alpha)$
      accepted $\leftarrow$ true; **break**
    end
  end
  `// 4. Adaptive Penalty Update`
  if accepted **then**
    Relax penalties $w_{\mathrm{vc}}, w_{\mathrm{tr}}$; increase step aggressiveness γ
  else
    Increase penalties $w_{\mathrm{vc}}, w_{\mathrm{tr}}$; reset γ and retain reference
  end
  `// 5. Stopping Criteria`
  if $\|V\|_\infty \le \varepsilon_{\mathrm{vc}}$ ***and*** *trust-region error* $\le \varepsilon_{\mathrm{tr}}$ ***and*** *terminal tolerances met* **then**
    break
  end
end
`// 6. Verification`
Verify continuous-time feasibility via dense inter-sample rollout
return $(X^\star, U^\star, S^\star) \leftarrow (X^{(i)}, U^{(i)}, S^{(i)})$

4.9 Nominal Trajectory Results

The PTR nominal trajectory provides the terminal state of the optimized landing reference, including final position, body-axis velocity, angular rates, and attitude. Therefore quantities such as terminal pitch and body-axis vertical velocity are available directly from the PTR solution, while inertial sink rate must be computed from the terminal velocity and attitude. The final converged solution was obtained in eight PTR iterations and achieved a final horizon of $t_f^\star = 50.723$ s with final speed $u_f = 84.755$ m/s, the terminal

pitch is approximately $\theta_f = -1.999^\circ$, the terminal CG height is approximately 0.109 m above the runway, and the derived inertial sink rate is approximately 3.61 m/s downward. The final state errors in the NED frame had $dN = -1.643$m, $dE = 0.000$m, and $dD = -0.009m$. The dN error is seen to be larger than the other two states, and it was found during weight tuning that specifying the terminal state of the aircraft to slightly above the runway resulted in a greater dN error, but resulted in much better convergence behavior. These results indicate that the trajectory is sufficiently accurate to serve as a nominal reference for a tracking controller to be applied. Simulation parameters used for initial and final conditions as well as specified constraints are shown in Table 4.3.

Table 4.3: Selected geometric, solver, and constraint parameters for the nominal landing problem.

Category	Parameters
Initial conditions	$N_0 = -4500$ m, $-D_0 = 235.8$ m, $u_0 = 95.0$ m/s
Terminal targets	$D_f = -0.10$ m, $u_f = 85.0$ m/s
Approach geometry	Glide slope $= 3.0^\circ$, FA min range $= 1400$ m
Flare geometry	Height $= 30$ m, Min range $= 420$ m
Global bounds	Time $\in [42, 72]$ s, Speed $\in [75, 110]$ m/s
Flight envelope bounds	$\lvert\phi\rvert \leq 25^\circ$, $\lvert\theta\rvert \leq 15^\circ$, $\alpha \in [-5^\circ, 15^\circ]$
Solver & grid settings	Intervals $N = 90$, CT substeps $= 5$, Dense samples $= 20$

Table 4.4: Summary of the final converged nominal landing trajectory.

Category	Result
Solver performance	Iterations $= 8$, Total time $= 50.723$ s
Terminal position error	$dN = -1.643$ m, $dE \approx 0.000$ m, $dD = -0.009$ m
Terminal dynamics	$u_f = 84.755$ m/s, $\theta_f = -1.999^\circ$
Dense verification	$\lvert\phi\rvert_{\max} = 0.000^\circ$, $\lvert\theta\rvert_{\max} = 5.140^\circ$, $\alpha \in [-1.639^\circ, 1.063^\circ]$, Violations $= 0$

Figure 4.3 shows the final ground track and approach profile. The trajectory remains centered on the runway and captures the nominal glide geometry before entering the flare region. Figure 4.4 shows the corresponding body velocities, Euler angles, control histories, control deviations relative to trim, and time-interval dilation. The aircraft remains wings-level throughout the maneuver, the pitch angle stays modestly

negative, and the optimizer selects a nearly constant thrust level during the descent.

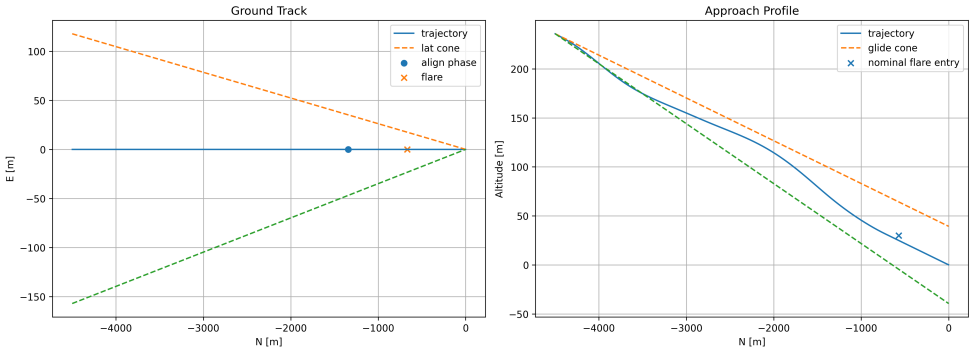


Figure 4.3: Ground-track and approach-profile views of the final nominal landing trajectory. The aircraft remains on the runway centerline and transitions from the approach into the flare region.

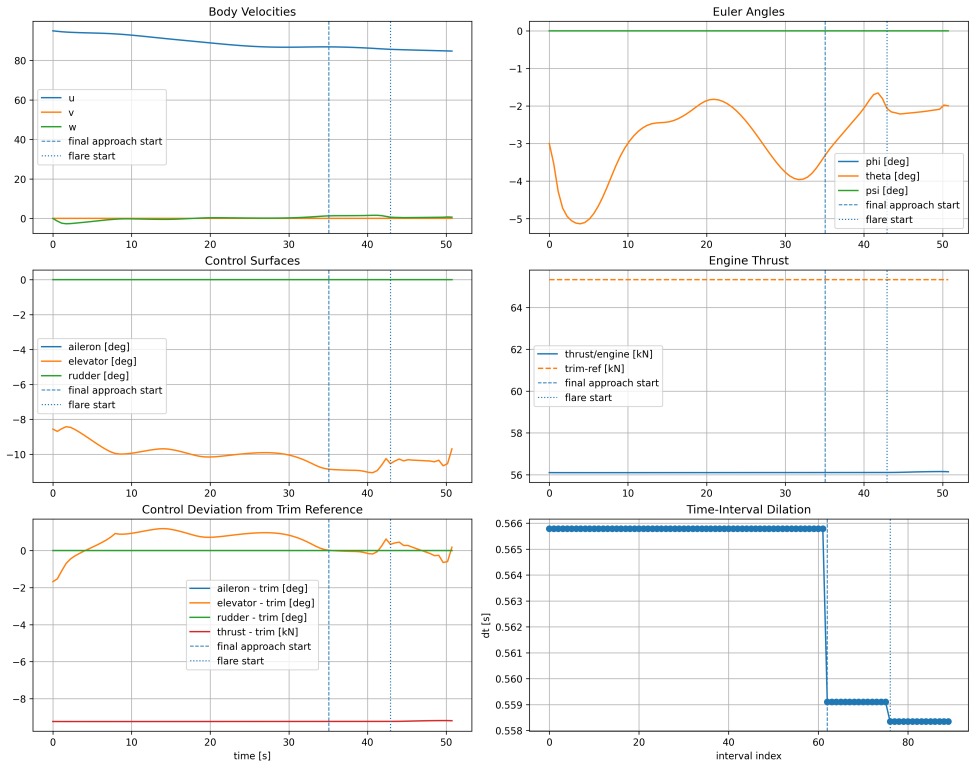


Figure 4.4: State, control, trim-deviation, and time-dilation histories for the converged nominal landing trajectory. The optimal nonlinear solution exhibits a smooth speed decay, small pitch variations and nearly constant thrust.

4.10 Convergence, and Feasibility Assessment

Figure 4.5 summarizes the internal xPTR convergence behavior. The nonlinear merit and path terms decrease by several orders of magnitude, virtual control is driven to a numerically negligible level after the early iterations, and the accepted step sizes remain large enough to converge in eight iterations.

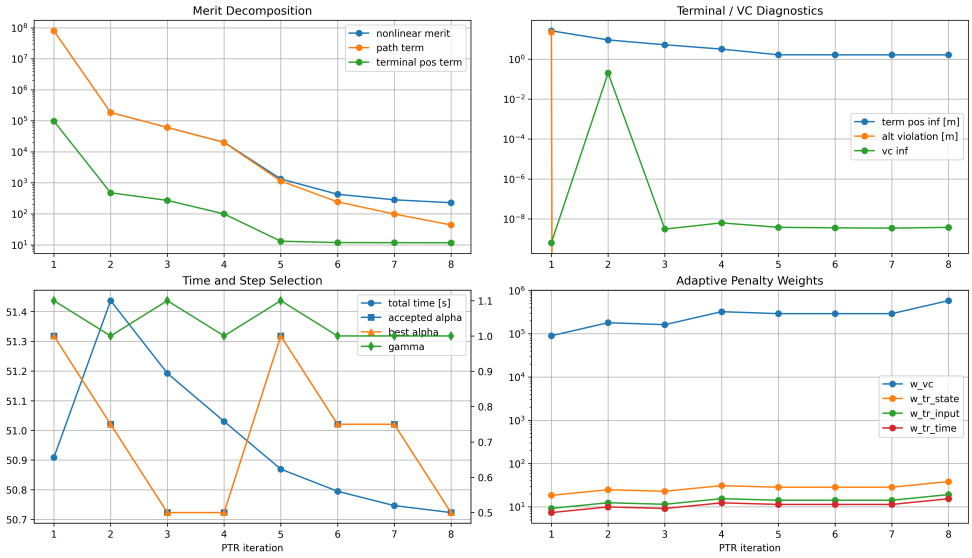


Figure 4.5: xPTR convergence diagnostics for the final nominal landing solution showing the merit decomposition, terminal error metrics, accepted step sizes, and adaptive penalty weights over each PTR iteration.

Dense inter-sample verification of the final accepted nominal trajectory showed no continuous-time violations of the enforced roll, pitch, or angle-of-attack bounds. Using $n_{\text{dense}} = 20$ subsamples per interval over $N = 90$ intervals, the nonlinear post-check evaluated the trajectory on a much finer time grid than the optimization mesh before the nominal was passed to the tracking controller. In the straight-in landing case, airplane roll remained essentially level, with $\max|\phi| \approx 0^\circ$, so the nominal stayed well inside both the pre-flare $\pm 25^\circ$ limit and the tighter flare-phase $\pm 8^\circ$ limit. The pitch remained within the global bound $|\theta| \leq 15^\circ$, varying from -5° to -1.7°. The (reconstructed) angle of attack values remained safely within its allowable limits of -5° and 15°, with dense-grid values of roughly $-1.7^\circ \leq \alpha \leq 1.0^\circ$. No dense-grid bound excesses were detected for for roll, pitch, and angle of attack. These results indicate that the accepted nominal trajectory remains accurate under finer continuous-time nonlinear analysis.

4.11 Limitations of the PTR Implementation

The current nominal formulation has several limitations. First, the planner does not model landing-gear contact, ground effect, or full touchdown physics. Therefore, the final state can be viewed as a late-approach or flare-completion reference rather than as an actual main-gear touchdown-like state. Additionally, the continuous-time integral reformulation has been implemented only for roll and pitch. These variables were the most important attitude constraints identified in the landing scenario, but it is not a full continuous-time verification of all path constraints. The current scenario is also a simple, straight-in landing case. Because airplane roll was not necessarily used in any turning maneuvers, it is therefore expected that the nominal remains almost perfectly wings-level. More aggressive cases, such as off-center initial conditions, crosswind, or even obstacle avoidance would provide a stronger test of the continuous-time constraint violations.

It was also noticed that the solution is sensitive to weight tuning. Small changes in flare weights, terminal pitch shaping, or terminal-altitude interpretation were found to degrade PTR convergence. The final tuning should therefore be interpreted as applicable to a very localized set of initial and final conditions rather than as a unique or globally optimal. The weight constraints of the PTR algorithm should ideally converge to some local optima given a range of varying initial and final conditions to the nonlinear optimal landing maneuver, however it is currently too sensitive to small changes in hard constraint values to be deployed on real-world systems. Finally, it can be seen in Figure 4.4 that the thrust profile holds steady at about $56\ \mathrm{kN}$ per engine, which sits below the trim-reference level of roughly $65\ \mathrm{kN}$. This suggests the optimizer may not be complex enough to model real-world engine thrust behavior over the landing maneuver. These issues suggest that the current landing logic to be too basic for more realistic landing maneuvers, especially in different operating conditions.

For these reasons, the final modeling decision was to keep the PTR problem focused on generating a stable airborne nominal reference rather than forcing it to solve touchdown realism directly. The tracking controller and later digital-twin terminal logic are responsible for handling the physically meaningful touchdown behavior. The main precaution is that closed-loop tracking must begin early enough that the plant remains airborne when it begins following the nominal terminal segment.

Chapter 5

TRACKING CONTROL ALONG THE LANDING TRAJECTORY

5.1 Application of the Nominal Optimal Landing Trajectory

The trajectory optimization section provided a nominal, dynamically feasible landing approach in the form of a discrete state and input history $\{x_k^\star, u_k^\star\}_{k=0}^N$. The nominal trajectory however, is not robust to disturbances or model mismatch when executed in an open loop setting. Thus, it is treated as a feedforward reference motion that can be stabilized by a closed-loop tracking controller on the higher-fidelity plant. The optimizer acts as a reference for where the aircraft should be and what nominal inputs to use on the planning model, while the tracking controller provides the feedback that's necessary to account for any deviations caused by unmodeled effects, which creates a discrepancy between the digital model and physical system.

There exists a wide range of trajectory tracking controllers that could be applied to the 6dof nonlinear aircraft system. These include gain-schedulized controllers that linearize around multiple trim conditions and interpolate controllers as a function of scheduling variables [33]. Nonlinear approaches include dynamic inversion-based control and Lyapunov-based nonlinear designs, however these methods are sensitive to model mismatch [34]. Another popular method is Model Predictive Control (MPC) which repeatedly solves a constrained finite-horizon optimal control problem online in a receding-horizon fashion. A drawback to MPC in this application is the computational complexity around solving an optimization problem at each timestep which can be computationally demanding if it's used as a high-rate flight controller.

In this thesis, a Time-Varying Linear Quadratic Regulator (TVLQR) is used as the tracking controller. In TVLQR, the nonlinear plant is linearized about the nominal trajectory and a stabilizing controller is designed for the resulting Linear time-varying (LTV) error dynamics [35]. A finite-horizon Riccati recursion is then solved (offline) along this LTV model to produce a time-indexed gain sequence that locally stabilizes deviations from the nominal state–input history.

TVLQR is particularly applicable in this thesis for many reasons including the ability to produce controller gains that are tuned to each point along the nominal trajectory's linearization, especially since the landing maneuver is time-varying. Additionally, while TVLQR may be computationally intensive offline

from the Ricatti recursion and backward integration step, this is done only once on the nominal trajectory, and the controller is much quicker during online deployment as it only needs to perform matrix multiplication.

5.2 Time-Varying Tracking Formulation

The nominal landing trajectory from the previous section provides a reference state and input history, but those planner nodes are too coarse to be applied on a closed-loop controller. A denser controller grid is therefore created, and the nominal state and commanded-input histories are interpolated onto that grid. This produces a sampled reference pair

$$(x_k^\star, u_{c,k}^\star) \qquad k = 0, \ldots, M$$

where $u_{c,k}^\star$ denotes the commanded control input applied to the plant-side actuators. The nonlinear plant is then linearized locally along the moving reference trajectory, and the feedback law is used to regulate deviations from that reference at each time step.

Because the higher-fidelity plant includes actuator dynamics, the tracking model is written in an augmented state. Let

$$z_k \triangleq \begin{bmatrix} x_k \\ u_{a,k} \end{bmatrix} \in \mathbb{R}^{16}, \qquad x_k = \begin{bmatrix} N_k & E_k & D_k & u_k & v_k & w_k & p_k & q_k & r_k & \phi_k & \theta_k & \psi_k \end{bmatrix}^\top \tag{5.1}$$

where $u_{a,k} \in \mathbb{R}^4$ contains the realized aileron, elevator, rudder, and (symmetric) thrust actuator states. The tracking input is the commanded actuator vector

$$u_{c,k} = \begin{bmatrix} \delta_{A,c,k} & \delta_{E,c,k} & \delta_{R,c,k} & T_{c,k} \end{bmatrix}^\top$$

The flaps, landing gear, ground spoiler, and brake commands are held fixed during this stage, therefore only the aileron, elevator, rudder, and thrust commands are actively being controlled. Thus, the controller is a 16 state augmented TVLQR which includes the aircraft dynamics and actuator states. Let the sampled nonlinear augmented dynamics be written as

$$z_{k+1} = F_k(z_k, u_{c,k})$$

where F_k denotes a single controller period propagation of the nonlinear plant together with actuator lag. Since actuator lag is included as a part of the state, the reference used for feedback must also include a

reference actuator history. Starting from the interpolated command sequence $u^\star_{c,k}$, a corresponding actuator reference $u^\star_{a,k}$ is generated by propagating the same actuator model along the nominal command history. This defines the augmented reference vector

$$z^\star_k = \begin{bmatrix} x^\star_k \\ u^\star_{a,k} \end{bmatrix}$$

The deviation variables are then defined as the difference between the actual and reference states

$$e_k = z_k - z^\star_k, \qquad \delta u_k = q u_{c,k} - u^\star_{c,k} \tag{5.2}$$

For the Euler-angle components, the error is wrapped onto the interval $(-\pi, \pi]$ so that equivalent attitudes on opposite sides of $\pm\pi$ are treated as nearby angles, rather than being mis-interpreted as discontinuities on the scale of 2π.

$$e_{\chi,k} = \mathrm{wrap}(\chi_k - \chi^\star_k), \qquad \chi \in \{\phi, \theta, \psi\}$$

A local deviation model is obtained by linearizing the sampled nonlinear plant about the moving reference pair $(z^\star_k, u^\star_{c,k})$:

$$z_{k+1} \approx F_k(z^\star_k, u^\star_{c,k}) + A_k(z_k - z^\star_k) + B_k(u_{c,k} - u^\star_{c,k})$$

where

$$A_k \approx \left.\frac{\partial F_k}{\partial z}\right|_{(z^\star_k, u^\star_{c,k})}, \qquad B_k \approx \left.\frac{\partial F_k}{\partial u_c}\right|_{(z^\star_k, u^\star_{c,k})}$$

A_k and B_k are the Jacobians of the sampled nonlinear augmented plant map with respect to the augmented state and commanded input, respectively. These matrices are used as a local deviation model around the nominal reference trajectory. Since the reference is propagated by the same sampled model

$$z^\star_{k+1} = F_k(z^\star_k, u^\star_{c,k})$$

subtracting the reference update from the linearized plant update results in the expression

$$z_{k+1} - z^\star_{k+1} \approx A_k(z_k - z^\star_k) + B_k(u_{c,k} - u^\star_{c,k})$$

Using the deviation definitions from (5.2), the tracking-error dynamics become

$$e_{k+1} \approx A_k e_k + B_k \delta u_k \tag{5.3}$$

Equation (5.3) is therefore the local linear time-varying (LTV) error model obtained by expressing the sampled nonlinear augmented dynamics in coordinates relative to the moving reference $z_k^\star$. The matrix A_k describes how the current augmented-state error propagates over one controller step, while B_k maps a commanded correction δu_k into the next-step tracking error. Because the linearization is performed along the nominal landing trajectory, these matrices will vary with k.

Finite-horizon tracking cost. The objective is to keep the plant close to the augmented reference trajectory $\{z_k^\star, u_{c,k}^\star\}_{k=0}^{M}$ while avoiding unnecessarily large corrective control actions. This is expressed with a finite-horizon quadratic cost: the matrices Q and Q_f penalize augmented-state tracking error along the horizon and at the terminal point, respectively, while R penalizes the size of the commanded input correction relative to the nominal command history.

$$J = \frac{1}{2}(z_M - z_M^\star)^\top Q_f (z_M - z_M^\star) + \frac{1}{2}\sum_{k=0}^{M-1}\left[(z_k - z_k^\star)^\top Q (z_k - z_k^\star) + (u_{c,k} - u_{c,k}^\star)^\top R (u_{c,k} - u_{c,k}^\star)\right]$$

denote $e_k = z_k - z_k^\star$ and $\delta u_k = u_{c,k} - u_{c,k}^\star$ and the above expression simplifies to

$$J = \frac{1}{2} e_M^\top Q_f e_M + \frac{1}{2}\sum_{k=0}^{M-1}\left(e_k^\top Q e_k + \delta u_k^\top R \delta u_k\right) \tag{5.4}$$

Here $Q \succeq 0$ and $Q_f \succeq 0$ penalize augmented-state tracking error over the horizon and at the terminal point, while $R \succ 0$ penalizes commanded-input correction effort. The matrix Q is the running state-error weighting in the stage cost, while Q_f is the terminal state-error weighting applied at the end of the horizon. These weights are applied to the full augmented coordinates, so the cost penalizes rigid-body state error as well as actuator-state error through the actuator components of $z_k^\star$. Larger weights are placed on the components that are most relevant to late-approach accuracy, which includes altitude, body-axis vertical velocity, pitch angle, pitch rate, and realized elevator behavior.

Backward Riccati recursion and optimal gain sequence. For the local time-varying (LTV) model (5.3) with quadratic cost (5.4), the optimal feedback law (cost-to-go function) is obtained from the finite-horizon discrete-time Bellman recursion. Define the value function $V_k(e_k)$ as the minimum remaining tracking cost

from step k to the terminal step M. Under the standard finite-horizon LQR approach, this optimal value function is quadratic:

$$V_k(e_k) = \frac{1}{2} e_k^\top P_k e_k, \qquad V_M(e_M) = \frac{1}{2} e_M^\top Q_f e_M$$

The function $V_k(e_k)$ is called the cost-to-go because it measures how much optimal tracking cost remains from time step k onward, assuming the current augmented tracking error is e_k and all future control corrections are chosen optimally. $P_k \in \mathbb{R}^{n_z \times n_z}$ is the symmetric matrix that parameterizes the minimum remaining quadratic cost at time step k, where n_z is the dimension of the augmented state. The matrix Q_f is the terminal state-weighting matrix from the cost function. The terminal condition is therefore given by

$$P_M = Q_f$$

That is, at the final time step, the remaining cost is exactly the terminal tracking penalty.

Bellman's principle of optimality states that the optimal cost-to-go at step k is equal to the current stage cost plus the optimal remaining cost from step $k+1$ onward, minimized over the current control correction δu_k. In discrete time, this gives the recursion

$$V_k(e_k) = \min_{\delta u_k} \frac{1}{2} \left(e_k^\top Q e_k + \delta u_k^\top R \delta u_k + e_{k+1}^\top P_{k+1} e_{k+1} \right)$$

This expression says that the best achievable remaining cost at step k is found by choosing the current control correction δu_k to balance the current state-tracking penalty, the current control-effort penalty, and the future cost induced through the next-step error e_{k+1}. Substituting in equation (5.3) gives

$$V_k(e_k) = \min_{\delta u_k} \frac{1}{2} \Big[e_k^\top (Q + A_k^\top P_{k+1} A_k) e_k + 2\, \delta u_k^\top B_k^\top P_{k+1} A_k e_k + \delta u_k^\top (R + B_k^\top P_{k+1} B_k) \delta u_k \Big]$$

The minimizing control correction can then be found by differentiating with respect to δu_k and setting the result equal to zero, resulting in the expression

$$(R + B_k^\top P_{k+1} B_k) \delta u_k + B_k^\top P_{k+1} A_k e_k = 0$$

Hence the optimal correction has the linear state-feedback form

$$\delta u_k = -K_k e_k, \qquad u_{c,k} = u_{c,k}^\star - K_k e_k$$

where

$$K_k = \left(R + B_k^\top P_{k+1} B_k\right)^{-1} B_k^\top P_{k+1} A_k \tag{5.5}$$

Substituting this minimizing policy back into the quadratic form yields the backward discrete Riccati recursion

$$P_M = Q_f, \qquad P_k = Q + A_k^\top P_{k+1} A_k - A_k^\top P_{k+1} B_k \left(R + B_k^\top P_{k+1} B_k\right)^{-1} B_k^\top P_{k+1} A_k \tag{5.6}$$

Thus, the gain, K_k gets computed offline and the reference trajectories $z_k^\star$ and $u_{c,k}^\star$ gets stored. During Online implementation, the current augmented state z_k is estimated, and the tracking error is computed as $e_k = z_k - z_k^\star$. This error is then used to compute the actual command correction, $-K_k e_k$. In an idealized case where there is no unmodeled disturbance, $e_k = 0$ so the controller only needs to apply the nominal command $u_{c,k}^\star$ and the plane would stay on the nominal trajectory exactly. In the closed loop implementation, the feedback correction is defined by $\delta u_k = -K_k e_k$ and is computed online and changes at every time step depending on the estimated error.

Equations (5.5)–(5.6) show that the gain schedule is obtained by propagating the quadratic cost-to-go matrix P_k backward in time from the terminal weight Q_f to the initial tracking step. Because A_k and B_k vary along the nominal landing trajectory, the resulting feedback is time-varying as well. At each controller step, the nominal command $u_{c,k}^\star$ supplies the feedforward action associated with the reference trajectory, while the correction $-K_k e_k$ provides local stabilization of deviations from the augmented reference $z_k^\star$.

5.3 Gain-Schedule Construction and Online Execution

The gain schedule is built numerically from the same nonlinear sampled augmented plant used in closed-loop simulation. Starting from the nominal landing outputs $\{x_k^\star, u_k^\star, S_k\}$, the first step is to construct a denser augmented reference on a gain-schedule grid with sample rate f_g. Let

$$\Delta t_g = \frac{1}{f_g}, \qquad t_0^g = 0, \qquad t_M^g = t_{\text{end}}$$

where $t_{\text{end}} = t_h$. The nominal state and commanded input histories are interpolated onto this grid to obtain

$$x^\star(t_k^g), \qquad u_c^\star(t_k^g), \qquad k = 0, \ldots, M$$

Because the controller is designed on the augmented state $z = [x^\top, u_a^\top]^\top$, the actuator component of the reference must also be reconstructed. This is done by simulating the same first-order actuator model along the nominal commanded-input history:

$$u^\star_{a,k+1} = u^\star_{a,k} + \alpha_g\big(u^\star_{c,k} - u^\star_{a,k}\big) \tag{5.7}$$

where $\alpha_g = 1 - e^{-\Delta t_g/\tau_a}$ and $u^\star_{a,0} = u^\star_{c,0}$. This augmented reference is the linearization point used throughout the finite-horizon TVLQR construction. At each schedule node, the discrete nonlinear map

$$z_{k+1} = F_k(z_k, u_{c,k})$$

is evaluated numerically by propagating the nonlinear RCAM plant and actuator states over one schedule interval Δt_g. The Jacobians A_k and B_k are then computed by forward finite differences about each sampled reference pair $(z^\star_k, u^\star_{c,k})$:

$$A_k^{(:,i)} \approx \frac{F_k\Big(z^\star_k + \varepsilon_{z,i}e_i,\ u^\star_{c,k}\Big) - F_k\Big(z^\star_k,\ u^\star_{c,k}\Big)}{\varepsilon_{z,i}} \tag{5.8}$$

$$B_k^{(:,j)} \approx \frac{F_k\Big(z^\star_k,\ u^\star_{c,k} + \varepsilon_{u,j}e_j\Big) - F_k\Big(z^\star_k,\ u^\star_{c,k}\Big)}{\varepsilon_{u,j}} \tag{5.9}$$

Here, $\varepsilon_{z,i}$ and $\varepsilon_{u,j}$ denote small perturbations used to approximate the local Jacobians by finite differences. Once $\{A_k, B_k\}$ are available, the regularized backward Riccati recursion (5.6) is solved over the pre-handoff horizon to produce the gain sequence $\{K_k\}$. The weighting matrices are expressed in diagonal form,

$$Q = \mathrm{diag}(q_1, \ldots, q_{16}), \qquad R = \mathrm{diag}(r_1, \ldots, r_4), \qquad Q_f = \lambda_f Q$$

with $\lambda_f > 1$ being a terminal-weight multiplier so that terminal tracking error is penalized more heavily than intermediate error. Each weight is interpreted as the inverse square of a tracking deviation. That is,

$$q_i \sim \frac{1}{\sigma_i^2}, \qquad r_j \sim \frac{1}{\rho_j^2} \tag{5.10}$$

where σ_i is the characteristic acceptable error in the ith augmented-state channel and ρ_j is the acceptable correction size in the jth commanded-input channel.

The TVLQR gain schedule is constructed on a 10 Hz gain grid, while closed-loop controller updates, logging, and EKF updates are performed at 20 Hz. The RCAM plant dynamics are integrated internally at

200 Hz using substeps within each controller interval. During online deploymnt, the controller evaluates the current plant state, forms the augmented tracking error e_k, and applies the nominal command together with the TVLQR correction. The pre-handoff command law can be written as

$$u_{c,k} = \mathrm{sat}\big(u^{\star}_{c,k} - \gamma_{\mathrm{fb}}\,\mathrm{clip}(K_k e_k, \Delta u_{\max})\big) \tag{5.11}$$

Here, $u^{\star}_{c,k}$ is the nominal commanded input, K_k is the scheduled TVLQR gain, e_k is the augmented tracking error, $\gamma_{\mathrm{fb}} \in (0, 1]$ is a scalar feedback scale, and $\Delta u_{\max}$ is a increment bound. This clipping layer is included because the unconstrained linear feedback increment can request unrealistically large command changes when the plant departs from the nominal trajectory, especially near the end of the maneuver. This controller has actuator states included in the tracked augmented state so the feedback law penalizes realized surface and thrust deviations in addition to rigid-body state error. The saturated command is passed through the plant side actuator dynamics before affecting the aerodynamic and propulsion forces. Algorithm 2 summarizes the execution sequence of the controller, from offline gain-schedule construction through online mode switching and nonlinear plant deployment. Figure 5.1 then provides a visual summary of the same logic.

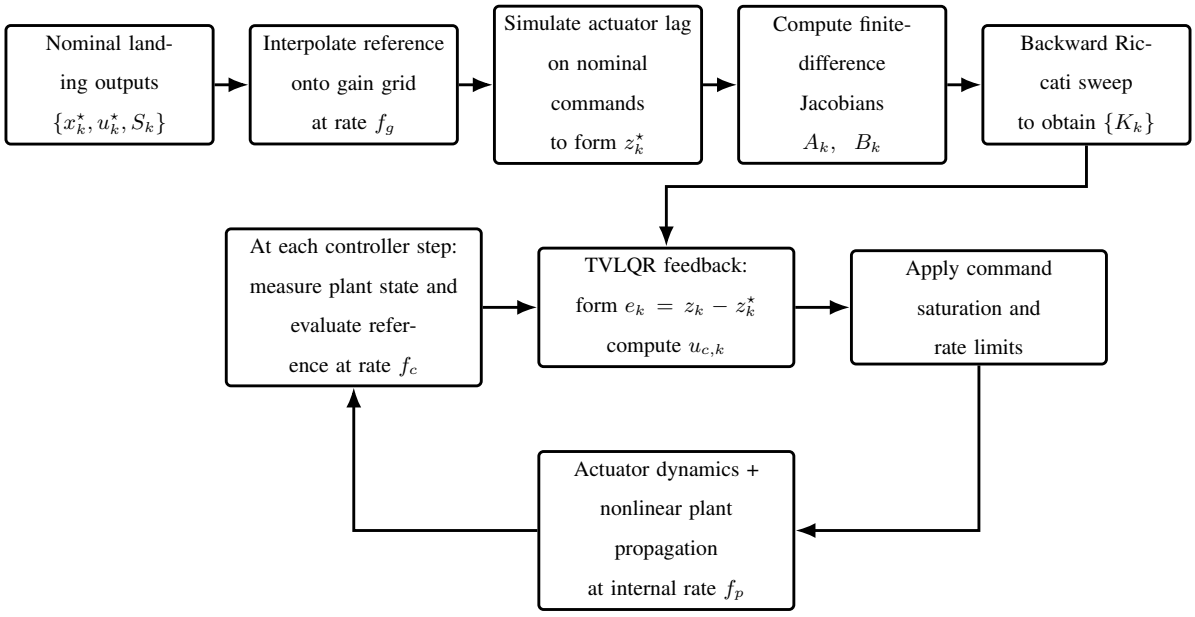


Figure 5.1: Schematic of the augmented-state TVLQR tracking-controller architecture. The top row summarizes the offline gain-schedule construction from the nominal landing trajectory and the sampled nonlinear augmented plant. The lower loop summarizes online execution, where the plant state is measured, the reference and gain are evaluated, the TVLQR feedback command is saturated and rate-limited, and the nonlinear plant with actuator dynamics is propagated at the internal integration rate.

Algorithm 2: Augmented-state TVLQR tracking along the nominal landing trajectory

Input: Nominal outputs $\{x_k^\star, u_k^\star, S_k\}$, rates f_c, f_g, f_p
Output: Closed-loop trajectory $\{z_k, u_{c,k}\}$ and diagnostics
`// 1. Offline Gain Scheduling`
Build augmented reference $(Z^\star, U^\star)$ via interpolation and simulated actuator lag
for $k = 0, \ldots, M-1$ **do**
  Linearize augmented plant about $(z_k^\star, u_{c,k}^\star)$ to obtain Jacobians A_k, B_k
end
Solve backward Riccati recursion for time-varying gains $\{K_k\}$
`// 2. Online Execution`
Initialize perturbed nonlinear plant at t_0
for *each controller step* t_k **do**
  Interpolate reference and form wrapped tracking error $e_k = z_k - z_k^\star$
  Compute increment-limited TVLQR command (Eq. 5.11)
  Propagate nonlinear plant, apply actuator dynamics, and log telemetry
end

5.4 Tracking Results and Controller Behavior

Three tracking scenarios were evaluated using the nominal trajectory generated in the previous section. Across all cases, the same nominal reference and cached TVLQR gain schedule were used. The gain schedule was computed using the nominal zero-wind model, so the disturbed cases evaluate closed-loop tracking under plant disturbance mismatch rather than a wind-specific redesign of the controller.

- **Case 1, no-disturbance verification case:** Uses zero wind, no gusts, no turbulence, and zero initial state and actuator perturbations. This case checks whether the controller can track the nominal trajectory when the plant is nearly aligned with the reference model.

- **Case 2, moderate wind case:** Adds a deterministic NED base wind, windowed gust variation, colored turbulence, and small initial state and actuator perturbations. The colored turbulence component uses a Gauss–Markov process with standard deviation $[0.55,\ 0.45,\ 0.18]\ \mathrm{m/s}$ and time constants $[3.0,\ 4.0,\ 2.5]\ \mathrm{s}$ in the NED frame.

- **Case 3, severe wind case:** Uses a stronger base wind, stronger gusts, larger turbulence intensity, and larger initial state and actuator perturbations. The colored turbulence component uses a Gauss–Markov process with standard deviation $[1.10,\ 0.90,\ 0.30]\ \mathrm{m/s}$ and time constants $[2.2,\ 2.8,\ 1.8]\ \mathrm{s}$ in the NED frame.

Table 5.1 summarizes the wind levels used in each case. The base wind is the constant NED wind bias

applied to the plant. The remaining columns summarize the magnitude of the total applied wind vector,

$$|\mathbf{V}_w(t)| = \sqrt{V_{w,N}^2(t) + V_{w,E}^2(t) + V_{w,D}^2(t)},$$

after the base wind, gust, and turbulence components are combined. The mean wind speed gives the average disturbance level over the simulated maneuver while the RMS value, or root-mean-square value, gives a magnitude-weighted average. The maximum value reports the largest instantaneous wind speed reached during the maneuver and used for interpreting the strongest gust/turbulence experienced by the aircraft

Table 5.1: Wind settings and realized wind-speed magnitudes for the tracking scenarios.

Case	Base wind [N,E,D] (m/s)	Mean $\lvert\mathbf{V}_w\rvert$ (m/s)	RMS $\lvert\mathbf{V}_w\rvert$ (m/s)	Max $\lvert\mathbf{V}_w\rvert$ (m/s)
No-disturbance	$[0.00,\ 0.00,\ 0.00]$	0.000	0.000	0.000
Moderate wind	$[1.50,\ -0.40,\ 0.00]$	2.105	2.310	4.014
Severe wind	$[3.00,\ -1.25,\ 0.00]$	5.781	6.026	9.836

The disturbed cases also include initial offsets in the aircraft state and actuator states, summarized in Table 5.2. The aircraft-state perturbation is applied to the initial local NED position, body velocity, body rate, and Euler attitude states. The actuator perturbation is applied to the initial aileron, elevator, rudder, and thrust actuator states. These offsets prevent the disturbed cases from being evaluated from an idealized initial condition that matches perfectly with the plant.

Table 5.2: Initial state and actuator perturbations used in the disturbed tracking scenarios.

Case	Initial aircraft-state perturbation	Initial actuator perturbation
No-disturbance	None	None
Moderate wind	$[N, E, D] = [2.0,\ 0.4,\ -0.15]$ m $[u, v, w] = [0.35,\ 0.05,\ 0.08]$ m/s $[p, q, r] = [0.10,\ 0.06,\ 0.08]$ deg/s $[\phi, \theta, \psi] = [0.00,\ 0.08,\ 0.00]$ deg	$[\delta_a, \delta_e, \delta_r] = [0.10,\ -0.12,\ 0.06]$ deg $\Delta T = 120$ N
Severe wind	$[N, E, D] = [6.0,\ 1.0,\ -0.40]$ m $[u, v, w] = [0.80,\ 0.15,\ 0.18]$ m/s $[p, q, r] = [0.35,\ 0.20,\ 0.25]$ deg/s $[\phi, \theta, \psi] = [0.20,\ 0.20,\ 0.25]$ deg	$[\delta_a, \delta_e, \delta_r] = [0.25,\ -0.28,\ 0.18]$ deg $\Delta T = 300$ N

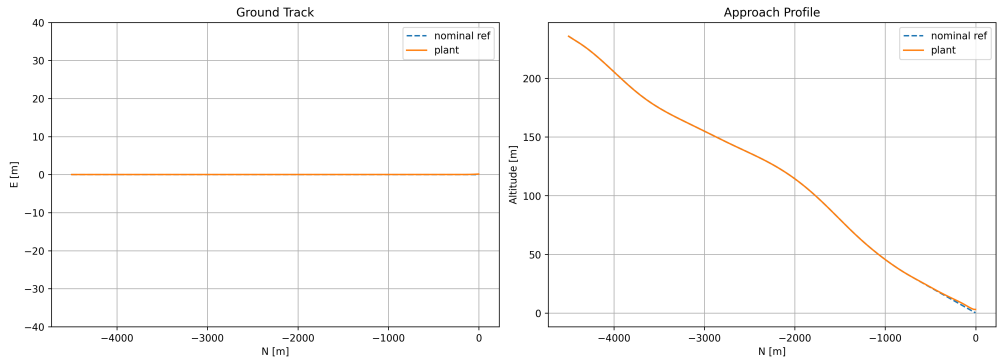


Figure 5.2: Ground track and approach profile for Case 1 (verification). The plant trajectory remains nearly coincident with the nominal reference.

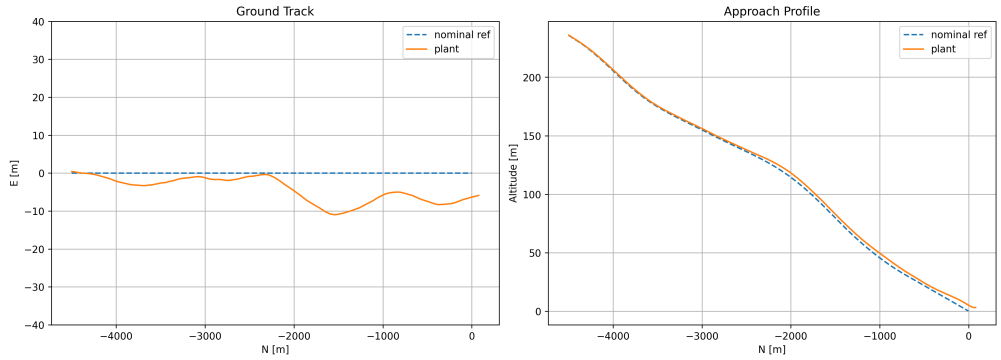


Figure 5.3: Ground track and approach profile for Case 2 (moderate realism). With steady wind, gusts, turbulence, and small initial perturbations, the plant develops visible but still smooth geometric drift relative to the nominal reference.

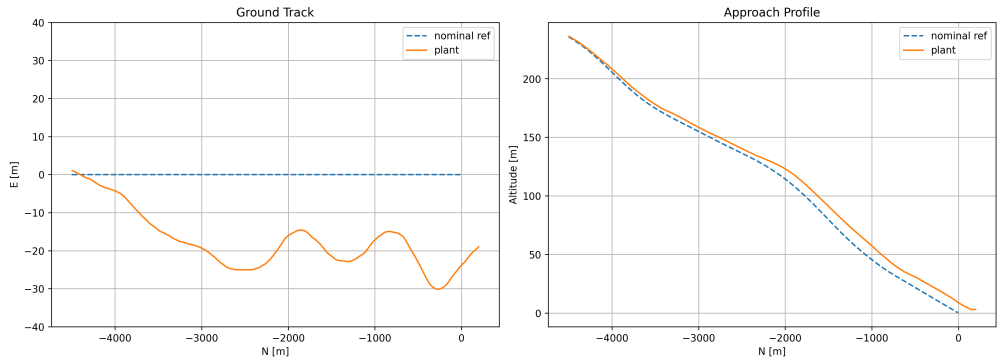


Figure 5.4: Ground track and approach profile for Case 3 (stress test). Under the strongest mismatch, the geometric drift increases, especially in the ground-track deviation and the separation between the plant and nominal approach profiles.

Table 5.3: Baseline TVLQR tracking-error metrics for the three tracking scenarios.

Case	Pos. RMSE (m)	Max pos. err. (m)	N-RMSE (m)	E-RMSE (m)	D-RMSE (m)	Vel. RMSE (m/s)	θ-RMSE (deg)
No-disturbance	0.541	3.655	0.289	0.012	0.458	0.578	0.331
Moderate wind	48.551	81.454	48.230	5.536	0.703	0.782	0.357
Severe wind	128.704	202.902	127.251	19.250	1.176	1.546	0.351

Case 1 remains very close to the nominal reference over the entire airborne segment. As the wind disturbance is increased in Cases 2 and 3, the tracking errors grow, but the closed-loop response remains bounded. No command or command-rate saturation events were recorded in any scenario, which indicates that the observed loss of tracking quality is not caused by actuator clipping. Table 5.3 summarizes the baseline TVLQR tracking-error metrics before EKF synchronization or digital-twin terminal intervention is introduced. The no-disturbance case has small position error, with a total position RMSE of $0.541\ \mathrm{m}$, confirming that the cached TVLQR schedule can track the nominal trajectory when the plant remains close to the planning model. As the disturbance level increases, the total position RMSE grows to $48.551\ \mathrm{m}$ in the moderate wind case and $128.704\ \mathrm{m}$ in the severe wind case. Most of this increase comes from the downrange N component, while the vertical D-error remains much smaller. The pitch RMSE remains nearly unchanged across the three cases, which suggests that the main degradation is an accumulated ground-track and phase error caused by wind, gusts, turbulence, and initial perturbations.

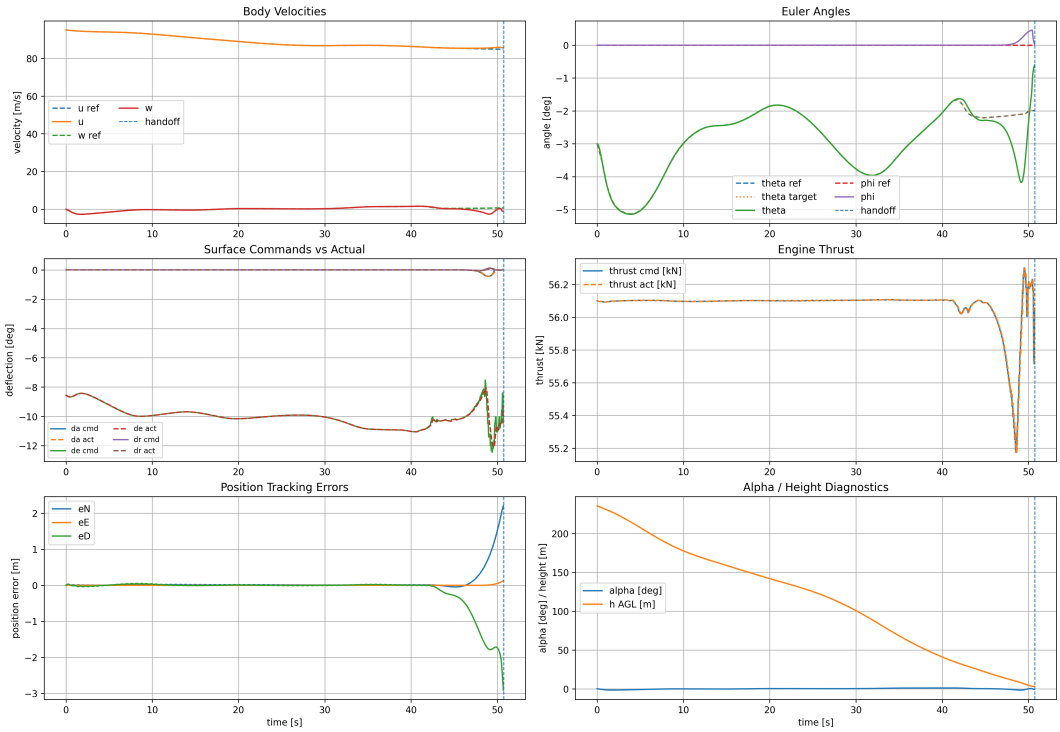


Figure 5.5: State and control histories for Case 1 (verification). The body-velocity, actuator, and engine thrust remain closely aligned with the nominal reference.

Figures 5.2–5.4 show how the tracking behavior changes over the three tested cases. In Case 1, the plant remains nearly coincident with the nominal ground track and approach profile. In Case 2, a visible but smooth drift appears in both the lateral ground track and the vertical approach geometry. In Case 3, these discrepancies become larger, especially in the ground-track deviation, while the closed-loop response remains bounded and does not show actuator saturation. The Euler-angle subplot for the no-wind case in Figure 5.5 shows that the pitch angle begins to move away from the nominal reference near the end of the maneuver. This behavior is best interpreted as a finite-horizon TVLQR terminal-edge effect since the verification case contains no wind, no initial perturbation, and no command saturation. The gain schedule is being used up to the end of a finite reference horizon where the nominal trajectory is already near the main-gear contact condition, but the baseline controller has no dedicated terminal landing or flare policy. Small residual altitude, velocity, and pitch errors therefore become more visible near the endpoint and can produce sharper elevator activity. Actuator dynamics also contribute to the mismatch, but the pitch departure remains primarily a terminal-schedule limitation of the baseline TVLQR implementation rather than a failure of the

full trajectory tracking behavior.

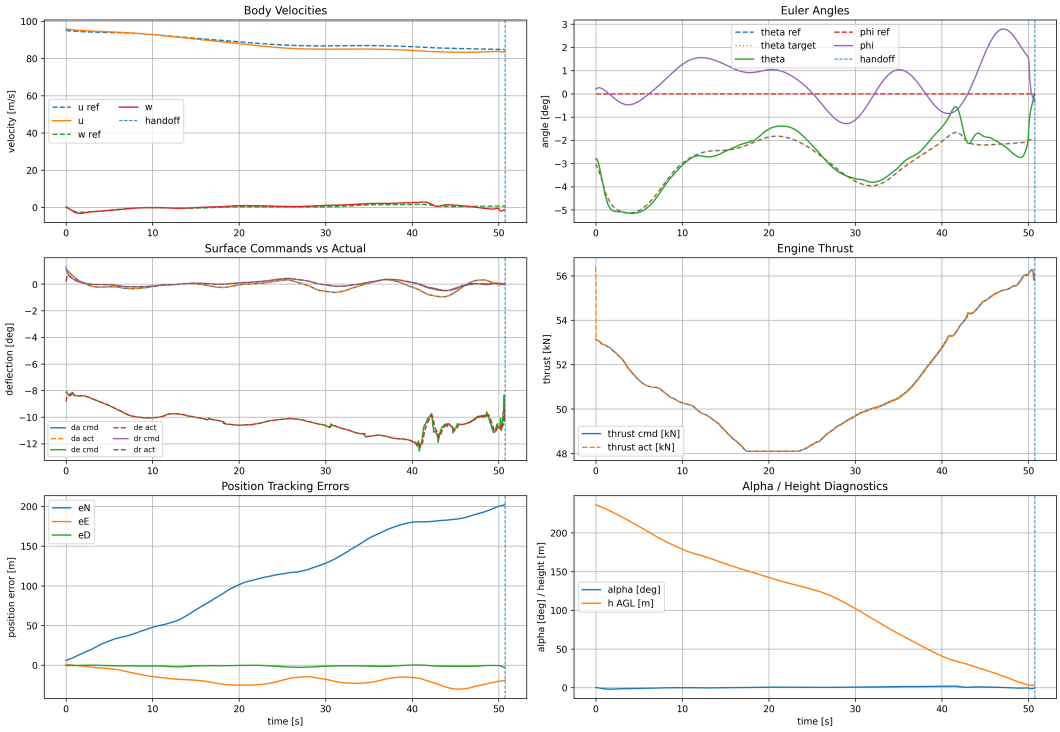


Figure 5.6: State and control histories for Case 3 (stress test). Relative to Case 1, the Euler-angle subplot shows a larger discrepancy between the plant and reference, particularly in bank and pitch during the later approach and flare segments. The position-tracking-error subplot show substantial growth in downrange error and smaller but visible cross-track deviation, while the altitude error remains comparatively small. The actuator and thrust histories remain smooth, and the commanded and actual inputs stay closely aligned.

The state and control histories in Figure 5.6 show that the body velocities remain close to the reference, and the commanded and realized actuator histories are closely aligned. This indicates that actuator lag is present, but not large enough to cause divergent behavior in this case. Engine thrust also remains smooth, and no command saturation events were recorded in any scenario. Figure 5.7 shows that in the verification case, the corrections remain very small for nearly the entire trajectory, which is consistent with the near-nominal tracking performance already observed in the state and path histories. Under moderate realism, the controller must apply visibly larger corrections in all channels, with the thrust correction becoming more pronounced over the later portion of the approach. In the stress-test case, the required corrections increase, and the thrust channel exhibits the largest sustained deviation from the nominal command, indicating that

the controller is working significantly harder to counter the stronger wind mismatch, turbulence, and initial offsets. These results indicate that the loss in tracking quality observed in the more disturbed scenarios corresponds to an increase in feedback effort, even though the same nominal reference and gain schedule are used in all three cases.

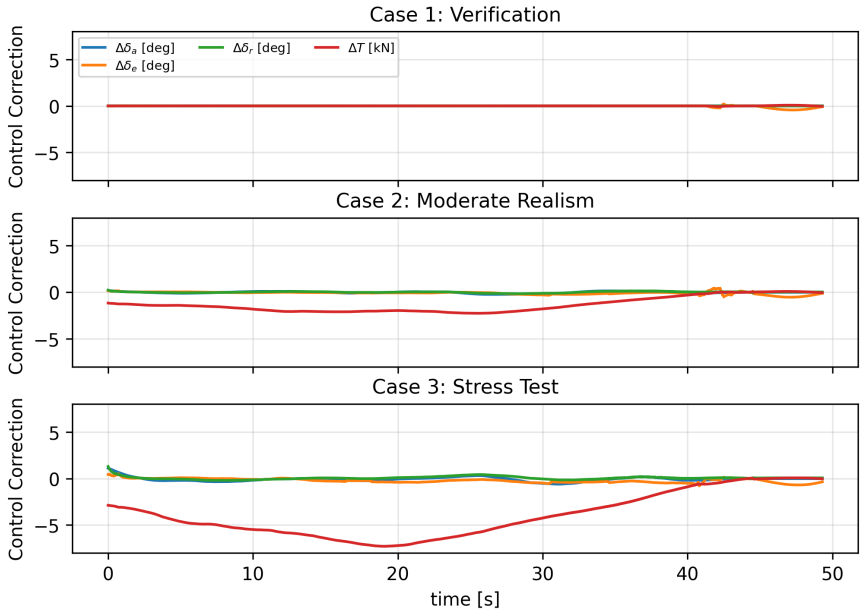


Figure 5.7: Feedback control corrections relative to the nominal command history for the three tracking cases. The dashed vertical line marks the time-based handoff from the main TVLQR tracking phase to the post-handoff control mode.

Terminal metric convention. Terminal landing metrics are reported using the main-gear clearance rather than the aircraft center-of-gravity altitude. The RCAM state gives the vehicle pose at the center of gravity, so the modeled front and rear main-gear locations are transformed through the aircraft attitude into the local runway frame. The main-gear clearance, h_{mg}, is then defined as the vertical clearance between the runway and the lowest main-gear contact point. This quantity is used for terminal reporting because it is closer to the physical touchdown geometry than the center-of-gravity altitude. A landing is counted as reaching main-gear contact when $h_{\mathrm{mg}} \leq 0.09\ \mathrm{m}$. This threshold is a near-touchdown tolerance around the nominal touchdown-clearance target of $0.08\ \mathrm{m}$. The additional $0.01\ \mathrm{m}$ margin prevents the contact classification from depending on numerical penetration, gear compression, or the exact logging sample at which the trajectory crosses the runway contact boundary. Thus, "main-gear contact" in the terminal tables means entry into the

intended touchdown band, not a simulated wheel-compression event.

In the terminal tables, the subscript c denotes the first time this clearance threshold is reached. The quantities N_c and E_c are the downrange and lateral positions at threshold contact, $h_{\mathrm{mg},c}$ is the main-gear clearance at that instant, and $\dot{h}_{\mathrm{mg},c}$ is the corresponding clearance rate. Negative $\dot{h}_{\mathrm{mg},c}$ indicates that the main gear is moving downward toward the runway. The sink rate $V_{\mathrm{sink},c}$ is reported as a positive downward speed, and θ_c is the aircraft pitch angle at threshold contact.

Table 5.4 summarizes the terminal landing metrics for the baseline TVLQR-only cases. This baseline uses the nominal zero-wind gain schedule with no EKF synchronization and no digital-twin terminal intervention. Here, the subscript c denotes the first main-gear contact event. N_c and E_c are the aircraft downrange and lateral positions at first contact, expressed in the local NED runway frame. $h_{\mathrm{mg},c}$ is the main-gear clearance above the runway at first contact, and $\dot{h}_{\mathrm{mg},c}$ is its time derivative. A negative value of $\dot{h}_{\mathrm{mg},c}$ means the main gear is moving downward toward the runway. The sink rate $V_{\mathrm{sink},c}$ is reported as a positive downward vertical speed at contact. Finally, θ_c is the aircraft pitch angle at first contact, where negative values indicate a nose-down attitude relative to the local level frame. Main-gear contact is defined using the clearance threshold from the terminal diagnostic script, with first contact recorded when the main-gear clearance reaches approximately 0.09 m. The table reports both the touchdown condition at first contact and the minimum main-gear clearance reached later in the terminal segment.

Table 5.4: Terminal landing metrics for the baseline TVLQR-only tracking cases.

Case	Main Gear contact?	t_c (s)	N_c (m)	E_c (m)	$h_{\mathrm{mg},c}$ (m)	$\dot{h}_{\mathrm{mg},c}$ (m/s)	$V_{\mathrm{sink},c}$ (m/s)	θ_c (deg)
No-disturbance	Yes	50.390	-28.0	0.09	0.089	-2.608	2.401	-1.35
Moderate wind	Yes	50.315	44.6	-6.12	0.090	-2.656	2.411	-1.53
Severe wind	Yes	49.960	133.5	-20.36	0.088	-3.474	3.276	-1.99

Table 5.5: Additional terminal diagnostics for the baseline TVLQR-only tracking cases.

Case	α_c (deg)	u_c (m/s)	$h_{\mathrm{mg,min}}$ (m)	$t(h_{\mathrm{mg,min}})$ (s)
No-disturbance	0.25	85.81	-0.127	50.620
Moderate wind	0.22	84.95	-0.119	50.510
Severe wind	-0.37	83.80	-0.269	50.200

In the no-disturbance case, the aircraft reaches contact near the runway centerline with a sink rate of

about $2.40\ \mathrm{m/s}$ and a slightly nose-down pitch attitude. The moderate wind case still achieves contact with similar sink rate and clearance rate, but the touchdown point shifts downrange and laterally, indicating that the nominal TVLQR schedule alone does not fully reject the wind-driven position error. The severe wind case shows the clearest degradation. Contact occurs earlier in time but much farther downrange and laterally offset, with the sink rate increasing to $3.28\ \mathrm{m/s}$, the main-gear clearance rate increasing in magnitude to $3.47\ \mathrm{m/s}$, and the minimum clearance dropping to $-0.269\ \mathrm{m}$.

The tracking results suggests that the TVLQR controller is able to very closely track the nominal trajectory when there are no modeled perturbations. The introduction of wind, gusts, turbulence, and initial perturbations causes a monotonic degradation in tracking accuracy, with the dominant error appearing as accumulated drift in the lateral ground track position over time. Even under the stress-test case, the closed-loop response remains bounded and actuator saturation is avoided. The controller is therefore still actively course-correcting, but it is doing so with a local linear design synthesized about the nominal trajectory. Under the presence of unmodeled disturbances, a phase drift accumulates in the airplanes position.

Chapter 6

DIGITAL TWIN SYNCHRONIZATION USING EXTENDED KALMAN FILTERING

The Extended Kalman Filter (EKF) is used as the synchronization method between an imperfect digital twin and the higher-fidelity plant proxy during closed-loop landing trajectory tracking. The EKF filters noisy measurements and recursively estimates the wind-related mismatch state used to synchronize the digital twin with the plant proxy during closed-loop execution. The estimator must also operate in near real-time with minimal computational latency. Unlike the standard linear Kalman filter, the EKF may be used with nonlinear process and measurement models while retaining a compact recursive structure which makes it more suitable for navigation compared to more computationally demanding methods such as nonlinear Bayesian estimators. [36, 37].

Whether augmented wind, air-data, or sensor-bias states can be reliably identified depends on the available measurements, flight conditions, and the degree of trajectory excitation. These limitations can be seen in many existing aircraft navigation and air-data estimation literature. Onse such study used a single-antenna GPS receiver and pitot tube to estimate horizontal wind and an airspeed scaling factor for a small UAV, but showed that the sensor combination alone does not directly determine the wind without changes in aircraft heading, such as banking turns or circular maneuvers [38]. Another study implemented a cascaded EKF architecture using MEMS IMU, GNSS, and air-data measurements, emphasizing a modular estimator structure that avoids unnecessary state augmentation and reduces covariance propagation cost and tuning complexity [39]. Others have further showed that model-free air-data estimation using GPS, IMU, and pitot-tube measurements is only conditionally observable and that wind and flow-angle estimates improve with sufficiently informative maneuvers, but their accuracy remains limited by wind uncertainty, attitude accuracy, airspeed, and sensor quality [40].

A digital twin can generally be described as a synchronized model that uses data from the physical system to represent, predict, optimize, and support decision making [41]. In this thesis, the digital twin contains the 6-DoF nonlinear aircraft dynamics introduced in Section 3, while the EKF acts as the data-assimilation layer used to keep that twin synchronized during closed-loop execution. The EKF process

model uses the measured body-axis velocity and angular-rate channels as kinematic inputs. Position is propagated by rotating body-axis air-relative velocity into the local NED frame and adding the estimated wind velocity. Attitude is propagated using the Euler-rate kinematics, and the wind state is modeled as a first-order Gauss–Markov process. The measurement model uses noisy local pose measurements and ground-referenced velocity measurements. In this structure, discrepancies between measured ground motion and the air-relative motion implied by the current pose and body velocity provide information about the wind state, while the pose measurements prevent the synchronized twin from receiving direct, perfect plant-state access.

The digital twin remains responsible for nonlinear aircraft propagation, tracking-support, and short-horizon terminal consequence prediction. The EKF works to provide a filtered estimate of the current pose and wind condition used to initialize and correct the synchronized digital twin. The process and measurement mappings are nonlinear because the navigation update depends on attitude-dependent frame transformations, the Euler-rate equations depend on the current attitude, and the ground-velocity measurement depends jointly on attitude, body-axis velocity, and wind. The extended Kalman filter preserves the recursive prediction-update structure of the Kalman filter while relinearizing these nonlinear mappings about the current estimate through local Jacobians [42, 43, 44]. This keeps the estimator compact enough for online use while still capturing the main nonlinearities needed for pose-and-wind synchronization.

6.1 Estimation Problem for Digital Twin Synchronization

The purpose of the estimator is to keep the online digital twin aligned with the plant proxy during closed-loop landing execution. The estimator provides a filtered estimate of the current local pose and wind condition. This estimate is then used to keep the synchronized twin initialized near the measured plant trajectory without directly assigning the full true plant state to the twin.

The EKF is formulated as a discrete-time stochastic state-estimation problem. Let X_k denote the estimator state, U_k denote known estimator inputs, and Y_k denote the measurement vector available at sample time t_k. The general nonlinear stochastic model can be written as

$$X_{k+1} = f_d(X_k, U_k) + V_k \tag{6.1a}$$

$$Y_k = h(X_k, U_k) + W_k \tag{6.1b}$$

where V_k is the process noise and W_k is the measurement noise. The estimator assumes $\mathbb{E}[V_k] = 0$, $\mathbb{E}[W_k] = 0$, $Q_k = \mathrm{Cov}(V_k)$, and $R_k = \mathrm{Cov}(W_k)$. The process covariance Q_k represents uncertainty in the state propagation model, while the measurement covariance R_k represents uncertainty in the measured sensor channels. These matrices are estimator covariance parameters and are different from the TVLQR cost weights used in Section 5. In the present landing problem, the nonlinearities come from attitude-dependent frame transformations, Euler-rate kinematics, and wind-corrected ground-velocity relations. The EKF therefore, propagates the nonlinear airplane model and uses local Jacobians to approximate the nearby error dynamics. To avoid confusing the EKF estimator state, inputs, and measurement vectors with previously defined aircraft state and input definitions, denote the EKF variables with $X_k \equiv x_{\mathrm{ekf},k}$ and $U_k \equiv u_{\mathrm{ekf},k}$. The EKF estimate and covariance after conditioning on measurements through time t_k are denoted by

$$\hat{x}_{\mathrm{ekf},k|k} = \mathbb{E}[x_{\mathrm{ekf},k} \mid y_{1:k}] \quad \text{under the EKF approximation,} \qquad P_{k|k} = \mathrm{Cov}\left(x_{\mathrm{ekf},k} - \hat{x}_{\mathrm{ekf},k|k}\right)$$

Here $y_{1:k} = \{y_1, \ldots, y_k\}$ denotes the measurement history. The covariance $P_{k|k}$ is the filter's internal estimate of uncertainty in the pose-and-wind state. The EKF state is chosen to capture the reduced set of variables needed for digital-twin synchronization:

$$x_{\mathrm{ekf},k} = \begin{bmatrix} r_{N,k}^\top & \eta_k^\top & w_{N,k}^\top \end{bmatrix}^\top = \begin{bmatrix} N_k & E_k & D_k & \phi_k & \theta_k & \psi_k & w_{N,k} & w_{E,k} & w_{D,k} \end{bmatrix}^\top \tag{6.2}$$

where $r_N = [N, E, D]^\top$ is the local runway-aligned NED position, $\eta = [\phi, \theta, \psi]^\top$ is the Euler attitude, and $w_N = [w_N, w_E, w_D]^\top$ is the wind vector expressed in the same NED frame. This state is lower dimensional than the full aircraft state introduced in Section 3 and lower dimensional than the augmented TVLQR tracking state in Section 5. The EKF is therefore not a full aircraft-dynamics estimator. It estimates the local pose and wind information needed to synchronize the digital twin.

At each estimator update, the simulated plant proxy provides a measurement stream that is treated as the available sensor information for synchronization. The measurement vector contains local pose and ground-referenced velocity:

$$y_k = \begin{bmatrix} N_k & E_k & D_k & \phi_k & \theta_k & \psi_k & V_{N,k} & V_{E,k} & V_{D,k} \end{bmatrix}^\top + W_k, \qquad W_k \sim \mathcal{N}(0, R_k) \tag{6.3}$$

The position and attitude measurements provide direct information about the aircraft pose, while the ground-velocity measurement provides information about wind when compared against the air-relative velocity implied by the current attitude and body-axis velocity. The measurement noise covariance R_k controls how

strongly the filter trusts these measured pose and velocity channels. The EKF process model is driven by the rigid-body velocity and angular-rate channels

$$u_{\mathrm{ekf},k} = \begin{bmatrix} u_k & v_k & w_k & p_k & q_k & r_k \end{bmatrix}^\top \tag{6.4}$$

where (u, v, w) are body-axis air-relative velocity components and (p, q, r) are body-axis angular rates. These quantities are part of the aircraft rigid-body state defined in Section 3. They are treated as known kinematic inputs to the EKF over one estimator update. This input vector is not the same as the commanded aircraft control input $u_{c,k}$ used by the TVLQR controller. The commanded input is still logged and used by the plant and digital-twin propagation, but the EKF itself uses the kinematic input in Eq. (6.4) to propagate the reduced pose-and-wind state.

The position update is based on rotating the body-axis air-relative velocity into the NED frame and adding the estimated wind contribution. The attitude update is based on the Euler-rate kinematics. The wind update is modeled as a first-order Gauss–Markov process. The process noise covariance Q_k accounts for uncertainty in this reduced propagation model, including unmodeled pose propagation error and wind variability. This avoids requiring the filter to identify actuator lag, aerodynamic parameters, propulsion states, or contact behavior.

The plant proxy simulation integrates the RCAM aircraft dynamics at a higher internal rate than the outer controller and estimator updates. In the runs reported here, the outer controller and logged results are evaluated at 20 Hz, corresponding to $\Delta t_s = 0.05$ s. Within each outer step, the RCAM dynamics are propagated using ten internal substeps, giving an internal dynamics step of 0.005 s, or 200 Hz. The augmented EKF is updated on the same 20 Hz controller grid. Defining $t_k = k\Delta t_s$, the EKF receives the measurement vector y_k and kinematic input $u_{\mathrm{ekf},k}$ at each estimator sample time. Between estimator updates, the input is treated as piecewise constant over $[t_k, t_{k+1})$. This is an estimator approximation and is separate from the higher-rate numerical integration used inside the plant simulation.

Because the plant proxy is simulated, the logged variables are treated as true state variables before measurement noise is added. Noise can therefore be introduced at the measurement-model level to avoid giving the synchronization algorithm unrealistically accurate information. Under this setup, R_k describes the injected sensor uncertainty, Q_k describes uncertainty in the reduced pose-and-wind propagation model, and $P_{k|k}$ describes the EKF's evolving uncertainty in the synchronized pose-and-wind estimate.

6.2 Extended Kalman Filter Formulation

At each estimator step, the filter receives the input $u_{\mathrm{ekf},k}$, the measurement y_k, and the previous estimate $\hat{x}_{\mathrm{ekf},k|k}$ with covariance $P_{k|k}$. It then computes a predicted estimate, compares the predicted measurement against the incoming measurement, and applies a covariance-weighted correction. Using the notation from Section 6.1, the nonlinear discrete-time estimation model is

$$x_{\mathrm{ekf},k+1} = f_d(x_{\mathrm{ekf},k}, u_{\mathrm{ekf},k}) + q_k, \qquad q_k \sim \mathcal{N}(0, Q_k) \tag{6.5a}$$

$$y_k = h(x_{\mathrm{ekf},k}, u_{\mathrm{ekf},k}) + r_k, \qquad r_k \sim \mathcal{N}(0, R_k) \tag{6.5b}$$

where $f_d(\cdot)$ is the discrete-time process map over one estimator sample interval, and $h(\cdot)$ maps the EKF state and kinematic input to the expected measurement. The covariance Q_k models uncertainty in the pose-and-wind propagation, while R_k models uncertainty in the measured pose and ground-velocity channels. The input $u_{\mathrm{ekf},k}$ is treated as known over the estimator update interval.

If f_d and h were linear, the Kalman filter would propagate the conditional mean and covariance exactly under the standard Gaussian assumptions. In this problem, both maps are nonlinear because of attitude-dependent frame transformations, Euler-rate kinematics, and wind-corrected ground-velocity relations. The EKF handles this by replacing the nonlinear maps with local first-order approximations about the current estimate. This makes the filter locally valid. If the estimate moves too far from the true state, the linearized error model may no longer represent the actual local dynamics accurately. For a differentiable map $g(x_{\mathrm{ekf}})$, the local approximation about $\bar{x}_{\mathrm{ekf}}$ is

$$g(x_{\mathrm{ekf}}) \approx g(\bar{x}_{\mathrm{ekf}}) + \left.\frac{\partial g}{\partial x_{\mathrm{ekf}}}\right|_{\bar{x}_{\mathrm{ekf}}} (x_{\mathrm{ekf}} - \bar{x}_{\mathrm{ekf}})$$

The prediction step applies this idea to the process map f_d, while the conditioning step applies it to the measurement map h.

6.2.1 Prediction Step

Suppose that at time t_k, the filter has the estimate $x_{\mathrm{ekf},k|k}$ and covariance $P_{k|k}$. Define the estimation error as

$$e_{k|k} = x_{\mathrm{ekf},k} - \hat{x}_{\mathrm{ekf},k|k}$$

The process model can be expanded around the current estimate:

$$\begin{aligned} x_{\text{ekf},k+1} &= f_d(x_{\text{ekf},k}, u_{\text{ekf},k}) + q_k \\ &= f_d(\hat{x}_{\text{ekf},k|k} + e_{k|k}, u_{\text{ekf},k}) + q_k \\ &\approx f_d(\hat{x}_{\text{ekf},k|k}, u_{\text{ekf},k}) + F_k e_{k|k} + q_k \end{aligned}$$

where the Jacobian is

$$F_k = \left.\frac{\partial f_d}{\partial x_{\text{ekf}}}\right|_{\hat{x}_{\text{ekf},k|k}, u_{\text{ekf},k}}$$

Taking the conditional expectation given the measurement history through t_k gives the predicted mean

$$\hat{x}_{\text{ekf},k+1|k} = f_d(\hat{x}_{\text{ekf},k|k}, u_{\text{ekf},k}) \tag{6.6}$$

This follows because the current estimation error is assumed to be zero mean under the filter approximation, and the process noise is modeled as zero mean. The predicted error is then

$$e_{k+1|k} = x_{\text{ekf},k+1} - \hat{x}_{\text{ekf},k+1|k} \approx F_k e_{k|k} + q_k$$

Using this linearized error propagation, the predicted covariance becomes

$$P_{k+1|k} = F_k P_{k|k} F_k^\top + Q_k \tag{6.7}$$

Thus, the prediction step propagates the best current estimate through the nonlinear process model, while the covariance is propagated through the local linearization of that model. For the i-th EKF state component, a first-order finite-difference approximation is used:

$$F_k^{(:,i)} \approx \frac{f_d(\hat{x}_{\text{ekf},k|k} + \epsilon_i e_i, u_{\text{ekf},k}) - f_d(\hat{x}_{\text{ekf},k|k}, u_{\text{ekf},k})}{\epsilon_i},$$

where e_i is the i-th unit vector and ϵ_i is the finite-difference perturbation for that state component. Let

$$v_{B,k} = \begin{bmatrix} u_k & v_k & w_k \end{bmatrix}^\top, \qquad \omega_{B,k} = \begin{bmatrix} p_k & q_k & r_k \end{bmatrix}^\top$$

and recall that $x_{\text{ekf},k} = \begin{bmatrix} r_{N,k}^\top & \eta_k^\top & w_{N,k}^\top \end{bmatrix}^\top$. For the pose-and-wind process model used in Section 6.3, the discrete update is

$$r_{N,k+1} \approx r_{N,k} + \Delta t_s \left[R_{NB}(\eta_k) v_{B,k} + w_{N,k} \right],$$

$$\eta_{k+1} \approx \eta_k + \Delta t_s \mathcal{E}(\eta_k) \omega_{B,k},$$

$$w_{N,k+1} = \Phi_w w_{N,k}$$

The Jacobian has the form

$$F_k \approx \begin{bmatrix} I_3 & \Delta t_s \left. \dfrac{\partial (R_{NB}(\eta) v_{B,k})}{\partial \eta} \right|_{\hat{\eta}_{k|k}} & \Delta t_s I_3 \\ 0 & I_3 + \Delta t_s \left. \dfrac{\partial (\mathcal{E}(\eta) \omega_{B,k})}{\partial \eta} \right|_{\hat{\eta}_{k|k}} & 0 \\ 0 & 0 & \Phi_w \end{bmatrix} \tag{6.8}$$

Position uncertainty depends on position, attitude, and wind uncertainty. Attitude uncertainty builds from the propagation of the Euler-rate kinematics. The wind state evolves independently from position and attitude using the first-order Gauss–Markov model, which gives the lower-right block Φ_w.

6.2.2 Conditioning Step

After prediction, the estimator receives the next measurement y_{k+1}. The measurement model is linearized about the predicted state $\hat{x}_{\text{ekf},k+1|k}$

$$\begin{aligned} y_{k+1} &= h(x_{\text{ekf},k+1}, u_{\text{ekf},k+1}) + r_{k+1} \\ &= h(\hat{x}_{\text{ekf},k+1|k} + e_{k+1|k}, u_{\text{ekf},k+1}) + r_{k+1} \\ &\approx h(\hat{x}_{\text{ekf},k+1|k}, u_{\text{ekf},k+1}) + H_{k+1} e_{k+1|k} + r_{k+1} \end{aligned} \tag{6.9}$$

where the measurement Jacobian is

$$H_{k+1} = \left. \frac{\partial h}{\partial x_{\text{ekf}}} \right|_{\hat{x}_{\text{ekf},k+1|k}, u_{\text{ekf},k+1}}$$

The innovation is the difference between the incoming measurement and the measurement predicted from the current estimate:

$$\nu_{k+1} = \mathcal{W} \left(y_{k+1} - h(\hat{x}_{\text{ekf},k+1|k}, u_{\text{ekf},k+1}) \right) \tag{6.10}$$

The operator $\mathcal{W}(\cdot)$ wraps the Euler-angle components of the residual onto $(-\pi, \pi]$ and leaves the position and velocity components unchanged. This avoids treating equivalent attitudes near the $\pm\pi$ boundary as large errors. Using the linearized measurement model, the innovation can be approximated as

$$\nu_{k+1} \approx H_{k+1} e_{k+1|k} + r_{k+1} \tag{6.11}$$

The innovation covariance is then

$$S_{k+1} = H_{k+1} P_{k+1|k} H_{k+1}^{\top} + R_{k+1} \tag{6.12}$$

The Kalman gain is

$$K_{k+1} = P_{k+1|k} H_{k+1}^{\top} S_{k+1}^{-1} \tag{6.13}$$

This gain determines how strongly the filter corrects the predicted estimate using the new measurement. (It's important to note that the EKF gain in this section is different from the TVLQR gains computed offline around the nominal landing trajectory in the previous section. The EKF gain is computed online from the current covariance and local measurement linearization). A larger predicted covariance or smaller measurement covariance generally increases the correction. A smaller predicted covariance or larger measurement covariance makes the update more conservative. The conditioned state estimate is

$$\hat{x}_{\mathrm{ekf},k+1|k+1} = \hat{x}_{\mathrm{ekf},k+1|k} + K_{k+1} \nu_{k+1} \tag{6.14}$$

This update can be interpreted as first predicting the pose-and-wind state with the process model, then correcting the predicted error using the part of the measurement residual that is consistent with the local linearized observation model. The covariance update can then be expressed as

$$P_{k+1|k+1} = P_{k+1|k} - P_{k+1|k} H_{k+1}^{\top} S_{k+1}^{-1} H_{k+1} P_{k+1|k}$$

The Joseph form is algebraically equivalent to the compact form under exact arithmetic, but it is usually more reliable numerically because it better preserves symmetry and positive semidefiniteness under finite precision.

$$P_{k+1|k+1} = (I - K_{k+1} H_{k+1}) P_{k+1|k} (I - K_{k+1} H_{k+1})^{\top} + K_{k+1} R_{k+1} K_{k+1}^{\top} \tag{6.15}$$

Similar to the prediction step, the measurement Jacobian is evaluated using finite differences. For the i-th EKF state component,

$$H_{k+1}^{(:,i)} \approx \frac{h(\hat{x}_{\mathrm{ekf},k+1|k} + \epsilon_i e_i, u_{\mathrm{ekf},k+1}) - h(\hat{x}_{\mathrm{ekf},k+1|k}, u_{\mathrm{ekf},k+1})}{\epsilon_i}$$

For the pose-and-wind measurement model, the measurement Jacobian contains direct pose-measurement terms and ground-velocity terms that depend on attitude and wind. These dependencies are what allow the filter to estimate wind from the difference between measured ground-referenced velocity and the air-relative velocity implied by the current body velocity and attitude estimate.

Innovation and Covariance Diagnostics The EKF covariance is an internal estimate of uncertainty and a useful diagnostic that shows how much uncertainty the filter assigns to each state channel and how strongly new measurements are being trusted. The innovation sequence is also useful because it measures the discrepancy between the predicted measurement and the incoming measurement before the correction is applied. A scalar consistency diagnostic is the normalized innovation squared (NIS)

$$\mathrm{NIS}_{k+1} = \nu_{k+1}^{\top} S_{k+1}^{-1} \nu_{k+1}$$

Large NIS values indicate that the observed measurement residual is large relative to the covariance predicted by the filter. This may occur when the measurement noise is underestimated, the process model is too confident, or the nonlinear approximation is poor. Small values can indicate conservative covariance tuning or measurement residuals that are smaller than expected. The component-wise normalized innovation can also be written as

$$\bar{\nu}_{j,k+1} = \frac{\nu_{j,k+1}}{\sqrt{S_{jj,k+1}}}$$

where $S_{jj,k+1}$ is the j-th diagonal entry of the innovation covariance. These normalized residuals are useful for checking whether a specific measurement channel is poorly tuned or systematically biased.

6.3 Pose-and-Wind EKF for the Aircraft Landing Problem

The continuous-time process model is written as

$$\dot{r}_N = R_{NB}(\eta)\, v_B + w_N \tag{6.16a}$$

$$\dot{\eta} = \mathcal{E}(\eta)\, \omega_B \tag{6.16b}$$

$$\dot{w}_N = -T_w^{-1} w_N + n_w \tag{6.16c}$$

In Eq. (6.16a), $R_{NB}(\eta)$ maps body-frame velocity components into the local NED frame. The resulting air-relative NED velocity is then corrected by the wind estimate to obtain the ground-referenced position rate. In Eq. (6.16b), $\mathcal{E}(\eta)$ is the standard Euler-rate transformation for the 3-2-1 yaw-pitch-roll convention. The wind model in Eq. (6.16c) is a first-order Gauss–Markov process, where

$$T_w = \mathrm{diag}(\tau_N, \tau_E, \tau_D) \tag{6.17}$$

contains the wind correlation time constants and n_w represents zero-mean process excitation. Over one estimator sample interval Δt_s, the wind update can be written in discrete form as

$$w_{N,k+1} = \Phi_w w_{N,k} + q_{w,k}, \qquad \Phi_w = \mathrm{diag}\left(e^{-\Delta t_s/\tau_N}, e^{-\Delta t_s/\tau_E}, e^{-\Delta t_s/\tau_D}\right) \tag{6.18}$$

The nonlinear process map used by the EKF is the discrete-time version of Eqs. (6.16a)–(6.16c):

$$x_{\mathrm{ekf},k+1} = f_d(x_{\mathrm{ekf},k}, u_{\mathrm{ekf},k}) + q_k, \qquad q_k \sim \mathcal{N}(0, Q_k)$$

The local nonlinearities in this process model come from the attitude-dependent rotation $R_{NB}(\eta)$, the Euler-rate map $\mathcal{E}(\eta)$, and the coupling between the wind estimate and the local position propagation. The nonlinear measurement model is

$$h(x_{\mathrm{ekf},k}, u_{\mathrm{ekf},k}) = \begin{bmatrix} r_{N,k} \\ \eta_k \\ R_{NB}(\eta_k) v_{B,k} + w_{N,k} \end{bmatrix}$$

The first two blocks return the estimated local position and attitude. The third block predicts the ground-referenced velocity by rotating the measured body-axis air-relative velocity into the NED frame and adding the estimated wind. Therefore, the velocity measurement helps identify wind through the difference between

measured ground motion and the air-relative motion implied by the current attitude and body velocity. The process and measurement covariance matrices are written using diagonal standard-deviation matrices. Let

$$Q_k = \Sigma_{q,k}\Sigma_{q,k}^{\top}, \qquad R_k = \Sigma_{y,k}\Sigma_{y,k}^{\top} \tag{6.19}$$

where $\Sigma_{q,k}$ contains the assumed one-step process standard deviations and $\Sigma_{y,k}$ contains the measurement standard deviations. For the pose uncertainty, the discrete pose increments are computed using

$$\Sigma_{q,r} = \Delta t_s \,\mathrm{diag}\left(\sigma_{\dot{N}}, \sigma_{\dot{E}}, \sigma_{\dot{D}}\right), \qquad \Sigma_{q,\eta} = \Delta t_s \,\mathrm{diag}\left(\sigma_{\dot{\phi}}, \sigma_{\dot{\theta}}, \sigma_{\dot{\psi}}\right) \tag{6.20}$$

For the Gauss–Markov wind state, the discrete wind process variance for component $i \in \{N, E, D\}$ is calculated as

$$\sigma_{q,w_i}^2 = \frac{1}{2}\sigma_{w_i}^2 \tau_i \left(1 - e^{-2\Delta t_s/\tau_i}\right) \tag{6.21}$$

where σ_{w_i} is the wind process excitation parameter and τ_i is the corresponding time constant. The full process standard-deviation matrix is then

$$\Sigma_{q,k} = \mathrm{diag}\left(\Sigma_{q,r}, \Sigma_{q,\eta}, \sigma_{q,w_N}, \sigma_{q,w_E}, \sigma_{q,w_D}\right) \tag{6.22}$$

with the block notation in Eq. (6.22) representing the placement of the diagonal entries of $\Sigma_{q,r}$ and $\Sigma_{q,\eta}$ into the full diagonal matrix.

The measurement standard-deviation matrix is

$$\Sigma_{y,k} = \mathrm{diag}\left(\sigma_N, \sigma_E, \sigma_D, \sigma_\phi, \sigma_\theta, \sigma_\psi, \sigma_{V_N}, \sigma_{V_E}, \sigma_{V_D}\right) \tag{6.23}$$

Thus, R_k influences how much the EKF trusts the measured local pose and ground-velocity channels relative to the process prediction. Larger measurement standard deviations make the update more conservative, while smaller values cause the filter to follow the measurement stream more closely. The nominal covariance parameters used for the pose-and-wind EKF are summarized in Table 6.1. Attitude values are shown in degrees

Table 6.1: Nominal pose-and-wind EKF covariance and wind-model parameters.

Quantity	Symbol	Value
Wind time constants	$[\tau_N, \tau_E, \tau_D]$	$[35,\ 35,\ 18]$ s
Wind process excitation	$[\sigma_{w_N}, \sigma_{w_E}, \sigma_{w_D}]$	$[0.20,\ 0.20,\ 0.12]$ m/s
Position process std.	$[\sigma_{\dot{N}}, \sigma_{\dot{E}}, \sigma_{\dot{D}}]$	$[0.08,\ 0.08,\ 0.04]$ m/s
Attitude process std.	$[\sigma_{\dot{\phi}}, \sigma_{\dot{\theta}}, \sigma_{\dot{\psi}}]$	$[0.08,\ 0.08,\ 0.10]$ deg/s
Position measurement std.	$[\sigma_N, \sigma_E, \sigma_D]$	$[0.50,\ 0.50,\ 0.25]$ m
Attitude measurement std.	$[\sigma_\phi, \sigma_\theta, \sigma_\psi]$	$[0.15,\ 0.15,\ 0.25]$ deg
Ground-velocity measurement std.	$[\sigma_{V_N}, \sigma_{V_E}, \sigma_{V_D}]$	$[0.35,\ 0.35,\ 0.20]$ m/s
Initial position covariance std.	$[\sigma_{N,0}, \sigma_{E,0}, \sigma_{D,0}]$	$[2,\ 2,\ 1]$ m
Initial attitude covariance std.	$[\sigma_{\phi,0}, \sigma_{\theta,0}, \sigma_{\psi,0}]$	$[1,\ 1,\ 2]$ deg
Initial wind covariance std.	$[\sigma_{w_N,0}, \sigma_{w_E,0}, \sigma_{w_D,0}]$	$[3,\ 3,\ 1.5]$ m/s

The process covariance determines how much model error and wind variability the filter expects between updates. The measurement covariance determines how strongly the filter trusts the pose and ground-velocity measurements. In the landing simulations, these choices were tuned to keep the synchronized digital twin responsive to plant motion while avoiding aggressive corrections.

6.4 EKF Synchronization and Tuning

The EKF is treated as an online synchronization mechanism that runs concurrently with the nonlinear plant proxy and the synchronized digital twin. At each estimator update, the EKF uses the measurement vector y_k and kinematic input $u_{\mathrm{ekf},k}$ to update the filtered pose-and-wind estimate $\hat{x}_{\mathrm{ekf},k|k}$. The synchronized digital twin continues to propagate with the nonlinear aircraft model, while the EKF supplies filtered pose and wind information used to keep the digital twin close to the current plant trajectory.

In this section, $\hat{x}_k^s$ denotes the synchronized digital-twin state, while $\hat{x}_{\mathrm{ekf},k|k}$ denotes the conditioned EKF estimate. The synchronized twin state contains the aircraft variables propagated by the nonlinear twin model. The EKF state is lower dimensional and contains only the local position, attitude, and wind components defined in Eq. (6.2). The corresponding EKF components are denoted by $\hat{r}_{N,k}^{\mathrm{ekf}}$, $\hat{\eta}_k^{\mathrm{ekf}}$, and $\hat{w}_{N,k}^{\mathrm{ekf}}$.

The synchronized twin is propagated with the nonlinear digital twin dynamics between estimator updates and then aligned using the filtered EKF estimate. This produces a data-assimilated twin state that is used for tracking support and short-horizon terminal prediction. The digital twin is first propagated using the

commanded input and the current EKF wind estimate:

$$\hat{x}_{k+1}^{s,-} = f_t\left(\hat{x}_k^s, u_{c,k}; \theta_t, \hat{w}_{N,k}^{\text{ekf}}\right) \tag{6.24}$$

where $f_t(\cdot)$ is the nonlinear digital-twin propagation model and the superscript $(-)$ denotes the pre-synchronization value. The EKF wind estimate is the main environmental synchronization channel. Without this update, the digital twin would continue to propagate under the planning-model assumption of no wind, which leads to accumulated position and timing drift in the disturbed plant cases. After the nonlinear propagation, selected components of the synchronized twin are updated with the conditioned EKF estimate:

$$\hat{x}_{k+1}^s = \mathcal{C}_{\text{sync}}\left(\hat{x}_{k+1}^{s,-}, \hat{x}_{\text{ekf},k+1|k+1}\right) \tag{6.25}$$

where $\mathcal{C}_{\text{sync}}(\cdot)$ applies the corrections to the digital twin state using the filtered pose-and-wind estimate. It does not replace the full aircraft state with the true plant state. Unestimated states, actuator states, and terminal contact behavior remain part of the nonlinear twin and plant propagation. The current state is informed by measurements through the EKF, while the future rollout used for terminal prediction is still generated by the nonlinear digital-twin dynamics. Algorithm 3 summarizes the online synchronization loop. The algorithm is written at the estimator and synchronization level. The tracking-controller and terminal-policy logic that use the synchronized state are described in the following sections.

Algorithm 3: Online pose-and-wind EKF synchronization of the digital twin

Input: Initial EKF estimate $\hat{x}_{\text{ekf},0|0}$, covariance $P_{0|0}$, synchronized twin state $\hat{x}_0^s$, covariances Q_k, R_k, sample time Δt_s
Output: Synchronized twin state, EKF estimate history, and diagnostic logs
Initialize the plant proxy, synchronized twin, and EKF histories
for *each estimator update* t_k **do**
  Receive the measurement y_k, kinematic EKF input $u_{\text{ekf},k}$, and commanded input $u_{c,k}$
  `// EKF prediction and conditioning`
  Propagate the EKF state and covariance using the prediction step in Eqs. (6.6)–(6.7)
  Compute the wrapped innovation, Kalman gain, and conditioned estimate using Eqs. (6.10)–(6.14)
  Update the covariance using the Joseph form in Eq. (6.15), then symmetrize and floor it as needed
  `// Synchronized twin update`
  Propagate the nonlinear digital twin with $u_{c,k}$ and the EKF wind estimate using Eq. (6.24)
  Apply the synchronization operator in Eq. (6.25) using the conditioned EKF estimate
  `// Diagnostics`
  Log the EKF estimate, innovation, NIS, covariance diagonal, minimum covariance eigenvalue, wind-estimation error, and synchronized-twin mismatch
end

The pose-and-wind synchronization uses small correction gains so that the digital twin simulated states get influenced by the EKF estimate and not overwritten by it. The digital twin model still propagates the

trajectory with a nonlinear airplane model, while the EKF is only the measurement model able to provide accurate estimates of the pose and unmodeled disturbance. The final configuration uses a position synchronization gain of $\gamma_r = 0.08$, an attitude synchronization gain of $\gamma_\eta = 0.02$, and a velocity synchronization gain of $\gamma_v = 0.02$. The velocity correction is limited to $1.5\ \mathrm{m/s}$. This helps prevent the synchronized twin from being aggressively corrected during the final contact geometry, where small velocity changes can strongly affect predicted clearance rate and touchdown behavior.

Several diagnostics are recorded during each run. The EKF logs the state estimate, innovation vector, normalized innovation squared, covariance diagonal, minimum covariance eigenvalue, and wind-estimation error. The digital-twin synchronization layer logs the baseline twin mismatch and the synchronized twin mismatch relative to the plant. The innovation and NIS histories assess whether the measurement residuals are consistent with the assumed covariance tuning. The covariance diagonal and minimum eigenvalue track numerical health. The wind error evaluates whether the Gauss–Markov wind state is being recovered. The baseline and synchronized twin mismatch histories show whether EKF synchronization actually improves the model state used for terminal forecasts.

6.5 Online Tracking Support

The synchronized digital twin is used during online tracking to reduce the phase drift and lateral bias that appear when the plant is exposed to wind disturbances that are not included in the nominal planning model. The nominal trajectory and cached TVLQR gain schedule are generated using the simplified twin model. Over the trajectory, wind perturbations change the plant ground track and timing relative to that nominal reference. Without online synchronization, this mismatch can cause the plant to separate gradually from the reference. In other works, the desired air-relative direction or path geometry is modified using various path-following approaches. For instance, Lu et al. [45] use a wind disturbance estimate in the nominal guiding vector to produce a desired airspeed direction in the inner loop of the controller, while Beard et al. [46] apply robust path-following techniques for UAVs tracking linear segments. Furthermore, real-time wind velocity estimates have been successfully utilized to continuously adapt path-planning and tracking references mid-flight, ensuring the kinematic feasibility of the trajectory under varying local wind fields [47]. In this thesis, the offline Riccati gain schedule is retained, while the EKF-synchronized digital twin updates the reference progress and heading bias used by the online controller.

At each controller update, the tracking logic uses the most recent conditioned EKF estimate, $\hat{x}_{\mathrm{ekf},k|k}$, and the current synchronized twin state, $\hat{x}_k^s$. The EKF estimate provides the local pose and wind components defined in Eq. (6.2), while the synchronized twin provides the nonlinear aircraft state used for tracking support and short-horizon prediction. If the estimator and controller are evaluated at different rates, the controller uses the latest available EKF estimate until the next estimator update. The tracked reference remains the nominal TVLQR reference. Synchronization does not recompute the TVLQR gains or replace the nominal trajectory. Instead, it modifies how the reference is evaluated online through two support terms: trajectory phase synchronization and a bounded lateral heading feedforward. The first term adjusts the reference timing using the synchronized twin position, while the second biases the reference heading into the estimated crosswind. The synchronized digital twin is updated using the filtered EKF estimate so that the state used for progress estimation remains aligned with the measured trajectory.

Trajectory Phase Synchronization. Standard trajectory-tracking controllers enforce strict temporal constraints, evaluating the reference state strictly as a function of clock time t_k. In the presence of a headwind or tailwind, clock time may no longer be an accurate indicator of the aircraft position along the nominal reference. A headwind reduces ground speed relative to airspeed, while a tailwind increases it. If the controller evaluates the reference only using clock time, the plant can become phase-shifted relative to the nominal trajectory even when the local attitude and velocity tracking remain reasonable. Parameterizing the reference curve with a spatial progress variable, often referred to as a *virtual target*, is a common path-following technique used to decouple spatial geometry from temporal disturbances [48]. Other related projection based maneuvering methods select a path parameter from the current vehicle state which adds adaptive capability to accommodate for speed changes due to external disturbances [49]. The controller therefore estimates a reference progress time by comparing the EKF-synchronized position estimate with nearby positions on the nominal trajectory. This synchronized position is a data-assimilated estimate obtained from the EKF and synchronization layer. Let s_k denote the progress time used to evaluate the nominal reference. At each controller update, a finite set of candidate progress times is constructed over a local search interval,

$$\mathcal{S}_k = \{s_i : s_i \in [s_{k-1},\, t_k + T_{\mathrm{look}}], \quad s_{i+1} - s_i = \Delta s_{\mathrm{grid}}\}$$

Each candidate is scored using the scaled position mismatch between the nominal reference and the synchronized position estimate,

$$J_{\text{prog}}(s_i) = \left\| L_r^{-1} \left(r_N^\star(s_i) - \hat{r}_{N,k}^{s} \right) \right\|_2^2 + \rho_t \left(\frac{s_i - t_k}{T_{\text{look}}} \right)^2$$

Here, L_r is a diagonal position-scaling matrix, T_{look} sets the maximum look-ahead time, and ρ_t discourages progress estimates that move unnecessarily far from the current clock time. The desired progress time is selected as

$$s_k^{\text{des}} = \arg\min_{s_i \in \mathcal{S}_k} J_{\text{prog}}(s_i)$$

The selected progress time is then passed through a rate-limited first-order update:

$$s_k = s_{k-1} + \text{clip}\left(\frac{s_k^{\text{des}} - s_{k-1}}{\tau_s}, \dot{s}_{\min}, \dot{s}_{\max} \right) \Delta t_k$$

This filtering prevents abrupt jumps between neighboring points on the nominal trajectory. The result is a reference timing update that reduces accumulated phase drift without forcing the controller to chase noisy position changes at each time step.

Disturbance-Rejection Wind-Correction Angle (WCA) Feedforward. Feedback-only lateral controllers inherently require a non-zero cross-track error to generate a steady-state correction effort against a persistent crosswind. To eliminate this steady-state tracking bias, kinematic feedforward terms are frequently utilized. For example, Rysdyk [50] demonstrated that augmenting Line-of-Sight (LOS) guidance laws with a feedforward crosswind crab angle significantly improves trajectory-following accuracy for unmanned aircraft. According to the FAA [51], the preferred method of correcting for wind drift is to angle the airplane sufficiently into the wind to cancel the effect of the sideways drift caused by the wind. This angle is commonly referred to as a *crab angle* or *wind correction angle*. To implement this, the EKF wind estimate is decomposed into along-track and cross-track components relative to the nominal runway-aligned heading. For a reference heading $\psi^\star(s_k)$,

$$\hat{w}_{\perp,k} = \begin{bmatrix} -\sin\psi^\star(s_k) & \cos\psi^\star(s_k) \end{bmatrix} \hat{w}_{NE,k}^{\text{ekf}}$$

Let $\hat{w}_{NE,k}^{\text{ekf}}$ denote the horizontal wind estimate extracted from the conditioned EKF state,

$$\hat{w}_{NE,k}^{\text{ekf}} = \begin{bmatrix} \hat{w}_{N,k}^{\text{ekf}} & \hat{w}_{E,k}^{\text{ekf}} \end{bmatrix}^\top$$

The along-track and cross-track wind components are then

$$\hat{w}_{\parallel,k} = \left(\hat{w}_{NE,k}^{\mathrm{ekf}}\right)^{\top} \hat{e}_{\parallel,k}, \qquad \hat{w}_{\perp,k} = \left(\hat{w}_{NE,k}^{\mathrm{ekf}}\right)^{\top} \hat{e}_{\perp,k}$$

The cross-track component is used to compute a bounded heading-bias command,

$$\psi_{\mathrm{ff},k}^{\mathrm{des}} = \mathrm{sat}_{\psi_{\max}}\left[-\tan^{-1}\left(\frac{\hat{w}_{\perp,k}}{\max\left(V_a^{\star}(s_k), V_{\min}\right)}\right)\right]$$

where $V_a^{\star}(s_k)$ is the nominal airspeed at the synchronized progress time, $V_{\min}$ prevents unrealistically large corrections at low speed, and $\psi_{\max}$ limits the maximum allowed heading bias. The desired bias is then rate-limited before being applied and the synchronized heading reference used in the TVLQR tracking error is

$$\psi_{\mathrm{ref},k}^{\mathrm{sync}} = \mathrm{wrap}\left(\psi^{\star}(s_k) + \psi_{\mathrm{ff},k}\right)$$

The synchronized reference state $z_{\mathrm{ref},k}^{\mathrm{sync}}$ is formed by evaluating the nominal augmented reference at the synchronized progress time s_k and replacing the heading component with $\psi_{\mathrm{ref},k}^{\mathrm{sync}}$. All other reference components remain tied to the nominal trajectory unless they are modified later by the terminal policy layer. The resulting command keeps the cached TVLQR structure,

$$u_{c,k} = u_c^{\star}(s_k) - K(s_k)\left(z_k - z_{\mathrm{ref},k}^{\mathrm{sync}}\right)$$

followed by the same command saturation, rate limits, and actuator dynamics used in the baseline tracking simulation. This heading feedforward is not able to remove all lateral displacement from wind gusts. It biases the commanded heading into the estimated crosswind so that the ground track remains closer to the runway-aligned reference under persistent lateral wind. The synchronized digital twin therefore supports tracking by supplying wind-aware reference timing and a bounded heading correction, while the main feedback control still comes from the cached TVLQR gain schedule. Large gusts, actuator lag, and high-frequency wind content can still produce temporary displacement.

6.6 Results

6.6.1 Tracking Support under Wind Disturbances

The terminal metrics in Table 6.2 show that the EKF-supported controller reaches the main-gear threshold in both disturbed cases when terminal policy selection is disabled. Compared with the baseline TVLQR-only terminal behavior reported earlier, the threshold contact locations are closer to the nominal touchdown

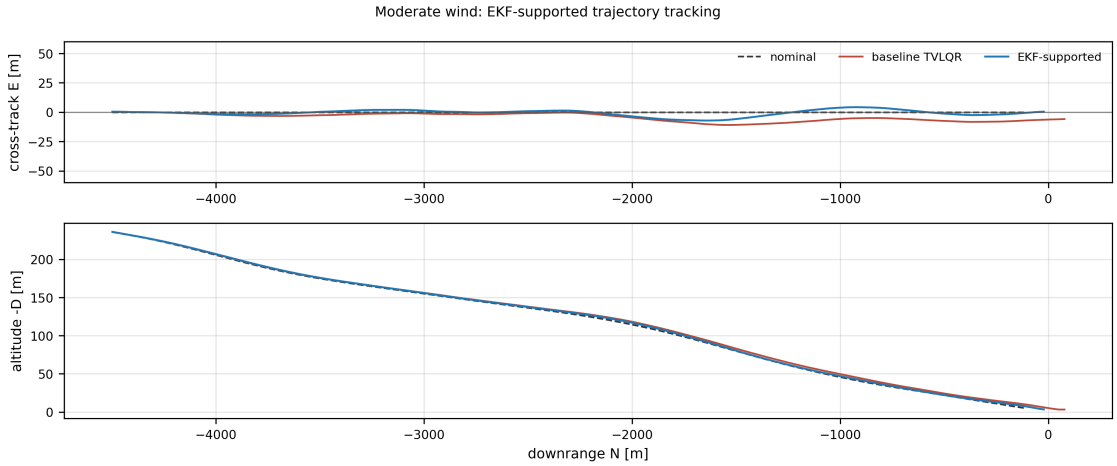
region in downrange position. The moderate and severe cases reach threshold contact at $N_c = -23.02\ \mathrm{m}$ and $N_c = -26.05\ \mathrm{m}$, respectively, rather than landing far past the runway reference point. This suggests that EKF-supported progress synchronization reduces the downrange phase error that appeared in the disturbed TVLQR-only cases. However, the contact rates are still large, especially in the severe case, where $\dot{h}_{\mathrm{mg},c} = -5.12\ \mathrm{m/s}$ and $V_{\mathrm{sink},c} = 4.88\ \mathrm{m/s}$. Therefore, this result should be interpreted as improved tracking support, not as a satisfactory terminal landing policy by itself.

Table 6.2: Terminal landing metrics for TVLQR+EKF tracking with terminal policy selection disabled. First contact is reported using the $h_{\mathrm{mg}} \leq 0.09\ \mathrm{m}$ threshold-contact convention.

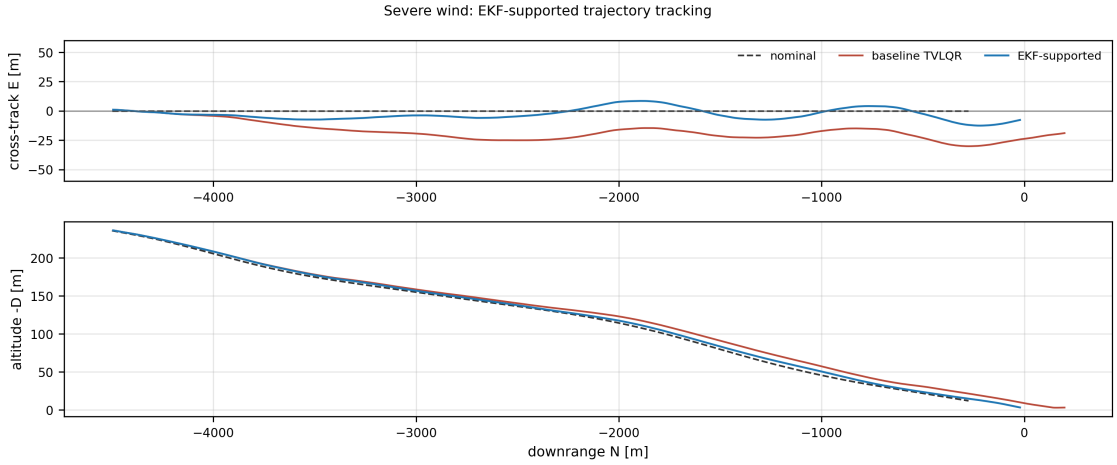
Case	Main gear contact?	t_c (s)	N_c (m)	E_c (m)	$h_{\mathrm{mg},c}$ (m)	$\dot{h}_{\mathrm{mg},c}$ (m/s)	$V_{\mathrm{sink},c}$ (m/s)	θ_c (deg)
Moderate wind	Yes	49.34	−23.02	0.42	0.090	−3.29	3.07	−1.68
Severe wind	Yes	47.45	−26.05	−7.94	0.090	−5.12	4.88	−2.72

Table 6.3: Additional terminal diagnostics for TVLQR+EKF tracking with terminal policy selection disabled.

Case	α_c (deg)	u_c (m/s)	$h_{\mathrm{mg,min}}$ (m)	$t(h_{\mathrm{mg,min}})$ (s)
Moderate wind	0.18	85.69	0.058	49.35
Severe wind	−0.15	86.54	−0.147	47.50



(a) Moderate wind case: Stable tracking with minimal cross-track variance and robust alignment to the nominal glide slope.



(b) Severe wind case (stress test): EKF feedforward actively counteracts cross-track drift, while progress synchronization reduces phase lag along the glide path.

Figure 6.1: EKF-supported trajectory tracking performance compared against baseline TVLQR under two wind disturbance cases. The upper axes show cross-track position E, and the lower axes show altitude $-D$ against downrange position N.

The additional terminal diagnostics in Table 6.3 are consistent with the same interpretation. The forward speeds at contact remain close to the intended approach range, with $u_c = 85.69\ \mathrm{m/s}$ in the moderate case and $u_c = 86.54\ \mathrm{m/s}$ in the severe case. The angle of attack also stays small in both cases, indicating that the EKF-supported tracking changes do not create an obvious terminal angle-of-attack excursion. The minimum main-gear clearance is more concerning in the severe case, where $h_{\mathrm{mg,min}} = -0.147\ \mathrm{m}$, indicating pene-

tration below the threshold-contact band after first contact. This reinforces the need for the later terminal policy layer. The EKF-supported controller helps the aircraft arrive in the right terminal region, but it does not regulate touchdown softness on its own.

Figure 6.1 compares the baseline TVLQR response with the EKF-supported response for the moderate and severe wind cases. The baseline controller uses the same nominal trajectory and cached gain schedule, but it does not receive EKF-supported reference timing or wind-aware heading feedforward. The EKF-supported case uses the same TVLQR gain schedule, while also using the synchronized twin for phase synchronization and the EKF wind estimate $\hat{w}_{N,k}^{\mathrm{ekf}}$ for lateral disturbance rejection feedforward. In the moderate wind case, the EKF-supported trajectory remains close to the nominal cross-track path and avoids the larger negative lateral drift seen in the baseline response. The vertical profile is also similar to the nominal descent for both controllers, but the EKF-supported response reaches the near-runway region with less apparent downrange delay. This behavior is consistent with the terminal results in Table 6.2, where the moderate EKF-supported case reaches the contact threshold slightly before the runway reference point instead of drifting late downrange.

In the severe wind case, the baseline response develops a sustained cross-track offset under the stronger lateral disturbance, while the EKF-supported response remains much closer to the runway centerline, although with visible oscillation about the nominal path, likely due to wind gust. The vertical profile also tracks closer to the nominal descent over much of the approach. These improvements are useful, but they do not remove the terminal landing-quality problem. The severe EKF-supported case still reaches contact with a large downward clearance rate and a nose-down pitch angle, as shown in Table 6.2.

Table 6.4: TVLQR+EKF tracking-error metrics with terminal policy selection disabled.

Case	Pos. RMSE (m)	Max pos. err. (m)	N-RMSE (m)	E-RMSE (m)	D-RMSE (m)	Vel. RMSE (m/s)	θ-RMSE (deg)
Moderate wind	13.77	18.86	13.47	2.77	0.67	0.67	0.39
Severe wind	19.80	28.72	18.87	5.71	1.83	1.02	0.62

The tracking-error metrics in Table 6.4 supports the tracking improvements seen in the trajectory plots. The moderate case has a position RMSE of $13.77\ \mathrm{m}$, while the severe case has a position RMSE of $19.80\ \mathrm{m}$. Most of this remaining error is still in the downrange direction, with N-RMSE values of $13.47\ \mathrm{m}$ and $18.87\ \mathrm{m}$.

6.6.2 *Wind and Pose Estimation Accuracy*

Figure 6.2 shows the pose-filtering trends of the cross-track position. The measurement stream contains visible sensor noise, while the EKF estimate stays close to the true E-position trajectory in both the moderate and severe wind cases. The lower panels show that the EKF estimation error is much smaller than the raw measurement error and remains centered near zero for most of the maneuver. The error however, can be seen to occasionally leave the $\pm 2\sigma$ region. The severe wind case has larger overall cross-track motion, but the filtered position estimate can be seen to continue to stay coincident with the actual position.

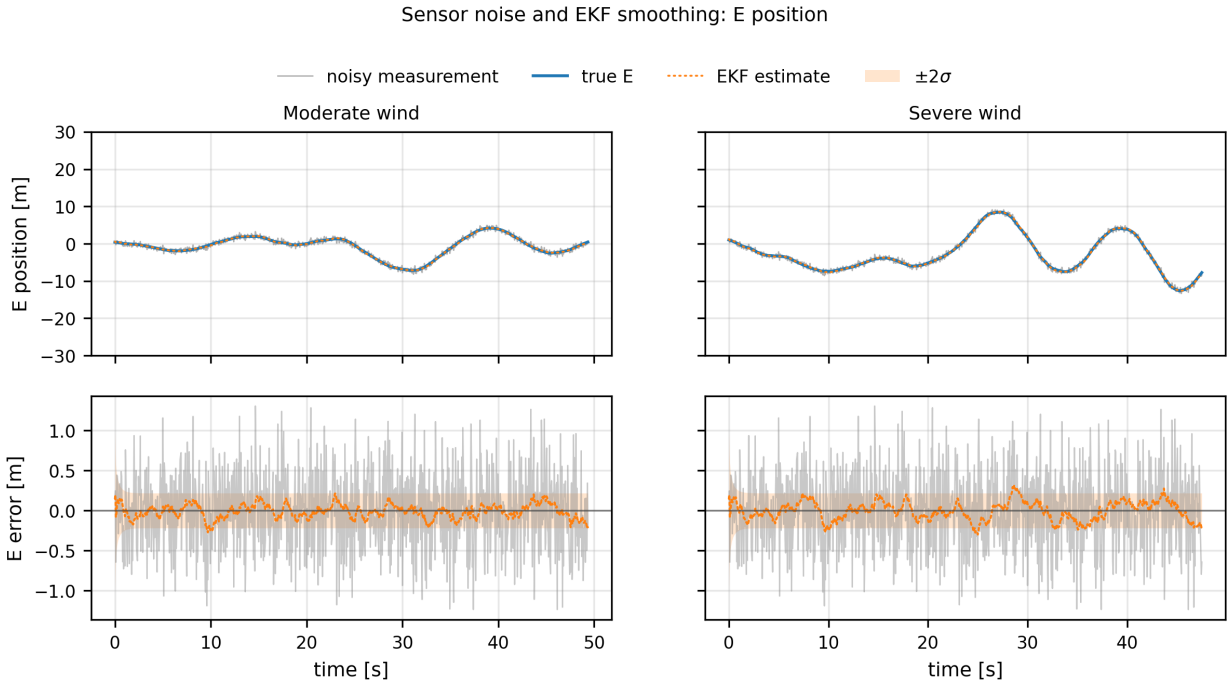
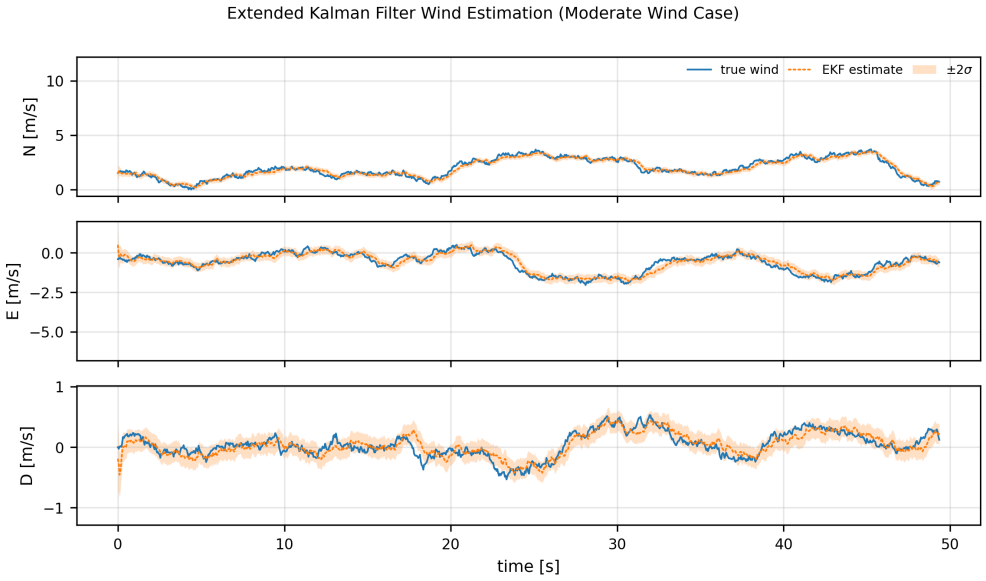
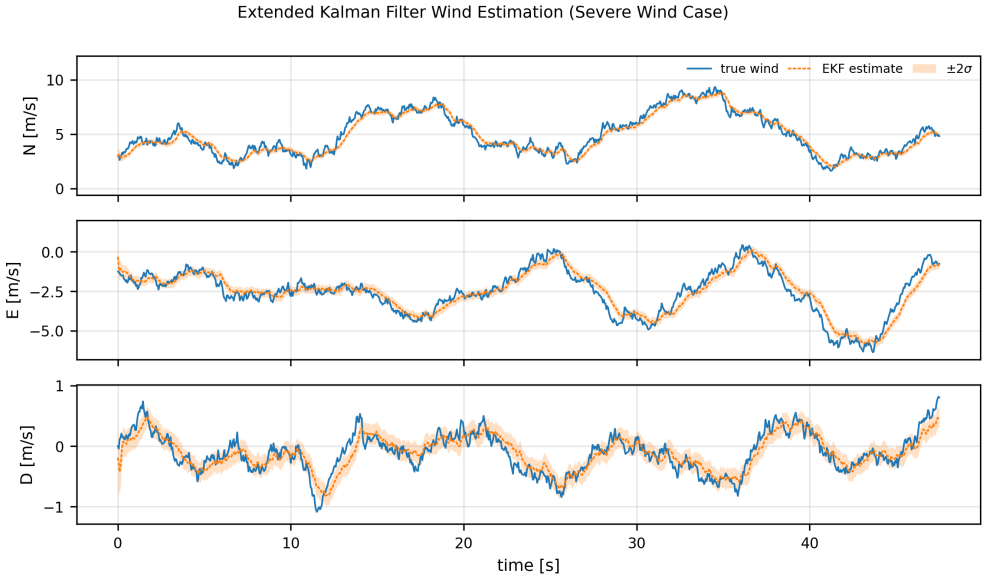


Figure 6.2: EKF pose-smoothing diagnostic for the cross-track position E under moderate and severe wind cases. The top axes compare the true trajectory, noisy measurements, and EKF estimate. The bottom axes show the corresponding estimation error and internal $\pm 2\sigma$ covariance bounds.



(a) Moderate wind case: EKF wind estimation in the local NED frame. The estimate follows the dominant wind trends while smoothing higher-frequency variation.



(b) Severe wind case: EKF wind estimation under stronger wind variation. The estimate remains bounded and tracks the main wind trend, but visible lag appears during sharper gust transitions.

Figure 6.3: Extended Kalman Filter wind velocity estimation compared against the true simulated wind vectors in the local NED frame. The shaded regions denote the internal $\pm 2\sigma$ covariance bounds from $P_{k|k}$.

Figure 6.3 shows that the EKF successfully estimates the wind trends in both disturbed cases. In the moderate wind case, the estimated N, E, and D wind components remain close to the true simulated wind over most of the maneuver. The estimate is visibly smoother than the true wind disturbance, especially in the vertical wind component, but it follows the main low-frequency variation without divergence. This behavior is consistent with the intended role of the wind state as it is keeping the synchronized digital twin from drifting from the nominal trajectory over time, but is is unable to perfectly reconstruct every gust fluctuation. The severe wind case is able to show this behavior more clearly where the estimate lags during sharper gust transitions, especially in the E and D components. This lag is expected because the wind state is modeled as a first-order Gauss–Markov process and the covariance tuning favors a smoothed estimate over direct tracking of noisy or rapidly varying measurements.

The wind estimation accuracy is quantified in Table 6.5 where it can be seen that the wind RMSE increases from $0.311\ \mathrm{m/s}$ in the moderate case to $0.657\ \mathrm{m/s}$ in the severe case, and the final wind-error norm increases from $0.089\ \mathrm{m/s}$ to $0.407\ \mathrm{m/s}$. The pose and attitude estimates remain much cleaner than the noisy measurements in both cases. The moderate pose RMSE is $0.153\ \mathrm{m}$, compared with a measurement RMSE of $0.729\ \mathrm{m}$, while the severe pose RMSE is $0.198\ \mathrm{m}$ against a similar measurement RMSE indicating that the EKF is reducing measurement noise while still tracking the motion needed for synchronization.

The measurement vector has nine channels, so a mean NIS near the state dimension is broadly consistent with reasonable covariance tuning.The innovation diagnostics show that the moderate case has a mean NIS of 9.09, close to the nine-channel measurement dimension, while the severe case increases to 11.57. The 95th percentile also increases from 17.05 to 22.95, showing that the severe case produces larger residuals relative to the assumed covariance. This indicates that the covariance tuning is less consistent under stronger disturbances. The minimum covariance eigenvalue remained positive in both cases, approximately 4.8×10^{-7}, which indicates that the covariance update remained numerically well behaved in these runs.

Table 6.5: EKF estimation diagnostics for the moderate and severe wind cases. Pose and attitude entries are reported as EKF RMSE divided by the corresponding measurement RMSE.

Case	Wind RMSE [m/s]	Final wind err. [m/s]	Pose RMSE / meas. [m]	Att. RMSE / meas. [deg]	NIS mean	NIS p95
Moderate wind	0.311	0.089	0.153/0.729	0.084/0.326	9.09	17.05
Severe wind	0.657	0.407	0.198/0.728	0.124/0.324	11.57	22.95

6.7 Discussion

The EKF-supported tracking results indicate that pose and wind synchronization improves the results of the wind-disturbed tracking problem. Supporting the controller with trajectory phase synchronization and disturbance-rejection feedforward using real-time estimated pose and wind keeps the landing trajectory much more closely aligned with the nominal trajectory. These trends can be seen in Figure 6.1, and are valid for both the moderate wind case as well as the severe-wind stress-test case. Comparing the EKF-supported controller setup with the baseline TVLQR controller detailed in Section 5, the baseline TVLQR showed an N-RMSE of $127.25\ \mathrm{m}$, showing that under a headwind, the cached gain reference becomes inaccurate relative to the actual location of the airplane. Progress synchronization reduced the N-RMSE to $18.87\ \mathrm{m}$. Lateral crosswind caused the baseline tracking controller to exhibit an E-RMSE of about $20\ \mathrm{m}$, but the crab-angle correction mechanism, updated using real-time wind information, reduced this to $5.71\ \mathrm{m}$. In the severe wind case, this led to an 84% position error reduction.

Figure 6.2 shows that the EKF filters the noisy cross-track position measurement while remaining close to the true trajectory in both disturbed cases. Figure 6.3 shows that the EKF follows the dominant wind trends in the local NED frame, especially in the moderate case. In the severe case, the estimate remains bounded and tracks the main wind structure, but it visibly lags during sharper gust transitions. This behavior is consistent with the first-order Gauss–Markov wind model and the covariance tuning. The EKF estimates a nine-state local pose-and-wind vector, but the mismatch component of that state is still only a three-component Gauss–Markov wind model. It does not reconstruct the full spatiotemporal wind field, aerodynamic coefficient error, lift and drag uncertainty, actuator mismatch, ground effect, or runway-contact behavior. If the dominant mismatch comes from wind shear, gust gradients, near-ground aerodynamics, or engine lag, the wind estimate can only absorb the portion of that error that appears as a low-frequency ground-velocity discrepancy.

Limitations to the current extended Kalman filter setup include the tracking controller still using an offline gain schedule designed around the nominal trajectory. EKF synchronization allows for reference timing that considers wind disturbance effects, but it does not redesign the trajectory or recompute the Riccati gains. While this preserves the computational advantage of TVLQR, it also means the controller remains local to the nominal path and its associated timing. If the aircraft moves too far from the reference, or if the progress estimate becomes poor, the linearized tracking model becomes less representative of the actual closed-loop

behavior. Another limitation is that the measurement and truth information come from a simulation environment. Sensor noise is injected to avoid giving the estimator unrealistically clean information, but the plant proxy still provides known truth values for diagnostic comparison. This is useful for development and verification, but it is not equivalent to flight-test validation. Finally, the nominal planner does not model landing-gear contact, runway reaction forces, gear compression, or ground effects. Because the PTR planner leaves out these effects, the final nominal state should not be interpreted as a physically realizable touchdown condition. For the RCAM geometry used in the higher-fidelity execution model, main-gear contact occurs while the aircraft reference point is still several meters above the runway. Therefore, the nominal terminal altitude target of $0.10\ \mathrm{m}$ above the runway is best interpreted as a terminal reference that pulls the airborne trajectory toward the runway region.

For terminal conditions, the EKF-supported controller reaches the main-gear threshold in both tested wind cases, and the threshold contact locations are closer to the nominal touchdown region than in the baseline disturbed TVLQR-only results. This supports the use of progress synchronization and wind-aware heading feedforward as tracking-support terms. However, Table 6.2 also shows that the severe case still reaches contact with a large downward clearance rate and a nose-down pitch angle. The additional diagnostics in Table 6.3 show that the severe case reaches a minimum main-gear clearance of $-0.147\ \mathrm{m}$, which indicates that the terminal behavior remains too aggressive after threshold contact. Thus, the EKF-supported controller helps the aircraft arrive in the right terminal region, but it does not regulate touchdown softness by itself.

The current nominal trajectory's terminal behavior depends strongly on SCP convergence, terminal constraint choices, and tuning. As a result, accurately tracking the current nominal trajectory can still produce an undesirable landing condition. In fact, because the EKF-supported controller follows the nominal trajectory more closely while also rejecting wind perturbations, it can guide the aircraft more directly into the terminal segment without necessarily reducing the sink rate. The sink rate increases from $2.40\ \mathrm{m/s}$ in the no-wind baseline case to $4.88\ \mathrm{m/s}$ in the severe-wind EKF-supported case, which is large enough to be considered a potentially damaging touchdown condition. This shows that estimation and filtering alone, even when they accurately synchronize the digital twin with the plant model, are not enough to produce a smooth landing if the SCP formulation was not able to produce a smooth landing. The continuously updated nonlinear aircraft model must also be used for terminal decision support, so that the controller can respond to poor predicted touchdown behavior rather than simply tracking a nominal trajectory into it.

Chapter 7

DIGITAL-TWIN FORMULATION FOR LANDING DECISION SUPPORT

The previous chapters developed the nonlinear aircraft model in Section 3, generated a nominal landing trajectory using the imperfect digital twin in Section 4, designed an offline TVLQR tracking controller in Section 5, and introduced EKF-based synchronization in Section 6. During execution, the EKF acts as a data-assimilation layer that estimates a reduced local pose-and-wind state from noisy pose and ground-velocity measurements. This estimate provides filtered pose corrections and a wind estimate to the synchronized digital twin, which continues to propagate the nonlinear aircraft dynamics under the current command history. The value of the synchronized digital twin is therefore assessed through its ability to support tracking, forecast terminal landing behavior, and guide policy selection during the final phase of the landing maneuver.

7.1 Digital twin prediction based Terminal Policy Supervisor

The synchronized digital twin is used during the terminal phase of the landing maneuver as a short-horizon consequence prediction model for landing decision support. A trajectory can remain close to the nominal path while still approaching the runway with undesirable terminal geometry. This may include include a high downward clearance rate, nose-down pitching angle, main-gear penetration, excessive thrust, or delayed touchdown. Near the runway, small differences in clearance, pitch attitude, sink rate, and touchdown timing can determine whether the aircraft reaches a reasonable first-contact condition or arrives nose-down, too hard, too late, or still floating above the runway.

A digital twin that is updated with current flight position can be useful in this application as it contains the nonlinear airplane dynamics that can be propagated forward in time. The predicted future landing metrics can be evaluated based on the current state of the aircraft and the decisions that can be taken that leads to more desirable landing outcomes. At each update, the current EKF-synchronized state and wind estimate define the best available estimate of the aircraft and disturbance state. The twin then propagates that state forward under several candidate terminal policies and evaluates the predicted landing desirability of each candidate. The predicted consequences of several terminal policies are compared before selecting the policy

to apply to the controller.

Activation logic. The terminal policy selector is activated only in the final portion of the maneuver. Let t_f denote the terminal time of the nominal landing reference and let t_j denote the jth selector update time. The terminal policy selector starts up to 10 s before the nominal terminal landing time but only if the main gear has not touched the runway yet, and the main gear is within a clearance gate of 0–20 m. The terminal policy selector then refreshes at a rate of 2 Hz (about every 0.5 sec). At each selector update, the synchronized twin propagates each candidate policy (6 total) over a 5.0 s horizon using internal nonlinear propagation at 25 Hz. Each candidate deployment is scored using predicted landing-quality metrics, and the policy with the most favorable score is selected. The selected policy is determined by a touchdown quality scoring function. Policy scoring is based on repeated finite-horizon candidate deployments and ranking. The landing score includes desirable touchdown metrics and penalties for undesirable effects such as hard runway contact, high sink rate, and constraint violations.

Candidate terminal policies. Before the selected policy is applied to the plant, each candidate policy is tested inside the synchronized twin using the same initial time aircraft state and wind estimate. The policies modify the terminal reference, command target, feedback scaling, command limits, or closure behavior to represent different ways the controller could manage the final approach. The synchronized digital twin then predicts the consequence of applying each behavior over the finite prediction horizon. This allows the supervisor to choose among interpretable terminal actions without needing to solve a new nonlinear optimal-control problem online. Within the policy selector, continuing with the nominal trajectory would score the aircraft based on if it can safely continue tracking the existing TVLQR reference. The descent and roundout policies modify the terminal reference behavior to maintain runway closure or reduce excessive sink. The flare policy applies low-altitude pitch and sink-rate shaping. The float-guard and direct-contact policies are more aggressive contact-capture behaviors used when the forecast indicates delayed touchdown, insufficient closure, or persistent wheel clearance.

At each selector update, the synchronized twin evaluates a finite candidate set $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$. In this landing scenario, the candidate set is defined using

$$\Pi = \{\pi_{\mathrm{nom}}, \pi_{\mathrm{closure}}, \pi_{\mathrm{sink}}, \pi_{\mathrm{flare}}, \pi_{\mathrm{float}}, \pi_{\mathrm{contact}}\}$$

Each policy corresponds to an interpretable terminal behavior, their effect on the landing characteristics are summarized below in Table 7.1

Table 7.1: Candidate terminal policies evaluated by the forecast-selected supervisor.

Candidate policy	**Terminal behavior represented in the selector**
Nominal Tracking	Continue tracking the nominal reference.
Runway-Closure Tracking	Maintain descent toward the runway when the prediction indicates that the aircraft may remain too high.
Sink-Rate Reduction	Reduce excessive downward clearance rate while still preserving runway closure.
Flare Shaping	Apply a low-altitude pitch increase and sink-rate decrease.
Float Prevention	Bias the terminal behavior toward contact when the forecast indicates delayed touchdown or insufficient closure.
Contact Capture	Apply stronger touchdown-oriented terminal modifications when the other candidates are predicted to float, touch down late, or miss main-gear contact.

Policy-dependent influence on the TVLQR command. The candidate policies modify the reference, command target, feedback scaling, command limits that enter the original cached TVLQR feedback structure. Recall from Section 6 that the augmented reference $z_{\mathrm{ref},k}$ and commanded target $u_{\mathrm{ref},k}$ denote the nominal scheduled reference state and command at the current progress time, and K_k denotes the stored TVLQR gain. For a candidate policy π, the terminal supervisor constructs a policy-dependent reference and command target,

$$z_{\mathrm{ref},k}^{(\pi)} = z_{\mathrm{ref},k} + \Delta z_{\mathrm{ref},k}^{(\pi)}, \qquad u_{\mathrm{ref},k}^{(\pi)} = u_{\mathrm{ref},k} + \Delta u_{\mathrm{ref},k}^{(\pi)}$$

The tracking error used by the feedback law is therefore also policy dependent,

$$e_k^{(\pi)} = z_k - z_{\mathrm{ref},k}^{(\pi)}$$

The resulting command correction can be represented as

$$\Delta u_k^{(\pi)} = -\alpha_\pi K_k e_k^{(\pi)} + \Delta u_{\mathrm{overlay},k}^{(\pi)}$$

where α_π is a policy-dependent feedback scale and $\Delta u^{(\pi)}_{\mathrm{overlay},k}$ represents an additive terminal command overlay, such as thrust reduction. The command applied during the deployment is then clipped, rate-limited, and saturated

$$u^{(\pi)}_{c,k} = \mathrm{sat}\left[\mathrm{rate}\left(u^{(\pi)}_{\mathrm{ref},k} + \mathrm{clip}\left(\Delta u^{(\pi)}_{k}, -\Delta u^{(\pi)}_{\max}, \Delta u^{(\pi)}_{\max}\right)\right)\right]$$

to represent actuator magnitude and rate limits. Each policy is tested by propagating the synchronized nonlinear twin forward and scoring the predicted landing outcome before the policy is allowed to influence the plant.

Nominal Tracking:

This policy is the baseline candidate and applies no additional terminal shaping. The reference state $z_{\mathrm{ref},k}$, nominal command target $u_{\mathrm{ref},k}$, feedback scale, command increment limits, and rate limits remain those of the nominal synchronized TVLQR controller.

Runway-Closure Tracking:

This corresponds to changing $z^{(\pi)}_{\mathrm{ref},k}$ through terminal range or descent-oriented reference shaping and using a reduced α_π so that the aircraft continues closing toward the runway without relying on an overly aggressive feedback correction.

Sink-Rate Reduction:

It changes $z^{(\pi)}_{\mathrm{ref},k}$ to manage clearance-rate behavior while preserving runway closure, and it modifies $u^{(\pi)}_{\mathrm{ref},k}$ through pitch, elevator, or thrust shaping when the predicted sink rate is too large. Its feedback authority is also reduced relative to the nominal TVLQR behavior so that the terminal response is smoother.

Flare Shaping:

The Flare shaping policy modifies the low-altitude reference state. It changes $z^{(\pi)}_{\mathrm{ref},k}$ by imposing flare-oriented pitch and sink-rate behavior, including pitch-floor logic that discourages nose-down contact. It also uses tighter command limits and reduced feedback inputs so that the final correction remains smooth near the runway. Physically, this policy is intended to improve the contact attitude and reduce excessive descent near touchdown and used when the aircraft is close enough to the runway for flare behavior to matter.

Float Prevention:

The Float Prevention policy modifies the reference, command target, and additive terminal command overlay when the synchronized twin predicts insufficient runway closure or delayed contact. It biases $z_{\mathrm{ref},k}^{(\pi)}$ toward a lower clearance target, adjusts $u_{\mathrm{ref},k}^{(\pi)}$ through closure-oriented command shaping, and introduces $\Delta u_{\mathrm{overlay},k}^{(\pi)}$ contributions such as thrust reduction or an elevator command bias that encourages continued runway closure. Physically, this policy is meant to prevent the aircraft from remaining above the runway when the forecast indicates excessive float, insufficient descent closure, or delayed main-gear contact.

Contact Capture:

The Contact Capture policy is the strongest modification of the baseline TVLQR structure. It modifies $z_{\mathrm{ref},k}^{(\pi)}$ through low-clearance or contact-oriented terminal targets, modifies $u_{\mathrm{ref},k}^{(\pi)}$ through stronger landing-oriented command shaping, and specifies terminal command increments $\Delta u_{\mathrm{overlay},k}^{(\pi)}$ to reduce thrust and bias the elevator command toward a more contact-oriented terminal response. Physically, this policy is used when the other candidates are predicted to float, touch down late, or fail to produce main-gear contact. Because this candidate can produce the strongest terminal command modification, it is penalized in the selector cost and is selected only when its predicted terminal outcome is better than the softer alternatives.

These policies can be interpreted as increasingly stronger modifications to the same underlying TVLQR tracking architecture. Each candidate changes how the terminal reference and command target are shaped before the TVLQR feedback correction $-K_k e_k$ is computed. The nominal candidate leaves the reference state $z_k^\star$ and nominal command target $u_{c,k}^\star$ unchanged. Runway-Closure Tracking, Sink-Rate Reduction, and Flare Shaping modify the terminal reference and sink-rate behavior. Float Prevention and Contact Capture add stronger landing-oriented corrections when the predicted future trajectory indicates that the aircraft may remain above the runway or reach contact too late. This structure keeps the online decision problem computationally light. The expensive trajectory optimization and gain-schedule construction are done offline, while the online supervisor only performs short nonlinear deployments of a finite set of candidate policies. The selected policy is therefore the candidate whose predicted terminal outcome gives the best tradeoff among clearance, clearance rate, pitch attitude, touchdown location, contact timing, constraint satisfaction, and command smoothness.

Candidate policy deployment model. Let $\hat{z}_j$ denote the synchronized augmented twin state at selector update time t_j, where z includes the aircraft state and actuator states used by the tracking simulation. Let $\hat{w}_j$

denote the EKF wind estimate in the local NED frame. For each candidate policy $\pi \in \Pi$, the synchronized twin is initialized from the same issue-time state,

$$\hat{z}_{j,0}^{(\pi)} = \hat{z}_j, \qquad \hat{w}_{j,0}^{(\pi)} = \hat{w}_j$$

The candidate decision is then deployed inside the synchronized twin over the selector horizon T_h. This deployment is a forecast calculation, not the command applied to the plant. The deployed candidate is propagated with the nonlinear synchronized-twin model,

$$\hat{z}_{j,\ell+1}^{(\pi)} = F_{\Delta t}\left(\hat{z}_{j,\ell}^{(\pi)}, u_{j,\ell}^{(\pi)}, \hat{w}_{j,\ell}; \theta_t\right), \qquad \ell = 0, \ldots, N_h - 1$$

where $F_{\Delta t}$ denotes numerical integration of the synchronized nonlinear twin over one deployment step, θ_t denotes the digital-twin model parameters, and N_h is the number of deployment steps. The wind estimate is propagated open-loop using the same first-order Gauss–Markov structure used by the EKF wind model:

$$\hat{w}_{j,\ell} = \Phi_w(\tau_\ell)\hat{w}_j, \qquad \Phi_w(\tau_\ell) = \mathrm{diag}\left(e^{-\tau_\ell/\tau_N}, e^{-\tau_\ell/\tau_E}, e^{-\tau_\ell/\tau_D}\right)$$

where $\tau_\ell = \ell\Delta t$ and τ_N, τ_E, τ_D are the wind correlation time constants. This allows the policy selector to use the current EKF wind estimate while avoiding a separate stochastic forecast inside each candidate deployment. No measurement update is applied during the candidate deployment, so the wind is treated as an open-loop forecast initialized from the most recent EKF estimate.

At each deployment step, the candidate policy modifies the scheduled reference and feedforward command used by the tracking controller. Let $z_{\mathrm{ref}}(s)$ and $u_{\mathrm{ref}}(s)$ denote the scheduled TVLQR reference state and command at progress time s, and let $K(s)$ denote the stored TVLQR gain. The candidate-specific shaping map G_π produces the policy-dependent reference and command target described above:

$$\left(z_{\mathrm{ref},j,\ell}^{(\pi)}, u_{\mathrm{ref},j,\ell}^{(\pi)}\right) = G_\pi\left(z_{\mathrm{ref}}(s_{j,\ell}), u_{\mathrm{ref}}(s_{j,\ell}), \hat{z}_{j,\ell}^{(\pi)}, h_{\mathrm{mg},j,\ell}^{(\pi)}, \dot{h}_{\mathrm{mg},j,\ell}^{(\pi)}\right) \tag{7.1}$$

where G_π represents the candidate-specific terminal reference shaping. The candidate tracking error is

$$e_{j,\ell}^{(\pi)} = \hat{z}_{j,\ell}^{(\pi)} - z_{\mathrm{ref},j,\ell}^{(\pi)}$$

Angle components of the tracking error are wrapped before feedback is applied, consistent with the TVLQR tracking controller. The clipped feedback increment is

$$\Delta u^{(\pi)}_{\mathrm{fb},j,\ell} = \mathrm{clip}\left(-\alpha_\pi K(s_{j,\ell}) e^{(\pi)}_{j,\ell}, -\Delta u^{(\pi)}_{\max}, \Delta u^{(\pi)}_{\max}\right)$$

The command target before rate limiting is then

$$\tilde{u}^{(\pi)}_{j,\ell} = u^{(\pi)}_{\mathrm{ref},j,\ell} + \Delta u^{(\pi)}_{\mathrm{fb},j,\ell} + \Delta u^{(\pi)}_{\mathrm{overlay},j,\ell}$$

where $\Delta u^{(\pi)}_{\mathrm{overlay},j,\ell}$ is zero for candidates without a command overlay. This overlay is used by the contact-capture candidate to modify the command target after the clipped feedback increment has been computed. The final command used inside the candidate deployment is rate-limited and saturated:

$$u^{(\pi)}_{j,\ell} = \mathrm{sat}\left[\mathrm{rate}\left(\tilde{u}^{(\pi)}_{j,\ell}, u^{(\pi)}_{j,\ell-1}, \dot{u}^{(\pi)}_{\max}, \Delta t\right)\right]$$

Thus, each candidate deployment uses the same nonlinear aircraft model, EKF-synchronized initial condition, gain reference schedule, and actuator and command-limiting structure. The only difference between each terminal policy is their policy-dependent reference shaping, command shaping, feedback scaling, command limits, and any additive command overlay.

Main-gear clearance and contact detection. After each candidate policy deployment, the policy selector extracts landing-quality metrics from the predicted trajectory. For each predicted state, the main-gear geometry and aircraft attitude are used to compute the height of the lower main gear relative to the runway. Hence why the aircraft reference point can remain above the runway even when the main gear reaches the contact threshold. Let $h^{(\pi)}_{\mathrm{mg},j,\ell}$ denote the predicted main-gear clearance for candidate policy π at deployment step ℓ. A predicted contact event occurs at the first index

$$\ell^{(\pi)}_c = \min\left\{\ell : h^{(\pi)}_{\mathrm{mg},j,\ell} \le h_{\mathrm{td}} \ \text{ or } \ \text{main gear contact}\right\}$$

where h_{td} is a small positive near-contact threshold on the computed main-gear clearance. The minimum predicted clearance, final predicted clearance, clearance rate, pitch angle, speed, and predicted touchdown location are also recorded. If h_{mg} becomes negative, the negative clearance is interpreted as geometric gear penetration or compression for scoring and safety-flag logic. If no contact event is found over the deployment horizon, contact-index quantities are evaluated at the final deployment step and the no-contact flag is set allowing each candidate policy to be scored while still penalizing policies that remain above the runway or fail to produce main-gear contact.

Terminal policy scoring. Each candidate deployment is reduced to a small set of predicted terminal quantities that describe whether the aircraft is approaching an acceptable first-contact condition. The selector evaluates landing desirability using predicted main-gear clearance, clearance rate, pitch attitude, speed, touchdown location, and discrete risk flags. This enables distinction between policies that produce similar tracking error while also producing different touchdown behavior. For a candidate policy π, let the synchronized-twin deployment produce the predicted terminal feature vector

$$\hat{y}^{(\pi)} = \begin{bmatrix} \hat{h}_c^{(\pi)} & \hat{\dot{h}}_c^{(\pi)} & \hat{\theta}_c^{(\pi)} & \hat{V}_c^{(\pi)} & \hat{N}_c^{(\pi)} & \hat{h}_{\min}^{(\pi)} \end{bmatrix}^T$$

where $\hat{h}_c^{(\pi)}$ is the predicted main-gear clearance at first contact, $\hat{\dot{h}}_c^{(\pi)}$ is the corresponding clearance rate, $\hat{\theta}_c^{(\pi)}$ is the pitch angle, $\hat{V}_c^{(\pi)}$ is the speed, and $\hat{N}_c^{(\pi)}$ is the predicted downrange contact location. If no contact event is found during the deployment, these contact-index quantities are evaluated at the final deployment step and the no-contact flag is set. The quantity $\hat{h}_{\min}^{(\pi)}$ is the minimum predicted main-gear clearance over the deployment. A compact representation of the scoring function is

$$\begin{aligned} J^{(\pi)} = {} & w_h e_h^2 + w_{\dot{h}} e_{\dot{h}}^2 + w_{\text{hard}} e_{\text{hard}}^2 + w_\theta \left(\frac{e_\theta}{3}\right)^2 + w_{\text{nose}} e_{\text{nose}}^2 \\ & + w_{\text{pen}} e_{\text{pen}}^2 + w_V e_V^2 + w_N e_N^2 + w_{\text{late}} e_{\text{late}}^2 + w_u J_u \\ & + w_{\text{cv}} e_{\text{cv}}^2 + w_{\text{float}} b_{\text{float}}^{(\pi)} + w_{\text{nc}} b_{\text{nc}}^{(\pi)} + J_{\text{bias}}^{(\pi)} \end{aligned} \tag{7.2}$$

Here e_h is the clearance error relative to the target contact clearance. If contact is not predicted, this error is clipped so that only remaining clearance above the target is penalized. The clearance-rate error $e_{\dot{h}}$ is measured relative to the desired negative closure rate. The hard-landing term e_{hard} penalizes predicted descent rates above the soft and hard sink-rate thresholds, while e_{nose} penalizes contact pitch below the selected nose-down limits. The penetration term is

$$e_{\text{pen}} = \frac{\left[-\hat{h}_{\min}^{(\pi)}\right]_+}{s_{\text{pen}}}$$

where $[x]_+ = \max(x, 0)$ and s_{pen} is the clearance-penetration scale. The binary terms $b_{\text{float}}^{(\pi)}$ and $b_{\text{nc}}^{(\pi)}$ penalize predicted float and near-terminal no-contact cases. The term J_u penalizes normalized command variation during the deployment, e_{cv} is the maximum predicted constraint violation, and $J_{\text{bias}}^{(\pi)}$ represents fixed implementation-level bias penalties, such as the added penalty on the direct contact-capture candidate.

The selected policy is chosen from the same deployment table used to compute Eq. (7.2). Since the score contains both continuous landing-quality penalties and discrete event penalties, the lowest score is

preferred among candidates that pass the acceptance checks. However, the selector is framed as a ranking rule rather than an unconstrained minimization. A candidate is treated as unacceptable when it predicts an unsafe terminal event such as excessive float, hard contact, nose-down contact, or clearance penetration. These conditions define an acceptable candidate set:

$$\Pi_{\text{acceptable},j} = \{\pi \in \Pi : \text{valid}_j^{(\pi)} = 1,\ b_{\text{float},j}^{(\pi)} = 0,\ b_{\text{hard},j}^{(\pi)} = 0,\ b_{\text{nose},j}^{(\pi)} = 0,\ b_{\text{pen},j}^{(\pi)} = 0\} \tag{7.3}$$

When at least one acceptable candidate is available, the selected terminal policy is the lowest cost candidate in the set:

$$\pi_j^\star = \arg \min_{\pi \in \Pi_{\text{acceptable},j}} J_j^{(\pi)} \tag{7.4}$$

If $\Pi_{\text{acceptable},j}$ is empty, then the terminal policy selector chooses the lowest-cost valid candidate. The selector also includes dwell-time and minimum-improvement logic to prevent excessive switching between similar candidates. The pseudo code shown in Algorithm 4 summarizes the terminal policy selector process.

Algorithm 4: Forecast-selected terminal policy supervisor

Input: Synchronized twin state $\hat{z}_j$, EKF wind estimate $\hat{w}_j$, TVLQR schedule, candidate set Π
Output: Selected terminal policy $\pi_j^\star$ and deployment diagnostics
if *selector gates are inactive or main-gear contact has occurred* **then**
  Continue synchronized TVLQR tracking and return the current terminal mode
end
`// 1. Candidate deployment`
foreach $\pi \in \Pi$ **do**
  Initialize candidate deployment from $(\hat{z}_j, \hat{w}_j)$
  Propagate the synchronized twin over T_h using policy-dependent shaping and cached TVLQR feedback
  Extract predicted contact metrics, minimum clearance, and risk flags
  Compute candidate score $J_j^{(\pi)}$ using Eq. (7.2)
end
`// 2. Policy selection`
Construct $\Pi_{\text{acceptable},j}$ using the terminal safety checks
if $\Pi_{\text{acceptable},j} \neq \emptyset$ **then**
  Select the lowest-cost acceptable candidate using Eq. (7.4)
else
  Select the lowest-cost valid candidate
end
Apply dwell-time and minimum-improvement guards
Map $\pi_j^\star$ to the corresponding terminal controller behavior
Store candidate predictions, scores, risk flags, and selected policy

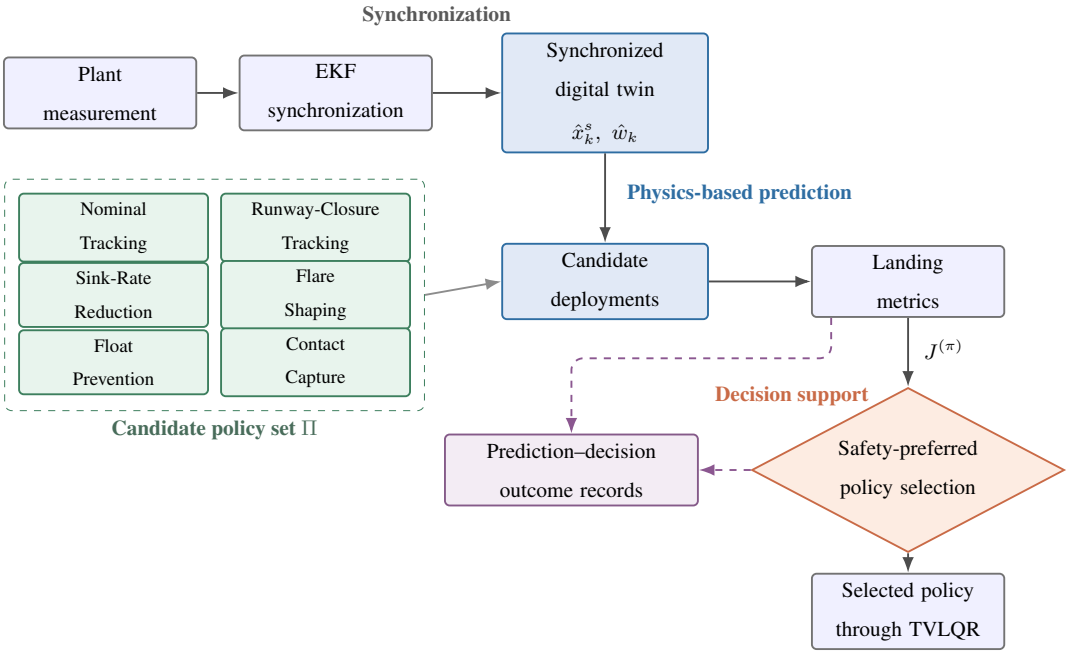


Figure 7.1: The EKF-synchronized DT initializes short-horizon deployments from the current estimated aircraft and wind state. Candidate terminal policies are propagated through the DT, scored using predicted landing metrics, screened by safety-preference logic, and applied to the TVLQR-based terminal controller. The candidate predictions are stored for prediction–decision–outcome analysis.

7.2 Residual Mismatch Diagnostics

The short-horizon forecast comparison evaluates whether the synchronized digital twin improves prediction accuracy over deployment horizons. It is then important to also determine whether synchronization also reduces the local mismatch between the plant and the digital twin along the tracked landing trajectory. The tracking controller is built around TVLQR gains constructed along the nominal landing trajectory, and this structure provides a local linear model for analyzing deviations about the reference. In robust trajectory-tracking work, LTV feedback laws and local uncertainty descriptions are often used because they allow deviation propagation to be bounded using quadratic or set-based arguments [52]. Existing uncertainty descriptions may be model-based, using bounds such as Lipschitz constants, sector conditions, or incremental quadratic constraints [53, 54, 55, 56], or data-driven, where uncertainty is inferred from measured state and input trajectories rather than imposed from a global analytic model [57, 58]. Here, a local-deviation diagnostic is used to quantify the amount of residual mismatch that remains after digital-twin synchronization.

Let ζ_k^{plant} denote the sampled plant state used for TVLQR closed-loop execution, and let u_k denote the corresponding commanded control input on the controller grid. The nominal reference trajectory generated by the digital twin and interpolated onto this grid is denoted by $\{\zeta_k^\star, u_k^\star\}_{k=0}^N$. The same diagnostic is applied to three reference histories: the nominal trajectory, the unsynchronized baseline twin, and the EKF-synchronized twin. For a given reference history, let ζ_k^{ref} and u_k^{ref} denote the state and input used for comparison. The state and input deviations are defined as

$$\delta\zeta_k = \zeta_k^{\text{plant}} - \zeta_k^{\text{ref}}, \qquad \delta u_k = u_k - u_k^{\text{ref}} \tag{7.5}$$

For the nominal reference, $\zeta_k^{\text{ref}} = \zeta_k^\star$ and $u_k^{\text{ref}} = u_k^\star$. For the baseline and synchronized twin references, ζ_k^{ref} is the corresponding twin state history and the executed command is used as the reference input, so $u_k^{\text{ref}} = u_k$ and $\delta u_k = 0$. For angular components, the error is wrapped onto the interval $(-\pi, \pi]$ so that attitude errors remain locally meaningful along the trajectory. The local residual is evaluated using the time-varying sampled deviation model on the controller grid,

$$\delta\zeta_{k+1} \approx A_k\delta\zeta_k + B_k\delta u_k \tag{7.6}$$

where A_k and B_k are the sampled local state and input Jacobians associated with the TVLQR linearization schedule. The residual is then estimated directly from the recorded trajectory as

$$r_k \triangleq \delta\zeta_{k+1} - A_k\delta\zeta_k - B_k\delta u_k \tag{7.7}$$

This residual measures the part of the recorded deviation update that is not explained by the local linear model at that controller sample. Using the time-varying matrices A_k and B_k avoids the extra segment-length artifact introduced by freezing one linearization over a longer trajectory segment. The residual histories are scaled using state, input, and residual normalization factors so that large translational coordinates do not dominate smaller angular or actuator components purely because of unit choice. The reported diagnostics include scaled deviation RMS, maximum scaled deviation, scaled residual RMS, and maximum scaled residual. These quantities are interpreted as empirical summaries of local mismatch severity and are used to assess whether EKF synchronization makes the digital twin locally more consistent with the plant over the pre-selector airborne portion of the landing maneuver.

Chapter 8

DIGITAL-TWIN EVALUATION AND RESULTS

8.1 VVUQ based Digital-Twin Credibility Assessment

The National Academies of Science emphasize Verification, Validation, and Uncertainty Quantification (VVUQ) as essential for the responsible development, implementation, monitoring, and sustainability of digital twins [59]. The evaluation of the synchronized digital twin can therefore be framed as a VVUQ-based credibility assessment for the intended landing decision-support application. It should be noted that in this thesis, VVUQ is only used to organize the evidence that the implemented digital-twin architecture is computationally consistent, accurate enough for the plant-proxy prediction tasks considered, and accompanied by empirical diagnostics of remaining plant–twin mismatch.

Verification Verification relates to ensuring that the algorithms and code used for the digital twin correctly implement the equations and computational procedures of the mathematical model. In this thesis, the airplane dynamics model was imported from PSim-RCAM [24], which follows the parameters and specifications set by GARTEUR [23] with the assumptions and limitations described in Section 3. The supporting algorithms for trajectory tracking, EKF synchronization, short-horizon prediction, terminal policy selection, and data logging were checked through implementation tests, no-wind verification cases, and controlled wind-induced mismatch cases. In the baseline no-wind cases, the plant and digital twin remain nearly coincident with the nominal trajectory, which verifies that the simulation does not introduce large discrepancies when the imposed disturbance is removed. When wind and initial perturbations are introduced, the observed deviations are therefore primarily associated with the intended controlled mismatch between the plant proxy and the digital twin. These checks provide verification of the algorithms used, but they do not amount to any sort of formal code verification or numerical-method verification.

Validation Validation measures the degree to which the digital twin accurately represents the system of interest with regards to it's intended use. In this thesis, the system being represented is not a physical aircraft

flight test, but a high-fidelity plant proxy based on the Research Civil Aircraft Model. The synchronized twin is validated relative to this plant-proxy use case by comparing forecast error growth, landing-relevant terminal quantities, and selector-on/selector-off ablation results. These comparisons evaluate whether the synchronized twin is accurate enough to support short-horizon prediction and terminal policy selection during the simulated landing maneuver. Since no physical flight-test data was used, validation is limited to only the simulation environment. The RCAM model itself is simplified compared to an actual airplane because it assumes rigid-body dynamics, coefficient-based aerodynamics, simplified propulsion behavior, idealized sensing, and simplified actuator and environmental effects. Therefore, the results do not validate the digital twin as a complete representation of a real aircraft and only its usefulness for the controlled plant-proxy decision-support problem considered in this thesis.

Uncertainty Quantification Uncertainty quantification is treated as an empirical residual diagnostic for the synchronized digital twin. Segment-wise deviation dynamics are used to quantify how much local plant–twin mismatch remains after EKF synchronization. These residual summaries provide analysis of whether synchronization reduces local mismatch in the analyzed airborne portions of the trajectory. The UQ results should be interpreted as diagnostic measures of synchronization rather than formal probabilistic uncertainty bounds, robustness certificates, or guaranteed reachable sets for the landing maneuver. The residual analysis is local to the tested trajectories, scaling choices, and retained airborne segments. Touchdown quality is therefore evaluated separately using the short-horizon forecast metrics and terminal policy-selection ablations, while the residual mismatch diagnostics are used to assess whether EKF synchronization produces locally smaller plant–twin mismatch.

8.2 Short-Horizon Forecast Accuracy

The short-horizon forecast study evaluates whether the synchronized digital twin provides useful predictive information beyond an instantaneous EKF-synchronized state estimate. At each forecast issue time, each prediction method is initialized from the same current condition and propagated forward over horizons from 0.5 s to 8.0 s. The predicted state at $t + H$ is then compared with the logged plant state at the same future time. Figure 8.1 shows the resulting forecast error growth for 3D position, main-gear clearance, pitch, and sink rate. The results are simulated from the moderate wind case, using 10 independent runs. The base NED wind is $[1.5,\ -0.4,\ 0.0]$ m/s with variation in gust and turbulence across runs. The average wind magnitude

ranged from about 2.00 to $2.59\ \mathrm{m/s}$, while the peak wind magnitude ranged from about 3.84 to $5.23\ \mathrm{m/s}$.

The EKF-only kinematic prediction is a strong short-horizon baseline, especially for pitch, because it extrapolates from the current pose and wind estimate without introducing additional closed-loop model response. However, the synchronized closed-loop digital twin provides the strongest landing-relevant prediction behavior. It is propagated forward using the nonlinear digital-twin model, the EKF wind estimate, and the stored TVLQR gain schedule. In Figure 8.1, this synchronized closed-loop forecast maintains lower main-gear clearance error over the full plotted horizon range and shows slower sink-rate error growth than the baselines. These quantities are directly tied to terminal landing quality because they describe whether the aircraft is approaching the runway with acceptable wheel clearance, vertical closure rate, and touchdown geometry.

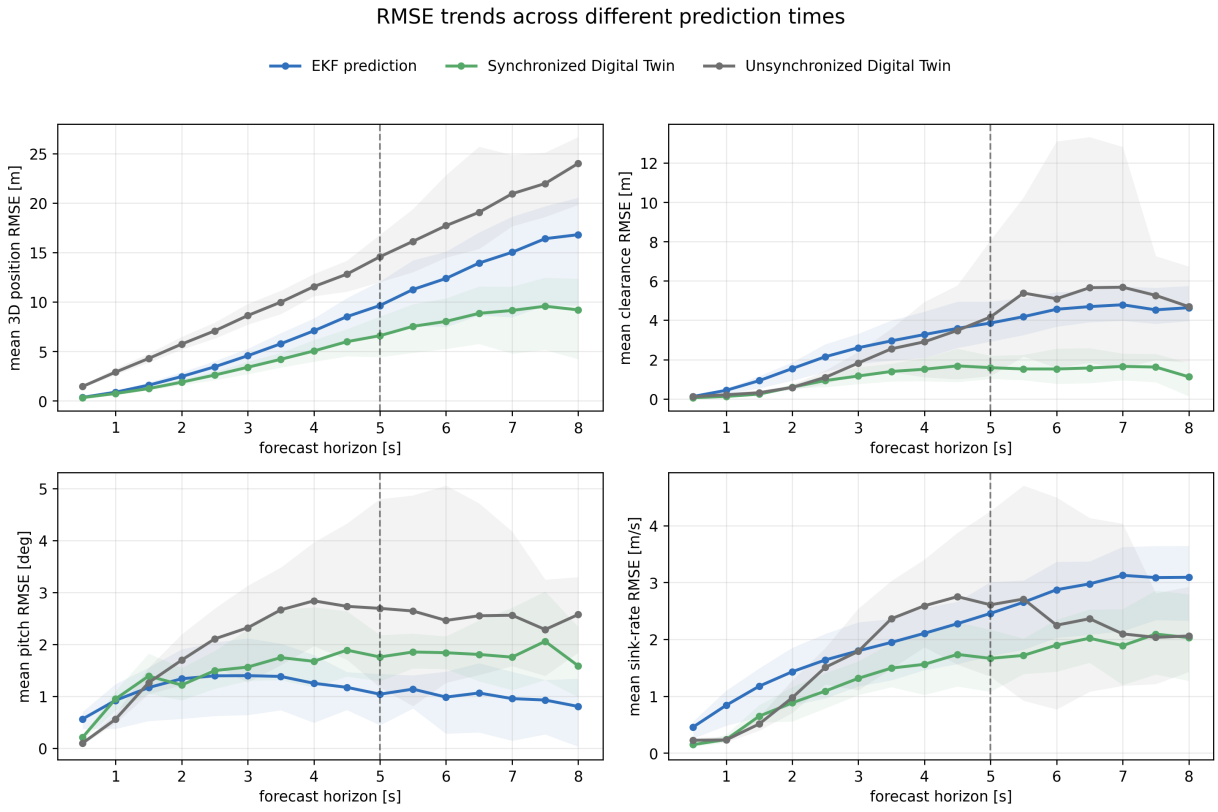


Figure 8.1: Forecast error growth with prediction horizon for the common plant-initial-condition evaluation. The synchronized closed-loop digital twin maintains low landing-relevant error growth, especially for main-gear clearance and sink rate, while also remaining accurate in 3D position prediction over longer horizons.

Table 8.1: Forecast RMSE at the 5.0 s selector horizon for the moderate-realism campaign. Lowest RMSE in each row is shown in bold.

Metric at 5.0 s	EKF prediction	Synchronized DT	Unsynchronized DT
3D position RMSE [m]	9.63	**6.58**	14.57
Main-gear clearance RMSE [m]	3.86	**1.59**	4.17
Sink-rate RMSE [m/s]	2.46	**1.66**	2.61
Pitch RMSE [deg]	**1.04**	1.76	2.69

Figure 8.1 summarizes how forecast error grows as the prediction horizon increases from 0.5 s to 8.0 s. The plotted lines show the campaign-mean RMSE at each horizon, while the shaded regions show the 10th and 90th percentile of the RMSE at a given horizon. A narrow band indicates that the predictor behaves similarly across the randomized wind and initial-condition cases, while a wider band indicates stronger run-to-run sensitivity. Table 8.1 gives the RMSE values at the 5.0 s selector horizon. The EKF-only predictor performs well for pitch because it is a local kinematic extrapolation from the current estimated state. However, it does not propagate the nonlinear aircraft dynamics, actuator response, command limits, or future closed-loop TVLQR corrections. The synchronized closed-loop digital twin includes these effects, which gives it better prediction accuracy for the landing quantities used by the terminal selector. At 5.0 s, it has lower RMSE than both baselines for 3D position, main-gear clearance, and sink rate. The pitch result is the main exception, where the EKF-only prediction has the lowest RMSE. For terminal decision support, the clearance and sink-rate trends are important because they determine whether the aircraft is closing toward the runway with acceptable wheel clearance and vertical closure behavior. The synchronized closed-loop digital twin has the strongest behavior in the main-gear clearance subplot because it predicts how the aircraft continues closing toward the runway under the same TVLQR-based control structure used during execution.

The 5 s terminal forecast horizon used by the terminal policy selector was arbitrarily chosen to compromise between better short horizon accuracy, and more time for landing maneuver to be corrected. A shorter horizon would generally reduce RMSE because there is less time for wind mismatch, actuator lag, nonlinear model error, and controller prediction error to accumulate. However, a very short forecast provides limited time for the terminal policy selector to influence the maneuver before touchdown. The selected policy acts through a rate-limited TVLQR command structure and actuator dynamics, so the terminal trajectory cannot be changed instantaneously. It can be seen in Fig. 8.1 that the synchronized closed-loop digital twin

maintains low RMSE at and beyond the 5 s horizon. The longer prediction horizons should be interpreted with more caution because the number of valid forecast samples decreases as H increases. A forecast at horizon H can only be scored when the logged plant trajectory contains truth data at the future time $t + H$. Therefore, late-terminal issue times remain available for short horizons but not for longer horizons. This can be seen in the wider shaded band as the forecast horizon time increases, suggesting that the 7–8 s values should not be interpreted with the same confidence as the shorter and mid-horizon results.

8.3 Pre-Selector Residual Mismatch Results

The residual-mismatch diagnostics are evaluated on the controller grid with sample period $\Delta t = 0.05$ s. The analysis window is restricted to the pre-selector airborne portion of each run, ending at $t = 42.0$ s. This gives a comparison across the no-wind, moderate-wind, and stress-wind cases before the terminal policy selector begins modifying the landing behavior. Table 8.2 compares the plant trajectory against three reference histories: the nominal trajectory, the unsynchronized baseline twin, and the EKF-synchronized twin.

Table 8.2: Pre-selector reference-alignment and time-varying residual diagnostics. Lower values indicate smaller scaled deviation or smaller empirical residual. Within each wind case, the best value in each metric column is shown in bold.

Wind scenario	**Reference case**	**Deviation RMS** *(scaled)*	**Max deviation** *(scaled)*	**Residual RMS** *(scaled)*	**Max Residual** *(scaled)*
No Wind Case	nominal	1.699×10^{-2}	3.684×10^{-2}	2.230×10^{-3}	1.409×10^{-2}
	baseline_twin	$\mathbf{0.000 \times 10^{0}}$	$\mathbf{0.000 \times 10^{0}}$	$\mathbf{0.000 \times 10^{0}}$	$\mathbf{0.000 \times 10^{0}}$
	synchronized_twin	3.470×10^{-3}	7.316×10^{-3}	2.384×10^{-4}	6.129×10^{-4}
Moderate Wind Case	nominal	1.026×10^{-1}	1.592×10^{-1}	4.724×10^{-3}	3.704×10^{-2}
	baseline_twin	6.045×10^{-2}	1.235×10^{-1}	$\mathbf{2.156 \times 10^{-4}}$	$\mathbf{4.987 \times 10^{-4}}$
	synchronized_twin	$\mathbf{5.751 \times 10^{-3}}$	$\mathbf{1.362 \times 10^{-2}}$	3.398×10^{-4}	9.196×10^{-4}
Stress Wind Case	nominal	1.933×10^{-1}	2.698×10^{-1}	5.991×10^{-3}	3.685×10^{-2}
	baseline_twin	2.881×10^{-1}	5.123×10^{-1}	6.932×10^{-4}	1.529×10^{-3}
	synchronized_twin	$\mathbf{1.204 \times 10^{-2}}$	$\mathbf{2.534 \times 10^{-2}}$	$\mathbf{5.971 \times 10^{-4}}$	$\mathbf{1.507 \times 10^{-3}}$

The scaled deviation metrics measure how far the reference trajectory is from the plant, while the residual metrics measure how much of the deviation update is not explained by the local linear model in Eqs. 7.6–7.7. Lower deviation values indicate a closer plant–reference alignment. Lower residual values indicate that the local deviation model is more consistent with the recorded trajectory over one controller step. A

larger residual would indicate that the deviation update contains behavior that the local linear model does not explain well. This behavior can come from plant–twin mismatch, nonlinear effects, synchronization corrections, or other sample inconsistencies.

In the no-wind case, the baseline twin is exactly aligned with the plant because the imposed wind disturbance is removed, as expected. In the moderate wind case, the synchronized twin reduces the scaled deviation RMS by about 94.4% relative to the nominal reference and about 90.5% relative to the baseline twin. In the stress-wind case, the reduction is about 93.8% relative to the nominal reference and about 95.8% relative to the baseline twin. These reductions indicate that EKF synchronization provides a much closer local reference for the plant trajectory. This suggests that the digital twin synchronization using the EKF is successfully reducing local mismatch between the digital twin and the plant proxy.

The residual RMS values however show that in the moderate-wind case, the synchronized twin reduces the residual RMS by about 92.8% relative to the nominal reference, but it is higher than the baseline-twin residual, even though the baseline unsynchronized digital twin has a higher deviation. In the stress-wind case, the synchronized twin reduces the residual RMS by about 90.0% relative to the nominal reference and about 13.9% relative to the baseline twin. This indicates that especially in lower wind settings, when the unsynchorized twin is perturbed far from the plant, its local linear model can more reliably describe the error evolution, even though the reference may be further from the actual plant position. Even though the synchronized twin is much closer to the plant, it is being actively corrected using the EKF support layer, which may be introducing small effects that are well represented by the local linear model. In practice, this indicates the synchronized twin is more sensitive to updates.

The residual RMS values show that in the moderate-wind case, the synchronized twin reduces the residual RMS by about 92.8% relative to the nominal reference, but the baseline twin has the lowest residual RMS even though it has a much larger trajectory deviation. This can happen because the residual measures how well the local linear model explains the evolution of the deviation. The baseline twin can therefore have a smaller residual if its error evolves smoothly, even while the reference itself is farther from the plant. The synchronized twin is much closer to the plant, but the EKF correction and synchronization updates can introduce effects that are not fully represented by the local TVLQR linearization. In the stress-wind case, the synchronized twin gives both the lowest deviation RMS and the lowest residual RMS, suggesting that synchronization becomes more important as the unsynchronized baseline drifts farther from the plant. Figure 8.2 acts as another synchonization check and shows the unscaled plant–twin position mismatch for the

moderate-wind selector-on run.

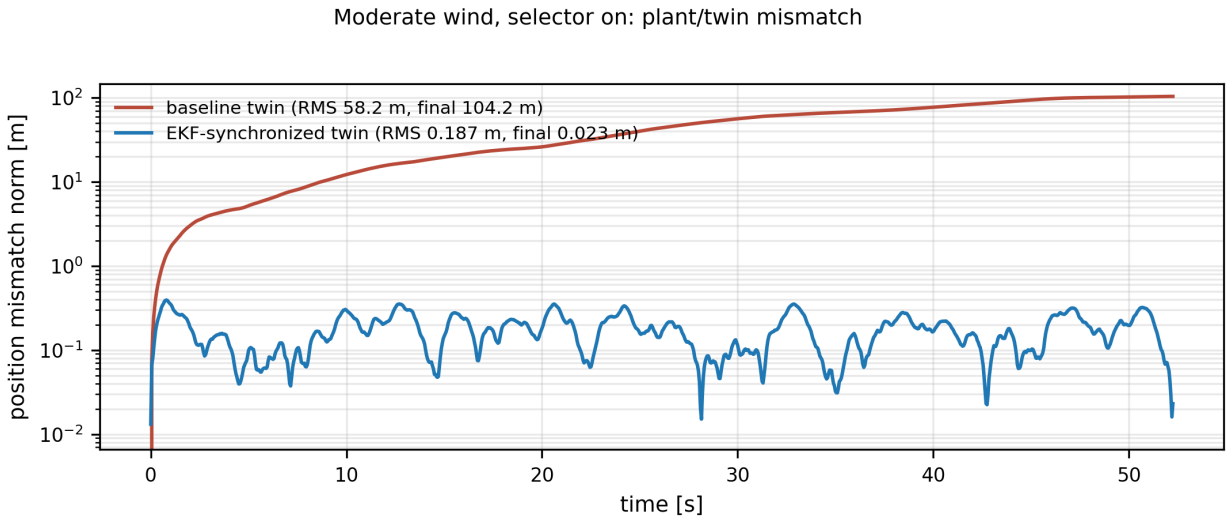


Figure 8.2: Plant–twin position-mismatch norm for the unsynchronized baseline twin and the EKF-synchronized twin in the moderate-wind selector-on run. The vertical axis is shown on a logarithmic scale. The unsynchronized twin accumulates drift over the maneuver, reaching an RMS position mismatch of $58.2\ \mathrm{m}$ and a final mismatch of $104.2\ \mathrm{m}$. The EKF-synchronized twin remains near the plant trajectory, with an RMS position mismatch of $0.187\ \mathrm{m}$ and a final mismatch of $0.023\ \mathrm{m}$.

Figure 8.2 provides a position-level validation on the synchronization behavior in the moderate-wind case. Unlike the scaled residual diagnostics in Table 8.2, this figure shows the unscaled position-mismatch norm over the full maneuver. The unsynchronized baseline twin drifts steadily away from the plant because it is propagated without the wind and correction information used by the synchronized twin. By the end of the simulation, the baseline position mismatch exceeds $100\ \mathrm{m}$, while the synchronized twin remains in the sub-meter range for almost the entire trajectory. This result indicates that EKF synchronization is keeping the digital twin state close enough to the plant to be useful for tracking support and terminal prediction. The RMS position mismatch decreases from $58.2\ \mathrm{m}$ for the baseline twin to $0.187\ \mathrm{m}$ for the synchronized twin, which is a reduction of more than 99%. The final position mismatch shows the same behavior, decreasing from $104.2\ \mathrm{m}$ to $0.023\ \mathrm{m}$.

8.4 Forecast-Selected Terminal Policy Landing Results

The forecast-selected terminal policy supervisor was evaluated using selector-on/selector-off ablation studies under two wind-disturbance cases. In the selector-off case, the aircraft uses the EKF-synchronized tracking

architecture with the forecast-selected terminal policy selector disabled. In the selector-on case, the same EKF-synchronized tracking architecture is used, but the synchronized digital twin evaluates candidate terminal policies using the predicted landing quality score (Eq. (7.2)) during the final approach and selects the terminal intervention with the most favorable predicted touchdown quality. This comparison evaluates whether the synchronized digital twin changes the terminal behavior of the airplane before runway contact.

In this thesis, an undesirable predicted landing condition is defined using the main-gear contact geometry predicted by the synchronized digital twin over the selector horizon. A candidate deployment is considered undesirable if the predicted touchdown has a hard-contact sink rate greater than $1.80\ \mathrm{m/s}$ downward, a nose-down contact pitch angle below -1.0°, or a main-gear clearance penetration below $-0.03\ \mathrm{m}$. The selector also penalizes softer departures from the desired touchdown condition, including sink rates above the soft limit of $1.20\ \mathrm{m/s}$, pitch angles below the soft contact limit of -0.50°, and touchdown locations more than $120\ \mathrm{m}$ beyond the nominal touchdown station. The nominal desired contact condition is a main-gear clearance of approximately $0.08\ \mathrm{m}$, a downward sink rate of $0.80\ \mathrm{m/s}$, and a pitch attitude near 0°. If the synchronized twin predicts any hard-contact, nose-down-contact, clearance-penetration, floating/no-contact, or late-touchdown condition within the $5.0\ \mathrm{s}$ forecast horizon, the terminal policy selector evaluates the available candidate terminal policies and selects the policy whose predicted rollout gives the best landing-quality score subject to these safety penalties.

Table 8.3: Terminal policy selector timing, rollout, and scoring parameters used in the selector-on runs. The weights are grouped because the absolute cost values are only used for ranking candidate policies inside the selector.

Category	**Quantity**	**Value**
Selector update	Refresh rate	2 Hz
Forecast rollout	Horizon, output step, internal rate	5.0 s, 0.20 s, 25 Hz
Landing-quality weights	Clearance, clearance-rate, pitch, speed	$w_h = 10$, $w_{\dot{h}} = 8$, $w_\theta = 30$, $w_V = 0.5$
Regularization/location weights	Command change, touchdown location	$w_u = 0.05$, $w_N = 0.08$
Event penalty weights	No contact, float, constraint violation	$w_{\mathrm{nc}} = 8$, $w_{\mathrm{float}} = 25$, $w_{\mathrm{cv}} = 100$
Landing penalty weights	Hard contact, nose-down, penetration, late TD	$w_{\mathrm{hard}} = 220$, $w_{\mathrm{nose}} = 120$, $w_{\mathrm{pen}} = 220$, $w_{\mathrm{late}} = 160$
Touchdown targets	Clearance, sink rate, pitch, speed	$h^\star = 0.08$ m, $V^\star_{\mathrm{sink}} = 0.80$ m/s, $\theta^\star = 0^\circ$, $V^\star = 90$ m/s
Hard safety flags	Hard contact, nose-down, penetration	$\dot{h}_{\mathrm{mg},c} < -1.80$ m/s, $\theta_c < -1.0^\circ$, $h_{\mathrm{mg,min}} < -0.03$ m
Soft penalty / preference thresholds	Soft sink, soft pitch, late touchdown	$V_{\mathrm{sink},c} > 1.20$ m/s, $\theta_c < -0.50^\circ$, $N_c - N^\star > 120$ m

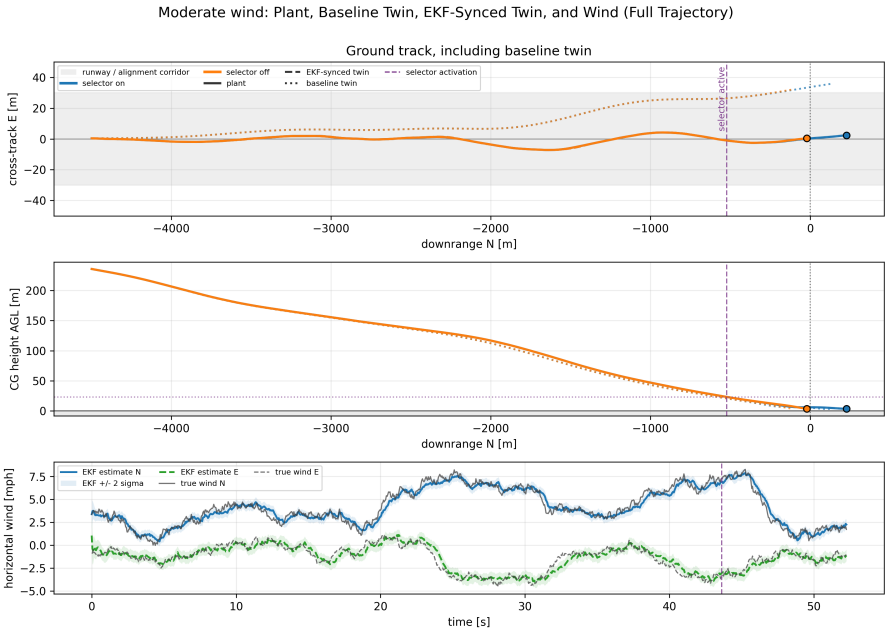


Figure 8.3: Full-trajectory moderate-wind case showing the plant trajectory, selector-on and selector-off terminal outcomes, baseline twin, EKF-synchronized twin, and EKF wind estimates.

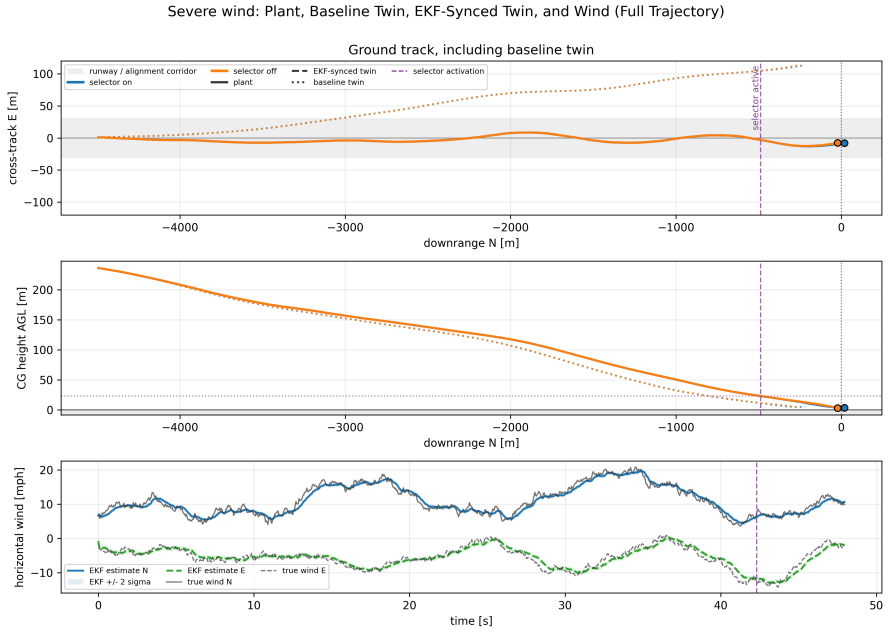


Figure 8.4: Full-trajectory severe-wind case showing the plant trajectory, selector-on and selector-off terminal outcomes, baseline twin, EKF-synchronized twin, and EKF wind estimates.

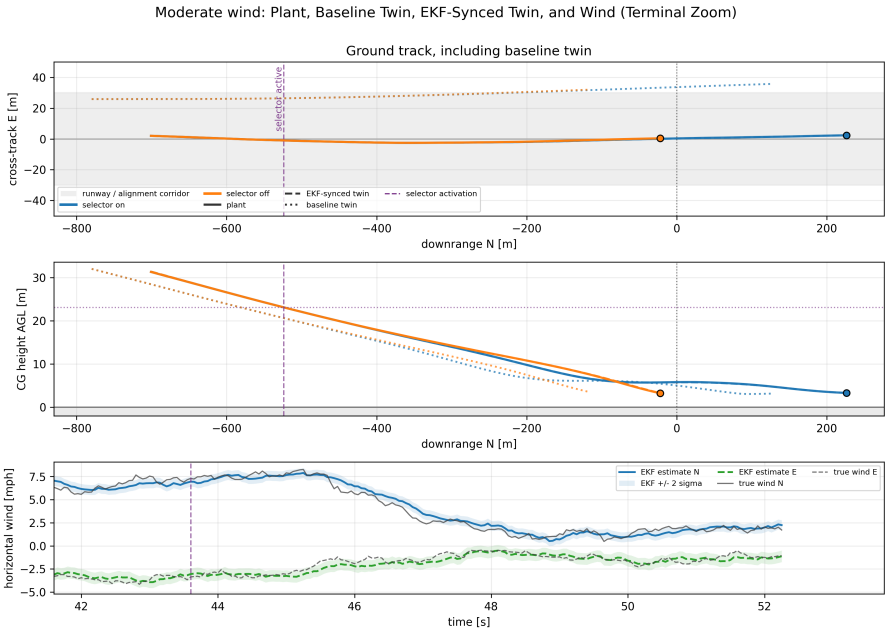


Figure 8.5: Terminal-zoom moderate-wind case showing the selector-on and selector-off plant trajectories, baseline twin, EKF-synchronized twin, and EKF wind estimates near terminal policy activation.

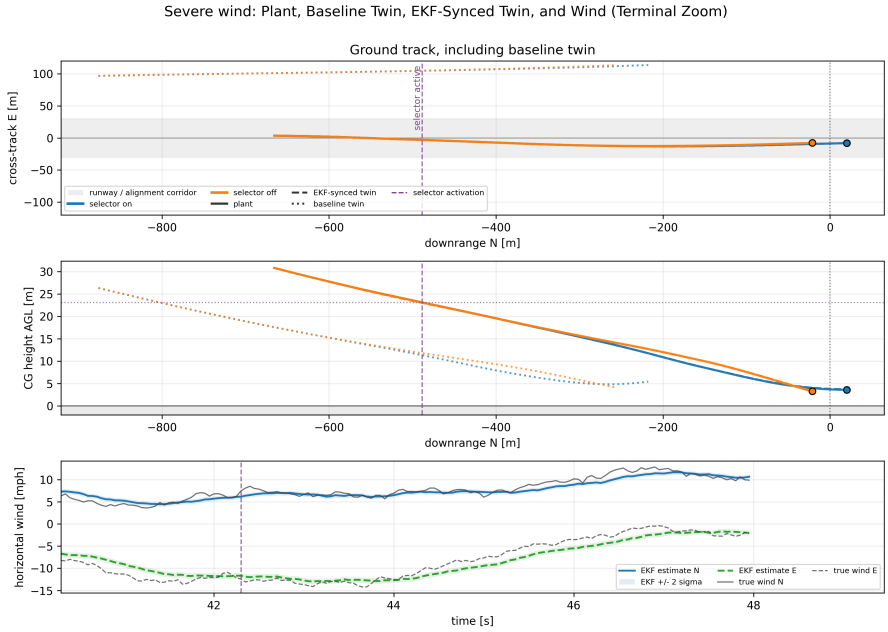


Figure 8.6: Terminal-zoom severe-wind case showing the selector-on and selector-off plant trajectories, baseline twin, EKF-synchronized twin, and EKF wind estimates near terminal policy activation.

Table 8.3 summarizes the selector timing and the main scoring parameters used in the reported selector-on runs. The full scoring rule is defined in Section 7. The largest penalties are assigned to hard contact, clearance penetration, late touchdown, and nose-down contact. Lower-weight terms rank the remaining candidates by clearance, clearance rate, pitch, speed, touchdown location, and command variation. Candidate deployments are declared hard-unsafe when they predict excessive contact sink rate, nose-down contact, main-gear penetration, or floating/no-contact behavior. Late touchdown is only subjected to a cost penalty. The selector therefore prefers earlier touchdown when it can do so without producing hard contact, nose-down contact, or penetration, but it may accept a farther-downrange touchdown if that deployment gives a better overall terminal condition.

Figures 8.3 and 8.4 show the full-trajectory behavior for the moderate and severe wind selector ablations. In both cases, the selector-on and selector-off plant trajectories remain nearly identical through most of the approach, and the visible difference appears mainly after the terminal selector becomes active. The zoomed in plots in Figures 8.5 and 8.6 shows only the region where the terminal selector actually affects the closed-loop response. After activation, the trajectories begin to separate mainly in the vertical profile and terminal downrange behavior rather than in the lateral ground track. This behavior is expected as the purpose of the digital twin is to adjust the terminal approach trajectory so that the touchdown is softer.

In the moderate wind case, the altitude profile shows that the digital twin chose to deviate from the nominal trajectory and continue keeping the airplane afloat to increase the pitch rate and provide a softer landing further down the runway. This is consistent with a policy choice that reduces aggressive closure near the ground at the cost of delayed contact. The ground-track panel shows that this change is not primarily a lateral correction. The severe-wind terminal behavior shows a similar trend, but was able to achieve main gear contact much closer to the nominal endpoint. The bottom subplot shows the EKF estimated and actual wind in the N and E directions along the simulation time. The altitude panels show aircraft center-of-gravity height above ground, not main-gear clearance. Main-gear contact is evaluated separately using the gear-geometry-based clearance variable h_{mg}, which is reported in the terminal metric tables.

Table 8.4: Terminal landing metrics for EKF-supported tracking with and without digital-twin terminal policy selection. First contact is reported using the $h_{\mathrm{mg}} \leq 0.09\ \mathrm{m}$ threshold-contact convention.

Case	Configuration	Main gear contact?	t_c (s)	N_c (m)	E_c (m)	$h_{\mathrm{mg},c}$ (m)	$\dot{h}_{\mathrm{mg},c}$ (m/s)	$V_{\mathrm{sink},c}$ (m/s)	θ_c (deg)
Moderate wind	EKF, selector off	Yes	49.34	−23.02	0.42	0.090	−3.29	3.07	−1.68
Moderate wind	DT selector on	Yes	52.22	224.0	2.35	0.090	−0.73	0.70	−0.83
Severe wind	EKF, selector off	Yes	47.45	−26.05	−7.94	0.090	−5.12	4.88	−2.72
Severe wind	DT selector on	Yes	47.92	17.1	−8.14	0.090	−0.25	0.29	1.36

Table 8.5: Additional terminal diagnostics for EKF-supported tracking with and without digital-twin terminal policy selection.

Case	Configuration	α_c (deg)	u_c (m/s)	$h_{\mathrm{mg,min}}$ (m)	$t(h_{\mathrm{mg,min}})$ (s)
Moderate wind	EKF, selector off	0.18	85.69	0.058	49.35
Moderate wind	DT selector on	−0.36	84.61	0.072	52.25
Severe wind	EKF, selector off	−0.15	86.54	−0.147	47.50
Severe wind	DT selector on	0.74	86.23	0.085	47.95

Table 8.6: Tracking-error metrics for EKF-supported tracking with and without digital-twin terminal policy selection. The selector-on values should be interpreted as closed-loop outcome metrics, not as a pure tracking improvement, because the terminal policy selector intentionally modifies the terminal behavior.

Case	Configuration	Pos. RMSE (m)	Max pos. err. (m)	N-RMSE (m)	E-RMSE (m)	D-RMSE (m)	Vel. RMSE (m/s)	θ-RMSE (deg)
Moderate wind	EKF, selector off	13.77	18.86	13.47	2.77	0.67	0.67	0.39
Moderate wind	DT selector on	35.37	136.51	35.19	2.71	2.26	1.00	0.61
Severe wind	EKF, selector off	19.80	28.72	18.87	5.71	1.83	1.02	0.62
Severe wind	DT selector on	28.43	115.48	27.67	5.83	3.00	0.99	0.62

Both selector-off wind cases successfully achieves main gear runway contact, but the contact geometry was seen to be harsh as outlined in Section 6.6. With the terminal policy selector enabled, the magnitude of the sink rate and main-gear clearance rate at contact is substantially reduced, as seen in Tables 8.4–8.5 which compare the EKF-supported tracking case with the digital-twin terminal policy selector enabled and disabled. With the selector disabled, the EKF-supported controller reaches the runway region in both wind cases, but the contact geometry remains aggressive. In the moderate-wind case, the aircraft reaches the main-gear contact threshold with a clearance rate of $-3.29\ \mathrm{m/s}$, a sink rate of $3.07\ \mathrm{m/s}$, and a pitch angle

of -1.68°. With the terminal policy selector enabled, the same case reaches contact later in time and contact location, but the clearance-rate magnitude drops to $0.73\ \mathrm{m/s}$, the sink rate drops to $0.70\ \mathrm{m/s}$, and the pitch angle moves closer to level at -0.83°.

The severe-wind case shows stronger landing improvements. In the EKF-supported tracking (without digital twin policy selection), the aircraft reaches the contact threshold with a clearance rate of $-5.12\ \mathrm{m/s}$, a sink rate of $4.88\ \mathrm{m/s}$, and a pitch angle of -2.72°. The digital twin supported policy selection resulted in a clearance-rate magnitude of $0.25\ \mathrm{m/s}$, the sink rate was reduced to $0.29\ \mathrm{m/s}$, and the contact pitch became positive at 1.36°. The minimum main-gear clearance also changes from $-0.147\ \mathrm{m}$ to $0.085\ \mathrm{m}$, removing the penetration event seen in the selector-off case.

This improvement however, needed to come at the cost of the main gear contact time and location. In the moderate-wind case, the selector-on trajectory reaches contact about $2.9\ \mathrm{s}$ later and roughly $247\ \mathrm{m}$ farther downrange than the selector-off trajectory. This is the main tradeoff in the current terminal policy set. The selector reduces the severity of first contact by allowing the aircraft to spend more time arresting descent, but that softer terminal behavior delays touchdown. In the severe-wind case, the downrange shift is much smaller, with contact moving from $N_c = -26.05\ \mathrm{m}$ to $N_c = 17.1\ \mathrm{m}$. Hence, the policy selector does not introduce the same range penalty in both cases. Its effect depends on the terminal state at activation and on which candidate policies are predicted to avoid hard or nose-down contact.

Table 8.6 shows the RMSE values for the EKF supported tracking controller and the synchronized digital twin terminal policy supported controller, but should be interpreted differently from the earlier tracking-error tables. The policy selection mechanism deliberately changes the terminal behavior when the synchronized digital twin predicts poor contact quality. Hence, the position RMSE and maximum position error increase in the selector-on cases, especially for the moderate-wind run where touchdown is delayed. The RMSE table however is still included for consistency when comparing the different control architectures on following the nominal trajectory.

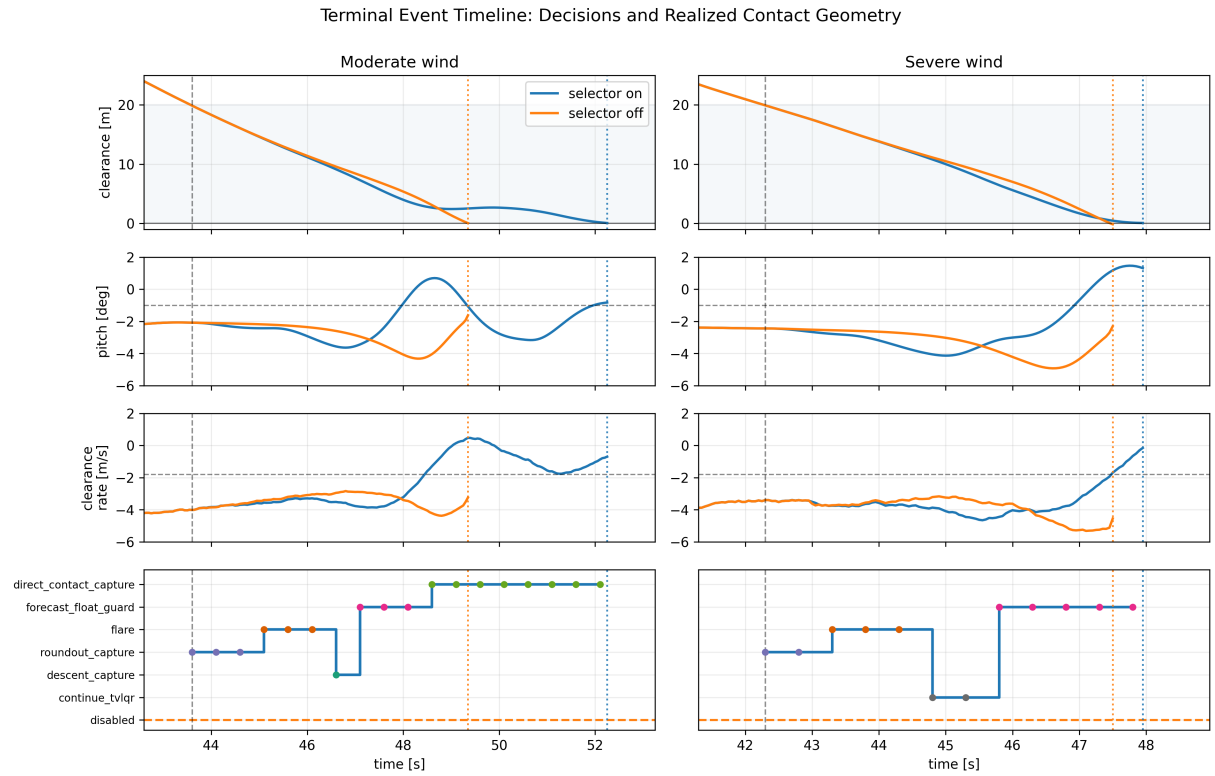


Figure 8.7: Terminal event timeline for the moderate- and severe-wind ablation cases. The upper rows compare selector-on and selector-off main-gear clearance, pitch, and main-gear clearance rate. The lower row shows the selected terminal policy sequence for the selector-on case and the disabled selector-off baseline. Vertical dotted lines indicate first-contact times for the corresponding runs.

Figure 8.7 ties the realized terminal response to the selected policy history. In both wind cases, the selector remains inactive until the late terminal window, then begins with softer terminal-shaping policies before switching to more contact-oriented policies. The early selections, such as *Sink Rate Reduction* and *Flare Shaping*, attempt to manage pitch and reduce aggressive runway closure. Later selections, especially *Float Prevention* and *Contact Capture*, indicate that the main predicted risk has shifted toward floating, delayed contact, or insufficient closure behavior. The selector then chose the stronger touchdown policies, which is why we can see that *Contact Capture* is engaged in the moderate wind case but not the severe wind case due to the float behavior just above the runway.

In the moderate-wind case, the selected sequence progresses from *Sink Rate Reduction* to *Flare Shaping*, briefly through *Runway Closure Tracking*, then to *Float Prevention*, and finally to *Contact Capture*. By that

point, the trajectory had already reduced the steep terminal clearance-rate trend, but the aircraft remains above the runway region likely due to the thrust being too high. The final contact-capture behavior is therefore trying to prevent the softened terminal response from turning into excessive float. In the severe-wind case, the selector is seen to choose a different sequence of policies. Starting with *Sink Rate Reduction* and *flare*, then briefly returning to *Nominal Tracking*, and then switching to *Float Prevention* through the final terminal interval. The corresponding pitch and clearance-rate traces shows better recovery from the nose-down and similar closure behavior as seen in the selector-off case. Unlike the moderate-wind run, the final selected behavior does not end in *Contact Capture*. This suggests that the forecasted terminal condition could be managed with float-guard behavior rather than the strongest contact-capture policy, and that the policies themselves can be better tuned so that *Contact Capture* does not become engaged, as it uses the most aggressive corrective actions.

Chapter 9

DISCUSSION, LIMITATIONS, AND FUTURE WORK

The results in the previous chapter suggests that the synchronized digital twin is able to act as a decision-support layer around the nominal TVLQR landing controller. This thesis considers whether an imperfect physics-based model can be updated from the measured aircraft response and then used to evaluate near-term landing consequences before touchdown. The nominal trajectory and tracking controller may bring the aircraft into the runway region while still producing poor terminal geometry. The digital twin therefore adds value by providing prediction of future states and trajectory adjustments that results in a smoother landing.

The synchronization and residual-mismatch results show that before the terminal selector becomes active, the EKF-synchronized twin gives a much closer local reference for the plant trajectory than the baseline unsynchronized digital twin. This is shown by the reduction in scaled deviation RMS in Table 8.2. The baseline digital twin accumulates large position drift, while the synchronized digital twin remains close to the plant over the maneuver. These results support the use of the EKF synchronization as an accurate and real time data assimilation layer which keeps the digital twin initialized near the current aircraft response, keeping the digital model of the aircraft updated with the actual aircraft state before any short-horizon forecast or terminal policy ranking is applied.

The results then show that the synchronized digital twin provides useful predictive information by propagating the current aircraft state forward in time. The synchronized closed-loop digital twin was shown to be accurate in extrapolating the current nonlinear aircraft state forward in time by a $5.0\ \mathrm{s}$ horizon and has lower RMSE than the unsynchronized digitaL twin for 3D position, main-gear clearance, and sink rate in Table 8.1. These quantities are directly tied to terminal landing behavior because they describe whether the aircraft is closing toward the runway with acceptable wheel clearance and vertical speed. The synchronized twin is useful here because it propagates the nonlinear aircraft model together with the stored TVLQR feedback structure, allowing for the evaluation of future aircraft states, depending on actions taken at the current time. Because the digital twin contains up to date information on the airplane's pose and accurately estimates unmodeled wind disturbance, different control actions can be propagated forward in time simultaneously to

guide decision making in a computationally efficient, and tractable manner.

The terminal policy selector architecture is used as a demonstration of the digital twin as a decision-support mechanism. In both wind cases, the selector-off configuration reaches main-gear contact, but does so with a high sink rate. The synchronized twin was used to identify ahead of time, the poor terminal contact behavior early enough to select a better terminal policy. Tables 8.4 and 8.5 show that enabling the selector substantially reduces the magnitude of the main-gear clearance rate and sink rate at contact. In the moderate-wind case, this produces a softer contact at the cost of delayed touchdown farther downrange. In the severe-wind case, the selector gives a greater improvement in contact quality while keeping the touchdown location much closer to the nominal endpoint. This indicates that the selector behavior depends on the terminal state at activation and on which candidate policy is predicted to best balance closure, pitch, sink rate, and contact location. Given the long runway for airplanes, the exact location of touchdown could differ, but the individual policies as well as their logic can be adjusted depending on the specific scenario and its hard constraints.

The policy history results also show that the selector is not acting as a fixed flare command. Figure 8.7 shows that the selected policy changes as the forecasted terminal condition changes. Early selections emphasize sink-rate reduction and flare shaping, while later selections shift toward float prevention or contact capture when the predicted risk becomes delayed contact or insufficient runway closure. This behavior is important because it shows that the synchronized twin is being used to compare future consequences of multiple terminal actions, not just to apply a preprogrammed correction at a fixed time. The selector intentionally moves the aircraft away from the nominal terminal profile when the forecast predicts poor touchdown quality, so the increase in position error is a closed-loop outcome of the terminal decision logic.

The results support the thesis claim that a synchronized physics-based digital twin can provide useful decision support for nonlinear aircraft landing under wind-induced plant–twin mismatch. The EKF synchronization layer reduces the state mismatch between the plant and the twin, the short-horizon forecast study shows that the synchronized closed-loop twin has accurate future prediction for landing-relevant quantities, and the terminal selector uses those predictions to improve contact quality compared to the EKF-supported tracking alone. The resulting architecture keeps the computationally expensive trajectory optimization and TVLQR gain construction offline, while using the synchronized twin online to evaluate a small set of interpretable terminal behaviors. This makes the digital twin a consequence prediction and policy-selection layer around the controller.

Limitations include the plant being represented by the PSim-RCAM proxy, and the synchronized digital twin being built from the same aircraft model, but with deliberate mismatch in wind, actuator response, and initial condition assumptions. Without validating on physical systems using actual flight-test data, the results can only support the claim that the synchronized twin improves prediction and terminal decision-making in the tested simulation cases.

The terminal policy selector performance is also tied to the specific candidate set, scoring weights, thresholds, and forecast horizon used in the reported runs. The 5.0 s horizon was used to give the selector enough time to influence the terminal trajectory, while minimizing the loss in accuracy as the prediction horizon increases. Changing the horizon would change which part of the landing sequence is being scored and could alter the ranking of the candidate policies. In the moderate wind case, the selector reduced contact severity but accepted a farther-downrange touchdown, showing that the current scoring logic trades touchdown softness against location error rather than enforcing all touchdown preferences as hard constraints. A more robust policy selection method, with better tuned weights and more complex logic may be able to address this issue.

Finally, the terminal selector is used on a small number of wind cases rather than on a broad Monte Carlo study with varied approach geometries, turbulence levels, sensor noise, and other discrepancies. The results currently do not characterize the limits of failure of the existing architecture. More extensive testing would be needed to determine how sensitive the selector is to tuning choices, forecast errors, late activation timing, and candidate-policy design. Future iterations should separate hard safety vetoes from soft preference penalties more systematically and evaluate whether multi-horizon scoring or adaptive cost tuning can reduce delayed-touchdown behavior without reintroducing hard or nose-down contact.

Future work includes the use of stored digital-twin decision data for post-flight learning and tuning. The aircraft state, wind estimate, selected policy, predicted terminal metrics, and the actual terminal outcome is recorded. This creates a dataset that connects forecast conditions to terminal landing behavior which could then be used to improve the terminal policy selector through prediction error correction or data-driven cost tuning. The wind disturbance model could also be improved upon as the current EKF estimates the pose-and-wind state, with wind represented as a first-order Gauss–Markov bias. This does not represent spatially varying wind, vertical shear, gust gradients, wake effects, or turbulence that vary across the approach corridor. Another extension is to make the landing using the digital twin more realistic near touchdown. The

current controller uses main gear clearance and geometric contact logic to evaluate terminal behavior, but it does not model full landing-gear compression, tire-runway interaction, ground forces, runway friction, spoiler deployment, and braking. A more complete landing model would allow the digital twin to score touchdown quality using the additional metrics.

The policy-selection layer could also be extended toward constrained optimal control. The finite-candidate terminal selector could be compared against a synchronized-twin MPC formulation. The MPC version could optimize terminal inputs directly subject to actuator limits, clearance constraints, touchdown-location constraints, and sink-rate limits. The current selector is simpler and computationally lighter because it evaluates a fixed set of interpretable terminal policies instead of solving a new optimization problem online, but future work could quantify this tradeoff. Control Barrier Functions (CBFs) or an MPC-CBF formulation could also be used to provide forward invariance guarantees against hard constraints to provide for more explicit safety constraints and reject candidate policies that would violate safety constraints.

Finally, the same architecture could be extended to other applications such as smaller UAVs and quadrotor drone landing. The general structure of the method is to generate a nominal trajectory offline, use a tracking controller executes the reference, using a data assimilation method to synchronize the digital twin with measured vehicle response, then the synchronized twin predicts the consequences of candidate terminal behaviors. A drone landing problem for example would require different dynamics, sensors, disturbance models, contact metrics, and terminal policies, which explores the same decision support idea, but applied to a different setting.

Appendix A

REPOSITORY AND REPRODUCIBILITY NOTES

The code and simulation artifacts associated with this thesis are available at:

https://github.com/tralalerro-tralala/Digital-Twin-Aircraft-Landing

The repository contains the implementation used for the nominal landing trajectory generation, TVLQR tracking controller, EKF-based synchronization, digital-twin forecast evaluation, and terminal-policy selector. The main simulation results in Chapters 6 and 8 were generated from this codebase.

BIBLIOGRAPHY

[1] M. Soori, B. Arezoo, and R. Dastres, "Digital Twin for Smart Manufacturing, A Review," *Sustainable Manufacturing and Service Economics*, June 2023.

[2] L. Li, S. Aslam, A. Wileman, and S. Perinpanayagam, "Digital Twin in Aerospace Industry: A Gentle Introduction," *IEEE Access*, vol. 10, pp. 9543–9562, 2022.

[3] G. Bisanti, L. Mainetti, T. Montanaro, L. Patrono, and I. Sergi, "Digital twins for aircraft maintenance and operation: A systematic literature review and an IoT-enabled modular architecture," *Internet of Things*, vol. 24, p. 100991, Nov. 2023.

[4] W. Meng, Y. Yang, J. Zang, H. Li, and R. Lu, "DTUAV: a novel cloud–based digital twin system for unmanned aerial vehicles," *SIMULATION*, vol. 99, p. 003754972211095, Aug. 2022.

[5] E. Fakhraian, I. Semanjski, S. Semanjski, and E.-H. Aghezzaf, "Towards Safe and Efficient Unmanned Aircraft System Operations: Literature Review of Digital Twins' Applications and European Union Regulatory Compliance," *Drones*, vol. 7, p. 478, July 2023.

[6] A. McClellan, J. Lorenzetti, M. Pavone, and C. Farhat, "A physics-based digital twin for model predictive control of autonomous unmanned aerial vehicle landing," *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, vol. 380, p. 20210204, Aug. 2022.

[7] A. Colagrossi, S. Silvestrini, A. Brandonisio, and M. Lavagna, "A Digital Twin Approach for Spacecraft On-Board Software Development and Testing," *Aerospace*, vol. 13, p. 55, Jan. 2026.

[8] H. Wang, J. Yang, Y. Shi, H. Huo, and X. Ye, "Review of the digital twin design of the smart satellite thermal control structure," *Acta Astronautica*, vol. 239, pp. 491–521, Feb. 2026.

[9] M. Reza, F. Faraji, and A. Knoll, "Digital twins for electric propulsion technologies," *Journal of Electric Propulsion*, vol. 3, p. 25, Oct. 2024.

[10] S. Gudeta and A. Karimoddini, "Design of a Smooth Landing Trajectory Tracking System for a Fixed-wing Aircraft," in *2019 American Control Conference (ACC)*, pp. 5674–5679, 2019.

[11] R. Lungu and M. Lungu, "Application of H2/H and dynamic inversion techniques to aircraft landing control," *Aerospace Science and Technology*, vol. 46, pp. 146–158, Oct. 2015.

[12] F. Liao, J. L. Wang, E. K. Poh, and D. Li, "Fault-Tolerant Robust Automatic Landing Control Design," *Journal of Guidance, Control, and Dynamics*, vol. 28, pp. 854–871, Sept. 2005.

[13] T. Wagner and J. Valasek, "Digital Autoland Control Laws Using Direct Digital Design and Quantitative Feedback Theory," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2006.

[14] A. Tripathi, V. Patel, and R. Padhi, "Autonomous Landing of Fixed Wing Unmanned Aerial Vehicle with Reactive Collision Avoidance," *IFAC-PapersOnLine*, vol. 51, pp. 474–479, Jan. 2018.

[15] M. Lungu, "Backstepping and dynamic inversion control techniques for automatic landing of fixed wing unmanned aerial vehicles," *Aerospace Science and Technology*, vol. 120, p. 107261, Jan. 2022.

[16] Z. Latif, A. Shahzad, A. I. Bhatti, J. F. Whidborne, and R. Samar, "Autonomous Landing of an UAV Using H Based Model Predictive Control," *Drones*, vol. 6, p. 416, Dec. 2022.

[17] L. Wang, Z. Zhang, Q. Zhu, and R. Dong, "Longitudinal automatic carrier landing system guidance law using model predictive control with an additional landing risk term," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, p. 095441001774643, Dec. 2017.

[18] S. R. Nekoo and A. Ollero, "A robust state-dependent Riccati equation controller with parameter uncertainty and matched disturbance," *Journal of the Franklin Institute*, vol. 360, pp. 14584–14595, Dec. 2023.

[19] V. Azimi, S. Farzan, and S. Hutchinson, "A Robust Time-Varying Riccati-Based Control for Uncertain Nonlinear Dynamical Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, July 2022.

[20] T. Çimen, "State-Dependent Riccati Equation (SDRE) Control: A Survey," *IFAC Proceedings Volumes*, vol. 41, pp. 3761–3775, Jan. 2008.

[21] O. Tekinalp and A. Prach, "Development of a State Dependent Riccati Equation based Tracking Flight Controller for an Unmanned Aircraft," Aug. 2013.

[22] M. Keshvarinia, C. MacKenzie, and Z. Zhao, "A simulation-based digital twin model for data-driven decision optimization," Oct. 2025.

[23] "Robust Flight Control Design Challenge Problem Formulation and Manual: the Research Civil Aircraft Model (RCAM)," June 1995.

[24] F. T. Engineering, "flight-test-engineering/PSim-RCAM," Apr. 2026. original-date: 2022-09-27T09:47:57Z.

[25] T. Reynolds, D. Malyuta, M. Mesbahi, B. Acikmese, and J. M. Carson, "A Real-Time Algorithm for Non-Convex Powered Descent Guidance," in *AIAA Scitech 2020 Forum*, AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, Jan. 2020.

[26] T. P. Reynolds and M. Mesbahi, "The Crawling Phenomenon in Sequential Convex Programming," in *2020 American Control Conference (ACC)*, pp. 3613–3618, July 2020. ISSN: 2378-5861.

[27] "Convex Optimization – Boyd and Vandenberghe."

[28] S. Tafazzol and E. Taheri, "Incorporating the nonlinearity index into adaptive-mesh sequential convex optimization for minimum-fuel low-thrust trajectory design," Nov. 2025. arXiv:2511.08837 [eess.SY].

[29] R. Bonalli, T. Lew, and M. Pavone, "Analysis of Theoretical and Numerical Properties of Sequential Convex Programming for Continuous-Time Optimal Control," Sept. 2022. arXiv:2009.05038 [math.OC].

[30] L. Xie, X. Zhou, H.-B. Zhang, and G.-J. Tang, "Higher-Order Soft-Trust-Region-Based Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, vol. 46, pp. 2199–2206, Nov. 2023.

[31] T. Kim, A. G. Kamath, N. Rahimi, J. Corleis, B. Açıkmeşe, and M. Mesbahi, "Six-Degree-of-Freedom Aircraft Landing Trajectory Planning with Runway Alignment," June 2025. arXiv:2405.16680 [math.OC].

[32] S. Diamond and S. Boyd, "CVXPY: A Python-Embedded Modeling Language for Convex Optimization," June 2016. arXiv:1603.00943 [math.OC].

[33] A. Pascoal and I. Kaminer, "On the design of gain-scheduled trajectory tracking controllers," *International Journal of Robust and Nonlinear Control*, vol. 12, pp. 797–839, July 2002.

[34] C. V. Girish, F. Emilio, H. P. Jonathan, and L. Hugh, "Nonlinear Flight Control Techniques for Unmanned Aerial Vehicles," in *Handbook of Unmanned Aerial Vehicles*, pp. 577–612, Springer, Dordrecht, 2015.

[35] G. Wu and K. Sreenath, "Variation-Based Linearization of Nonlinear Systems Evolving on SO(3) and \mathbb S^2," *IEEE Access*, vol. 3, pp. 1592–1604, 2015.

[36] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, Mar. 1960.

[37] A. H. Jazwinski, "Stochastic processes and filtering theory," Jan. 1970. NTRS Author Affiliations: NTRS Document ID: 19700053459 NTRS Research Center: Legacy CDMS (CDMS).

[38] A. Cho, J. Kim, S. Lee, and C. Kee, "Wind Estimation and Airspeed Calibration using a UAV with a Single-Antenna GPS Receiver and Pitot Tube," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 1, pp. 109–117, 2011.

[39] M. Alam, J. Whidborne, and M. Westwood, "Navigation Algorithm for a Twin-Engine Turboprop Aircraft using an Extended Kalman Filter," in *Proceedings of the 2024 CEAS EuroGNC conference*, June 2024.

[40] K. Sun, C. D. Regan, and D. Gebre-Egziabher, "Observability and Performance Analysis of a Model-Free Synthetic Air Data Estimator," 2019.

[41] M. Abdelrahman, E. Macatulad, B. Lei, M. Quintana, C. Miller, and F. Biljecki, "What is a Digital Twin Anyway? Deriving the Definition for the Built Environment from over 15,000 Scientific Publications," 2025.

[42] J. R. Carpenter and C. N. D'Souza, "Navigation Filter Best Practices," Mar. 2025. NTRS Author Affiliations: Goddard Space Flight Center, Johnson Space Center NTRS Report/Patent Number: 20180003657 NTRS Document ID: 20250002787 NTRS Research Center: Langley Research Center (LaRC).

[43] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. Jan. 2006.

[44] G. Chowdhary and R. Jategaonkar, "Aerodynamic parameter estimation from flight data applying extended and unscented Kalman filter," *Aerospace Science and Technology*, vol. 14, no. 2, pp. 106–117, 2010.

[45] H. Lu, L. Gao, Y. Yan, M. Hou, and C. Wang, "Wind disturbance compensated path-following control for fixed-wing UAVs in arbitrarily strong winds," *Chinese Journal of Aeronautics*, vol. 37, pp. 431–445, Feb. 2024.

[46] R. W. Beard, J. Ferrin, and J. Humpherys, "Fixed Wing UAV Path Following in Wind With Input Constraints," *IEEE Transactions on Control Systems Technology*, vol. 22, pp. 2103–2117, Nov. 2014.

[47] S. Benders, A. Wenz, and T. A. Johansen, "Adaptive Path Planning for Unmanned Aircraft Using In-flight Wind Velocity Estimation," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 483–492, June 2018. ISSN: 2575-7296.

[48] A. P. Aguiar and J. P. Hespanha, "Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty," *IEEE Transactions on Automatic Control*, vol. 52, pp. 1362–1379, Aug. 2007.

[49] S. Park, J. Deyst, and J. P. How, "Performance and Lyapunov Stability of a Nonlinear Path Following Guidance Method," *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 1718–1728, Nov. 2007.

[50] R. Rysdyk, "Unmanned Aerial Vehicle Path Following for Target Observation in Wind," *Journal of Guidance, Control, and Dynamics*, vol. 29, pp. 1092–1100, Sept. 2006.

[51] "Airplane Flying Handbook | Federal Aviation Administration."

[52] Z. Manchester and S. Kuindersma, "Robust direct trajectory optimization using approximate invariant funnels," *Autonomous Robots*, vol. 43, pp. 375–387, Feb. 2019.

[53] T. P. Reynolds, D. Malyuta, M. Mesbahi, and B. Acikmese, "Temporally-Interpolated Funnel Synthesis for Nonlinear Systems," July 2020.

[54] T. Kim, P. Elango, T. P. Reynolds, B. Açıkmeşe, and M. Mesbahi, "Optimization-based Constrained Funnel Synthesis for Systems with Lipschitz Nonlinearities via Numerical Optimal Control," *IEEE Control Systems Letters*, vol. 7, pp. 2875–2880, 2023. arXiv:2303.10504 [math].

[55] T. Kim, P. Elango, and B. Acikmese, "Joint Synthesis of Trajectory and Controlled Invariant Funnel for Discrete-time Systems with Locally Lipschitz Nonlinearities," Jan. 2024. arXiv:2209.03535 [math].

[56] T. Kim, D. Luo, and B. Açıkmeşe, "Continuous-time Constrained Funnel Synthesis for Incrementally Quadratic Nonlinear Systems," Nov. 2025. arXiv:2511.08868 [math].

[57] H. J. v. Waarde, M. K. Camlibel, and M. Mesbahi, "From noisy data to feedback controllers: non-conservative design via a matrix S-lemma," Dec. 2020. arXiv:2006.00870 [math].

[58] S. Shakeri and M. Mesbahi, "Robust Data-Driven Control for Nonlinear Systems Using their Digital Twins and Quadratic Funnels," Oct. 2025. arXiv:2510.01406 [eess.SY].

[59] E. National Academies of Sciences, N. A. o. Engineering, D. o. E. a. L. Studies, D. o. E. a. P. Sciences, B. o. A. S. Climate, , B. o. L. Sciences, C. S. a. T. Board, C. o. A. a. T. Statistics, B. o. M. S. Analytics, , and C. o. F. R. G. a. F. D. f. D. Twins, "The Digital Twin Landscape," in *Foundational Research Gaps and Future Directions for Digital Twins*, National Academies Press (US), Mar. 2024.