SSC26-FT-34

A Feasibility-Aware Attitude Trajectory Planner for Small Satellites

Patrick McKeen
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139; 336-671-3401
pmckeen@mit.edu

Niclas Scheuer
Department of Mechanical and Process Engineering, ETH Zürich
Rämistrasse 101, 8092 Zürich, Switzerland; +41 78 230 43 50
nscheuer@ethz.ch

Kerri Cahoy
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139
kcahoy@mit.edu

ABSTRACT

Spacecraft attitude control traditionally requires extensive per-mission engineering—hand-tuned gains, mission-specific mode logic, and conservative operating envelopes—that breaks down when on-orbit conditions deviate from ground assumptions. This is especially costly for small satellites, where limited ground support makes autonomous adaptability essential. We present a feasibility-aware attitude trajectory planner that replaces this hand-engineered stack with continuous planning from a digital twin: less a new feedback law than a different division of labor, in which the mode logic, gain tuning, and desaturation a conventional ADCS builds as separate pieces are instead absorbed into a single optimization.

Built on the ALTRO trajectory optimization algorithm, the planner accepts high-level pointing goals and reasons over future time-varying control authority, respecting the full nonlinear dynamics and actuator constraints directly—without collocation or slack variables. Because trajectories are produced by forward integration, every returned trajectory is dynamically consistent with the modeled spacecraft; when a goal cannot be met within the modeled constraints, the planner returns a bounded, best-effort trajectory rather than exceeding actuator authority or diverging. It also emits time-varying LQR gains for closed-loop tracking, providing robustness to unmodeled disturbances, estimation error, and sensor noise without anticipating them at the planning stage.

Optimizing directly against the dynamics, the planner also finds strategies a designer would otherwise have to anticipate: facing a body-fixed disturbance beyond its actuator authority, it autonomously adopted a spinning maneuver that averages the disturbance out over each revolution—a solution it was never directed toward. Monte Carlo simulations (100 runs, 1000s, randomized orientation, goal, rate, and orbit, under disturbances excluded from the planner's model) show 84% of slews converging (<5 degrees, mean 3.8 degrees) for a magnetorquer-only spacecraft versus 27% (mean 21.5 degrees) for PD control, improving to 94% (mean 1.2 degrees) versus 90% (mean 9.1 degrees) with a single reaction wheel added.

The same framework spans magnetorquer-only, hybrid, and full reaction-wheel configurations with minimal reconfiguration, handling momentum management and desaturation implicitly within slews and pointing. Implemented in C++ for embedded-class processors, it replans continuously on overlapping horizons—bringing trajectory-planned autonomy to spacecraft historically limited to reactive control.

I. INTRODUCTION

Pointing in the correct direction is critical for many satellites, whether that means pointing an imager at a target (and not at the sun!), a radio at a ground station, or solar panels towards the sun. Current methods of ADCS rely on separate modes, such as slewing, pointing, and momentum desaturation [3–6], with pre-specified transitions, mission- and hardware-specific gain tuning, and, in the worst case, ground-planned maneuvers that are then uplinked to the satellite. This is fragile—mode transitions must be anticipated, gains must be tuned per satellite/actuator/disturbance, and off-nominal conditions require ground intervention. This paper comes from a different direction: what if the spacecraft could plan its own attitude trajectory continuously, from a digital twin of itself, constantly balancing between its goals? This work builds on the authors' generalized small-satellite ADCS framework [9] and prior trajectory-optimization attitude-control methods [7, 8, 13].

A planner-based ADCS, as laid out in this work, changes not just performance but CONOPS: operators specify goals and times, the planner handles everything else. This simplifies operations and pre-mission ADCS development, but has other opportunities as well: planning with foresight over time-varying control authority enables strategies that instantaneous feedback cannot find. If the planner always generates feasible trajectories, a simpler control law (e.g., TVLQR) can track them, correcting for any residual model mismatch or disturbances and making the system robust.

In Section II, we lay out a feasibility-aware attitude trajectory planner based on ALTRO [1, 2], though the formulation and operational architecture are largely solver-agnostic (Section II-B). Section III describes how to use such a planner onboard on embedded systems in real-time operations, including questions of scheduling, following, and fallback methods. The results (Section IV) make the case that this is less a new controller than a different division of labor: jobs a conventional ADCS hand-engineers as separate pieces are absorbed instead by the single optimization. It unifies pointing, slewing, and momentum management with no mode-switching; degrades gracefully toward the achievable when a goal is infeasible; discovers control strategies—an emergent spin, a pre-positioning slew—with no human input; and generalizes across actuator configurations with no per-satellite gain tuning or control-law redesign. Discussion of CONOPS implications, objective function adjustments, computational performance, and limitations can be found in Section V.

II. PLANNER DESIGN

II-A. Problem Formulation

The planner formulates attitude control as a constrained trajectory optimization problem over a finite time horizon. The state vector is

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\omega}^\top & \mathbf{q}^\top & \mathbf{h}_w^\top \end{bmatrix}^\top \tag{1}$$

where $\boldsymbol{\omega} \in \mathbb{R}^3$ is the body angular velocity, $\mathbf{q} \in \mathbb{R}^4$ is the attitude quaternion, and $\mathbf{h}_w \in \mathbb{R}^{n_w}$ is the stored angular momentum in the reaction wheels (omitted for magnetorquer-only configurations). The control vector $\mathbf{u}$ contains commanded inputs to all actuators—magnetorquer dipole moments and reaction wheel torques.

The dynamics are propagated using a fourth-order Runge-Kutta integrator with fully analytical Jacobians and Hessians, providing exact gradient information to the optimizer. The equations of motion include rigid-body attitude dynamics, reaction wheel dynamics, and environmental models for the geomagnetic field and disturbance torques. The planner operates on a digital twin of the spacecraft: an internal model whose initial state comes from the onboard estimator, whose disturbance torques and actuator biases reflect current estimates, and whose physical parameters (inertia, actuator geometry) are loaded from the spacecraft model.

A key property of this formulation is that trajectories are generated through direct forward integration of the dynamics—there are no collocation schemes or slack variables. It is therefore worth separating three notions that "convergence" tends to conflate. A trajectory is *dynamically feasible* if it is a valid rollout of the true equations of motion; because the planner integrates the dynamics directly, this holds at every iteration, not only at the end. It is *constraint-satisfying* if the actuator, rate, momentum, and keep-out limits are met to tolerance, and *goal-converged* if the pointing objective is met to tolerance—both of which the augmented-Lagrangian iteration drives toward but guarantees only at convergence. A converged solve is all three; an incomplete solve still returns a dynamically feasible trajectory, but one whose constraints or goal may not yet be met. Constraints are enforced within the optimization:

- **Actuator bounds**: commanded outputs are limited to hardware maxima (dipole moment for magnetorquers, torque for reaction wheels).
- **Angular velocity limits**: bounding the spacecraft rate to protect hardware and maintain sensor validity.
- **Momentum storage limits**: reaction wheel speeds are constrained to prevent saturation.
- **Stiction avoidance**: reaction wheels are prevented from dwelling at zero speed, where static friction causes unpredictable torque response.
- **Keep-out zones**: instrument boresights can be prevented from crossing exclusion regions (e.g., solar avoidance for cameras).

II-B. ALTRO for Attitude Planning

The optimization is solved using SALTRO (satellite ALTRO), our modified version of the Augmented Lagrangian Trajectory Optimizer (ALTRO) [1], with quaternion-aware state reduction following Manchester et al. [2]. ALTRO solves constrained trajectory optimization through two nested loops (Figure 1). The inner loop is an iterative LQR (iLQR) solver: a backward pass computes feedback gains by quadratically approximating the

cost-to-go, and a forward pass rolls out the nonlinear dynamics under these gains with a line search. The outer loop is an augmented Lagrangian method that enforces constraints through multiplier and penalty terms, tightening enforcement each iteration. The backward pass operates on a reduced three-dimensional attitude error state via the quaternion kinematics matrix $W(q_k)$ [2], while the forward pass integrates the full nonlinear quaternion dynamics. The Cayley map is used by default to compute the three-parameter attitude error for feedback, though the implementation supports several alternatives (Modified Rodrigues Parameters, quaternion vector part).

This structure suits the attitude planning problem well. iLQR operates directly on the nonlinear dynamics without collocation, the augmented Lagrangian handles mixed inequality constraints without an interior-point solver, and the time-varying gains K_k produced by the backward pass can be reused for closed-loop tracking (Section III).

Much of what follows is independent of this particular solver. The attitude-planning formulation (reduced- and full-attitude goal handling, the cost structure with its momentum, stiction, and keep-out terms, and goal scheduling) and the operational architecture (digital-twin planning, overlapping-horizon replanning, the fallback hierarchy, and the plan-as-reference-with-a-tracker pattern of Section III) are solver-agnostic and could be built on other trajectory optimizers. Two of the advantages above, however, are specific to the differential-dynamic-programming family: dynamic feasibility by forward rollout (a shooting property—a collocation or general-NLP solver yields a trajectory feasible only at convergence) and the tracking gains K_k obtained as a byproduct of the backward pass (which a general NLP would require a separate Riccati recursion to produce). We adopt ALTRO because it delivers both of these together while remaining fast enough for embedded, real-time replanning; a practitioner preferring a different optimizer could reuse the formulation and architecture and recover tracking gains separately.

Our implementation makes several practical modifications to standard ALTRO; collectively, these define SALTRO:

- **Per-constraint penalty scaling (optional):** Rather than a single scalar penalty, the implementation can maintain per-constraint, per-timestep multipliers and penalties $(\lambda_{i,k}, \mu_{i,k})$, allowing tight enforcement of active constraints without over-penalizing those already satisfied.
- **Constraint margin in lieu of projected Newton:** ALTRO's projected Newton step for final constraint polishing is omitted. Instead, constraints are formulated with built-in margin—actuator limits are set to approximately 75–80% of hardware maxima. This simplifies the solver while reserving control authority for the tracking controller to handle disturbances and model mismatch at execution time.
- **Initialization strategies:** The optimizer requires a dynamically feasible initial trajectory. Several options are supported, including a gain-scheduled magnetic feedback law that accounts for the target attitude and current pointing error, a B-dot or PD control rollout, and random perturbation of controls.
- **Two-pass time refinement:** An optional two-pass strategy first solves at a coarse timestep with relaxed tolerances, then upsamples and re-optimizes at the finer resolution needed for tracking gain computation.
- **Homotopy spike removal:** If the trajectory has a wrap around 360°, the optimizer can struggle to remove it, as "unwrapping" the spike requires making the rest of the trajectory temporarily worse. This is a manifestation of the quaternion double cover: q and $-q$ represent the same physical attitude, so the optimizer can settle into a path that winds through a full revolution rather than the shorter, homotopically distinct path to the same orientation. Our approach detects spikes as they begin to appear (a smooth, good trajectory, then a sudden spike above 90°, then a return to the prior trajectory) and replaces that segment with a PD-based control toward the spike-exit state. This disrupts the remainder of the trajectory, but the optimizer quickly repairs the disruption.

II-C. Cost Function

The stage cost decomposes into attitude error, angular velocity, control effort, and (when reaction wheels are present) momentum management terms:

$$\ell_k(x_k, u_k) = \ell_q(x_k) + \tfrac{1}{2} c_\omega\, \omega_k^\top \omega_k + \tfrac{1}{2} c_u\, u_k^\top W u_k + \ell_{\mathrm{RW}}(x_k) \tag{2}$$

A separate set of terminal weights is applied at the final timestep, which excludes the control term. All cost Jacobians and Hessians are computed analytically.

The attitude error cost ℓ_q depends on the goal specification. A *full-attitude* goal specifies a target quaternion q_{goal}, with error based on the quaternion inner product $|q_{\mathrm{goal}} \cdot q_k|$. A *reduced-attitude* goal specifies only the

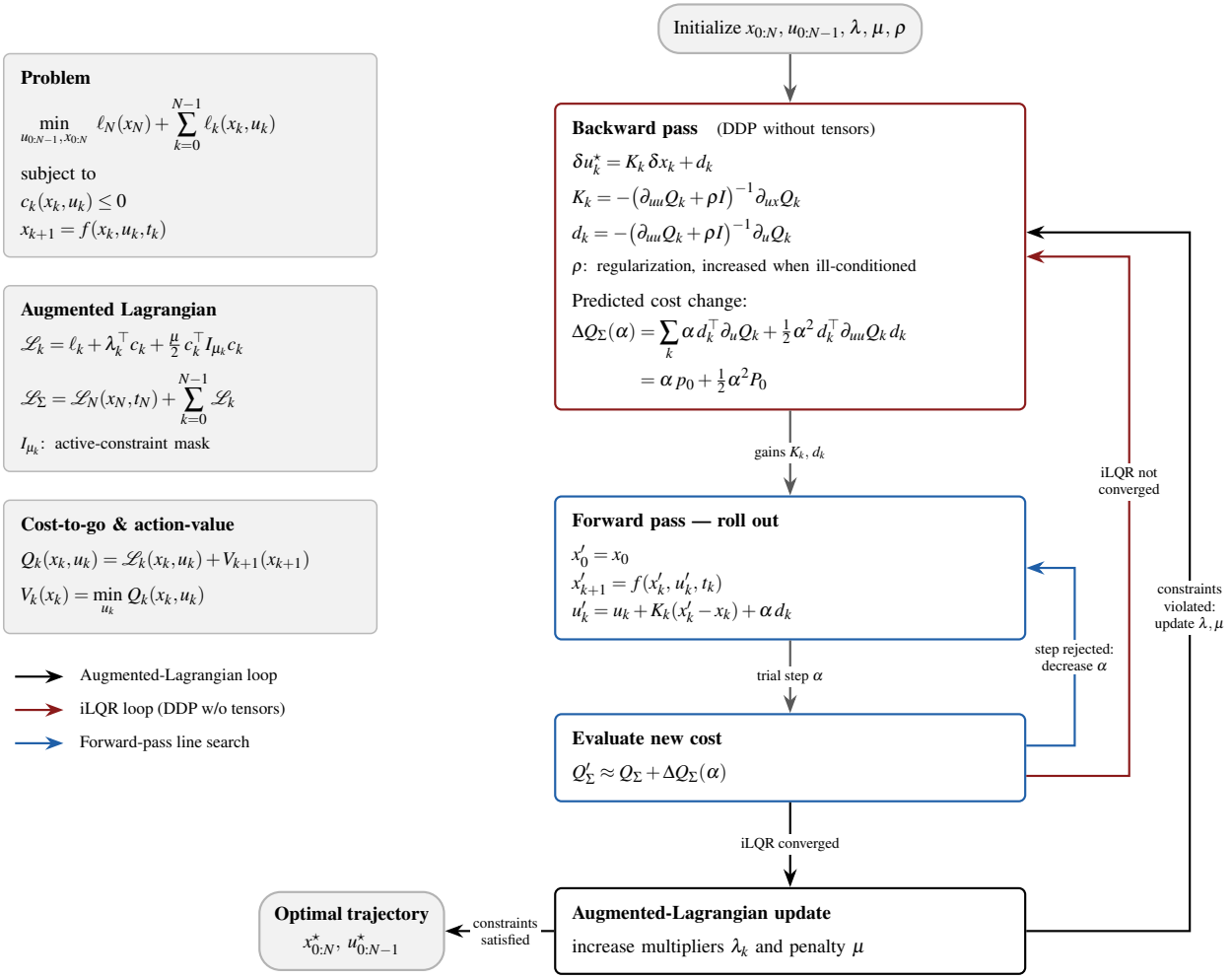


Figure 1: The ALTRO solver. The outer augmented-Lagrangian loop (black) enforces actuator, rate, momentum, stiction, and keep-out constraints through multipliers and penalties that tighten each iteration; the inner iLQR loop (red) computes a backward pass over a reduced three-parameter attitude error state (Cayley map by default) to produce time-varying feedback gains K_k, and the forward-pass line search (blue) rolls the full nonlinear quaternion dynamics forward under those gains, reducing the step α until the cost decreases. The gains K_k are reused at runtime by the TVLQR tracker (Section III-C). Left: the underlying problem, augmented Lagrangian, and cost-to-go definitions.

alignment of a body axis $\hat{s}$ with an inertial direction $\hat{e}$, with error based on $\hat{s} \cdot \mathrm{rot}(q_k)^\top \hat{e}$, leaving one rotational degree of freedom unconstrained [7]. Several functional forms of these error metrics are supported; the choice affects convergence behavior and is part of ongoing tuning work. The control weight matrix W is diagonal with per-actuator weights scaled inversely with the square of each actuator's maximum output.

Goals can vary over the trajectory, and the attitude error weight can be set to zero for selected intervals (see Section IV-C). During these intervals, the planner is free to use the available time for whatever is most useful given the remaining cost terms—slewing toward a future target, desaturating reaction wheels, or repositioning to prepare for an upcoming goal. This flexibility is particularly valuable for multi-target operations where transition periods between imaging passes would otherwise be wasted.

When reaction wheels are present, two soft penalty terms manage stored momentum, following the formulation introduced in [8]: a saturation cost (a shifted softplus) that penalizes stored momentum above a threshold, and a stiction-avoidance cost (a smoothstep) that discourages dwelling at zero wheel speed; we refer the reader to [8] for the exact functional forms. The stiction term is kept small, so it discourages lingering near zero without preventing the wheel from crossing through zero when necessary. In its current smoothstep form it introduces regions of negative curvature in the cost—which the optimizer tolerates but which we have not fully resolved; a better-conditioned stiction penalty remains an open item. Both terms complement the hard constraints in Section II-A.

A practical challenge with this cost structure is that terms can work against each other. For example, a high angular velocity weight discourages the fast slewing needed to reach a pointing target, while a high pointing weight may drive the optimizer to command aggressive maneuvers that violate rate preferences. The relative weighting must be chosen with the primary mission objective in mind, and hard constraints (such as angular velocity limits) can be used to enforce bounds that would otherwise require delicate cost tuning.

II-D. Feasibility and Graceful Degradation

A key property of the trajectory optimization approach is that infeasible goals do not cause failure. When the requested attitude cannot be reached exactly—due to underactuation, limited torque authority, or insufficient time—the optimizer converges to the lowest-cost behavior it can find as measured by the cost function. Because ALTRO minimizes a weighted objective subject to constraints on actuators, angular velocity, and keep-out zones, a converged trajectory respects all of these limits regardless of how ambitious the goal is. The planner does not saturate actuators indefinitely, does not induce unbounded angular rates, and does not need to be told that a goal is infeasible. It simply returns the best trajectory it can find.

This stands in contrast to feedback controllers, which respond to a persistent error signal with persistent actuation. A PD controller commanding a physically unreachable attitude will drive actuators to their limits for the duration of the attempt, potentially building momentum, violating rate constraints, or exciting undesirable dynamics. The planner, by construction, produces a trajectory that is dynamically feasible and constraint-satisfying even when the goal is not fully met.

This property is a practical enabler for operations. Operators and autonomous goal-selection systems can request ambitious targets—tight pointing during a brief pass, rapid multi-target transitions, or orientations that strain the available control authority—without risk of destabilizing the spacecraft. The worst outcome is that the planner achieves less pointing accuracy than requested, and it does so in a bounded, predictable manner.

II-E. Reduced Attitude Goals

Many spacecraft pointing tasks require only that a body axis (e.g., a camera boresight) be aligned with an inertial direction, rather than specifying the full three-axis orientation. These *reduced-attitude* goals leave one rotational degree of freedom—rotation about the aligned axis—unconstrained. The set of quaternions satisfying such a goal can be described analytically [7].

This distinction matters for the planner because the unconstrained axis expands the feasible solution space. Rather than searching for a trajectory to a single target quaternion, the optimizer can converge to any orientation on a continuous set of equally valid solutions. This is particularly beneficial for underactuated systems: a magnetorquer-only satellite that struggles to reach a specific full-attitude target may have many reduced-attitude solutions accessible from its current state. The extra degree of freedom also allows the optimizer to exploit the free axis for purposes beyond pointing—such as minimizing momentum buildup, finding faster slew paths, or accommodating disturbance torques—since the cost function does not penalize rotation about the boresight. For the magnetorquer-only case this relaxation and the planner's foresight are *multiplicative* rather than additive: the free axis helps only because the planner can

place it where the time-varying field supplies the needed authority over the horizon. Neither the reduced-attitude relaxation nor the field's time variation alone makes underactuated pointing feasible—the underactuated convergence results of Section IV rest on both at once.

In practice, this can lead to emergent strategies that were not explicitly designed. For example, when subject to a persistent off-axis disturbance torque, the planner has discovered spinning maneuvers about the pointing axis [13] that average out the disturbance while maintaining boresight alignment—a strategy that was not prescribed but arises naturally from the optimizer exploiting the available freedom. Results demonstrating this behavior are presented in Section IV-D.1.

III. REAL-TIME IMPLEMENTATION

The planner is not a standalone algorithm—it runs continuously as part of a larger ADCS (Figure 2), alongside estimation, control, and actuator management. This section addresses the practical questions that arise from that integration: how to account for the time the planner takes to solve, how to hand off between successive trajectories, how to follow a planned trajectory under real-world disturbances, and what to do when the planner fails to converge.

III-A. Compute Scheduling and Start-State Prediction

The planner requires a finite preparation time t_{prec} before a new trajectory can begin, covering orbit propagation, magnetic-field evaluation, disturbance estimation along the planned horizon, and the trajectory solve itself. During this window the spacecraft continues to move, so the planner cannot always use the current state as its initial condition—it must predict where the spacecraft will be when the new trajectory begins.

After the first trajectory, this prediction is straightforward: the planner reads the state from the current trajectory at the planned handoff time. Because the TVLQR or MPC tracker keeps the spacecraft close to the planned trajectory (Section III-C), the predicted start state is typically accurate to within the tracker's steady-state error. For the very first trajectory after startup—when no prior plan exists—the planner propagates the estimated state forward under a simple control law (e.g., B-dot or PD) for the duration t_{prec}.

The precalculation step also incorporates the most recent state estimates, including estimated disturbance torques and, where the estimator tracks them, actuator biases [10]. This means each replanning cycle uses updated environmental knowledge, so slowly varying errors (e.g., a changing residual magnetic dipole or a growing actuator bias) are naturally incorporated without explicit fault handling. One limit bounds what this feedforward can correct: a constant body-fixed torque that the wheels absorb into a constant momentum offset is unobservable from attitude alone—only the momentum *rate*, revealed as the orbital geometry changes, distinguishes it from an actuator bias. Slowly varying or attitude-coupled disturbances are therefore recovered, while a perfectly constant wheel-absorbed leak is held by the wheels until momentum management acts.

III-B. Overlapping Trajectories

Each trajectory extends beyond its nominal endpoint by an overlap duration t_{overlap}, so that a trajectory covering time t to $t+T$ is actually planned through $t+T+t_{\text{overlap}}$. The next trajectory begins at $t+T$ and is planned through $t+2T+t_{\text{overlap}}$, and so on. This overlap serves two purposes.

First, it provides a time buffer. If the next solve runs long or fails to converge, the spacecraft can continue following the overlap portion of the current trajectory rather than running out of plan. A typical overlap of one-third to one-half the trajectory duration provides substantial margin—for a 500-second trajectory, this is 170–250 seconds of buffer.

Second, it avoids trajectory endpoint effects. Near the end of its planning horizon, the planner has less future cost to optimize over, which can produce artifacts: angular velocities that drop to zero prematurely (because there are no future goals to justify maintaining them), TVLQR gains that shrink as the remaining cost-to-go decreases, and generally less well-conditioned behavior. By always transitioning to the next trajectory well before the current one ends, these endpoint effects never reach the executed portion of the plan.

Larger overlaps (e.g., 75% or more) would increase responsiveness but also increase computational cost and can introduce trajectory chatter—the planner oscillating between near-equivalent solutions (e.g., going above vs. below a keep-out zone) on successive replanning cycles due to small differences in initialization. With overlaps of one-third to one-half, each segment of the trajectory is only replanned once or twice, which keeps this effect manageable.

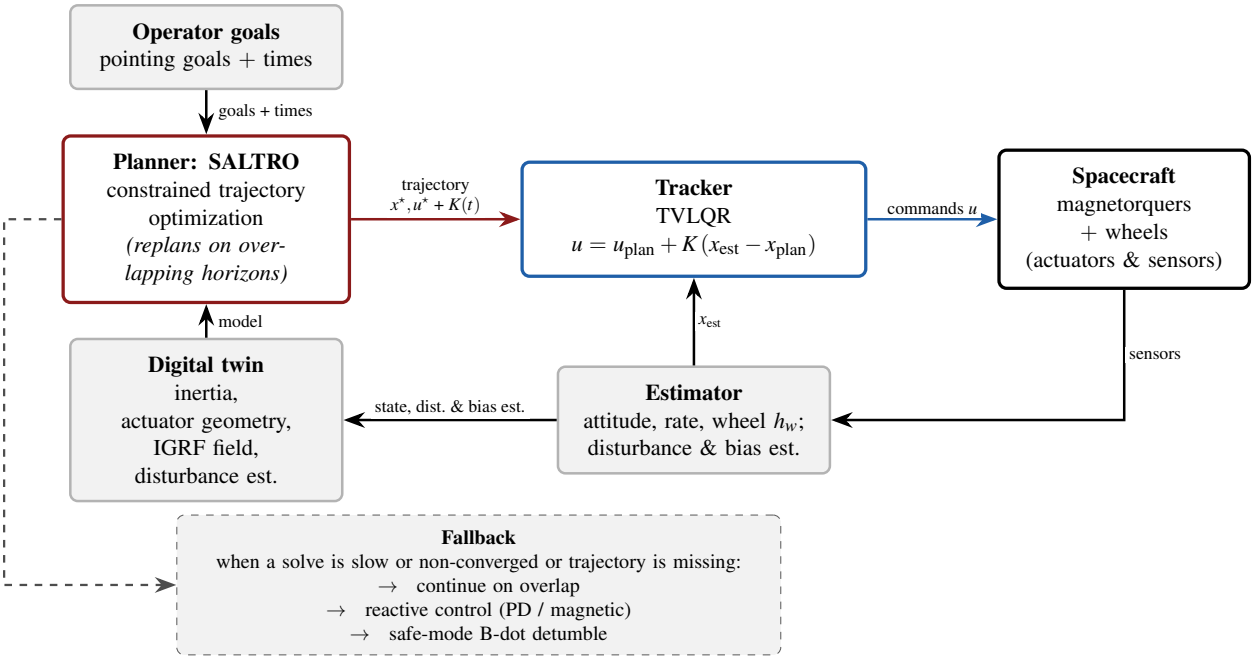


Figure 2: Operational architecture: the planner runs inside a continuously replanning ADCS loop, not as a one-shot maneuver generator. The operator supplies a goal timeline; the estimator and digital twin supply the spacecraft's current state and model; SALTRO solves a constrained trajectory and emits both the reference $(x^\star, u^\star)$ and the tracking gains $K(t)$; the tracker (e.g. TVLQR, Section III-C) follows it using the live estimate; and sensor feedback closes the loop into the estimator and twin, which the next replan consumes. Planning runs on overlapping horizons (Section III-B); a slow or non-converging solve drops to the fallback hierarchy (Section III-D) rather than leaving the spacecraft uncommanded. Unlike a mode-switching ADCS, there are no discrete slew/point/desaturate modes—one optimization produces all of these behaviors.

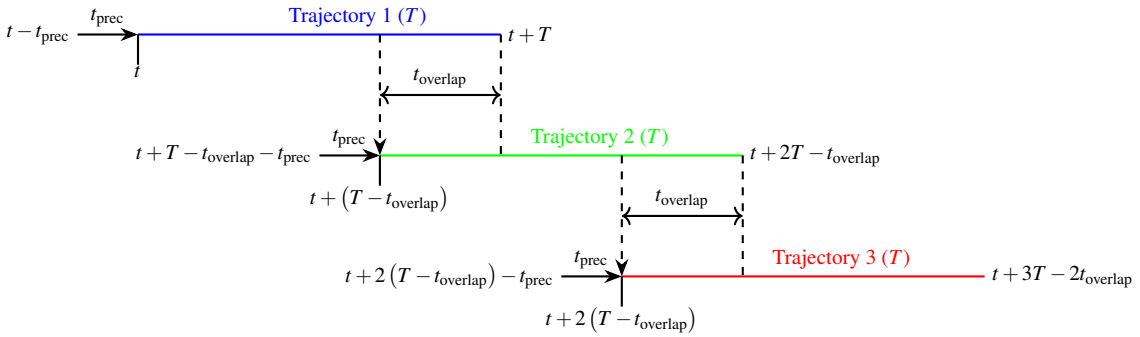


Figure 3: Diagram showing timing of trajectories, including trajectory duration, overlap between trajectories, and precalculation time.

III-C. Trajectory Following

A planned trajectory is only useful if the spacecraft can follow it under real-world conditions—sensor noise, estimation error, unmodeled disturbances, and actuator imperfections. Two trajectory-following approaches are implemented: a time-varying LQR (TVLQR) controller and a single-step model predictive controller (MPC).

III-C.1 Time-Varying LQR

The TVLQR controller uses gain matrices K_k computed along the planned trajectory during the SALTRO solve (as part of the iLQR inner loop); reusing these planner-derived gains for closed-loop tracking follows the trajectory-optimization attitude-control approach of [1, 2]. At each timestep, the control command is

$$\mathbf{u}_k = \mathbf{u}_{k,\text{plan}} + K_k \left(\mathbf{x}_{k,\text{est}} - \mathbf{x}_{k,\text{plan}}\right) \tag{3}$$

where $\mathbf{u}_{k,\text{plan}}$ and $\mathbf{x}_{k,\text{plan}}$ are the planned control and state, and $\mathbf{x}_{k,\text{est}}$ is the estimated state from the onboard filter. The state difference uses appropriate representations for each component (e.g., quaternion error rather than arithmetic subtraction for the attitude).

This is computationally inexpensive—a single matrix-vector multiply per timestep—and provides closed-loop robustness to moderate disturbances and estimation errors. An extended form includes estimated disturbance torques and residual dipole in the state vector, so that the TVLQR gains can compensate for differences between planned and actual disturbance environments:

$$\mathbf{u}_k = \mathbf{u}_{k,\text{plan}} + K_k \left(\begin{bmatrix} \mathbf{x}_{k,\text{plan}} \\ \tau_{k,\text{exp}} \\ \mathbf{m}_{k,\text{exp}} \end{bmatrix} - \begin{bmatrix} \mathbf{x}_{k,\text{est}} \\ \tau_{k,\text{est}} \\ \mathbf{m}_{k,\text{est}} \end{bmatrix} \right) \tag{4}$$

where τ and $\mathbf{m}$ are the disturbance torque and residual magnetic dipole, with subscripts denoting the values the planner expected versus the values the estimator currently reports. This disturbance-augmented variant is included as a direction for rejecting structured, slowly varying disturbances; the campaigns in this paper use the standard TVLQR above, and a thorough evaluation of the augmented form is left to future work.

III-C.2 Single-Step MPC

A single-step model predictive controller is implemented as an alternative tracker. Rather than a full multi-step MPC (prohibitively expensive per control cycle), it optimizes one control step ahead against the full nonlinear dynamics:

$$\min_{\mathbf{u}_k \in \mathscr{U}} \quad \frac{1}{2} \left\| K_{k+1} \left(\text{RK4}(\mathbf{x}_{k,\text{est}}, \mathbf{u}_k, t_k) - \mathbf{x}_{k+1,\text{plan}} \right) \right\|^2 \tag{5}$$

where $\text{RK4}(\cdot)$ propagates the estimated state forward one timestep through the full nonlinear dynamics (including disturbances), $\mathscr{U}$ is the feasible set of actuator commands, and K_{k+1} is the TVLQR gain matrix from the next timestep, used as a weighting that reflects the planner's assessment of which state errors are most costly. This avoids introducing separate tuning parameters for the tracker and naturally incorporates the trajectory's cost structure.

Because it uses the full nonlinear model, it can respect actuator bounds directly rather than clipping a linear command, at the cost of one small optimization per control cycle.

In a 100-trial head-to-head on the 3+1 reduced testbed ($t_f = 1000$ s, $\mathrm{d}t = 1$ s), the single-step MPC did not outperform TVLQR in its current formulation: TVLQR converged in every trial to 0.12° mean final error, while the bounded-QP MPC converged in 62% of trials to 13° mean, at higher control effort. (This tracker head-to-head is a separate campaign from the 2×2 matrix of §IV-B, so its 3+1 reduced figures differ slightly from Table 2.) We therefore use TVLQR as the default and retain the MPC only for hard-saturation regimes (not exercised in this nominal campaign), where re-optimizing against actuator bounds rather than clipping a linear command may help. We do not have a confirmed explanation for the current gap; improving the MPC formulation is left to future work.

III-D. Fallback Strategies

The planner is not guaranteed to *converge*—to meet the goal and satisfy every constraint to tolerance—for every scenario; the returned trajectory is always dynamically feasible, but convergence is what adds constraint satisfaction and goal accuracy. Near-singular magnetic field geometry, very large required slews with minimal actuation, or poor initial guesses can cause the SALTRO solver to exhaust its iteration budget before converging. The ADCS architecture must handle this gracefully.

The fallback hierarchy is:

1. **Continue on overlap.** If the current trajectory has remaining overlap, the spacecraft continues following it while the planner retries. This is the most common fallback and is invisible to the operator.
2. **Reactive control.** If the overlap expires without

a new trajectory, the spacecraft reverts to a reactive controller (PD or magnetic feedback) using the current goal. Performance degrades compared to planned operation, but the spacecraft remains controlled.

3. **Safe mode.** Under adverse conditions (e.g., high angular rates, repeated non-convergence), the system falls back to B-dot detumbling until conditions improve enough to attempt planning again.

In practice, non-convergence is infrequent—the combination of coarse-to-fine timestep refinement and the augmented Lagrangian formulation in ALTRO handles most scenarios. When non-convergence does occur, it is more often a sign that the problem is genuinely difficult (near the boundary of feasibility) than that the solver has failed arbitrarily. Distinguishing between a struggling solve and a truly infeasible problem is an open area: certain patterns during the solve—persistent oscillations in the cost function, unresolved high penalty terms, or sudden trajectory jumps in an otherwise smooth iterate—are informative but not yet automated as formal diagnostics. Developing reliable mid-solve convergence indicators, possibly using the cost-to-go trajectory or constraint violation trends, would allow the system to abandon unpromising solves earlier and retry with adjusted parameters.

IV. RESULTS

IV-A. Simulation Setup

Unless a scenario states otherwise, the campaigns below use the spacecraft and environment of Table 1. The emergent-spin scenario (§IV-D.1) and the cold-gas-leak scenario (§IV-E) use a different spacecraft and orbit; the twin-sensitivity sweep (§IV-G) keeps the nominal platform but deliberately perturbs the digital twin. All such changes are noted in place.

The validation campaign (§IV-B) randomizes, per trial and with seeds paired across planner and baseline: the initial attitude (uniform on $SO(3)$), the initial body rate (a uniformly random axis at 0.1–1.0°/s), the pointing target (a random inertial direction for reduced-attitude pointing; a random orientation for full attitude), and the circular orbit (random inclination, ascending node, and phase at 7000 km radius); reaction wheels start at rest. The sequential-multigoal campaign (§IV-C) draws three independent random ECI directions per trial; the hyperparameter sweep (§IV-G) fixes the scenario and varies one knob at a time.

Table 1: Simulation parameters for the planner campaigns. Scenario-specific parameter changes (the spin demo, the cold-gas-leak scenario, and the twin-sensitivity sweep) are detailed in the relevant subsections.

Quantity	Value
Spacecraft	
Inertia J (nominal)	$\mathrm{diag}(0.0314, 0.0341, 0.0100)$ kg·m^2 (BeaverCube; spin demo: $\mathrm{diag}(0.10, 0.05, 0.005)$)
Magnetorquers	3 orthogonal; max dipole 0.2 A·m^2
Reaction wheels	3+1: one on $+\hat{y}$; 3+3: on $\hat{x}, \hat{y}, \hat{z}$; max torque 2.3 mN·m, max momentum 3.6 mN·m·s (spin: $h_{\max}$=2)
Orbit & environment	
Orbit (MC)	circular, 7000 km orbital radius, random incl./RAAN/phase (spin: 429 km altitude, 51.5°)
Field model (planner)	nominal campaigns plan on the plant's sampled IGRF-13 field (planner field ≡ plant field, no mismatch)
Field model (plant)	IGRF-13 (`ppigrf`) with full Skyfield Earth-orientation frames
Disturbances (MC)	gravity-gradient ($\sim 10^{-8}$ N·m, peak 4.3×10^{-8}), in the plant and excluded from the planner; drag and SRP enabled but zero-torque (no cp–cg offset)
Solver & tracker	
Integration	RK4, analytical Jacobians/Hessians
Actuator margin	75–80% of hardware maxima reserved for the tracker
Control rate Δt	1 s
Tracker	TVLQR (default)
Convergence threshold	5° final pointing error

IV-B. Planner vs. Classical Feedback

A conventional ADCS fields a different, separately tuned controller for each actuator suite; here the same planner—one formulation, one cost structure—is pointed at each platform. We evaluate it against classical feedback control over a 2×2 matrix of actuator configurations and pointing tasks. The two configurations are a magnetorquer-only platform (three orthogonal magnetorquers, denoted 3+0, equivalently 3MTQ+0RW) and the same platform augmented with a single reaction wheel (3+1, i.e. 3MTQ+1RW). The two tasks are *reduced-attitude* pointing, in which only a body boresight is aligned to an inertial target direction, and *full-attitude* pointing, in which a complete target orientation is tracked. Each platform thus has a *controllability-appropriate* task—reduced-attitude for

Table 2: Planner versus configuration-appropriate classical feedback, across actuator configurations and pointing tasks (100 Monte Carlo trials per cell, 1000 s horizon; convergence defined as final pointing error $< 5°$). Daggers (†) mark each configuration's controllability-appropriate task. The classical baseline is Lovera's magnetic PD for 3+0 and an LP-allocated PD for 3+1.

		Planner		Classical PD	
Config	**Task**	Conv.	Mean err.	Conv.	Mean err.
3+0	Reduced†	84.0%	3.85°	27.0%	21.54°
	Full	18.0%	18.31°	0.0%	85.17°
3+1	Reduced	99.0%	0.26°	97.0%	1.58°
	Full†	94.0%	1.19°	90.0%	9.06°

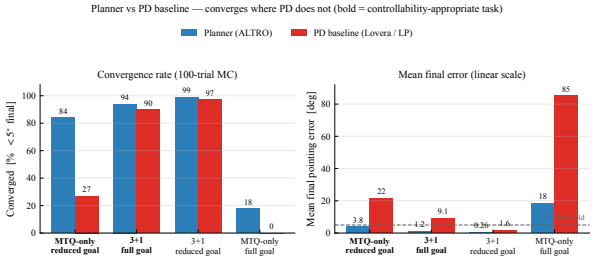


Figure 4: On each platform's controllability-appropriate task, the planner converges where classical feedback does not. Planner vs. PD baseline across the 2×2 matrix of Table 2 (bold marks the controllability-appropriate task): MTQ-only reduced pointing, 84% vs. 27%; 3+1 full attitude, 94% vs. 90% but at 1.2° vs. 9.1° mean error and a far lighter tail. Left: convergence rate; right: mean final error.

the magnetorquer-only 3+0, full-attitude for the wheel-augmented 3+1—and a harder, *mismatched* task (full attitude for the magnetorquer-only platform, the easier reduced-attitude goal for the wheel-augmented one). For each of the four cells we run 100 Monte Carlo trials over a 1000 s horizon, with the randomization and paired-seed protocol of §IV-A. We compare the feasibility-aware planner against the classical feedback law appropriate to each configuration—Lovera's magnetic PD [3] for the magnetorquer-only platform and an LP-allocated PD for the reaction-wheel platform. A trial is counted as converged if its final-timestep pointing error falls below 5°; Table 2 reports convergence rate and mean final error, and Figure 4 shows them graphically.

On each configuration's controllability-appropriate task—reduced-attitude pointing for the magnetorquer-only platform, full-attitude pointing for the reaction-wheel platform—the planner converges where classical feedback struggles. For the magnetorquer-only platform the planner aligns the boresight in 84% of trials against 27% for the magnetic PD, a threefold improvement, and reduces mean final error from 21.5° to 3.8°. The residual non-convergences split into two populations: trajectories still slewing at the 1000 s cutoff, and trajectories whose terminal residual requires a torque near the instantaneous field direction—the magnetorquer-only hold limit—which drift slightly rather than holding cleanly—a degraded terminal hold rather than a failure to acquire. For the reaction-wheel platform both controllers converge on most trials (94% versus 90%), but the planner holds a mean error of 1.2° against the PD's 9.1°—a 7.6× reduction. The gap is largest precisely where the actuation is hardest: a magnetorquer can produce torque only in the plane perpendicular to the instantaneous geomagnetic field, so feedback laws that command an instantaneously infeasible torque waste control authority, whereas the planner accounts for the time-varying feasibility envelope when it constructs the trajectory.

The mismatched pairings show the planner cannot pass controllability limits. Asked to perform reduced-attitude pointing, the over-capable reaction-wheel platform is near-perfect under both controllers (99% planner, 97% PD): when the task is well within the actuator envelope, the planner's advantage narrows because little authority is wasted to begin with. The harder mismatch is the informative one. Asked to perform *full*-attitude pointing with magnetorquers alone, even the planner converges in only 18% of trials (the PD law in 0%). This is not a planner failure but a hardware limit: a magnetorquer-only platform is not robustly controllable for full three-axis attitude, because no magnetorquer command can produce torque about the field direction (as established by the companion controllability analysis [12]). The planner extracts the performance the actuator geometry permits, but it cannot manufacture controllability the hardware does not have. The two controllability-appropriate pairings are therefore the headline comparison: each configuration is evaluated on the task its actuators are suited to, and on that task the planner's feasibility-awareness translates directly into higher convergence and markedly tighter pointing.

IV-C. Sequential Multi-Goal Operation

A feasibility-aware planner makes operational sense only if it handles routine sequencing of mission goals as cleanly as it handles single-goal acquisition. We probe this with the simplest case: chaining *several* terminal states inside a single optimization— point at A, hold,

Table 3: Sequential multi-goal results ($N = 100$ trials per configuration). *Held%* is the sustained-acquisition metric (continuous $< 5^\circ$ for ≥ 100 s within the window); the *(old)* column is the previous final-at-window-end definition, shown for contrast. t_{acq}: median time to first acquisition; SS: steady-state error over the held interval.

Config	Goal	Acq.%	Held%	(old)	t_{acq} [s]	SS [°]
3+1	A	96	77	10	232	1.28
3+1	B	99	99	14	48	0.52
3+1	C	100	100	98	37	0.29
3+3	A	100	100	22	156	0.43
3+3	B	100	100	25	19	0.31
3+3	C	100	100	100	23	0.17

slew to B, hold, slew to C. A conventional ADCS would sequence three pointing modes with engineered transitions; the planner instead receives one goal timeline and produces one trajectory through it.

The timeline is symmetric: goal A active over $[0, 500]$ s, no-goal coast over $[500, 700]$ s, goal B active over $[700, 1200]$ s, no-goal coast over $[1200, 1400]$ s, goal C active over $[1400, 1900]$ s. Each of A, B, C is an independent random ECI *pointing direction* (a reduced-attitude, 2-DOF goal; there are no full-attitude or quaternion targets), paired across configurations. We run 100 trials on each of 3+1 and 3+3, with one plan per trial through all three goals—no per-goal replanning. For each active window we report *acquired*, defined as the minimum pointing error inside the window dropping below 5°, and *held*, redefined here as a *sustained* acquisition—a continuous interval below 5° lasting at least 100 s within the window. We also report the median time to acquire (t_{acq}) and the steady-state error (SS, the mean over that held interval). For contrast the table also lists the previous definition of held—error still below 5° exactly at window end—which, as we will see, badly undersells the result.

The planner acquires every goal on both configurations (3+1 at 96–100%, 3+3 at 100% across the board), and under the redefined held metric it also *holds* them: held rates are 77–100%, the only sub-90% entry being the first interior goal on the more constrained 3+1 platform (77%). This is a sharp correction to the previous final-at-window-end definition, which reported only 10–25% for the interior goals—not because the planner failed to hold them, but because a single-shot plan through all three goals begins slewing toward the next goal before the active window closes, so the error sampled exactly at window-end has already lifted off. Reading that artifact as a hold failure would be reading the experiment backwards; the sustained metric (a continuous 100 s below 5°) credits the acquire-and-hold behavior correctly. The hold is also fast and tight: median time-to-acquire ranges from 19 to 232 s and steady-state error from 0.17° to 1.28°. A configuration gradient remains—the extra wheels on 3+3 acquire each interior goal faster and hold it longer than 3+1, while both nail the terminal goal. Reaction-wheel momentum profiles across the sequence remain bounded on both configurations (Figure 5); the planner manages momentum implicitly through the same optimization that lays down the trajectory.

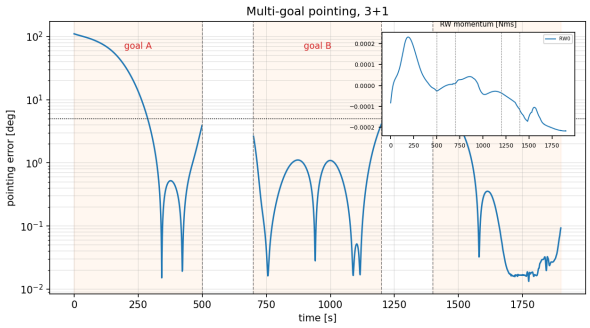


Figure 5: One plan acquires three sequential goals with momentum management folded in. Representative 3+1 trial (aggregates over all 100 in Table 3): pointing error against the active goal on a log axis, goal-active windows shaded. Error dips to near-zero at each goal and lifts as the next slew begins—the apparent "loss" the window-end metric misreads—while the terminal goal C is held to the horizon. Inset: reaction-wheel momentum stays smooth and unsaturated across the sequence.

IV-D. Emergent Control Strategies

Because the planner optimizes a trajectory directly against the dynamics and constraints rather than executing a fixed control law, it sometimes arrives at strategies a designer would otherwise have to anticipate and encode. We show two that emerged without being prescribed: a spin-stabilized terminal state that absorbs a disturbance the actuators cannot directly oppose, and a foresight maneuver that pre-positions a reaction wheel for an upcoming slew.

IV-D.1 Emergent Spin Stabilization Under a Body-Fixed Disturbance

A small body-fixed disturbance—a thruster misalignment during a propulsive maneuver, an offset center of pressure, a constant magnetic dipole—makes static inertial pointing infeasible with finite actuators. Integrated

in the inertial frame, the disturbance accumulates angular momentum that the spacecraft's actuators must absorb until they exhaust their authority, after which attitude is no longer regulated. The classical resolution is *spin stabilization*: rotate the body about the pointing axis so that the body-fixed disturbance averages out in the inertial frame. Conventional mode-switching ADCS implements this with an explicit spin-up controller, a separate pointing-while-spinning controller, momentum-management logic, and engineered handoffs. We show that the planner produces this entire strategy automatically, given only the pointing goal, the dynamics, and the actuator constraints.

This demonstration is motivated by BeaverCube [11], whose off-axis thrust during a propulsive maneuver imposes exactly this kind of body-fixed disturbance; we include a reaction wheel because magnetorquers alone cannot substitute for it in absorbing the demand. The scenario is an oblate spacecraft (principal inertias $\mathrm{diag}(0.10, 0.05, 0.005)$ kg·m^2; oblateness ratio $J_z/J_x \approx 20$, with the boresight along the minor axis $+\hat{z}_b$), three orthogonal magnetorquers, and a single reaction wheel on the $+\hat{y}_b$ body axis with momentum limit $h_{\max} = 2$ mN·m·s. A constant body-fixed disturbance $\tau_d = 0.3$ mN·m along $+\hat{x}_b$, representative of a small propulsion-thrust offset, acts throughout the maneuver. The spacecraft begins at rest 179° from the anti-ram (anti-velocity) target on a 429 km circular orbit at 51.5° inclination. The planner is given a vector-pointing goal and a 500 s horizon; *no spin rate or terminal attitude is specified*.

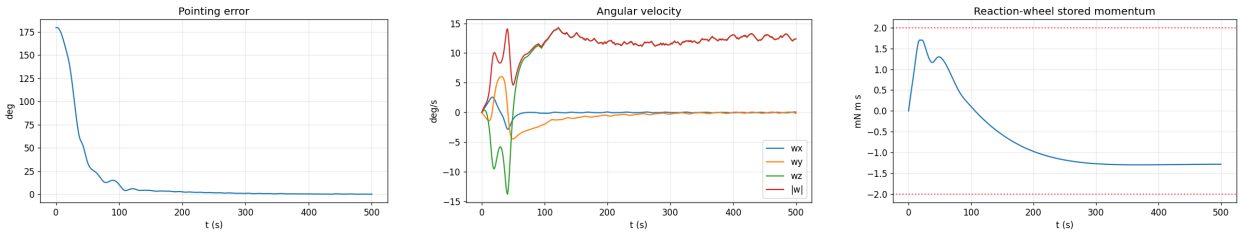
The planner's solution here is an open-loop *plan*—a single optimizer trajectory, shown in Figure 6, not a closed-loop Monte-Carlo campaign like the preceding sections. It acquires pointing within 1° in 151 s (Figure 6a), then settles to a steady-state mean pointing error of 0.12° (steady-state maximum 0.30°). The terminal state is not a static attitude but a steady spin of 13.8°/s about the boresight, with the spin axis aligned with $+\hat{z}_b$ to within 0.1° (Figure 6b). The chosen spin rate has a clean physical interpretation, primarily in terms of *disturbance smearing* rather than gyroscopic stiffness. A torque that is constant in the body frame rotates in the inertial frame at the spin rate Ω; its secular impulse over each rotation vanishes, and what remains is a sinusoidal angular-momentum demand of amplitude $|\tau_d|/\Omega$, which the wheel can absorb. Equivalently, viewed in the body frame, storing a constant wheel momentum $h_w \approx -|\tau_d|/\Omega$ on $+\hat{y}_b$ generates a gyroscopic back-torque $\omega \times h_w \hat{y}_b$ that cancels the disturbance on $+\hat{x}_b$ to leading order, with the magnetorquers absorbing the small residual. Requiring the implied wheel excursion to fit within $h_{\max}$ gives the gyro-balance minimum $\Omega_{\min} = |\tau_d|/h_{\max} = 8.6°/\mathrm{s}$; the planner selects $1.6\,\Omega_{\min}$, with the wheel settling at -1.17 mN·m·s in steady state and reaching the 1.7 mN·m·s margin only during the acquisition transient (Figure 6c). A separate effect—the body's own angular momentum about the spin axis providing gyroscopic stiffness against transient transverse perturbations—is a useful concurrent benefit, but it is the smearing mechanism, not the stiffness, that sets $\Omega_{\min}$.

This terminal mode was not specified. The pointing goal is roll-free, so the planner is free to choose any terminal angular velocity—and static pointing is simply not among the feasible options here. The disturbance is a steady 0.3 mN·m along $+\hat{x}_b$, while the single wheel spins about $+\hat{y}_b$: it is on the *wrong axis* and cannot oppose the disturbance's body-frame secular component at all. Nor would a correctly mounted wheel rescue static pointing—absorbing a constant 0.3 mN·m torque, it would wind up to its $h_{\max} = 2$ mN·m·s limit in under 7 s ($h_{\max}/|\tau_d| \approx 6.7$ s) and saturate. The magnetorquers, meanwhile, can produce at most $\sim 30\,\mu$N·m in the disturbance direction at the orbital field strength, an order of magnitude below the disturbance. Holding the body still therefore accumulates angular momentum the actuators cannot bleed off, and the body begins to tumble within the first minute. The planner has no way to make the static solution work and selects instead from the feasible class—spin-stabilized solutions with $\Omega > \Omega_{\min}$—the member that minimizes control effort. Spin stabilization is not built into the controller; it emerges as the optimization's response to the constraints.

The practical content of this is the framework claim made operational. A conventional ADCS would partition the scenario into modes—initial acquisition, spin-up, pointing while spinning, momentum management, and desaturation—each requiring its own controller and engineered transitions, and the spin-up controller in particular requires a designer to recognize that spin stabilization is the right strategy. The planner collapses all of this into a single optimization. The same pipeline that produces the static-pointing results of Section IV-B produces the spin-stabilized solution here, with no change beyond the disturbance model and initial conditions handed in.

IV-D.2 Foresight: Pre-Positioning a Wheel for a Future Slew

The spin result exploits freedom in the *terminal* state; a second emergent behavior exploits freedom in the *intermediate* state. A vector-pointing goal fixes only a



(a) Pointing error: the anti-ram target (179° offset) is acquired in 151 s and held to a 0.12° mean.

(b) Body rate: a steady, uncommanded 13.8°/s spin about the boresight (ω_z); transverse rates damped to zero.

(c) Wheel momentum: settles to −1.17 mN·m·s, touching the 85% margin (1.7) only during acquisition.

Figure 6: Emergent spin stabilization (single open-loop plan; 3+1, oblate bus, constant body-fixed 0.3 mN·m disturbance on $+\hat{x}_b$). Given only a roll-free anti-ram goal, the planner acquires the target and then holds it not by standing still but by spinning the body about the boresight at 13.8°/s. The spin smears the body-fixed disturbance so its secular impulse vanishes each revolution, leaving a sinusoidal momentum demand the single wheel absorbs at a constant offset. The whole strategy—acquire (a), spin up (b), park the wheel momentum (c)—was not specified; it is the optimizer's response to a disturbance no static-pointing solution can survive.

boresight direction, leaving the roll about that boresight free—and a planner that sees the whole goal timeline can spend that freedom now to make a later maneuver easier. We test this on the 3+1 platform with the single wheel mounted on $+\hat{z}_b$ (boresight $+\hat{y}_b$) and two sequential vector-pointing goals, A then B, separated by 90°, with the goal switching directly from A to B at $t = 700$ s; the upcoming $A \to B$ slew is about the ECI axis $[1,0,0]$.

While holding A, the planner rolls the body about the free boresight, swinging the wheel axis toward the upcoming slew axis—the wheel-axis/slew-axis alignment rises from 0.03 to 0.55 over the A-hold—and pre-loads a small wheel momentum ($|h| = 0.15$ mN·m·s at the $A \to B$ switch). The roll is gentle (peak rate 0.14°/s) and leaves A-pointing undisturbed through the hold; B is then acquired 144 s after the slew begins (Figure 7). Nothing in the goal specification asked for this re-orientation—the planner reorients the wheel *because* it can see that the next slew will need authority about an axis the wheel does not currently serve. Like the spin, it is look-ahead that a mode-based controller would have to be told to perform; here it falls out of planning over the full goal timeline.

IV-E. Incorporating New Information Online

A planner that is feasibility-aware offline is far more capable when it can also take in new information in flight and adapt. Conditions change after a plan is laid down—a disturbance appears, an estimate resolves—and the value of planning from a digital twin is that any newly available information can simply be added to the model, so that the next plan accounts for it. There is no disturbance-detection trigger and no contingency mode to fall into; the estimator updates the twin, and the next scheduled replan plans against it like any other. The same replanning machinery that handles scheduling and multi-goal sequencing (Sections IV-B–IV-C) absorbs new information about the environment without a dedicated mode or any controller retuning.

We demonstrate this with a constant cold-gas leak that the spacecraft has no prior knowledge of. The scenario is anti-ram (anti-velocity) pointing on an ESPA-class bus ($J = \mathrm{diag}(1, 1.2, 0.7)$ kg·m^2, three 50 A·m^2 magnetorquers, three reaction wheels with $h_{\max} = 0.5$ N·m·s). The plan uses a 600 s horizon, replanned every 500 s on a fixed schedule (at $t = 0, 500, 1000, 1500$ s), with a 1 s integration and control step. A momentum-rate observer estimates the net external torque on the spacecraft; once that estimate is available it is added to the planner's model, so the next replan plans against it. At $t = 600$ s a constant body-fixed leak $\tau_{\text{leak}} = 2 \times 10^{-4}\,\mathrm{normalize}([1,1,0.5])$ N·m switches on.

Figure 8 shows the adaptation. The spacecraft acquires anti-ram from a 57.6° initial offset, converging to 0.26° by $t \approx 100$ s, and tracks the rotating anti-ram direction at $\sim 0.3°$. The leak switches on at $t = 600$ s before it is part of the planner's model, and the pointing error grows, reaching 16.0° by $t \approx 1000$ s. Once the observer has resolved the leak and the $t = 1000$ s replan incorporates it, the planner adapts: it lays down a trajectory that accounts for the estimated torque, and the error collapses from 16.0° to 0.37° within 50 s, holding anti-ram at the 1–2° level thereafter.

The point is the architecture's openness to new information. Nothing about the leak was anticipated in the original plan, yet the moment it becomes known it is folded

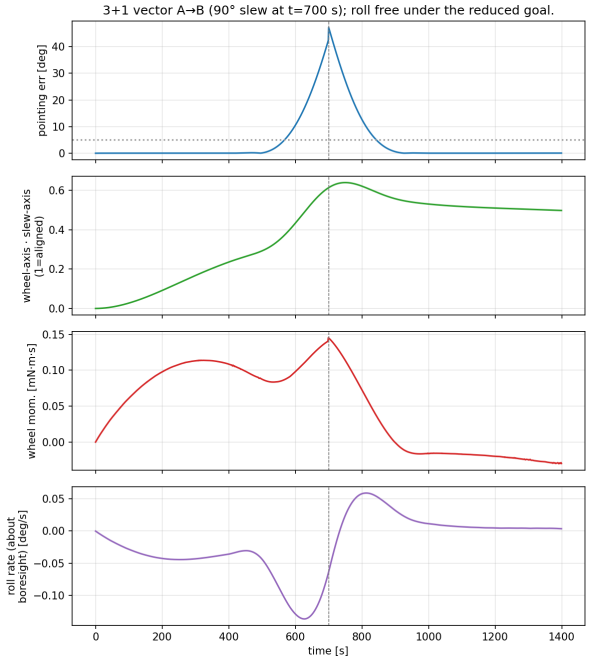


Figure 7: The planner spends a free rotational degree of freedom now to make a later slew easier. Foresight pre-positioning on 3+1, two sequential vector-pointing goals (A, then B, 90° apart; single wheel on $+\hat{z}_b$): while holding A (roll free under the reduced goal) it rolls about the free boresight to swing the wheel axis toward the upcoming slew axis (alignment $0.03 \rightarrow 0.55$) and pre-loads the wheel, leaving A-pointing undisturbed; B is acquired 144 s into the slew. The re-orientation was not commanded.

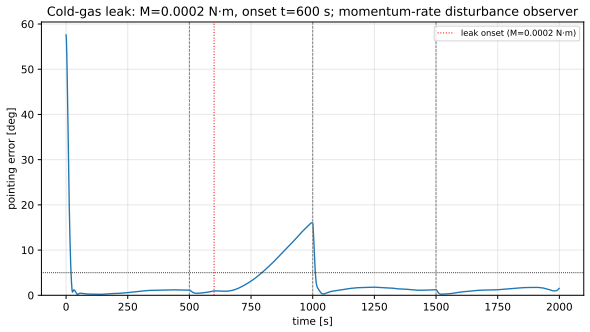


Figure 8: A disturbance unknown to the plan is absorbed the moment the model learns it. Anti-ram pointing on an ESPA-class 3MTQ+3RW bus under a constant cold-gas leak (onset $t = 600$ s, dotted line): error grows to 16° until the momentum-rate observer resolves the leak; the first replan to receive the estimate (dashed lines mark replan times) folds it into the model and pointing recovers to $\sim 0.3^\circ$. No event-detection logic—just the regular replanning cadence.

into the model and the planner adapts to it—the same way the planner incorporates the estimated disturbance, an updated goal, or a revised authority budget. The interval between onset and the next informed replan is simply the latency of the regular replanning cadence; a shorter cadence shrinks it. This adaptation comes from replanning continuously rather than from any event-detection or fault-trigger logic.

IV-F. Behavior at the Underactuated Boundary

Section II-D argued that an infeasible goal degrades gracefully—the planner returns a bounded, constraint-respecting trajectory and leaves the unmet error where the hardware cannot act, rather than saturating actuators or diverging. This section is that claim's hardest test: a *persistent* infeasibility, a goal the hardware cannot achieve at all (the recovery result above, by contrast, probed a transient—a feasible scenario perturbed by a discrete event). We use the most underactuated configuration in the study, 3MTQ+0RW, asked to hold a random full 3-axis attitude. Controllability analysis [12] tells us in advance that three magnetorquers cannot instantaneously command an arbitrary body torque (the rank is two, perpendicular to the local field), so the full-attitude task is at the edge of controllability and is only marginally reachable through the LTV averaging argument.

Across 100 Monte-Carlo trials the planner converges to within 5° in 18% of cases; the remaining 82 do not meet the strict bar. The shape of those 82 is the point. None diverge, none NaN, and none tumble: the planner lays down a smooth, low-rate trajectory with body rate held near $0.05^\circ/\text{s}$ at the median, and the single largest excursion in the entire run sweeps through the antipode of a near-180° eigenaxis slew and recovers to 12° final error. Read at a 30° acquisition threshold rather than the strict 5° bar, 87% of trials get close, and 99% finish within 90°. The honest summary is "nearly always gets close, always stays bounded," not "fails."

Figure 9 shows three representative trials: the envelope median, the trial with the highest peak pointing error during execution (a dramatic but recovering case), and the trial with the worst final error (the genuinely-stuck case). Two features of the figure are worth flagging. First, the error that remains concentrates in the axis the actuators cannot control. Decomposed, the *boresight* component (a 2-DOF projection onto the spacecraft-fixed $\hat{y}$-axis

pointing direction) is consistently tighter than the full 3-axis error—median boresight 11° versus median full 18°, with 15% of the non-converging set holding boresight within 5°. This is not the planner choosing to favor boresight pointing: it is solving the full-attitude problem, and the magnetorquer-only platform simply cannot control roll about the boresight, so the unmet error collects there. The practical lesson is about *matching the goal to the hardware*—the same 3MTQ+0RW platform asked for a reduced-attitude (vector-pointing) goal, the task its controllability actually supports, converges in 84% of trials (Section IV-B) against the 18% here for full attitude. A mission that only needs to aim a boresight should request reduced-attitude pointing on this hardware and will get it reliably; the full-attitude numbers above characterize behavior when the platform is pushed past what it can control. Second, the worst-final trial does pin its magnetorquers at rate limits during the first $\sim$250 s before settling to low-amplitude command activity; this is bang-bang allocation under high error, not chatter, and the body rate is bounded throughout. Handed an infeasible command the framework neither forces a bad solution nor fails unsafely—it stays bounded, achieves what the actuators allow, and leaves the operator a stable platform to re-task.

IV-G. Robustness to Model Error, Unmodeled Disturbances, and Tuning

The planner plans from a digital twin that is never exactly right, and from hyperparameters chosen once rather than per task. Three sweeps bound how far each can drift before performance degrades: a twin that is *wrong* (model error), one that is *incomplete* (unmodeled disturbances), and a tuning that is *approximate*. In every case the closed loop absorbs the gap, because the plan is a reference the TVLQR tracker (Section III-C) corrects against rather than a command it executes blindly.

Model error. On 3+1 ECI-pointing acquisition ($t_f =$ 1000 s) we independently sweep inertia ($\hat{J}/J \in [0.7, 1.3]$), magnetic-field magnitude ($\mathbf{B}_{\text{plant}}/\mathbf{B}_{\text{planned}} \in [0.7, 1.3]$), and field *rotation* ($\theta \in [0°, 60°]$, the adversarial case, since it turns the magnetorquer torque plane away from where the plan expects it). Final pointing accuracy stays between 0.01° and 0.03°—indistinguishable from nominal—across the full inertia and magnitude ranges and field rotation out to 45° (Figure 11). The only measurable effects are a mild, monotonic settling-time rise with overestimated inertia (181 s at $\hat{J}/J = 0.7$ to 230 s at 1.3; a heavier model gives a more sluggish reference) and a soft threshold at 60° rotation, where settling nearly doubles to 372 s with a 3.6° transient before recovery—a soft boundary, not a cliff. Because this is static-target regulation, model error shows up almost entirely in the transient; a campaign in which the plan is itself the desired behavior (a time-constrained slew, or the spin of Section IV-D.1) would stress mismatch harder, and a Monte Carlo extension over the mismatch distribution is left for future work. One such mismatch is already present, uninjected: in the replanning campaign (Section IV-E) the planner's IGRF implementation and frame differ from the plant's by roughly 4° in direction and 4% in magnitude—within the band established here, and absorbed by the tracker.

Unmodeled disturbances. The complementary question is disturbances the planner does not model *at all*. On the 3+1 reduced-attitude testbed ($N = 20$ paired seeds, $t_f = 1000$ s) we run six environments—a residual magnetic dipole of 0.05 or 0.10 A·m^2 (bracketing a typical uncompensated 3U residual), gravity-gradient with drag and solar radiation pressure (made nonzero by a 2.1 cm cp/cg offset), and the full combination at each dipole level—with the planner modeling none of them. Convergence is 100% in all six (Figure 10). The mean final boresight error scales with the residual dipole (0.06° with none, rising to 0.36° and 0.80° at 0.05 and 0.10 A·m^2), while gravity-gradient, drag, and SRP alone leave it essentially unchanged: the residual dipole is the term that matters, and even an order of magnitude of it holds sub-degree pointing unaided. This is a deliberate *blind-rejection floor*. The momentum-rate observer that resolved the cold-gas leak (Section IV-E) would estimate the dipole equally well, and either the next replan or the disturbance-augmented TVLQR of Section III-C would then drive the residual lower—we omit both to show the unaided floor is already sub-degree.

Tuning. The framework's value collapses if its hyperparameters need bespoke tuning per task. Sweeping the three knobs most likely to be retuned—the augmented-Lagrangian initial penalty, the attitude cost weight, and the control-effort multiplier—one at a time on the 3+1 reduced campaign (Table 4), convergence is 100% in every cell and the mean final error stays below 0.15° throughout. The AL penalty is inert (two decades of variation move nothing), and the control-effort weight is a weak, non-monotone lever. The one real trade is the attitude weight against solve time: from 1 to 100 it spans an 8$\times$ range in solve time (2.2 to 17.5 s) for a modest accuracy gain (0.103° to 0.056°), making the attitude weight the natural knob to lower the planner's onboard compute footprint without breaking convergence.

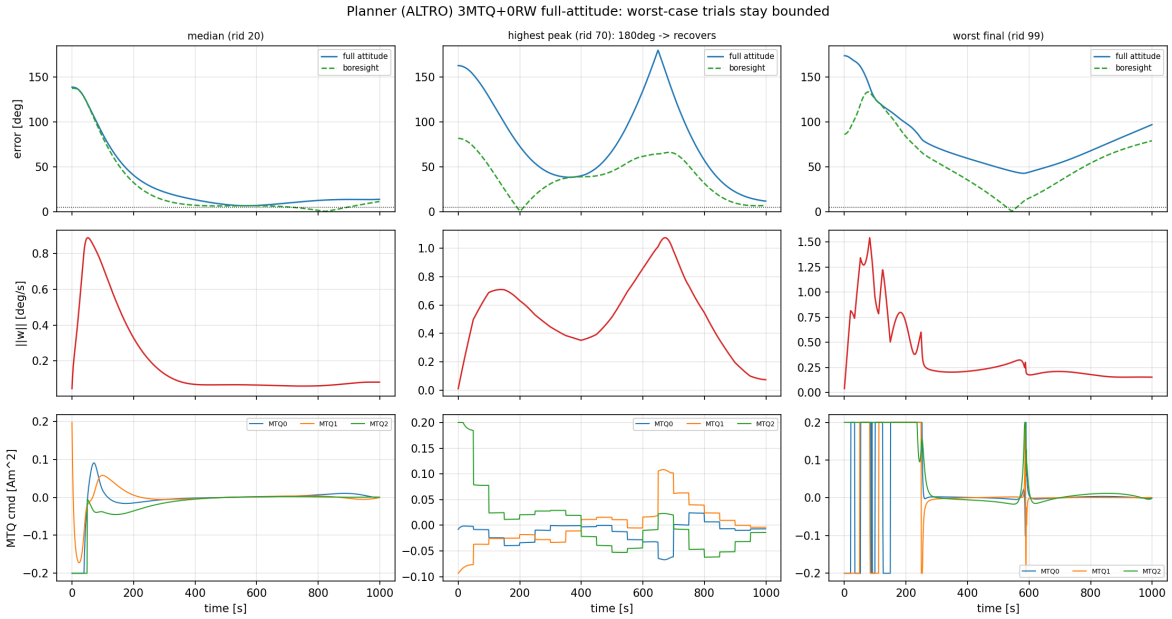


Figure 9: Pushed past controllability, the planner degrades gracefully rather than diverging. Behavior on the 3MTQ+0RW full-attitude task across three representative Monte-Carlo trials: envelope median (left), highest peak error (center, a 180° sweep recovering to 12°), and worst final error (right, the stuck case at 97°). Rows: full 3-axis error (solid) with 2-DOF boresight error (dashed); body-rate magnitude; magnetorquer dipole commands. No trial diverges or tumbles; the worst case briefly saturates the magnetorquers before settling. Boresight error undercuts full-attitude error because the unmet error collects in the uncontrollable roll axis, not because boresight is prioritized.

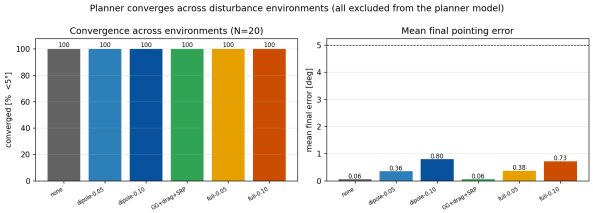


Figure 10: Residual magnetic dipole dominates blind disturbance rejection. Six unmodeled disturbance environments on 3+1 reduced-attitude pointing ($N = 20$ paired, $t_f = 1000$ s): 100% convergence in every environment, with mean final boresight error scaling with the residual dipole (0.06° clean to 0.80° at 0.10 A·m^2); gravity-gradient, drag, and SRP alone are negligible.

Table 4: Hyperparameter sensitivity, one-at-a-time around canonical settings (3+1 reduced, 6 trials each, paired seeds). Convergence is 100% in every cell.

Knob	Value	Mean (°)	Solve (s)
baseline	—	0.139	6.5
penalty_init	10^{-4}	0.144	7.3
penalty_init	10^{-2}	0.141	7.4
cost_main.angle	1	0.103	2.2
cost_main.angle	100	0.056	17.5
cost_main.control_mult	0.5	0.078	6.8
cost_main.control_mult	2.0	0.085	5.5

V. DISCUSSION

V-A. CONOPS Implications

The planner changes what operators need to specify. Rather than defining mode transitions, slew profiles, desaturation windows, and gain schedules, the operator provides a list of pointing goals and times. The planner handles maneuver generation, momentum management, and feasibility assessment internally. When conditions change—an unexpected disturbance, an updated goal, a partially failed actuator—the planner incorporates the new information on its next replanning cycle without ground intervention.

Consider a routine LEO Earth-observation pass: the

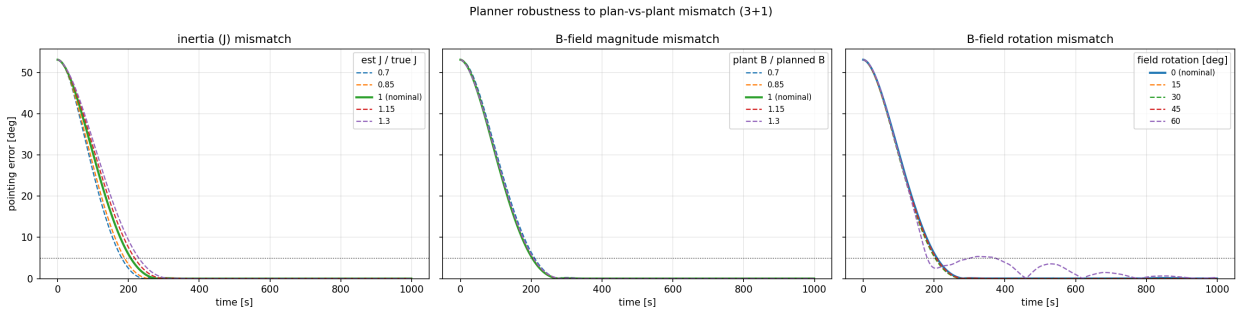


Figure 11: The closed loop absorbs digital-twin error; only an extreme field-rotation case is even visible. Pointing-error trajectories under three plan-vs-plant model-error sweeps on 3+1 ECI pointing—left: inertia $\hat{J}/J \in [0.7, 1.3]$; middle: field-magnitude scaling $\hat{\mathbf{B}}_{\text{plant}}/\mathbf{B}_{\text{planned}} \in [0.7, 1.3]$; right: field rotation $\theta \in [0^\circ, 60^\circ]$. All but the 60° rotation converge to the same final accuracy in similar time; 60° nearly doubles settling.

spacecraft must image a ground target during a visibility window, then return to sun-pointing for power generation. With conventional ADCS, this requires a slew mode, a pointing mode, a transition back, and logic governing when to switch between them—plus a separate momentum desaturation strategy running alongside or scheduled between maneuvers. With the planner, this is a two-entry goal list. The planner generates a trajectory that slews to the target, holds pointing during the pass, transitions back to sun-pointing, and manages wheel momentum throughout, all from a single optimization.

The same architecture scales to more demanding sequences without additional development. The sequential multi-goal results in Section IV-C demonstrate transitions between three pointing objectives with implicit momentum management and no mode switching. The spinning maneuver in Section IV-D.1 shows the planner autonomously discovering a strategy to maintain pointing under a disturbance that exceeds direct actuator authority—a solution that would be difficult to anticipate, let alone encode, in a conventional mode-based architecture. These are not separate capabilities requiring separate engineering effort; they emerge from the same planner solving the same type of problem.

The planner produces these behaviors only when the goal timeline leaves it room to act, and there are two kinds. The first is temporal: a no-goal coast between active windows—the multi-goal timeline of Section IV-C leaves 200 s between each goal—gives the planner room to schedule a slew into the gap, beginning each transition before the current window closes, which is why a hold metric read at window-end undersells the result. The cost of omitting that slack is visible in Figure 7, where two goals sit back-to-back with no coast: the $A \to B$ slew has nowhere to go but the convergence windows themselves, so the boresight begins leaving A before its window ends and B is not reacquired until 144 s into the next. The second kind of room is spatial: a reduced (vector) goal leaves the roll about the boresight free, and the planner spends it—pre-positioning the wheel during the A-hold in that same figure, or settling into the spin-stabilized terminal state of Section IV-D.1—the same freedom that lets reduced goals converge far more reliably than full three-axis targets on underactuated platforms (Sections IV-B, IV-F). Over-constraining on either axis—packing goals back-to-back, or demanding full attitude where a boresight suffices—removes the room and pushes the planner back toward the reactive, mode-like behavior it was meant to replace. The practical rule is to specify only what the mission needs, a pointing direction by a deadline, and leave both the time between goals and the unforced axes to the optimization.

V-B. Computational Performance

The SALTRO solver runs in C++ with a Python interface for problem setup. Table 5 reports cold-start solve times on a Raspberry Pi 4—a representative CubeSat-class flight processor—across the three actuator configurations and two horizon lengths.

Solve time reflects the size and richness of the optimization rather than the difficulty of control. The fully actuated 3MTQ+3RW case is effectively free—three wheels command torque directly, so the solver converges in a few iterations—while the rank-deficient, time-varying authority of magnetorquer-only 3MTQ+0RW makes it harder. The hybrid 3MTQ+1RW is the most demanding, and counterintuitively slower than mag-only despite its extra actuator: the added reaction-wheel momentum state enlarges the problem and the wheel opens a richer space of feasible trajectories to search. Cost grows roughly linearly with horizon length—a tenfold increase

Table 5: SALTRO cold-start solve time (seconds) on a Raspberry Pi 4, by actuator configuration and horizon. Each cell is the median (and worst) of ten random LEO scenarios—400–600 km circular orbits, random initial attitude and goal—each a single SALTRO solve at $\Delta t = 10$ s (50- and 500-step trajectories).

	500 s		5000 s	
Config	Med.	Max	Med.	Max
3MTQ + 0RW	0.37	0.56	2.08	3.96
3MTQ + 1RW	0.78	0.98	8.25	14.39
3MTQ + 3RW	0.03	0.03	0.27	0.72

in steps, from 50 to 500, costs roughly tenfold in time.

These times sit far inside the replanning budget. At the 500 s horizon used in the scenarios above, even the worst of ten hybrid solves is under a second; at a full 5000 s horizon it is 14.4 s. Both are a small fraction of the overlap buffer (Section III-B)—170–250 s for a 500 s trajectory, proportionally more for longer ones—so the next trajectory is produced with wide margin while the current one is flown.

Larger timesteps reduce solve time further at the cost of trajectory resolution; for many CubeSat missions 10-second steps are adequate, since maneuver timescales are long relative to the dynamics and the TVLQR tracker (Section III-C) handles inter-step corrections. Where margin is tight, the horizon can be shortened, the iLQR iteration cap lowered, or the solve warm-started from the previous trajectory—tuning adjustments, not architectural changes—and the fallback hierarchy (Section III-D) catches a slow or non-converging solve by continuing on overlap or reverting to reactive control.

V-C. Objective Function Tuning

The planner requires tuning of cost function weights—penalties on pointing error, momentum buildup, control effort, and constraint violation. This is a form of tuning, but it is fundamentally different from gain tuning in conventional controllers. Cost weights specify *what the mission values*: how much pointing accuracy matters relative to momentum management, how aggressively to use actuators, how hard to penalize constraint violations. They do not specify *how to achieve it*—the optimizer handles that. A gain change in a PD controller alters the dynamics of the closed-loop system and can introduce instability; a cost weight change in the planner shifts the tradeoff between competing objectives while the optimizer continues to produce feasible trajectories.

In practice, the cost weights are less sensitive than one might expect. The planner converges to reasonable trajectories across a range of weightings; the trajectories differ in how they balance competing objectives, but the qualitative behavior—converge to the goal, manage momentum, respect actuator limits—is consistent. The hyperparameter sensitivity sweep in Section IV-G quantifies this.

V-D. Limitations and Future Work

The primary limitation is computational cost. The planner is orders of magnitude slower than a reactive controller, and while the solve times in Table 5 are compatible with CubeSat-class replanning cadences, they leave less margin than a conventional control loop. The fallback architecture (Section III-D) addresses the operational risk—if a solve is slow or fails, the spacecraft continues on its current trajectory or reverts to reactive control—but this remains the largest trade-off compared to classical control laws.

Convergence—meeting the goal and satisfying all constraints to tolerance—is not guaranteed for every scenario; the SALTRO solver can exhaust its iteration budget on particularly difficult problems (e.g., near-singular magnetic field geometry, very large required slews with minimal actuation). The trajectory it returns is still dynamically feasible, and in testing non-convergence is infrequent and is handled by the fallback architecture, making it an operational nuisance rather than a safety concern. The robustness results in Section IV-G provide concrete guidance on how much model error the system tolerates and how insensitive it is to cost-weight choices.

Objective function tuning, while simpler than gain tuning, still requires the user to make choices about mission priorities. Developing default cost-weight profiles for common mission types (Earth observation, sun-tracking, detumble-and-stabilize)—and, beyond static defaults, automatically tuning the cost weights from a high-level objective specification—would further lower the barrier to adoption. Such auto-tuning appears feasible but finicky, and is left to future work.

Near-term extensions include flight demonstration on BeaverCube 2 [9], warm-starting from prior trajectories to reduce solve time across replanning cycles, and a fully compiled solver for tighter embedded timing margins.

On the modeling side, the formulation extends naturally to further actuators—thrusters as a bounded torque input, and control-moment gyros, whose gimbal dynamics

and singularity-avoidance steering would test whether the mode-free behavior carries over to actuators with richer internal state—and to a fuller treatment of disturbances: predicting and planning against them over the horizon, rather than leaving the tracker and the next replan to absorb what the twin omits, could lower the blind-rejection floor of Section IV-G.

On the solver side, several internal choices invite study. A coarse-to-fine solve—planning at a large timestep and refining—may converge faster or more reliably than the single-resolution pass used here, and the timestep itself trades solve time against trajectory fidelity in ways worth characterizing systematically (partly another tuning knob). The backward-pass regularization scheme and the form of the attitude-error cost, both fixed here by experience, likewise shape convergence and the character of the trajectories produced, and principled comparison could help most on the hardest, near-singular cases.

VI. CONCLUSION

We have presented a feasibility-aware attitude trajectory planner that replaces mode-switching ADCS with continuous planning from a digital twin. Operators specify high-level pointing goals; the planner returns dynamically feasible trajectories that respect the full nonlinear dynamics, time-varying actuator authority, and operational constraints directly—without collocation or slack relaxations—together with the time-varying LQR gains to track them. One architecture handles slews, pointing, momentum management, and goal sequencing, with no explicit modes.

Across Monte Carlo campaigns the planner outperforms classical feedback on the task each platform can support: 84% convergence versus 27% for a magnetorquer-only spacecraft, and 94% versus 90%—at 1.2° mean final error against 9.1°—with one reaction wheel added. The same architecture chains multiple terminal goals in a single solve and adapts online, re-solving from a perturbed state in seconds. Pushed past what its actuators can control, it degrades gracefully rather than diverging: it stays bounded, holds the pointing the hardware can deliver, and leaves the residual in the uncontrollable axis. It also discovers strategies a mode-based design would have to anticipate explicitly—most strikingly, an emergent spin-stabilized terminal state that absorbs a body-fixed disturbance the actuators cannot directly oppose. Throughout, the closed-loop pipeline tolerates the plan-vs-plant model error a real operator cannot eliminate.

The larger claim is not a better controller but a different paradigm for attitude operations: the ground specifies goals rather than maneuvers, the spacecraft plans from its own model of itself, and the architecture absorbs the off-nominal conditions— unexpected disturbances, infeasible goals, model mismatch, hardware aging—that otherwise demand mode switching, fault management, or ground intervention. The same flight software serves a 3MTQ+0RW, 3MTQ+1RW, or 3MTQ+3RW spacecraft through a configuration change rather than a retuned controller. What remains is operational: hardware-in-the-loop validation, a compiled embedded solver for tighter timing margins, and the flight demonstration that turns these results into an on-orbit track record.

ACKNOWLEDGMENTS

The authors thank Dr. Alex Rose, whose contributions in the early stages of this work were invaluable.

REFERENCES

[1] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Macau, China, Nov. 2019, pp. 7674–7679.

[2] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning with attitude," *IEEE Robot. Autom. Lett.*, vol. 6, pp. 5658–5664, 2021.

[3] M. Lovera and A. Astolfi, "Global magnetic attitude control of spacecraft in the presence of gravity gradient," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 42, no. 3, pp. 796–805, Jul. 2006.

[4] R. Wisniewski, "Sliding mode attitude control for magnetic actuated satellite," *IFAC Proc. Volumes*, vol. 31, no. 22, pp. 179–184, 1998.

[5] B. Wie, *Space Vehicle Dynamics and Control*, 2nd ed. Reston, VA, USA: AIAA Education Series, 2008.

[6] E. A. Hogan and H. Schaub, "Three-axis attitude control using redundant reaction wheels with continuous momentum dumping," *J. Guid. Control Dyn.*, vol. 38, no. 10, pp. 1865–1871, 2015.

[7] P. McKeen and K. Cahoy, "Method for pointing spacecraft while minimizing change in orientation," *J. Guid. Control Dyn.*, vol. 45, no. 8, pp. 1554–1557, 2022.

[8] P. McKeen, A. Meredith, and K. Cahoy, "Trajectory-planning attitude control system

for satellites with magnetic attitude control and one reaction wheel," in *AAS/AIAA Astrodynamics Specialist Conf.*, 2022, paper AAS 22-810.

[9] P. McKeen, N. Scheuer, and K. Cahoy, "A modular open-source Python framework for small satellite attitude determination and control," presented at the Small Satellites, Systems & Services Symp. (4S/SmallSat Europe), 2026.

[10] P. McKeen and K. Cahoy, "Generalized attitude estimation for spacecraft," in *Proc. 38th Annu. Small Satellite Conf.*, Logan, UT, USA, 2024, paper SSC24-WV-08.

[11] P. do Vale Pereira *et al.*, "BeaverCube: Coastal imaging with VIS/LWIR CubeSats," in *Proc. 34th Annu. Small Satellite Conf.*, Logan, UT, USA, 2020.

[12] P. McKeen, N. Scheuer, and K. Cahoy, "Generalized attitude control for small spacecraft," in *Proc. 40th Annu. Small Satellite Conf.*, Logan, UT, USA, 2026, paper SSC26-P2-54.

[13] P. McKeen, "Computational methods to improve satellite attitude determination and control with a focus on autonomy, generalizability, and underactuation," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2024.