aerospace

MDPI

Project Report

MAE 5830 Astronautic Optimization: Final Project

Cyrus Nolan [1]

[1] Cornell University; can62@cornell.edu

1. Introduction

Rigid body rotation has many applications. One interesting example is satellite re-orientation. The following dynamics describe 1D rotation about an axis, where I is the moment of inertia and θ is the rotation about that axis:

$$\mathrm{T} = I\ddot{\theta}$$

In rest-to-rest reorientation with inertia, timespan, and endpoint angular position normalized to unity, the plant dynamics and boundary values are:

$$\begin{aligned} \mathrm{T} &= \ddot{\theta} \\ \theta(t_0) &= 0 \\ \omega(t_0) &= 0 \\ \theta(1) &= 1 \\ \omega(1) &= 0 \end{aligned}$$

Control systems execute autonomous re-orientation of rigid bodies given a commanded rotation angle θ. Control design using classical control theory requires first linearizing the system about a desired equilibrium point. The linearized model is good enough in many cases. However, the set of possible control outputs and resulting state trajectories are severely limited because the linearized model requires gains to be time invariant. There are cases where classical control theory is unable to produce the most efficient solutions. Two examples are swinging up a pendulum to the inverted position and large angle satellite reorientation. For state-to-state maneuvers such as these, we can use Pontryagin's method to analytically find an optimal open loop control, u^*, that isn't constrained to be linear and time invariant. The set of possible trajectories is much broader, so more efficient solutions than those produced by classical control theory may be found. After finding u^*, the problem becomes implementation: How do we use this newfound knowledge to execute the more efficient state-to-state maneuver in a robust way? In the presence of disturbances like inertia variation and limitations on sensor accuracy, can u^* still outperform classical controllers? This report presents the results of a rest-to-rest reorientation maneuver of a second order system using five different implementations of u^*, as well as an implementation using a P+V controller for comparison. Several key figures of merit (FoM) describe the performance of each control system architecture. The first two FoM are endpoint state error and endpoint state spread, which represent the accuracy and precision

possible using each control system. The next FoM is quadratic cost J, where $J = \frac{1}{2}\int u^2\,dt$. J represents fuel usage. The final FoM is average runtime, which represents computational burden.

2. Control System Architectures

I. P+V Control

This control system is used as a benchmark representing a high-performing classical control architecture. The P+V control law is high-performing because it has 0 steady state error. The control law is:

$$u = k_p(\theta_d - \theta) - k_v\omega$$

Given this control law, the equation of motion becomes:

$$k_p(\theta_d - \theta) - k_v\omega = I\ddot{\theta}$$

The gains used for the Monte Carlo simulation are $k_p = 27.9791$ and $k_v = 7.4053$, determined by from $t_s = 0.9$ and $\zeta = 0.7$. The transfer function from commanded attitude θ_d to steady state error $\theta_d - \theta$ is:

$$\frac{\theta_d - \theta}{\theta_d} = \frac{(k_v s + Is^2)}{k_p + k_v s + Is^2}$$

Using the final value theorem, we can see the steady state error is $\frac{0}{k_p} = 0$.

II. Open Loop Guidance

This implementation feeds u^* to the plant. Position θ and angular rate ω are measured but not fed back. u^* is derived by solving the double integrator quadratic control (DQC) problem with Pontryagin's method. The problem is defined as:

$$\text{Minimize: } J[\theta(\cdot), \omega(\cdot), u(\cdot)] = \frac{1}{2}\int_{t_0}^{t_f} u^2 dt$$

$$\text{Subject to: } \dot{\theta} = \omega,\ \dot{\omega} = u,\ (\theta_0, \omega_0) = (0,0),\ \left(\theta_f, \omega_f\right) = (1,0)$$

The result is:

$$u^* = at + b, \qquad a = -12, \qquad b = 6$$

The implemented equation of motion is:

$$6 - 12t = I\ddot{\theta}$$

III. RTOC

Real time optimal control re-solves the boundary value problem from the previous section for a and b at each time step using the current θ and ω. The system of equations is defined as follows, where t_0, θ_0, and ω_0 are the current time, angular position, and angular velocity:

$$\begin{bmatrix} t_0^3/6 & t_0^3/2 & t_0 & 1 \\ t_0^2/2 & t_0 & 1 & 0 \\ 1/6 & 1/2 & 1 & 1 \\ 1/2 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \omega_0 \\ 1 \\ 0 \end{bmatrix}$$

Defining the 4x4 matrix above as T, the 4x1 matrix of unknowns as p, and the 4x1 matrix on the right-hand side of the equation as b, the controller performs the following calculation at each time step to solve for p:

$$p = \mathrm{T}^{-1} b$$

This increases robustness because the loop is closed. In the Monte Carlo analysis with RTOC, T^{-1} is calculated using pinv(T), which is the Moore-Penrose Pseudoinverse. There are multiple choices for taking the inverse. This analysis uses the pinv because the simulation obtains analytic and machine precision results using pinv without disturbances and with unlimited sensor accuracy.

Close to the endpoint, pinv(T) becomes inaccurate because the condition number of T becomes large. To handle this, the simulation calculates the condition number of T at each time step. When $1/cond(\mathrm{T})$ goes to 0, a switch is thrown and the control input becomes $6 - 12t$ for the final few time steps.

IV. Double Integrator Patching Filter

The final three control system architectures consist of a legacy P+V feedback control system and a patching filter that inputs u^* and outputs θ_d^*. θ_d^* is then the input for the legacy P+V system. The double integrator patching filter calculates θ_d^* by integrating $u^* = 6 - 12t$ twice with respect to time. Arbitrarily setting $t_s = 2$ and $t_r = 0.3$ produces the gains of the legacy system that the Monte Carlo simulation uses. We are simulating a scenario in which all we know is that the legacy system is a P+V feedback control system and we can't change it at all.

V. Double Integrator Patching Filter With Gain Tuning

This architecture consists of the same patching filter as the previous architecture. This time, gain tuning of the legacy P+V system is allowed. Gain tuning prioritizes minimization of endpoint angular velocity error because the rest-to-rest maneuver isn't very successful if $\omega(1)$ is far from 0. For example, some gains produced small angular position error with large angular rate error, which just means the angle happened to pass through 1 at $t = 1$. Initial experimentation with gains led to discovery of a local endpoint angular rate error minimum somewhere within $350 \le k_p \le 450$ and $1 \le k_v \le 10$, and iterating all possible integer combinations produced $k_p = 424$ and $k_v = 3$.

VI. Control Law Inversion Patching Filter

The transfer function from θ_d to u within the legacy P+V controller is:

$$\frac{u^*}{\theta_d} = \frac{k_p s^2}{s^2 + k_v s + k_p}$$

The control law inversion patching filter calculates θ_d by passing u^* through the inverse of transfer function above, which makes the patching filter transfer function from u^* to θ_d:

$$\frac{\theta_d}{u^*} = \frac{s^2 + k_v s + k_p}{k_p s^2}$$

With this patching filter, the transfer function from u^* to τ, the torque input to the plant, is 1, and the optimal control effectively bypasses the legacy system.

3. Results and Analysis

Each Monte Carlo simulation consists of 1500 runs. Each simulation uses ode8 (Dormand-Prince) with a 0.01 second step size. The simulations use this solver and step combination because it produces $\theta(1) = 1$ and $\omega(1) < eps$ when disturbances and sensor limitations are turned off. Throughout each run, inertia varies uniformly by +/- 5% to model fuel slosh. The angular position and rate sensors vary uniformly by +/- 0.005 rad and rad/s to model the accuracy of lower end attitude determination sensors like magnetometers and sun sensors. Note that all plots of Monte Carlo simulation results use the same x and y scales except for the double integrator patching filter because it missed the mark by so much.

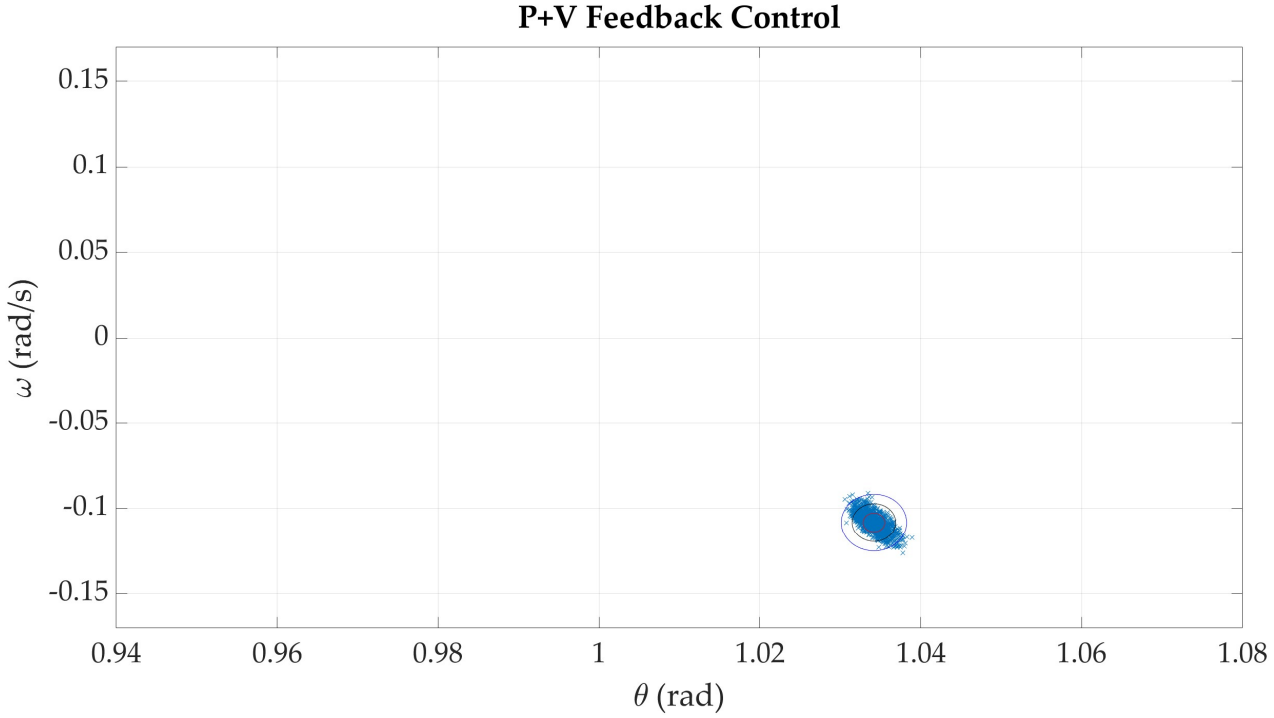

Figure 1: P+V Feedback Control

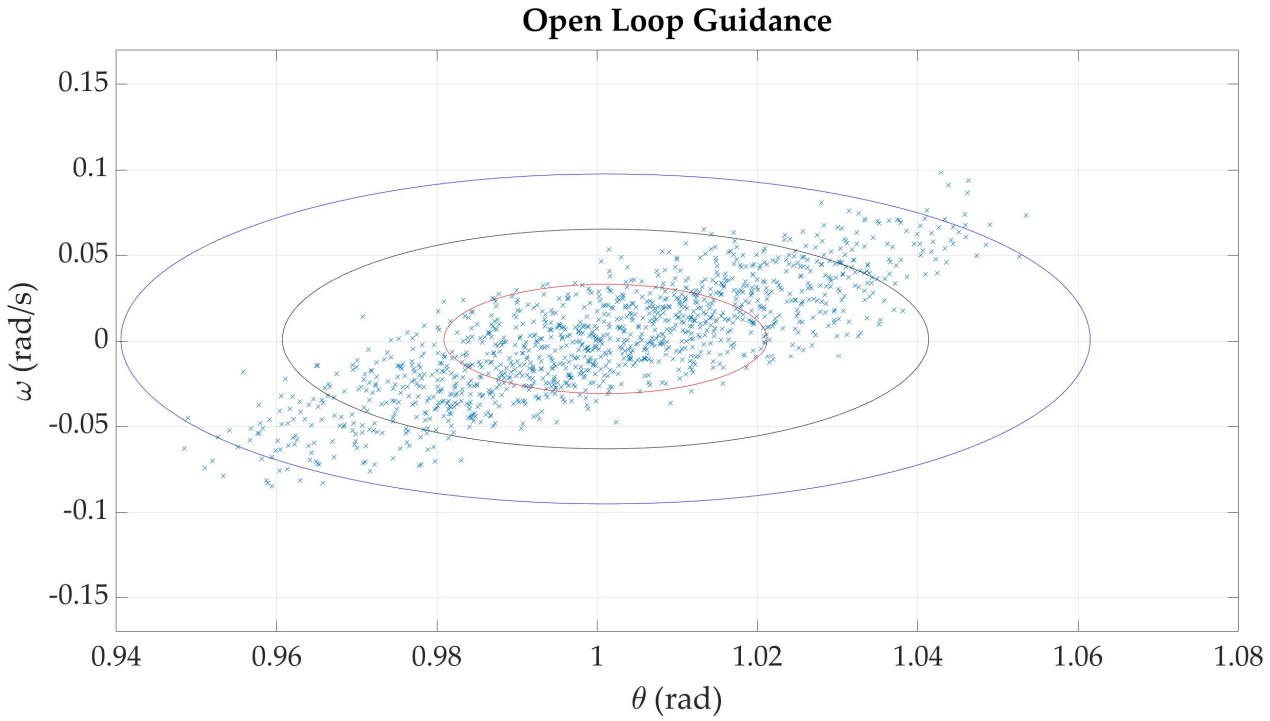

Figure 2: Open Loop Guidance

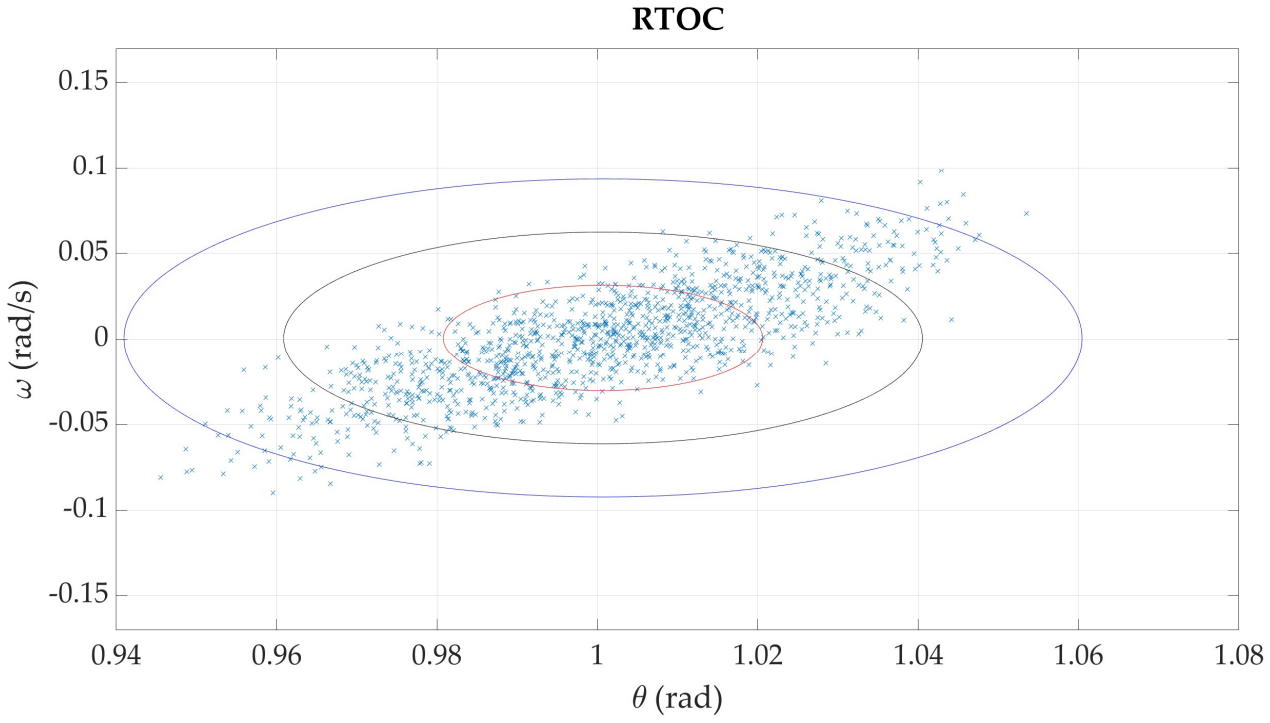

Figure 3: RTOC

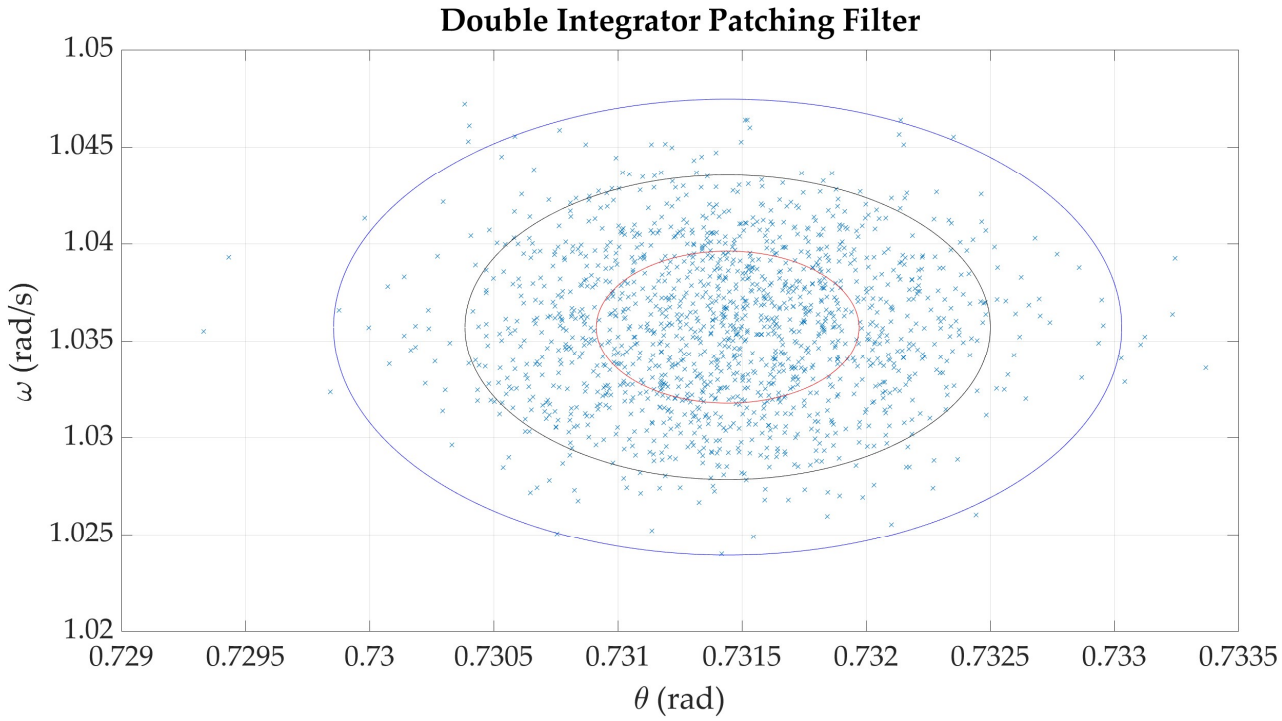

Figure 4: Double integrator patching filter (note: different scale)

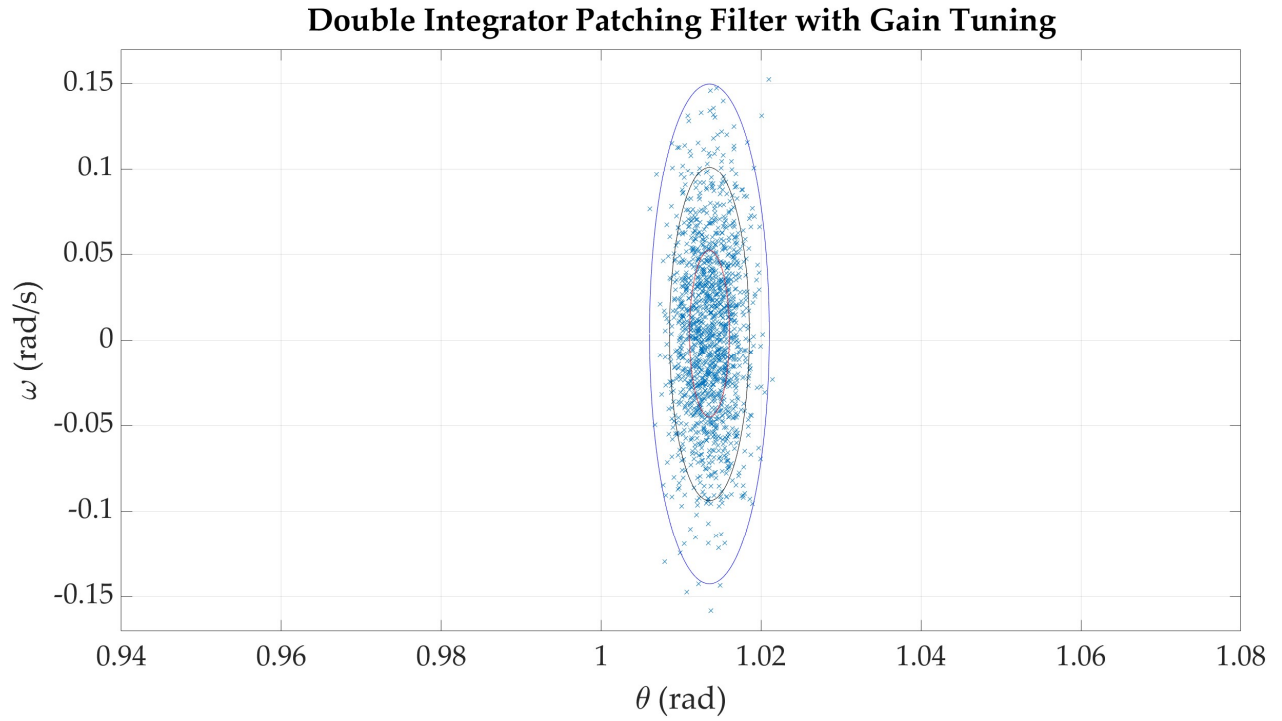

Figure 5: Double integrator patching filter with gain tuning

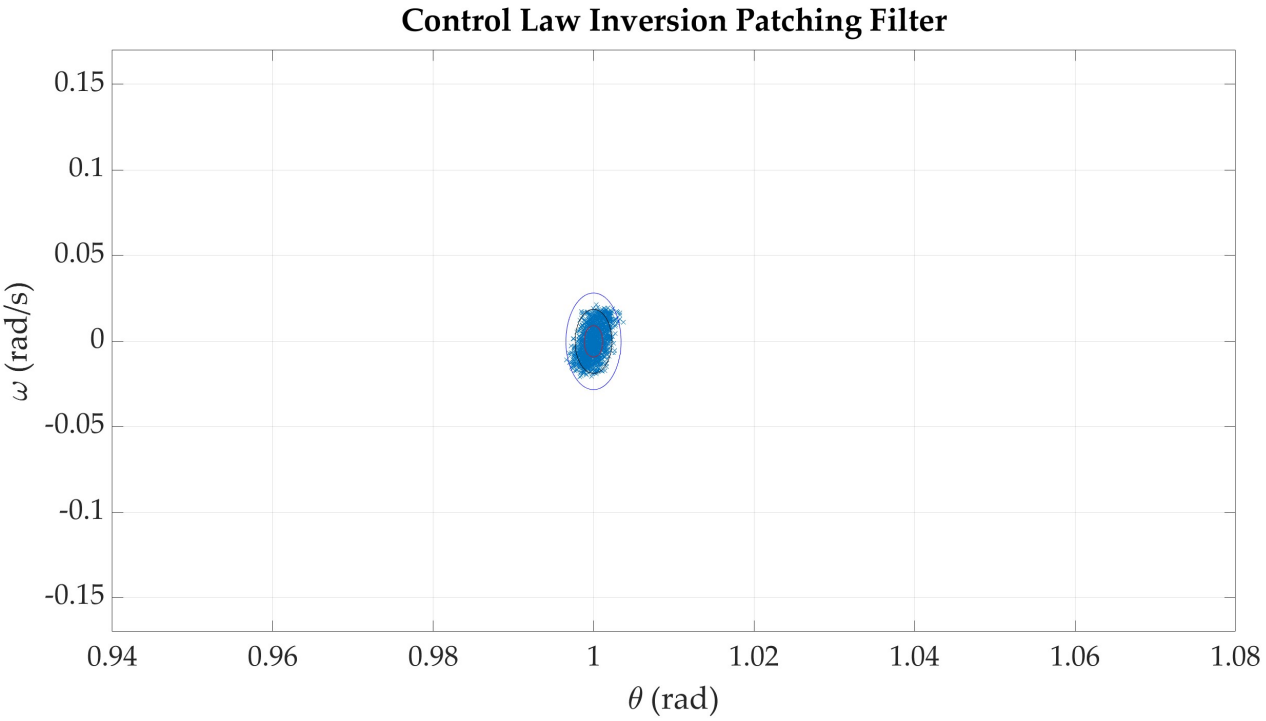

Figure 6: Control law inversion patching filter

Table 1. Monte Carlo simulation results for various sensor sample rates

Control System Architecture	Error θ_f (rad)	Std θ_f (rad)	Error ω_f (rad/s)	Std ω_f (rad/s)	Cost (CU)	Computational Burden (s)
P+V	3.4174e-2	1.3534e-3	-0.10839	5.5644e-3	26.4069	1.0457
Open Loop Guidance	1.0323e-3	2.0152e-2	1.0793e-3	3.2051e-2	6	1.0809
RTOC	6.9465e-4	1.9912e-2	5.2014e-4	3.0902e-2	6	1.0697
Double-Integrator Patching Filter (DIPF)	-0.26856	5.2913e-4	1.0357	3.9214e-3	1.4657	1.0244
Gain-Tuning with DIPF	1.354e-2	2.496e-3	3.589e-3	4.8717e-2	10.1949	0.27625
Control Law Inversion Patching Filter	-2.1934e-5	1.1462e-3	-3.0305e-4	9.4504e-3	6.0107	0.26856

* eps = 2.2204e-16

The double-integrator patching filter is the worst performer by a lot. Additionally, its performance is entirely dependent on the gains of the legacy P+V system. The top performing architecture is the control law inversion patching filter. It is the best in terms of endpoint position and accuracy. Its cost is nearly analytic 6, and it has the lowest computational burden. RTOC has similarly small mean error in θ and ω, and it's centered nicely around $(1,0)$ in figure 3, but the spread of endpoint θ values is the largest. Interestingly, open loop was only marginally worse than RTOC in accuracy and precision.

4. Conclusions

The preferred architecture for integrated guidance and control of a second order system with dynamics $T = I\ddot{\theta}$ is a control inverting patching filter that connects our optimal control u^* to a P+V feedback control system.

The patching filter should not ignore the gains of the legacy system. The double-integrator patching filter took this approach and performed horribly. If the gains of the P+V system are known, we can implement the preferred method—a control law inverting patching filter. If the gains are unknown but tunable, we can implement the double inverting patching filter with gain tuning and iteratively tune the gains. If the gains are both known and tunable, we should implement the control-inverting patching filter. An interesting follow up question would be to see how changing the gains affects the performance of the control inverting patching filter. Is there some combination of gains that increases or decreases performance? Or do the gains have absolutely no effect on performance? The latter case is likely because that's the point of the control inverting patching filter.

With 10% inertia variation and sensor accuracy approximately modeled after magnetometers or sun sensors, RTOC performed only marginally better than open loop guidance. The difference in performance would favor RTOC more and more as the magnitudes of disturbances increase. If the current model accurately represents the 3σ worst-case scenario disturbances for a particular application, engineers should consider discarding RTOC for the simplicity of open loop guidance.