# Gain Scheduled LQI Trajectory Tracking Control of a Quadrotor

*Samuel Bednarski and Michael Dermksian*

## Problem Motivation

In the first part of this project, a fixed-gain LQR controller was designed for the quadrotor system linearized about a single operating point. While this controller was found to stabilize the quadrotor well near the operating point, it cannot track complex trajectories very close, nor can it stabilize around other operating points. In particular, the linearized LQR design approach can only stabilize the quadrotor around a small range of yaw. For large changes in yaw the quadrotor becomes unstable.

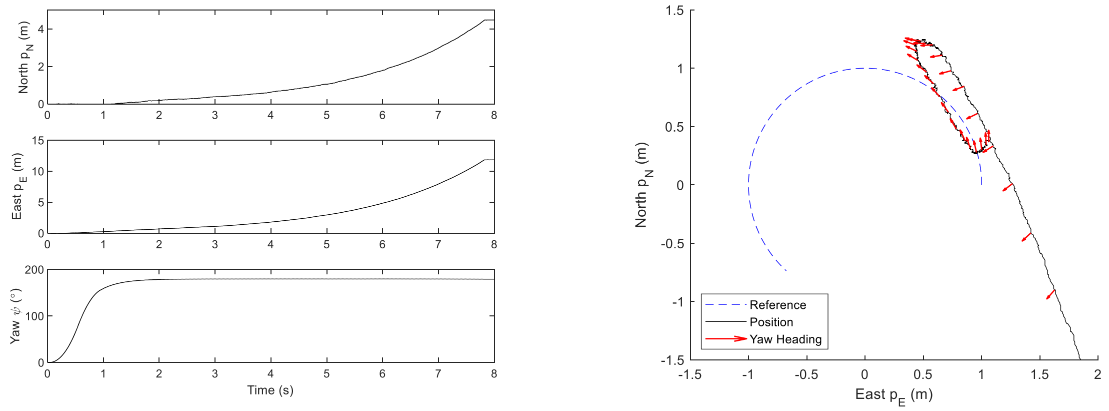

Figure 1. (*Left*) Loss of quadrotor stability when attempting to stabilize at 180° yaw. (*Right*) Inability to track a circular reference after significant rotation about yaw.

Figure 1 shows examples of the shortcomings of the fixed-gain LQR design from PX4 simulations. In the left image, the quadrotor attempts to stabilize about a yaw which is 180° from the linearized operating point. Though it actually converges to this yaw, the position drifts far until the quadrotor crashes. In the right image, the quadrotor gradually becomes unable to track the circular reference as the yaw changes too far from the linearized model.

Ideally, the quadcopter should be able to achieve steady-state behavior at any 3D position coordinate and yaw angle. Since the nonlinear dynamic model does not directly depend on the position, steady state at any 3D position can be achieved even with the linearized approach. However, the dynamic model does depend on the yaw angle, so the linearized approach cannot achieve steady state behavior for any arbitrary yaw.

We are interested in extending the capabilities of the quadrotor to tracking 3D trajectories with arbitrary yaw references. To achieve this, multiple gain scheduling approaches are utilized to extend the capabilities of LQR design. Additionally, integral control is included to provide asymptotic tracking of ramp references for the 3D position and yaw angle.

## Methods

### *Dynamics Modeling*

Modeling of the nonlinear system dynamics was done following the methods of Beard. The nonlinear kinematic and dynamic equations used for controller design and MATLAB simulation are as follows.

$$\begin{pmatrix}\dot{p}_{\mathrm{N}}\\ \dot{p}_{\mathrm{E}}\\ \dot{h}\end{pmatrix}=\begin{pmatrix}c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi\\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi\\ s\theta & -s\phi c\theta & -c\phi c\theta\end{pmatrix}\begin{pmatrix}u\\ v\\ w\end{pmatrix}$$

$$\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix}=\begin{pmatrix}rv-qw\\ pw-ru\\ qu-pv\end{pmatrix}+\begin{pmatrix}-g\sin\theta\\ g\cos\theta\sin\phi\\ g\cos\theta\cos\phi\end{pmatrix}+\frac{1}{m}\begin{pmatrix}0\\ 0\\ -F\end{pmatrix}$$

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix}=\begin{pmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{pmatrix}\begin{pmatrix}p\\ q\\ r\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix}=\begin{pmatrix}\frac{J_y-J_z}{J_x}qr\\ \frac{J_z-J_x}{J_y}pr\\ \frac{J_x-J_y}{J_z}pq\end{pmatrix}+\begin{pmatrix}\frac{1}{J_x}\tau_\phi\\ \frac{1}{J_y}\tau_\theta\\ \frac{1}{J_z}\tau_\psi\end{pmatrix}$$

$(p_{\mathrm{N}}, p_{\mathrm{E}}, h)$ are the North, East, and elevation coordinates in the world frame, $(u, v, w)$ are the body velocities along the North-East-Downward frame, $(\phi, \theta, \psi)$ are the roll-pitch-yaw Euler angles, and $(p, q, r)$ are the body angular velocities along the North-East-Downward frame. $F$ is the upward input thrust, and $(\tau_\phi, \tau_\theta, \tau_\psi)$ are the input torques aligned to the Euler angles. $m$ is the quadrotor mass, $\mathrm{diag}(J_x, J_y, J_z)$ is the inertial tensor, and $g$ is the gravitational acceleration constant.

The physical states and control inputs of the quadrotor are defined in the orders below.

$$\frac{d\boldsymbol{x}}{dt}=\frac{d}{dt}[p_{\mathrm{N}} \quad p_{\mathrm{E}} \quad h \quad u \quad v \quad w \quad \phi \quad \theta \quad \psi \quad p \quad q \quad r]^T=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$$

$$\boldsymbol{u}=[F \quad \tau_\phi \quad \tau_\theta \quad \tau_\psi]^T$$

Full derivation of these equations, along with definitions of constants and state variables can be found in the paper by Beard. [1]

### *Trajectories/Reference Signals*

To evaluate tracking performance, three unique time-parameterized trajectories are generated and supplied to the system as a reference signal. The chosen signals were a ramp, helix, and Lissajous curve. Each reference signal is comprised of four sub-signals, each corresponding to one state of the system:

$$r(t) = \begin{bmatrix} p_{\mathrm{N}}(t) \\ p_{\mathrm{E}}(t) \\ h(t) \\ \psi(t) \end{bmatrix}$$

The ramp trajectory is selected to demonstrate that incorporating a single integrator into the design of the controller allows the system to asymptotically track a ramp input in the corresponding state. The helix trajectory is selected because it is comprised of constant-frequency sinusoids to the $p_{\mathrm{N}}$ and $p_{\mathrm{E}}$ states, with a ramp input to $h$. Finally, the Lissajous trajectory is selected as a challenge to the system. It is comprised of sinusoids in $p_{\mathrm{N}}$ and $p_{\mathrm{E}}$, a stepped input to $h$, and $\psi$ is selected to ensure the quadcopter maintains a constant forward heading (tangent to the curve).

## *Controller Design*

This controller design is largely inspired by Sawyer [2]. To accomplish more effective tracking performance, integrators were added to each of the states for which a reference signal is supplied. These derived states – denoted $\sigma_{p_{\mathrm{N}}}, \sigma_{p_{\mathrm{E}}}, \sigma_h$, and $\sigma_\psi$ – are incorporated into the state vector, resulting in a 16-state system. Linear state feedback control was then used to generate an optimal state feedback controller. The feedback gain matrix was generated by first linearizing the system about various equilibrium points, and then solving the infinite horizon linear-quadratic regulator (LQR) problem.

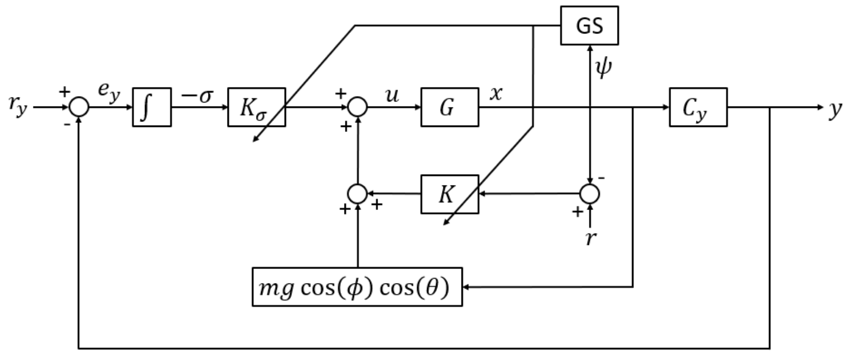

Figure 2. Block diagram of the full control structure, including the reference signal, integrator, and gain scheduling control updates.

Figure 2 shows the complete block diagram, including the integrator (denoted $\int$ ), the state feedback controllers (denoted $K$ and $K_\sigma$), the nonlinear plant (denoted $G$) and the gain scheduler (denoted GS). The control scheme also includes a feedback gravity compensation term which is scaled relative to the pitch and roll of the quadrotor.

## *Integrator*

Though a standalone LQR controller is capable of tracking trajectories, it is generally not able to do so without persistent tracking errors. To counteract this, integrators for the 3D position components and yaw angle are included to reduce, and in the case of ramp inputs eliminate, tracking error. The integrator dynamics are defined by the error signal.

$$\dot{\boldsymbol{\sigma}} = -\boldsymbol{e} = \boldsymbol{y}_{\boldsymbol{\sigma}} - \boldsymbol{r} = \boldsymbol{C}_{\boldsymbol{\sigma}}\boldsymbol{x} - \boldsymbol{r}$$

$$\boldsymbol{C}_{\boldsymbol{\sigma}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

*LQR Tuning*

To use the dynamics for LQR design, they first need to be linearized about an operating point. Since the 3D position and yaw angle need to track a reference, it is necessary to keep the linearization general with respect to with these parameters. All other states should be made 0 in order to achieve steady-state. However, as the dynamics do not depend on the 3D position, only the yaw angle becomes a parameter of the general linearized system. The augmented state dynamics with integrators is defined below.

$$\frac{d\boldsymbol{z}}{dt} = \frac{d}{dt}\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\sigma} \end{bmatrix} = \frac{d}{dt}[p_{\mathrm{N}} \quad p_{\mathrm{E}} \quad h \quad u \quad v \quad w \quad \phi \quad \theta \quad \psi \quad p \quad q \quad r \quad \sigma_{p_{\mathrm{N}}} \quad \sigma_{p_{\mathrm{E}}} \quad \sigma_h \quad \sigma_\psi]^T = \boldsymbol{\mathcal{F}}(\boldsymbol{z}, \boldsymbol{u})$$

Linearizing the physical system about an arbitrary yaw yields the following matrices.

$$\boldsymbol{A}(\psi) = \left.\frac{d\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}{d\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^*,\boldsymbol{u}=\boldsymbol{0}} = \begin{bmatrix} 0 & 0 & 0 & \cos\psi & -\sin\psi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sin\psi & \cos\psi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{B} = \left.\frac{d\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}{d\boldsymbol{u}}\right|_{\boldsymbol{x}=\boldsymbol{x}^*,\boldsymbol{u}=\boldsymbol{0}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -m^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & J_x^{-1} & 0 & 0 \\ 0 & 0 & J_y^{-1} & 0 \\ 0 & 0 & 0 & J_x^{-1} \end{bmatrix}$$

The state-space definition of $\boldsymbol{x}$ is augmented with that of $\boldsymbol{\sigma}$ for the LQR design.

$$\dot{\boldsymbol{z}} = \boldsymbol{\mathcal{A}}(\psi)\boldsymbol{z} + \boldsymbol{\mathcal{B}}\boldsymbol{u}$$
$$\boldsymbol{\mathcal{A}}(\psi) = \begin{bmatrix} \boldsymbol{A}(\psi) & \boldsymbol{0} \\ \boldsymbol{C}_{\boldsymbol{\sigma}} & \boldsymbol{0} \end{bmatrix}$$
$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix}$$

The LQR control is defined to minimize the infinite-horizon cost function.

$$J(\boldsymbol{u}) = \int_0^\infty \left(\boldsymbol{z}^T(t)\boldsymbol{Q}\boldsymbol{z}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}\boldsymbol{u}(t)\right) dt$$

Solving $\min_{\boldsymbol{u}} J(\boldsymbol{u})$ yields the optimal state feedback law $\boldsymbol{u}^* = -\boldsymbol{K}(\psi)\boldsymbol{z}$ about a particular operating point for the yaw. After tuning the weighting matrices such that the tracking performance is sufficiently good, the following weights were used.

$$\boldsymbol{Q} = \mathrm{diag}([50 \quad 50 \quad 500 \quad 1 \quad 1 \quad 5 \quad 10 \quad 10 \quad 20 \quad 10 \quad 10 \quad 1 \quad 500 \quad 100 \quad 100 \quad 10])$$
$$\boldsymbol{R} = \mathrm{diag}([1 \quad 150 \quad 120 \quad 20])$$

The LQR weighting matrices are tuned such that the quadrotor will not saturate the controls when the errors are not large. To ensure controls are within saturation limits, the components of $\boldsymbol{R}$ which correspond to the pitch and roll torques are set relatively large. Additionally, iterative tuning by simulating in MATLAB and PX4 enables the coordinates $p_\mathrm{N}$ and $p_\mathrm{E}$ to track sinusoids with minimal phase delay and amplitude gain.

## *Gain Scheduling Techniques*

To enable the quadrotor to track an arbitrary yaw angle, gain scheduling is used to extend the capability of LQR control. Three different scheduling techniques are used to design this portion of the controller. The scheduling variable is the yaw angle in all three methods. Due to the rotational symmetry of the quadrotor, the yaw is scheduled between 0° and 360° to encompass its entire range.

1. Switched Gain Scheduler

The first gain scheduling technique used is the switching controller. To design this scheduler, the dynamics are linearized about a collection of yaw operating points. For each operating point, a lower and upper bound on yaw determines the range over which the controller is applied. So long as the yaw is within this region, the corresponding fixed-gain state feedback gains are applied.

Since each linear controller has a restricted region of attraction around the operating point, the minimum number of operating points must be determined to ensure the quadrotor will be stable for any arbitrary yaw angle. Using the LQR gains finalized through the tuning process, it was found that the quadrotor could stabilize within $\pm 33.1°$ of the operating point for initial conditions sufficiently close to the desired reference coordinate.

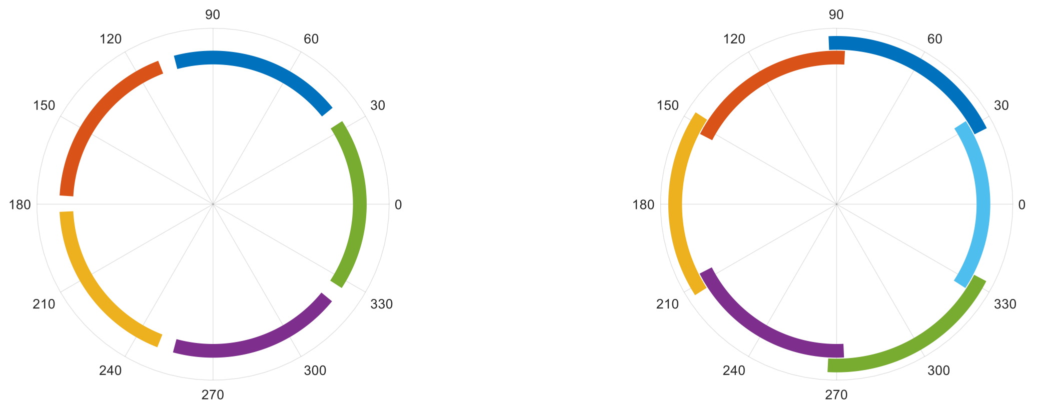

Figure 3. Visualization of the stabilizing regions of each fixed-gain LQR controller. 5 equally spaced operating points (*left*) do not meet the stability covering condition, however 6 do (*right*).

Based on this, each linear controller has a stabilizing range of 66.2°. So, at least 6 equally spaced operating points are needed to provide stability over the entire range of yaw for any desired value. This condition is visualized in Figure 3, showing that 5 operating points is not enough to ensure stability for any arbitrary yaw angle. Therefore, 6 or more operating points satisfies the stability covering condition. [3] Our designed switched gain scheduler uses 8 operating points, though a comparison to one using 16 operating points is discussed later.

The next step in designing the switched gain scheduler is determining the switching bounds. Due to the rotational symmetry of the quadrotor dynamics, and that the operating points are chosen to be equally spaced apart, the naïve choice is to have two adjacent controllers switch halfway between the operating points. While this may work well for a deterministic system, in the presence of noise and disturbances this can cause undesirable rapid switching back and forth between adjacent regions.

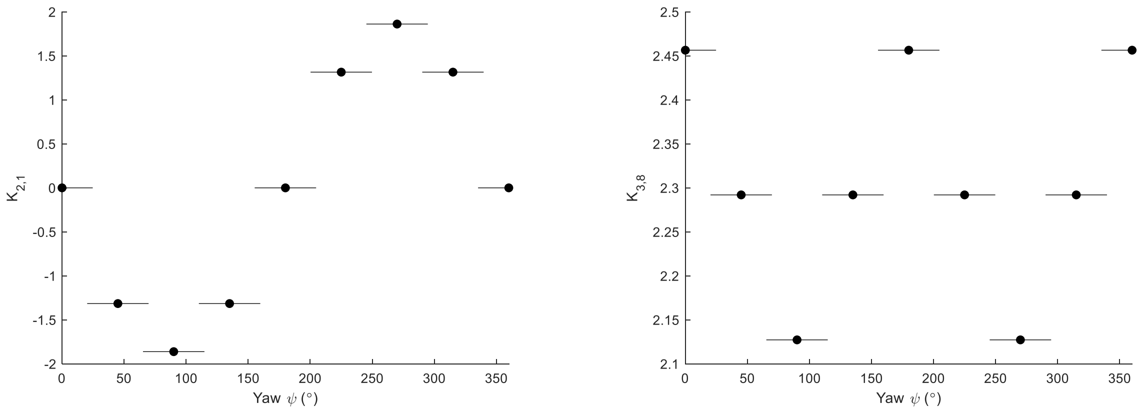

Figure 4. Switched state feedback gains for indices (2, 1) (*left*) and (3, 8) (*right*) with overlap. The black points represent the operating points.

To remedy this, each region can be extended such that the ends of adjacent operating regions overlap. So, when the yaw goes beyond the bound of one controlled region and switches to another, the closest bound is sufficiently far that noise and disturbances do not cause it to immediately switch back. Based on PX4 simulation results, each region is extended by 2.25° on either side, creating overlaps of 4.5° between regions. Two example elements of the tuned state feedback matrix as switched gains are shown in Figure 4. The outcomes of this choice as well as further switching improvements are discussed later.

2. Linearly Interpolated Gain Scheduler

A linearly interpolated gain scheduler is built upon the switched gain scheduler already designed. One issue which the switched gain scheduler faces is that it is a discontinuous controller. As currently designed, when the yaw angle enters an adjacent region the controller immediately switches to a new set of state feedback gains. This likewise causes discontinuous jumps in the control effort, which may be undesirable. Though there are direct methods to remedy this for the switched scheduler, an alternative solution is the linearly interpolated scheduler.

Rather than applying a single fixed-gain controller over a particular region, the interpolated controller mixes adjacent controllers depending upon the yaw value. More precisely, the linear state feedback gains designed at each operating point nearest the yaw angle are linearly interpolated to determine a unique set of state feedback gains for that particular yaw angle.

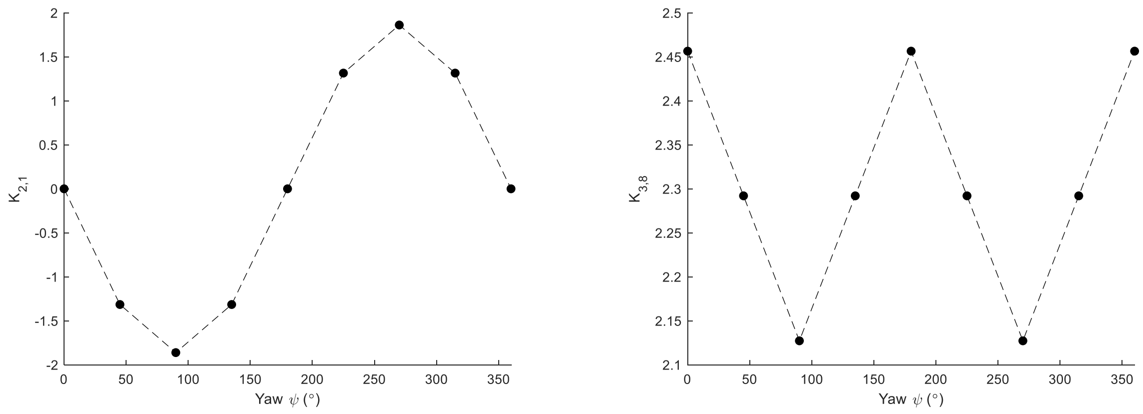

Figure 5. Linearly interpolated state feedback gains for indices (2, 1) (*left*) and (3, 8) (*right*). The black points represent the operating points.

By linearly interpolating between operating points, the designed controller is continuous. This will ensure that the applied control efforts will not have discontinuities due to switching gains. Two example elements of the tuned state feedback matrix as linearly interpolated gains are shown in Figure 5.

By interpolating the designed linear state feedback controllers, stability is no longer guaranteed by the stability covering condition. Instead, it must be shown that any interpolated controller has a region of stability such that the system will asymptotically converge to the setpoint. To demonstrate that the linearly interpolated controller provides asymptotic stability, simulations are used to test the controller. Here we test if for a reference yaw halfway in-between two operating points whether the two adjacent controllers designed at the operating points as well as the controller interpolated at the reference are all able to stabilize the system. If so, then any interpolated controller within this range should be capable of stabilizing the system for that reference.

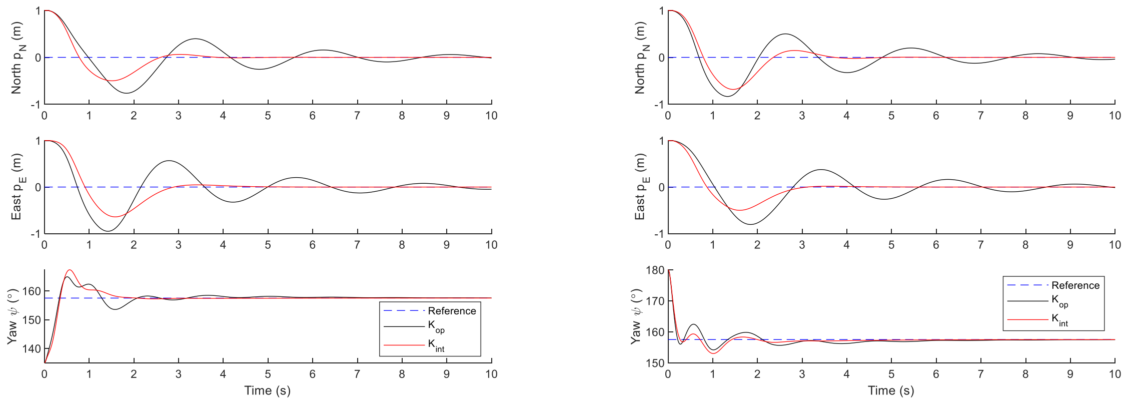

Figure 6. Stable trajectories controlled with fixed-gain state feedback controllers designed at the operating points (*black*) or interpolated between operating points (*red*). Plots are comparing controllers designed at operating point (*left*) $\psi = 135°$ and (*right*) $\psi = 180°$ to the controller interpolated at $\psi = 157.5°$.

Figure 6 shows one example of this, where a reference of $\psi = 157.5°$ is used, halfway in-between the operating points $\psi = 135°$ and $\psi = 180°$. On the left, the black trajectory is using the fixed-gain controller designed at $\psi = 135°$ and the red trajectory is using the fixed-gain controller interpolated at $\psi = 157.5°$. Starting from $(p_N, p_E, \psi) = (1,1,135°)$, both controllers are able to provide asymptotic stability towards the reference. Similarly, on the right, the black trajectory is using the fixed-gain controller designed at $\psi = 180°$ and the red trajectory is using the fixed-gain controller interpolated at $\psi = 157.5°$. Starting from $(p_N, p_E, \psi) = (1,1,180°)$, both controllers are again able to provide asymptotic stability towards the reference. This confirms that the steady-state behavior is unchanged by the gain scheduler [4]. Since all three controllers provide asymptotic stability, then any controller interpolated within this region will be able to stabilize the system for some set of initial conditions. To verify that the interpolated gain scheduling control provides stability over the entire range of yaw, this process is repeated for each pair of adjacent operating points. The same results are found for all pairs, so the linearly interpolated gain scheduling controller provides asymptotic stability for any yaw angle.

## 3. Continuously Smooth Gain Scheduler

Since 8 operating points is relatively course, patterns in the interpolated state feedback gains may be difficult to see. However, by increasing the number of operating points the gains appear to approach a continuous function. For a very large number of operating points, both the linearly interpolated and switched gain schedulers approach a scheduler which is a continuously smooth function of yaw. The rotational symmetry of the quadrotor dynamics is taken advantage of again for this gain scheduler. It is observed that every scalar element of the state feedback matrix which is not constant tends to follow a sinusoid of the form $K_{i,j} = A\sin(f\psi + \delta) + b$. Using enough operating points, a sinusoidal curve is fit to each non-constant element of the gain matrix.

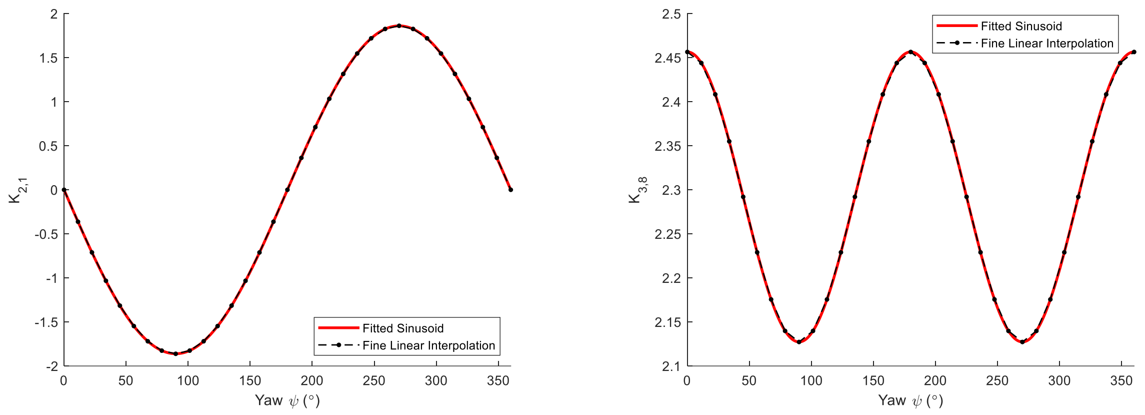

Figure 7. Continuously smooth state feedback gains for indices (2, 1) (*left*) and (3, 8) (*right*) and their linear interpolation approximations using 32 operating points.

Two example elements of the tuned state feedback matrix as smooth sinusoids are shown in Figure 7, along with corresponding linear interpolations of the designed gains using 32 operating points. At this finer resolution of operating points, it is clearer that the gains follow distinct sinusoids. The left image has a fit of $K_{2,1} = -1.861 \sin(\psi)$ and the right image $K_{3,8} = 0.165 \sin(2\psi + 90°) + 2.292$.

As with the linearly interpolated gain scheduler, the continuously smooth state feedback matrix is not guaranteed to be stable. With the continuously smooth gain scheduler, the concept of operating points for designing a collection of linear controllers does not exist, but the controller itself is a smooth continuum dependent upon the yaw. So, the same approach as before is no longer applicable. Instead, it must be shown that the system will be stable so long as the initial condition is close enough to the equilibrium and that the scheduling variable changes slowly. This will guarantee that the tracking error is uniformly bounded [4].

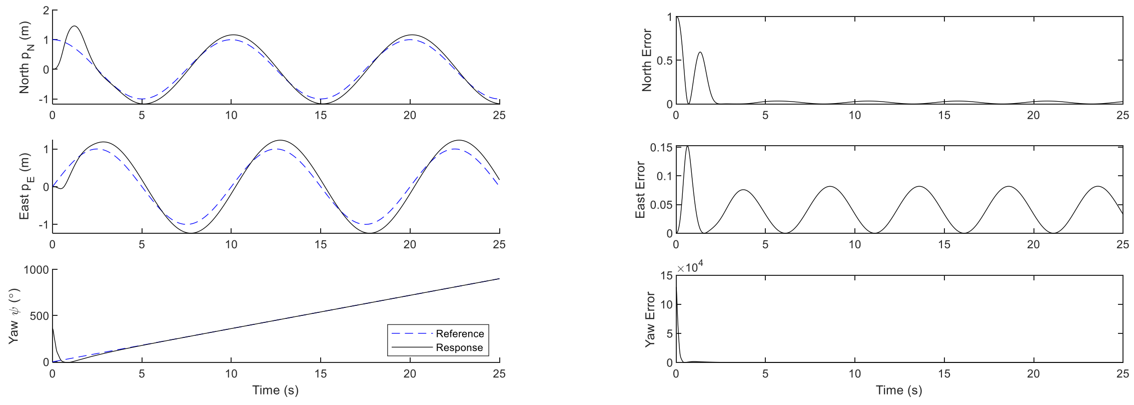

Figure 8. (*Left*) Tracking performance for linearly varying yaw such that the derivative is bounded. (*Right*) Bounded tracking errors for planar position and yaw.

Figure 8 demonstrates that the continuously smooth gain scheduler is stable. Since the yaw rate eventually becomes bounded as the yaw converges to the reference, the tracking errors for each state become bounded. This is true even when the quadcopter is initially a full rotation away from the reference yaw. This observation agrees with the idea that so long as the scheduling variable changes slowly enough, the controller will stabilize the system.

### *Gravity Compensation*

During the first part of the project, it was observed that an LQR controller for the states alone is not capable of regulating the quadrotor to the reference height without steady-state error. A simple feedforward thrust of $F_{\mathrm{GC}} = mg$ is sufficient to eliminate this error. With the addition of an integrator for the height, this feedforward is no longer necessary. However, it is still able to improve the transient performance of the quadrotor as well as eliminate a small nonlinearity.

Rather than compensating for gravity using a pure feedforward thrust, the compensation can be derived as a feedback component. The new gravity compensation thrust is defined below.

$$F_{\mathrm{GC}} = mg\cos(\phi)\cos(\theta)$$

Then the total thrust is defined as $F = F_{\mathrm{LQR}} + F_{\mathrm{GC}}$, where $F_{\mathrm{LQR}}$ is the thrust component calculated using the LQR state feedback. The gravity compensation will perform a cancellation in the downward velocity dynamics.

$$\begin{aligned}\dot{w} &= qu - pv + g\cos(\phi)\cos(\theta) - \frac{1}{m}F \\ &= qu - pv + g\cos(\phi)\cos(\theta) - \frac{1}{m}\left(F_{\mathrm{LQR}} + mg\cos(\phi)\cos(\theta)\right) = qu - pv - \frac{1}{m}F_{\mathrm{LQR}}\end{aligned}$$

Intuitively, including gravity compensation in a feedback manner allows it to scale the thrust magnitude according to the overall tilt of the quadrotor. If it is significantly tilted, then more of the thrust would be working against the planar components rather than counteracting gravity.

### *Simulation*

Simulation of the system is accomplished in two environments: PX4/jMAVsim and MATLAB. Simulation in MATLAB is handled by *ode45*, a non-stiff Runge-Kutta (4,5) based solver. Simulation in PX4/jMAVsim involves low-level C++ programming of quadcopter software that runs on a simulated quadcopter in a highly realistic environment. This environment includes factors like wind disturbance and simulated sensors. It is not uncommon to be able to achieve excellent controller performance in idealized environments like MATLAB but have poor controller performance in realistic tests. As a result, comparing results between the two environments became a point of particular interest for this project, where real-world results in hardware were not possible.

## Results

### *MATLAB and PX4 Simulation Comparison*

MATLAB and PX4 simulations were compared to investigate the extent to which the idealized simulations in MATLAB's *ode45* solver translate into more realistic environments. To compare these environments, a switched gain scheduler with 8 regions was simulated, and the trajectories were plotted. It was observed that the trajectory features periodic regions of divergence from the reference trajectory. Squared errors of each state were plotted against time to indicate at which points in time the simulated trajectory deviates from its reference.

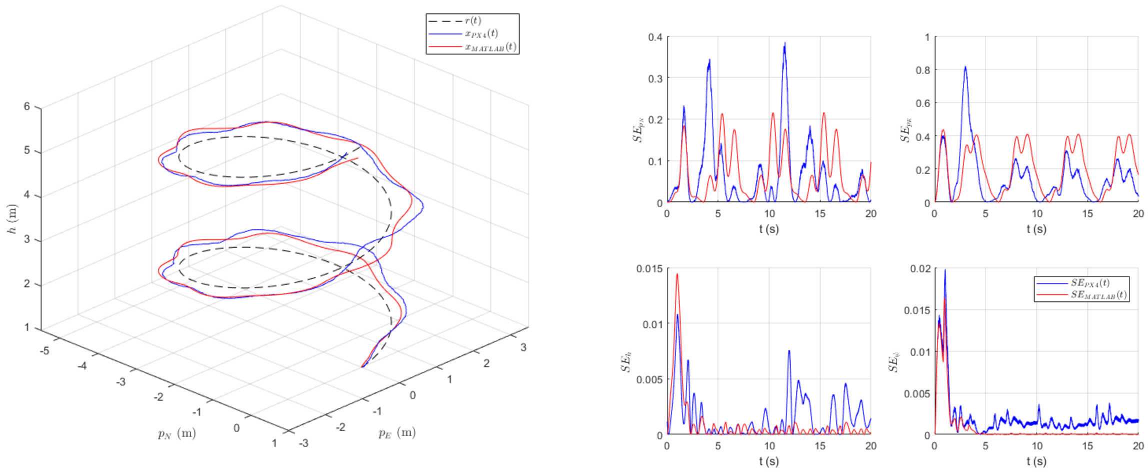

Figure 9. (*Left*) Trajectories in PX4 and MATLAB along with the commanded reference. (*Right*) Squared errors of each state over time.

From Figure 9, it is visually apparent that the three-dimensional spatial trajectories have similar features. In particular, the trajectories both diverge slightly from the nominal reference at consistent points along the trajectory (albeit with slight shifts and varying magnitudes). This is further supported by the squared errors. In most cases, the trends of the errors match closely. In the case of this 8-region switching gain scheduler, these deviations correspond to the portions of the trajectory where the controller is in-between operating points. In this region, the linearized approximation of the system dynamics is less accurate, and the controller is therefore inadequately designed. Additionally, the control signal switches in this region discontinuously, inducing transients in the response.

### *Impact of Region Quantity on Tracking Capability*

To evaluate the impact of the number of regions on the capability of the tracking controller the controller was tasked with tracking a helix. Switching and linearly interpolated gain schedulers were designed with the number of regions varying from 8 – 16. Each controller was simulated in MATLAB and then the mean squared tracking error was computed over the course of the run to indicate (on average) how much the quadcopter deviates from the commanded trajectory. Figure 10 shows the MSE of each critical position state against the number of regions. As the number of regions increases, the MSE monotonically decreases. However, rather than approaching zero, the

MSE trends towards some non-zero asymptote. This indicates that increasing the number of regions comes with marginal returns to tracking performance.

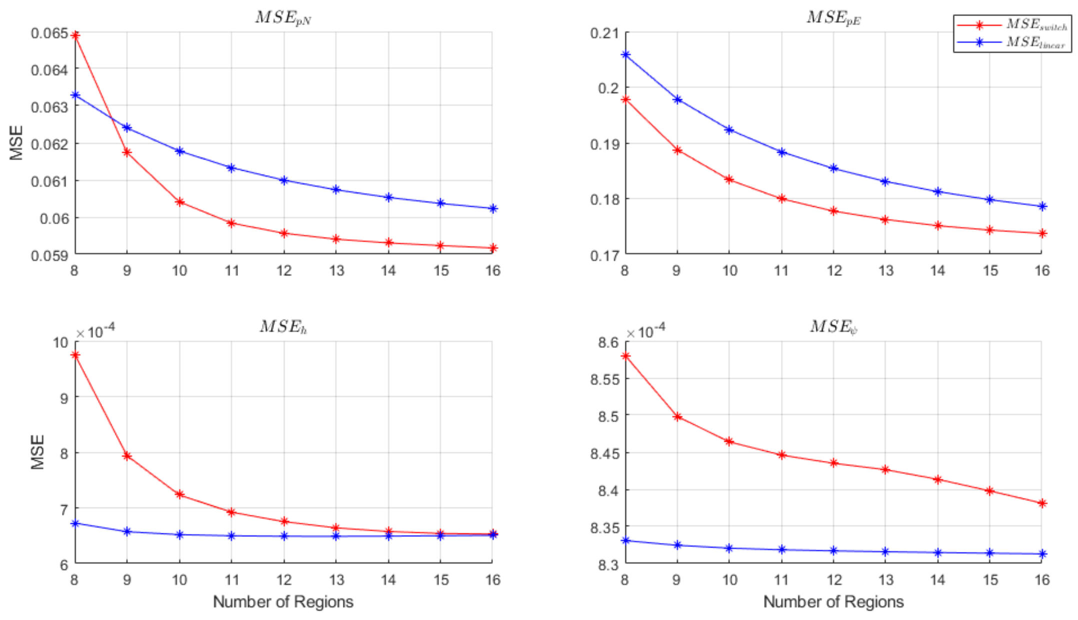

Figure 10. Mean squared error of $p_{\mathrm{N}}$, $p_{\mathrm{E}}$, $h$, and $\psi$ plotted against number of regions for both the switched and linearly interpolated gain schedulers. The controller is tasked with tracking a helix.

## *Comparison of Three Gain Scheduling Techniques*

The three discussed gain scheduling techniques were compared by simulating the performance of the controller tracking a helix in PX4. The three trajectories are depicted in Figure 11. A qualitative look at the trajectories indicates that all three controllers are stable in the execution of this task. However, the switching gain scheduler suffers periodically when it is in between operating points. As aforementioned, this is because the linearized approximation is least accurate in this region and the discontinuous switching of controller gains induces some oscillations in the system. The linearly interpolated and continuously smooth gain schedulers have visually similar performance.

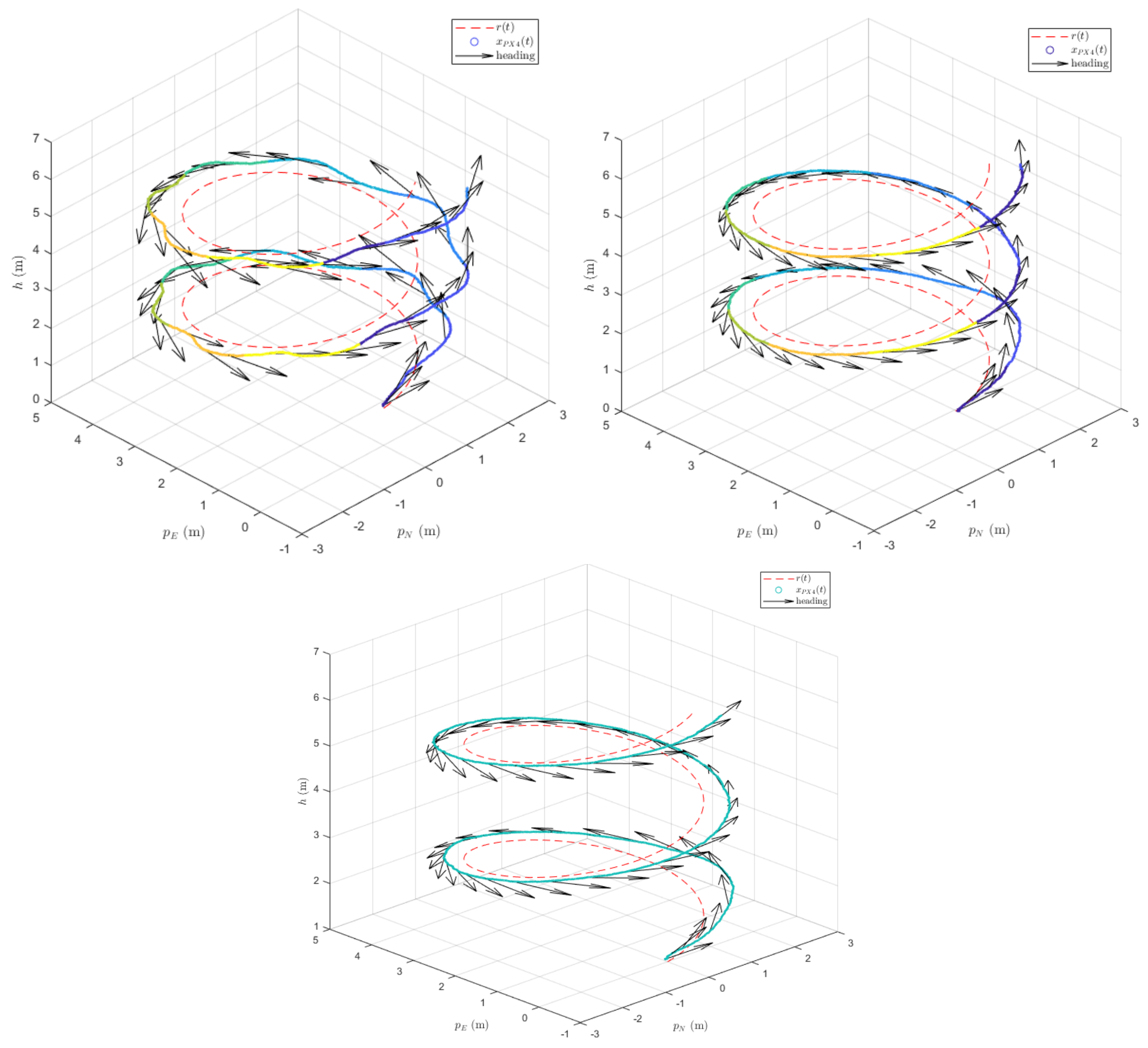

Figure 11. Gain scheduled tracking of three different tracking controllers: (*upper left*) 8-region switching, (*upper right*) 8-region linear, and (*bottom*) continuously smooth. Where applicable, color changes correspond to region switches.

The controllers can also be evaluated by looking at the squared tracking error as in Figure 12. For both $h$ and $\psi$, the squared error of all three controllers trends nearly to 0, with slight oscillations in the switched controller. In the case of the helix, both states are a ramp reference, so this result is unsurprising due to the inclusion of integral control. In $p_\mathrm{N}$ and $p_\mathrm{E}$, the squared error of all controllers oscillates. However, the linear and continuous controllers, the amplitude of the error is significantly lower than that of the switching controller.

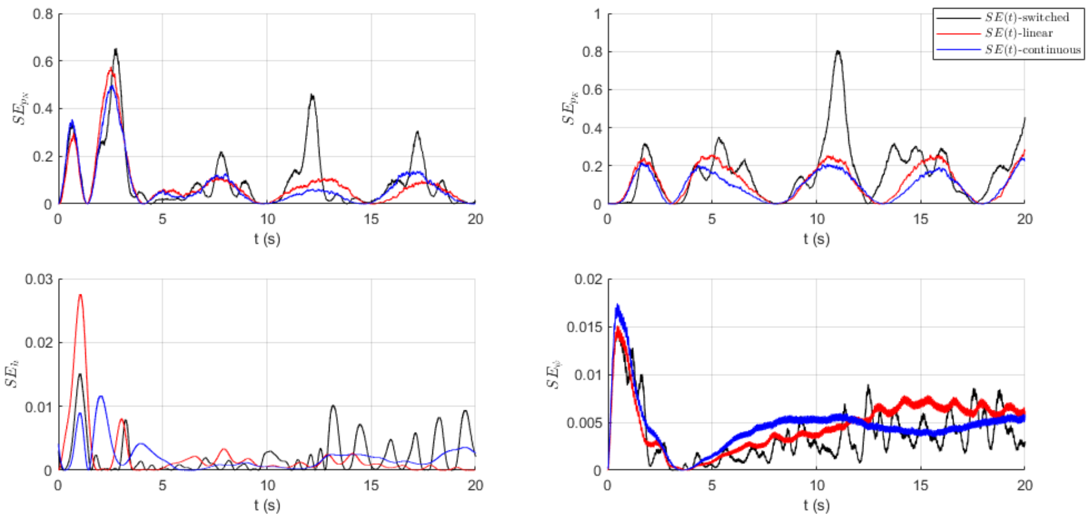

Figure 12. Squared error in $p_n, p_e, h$, and $\psi$ with the reference signal for each type of gain scheduled controller.

This is further demonstrated by examining MSE of each signal for each state as in Table 1. The continuously smooth gain scheduler exhibits the lowest MSE in $p_{\mathrm{N}}$ and $p_{\mathrm{E}}$. Interestingly, it does not have the lowest MSE in $h$ and $\psi$, but because each simulation was run at different times, it is likely that each run was subject to slightly different disturbances, as the wind disturbance in PX4 is randomly generated. The difference in variation in these two states is small enough to be attributable to differences in disturbances.

Table 1: MSE for each state in Figure 12.

| State | Switched | Linearly Interpolated | Continuously Smooth |
|---|---|---|---|
| $p_{\mathrm{N}}$ | 0.0988 | 0.0804 | 0.0763 |
| $p_{\mathrm{E}}$ | 0.1925 | 0.1288 | 0.0991 |
| $h$ | 0.0022 | 0.0019 | 0.0020 |
| $\psi$ | 0.0037 | 0.0048 | 0.0046 |

## *Evaluation of Gain Scheduling Controller on Various Trajectories*

Based on the results of the previous investigation, the continuously smooth gain scheduler was considered the most successful tracking control scheme. It was then evaluated on two further trajectories: a 3-dimensional ramp and a Lissajous curve with stepped height changes. It is clear from Figure 13 that the controller is capable of asymptotically tracking ramp inputs in $p_{\mathrm{N}}, p_{\mathrm{E}}$, and $h$ while regulating to stepped inputs in $\psi$ with roughly 30% overshoot.

Figure 14 shows the performance on the Lissajous curve. Again, the controller demonstrates some non-zero tracking error at all points but tracks the nonlinear reference signal in $p_{\mathrm{N}}, p_{\mathrm{E}}$ and $\psi$ closely while regulating to the stepped input to $h$ with roughly 25% overshoot.

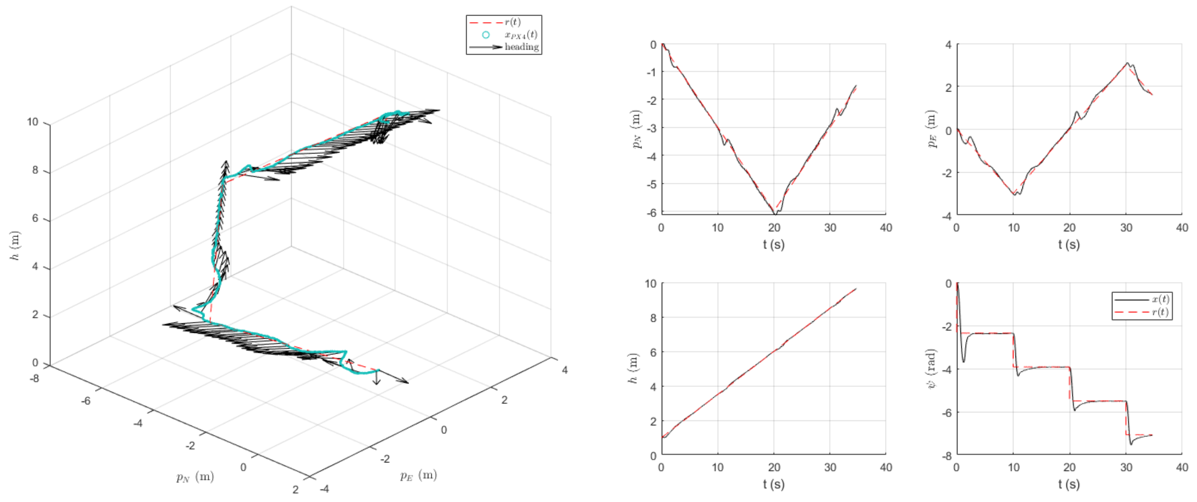

Figure 13. Continuously smooth gain scheduler tracking a ramp. (*left*) 3D trajectory with heading. (*right*) Tracked states vs. time.

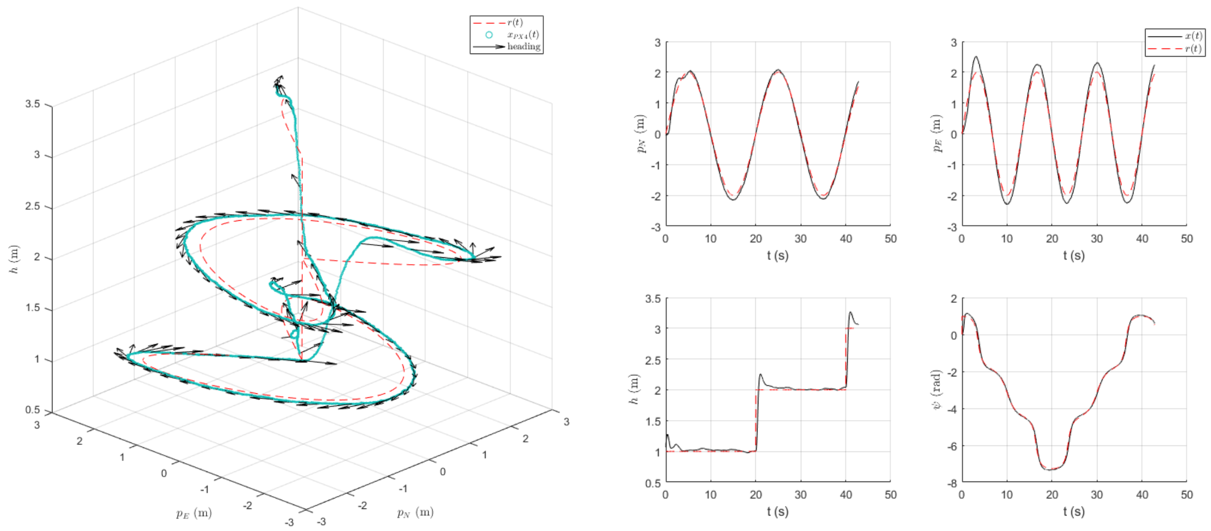

Figure 14. Continuously smooth gain scheduler tracking a Lissajous curve. (*left*) 3D trajectory with heading. (*right*) Tracked states vs. time

## Conclusions

These three methods of gain scheduling allow straightforward linear control principles to be applied to nonlinear systems. The switching gain scheduler is capable of tracking simple trajectories, but clearly suffers from the discontinuities induced when switching between regions. The linearly interpolated gain scheduler is a simple extension of the switching controller that removes these discontinuities. Additionally, when the gains at various operating points follow some discernable trend, curve fitting can be employed to produce gain matrices which are a continuous and smooth function of the state.

From our experiments in simulation, the continuous smooth gain scheduler allows for the best tracking performance. When it is not possible to smooth curves to the control gains, linear interpolation of gains is a powerful way of improving the discretely switched gain scheduler to remove the negative effects or instantaneous changes in control gains. In fact, simply interpolating the control gains resulted in a 18.4% and 31% reduction in MSE of $p_{\mathrm{N}}$ and $p_{\mathrm{E}}$, respectively, when tracking a helix. Smooth gain curve fitting allowed for a 22.8% and 48.5% reduction in the same states.

One acknowledgement must be made that it is possible the method of gain switching implemented in in our case could be improved to reduce the discontinuities induced by switching. Allowing the controller associated with the “next region” to begin working while in the overlapping region before switching to it could work to reduce the oscillations we observed at the switching points.

Performance may also be increased by simply increasing the number of regions. In the case of the quadrotor, this region increase comes with marginal gains. Therefore, it is imperative to make an application-specific choice as increasing the number of regions comes with a tradeoff cost of increasing the memory needed to store the implemented controller. In our application, there is only one scheduling variable, but in applications with several scheduling variables increasing the number of regions may result in an exponential cost to memory requirements.

## Challenges

Implementation of these gain scheduled controllers came with several challenges. The first is that a gain scheduler is only as effective as its underlying LQR controllers. Tuning of these controllers can require significant trial and error, and tradeoffs must be made with objectives such as response time, settling time, overshoot, and control saturation.

The bulk of the challenges relate to implementation of the controller in PX4. Accurate computation of the integral states is challenging. Simple Euler integration is used here, but more accurate methods could be employed at a cost to computation time. Secondly, coordination between the gain scheduler, the reference trajectory generator, and the control loop was not trivial. Further, different modules required different representations of yaw angle, and this proved to be trickier than expected. The gain scheduler expected $\psi \in [0, 360)$ while the reference generator and controller allowed $\psi \in (-\infty, \infty)$. Particularly in the case of the switching gain scheduler, this introduced challenges when the yaw angle transitioned from the region associated with 360° to the region associated with 0°.

## Appendix A – References

[1] Beard, "Quadrotor Dynamics and Control," *Brigham Young University*, 2008.

[2] Sawyer, "Gain-Scheduled Control of a Quadcopter UAV." University of Waterloo, 2015.

[3] Stilwell, Rugh, "Interpolation of Observer State Feedback Controllers for Gain Scheduling." IEEE Transactions on Automatic Control, vol. 44, no. 6, pp. 1225-1229, 1999.

[4] Khalil, "Nonlinear Systems." Prentice Hall, Upper Saddle River, ed. 3, 2002.