2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
June 28-30, 2023. Seattle, Washington, USA

Quasi-static state feedback output tracking for a slung load system with rotor drag compensation: PX4 SITL validation

Zifei Jiang, and Alan F. Lynch

***Abstract*— This paper presents a quasi-static state feedback (QSF) for motion control of a Slung Load System (SLS) which is a flat system consisting of a multirotor drone and suspended payload. The design exactly linearizes the closed-loop in new state coordinates. The linearizing feedback has an important static dependence on state and does not require a dynamic controller. After linearization, a straightforward output tracking control ensures that error dynamics in the design coordinates is linear and exponentially stable. The control design compensates for rotor drag and is validated in an open-source PX4 Software-in-the-Loop (SITL) simulation.**

I. INTRODUCTION

There has been an increase in interest in using drones for cargo transport [13]. A popular mode of transport is a multirotor drone Slung Load System (SLS), where a cable attached to the drone suspends the load. Benefits of the SLS include manoeuvrability and a safe distance between the payload and the vehicle. In addition, SLSs are lightweight and do not require powered computer control as with gripper-based solutions. Although SLSs are attractive for cargo transportation, creating a rigorous high-performance motion control is difficult due to underactuated nonlinear dynamics. Drone SLSs are a part of the recent trend of "unmanned aerial manipulation," which studies the interaction of aerial robots with their environment [13].

The concept of differential flatness was first presented in [3]. Flatness defines a so-called flat output which differentially parameterizes the input and state. Open-loop motion planning frequently employs this parameterization. Work [7] used flatness for open-loop trajectory planning for drone swarms and wheeled mobile robots. The nonlinear SLS model has been proven flat under common modelling assumptions in [10]. That paper uses flatness to perform open-loop motion planning and takes a different approach to feedback design. For traditional quadrotors, flatness-based feedback control is developed in [1], in which convincing experimental results are presented for aggressive trajectory tracking. Flatness-based control provides high performance with low computation compared to model predictive controllers (MPC) [12]. Although flatness has been used for open-loop trajectory planning for SLSs [10], it has not been used for *feedback control*. A main benefit of adopting a flatness-based closed-loop control is exact linearization of the error dynamics. This provides a simple and immediate stability result. This is in contrast to results in [10, 14] where nonlinear multi-loop dynamics complicate stability analysis.

Compensating for rotor drag force in multirotor motion control is an important problem [8, 6, 2]. Work [2] proved a quadrotor dynamics with drag force is still flat and proposed a flatness-based feedback control with drag compensation. There are no known results on drag force compensation for SLSs. Hence, our proposed design makes a contribution in this regard. Compared to our work [5], this paper extends the SLS model to include rotor drag. Another major extension relative to [5] is on the implementation side. We present an implementation of the proposed QSF in a PX4 software-in-the-loop (SITL) environment to validate performance. This is important for proving the developed controller can be implemented on typical autopilots and is robust to model error introduced in the SITL simulation environment. For example, the controller in [11, 10] involves complex expressions which are approximated for implementation. A video shows the practical performance of the proposed control in a SITL simulation `https://youtu.be/fcOKLIxspCw`.

II. SLS MODELLING

The suspended load is modelled as a pendulum attached to the drone's Center of Mass (CoM). Two reference frames are used: a navigation frame $\mathscr{N}$ fixed to the earth and a body frame $\mathscr{B}$ attached to the drone. We assume $\mathscr{N}$ is inertial and has an orthonormal basis $\{n_1, n_2, n_3\}$ of vectors oriented north, east, and down, respectively. The origin of $\mathscr{B}$ is the drone's CoM, and its basis $\{b_1, b_2, b_3\}$ has vectors oriented forward, right, and down, respectively. We denote pendulum position $p_L \in \mathbb{R}^3$, pendulum attitude $q \in \mathbb{S}^2$, and drone attitude $R \in SO(3)$. The SLS configuration variable is $[p_L, R, q] \in SE(3) \times \mathbb{S}^2$. The unit vector q is expressed in $\mathscr{N}$ and parameterized with angles α and β where α is a rotation about n_1 and β is about n_2. Load position p_L and quadrotor position p_Q are related by

$$p_L = p_Q + Lq = p_Q + L[s_\beta, -s_\alpha c_\beta, c_\alpha c_\beta]^T \tag{1}$$

where L is pendulum length. The UAV dynamics is

$$\dot{p}_Q = v_Q \tag{2a}$$
$$m_Q \dot{v}_Q = m_Q g n_3 - R\bar{u} n_3 + Tq - RDR^T v_Q \tag{2b}$$
$$\dot{R} = RS(\omega) \tag{2c}$$
$$J\dot{\omega} = -\omega \times J\omega + \tau \tag{2d}$$

where $J = \mathrm{diag}(J_1, J_2, J_3) \in \mathbb{R}^{3\times 3}$ is the inertia matrix for the drone, $\omega \in \mathbb{R}^3$ is drone angular velocity, $T \in \mathbb{R}_{\geq 0}$ is internal force in the pendulum, $\bar{u} \in \mathbb{R}_{\geq 0}$ is total thrust from the four

Zifei Jiang, and Alan F. Lynch are with the Department of Electrical and Computer Engineering, University of Alberta, AB, T6G 2R3, Canada. Email: {zifei.jiang, alan.lynch}@ualberta.ca
* Corresponding author. Email: alan.lynch@ualberta.ca

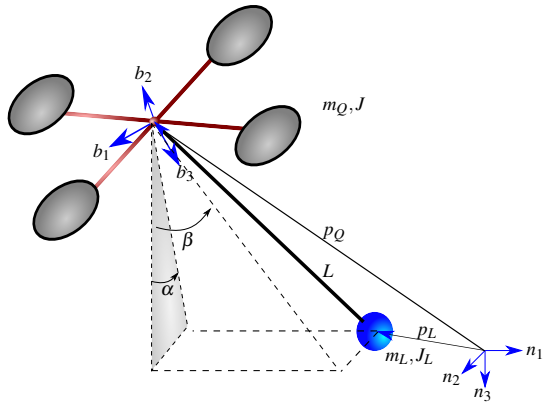

Fig. 1: SLS Modelling

rotors, $\tau \in \mathbb{R}^3$ is body torque, g is the acceleration of gravity, and m_Q is UAV body mass. Matrix $D = \text{diag}(d_x, d_y, d_z) \in \mathbb{R}^{3\times3}$ includes mass-normalized rotor drag coefficients. The rotor drag model assumes zero wind velocity. For details on rotor drag modelling see [8, 2, 6]. A ZYX Euler angle parameterization is used. We define $\eta = [\phi, \theta, \psi]^T \in \mathbb{R}^3$ and have

$$\dot{\eta} = W(\eta)\omega, \text{ with } W(\eta) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi/c_\theta & c_\phi/c_\theta \end{bmatrix} \tag{3}$$

where $t_\theta = \tan\theta, s_\theta = \sin\theta, c_\theta = \cos\theta$. The pendulum's dynamics is

$$\dot{p}_L = v_L \tag{4a}$$
$$m_L\dot{v}_L = -Tq + m_L g n_3 \tag{4b}$$
$$\dot{q} = \omega_L \times q \tag{4c}$$
$$J_L\dot{\omega}_L = -\omega_L \times J_L\omega_L + Lq \times (m_L g n_3 - m_L\dot{v}_Q) \tag{4d}$$

where $\omega_L \in \mathbb{R}^3$ is angular velocity of load in $\mathcal{N}$, and J_L is the load inertia matrix about the UAV CoM. For simplicity, we have assumed a pendulum with massless rod and point mass payload m_L. We can express J_L as

$$J_L = m_L L^2(I - qq^T) \tag{5}$$

Substituting for T in (4b) and (2b), we have

$$m_Q\dot{v}_Q + m_L\dot{v}_L = (m_Q + m_L)gn_3 - R\bar{u}n_3 - RDR^T v_Q \tag{6}$$

Eliminating $\dot{v}_Q$ in (6) using (1), we have

$$(m_L + m_Q)\dot{v}_L = (m_L + m_Q)gn_3 - R\bar{u}n_3 + m_Q L\ddot{q} - RDR^T v_Q \tag{7}$$

where $\ddot{q} = \dot{\omega}_L \times q + \omega_L \times \dot{q}$ from (4c) is used. Combining (7) and (1) yields

$$\dot{v}_Q = -\frac{m_L}{m_Q + m_L}L\ddot{q} + gn_3 - \frac{R\bar{u}n_3 + RDR^T v_Q}{m_Q + m_L} \tag{8}$$

Substituting (5), and (8) into (4d), the load rotational dynamics in $\mathcal{N}$ is

$$m_Q L(I - qq^T)\dot{\omega}_L = q \times (R\bar{u}n_3 + RDR^T v_Q) - m_L L(q^T\omega_L)\dot{q} + (m_Q + m_L)L(q^T\omega_L)\omega_L \times q \tag{9}$$

Angular velocity ω_L can be written in terms of $[\gamma_\alpha, \gamma_\beta] = [\dot{\alpha}, \dot{\beta}]$:

$$\omega_L = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\gamma_\alpha + \begin{bmatrix} 0 \\ c_\alpha \\ s_\alpha \end{bmatrix}\gamma_\beta, \dot{\omega}_L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha \\ 0 & s_\alpha & c_\alpha \end{bmatrix}\begin{bmatrix} \dot{\gamma}_\alpha \\ \dot{\gamma}_\beta \\ \gamma_\alpha\gamma_\beta \end{bmatrix} \tag{10}$$

Substituting (10) into (7) and (9), and solving for $\dot{v}_L, \dot{\gamma}_\alpha, \dot{\gamma}_\beta$, we obtain a state space form for the SLS dynamics

$$\dot{x} = f_1(x) + f_2(x) + g(x)u \tag{11}$$

where

$$f_1(x) = \begin{bmatrix} v \\ \gamma_\alpha \\ \gamma_\beta \\ W(\eta)\omega \\ -\frac{s_\beta(\gamma_\alpha^2 c_\beta^2 + \gamma_\beta^2)Lm_Q}{m_Q + m_L} \\ \frac{s_\alpha c_\beta(\gamma_\alpha^2 c_\beta^2 + \gamma_\beta^2)Lm_Q}{m_Q + m_L} \\ g - \frac{c_\alpha c_\beta(\gamma_\alpha^2 c_\beta^2 + \gamma_\beta^2)Lm_Q}{m_Q + m_L} \\ 2\gamma_\alpha\gamma_\beta t_\beta \\ -\gamma_\alpha^2 c_\beta s_\beta \\ -J^{-1}\omega \times J\omega \end{bmatrix}, g(x) = \begin{bmatrix} 0_{8\times1} & 0_{8\times3} \\ \bar{g}(x) & 0_{5\times3} \\ 0_{3\times1} & J^{-1} \end{bmatrix} \tag{12}$$

$$f_2(x) = [0_8, \bar{f}_2(x), 0_{3\times1}]^T \in \mathbb{R}^{16}$$
$$x = [p_L^T, \alpha, \beta, \eta^T, v_L^T, \gamma_\alpha, \gamma_\beta, \omega^T]^T \in \mathbb{R}^{16}$$
$$u = [\bar{u}, \tau^T]^T \in \mathbb{R}^4$$

and the expression for $\bar{g}$ is given in [5]. Rotor drag is modelled by $\bar{f}_2$ whose expression is too large to present here. The drift vector field of (11) has singularities at $c_\beta = c_\theta = 0$ due to parametrizations used for the orientation of the drone and pendulum.

III. Quasi-Static Feedback (QSF) Linearization

The goal of QSF is to statically linearize flat systems that do not satisfy the conditions for static state feedback linearization. Compared with dynamic state feedback, QSF is a static function of the state, i.e., it requires no state augmentation. Having a simpler controller structure is practically important when implementing onboard the autopilot where computing resources are limited.

The QSF begins with a flat output $y = [p_L^T, \psi]^T$ and dynamics (11). During the design we approximate the drag force by taking $RDR^T v_Q \approx Dv_Q$, this approximation is valid when $d_x \approx d_y \approx d_z$. Below, variables with superscript $^{\langle i \rangle}$ represent the step number of the QSF design. Each step of

the QSF design defines components of an auxiliary input $\nu \in \mathbb{R}^4$. Components of input ν are constructed in steps using Lie derivatives of the output so that an invertible relation between ν and original system input u is obtained.

Step 0**.** In Step 0 we verify $D^{\langle 0\rangle} = D$ has a constant rank less than 4 (the number of SLS inputs) in some neighbourhood of $x_0 = 0$, where $\ddot{y} = a_0(x) + D^{\langle 0\rangle}u$. We have

$$D^{\langle 0\rangle} = \begin{bmatrix} d_{11}^{\langle 0\rangle} & 0 & 0 & 0 \\ d_{21}^{\langle 0\rangle} & 0 & 0 & 0 \\ d_{31}^{\langle 0\rangle} & 0 & 0 & 0 \\ 0 & 0 & \frac{\mathrm{s}_\phi}{J_2\mathrm{c}_\theta} & \frac{\mathrm{c}_\phi}{J_3\mathrm{c}_\theta} \end{bmatrix} \tag{13}$$

where $d_{11}^{\langle 0\rangle}, d_{21}^{\langle 0\rangle}, d_{31}^{\langle 0\rangle}$ are functions of state, and $\mathrm{rank}(D^{\langle 0\rangle}) = 2$ about x_0 where

$$d_{31}^{\langle 0\rangle}(x) = -\frac{[\mathrm{s}_\beta, -\mathrm{s}_\alpha\mathrm{c}_\beta, \mathrm{c}_\alpha\mathrm{c}_\beta]\cdot Rn_3}{m_Q + m_L} \neq 0 \tag{14}$$

We decompose $y^{\langle 0\rangle} = \ddot{y}$ into independent $\tilde{y}^{\langle 0\rangle} = [\ddot{y}_3, \ddot{y}_4]^T$, and dependent $\hat{y}^{\langle 0\rangle} = [\ddot{y}_1, \ddot{y}_2]^T$ components. We introduce auxiliary input $v_1 = [\ddot{y}_3, \ddot{y}_4]^T$ so that

$$v_1 = \tilde{y}^{\langle 0\rangle} = \tilde{a}_0(x) + \tilde{b}_0(x)u \tag{15}$$

Then, $\hat{y}^{\langle 0\rangle}$ can be written in terms of v_1 and state:

$$\hat{y}^{\langle 0\rangle} = \begin{bmatrix} \left([1,0]^T v_1 - g\right)\mathrm{t}_\beta/\mathrm{c}_\alpha \\ \left(g - [1,0]^T v_1\right)\mathrm{t}_\alpha \end{bmatrix} \tag{16}$$

Step 1**.** Now that we have computed v_1, Step 1 computes additional independent auxiliary inputs from higher order Lie derivative of the output. The time derivative of $\hat{y}^{\langle 0\rangle}$ is

$$\dot{\hat{y}}^{\langle 0\rangle} = \begin{bmatrix} \frac{\dot{v}_1\mathrm{s}_\beta}{\mathrm{c}_\beta\mathrm{c}_\alpha^2} - \frac{\gamma_\beta(g-v_1)}{\mathrm{c}_\beta^2\mathrm{c}_\alpha} - \frac{\gamma_\alpha\mathrm{s}_\alpha\mathrm{s}_\beta(g-v_1)}{\mathrm{c}_\beta\mathrm{c}_\alpha} \\ \frac{\gamma_\alpha(g-v_1)}{\mathrm{c}_\alpha^2} - \dot{v}_1\mathrm{t}_\alpha \end{bmatrix} \tag{17}$$

Since the input does not appear in (17), we take another time derivative of $\hat{y}^{\langle 0\rangle}$ to get

$$\ddot{\hat{y}}^{\langle 0\rangle} = a_1(x, v_1, \dot{v}_1, \ddot{v}_1) + b_1(x, v_1, \dot{v}_1)u \tag{18}$$

where $a_1(x, v_1, \dot{v}_1, \ddot{v}_1) \in \mathbb{R}^{2\times 1}$, $b_1(x, v_1, \dot{v}_1) \in \mathbb{R}^{2\times 4}$. Matrix b_1 has the structure

$$b_1(x, v_1, \dot{v}_1) = \begin{bmatrix} b_{11} & 0 & 0 & 0 \\ b_{21} & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

We observe that both rows of (19) are linearly dependent on the third row of $D^{\langle 0\rangle}$. Hence, $\ddot{\hat{y}}^{\langle 0\rangle}$ can be expressed in terms of v_1. As a result, (18) can be written as $\ddot{\hat{y}}^{\langle 0\rangle} = \ddot{\hat{y}}^{\langle 0\rangle}(x, v_1, \dot{v}_1, \ddot{v}_1)$ with u eliminated. Similarly we eliminate u in $\left(\hat{y}^{\langle 0\rangle}\right)^{(3)}$ to obtain a function $\left(\hat{y}^{\langle 0\rangle}\right)^{(3)}(x, v_1, \dot{v}_1, \ddot{v}_1, v_1^{(3)})$. Calculating $\left(\hat{y}^{\langle 0\rangle}\right)^{(4)}$ we obtain

$$\left(\hat{y}^{\langle 0\rangle}\right)^{(4)} = \tilde{a}_1(x, \dot{v}_1, \ldots, v^{(4)}) + D^{\langle 1\rangle}u \tag{20}$$

where

$$D^{\langle 1\rangle} = \begin{bmatrix} d_{11}^{\langle 1\rangle} & d_{12}^{\langle 1\rangle} & d_{13}^{\langle 1\rangle} & 0 \\ d_{21}^{\langle 1\rangle} & d_{22}^{\langle 1\rangle} & d_{23}^{\langle 1\rangle} & 0 \end{bmatrix} \tag{21}$$

where $d_{ij}^{\langle 1\rangle}$ are functions of $(x, v_1, \dot{v}_1, \ldots, v_1^{(3)})$ with $\mathrm{rank}(D^{\langle 1\rangle}) = 2$ and its rows are linearly independent of all rows of $D^{\langle 0\rangle}$. We define auxiliary input

$$v_2 = \left(\hat{y}^{\langle 0\rangle}\right)^{(4)} \tag{22}$$

Combining (15), (22), we have an invertible relation between u and v

$$v = \tilde{a}(x, v_1, \dot{v}_1, \ldots, v_1^{(4)}) + D^{\circ}u \tag{23}$$

where

$$\tilde{a} = \begin{bmatrix} \tilde{a}_0(x) \\ \tilde{a}_1(x, v_1, \dot{v}_1, \ldots, v_1^{(4)}) \end{bmatrix}, D^{\circ} = \begin{bmatrix} d_{31}^{\langle 0\rangle} & 0 & 0 & 0 \\ 0 & 0 & \frac{\mathrm{s}_\phi}{J_2\mathrm{c}_\theta} & \frac{\mathrm{c}_\phi}{J_3\mathrm{c}_\theta} \\ d_{11}^{\langle 1\rangle} & d_{12}^{\langle 1\rangle} & d_{13}^{\langle 1\rangle} & 0 \\ d_{21}^{\langle 1\rangle} & d_{22}^{\langle 1\rangle} & d_{23}^{\langle 1\rangle} & 0 \end{bmatrix}$$

and D° is a nonsingular decoupling matrix.

By setting $v = [v_1, v_2]^T = [y_3^{(2)}, y_4^{(2)}, y_1^{(6)}, y_2^{(6)}]^T$ and using (23), a linearizing QSF is obtained. The tracking error is defined as

$$\begin{aligned} \tilde{z} &= [\tilde{z}_1, \ldots, \tilde{z}_{16}]^T \\ &= [y_1 - y_{d1}, \ldots, y_1^{(5)} - y_{d1}^{(5)}, y_2 - y_{d2}, \ldots, y_2^{(5)} - y_{d2}^{(5)}, \\ &\quad y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}, y_4 - y_{d4}, \dot{y}_4 - \dot{y}_{d4}]^T \end{aligned} \tag{24}$$

where $y_{di}, 1 \leq i \leq 4$ are desired outputs. Applying the linearizing control $u = D^{\circ -1}(K\tilde{z} - \tilde{a} + y_d^{(\bar{r})})$ with $y_d^{(\bar{r})} = [y_{d1}^{(6)}, y_{d2}^{(6)}, y_{d3}^{(2)}, y_{d4}^{(2)}]^T$ we obtain

$$\dot{\tilde{z}} = (A_c + B_c K)\tilde{z} \tag{25}$$

where $A_c \in \mathbb{R}^{16\times 16}, B_c \in \mathbb{R}^{16\times 4}$ are in Brunovsky Controller form, and $K \in \mathbb{R}^{4\times 16}$ is a control gain chosen so that $A_c + B_c K$ is Hurwitz and the closed-loop system has appropriate transient performance. Because $\dot{v}_1, \ldots, v_1^{(4)}$ in (23) are calculated using $y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}, y_4 - y_{d4}, \dot{y}_4 - \dot{y}_{d4}$ and their time derivatives, the controller depends only on x and the reference trajectory. Hence, it is a static state feedback. The expressions for $v_1, \dot{v}_1, \ddot{v}_1$ are

$$\begin{aligned} v_1 &= [\ddot{y}_{d3}, \ddot{y}_{d4}]^T - k_1[\tilde{z}_{13}, \tilde{z}_{15}]^T - k_2[\tilde{z}_{14}, \tilde{z}_{16}]^T \\ \dot{v}_1 &= [y_{d3}^{(3)}, y_{d4}^{(3)}]^T - k_1[\tilde{z}_{14}, \tilde{z}_{16}]^T - k_2(v_1 - [\ddot{y}_{d3}, \ddot{y}_{d4}]^T) \\ \ddot{v}_1 &= [y_{d3}^{(4)}, y_{d4}^{(4)}]^T - k_1(v_1 - [\ddot{y}_{d3}, \ddot{y}_{d4}]^T) - k_2(\dot{v}_1 - [y_{d3}^{(3)}, y_{d4}^{(3)}]^T) \end{aligned}$$

where k_1, k_2 are control gains from K. Similar expressions can be obtained for $v_1^{(3)}, v_1^{(4)}$. The set of point where the QSF is singular is discussed in [5].

IV. Matlab Simulation

In this section, the QSF is validated using Matlab simulation. We consider output tracking and stabilization. Our simulations compare QSF with and without drag force compensation. The system parameters used in the model and controller are $m_Q = 1.6\,\mathrm{kg}, m_L = 0.16\,\mathrm{kg}, L = 1\,\mathrm{m}, J_1 = J_2 = 0.03\,\mathrm{kg}\cdot\mathrm{m}^2, J_3 = 0.05\,\mathrm{kg}\cdot\mathrm{m}^2, d_x = 0.5\,s^{-1}, d_y = 0.4\,s^{-1}, d_z = 0.4\,s^{-1}$.

Stabilization: This section considers stabilization of the SLS at $x = 0$. We take initial position $p_L(0) = [2,2,2]^T$m and the remaining states are set to zero. Figs. 2 and 3 show the configuration variables and control input, respectively. For the stabilization task, the QSF performance with and without drag compensation (labelled "QSF Drag" and "QSF", respectively) is similar. Both controller show good transient convergence to the setpoint. This is to be expected as linear velocity and rotor drag converge to zero.

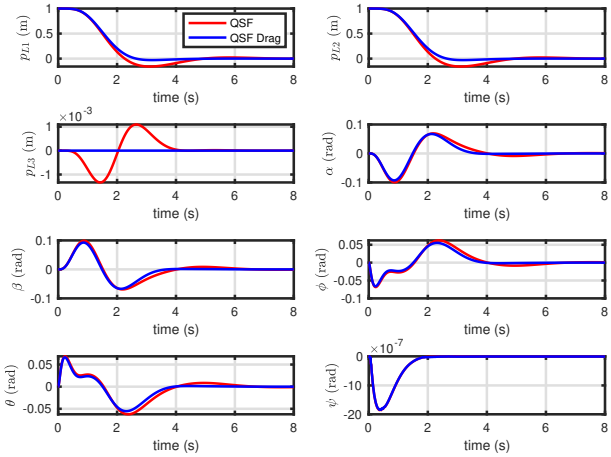

Fig. 2: System states p_L, α, β, η.

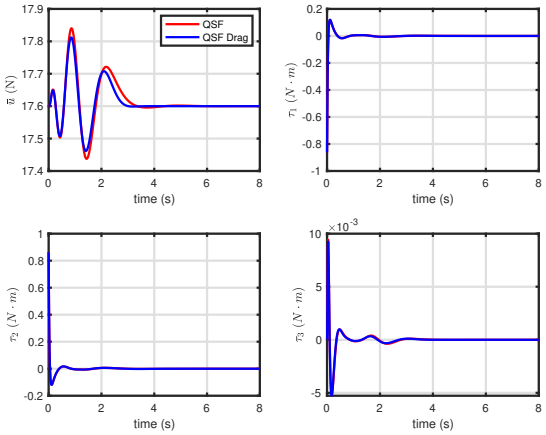

Fig. 3: Inputs $\bar{u}, \tau$.

Trajectory Tracking: The tracking error for complex reference trajectories is exponentially stable with QSF control. A "figure-8" reference is given as

$$y_d(t) = [3\sin(\pi t/4), 1.5\sin(\pi t/2), 0.5\sin(\pi t/4) - 9, 0.02t]^T \tag{26}$$

Fig. 4 and Fig. 5 show the tracking errors and configuration variables, respectively. The input trajectories are in Fig. 6. When drag is compensated, the tracking error shows exponential convergence with good transient performance. The trajectories of inputs remain within a feasible range. The gain for the designs were obtained using LQR with $Q = \text{blockdiag}(Q_1, Q_2, Q_3, Q_4), R = 0.1 \cdot I_4$, where $Q_1 = Q_2 = \text{diag}(100, 100, 100, 1, 1, 1), Q_3 = Q_4 = I_2$. As shown in Fig. 4, the tracking errors of QSF without drag compensation do not converge to 0 in the x and y directions.

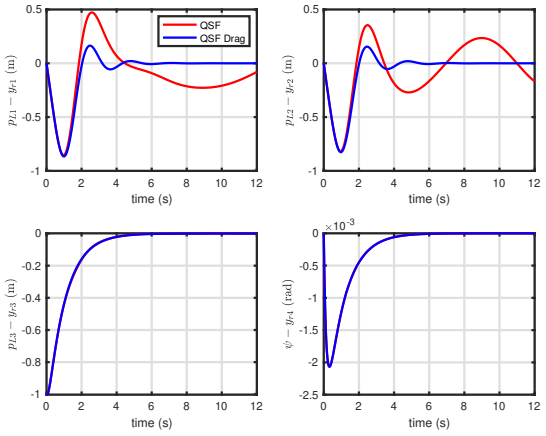

Fig. 4: Output tracking error $y - y_d$.

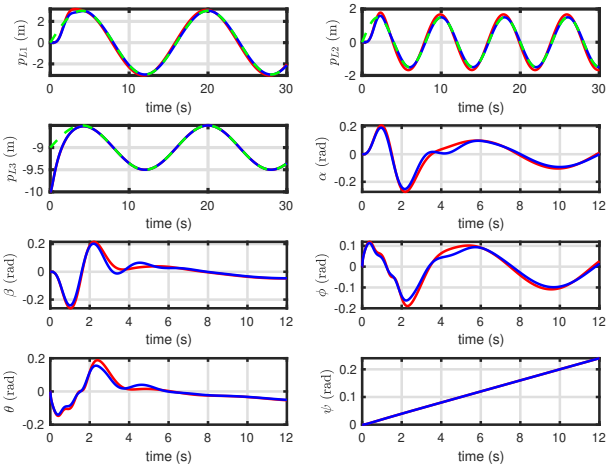

Fig. 5: System states p, α, β, η. The red line is QSF, the blue line is QSF with rotor drag compensation, and the green dotted line is the reference.

V. PX4 SITL Simulation

This section presents Software-In-The-Loop (SITL) simulation of the QSF. SITL emulates a real autopilot environment so that performance can be validated close to flight conditions. SITL ensures robustness to unmodeled effects such as controller saturation, multi-rate sampling, and computational delay. We choose the open source PX4 SITL framework [9] with the Gazebo simulator using the open dynamics engine (ODE) as the physics engine. Unlike the Matlab simulation in Section III where the SLS model directly uses the differential equations (DEs) (11), a Gazebo model requires no DE model, but rather based on the geometry and inertial properties of the system's bodies. The PX4-Gazebo SITL simulation leverages the RotorS simulation plugins for UAV modelling [4].

A. QSF SITL Simulation Pipeline

Model-based controllers such as the QSF benefit from precise theoretical statements about their stability and performance. Their mathematical derivation means they can be extended systematically to different applications. However, they can suffer from complex expressions. The QSF proposed in this paper uses Maple to compute the high-order Lie

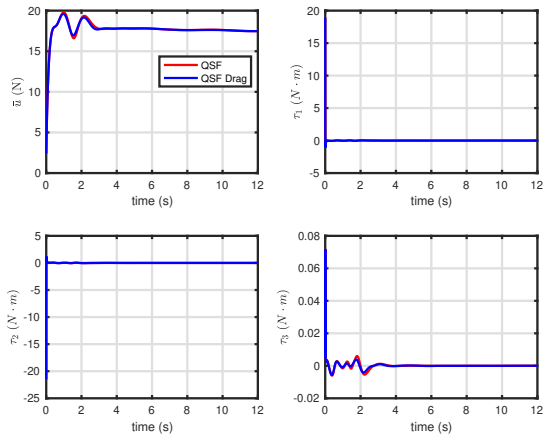

Fig. 6: Inputs $\bar{u}, \tau$.

derivatives required. The resulting controller expression is complex and must be automatically moved from Maple to C++ to avoid errors and streamline debugging. Therefore, we developed a Maple-Matlab-SITL pipeline for controller development. First, we do all symbolic calculations, including system modelling and the QSFA in Maple. Next, the resulting symbolic expression for the system model and controller are exported to Matlab for efficient simulation. Finally, the Matlab controller is exported to C++ for SITL using Matlab's `Coder Toolbox`.

B. Gazebo Simulator for Drones

The output of the QSF is total thrust $\bar{u}$ and torque τ in units of N and N$\cdot$m, respectively. The relation between $[\bar{u}, \tau]$ and rotor speed Ω is

$$\begin{bmatrix} \bar{u} \\ \tau \end{bmatrix} = C_T \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\ell_1 & \ell_3 & \ell_1 & -\ell_3 \\ \ell_2 & -\ell_4 & \ell_2 & -\ell_4 \\ C_M & C_M & -C_M & -C_M \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{27}$$

where $\Omega_i, 1 \le i \le 4$ is the ith rotor speed, C_T is the rotor thrust constant, and C_D is the torque constant. We take $C_T = 5 \times 10^{-6}\,\mathrm{N \cdot s^2}$ and $C_M = 0.05\,\mathrm{m}$. In this paper, we choose a 3DR Iris quadrotor whose geometry is described by $\ell = [0.22, 0.13, 0.2, 0.13]$ m.

The PX4 expects a normalized thrust $\widetilde{u} \in [0,1]$ and torque $\widetilde{\tau} \in [-1,1]$. PX4 converts $\widetilde{u}, \widetilde{\tau}$ with its Mixer into normalized rotor speed commands $\widetilde{\Omega}_i, 1 \le i \le 4$, using parameters describing the geometry of the multirotor's frame. Gazebo receives a normalized rotor speed command $[\widetilde{\Omega}_1, \widetilde{\Omega}_2, \widetilde{\Omega}_3, \widetilde{\Omega}_4]$ which it scales to obtain a rotor speed command $\Omega_i = C_\Omega \widetilde{\Omega}_i, 1 \le i \le 4$, where $C_\Omega = 1000$ rad/s is the scaling constant. Assuming that modelling is known exactly, we can scale the QSF output before feeding it to PX4 so that Gazebo applies the desired values of $\bar{u}$ and τ. Given $\bar{u}, \tau = [\tau_1, \tau_2, \tau_3]^T$ output from the QSF in (23) we scale $[\widetilde{u}, \widetilde{\tau}^T]$ as $[\bar{u}/S_1, \tau_1/S_2, \tau_2/S_3, \tau_3/S_4]$, where $S_1 = 20, S_2 = 1.8383, S_3 = 1.8383, S_4 = 0.9546$, so that the Gazebo simulation receives the correct control command from PX4.

C. Simulation Results

A video of the stabilization and time-varying tracking discussed in this subsection is at `https://youtu.be/fcOKLIxspCw`. All simulations compensate for rotor drag. Wind velocity is set to zero in the simulator.

Stabilization: Stabilization is performed with the set point $y_d = [0,0,-10,0]^T$ m. The quadrotor takes off with the built-in PX4 motion controller and commanded to a position away from the set point. The QSF is activated once the SLS is at rest. The configuration variables are in Fig. 7 with the QSF activated at $t = 138\,\mathrm{s}$. Good stabilization performance is achieved with a well-damped transition to the setpoint in about 4 s. This performance is similar to the stabilization of "QSF Drag" in Matlab given in Section IV. Corresponding input trajectories are given in Fig. 8 which are physically realizable and unsaturated.

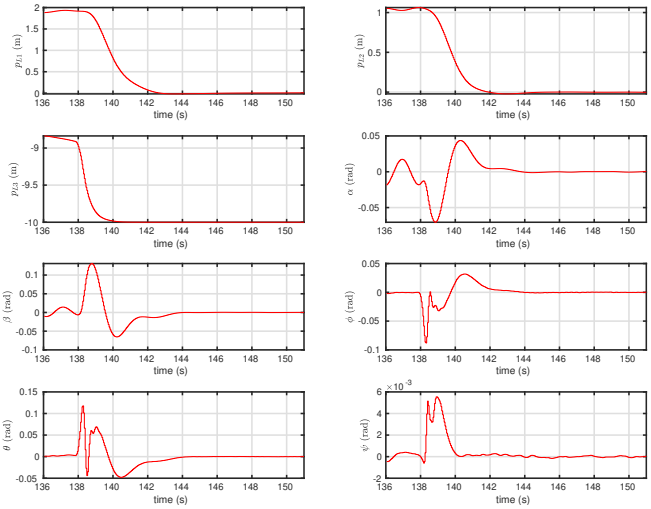

Fig. 7: Stabilization SITL simulation: system states p, α, β, η.

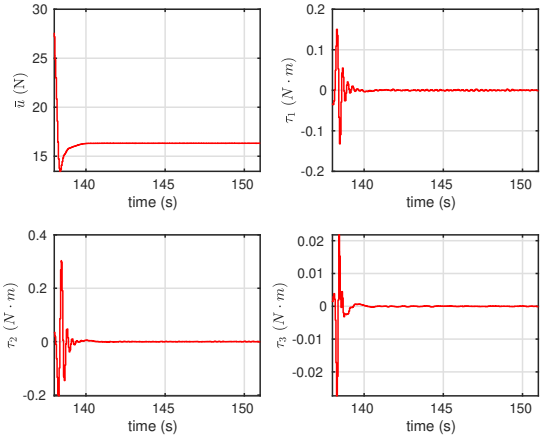

Fig. 8: Stabilization SITL simulation: inputs $\bar{u}, \tau$.

Trajectory Tracking: One of the advantages of the QSF is its guaranteed tracking performance for any smooth bounded reference trajectory. We reuse trajectory (26). As shown in Fig. 9 and Fig. 10, the tracking error converges to zero exponentially with a good transient performance. The control input remains unsaturated as shown in Fig. 11.

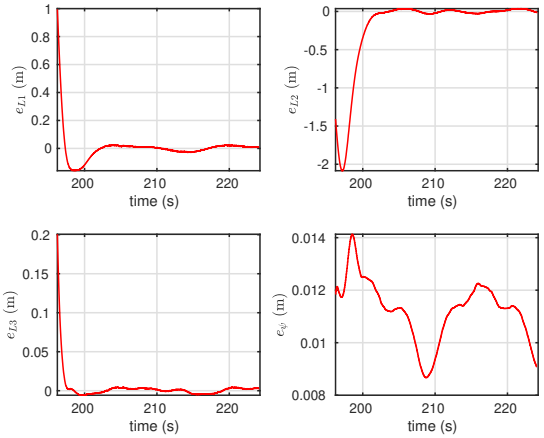

Fig. 9: Output tracking SITL simulation: tracking error $e = y - y_d$.

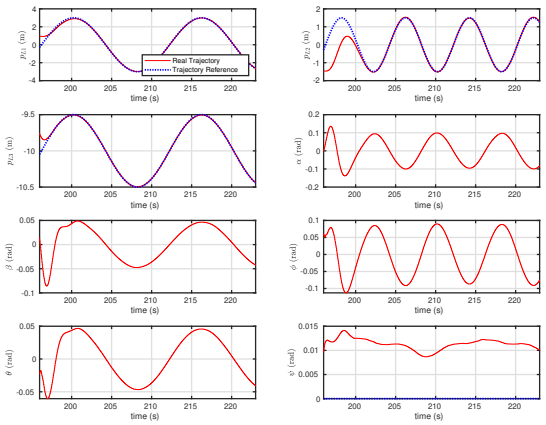

Fig. 10: Output tracking SITL simulation: SLS states p, α, β, η.

VI. Conclusions

This paper presented a QSF for the SLS which compensates for rotor drag force. The QSF achieves LTI exponentially stable tracking error dynamics on a well-defined and practical region of state space. Matlab simulation shows the importance of compensating rotor drag in order to achieve output tracking. In order to demonstrate the proposed QSF can be implemented on a real autopilot, we present a software pipeline for implementing a PX4/Gazebo SITL simulation. SITL simulation is important for developing flyable controllers as it proves the design is robust to unmodelled effects and can be implemented on autopilot hardware.

References

[1] M. Faessler, A. Franchi, and D. Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett.*, 3(2):620–626, 2017.

[2] M. Faessler, A. Franchi, and D. Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett.*, 3(2):620–626, 2017.

[3] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *Int. J. Control*, 61(6):1327–1361, 1995.

[4] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. *Robot Operating System (ROS)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer, 2016.

[5] Z. Jiang, M. Al Lawati, A. Mohammadhasani, and A. F. Lynch. Flatness-based motion control of a uav slung load system using quasi-static feedback linearization. In *Proc. ICUAS*, pages 361–368, 2022.

[6] J.-M. Kai, G. Allibert, M.-D. Hua, and T. Hamel. Nonlinear feedback control of quadrotors exploiting first-order drag effects. *IFAC-PapersOnLine*, 50(1):8189–8195, 2017.

[7] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[8] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robot. Autom. Mag.*, 19(3):20–32, 2012.

[9] L. Meier. Px4/px4-autopilot: Stable release v1.13.0, June 2022.

[10] K. Sreenath, T. Lee, and V. Kumar. Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load. In *Proc. IEEE CDC*, pages 2269–2274, 2013.

[11] K. Sreenath, N. Michael, and V. Kumar. Trajectory generation and control of a quadrotor with a cable-suspended load - a differentially-flat hybrid system. In *Proc. IEEE ICRA*, pages 4888–4895, 2013.

[12] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza. A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight. *IEEE Trans. Robot.*, 2022.

[13] D. K. D. Villa, A. S. Brandão, and M. Sarcinelli-Filho. A survey on load transportation using multirotor UAVs. *J. Intell. Robot. Syst.*, 98(2):267–296, 2020.

[14] S. Yang and B. Xian. Exponential regulation control of a quadrotor unmanned aerial vehicle with a suspended payload. *IEEE Trans. Contr. Syst. Technol.*, 28(6):2762–2769, 2020.

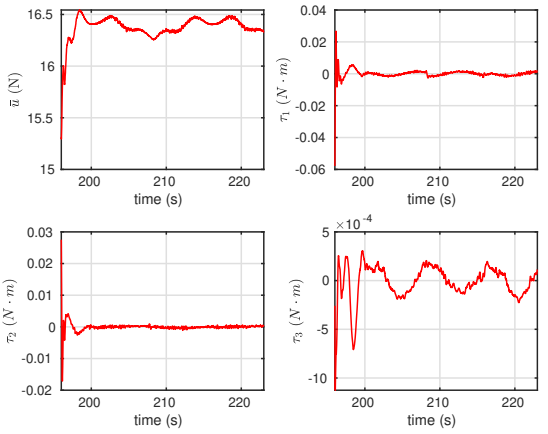

Fig. 11: Output tracking SITL simulation: inputs $\bar{u}, \tau$.