Perching Maneuver for Flapping-Wing Micro-Aerial Vehicles

Christian M. Chan

***Abstract*—The task of landing on various surfaces is challenging for flapping-wing micro-aerial vehicles (FWMAV) due to their inherently unstable, non-linear dynamics. This work presents a control solution for the Harvard RoboBee (a 90 mg FWMAV) to fly towards a wall, perform a flip-turn maneuver, and land/perch on the wall in simulation. We build upon a closed-form dynamic model of the RoboBee and leverage non-linear trajectory optimization techniques to achieve this aggressive aerial maneuver. First, an open-loop, nominal trajectory was constructed for the vehicle to follow via the direct collocation method. The nominal open-loop trajectory is then locally linearized using a finite-horizon, time-varying linear quadratic regulator controller to stabilize the trajectory about the perching position. The results are simulated in Drake to demonstrate the controller subject to the RoboBee dynamics.**

***Index Terms*—Aerial systems: mechanics and control, biologically-inspired robots, micro/nano robots.**

I. Introduction

In nature, we observe flapping-winged insects to frequently land and take-off from a variety of surface conditions for executing tasks such as scavenging for food, pollinating plants, escaping danger, or finding mates. Over two decades ago, a sub-field of robotics interested in mimicking the behaviour of flapping-winged insects was conceived [1]. With insect-inspired flapping-wing micro-aerial vehicles (FWMAV), applications such as environmental monitoring, artificial pollination, search and rescue, and swarm control for collective robotic intelligence can be achieved. Currently, one of the most promising insect-scale FWMAV platforms is the piezoelectric (PZT) actuator powered Harvard RoboBee [2]. Thus-far, the RoboBee has been shown to achieve stable, controlled hover flight through the use of a geometric tracking controller on $SE(3)$, as typically used on quadrotor vehicles [3][4]. However, landing for the RoboBee is a challenging problem.

Not only are insects able to interact with a variety of surfaces, but they typically approach these surfaces for landing with high velocities, undergoing aggressive aerobatic maneuvers while still able to stick the landing. On the contrary, the RoboBee is constrained to well-controlled flight regimes, avoiding aggressive maneuvers and contact and collisions with surfaces. The reason for this is to protect the PZT actuators from collision-induced stresses that would induce and/or propagate surface crack defects, or worse, fracture the brittle actuators such that the vehicle is no longer operational. Additionally, when hovering near surfaces in preparation for landing, the air vortexes produced by the RoboBee's flapping-wing motion that provides lift causes turbulent interactions with the ground called the ground effect. This ground effect exacerbates the vehicle's already unstable dynamics, making it difficult for the vehicle to land safely (with an almost ideal quasi-static zero velocity). The current solution for landing for the RoboBee is the drop landing method, where the vehicle is controlled to hover just above the turbulent ground effect region and dropped over the desired target landing position by turning off power and control. However, this hover-and-drop landing method is only suitable for flat, horizontal surfaces. If we wish to push the capabilities of FWMAVs towards mimicry of real insects that are capable of landing on more exotic surfaces and orientations, we must explore controllers for achieving aggressive, dynamical maneuvers for landing as observed in nature with insects.

In this work, we are interested in developing a controller that enables FWMAVs to mimic insects' ability to perch on a vertical surface as illustrated in Fig. 2. We present a

C. M. Chan was with the Department of Harvard Microrobotics Laboratory, John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02138 USA. e-mail: (christianchan@g.harvard.edu).

Manuscript received May 16, 2023.

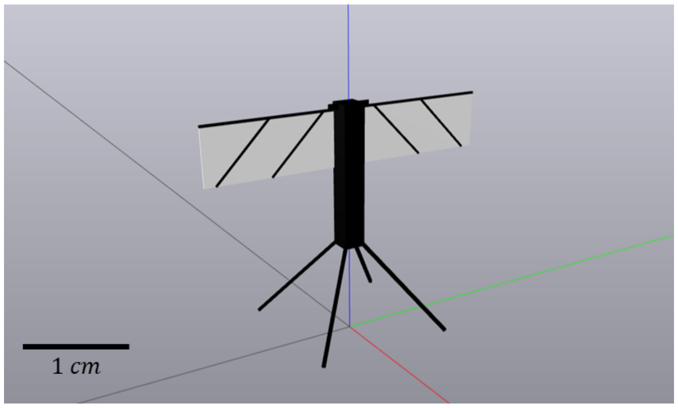

Fig. 1. RoboBee URDF model in Drake Meshcat.

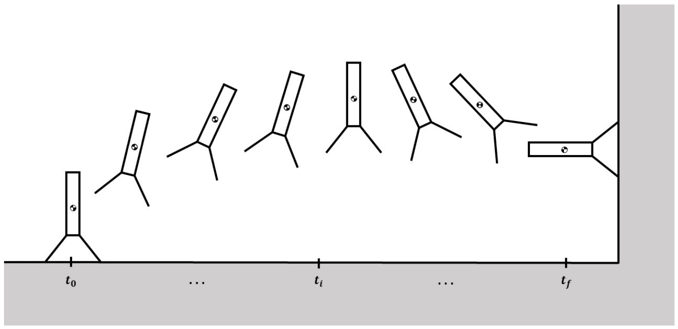

Fig. 2. Illustration of the perching maneuver trajectory. The RoboBee model is simplified to 2D for the trajectory optimization problem.

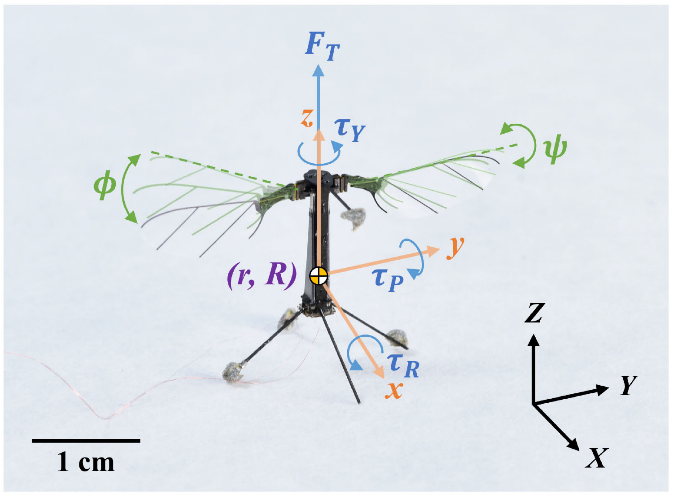

Fig. 3. RoboBee free-body diagram from [4] showing the parameters used to derive the flapping-wing lift and drag models, and thrust and torque forces.

controller that commands the RoboBee to fly towards a wall, perform a flip-turn maneuver, and land/perch on the wall. Previous work has been conducted on perching for macro-scale gliding vehicles such as in [5][6][7]. Additionally, past work on the RoboBee has explored perching through the use of electrostatic adhesion to stick to a variety of surfaces [8] and perching on a wall through iterative learning-based control with the aid of magnetic adhesion [9]. Here, we wish to achieve the results of the magnet-assisted learning-based perching control of the RoboBee in [8] using the trajectory optimization methods used in [7] and demonstrate them in simulation. We first summarize the closed-form model of the RoboBee as derived in [4] to define the required input voltages for controlling the vehicle state. Then, we design a target nominal trajectory in terms of vehicle state and voltage inputs for the perching maneuver using direct collocation. Finally, we stabilize the vehicle about the perching target position with a closed-loop finite-horizon, time-varying linear quadratic regulator (TVLQR) controller.

II. Vehicle Model

This section summarizes the lift and drag models of the flapping-wing dynamics of the RoboBee as well as the body dynamics of the RoboBee as defined by [4]. There are three main assumptions made by [4] to attain a closed-form model of the RoboBee.

Assumption 1: The flapping-wing dynamics and the body dynamics of the RoboBee are decoupled in order to simplify the model. This assumption is justified because the weight of the wings is negligible compared to the weight of the body (actuators and airframe). The wings weigh $\sim 1\%$ of the total weight of the vehicle.

Assumption 2: The flapping-wing dynamics are approximated as cycle-averaged blade elements to simplify the model. The body dynamics are dependant on the cycle-averaged wing aerodynamics. This assumption is justified because it is difficult to attain an accurate instantaneous wing configuration state (ϕ and ψ in Fig. 3) when flapping at high frequencies ($\sim$ 150 Hz).

Assumption 3: A constant angle-of-attack is assumed throughout the cycle-averaged flapping-wing motion in order to attain a closed-form solution for the lift and drag forces.

A. Lift and Drag Models

The high frequency flapping-wing motion of the RoboBee make the wing dynamics difficult to model. Instead, [4] treats the wings as cycle-averaged quasi-steady blade elements, using *Assumption 1* that the wings are decoupled from the vehicle body. The model for the lift and drag forces produced by the wing over a cycle period T for the i—th wing, where $i \in L, R$, is

$$\overline{F}^i_L = \frac{\omega\rho\beta}{4\pi}\int_0^{\frac{2\pi}{\omega}} C_L(\alpha_i)\dot{\phi}^2 dt, \tag{1}$$

$$\overline{F}^i_D = \frac{\omega\rho\beta}{4\pi}\int_0^{\frac{2\pi}{\omega}} C_D(\alpha_i)\dot{\phi}|\dot{\phi}|dt, \tag{2}$$

where ϕ is the wing stroke angle, $\omega = 2\pi/T$ is the flapping frequency of the wings, ρ is the density of air as shown in Fig. 3. β is a wing geometry parameter, and $C_L(\alpha_i)$ and $C_D(\alpha_i)$ are the lift and drag coefficients of the wing as a function of the angle of attack α which were gathered from experimental data of a *Drosphilia* wing in [10].

Using the lift and drag forces defined in Eq. 1 and 2, the thrust and torque forces are defined as

$$F_T = \overline{F}^L_L + \overline{F}^R_L, \tag{3}$$

$$\tau_r = r_{cp}\left[\overline{F}^L_L cos(\phi^L_{mean}) - \overline{F}^R_L cos(\phi^R_{mean})\right], \tag{4}$$

$$\tau_p = r_{cp}\left[\overline{F}^L_L sin(\phi^L_{mean}) - \overline{F}^R_L sin(\phi^R_{mean})\right], \tag{5}$$

$$\tau_y = r_{cp}(\overline{F}^L_D - \overline{F}^R_D), \tag{6}$$

where r_{cp} is the distance from the body z-axis to the (left or right) wing's center of pressure.

The PZT actuators that power the RoboBee require input voltage signals to control the flapping-wing motion (ϕ and ψ). From *Assumption 3*, we set ψ to have a constant angle-of-attack of 45°. The relationship between stroke angle ϕ and input voltage is defined as

$$\Phi(\omega) = H(\omega)V(\omega), \tag{7}$$

where [4] defines the transfer function $H(\omega)$ as a second-order linear system approximation of the actuator to wing transmission dynamics of the form

$$H(\omega) = \frac{A_H}{m_{eq}(\omega j)^2 + b_{eq}(\omega j) + k_{eq}}. \tag{8}$$

The wings are actuated by applying a sinusoidal voltage signal of the form

$$V^i(t) = V^i_{off} + V^i_{amp}\sum_{n=1}^{3} a_n sin(n\omega t), \tag{9}$$

for the i—th wing, where $i \in L, R$. The coefficients a_n are harmonic signals of the wing's first, second, and third

TABLE I
MODEL PARAMETERS

Parameter	Value	Units
m	90	$[mg]$
g	9.81	$[m/s^2]$
ρ	1.204	$[kg/m^3]$
I_{yy}	1.34	$[g \cdot mm^2]$
γ_1	$3.1955 \cdot 10^{-9}$	
γ_2	$8.0088 \cdot 10^{-4}$	
γ_3	0.4774	
δ_1	4.6154	
δ_2	1.8648	
δ_3	-0.0038	
η	1.0970	
ν	-5.6520	
μ	-0.1078	

harmonics that provide control authority of the wing stroke amplitude, stroke bias, and yaw torque respectively.

Finally, Eq. 3-6 are grouped into non-dimensional parameters which are functions of the transfer function $H(\omega)$ from Eq. 8, and the input voltages, which are grouped into the vector u_i, where $i \in [1, 5]$. This was done to simplify the system identification process which is performed on the real RoboBee. The thrust and torque definitions in Eq. 3-6 become

$$F_T = \gamma_1(\delta_1^2 + \delta_2^2 u_3^2)(\eta u_1 + u_2), \tag{10}$$
$$\tau_r = \gamma_1\gamma_2(\delta_1^2 + \delta_2^2 u_3^2)(\eta u_1 - u_2), \tag{11}$$
$$\tau_p = \gamma_1\gamma_2\delta_3(\delta_1^2 + \delta_2^2 u_3^2)(\eta u_1(u_4 + \nu) + u_2(u_5 + \nu)), \tag{12}$$
$$\tau_y = \gamma_1\gamma_2\gamma_3\delta_1\delta_2(u_3 + \mu)(\eta u_1 + u_2), \tag{13}$$

with non-dimensional parameters

$$\gamma_1 = \frac{1}{4}\rho\beta C_L \omega^2, \tag{14}$$
$$\gamma_2 = r_{cp}, \tag{15}$$
$$\gamma_3 = \frac{C_D}{C_L} \cdot \frac{8}{3\pi} cos(2\angle H(\omega) - \angle H(2\omega)), \tag{16}$$
$$\delta_1 = a_1\omega|H(\omega)|, \tag{17}$$
$$\delta_2 = 2\omega|H(2\omega)|, \tag{18}$$
$$\delta_3 = |H(0)|^* sgn\angle H(0)^*. \tag{19}$$

The input vector u is defined by the input voltages

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 \end{bmatrix}^T, \tag{20}$$
$$= \begin{bmatrix} V_L^2 & V_R^2 & a_2 & V_{off}^L & V_{off}^R \end{bmatrix}^T. \tag{21}$$

The non-dimensional parameters in Eq. 14-19 were determined using an optimization-based system identification on an open-loop hover test of the real RoboBee. The parameters that were determined from the system identification are summarized in Table I.

B. Body Dynamics

For trajectory optimization, the state of the decoupled RoboBee body was simplified to 2D. The state of the simulated RoboBee used for trajectory optimization is

$$\mathbf{x} = \begin{bmatrix} x & z & \theta & \dot{x} & \dot{z} & \dot{\theta} \end{bmatrix}^T. \tag{22}$$

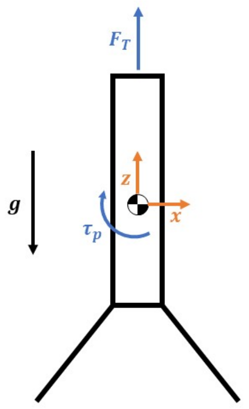

Fig. 4. 2D free-body diagram of the RoboBee used for trajectory optimization.

The dynamics of the decoupled vehicle body in 2D are defined by

$$m\ddot{x} = F_T sin(\theta), \tag{23}$$
$$m\ddot{z} = F_T cos(\theta) - g, \tag{24}$$
$$I_{yy}\ddot{\theta} = \tau_p, \tag{25}$$

and was derived using the free-body diagram in Fig. 4 which illustrates the RoboBee body in the x-z plane.

The wing thrust and torque forces from Eq. 10-13 in 2D are reduced to

$$F_T = \gamma_1(\delta_1^2 + \delta_2^2 u_3^2)(\eta u_1 + u_2), \tag{26}$$
$$\tau_p = \gamma_1\gamma_2\delta_3(\delta_1^2 + \delta_2^2 u_3^2)(\eta u_1(u_4 + \nu) + u_2(u_5 + \nu)). \tag{27}$$

In 2D, we assume that $V_L = V_R$ and that $V_{off}^L = V_{off}^R$. Additionally, the second harmonic frequency coefficient a_2 is responsible for inducing yaw torque (torque about the z-axis). Thus, we set $a_2 = 0$ since yaw is neglacted in 2D. The control input has the constraints that $u_1 = u_2$, $u_3 = 0$, and $u_4 = u_5$.

III. TRAJECTORY GENERATION

From the vehicle model defined by Eq. 23-27, the dynamics are non-linear in the voltage inputs $\mathbf{u}(t)$. Thus, we must use non-linear trajectory optimization techniques to formulate the perching maneuver problem. We follow the trajectory optimization technique presented in [7]. In this method, a nominal trajectory for the perching maneuver is first determined. Then, the trajectory is feedback-stabilized using a time-varying linearization of the vehicle dynamics about the nominal trajectory. The solutions for the nominal and linearized trajectories were found using the SNOPT solver and simulated in Drake as shown in Fig. 5.

We formulate the nominal trajectory by using the direct collocation method in which the trajectory $\mathbf{x}(t)$ and input $\mathbf{u}(t)$

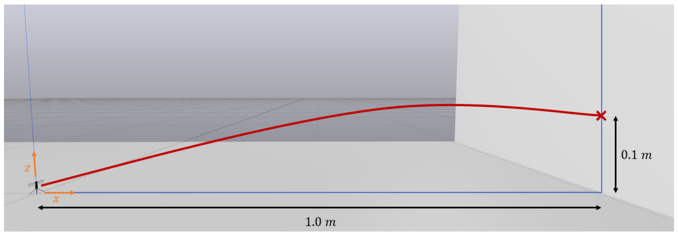

Fig. 5. Simulation environment in Drake Meshcat for visualizing the perching trajectory. The red curve illustrates what the trajectory optimization solution might look like.

are modelled as continuous, smooth piece-wise polynomial functions. The direct collocation problem is formulated as

$$\min_{\mathbf{x}[\cdot],\mathbf{u}[\cdot]} \left[\mathbf{x}[N]^T\mathbf{Q}_f\mathbf{x}[N] + \sum_{n=0}^{N-1}\mathbf{u}[n]^T\mathbf{R}\mathbf{u}[n]\right], \quad (28)$$

$$s.t. \quad \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u}), \quad (29)$$

$$x_0, z_0, \theta_0, \dot{x}_0, \dot{\theta}_0 = 0, \quad (30)$$

$$\dot{z}_0 = 0.35, \quad (31)$$

$$x_0 \le x_i \le 1, \quad (32)$$

$$0 \le z_i \le 0.2, \quad (33)$$

$$-\pi \le \theta_i \le \pi, \quad (34)$$

$$x_f = 1, \quad (35)$$

$$z_f = 0.1, \quad (36)$$

$$\theta_f = -\frac{\pi}{2}, \quad (37)$$

$$-0.2 \le \dot{x}_f \le 0, \quad (38)$$

$$-0.2 \le \dot{z}_f \le 0.2. \quad (39)$$

$$-\infty \le \dot{\theta}_f \le \infty. \quad (40)$$

Constraints from Eq. 30-31 enforce the desired initial state of the RoboBee, $\mathbf{x}_0$, to an upright, stationary position

$$\mathbf{x}_0 = \begin{bmatrix}0 & 0 & 0 & 0 & 0.35 & 0\end{bmatrix}^T. \quad (41)$$

The vehicle is given an initial vertical velocity $\dot{z}_0 = 0.35\ m/s$ in consideration of future implementation on the real RoboBee. With an initial vertical velocity constraint, the vehicle must fly upwards at the start of the trajectory as shown by the results in Fig. 6. This will help the vehicle clear the ground effect region before executing the perching maneuver trajectory when applied to the real robot in future work.

The constraints in Eq. 35-39 enforce the desired final state of the RoboBee $\mathbf{x}_f$, to the horizontal, perched position

$$\mathbf{x}_f = \begin{bmatrix}1 & 0.1 & -\frac{\pi}{2} & \dot{x}_f & \dot{z}_f & \dot{\theta}_f\end{bmatrix}^T. \quad (42)$$

The final velocity constraints $\dot{x}_f$, $\dot{z}_f$, and $\dot{\theta}_f$ in Eq. 38-40 are left as bounding box constraints. These relaxed velocity constraints are in consideration of the mechanical design of the real RoboBee which features compliant legs that are robust to non-zero velocity landings by damping the vehicle's kinetic energy during impact. Thus, the trajectory optimization need not enforce a perfect quasi-static zero velocity landing for the perching maneuver. These relaxed final state constraints helped the SNOPT solver find a feasible solution.

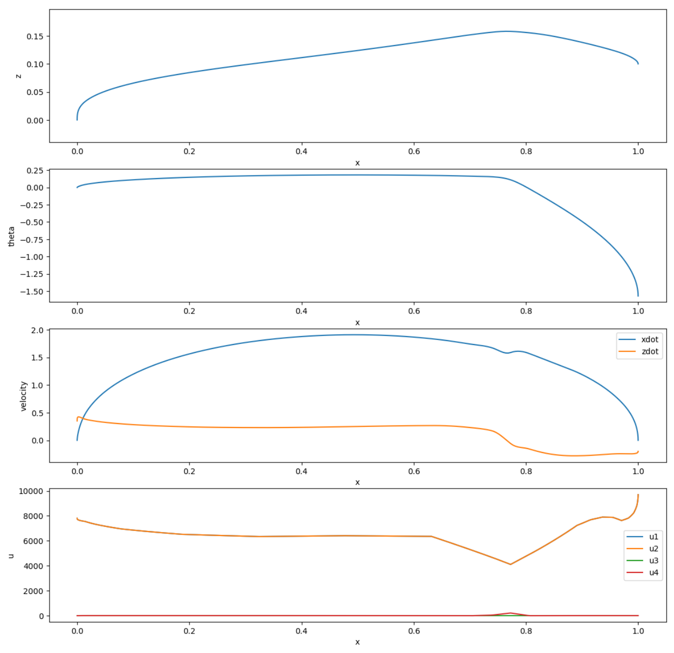

Fig. 6. Results from direct collocation trajectory optimization with SNOPT.

The constraints for the intermediary collocation points in Eq. 32-34 were chosen to restrict the trajectory solution to z positions not too far away from the target and to restrict the x positions to within the starting position and the wall. The pitch constraints on θ restrict body rotations to within 2π.

The cost function is formulated as a quadratic cost on the input voltage vector $\mathbf{u}(t)$ and the final, perched state $\mathbf{x}[N]$ as defined by Eq. 28 where $\mathbf{R}$ is the cost on $\mathbf{u}(t)$ and $\mathbf{Q}_f$ is the cost on $\mathbf{x}[N]$ which are both diagonal matrices. The cost on input effort was chosen as $\mathbf{R} = 50 \cdot I$, where I is the identity matrix. The cost on the final state was chosen to be a diagonal matrix with elements $diag(\mathbf{Q}_f) = \begin{bmatrix}100 & 100 & 100 & 1 & 1 & 1\end{bmatrix}$, corresponding to the gains of the state elements $\mathbf{x} = \begin{bmatrix}x & z & \theta & \dot{x} & \dot{z} & \dot{\theta}\end{bmatrix}$. The cost matrices $\mathbf{R}$ and $\mathbf{Q}_f$ were hand-tuned by adjusting the cost values to allow the SNOPT solver to find a feasible solution that satisfies the problem constraints at all collocation points. The nominal trajectory solution to the direct collocation problem formulation was found using Drake's `DirectCollocation` method with SNOPT solver. The nominal trajectory results are plotted in Fig. 6. From Fig. 6, we see that the body angle θ tilts slightly forward before performing a flip-turn at around $x = 0.8\ m$ in order to gain enough momentum to reach the wall. During the flip-turn, the input voltage $u_1 = u_2 = V_L^2 = V_R^2$ drops to a minimum before ramping up again right before the final perched state to decelerate the vehicle as shown in the velocity plot.

When designing the nominal trajectory, I found that the SNOPT solver was quite sensitive to slight changes in cost function values and number of collocation points. In addition, the choice of initial state vertical velocity impacted the ability for the solver to determine a feasible solution. The choices of cost values was chosen to give a high penalty for input effort, and also a high penalty to deviations from the

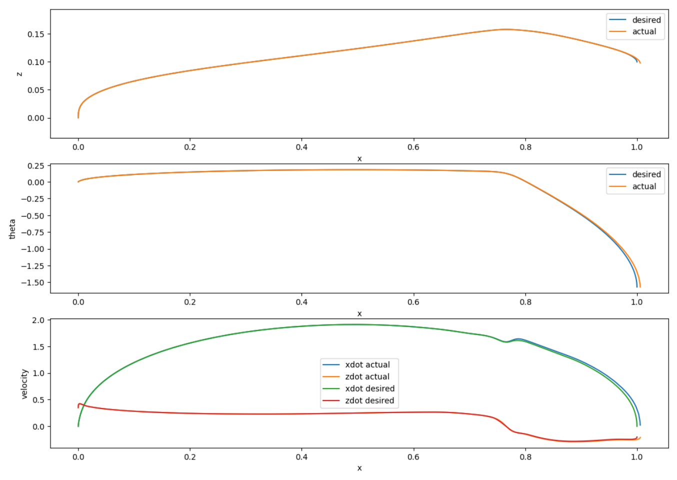

Fig. 7. Comparison between nominal trajectory from direct collocation with open-loop simulation of the RoboBee using the determined control inputs.

final x, z, and θ positions. The choice of the initial state velocity constraint in Eq. 31, was designed in parallel with the intermediary collocation point bounding box constraints in Eq. 32-33. The intermediary constraints in Eq. 32-33 were chosen such that the trajectory is restricted to a region near the final z position of $0.1m$ and also restricted within the start and end positions (we do not want the vehicle to go behind the starting position or through the wall). If these constraints were not set, it was found that the trajectory would undergo unpredictable oscillatory trajectories in the x-z plane and/or SNOPT would not be able to find a feasible solution. Sometimes the solver would even plan a trajectory that goes under the ground ($z < 0$), or through the wall ($x > 1.0$). Additionally, if the initial state velocity constraint $\dot{z}_0$ in Eq. 31 was chosen to be too small, the solver would sometimes find solutions that go under the ground. Alternatively, if $\dot{z}_0$ was chosen to be too large, the solver would command the vehicle vertically upwards, then straight back down, before commanding a smooth trajectory to the final perched position. From the design of the direct collocation problem formulation, I learned that a good choice of constraints can help push the solver towards a desired trajectory behavior.

The nominal trajectory solution was also compared to an open-loop simulation of the RoboBee dynamics. Here, the input trajectory solution $\mathbf{u}(t)$ found from direct collocation was used as the input for an open-loop simulation of the RoboBee model. The open-loop simulation results are plotted alongside the desired nominal trajectory in Fig. 7. The open-loop simulation seemed to match the nominal trajectory fairly well with slight deviation near the final perched state, causing the trajectory to overshoot the wall x position. While this is not an issue knowing that the real vehicle is equipped with energy-damping leg appendages, we would like to develop a feedback controller to account for deviations from the trajectory earlier in time.

IV. Trajectory Stabilization

To correct for the deviation between simulation and the desired nominal trajectory, the RoboBee dynamics can be stabilized using a finite-horizon, time-varying linear quadratic regulator (TVLQR) controller. In order to stabilize the vehicle about the nominal trajectory, the RoboBee state and input dynamics ($\mathbf{x}(t)$ and $\mathbf{u}(t)$) are linearized about the nominal trajectory ($\mathbf{x}_0(t)$ and $\mathbf{u}_0(t)$) as

$$\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_0(t), \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_0(t). \tag{43}$$

Using Eq. 43, the dynamics can be linearized using a first-order Taylor expansion approximation

$$\begin{aligned} \dot{\bar{\mathbf{x}}} &= \dot{\mathbf{x}} - \dot{\mathbf{x}}_0, & (44)\\ &= f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}_0, \mathbf{u}_0), & (45)\\ &\approx \mathbf{A}(t)\bar{\mathbf{x}} + \mathbf{B}(t)\bar{\mathbf{u}}, & (46) \end{aligned}$$

yielding a linear, time-varying version of the RoboBee body dynamics about the nominal trajectory

$$\dot{\bar{\mathbf{x}}} = \mathbf{A}(t)\bar{\mathbf{x}} + \mathbf{B}(t)\bar{\mathbf{u}}. \tag{47}$$

If we formulate the problem using a quadratic cost function, of the form

$$\begin{aligned} \mathbf{J} &= \mathbf{x}(t_f)^T \mathbf{Q}_f \mathbf{x}(t_f) \\ &\quad + \int_{t=0}^{t_f} \left[\mathbf{x}(t)^T \mathbf{Q} \mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R} \mathbf{u}(t) \right] dt, \end{aligned} \tag{48}$$

then the corresponding solution for the TVLQR feedback controller $\mathbf{u}(t)$ is defined by solving the associated Riccati equation backwards in time as derived in section 2.4.4 in [7]

$$\begin{aligned} \dot{\mathbf{S}}(t) &= -\mathbf{A}^T(t)\mathbf{S}(t) - \mathbf{S}(t)\mathbf{A}(t) \\ &\quad + \mathbf{S}(t)\mathbf{B}(t)\mathbf{R}^{-1}\mathbf{B}^T(t)\mathbf{S}(t) - Q, & (49)\\ \mathbf{u}(t) &= -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}(t)\mathbf{x}(t), & (50) \end{aligned}$$

with the terminal condition $\mathbf{S}(t_f) = \mathbf{Q}_f$ and a cost-to-go function of the form $\mathbf{J}^*(\mathbf{x}, t) = \mathbf{x}^T\mathbf{S}(t)\mathbf{x}$.

The matrices $\mathbf{Q}_f$, $\mathbf{Q}$, and $\mathbf{R}$ for the cost function in Eq. 48 are positive definite diagonal matrices. The diagonal matrix values were hand-tuned in a similar manner as the direct collocation problem formulation such that the trajectories with different initial conditions would stabilize towards the final, perched position as shown by the results in Fig. 8. The diagonal cost values on the state were chosen to be $diag(\mathbf{Q}) = \begin{bmatrix}100 & 100 & 100 & 1 & 1 & 1\end{bmatrix}$. The cost on the input was chosen as $\mathbf{R} = 0.1 \cdot I$. The diagonal cost values on the final state were chosen to be $diag(\mathbf{Q}_f) = \begin{bmatrix}(1/0.05)^2 & (1/0.05)^2 & (1/3.0)^2 & 1 & 1 & 1\end{bmatrix}$. The stabilized, TVLQR feedback controller was implemented using Drake's `MakeFiniteHorizonLinearQuadraticRegulator` method. The TVLQR controller was connected to the RoboBee model and simulated 5 times, each with a different random vertical z initial position. This shows the controller's robustness to deviations from the desired trajectory earlier in time, while still able to stick the landing at the end. The simulated, feedback controlled trajectories were plotted with the desired, nominal trajectory found using direct collocation in Fig. 8. The simulated trajectories all reach the desired, perched position regardless of the initial vertical z position.

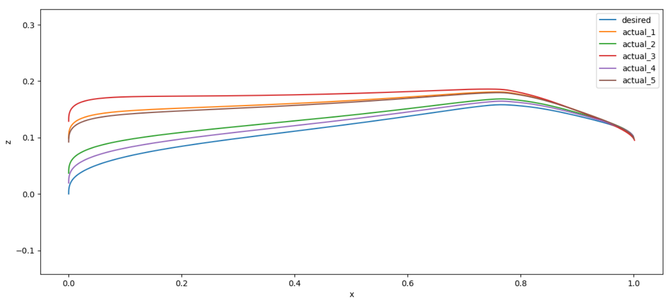

Fig. 8. Stabilization of the nominal trajectory about the perching point using finite horizon LQR.

V. Conclusion

This work tackles the control problem of landing for FWMAVs by building upon the closed-form control model of the Harvard RoboBee (an FWMAV) and applying trajectory optimization techniques to achieve an acrobatic, feedback-controlled, perching aerial maneuver in simulation. We successfully combined previous work in modelling of the RoboBee wing forces and body dynamics, with trajectory optimization techniques using direct collocation and TVLQR methods, while leveraging Drake and SNOPT solver for developing solutions to our problem formulations and constraints. Future work includes the extension of the trajectory optimization to 3D, and the addition of disturbance rejection from environmental sources such as wind. I am interested in exploring other feedback control techniques such as LQR-Trees which leverages a library of stabilized trajectories to expand the region of attraction for a desired trajectory as studied in [7]. For hardware implementation, the current RoboBee controller is developed in MATLAB and Simulink and would require translation from Python which was used in this work. I am also interested in exploring the implementation of onboard sensors for the RoboBee for state-estimation feedback using Extended Kalman Filtering. For example, the implementation of an onboard magnetometer could enable magnetic-field sensing for perching on a power-line or landing pad as explored in [11]. This would be the next step towards autonomous flight and recharging for the RoboBee. The trajectory optimization framework presented in this work serves as a platform for which we can build upon, with the intent of achieving these future goals.

References

[1] J. Yan, R. Wood, S. Avadhanula, M. Sitti, and R. Fearing, "Towards flapping wing control for a micromechanical flying insect," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 4, 2001, pp. 3901–3908 vol.4.

[2] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science*, vol. 340, no. 6132, pp. 603–607, 2013.

[3] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

[4] R. McGill, N.-s. Hyun, and R. Wood, "Frequency-modulated control for insect-scale flapping-wing vehicles," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–8, 10 2022.

[5] R. Cory and R. Tedrake, *Experiments in Fixed-Wing UAV Perching*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2008-7256

[6] J. W. Roberts, R. Cory, and R. Tedrake, "On the controllability of fixed-wing perching," in *2009 American Control Conference*, 2009, pp. 2018–2023.

[7] R. E. Cory, "Supermaneuverable perching," PhD Thesis, Massachusetts Institute of Technology, June 2010.

[8] M. A. Graule, P. Chirarattananon, S. B. Fuller, N. T. Jafferis, K. Y. Ma, M. Spenko, R. Kornbluh, and R. J. Wood, "Perching and takeoff of a robotic insect on overhangs using switchable electrostatic adhesion," *Science*, vol. 352, no. 6288, pp. 978–982, 2016.

[9] P. Chirarattananon, K. Y. Ma, and R. J. Wood, "Perching with a robotic insect using adaptive tracking control and iterative learning control," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1185–1206, 2016.

[10] M. H. Dickinson, F.-O. Lehmann, and S. P. Sane, "Wing rotation and the aerodynamic basis of insect flight," *Science*, vol. 284, no. 5422, pp. 1954–1960, 1999. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.284.5422.1954

[11] J. L. Moore, "Powerline perching with a fixed-wing uav," Master's thesis, Massachusetts Institute of Technology, June 2011.