Landing Trajectory Optimization for the RoboBee

Rebecca Steinmeyer
Harvard University
rhsteinmeyer@g.harvard.edu

Irina Tolkova
Harvard University
itolkova@g.harvard.edu

Abstract

The Harvard RoboBee is an insect-scale flapping-wing micro-aerial vehicle, subject to extreme challenges in maneuvering due to its small scale and control authority, and inherent instability. Landing the vehicle, in particular, remains an unsolved problem (and is typically bypassed during experiments); determining an optimal landing trajectory would increase system versatility. We sought to apply optimization techniques to determine control inputs (forces and torques) for trajectories which result in a desired final position and orientation (from a variety of initial states), using IPOPT, SNOPT, and an ADMM-based method. Furthermore, we found optimal wingstroke signal parameters to produce the resulting control inputs, both by expansion of the initial optimization methods and by separate implementation of Projected Gradient Descent. We found that all three solvers were able to achieve physically feasible control parameters for the landing problem with similar degrees of optimality. We compare the advantages and challenges of each solver and problem formulation for trajectory optimization of the RoboBee.

1 Introduction

The Harvard RoboBee is an 87 mg flapping-wing micro-aerial vehicle (MAV) which achieves flight via two independently flapping wings [16], each actuated by its own piezoelectric bimorph actuator [17]. The MAV is characterized by a resonant flapping frequency of 170 Hz, at which it produces maximal lift, with a lift-to-weight ratio of 3.5 [10]. While an MAV at this scale opens opportunities for applications unsuitable for larger, traditional aerial systems (e.g., assisted pollination, search-and-rescue, and surveillance), achieving flapping-wing flight at the insect scale comes with a unique set of challenges: The RoboBee is inherently unstable; has extreme limitations in power, sensing, and control; and is very small (in both size and control authority) relative to external disturbances. As such, agile and robust maneuvering remains an active area of research.

Landing the RoboBee, in particular, is a crucial task for robust vehicle performance (especially as the vehicle approaches autonomy, which will eventually allow it to operate outside of the laboratory setting). Previous research has achieved upright, stable landing only with large carbon fiber struts added to the base of the vehicle as "landing gear" [2] or via perching on overhead or vertical surfaces using electromagnetic pads to adhere to the landing surface [7, 3]. Basic landing of the system, however, remains an obstacle, and is simply avoided during current experiments: either the RoboBee hangs from a string as its starting and landing position, or it simply crashes after its maneuver.

We aim to apply the optimization strategies to determine an acceptable landing trajectory for the RoboBee (in terms of force and torque control inputs) starting from an arbitrary initial state. From there, we will solve for optimal wingstroke parameters (i.e., the wingstroke sinusoid directly governing actuator motion, and therefore determining wingstroke kinematics). To do so, we will apply a set of solvers (proprietary solvers SNOPT and IPOPT, along with an iterative ADMM-based solver) to determine control inputs. We will find wingstroke parameters via two strategies: first, with expansion of the SNOPT, IPOPT, and ADMM methods (the "single-stage" method); and second, with a separate nonconvex optimization (the "two-stage" method), specifically using Projected Gradient Descent,

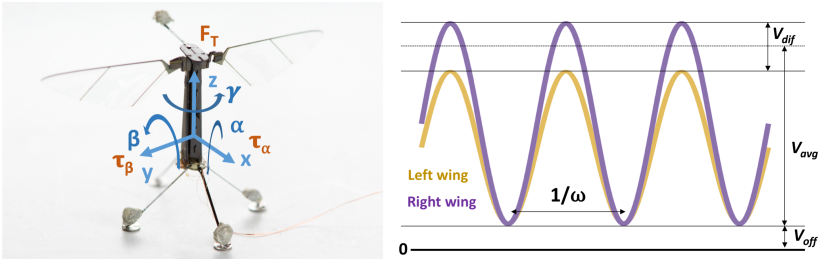

Figure 1: State, control, and wingstroke signal parameters for the Harvard RoboBee; (left) the RoboBee with position (x, y, z) and orientation (α, β, γ) indicated, with associated control parameters thrust (F_T), roll torque (τ_α), and pitch torque (τ_β), (right) wingstroke signal construction, with associated signal parameters flapping frequency (ω), average voltage (V_{avg}), voltage difference between the wings (V_{dif}), and voltage offset (V_{off}).

with a stochastic element added to avoid simply finding local optima. We will then compare the results of the two pipelines for optimization-based trajectory planning for the Harvard RoboBee.

2 Background

2.1 The Harvard Robobee

We model the Harvard RoboBee using the quasi-static approach described by Ma et al. [13], wherein RoboBee dynamics are determined by time-averaged force and torque control parameters.

At any point in time, the state of the RoboBee, $\mathbf{X} \in \mathbb{R}^{12\times 1}$, may be described as the position of the vehicle (x, y, z), the orientation of the vehicle (α, β, γ), the body translational velocity $(\dot{x}, \dot{y}, \dot{z})$, and the body rotational velocity $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$, in that order (see Figure 1). (We note that although the rotation about the yaw axis, γ, is included in our state-space representation of the RoboBee, control authority about this axis remains an active area of research, so we will not include yaw torque in our control parameters.) Our state-space representation of RoboBee dynamics is as follows, where $\mathbf{U} = [F_T \quad \tau_\alpha \quad \tau_\beta]^\intercal$ (with thrust force F_T, roll torque τ_α, and pitch torque τ_β; also appended here as $\mathbf{U}'$ to include mg at the end of the vector to take gravity into account in our model):

$$\dot{\mathbf{X}} = \begin{bmatrix} \mathbf{0}^{(6)} & \mathbf{I}^{(6)} \\ \mathbf{0}^{(6)} & \mathbf{0}^{(6)} \end{bmatrix} \mathbf{X} + \left[\begin{array}{c|ccccc} \mathbf{0}^{(4\times 6)} & \frac{\sin\beta}{m} & -\frac{\sin\alpha\cos\beta}{m} & \frac{\cos\alpha\cos\beta}{m} & 0 & 0 & 0 \\ & 0 & 0 & 0 & \frac{1}{I_\alpha} & 0 & 0 \\ & 0 & 0 & 0 & 0 & \frac{1}{I_\beta} & 0 \\ & 0 & 0 & -\frac{1}{m} & 0 & 0 & 0 \end{array}\right]^\intercal \mathbf{U}'. \quad (1)$$

In the above state-space model, m is the mass of the RoboBee, and I_α and I_β are the moment of inertia about the x and y axes (that is, associated with roll and pitch), respectively. All constants for the state space model and the control parameter equations below may be found in Table 2.1.

To produce the desired control input $\mathbf{U} = [F_T \quad \tau_\alpha \quad \tau_\beta]^\intercal$, we design a set of wingstroke signal parameters $\mathbf{Y} = [\omega \quad V_{avg} \quad V_{dif} \quad V_{off}]^\intercal$ to directly apply a sinusoidal voltage signal to the actuators in order to govern the wingstroke kinematics (as shown in Figure 1). These parameters are related to the force and torque control parameters ($\mathbf{U}$) as follows:

$$??F_T = \frac{1}{2}\rho B C_L \big(\omega G(\omega)\big)^2 \big(V_{avg}^2 + V_{dif}^2\big) \quad (2)$$

$$\tau_\alpha = r_{cp}\rho B C_L \big(\omega G(\omega)\big)^2 \big(V_{avg} V_{dif}\big) \quad (3)$$

$$\tau_\beta = r_{cp} V_{off} G(0) F_T \quad (4)$$

$$= \frac{1}{2} r_{cp}\rho B C_L G(0) \big(\omega G(\omega)\big)^2 V_{off} \big(V_{avg}^2 + V_{dif}^2\big) \quad (5)$$

where ρ is the ambient air density, B is a wing geometry parameter [1], C_L is the average lift coefficient over the wingstroke, and r_{cp} is the shoulder width of the vehicle. Additionally, the transfer

function $G(\omega) = |\frac{A}{m_{eq}(j\omega)^2 + b_{eq}(j\omega) + k_{eq}}|$ describes the frequency response of the wingstroke to the actuator and transmission dynamics and filtering.

To design acceptable constraints for the control and wingstroke parameters for the RoboBee landing trajectory, we also note typical operating conditions for the vehicle: the flapping frequency lies at the resonant point for maximum lift (here, 170Hz), and can achieve lift down to approximately 140Hz; we constrain it therefore to $\omega \in [100, 180]$ Hz. Operating voltage is actuator-limited (a voltage amplitude higher than 200 V will likely cause microcracks in the piezoelectric material); we constrain our average voltage amplitude therefore to $V_{avg} \in [140, 200]$ V. Voltage difference (between the wings) and voltage offset should both only be small perturbations in the wingstroke signals; we therefore constrain them to $V_{dif} \in [-30, 30]$ V and $V_{off} \in [-30, 30]$ V.

Constant	Value	Units
m	8.7×10^{-7}	kg
I_α	2.45×10^{-12}	$kg\ m^2$
I_β	1.34×10^{-12}	$kg\ m^2$
ρ	1.2041	kg/m^3
B	2.0933×10^{-9}	
C_L	1.8	
r_{cp}	0.0136	m
A	0.9538	
m_{eq}	3.82×10^{-4}	kg
b_{eq}	0.1959	Ns/m
k_{eq}	500.4331	N/m

We note that the state space model provided herein is a simplified representation of RoboBee dynamics. For example; flapping-wing flight is inherently unsteady; our quasi-steady model reflects a simplified (time-averaged) version which does not take into account flapping-wing aerodynamic effects such as vortex shedding [4]. Additionally, we do not take external disturbances (e.g., ambient air flow) into account in our simulation. We also do not take yaw torque into account in this work; it is a historically weak control effort for the vehicle, and is an active area of research.

2.2 Optimization-based Planning

There have been a wide range of approaches for path planning in robotics. One class of algorithms that have proven successful in some studies are sampling-based methods, such as rapidly-exploring random trees or probabilistic roadmaps [12]. However, while these approaches can work well for simple systems, they fall under the colloquial "curse of dimensionality": to find dynamically feasible solutions, sampling must be done in the robot's state space, which grows exponentially as the number of degrees of freedom of the system increases. As a result, most sampling-based algorithms become impractical for complex robotic systems. Alternatively, path planning can be formulated as a (nonconvex) optimization problem, which may not have runtime guarantees, but is also not reliant on exponentially expensive operations. Dynamical feasibility is encoded as numerical integration constraints between consecutive knot points, with different integration schemes leading to different established approaches, such as direct transcription and direct collocation. In this project, we formulate two variants of the trajectory optimization problem using the midpoint integration scheme.

3 Problem Formulation

3.1 Single-Stage Problem

A straightforward application of methods such as direct transcription to the RoboBee problem would lead to an optimization problem which considers the state $\mathbf{x} \in \mathbb{R}^{12}$ (of positions, angles, velocities, and angular velocities) to be a highly nonlinear function of the control parameters $\mathbf{y}$. In other words, consider $f : \mathbb{R}^{12} \times \mathbb{R}^3 \to \mathbb{R}^{12}$ to be the dynamics function $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ described by Equation 1, and $g : \mathbb{R}^4 \to \mathbb{R}^3$ to be the function mapping wingstroke signal parameters $\mathbf{y}$ to control inputs $\mathbf{u}$ as described by Equations (2)-(5). Then, using a midpoint integration scheme, the optimization problem would minimize some objective function $c(\mathbf{x}_i, \mathbf{y}_i)$ under initial and final positions constraints,

physical dynamics constraints, and constraints on lower and upper bounds on states and inputs:

$$\begin{aligned}
\min_{\mathbf{x}_i, \mathbf{y}_i \forall i \in [0,n]} \quad & c(\mathbf{x}_i, \mathbf{y}_i) \\
\text{such that} \quad & \mathbf{x}_0 = \mathbf{x}_{\text{initial}}, \quad \mathbf{x}_m = \mathbf{0}^{12} \\
& \mathbf{x}_{i+1} = \mathbf{x}_i + \frac{1}{\Delta t} \cdot f\left(\frac{\mathbf{x}_i + \mathbf{x}_{i+1}}{2}, g(\frac{\mathbf{y}_i + \mathbf{y}_{i+1}}{2})\right), \quad \forall i \in [0, n-1] \\
& \mathbf{y}_{min} \preceq \mathbf{y}_i \preceq \mathbf{y}_{max}, \quad \forall i \in [0, n] \\
& \mathbf{x}_{min} \preceq \mathbf{x}_i \preceq \mathbf{x}_{max}, \quad \forall i \in [0, n]
\end{aligned}$$

While this formulation is a valid approach to the problem we are interested in, it is clearly highly nonlinear and nonconvex due to the complexity of the dynamics constraints. Since nonconvex solvers often rely on some form of low-order approximations for constraints functions, high nonlinearity leads to higher error within every step of a solver, resulting in increased computational complexity and a higher chance of divergence.

3.2 Two-Stage Problem

To mitigate the potential issues caused by high nonlinearity, we consider an alternative approach which breaks the overall optimization problem into two stages by solving for optimal control inputs $\mathbf{u}$, and then computing the wingstroke parameters $\mathbf{v}$ that could correspond to $\mathbf{u}$. More specifically, the first stage can be formulated very similarly to the problem above:

$$\begin{aligned}
\min_{\mathbf{x}_i, \mathbf{u}_i \forall i \in [0,n]} \quad & c(\mathbf{x}_i, \mathbf{u}_i) \\
\text{such that} \quad & \mathbf{x}_0 = \mathbf{x}_{\text{initial}}, \quad \mathbf{x}_m = \mathbf{0}^{12} \\
& \mathbf{x}_{i+1} = \mathbf{x}_i + \frac{1}{\Delta t} \cdot f\left(\frac{\mathbf{x}_i + \mathbf{x}_{i+1}}{2}, \frac{\mathbf{u}_i + \mathbf{u}_{i+1}}{2})\right), \quad \forall i \in [0, n-1] \\
& \mathbf{u}_{min} \preceq \mathbf{u}_i \preceq \mathbf{u}_{max}, \quad \forall i \in [0, n] \\
& \mathbf{x}_{min} \preceq \mathbf{x}_i \preceq \mathbf{x}_{max}, \quad \forall i \in [0, n]
\end{aligned}$$

This problem can intuitively be considered simpler, as it does not involve the nonlinear relationships between $\mathbf{u}$ and $\mathbf{v}$, and may be more likely to be solved successfully by optimization software. However, we introduce several challenges. First, it would become more difficult to represent any objective function that depends on y, so we only work with state-dependent objectives in this project. More significantly, since g is not isomorphic, calculating $y_i = g^{-1}(u_i)$ is not fully well-defined. Instead, based on the knowledge of the form (and gradients) of g, we calculate the least-squares fit to wingstroke parameters through the development of another optimization procedure.

Specifically, we solve an independent second nonconvex problem to find a set of wingstroke parameters over the trajectory, to recreate the desired control inputs in $\mathbf{v}$. To do so, we minimize some objective function $d(\mathbf{x}_i, \mathbf{u}_i)$ under wingstroke parameter constraints, solving the following problem:

$$\begin{aligned}
\min_{\mathbf{u}_i, \mathbf{v}_i \forall i \in [0,n]} \quad & d(\mathbf{u}_i, \mathbf{v}_i) \\
\text{such that} \quad & \mathbf{u}_{min} \preceq \mathbf{u}_i \preceq \mathbf{u}_{max}, \quad \forall i \in [0, n].
\end{aligned}$$

While this approach augments variability in the actual control inputs achieved, it is advantageous in that, although this remains a nonconvex problem, we may consider the nonlinear relationship between wingstroke parameters and control inputs as an objective rather than a constraint, allowing implementation of more traditional optimization methods.

3.3 Objective Functions

In the context of path planning for robotic systems, it is often desirable to minimize energy expenditure, minimize the navigation time, control orientation during movement, or use shaping costs in the

objective to encourage satisfaction of constraints. For the RoboBee system, a possible objective function could place a cost on the average voltage V_{avg} or the flapping frequency w, or on the imparted thrust force and torques. However, to directly compare the two approaches outline above, we limited the problem scope to only consider an objective function that was independent of input values. One class of such common objective functions has the form:

$$c(\mathbf{x}) = \sum_{i=0}^{n} (\mathbf{x}_i - \mathbf{x}_{final})^T Q (\mathbf{x}_i - \mathbf{x}_{final}) \tag{6}$$

This function can be considered a "shaping" cost that encourages the trajectory to satisfy the final state constraint by distributing the constraint across all waypoints rather than setting it to only influence the variable $\mathbf{x}_n$. The square matrix Q is often a diagonal matrix of weights on each quadratic term. However, if this is used as the full objective, and if the system can maintain a static state at x_f, then it can be thought of as a representation of a minimum time problem, since the cost will be minimal if $\mathbf{x}$ arrives at x_f in the fewest time points, and remains there indefinitely afterwards. We choose Q equal to the identity matrix for our project.

During the two-stage method (specifically considering the wingstroke trajectories U and the control inputs V for the RoboBee) we designed an objective function specifically to include the input values (U, for the second stage), as we aimed to minimize the discrepancy between the physical control inputs generated through both methods:

$$d(U, V) = \sum_{i=0}^{n} \big(f\big(\mathbf{u}_i\big) - \mathbf{v}_i\big)^2. \tag{7}$$

In the cost function below, the function $f(\mathbf{u})$ applies Equations (2)-(5) to back-calculate resulting forces and torques from the optimized wingstroke parameters.

4 Methods and Implementation

4.1 Experiment Overview

In this project, we implement two variations of dynamical systems representing the RoboBee (corresponding to the two different solution methodologies), along with the described optimization problems, within the Drake Robotics Toolbox. We will then solve the single-stage and two-stage methods with SNOPT, IPOPT, and ADMM (described in more detail below) for 20 problem instances, with initial positions chosen uniformly at random with $x \in [-5, 5]$, $y \in [-5, 5]$, $z \in [0, 5]$ and final positions fixed at $[0, 0, 0.01]$. The number of discretized waypoints is set to $N = 40$, the total time is set to $T = 4.0$ seconds, and the (scaled) constraint tolerance is set to 10^{-6}. We will then evaluate the performance of all three solvers by comparing solve success rate, objective value, magnitude of constraint violation, and runtime. Then, for the two-stage variant, we will run projected gradient descent as described in Algorithm 1 to recover wingstroke paramaters. The parameters obtained from both approaches will be compared, with an analysis of patterns in deviation or alignment in values across time.

4.2 Solvers

For both of the approaches outlined above, the initial trajectory optimization component is solved with the use of two proprietary solvers – SNOPT and IPOPT – along with an iterative ADMM-based solver which is under concurrent development through research with the Agile Robotics Laboratory.

SNOPT and IPOPT are both implementations of optimization algorithms designed for general, large-scale nonconvex problems. In particular, SNOPT (short for "Sparse Nonlinear OPTimizer") falls under the class of sequential quadratic programming (SQP) methods [6]. The package relies on iteratively creating quadratic subproblems by computing linear approximations of the constraints and a quadratic approximation of the objective functions. Then, the these outer subproblems are solved with a reduced-Hessian quasi-Newton method, which constructs and maintains an efficient approximation of the Hessian within the structure of the standard Newton method [5]. Similarly, IPOPT ("Interior Point OPTimizer") is a solver designed by the *Computational Infrastructure for Operations Research* project, a community for open-source mathematical software. The algorithm is a primal-dual barrier

method: a logarithmic penalty is applied to the constraints, becoming increasingly steep across iterations, and optimality is evaluated by a residual calculated from primal-dual conditions. The barrier subproblems are solved with a Hessian-based method equipped with a filter line search, which "accepts" or "rejects" candidate points in an attempt to achieve faster local convergence [15]. SNOPT and IPOPT are both widely used across interdisciplinary applications, ranging from geotechnical modeling to epidemiology to electrochemistry [14][8][11], have bindings to common programming languages, and are integrated with the Drake toolbox, making them reasonable choices for our project. Importantly, while we do not provide gradients for the dynamics constraints, Drake supports auto-differentiation of the dynamics, which is used for our project.

Along with the proprietary solvers, we will also evaluate the performance of a simple iterative solver intended specifically for non-convex trajectory planning problems, under development as part of graduate research with the Agile Robotics Laboratory. At each iteration, the algorithm takes the first-order Taylor approximation of the constraints relative to the previous iteration, resulting in a linear form of the dynamics and of additional constraints such as variable bounds of obstacle-avoidance conditions. The algorithm then formulates the minimization as a consensus problem with a quadratic penalty on constraints, and applies an iteration of the consensus variant of the Alternating Direction Method of Multipliers – an algorithm that constructs the augmented Lagrangian and alternates minimizing with respect to two "duplicate" variables, one associated with the objective function and the other with constraints. By increasing the penalty weights with every iteration, the problem approaches true satisfaction of constraints. This algorithm is also integrated with Drake, and has been previously used for planning for quadrotor and manipulator systems, but has not been applied to or tested on the RoboBee.

4.3 Projected Gradient Descent

During the two-stage method, to determine appropriate wingstroke signal parameters from the control inputs found using the IPOPT, SNOPT, and ADMM solvers, we apply a variant on Projected Gradient Descent (chosen to provide an optimal solution while adhering to the parameter constraints described in Section 2.1). Due to the nonconvex nature of the problem, we also applied an element of stochasticity to the algorithm in order to avoid locally optimal solutions.

As described in Algorithm 1, the optimization method applied herein iterates over a selection of random wingstroke parameter starting conditions. For each starting condition, the solver iterates through the set of N discrete control inputs in $V \in \mathbb{R}^{3\times N}$, and for each, applies Projected Gradient Descent to find an optimal associated set of wingstroke signal parameters U. During each Projected Gradient Descent step, the solution is projected to lie within the parameter constraints in $U_{bnd} \in \mathbb{R}^{3\times N}$.

Algorithm 1 Projected Gradient Descent *(with stochasticity for nonconvex optimization)*

input: $V \in \mathbb{R}^{3\times N}, U_{\text{bnd}} \in \mathbb{R}^{4\times 2}, \eta \in \mathbb{R}^4, M, N, T$
for $i = 1$ to M **do**
    for $j = 1$ to N **do**
        if $j = 1$ **then**
            $\mathbf{a}_j^1 \leftarrow$ random vector within bounds described by U_{bnd}
        else
            $\mathbf{a}_j^1 \leftarrow \mathbf{u}_j'^{\,i}$
        end if
        for $t = 1$ to T **do**
            $\mathbf{y}^{t+1} \leftarrow \mathbf{a}_j^t - \eta \nabla f(\mathbf{a}_j^t, \mathbf{v}_j)$
            $\mathbf{a}_j^{t+1} \leftarrow \Pi_K \mathbf{y}^{t+1}$
        end for
        $\mathbf{u}_j'^{\,i} \leftarrow \mathbf{a}_j^{T+1}$
    end for
end for
$U \leftarrow \arg\min \sum_{j=1}^{N} f(\mathbf{u}_j'^{\,i}, \mathbf{v}_j)$
return U

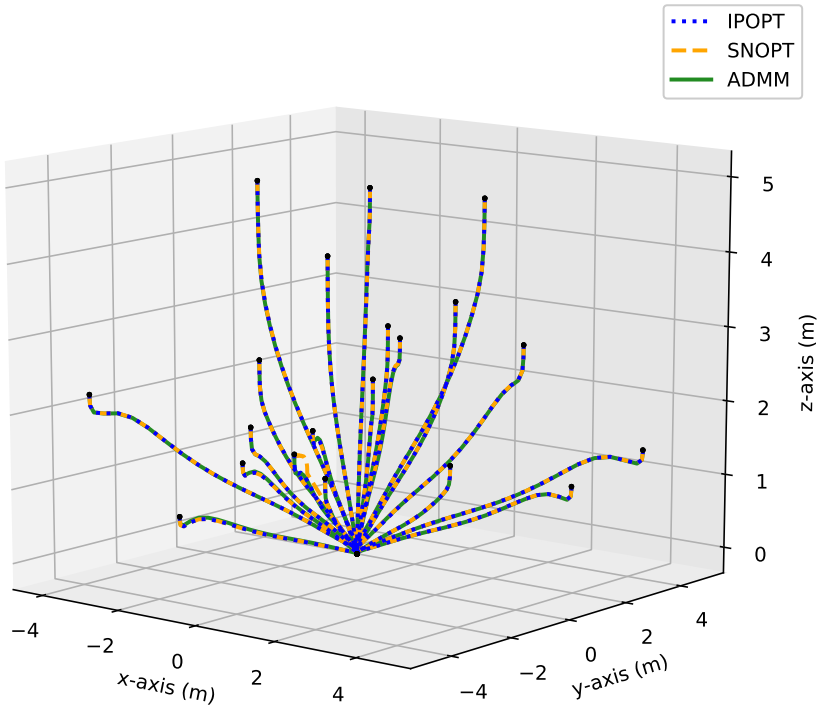

Figure 2: Solution landing trajectories of 20 randomized problem instances for the single-stage formulation.

Over all randomized starting points, the algorithm takes the solution with minimal total cost; that is, the solution in which the control parameters resulting from the solved wingstroke signal parameters match most closely to the original control parameters. This method finds a solution for all given control inputs so long as the control parameters are feasible within the parameter constraints of the RoboBee; this would result in a set of identical cost minima instead of a distinct solution.

5 Results

Figure 2 shows the 3D trajectories of the solutions for the single-stage problem formulation. The appearance of trajectories is very similar to those resulting from the two-stage formulation, such as the specific example shown in Figure 3. The state and input valiues corresponding to this example are shown in Figure 4.

Of the 20 problem instances for the full, single-stage model, SNOPT and IPOPT successfully solved all problems, and iterative ADMM solved all but one (where it stopped due to an iteration limit). In comparison, for the two-stage model, all solvers solved all problems successfully. Boxplots of the distribution of objective values, infinity-norm dynamics constraint error, and runtimes for both formulations are shown in Figure 5.

The two-stage method successfully determined wingstroke signal parameters for all 20 trials. This required successful solutions for both stages; all IPOPT, SNOPT, and ADMM solutions for the control input stage, and all Projected Gradient Descent solutions for the wingstroke parameter stage). For the second stage, wingstroke parameters were found over an average of 18.9 seconds per trial.

The mean absolute difference between the wingstroke parameters found with the single-stage and two-stage formulations are shown in Figure 6. Applying Equations (2)-(5) to find the control inputs from the wingstroke parameter in the two-stage method (the control inputs were provided directly in

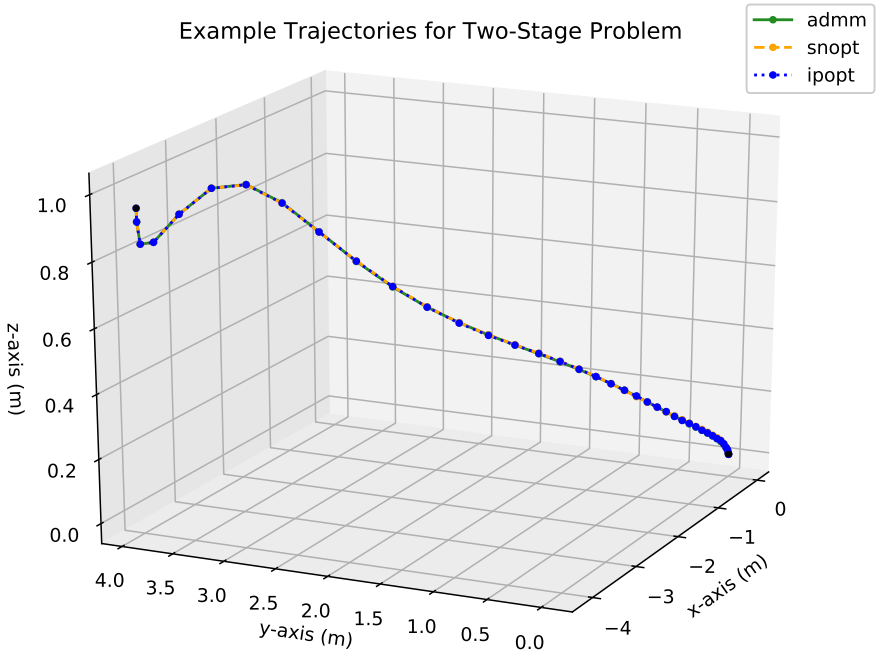

Figure 3: Example of 3-D solution trajectory for a specific instance of the two-stage problem formulation.

the single-stage formulation), we compare the resulting control inputs for each solver across both formulations in Figure 7. We note that across all trajectories, we observed varied behavior (and subsequent agreement) between the single-stage and two-stage methods (which we will discuss further in Section 6.2), but the single trajectory result shown above in Figure 7 is demonstrative of variation between control parameter solutions for across all solvers, and discrepancies and convergence between the single-stage and two-stage formulations.

6 Discussion

6.1 Solver Comparison

Overall, contrary to the anticipation of difficulties due to the high nonlinearity and nonconvexity of the problem, the solvers were found to be very successful. In Figure 2, it is difficult to make out differences between the state trajectories returned by the solvers, as the state values across solvers agree almost perfectly for most problem instances within the single-stage approach. Interestingly, this does not imply agreement in input trajectories, which often differ between solvers. This appears to be a product of the problem formulations: the solvers agree on a global or very good local optimum, which has multiple mappings to corresponding input values. Trajectory structure resulting from the two-stage approach is very similar (such as the example in Figure 3, though IPOPT finds different local minima than the other solvers for some problem instances. There also appears to be closer alignment in (force and torque) input values for matching trajectories, which is intuitive, as the mapping between states and control authority parameters is more direct. The concentration of trajectory waypoints at the origin is a demonstration of the minimum-time objective.

From the boxplots in Figure 5, we can conclude that SNOPT has the highest performance of all of the solvers for both problem approaches: the distribution of objective values is comparable to those found by ADMM and better than those of IPOPT, and it has the fastest runtimes with the lowest constraint error. Runtimes for the ADMM solver are about 1.5 times longer for the two-stage approach than those for the single-stage approach, though this constrast is more comparable for SNOPT and IPOPT,

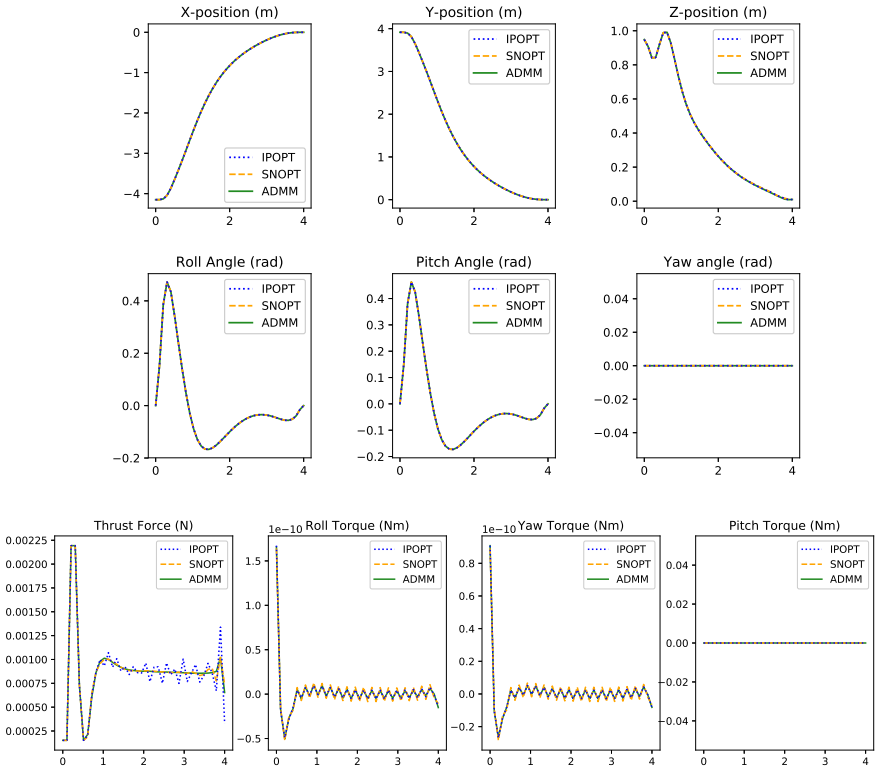

Figure 4: Example of individual state and input trajectories corresponding to the specific instance of the two-stage problem formulation in Figure 3.

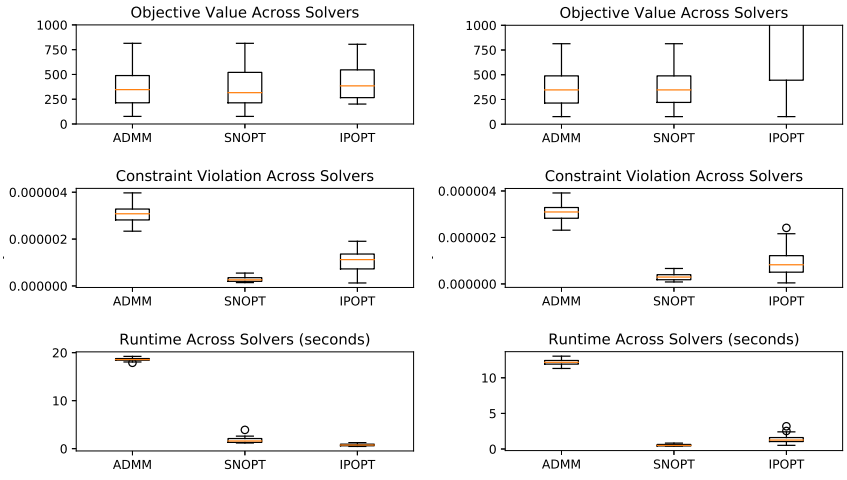

Figure 5: Objective values, infinity-norm dynamics constraint error, and runtime for SNOPT, IPOPT, and the iterative ADMM solver across the successful solves within 20 randomized problem instances. Left figure: single-stage problem. Right figure: two-stage problem.

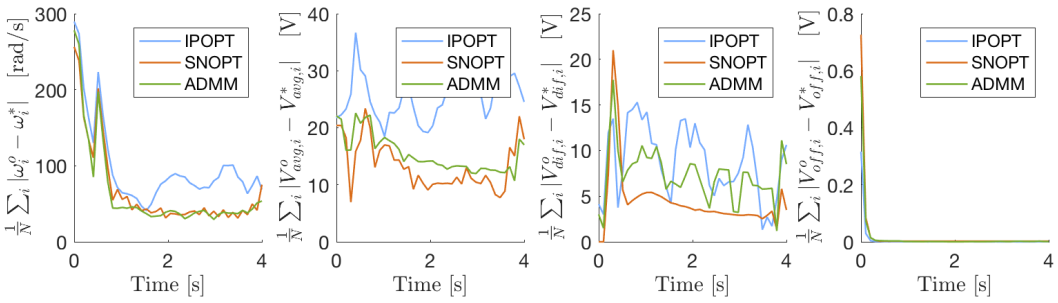

Figure 6: Mean absolute difference between the wingstroke parameters found via the single-stage and two-stage formulations, based on control input results (and corresponding wingstroke parameters) for each optimization method.

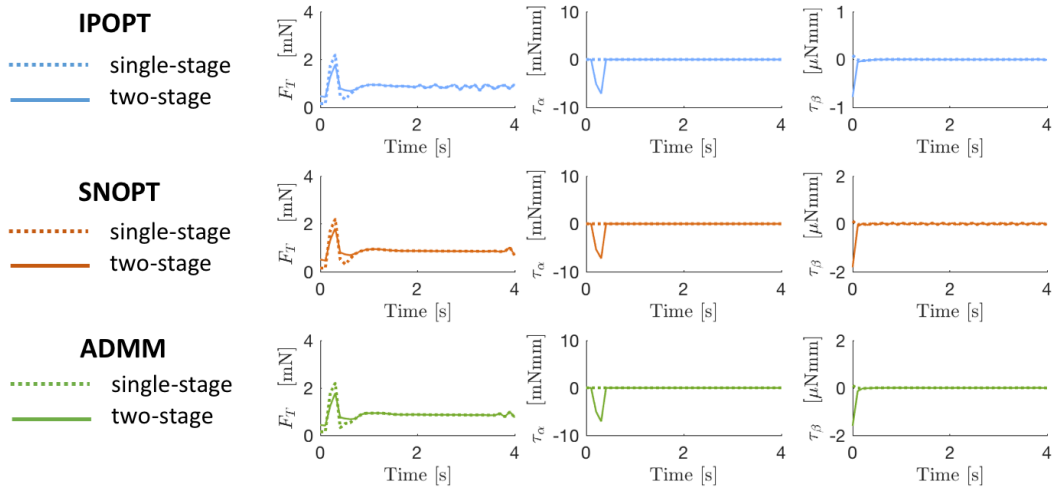

Figure 7: Control inputs found directly via IPOPT, SNOPT, and ADMM in the single-stage formulation, and from the wingstroke parameters in the two-stage formulation, for a single trajectory.

suggesting that they have better infrastructure for handling high nonconvexity. However, the iterative ADMM algorithm is very dependent on the choice of initial penalty parameters, and on their rates of increase, which determine the priorities placed on constraints satisfaction against optimality between subsequent iterations. It is likely that ADMM runtimes can be improved by conducting a parameter sweep for more informed initial parameter values.

Another significant issue during algorithm implementation was problem scaling. Due to the physical scale and mass of the RoboBee, the input parameters needed for a simple path are on the order of 10^{-10} in standard SI units, even when the desired positions are on the order of several meters. Since these values are much lower than the desired tolerance, and close to the order of machine precision when considering squared-input costs, the optimization becomes very poorly conditioned. Initial attempts to run the solvers would lead to solutions with very high-frequency oscillation in input parameters. To mitigate this issue, we scaled input values within the dynamical system such that all inputs were expected to be approximately unity in magnitude, and unscaled after the problem was solved. Some residual oscillation is still visible in the roll and yaw torques of Figure 4.

6.2 Single- vs. Two-Stage Formulation

We may directly compare the results of the single-stage and two-stage formulations (as outlined in Section 3) by observing both the resulting wingstroke signal parameters **u** and the force and torque control inputs **v** (found for the two-stage method via application of Equations (2)-(5)).

First, considering the control parameters $\mathbf{v} = [F_T \quad \tau_\alpha \quad \tau_\beta]$, we note (as observed in Figure 7) that while we observe differences between the control inputs from each strategy (especially within the first second of the trajectory), we largely see convergence and near-identical agreement between the control inputs from each formulation, validating the use of the both formulations. Considering the differences, however, one particular characteristic to note is that while the single-stage formulation results in a larger initial thrust force spike (across all solvers), the two-stage formulation results in a roll torque spike in the same location. Since the wingstroke parameters V_{avg} and V_{dif} are largely responsible for the magnitudes of both of these control inputs, we may perhaps attribute this phenomenon to each formulation attempting to appropriately balance V_{avg} and V_{dif} between these two control inputs.

The control inputs shown in Figure 7, however, are from a single trajectory (of twenty total). While we observed a similar balance phenomenon across all trajectories, we note that the agreement and convergence between the single- and two-stage formulations varied, most significantly when an extreme initial condition $\mathbf{x}_0$ resulted in railing of input parameters against their defined limits. Overall, however, we observed a similarly convergent trend throughout other trajectories.

Finally, comparing the resulting wingstroke signal parameters $\mathbf{u} = [\omega \quad V_{avg} \quad V_{dif} \quad V_{off}]$ between the single- and two-stage formulations, we observe high variability between the results of the two methods, as demonstrated in Figure 6. While we note that (based on the control result) it is reasonable to assume that the differences between wingstroke parameter solutions do not necessarily indicate drastic differences in control input, we do expect the independent solutions here to differ in power efficiency: input power is proportional to $\omega * V^2$ [10], so different solutions will certainly vary in input power. While we would therefore benefit from including power constraints in future objective functions, we expect a variety of parameter solutions using the current formulations.

6.3 Implications for Control

While the run times for the optimization methods described herein are prohibitive for direct application in a RoboBee control scheme (which requires a feedback rate on the order of 250 Hz, at minimum [9]), we expect that finding optimal landing trajectories (and corresponding control and wingstroke inputs) will be directly applicable in creating ideal reference trajectories for the RoboBee to attempt during flight: The methods described herein create trajectories within the bounds of feasible flight, therefore encouraging an optimal landing trajectory tailored to the vehicle. While during flight the MAV will certainly still be subject to unpredictable disturbances (thus precluding simply applying the control inputs from the optimal trajectories), we expect that the trajectory optimizations in this work will provide a suitable reference for aggressive maneuvers, including but not limited to landing.

7 Conclusions

In this work, we sought to design and evaluate strategies to provide optimal landing trajectories for the Harvard RoboBee. We considered optimal solutions in terms of input control parameters (forces and torques) and wingstroke signal parameters (flapping frequency and signal voltage parameters). We applied commercial solvers (IPOPT and SNOPT) and an iterative ADMM-based method to solve for control inputs, and branched to consider single- and two-stage formulations to find optimal wingstroke signal parameters (where the single-stage formulation expanded the initial solver process to include the wingstroke parameters, and the two-stage process applied a Projected Gradient Descent method to find wingstroke parameters from predetermined optimal control inputs.

We found that landing trajectories could be successfully determined using the methods described above, generating physically-reasonable control inputs for the RoboBee across all optimization strategies, with agreement in control solutions for the single- and two-stage formulations and varied corresponding wingstroke parameters. In particular, SNOPT was found to have the best performance, with lower runtimes and constraint errors while arriving at comparable objective values to IPOPT and ADMM. Additionally, we found that due to the physical scale of the RoboBee system, problem scaling was vital for successful optimization.

We expect that the methods herein will be directly applicable to trajectory design for RoboBee landing, and may be expanded to apply to a larger set of maneuvers, particularly in the design of ideal reference trajectories which cater to the physical constraints of the system. Future work should also

consider restructuring of the cost functions related to wingstroke parameters in both formulations in order to consider efficiency, to reach a single set of optimal parameters in lieu of a variety of parameter sets with similar control outputs.

Overall, the work presented herein will guide landing trajectories for the RoboBee to assist in robust autonomous flight.

References

[1] Yufeng Chen, Kevin Ma, and Robert J Wood. Influence of wing morphological and inertial parameters on flapping flight performance. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2329–2336. IEEE, 2016.

[2] Pakpong Chirarattananon, Kevin Y Ma, and Robert J Wood. Adaptive control for takeoff, hovering, and landing of a robotic fly. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3808–3815. IEEE, 2013.

[3] Pakpong Chirarattananon, Kevin Y Ma, and Robert J Wood. Fly on the wall. In *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 1001–1008. IEEE, 2014.

[4] CP Ellington. The aerodynamics of hovering insect flight. 5. a vortex theory. *Philosophical Transactions of the Royal Society of London Series B-Biological Sciences*, 305(1122):115–144, 1984.

[5] Philip E Gill and Michael W Leonard. Reduced-hessian quasi-newton methods for unconstrained optimization. *SIAM Journal on Optimization*, 12(1):209–237, 2001.

[6] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[7] MA Graule, P Chirarattananon, SB Fuller, NT Jafferis, KY Ma, M Spenko, R Kornbluh, and RJ Wood. Perching and takeoff of a robotic insect on overhangs using switchable electrostatic adhesion. *Science*, 352(6288):978–982, 2016.

[8] Quan Gu, Michele Barbato, Joel P Conte, Philip E Gill, and Frank McKenna. Opensees-snopt framework for finite-element-based optimization of structural and geotechnical systems. *Journal of Structural Engineering*, 138(6):822–834, 2011.

[9] E Farrell Helbling and Robert J Wood. A review of propulsion, power, and control architectures for insect-scale flapping-wing vehicles. *Applied Mechanics Reviews*, 70(1):010801, 2018.

[10] Noah T Jafferis, Moritz A Graule, and Robert J Wood. Non-linear resonance modeling and system design improvements for underactuated flapping-wing vehicles. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3234–3241. IEEE, 2016.

[11] P Jain, LT Biegler, and MS Jhon. Optimization of polymer electrolyte fuel cell cathodes. *Electrochemical and Solid-State Letters*, 11(10):B193–B196, 2008.

[12] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

[13] Kevin Y Ma, Samuel M Felton, and Robert J Wood. Design, fabrication, and modeling of the split actuator microrobotic bee. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1133–1140. IEEE, 2012.

[14] Helena Sofia Rodrigues, M Teresa T Monteiro, and Delfim FM Torres. Optimization of dengue epidemics: a test case with different discretization schemes. In *AIP Conference Proceedings*, volume 1168, pages 1385–1387. AIP, 2009.

[15] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[16] J. P. Whitney and R. J. Wood. Conceptual design of flapping-wing micro air vehicles. *Bioinspiration & biomimetics*, 7(3):036001, 2012.

[17] Robert J. Wood, E Steltz, and R. S. Fearing. Optimal energy density piezoelectric bending actuators. *Sensors and Actuators A: Physical*, 119(2):476–488, 2005.