Two-Stage Trajectory Optimization for Flapping Flight with Data-Driven Models

Jonathan Hoff[1] and Joohyung Kim[1]

***Abstract*— Underactuated robots often require involved routines for trajectory planning due to their complex dynamics. Flapping-wing aerial vehicles have unsteady aerodynamics and periodic gaits that complicate the planning procedure. In this paper, we improve upon existing methods for flight planning by introducing a two-stage optimization routine to plan flapping flight trajectories. The first stage solves a trajectory optimization problem with a data-driven fixed-wing approximation model trained with experimental flight data. The solution to this is used as the initial guess for a second stage optimization using a flapping-wing model trained with the same flight data. We demonstrate the effectiveness of this approach with a bat robot in both simulation and experimental flight results. The speed of convergence, the dependency on the initial guess, and the quality of the solution are improved, and the robot is able to track the optimized trajectory of a dive maneuver.**

I. INTRODUCTION

Many robotics applications rely on trajectory planning to determine feasible and optimal paths for them to follow [1]–[3]. These systems require high levels of both speed and accuracy, and this can be challenging because they are often inversely related and trade offs must be considered. In the field of flapping flight, trajectory planning is particularly difficult because flapping dynamics are time-varying due to the periodic gait from flapping. Past works in flapping flight have used the technique called averaging to approximate the dynamics without the time-varying component by averaging the net forces over a wingbeat [4]–[6]. This technique has been very successful for control and prediction of flapping-wing micro aerial vehicles (FWMAV), as the wingbeat frequencies of these small systems are much faster than the body dynamics. However, this technique is less accurate with larger flapping systems in which the ratio of the flapping frequency to the natural body frequency is small (< 50), such as the hawkmoth flapping at 26 Hz [7]. Additionally, it can be difficult to incorporate the effects from forward velocity and body orientation.

Consequently, larger flapping systems require the use of time-varying models with periodic flapping for applications that need higher levels of accuracy like flying through a narrow gap. Another layer of difficulty is added when the time duration of the trajectory is a decision variable in the optimization. Changing this value results in changing the number of wingbeats of the trajectory, assuming the flapping frequency remains constant, and this can cause difficulties with convergence. The averaging method may improve speed but suffer from critical inaccuracies, while the full dynamics model will improve the accuracy but suffer from higher computation time and even local optimality.

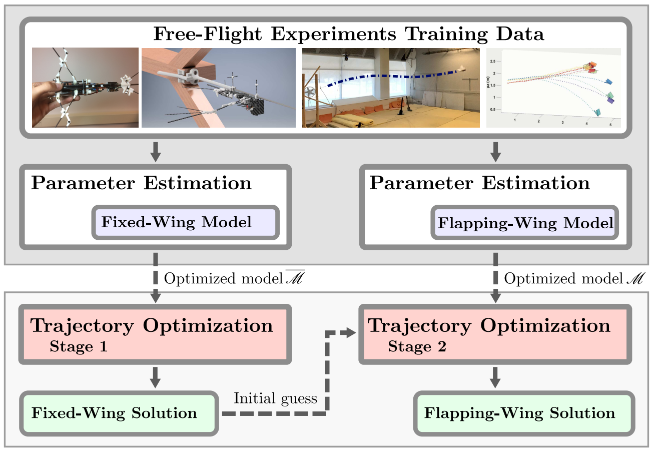

Free-Flight Experiments Training Data
Parameter Estimation
Fixed-Wing Model
Parameter Estimation
Flapping-Wing Model
Optimized model $\overline{\mathscr{M}}$
Optimized model $\mathscr{M}$
Trajectory Optimization Stage 1
Trajectory Optimization Stage 2
Fixed-Wing Solution
Initial guess
Flapping-Wing Solution

Fig. 1: Flow chart depicting steps of parameter estimation of models and the two-stage trajectory optimization formulation.

In this paper, we propose a two-stage optimization approach that uses both fixed-wing (similar to averaging) and flapping-wing data-driven models to reduce the computation time of generating trajectories while maintaining solution accuracy. Figure 1 presents the steps of this approach. We use the parameter estimation formulation and data set from [8] to train the fixed-wing and flapping-wing models. The first stage of the method solves the trajectory optimization problem using the fixed-wing model, and the second stage uses this solution as the initial guess to the next optimization with the flapping-wing dynamics model. We apply this method to Bat Bot (B2), a bio-inspired robot mimicking bat flight [9]–[12]. Through rigorous simulation, we demonstrate that this two-stage approach outperforms a single-stage approach in terms of computation time, optimality of the final result, and invariance to poor initial guesses. Open-loop tracking experimental results with B2 demonstrate the applicability of this method on physical systems. A supplementary video[1] describes the methods and results of this paper.

A. Background

The works in this paper are related to those in [13]. Hassan and Taha used an averaging model to supply the initial guess to a shooting method with the full model for analyzing stability characteristics of a FWMAV constrained on two vertical

[1]Authors are with the Coordinated Science Laboratory and Department of Electrical and Computer Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA. `jehoff2@illinios.edu`

[1]`https://youtu.be/-shurpI4kzw`

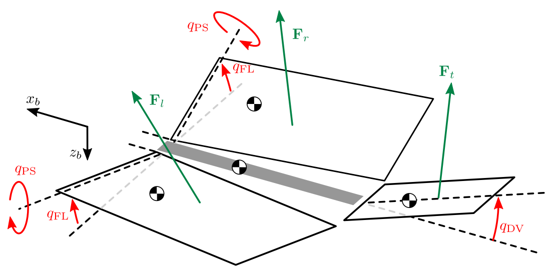


Fig. 2: Planar model of B2 dynamics and aerodynamics [25].

rails. The fixed-wing model for B2 is similar to the averaging method because the model approximates the thrust produced from flapping as a time-invariant external force. However, we introduce a fixed aerodynamic surface that incorporates the effects of forward velocity and body orientation which are nontrivial for this system. These effects can be ignored in hovering FWMAVs [14], [15]. Additionally, the model is data-driven: it is trained using the free-flight data and parameter estimation methods presented in [8].

Similar research has been performed in the legged systems community. Reduced-order models such as the linear inverted pendulum [16] have become essential for trajectory planning in the area of bipedal locomotion [17]. Past works have used a combination of a reduced-order model and the full model to simplify the planning problem [3]. Marcucci *et al.* [18] developed a two-stage procedure for humanoid models in which the first stage solves a direct collocation problem with simplifications such as reducing the number knot points, using a pseudo-static model, and relaxing contact parameters. This solution is used to warm-start the second stage.

While trajectory planning methods have been extensively studied for fixed and rotary-winged systems, there have been limited works in the realm of planning for flapping flight [19]–[25]. Only a few of these have used optimization-based approaches for generating feasible trajectories [22]–[25]. Solving this problem is important for extending the abilities of these flapping systems to allow for more complex maneuvers and operation. Specifically for mimicking bat flight, maneuvers like banked turning [26], upside-down perching [27], and recovery from fall [28] have sophisticated movement strategies. Improving the speed of trajectory planning for flapping systems is a relatively open problem that has yet to be studied.

II. MODELS

In past works [8], we presented a data-driven model of B2 that was trained using free-flight experiments data. The resulting model showed favorable performance over the existing model as a result of tuning parameters with data. In this work, we utilize these same parameter estimation methods to train a second model of B2 in which the wings are fixed and the model accounts for thrust.

A. *Flapping-wing model*

The planar longitudinal model of B2 is displayed in Figure 2. This model consists of four rigid links: two wings, a tail, and a body. The moment of inertia of each is considered, making this a multi-body system. The wings are coupled and flap up and down on revolute joints. The flapping angle between the body xy plane and each wing is labeled q_{FL}. The rotation of the wings about the span-wise axis is called pronation-supination, and this angle is denoted q_{PS}. The last actuated degree of freedom (DoF) is the dorsoventral movement of the tail, and this angle between the body and the tail surface is labeled q_{DV}. The unactuated coordinates are written with respect to the center of mass (CoM) of the body link. The pitching up and down is written as q_y, x translation (horizontal) is p_x, and the z translation (altitude) is p_z. These coordinates along with the inputs driving the actuated coordinates are grouped in the configuration variable and input vectors

$$\begin{aligned}\mathbf{q} &= \begin{bmatrix} q_y & p_x & p_z & q_{\mathrm{FL}} & q_{\mathrm{PS}} & q_{\mathrm{DV}} \end{bmatrix}^\top \\ \mathbf{u} &= \begin{bmatrix} u_{\mathrm{FL}} & u_{\mathrm{PS}} & u_{\mathrm{DV}} \end{bmatrix}^\top .\end{aligned} \tag{1}$$

As in our past works [8], [25], we use the flapping-wing aerodynamic model structure proposed by Wang [29], [30]

$$C_L = C_{L_1} \sin 2\alpha, \quad C_D = C_{D_0} + C_{D_1} \cos 2\alpha \tag{2}$$

for the lift C_L and drag C_D coefficients with constants C_{L_1}, C_{D_0}, and C_{D_1}. Additionally, we include the coupling term between the wings and tail $\bar{q}^r_{\mathrm{PS}}(t) = q^r_{\mathrm{PS}}(t) + a_{\mathrm{coup}} q_{\mathrm{DV}}(t)$, the tail mapping function, and the body plate given their improvements in model accuracy [8]. The coupling term with constant a_{coup} accounts for the modeling choice of using separate wing and tail aerodynamic surfaces. The tail mapping function accounts for the high sensitivity of movement about $q_{\mathrm{DV}} = 0°$.

The dynamics of this system are written as

$$D(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = B\mathbf{u} + \Gamma(\mathbf{q},\dot{\mathbf{q}}), \tag{3}$$

where D is the inertial matrix, C is the Coriolis matrix, G is the gravity vector, B is the matrix mapping inputs to configuration variables, and Γ is the aerodynamic force vector mapped to the configuration space via virtual work. The input $\mathbf{u}$ enforces the periodic reference trajectories of q_{FL} and q_{PS} as

$$\begin{aligned} q^r_{\mathrm{FL}}(t) &= a_{\mathrm{FL}} \sin(\omega_{\mathrm{FL}} t + b_{\mathrm{FL}}) + c_{\mathrm{FL}} \\ q^r_{\mathrm{PS}}(t) &= a_{\mathrm{PS}} \sin(\omega_{\mathrm{FL}} t + b_{\mathrm{PS}}) + c_{\mathrm{PS}} + a_{\mathrm{coup}} q_{\mathrm{DV}}(t). \end{aligned} \tag{4}$$

The constant ω_{FL} is the flapping frequency of the system. The flapping model is denoted $\mathscr{M}$. Its model parameters C_{L_1}, C_{D_1}, C_{D_0}, and a_{coup} along with w_c (wing chord length), t_s (tail span), A_b (area of body plate), and c^+_{DV} (tail function mapping parameter) are grouped into the vector P and selected using the optimization methods from [8]. This method is summarized in the following section.

B. *Fixed-wing model*

The fixed-wing model differs from the flapping model in two ways. First, the wingbeat frequency is set to $\omega_{\mathrm{FL}} = 0$ such that the wings are in the fixed configuration. Second, an additional term is added to the model to incorporate the average affect of thrust from flapping the wings at some fixed frequency ω_{FL} with some dependency of the speed of flight v_b. This thrust term is given as

$$\|F_T\| = c_{\mathrm{T}} - c_{\mathrm{D}} v_b^2, \tag{5}$$

and it acts at the body CoM position aligned with the body x-axis (forward toward the nose). The fixed-wing model parameter vector $\overline{P}$ includes the elements of P and the constants c_{T} and c_{D}.

We train this fixed-wing model of B2 using the data set and parameter estimation methods in [8]. This data set consists of 43 free-flight experiments of 1 s duration that were captured at 100 Hz with eight Vicon T40 motion capture cameras in the Intelligent Robotics Laboratory (IRL) flight arena at the University of Illinois Urbana-Champaign (UIUC). The orientation data was estimated with the VectorNav VN-100 inertial measurement unit (IMU) on B2 and was fused with the position data from the Vicon system. The data set is broken up into a training subset of $n_{\mathrm{train}} = 13$ flights and a testing subset of the remaining $n_{\mathrm{test}} = 30$ flights.

We minimize the multistep prediction error with the objective function

$$\mathscr{J}_P(\overline{P}) = \sum_{i=1}^{n_{\mathrm{train}}} \sum_{k=0}^{N-1} \left\| \bar{\mathbf{x}}^i_{\mathrm{sim}}(t_k) - \bar{\mathbf{x}}^i_{\mathrm{exp}}(t_k) \right\|^2, \tag{6}$$

where $\bar{\mathbf{x}} = \begin{bmatrix} q_y & p_x & p_z & \dot{q}_y & \dot{p}_x & \dot{p}_z \end{bmatrix}^\top$ is the underactuated CoM coordinates. The simulated states $\mathbf{x}^i_{\mathrm{sim}}(t_k)$ at each time step t_k of trial i were generated using forward Euler integration with a step size of 0.001 s. The simulation was provided the initial condition of the experimental data $\mathbf{x}^i_{\mathrm{exp}}(t_0)$ and the actuator trajectories over the flight duration.

We solved the nonlinear programming problem using the interior-point algorithm of MATLAB's *fmincon*. The original model parameters from [25] were the initial guess for $\overline{P}$. The new fixed-wing model with optimized parameters $\overline{P}$ is denoted as $\overline{\mathscr{M}}$. The resulting matrices $\overline{D}$, $\overline{C}$, $\overline{G}$, and $\overline{\Gamma}$ are functions of $\overline{P}$. We evaluate the model performance using the normalized root-mean-square error (NRMSE) metric $\mathrm{NRMSE}_{i,j} = \frac{1}{x_{j,\max} - x_{j,\min}} \sqrt{\sum_{k=0}^{N-1} (x^i_j(t_k) - \hat{x}^i_j(t_k))^2 / N}$ of the long-term prediction error between each underactuated state $x^i_j \in \bar{\mathbf{x}}$ of the experimental data and $\hat{x}^i_j \in \bar{\mathbf{x}}$ of the model prediction for flight test number i. The terms $x_{j,\min}$ and $x_{j,\max}$ are the state upper and lower bounds to normalized the states. We average the NRMSE for all of the states $\bar{\mathbf{x}}$ for a given flight i with the equation $\mathrm{NRMSE}_i = \sum_{j=1}^{n_{\bar{\mathbf{x}}}} \mathrm{NRMSE}_{i,j} / n_{\bar{\mathbf{x}}}$.

In our analyses, we compute the NRMSE_i for the fixed-wing and flapping-wing models before and after optimization, and we plot this against the average q_{DV} angle of a given flight in Figure 3. The optimized results for both the fixed-wing and flapping-wing models outperform those not optimized for the range of different tail configurations. The NRMSE results for the optimized fixed-wing case show good performance, close to that of the flapping-wing model.

III. Trajectory Optimization

Using the fixed-wing and flapping-wing models from the previous section, we set up the optimization problems to be solved for the first and second stages of our method.

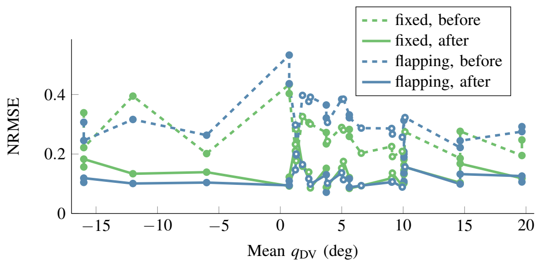


Fig. 3: NRMSE for fixed and flapping models before and after parameter estimation. Each flight for a given model is plotted against the average q_{DV} position. Filled circles represent flights with q_{DV} fixed at some angle, and open circles denote flights with q_{DV} varying in position.

A. Flapping-wing model

We enforce the flapping-wing dynamics (3) using the direct collocation Hermite-Simpson separated discretization [31]. We discretize the states and inputs into N knot points, and we formulate the dynamics as equality constraints

$$\begin{aligned} \mathbf{c}_1: &\quad D_k \ddot{\mathbf{q}}_k + C_k \dot{\mathbf{q}}_k + G_k - B\mathbf{u}_k - \Gamma_k = \mathbf{0} \\ \mathbf{c}_2: &\quad \begin{cases} \mathbf{x}_{k+\frac{1}{2}} - \frac{1}{2}(\mathbf{x}_k + \mathbf{x}_{k+1}) - \frac{h_k}{8}(\dot{\mathbf{x}}_k - \dot{\mathbf{x}}_{k+1}) = \mathbf{0} \\ \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{6}(\dot{\mathbf{x}}_k + 4\dot{\mathbf{x}}_{k+\frac{1}{2}} + \dot{\mathbf{x}}_{k+1}) = \mathbf{0}, \end{cases} \end{aligned} \tag{7}$$

where $\mathbf{x}_k = \begin{bmatrix} \mathbf{q}_k & \dot{\mathbf{q}}_k \end{bmatrix}^\top$, $\dot{\mathbf{x}}_k = \begin{bmatrix} \dot{\mathbf{q}}_k & \ddot{\mathbf{q}}_k \end{bmatrix}^\top$, and $h_k = (t_{k+1} - t_k)$. Here, we have introduced the midpoints between knots $\mathbf{x}_{k+\frac{1}{2}}$, $\dot{\mathbf{x}}_{k+\frac{1}{2}}$, and $\mathbf{u}_{k+\frac{1}{2}}$ as extra decision variables in the optimization. The dynamics constraints $\mathbf{c}_1$ act at the $k = 0, 1, \cdots, N-1$ knot points, and the Hermite-Simpson constraints $\mathbf{c}_2$ enforce the continuity of the trajectory in between knot points. We add $\ddot{\mathbf{q}}_k$ at each knot point as decision variables to remove the need for explicit derivation of the Jacobian of the dynamics in the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, as well as to avoid computing the matrix inverse of D [32]. This separates the dynamics constraints $\mathbf{c}_1$ from the Hermite-Simpson constraints $\mathbf{c}_2$. We note that q_{FL}, $\dot{q}_{\mathrm{FL}}$, $\ddot{q}_{\mathrm{FL}}$, q_{PS}, $\dot{q}_{\mathrm{PS}}$, and $\ddot{q}_{\mathrm{PS}}$ are directly set to their corresponding reference trajectories from (4). This embeds the reference trajectory constraints in the optimization. As such, $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are functions of t_f.

B. Fixed-wing model

The fixed-wing model uses a similar form of constraints as (7), with some modifications. First, we use the fixed-wing model dynamics and write the constraints as

$$\bar{\mathbf{c}}_1: \quad \overline{D}_k \ddot{\mathbf{q}}_k + \overline{C}_k \dot{\mathbf{q}}_k + \overline{G}_k - B\mathbf{u}_k - \overline{\Gamma}_k = \mathbf{0}. \tag{8}$$

Second, the reference trajectories embedded in the dynamics are $q_{\mathrm{FL}}(t_k) = c_{\mathrm{FL}}$ and $q_{\mathrm{PS}}(t_k) = \overline{a_{\mathrm{coup}}} q_{\mathrm{DV}_k} + c_{\mathrm{PS}}$ because $\omega_{\mathrm{FL}} = 0$. Third, the number of knot points are reduced to $\overline{N} = \alpha N$ ($\alpha < 1$). This reduction is a critical element of this method because without the flapping dynamics, significantly fewer knots are needed to enforce the dynamics because the fast oscillations from flapping are removed. This reduces the computation time of this stage, and in turn, the overall method.

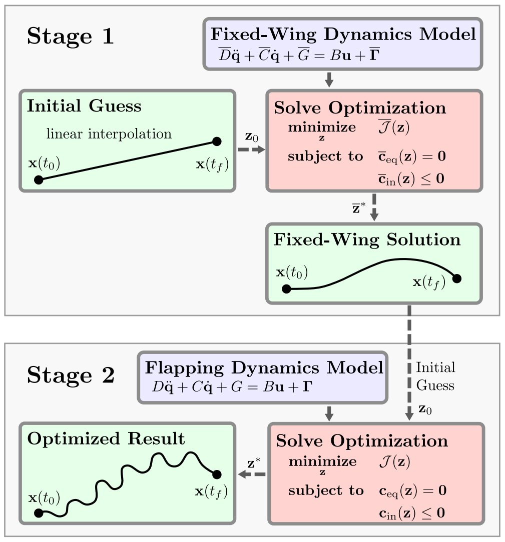

Stage 1
Fixed-Wing Dynamics Model
$\overline{D}\ddot{\mathbf{q}} + \overline{C}\dot{\mathbf{q}} + \overline{G} = B\mathbf{u} + \overline{\Gamma}$
Initial Guess
linear interpolation
$\mathbf{x}(t_0)$
$\mathbf{x}(t_f)$
$\mathbf{z}_0$
Solve Optimization
minimize $\overline{\mathcal{J}}(\mathbf{z})$
subject to $\overline{\mathbf{c}}_{\text{eq}}(\mathbf{z}) = \mathbf{0}$
$\overline{\mathbf{c}}_{\text{in}}(\mathbf{z}) \leq \mathbf{0}$
$\overline{\mathbf{z}}^*$
Fixed-Wing Solution
Stage 2
Flapping Dynamics Model
$D\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + G = B\mathbf{u} + \Gamma$
Initial Guess
Optimized Result
$\mathbf{z}^*$
minimize $\mathcal{J}(\mathbf{z})$
subject to $\mathbf{c}_{\text{eq}}(\mathbf{z}) = \mathbf{0}$
$\mathbf{c}_{\text{in}}(\mathbf{z}) \leq \mathbf{0}$

Fig. 4: Flow chart depicting steps of two-stage trajectory optimization.

C. Two-stage optimization formulation

The steps of the two-stage optimization procedure can be summarized by Figure 4. First, the trajectory optimization problem with the fixed-wing model is solved. The solution to this is used as the initial guess for the trajectory optimization with the flapping model. The solver computes the final solution to this problem.

The first stage of the optimization finds a locally optimal trajectory using the fixed-wing model $\overline{\mathcal{M}}$. We set up the boundary conditions and decision variables upper and lower bounds as

$$\begin{aligned} \mathbf{c}_3: &\quad \underline{z}_i \leq z_i \leq \overline{z}_i,\ i = 0,\ldots,n_z-1 \\ \mathbf{c}_4: &\quad |x_i(t_0) - x_i^d(t_0)| \leq \varepsilon_i^0,\ x_i \in \mathbf{x} \\ \mathbf{c}_5: &\quad |x_i(t_f) - x_i^d(t_f)| \leq \varepsilon_i^f,\ x_i \in \mathbf{x}. \end{aligned} \tag{9}$$

The constraints $\mathbf{c}_3$ are the upper and lower bounds of the decision variables, for example $-\pi/2 \leq q_y(t_k) \leq \pi/2$. The constraints $\mathbf{c}_4$ are the boundary conditions for the initial state with slackness ε^0, and $\mathbf{c}_5$ are the conditions for the final state with slackness ε^f. These boundary values are selected depending on the trajectory type in Section IV.

We can separate the constraints into equality $\overline{\mathbf{c}}_{\text{eq}}$ and inequality $\overline{\mathbf{c}}_{\text{in}}$ constraint vectors

$$\overline{\mathbf{c}}_{\text{eq}} = \begin{bmatrix}\overline{\mathbf{c}}_1^\top & \mathbf{c}_2^\top\end{bmatrix}^\top,\ \overline{\mathbf{c}}_{\text{in}} = \begin{bmatrix}\mathbf{c}_3^\top & \mathbf{c}_4^\top & \mathbf{c}_5^\top\end{bmatrix}^\top. \tag{10}$$

Given these constraints, we propose the optimization

$$\begin{aligned} \underset{\mathbf{z}}{\textbf{minimize}} &\quad \overline{\mathcal{J}}(\mathbf{z}) = \sum_{k=0}^{\overline{N}-2} \frac{h_k}{2}\left(\ddot{q}_{\text{DV}}(t_{k+1})^2 + \ddot{q}_{\text{DV}}(t_k)^2\right) \\ \textbf{subject to} &\quad \overline{\mathbf{c}}_{\text{eq}}(\mathbf{z}) = \mathbf{0}, \quad \overline{\mathbf{c}}_{\text{in}}(\mathbf{z}) \leq \mathbf{0}. \end{aligned} \tag{11}$$

The cost function $\overline{\mathcal{J}}$ penalizes the control effort spent moving the hindlimbs. The decision variables of the optimization are $\mathbf{z} = \begin{bmatrix}t_f & \mathbf{z}_0^\top & \cdots & \mathbf{z}_k^\top & \cdots & \mathbf{z}_{N-1}\end{bmatrix}$, where $\mathbf{z}_k = \begin{bmatrix}\mathbf{q}_k^\top & \dot{\mathbf{q}}_k^\top & \ddot{\mathbf{q}}_k^\top & \mathbf{u}_k^\top\end{bmatrix}^\top$ and t_f is the final time of the trajectory, assuming the initial time $t_0 = 0$. The time at each knot point is given as $t_k = \frac{k}{N-1}t_f$. The number of decision variables is equal to $n_z = 1 + (2\overline{N})(3n_q + n_u)$, where $n_q = 6$ is the number of configuration variables and $n_u = 3$ is the number of inputs. We construct the initial guess for the decision variables $\mathbf{z}$ by linear interpolation between the initial $\mathbf{x}_0$ and final $\mathbf{x}_f$ states from the boundary conditions. We solve this nonlinear optimization problem with IPOPT [33], a general purpose solver that excels at solving trajectory optimization problems. We supply the analytically derived constraints Jacobian to the optimizer along with the sparsity structure to reduce computation time.

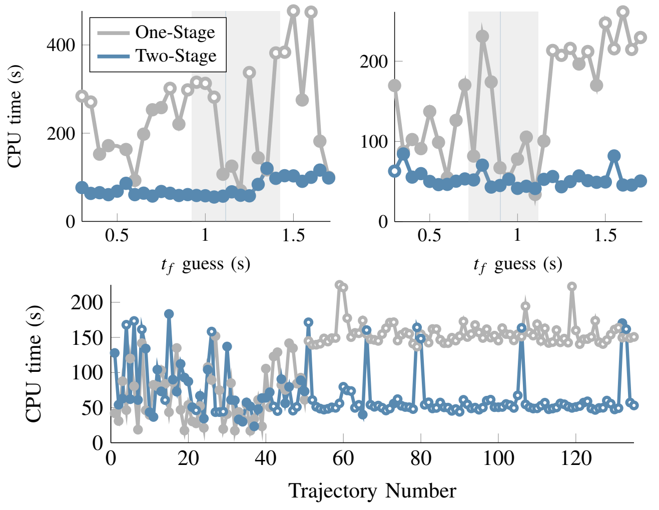


Fig. 5: Comparison of one-stage and two-stage approaches for changing the final time t_f guess given to the optimizer for launch (top-left) and dive (top-right) trajectories, and for varying the boundary conditions for a trajectory (bottom). Open circles denote cases where the max iteration threshold was reached before convergence. The shading shows the mean ± standard deviation of the optimized value of t_f for all cases that converged.

If the optimization converges and finds a feasible solution $\overline{\mathbf{z}}^*$, this solution is cubically interpolated at N knot points and used as the initial guess for the decision variables of the optimization with the flapping-wing model. This problem uses the objective function $\mathcal{J}(\mathbf{z})$ which matches $\overline{\mathcal{J}}(\mathbf{z})$ except N replaces $\overline{N}$. The constraints are

$$\mathbf{c}_{\text{eq}} = \begin{bmatrix}\mathbf{c}_1^\top & \mathbf{c}_2^\top\end{bmatrix}^\top,\ \mathbf{c}_{\text{in}} = \begin{bmatrix}\mathbf{c}_3^\top & \mathbf{c}_4^\top & \mathbf{c}_5^\top\end{bmatrix}^\top. \tag{12}$$

This second stage is run only if the first stage converges because the simulation results in the following section have demonstrated the second stage typically will not converge if the first fails to converge. If running a significant number of tests, eliminating the second stage provides large computational savings.

IV. RESULTS

We compared the performance of the two-stage optimization with the previous one-stage method, i.e. running stage two without the initial guess produced from stage one. We considered the effects of the initial guess of t_f, given the challenges of time scaling in flapping flight, by running the optimization for a range of different t_f guesses for a launch trajectory and a dive trajectory. Then, we ran the optimization

TABLE I: Simulation results comparing one and two stage methods for number of cases run (N), mean cost of converged cases ($\mathscr{J}(\mathbf{z}^*)$), percent of cases that converged (C), mean CPU time for cases that converged ($\bar{t}_c$), and mean time for cases that reached the maximum number of iterations before convergence ($\bar{t}_m$).

Type	Case	N	$\mathscr{J}(\mathbf{z}^*)$	C	$\bar{t}_c$	$\bar{t}_m$
Launch	One Stage	29	2.4e+03	55%	164.4 s	343.3 s
	Two Stage	29	**2.4e-01**	**100%**	**75.3 s**	**0 s**
Dive	One Stage	29	1.0e+05	69%	116.5 s	224.2 s
	Two Stage	29	**2.3e+02**	**97%**	**52.5 s**	**63.0 s**
Varied	One Stage	135	5.7e+04	**37%**	**69.0 s**	155.4 s
	Two Stage	135	**4.4e+03**	30%	74.8 s	**64.3 s**

for a variety of different initial and final states to get a rich set of trajectories to demonstrate the improved performance of the two-stage method over a wider range of flight conditions. Additionally, we varied the knot factor α over the different simulations. All simulations were run on a Windows laptop with an i7 4600U processor and 16GB of RAM using MATLAB 2014a. Finally, we performed experimental flight results tracking the dive maneuver trajectory.

A. Simulation results

The launch trajectory was set by the boundary conditions $q_{y_0} = 0°$, $p_{x_0} = 0$, $p_{z_0} = 0$, $p_{x_f} \in [p^{\text{d}}_{x_f} \pm \varepsilon_1]$, $p_{z_f} \in [p^{\text{d}}_{z_f} \pm \varepsilon_2]$, $\dot{q}_{y_0} = 7\,\text{rad/s}$, $\dot{p}_{x_0} = v_0 \cos(\theta_0)$, $\dot{p}_{z_0} = v_0 \sin(\theta_0)$, $\dot{q}_{\text{DV}_0} = 0$, and $\dot{q}_{\text{DV}_f} = 0$. The initial launch velocity was $v_0 = 9\,\text{m/s}$, and the initial launch angle (i.e. direction of velocity) was $\theta_0 = 2°$. The desired final position was $p^{\text{d}}_{x_f} = 6.5\,\text{m}$ and $p^{\text{d}}_{z_f} = 0.75\,\text{m}$. The tolerances $\varepsilon_1, \varepsilon_2 = 0.01\,\text{m}$ gave a small amount of slackness to the optimization. There were $N = 36$ knot points used in the optimization with knot factor $\alpha = 1/2$. We set the boundary constraints of the dive trajectory as those defined for the launch maneuver with the addition of $q_{\text{DV}_0} = 4°$ to force more tail movement over the trajectory for a dynamically different maneuver, a lower value of p_{z_f} to force the robot to dive downward, and $q_{y_f} \in [10 \pm 2°]$ to ensure the robot recovered its orientation at the end of the maneuver. The values for these boundary conditions were $v_0 = 6.5\,\text{m/s}$, $\theta_0 = 4°$, $q_{y_0} = 12°$, $\dot{q}_{y_0} = 6\,\text{rad/s}$, $p^{\text{d}}_{x_f} = 4\,\text{m}$, $p^{\text{d}}_{z_f} = 0\,\text{m}$, $\varepsilon_1, \varepsilon_2 = 0.01\,\text{m}$, $N = 31$, and $\alpha = 1/4$. For both the launch and dive maneuvers, we ran the optimization with varying t_f guesses from 0.3 s to 1.7 s at intervals of 0.05 s.

Using the same boundary constraints as the launch maneuver, we varied the values of initial and final conditions as $v_0 \in \{6, 7, 8\text{m/s}\}$, $q_{y_0} \in \{-15, -7.5, 0, 7.5, 15°\}$, and $p^{\text{d}}_{z_f} \in \{-2, -1.5, -1, \cdots, 2\text{m}\}$ in order to generate more variation in the type of trajectory. We provided the guess $t_f = \left\| \mathbf{p}_f - \mathbf{p}_0 \right\| / (\frac{1}{2}(v_0 + v_f))$ as a good estimate for the final time, where $\mathbf{p}_0 = \begin{bmatrix} p_{x_0} & p_{z_0} \end{bmatrix}^\top$ and $\mathbf{p}_f = \begin{bmatrix} p_{x_f} & p_{z_f} \end{bmatrix}^\top$. For all tests, we set $\theta_0 = q_{y_0}$, $\dot{q}_{y_0} = 0\,\text{rad/s}$, $p^{\text{d}}_{x_f} = 4.5\,\text{m}$, $\varepsilon_1, \varepsilon_2 = 0.01\,\text{m}$, $N = 21$, and $\alpha = 1/4$. We ran the optimization routine for every combination of v_0, q_{y_0}, and p_{z_f}, resulting in 135 different combinations.

The comparison of the computation time between the one-stage and two-stage methods when varying t_f is displayed in Figure 5 and Table I. The plot demonstrates significant improvements in computation time for the vast majority of the different cases. We noticed that a large number of the cases using the one-stage approach failed to converge within 200 iterations, denoted with unfilled circles at those points. The final results for t_f had a higher variance for the one-stage approach, suggesting that the routine has a higher dependency on the initial condition given to the optimization. In all three types of simulations, the mean objective function $\mathscr{J}(\mathbf{z}^*)$ was improved with the two-stage method.

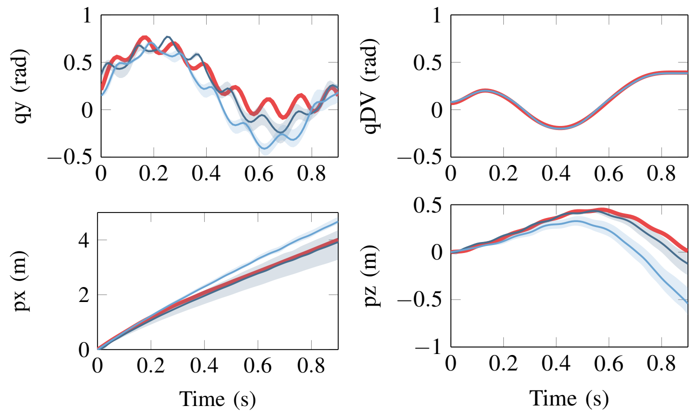
The two-stage approach also reduced computation time for trajectories that were infeasible. Many of the proposed optimization problems for varying boundary conditions were infeasible because of the random combinations of boundary conditions. Table I and Figure 5 show that the two-stage method more quickly determined if the problem would not converge and reach the maximum number of iterations. Stage two would not be run if stage one did not converge, thus reducing computation. In order to validate not running the second stage, we ran stage two for the 84 cases where stage one did not converge, and we found that 91.7% of these cases also did not converge in stage two. Thus, stage two likely will not converge if stage one does not converge. The guess for t_f was close to the optimized result for each of the varied boundary tests, so it was expected that the one-stage method would have similar computation time to the two-stage method for the cases with convergence.

In summary, the primary improvements were robustness to poor guesses for t_f, improved solution quality, and faster solving time.

B. Experimental results

We validated the two-stage approach through experimental open-loop flight tracking tests. We used the dive trajectory generated using the two-stage approach from the simulation results for these experiments. The tail actuator trajectory was embedded on B2's microprocessor, the IMU detected the launch acceleration, and the IMU and Vicon systems recorded the states of the system. The experiments were performed in the IRL flight arena at UIUC. The custom-built launcher [8] accelerated the robot for each test. The launcher was positioned carefully such that the initial conditions of the experiments matched those of the optimized trajectory as close as possible. The hardware and experiments are also shown in the accompanying video.

The optimized trajectory and experiments are displayed in Figure 6. The tail initially tilted up to increase the altitude, then tilted down to bring the robot into a dive, and finally tilted up to increase the pitch to the desired final condition. Due to the difficulty to control the initial condition of the phase offset of flapping angle q_{FL} at launch, we separated the flight results into three sets: flights with the phase of q_{FL} shifted slightly earlier than the optimized trajectory (3 flights), flights with it shifted slightly later (4 flights), and flights with it completely out of phase (13 flights, not displayed in plots). The experiments with close initial



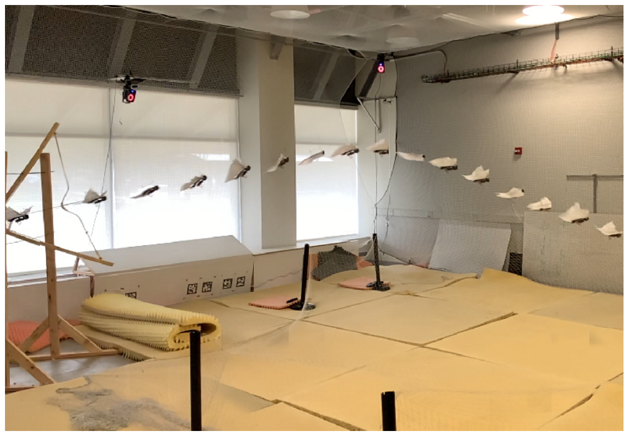


Fig. 6: Open-loop tracking results of 7 flight experiments of the dive maneuver (dark blue for early-phase q_{FL} flights, light blue for late-phase q_{FL} flights) compared with the optimized trajectory (red). The mean of the experiments at each time sample is plotted in solid blue, and the minimum and maximum are plotted in the lighter shadows. The right image displays a single flight result from the early-phase q_{FL} group.

conditions tracked the pitch angle q_y and horizontal position p_x fairly well. The early-phase flights tracked the altitude p_z quite well, but there was some long-term error of the altitude p_z for the flights that were launched slightly late. The data-driven model of B2 has lower accuracy for angles between $-5°$ and $0°$ due to the tail mapping function, and this could explain the deviation of the trajectory p_z when the tail tilted down to this range. Additionally, the flapping frequencies between the actual and optimized varied slightly as the flapping frequency was set by hand, and this contributed to q_y being out of phase.

C. Discussion

One of the contributing factors for failed or slow convergence of the one-stage flapping model optimization is the effect of the reference trajectories when t_f is a decision variable. When t_f is changed by the optimizer, q_{FL} and q_{PS} are consequently changed because they are generated from (4) and depend on $t_k = \frac{k}{N-1}t_f$. As a result, there may be a mismatch between the dynamics constraints at the knot points that previously matched because changing t_f changes the number of wingbeats in the trajectory. First optimizing with the fixed-wing model in which q_{FL} and q_{PS} are independent of t_f simplifies the process of selecting the optimal t_f^* and provides a greater robustness to poor initial guesses for this variable.

The optimization with the fixed-wing model requires considerably less computation time than with the flapping-wing model. When solving this nonlinear program, reducing the number of knot points for the first stage to $\overline{N} = \alpha N$, where $\alpha < 1$, significantly reduces the number of decision variables. Removing the time-periodic elements also results in fewer iterations of the optimization.

While the two-stage method significantly outperforms the one-stage approach, there are two minor limitations. There are a few cases in which the first stage of the two-stage optimization does not converge but the one-stage optimization does converge. A solution may be possible, but the first stage reaches the maximum allowable iterations before convergence and the second stage is not run. Always running the second stage would alleviate this issue, but it comes with the trade off of increasing computation time. Perturbing the initial guess for stage one and rerunning could improve convergence without increasing computation as much. Additionally, while in the majority of cases the objective function is lower for the two-stage method, the one-stage method does have a few cases in which it finds a lower final objective function value. It is likely that in these cases, the first stage gave the second stage an initial guess that results in a local optimimum.

Finally, we note that the physical limitations of B2 prevent more dynamic maneuvers. Given the relatively low thrust-to-weight ratio of the robot, it is difficult for it to follow trajectories like perching, lift off rest, or dynamic turning. Currently, we are developing aspects of the robot to improve its capabilities and allow for complex maneuvers.

V. CONCLUSIONS

We have presented a novel approach for planning flapping flight maneuvers using a two-stage trajectory optimization method with fixed-wing and flapping-wing data-driven models of the bat robot B2 trained using free-flight data. The first stage of the approach uses direct collocation to optimize the trajectory of the fixed-wing model, and the second stage uses the solution to this optimization as the initial guess to the trajectory optimization of the flapping-wing model. Compared to a single-stage method that only optimizes the flapping-wing model, this approach improves computation time and the quality of solution while being more robust to poor initial guesses. The experimental untethered tracking experiments with B2 demonstrate that this approach is applicable to physical systems.

ACKNOWLEDGMENT

The flight experiments were performed in IRL at UIUC.

References

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 2520–2525.

[2] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[3] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[4] L. Schenato, D. Campolo, and S. Sastry, "Controllability issues in flapping flight for biomimetic micro aerial vehicles (MAVs)," in *42nd IEEE International Conference on Decision and Control*, vol. 6. IEEE, 2003, pp. 6441–6447.

[5] X. Deng, L. Schenato, W. C. Wu, and S. S. Sastry, "Flapping flight for biomimetic robotic insects: Part I-system modeling," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 776–788, 2006.

[6] X. Deng, L. Schenato, and S. S. Sastry, "Flapping flight for biomimetic robotic insects: Part II-flight control design," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 789–803, 2006.

[7] H. E. Taha, A. H. Nayfeh, and M. R. Hajj, "Effect of the aerodynamic-induced parametric excitation on the longitudinal stability of hovering mavs/insects," *Nonlinear Dynamics*, vol. 78, no. 4, pp. 2399–2408, 2014.

[8] J. Hoff and S. Hutchinson, "Data-driven modeling of a flapping bat robot with a single flexible wing surface," in *Robotics: Science and Systems*, 2020.

[9] A. Ramezani, X. Shi, S.-J. Chung, and S. Hutchinson, "Bat Bot (B2), a biologically inspired flying machine," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3219–3226.

[10] J. Hoff, A. Ramezani, S.-J. Chung, and S. Hutchinson, "Synergistic design of a bio-inspired micro aerial vehicle with articulated wings," in *Robotics: Science and Systems*, 2016.

[11] A. Ramezani, S.-J. Chung, and S. Hutchinson, "A biomimetic robotic platform to study flight specializations of bats," *Science Robotics*, vol. 2, no. 3, p. eaal2505, 2017.

[12] J. Hoff, A. Ramezani, S.-J. Chung, and S. Hutchinson, "Optimizing the structure and movement of a robotic bat with biological kinematic synergies," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1233–1252, 2018.

[13] A. M. Hassan and H. E. Taha, "Combined averaging–shooting approach for the analysis of flapping flight dynamics," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 2, pp. 542–549, 2018.

[14] M. Keennon, K. Klingebiel, and H. Won, "Development of the nano hummingbird: A tailless flapping wing micro air vehicle," in *50th AIAA Aerospace Science Meeting*, 2012, p. 588.

[15] 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.

[16] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1. IEEE, 2001, pp. 239–246.

[17] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, "Optimization based controller design and implementation for the Atlas robot in the DARPA robotics challenge finals," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 1028–1035.

[18] T. Marcucci, M. Gabiccini, and A. Artoni, "A two-stage trajectory optimization strategy for articulated bodies with unscheduled contact sequences," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 104–111, 2016.

[19] A. A. Paranjape, S.-J. Chung, and J. Kim, "Novel dihedral-based control of flapping-wing aircraft with application to perching," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1071–1084, 2013.

[20] C. J. Rose, P. Mahmoudieh, and R. S. Fearing, "Modeling and control of an ornithopter for diving," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 957–964.

[21] L. J. Roberts, H. A. Bruck, and S. Gupta, "Modeling of dive maneuvers for executing autonomous dives with a flapping wing air vehicle," *Journal of Mechanisms and Robotics*, vol. 9, no. 6, p. 061010, 2017.

[22] J. M. Dietl and E. Garcia, "Ornithopter optimal trajectory control," *Aerospace Science and Technology*, vol. 26, no. 1, pp. 192–199, 2013.

[23] 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.

[24] A. A. Hussein, A. E. Seleit, H. E. Taha, and M. R. Hajj, "Optimal transition of flapping wing micro-air vehicles from hovering to forward flight," *Aerospace Science and Technology*, vol. 90, pp. 246–263, 2019.

[25] J. Hoff, U. Syed, A. Ramezani, and S. Hutchinson, "Trajectory planning for a bat-like flapping wing robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6800–6805.

[26] J. Iriarte-Díaz and S. M. Swartz, "Kinematics of slow turn maneuvering in the fruit bat Cynopterus brachyotis," *Journal of Experimental Biology*, vol. 211, no. 21, pp. 3478–3489, 2008.

[27] D. K. Riskin, J. W. Bahlman, T. Y. Hubel, J. M. Ratcliffe, T. H. Kunz, and S. M. Swartz, "Bats go head-under-heels: The biomechanics of landing on a ceiling," *Journal of Experimental Biology*, vol. 212, no. 7, pp. 945–953, 2009.

[28] A. J. Bergou, S. M. Swartz, H. Vejdani, D. K. Riskin, L. Reimnitz, G. Taubin, and K. S. Breuer, "Falling with style: Bats perform complex aerial rotations by adjusting wing inertia," *PLoS Biology*, vol. 13, no. 11, p. e1002297, 2015.

[29] Z. J. Wang, J. M. Birch, and M. H. Dickinson, "Unsteady forces and flows in low Reynolds number hovering flight: Two-dimensional computations vs robotic wing experiments," *Journal of Experimental Biology*, vol. 207, no. 3, pp. 449–460, 2004.

[30] Z. J. Wang, "Dissecting insect flight," *Annual Review of Fluid Mechanics*, vol. 37, pp. 183–210, 2005.

[31] J. T. Betts, *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*. Siam, 2010, vol. 19.

[32] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1447–1454.

[33] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.