Control of uniflagellar soft robots at low Reynolds number using buckling instability

Mojtaba Forghani, Weicheng Huang, and M. Khalid Jawed

Abstract—In this paper, we analyze the inverse dynamics and control of a bacteria-inspired uniflagellar robot in a fluid medium at low Reynolds number. Inspired by the mechanism behind the locomotion of flagellated bacteria, we consider a robot comprised of a flagellum—a flexible helical filament—attached to a spherical head. The flagellum rotates about the head at a controlled angular velocity and generates a propulsive force that moves the robot forward. When the angular velocity exceeds a threshold value, the hydrodynamic force exerted by the fluid can cause the soft flagellum to buckle, characterized by a dramatic change in shape. In this computational study, a fluid-structure interaction model that combines Discrete Elastic Rods (DER) algorithm with Lighthill’s Slender Body Theory (LSBT) is employed to simulate the locomotion and deformation of the robot. We demonstrate that the robot can follow a prescribed path in three dimensional space by exploiting buckling of the flagellum. The control scheme involves only a single (binary) scalar input—the angular velocity of the flagellum. By triggering the buckling instability at the right moment, the robot can follow an arbitrary path in three dimensional space. We also show that the complexity of the dynamics of the helical filament can be captured using a deep neural network, from which we identify the input-output functional relationship between the control inputs and the trajectory of the robot. Furthermore, our study underscores the potential role of buckling in the locomotion of natural bacteria.

Index Terms—Motion control, bio-inspired robots, helical flagella, deep neural network.

I. INTRODUCTION

THANKS to recent advances in MEMS technology, design, fabrication, and control of millimeter and submillimeter sized robots have received numerous attention with particular emphasis on biomedical applications [11]–[15]. Inspired by the motion of microorganisms in fluid, a number of these works have focused on design of bio-inspired robots for the eventual purpose of placing them inside biological systems to perform non-invasive (or minimally invasive) tasks such as drug/cargo delivery [5]–[8], surgery [2], [9], targeted therapy [1], material removal [5], and imaging [10]. The majority of these works are centered around efficient fabrication and propulsion mechanism of micro-robots [3], [4], [11]–[15], while others focus on characterization of motion dynamics of microorganisms and its relevance to efficient control of micro-robots [6], [15], [16].

A variety of propulsion mechanisms, and subsequently their dynamics and control analyses, have been utilized by researchers. Notable among them are traveling wave propulsion [18], [19], chemically powered propulsion [20], magnetic field [3]–[6], [10]–[16], [21], [22], and electric field [23]. The vast majority of these works study the design of micro-robots in conjunction with their proposed propulsion mechanisms. In some of these studies, the primary focus is on the development of efficient power source for micro/nano motors [20] which is not the topic of the present paper.

There are a number of limitations associated with the current propulsion mechanisms. For instance, requirement of external magnetic or electric fields and consequently the issues associated with supply of sufficient field strength (such as the distance between the external source and the robot or the size of the external magnets) [3], rigidity of the flagellum attached to the robot body (bacteria have flexible flagella) which consequently requires multiple actuators to provide forward motion as well as steering [9], limited controllability of the robot [24], constraining the robot to operate in two-dimensional (2D) space [22], and the need for more than one flagellum (more than 90% of bacteria are uniflagellar [25]), not only add to the complexity of the propulsion mechanism, it leads to inefficient designs (for instance, multiple flagella are not necessarily a means for bacteria to increase their maneuverability). The complexities observed in many of the current designs can be traced back to our lack of understanding of the locomotion of microorganisms. This necessitates the search for a truly bio-inspired motion mechanism that mimics that of microbial locomotion.

Since reciprocal motion mechanisms (deformations with time-reversal symmetry) such as the ones used by fish cannot lead to any net propulsion at low Reynolds numbers (ratio of the inertial forces to viscous forces) as a consequence of scallop theorem [11], microorganisms are bound to employ alternative mechanisms. Among the different locomotion mechanisms used by microorganisms, propulsion of bacteria by rotation of flagella (flexible helical filaments) has attracted many researchers [25]–[29]. By exploiting buckling instability, uni-flagellated bacteria alternate their motion between turning and straight path to follow consecutive environmental cues [29]. In particular, by choosing the proper angular velocity of the flagellum attached to their body, bacteria can maneuver in three-dimensional (3D) space.

Uni-flagellated bacteria-inspired actuation mechanism has significant advantages over other mechanisms. For instance, we will show in this paper that a robot equipped with such mechanism can explore the 3D space using only one input (one
rotating actuator). Other actuation methods such as traveling-wave propellers \[18, 19\] require distributed actuation which has been proven to be very challenging in particular at the microscale \[5\]. Furthermore, depending on the cargo that the robot carries, long strips of drug, shaped into a helical shape, may be used as the flagellum itself, significantly minimizing the weight of the robot \[6\]. This propulsion method is comparatively more pliable than, for instance, magnetically driven actuators that are limited by rigidity of the material used as the flagellum \[10\] and require specific material properties (such as ferromagnetism) for efficient interaction with the magnetic field \[12, 13\]. Another advantage of such propellers is their higher maneuverability; due to crawling-like motion of elastic flagellum and flexibility of its dimension \[6\], these robots may potentially be suitable for efficiently moving through lumens \[5\]. Inspired by these potentials of uniflagellar robots, we consider the following problem: how can a uniflagellar robot follow a prescribed trajectory in three dimensional space simply by varying the angular velocity of the flagellum?

Due to the complexity of the mechanics of the helical rod and its hydrodynamic interaction with the viscous fluid that it is operating in, numerical methods for simulation of such systems must be able to couple the geometrically nonlinear deformation in the flexible flagellum and the hydrodynamic forces exerted by the fluid. Resistive Force Theory (RFT) is often used as a hydrodynamic model to analyze and predict slender body motion (such as flagella) in fluid \[6, 9, 11, 16\]. However, such methods, by neglecting non-local hydrodynamics (such as interaction among flows generated by distant parts on the flagellum), introduce significant approximations, leading to disagreement between theoretical predictions and experimental observations \[30\]. Moreover, the mechanical model for the flagellum should allow large deformation, in contrast to the often used assumption of rigid flagellum \[30\] or small deflection \[31\].

In our work, we model the flagellated soft robot as a helical flagellum attached to a rigid spherical head. As the robot (or the bacterium) moves in the fluid medium, it experiences a hydrodynamic drag force. Due to the microscopic size of bacteria, inertial forces are negligible compared with viscous forces and the fluid flow is at low Reynolds number regime (i.e. $Re \ll 1$). For numerical simulation of this fluid-structure interaction problem, we employ the Discrete Elastic Rods (DER) algorithm \[32-35\] – a computational tool for geometrically nonlinear deformation of rod-like structures – in conjunction with Lighthill’s Slender Body Theory (LSBT) \[35\] – a model for the hydrodynamic force on rods at low Reynolds number. The accuracy of the said fluid-structure interaction model has been partially validated against experiments in previous works \[30, 36, 37\]. Furthermore, in our numerical simulator, we have integrated the coupled interaction between the rigid head and the soft filament in low Reynolds environment by balancing the forces and torques in the whole structure \[38, 39\]. Details behind this model can be found in Ref. \[40\]. When the angular velocity of the flagellum is lower than a threshold for buckling, the robot travels along a straight path. However, if the angular velocity exceeds the threshold value, it leads to a buckling instability in the soft filamentary structure and the robot begins to follow a nonlinear trajectory.

Once the proper numerical setup has been established to solve the forward dynamics (the problem of generating robot trajectory from given angular velocity of its flagellum), we solve the inverse problem (the problem of generating angular velocities of flagellum from a given trajectory) by combining the input that characterizes the path that the robot should follow with a function approximation method (multi-layer neural network) that identifies the functional relationship between the properly projected information of the trajectory and the time-dependent angular velocities. Such inverse problem frameworks have been utilized successfully to control complex dynamical systems where identification of the system, due to blackbox-like behavior of the forward dynamics, is possible only from its input-output observations \[41\]. Exemplary applications include robot-assisted source identification \[42\], stochastic reachable sets parameterization \[43\], robot-human handover tasks \[44\], and human response time identification in semi-autonomous systems \[45\].

The remainder of the paper is organized as follows. In section \[II\] we formulate the problem that we are addressing here. In section \[III\] we briefly discuss the dynamic of the helical flagellum. In section \[IV\] we propose a learning process that can be employed to extract the inverse dynamics for the eventual purpose of controlling robot motion. In section \[V\] we provide the detail of the control algorithm. In section \[VI\] we discuss the result of the implementation of the algorithm, and finally, in section \[VII\] we provide the conclusion and outline our future direction.

### II. PROBLEM FORMULATION

Our goal is to provide an algorithm that takes state of the robot (a structure comprising of a head and a soft filament attached to it) and the set of inputs characterizing a desired trajectory (a path that the robot head is expected to follow), and provides proper control input in the form of a time-dependent angular velocity. The characterization of the trajectory consists of two points in space that the robot is expected to cover in its upcoming time steps. Any curve in the 3D space can be discretized into linear segments, allowing implementation of the algorithm as close as possible on arbitrary trajectories. Fig. \[I\] shows a sketch of the problem that we are addressing here.

The input to the control algorithm at time $t$ consists of the state of the robot and the characterization of the trajectory that it should follow:

$$
s(t) = (x_0(t - (0: k)\delta t), x_{1:2}(t), p_{1:2}, \omega(t))^T,
$$

where $i : j = \{i, i + 1, ..., j\}$, $\delta t$ is the time interval between two consecutive observations of the position of the system (the robot head), $x_i(t)$ is the position of the $i$-th node of the flagellum (with $x_0(t)$ showing position of the head of the robot; see section \[III\] for further detail of the node definition), $p_1$ and $p_2$ are the two successive points that the robot is expected to follow, $\omega(t)$ is the current angular velocity (the known input to the actuator), and the $T$ superscript stands for transposition. Here, the information of the position of the head
node of the k previous time steps, x₀(t − (1 : k)Δt), has been used in order to determine the direction of the motion (vector v in Fig. 1) and in order to check whether the trajectory is linear or not. Therefore, it is not essential that this information be available for every time step (Δt can be larger than the time step we use for our forward dynamics simulation, Δt, in section III). The information of the position of the two most adjacent nodes to the head has been used to determine the orientation of the robot (the unit vector n in Fig. 1). If the orientation is measurable through other sensory information, x₁:₂(t) in equation (1) can be replaced with n.

The control policy determined at time t that characterizes the future trajectory of the robot (for time t_f > t) amounts to determining the time dependent angular velocity Ω(t_f, s(t)). If the information of the desired trajectory is provided at time t = 0 through a set of points p₁:k (instead of the online information in the form of p₁:2), online conversion of these data into the form used in equation (1) is straightforward. For instance, if x₀(t) is between pᵢ and pᵢ₊1, then p₁:2 = pᵢ₊1:i+2.

Physical parameters: In this fluid-structure interaction problem, the structure is comprised of a Kirchhoff elastic rod [46] – our model for the flagellum – attached to a rigid head at one end. Based on prior experiments [39] on the dynamics of soft helical rods in viscous fluid, the physical parameters of the right-handed helical flagellum shown schematically in Fig. 2 are: axial length L = 13 cm, pitch λ = 32.6 mm, helix radius R = 6.04 mm, radius of circular cross-section r₀ = 1 mm, Young’s modulus E = 1 MPa, and Poisson’s ratio ν = 0.5 (incompressible material). The head is spherical with a radius b = 1 cm. The fluid medium has a viscosity of μ = 2.7 Pa·s. The density of the rod is ρ = 127 g/cm³ such that the Reynolds number Re remains in Stokes regime. While this density value does not affect the dynamics as long as we maintain low Reynolds number, a larger value of density typically allows us to take larger time step, Δt, in the simulation and reduces the computation time. For this study, we chose Δt = 1 ms. The number of nodes used in all of the simulations is N = 122. These parameters are similar to the ones used in our prior study [40].

III. FORWARD DYNAMICS

In this section, we provide an overview of the simulation procedure of the dynamics of the robot at low Reynolds number. Detailed description of the numerical method used for the forward dynamics is provided in Ref. [40]. As shown in Fig. 2 (Inset), the rod in DER [33], [34] is discretized into N nodes located at xⱼ with j = 0,...,N − 1. The segment of the rod between two nodes, xⱼ and xⱼ₊1, is the edge vector eⱼ = xⱼ₊1 − xⱼ. We use subscripts for node-based quantities and superscripts for edge-based quantities. Associated with each of N − 1 edges is the orthonormal material frame (m₁, m₂, tⱼ) that stays adapted to the centerline, i.e. tⱼ is the unit tangent vector along eⱼ. A second orthonormal adapted frame (d₁, d₂, tⱼ) is used as the reference frame that stays adapted to the centerline through parallel transport in time [34]. The relative angular orientation between the reference frame and the material frame at each edge is the twist angle θⱼ with j = 0,...,N − 2. The nodal coordinates, xⱼ, and the twist angles, θⱼ, constitute the 4N − 1 sized degrees of freedom (DOF) vector, q = [x₀, θ₀, ..., x_N₋₂, θᴺ₋², xᴺ₋₁], that completely describes the configuration of the rod. The DER algorithm marches forward in discrete time steps and updates q based on the balance of forces at each DOF. The equation of motion at i-th DOF, qᵢ, (i = 0,...,4N − 2) to move from time t to t + Δt is

\[
mᵢ \dot{qᵢ}(t + Δt) - qᵢ(t) - mᵢ \ddot{qᵢ}(t) \frac{Δt}{Δt} - fₖᵢ(q(t + Δt)) - fₑᵢ(q(t)) = 0,
\]

where mᵢ is the lumped mass at qᵢ, qᵢ(t) is the velocity at time t (i.e. time derivative of the DOF), fₖᵢ(q(t + Δt)) is the sum of elastic stretching, twisting, and bending forces that can be evaluated from q(t + Δt) (details can be found in Ref. [33]), and fₑᵢ(q(t)) is the external hydrodynamic force computed from q(t), described later in this section. When qᵢ(t) and qᵢ(t) are known, the system of 4N − 1 equations in (2) can be solved.
using Newton’s method to obtain $q_i(t + \Delta t)$. The velocity at $t + \Delta t$ is simply 
$$\dot{q}_i(t + \Delta t) = (q_i(t + \Delta t) - q_i(t)) / \Delta t.$$ 
In order to incorporate the rigid head in this framework, the first node, $x_0$, is assumed to be the center of the head, and, therefore, the velocity of the head is simply $U_h = x_0$. As shown in Fig. 2 the second node, $x_1$, is located on the axis of the helix such that the first edge $e^0 = x_1 - x_0$ is parallel to this axis. The flagellum rotates at a prescribed angular speed, $\omega$, about $e^0$ – the control parameter of this problem. When $\omega$ is positive, the flagellum rotates counter clockwise as viewed from above the head. The torque generated by the hydrodynamic force on the flagellum causes the head to rotate at an angular velocity $\Omega_h$.

As the flagellum and the head move in the fluid, they experience an external hydrodynamic force, represented by $f^\text{ext}_t$ in equation (2). The motion of the flagellum influences the force exerted by the fluid on the head; reciprocally, the motion of the head affects the force on the flagellum, resulting in a highly coupled problem, as summarized next.

**Force on the flagellum:** The velocity at each node on the rod is equal to the fluid velocity at that point (no-slip boundary condition). The velocity at the $j$-th node on the flagellum, $u_j \equiv [\dot{q}_{3j}, \dot{q}_{3j+1}, \dot{q}_{3j+2}]$, can be decomposed into two components: (i) a flow $(u_f)_j$ that is generated by the force exerted by the flagellum onto the fluid (equal and opposite of the boundary condition). The velocity at the $j$-th node relative to the center of the head, $(r_h)_j = ||(r_h)_j||$, and $\times$ is the cross product. Combining equations (3) and (4), the actual velocity at the $j$-th node is $u_j = (u_f)_j + (u_h)_j$.

**Force on the head:** The flow around the head is influenced by both the moving fluid current around it and the motion of the flagellum. The viscous fluid generates the drag force $-6\pi \mu b U_h$ and torque $-8\pi \mu b^2 \Omega_h$ on the moving head, while the flow caused by the force at each node on the flagellum (see equation (3)) results in the following force and torque on the head [38], [39]:

$$f_h = \sum_{j=1}^{N-1} \left( -3 \frac{b}{2(r_h)_j} + \frac{1}{2} \frac{b^3}{(r_h)_j^3} \right) f_j + \frac{f_j \cdot (r_h)_j}{(r_h)_j^2} \left( -3 \frac{b}{4(r_h)_j} + \frac{3}{4} \frac{b^3}{(r_h)_j^3} \right),$$

$$t_h = \sum_{j=1}^{N-1} - \frac{b^3}{(r_h)_j^3} (r_h)_j \times f_j.$$  

Here, we briefly review the scheme to compute the $4N-1$ sized external force vector (represented by $f^\text{ext}_t$ in equation (2)). At each time step of the simulation (between time $t$ to $t + \Delta t$), the DOF vector, $q$, the velocity, $\dot{q}$, the angular velocity of the first edge on flagellum, $\omega$, and the angular velocity of the head, $\Omega_h$, from the previous time step, $t$, are known. We use equation (4) to evaluate the flow caused by the moving head, $(u_h)_j$. Then the flow generated by the flagellum is $(u_f)_j = u_j - (u_h)_j$. Equation (3) can then be solved to compute the external force at any $j$-th node, $f_j \equiv [f^\text{ext}_j, f^\text{ext}_{j+1}, f^\text{ext}_{j+2}]$, where $j = 0, \ldots, N - 1$. Once $f_j$ is calculated, equation (2) can be used to obtain the new DOF vector and velocity at time $t + \Delta t$, treating the elastic forces implicitly and external forces explicitly. Lastly, the condition of torque balance on the whole robot is used to obtain the angular velocity of rotating head, $\Omega_h$.

In summary, the input of our numerical tool is the control parameter $\omega(t)$ that can vary as a function of time and the output is the configuration of the rod (e.g. the DOF vector, $q(t)$) that evolves with time. In this study, we are primarily concerned with the coordinates of the first node (i.e. $x_0$ – the first three elements of the DOF vector) that represents the trajectory of the head.

![Fig. 3: Schematic of (a) an unbuckled flagellum at $\omega(t) = \omega_L < \omega_h$ and (b) a buckled flagellum at $\omega(t) = \omega_H > \omega_h$.](image)
A key feature of the dynamics of the robot is the presence of a buckling instability [36, 47]. In Fig. 3(a), when the angular velocity \( \omega \) stays below the threshold angular velocity for buckling, \( \omega_b \), the flagellum retains its helical shape and moves along a straight line parallel to the axis of the helix. However, as shown schematically in Fig. 3(b), if \( \omega \) exceeds the threshold \( \omega_b \), the excessive hydrodynamic drag can cause the flagellum to buckle and a nonlinear trajectory ensues. For the model robot considered in our study, the threshold buckling velocity is \( \omega_b \approx 10 \text{ rpm} \).

IV. INVERSE DYNAMICS

The goal here is to find the functional relationship between a set of inputs characterizing a given path and the time dependent angular velocities. In particular, we seek to find the functions \( t_H = f_H(h, \alpha) \), \( t_L = f_L(h, \alpha) \), \( \beta = f_\beta(t_H, t_L) \), and \( l = f_l(t_H, t_L) \) such that the input

\[
\omega(t; t_0, t_H, t_L) = \begin{cases} 
\omega_L & \text{if } t - t_0 < 0, \\
\omega_H & \text{if } 0 \leq t - t_0 < t_H, \\
\omega_L & \text{if } t_H \leq t - t_0 < t_H + t_L,
\end{cases}
\]

the angular velocity parameterized with \( t_0, t_H, \) and \( t_L \), generates a trajectory \( x_0(t_0 \leq t < t_0 + t_H + t_L) \) that is parameterized with \( h, \alpha, \beta, \) and \( l \), in which

\[
\alpha = \frac{180}{\pi} \cos^{-1} \left( \frac{(p_2 - p_1) \cdot v}{|p_2 - p_1|} \right),
\]

\[
\beta = \frac{180}{\pi} \tan^{-1} \left( \frac{(p_2 - p_1) \cdot (v \times n)}{|p_2 - p_1| |v \times n|} \right),
\]

characterize the angles (in degrees units) between after-steering linear trajectory and the robot orientation, where \( v \) is the unit vector defining the direction of motion (see Fig. 1). In equation (6), \( \omega_L \) is an angular velocity below the flagellum buckling \( (\omega_L < \omega_b) \) for which the path of the robot can be well approximated by a line, while \( \omega_H \) is an angular velocity above the buckling \( \omega_b \) for which the robot follows a complex path in 3D which is responsible for steering of the robot. The time \( t_0 > 0 \) is a small time interval chosen such that the characterization of the trajectory starts when sufficient time has passed since the start of the motion (time 0) to ensure that the non-linearity due to starting from the static configuration has vanished. The detail of the reason for the choice of functional relationships considered here is provided in section 4.

The inverse dynamics identification begins with generation of trajectories with inputs in the form of equation (6). These trajectories can either be generated incrementally (by increasing \( t_H \) and \( t_L \) gradually) or randomly (by generating random \( t_H \) and \( t_L \)). Here, we have generated long trajectories (the total time \( t_{total} = 2500 \text{ s} \) with different \( t_H \) (gradually increasing from 0 to 55 s), and for each long trajectory, we picked different segments in the form \( t_0 \leq t < t_e \), where \( t_0 + t_H + k\delta t < t_e \leq t_{total} (t_e \text{ is the end point}) \), as one datapoint (one dataset in the form \( (t_H, t_L, h, \alpha, \beta, l) \)). This means that we simulate the forward dynamics based on a long simulation with \( \omega(t; t_0, t_H, t_{total} - t_H - t_0) \) input (see equation (6)) and then extract the information of \((t_H, t_L, h, \alpha, \beta, l)\) for different segments corresponding to \( \omega(t; t_0, t_H, t_L) \) input, where \( k\delta t < t_L \leq t_{total} - t_H - t_0 \), as one datapoint. Note that this process is necessary since the parameters \( \alpha \) and \( \beta \) change as the robot moves along a long trajectory. Since the trajectory does not become linear immediately after time \( t_0 + t_H \), due to the non-linearities associated with the above-buckling regime, and even after reaching linear regime, the two linear trajectories before and after steering are not coplanar, the variation in parameters \( \alpha \) and \( \beta \) for a given long trajectory, albeit small, is non-zero for the interval \( t_0 + t_H + k\delta t < t < t_{total} \). Moreover, even in the linear regime, the trajectory in reality does not follow a linear path; in fact, the trajectory is a helical path with a small radius that has a non-varying axis of rotation (see Fig. 4).

After generation of sufficient datapoint (different segments of long trajectories), we proceed to convert the information contained in \( x_0(t_0 \leq t < t_0 + t_H + t_L) \) to the form \( (t_H, t_L, h, \alpha, \beta, l) \). Since the linear trajectories before and after turning are not coplanar, this is done by finding a point \( p_1 \) located on the unique plane defined by \( p_2 = x_0(t_0 + t_H + t_L) \) and \( x_0(t_0) \), that contains \( v \), the unit vector defining the direction of motion (see Fig. 4). This ensures that the two lines characterizing the steering are coplanar and as close as possible to the true trajectory. The details of finding the optimal \( p_1 \) are provided in Appendix A.

Once the dataset \( (t_H, t_L, h, \alpha, \beta, l) \) is created, we find functions \( t_H = f_H(h, \alpha) \), \( t_L = f_L(h, \alpha) \), \( \beta = f_\beta(t_H, t_L) \), and \( l = f_l(t_H, t_L) \). Due to the complexity of the behavior.
of the robot in the above-buckling regime, fitting a simple function to \((t_H, t_L, h, \alpha, \beta, l)\) datapoint is inadequate. Fig. 5 shows the trajectory of the robot when \(\omega_H\) is applied. Although this complex motion occurs in a length scale significantly smaller than the robot length, since \(\beta\) and \(\alpha\) critically depend on this trajectory, it is very essential to parameterize it accurately. Here, we employed a three-layer neural network with \(N_1 = 20, N_2 = 10\) and \(N_3 = 5\) neurons for each of these functions and trained the data using Bayesian regularization (see Fig. 6). The number of layers and neurons have been determined experimentally. We did not see significant improvement by using more complex architecture.

V. CONTROL

The goal here is to find the time-dependent angular velocity function \(\Omega(t_f, s(t))\) for time \(t_f > t\), given the input \(s(t)\) at time \(t\) (see equation (1)). The proposed algorithm in this section assumes the existence of only two control inputs: one below buckling, \(\omega_L\), providing a linear trajectory, and one above buckling, \(\omega_H\), providing rotation in 3D space. Extension of the single below-buckling angular velocity to multiple (possibly continuously varying) angular velocities is straightforward due to simplicity of the linear trajectory which amounts to only characterization of the linear velocity of the robot along the linear trajectory. The functions \(f_H(h, \alpha), f_L(h, \alpha), f_b(t_H, t_L),\) and \(f_s(t_H, t_L)\) used in this section are assumed to be available to the algorithm based on the learning process described in section [IV].

If the angular velocity at the current time, \(t\), is below buckling (\(\omega(t) < \omega_h\)), the current trajectory of the robot can be calculated by fitting a line to the head position of the current and previous motion steps \(x_0(t - (0:k)\delta t)\), which amounts to solving the following minimization problem

\[
\mathcal{L}(a_{1:4}) = \min_{a_{1:4}} \sum_{j=0}^{k} \left[ (x_{0_y}(t - j\delta t) - a_1 x_{0z}(t - j\delta t) - a_2)^2 + (x_{0_z}(t - j\delta t) - a_3 x_{0x}(t - j\delta t) - a_4)^2 \right],
\]

where \(a_{1:4}\) are the parameters characterizing the equation of the trajectory; in particular, \(y = a_1 x + a_2\) and \(z = a_3 x + a_4\) is the trajectory. Also, \(x_0\) for \(i \in \{x, y, z\}\) is the \(i\)-th component of \(x_0\). Equation (8) has the following solutions

\[
a_1 = \frac{k + 1}{k + 1} \sum_{j=0}^{k} x_{0z}(t - j\delta t) x_{0y}(t - j\delta t)
\]

\[
- \frac{k + 1}{k + 1} \sum_{j=0}^{k} x_{0x}(t - j\delta t) \sum_{j=0}^{k} x_{0y}(t - j\delta t)
\]

\[
(9)
\]

\[
a_2 = \frac{1}{k + 1} \sum_{j=0}^{k} x_{0y}(t - j\delta t) - a_1 \sum_{j=0}^{k} x_{0x}(t - j\delta t)
\]

\[
a_3 = \frac{1}{k + 1} \sum_{j=0}^{k} x_{0z}(t - j\delta t) x_{0y}(t - j\delta t)
\]

\[
- \frac{1}{k + 1} \sum_{j=0}^{k} x_{0x}(t - j\delta t) \sum_{j=0}^{k} x_{0z}(t - j\delta t)
\]

\[
(9)
\]

\[
a_4 = \frac{1}{k + 1} \sum_{j=0}^{k} x_{0z}(t - j\delta t) - a_3 \sum_{j=0}^{k} x_{0x}(t - j\delta t)
\]

Equation (9) can be used to determine the direction of motion and the degree of linearity, that is, if \(\mathcal{L}(a_{1:4}) > \delta_t\) for a preset small \(\delta_t\), then the linear motion assumption is not correct, implying that either the robot is turning (if also \(\omega(t) > \omega_h\)) or the robot has recently turned but not sufficient time has passed for it to adjust to new below-buckling dynamics (if \(\omega(t) < \omega_h\)). This information is useful in determining \(\Omega(t_f, s(t))\), since the functions trained in section [V] are valid only if the current trajectory is linear. The unit vector determining the direction of motion can be calculated using the result provided in (9) via

\[
v = \text{sgn}\left(\{x_0(t) - x_1(t)\} \cdot \{1, a_1, a_3\}\right) \frac{1}{\|\{1, a_1, a_3\}\|},
\]

where \(\text{sgn}\) is the sign function.

If the robot follows the successive \(p_{1:2}\) points exactly, the vector \(v\) and points \(x_0(t)\) and \(p_{1:2}\) should be coplanar. However, in practice, due to multiple sources of error, noise,
or inaccuracy in function training, these points may not be reached exactly. In order to guarantee that the destined points are not compromised due to errors in following the previous trajectories, we define an alternative trajectory at each time that given \( v, x_0(t), \) and \( p_2 \) (which define a unique plane together), finds an alternative point \( \hat{p}_1 \) that is as close as possible to \( p_1 \); this amounts to finding the projection of point \( p_1 \) on the plane defined by \( v, x_0(t), \) and \( p_2 \) which itself is parameterized via

\[
[v \times (p_2 - x_0(t))] \cdot (x - x_0(t)) = 0,
\]

where \( x \) is a point in 3D space. The components of the projected point \( \hat{p}_1 \) are

\[
\hat{p}_{1x} = p_{1x} + \frac{v \times (p_2 - x_0(t))}{\|v \times (p_2 - x_0(t))\|^2} \cdot (x_0(t) - p_1),
\]

\[
\hat{p}_{1y} = \frac{v \times (p_2 - x_0(t))}{\|v \times (p_2 - x_0(t))\|^2} \cdot (p_{1x} - p_1x) + p_{1y},
\]

\[
\hat{p}_{1z} = \frac{v \times (p_2 - x_0(t))}{\|v \times (p_2 - x_0(t))\|^2} \cdot (p_{1x} - p_1x) + p_{1z}.
\]

In order to transform the information embedded in the \( x_0(t) \rightarrow \hat{p}_1 \rightarrow p_2 \) trajectory into the spherical body-fixed coordinates, we define the following three principal directions:

\[
v, n = \frac{v \times (x_1(t) - x_2(t))}{\|v \times (x_1(t) - x_2(t))\|}, \quad w = \frac{v \times n}{\|v \times n\|}.
\]

The projection of \( p_2 - \hat{p}_1 \), vector on these principal directions provides us with the characterization of the trajectory in the body-fixed spherical coordinates. In particular, the four parameters defined below uniquely characterize the immediate desired trajectory:

\[
h_d = \|p_2 - \hat{p}_1\|, \quad l_d = \text{sgn}((\hat{p}_1 - x_0(t)) \cdot v) \|\hat{p}_1 - x_0(t)\|,
\]

\[
\alpha_d = \frac{180}{\pi} \cos^{-1} \left( \frac{(p_2 - \hat{p}_1) \cdot v}{\|p_2 - \hat{p}_1\|} \right),
\]

\[
\beta_d = \frac{180}{\pi} \tan^{-1} \left( \frac{(p_2 - \hat{p}_1) \cdot (v \times n)}{\|p_2 - \hat{p}_1\| \|v \times n\|} \right),
\]

where the subscript \( d \) stands for “desired” and the angles \( \alpha_d \) and \( \beta_d \) are represented in degrees units.

Once these parameters are determined, we can find the proper control inputs which amounts to specification of three parameters, \( t_{\text{app}}, t_H, \) and \( t_L \) in the function

\[
\omega(t_f) = \begin{cases} 
\omega_L & \text{if } 0 \leq t_f < t_{\text{app}}, \\
\omega_H & \text{if } t_{\text{app}} \leq t_f < t_{\text{app}} + t_H, \\
\omega_L & \text{if } t_{\text{app}} + t_H \leq t_f < t_{\text{app}} + t_H + t_L,
\end{cases}
\]

where “app” corresponds to the time that the steering angular velocity must be applied. For a given \( h_d \) and \( \alpha_d \), the time intervals \( t_H \) and \( t_L \) in equation \ref{eq:time_intervals} can be uniquely determined via

\[
t_H = f_H(h_d, \alpha_d), \quad t_L = f_L(h_d, \alpha_d),
\]

from which we can calculate their corresponding \( \beta \) and \( l \) via

\[
\beta = f_\beta(t_H, t_L), \quad l = f_l(t_H, t_L),
\]

however, \( \beta \), the angle corresponding to \( t_H \) and \( t_L \), is not necessarily equal to \( \beta_d \) and is not controllable either independent of other parameters once \( t_H \) and \( t_L \) are determined; the same statement is true for \( l \) and \( l_d \). The remaining control parameter in \ref{eq:b_t}, \( t_{\text{app}} \), must be chosen such that \( l \) and \( \beta \) are as close as possible to \( l_d \) and \( \beta_d \), respectively. This problem can be formulated as an optimization problem where some \( l \) and \( \beta \) that provide a trajectory as close as possible to the desired one are calculated. However, here, by exploiting an important property of the flagellum actuator, its periodic orientation, we only compromise on the value of \( l_d \). Note that the orientation of the robot during its linear motion has a period of \( 60/\omega_L \) seconds (\( \omega_L \) is in rpm units), that is

\[
n(t + 60/\omega_L) = n(t).
\]

On the other hand, the robot travels a distance of \( 60v_{\omega_L}/\omega_L \) during one period, where \( v_{\omega_L} \) is the linear velocity when \( \omega_L \) is applied. This is a very small distance compared with the size of the robot; for instance, for the parameters considered in this work, it is about 4 mm while the length of the flagellum is 20 cm. Taking into account these considerations, we choose the following \( t_{\text{app}} \)

\[
t_{\text{app}} = \frac{\beta_d - \beta}{6\omega_L} + \frac{60}{\omega_L} \arg \min_{\kappa} \left\{ \frac{\beta_d - \beta + 360\kappa}{6\omega_L} - v_{\omega_L} + l - l_d \right\}.
\]

The first term in \ref{eq:time_app} guarantees that the robot will be able to follow \( \beta_d \) accurately while the second term (the \( \arg \min \) expression) guarantees that \( l_d \) is reproduced as close as possible.

The solution to \ref{eq:time_app} is

\[
t_{\text{app}} = \frac{\beta_d - \beta}{6\omega_L} + \frac{60}{\omega_L} \text{round} \left\{ \frac{\omega_L(l_d - l) - \beta_d - \beta}{360} \right\},
\]

where “round” implies rounding to the nearest integer. This approximation leads to the maximum error of \( 30\omega_{\omega_L}/\omega_L \) in following \( l_d \) which is very negligible; for instance, it is 2 mm (compare with the length of the flagellum which is 20 cm) at \( \omega_L = 3 \text{ rpm} \) (the value chosen in our work).

The control algorithm \( \Omega(t_f, s(t)) \) based on the results of this section is provided in Algorithm \ref{alg:control_algorithm}. The used time in this algorithm, \( t \), is the universal clock, therefore, after each time step, we have \( t \leftarrow t + \delta t \). The algorithm first initializes the angular velocity of the first \( k \) time steps to \( \omega_L \) to allow sufficient time from the initial non-linearity associated with starting from the static state. Then it checks whether all the elements of \( s(t) \) are non-empty, and if this is true, it proceeds to check whether the assumption of linear trajectory is valid, since otherwise, the results provided in this section and section \ref{sec:linear_trajectory} do not hold anymore. If the algorithm passes all these tests, it calculates the future angular velocities and stores them in \( \omega(t_f) \). Note that here we are assuming that the algorithm does not erase any of its information over the sequence of time steps. For instance, if there is incoming information of \( P_{1/2} \) while the robot is turning (\( L(a_{1/4}) > \delta t \)), it stores \( P_{1/2} \) and will use it as soon as other conditions hold as well. The same is true
for the parameter \( j \) in the algorithm. Finally, if the condition \( \omega(t + \delta t) = \emptyset \) is satisfied at time \( t \), it sets \( \omega(t + \delta t) = 0 \) to stop the robot (\( \emptyset \) is the empty set).

**Algorithm 1** Online control \( \Omega(t_f, s(t)) \)

**Require:** \( s(t) \): input (see equation [1])

\[
\text{if } t = 0 \text{ then }
\quad \text{for } j = 1 \text{ to } j = k \text{ do }
\quad \quad \omega(t + j\delta t) = \omega_L
\quad \text{end for }
\text{end if }
\]

if \( t > k\delta t \), \( x_1(t) \neq \emptyset \), \( x_2(t) \neq \emptyset \), \( \mathbf{p}_1 \neq \emptyset \), \( \mathbf{p}_2 \neq \emptyset \), and \( \forall i \in \{0, \ldots, k\} : x_0(t - i\delta t) \neq \emptyset \) then

Calculate \( a_{1:4} \) from (9); calculate \( \mathcal{L}(a_{1:4}) \) from (8);

if \( \mathcal{L}(a_{1:4}) \leq \delta_l \) then

Calculate \( v \) from (10); \( h_1 \) from (12); \( h_d \), \( l_d \), \( \alpha_d \), and \( \beta_d \) from (14); \( t_H \) and \( t_L \) from (16); \( \beta \) and \( l \) from (17); \( t_{app} \) from (20);

\( j \leftarrow 1 \)

while \( j\delta t < t_{app} + t_H + t_L \) do

if \( j\delta t \geq t_{app} \) and \( j\delta t < t_{app} + t_H \) then

\( \omega(t + j\delta t) \leftarrow \omega_H \)

else

\( \omega(t + j\delta t) \leftarrow \omega_L \)

end if

\( j \leftarrow j + 1 \)

end while

end if

\( \omega(t_f) \leftarrow \omega(t + \lfloor 1 : j \rfloor \delta t) \)

\( j \leftarrow j - 1 \)

if \( \omega(t_f) = \emptyset \) then

\( \omega(t_f) = 0 \)

end if

return \( \omega(t_f) \)

**VI. RESULTS**

We have used Algorithm 1 to follow a helical path. The points characterizing the desired trajectory (the sequence \( \mathbf{p}_{1:2} \)), the robot trajectory, and a few screenshots of the robot itself are shown in Fig. 7. We observe that overall the robot has been able to follow the desired points with reasonable accuracy. The robot head (blue spheres in the figure) is the point that is expected to follow the desired points (red asterisks) sequentially.

One of the advantages of the algorithm used here is that it has binary input, \( \omega_L \) and \( \omega_H \), which simplifies both control policy and design of the actuator. Fig. 9a shows the control input used for generation of the trajectory of Fig. 7. We observe that the periods at which the above-buckling angular velocity has been input are significantly shorter than the below-buckling periods. This is also partly due to small steering angles, \( \alpha_r \), used for the trajectory of Fig. 7. Larger values of \( \alpha_r \) require longer periods of \( \omega_H \). These periods also depend on the material and structural properties of the robot and the fluid in which it is operating. We have also plotted the tracking error in Fig. 9b. The tracking error in this figure is defined as the minimum distance (Euclidean distance) between robot head and the prescribed helical path. We observe that overall the error is small. We also notice that for the most part, the largest errors correspond to steering incidents – the same trend observed in Fig. 4. This error can decrease if a set of material properties is chosen such that the deviation of robot trajectory from coplanar before- and after- steering lines is minimal. The overall error may also be reduced by generating more data points during the learning process, choosing a smaller linearity criteria, \( \delta_l \), or increasing the time window for estimation of direction of motion (increasing \( k \)).

**Fig. 7:** The robot trajectory (solid black curve) and the points that it was expected to follow (red asterisks).

**Fig. 8:** (a)- Output (velocity of the head and its components) and (b)- input (angular velocity) during a steering incident.
Similar to the time duration required for applying the above-buckling angular velocity for providing a given steering angle, the time taken for the flagellum to reach its unbent straight-line configuration also varies depending on the material property of the flagellum. Fig. 8a shows the robot velocity magnitude and its three components during a steering incident and Fig. 8b shows its corresponding input angular velocity. The complexity of the robot path when the above-buckling angular velocity is applied is well captured in Fig. 8a. Another important observation is the long post-buckling effect in robot trajectory and velocity which lasts about three times the applied $\omega_H$ duration. This period can be shortened by performing an optimization over different $\omega_L$ and $\omega_H$ values, different material properties, and different structural properties, aimed at minimizing the total buckling effect duration. A similar inverse problem framework can be utilized to identify the relationship between these inputs and the robot trajectory.

VII. Conclusion

Inspired by motion of bacteria, we have proposed a method for locomotion of robots in low Reynolds number using one single input. Restricting the angular velocity to obtain a binary input ($\omega_L$ and $\omega_H$) provides simple design of actuators, addressing one of the main challenges that millimeter/sub-millimeter robots face. By combining an accurate numerical model of the dynamics of the helical filament with neural network, we have been able to capture the complexity associated with the inverse dynamics, bypassing the issues associated with approximate analytical/semi-analytical approaches to characterization of the robot motion. Since it has been verified previously that the simulation tools employed here resemble the dynamics of the helical filament accurately, in the present work, we have primarily focused on the numerical and algorithmic aspects of the motion control. Our future emphasis is on the implementation of the algorithm provided here on small scale soft robots.

In this work, we have focused on uniflagellar robots due to the simplicity of their input (binary input) that will consequently lead to simpler design of actuators. Such designs are suitable when the primary objective of the robot is reaching a target, such as imaging or sensing. However, higher maneuverability of robot requires more complicated inputs. Here, we did not assess the instability in motion due, for instance, to carrying a cargo, such as drug delivery tasks. Such designs require both more inputs, such as multiple flagella instead of one, and more advanced control policy such as adaptive and/or sliding mode control strategies [49]. In future, we will study both algorithmic and experimental aspects of such designs.

In the present work, we proposed an algorithm assuming perfect knowledge of the system state which consequently led to designs based on only the next two points of the desired trajectory (assuming that the state of the system is readily available in future time steps). If the system state is imperfect, for instance due to communication and sensory issues, the algorithm may need to output angular velocity function for more than one turning event (trajectory characterization with more than two points). Model predictive control strategies (MPC) based on online sequential optimization of the given sequential trajectories can address such issues [50]; however, here, due to computational cost of the forward dynamics simulation, we did not study them. In future, we will explore further aspects of the numerical method used to solve the forward dynamics in order to reduce its cost and consequently use MPC for sequential motion control.

ACKNOWLEDGMENT

We acknowledge support from the Henry Samueli School of Engineering and Applied Science, University of California, Los Angeles.

APPENDIX A

PARAMETERIZATION OF THE GENERATED TRAJECTORIES

Here, we provide the details of converting the information embedded in the generated trajectories, $x_0(t_0 \leq t < t_0 + t_H + t_L)$ into the form of the data we use in the control algorithm, $(t_H, t_L, h, \alpha, \beta, l)$. Since the robot’s before-turning trajectory, initial direction of motion, and initial orientation do
where trajectories before and after the rotation and the projection of the image on the lines. The next step is to parameterize the trajectory of the robot to satisfying the following conditions

\[ a_1 x + a_2 = b_1 x + b_2 \]
\[ a_3 x + a_4 = b_3 x + b_4, \]
\[ b_1 x_{0x}(t_e) + b_2 = x_{0y}(t_e) \quad \text{and} \quad b_3 x_{0x}(t_e) + b_4 = x_{0z}(t_e), \]

where the notation \( t_e = t_0 + t_{th} + t_L \) (the end point) has been used for compactness. The first condition in (21) guarantees that the trajectories before and after turning are coplanar, and the second conditions guarantee that the end position is on the trajectory, leading to two additional equations; in fact, \( x_{0l}(t_e) \) for different values of \( t_e \) corresponds to different \( p_2 \) points in Fig. 3. The fourth equation required to determine \( b_{1:4} \) is obtained by fitting the best line to the trajectory of the robot, that is, solving the following optimization problem

\[
\min_{b_{1:4}} \sum_{j=1}^{m} \left[ (x_{0y}(t_e) - b_1 x_{0x}(t_e) - b_2)^2 + (x_{0z}(t_e) - b_3 x_{0x}(t_e) - b_4)^2 \right].
\]

where \( m = \text{round} \left\{ \frac{t_e - h \delta t}{\delta t} \right\} \). Solving (21) and (22) leads to following solutions

\[
b_1 = \sum_{j=1}^{m} \left[ x_{0x}(t_e) - j \delta t - x_{0y}(t_e) \right] \left[ x_{0y}(t_e) - j \delta t - x_{0y}(t_e) \right] \frac{1 + \chi^{-2}}{1 + \chi^2} + \left[ x_{0z}(t_e) - j \delta t - x_{0z}(t_e) \right] \frac{1 + \chi^{-2}}{1 + \chi^2},
\]

\[
b_2 = x_{0y}(t_e) - b_1 x_{0x}(t_e), \quad b_3 = (a_1 - b_1) x_{0x}(t_e) - a_4, \quad b_4 = x_{0z}(t_e) - b_3 x_{0x}(t_e) + a_2 + a_3 x_{0x}(t_e),
\]

where

\[
\chi = \frac{a_1 x_{0x}(t_e) + a_2 - x_{0y}(t_e) + a_3 x_{0x}(t_e) + a_4 - x_{0z}(t_e)}{a_3 x_{0x}(t_e) + a_4 - x_{0z}(t_e)}.
\]

Next, we find the location of the intersection of the linear trajectories before and after the rotation and the projection of \( x_{0}(t_0) \) on the \( y = a_1 x + a_2 \) and \( z = a_3 x + a_4 \) lines, as it was explained previously, due to helical path of the robot in the linear regime (Fig. 4). \( x_{0}(t_0) \) is slightly distorted from its image on the lines.

\[ b_{1:4} \]

\[ \text{from which the position of the intersection can also be calculated in the following form} \]

\[
p_1 = \left[ \frac{b_2 - a_2}{a_1 - b_1}, \frac{a_1 b_2 - b_1 a_2}{a_1 - b_1}, \frac{a_3 b_4 - b_3 a_4}{a_3 - b_3} \right].
\]

The parameters \( h, \alpha, \beta, l \) used in (16) and (17) can be calculated based on equations similar to those used in (14). Also, we can replace \( x_{0l}(t_e) \) with \( p_2 \), therefore,

\[
h = \| p_2 - p_1 \|, \quad l = \text{sgn} \left( (p_1 - x_{0l}) \cdot v \right) \| p_1 - x_{0l} \|, \quad \alpha = \frac{180}{\pi} \cos^{-1} \left( \frac{(p_2 - p_1) \cdot v}{\| p_2 - p_1 \|} \right), \quad \beta = \frac{180}{\pi} \tan^{-1} \left( \frac{(p_2 - p_1) \cdot (v \times n)}{\| p_2 - p_1 \| \| v \times n \|} \right).
\]

The set \( (t_{th}, t_L, h, \alpha, \beta, l) \) is one unique datapoint used in the neural network training algorithm.

An alternative to training two separate functions, \( t_{th} = f_{HL}(h, \alpha) \) and \( t_L = f_{L}(h, \alpha) \), is to train one function \( (t_{th}, t_L) = f_{HL}(h, \alpha) \). An important consideration in identification of function \( (t_{th}, t_L) = f_{HL}(h, \alpha) \) is to ensure that the trained network is not biased toward either of the outputs. A reasonable characterization of the error between the true functional relationship of the datapoint inputted to the network and the trained function is the error observed in the value of the final target point \( p_2 \) (see Fig. 4), since the position of \( p_2 \) is much more sensitive to \( h \) and \( \alpha \) than \( p_1 \), therefore, \( p_1 \) can be assumed to be unaffected by error in \( h \) and \( \alpha \). In particular, if we constrain the trained function to always output a set \( (t_{th}, t_L) \) for which point \( p_2 \) is inside a ball of radius \( \epsilon \), it must satisfy \( |h \Delta | < \epsilon \) and \( |\Delta h| < \epsilon \), assuming \( \Delta \alpha \) to be small. An estimate of the relation between \( \Delta h \) and \( \Delta t_L \) is \( \Delta h \sim v_{ul} \Delta t_L \). A linear fitting can also be used to obtain an estimate to \( \Delta \alpha \sim c \Delta t_H \). Therefore, the error constraints can be written in the form of \( |h c \Delta t_H| \lesssim \epsilon \) and \( |v_{ul} \Delta t_L| \lesssim \epsilon \), implying that \( h c \Delta t_H \) and \( v_{ul} \Delta t_L \) must be trained instead of \( t_{th} \) and \( t_L \). We have tested this training process and have observed it gives an error similar to that of separate training of \( f_{HL} \) and \( f_{L} \), while parameterization based on \( t_{th} \) and \( t_L \) is highly biased toward \( t_L \) (due to \( h c/v_{ul} \gg 1 \)).

REFERENCES

1. S. Ornes, Medical microrobots have potential in surgery, therapy, imaging, and diagnostics, Proceedings of the National Academy of Sciences 114, 12356-12358 (2017).

2. The parameters \( h, \alpha \) used here are similar to \( h_{2}, \alpha_{2} \) used in equation (16), but in order to distinguish between the desired parameters input to the control algorithm and the ones obtained based on the data training, we have used subscript \( d \).
