Data-driven optimal control of undulatory swimming

Karl Maroun, Philippe Traoré, Michel Bergmann

Data-driven optimal control of undulatory swimming

Karl Maroun,[1] Philippe Traoré,[1] and Michel Bergmann[2]
[1)]*Institut PPRIME, Département Fluide-Thermique-Combustion, Boulevard Pierre et Marie Curie, BP 30179, 86962, Futuroscope-Chasseneuil, France*
[2)]*INRIA Bordeaux Sud-Ouest, MEMPHIS team, 200 avenue de la Vielle Tour, 33405 Talence, France.*

(Dated: 25 September 2024)

Achieving precise control over self-propelled undulatory swimmers requires a deep understanding of their intricate dynamics. This paper presents a method for addressing optimal control problems in this context by leveraging surrogate models. We develop a Navier-Stokes solver using a volume penalization method to simulate the fluid-structure interaction inherent in swimming dynamics. An offline phase generates training data through open-loop simulations across a defined range of control inputs, enabling the training of a surrogate model. This model significantly reduces computational costs, particularly in optimization and control contexts. Utilizing these surrogate models, we compute control strategies to address two key challenges: precise velocity tracking and optimizing swimmer efficiency. Firstly, we employ model predictive control (MPC) to enable velocity tracking against a reference signal, allowing swift adjustments of the swimmer's frequency and amplitude. Secondly, we tackle the minimization of the swimmer's cost of transport, resulting in a solution akin to a burst-and-coast strategy. Despite achieving energy performance comparable to continuous swimming cases, mismatches between the surrogate model and the high fidelity simulation significantly impact the quality of the obtained solution. This work sheds light on the potential of surrogate models in optimizing self-propelled swimming behavior and underscores the importance of addressing model mismatches for more accurate control strategies in the future.

I. INTRODUCTION

Aquatic animals employ different control mechanisms to enhance hydrodynamic performance. These flow control methods are either passive, involving anatomical and morphological changes driven by evolution, or active, where muscles are used to alter body shape and manipulate the flow[1]. Among animals relying on undulations for thrust generation, examples of active flow control include adjusting swimming frequency, amplitude, and body curvature. Active flow control has been a long standing topic in the fluid mechanics literature[2,3]. The use of actuators coupled with robust control laws has the potential to improve the hydrodynamic performance of many engineering systems of interest. Drag reduction, lift augmentation, improved energy efficiency are just some of many benefits that could be achieved. In computational contexts, active flow control is broadly categorized into two approaches: model-based and model-free control. In the first one, a mathematical model of the dynamics is known a priori and used to compute control laws. Model-free approaches on the other hand, do not rely on a model. Control laws are computed using data generated from the real system. Both methods have their own advantages and drawbacks. As an example, mismatches between the real system and the model is one important limitation of model-based approaches. Model-free approaches circumvent this problem by directly optimizing the real system. Nonetheless, without the use of a model, the search for a good control law becomes computationally demanding and exhibits poor sample efficiency[4–6]. Furthermore, the incorporation of constraints on control inputs and states is easier in model-based approaches because these constraints can be formulated mathematically and added to the optimization problem. Recent advances in the field of machine learning (ML) have brought many tools that can complement and improve classical approaches in active flow control. In model-free methods, genetic programming[7], deep reinforcement learning[8–10] and cluster based methods[11,12] have all been used to control several interesting flows. In model-based approaches, the most obvious use of ML is to learn the model using a supervised learning algorithm. The learned model is then used in a model-based method such as model predictive control. The dimension of the state space in active flow control problems is often very high . Reduced order models, obtained with methods such as Proper Orthogonal Decomposition (POD)[13,14], are commonly employed for model-based control to alleviate computational concerns.

For most aquatic animals, deriving mathematical models from first principles is far from an easy task. Many aquatic animals propel themselves through water by generating periodic undulatory motions. The hydrodynamic forces that are applied on the swimmer result from complex interactions with the surrounding fluid. In the 1960s, Lighthill pioneered the development of analytical expressions for these forces through his renowned elongated-body theory[15]. In his theory, Lighthill makes the assumption that the fluid is potential and derives formulas for both the thrust and the energy cost. Dynamical models based on Lighthill's theory have been used to predict and control the motion of bio-inspired aquatic robots[16–18].
Due do the above mentioned difficulty, many studies have focused on combining high fidelity simulations with model free methods to solve control tasks for undulatory swimming[8,19,20]. In this study, we explore an alternative approach by combining high-fidelity numerical simulations with modern machine learning techniques for model-based control. Our strategy involves utilizing data generated from computational fluid dynamics simulations to perform nonlinear system identification. The obtained models are then utilized for control. One notable advantage of this approach is its reduced reliance on simplifying assumptions when compared to the

conventional analytical approach typified by Lighthill's theory. In our approach, we solve the incompressible Navier-Stokes equations numerically to accurately calculate the hydrodynamic forces acting on the swimmer. The computed forces account for viscous effects, which are not considered in the model relying on Lighthill's theory because of the potential flow assumption.

Many nonlinear system identification methods exist in the literature. They can be classified into two categories: parametric and nonparametric methods. For both of these approaches the goal is the same: learn the system dynamics during an offline phase to build a model that adequately represents the system's behavior. The model is then used for control. Recent years have seen a significant upsurge in the exploration of machine learning techniques for system idenfication and control. For instance, in the work by Hewing, Kabzan, and Zeilinger[21], gaussian process regression is used to learn model uncertainties with respect to a nominal model. The resulting stochastic dynamics are then used for safe model predictive control. Similarly in a study by Wu *et al.*[22], a recurrent neural network is trained using open loop simulations to obtain the system dynamics. In Ref. 23, a parametric method known as the sparse identification of nonlinear dynamics (SINDy)[24] is used to learn a model to control the plasma in a tokamak. What distinguishes SINDy is its ability to provide interpretable models, making it an attractive option, especially when a nominal model is unavailable. Additionally, SINDy needs significantly less data than methods like neural networks[25].

In this work, for the reason mentioned above, we decided to use SINDy to model the swimming dynamics. The data required to train the model is obtained by performing a certain number of high fidelity numerical simulations. The remainder of the paper is organized as follows: In the following section, the numerical modelling procedure required for the simulation is introduced. Then in section III, the SINDy framework used for system identification is presented. The numerical optimal control methods that are used are detailed in section IV. Section V is devoted to the presentation of results on both velocity tracking and cost of transport optimization. Finally, a brief conclusion is drawn.

II. NUMERICAL MODELLING

A. Swimming kinematics

The kinematics of most undulatory swimming aquatic animals can be approximated by a backward traveling wave[26] of the following form:

$$y(x,t) = a(x)\,sin\left(kx - \omega_f t\right). \tag{1}$$

The wavenumber k is related to the wavelength λ by the following formula: $k = \frac{2\pi}{\lambda}$. The angular frequency, ω_f, is given by: $\omega_f = 2\pi f$, where f is the frequency of the wave. The curve envelope, $a(x)$ is approximated by a quadratic function:

$$a(x) = c_0 + c_1 x + c_2 x^2. \tag{2}$$

The constants (c_0, c_1 and c_2) are defined according to the undulatory swimming mode. There are four types of undulatory swimming modes: anguilliform, subcarangiform, carangiform and thunniform. A detailed review of these modes can be found in Ref. 27. In this work, the values of $c_0 = 0.02, c_1 = -0.12$ and $c_2 = 0.2$ were chosen. They correspond to the carangiform mode of swimming. Equation (1) is only valid if the frequency, f, is constant. For a modulating frequency, the formula for the wave can be written as a function of the instantaneous phase:

$$y(x,t) = a(x)\,sin\left(kx - \phi(t)\right). \tag{3}$$

The instantaneous phase, $\phi(t)$ depends on the instantaneous frequency in the following way:

$$\phi(t) = 2\pi \int_{t_0}^{t} f(t)\,dt. \tag{4}$$

The amplitude of swimming can also be changed by multiplying the curve envelope by a time-varying gain $b(t)$. The final formula for the travelling wave, becomes:

$$y(x,t) = b(t)\,a(x)\,sin\left(kx - \phi(t)\right). \tag{5}$$

Equation (5) models the undulation of the swimmer's backbone. To complete the geometrical description of the swimmer, its shape should also be defined. In this work, an approach similar to the one taken in Ref.28 is used. The shape of the swimmer is modelled by an airfoil parametrized by a karman-trefftz transformation. The karman-trefftz transformation is defined by the following conformal map:

$$z = nb\frac{(\zeta+m)^n + (\zeta-m)^n}{(\zeta+m)^n - (\zeta-m)^n}, \tag{6}$$

Where $z = xi + y$ is the the complex variable in the new space and $\zeta = \eta i + \theta$ is the complex variable in the original space. The constant m determines the positions where $\frac{d\zeta}{dz} = 0$ and n is related to the tail angle, β, through the following formula: $\beta = (2-n)\,\pi$. To set the length of swimmer to some value l, the shape is first translated so that the leading edge coincides with $x = 0$ and then the shape is re-scaled to ensure that the backbone satisfies $0 \leq x \leq l$. The shape of the swimmer can be obtained with the 5 following parameters: $(\eta_c, \theta_c), n, m, l$. The variables (η_c, θ_c) correspond to the coordinates of the center of the circle in the original space. When the swimmer's backbone undulates according to equation (5), its original shape is deformed and must be accordingly updated. In this work, the shape is always deformed based on the original shape depicted in figure 1 (c). Let $\mathscr{B}_s$ be the set containing the backbone coordinates defined at N discrete points: $\mathscr{B}_s = \{(\vec{x}_{bs})_i\}_{i\in\{0,\ldots N-1\}}$. Each backbone coordinate corresponds to a pair of shape coordinates that are grouped in some set $\mathscr{T}_s = \{(\vec{x}_s)_{i,j}\}_{i,j\in\{0,\ldots,N-1\}\times\{1,2\}}$ (see figure 2 (a)). At a given time t, each backbone coordinate in $\mathscr{B}_s$ can be updated according to equation (5). A new set of deformed backbone coordinates, $\mathscr{B}_d^t = \{(\vec{x}_{bd})_i\}_{i\in\{0,\ldots,N-1\}}$ is then obtained (see figure 2 (b)). To update the shape coordinates, the Euler-Bernoulli beam assumption is made. This means that

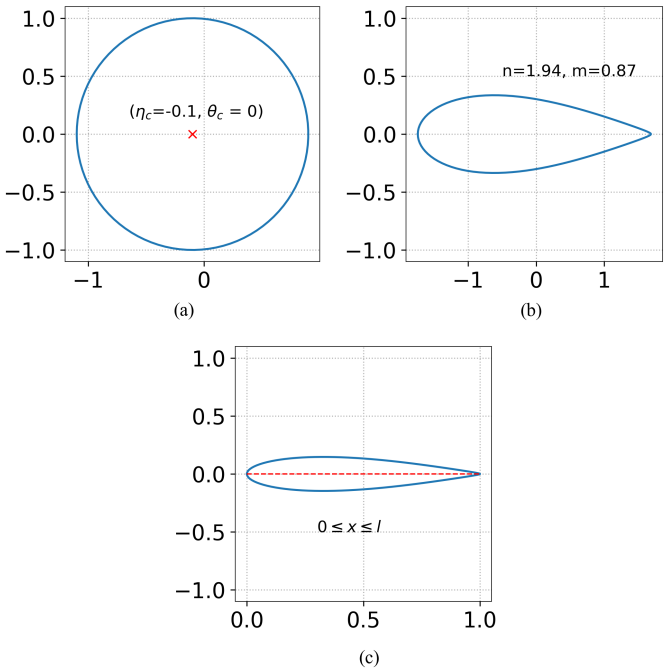

FIG. 1. Karman trefftz transformation from the original space (a) to the new space. (b) represents the shape before scaling and (c) corresponds to the shape after scaling.

the segment connecting a pair $(\vec{x_s})_{i,1}$ and $(\vec{x_s})_{i,2}$, is perpendicular to the backbone at all times. Hence, after computing the local backbone angles $\{\alpha_i\}_{i\in\{0,...,N-1\}}$ (see figure 3), the position of each pair is updated as follows:

$$\begin{aligned}(\vec{x_d})_{i,j} &= \mathbf{R}(\alpha_i)\left((\vec{x_s})_{i,j} - (\vec{x_{bs}})_i\right) + (\vec{x_{bd}})_i \\ i &= \{1,...,N-1\} \quad j = \{1,2\},\end{aligned} \tag{7}$$

Where $\mathbf{R}$ corresponds to the 2D rotation matrix of angle α:

$$\mathbf{R}(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}.$$

Applying equation (7) at a given time t, we obtain a new set of deformed shape coordinates: $\mathcal{T}_d^t = \{(\vec{x_d})_{i,j}\}_{i,j\in\{0,...,N-1\}\times\{1,2\}}$. The deformation induced by equation (7) can cause a slight change in the location of the center of mass. We assume that this deformation is independent from the rigid motion. Therefore, the center of mass is systematically reset to its previous position.

B. Governing Equations

To simulate the fluid structure interaction between a swimmer and water, we chose to solve the 2D incompressible Navier-Stokes equations. The simulation domain is represented by Ω. The domain is split into Ω_f (for the fluid) and Ω_s (for the solid/swimmer). The interface between the fluid

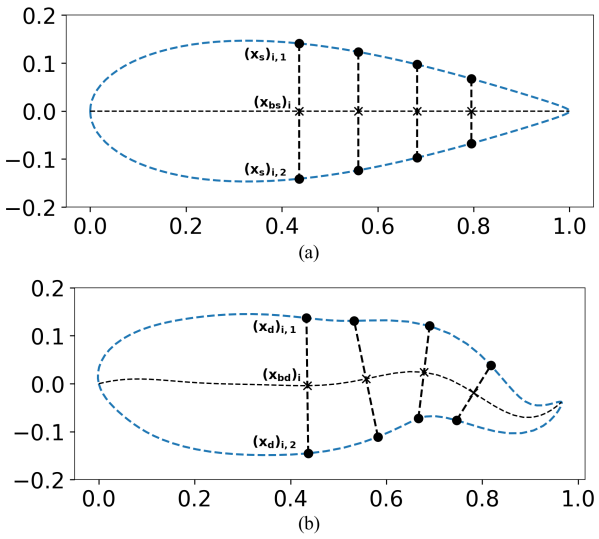

FIG. 2. Shape coordinates before deformation (a) and after deformation (b)

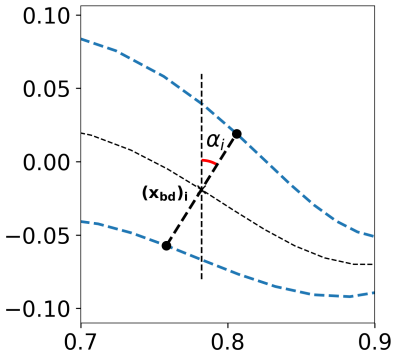

FIG. 3. Local backbone angle

and the solid is represented by Γ_s. A sketch of the 2D domain is shown in figure 4.

The incompressible Navier-Stokes equations on the domain illustrated in figure 4 is written as follows:

$$\begin{cases} \rho\left(\frac{\partial\vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u}\right) = -\nabla p + \mu\nabla^2\vec{u} & \text{in } \Omega_f, \\ \nabla\cdot\vec{u} = 0 & \text{in } \Omega_f, \end{cases} \tag{8}$$

Where $\vec{u}$ is the fluid velocity field and p is the pressure. A no-slip boundary condition (Dirichlet boundary condition) is applied to the solid/fluid interface:

$$\vec{u}(\vec{x},t) = \vec{u}_s(\vec{x},t) \qquad \text{on } \Gamma_s, \tag{9}$$

Where $\vec{u}_s(\vec{x},t)$ is the body velocity field (inside Ω_s). The body velocity is decomposed into translation, rotation and deformation components:

$$\vec{u}_s(\vec{x},t) = \underbrace{\vec{u}_G(t)}_{\text{translation}} + \underbrace{\vec{u}_\theta(\vec{x},t)}_{\text{rotation}} + \underbrace{\vec{u}_{def}(\vec{x},t)}_{\text{deformation}} \tag{10}$$

Additionally, appropriate boundary conditions should be prescribed at the borders of the domain $\delta\Omega$:

$$B(\vec{u}) = 0 \quad \text{on } \delta\Omega. \tag{11}$$

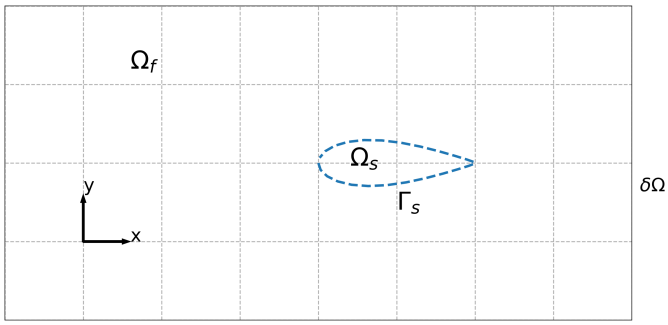

FIG. 4. 2D computational domain

C. Volume penalization method

To solve the system of equations defined in the previous section (Eq. (8)), one can use body-fitted grids and impose the no-slip boundary condition explicitly. This approach increases in complexity when the solid/fluid interface is moving and deformable. In fact, the combination of a body-fitted grid and a moving interface requires a numerical method that updates the grid at each time step. To avoid these issues, an immersed boundary method based on volume penalization is used[29] and the Navier-Stokes equation is solved on an uniform cartesian grid. In this approach, the solid body is modelled by adding an appropriate volume force to the Navier-Stokes equation. The resulting equation, also known as the Navier-Stokes Brinkman equation is written as follows:

$$\begin{cases} \rho\left(\frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u}\right) = -\nabla p + \mu\nabla^2\vec{u} + \underline{\frac{\chi_s}{K}(\vec{u}_s - \vec{u})} & \text{in } \Omega. \\ \nabla\cdot\vec{u} = 0 & \text{in } \Omega. \end{cases} \tag{12}$$

Adding the underlined term in equation (12) can be interpreted as considering the solid region, Ω_s, as a porous media with very small permeability K. The indicator function of the body, χ_s, is defined as follows:

$$\begin{cases} \chi_s(\vec{x},t) = 1 & \text{if } \vec{x} \in \Omega_s. \\ \chi_s(\vec{x},t) = 0 & \text{elsewhere.} \end{cases}$$

Equation (12) adds two additional unknowns to the system of equations. The first one is the body velocity field, $\vec{u}_s(\vec{x},t)$, defined in the region Ω_s. The second unknown is the indicator function, $\chi_s(\vec{x},t)$. The position ($\chi_s(\vec{x},t)$) and velocity ($\vec{u}_s(\vec{x},t)$) of the swimmer's body depend on both its deformation (equation (5)) as well as on the forces and torques applied by the fluid on the swimmer. The hydrodynamic forces and torques are computed by integrating the total stress tensor on the surface of the body:

$$\vec{F} = \int_{\Gamma_s} \sigma(\vec{u},p)\vec{n}\,\mathrm{d}S, \qquad \vec{M} = \int_{\Gamma_s} \sigma(\vec{u},p)\vec{n}\times\vec{r}\,\mathrm{d}S, \qquad \vec{r} = \vec{x} - \vec{x}_G, \tag{13}$$

Where $\sigma(\vec{u},p)$ is the total stress tensor, $\mathrm{d}S$ is the differential surface element and $\vec{n}$ is the normal vector to the surface. The total stress tensor depends on the velocity and pressure fields through the following relation: $\sigma(\vec{u},p) = -pI + \tau(\vec{u})$. For a Newtonian fluid, the viscous stress tensor, $\tau(\vec{u})$, is written as follows: $\tau(\vec{u}) = \mu\left(\nabla\vec{u} + \nabla\vec{u}^T\right)$. The total torque exerted by the fluid is calculated with respect to the swimmer's center of mass $\vec{x}_G$. The rigid components of the swimmer's velocity are governed by Newton's second law:

$$m\frac{\mathrm{d}\vec{u}_G}{\mathrm{d}t} = \vec{F}, \qquad \frac{\mathrm{d}(J\omega)}{\mathrm{d}t} = \vec{M}, \tag{14}$$

Where $\omega = \vec{u}_\theta \times \vec{r}$ is the angular velocity of the body and J its moment of inertia. One can see that the motion and location of the swimmer depends on the velocity and pressure fields which, in turn, depend on the motion and location of the swimmer. Therefore, the fluid-structure interaction is a coupled problem and its solution will be outlined in the following section.

D. Numerical Solution

The main idea is to solve the Navier-Stokes Brinkman equation (equation (12)) and the swimmer's kinematics (equations (7)) and dynamics ((14)) sequentially. During a time step Δt, the swimmer's position is assumed to be constant. This means that χ_s and $\vec{u}_s$ in equation (12) are known and the Navier-Stokes Brinkman equations can be solved to obtain the velocity and the pressure field at $t+\Delta t$. Once $\vec{u}(\vec{x},t+\Delta t)$ and $p(\vec{x},t+\Delta t)$ are computed, the forces and torques (equation (13)) are calculated and the swimmer is deformed (equation (7)), rotated and translated (equation (14)). The Navier-Stokes solver is based on a time explicit finite difference approach, using the well established Chorin-Temam projection scheme[30,31]. The method is first order in time, second order in space for the diffusive terms and fifth order for convective ones. Periodic boundary conditions are used in both directions. The fluid-body time integration scheme can be summarized as follows:

1. Compute the velocity and pressure fields $\vec{u}^{n+1}$ and p^{n+1}. $\vec{u}^n$ and p^n are given as well as the swimmer's position and velocities (χ_s^n, $\vec{u}_s^n$):

$$\begin{aligned} &\frac{\vec{u}^{n+1} - \vec{u}^n}{\Delta t} = -\nabla p^n + \mu\nabla^2\vec{u}^n - (\vec{u}^n\cdot\nabla)\vec{u}^n + \frac{\chi_s^n}{K}(\vec{u}_s^n - \vec{u}^n) \quad \text{in } \Omega. \\ &\nabla\cdot\vec{u}^{n+1} = 0 \quad \text{in } \Omega. \end{aligned} \tag{15}$$

2. Compute the position and velocity of the swimmer at iteration $n+1$ by deforming, rotating and translating the body.

The reader is referred to Bergmann and Iollo [28] for a complete description of the method. Both 2D and 3D validations were already performed in previous publications. One validation involves the sedimentation of a sphere[32] with various sphere diameters and fluid viscosities. For bio-inspired problems involving large deformations, another validation was conducted on an eel swimming model[33], initially proposed by Kern and Koumoutsakos [34] and later studied by Bhalla *et al.* [35]. In both cases, the numerical results align well with the reference data.

III. SPARSE IDENTIFICATION OF NONLINEAR DYNAMICS

The SINDy method is a machine learning algorithm used to identify the underlying equations that describe a dynamical system from experiments/numerical data. This approach is particularly suited in the context of optimal control where the aim is to model and extract the dynamics of a system in order to design effective control strategies. In fact, the SINDy algorithm can be readily extended to systems where control parameters drive the system's performance[36]. Based on the numerical strategies described in previous sections, simulations are performed in order to gather data and system identification using SINDy is used to learn the swimming dynamics. We assume that the system is governed by the following nonautonmous dynamical system:

$$\frac{\mathrm{d}\vec{s}}{\mathrm{d}t} = \mathbf{f}(\vec{s},\vec{c}),$$

Where $\vec{s} \in \mathbb{R}^{n_s}$ is a vector that contains the states of the dynamical system and $\vec{c} \in \mathbb{R}^{n_c}$ contains the control inputs. The variable n_s stands for the dimension of the state space, while n_c represents the dimension of the control space. The SINDy method seeks to approximate the vector field $\mathbf{f}$ by a nonlinear model of the following form:

$$\mathbf{f}(\vec{s},\vec{c}) \approx \Xi \vec{\theta}(\vec{s},\vec{c}),$$

Where $\vec{\theta} : \mathbb{R}^{n_s} \times \mathbb{R}^{n_c} \to \mathbb{R}^{n_f}$ is a library of n_f candidate functions and $\Xi \in \mathbb{R}^{n_s \times n_f}$ is matrix whose rows contain the coefficients of the candidate functions. Time series of the system's states are collected in matrix $\mathbf{X} \in \mathbb{R}^{n_s \times N_t}$, and control inputs in the matrix $\mathbf{Y} \in \mathbb{R}^{n_c \times N_t}$. The variable N_t represents the total number of samples. The following regularized least-squares problem is solved to determine the coefficients:

$$\underset{\vec{\xi}_k}{min} \quad \|\dot{\mathbf{X}}_k - \vec{\xi}_k \Theta(\mathbf{X},\mathbf{Y})\|_2 + R\left(\vec{\xi}_k\right), \tag{16}$$

The vector $\dot{\mathbf{X}}_k$ corresponds to the k_{th} row of the derivative of the states' time series and $\vec{\xi}_k$ is a vector whose components correspond to the k_{th} row of the coefficient matrix Ξ. To construct the matrix $\dot{\mathbf{X}}$, the initial preprocessing step involves estimating the derivatives of the state time series using smoothed finite differences. This method is chosen due to the oscillatory and potentially noisy nature of the time series, as illustrated in Figure 5. Before solving the least-square problem (16), a second preprocessing step is required to construct $\Theta(\mathbf{X},\mathbf{Y})$. The matrix $\Theta(\mathbf{X},\mathbf{Y}) \in \mathbb{R}^{n_f \times N_t}$ is obtained by applying the function $\vec{\theta}$ to every state and control vector in the time series. A visualization of the various matrices used is shown below:

$$\mathbf{X} = \begin{bmatrix} \vec{s}_1 & \vec{s}_2 & \dots & \vec{s}_{N_t} \end{bmatrix} \qquad \dot{\mathbf{X}} = \begin{bmatrix} \frac{\mathrm{d}\vec{s}_1}{\mathrm{d}t} & \frac{\mathrm{d}\vec{s}_2}{\mathrm{d}t} & \dots & \frac{\mathrm{d}\vec{s}_{N_t}}{\mathrm{d}t} \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} \vec{c}_1 & \vec{c}_2 & \dots & \vec{c}_{N_t} \end{bmatrix} \qquad \Xi = \begin{bmatrix} \vec{\xi}_1 \\ \vec{\xi}_2 \\ \vdots \\ \vec{\xi}_{n_s} \end{bmatrix}$$

$$\Theta(\mathbf{X},\mathbf{Y}) = \begin{bmatrix} \vec{\theta}(\vec{s}_1,\vec{c}_1) & \vec{\theta}(\vec{s}_2,\vec{c}_2) & \dots & \vec{\theta}(\vec{s}_{N_t},\vec{c}_{N_t}) \end{bmatrix}$$

Several methods can be used to solve the optimization problem stated in equation (16). Some examples are LASSO[37], sparse relaxed regularized regression[38] (SR3) and sequential thresholded least squares (STLS)[24]. The form of the regularizing term $R\left(\vec{\xi}_k\right)$ depends on the method used. In this work, the STLS algorithm is used. The method consists in repeating the following two steps until convergence:

1. Solve problem (16) with $R(\xi_k) = \alpha\|\xi_k\|_2$ (ridge regression).
2. Remove all candidate functions that are associated with a coefficient smaller than a prescribed threshold λ

The second step of STLS is key to obtaining a sparse solution and minimizing the number of nonzero terms in the coefficient matrix. In this work, the PySINDy open source library[39] is used to implement the method.

IV. OPTIMAL CONTROL

The optimal control problem formulation considered in this work is the following:

$$\underset{\vec{c}(t),\vec{s}(t)}{min} \quad \Psi\left[\vec{s}\left(t_f\right)\right] + \int_{t_0}^{t_f} \mathscr{L}\left[\vec{s}(t),\vec{c}(t)\right] \mathrm{d}t.$$

$$\begin{aligned} \text{s.t} \quad & \frac{\mathrm{d}\vec{s}}{\mathrm{d}t} = \mathbf{f}\left[\vec{s}(t),\vec{c}(t)\right], \\ & \vec{s}(t_0) = \vec{s}_0, \\ & \vec{s}\left(t_f\right) = \vec{s}_f, \\ & \vec{c}_l(t) \le \vec{c}(t) \le \vec{c}_u(t). \end{aligned} \tag{17}$$

The total cost is divided into two components. The running cost, $\mathscr{L}$, and the final cost, Ψ. We consider four constraints: the learned dynamical system, initial/terminal constraints and bounds on the control inputs. Other constraints such as path constraints on the states can be added to problem (17) but they are not considered in this work. The terminal constraint, $\vec{s}(t_f) = \vec{s}_f$, defines a goal state that we want to reach at the end of the time horizon. It will be used in the second optimal control problem (cost of transport minimization). A vast array of numerical methods have been developed over the years to solve problems similar to the one related to equations (17). They can be classified into 3 categories: direct methods, indirect methods and dynamic programming. Direct methods, especially simultaneous or all-at-once approaches, have become the preferred methods for the majority of real-world applications. Direct methods transform the infinite-dimensional optimization problem (equation (17)) to a nonlinear program (NLP) by discretizing the control space into a finite number of control parameters. Among these methods, the most prominent ones are direct collocation methods[40,41], and direct multiple shooting methods[42]. A detailed discussion of both methods can be found in Ref. 43 and Ref. 44. In this work, we use the CasADi library[45] to facilitate the implementation of direct methods. The resulting nonlinear program is solved using the open source optimizer IPOPT[46,47].

A. Model predictive control

The goal of model predictive control (MPC) is to compute an approximate solution of the problem stated in equation (17) by repeatedly solving a closed loop version for a shorter time horizon. The modified problem is formulated as follows:

$$\begin{aligned} &\min_{\vec{c}(t),\vec{s}(t)} V\left[\vec{s}(t_i+\Delta t_h)\right] + \int_{t_i}^{t_i+\Delta t_h} \mathscr{L}\left[\vec{s}(t),\vec{c}(t)\right] \mathrm{d}t \\ &\text{s.t} \quad \frac{\mathrm{d}\vec{s}}{\mathrm{d}t} = \mathbf{f}\left[\vec{s}(t),\vec{c}(t)\right], \\ &\qquad \vec{s}(t_i) = \vec{s}_m, \\ &\qquad \vec{c}_l(t) \leq \vec{c}(t) \leq \vec{c}_u(t), \end{aligned} \tag{18}$$

Where Δt_h is the time horizon and $\vec{s}_m$ is the state measured from the high-fidelity simulation. The time at which the time horizon starts is denoted by t_i. The function $V\left[\vec{s}(t_i+\Delta t_h)\right]$, known as the value function, approximates the tail of the cost beyond the prediction horizon. After solving the optimization problem in equation (18), a discrete optimal control sequence, $\vec{c}^{\,0:N_h-1} = \begin{bmatrix}\vec{c}^{\,0} & \vec{c}^{\,1} & \cdots & \vec{c}^{\,N_h-1}\end{bmatrix}$, is obtained and only the first element ($\vec{c}^{\,0}$) is applied. In this work, the do-mpc[48] open-source python library is used to solve equation (18) . The library is built on top of CasADi and makes use of the direct collocation method to solve the optimal control problem numerically.

V. RESULTS

The results section will comprise three parts: 1) Description of data generation and SINDy model; 2) Demonstration of velocity tracking via model predictive control; 3) Presentation of findings on minimizing cost of transport.

A. SINDy model

The rigid motion of the swimmer is governed by Newton's second law (equations (14)). For the rest of the paper, we assume that the swimmer's motion is constrained in the x direction. This is done by removing the contribution of the force in the y direction and the torque from equation (14). The reason for this is that frequency and amplitude gain alone are not enough to effectively control the torque applied by the fluid on the swimmer. A body curvature control input can be added for maneuvering purposes as done in Ref. 49. The explicit relationship between the thrust and the kinematic parameters is not known. To overcome this issue, the SINDy framework is used. The data is generated from the numerical simulation presented in section 2. The data correspond to time series of the velocity magnitude of the swimmer, $u_G(t)$, for multiple frequencies f, and amplitude gains b:

$$\mathscr{D}_k = \left\{ \left(\vec{s}_i, \frac{d\vec{s}_i}{dt}, \vec{c}_k \right) \right\}_{i=1}^{N_{samp}}. \tag{19}$$

The set $\mathscr{D}_k$ corresponds to the data generated by one simulation. The one dimensional state, $u_G \in \mathbb{R}$, is the velocity of the swimmer. The control vector , $\vec{c}_k = \begin{pmatrix} f_k \\ b_k \end{pmatrix} \in \mathbb{R}^2$, contains the frequency and the amplitude gain. The control vector is constant during one simulation. The index i varies from 1 to the number of samples per simulation N_{samp}. The subscript k represents the k^{th} simulation which varies from 1 to N_{sim} which is the number of simulations performed. The total number of samples, N_t, is equal to $N_{samp} \times N_{sim}$. The sampling strategy for the offline phase consists of running multiple high fidelity simulations with different constant control vectors. Fifteen high fidelity simulations ($N_{sim} = 15$) are run with the following control parameters: $f = \{1.0, 1.5, 2.0, 2.5, 3.0\}$ and $b = \{0.5, 1.0, 1.5\}$. The length of the swimmer is chosen to be $l = 1\ m$. The kinematic viscosity of the fluid is set to: $\nu = 0.001\ m^2/s$. These parameters were selected to maintain a laminar flow. The critical Reynolds number for undulatory swimmers is $Re_{critical} \approx 3000$[50]. Based on the swimming speeds obtained in the dataset, the flow Reynolds number, $\mathrm{Re} = \frac{u_G^s l}{\nu}$ is between 100 and 2000. The swimming speed reached at steady-state is denoted by u_G^s. The basis functions chosen in the SINDy model are polynomials of degree up to 4. The optimization problem (equation (16)) results in the following one state dynamical system:

$$\begin{aligned}\frac{du_G}{dt} =& -0.209u_G - 0.276u_G^2 - 0.436u_G^3 - 0.067fb - 0.109f^2b^2 \\ &+ 0.148u_Gfb - 0.261u_G^2fb - 0.071u_Gf^2b - 0.344u_Gfb^2.\end{aligned} \tag{20}$$

The model's fit on the training data is illustrated in figure 5. The dashed lines correspond to the solution of the SINDy model whereas the solid lines correspond to the data obtained from the high fidelity simulations.

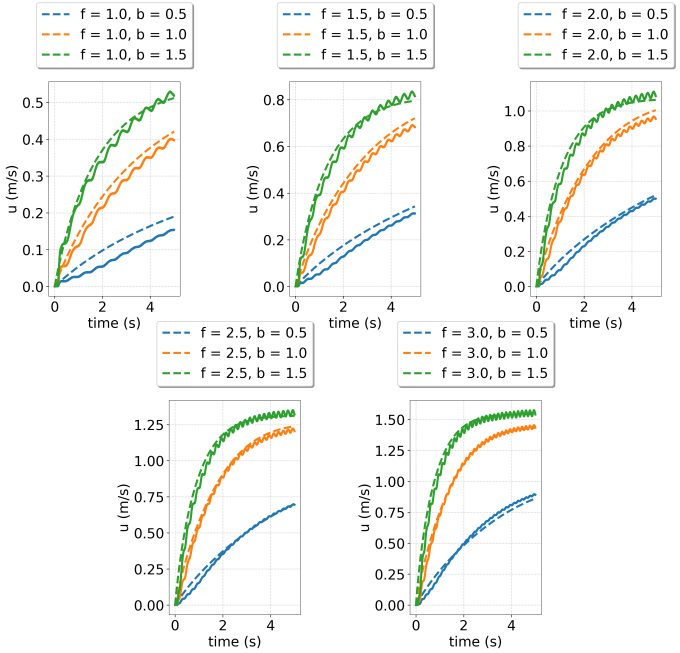

FIG. 5. Sindy model's fit on the training data

B. Velocity tracking

In this section, the goal for the swimmer is to track a predefined reference velocity. This velocity tracking problem is solved with MPC. The discrete version of the optimal control problem is written as follows:

$$\begin{aligned}\underset{u_G^{0:N_h},\vec{c}^{0:N_h-1}}{min} \quad & \left(u_G^{N_h} - u_{ref}^{N_h}\right)^2 \\ & + \sum_{l=0}^{N_h-1}\left[\left(u_G^l - u_{ref}^l\right)^2 + \Delta\vec{c}^{\,l}R\Delta\vec{c}^{\,l}\right] \\ \text{s.t} \quad & u_G^{l+1} = G\left(u_G^l, \vec{c}^{\,l}\right), \\ & u_G^0 = (u_G)_m \\ & f_{min} \le f^l \le f_{max}, \\ & b_{min} \le b^l \le b_{max}.\end{aligned} \tag{21}$$

The variable N_h represents the number of time intervals within the given time horizon. The time step for each interval is denoted by $\Delta\tau$. The index l specifies the time, $t = t_i + l\Delta\tau$, at which a given variable is evaluated. The function $G\left(u_G^l, \vec{c}^{\,l}\right)$ corresponds to the time integration of the learned model (equation (20)). The optimal control sequence, $\vec{c}^{\,0:N_h-1}$, is the result of solving problem 21. As mentioned before, only the first element of the sequence is applied. A test case with the following parameters is solved: $t_0 = 0\,s$, $t_f = 6\,s, f_{min} = 0\,Hz$, $f_{max} = 3\,Hz, b_{min} = 0$, $b_{max} = 1.5$. The same bounds on the control inputs will be used for the rest of this study. The time step for the MPC scheme is chosen to be $\Delta\tau = 0.03\,s$ and N_h is set to 19. This corresponds to a time horizon of 0.6 seconds. The weighting matrix R is set to the identity matrix. A change in reference velocity, $u_{ref}(t)$, is also considered in this case:

$$\begin{cases} u_{ref}(t) = 0.5\,m/s & 0 \le t \le 3. \\ u_{ref}(t) = 1.0\,m/s & 3 < t \le 6. \end{cases}$$

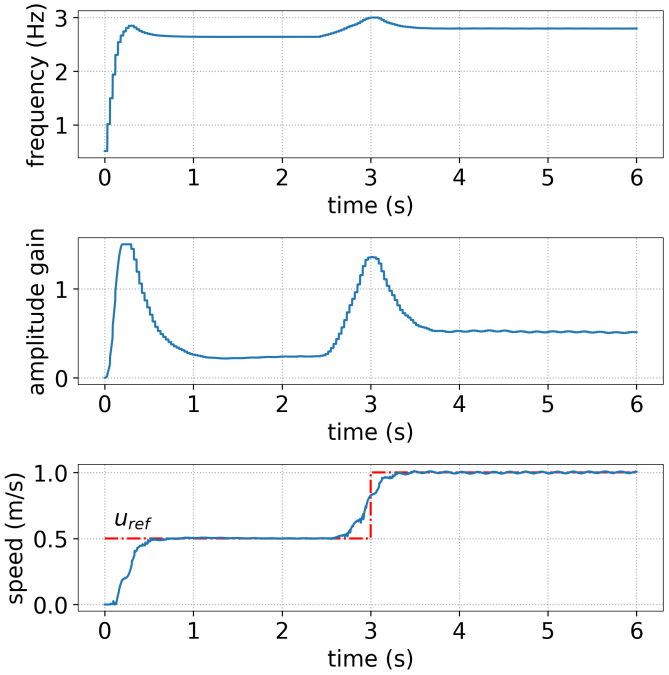

FIG. 6. Trajectory tracking with MPC. First two plots are the control inputs. Last plot is the speed of the swimmer

The results are illustrated in figure 6. It shows a good tracking with respect to the reference velocity. Additionally, the predictive nature of MPC allows the swimmer to anticipate the change in $u_{ref}(t)$ at 3 seconds. To quickly reach the desired reference speed, the swimmer adjusts its frequency and amplitude gain to their maximum allowable values and then decreases them to some constant value to stay at the desired speed.

C. Cost of transport optimization

The cost of transport of the swimmer is a measure of how much energy is spent to travel a certain distance. The cost of transport[8] is expressed as :

$$CoT\left(t_f\right) = \frac{\int_{t_0}^{t_f} P_{def}\mathrm{d}t}{\int_{t_0}^{t_f} |u_G|\mathrm{d}t} = \frac{e\left(t_f\right)}{d\left(t_f\right)}, \tag{22}$$

Where $e\left(t_f\right)$ is the total energy spent and $d\left(t_f\right)$ is the total distance traveled during time t_f. The instantaneous deformation power, P_{def}, is a measure of the mechanical power spent by the swimmer to deform its body in order to fulfill the kinematic law (5):

$$P_{def} = -\int\int_{\Gamma_s} \max\left(\vec{u}_{def}\cdot \mathrm{d}\vec{F},\quad 0\right). \tag{23}$$

The deformation velocity, $\vec{u}_{def}$, is estimated by a finite difference between two sets of deformed shape coordinates ($\mathscr{T}_d^t$ and $\mathscr{T}_d^{t-\Delta t}$). The hydrodynamic force applied on a differential surface area element of Γ_s is denoted by $\mathrm{d}\vec{F}$. To find a control law that minimizes the cost of transport, it is first necessary to learn an appropriate dynamical model for the evolution of the total energy $e(t)$. We assume that $e(t)$ and $u_G(t)$ are independent. Moreover, if either the frequency or amplitude gain is zero, the derivative of e with respect to time ($\frac{\mathrm{d}e}{\mathrm{d}t}$) should be 0. Thus, polynomials solely involving e are not utilized. The SINDy algorithm is then applied using the same dataset, resulting in the following three-state dynamical system:

$$\begin{cases} \dfrac{\mathrm{d}x_G}{\mathrm{d}t} = u_G. \\ \dfrac{\mathrm{d}u_G}{\mathrm{d}t} = -0.209u_G - 0.276u_G^2 - 0.436u_G^3 - 0.067fb \\ \qquad -0.109f^2b^2 + 0.148u_Gfb - 0.261u_G^2fb \\ \qquad -0.071u_Gf^2b - 0.344u_Gfb^2. \\ \dfrac{\mathrm{d}e}{\mathrm{d}t} = 0.073fb - 0.068f^2b - 0.050fb^2 + 0.050f^2b^2 \\ \qquad +0.017f^3b + 0.031efb + 0.013e^2fb \\ \qquad -0.018ef^2b - 0.018efb^2. \end{cases} \tag{24}$$

The following optimal control is solved to minimize the cost of transport:

$$\begin{aligned} \underset{\vec{s}^{0:N_f},\vec{c}^{0:N_f-1}}{min} \quad & e\left(t_f\right) + \sum_{l=0}^{N_f-1}\left[\Delta\vec{c}^{\,l} R \Delta\vec{c}^{\,l} + \left(\vec{c}^{\,l}\right)^T Q\vec{c}^{\,l}\right], \\ \text{s.t} \quad & \vec{s}^{\,l+1} = G\left(\vec{s}^{\,l},\vec{c}^{\,l}\right), \\ & \vec{s}(t_0) = \vec{s}_0, \\ & x_f - \varepsilon \le x\left(t_f\right) \le x_f + \varepsilon, \\ & f_{min} \le f^l \le f_{max}, \\ & b_{min} \le b^l \le b_{max}. \end{aligned} \tag{25}$$

The dynamical system constraint used in the optimal control problem corresponds to the one learned with SINDy (equation (24)). Problem (25) is more difficult to solve with model predictive control than the tracking problem tackled in the previous section. This is due to the terminal cost, $e\left(t_f\right)$, and the terminal constraint, $x_f - \varepsilon \le x\left(t_f\right) \le x_f + \varepsilon$. In fact, utilizing a shorter time horizon could result in infeasible solutions $\left(x\left(t_f\right) > x_f + \varepsilon \quad \text{or} \quad x\left(t_f\right) < x_f - \varepsilon\right)$. Additionally, this shorter horizon necessitates an accurate value function (V in equation (18)). For these reasons, an open-loop solution to the optimal control problem (25) will be computed and compared to two baseline solutions. In what follows, all solutions that are compared are verified to satisfy the terminal constraint. A test case is solved for the following parameters: $\vec{s}_0 = (0,0,0)$, $x\left(t_f\right) = 4\,m$, $t_0 = 0\,s$, $t_f = 12\,s$, $N_f = 60$, $\varepsilon = 0.04$, $R = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.01 \end{pmatrix}$ and $Q = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.001 \end{pmatrix}$. The open loop solution, $\vec{c}^{\,*}(t)$, shown in figure 7(b) leads to a different trajectory when used in the high fidelity simulation. As depicted in figure 8, we observe a discrepancy between the high fidelity results in which the open loop solution has been applied and the SINDy solution. Indeed, it is not guaranteed that the resulting trajectory will satisfy the terminal constraint (even though it does in this case). This discrepancy primarily stems from the generalization error that arises when utilizing out-of-sample control inputs. We suspect that using time-varying control parameters to obtain the training data could reduce the discrepancies between the SINDy model and the numerical high-fidelity results.

Figure 7(b) illustrates that the open loop solution closely resembles a burst and coast strategy which is a well-documented technique observed in fish to minimize energy consumption[51,52]. It consists of a phase of active undulation followed by a gliding phase. It has been shown that this strategy results in higher energy savings at lower average velocities[52]. However, the effectiveness depends on the specific frequency and amplitude of the burst phase, and unoptimized burst and coast swimming can be less efficient than continuous swimming[52]. In the simulation, the swimmer actively undulates during 6 seconds ($f > 0, b > 0$) and glides for the remaining 6 seconds ($f \approx 0, b \approx 0$). During the active phase the swimmer progressively accelerate to a velocity that is greater than $0.33\,m/s$ (constant velocity required to travel a distance of $4\,m$ in $12\,s$). The swimmer spends most of the

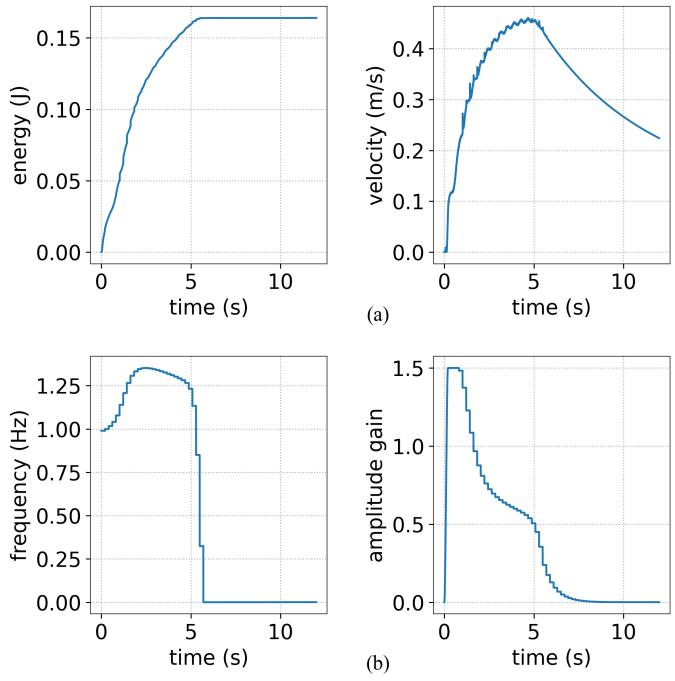

FIG. 7. Open loop solution of problem (25). Part (a) represents trajectories for states e and u_G and part (b) shows control inputs trajectories

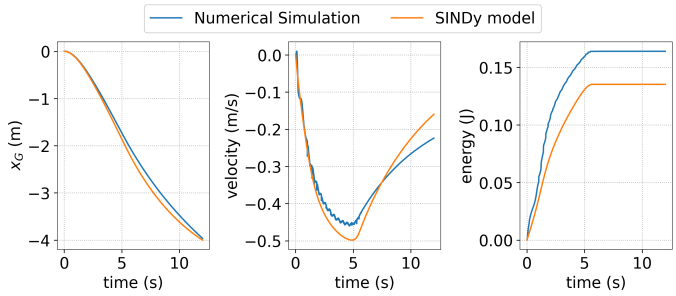

FIG. 8. Discrepancy between CFD and SINDy results

energy during the acceleration phase and uses the acquired inertia to glide for the rest of the simulation while spending no energy at all. A visual representation of the vorticity field is illustrated in figures 10 and 11. Three snapshots of the vorticity field are shown during the burst phase (figure 10) and 3 other ones are shown during the coast phase (figure 11). One can see clearly that vortex shedding stops during coasting due to the absence of active undulation. We compare the results shown in figure 7 (a) with two cases of continuous swimming (constant frequency and amplitude gain). Control inputs are determined through trial and error to meet the terminal constraint. The state and control trajectories are shown in figure 9. Table I compares the burst-and-coast solution to the two continuous cases.

All compared trajectories meet the terminal constraint. The continuous case with high frequency and low amplitude ($f = 1.55, b = 0.5$) performs the worst in terms of energy consumption. The burst-and-coast solution and the low frequency/high amplitude gain case ($f = 1, b = 0.85$) exhibit comparable energetic performance (around 1% difference). It is important to note that the burst and coast solution is a local optimum to the optimization problem (25). This optimization problem uses the learned SINDy model. Therefore, the local optimum differs from that of the high-fidelity simulation due to model mismatches.

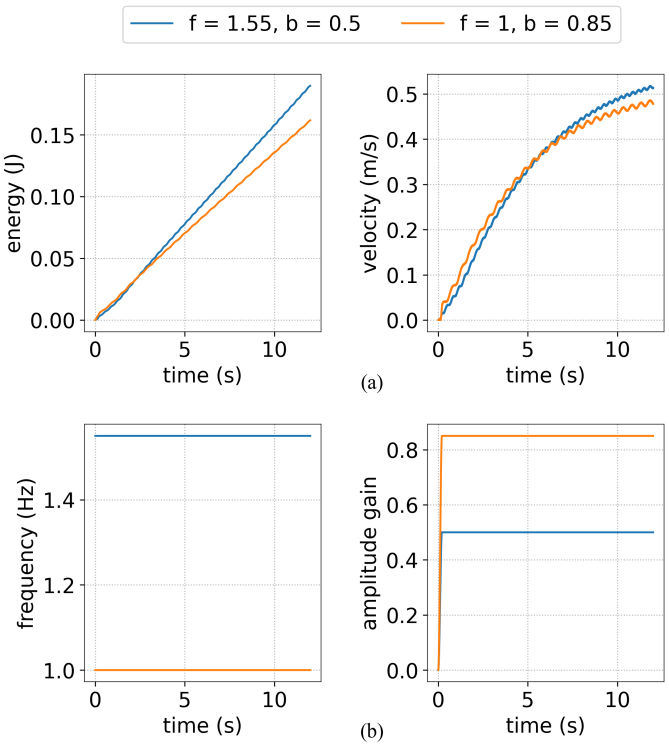

FIG. 9. Continuous swimming cases. Part (a) represents trajectories for states e and u_G and part (b) shows control inputs trajectories

TABLE I. Table comparing burst-and-coast with the two continuous swimming cases

Cases	Total energy $e(t_f)$ (J)	Terminal position $x(t_f)$ (m)
Burst-and-coast	0.164	3.96
$f = 1.55, b = 0.5$	0.19	3.98
$f = 1, b = 0.85$	0.162	3.96

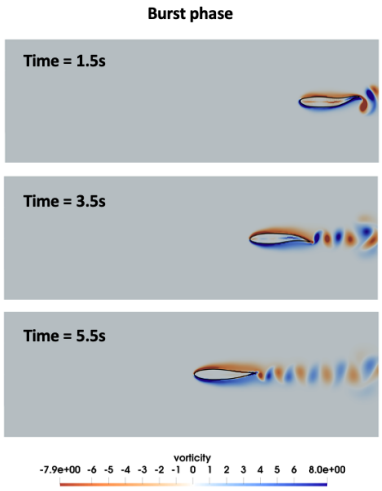

FIG. 10. Vorticity field during burst phase: active undulation during the first half of the simulation

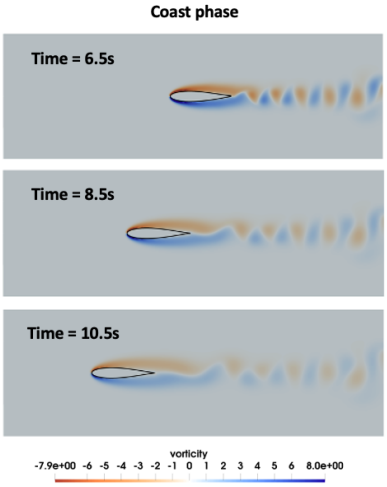

FIG. 11. Vorticity field during the coast phase: the swimmer stops undulating until the simulation ends

VI. CONCLUSION

This paper introduces a simple approach for computing numerical solutions to optimal control problems in self-propelled swimming. A high-fidelity Navier-Stokes solver based on a volume penalization method has been developed to simulate the fluid-structure interaction problem associated with self-propelled swimming. During an offline phase, training data is generated by carrying out several open loop simulations for a particular range of control inputs. The data gathered is used to train a surrogate model. Once the model is trained, the use of the model instead of the high-fidelity solver considerably reduces the computational cost, especially in many query contexts such as optimization and control. Besides solving trajectory optimization problems for self-propelled undulatory swimmers, the proposed framework can potentially be used for the design and control of bio-inspired aquatic robots. Access to accurate high-fidelity simulations for such systems can significantly aid in developing effective control policies by learning simple models from data. In this study, two model-based control problems have been explored. In both, direct methods are used to transform the optimal control problem into a finite dimensional optimization one. First, we have considered a velocity tracking problem in which model predictive control (MPC) has been used to allow the swimmer to accurately track a reference velocity signal. The MPC method allows the swimmer to quickly reach the desired speed by adjusting its frequency and amplitude. In the second problem, the swimmer's cost of transport has been minimized. By solving the optimal control problem, a solution similar to a burst-and-coast strategy has been obtained. For the test case considered, the intermittent solution shows comparable energy performance to one of the two continuous swimming cases. However, certain limitations must be acknowledged. The discrepancies between the SINDy model and the high-fidelity simulations can impact the optimality of the open-loop solution. Additionally, The current sampling strategy poses challenges. Increasing the number of control inputs, results in a large augmentation in required simulations for model training. Future research can explore several avenues to address these limitations. Methods such as modifier adaptation[53] can potentially improve the open loop solution by using high-fidelity simulation data to modify the optimization problem. Using online data to refine and retrain the surrogate model[23] also has the potential to improve the solution. The sampling strategy could also be improved. Instead of using constant control inputs, generating training data with time-varying control inputs could be more effective. Active learning strategies[54] to select control inputs can enhance both the sample efficiency and the quality of the trained model.

ACKNOWLEDGMENTS

This research was funded by the French government by means of the National Research Agency (ANR) as part of the ANR DRAGON-2 project (ANR-20-CE02-0010). Computer time for this study was provided by the computing facilities of the MCIA (Mésocentre de Calcul Intensif Aquitain).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Karl Maroun: Conceptualization (equal); writing – review and editing (equal); Writing – original draft (lead); Formal analysis (lead); Software (equal); Methodology (equal).

Michel Bergmann: Conceptualization (equal); review and editing (equal); Software (equal); Methodology (equal); Supervision (equal); Funding Acquisition (equal).
Philippe Traoré: Supervision (equal); review and editing (equal); Funding Acquisition (equal).

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]F. Fish and G. Lauder, "Passive and active flow control by swimming fishes and mammals," Annual Review of Fluid Mechanics **38**, 193–224 (2006).
[2]S. Scott Collis, R. D. Joslin, A. Seifert, and V. Theofilis, "Issues in active flow control: theory, control, simulation, and experiment," Progress in Aerospace Sciences **40**, 237–289 (2004).
[3]L. N. Cattafesta and M. Sheplak, "Actuators for active flow control," Annual Review of Fluid Mechanics **43**, 247–272 (2011).
[4]S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 48, edited by M. F. Balcan and K. Q. Weinberger (PMLR, New York, New York, USA, 2016) pp. 2829–2838.
[5]Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, 2017) pp. 703–711.
[6]A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018).
[7]N. Gautier, J.-L. Aider, T. Duriez, B. R. Noack, M. Segond, and M. Abel, "Closed-loop separation control using machine learning," Journal of Fluid Mechanics **770**, 442–457 (2015).
[8]G. Novati, S. Verma, D. Alexeev, D. Rossinelli, W. M. van Rees, and P. Koumoutsakos, "Synchronisation through learning for two self-propelled swimmers," Bioinspiration & Biomimetics **12**, 036001 (2017).
[9]J. Rabault, F. Ren, W. Zhang, H. Tang, and H. Xu, "Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization," Journal of Hydrodynamics **32**, 234–246 (2020).
[10]F. Ren, J. Rabault, and H. Tang, "Applying deep reinforcement learning to active flow control in weakly turbulent conditions," Physics of Fluids **33** (2021), 10.1063/5.0037371.
[11]A. G. Nair, C.-A. Yeh, E. Kaiser, B. R. Noack, S. L. Brunton, and K. Taira, "Cluster-based feedback control of turbulent post-stall separated flows," Journal of Fluid Mechanics **875**, 345–375 (2019).
[12]X. Wang, N. Deng, G. Y. Cornejo Maceda, and B. R. Noack, "Cluster-based control for net drag reduction of the fluidic pinball," Physics of Fluids **35** (2023), 10.1063/5.0136499.
[13]M. Bergmann and L. Cordier, "Optimal control of the cylinder wake in the laminar regime by trust-region methods and pod reduced-order models," Journal of Computational Physics **227**, 7813–7840 (2008).
[14]A. BARBAGALLO, D. SIPP, and P. J. SCHMID, "Closed-loop control of an open cavity flow using reduced-order models," Journal of Fluid Mechanics **641**, 1–50 (2009).
[15]M. J. Lighthill, "Large-amplitude elongated-body theory of fish locomotion," Proceedings of the Royal Society of London. Series B. Biological Sciences **179**, 125–138 (1997), publisher: Royal Society.
[16]M. Porez, F. Boyer, and A. Ijspeert, "Improved Lighthill fish swimming model for bio-inspired robots: Modeling, computational aspects and experimental comparisons," The International Journal of Robotics Research **33**, 1322–1341 (2014).
[17]S. Verma and J.-X. Xu, "Data-Assisted Modeling and Speed Control of a Robotic Fish," IEEE Transactions on Industrial Electronics **PP**, 1–1 (2016).
[18]S. Verma and J.-X. Xu, "Analytic Modeling for Precise Speed Tracking of Multilink Robotic Fish," IEEE Transactions on Industrial Electronics **65**, 5665–5672 (2018), conference Name: IEEE Transactions on Industrial Electronics.
[19]H. Yu, B. Liu, C. Wang, X. Liu, X.-Y. Lu, and H. Huang, "Deep-reinforcement-learning-based self-organization of freely undulatory swimmers," Physical Review E **105** (2022), 10.1103/physreve.105.045105.
[20]I. Mandralis, P. Weber, G. Novati, and P. Koumoutsakos, "Learning swimming escape patterns for larval fish under energy constraints," Physical Review Fluids **6** (2021), 10.1103/physrevfluids.6.093101.
[21]L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control Using Gaussian Process Regression," IEEE Transactions on Control Systems Technology **28**, 2736–2743 (2020), conference Name: IEEE Transactions on Control Systems Technology.
[22]Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. part i: Theory," AIChE Journal **65** (2019), 10.1002/aic.16729.
[23]J. Lore, S. De Pascuale, P. Laiu, B. Russo, J.-S. Park, J. Park, S. Brunton, J. Kutz, and A. Kaptanoglu, "Time-dependent solps-iter simulations of the tokamak plasma boundary for model predictive control using sindy," Nuclear Fusion **63**, 046015 (2023).
[24]S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," Proceedings of the National Academy of Sciences **113**, 3932–3937 (2016).
[25]U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton, "Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **478** (2022), 10.1098/rspa.2021.0904.
[26]D. S. BARRETT, M. S. TRIANTAFYLLOU, D. K. P. YUE, M. A. GROSENBAUGH, and M. J. WOLFGANG, "Drag reduction in fish-like locomotion," Journal of Fluid Mechanics **392**, 183–212 (1999).
[27]C. Connaboy, S. Coleman, and R. H. Sanders, "Hydrodynamics of undulatory underwater swimming: A review," Sports Biomechanics **8**, 360–380 (2009).
[28]M. Bergmann and A. Iollo, "Modeling and simulation of fish-like swimming," Journal of Computational Physics **230**, 329–348 (2011).
[29]P. Angot, C.-H. Bruneau, and P. Fabrie, "A penalization method to take into account obstacles in incompressible viscous flows," Numerische Mathematik **81**, 497–520 (1999).
[30]A. J. Chorin, "On the convergence of discrete approximations to the navier-stokes equations," Mathematics of Computation **23**, 341–353 (1969).
[31]R. Témam, "Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (ii)," Archive for Rational Mechanics and Analysis **33**, 377–385 (1969).
[32]M. Bergmann, J. Hovnanian, and A. Iollo, "An accurate cartesian method for incompressible flows with moving boundaries," Communications in Computational Physics **15**, 1266–1290 (2014).
[33]M. Bergmann, A. Iollo, and R. Mittal, "Effect of caudal fin flexibility on the propulsive efficiency of a fish-like swimmer," Bioinspiration and Biomimetics **9**, 046001 (2014).
[34]S. Kern and P. Koumoutsakos, "Simulations of optimized anguilliform swimming," Journal of Experimental Biology **209**, 4841–4857 (2006).
[35]A. P. S. Bhalla, R. Bale, B. E. Griffith, and N. A. Patankar, "Fully resolved immersed electrohydrodynamics for particle motion, electrolocation, and self-propulsion," Journal of Computational Physics **256**, 88–108 (2014).
[36]E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **474**, 20180335 (2018).
[37]R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," Journal of the Royal Statistical Society: Series B (Methodological) **58**, 267–288 (1996).
[38]K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, and J. N. Kutz, "A unified sparse optimization framework to learn parsimonious physics-

informed models from data," IEEE Access **8**, 169259–169271 (2020).
[39]A. Kaptanoglu, B. de Silva, U. Fasel, K. Kaheman, A. Goldschmidt, J. Callaham, C. Delahunt, Z. Nicolaou, K. Champion, J.-C. Loiseau, J. Kutz, and S. Brunton, "Pysindy: A comprehensive python package for robust sparse system identification," Journal of Open Source Software **7**, 3994 (2022).
[40]L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," Computers & Chemical Engineering **8**, 243–247 (1984).
[41]C. Hargraves and S. Paris, "Direct trajectory optimization using nonlinear programming and collocation," Journal of Guidance, Control, and Dynamics **10**, 338–342 (1987), publisher: American Institute of Aeronautics and Astronautics.
[42]H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems *," IFAC Proceedings Volumes **17**, 1603–1608 (1984).
[43]M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," SIAM Review **59**, 849–904 (2017).
[44]C. Kirches, "The direct multiple shooting method for optimal control," in *Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control* (Vieweg+Teubner Verlag, 2011) p. 13–29.
[45]J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," Mathematical Programming Computation **11**, 1–36 (2018).
[46]A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," Mathematical Programming **106**, 25–57 (2005).
[47]A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Local convergence," SIAM Journal on Optimization **16**, 32–48 (2005).
[48]F. Fiedler, B. Karg, L. Lüken, D. Brandner, M. Heinlein, F. Brabender, and S. Lucia, "do-mpc: Towards fair nonlinear and robust model predictive control," Control Engineering Practice **140**, 105676 (2023).
[49]M. Bergmann, "Numerical modeling of a self-propelled dolphin jump out of water," Bioinspiration & Biomimetics **17**, 065010 (2022).
[50]M. Gazzola, M. Argentina, and L. Mahadevan, "Scaling macroscopic aquatic locomotion," Nature Physics **10**, 758–761 (2014).
[51]G. Li, I. Ashraf, B. François, D. Kolomenskiy, F. Lechenault, R. Godoy-Diana, and B. Thiria, "Burst-and-coast swimmers optimize gait by adapting unique intrinsic cycle," Communications Biology **4** (2021), 10.1038/s42003-020-01521-z.
[52]G. Li, D. Kolomenskiy, H. Liu, R. Godoy-Diana, and B. Thiria, "Intermittent versus continuous swimming: An optimization tale," Physical Review Fluids **8** (2023), 10.1103/physrevfluids.8.013101.
[53]A. Marchetti, G. François, T. Faulwasser, and D. Bonvin, "Modifier adaptation for real-time optimization—methods and applications," Processes **4**, 55 (2016).
[54]H. Mania, M. I. Jordan, and B. Recht, "Active Learning for Nonlinear System Identification with Guarantees," Journal of Machine Learning Research **23**, 1–30 (2022).