Aquarium: A Fully Differentiable Fluid-Structure Interaction Solver for Robotics Applications

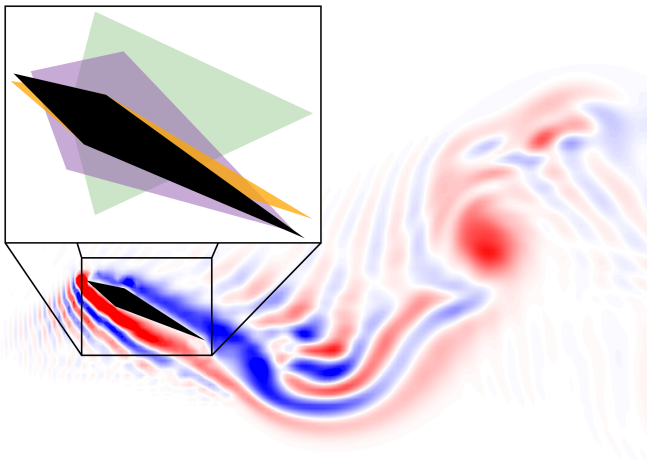
Jeong Hun Lee[1], Mike Y. Michelis[2], Robert Katzschmann[2], and Zachary Manchester[1]

***Abstract*— We present Aquarium, a differentiable fluid-structure interaction solver for robotics that offers stable simulation, accurately coupled fluid-robot physics in two dimensions, and full differentiability with respect to fluid and robot states and parameters. Aquarium achieves stable simulation with accurate flow physics by directly integrating over the incompressible Navier-Stokes equations using a fully implicit Crank-Nicolson scheme with a second-order finite-volume spatial discretization. The fluid and robot physics are coupled using the immersed-boundary method by formulating the no-slip condition as an equality constraint applied directly to the Navier-Stokes system. This choice of coupling allows the fluid-structure interaction to be posed and solved as a nonlinear optimization problem. This optimization-based formulation is then exploited using the implicit-function theorem to compute derivatives. Derivatives can then be passed to downstream gradient-based optimization or learning algorithms. We demonstrate Aquarium's ability to accurately simulate coupled fluid-robot physics with numerous 2D examples, including a cylinder in free stream and a soft robotic fish tail with hardware validation. We also demonstrate Aquarium's ability to provide analytical gradients by performing gradient-based shape-and-gait optimization of an oscillating diamond foil to maximize its generated thrust.**

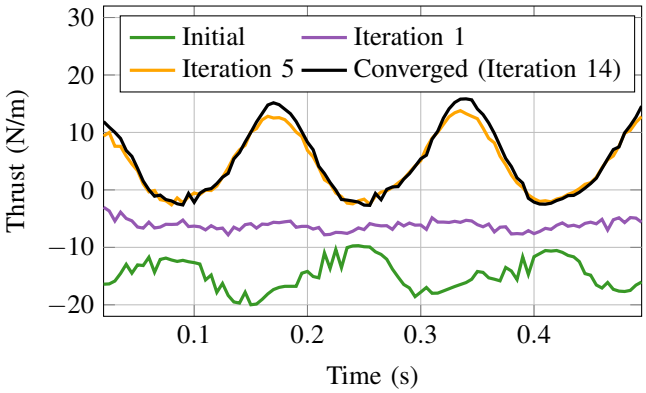
I. INTRODUCTION

In recent years, there has been considerable interest in bio-inspired locomotion for underwater [1]–[6] and aerial vehicles [7]–[10] involving complex interactions with the fluid environment. For example, the energy efficiency and high maneuverability of fish-like propulsion are attributed to the sensing and shedding of vortices [11], [12]. This complex fluid interaction is also present in the flight of flapping-wing aerial vehicles [13] and high-angle-of-attack perching maneuvers of fixed-wing airplanes [14], [15], and has led to the development of various robotic hardware systems, including oscillating-foil propulsion for marine vehicles [16], various biomimetic underwater systems [4], [17]–[19], and flapping-wing, micro-aerial vehicles [20]–[22].

In contrast, there has been comparatively little work *jointly* optimizing both the systems' design and control parameters [23], [24] while reasoning about the fluid environment [25], [26]. Such an optimization could provide insight into the well-known passive swimming dynamics of a dead trout [27] and lead to robotic systems with similar capabilities. We believe that a major reason for the lack of work in this direction is the absence of an open-source, accurate, stable simulator with *full differentiability*.



(a) Progression of foil shape-gait profiles throughout the optimization process. Shape colors are in reference to those in Figure 1b.



(b) Time history of generated thrust by foil shape-gait profiles throughout the optimization process.

Fig. 1: Shape-and-gait co-optimization of an oscillating diamond foil using the limited-memory BFGS (L-BFGS) algorithm with analytical gradients provided by Aquarium. The foil thickness, heave, pitch, and pitch-heave phase are optimized to maximize the generated thrust. A fully converged solution (black) is found after only 14 L-BFGS iterations from a drag-inducing initial guess (green).

In recent years, various differentiable simulators [28]–[37] have been deployed for a wide range of uses in robotics. These solvers are able to provide gradients that can be passed directly to optimization-based frameworks such as reinforcement learning and model-predictive control.

*This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE2140739.

[1] The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA jeonghunlee@cmu.edu, zacm@cmu.edu

[2] Soft Robotics Lab, ETH Zurich, Zurich, Switzerland michelism@ethz.ch, rkk@ethz.ch

Differentiable simulators can also be integrated into neural networks during backpropogation to achieve high sample efficiency and accuracy [24], [38], [39].

In the fluids community, various computational fluid dynamics (CFD) simulators also exist [26], [40]–[50]. However, these solvers each possess at least one key deficiency that limits their usability for robotics. These deficiencies include poor accuracy, generalizability, and computational efficiency; instability over larger time steps due to explicit integration; lack of full differentiability for robotics tasks (e.g., control) in unsteady flow; and inability to handle fluid-structure interaction (FSI) to properly simulate the unsteady, multi-physics coupling between the fluid and robot dynamics.

We propose **Aquarium**, an *open-source, physics-based, fully differentiable* solver for simulating the two-dimensional (2D) coupled dynamics between robotic systems and their surrounding fluid environment. The single-phase fluid dynamics are solved by integrating over the governing Navier-Stokes equations directly with a fully implicit Crank-Nicolson scheme to preserve stability over large time steps. The fluid discretization is handled using a second-order finite-volume method, while the fluid-robot coupling is achieved using the immersed boundary method [51], which separates the fluid and robot meshes to avoid computationally expensive re-meshing. Specifically, we build upon the work of Taira et. al. [52] and Perot [53] to pose the FSI dynamics as an optimization problem, with the fluid-robot coupling acting as equality constraints to satisfy the no-slip boundary condition. Analytical gradients are computed by applying the implicit function theorem directly to the FSI problem. Aquarium is currently implemented in 2D for rigid bodies, but the methodology is generalizable to 3D flow and soft bodies. In summary, Aquarium offers:

- Simulations that solve the discretized 2D Navier-Stokes equations directly with multi-physics coupling between rigid bodies and a fluid environment.
- Fully implicit time integration using Crank-Nicholson to achieve stable simulation at reasonable sample rates for control and optimization.
- Full differentiability to calculate analytical gradients with respect to fluid and robot states and parameters for use in gradient-based optimization and learning frameworks.

The remainder of the paper is organized as follows: In Section II, we provide some background on existing CFD and FSI solvers, including their uses and limitations. Section III then describes the proposed Aquarium solver. In Section IV, we provide simulation results and hardware validation on a variety of examples, including a cylinder in free stream and a flapping, soft robotic fish tail in initially still water. We then showcase the differentiability of Aquarium by performing gradient-based shape-and-gait co-optimization of an oscillating diamond foil to maximize its generated thrust. In Section V we provide final concluding remarks and discuss future work to address current limitations.

II. RELATED WORKS

A. Differentiable Fluid Dynamics

Industry-standard CFD solvers such as OpenFOAM [43], SU2 [44], ANSYS FLUENT [45], and STAR-CCM+ [46] provide accurate results for complex flows (e.g., multi-phase, heat transfer, etc.) but at high computational costs with only OpenFOAM and SU2 being open-source. Consequently, running parameter sweeps for gradient-free optimization can quickly become prohibitively expensive for high-dimensional problems. In addition, gradient-free approaches tend to be less stable and slow to convergence when compared to their gradient-based counterparts [23], [36].

The class of differentiable simulators addresses this shortcoming. Though many differentiable solvers were developed for robotics [28]–[37], there are also several works aimed towards the fluids community. PhiFlow [47] was developed as a differentiable PDE solver for deep-learning, written in frameworks that allow for automatic differentiation, such as JAX [54], PyTorch [55], and TensorFlow [56]. However, this method is mainly directed at controlling fluids directly by solving the governing Navier-Stokes equations and does not support FSI. Similarly, JAX-FLUIDS [48] developed a level-set method for differentiable, compressible, two-phase fluid simulations in JAX. The work implements various boundary conditions, including immersed boundaries for rigid bodies, but only supports explicit integration and, similar to PhiFlow, focuses on deep learning of fluid dynamics and does not currently support FSI.

B. Design Optimization using Adjoint Methods

While not usually called "differentiable simulators," several conventional CFD frameworks [43]–[46] implement adjoint methods [57], [58] to efficiently provide gradients for large-scale shape optimization problems in steady-state flow conditions. SU2 additionally offers shape optimization capabilities in unsteady environments. Rather than solving for the fluid-model derivatives explicitly, these frameworks calculate Jacobian-vector products to efficiently calculate gradients of the optimization problem. This is equivalent to reverse-mode automatic differentiation that is widely used in machine learning [54]–[56]. However, the extension of adjoint methods to other applications that are critical for robotics (e.g., control in unsteady flow) are relatively unexplored and not available as open-source platforms [59]–[61].

C. Fluid-Structure Interaction for Optimization

As previously mentioned, conventional CFD simulators [43]–[46] are currently limited to gradient-based shape optimization via adjoint methods. Using FSI for gradient-based, non-shape (e.g., gait) optimization in unsteady flow is still challenging, with previous work simplifying the fluid model to potential flow [6] or Stokes flow [49], [62], [63] for low-Reynolds-number regimes. Recently, Nava et al. proposed a physics-informed, neural-network model of FSI for the optimization of soft robotic swimmers [26]. However, no guarantees can be given when generalizing to new shapes and flow conditions.

Finally, Liu et al. [50] have implemented an FSI extension based in the OpenAI Gym environment. While not differentiable, their fluid solver implementation, a GPU-optimized lattice-Boltzmann method, is highly efficient and can be integrated with reinforcement-learning pipelines. The FSI is achieved with an immersed-boundary method and various swimming bodies are modeled as articulated rigid bodies. The advantages of coupling robot dynamics with full fluid solvers, as opposed to approximated fluid models [14], [64], [65] are shown in several applications, such as leveraging Kármán vortices for faster propulsion in swimming [1], [25] and greater lift in flight of flapping-wing microrobots [40].

III. DIFFERENTIABLE FLUID-STRUCTURE INTERACTION

Computationally modeling fluid dynamics and fluid-structure interaction using the Navier-Stokes equations has been extensively studied [51]–[53], [66]–[68]. Rather than describing the methods in detail, we highlight the key concepts and refer the reader to existing literature on CFD and FSI for more details. Specifically, our work is most closely related to that of Taira et. al [52] and Perot [53].

A. Implicit Fluid Model

We begin with the non-dimensionalized, incompressible Navier-Stokes equations, which express conservation of momentum and mass for Newtonian fluids:

$$\frac{du}{dt} + (u \cdot \nabla)u = -\nabla p + \frac{1}{Re}\nabla^2 u + a_{ext}, \tag{1}$$

$$\nabla \cdot u = 0, \tag{2}$$

where u, p, a_{ext}, and Re are the fluid velocities, pressure, acceleration due to external forces (i.e., gravity, etc.), and non-dimensional Reynolds number, respectively. The Reynolds number is further defined as

$$Re = \frac{\rho u_{ref} l_{ref}}{\mu}, \tag{3}$$

where ρ is the fluid density, μ is the dynamic viscosity of the fluid, u_{ref} is a reference velocity (e.g., free-stream velocity), and l_{ref} is a reference length (e.g., width of the robot).

Using a second-order finite-volume method, we express the continuous, partial derivatives of (1) and (2) as discrete operations over a spatial fluid grid:

$$(u \cdot \nabla)u \Rightarrow N(u), \qquad \nabla^2 u \Rightarrow Lu + bc_L,$$
$$\nabla p \Rightarrow Gp, \qquad \nabla \cdot u \Rightarrow Du + bc_D,$$

where $N(u) \in \mathbb{R}^{n_u}$ is the nonlinear convective term, $G \in \mathbb{R}^{n_u \times n_p}$ is the gradient operator, $L \in \mathbb{R}^{n_u \times n_u}$ is the Laplacian operator, $D \in \mathbb{R}^{n_p \times n_u}$ is the divergence operator, $bc_L \in \mathbb{R}^{n_u}$ is the boundary condition term corresponding to the Laplacian, and $bc_D \in \mathbb{R}^{n_p}$ is the boundary condition term corresponding to the divergence. Then, using implicit Crank-Nicolson integration over a time step, we have the discrete, incompressible Navier-Stokes equations,

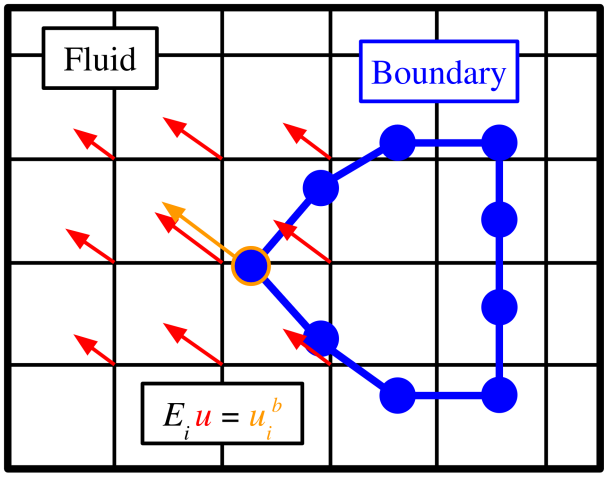


Fig. 2: The immersed-boundary method, where the fluid domain is represented by a fixed Eulerian grid (black), and the boundary of the rigid body (i.e., robot) is represented by a moving Lagrangian mesh (blue). The meshes are coupled by a convolution matrix E that maps fluid-cell velocities (red) to those of the boundary nodes (orange).

$$R(u_{k+1}, u_k, p_{k+1}) = Au_{k+1} + \frac{1}{2}N(u_{k+1}) - r(u_k) + Gp_{k+1} = 0, \tag{4}$$

$$c_1(u_{k+1}) = Du_{k+1} + bc_D = 0, \tag{5}$$

where $u_{k+1} = [u_x, u_y]^T_{k+1} \in \mathbb{R}^{n_u}$, $p_{k+1} \in \mathbb{R}^{n_p}$, and A and r are defined as follows:

$$A = \frac{1}{\Delta t} I - \frac{1}{2Re} L, \tag{6}$$

$$r(u_k) = \left[\frac{1}{\Delta t} I + \frac{1}{2Re} L\right] u_k - \frac{1}{2}N(u_k) + \frac{1}{Re} bc_L + a_{ext}. \tag{7}$$

Interestingly, it has been noted that $-G^T = D$ [52]. Therefore, p_{k+1} can be interpreted as a Lagrange multiplier enforcing the conservation-of-mass constraint imposed by (5).

B. Fluid-Structure Interaction Model

To model the FSI, we extend the idea of treating the Navier-Stokes equations as an optimization problem. First, the FSI is modeled as an additional constraint in the form of no-slip boundary conditions, where the velocity of the fluid must equal the velocity of the robot at the boundary. To do so, we use the immersed-boundary method [51], which separates the fluid (u) and robot (x^b) meshes into a fixed Eulerian grid and a floating Lagrangian-boundary mesh, respectively, as illustrated in Figure 2. The coupling between the meshes is formulated as a convolution matrix, $E \in \mathbb{R}^{n_b \times n_u}$ that maps the fluid cell velocities to the boundary node locations, allowing us to formulate the no-slip constraint,

$$E(\theta)u_{k+1} = u^b_{k+1}, \tag{8}$$

where $u^b_{k+1} \in \mathbb{R}^{n_b}$ contains the velocities of the boundary nodes and $\theta \in \mathbb{R}^{n_\theta}$ represents parameters of the robot, such as shape parameters and the state, x_{k+1}. We then apply (8)

to our Navier-Stokes formulation as an additional constraint, providing us with the full FSI problem:

$$R\begin{pmatrix} u_{k+1}, u_k, p_{k+1}, \\ \tilde{f}^b_{k+1}, \theta \end{pmatrix} = \begin{aligned} &Au_{k+1} + \frac{1}{2}N(u_{k+1}) - r(u_k) \\ &+Gp_{k+1} + E(\theta)^T \tilde{f}^b_{k+1} = 0, \end{aligned} \tag{9}$$

$$c_1(u_{k+1}) = G^T u_{k+1} - bc_D = 0, \tag{10}$$

$$c_2(u_{k+1}, \theta) = E(\theta)u_{k+1} - u^b_{k+1} = 0. \tag{11}$$

We solve (9)–(11) using a Gauss-Newton method, in which the equations are locally linearized to compute an update step,

$$\begin{bmatrix} A + \frac{1}{2}\frac{\partial N}{\partial u_{k+1}} & G & E^T \\ G^T & 0 & 0 \\ E & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u_{k+1} \\ \Delta p_{k+1} \\ \Delta \tilde{f}^b_{k+1} \end{bmatrix} = \begin{bmatrix} -R \\ -c_1 \\ -c_2 \end{bmatrix} \tag{12}$$

where $\tilde{f}^b_{k+1} \in \mathbb{R}^{n_b}$ acts as the dual variable for (8). Equation (12) has the structure of a Karush-Kuhn-Tucker (KKT) system, which are common in constrained optimization [69]. Upon inspection, $\tilde{f}^b_{k+1}$ effectively acts as a non-dimensional acceleration that fluid particles experience at the boundary. Therefore, forces acting on each boundary node (i.e., pressure) can be calculated directly as

$$f^b_{k+1} = -\rho \frac{h_x h_y}{s} \tilde{f}^b_{k+1}, \tag{13}$$

where $f^b_{k+1} \in \mathbb{R}^{n_b}$ are the pressure forces acting along the boundary, ρ is the fluid density, h_x and h_y are the spatial step sizes of the fluid grid discretization, and s is the step size of the boundary discretization. f^b_{k+1} can then be integrated along the surface of the robot body to obtain net forces.

C. Simulation Gradients

We exploit the structure of the KKT system defined by (12) to calculate analytical Jacobians of our FSI model [70]. For clarity, we first start by looking at our system at time t_k and group (9)–(11) to simplify the model:

$$z = (u_{k+1}, u_k, p_{k+1}, \tilde{f}^b_{k+1}) \tag{14}$$

$$g(z;\theta) = \begin{bmatrix} R(u_{k+1}, u_k, p_{k+1}, \tilde{f}^b_{k+1}, \theta) \\ c_1(u_{k+1}) \\ c_2(u_{k+1}, \theta) \end{bmatrix} = 0. \tag{15}$$

By definition, $g(z^*;\theta) = 0$ is an implicit function, where z^* represents the converged solution at each time-step. Using the implicit function theorem [71], we then compute the derivative $\frac{\partial z}{\partial \theta}$:

$$\frac{\partial g}{\partial z}\delta z + \frac{\partial g}{\partial \theta}\delta\theta = 0 \implies \frac{\partial z}{\partial \theta} = -\left(\frac{\partial g}{\partial z}\right)^{-1}\frac{\partial g}{\partial \theta}. \tag{16}$$

Expanding (16), we arrive at

$$-\underbrace{\begin{bmatrix} A + \frac{1}{2}\frac{\partial N}{\partial u_{k+1}} & G & E^T \\ G^T & 0 & 0 \\ E & 0 & 0 \end{bmatrix}}_{D'_g} \begin{bmatrix} \frac{\partial u_{k+1}}{\partial \theta} \\ \frac{\partial p_{k+1}}{\partial \theta} \\ \frac{\partial \tilde{f}^b_{k+1}}{\partial \theta} \end{bmatrix} = \begin{bmatrix} -\frac{\partial r}{\partial u_k} \\ 0 \\ 0 \end{bmatrix} \frac{\partial u_k}{\partial \theta} + \frac{\partial g}{\partial \theta}, \tag{17}$$

where D'_g is the same KKT system that appears in (12) and $\frac{\partial u_k}{\partial \theta}$ has already been computed at the previous time step t_{k-1}. Therefore, we can re-use the matrix factorization computed during the simulation step to calculate derivatives for very little additional computational cost. These Jacobians can then be passed to gradient-based solvers to optimize shapes, gaits, controls, or trajectories. This method also generalizes to gradient computations with respect to fluid states and parameters (e.g., fluid density).

IV. EXPERIMENTAL RESULTS

This section presents the results of several simulation experiments to evaluate the FSI physics of Aquarium with comparisons to both other numerical works and hardware experiments. This includes the classic cylinder-in-free-stream benchmark and a real-to-sim demonstration of a real-world soft robotic fish tail. We also demonstrate the full differentiability of Aquarium with a gradient-based optimization example that involves maximizing the thrust of an oscillating diamond foil. These examples, along with the open-source implementation of Aquarium are available at: `https://github.com/RoboticExplorationLab/Aquarium.jl`

A. Cylinder in Free Stream

We simulate the classic benchmark example of a cylinder in free-stream conditions as shown in Figure 3. To do so, we define the simulation environment to have inflow and outflow boundary conditions on the left and right boundaries, respectively. The inflow boundary condition is defined to be the free-stream velocity, u_∞, while the outflow boundary condition allows vortices to exit freely. We define the top and bottom fluid boundaries to have far-field conditions (i.e., the same velocity as inflow) to also simulate free-stream conditions when sufficiently distanced from the cylinder.

To study Aquarium's generalizability to varying Reynolds numbers, we evaluate the resulting steady-state behaviors under various free-stream velocities. As seen in Figure 4, Aquarium is able to capture both the steady-state vortex pairs at lower Reynolds numbers ($Re = 40$) and the Kármán vortex street at higher Reynolds numbers ($Re = 100$).

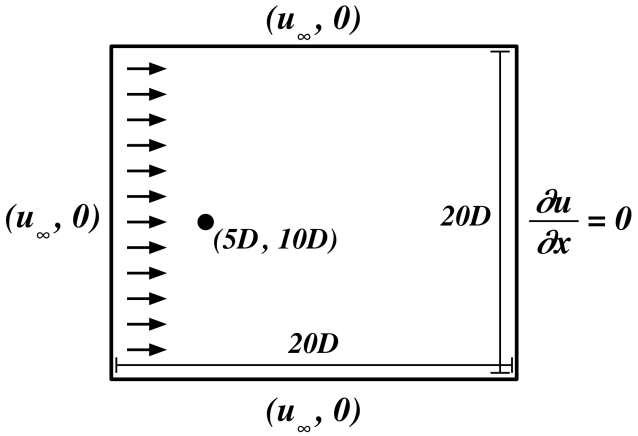
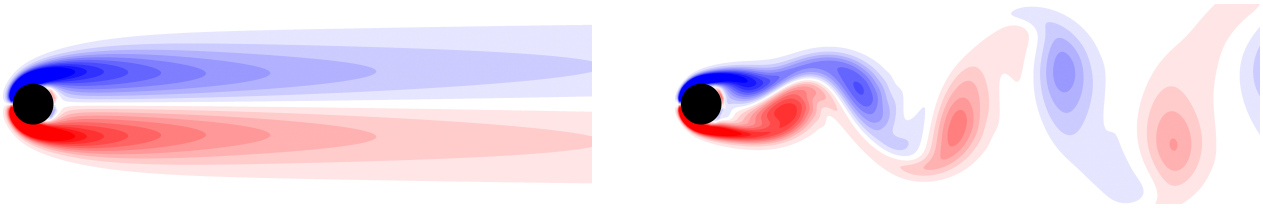
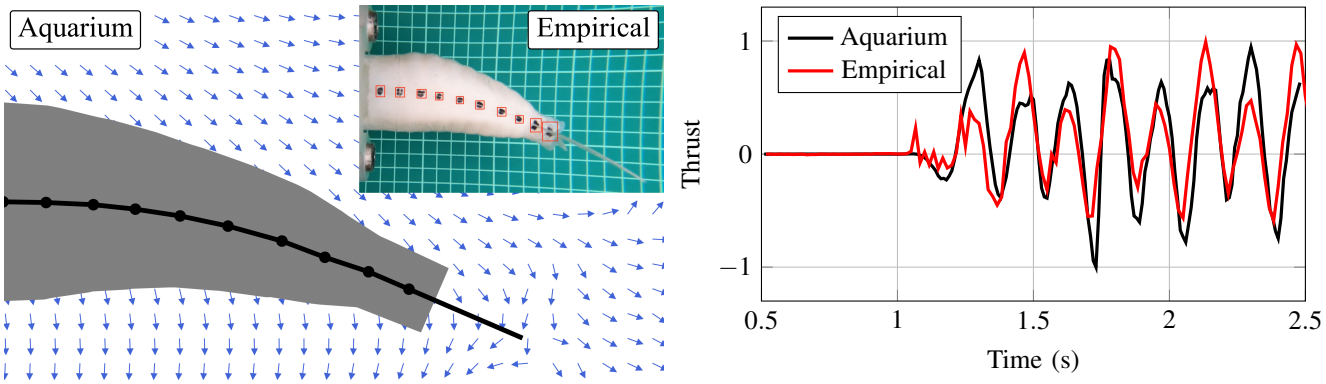


Fig. 3: Cylinder-in-free-stream simulation setup with inflow and outflow boundary conditions defined for left and right boundaries, respectively. Far-field boundary conditions are defined for the top and bottom boundaries.



(a) $Re = 40$ (b) $Re = 100$

Fig. 4: Vorticity contours of flow around a cylinder in steady-state, free-stream conditions at varying Reynolds numbers. Being Navier-Stokes based, Aquarium is able to properly simulate the vortex-shedding that occurs at higher Reynolds numbers.



(a) Aquarium simulation with corresponding hardware experiment. The tracked markers on hardware, modeled as joints of the multi-link, simulated representation, are boxed in red with corresponding center-line links and joints in simulation shown in black.

(b) Time history of normalized thrust force generated by soft robotic tail from both simulation (black) and empirical (red) results.

Fig. 5: Unsteady fluid-structure interaction of a fixed-base soft robotic fish tail with matching hardware demonstration. The robotic fish tail starts at rest in initially still water before being actuated at 3Hz. Aquarium is able to properly simulate the transient flow and resulting forcing effects that are also observed empirically, especially the phase and frequency.

To study the flow-induced forces on the cylinder, we also evaluate the resulting steady-state drag and lift coefficients. As seen in Tables I and II, there is good agreement between Aquarium and previous numerical and empirical studies. This is also true for the non-dimensional Strouhal number, which characterizes periodic vortex shedding in the wake and is critical for studying bio-inspired swimming [12].

TABLE I: Steady-state results at $Re = 40$

	Drag Coeff.
Tritton [73] (*empirical*)	1.65
Taira et. al [52]	1.55
Ren et. al [74]	1.57
Aquarium	1.75

TABLE II: Steady-state results at $Re = 100$

	Drag Coeff.	**Lift Coeff.**	**Strouhal #**
Braza et. al [75]	1.325 ± 0.008	± 0.280	0.164
Ren et. al [74]	1.335 ± 0.011	± 0.356	0.164
Aquarium	1.481 ± 0.010	± 0.362	0.174

B. Soft Robotic Fish Tail

To demonstrate Aquarium's ability to generalize beyond simple geometry in steady-state flow, we also simulate the periodic flapping of a soft robotic fish tail in initially still water, and validate it against a hardware experiment as shown in Figure 5a. The soft robotic fish tail is fabricated as described in [23], [72], and has a left and right chamber that are pneumatically actuated with 500 mbar at 3 Hz. The hardware experiment involves fixing the tail to a force sensor, which collects a time history of the net thrust force while the tail is actuated. We use previously collected video and force measurement data from [23].

In simulation, we approximate the soft robotic fish tail as a ten-link, serial-chain, rigid-body model with the joints located at corresponding marker positions along the center line of the robot, as shown in Figure 5a. The body profiles of each link are approximated by a linear interpolation between the respective joint widths, and the fin is represented as a 1D link. The simulated motion is prescribed using a forward-kinematics model, where the joint angles are determined using CSRT [76] marker tracking from a pre-recorded video of the hardware experiment. The fluid environment is also modeled to recreate the hardware experiment: a $0.6\,\text{m} \times 0.6\,\text{m}$ cavity with wall-like boundary conditions ($u_\infty = 0$) filled with initially still water. Simulated boundary pressure forces, f^b, are integrated over the boundary and compared to the empirical force-sensor measurements. To best match our 2D simulation to the 3D hardware experiment, we normalize both datasets' maximum thrust values to one.

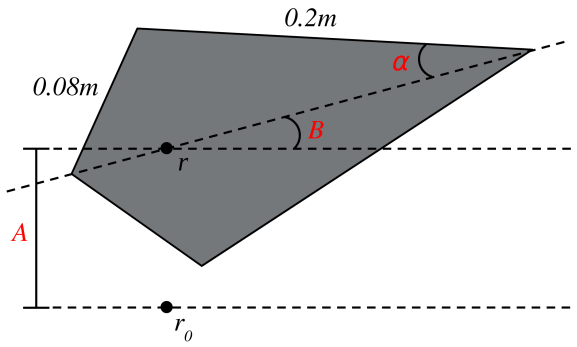


Fig. 6: Oscillating diamond-foil optimization setup with decision variables (1 shape and 3 gait parameters) shown in red. α is the angle varying the foil thickness, A is the heave amplitude, and B is the pitch amplitude. The relative pitch-heave phase, ϕ (not shown) is also a decision variable.

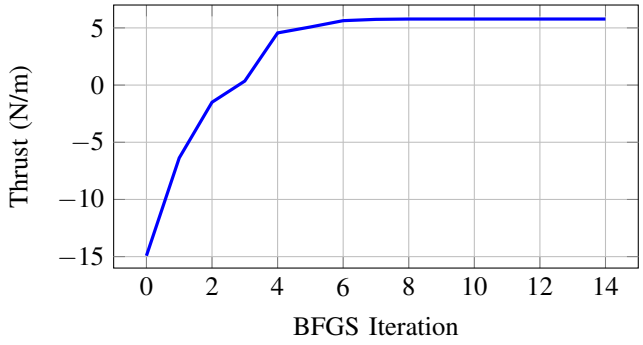


Fig. 7: The reward (time-averaged thrust) throughout the co-shape-gait optimization of an oscillating diamond foil. Using gradients provided by Aquarium, the L-BFGS algorithm quickly converges to the near-optimized solution in 7 iterations with full convergence at 14 iterations.

As seen in Figure 5b, there is good agreement between the phase and frequency of the normalized thrust forces with respect to time, demonstrating the simulation's ability to capture transient-flow effects on a moving boundary. Potential sources of error include Aquarium's inability to capture the full 3D effects (i.e., flow over the 3D contour of the fish tail) present in the experiment. This is a limitation of a 2D simulation, and future work is planned to extend Aquarium to 3D. Other potential sources of error include hardware fabrication error as well as the geometric and kinematic approximations of the multi-link representation.

C. Optimization using Aquarium Gradients

To showcase the full differentiability of Aquarium, we perform a shape-and-gait co-optimization of an oscillating diamond foil in free-stream conditions using the gradient-based, limited-memory BFGS (L-BFGS) algorithm [69] as shown in Figure 1. Specifically, we aim to maximize the thrust (i.e., minimize drag) generated by the foil and formulate the objective as the integral of the thrust force over an oscillation period. The optimization is performed over a range of chord-wise $Re \in [620, 828]$. This demonstrates Aquarium's ability to be used in optimization in unsteady flow environments. We represent the diamond foil's shape with constant edge lengths $(0.08\,\text{m}, 0.2\,\text{m})$ and an angle parameter α that determines the foil thickness as seen in Figure 6. Also seen in Figure 6, the gait is determined by the heave amplitude A, pitch angle amplitude B, and relative pitch-heave phase ϕ. To avoid degenerate cases, we impose box constraints on the decision variables with $6^\circ \leq \alpha \leq 23^\circ$, $0.05\,\text{m} \leq A \leq 0.15\,\text{m}$, $0^\circ \leq B \leq 60^\circ$, and $0^\circ \leq \phi \leq 180^\circ$.

Using gradients provided by Aquarium, L-BFGS achieves a 139% improvement in thrust after converging to a thrust-generating solution from a drag-inducing initial geometry and gait as seen in Figure 1b. Specifically, the foil thickness is minimized ($\alpha = 6^\circ$) and the heave amplitude is maximized ($A = 0.15\,\text{m}$) with a converged pitch and phase of $B \approx 27^\circ$ and $\phi \approx 74^\circ$ as seen in Figure 1a. Despite the differences in Re regime and overall foil shape, there is still good agreement between this optimized gait profile to the empirical findings from Anderson et. al. [16]. In addition, L-BFGS was able to quickly converge to the optimized solution in only 7 iterations as seen in Figure 7, showcasing the sample-efficiency achieved by gradient-based optimization with Aquarium.

The entire optimization process had a total runtime of 6 hours on a 64-core AMD Threadripper CPU. Each simulation rollout is performed over 100 time steps over a 300×300 fluid grid, with the KKT system in (12) and (17) solved using MKL Pardiso [77]. Aquarium is currently not optimized for computational performance, and we expect an order of magnitude improvement by solving the KKT system with a specialized sparse linear solver and efficiently implementing the adjoint method.

V. CONCLUSIONS

We have presented Aquarium, a fully differentiable fluid-structure interaction simulator that provides full, analytical gradients while accurately simulating coupled fluid and rigid-body dynamics in 2D. Aquarium improves on existing fluid simulators by offering three key features: 1) full differentiability with analytical gradients, which enables optimization in unsteady flow; 2) accurate and stable modeling of fluid dynamics by applying fully implicit integration to the full Navier-Stokes equations; and 3) explicitly formulated fluid-structure interaction that couples fluid physics with rigid-body dynamics. Aquarium enables a variety of optimization tasks — including gait optimization, reinforcement learning, and hardware-controller co-design — suited to robotics applications, where efficient locomotion may need to consider detailed flow physics both steady and unsteady.

In future work, we plan to address the current limitations of Aquarium, which include the lack of 3D simulation and soft bodies. Doing so will improve sim-to-real transfer over a wide-range of robotic systems while offering optimization over 3D geometries and gaits. We also plan to improve Aquarium's computational efficiency by implementing the adjoint method and a specialized sparse linear solver, making Aquarium more suitable for 3D simulation and high-dimensional optimization tasks such as the training of deep-learning models.

REFERENCES

[1] A. P. Maertens, A. Gao, and M. S. Triantafyllou, "Optimal undulatory swimming for a single fish-like body and for a pair of interacting swimmers," *Journal of Fluid Mechanics*, vol. 813, pp. 301–345, 2017. Publisher: Cambridge University Press.

[2] A. Gao and M. S. Triantafyllou, "Independent caudal fin actuation enables high energy extraction and control in two-dimensional fish-like group swimming," *Journal of Fluid Mechanics*, vol. 850, pp. 304–335, 2018. Publisher: Cambridge University Press.

[3] 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*, vol. 12, no. 3, p. 036001, 2017. Publisher: IOP Publishing.

[4] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," *Science Robotics*, vol. 3, no. 16, p. eaar3449, 2018.

[5] J. G. Miles and N. A. Battista, "Don't be jelly: Exploring effective jellyfish locomotion," *arXiv preprint arXiv:1904.09340*, 2019.

[6] Y. Jiao, F. Ling, S. Heydari, N. Heess, J. Merel, and E. Kanso, "Learning to swim in potential flow," *Physical Review Fluids*, vol. 6, May 2021. Publisher: American Physical Society (APS).

[7] M. F. Platzer, K. D. Jones, J. Young, and J. C. Lai, "Flapping wing aerodynamics: progress and challenges," *AIAA journal*, vol. 46, no. 9, pp. 2136–2149, 2008.

[8] W. Shyy, H. Aono, S. K. Chimakurthi, P. Trizila, C.-K. Kang, C. E. Cesnik, and H. Liu, "Recent progress in flapping wing aerodynamics and aeroelasticity," *Progress in Aerospace Sciences*, vol. 46, no. 7, pp. 284–327, 2010.

[9] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019. Publisher: IEEE.

[10] A. Appius, E. Bauer, M. Blöchlinger, A. Kalra, R. Oberson, A. Raayatsanati, P. Strauch, S. Suresh, M. von Salis, and R. K. Katzschmann, "RAPTOR: Rapid Aerial Pickup and Transport of Objects by Robots," *arXiv preprint arXiv:2203.03018*, 2022.

[11] M. Triantafyllou and G. Triantafyllou, "An Efficient Swimming Machine," *Scientific American - SCI AMER*, vol. 272, pp. 64–70, Mar. 1995.

[12] M. S. Triantafyllou, G. S. Triantafyllou, and D. K. P. Yue, "Hydrodynamics of Fishlike Swimming," *Annual Review of Fluid Mechanics*, vol. 32, no. 1, pp. 33–53, 2000.

[13] W. Shyy and H. Liu, "Flapping Wings and Aerodynamic Lift: The Role of Leading-Edge Vortices," *AIAA Journal*, vol. 45, no. 12, pp. 2817–2819, 2007.

[14] Z. R. Manchester, J. I. Lipton, R. J. Wood, and S. Kuindersma, "A variable forward-sweep wing design for enhanced perching in micro aerial vehicles," in *55th AIAA Aerospace Sciences Meeting*, p. 0011, 2017.

[15] J. Moore, R. Cory, and R. Tedrake, "Robust post-stall perching with a simple fixed-wing glider using LQR-Trees," *Bioinspiration & Biomimetics*, vol. 9, p. 025013, May 2014.

[16] J. M. ANDERSON, K. STREITLIEN, D. S. BARRETT, and M. S. TRIANTAFYLLOU, "Oscillating foils of high propulsive efficiency," *Journal of Fluid Mechanics*, vol. 360, pp. 41–72, 1998.

[17] J. M. Anderson and N. K. Chhabra, "Maneuvering and stability performance of a robotic tuna.," *Integrative and comparative biology*, vol. 42, pp. 118–126, Feb. 2002.

[18] C. Christianson, Y. Cui, M. Ishida, X. Bi, Q. Zhu, G. Pawlak, and M. Tolley, "Cephalopod-inspired robot capable of cyclic jet propulsion through shape change," *Bioinspiration & biomimetics*, vol. 16, Sept. 2020.

[19] C. A. Aubin, S. Choudhury, R. Jerch, L. A. Archer, J. H. Pikul, and R. F. Shepherd, "Electrolytic vascular systems for energy-dense robots," *Nature*, vol. 571, pp. 51–57, July 2019.

[20] Y. Chen, H. Wang, E. F. Helbling, N. T. Jafferis, R. Zufferey, A. Ong, K. Ma, N. Gravish, P. Chirarattananon, M. Kovac, and R. J. Wood, "A biologically inspired, flapping-wing, hybrid aerial-aquatic microrobot," *Science Robotics*, vol. 2, no. 11, p. eaao5619, 2017.

[21] N. T. Jafferis, E. F. Helbling, M. Karpelson, and R. J. Wood, "Untethered flight of an insect-sized flapping-wing microscale aerial vehicle," *Nature*, vol. 570, pp. 491–495, June 2019.

[22] E. Farrell Helbling and R. J. Wood, "A review of propulsion, power, and control architectures for insect-scale flapping-wing vehicles," *Applied Mechanics Reviews*, vol. 70, no. 1, 2018.

[23] J. Z. Zhang, Y. Zhang, P. Ma, E. Nava, T. Du, P. Arm, W. Matusik, and R. K. Katzschmann, "Learning Material Parameters and Hydrodynamics of Soft Robotic Fish via Differentiable Simulation," *arXiv preprint arXiv:2109.14855*, 2021.

[24] P. Ma, T. Du, J. Z. Zhang, K. Wu, A. Spielberg, R. K. Katzschmann, and W. Matusik, "Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–14, 2021. Publisher: ACM New York, NY, USA.

[25] W. M. van Rees, M. Gazzola, and P. Koumoutsakos, "Optimal morphokinematics for undulatory swimmers at intermediate Reynolds numbers," *Journal of Fluid Mechanics*, vol. 775, pp. 178–188, 2015. Publisher: Cambridge University Press.

[26] E. Nava, J. Z. Zhang, M. Y. Michelis, T. Du, P. Ma, B. F. Grewe, W. Matusik, and R. K. Katzschmann, "Fast Aquatic Swimmer Optimization with Differentiable Projective Dynamics and Neural Network Hydrodynamic Models," in *International Conference on Machine Learning*, pp. 16413–16427, PMLR, 2022.

[27] D. N. BEAL, F. S. HOVER, M. S. TRIANTAFYLLOU, J. C. LIAO, and G. V. LAUDER, "Passive propulsion in vortex wakes," *Journal of Fluid Mechanics*, vol. 549, pp. 385–402, 2006.

[28] T. A. Howell, S. Le Cleac', J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A Differentiable Simulator for Robotics," 2022.

[29] R. Tedrake and t. D. D. Team, "Drake: Model-based design and verification for robotics," 2019.

[30] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[31] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "NeuralSim: Augmenting Differentiable Simulators with Neural Networks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[32] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax–A Differentiable Physics Engine for Large Scale Rigid Body Simulation," *arXiv preprint arXiv:2106.13281*, 2021.

[33] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: Analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[34] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics for articulated rigid bodies with contact," *arXiv preprint arXiv:2103.16021*, 2021.

[35] J. Degrave, M. Hermans, J. Dambre, and others, "A differentiable physics engine for deep learning in robotics," *Frontiers in neurorobotics*, p. 6, 2019.

[36] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus, and W. Matusik, "Diffpd: Differentiable projective dynamics," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 2, pp. 1–21, 2021. Publisher: ACM New York, NY.

[37] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International conference on robotics and automation (ICRA)*, pp. 6265–6271, IEEE, 2019.

[38] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, pp. 136–145, PMLR, 2017.

[39] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[40] Y. Chen, A. L. Desbiens, and R. J. Wood, "A computational tool to improve flapping efficiency of robotic insects," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1733–1740, 2014.

[41] N. A. Battista, W. C. Strickland, and L. A. Miller, "IB2d: a Python and MATLAB implementation of the immersed boundary method," *Bioinspiration & biomimetics*, vol. 12, no. 3, p. 036003, 2017.

[42] C. Bernier, M. Gazzola, R. Ronsse, and P. Chatelain, "Simulations of propelling and energy harvesting articulated bodies via vortex particle-mesh methods," *Journal of Computational Physics*, vol. 392, pp. 34–55, 2019.

[43] H. Jasak, "OpenFOAM: open source CFD in research and industry," *International Journal of Naval Architecture and Ocean Engineering*, vol. 1, pp. 89–94, Dec. 2009.

[44] F. Palacios, T. D. Economon, A. Aranake, S. R. Copeland, A. K. Lonkar, T. W. Lukaczyk, D. E. Manosalvas, K. R. Naik, S. Padron,

B. Tracey, and others, "Stanford university unstructured (SU2): Analysis and design technology for turbulent flows," in *52nd Aerospace Sciences Meeting*, p. 0243, 2014.

[45] U. Manual, "ANSYS FLUENT 12.0," *Theory Guide*, 2009.

[46] Siemens Digital Industries Software, "Simcenter STAR-CCM+ User Guide v. 2021.1," 2021.

[47] P. Holl, V. Koltun, K. Um, and N. Thuerey, "phiflow: A differentiable pde solving framework for deep learning via physical simulations," in *NeurIPS Workshop*, vol. 2, 2020.

[48] D. A. Bezgin, A. B. Buhendwa, and N. A. Adams, "JAX-FLUIDS: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows," *arXiv preprint arXiv:2203.13760*, 2022.

[49] T. Du, K. Wu, A. Spielberg, W. Matusik, B. Zhu, and E. Sifakis, "Functional optimization of fluidic devices with differentiable stokes flow," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020. Publisher: ACM New York, NY, USA.

[50] W. Liu, K. Bai, X. He, S. Song, C. Zheng, and X. Liu, "FishGym: A High-Performance Physics-based Simulation Framework for Underwater Robot Learning," *arXiv preprint arXiv:2206.01683*, 2022.

[51] C. S. Peskin, "The immersed boundary method," *Acta numerica*, vol. 11, pp. 479–517, 2002.

[52] K. Taira and T. Colonius, "The immersed boundary method: A projection approach," *Journal of Computational Physics*, vol. 225, no. 2, pp. 2118–2137, 2007.

[53] J. B. Perot, "An Analysis of the Fractional Step Method," *Journal of Computational Physics*, vol. 108, no. 1, pp. 51–58, 1993.

[54] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and others, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[56] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015.

[57] A. Jameson, "Aerodynamic design via control theory," *Journal of Scientific Computing*, vol. 3, pp. 233–260, Sept. 1988.

[58] G. K. Kenway, C. A. Mader, P. He, and J. R. Martins, "Effective adjoint approaches for computational fluid dynamics," *Progress in Aerospace Sciences*, vol. 110, p. 100542, 2019. Publisher: Elsevier.

[59] Y. Bazilevs, M.-C. Hsu, and M. T. Bement, "Adjoint-based Control of Fluid-Structure Interaction for Computational Steering Applications," *Procedia Computer Science*, vol. 18, pp. 1989–1998, 2013.

[60] M. P. Rumpfkeil and D. W. Zingg, "The optimal control of unsteady flows with a discrete adjoint method," *Optimization and Engineering*, vol. 11, pp. 5–22, 2010. Publisher: Springer.

[61] X. An, D. Floryan, and C. Rowley, "Optimal Gaits of Fish-like Swimming," Jan. 2021.

[62] J. Grover, J. Zimmer, T. Dear, M. Travers, H. Choset, and S. D. Kelly, "Geometric Motion Planning for a Three-Link Swimmer in a Three-Dimensional low Reynolds-Number Regime," in *2018 Annual American Control Conference (ACC)*, pp. 6067–6074, 2018.

[63] J. Grover and D. Vedova, "Motion planning, design optimization and fabrication of ferromagnetic swimmers," in *Robotics science and systems*, 2019.

[64] X. Tu and D. Terzopoulos, "Artificial fishes: Physics, locomotion, perception, behavior," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 43–50, 1994.

[65] S. Min, J. Won, S. Lee, J. Park, and J. Lee, "Softcon: Simulation and control of soft-bodied animals with biomimetic actuators," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–12, 2019. Publisher: ACM New York, NY, USA.

[66] M.-C. Lai and C. S. Peskin, "An immersed boundary method with formal second-order accuracy and reduced numerical viscosity," *Journal of computational Physics*, vol. 160, no. 2, pp. 705–719, 2000. Publisher: Elsevier.

[67] J. Kim and P. Moin, "Application of a fractional-step method to incompressible Navier-Stokes equations," *Journal of computational physics*, vol. 59, no. 2, pp. 308–323, 1985. Publisher: Elsevier.

[68] S. V. Patankar and D. B. Spalding, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," in *Numerical prediction of flow, heat transfer, turbulence and combustion*, pp. 54–73, Elsevier, 1983.

[69] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, second ed., 2006.

[70] S. Barratt, "On the differentiability of the solution to convex optimization problems," *arXiv preprint arXiv:1804.05098*, 2018.

[71] U. Dini, *Lezioni di analisi infinitesimale*, vol. 1. Fratelli Nistri, 1907.

[72] Y. Zhang and R. K. Katzschmann, "Creation of a Modular Soft Robotic Fish Testing Platform," *arXiv preprint arXiv:2201.04098*, 2022.

[73] D. J. Tritton, "Experiments on the flow past a circular cylinder at low Reynolds numbers," *Journal of Fluid Mechanics*, vol. 6, no. 4, pp. 547–567, 1959. Publisher: Cambridge University Press.

[74] W. W. Ren, J. Wu, C. Shu, and W. M. Yang, "A stream function–vorticity formulation-based immersed boundary method and its applications," *International Journal for Numerical Methods in Fluids*, vol. 70, no. 5, pp. 627–645, 2012. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.2705.

[75] M. Braza, P. Chassaing, and H. H. Minh, "Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder," *Journal of Fluid Mechanics*, vol. 165, pp. 79–130, 1986. Publisher: Cambridge University Press.

[76] A. Lukezic, T. Vojir, L. ˇCehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6309–6318, 2017.

[77] O. Schenk, K. Gärtner, W. Fichtner, and A. Stricker, "PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation," *Future Generation Computer Systems*, vol. 18, no. 1, pp. 69–78, 2001.