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

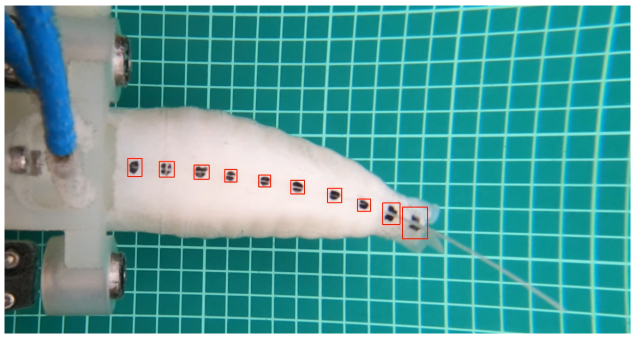
Jeong Hun Lee[1], Mike Y. Michelis[2], Robert Katzschmann[2], and Zachary Manchester[1]

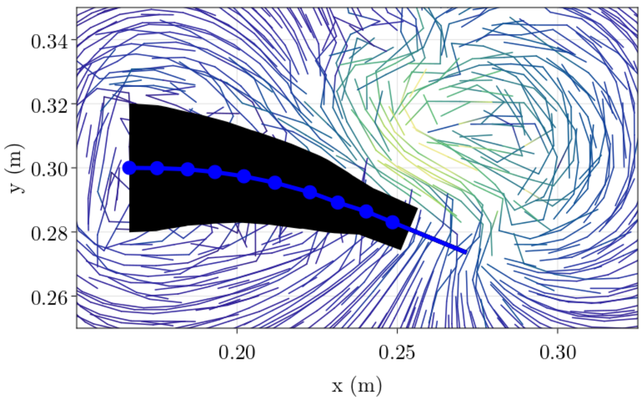
***Abstract*— We present Aquarium, a differentiable fluid-structure interaction solver for robotics that offers stable simulation, accurate coupled robot-fluid physics, and full differentiability with respect to fluid states, robot states, and shape parameters. Aquarium achieves stable simulation with accurate flow physics by integrating over the discrete, incompressible Navier-Stokes equations directly using a fully-implicit Crank-Nicolson scheme with a second-order finite-volume spatial discretization. The robot and fluid 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. The derivatives can then be passed to a gradient-based optimization or learning framework. We demonstrate Aquarium's ability to accurately simulate coupled fluid-solid physics with numerous examples, including a cylinder in free stream and a soft robotic tail with hardware validation. We also demonstrate Aquarium's ability to provide full, analytical gradients by performing both shape and gait optimization of a robotic fish tail to maximize generated thrust.**

I. INTRODUCTION

In recent years, there has been considerable interest in bio-inspired locomotion for underwater [1], [2], [3] and aerial vehicles [4], [5], [6], [7], which involve complex interactions with the fluid environment to achieve impressive performance. For example, the energy efficiency and high maneuverability of fish-like propulsion is attributed to the sensing and shedding of vortices [8]. This complex fluid interaction is also present in the flight of flapping-wing aerial vehicles [9] and high-angle-of-attack perching maneuvers of fixed-wing airplanes [10], [11], and has led to the development of various robotic hardware systems, including: oscillating-foil propulsion for marine vehicles [12], various biomimetic underwater systems [2], [13], [14], [15], and flapping-wing, micro-aerial vehicles [16], [17], [18].

In contrast, there has been comparatively little work in optimizing *both* the systems' design and control parameters [19], [20] while reasoning about the fluid environment [21]. Such an optimization could provide insight into the passive swimming behaviour of a dead trout [22], 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) Hardware experiment



(b) Aquarium simulation with streamlines

Fig. 1: Aquarium simulation of fluid-structure interaction of a soft robotic fish tail with hardware validation. The tracked markers, modeled as joints in the multi-link, simulated representation, are boxed in red with corresponding center-line links and joints in simulation shown in blue.

In recent years, various differentiable simulators [23], [24], [25], [26], [27], [28], [29], [30] have been deployed with a wide range of uses in robotics, from reinforcement learning to model-predictive control. With the added feature of differentiability, these solvers are able to provide gradients that can be passed directly to gradient-based optimization and control frameworks. Differentiable simulators can also be integrated into neural networks during backpropogation to achieve high sample efficiency and enable the use in gradient-based controllers [20], [31], [32].

In the fluids community, various computational fluid dynamics (CFD) solvers also exist [21], [33], [34], [35], [36], [37], [38], [39], [40]. However, these solvers suffer

*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

from various deficiencies for use in robotics, including: poor accuracy, generalizability, computational efficiency, and stability over larger time-steps (fully-implicit integration); lack of full differentiability for unsteady flow; and inability to handle fluid-structure interaction (FSI) to properly simulate unsteady, multi-physics coupling between the fluid and solid (*e.g.*, robot) dynamics.

We propose **Aquarium**, an *open-source, physics-based, fully differentiable* solver for simulating the coupled fluid-structure interactions between robotic systems and the surrounding fluid. The 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-solid coupling is achieved using the immersed boundary method [41], which separates the fluid and solid mesh to avoid computationally-expensive re-meshing. Specifically, we build upon the work of Taira et. al. [42] and Perot [43] to pose the FSI as an optimization problem, with the fluid-solid coupling acting as an equality constraint to satisfy the no-slip boundary condition. Analytical gradients are computed by applying the implicit function theorem directly to the fluid-structure interaction problem. Aquarium is currently implemented in 2D, but the methodology is generalizable to 3D to capture the complete fluid-solid-coupled physics. In summary, our contribution is a solver for robotics applications with the following features:

- A simulator that solves the discretized Navier-Stokes equations directly with multi-physics coupling between the robot and 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 both fluid and robot 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 still water. We then showcase the differentiability of Aquarium by performing gradient-based optimization of gait and shape. 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 like OpenFOAM [37] and ANSYS [40] provide accurate baseline results, but at high computational costs. Consequently, running parameter sweeps for gradient-free optimization can quickly become prohibitively expensive, especially for higher-dimensional problems.

Gradient-free approaches tend to be less stable and converge much slower than their gradient-based counterparts [19], [29]; however, most conventional simulation software does not provide gradients *w.r.t.* optimization parameters. The class of differentiable simulators addresses this shortcoming. Though many differentiable solvers were developed in fields other than CFD [24], [25], [26], [27], [28], [29], [30], there are also several works aimed towards the fluids community. PhiFlow [36] was developed as a differentiable PDE solver for deep-learning, written in frameworks that allow for automatic differentiation, such as JAX, PyTorch, and TensorFlow. 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 [35] developed a level-set method for differentiable fluid simulations in JAX. The work shows various implemented boundary conditions, including immersed boundaries. Therefore, FSI can be supported, but has only been extended to static rigid bodies and fluid-fluid interaction, and not yet deformable, moving, or articulated bodies [35].

B. Adjoint Methods

While not usually phrased as differentiable simulators, adjoint methods have been used to efficiently provide gradients within various CFD frameworks [37], [39], [40], [44], particularly for those that use iterative solvers. However, all of these methods are limited to shape optimization, primarily in steady flow, and extensions to, *e.g.*, gait optimization would require significantly more effort. For example, SU2 [39] provides topology optimization capabilities by computing gradients of shape surfaces using continuous and discrete adjoint methods. Conventional methods for FSI topology optimization specifically have used sensitivity analysis [39], [45], [46], which are quite restrictive in that they do not account for any other form of optimization besides topology, *e.g.*, control/actuation optimization, which are critical for the robotics community.

C. Fluid-Structure Interaction for Optimization

Out of the previous methods, aside from the traditional CFD solvers, only SU2 and JAX-FLUIDS [35] support fluid-structure interaction capabilities. Additionally, in the latter case, gradients are computed using automatic differentiation, which can be computationally expensive compared to computing analytic gradients with direct linear algebra. In addition, combining FSI with optimization continues to be a challenge. Previous work [47] has simplified the problem to modeling the fluid as a differentiable Stokes problem, and hence only doing steady-state optimization. More recent methods leverage neural-network surrogate models for fluids to speed up the differentiable FSI pipeline for optimization tasks [21] and show how it can be applied to actuation optimization, yet no guarantees can be given on the neural-network surrogate for the fluid simulation, hence it might not generalize to unknown scenarios.

Finally, Liu et al. [38] implement an FSI extension based on the OpenAI Gym environment. Though not differentiable, their fluid solver implementation, a GPU-optimized lattice-Boltzmann method, is highly efficient and can be integrated in an RL training pipeline. The solid skeleton is modeled as an articulated rigid body, and two-way coupling is achieved through an immersed boundary method. The advantages of having full fluid solvers as opposed to empirical approximations for hydrodynamics [20], [48], [49] are shown in several applications, such as leveraging Kármán vortices that are formed for faster forward propulsion of the robots.

III. DIFFERENTIABLE FLUID-STRUCTURE INTERACTION

Computationally modeling fluid dynamics and fluid-structure interaction using the Navier-Stokes equations has been extensively studied [41], [42], [43], [50], [51], [52]. Rather than describing the methods in detail, we highlight the key concepts and refer the reader to existing literature on CFD and FSI. Specifically, our work is most closely related to that of Taira et. al [42] and Perot [43].

A. *Implicit Fluids 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 solid in free-stream).

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

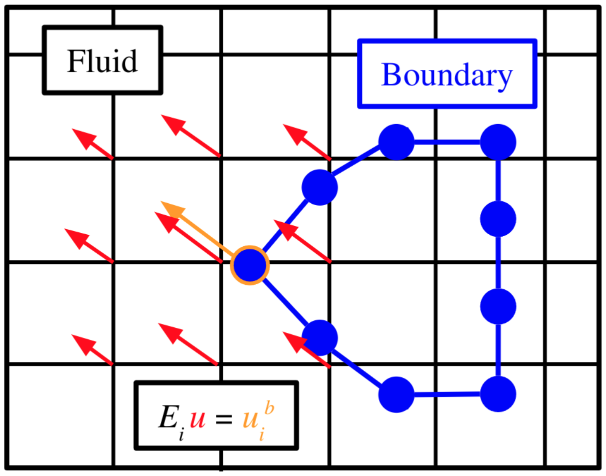


Fig. 2: Immersed boundary meshes where the fluid domain is represented by a fixed Eulerian grid, and the boundary of the solid body is represented by a moving Lagrangian mesh (blue). The meshes are coupled by a convolution matrix E that maps fluid-cell values (red) to the boundary nodes (orange).

equations,

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

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

where $u_{k+1} = [u_x, u_y]^T_{k+1} \in \mathbb{R}^{n_u}$, $p_{k+1} \in \mathbb{R}^{n_f}$, and A and r 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 $-G^T = D$ [42]. This means that 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 fluid-structure interaction, 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 solid at the boundary. To do so, we use the immersed boundary method [41], which separates the fluid (u) and solid (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 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 boundary, such as shape parameters and the boundary state, x_{k+1}. We then apply (8) to our Navier-Stokes formulation as an additional constraint, providing us with the full fluid-structure interaction 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 [53]. 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 $f^b_{k+1} \in \mathbb{R}^{n_f}$ are the pressure forces acting along the boundary, ρ is the density of the fluid, h_x, h_y are the spatial step sizes of the fluid grid discretization, and s is the step size of the boundary discretization. Values of f^b_{k+1} can then be integrated along the surface of the solid body to obtain net forces.

C. *Extraction of Simulation Gradients*

We exploit the structure of the KKT system defined by (12) to calculate analytic Jacobians of our FSI model [54]. For clarity, we first start by looking at our system at time t_k and group (9)–(11) to simplify the FSI 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 [55], 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

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

where D'_g is the same KKT system that appears in (12) and $\frac{\delta u_k}{\delta \theta}$ has already been computed at the previous time step t_k. 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 for to optimize shapes, gaits, trajectories, controls, etc.

IV. EXPERIMENTAL RESULTS

This section presents the results of several simulation experiments to evaluate the fluid-structure interaction 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 tail. We also demonstrate the full differentiability of the simulator in gradient-based optimization examples, including both shape and gait optimization. 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 steady-state flow around a cylinder under 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 of the fluid grid, 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 far away from the cylinder. The simulated fluid environment is set to have dimensions 20 times the cylinder diameter to ensure the boundary conditions are sufficiently distanced from the cylinder.

To study the generalizability of the simulator to varying Reynolds numbers (ratio of nonlinear convective forces in the

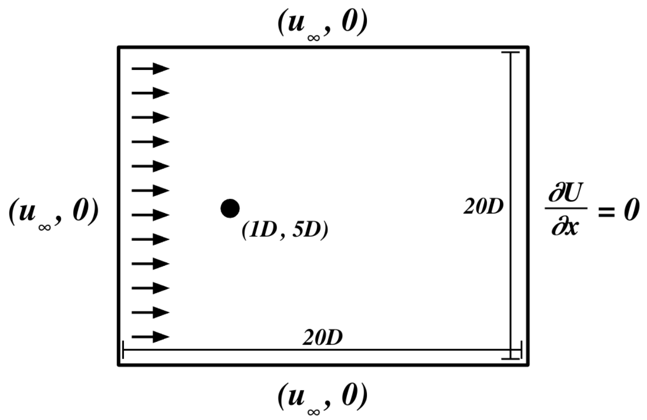
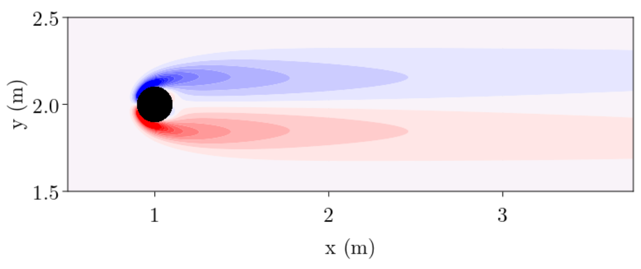
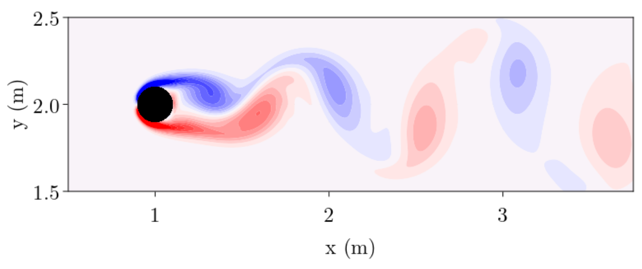


Fig. 3: Cylinder-in-free-stream simulation setup with inflow and outflow boundary conditions defined for left and right boundaries. Far-field boundary conditions are defined for the top and bottom boundaries.



(a) $Re = 40$



(b) $Re = 100$

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

fluid), we evaluate the resulting steady-state behaviors under various free-stream velocity conditions. As seen in Figures 4a and 4b, the Aquarium simulator is able to capture both the steady-state vortex pairs at low Reynolds numbers ($Re = 40$), and the Kármán vortex street that occurs at higher Reynolds numbers ($Re = 100$).

To study the flow-induced forces on the cylinder body, 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 especially true for the non-dimensional Strouhal number, which characterizes periodic vortex shedding in the wake, and is especially important for studying bio-inspired swimming [1].

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

	Drag Coeff.
Tritton [56] (*empirical*)	1.65
Taira et. al [42]	1.55
Ren et. al [57]	1.57
Aquarium	1.76

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

	Drag Coeff.	**Lift Coeff.**	**Strouhal #**
Braza et. al [58]	1.325 ± 0.008	± 0.280	0.164
Ren et. al [57]	1.335 ± 0.011	± 0.356	0.164
Aquarium	1.496 ± 0.010	± 0.368	0.167

B. Soft Robotic Fish Tail

To demonstrate Aquarium's ability to generalize beyond simple and still geometry in steady-state flow, we also simulate the periodic flapping of a soft robotic fish tail, and validate it against a hardware experiment as shown in Figure 1a. The soft robotic fish tail is fabricated as described in [19], [59], 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 as the tail is actuated in initially still water. We use previously collected video and

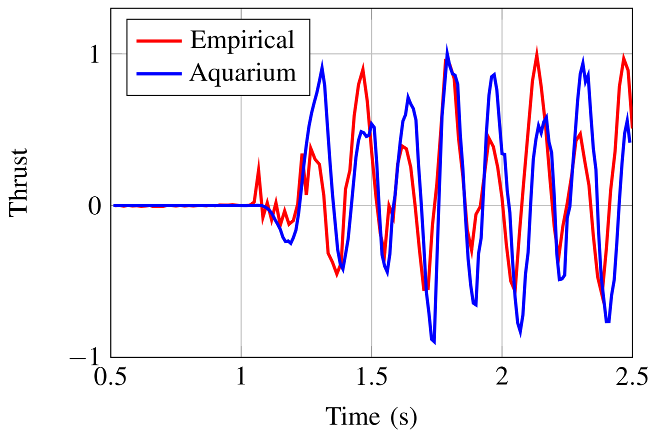


Fig. 5: Time history of normalized thrust force for both hardware (red) and simulation (blue) results.

force measurement data from [19].

In simulation, we approximate the 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 body, as shown in Figure 1b. The body profiles of each link are approximated by a linear interpolation between the respective joint widths with the fin represented as a 1D link. The simulated motion is prescribed using a forward-kinematics model, where the joint angles are determined using CSRT [60] marker tracking from a pre-recorded video of the hardware experiment. The fluid environment is also modeled to recreate the hardware experiment: a $0.6\,\mathrm{m} \times 0.6\,\mathrm{m}$ cavity with wall-like boundary conditions ($u_\infty = 0$) with no initial fluid velocity. The fluid is simulated with properties of water, with a density of $\rho = 997\,\mathrm{kg\,m^{-3}}$ and dynamic viscosity of $\mu = 8.9 \times 10^{-4}\,\mathrm{Pa\,s}$. Simulated boundary pressure forces, $\tilde{f}^b$, are integrated over the simulated fish-tail 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.

As seen in Figure 5, there is good agreement between the phase and frequency of the normalized thrust forces w.r.t 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

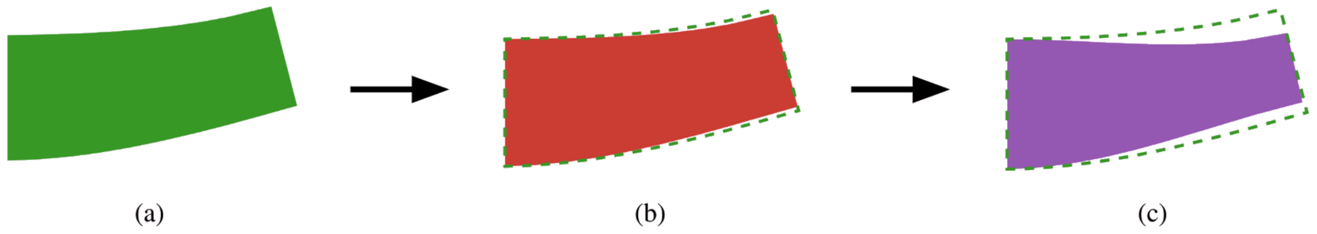


Fig. 6: Optimized fish-tail profiles for maximizing thrust, where (a) is the original geometry, (b) is the result of shape optimization, and (c) is the result of shape-and-gait co-optimization. Displayed tail configurations correspond to those seen during flapping. Results are obtained from running the BFGS algorithm using Aquarium gradients.

additional 3D flow and shape effects (i.e. contour of the fish-tail and fin in 3D) 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 fabrication error of the hardware as well as the geometric and kinematic approximations of the multi-link representation.

C. Optimization using Flow Gradients

To showcase the full differentiability of the Aquarium simulator, we perform both a shape and shape-and-gait co-optimization on the fish tail using the gradient-based BFGS algorithm [53]. Specifically, we aim to maximize the net thrust force (i.e. minimize drag force) generated over time and formulate the objective function as the discrete-time integral,

$$\min_{\theta} \quad -\sum_{k=1}^{N} q^T f_k^b \Delta t, \tag{18}$$

where $q = [s, 0] \in \mathbb{R}^{n_b}$ acts as an operator to integrate f_k^b over the boundary. The optimization is performed over one period of tail flapping under initially free-stream conditions of $Re = 320$ to demonstrate the simulator's ability to optimize over unsteady flow conditions. For shape optimization, we represent the fish-tail profile using a cubic polynomial and optimize over the respective coefficients. We then optimize over both the polynomial coefficients and the frequency of tail flapping to showcase the ability to perform shape-and-gait co-optimization.

Using gradients from Aquarium, BFGS is able to improve on the initial geometry to maximize thrust over unsteady flow conditions as shown in Table III with corresponding shapes in Figure 6. While shape optimization alone is able to improve thrust by 15%, the ability to also optimize over the gait results in a 60% improvement. This demonstrates the significant advantage of Aquarium's full differentiability, where additional gradient information can result in large performance improvements for robotics tasks.

TABLE III: BFGS Optimization Results

	Loss	**% Improvement**
Initial geometry	0.0225	-
Shape optimization	0.0176	15
Shape-and-gait co-optimization	0.0088	60

V. CONCLUSIONS

We have presented Aquarium, a fully differentiable fluid-structure simulator that provides full, analytical gradients while accurately simulating coupled fluid-robot physics. Aquarium improves on existing fluid simulators by offering three key features: 1) full differentiability with analytical gradients, which enables optimization beyond shape alone; 2) accurate and stable modeling of fluid dynamics by applying implicit integration to the full Navier-Stokes equations; and 3) explicitly formulated fluid-structure interaction that couples fluid physics with rigid-body robot 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 under varying conditions, both steady and unsteady. In future work, we plan to extend Aquarium to full 3D flow to improve sim-to-real transfer and better optimize 3D robot geometry.

REFERENCES

[1] 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. _eprint: https://doi.org/10.1146/annurev.fluid.32.1.33.

[2] 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. _eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.aar3449.

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

[4] 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. Publisher: Elsevier.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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. _eprint: https://doi.org/10.2514/1.33205.

[10] 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.

[11] 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.

[12] 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. Publisher: Cambridge University Press.

[13] 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. Place: England.

[14] 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.

[15] 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.

[16] 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. _eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.aao5619.

[17] 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.

[18] 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. Publisher: American Society of Mechanical Engineers Digital Collection.

[19] 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.

[20] 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.

[21] 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.

[22] 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. Publisher: Cambridge University Press.

[23] T. A. Howell, S. Le Cleac’, J. Z. Kolter, M. Schwager, and Z. Manchester, “Dojo: A Differentiable Simulator for Robotics,” 2022.

[24] 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.

[25] 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. Publisher: ACM New York, NY, USA.

[26] 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.

[27] 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.

[28] J. Degrave, M. Hermans, J. Dambre, and others, “A differentiable physics engine for deep learning in robotics,” *Frontiers in neurorobotics*, p. 6, 2019. Publisher: Frontiers.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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. Publisher: IOP Publishing.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] P. Kohnke, “Ansys,” in *Finite Element Systems*, pp. 19–25, Springer, 1982.

[41] C. S. Peskin, “The immersed boundary method,” *Acta numerica*, vol. 11, pp. 479–517, 2002. Publisher: Cambridge University Press.

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

[43] J. Perot, “An analysis of the fractional step method,” *Journal of Computational Physics*, vol. 108, no. 1, pp. 51–58, 1993.

[44] 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.

[45] C. S. Andreasen and O. Sigmund, “Topology optimization of fluid–structure-interaction problems in poroelasticity,” *Computer Methods in Applied Mechanics and Engineering*, vol. 258, pp. 55–62, 2013.

[46] W. J. P. Casas and R. Pavanello, “Optimization of fluid-structure systems by eigenvalues gap separation with sensitivity analysis,” *Applied Mathematical Modelling*, vol. 42, pp. 269–289, 2017.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

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

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

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

[56] D. J. Tritton, "Experiments on the flow past a circular cylinder at low reynolds numbers," *Journal of Fluid Mechanics*, vol. 6, no. 4, p. 547–567, 1959.

[57] 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.

[58] 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, p. 79–130, 1986.

[59] Y. Zhang and R. K. Katzschmann, "Creation of a modular soft robotic fish testing platform," *arXiv preprint arXiv:2201.04098*, 2022.

[60] 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.