Bulletin of Mathematical Biology (2022) 84:72
https://doi.org/10.1007/s11538-022-01027-1

Society for Mathematical Biology

ORIGINAL ARTICLE

Planktos: An Agent-Based Modeling Framework for Small Organism Movement and Dispersal in a Fluid Environment with Immersed Structures

W. C. Strickland[1] · N. A. Battista[2] · C. L. Hamlet[3] · L. A. Miller[4]

Received: 1 November 2021 / Accepted: 6 May 2022 / Published online: 10 June 2022

Abstract

Multiscale modeling of marine and aerial plankton has traditionally been difficult to address holistically due to the challenge of resolving individual locomotion dynamics while being carried with larger-scale flows. However, such problems are of paramount importance, e.g., dispersal of marine larval plankton is critical for the health of coral reefs, and aerial plankton (tiny arthropods) can be used as effective agricultural bio-control agents. Here we introduce the open-source, agent-based modeling software *Planktos* targeted at 2D and 3D fluid environments in Python. Agents in this modeling framework are relatively tiny organisms in sufficiently low densities that their effect on the surrounding fluid motion can be considered negligible. This library can be used for scientific exploration and quantification of collective and emergent behavior, including interaction with immersed structures. In this paper, we detail the implementation and functionality of the library along with some illustrative examples. Functionality includes arbitrary agent behavior obeying either ordinary differential equations, stochastic differential equations, or coded movement algorithms, all under

✉ W. C. Strickland
cstric12@utk.edu

N. A. Battista
battistn@tcnj.edu

C. L. Hamlet
ch051@bucknell.edu

L. A. Miller
lauram9@math.arizona.edu

1 Department of Mathematics, University of Tennessee, Knoxville, 227 Ayres Hall, Knoxville, TN 37996-1320, USA

2 Department of Mathematics and Statistics, The College of New Jersey, Ewing Township, NJ 08628, USA

3 Department of Mathematics, Bucknell University, Lewisburg, PA 17837, USA

4 Department of Mathematics, University of Arizona, 617 N. Santa Rita Ave., Tuscon, AZ 85721-0089, USA

the influence of time-dependent fluid velocity fields generated by computational fluid dynamics, experiments, or analytical models in domains with static immersed mesh structures with sliding or sticky collisions. In addition, data visualization tools provide images or animations with kernel density estimation and velocity field analysis with respect to deterministic agent behavior via the finite-time Lyapunov exponent.

Keywords Agent-based models · Multi-scale modeling · Plankton · Fluid–structure interaction · Computational fluid dynamics

1 Introduction

Marine or aquatic plankton consists of phytoplankton, zooplankton, bacteria, viruses, as well as the larvae of many marine invertebrates and fish that inhabit open water. Aerial plankton are tiny arthropods carried by the winds and include agriculturally important pests and biocontrol agents. Both types of plankton are dispersed by local flows but can also bias their distributions by timing their release and actively controlling some aspects of their locomotion during settlement. In terms of the ecology of both marine and aerial plankton, the resulting dispersal patterns can have a significant effect on the probability of mating, the ability to find adequate nutrient sources, genetic diversity, and recovery from disturbance (Cowen and Sponaugle 2009; Dingle 2014; Howe and Miriti 2004; Bohonak and Jenkins 2003; Pepper et al. 2015; Thompson et al. 2018; Kleypas et al. 2016; McManus et al. 2020).

In the air, predictive models of the dispersal of small insects and other organisms whose trajectories are significantly determined by both the local airflow and their own behavior are important for many agricultural applications. High-fidelity dispersal models of pests such as thrips and spider mites could inform the timing and location of the application of pesticides or release of biocontrol agents, and models of the dispersal of the biocontrol agents themselves could allow for more effective release of these organisms. In the simplest case, biocontrol will be unsuccessful if prey can escape control by out-dispersing its predator (Taylor 1990). However, high dispersal alone is not enough to guarantee success (Darrouzet-Nardi et al. 2006). It is recognized that individual behavioral decision rules (e.g., movement tendency, duration of flight, and active settlement) have a strong effect on biocontrol efficacy on a local scale (Potting et al. 2005).

The complex behavior of insects and other aerial plankton motivates the development of more detailed models of their movement. Insects use active behaviors to become entrained in the wind, which can be triggered by either the individual's internal state or environmental conditions. Examples of these behaviors include the raised stance employed by phytoseiid mites to undergo dispersal in response to starvation and temperature, (Johnson and Croft 1976) and how the flight window used by parasitoid wasps depends upon the time of day and the prevailing wind speed (Kristensen et al. 2013; Strickland et al. 2017). Insects can also modify their windborne flight by using behaviors that change their aerodynamic shape or by active flying, such as the selective timing of active flight used by aphids in response to the vertical direction of air currents (Reynolds and Reynolds 2009).

In water, predictive models of small-scale (tens of meters or less) plankton dispersal and settlement, filter feeding, and uptake of symbionts (family Symbiodiniaceae) and sperm can aid in constructing artificial reefs and shed light on the impact of lower plankton densities on organism fitness. Research on the dispersal of these organisms increased rapidly in the last decade of the twentieth century with a focus on mechanisms of transport, mortality rates, larval trajectories, settlement behaviors, and gene flow (Levin 2006). Interactions of these organisms with a flow are complex, and distances that larvae are transported can differ greatly between species with different life histories depending on their duration as plankton (Vance 1973; Treml et al. 2012; Winston 2012). Although currents drive dispersal and connectivity at the scales of a kilometer to thousands of kilometers (Thompson et al. 2018), within sheltered environments such as reefs, seagrass beds, and mangroves, behavior can significantly affect local movement and settlement (Reynolds et al. 2005; Koehl et al. 2007; Grünbaum and Strathmann 2003). The swimming speeds of the larvae of many fish species can dominate the speed of ambient currents so that their resulting dispersal is different than would have been predicted by currents alone (Leis 2007). Other common places where behavior and physical oceanography combine to influence plankton dispersal are aggregations at long-lived ocean fronts, thin layers in the subsurface, and places where eddies and internal tidal bores come together (McManus and Woodson 2012).

Settlement is as important as transport in plankton, with at least some plankton actively choosing settlement sites. For example, larvae of perciform fish often take into account the presence of resident fish when determining where to settle (Leis 2007). Similarly, larvae of reef-building corals settle in response to biological (adult conspecifics) (Baird et al. 2003), chemical (crustose coralline algae) (Morse et al. 1988), and physical (light) (Mundy and Babcock 1998) cues on the reef. Daigle et al. (2014) showed that organisms with similar behavior tended to disperse in similar ways, while groups that were not functionally similar did not.

As a whole, the dispersal of plankton can be mostly active, mostly passive, or a mixture of the two. To quantify this effect, one can use the velocity ratio, $V_r = \frac{u_a}{U}$, where u_a is the organism's average active swimming or flying velocity and U gives the typical background flow speed. This dimensionless number describes the ability of an organism to swim against a flow (McManus and Woodson 2012). For low values of V_r, passive drifting is the predominant dispersal mechanism. For higher values of V_r, active movement becomes important. For intermediate-sized organisms, the interaction between physical transport mechanisms and behavior is complex (McManus and Woodson 2012; Villarino et al. 2018). For example, in larval slugs, long-distance dispersal patterns may be primarily driven by currents, but local settlement on a reef is determined by a complex interaction of active behavior and small-scale local fluid dynamics (Koehl et al. 2007; Hata et al. 2017).

Most work on the fluid dynamics of locomotion and dispersal has considered the cases where either one organism actively locomotes in a quiescent fluid (Hoover and Miller 2015; Miller and Peskin 2004; Tytell et al. 2014; Santhanakrishnan et al. 2018) or many organisms passively drift with the fluid (McKenna and Chalupnicki 2011; O'Brien and Wroblewski 1973). Theoretical, computational, and experimental studies that consider both methods of transport are often either (1) limited to a small number of organisms across small scales (tens of body lengths) (Ramananarivo et al. 2016;

Oza et al. 2019) or (2) focus on fluid dynamic interactions between many microscopic individuals with simplified behavior (Nikora 2010; Gao et al. 2017). Agent-based simulations reveal complex emergent phenomena and collective behavior (Giardina 2008; Couzin 2009; Carturan et al. 2020), but nearly all studies neglect background flows. Recent efforts have also focused on the development of software libraries that will advect particles (e.g., plankton, particles, oil) in a given flow, particularly at the ocean and atmosphere scale (Dagestad et al. 2018). Although this software offers much flexibility for adding additional modules, it is focused on a different scale. In addition, it is not straightforward to include more complex agent behavioral models and interactions. Other studies consider background flows while modeling active locomotion as a simple diffusive process using advection–diffusion models (Broström 2002; McGillicuddy and Franks 2019). Understanding the complex interactions between dynamic flows and organism behavior for navigation, settlement, and dispersal has been a long-standing challenge (Willis (2011)).

To tackle problems concerning organisms that actively locomote in complex flows at intermediate values of V_r, we have developed the *Planktos* agent-based modeling software. Agent-based models (or individual-based models) are models in which individuals are described as autonomous entities which can then interact with each other and/or their environment. They have adaptive behavior, which can include continual adjustments to changes in their environment, the presence and behavior of other agents, or their internal states (Railsback and Grimm (2012)). These models are typically stochastic in nature and are particularly well suited to exploring how simple, single-organism dynamics can give rise to population-level, aggregate phenomena.

Agent-based models (ABMs) have seen widespread use across a variety of ecological contexts with varying levels of complexity (e.g., Huth et al. 2004; D'Orsogna et al. 2006; Liedloff and Cook 2007; Schulze et al. 2017 or see Netlogo references 2018 https://ccl.northwestern.edu/netlogo/references.shtml for a long list of citations). Parsimony is critical in forming an ABM that can inform theory from a mathematical perspective. When properly formulated in the context of laboratory or field observations, they can be powerful tools for forging a connection between key individual-level behaviors and macroscopic group properties (Topaz et al. 2008; Nilsen et al. 2013). This understanding can then be used to form robust mathematical characterizations of population movement.

While various software packages providing comprehensive simulation environments for agent-based modeling exist (Railsback et al. 2006), they tend to be for high-level, general use and not easily adaptable to specific challenges of incorporating fluid–structure interaction data for swimmers in continuous 2D or 3D environments. This is partially due to the packages being written for complex agent behavior and interactions without the infrastructure to read in large 2D and 3D stationary or time-varying flow fields and structures. *Planktos* is an ABM framework with the unique capability to conduct 2D or 3D simulations in spatially and temporally heterogeneous fluid environments which contain static immersed boundaries. It is organized as an object-oriented Python library that can be called as part of any Python script or Jupyter notebook (Kluyver et al. 2016). *Planktos* leverages the high-performance numerical routines of NumPy, SciPy, and Matplotlib (Harris et al. 2020; Virtanen et al. 2020; Hunter 2007) to provide a complete package to conduct and visualize simulations.

Planktos is both open-source and features online documentation (hosted at https://planktos.readthedocs.io) with tutorial-style working examples (Strickland 2018).

2 Agent-Based Modeling Framework

Planktos models individual organisms, or agents, in a rectangular, two- or three-dimensional fluid environment. A fundamental assumption of *Planktos* is that these organisms are small enough to have a negligible effect on the fluid environment around them. This assumption allows the fluid velocity field to be constructed ahead of time and completely independent of the agents' movement. Without this assumption, it would be necessary to calculate a complex fluid–structure interaction problem with possibly hundreds of organisms. Even single-organism problems of this nature are often highly complex and the subject of many studies and dedicated, high-performance software (Jones et al. 2016; Li et al. 2018; Samson et al. 2019; Hossain and Staples 2020; Borazjani 2020; Hoover et al. 2021; Battista et al. 2017a; Griffith 2019).

Planktos is capable of loading any fluid velocity field specified on a regular, rectangular grid, including time-varying fluid velocity fields. Supported data formats currently include ASCII VTK (Schroeder et al. 2006), ASCII VTU and NetCDF files, with others likely to be added as demand requires.

The Visualization Toolkit (VTK) is an open-source, object-oriented software system for computer graphics, visualization and image processing (Moreland 2021; Schroeder et al. 2000). The VTK file type is commonly used with the open-source CFD software library OpenFOAM (Weller et al. 1998) and the associated post-processing library paraFoam. Furthermore, COMSOL Multiphysics (www.comsol.com) and the open-source visualization libraries Paraview (Ahrens et al. 2005) and Visit (Childs et al. 2012) can read and export flow data to this file type. For the VTK format, special attention is paid to importing data that corresponds to the typical output of our two-dimensional immersed boundary software library (IB2D) (Battista et al. 2017b, a). Note flow field data readable by VisIt or Paraview, such as the Immersed Boundary Method with Adaptive Mesh Refinement (IBAMR Griffith 2019; Griffith et al. 2007), may also be exported to VTK format.

Planktos is capable of generating a selection of 1D, analytical fluid velocity fields out of the box which are then translated into 2D or 3D. These models include flow through a uniform porous layer of fixed height described by the Brinkman equations (Brinkman 1949) (see Strickland et al. 2017 for details) and wide-channel flow with a vegetation layer according to the two-layer model described by Defina and Bixio (2005). For terrestrial applications that consider flow within and above a uniform, homogenous canopy, we have added the model described by Finnigan and Belcher (2004). After generating the velocity fields for any of these models, the fluid data may be subsequently exported from *Planktos* in VTK format, with or without agent simulation.

Static immersed mesh structures may also be imported as either vertex points for line mesh boundaries (two spatial dimensions) or STL (Standard Tessellation Language) (Grimm 2004) formatted triangular meshes in three dimensions. Note that the STL is a file format commonly used for 3D printing and computer-aided design, and many 3D

repositories provide complex 3D biological objects in STL format (of Health 2022; Boyer et al. 2014; Digitalmorphology 2019). Agents respect these immersed meshes, undergoing sliding collisions (velocity vector projection onto the boundary and beyond it) upon contact.

Simulations within *Planktos* use an object-oriented programming (OOP) paradigm which aims to be both Pythonic and to follow the OOP principle of encapsulation, resulting in as little scripting by the end user as possible. To this end, *Planktos* is built around two main classes, the *environment* class (which collects and handles all information related to the domain, fluid and immersed structures) and the *swarm* class (a collection of individual agents that share a movement model), along with a library of supporting functions that aid the user in specifying motion models. The environment class holds all information about the fluid velocity field and simulation time while supplying a number of methods for loading, viewing and performing calculations and interpolations on the fluid. Agents with a similar motion model are then grouped together into an instance of the swarm class, which holds all information about its member agents and their motion model and provides methods for plotting and saving agent data. Every instance of the swarm class belongs to an environment object, and an environment object may hold several swarms.

2.1 Environment Class

Every instance of the environment class (an environment object) can be conceptualized as a rectangular, spatial domain in either two or three dimensions. The lower-left corner is located at the Euclidean origin. Boundary conditions are specified with respect to the agents on each side of the domain, with the options being to mask any agents exiting the domain from that side, periodic boundary conditions, or to have the agents undergo a sliding collision upon contact with the boundary (no flux). Unless the fluid velocity is everywhere zero within the environment, a fluid velocity field is specified on a regular mesh of grid points which always includes the domain boundaries. These data may vary in time, in which case they are specified at a given set of time points, but the spatial mesh on which they are specified must remain constant. Analytical fluid velocity fields are also available, as described before, and they are also specified on a mesh for consistency.

As an added feature, *Planktos* includes automation for tiling periodic fluid velocity fields in either the x- or y-directions to obtain a larger domain. This tiling automatically includes any immersed boundaries that have been loaded into the environment object. Extending the domain in a particular direction is also an option, in which case *Planktos* copies the fluid velocity at the previous boundary onto all-new grid points, resulting in spatially homogeneous flow in the direction of the extension. While this is not always appropriate, it can occasionally be of use to extend the domain away from the main fluid dynamics.

In order to provide the fluid velocity at an arbitrary location within the domain, the environment class is capable of interpolating the fluid velocity field using first a cubic spline in time (as implemented by the SciPy function `CubicSpline` in the interpolate package). A linear interpolation in space is implemented by the SciPy

function `interpn` in the same package (Virtanen et al. 2020). Tricubic interpolation in space is not currently available in SciPy and is planned as a future feature of *Planktos*. As stated previously, the environment class contains several methods for importing VTK, VTU and NetCDF fluid velocity data, particularly from VisIt, ParaView, IB2d and COMSOL, with more import functions likely to be added in the future.

In 2D, immersed mesh structures may be imported to the environment object from a list of vertex points corresponding to the format used in IB2d (Battista et al. 2017b). These vertices are then converted into a line mesh using one of several available algorithms within the environment class. The first option is to create a convex hull from the vertex data using the `ConvexHull` function in SciPy (Virtanen et al. 2020). This algorithm is highly efficient, leveraging the Qhull library (2020, www.qhull.org) under the hood, and this method of the environment class is also capable of transforming 3D vertex meshes, such as those used in IBAMR, Griffith et al. (2007) to convex triangular meshes. The second option, available for 2D vertex points only, is to associate nearby vertex points as endpoints of a line and then potentially add vertex points wherever lines cross away from an established vertex. With careful tuning, this algorithm is capable of recreating immersed mesh structures with significant concave features, such as the lobes of a leaf (see Fig. 1).

In 3D, the preferred way of importing mesh structures is via STL file, as this format is both ubiquitous and removes any ambiguity as to how the mesh should be configured. In either case, only static meshes are currently supported in *Planktos*; moving meshes represent a significant increase in complexity with respect to interaction with agents and are planned as a future feature of this software. Once an environment object is created, a fluid velocity field is loaded, and optionally an immersed mesh structure is loaded, the environment class provides for basic plotting of the domain without extra simulation elements to ensure that everything has loaded correctly.

The environment class contains several methods for calculating fluid field data that may be useful either in specifying agent behavior or for visualization and analysis. Examples include calculating the vorticity of the fluid velocity field (which can be exported from *Planktos* as a series of VTK files) or finding the temporally interpolated time derivative of the fluid velocity. Through the swarm class (discussed below), the gradient of the fluid speed (given as the magnitude of the fluid velocity) can also easily be interpolated at all agent positions. *Planktos* is also capable of calculating finite-time Lyapunov exponents for an unusually large number of motion models, an exciting functionality that is not duplicated in any other software package that we are aware of.

Finite-time Lyapunov exponent (FTLE) fields are a powerful tool for studying time-dependent, nonhomogeneous dynamical systems such as those frequently encountered in the context of fluid flow. In a similar fashion to the Lyapunov exponent of homogenous, time-independent dynamical systems, the FTLE provides a measure of how sensitive the spatial–temporal orbit is to its initial conditions. In this setting, an initial time for the analysis is chosen and initial spatial conditions are specified on a regularly spaced grid. The FTLE field is then calculated by integrating forward in time and using a finite-difference approximation for the gradient at each point, taking advantage of the grid geometry to define a stencil. The actual FTLE is based on the largest eigenvalue of a finite-time version of the Cauchy-Green deformation tensor and is sometimes

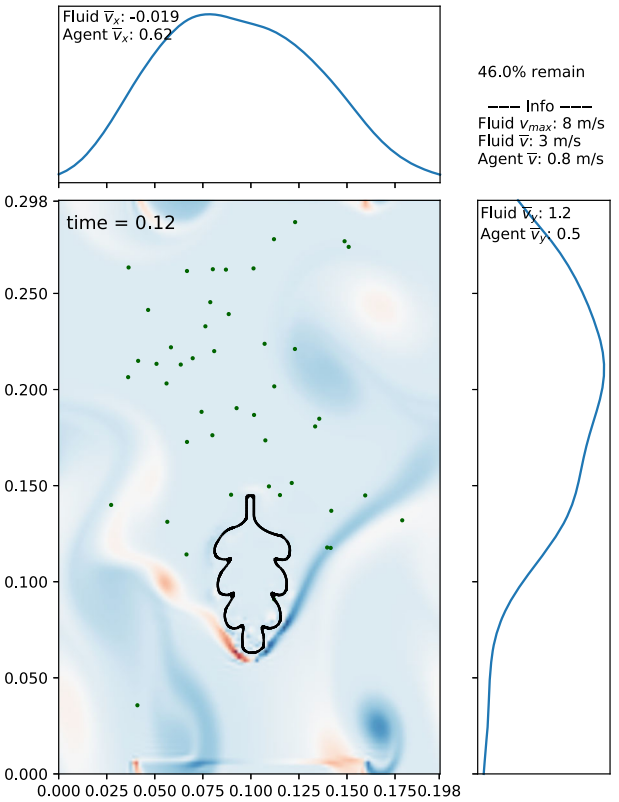

Fig. 1 *Planktos* simulation of tracer particles with constant diffusive noise in fluid flow around an immersed leaf structure. Fluid vorticity is shown in the background, along with Gaussian kernel density estimations for agent position in the x- and y-directions. Agents that reach any of the four edges of the environmental domain are no longer simulated; in this image, 46% of the original 100 agents remain. The leaf structure was loaded as a set of vertex points with nearby points associated on the mesh. Due to the geometry of the leaf, some mesh elements overlapped and vertices were added at these intersections

referred to as the largest FTLE for this reason; full details are provided in a number of sources that describe the mathematics in full detail (Haller and Yuan 2000; Haller 2002; Shadden 2005).

This process aims to identify separatrices (distinguished invariant manifolds) of the system, referred to as Lagrangian Coherent Structures (LCS), which divide regions of the fluid environment into areas of dynamically distinct activity with respect to a particular immersed particle motion model. When examining the fluid velocity field or collective particle motion, these regions may not be immediately apparent. Two basic types of regions (with respect to a time interval, $t \in [t_0, t]$) are referred to as attracting LCS and repelling LCS, which are material surfaces that attract or repel nearby trajectories, respectively, at the locally highest rate. While potential repelling LCS may be identified using forward-time integration (from t_0 to t) and the largest FTLE as described above, following the approach of (Haller and Sapsis 2011), potential attracting LCS at time t may be identified using the smallest FTLE which *Planktos*

calculates as well (Haller and Yuan (2000); Haller (2001, 2002); Haller and Sapsis (2011); Shadden (2005)).

Several programs exist with the capability of calculating FTLE fields in either 2D or 3D environments (Childs et al. 2012; Dabiri 2021; Liu 2017; Haller 2021). However, to our knowledge, all of these implementations are limited to models of tracer particles (i.e., with no active movement) or inertial particles. On the other hand, software that targets general ODE dynamical systems models may not be robust to nonautonomous systems or experimental data and does not encompass algorithmic models of motion and immersed 2D or 3D boundary collisions (Nave 2018).

2.2 Swarm Class

The swarm class acts as a collection of agents that share a similar motion model. All agents must be grouped into a swarm, a convention within *Planktos* that achieves two purposes: (1) computational efficiency, in that swarm methods can be implemented as vectorized operations on NumPy arrays (Harris et al. 2020), and (2) the convenience of quickly and easily performing operations and visualizations over a group of similar agents by default. Every swarm must also have an associated environment object, which at minimum defines the domain size, spatial dimension and boundary conditions while holding information about the current time and previous time points in the simulation (Fig. 2).

Swarms may be of arbitrary size, and agent positions may be initialized at arbitrary locations within the environment domain. A few convenient methods exist for initializing agents at a single point or randomly throughout the domain, but positions can also be supplied on an individual basis. Swarm objects may also carry an integer value that is used to seed a random number generator associated with the swarm object at the time of initialization. This provides a method for generating reproducible results even if the swarm's movement model is stochastic in nature.

Internally, information about agent locations, velocities and accelerations within the swarm is stored as masked NumPy arrays where each row is an agent and each column is a spatial dimension (Harris et al. 2020). The mask refers to whether or not the agent has left the spatial domain; if so, its data are typically no longer updated for computational efficiency. The mask provides an automatic way of excluding the agent from plots. Agent locations in previous time steps are stored similarly within a Python list for efficient appending. Past velocity and acceleration data are not stored to save on memory resources.

Any agent or motion model properties that do not vary from agent to agent are stored within a swarm attribute implemented as a Python dictionary. However, a key part of agent-based modeling is the ability for agents to possess individually varying states or properties. In some cases, individual agent states may update within the model in response to other agents or the environment (e.g., in a model of infectious disease, an agent may change state to represent the stage or compartment of the disease that the agent currently belongs to), and in other cases the agents may possess static properties which provide for individual variation in a way that roughly captures individual differences seen in nature.

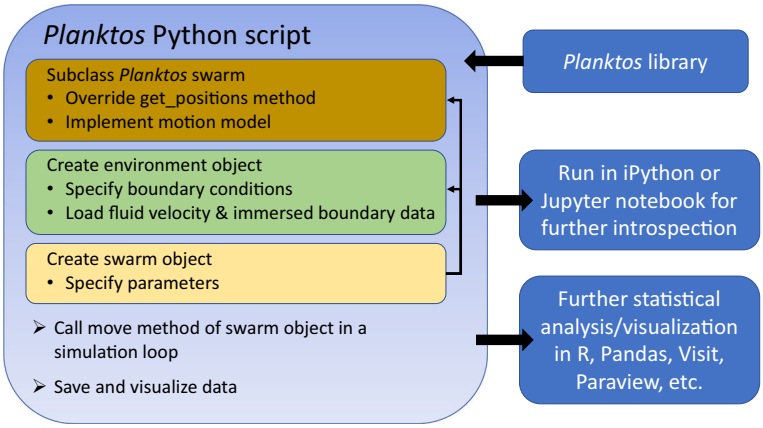

Fig. 2 Flowchart for a typical workflow in *Planktos*. Running a *Planktos* script within either iPython or a Jupyter notebook (Kluyver et al. 2016) allows for further object introspection after the simulation is finished (including continuation of the simulation), and data can be exported to a variety of formats for further analysis outside of Python

In order to provide this functionality in a way that is both self-contained and allows for easy inspection and statistical analysis on the part of the user, swarm objects store all individually varying properties as a pandas `DataFrame` (Wes McKinney 2010; pandas development team 2020). The `DataFrame` data structure is similar to the dataframe structure in R R Core Team (2017), but it is built on top of NumPy and intended to integrate well with the Python scientific computing environment. Additionally, the pandas library (like R) has many built-in tools for practical, real-world data analysis and robust IO methods for loading and saving to CSV, Excel files, HDF5 and many other formats which users of *Planktos* can take advantage of (Harris et al. 2020; Wes McKinney 2010; pandas development team 2020).

Position data for the agents are readily plotted in either 2D or 3D using built-in methods of the swarm class (see Figs. 1, 4, 7 and 8). Plots may be a static plot showing a single time step or an animation showing multiple time steps in sequence. Both images and animations can easily be saved to disk with options for still images, including both raster and vector graphics. In addition to showing the agents' location within the domain, all plots and animations include two or three additional plots showing a kernel-density estimate for agent positions in each spatial dimension. Options for these density estimates include a histogram (agents are spatially binned in 26 equally spaced intervals along each spatial dimension) or a Gaussian kernel estimation, where the user determines the estimator bandwidth, the amount of agent "jitter," or using Scott's rule (Scott 1992) as implemented in the SciPy stats library (Virtanen et al. 2020). In the case of 2D plots, fluid data can also be automatically plotted in the background, either as a quiver plot of velocity or as a pseudocolor plot of vorticity where vorticity is calculated within *Planktos* on-the-fly using the environment class (for example, see Figs. 1 and 4).

All plotting and saving of images and animations leverage the Python Matplotlib library, which also supports interaction with still images (e.g., panning and zooming)

(Hunter 2007). In 2D, environments whose length and width do not differ by a large amount will be plotted in *Planktos* with the proper aspect ratio; however, Matplotlib only supports 3D plotting to a limited degree, and therefore all 3D environments are plotted as a cube (the axes are labeled appropriately). 3D data visualization is a complex problem, and there exist several software solutions dedicated specifically to visualizing such data. VTK, VisIt and ParaView in particular have Python scripting interfaces (Schroeder et al. 2006; Childs et al. 2012; Ahrens et al. 2005). Alternatively, *Planktos* can save position data for all agents within the swarm either in CSV or ASCII VTK format for external visualization and analysis.

2.2.1 Collisions with Static Immersed Mesh Structures

While the details of specifying agent motion and interaction (both with each other and the environment) will be discussed below in Sect. 2.3, the process by which the swarm class detects and handles agent collisions with immersed mesh boundaries should be mentioned first, since it is automatically carried out by *Planktos* whenever an immersed boundary is present in the environment class, unless it is specifically turned off by the user.

First, collisions with immersed boundary mesh elements must be detected and any intersection points recorded. During a given time step of length Δt starting at a time t, an agent's trajectory may be approximated by the line segment connecting its position at time t with its position at time $t + \Delta t$. All mesh elements within a certain radius of the agent's starting position (the exact radius is calculated on a per-agent basis based on the distance the agent traveled) are then checked for potential intersection with the agent's line segment of travel, with only the intersection closest to the agent's start point being ultimately considered with respect to boundary conditions.

Intersections are detected using parametric equations for the line segment of the agent's travel and the current mesh element under consideration. Since mesh elements are either line segments (in 2D) or triangular portions of a plane (in 3D), this task amounts to finding the intersection of two lines (2D) or a line and a plane (3D) if it exists, followed by determining whether that intersection falls within both the agent's start and endpoints and the mesh element boundaries. This approach is well documented and has the benefit of being both general in its approach and computationally efficient (Sunday 2021).

Immersed boundary conditions in *Planktos* can be specified as either "sticky," in which case agents simply stop at the point of intersection until the next time step, or "sliding." In either case, a swarm property (`ib_collision`) is set to True for any agent that collided with an immersed boundary during that time step (False otherwise) to report to the user that a collision occurred. This information can be used to accomplish certain behaviors, e.g., halting all subsequent agent movement after that agent comes in contact with an immersed boundary, for instance in organism capture applications.

In the case of a sticky boundary condition, the actual intersection point used by *Planktos* is a perturbation of 10^{-7} from the mathematically determined point of intersection, taken in the direction opposite that of the agent's movement. This perturbation is for numerical stability: roundoff error will commonly result in the numerically cal-

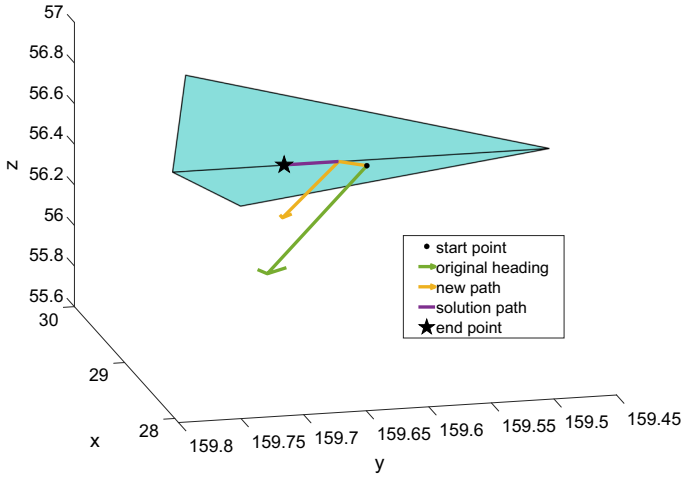

Fig. 3 3D projection example onto an immersed, triangular mesh structure with sliding collision. The original trajectory of the agent (before collisions are accounted for) is given by the green arrow, starting from the black dot. Using vector projection, the agent slides along the mesh element it intersected until an edge is reached, at which point it would normally continue along its original heading for a proportional distance (yellow path). If an adjoining mesh element is intersected, this is detected recursively, and the agent slides along this element as well (purple path). The black star gives the final location in this case.

culated point of intersection being found on the far side of the mathematically precise location of the immersed boundary element, with the resulting effect of the agent passing through the boundary. As a result, all calculations in *Planktos* involving immersed boundaries should be taken to have precision no greater than 10^{-6}, regardless of spatial scale.

In the case of sliding collisions, the portion of the agent's movement left over after the point of intersection is projected as a vector onto the corresponding mesh element perturbed a distance of 10^{-7} in a direction normal to the actual mesh element and toward the starting point of the agent's movement. The intersection of the projected segment with (perturbed) mesh element boundaries is then detected. If such an intersection occurs, the remaining portion of the agent's movement vector after the new intersection is projected back on its original heading. The algorithm then undergoes recursion to look for additional intersections, with care taken to avoid pathological cases at concave intersections of mesh elements. More specifically, intersecting the same mesh element more than once can result in the agent halting at a shared vertex of two adjacent mesh elements in 2D or being projected along the shared line of two adjacent mesh elements in 3D in order to avoid an infinite or computationally expensive loop (see Fig. 3).

2.3 Motion Models

Agent behavior is specified by subclassing the swarm class and then overriding the `get_positions` method of the class. This method takes in a positive time step

and returns the new agent positions after that time step has elapsed but before any boundary conditions have been applied. Subclassing `get_positions` is extremely straightforward: a full example with tutorial-style explanation is available both as an executable script within the examples folder of the source code and in the online documentation for *Planktos*. More importantly, this approach allows the user to write any type of algorithm for updating agent positions and states, with all attributes of the swarm object and the associated environment object (which is itself an attribute of the swarm) available for reading and writing.

Within this framework, *Planktos* provides a number of tools to aid in the specification of motion models based on ordinary or stochastic differential equations. These are located within the *motion* module of *Planktos*. Differential equations will typically need access to the fluid velocity field and other properties of the swarm and its associated environment object; for this reason, the *Planktos* documentation provides examples of how to create an ordinary differential equation (ODE) generator function. These generators take in a swarm object and return a function handle to the system of ODEs, thereby giving the ODEs access to all of the properties and methods of the swarm. Once this is done, they can subsequently be solved either with the provided RK45 function which implements the Runge–Kutta Dormand–Prince method (Dormand and Prince 1980) or within the context of a stochastic differential equation (SDE).

Since most agent-based models are stochastic in nature, it is often natural to express their movement as an SDE. In *Planktos*, it is assumed that this equation takes the form

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + \sigma(t)d\mathbf{W}_t \tag{1}$$

where $\mathbf{X}_t$ is a vector of agent positions at time t, $\boldsymbol{\mu}(\mathbf{X}_t, t)$ the drift coefficient (drift velocity), $\sigma(t)$ the diffusion coefficient, and $\mathbf{W}_t$ is the Wiener process. $\boldsymbol{\mu}$ may be specified as a system of differential equations which describes the deterministic behavior of the agents in the absence of stochastic effects, or $\boldsymbol{\mu}$ may be specified as a constant vector on a per-agent or global basis. $\sigma(t)$ may be given as a diffusion tensor or covariance matrix, again on either a per-agent or global basis. With spatially invariant σ, the Itô and Stratonovich interpretations of this equation are equivalent and *Planktos* provides an Euler-Maruyama solver which is strong order 1 for this problem (Kloeden and Platen 1999). A planned future feature of *Planktos* is to extend Eq. (1) to spatially varying values of σ (e.g., $\sigma(\mathbf{X}_t, t)$), providing a strong first-order solver based on the Milstein method which can be set to either Itô or Stratonovich interpretations (Mil'shtejn 1973). This functionality will be particularly useful for porous media and similar applications.

At run-time, the user calls the `move` method of the swarm object and provides it with a time step within a for-loop. The `move` method calls `get_positions` and also handles boundary conditions and velocity and acceleration data updates after `get_positions` returns. Explicitly updating agents within a for-loop in this manner presents the user with straightforward opportunities to both collect and respond to simulation data as it is generated by writing additional code before or after the `move` method executes.

3 Workflow

A typical workflow for creating a *Planktos* simulation includes the following steps within a Python script file (visualized in Fig. 2). All of these are demonstrated in the many tutorial-style example scripts within the example folder of the *Planktos* source code repository. Due to the abstraction of many common tasks, *Planktos* scripts can be quite short: 40–50 lines at times, fully commented.

1. Add the location of the main *Planktos* library to the Python path and import it, along with any other needed libraries.
2. Subclass the *Planktos* swarm class and override the `get_positions` method with the desired motion model and any other behavior agents should have.
3. Create a *Planktos* environment object with suitable dimensions and boundary conditions.
4. Load fluid velocity data or supply fluid parameter information and generate an analytic flow field within the environment object. Optionally load immersed boundary data.
5. Create a swarm object from the user-defined subclass with the desired number of agents, assigning it to the already created environment. Specify any agent parameters necessary for the simulation.
6. Call the `move` method of the swarm object in a loop along with any other necessary tasks for conducting the simulation or recording data on a per time step basis.
7. Use the `plot` and `plot_all` methods of the swarm object to visualize the result. Save and analyze data from the simulation as needed.

4 Examples

Planktos comes with multiple working examples (located within the examples folder of the repository), each of which is fully commented in tutorial style focusing on different aspects of the software's functionality. Five of these examples are written up as tutorials within the online documentation (located at https://planktos.readthedocs.io/en/latest/examples/index.html), complete with code snippets, references and accompanying images. Full documentation of the *Planktos* API is also available from the online documentation (at https://planktos.readthedocs.io/en/latest/api/index.html) and from the docstrings of the library itself, which are all immediately accessible using the built-in help function included with Python.

To demonstrate the computational and data visualization capabilities of *Planktos*, we will highlight several of the included examples below. Some of the examples also illustrate how *Planktos* might be used in applications, including problems involving the movement of marine plankton within sheltering layers, the capture of plankton by filter feeders and the collective behavior of agents in a wind tunnel.

4.1 Flow Around a Cylinder in a 2D Channel

In this example, IB2d (Battista et al. 2017b, a) was used to simulate flow past an immersed cylinder boundary in 2D with vortex shedding. The cylinder is within a

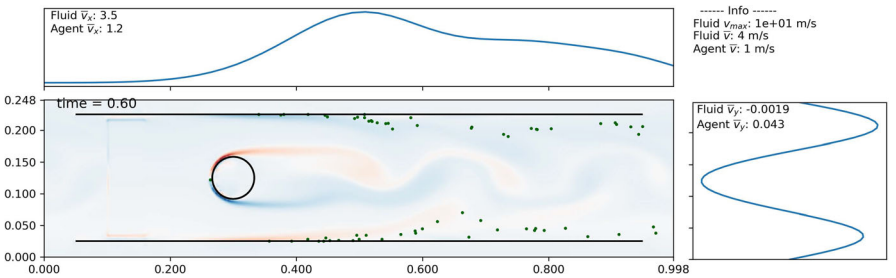

Fig. 4 *Planktos* result at time 0.6 for IB2d generated flow around a cylinder within a horizontal channel in 2D. Black lines represent immersed boundaries. Gaussian kernel density estimations for agent positions in the x- and y-directions are given, and the vorticity of the fluid (calculated within *Planktos*) is shown in the background of the simulation

channel, represented by two additional immersed boundaries. This simple model is representative of a study that one might perform in a flow tank or wind tunnel where tiny marine animals or insects are advected past a cylindrical obstacle. *Planktos* reads in both the time-dependent fluid velocity data (as a sequence of ASCII VTK files) and the vertex data representing all of the immersed boundaries. The vertex data are automatically converted into a mesh structure upon loading it into *Planktos* by associating nearby points within a given radius. The raw data from IB2d are available for download through a link given in the comments of this example and in the online documentation.

To keep the example simple, agent behavior was specified as fluid advection plus an unbiased random walk as given by the stochastic differential equation

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + 0.0001d\mathbf{W}_t \tag{2}$$

where $\boldsymbol{\mu}(\mathbf{X}_t, t)$ is the interpolated fluid velocity at agent location $\mathbf{X}_t$ at time t. The initial position for all agents was specified to be at 0.1 in the x-direction in the center of the channel. Figure 4 shows a still of the result at time $t = 0.60$, including Gaussian kernel density estimations and fluid vorticity which was calculated directly within *Planktos*. When interacting with the immersed boundaries, agents automatically used the sliding boundary condition.

4.2 Finite-Time Lyapunov Exponent (FTLE) with Generalized Motion Model and Immersed Mesh Structure

Most agent-based simulations involve stochasticity, but *Planktos* is also fully capable of simulating uniform, deterministic behavior described algorithmically or by a system of fluid-dependent ODEs. In the latter case, *Planktos* provides a Dormand–Prince Runge–Kutta 4(5) adaptive step solver to approximate agent trajectories (Dormand and Prince 1980). This capability allows *Planktos* to calculate the 2D FTLE field with respect to fluid velocity data and a large range of deterministic models of motion for independently acting agents initialized on a regular grid. For ODEs, the FTLE field relates to the deterministic behavior of agents as unbiased diffusive noise goes to zero

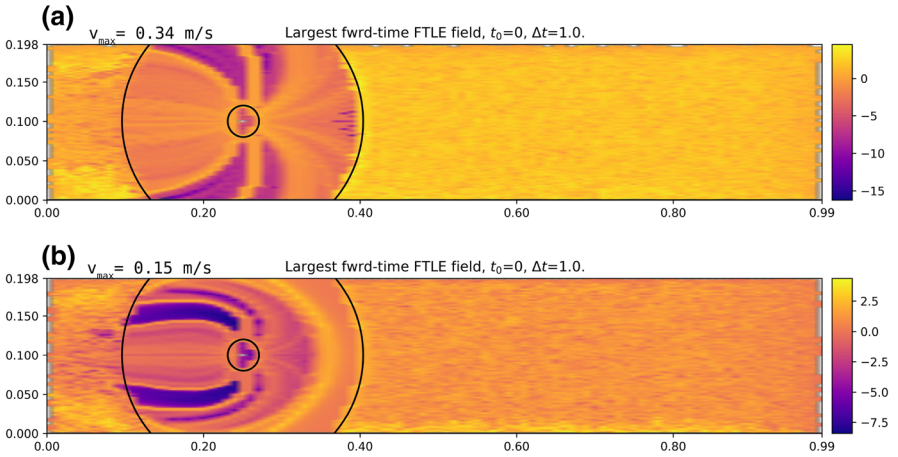

Fig. 5 FTLE fields for a parasitoid wasp flying toward a target cylinder (small black circle) in fluid flow moving left to right around the cylinder. The larger black circle describes the radius of cylinder detection for a parasitoid. **a** The maximum flight velocity of 0.34 m/s is similar to the fluid velocity behind the cylinder (0.35 m/s), resulting in primarily detection-limited success at landing on the cylinder. **b** The maximum flight velocity is 0.15 m/s instead, and the leading edge of an LCS now appears within the detection radius. Both images reveal clear substructures related to flight relative to the local fluid velocity

in an associated SDE motion model (see Sect. 2.3). This relation allows us to study the effect of such noise on the qualitative dynamics of potential trapping regions.

Furthermore, since *Planktos* is capable of handling agent collisions with immersed mesh structures, FTLE fields can potentially take these interactions into account as well—a functionality that, as far as we are aware, exists in no other software package. From the point of view of small organism dynamics in fluid, this capability is exciting because it allows one to directly compare information about LCS with respect to fluid field properties to the trapping potential of these structures under a given organism's motion and behavior model. One can easily imagine scenarios where LCS are more or less robust to a given organism's behavior, depending for instance, on if the organisms display shelter-seeking behavior, are attempting to escape predation or are blocked by immersed structures.

Figure 5 shows the resulting FTLE field for a simulation of agents moving toward a cylindrical target with a background flow field that moves from left to right. This simulation was inspired by a controlled wind tunnel experiment in which a parasitoid wasp flies upwind toward a cylinder (Sarig and Ribak 2021). In this experiment, the insects are released downwind of the cylinder and fly toward the cylinder if they are within a given detection distance that is proportional to the cylinder's diameter. In this simulation, the detection distance is denoted by the black line in the figure. Once the cylinder is detected, the agents use a burst flight speed for a set duration of time and actively fly toward the cylinder. After this time, the insects are fatigued and fly at 30% of that velocity. In the top panel of the figure, the burst flight speed and background flow speed are nearly balanced and set to 0.35 m/s and 0.34 m/s, respectively. Once within the detection distance, the agents reach the target, seen by the separatrix that corresponds to the detection distance. In the bottom panel, the active flight speed is

set to only 0.15 m/s, and some of the insects downstream of the cylinder but within the detection distance are unable to make it to the cylinder. In this case, the separatrix (yellow line) is upstream of the detection distance.

The main limitation of this approach is that the behavior of each agent must be independent of surrounding agents. While *Planktos* as a framework does not limit motion models in this way, calculation of the FTLE field requires it. Relaxation of this FTLE assumption is highly nontrivial and likely limited to specific scenarios, so it has not been a goal of our software development so far.

4.3 Movement of Brine Shrimp Through a Sheltering Layer of Cylinders

To illustrate an example that can be compared to experimental measurements of the movement of agents, we present previous results of a *Planktos* simulation of *Artemia spp.*, or brine shrimp, that actively swim within a flow tank (Ozalp et al. 2020). To introduce complexity in the flow field, we 3D printed cylindrical arrays that represent beds of idealized macrophytes, or aquatic plants such as sea grasses and reeds. The motivation behind this experiment is that macrophytes are thought to provide a sheltering region for small plankton in the open ocean. Both the height and the density of the cylinders were varied to consider different sheltering effects. All bases of all models were 150 mm long and 75 mm wide, and the water height in the flow tank was 75 mm. Each cylinder was 0.25 cm in diameter with heights varied from 0 to 3 cm. The brine shrimp were injected in the upstream portion of the flow tank and were advected downstream (see Fig. 6A). Video recordings were used to quantify the distribution of brine shrimp over time. Figure 6B shows the normalized count of brine shrimp over time in a region directly downstream of the model that was 4 cm wide and 8 cm high.

Planktos was then used to simulate the movement of the brine shrimp within the idealized models of macrophytes. To generate flow fields, COMSOL Multiphysics 4.5 was used to solve the steady Navier–Stokes equations (e.g., $\partial u/\partial t = 0$; see Fig. 6C. The dimensions of the fluid domain were matched to the actual flow tank and were set to 80 x 320 x 80 mm. The density and dynamic viscosity of the fluid was set to $\rho = 1025\ kg/m^3$ and $\mu = 0.00108\ kg/(m \cdot s)$, respectively. A parabolic inlet boundary condition with a maximum velocity of 44.4 mm/s and a zero pressure outlet condition was used. All other walls used a no-slip boundary condition.

For each *Planktos* simulation, 10,000 agents were initialized with a starting position 1 mm behind the center of the model and 3 mm from the bottom of the tank, similar to the release point of the actual brine shrimp. The agents were advected with the flow and were given additional 'jitter,' or unbiased Brownian motion with a variance of 2.5 mm^2/s. Arrival times were calculated as the first instant in which the agent entered the measurement zone (the region directly behind the model). Representative results are given in Table 1. Note that the plate case represents the case when there are no cylinders, corresponding to the solid red line in Fig. 6B. The subsequent models represent cases of increasing cylinder height and density, again corresponding to the experimental data shown in Fig. 6B. For both the experiments and simulations, the arrival times and the standard deviations increase with increasing cylinder density and height, indicating that macrophyte beds do have a significant sheltering effect.

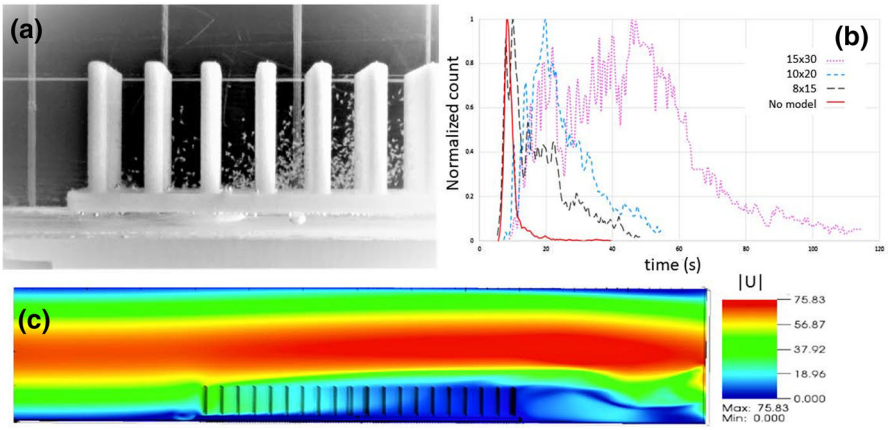

Fig. 6 Results from the dispersal of brine shrimp over time for Ozalp et al. (2020). **a** A 3D printed model of a cylindrical array inserted at the bottom of a flow tank with brine shrimp injected within the upstream section of the cylindrical array. **b** Normalized count of individuals over time in the measurement zone. The solid red line represents the dispersal when there is no cylindrical model in the tank. The 3D printed models considered include a plate with an 8×15 array of 1-cm-tall cylinders (black dashed line), a plate with a 10×20 array of 2-cm-tall cylinders (blue dashed line) and a plate with an array of 15x30 3cm high cylinders (pink dotted line). **c** A representative COMSOL simulation showing the magnitude of the velocity through a 2D slice taken through the center line of the model

Table 1 Statistics for the timing of brine shrimp agents entering the measurement zone

	Mean	Median	Mode	Std.	Skewness	Kurtosis
Plate	10.26556	7.60	4.8	6.644776	1.577448	5.648092
8×15_1 cm	16.60715	11.80	7.1	12.855364	1.867912	7.226498
10×20_2 cm	24.37854	20.55	15.9	12.792898	1.755368	7.025288
15×30_3 cm	53.52796	51.20	48.6	23.670932	0.971755	4.696066

All values are given in units of seconds, and both skewness and kurtosis are calculated as Pearson standardized moments

4.4 Flow Through a Uniform Porous Layer with User-Defined Agent Behavior

In this example, *Planktos* is used to internally generate a 1D Brinkman flow through a porous layer of fixed height with parallel shear flow above the layer. This represents a simplification of the previous example, where the porous layer might provide a sheltering environment for plankton. Brinkman's equation (Brinkman 1949) is given by

$$\rho(\mathbf{u}_t(\mathbf{x}, t) + \mathbf{u}(\mathbf{x}, t) \cdot \nabla \mathbf{u}(\mathbf{x}, t)) = -\nabla p(\mathbf{x}, t) + \mu \nabla^2 \mathbf{u}(\mathbf{x}, t) - \alpha \mu \mathbf{u}(\mathbf{x}, t) \quad (3)$$

where α is the inverse of the hydraulic permeability. We assume that the flow is steady ($\partial u/\partial t = 0$), fully developed ($\partial u/\partial x = 0$) and zero in all cross-stream directions. The fully developed 1-D fluid velocity $u(x)$ is then projected into the 3D environment.

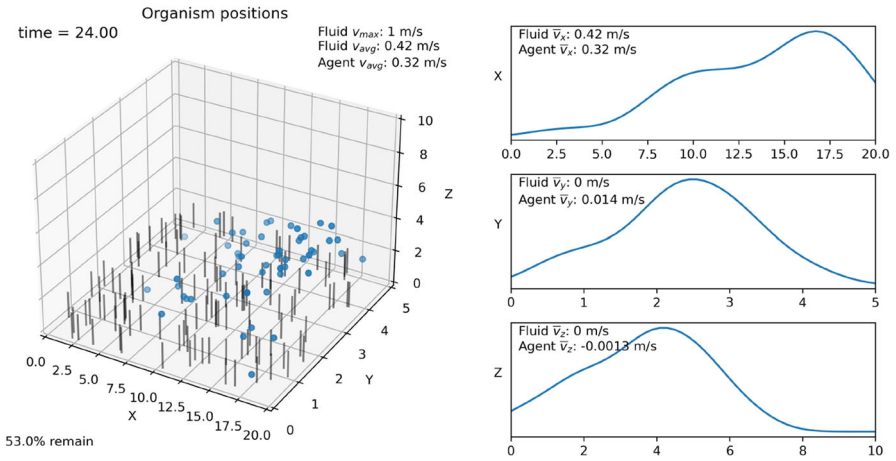

Fig. 7 Snapshot of a 3D simulation with Brinkman (1949) fluid velocity through a fixed-height porous layer and user-defined agent behavior. Gaussian kernel density estimations for agent positions in the x-, y- and z-directions are also shown. *Planktos* visualizes a porous layer via randomly placed "grass" of the correct height. This grass is for visualization purposes only: it is not a barrier to the agents and does not represent the magnitude of the hydraulic permeability, α

Our assumptions result in an analytical solution to Eq. (3), so it is this solution that *Planktos* uses to generate the fluid velocity (Strickland et al. 2017).

To specify agent behavior, it is necessary to subclass the *Planktos* swarm class and override the `get_positions` method. The new swarm class is then used to carry out the simulation. This example provides a full tutorial of how to accomplish these steps and is fully detailed in the online documentation at https://planktos.readthedocs.io/en/latest/examples/agent_behavior.html. For this example, 88% of the agents were made to move toward the mean position of the swarm. Membership in this biased group was stored as a Boolean agent property, and it was assumed that diffusive noise for all agents is scalar. Agents that moved toward the mean position of the swarm did so with a bias magnitude equal to one standard deviation of their scalar noise, and the magnitude of their scalar noise term was cut in half. The appropriate stochastic differential equation was then solved for each agent to specify its position throughout the simulation. A snapshot of the result can be seen in Fig. 7.

4.5 Filter Feeding in Sea Fans

In this example, we demonstrate the potential of *Planktos* to reveal how the morphology of filter-feeding organisms can enhance the rate at which they capture prey. Our simplified example is motivated by the structure of *Gorgonia spp.* or sea fans. Sea fans are sessile soft corals ubiquitous throughout the marine world that play an integral part in coral reefs and shallow reef formation in particular (Rossi et al. 2018). Gorgonian corals are believed to be the foundational support for the larger reef communities and crucial to the re-establishment of disturbed communities (Lacharite and Metaxas

2013). Sea fans comprise colonial polyps that form a fan-like structure whose growth is directed normal to environmental currents (Wainwright and Dillon 1969).

Sea fans possess elaborate branching structures which create feeding flows as they sway (see Fig. 8B). These flows appear optimized for a particular range of velocities, possibly utilized to capture prey of a preferred size. As water circulates around and through the branches, boundary interactions slow the flow, forming recirculation zones downstream of the organism. Individual tentacled polyps emerge from branches, expanding their tentacles and further slowing the flow at the smaller scale. At the most miniature scale, the tentacles are covered with digitata, which capture prey and exchange particles with the surrounding fluid.

The spacing between the branches of the sea fan can drastically alter the flow that moves between them and subsequently affect the likelihood of prey capture. If the flow velocity between the branches is too high, it can quickly sweep away prey before capture can occur. If the flow is too slow, the fan acts as a solid surface, and the prey is mostly swept around the sea fan, preventing capture. In this example, we have selected two biologically relevant choices of spacing that permit a small and relatively large amount of flow between the branches.

The sea fan is idealized as an infinite row of cylinders normal to the oncoming flow. To generate the flow fields used in the *Planktos* simulations, the steady Navier–Stokes equations were solved using COMSOL Multiphysics 4.5. More specifically, a stationary study (e.g., $\partial u/\partial t = 0$) was performed for single-phase laminar flow. Parameters typical of seawater, density $\rho = 1025\ kg/m^3$ and dynamic viscosity $\mu = 0.00108\ kg/(m \cdot s)$, were used. A cylinder 1 mm in diameter and 10 mm long was placed in the center of the working section and 20 mm downstream of the inlet. The fluid domain was 100 mm long and 40 mm high, and the width of the domain was set to 1.5 mm and to 3 mm to model different gap to diameter ratios ($G/D = 0.5, 2$). The inlet boundary condition of uniform flow was $U = 1$ or 8 mm/s for the simulations. The outlet conditions for the domain were set to zero pressure. The side walls (parallel to the central axis of the cylinder) used periodic boundary conditions to represent an infinite array of cylinders. The top and bottom of the domain (normal to the central axis of the cylinder) used symmetry boundary conditions. The complete flow field was formed by tiling the flow around each of the cylinders (see Fig. 8A).

For the *Planktos* simulations, the agents were initialized as a point source in the middle of the domain and upstream of the cylinders (see Fig. 8C). The agents were then advected with the flow and moved with an additional unbiased Brownian motion with a variance of 2.5 mm^2/s (representative of brine shrimp) (Fig. 8D). All simulations were conducted using 1000 agents with 100 repetitions to get the reported statistics shown in Table 2. When the agent contacted and remained stuck to a cylinder, it was considered captured.

For a background flow speed of 8 mm/s, the finer array of cylinders ($G/D = 0.5$) captures more plankton as there is slower flow that allows more time for the plankton to contact the cylindrical array. When $G/D = 2.0$, the slower background flow speed results in a much slower flow between the cylinders, representative of a nonlinear relationship between G/D, Reynolds number and normalized flow between cylinder (Cheer and Koehl 1987). In this case, the reduced flow speed between cylinders did not enhance capture rate as many of the agents were advected around the cylindrical

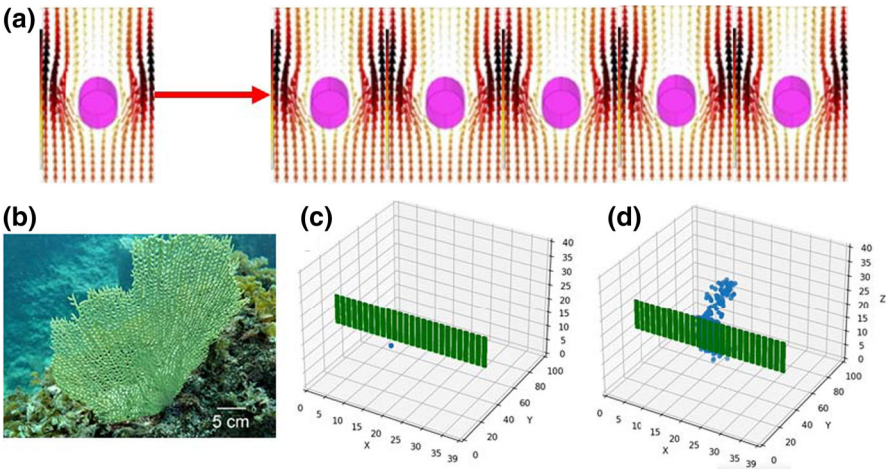

Fig. 8 Results from the simulation of plankton capture by sea fans. **a** COMSOL was used to generate 3D flow fields around a cylinder with periodic boundary conditions in the x-direction. The flow fields were then tiled to create flow through an array of cylinders. **b** An image of a sea fan taken in Grand Cayman illustrates the elaborate branching pattern that reconfigures to be normal to the direction of flow. **c** Output for the *Planktos* simulation where agents are released as a point source upstream of the cylindrical array. **d** Position of agents after 5.5 s. Note the cylinders have captured some agents, while others are advected above and below the cylindrical array

Table 2 Statistics showing the percent of agents captured by an idealized sea fan

G/D	U (mm/s)	Duration (s)	% Captured	Std.
0.5	8.0	60	54.3	1.5
2.0	8.0	60	31.8	1.5
2.0	1.0	90	27.4	1.4

G/D represents the ratio of the gap between cylinders to the diameter of the cylinder. U gives the background flow velocity for the simulation, and the duration is the length of time the *Planktos* simulations were performed

array. For all cases, the noise between trials is relatively small and can be considered normally distributed (based on failing to reject the null hypothesis of the Shapiro-Wilk test with $p \geq 0.35$ in all cases). These results suggest that sea fans living in faster flows should have smaller G/D ratios between branches.

4.6 Collective Behavior in Flow

In this example, we demonstrate how *Planktos* can be used to simulate the collective behavior of agents in flow. In this case, the flow field is representative of a low-speed wind tunnel that might be used for experiments with small insects. The wind tunnel has dimensions of $2.5 \times 0.2 \times 0.21$ m with an inlet velocity of 0.22 m/s. To perturb the flow field, a cylinder with a radius of 0.02 m and a height of 0.21 m is placed in the middle of the wind tunnel. The flow field is numerically determined by solving the

steady Navier–Stokes equations using COMSOL Multiphysics 4.5. Two flow fields are determined. The first models a traditional wind tunnel with velocity inlet and pressure outlet conditions and wall conditions ($u = 0$) on all sides. The second case is a periodic wind tunnel that is periodic in the x-direction.

Agent behavior followed the famous Vicsek model of collective behavior in self-driven particles (Vicsek et al. 1995). In this 2D spatial model, agent velocity in the absence of fluid flow has a fixed magnitude and an angle that is determined by the average direction of the velocities of agents within a radius r surrounding the given agent, plus some amount of noise. More specifically, for any agent i, its position after a single time step is updated with the equations

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t)\Delta t \tag{4}$$

$$\theta_i(t+1) = \langle \theta(t) \rangle_r + \Delta\theta \tag{5}$$

where $\mathbf{x}_i(t)$ is the position of the agent i at time t, $\mathbf{v}_i(t)$ is its velocity, Δt is the length of the time step, $\langle \theta(t) \rangle_r$ is the average direction of agents within a radius r around agent i, and $\Delta\theta$ is a random number chosen from a uniform probability within the interval $[-\eta/2, \eta/2]$.

We have included a direct implementation of this model in the *Planktos* package under the example folder as well as an extension into three spatial dimensions using spherical coordinates that we will discuss here. Both of these are constructed within the context of the low-speed wind tunnel flow which, to our knowledge, is a novel setting for collective motion of self-driven particles. For our implementation, agents move at a fixed speed of 0.02 m/s while also being advected by the fluid. That is, $\mathbf{v}_i(t) = \mathbf{v}_{i,p}(t) + \mathbf{v}_{i,f}(t)$ where $||\mathbf{v}_{i,p}(t)|| = 0.02$ and $\mathbf{v}_{i,f}(t)$ is the interpolated fluid velocity at the point $\mathbf{x}_i(t)$. The direction of $\mathbf{v}_{i,p}(t)$ is given by the two angles θ and φ, where θ is given counter-clockwise from the positive x-axis (to match the 2D Vicsek model) and φ is the inclination angle down from the positive z-axis. Both are updated in each time step according to Eq. (5), with $\Delta\theta$ a uniform random number in $[-0.25, 0.25]$, $\Delta\varphi$ a uniform random number in $[-0.125, 0.125]$, and $r = 0.1$.

3D *Planktos* simulations were conducted for 20 seconds using time steps of $\Delta t = 0.25$ in an environment containing fluid flows for both the traditional wind tunnel and the periodic wind tunnel as described above. In addition to agents following the Vicsek model (blue particles), simulations were also run using passively advecting particles (purple particles) and particles undergoing Brownian motion (green particles) with constant white noise that had a standard deviation of 0.02 m/s (matching the Vicsek particle speed). The results are displayed in Fig. 9, where the top row represents the top view of the wind tunnel and the bottom row represents the view looking down the length of the wind tunnel. The particle positions were exported to VTK and visualized using Paraview (Ahrens et al. 2005). Note that due to the periodic nature of the wind tunnel, agents that exit from the back of the wind tunnel (here visualized as the top) will re-enter at the inlet (bottom).

When comparing the cases of pure advection to advection with active movement driven by the Vicsek model, we find that both types of agents move in a similar manner, with the Vicsek agents moving faster (Fig. 9D–G). This is due to the fact that the Vicsek

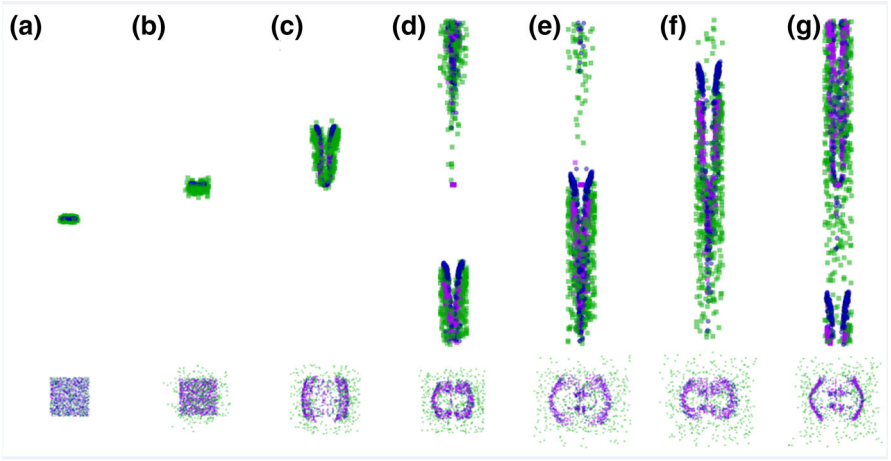

Fig. 9 3D simulations of particle motion in a periodic flow tank with an immersed cylinder in its center. The bottom row shows particle positions looking down the length of the wind tunnel, normal to the x, z-plane. The top row shows the particle positions looking from the top of the tunnel, normal to the x, y-plane. (**A**–**G**) correspond to 0, 1.25, 2.5, 5.0, 7.5, 10, and 12.5 seconds. The purple particles correspond to passive advection, the blue particles correspond to the case where the particles are advected and have an active motion given by the Vicsek model, and the green particles are advected with the flow and have some additional Brownian motion (uniformly distributed angle, speed standard distribution of 0.02 m/s to match self-propelled speed of Vicsek particles). Initial particle positions in all cases were given by a uniform random sheet at $y = 0.9$ behind the cylinder with $x \in [0.055, 0.155]$ and $z \in [0.05, 0.15]$

agents have an additional active velocity with a directional bias (e.g., downwind). The advection–diffusion agents disperse away from their initial configuration toward the edges of the wind tunnel. The differences between the Vicsek, pure advection and Brownian models can be easily seen by observing the structured patterns in the bottom row of Fig. 9 in contrast to the diffusive model. These differences highlight how the choice of an active motion model can dramatically affect the resulting spatial structures observed. It also suggests that the details of the agent interaction models can have a large impact on the spatial evolution of the swarming or schooling particles.

5 Discussion and Conclusion

The current implementation of *Planktos* can be applied to a wide range of problems where the innate velocity of the agents is on the order of the background flow velocity and when the movement of the agents does not significantly affect the background flow field. In particular, these applications are common for organisms on the order of 1 mm that swim or fly at Reynolds numbers near unity. In such instances, the boundary layers around the organisms and their densities are sufficiently small that one does not typically observe instances of active mixing as has been documented in bacteria and other microorganisms (Visser 2007; Kurtuldu et al. 2011; Wickramarathna et al. 2014; Simoncelli et al. 2017. On the other hand, the scale is small enough that the wake and vortices generated through active locomotion typically dissipate before such flow

structures are encountered by other organisms, which could either enhance or reduce their locomotory performance (Triantafyllou et al. (2016); Verma et al. (2018)).

Planktos can also be used to simulate interactions between organisms and stationary, rigid boundaries. Such interactions are important for a range of applications, including pollen capture (Niklas 1985a), filter feeding (Riisgard and Larsen 1995), settlement on complex substrates such as coral reefs (Koehl et al. 2007), and navigating to target structures or vegetation (Sarig and Ribak 2021). For example, it has been proposed in plants that female ovulate organs alter local airflow to enhance the interception of pollen with stigmas or micropyles (Niklas 1985a). The interaction between coral larvae and local water flow through reefs enables the larva to settle in beneficial locations (Hata et al. 2017). The structure of the deep sea glass sponge creates regions of local mixing within the body cavity that increases the residence time of particles and may enhance the uptake of food and sperm (Falcucci et al. 2021).

Thus far, our development of *Planktos* has focused on application problems of relevance to mathematicians, engineers and biologists working at the organismal level in fluid domains on the order of centimeters to meters. These applications include suspension and filter feeding by marine organisms (Jørgensen 1955), cue tracking at the level of a plant by tiny insects (Sarig and Ribak 2021) and larval settlement near the surface of an individual reef (Koehl and Hadfield 2004). Such applications are often studied in a laboratory environment and in a small wind tunnel or flow tank. Similarly, our applications have employed direct numerical simulations (DNS) of the Navier–Stokes equations and/or large eddy simulations (LES), using software packages such as OpenFOAM (Weller et al. 1998), IBAMR (https://code.google.com/p/ibamr/) and ANSYS (DeSalvo and Swanson 1985). Certainly, *Planktos* could be used to simulate plankton (or other aquatic animals) at much larger scales, including those of estuaries, lakes, bays, gulfs and oceans. These application problems would need to be chosen such that active animal movement is relevant relative to the background flow speed. For example, there is a large body of work on the biophysical mechanisms of larval dispersal at the ocean scale that could be studied using *Planktos* (see, for example, Hawkins et al. 2019). Moreover, additional consideration will need to be given to resolving the vastly different spatial and temporal scales between active organism movement and geophysical flows. In future work, we plan to extend the functionality of *Planktos* to import flow field data and geometries that are typically used by this geophysical scale community.

Future implementations of *Planktos* will also expand the types of applications that can be considered through the addition of moving boundaries, chemical cue tracking and response to flow disturbances, among other functionality. The motion of nearly all marine invertebrates in unsteady flow is non-negligible, particularly in the case of jellyfish and sea fans, and has consequences for the efficiency of particle capture by increasing encounter rates (Boudina et al. 2021; Krick and Ackerman 2015; Shimeta and Koehl 1997). The movement of plants, such as grasses, can also have a significant effect on the efficiency of pollen capture by increasing encounter rates between the pollen and stigma (Niklas 1985b). In terms of chemical tracking, many aquatic and aerial plankton, such as coral larvae and flies, track chemical cues that advect with and diffuse within the fluid (Koehl and Hadfield (2004); Hay (2009); Morehead and Feener (2000); Kim and Dickinson (2017)). To understand such chemical tracking

applications, one will also need to integrate odor tracking algorithms within *Planktos* (see van Breugel and Dickinson 2014; Villarreal et al. 2014; Pang et al. 2018; Baker et al. 2018). Additional model development could also include the detection of disturbances in the flow, triggering escape responses, e.g., short-duration bursts of fast swimming (Green et al. 2003; Buskey et al. 2002; Jakobsen 2001)..

Acknowledgements The authors would like to thank Diane Thompson and Jeremiah Hackett for their contributions on background information pertaining to reef systems and Shane Gladson for generating the FTLE examples for this paper. WCS would like to thank the Simons Foundation (585322) for their funding support regarding conference- and collaboration-related travel associated with this work. NAB would like to acknowledge support for computational resources from NSF OAC #1826915 and the NSF OAC #1828163 as well as support from the TCNJ Support of Scholarly Activity grant. CLH would like to acknowledge support from NSF CBET Fluid Dynamics #1916154. LAM would like to acknowledge support from NSF DMS Mathematical Biology #2111765 and NSF CBET Fluid Dynamics #2114309.

Data Availability The fluid datasets used for the examples in this publication are available either with the software package itself (at https://github.com/mountaindust/Planktos) or in the Google Drive repository cited within the examples directory of the software package.

References

Ahrens J, Geveci B, Law C (2005) Paraview: an end-user tool for large-data visualization. In: Hansen CD, Johnson CR (eds) Visualization handbook. Butterworth-Heinemann, Burlington

Ahrens J, Geveci B, Law C (2005) Paraview: an end-user tool for large data visualization. The visualization handbook 717(8)

Baird A, Babcock R, Mundy C (2003) Habitat selection by larvae influences the depth distribution of six common coral species. Mar Ecol Prog Ser 252:289–293

Baker KL, Dickinson M, Findley TM, Gire DH, Louis M, Suver MP, Verhagen JV, Nagel KI, Smear MC (2018) Algorithms for olfactory search across species. J Neurosci 38(44):9383–9389

Battista NA, Strickland WC, Miller LA (2017) Ib2d: a python and matlab implementation of the immersed boundary method. Bioinspir Biomim 12(3):036003

Battista N, Strickland C, Barrett A, Miller LA (2017) Ib2d reloaded: a more powerful python and matlab implementation of the immersed boundary method. Math Methods Appl Sci. https://doi.org/10.1002/mma.4708

Bohonak AJ, Jenkins DG (2003) Ecological and evolutionary significance of dispersal by freshwater invertebrates. Ecol Lett 6:783–796. https://doi.org/10.1046/j.1461-0248.2003.00486.x

Borazjani I (2020) Numerical simulations of flow around copepods: challenges and future directions. Fluids 5(2)

Boudina M, Gosselin F, Étienne S (2021) Vortex-induced vibrations: A soft coral feeding strategy? J Fluid Mech 916:A50. https://doi.org/10.1017/jfm.2021.252

Boyer D, Kaufman S, Gunnell G, Gomes E, Thostenson J (2014) Morphosource: a currently active project-based 3d digital web-accessible data archive for museums and individuals at due university usa. Digital Specimen Abstracts of Presentations

Brinkman HC (1949) A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles. Appl Sci Res 1:27–34

Broström G (2002) On advection and diffusion of plankton in coarse resolution ocean models. J Mar Syst 35:99–110

Buskey E, Len PH, Hartline D (2002) Escape behavior of planktonic copepods in response to hydrodynamic disturbances: high speed video analysis. Mar Ecol Progress Ser. Vol. 235

Carturan BS, Pither J, Maréchal JP, Bradshaw CJ, Parrott L (2020) Combining agent-based, trait-based and demographic approaches to model coral-community dynamics. eLife 9:e55993

Cheer A, Koehl M (1987) Paddles and rakes: fluid flow through bristled appendages of small organisms. J Theor Biol 129:17–39

Childs H, Brugger E, Whitlock B, Meredith J, Ahern S, Pugmire D, Biagas K, Miller MC, Harrison C, Weber GH, Krishnan H, Fogal T, Sanderson A, Garth C, Bethel EW, Camp D, Rubel O, Durant M, Favre JM, Navratil P (2012) VisIt: an end-user tool for visualizing and analyzing very large data. https://doi.org/10.1201/b12985. https://visit.llnl.gov

Childs H, Brugger E, Whitlock B, Meredith J, Ahern S, Pugmire D, Biagas K, Miller M, Harrison C, Weber GH, Krishnan H, Fogal T, Sanderson A, Garth C, Bethel EW, Camp D, Rübel O, Durant M, Favre JM, Navrátil P (2012) Visit: an end-user tool for visualizing and analyzing very large data. In: High Performance Visualization–Enabling Extreme-Scale Scientific Insight, pp. 357–372

COMSOL Multiphysics®. www.comsol.com. COMSOL AB Stockholm, Sweden

Couzin ID (2009) Collective cognition in animal groups. Trends Cogn Sci 13(1):36–43. https://doi.org/10.1016/j.tics.2008.10.002

Cowen RK, Sponaugle S (2009) Larval dispersal and marine population connectivity. Annu Rev Mar Sci 1:443–466. https://doi.org/10.1146/annurev.marine.010908.163757

Dabiri, J.: Lcs matlab kit. https://dabirilab.com/software (2021). Accessed: 2021-08-26

Dagestad KF, Röhrs J, Breivik Ø, Ådlandsvik B (2018) Opendrift v1. 0: a generic framework for trajectory modelling. Geosci Model Dev 11(4):1405–1420

Daigle RM, Metaxas A, deYoung B (2014) Bay-scale patterns in the distribution, aggregation and spatial variability of larvae of benthic invertebrates. Mar Ecol-Prog Ser 503:139–156. https://doi.org/10.3354/meps10734

Darrouzet-Nardi A, Hoopes MF, Walker JD, Briggs CJ (2006) Dispersal and foraging behaviour of Platygaster californica: hosts can't run, but they can hide. Ecol Entomol 31(4):298–306. https://doi.org/10.1111/j.1365-2311.2006.00798.x

Defina A, Bixio AC (2005) Mean flow and turbulence in vegetated open channel flow. Water Resour Res. 41(7)

DeSalvo GJ, Swanson JA (1985) ANSYS engineering analysis system user's manual. Swanson Analysis Systems, Houston, PA

Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, development team, J (2016) Jupyter notebooks - a publishing format for reproducible computational workflows. In: Loizides F, Scmidt B (eds) Positioning and power in academic publishing: players, agents and agendas. IOS Press, Netherlands

Digitalmorphology: a national science foundation digital library at the university of texas at austin. http://digimorph.org/index.phtml (2019)

Dingle H (2014) Migration: the biology of life on the move, 2nd edn. Oxford University Press, Oxford

Dormand J, Prince P (1980) A family of embedded Runge-Kutta formulae. J Comput Appl Math 6(1):19–26

D'Orsogna MR, Chuang YL, Bertozzi AL, Chayes LS (2006) Self-propelled particles with soft-core interactions: patterns, stability, and collapse. Phys Rev Lett 96(104302):1–4

Falcucci G, Amati G, Fanelli P.e.a (2021) Extreme flow simulations reveal skeletal adaptations of deep-sea sponges. Nature 595:537–541

Finnigan JJ, Belcher SE (2004) Flow over a hill covered with a plant canopy. Q J R Meteorol Soc 130(596):1–29. https://doi.org/10.1256/qj.02.177

Gao T, Betterton MD, Jhang A, Shelley MJ (2017) Analytical structure, dynamics, and coarse graining of a kinetic model of an active fluid. Phys Rev Fluids 2:093302

Giardina I (2008) Collective behavior in animal groups: theoretical models and empirical studies. HFSP J 2(4):205–219

Green S, Visser A, Titelman J.e.a (2003) Escape responses of copepod nauplii in the flow field of the blue mussel, mytilus edulis. Mar Biol 142:727–733. https://doi.org/10.1007/s00227-002-0996-1

Griffith BE (2019). An adaptive and distributed-memory parallel implementation of the immersed boundary (ib) method. https://code.google.com/p/ibamr/. Accessed: 2019-05-20

Griffith BE, Hornung RD, McQueen DM, Peskin CS (2007) An adaptive, formally second order accurate version of the immersed boundary method. J Comput Phys 223(1):10–49

Grimm T (2004) User's guide to rapid prototyping. Society of Manufacturing Engineers

Grünbaum D, Strathmann RR (2003) Form, performance and trade-offs in swimming and stability of armed larvae. J Mar Res 61:659–691. https://doi.org/10.1357/002224003771815990

Haller G (2021) Nonlinear dynamical systems group: Lcs tool. http://georgehaller.com/software/software.html. Accessed: 2021-08-26

Haller G (2001) Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Phys D 149(4):248–277

Haller G (2002) Lagrangian coherent structures from approximate velocity data. Phys Fluids 14(6):1851
Haller G, Sapsis T (2011) Lagrangian coherent structures and the smallest finite-time lyapunov exponent. Chaos 21:023115
Haller G, Yuan GC (2000) Lagrangian coherent structures and mixing in two-dimensional turbulence. Phys D 147(3):352–370
Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, Fernàndez del Río J, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with NumPy. Nature 585:357–362. https://doi.org/10.1038/s41586-020-2649-2
Hata T, Madin JS, Cumbo VR, Denny M, Figueiredo J, Harii S, Thomas CJ, Baird AH (2017) Coral larvae are poor swimmers and require fine-scale reef structure to settle. Sci Rep 7(1):2249
Hawkins S, Allcock A, Bates A, Firth L, Smith I, Swearer S, Todd P (2019) A review of biophysical models of marine larval dispersal. Ann Rev Oceanogr Mar Biol 57:325–356
Hay ME (2009) Marine chemical ecology: chemical signals and cues structure marine populations, communities, and ecosystems. Ann Rev Mar Sci 1:193–212
Hoover AP, Xu NW, Gemmell BJ, Colin SP, Costello JH, Dabiri JO, Miller LA (2021) Neuromechanical wave resonance in jellyfish swimming. PNAS 118(11)
Hoover AP, Miller LA (2015) A numerical study of the benefits of driving jellyfish bells at their natural frequency. J Theor Biol 374:13–25
Hossain MM, Staples AE (2020) Effects of coral colony morphology on turbulent flow dynamics. PLoS ONE 15:1–25. https://doi.org/10.1371/journal.pone.0225676
Howe HF, Miriti MN (2004) When seed dispersal matters. Bioscience 54:651–660. https://doi.org/10.1641/0006-3568(2004)054[0651:wsdm]2.0.co;2
Hunter JD (2007) Matplotlib: a 2d graphics environment. Comput Sci Eng 9(3):90–95. https://doi.org/10.1109/MCSE.2007.55
Huth A, Drechsler M, Köhler P (2004) Multicriteria evaluation of simulated logging scenarios in a tropical rain forest. J Environ Manage 71(4):321–333
Jakobsen HH (2001) Escape response of planktonic protists to fluid mechanical signals. Mar Ecol Prog Ser 214:67–78
Johnson DT, Croft BA (1976) Laboratory study of the dispersal behavior of Amblyseius fallacis (Acarina: Phytoseiidae). Ann Entomol Soc Am 69(6):1019–1023. https://doi.org/10.1093/aesa/69.6.1019
Jones SK, Yun YJJ, Hedrick TL, Griffith BE, Miller LA (2016) Bristles reduce the force required to 'fling' wings apart in the smallest insects. J Exp Biol 219(23):3759–3772
Jørgensen CB (1955) Quantitative aspects of filter feeding in invertebrates. Biol Rev 30(4):391–453
Kim IS, Dickinson MH (2017) Idiothetic path integration in the fruit fly drosophila melanogaster. Curr Biol 27(15):2227–2238
Kleypas JA, Thompson DM, Castruccio FS, Curchitser EN, Pinsky M, Watson JR (2016) Larval connectivity across temperature gradients and its potential effect on heat tolerance in coral populations. Glob Change Biol 22(11):3539–3549
Kloeden PE, Platen E (1999) Numerical solution of stochastic differential equations, 3rd edn. Springer-Verlag, Berlin
Koehl MAR, Hadfield MG (2004) Soluble settlement cue in slowly-moving water within coral reefs induces larval adhesion to surfaces. J Mar Syst 49:75–88. https://doi.org/10.1016/j.jmarsys.2003.06.003
Koehl MAR, Strother JA, Reidenbach MA, Koseff JR, Hadfield MG (2007) Individual-based model of larval transport to coral reefs in turbulent, wave-driven flow: behavioral responses to dissolved settlement inducer. Mar Ecol Prog Ser 335:1–18. https://doi.org/10.3354/meps335001
Krick J, Ackerman J (2015) Adding ecology to particle capture models: numerical simulations of capture on a moving cylinder in crossflow. J Theor Biol 368:13–26
Kristensen NP, Schellhorn NA, Hulthen AD, Howie LJ, De Barro PJ (2013) Wind-borne dispersal of a parasitoid: the process, the model, and its validation. Environ Entomol 42(6):1137–1148. https://doi.org/10.1603/en12243
Kurtuldu H, Guasto JS, Johnson KA, Gollub JP (2011) Enhancement of biomixing by swimming algal cells in two-dimensional films. Proc Natl Acad Sci 108(26):10391–10395. https://doi.org/10.1073/pnas.1107046108
Lacharite M, Metaxas A (2013) Early life history of deep-water gorgonian corals may limit their abundance. PLoS One 8(6):e65394

Leis JM (2007) Behaviour as input for modelling dispersal of fish larvae: behaviour, biogeogrpahy, hydrodynamics, ontogeny, physiology and phylogeny meet hydrography. Mar Ecol-Prog Ser 347:185–193. https://doi.org/10.3354/meps06977
Levin LA (2006) Recent progress in understanding larval dispersal: new directions and digressions. Integr Comp Biol 46(3):282–297. https://doi.org/10.1093/icb/icj024
Li C, Dong H, Zhao K (2018) A balance between aerodynamic and olfactory performance during flight in drosophila. Nat Commun 9(1):3215
Liedloff AC, Cook GD (2007) Modelling the effects of rainfall variability and fire on tree populations in an Australian tropical savanna with the FLAMES simulation model. Ecol Model 201:269–282
Liu Y (2017) Project lcs: Lagrangian coherent structure analysis. https://stevenliuyi.github.io/lcs/. Accessed: 2021-08-26
McGillicuddy DJ, Franks PJ (2019) Models of plankton patchiness. In: Cochran JK, Bokuniewicz HJ, Yager PL (eds) Encyclopedia of ocean sciences, 3rd edn. Academic Press, Oxford
McKenna JE Jr, Chalupnicki MA (2011) A heuristic simulation model of Lake Ontario circulation and mass balance transport. J. Freshwater Ecol 26(1):123–132. https://doi.org/10.1080/02705060.2011.553928
McManus M, Woodson C (2012) Plankton distribution and ocean dispersal. J Exp Biol 215:1008–1016. https://doi.org/10.1242/jeb.059014
McManus LC, Vasconcelos VV, Levin SA, Thompson DM, Kleypas JA, Castruccio FS, Curchitser EN, Watson JR (2020) Extreme temperature events will drive coral decline in the coral triangle. Glob Change Biol 26(4):2120–2133
Miller LA, Peskin CS (2004) When vortices stick: an aerodynamic transition in tiny insect flight. J Exp Biol 207(17):3073–3088. https://doi.org/10.1242/jeb.01138
Mil'shtejn GN (1973) Approximate integration of stochastic differential equations. Theor Probab Appl 19:557–562
Morehead S, Feener D (2000) Visual and chemical cues used in host location and acceptance by a dipteran parasitoid. J Insect Behav 13:613–625
Moreland K (2021) Vtk-m users' guide, version 1.7. Tech. rep., Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States)
Morse DE, Hooker N, Morse AN, Jensen RA (1988) Control of larval metamorphosis and recruitment in sympatric agariciid corals. J Exp Mar Biol Ecol 116(3):193–217
Mundy C, Babcock R (1998) Role of light intensity and spectral quality in coral settlement: Implications for depth-dependent settlement? J Exp Mar Biol Ecol 223(2):235–255
Nave G. Manifoldid. https://github.com/gknave/ManifoldID (2018). Accessed: 2021-08-26
Netlogo references (2018). https://ccl.northwestern.edu/netlogo/references.shtml. Accessed: 2018-09-03
Niklas KJ (1985) The aerodynamics of wind pollination. Bot Rev 51:328
Niklas KJ (1985) Wind pollination-a study in controlled chaos: aerodynamic studies of wind-pollinated plants reveal a high degree of control in the apparently random process of pollen capture. Am Sci 73:462–470
Nikora V (2010) Hydrodynamics of aquatic ecosystems: an interface between ecology, biomechanics and environmental fluid mechanics. River Res Appl 26(4):367–384
Nilsen C, Paige J, Warner O, Mayhew B, Sutley R, Lam M, Bernoff AJ, Topaz CM (2013) Social aggregation in pea aphids: experiment and random walk modeling. PLoS ONE 8(12):1–11. https://doi.org/10.1371/journal.pone.0083343
O'Brien JJ, Wroblewski JS (1973) On advection in phytoplankton models. J Theor Biol 38(1):197–202
of Health NI (2022) Nih 3d print exchange. https://3dprint.nih.gov/
Oza AU, Ristroph L, Shelley MJ (2019) Lattices of hydrodynamically interacting flapping swimmers. PHYSICAL REVIEW X
Ozalp MK, Miller LA, Dombrowski T, Braye M, Dix T, Pongracz L, Howell R, Klotsa D, Pasour V, Strickland C (2020) Experiments and agent based models of zooplankton movement within complex flow environments. Biomimetics 5(1):2
pandas development team T (2020) pandas-dev/pandas: Pandas. https://doi.org/10.5281/zenodo.3509134
Pang R, van Breugel F, Dickinson M, Riffell JA, Fairhall A (2018) History dependence in insect flight decisions during odor tracking. PLOS Comp Biol. https://doi.org/10.1371/journal.pcbi.1005969
Pepper RJ, Jaffe S, Variano E, Koehl MAR (2015) Zooplankton in flowing water near benthic communities encounter rapidly fluctuating velocity gradients and accelerations. Mar Biol 162:1939–1954

Potting RPJ, Perry JN, Powell W (2005) Insect behavioural ecology and other factors affecting the control efficacy of agro-ecosystem diversification strategies. Ecol Model 182(2):199–216. https://doi.org/10.1016/j.ecolmodel.2004.07.017

Qhull (2020). www.qhull.org

R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2017). https://www.R-project.org/

Railsback SF, Grimm V (2012) Agent-based and individual-based modeling: a practical introduction. Princeton University Press, Princeton

Railsback SF, Lytinen SL, Jackson SK (2006) Agent-based simulation platforms: review and development recommendations. Simulation 82(9):609–623. https://doi.org/10.1177/0037549706073695

Ramananarivo S, Fang F, Oza A, Zhang J, Ristroph L (2016) Flow interactions lead to orderly formations of flapping wings in forward flight. Phys Rev Fluids 1:071201

Reynolds AM, Reynolds DR (2009) Aphid aerial density profiles are consistent with turbulent advection amplifying flight behaviours: abandoning the epithet 'passive'. Proc R Soc Lond B 276(1654):137–143. https://doi.org/10.1098/rspb.2008.0880

Reynolds DR, Chapman JW, Edwards AS, Smith AD, Wood CR, Barlow JF, Woiwod IP (2005) Radar studies of the vertical distribution of insects migrating over southern Britain: the influence of temperature inversions on nocturnal layer concentrations. Bull Entomol Res 95(3):259–274

Riisgard HU, Larsen P (1995) Filter-feeding in marine macro-invertebrates: pump characteristics, modelling and energy cost. Biol Rev Camb Philos Soc 70:67–106. https://doi.org/10.1111/j.1469-185X.1995.tb01440.x

Rossi S, Shubert N, Brown D, de Oliveira Soares M, Grosso V, Rangel-Huerta E, Maldonado E (2018) Linking host morphology and symbiont performance in octocorals. Sci Rep 8:12823-1–14

Samson JE, Miller LA, Ray D, Holzman R, Shavit U, Khatri S (2019) A novel mechanism of mixing by pulsing corals. J Exp Biol. 222(15)

Santhanakrishnan A, Jones SK, Dickson WB, Peek M, Kasoju VT, Dickinson MH, Miller LA (2018) Flow structure and force generation on flapping wings at low Reynolds numbers relevant to the flight of tiny insects. Fluids 3:45

Sarig A, Ribak G (2021) To what extent can the tiny parasitoid wasps, eretmocerus mundus, fly upwind? J Appl Entomol 145(7):660–674. https://doi.org/10.1111/jen.12890

Schroeder WJ, Avila LS, Hoffman W (2000) Visualizing with vtk: a tutorial. IEEE Comput Graphics Appl 20(5):20–27

Schroeder W, Martin K, Lorensen B (2006) The visualization toolkit, 4th edn. Kitware, New York

Schulze J, Müller B, Groeneveld J, Grimm V (2017) Agent-based modelling of social-ecological systems: achievements, challenges, and a way forward. J Artif Soc Soc Simul 20(2):8

Scott D (1992) Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons, Chicester, New York

Shadden S (2005) Lagrangian coherent structures: analysis of time-dependent dynamical systems using finite-time Lyapunov exponents. https://shaddenlab.berkeley.edu/uploads/LCS-tutorial/. Accessed: 2021-07-16

Shimeta J, Koehl M (1997) Mechanisms of particle selection by tentaculate suspension feeders during encounter, retention, and handling. J Expl Mar Biol Ecol 209:47–73

Simoncelli S, Thackeray SJ, Wain DJ (2017) Can small zooplankton mix lakes? Limnol Oceanogr Lett 2(5):167–176

Strickland C (2018) Planktos agent-based modeling framework. https://github.com/mountaindust/Planktos. Accessed: 2019-05-20

Strickland C, Kristensen NP, Miller LA (2017) Inferring stratified parasitoid dispersal mechanisms and parameters from coarse data using mathematical and bayesian methods. R Soc Interface 14:20170005

Strickland C, Miller L, Santhanakrishnan A, Hamlet C, Battista NA, Pasour V (2017) Three-dimensional low reynolds number flows near biological filtering and protective layers. Fluids 2:62

Sunday D (2021) Practical Geometry Algorithms with C++ Code. Self-published, Amazon KDP

Taylor AD (1990) Metapopulations, dispersal, and predator-prey dynamics: an overview. Ecology 71:429–433. https://doi.org/10.2307/1940297

Thompson DM, Kleypas J, Castruccio F, Curchitser EN, Pinsky ML, Jönsson B, Watson JR (2018) Variability in oceanographic barriers to coral larval dispersal: Do currents shape biodiversity? Prog Oceanogr 165:110–122

Topaz CM, Bernoff AJ, Logan S, Toolson W (2008) A model for rolling swarms of locusts. Eur Phys J Spec Top 157(1):93–109. https://doi.org/10.1140/epjst/e2008-00633-y
Treml EA, Roberts JJ, Chao Y, Halpin PN, Possingham HP, Riginos C (2012) Reproductive output and duration of the pelagic larval stage determine seascape-wide connectivity of marine populations. Integr Comp Biol 52(4):525–537. https://doi.org/10.1093/icb/ics101
Triantafyllou MS, Weymouth GD, Miao J (2016) Biomimetic survival hydrodynamics and flow sensing. Ann Rev Fluid Mech 48(1):1–24
Tytell ED, Hsu CY, Fauci LJ (2014) The role of mechanical resonance in the neural control of swimming in fishes. Zoology 117(1):48–56
van Breugel F, Dickinson MH (2014) Plume-tracking behavior of flying drosophila emerges from a set of distinct sensory-motor reflexes. Curr Biol 24(3):274–286
Vance RR (1973) On reproductive strategies in marine benthic invertebrates. Am Nat 107(955):339–352. https://doi.org/10.1086/282838
Verma S, Novati G, Koumoutsakos P (2018) Efficient collective swimming by harnessing vortices through deep reinforcement learning. Proc Natl Acad Sci 115(23):5849–5854. https://doi.org/10.1073/pnas.1800923115
Vicsek T, Czirók A, Ben-Jacob E, Cohen I, Shochet O (1995) Novel type of phase transition in a system of self-driven particles. Phys Rev Lett 75(6):1226–1229
Villarino E, Watson JR, Jönsson B, Gasol JM, Salazar G, Acinas SG, Estrada M, Massana R, Logares R, Giner CR, Pernice MC, Olivar MP, Citores L, Corell J, Rodríguez-Ezpeleta N, Acuña JL, Molina-Ramírez A, González-Gordillo JI, Cózar A, Martí E, Cuesta JA, Agustí S, Fraile-Nuez E, Duarte CM, Irigoien X, Chust G (2018) Large-scale ocean connectivity and planktonic body size. Nat Commun 9(1):142
Villarreal B, Olague G, Gordillo J (2014) Odor plume tracking algorithm inspired on evolution. In: Martínez-Trinidad J, Carrasco-Ochoa J, Olvera-Lopez J, Salas-Rodríguez J, Suen C (eds) Pattern recognition, lecture notes in computer science, vol 8495. Springer, Berlin, pp 321–330
Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P (2020) SciPy 1.0 contributors: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods 17:261–272
Visser AW (2007) Biomixing of the oceans? Science 316(5826):838–839
Wainwright SA, Dillon JR (1969) On the orientation of sea fans (genus gorgonia). Biol Bull 136(1):130–139
Weller HG, Tabor G, Jasak H, Fureby C (1998) A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput Phys 12(6):620–631
Wes McKinney: data structures for statistical computing in Python. In: Stéfan van der Walt, Jarrod Millman (eds.) Proceedings of the 9th Python in Science Conference, pp. 56–61 (2010). https://doi.org/10.25080/Majora-92bf1922-00a
Wickramarathna LN, Noss C, Lorke A (2014) Hydrodynamic trails produced by daphnia: size and energetics. PLoS ONE 9(3):1–10. https://doi.org/10.1371/journal.pone.0092383
Willis J (2011) Modelling swimming aquatic animals in hydrodynamic models. Ecol Model 222(23–24):3869–3887
Winston JE (2012) Dispersal in marine organisms without a pelagic larval phase. Integr Comp Biol 52(4):447–457. https://doi.org/10.1093/icb/ics040