Integrative and Comparative Biology

Integrative and Comparative Biology, volume 55, number 5, pp. 901–911
doi:10.1093/icb/icv102

Society for Integrative and Comparative Biology

SYMPOSIUM

A Mathematical Model and MATLAB Code for Muscle–Fluid–Structure Simulations

Nicholas A. Battista,* Austin J. Baird† and Laura A. Miller[1,‡]

*Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599, USA; †Department of Mathematics, Duke University, Durham, NC 27708, USA; ‡Departments of Biology and Mathematics, University of North Carolina, Chapel Hill, NC 27599, USA

From the symposium “Leading Students and Faculty to Quantitative Biology Through Active Learning” presented at the annual meeting of the Society for Integrative and Comparative Biology, January 3–7, 2015 at West Palm Beach, Florida.

[1]E-mail: lam9@email.unc.edu

Synopsis This article provides models and code for numerically simulating muscle–fluid–structure interactions (FSIs). This work was presented as part of the symposium on *Leading Students and Faculty to Quantitative Biology through Active Learning* at the society-wide meeting of the Society for Integrative and Comparative Biology in 2015. Muscle mechanics and simple mathematical models to describe the forces generated by muscular contractions are introduced in most biomechanics and physiology courses. Often, however, the models are derived for simplifying cases such as isometric or isotonic contractions. In this article, we present a simple model of the force generated through active contraction of muscles. The muscles’ forces are then used to drive the motion of flexible structures immersed in a viscous fluid. An example of an elastic band immersed in a fluid is first presented to illustrate a fully-coupled FSI in the absence of any external driving forces. In the second example, we present a valveless tube with model muscles that drive the contraction of the tube. We provide a brief overview of the numerical method used to generate these results. We also include as Supplementary Material a MATLAB code to generate these results. The code was written for flexibility so as to be easily modified to many other biological applications for educational purposes.

Introduction

Conceptual and mathematical models describing muscular contraction are a standard topic in introductory biology, physiology, and mathematical biology courses (Schmidt-Nielsen 1997; Keener and Sneyd 1998; Vogel 2013). Often these models are introduced in the context of an isolated muscle performing isometric or isotonic contractions. The integration of muscle models within an organ or organismal system that include changes in velocities and loads can provide additional insights into the implications of the dynamics of contractions.

The ways that organisms pump fluid, swim, and fly are also standard topics in physiology and biomechanics courses (Vogel 1996; Schmidt-Nielsen 1997). It is often noted that biological materials are flexible, and the large deformations of these structures can have significant implications for the performance of the organisms. For example, the deformations of flexible fish fins (Clark and Smits 2006; Shoele 2008) and flexible jellyfish bells (Gemmell et al. 2013) can enhance swimming performance, and the dynamics of flexible leaflets on heart valves are important for the proper transport of blood through the heart (Griffith et al. 2009). Mathematical models that integrate flexible structures and fluids are, however, difficult to study in the classroom because analytical solutions are typically not available.

In this article, we incorporate a mathematical model of muscle mechanics into a numerical simulation of a flexible structure immersed in a fluid. We have developed MATLAB software written for versatility and ease of use rather than for computational performance. We have made the program flexible enough so that interested students and faculty can modify the code for a variety of

Advanced Access publication September 3, 2015

applications. We provide two simple examples to illustrate the interaction of fluid–structure interaction (FSI) and muscle mechanics. The first example is a simple elastic band immersed in a fluid that resists stretching. The second example is an elastic tube with muscles that drive a periodic contraction.

Methods

Simple models of muscle mechanics

One of the earliest mathematical models of the forces generated by muscular contraction dates back to Hill (1938). Since then, the Hill model has been extended to consider additional features of muscles, such as the compliance of filaments (Goldman and Huxley 1994; Campbell 2006) and the formation of individual myosin cross-bridges (Huxley 1957; Keener and Sneyd 1998). Here, for purposes of illustration, we present one simple muscle model that incorporates many of the salient properties of muscles.

The force a muscle can exert depends upon how fast it shortens as well as the length of the muscle while it contracts. Let's begin by first describing a model that captures the force generated as a function of the muscle's velocity of contraction. In general, the faster the muscle shortens, the less force it can exert. The Hill model is commonly used to describe this force–velocity relationship (Hill 1938; Fung 1993) and is given by the following equation:

$$V_{\mathrm{F}} = \frac{b(F_{\max} - F)}{F + a}, \tag{1}$$

where V_{F} is the shortening velocity of the muscle fiber, F is the force exerted by the fiber, and $F_{\max}$ is the maximum load with zero velocity of contraction. The constants a and b may be determined experimentally, and note that $V_{\max} = b\, F_{\max}/a$ is the maximum velocity under no load. An isometric contraction refers to the scenario when the velocity of contraction is zero. An isotonic contraction refers to the case when the tension generated remains unchanged as the muscle's length changes. The maximum load possible for a given velocity may be found on the force velocity curve.

It is also well known that the maximum force a muscle can generate is a function of its length. When the muscle is stretched, only a fraction of the myosin heads on each thick filament can reach a thin filament. In this case, the active force that can be generated is small. The maximum force is typically generated at intermediate lengths when all of the myosin heads are within reach of the thin filaments. With further shortening, the thick filaments interfere with the *Z*-disks, and the force generated falls rapidly. A simple model describing the length–tension relationship of muscle is as follows (Hatze 1981):

$$F_{\mathrm{I}} = F_{\mathrm{IO}} \exp\left[-\left(\frac{Q-1}{\mathrm{SK}}\right)^2\right], \quad Q = \frac{L_{\mathrm{F}}}{L_{\mathrm{FO}}}, \tag{2}$$

where F_{I} is the maximum isometric tension at a given length, F_{IO} is the maximum isometric force produced at the optimum length of the muscle fibers, L_{F} is the length of the muscle fibers, L_{FO} is the length at which the muscle fibers exert their maximum tension, and SK is a constant specific for each muscle where $\mathrm{SK} > 0.28$.

A simple model that combines the length–tension and force–velocity profiles is derived by simply taking the product of these normalized relationships (Challis and Kerwin 1994). Let's divide Equations (1) and (2) by $F_{\max}$ and F_{IO}, respectively. Let a_f represent the activation strength of the muscle which varies from 0 to 1. Also let $\tilde{F}_{\max}$ describe the maximum isometric force produced at the optimum length of the muscle fibers under full activation. Then the model describing the force the muscle generates for a given length and velocity, $F_{\mathrm{M}}(L_{\mathrm{F}}, V_{\mathrm{F}})$, can be written as

$$F_{\mathrm{M}} = a_f \tilde{F}_{\max} F_1(L_{\mathrm{F}}) F_2(V_{\mathrm{F}}), \tag{3}$$

where $F_1(L_{\mathrm{F}})$ and $F_2(V_{\mathrm{F}})$ are given by

$$F_1(L_{\mathrm{F}}) = \exp\left[-\left(\frac{(L_{\mathrm{F}}/L_{\mathrm{FO}}) - 1}{\mathrm{SK}}\right)^2\right], \text{ and} \tag{4}$$

$$F_2(V_{\mathrm{F}}) = \frac{1}{F_{\max}}\left(\frac{bF_{\max} - aV_{\mathrm{F}}}{V_{\mathrm{F}} + b}\right). \tag{5}$$

The immersed boundary method

Since its introduction by Peskin over 40 years ago (Peskin 1972), the immersed boundary (IB) method has been used successfully to model a variety of problems in the dynamics of biological fluids. Examples include insect flight (Miller and Peskin

2009), lamprey swimming (Tytell et al. 2010), cardiac blood flow (Peskin and McQueen 1996), cell deformation by shear flows (Bottino 1998), and the formation of blood clots by the aggregation of platelets (Skorczewski et al. 2014).

One of the main advantages of the IB method is that it is a straightforward way to compute FSIs in the sense that a uniform computational grid can be used to solve the equations of fluid motion with a standard fluid solver. The IB is represented by a collection of Lagrangian markers that move independently from the grid, and the effect of the motion of the boundary is transferred to the grid through a local stencil near each marker point that is simple to implement. The IB method does not require a non-uniform or moving mesh.

We provide a short overview of the mathematical formulation of the IB method and then provide some intuition for its implementation. The equations of two-dimensional (2D) motion for the fluid are given by the Navier–Stokes equations in Eulerian form:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}(\mathbf{x}, t)\cdot\nabla\mathbf{u}(\mathbf{x}, t)\right) = -\nabla \mathrm{p}(\mathbf{x}, t) + \mu\Delta\mathbf{u}(\boldsymbol{x}, t) + \mathbf{f}(\mathbf{x}, t), \quad (6)$$

$$\nabla\cdot\mathbf{u}(\mathbf{x}, t) = 0, \quad (7)$$

where $\mathbf{u}(\mathbf{x}, t)$ is the velocity of the fluid, $\mathrm{p}(\mathbf{x}, t)$ is the pressure, $\mathbf{f}(\mathbf{x}, t)$ is the force per unit area applied to the fluid by the IB, ρ is the density of the fluid, and μ is the dynamic viscosity of the fluid. The independent variables are the time t and the position $\mathbf{x}$. Figure 1 shows the IB defined on a curvilinear mesh immersed in a fluid defined on a fixed Cartesian grid. Note that bold letters represent vector quantities, lower-case letters represent Eulerian variables, and upper-case letters represent Lagrangian variables. Equation (7) is the condition that the fluid is incompressible.

The interaction and forcing equations between the fluid and the boundary are then given by:

$$\mathbf{f}(\mathbf{x}, t) = \int \mathbf{F}(n, t)\delta(\mathbf{x} - \mathbf{X}(n, t))\mathrm{d}n, \quad (8)$$

$$\frac{\partial \mathbf{X}(n, t)}{\partial t} = \mathbf{U}(\mathbf{X}(n, t)) = \int \mathbf{u}(\mathbf{x}, t)\delta(\mathbf{x} - \mathbf{X}(n, t))\mathrm{d}\mathbf{x}, \quad (9)$$

where $\mathbf{F}(n, t)$ is the force per unit length applied by the boundary to the fluid as a function of Lagrangian position and time, $\delta(\mathbf{x})$ is a 2D delta function, $\mathbf{X}(n, t)$ gives the Cartesian coordinates at time t of the material point labeled by the Lagrangian parameter n. Equation (8) applies force from the boundary to the fluid grid, and Equation (9) evaluates the velocity of the local fluid at the boundary. The boundary is then moved at the local fluid's velocity, and this enforces the no-slip condition. Each of these equations involves a 2D Dirac delta function δ, which acts in each case as the kernel of an integral transformation. These equations convert Lagrangian variables to Eulerian variables and vice versa.

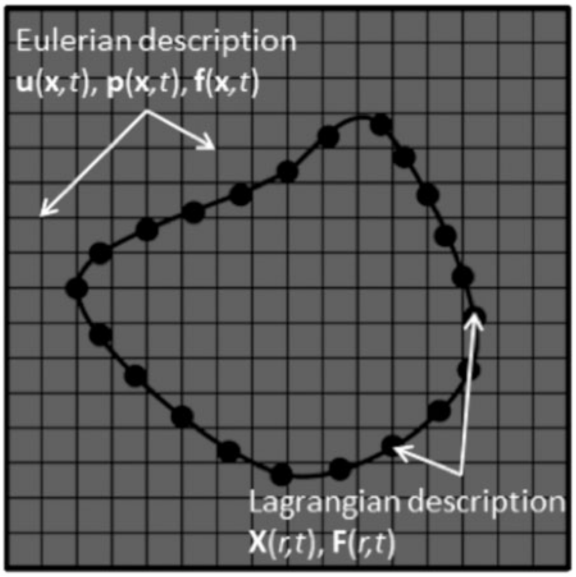


Fig. 1 An **IB** discretized on a curvilinear mesh immersed in a fluid discretized on a fixed Cartesian grid. An Eulerian description is used to describe the fluid where **u** is the velocity, p is the pressure, and **f** is the force's density. A Lagrangian description is used to describe the **IB** where **X** is the position of the boundary point labeled *r* and **F** is the force per unit length on the boundary.

Smoothed delta functions for force spreading and interpolation of velocity

Since the Lagrangian array of boundary points does not necessarily coincide with the fixed Eulerian lattice used for the computation of the fluid velocities, a smoothed approximation to the Dirac delta function is used to handle the fluid–boundary interaction. This approximate delta function is used to apply a force from the IB to the underlying fluid grid in a specific region of the Eulerian lattice, whose points are within two nodal points of the Lagrangian structure. The approximate delta function is also used to interpolate the fluid's velocity onto the Lagrangian structure, by considering only the velocity of Eulerian nodal points within two lattice points of the IB. The approximate 2D Dirac delta function, $\boldsymbol{\delta}_{\Delta x}$, used in these calculations is given by the following equations:

$$\boldsymbol{\delta}_{\Delta x}(\mathbf{x}) = \Delta x^{-2}\phi\left(\frac{x}{\Delta x}\right)\phi\left(\frac{y}{\Delta x}\right), \tag{10}$$

$$\phi(r) = \begin{cases} \frac{1}{4}\left(1+\cos\left(\frac{\pi r}{2}\right)\right), & |r| \leq 2 \\ 0, & \text{otherwise} \end{cases}$$

where Δx is the size of the spatial step in the Eulerian grid and $\mathbf{x} = (x, y)$. Peskin and McQueen (1996) previously described this choice of the delta function. Other approximate delta functions have also been used successfully (Peskin 2002). Note that since $\boldsymbol{\delta}_{\Delta x}(\mathbf{x})$ is nonzero only in a small region centered on the boundary, and the forces are spread to a thin layer whose width decreases as the grid is refined.

The numerical algorithm

In an IB simulation, we discretize the Navier–Stokes equations and solve for the velocities and pressures at times $t_k = k\Delta t$ and positions $\boldsymbol{x}_{ij} = (i\Delta x, j\Delta y) = (x_i, y_j)$. We also discretize our boundaries and solve for the position of each Lagrangian point n with respect to the Cartesian grid at each time. We denote the position of the point labeled n at time t_k as $\boldsymbol{X}_n(t_k)$. To update the system of equations from time t_k to time t_{k+1}, we performed the following steps:

(1) Solved elasticity equations and any other equations describing the forces applied to the boundary to determine the force $\mathbf{F}_n$ at each boundary point with position $\mathbf{X}_n$.

(2) Spread the forces on the Lagrangian curvilinear mesh to the fixed Cartesian grid on which the Navier–Stokes equations are solved. The total force density $\mathbf{f}_{ij}$ on the Cartesian grid at location $\boldsymbol{x}_{ij} = (x_i, y_j)$ is given by

$$\mathbf{f}_{ij} = \sum_{n=1}^{N} \mathbf{F}_n \boldsymbol{\delta}_{\Delta x}(\mathbf{x}_{ij} - \mathbf{X}_n)\Delta n. \tag{11}$$

Note that $\boldsymbol{\delta}_{\Delta x}(\mathbf{x}_{ij} - \mathbf{X}_n)$ is the smoothed approximation to the 2D delta function that was described in the section "Smoothed delta functions for force spreading and interpolation of velocity". The spacing between the IB points is given by Δn. A visual representation of this process is given in the left panel of Fig. 2. Note that the force is spread to nearby nodes on the fixed Cartesian grid.

(3) Solved the Navier–Stokes equations using the force density $\mathbf{f}_{ij}$ to drive the fluid. Most Navier–Stokes solvers can be used for this purpose.

(4) Moved the IB points at the velocity of the local fluid, $\mathbf{u}_{ij}$, to enforce the no-slip condition. It is necessary to interpolate the grid velocity to the IB point to obtain the fluid's velocity at the boundary point, $\mathbf{U}_n$, as follows:

$$\frac{\mathbf{X}_n^{k+1} - \mathbf{X}_n^k}{\Delta t} = \mathbf{U}_n = \sum_{n=1}^{N} \mathbf{u}_{ij}\boldsymbol{\delta}_{\Delta x}(\mathbf{x}_{ij} - \mathbf{X}_n)\Delta x^2 \tag{12}$$

(5) A visual representation is given on the right panel of Fig. 2. Note that nearby nodes on the fixed Cartesian grid are used to determine the velocity at the IB point.

Any updates in target-point positions, resting curvatures, or resting spring lengths can be made at this time.

The elastic boundaries

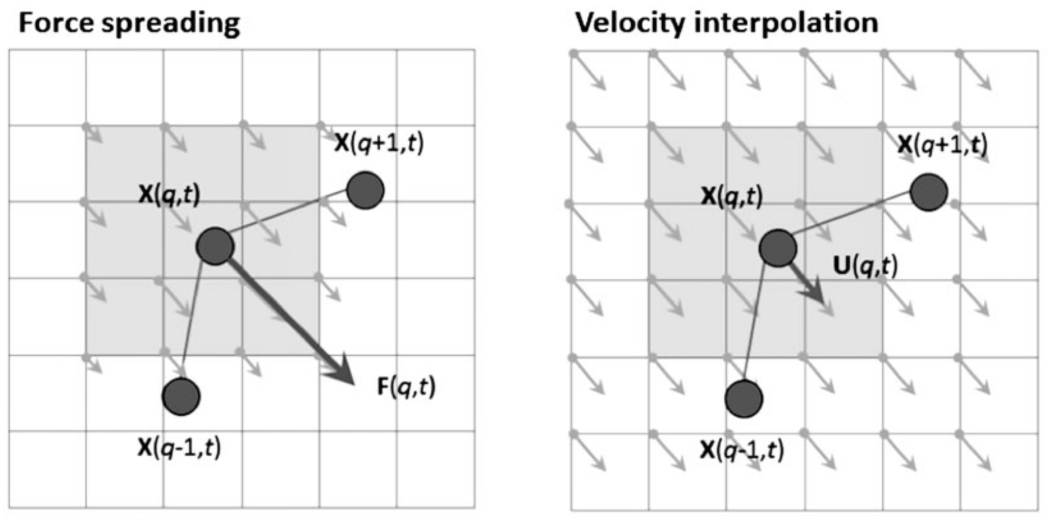
The immersed boundaries will be made of elastic springs that resist stretching (Hookean springs) and bending (torsional springs). To model the resistance to stretching, assume that elastic links connecting adjacent boundary points act as linear springs. Let boundary points m and n have the corresponding position coordinates $\mathbf{X}_m$ and $\mathbf{X}_n$, and let these points be connected by elastic link w. The stretching energy function for this link can then be written as

$$E_S(\mathbf{X}_m, \mathbf{X}_n) = \frac{1}{2}k_s(\|\mathbf{X}_m - \mathbf{X}_n\| - l_w)^2, \tag{13}$$

where l_w is the resting length of the spring and k_s is its stiffness coefficient. Note that E_S is equal to zero when the distance between the points equals the resting length.

Resistance to bending can be modeled by a pair of elastic links that emanate from the same IB point. Deviations in the angle between these links from a prescribed angle, θ, are penalized. Consider a triplet of consecutive points labeled m, n, and o with corresponding position coordinates $\mathbf{X}_m$, $\mathbf{X}_n$, and $\mathbf{X}_o$. The bending energy is then defined as

$$E_B(\mathbf{X}_m, \mathbf{X}_n, \mathbf{X}_o) = \frac{1}{2}k_b(\mathbf{z}\cdot(\mathbf{X}_o - \mathbf{X}_n)\times(\mathbf{X}_n - \mathbf{X}_m) - \kappa)^2, \tag{14}$$



$$\mathbf{f}(\mathbf{x},t) = \int \mathbf{F}(n,t)\delta\big(\mathbf{x}-\mathbf{X}(n,t)\big)\,dn$$

$$\mathbf{f}_{ij} = \sum_{n=1}^{N} \mathbf{F}_{\boldsymbol{n}} \boldsymbol{\delta}_{\Delta x}(\mathbf{x}_{\boldsymbol{ij}} - \mathbf{X}_{\boldsymbol{n}})\,\Delta n$$

$$\frac{\partial \mathbf{X}(n,t)}{\partial t} = \mathbf{U}\big(\mathbf{X}(n,t)\big) = \int \mathbf{u}(\mathbf{x},t)\delta\big(\mathbf{x}-\mathbf{X}(n,t)\big)d\mathbf{x}$$

$$\frac{\mathbf{X}_{\boldsymbol{n}}^{\boldsymbol{k+1}} - \mathbf{X}_{\boldsymbol{n}}^{\boldsymbol{k}}}{\Delta t} = \mathbf{U}_{\boldsymbol{n}} = \sum_{n=1}^{N} \mathbf{u}_{ij} \boldsymbol{\delta}_{\Delta x}(\mathbf{x}_{\boldsymbol{ij}} - \mathbf{X}_{\boldsymbol{n}})\Delta x^2$$

Fig. 2 Visual representation of force spreading and interpolation of velocity. Left panel: The three points represent a beam that resists bending. The large vector labeled **F**(*q,t*) gives the elastic force due to the resistance to bending. This force is spread to the fluid using a smoothed approximation to the delta function. The points of the fluid grid that feel this force are shaded in gray. Right panel: The fluid velocity at the Lagrangian point labeled **X**(*q,t*) is determined by taking a weighted average of local fluid grid-points highlighted in the gray box.

where k_b is the bending stiffness, κ is the prescribed curvature, and $\mathbf{z} = [0,0,1]$. κ is determined using the preferred angle, θ, and the resting lengths between the IB points, $l_{m,n}$ and $l_{n,o}$, using the following equation:

$$\kappa = l_{m,n} l_{n,o} \sin(\theta). \tag{15}$$

In some applications, it is desirable to move the boundary with some prescribed motion. In the framework of the IB, the position of the boundary is not prescribed. Preferred motion, however, can be imposed by penalizing deviations from a prescribed location that may change in time. This is done using a "tether" energy. Assume that the IB point n with coordinates $\mathbf{X}_n$ is connected to a tether point with position $\mathbf{X}_n^T$ by a linear spring of zero resting length. The resulting energy is then given as

$$E_T(\mathbf{X}_n) = \frac{1}{2} k_\mathrm{T} \big(\|\mathbf{X}_n - \mathbf{X}_n^\mathrm{T}\|\big)^2, \tag{16}$$

where k_T is the stiffness of the tether spring. The difference between the actual location of the IB point and its preferred position can be controlled with the constant k_T. Note that tether points do not interact with the fluid and are not IB points.

The total elastic energy is calculated as the sum of the stretching, bending, and target energies for each IB point. For example, if one side of a 2D heart tube is made up of a line of N IB points arranged in order so that each pair of consecutive points is joined by a linear spring that resists stretching and each consecutive triplet resists bending, then

$$E(\mathbf{X}, t) = \sum_{i=1}^{N-1} E_\mathrm{S}(\mathbf{X}_i, \mathbf{X}_{i+1}) + \sum_{i=2}^{N-1} E_\mathrm{B}(\mathbf{X}_{i-1}, \mathbf{X}_i, \mathbf{X}_{i+1}) + \sum_{i=1}^{N} E_\mathrm{T}(\mathbf{X}_i). \tag{17}$$

The elastic force at point n is then calculated using the derivatives of the elastic energy with respect to the coordinates in $\mathbf{X}_n$:

$$\mathbf{F}_n(\mathbf{X}, t) = -\frac{\partial \mathbf{E}(\mathbf{X}, t)}{\partial \boldsymbol{X}_n}. \tag{18}$$

Values of the constants k_s, k_b, κ, and l must be chosen to specify reasonable energies and forces associated with the boundary. Each of these constants can also vary in space and time. More details on this approach are provided by Peskin (2002).

The MATLAB code

The version of the MATLAB code at the time of submission of this article is available as a Supplementary File. As we continue to update the code with additional features, the most recent version will be available from https://github.com/fairyflies9/2D_IBM_MATLAB. The main folder includes an "Examples" folder and an "IBM_Blackbox" folder. The README file provides a general overview of how the code may be modified to simulate new applications. To run one of the example simulations, go into that simulation folder and type "main2d" in the MATLAB command window. You can also modify the input parameters in the *input2d* file. This allows one to change properties of the fluid, temporal information, the Eulerian grid structure, flags for the mathematical model(s) used in constructing the IB, and flags for printing and displaying output. More details on the *input2d* file are given in Appendix 2. The "IBM_Blackbox" folder includes all of the functions necessary to run the IB simulations, including functions to calculate elastic deformation forces, functions to spread the forces from the boundary to the fluid grid, the Navier–Stokes solver, and functions to interpolate the fluid velocity at the boundary and update its position.

The particular numerical scheme used in this article has been described in detail by Peskin and McQueen (1996), and more details on the method may be found in Peskin (2002). The system of integro-differential equations given by Equation (6) was solved on a rectangular grid with periodic boundary conditions in both directions. The Navier–Stokes equations were discretized on a fixed Eulerian grid, and the immersed boundaries were discretized on a Lagrangian array of points. Details of the fluid solver used are given in Appendix 1.

The computational domain for the fluid was set to 128×128 spatial steps. Note that this allows for fast simulations, but complicated boundaries and flows will require a much finer grid to accurately describe the dynamics of the fluid. The boundary was constructed such that the distance between each boundary point was set to $\mathrm{d}n = \mathrm{d}x/2$, where $\mathrm{d}x$ is the spatial step size of the fluid grid. The stiffness coefficient of the virtual springs attaching the boundary to tether points was chosen to minimize deviations from the preferred position. The time step size, $\mathrm{d}t$, was chosen to ensure stability of the numerical method. If the code is modified such that the simulation becomes unstable where very large and unrealistic velocity fields are generated, the size of the time step should likely be reduced.

Dimensionless units are used in the code, and the Reynolds number (Re) is varied from about 0.1 to 100. The Re may be found by nondimensionalizing the Navier–Stokes equations as follows:

$$\frac{\partial \mathbf{u}'}{\partial t} = \mathbf{u}' \cdot \nabla' \mathbf{u}' = \nabla' p' + \frac{1}{\mathrm{Re}} \nabla'^2 \mathbf{u}' + \mathbf{f}',$$

$$\nabla' \cdot \mathbf{u}' = 0, \tag{19}$$

$$\mathrm{Re} = \frac{\mu L U}{\rho}, \tag{20}$$

$$\mathbf{u}' = \frac{\mathbf{u}}{U} \qquad t' = \frac{tU}{L}$$

$$p' = \frac{p}{\rho U^2} \qquad \mathbf{x}' = \frac{\mathbf{x}}{L}$$

$$\mathbf{f}' = \frac{\rho U^2 \mathbf{f}}{L} \qquad \nabla' = L\nabla. \tag{21}$$

where L is the characteristic scale of the length (such as the diameter of a rubber band or the tube) and U is the characteristic velocity. In the case of the pumping-tube example, $U = L\,t'$, where t' is the characteristic time and is set equal to the pumping period. The dimensionless variables are $\mathbf{u}'$, $\mathbf{x}'$, p', t', and $\mathbf{f}'$ which represent the dimensionless velocity, position, pressure, time, and force per unit area, respectively. Re may be thought of as being roughly proportional to the ratio of inertial to viscous forces in the fluid.

We have written the examples using dimensionless numbers to make it easier to apply the code to new situations. We have found that students find it challenging to change all of the necessary parameters when moving between problems that differ in scale by orders of magnitude. Such changes require modifications of spring stiffnesses, bending stiffnesses, velocities, and scales of length and time.

For 2D simulations, stiffnesses are given for a sheet of unit length in the third dimension. In dimensional form, k_s and k_{T} have units of N/m. Both can be nondimensionalized using the following equations:

$$k'_{s,\mathrm{T}} = \begin{cases} \dfrac{k_{s,\mathrm{T}}}{\rho U^2 L}, \mathrm{Re} \gg 1 \\ \dfrac{k_{s,\mathrm{T}}}{\mu U}, \mathrm{Re} \ll 1. \end{cases} \tag{22}$$

Similarly, k_{b} has units of N*m and may be nondimensionalized as follows:

$$k'_b = \begin{cases} \dfrac{k_{\mathrm{b}}}{\rho U^2 l_n^2 L}, \mathrm{Re} \gg 1 \\ \dfrac{k_{\mathrm{b}}}{\mu U l_n^2}, \mathrm{Re} \ll 1, \end{cases} \tag{23}$$

where l_n is the length of the elastic links making up the triplet that forms the torsional spring. Typically, l_n will be set to ds, the distance between boundary points. Note that the exact nondimensionalization for intermediate Re is not straightforward, but using the higher Re approximation for Re > 1 and the lower Re approximation for Re < 1 should provide a reasonable estimate. Since elastic boundaries are approximated using linear and torsional springs, the values of $k_{b,s,\mathrm{T}}$ are not equivalent to the flexural stiffness or tensile stiffness used in a continuum model of the material.

Results

A variety of examples are included in the current release of the MATLAB code to illustrate the various capabilities of the program. We highlight a simple FSI problem of a rubber band and a more complicated muscle–FSI simulation of a pumping tube below. Some of the additional examples in the current release include a deformed beam that is free to move in a fluid, flow past a cylinder, and flow driven by an impedance pump. We will continue to develop and post examples, particularly those that include the use of muscle fibers. Specific lesson plans that make use of this code will be added to the github repository in the future, and the best examples will be submitted to the SICB Digital Library. We strongly encourage educators who use the code in their courses to send us new examples and lesson plans. We will be happy to add these examples to the github repository.

The rubber band

The rubber band is one of the simplest examples of a problem in fully coupled FSI. We have found it useful to begin with this example to allow students to gain intuition for FSI before moving to more complicated models. In this problem, we modeled an elastic band using spring connections between adjacent Lagrangian points. The springs all have a preferred zero resting length, as well as fixed stiffness of the spring.

The simulation is initialized with the rubber band stretched into an elliptical shape with a fixed volume of fluid trapped within the elastic band. Since the resting length is zero, the rubber band will be driven toward the lowest state of energy that minimizes length for a given internal volume, i.e., a circle. As it moves toward this equilibrium position, it will contract and expand periodically across the semi-major and semi-minor directions of the axis.

We illustrate three different examples of the rubber-band model, each with the same spring parameters, e.g., the same resting lengths, spring stiffnesses, and equivalently perturbed initial state (see Fig. 3). The structures are, however, immersed within fluids of different dynamic viscosities. The case with the highest viscosity, $\mu = 10$, damps the oscillations such that the structure reaches its equilibrium shape within 0.05 s. For the lowest viscosity case, $\mu = 0.1$, the rubber band oscillates along the long and short semi-major and semi-minor axes.

The elastic tube

A straightforward example of muscular driven motion of a fluid is the contraction of elastic tubes within organisms. Muscle-driven contraction of a tube can be seen in the hearts of many invertebrates, all vertebrate embryonic hearts, and the lymphangions of the lymphatic system.

We have constructed a simplified 2D (not axisymmetric) example of an elastic tube. Note that this is analogous to the case of two infinitely long elastic plates that are driven by muscle fibers that run normal to the two plates. We immerse this 2D elastic tube in fluid and tether it in place at both ends. The walls of the tube resist bending and stretching. The strength of the muscle's activation is given by a sinusoidal function of time. The muscles are activated by a traveling contractile wave throughout the middle section of the elastic tube, generating a

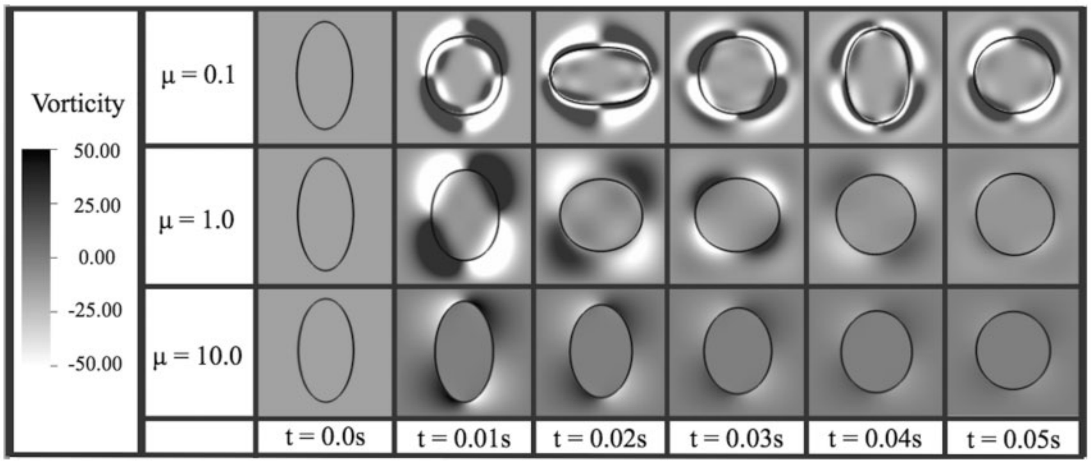


Fig. 3 Vorticity plots showing the evolution of an elastic band in a fluid during six snapshots in time. For the most viscous fluid with dynamic viscosity, μ, set to 10, the band deforms continuously from the initial ellipse ($t = 0$) to the circle. For the least viscous fluid, the band oscillates about the major and minor axes.

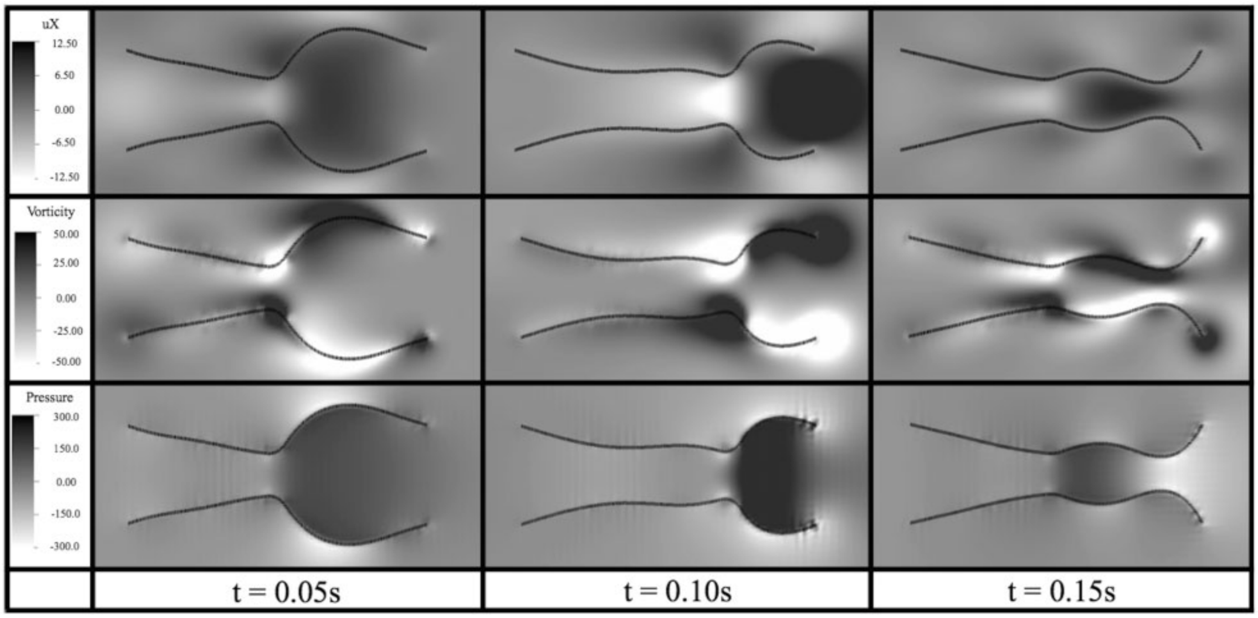


Fig. 4 *x*-Component of velocity (top), vorticity (middle), and pressure (bottom) illustrating the activation wave traveling down the channel at three time-points. Note that the pumping frequency is set to 5 Hz.

contraction that is similar to that observed in the lymphangions under some conditions.

Snapshots showing the *x*-component of velocity, vorticity, and pressure fields at several instances in time are shown in Fig. 4. The output can be changed in the input2d file to view vectors and magnitude of velocity, and other quantities of interest (Appendix 2). The user is encouraged to view the output files in vtk file format using freely available software packages such as VisIt and Paraview. These packages allow the user to easily modify plots, analyze volumetric flow rates, and include colorbars through a GUI interface.

A simple exercise using this example would be to ask students to vary the dynamic viscosity of the tube. At low Re, it becomes difficult for the tube to re-expand. To compensate for this, the bending and stretching stiffnesses of the tube can be increased. Note that the maximum force generated by the muscles should also be increased to generate a significant contraction. At higher Re, strong mixing can be observed through the action of separated vortices that persist in the fluid.

Another modification that students can make is to change the code to consider the case of a uniform contraction rather than a traveling wave. This results in a situation in which the fluid moves symmetrically from each side of the tube during contraction. The fluid then moves back into the tube during the passive expansion. The symmetry in flow may be broken either by introducing one-way valves, as in the case of the lymphangions, or by introducing a unidirectional traveling wave of contraction.

Discussion

In this article, we developed 2D MATLAB software for solving muscle—FSI problems with the IB method. We wrote the code such that students can create their own geometries and elastic properties by connecting boundary points with linear and torsional springs. Each linear spring requires a specified stiffness and resting length of the spring as parameters; however, both can be changed at each time-step if desired. Torsional springs need a specified stiffness and curvature as parameters, and can also be modified at each time-step. Prescribed motion can be achieved using a target point formulation and updating the target-point positions at each time-step. We also included a simple muscle mechanics model that can be used to apply a force to the boundary.

We have used this code, or a similar code, for undergraduate research experiences, an undergraduate course in quantitative biology, and a graduate course in mathematical modeling. In the research experiences and graduate course, students were expected to create new examples of FSI and muscle–FSI problems as course projects. Some example projects have included the swimming of jellyfish and of copepods, cross-sections of wings in flow, and the pumping of tubular hearts. In the undergraduate courses, students used existing examples to explore suggested parameter spaces. We intend to more extensively use this code in the following year as part of a laboratory component for the introductory quantitative biology course. The only prerequisite for this course is the first semester of calculus, and the majority of students enrolled are biology majors. We will include additional examples and lesson plans in future releases of the code available on the github repository.

The current program is written for ease of use. There are also some limitations. This IB method works best for Reynolds numbers between 10^{-2} and 10^{2}. It is not well-suited for very high Reynolds numbers with significant turbulence. The numerical method works best when all terms are balanced, including forces due to the resistance to bending, resistance to stretching, inertial forces, and viscous forces. The 2D simulations run relatively quickly for problems of modest size. Finer grids and stiffer springs and beams require smaller time-steps. One can quickly design a simulation that requires on the order of 10^5 to 10^6 times-steps. For simulations with complex boundaries or complex fluid motion, the spatial grids required for solving the Navier–Stokes equations can become prohibitively large.

For the student or faculty who would like to perform more sophisticated simulations, there are alternative packages available. One example is the IB Method with Adaptive Mesh Refinement, or IBAMR (Griffith 2015). IBAMR is a distributed-memory parallel implementation of the IB method with support for Cartesian-grid adaptive-mesh refinement. This library is used broadly within the immersed-boundary community. In addition to the immersed-boundary method, there are other approaches for numerically solving FSI problems. Some of these methods include front-tracking methods (Unverdi and Tryggvason 1992), level-set methods (Sussman et al. 1994; Legaya et al. 2006), the segment-projection method (Tornberg and Engquist 2003), the immersed-interface method (Lee and LeVeque 2003), and other variations on the immersed-boundary method (Mittal and Iaccarino 2005).

Acknowledgment

The authors would like to thank Virginia Pasour for her support of this symposium and Boyce Griffith for the inspiration of this MATLAB code.

Funding

This work was supported by the grant from the US Army Research Office to L. Waldrop [W911NF-14-1-0326]; the Society for Integrative and Comparative Biology's Divisions of Animal Behavior, Comparative Physiology and Biochemistry, Comparative

Biomechanics, and Vertebrate Morphology. This work was also supported by a National Science Foundation Grant [DMS 1151478 to L.M.].

Supplementary data

Supplementary Data available at ICB online.

References

Bottino DC. 1998. Modeling viscoelastic networks and cell deformation in the context of the immersed boundary method. J Comput Phys 147:86–113.

Campbell KS. 2006. Filament compliance effects can explain tension overshoots during force development. Biophys J 91:4102–9.

Challis JH, Kerwin DG. 1994. Determining individual muscle forces during maximal activity: Model development, parameter determination, and validation. Hum Movement Sci 13:29–61.

Clark RP, Smits AJ. 2006. Thrust production and wake structure of a batoid-inspired oscillating fin. J Fluid Mech 562:415–29.

Fung YC. 1993. Biomechanics: mechanical properties of living tissues. New York: Springer-Verlag.

Gemmell BJ, Costello JH, Colin SP, Stewart CJ, Dabiri JO, Tafti D, Priya S. 2013. Passive energy recapture in jellyfish contributes to propulsive advantage over other metazoans. Proc Natl Acad Sci USA 110:17904–9.

Goldman YE, Huxley AF. 1994. Actin compliance: are you pulling my chain? Biophys J 67:2131–3.

Griffith BE. 2015. An adaptive and distributed-memory parallel implementation of the immersed boundary (IB) method (IBAMR) (https://github.com/IBAMR/IBAMR).

Griffith BE, Luo X, McQueen DM, Peskin CS. 2009. Simulating the fluid dynamics of natural and prosthetic heart valves using the immersed boundary method. Int J Appl Mech 1:137–77.

Hatze H. 1981. A comprehensive model for human motion simulation and its application to the take-off phase of the long jump. J Biomech 14:135–42.

Hill AV. 1938. The heat of shortening and the dynamic constants of muscle. Proc R Soc Lond 126:136–95.

Huxley AF. 1957. Muscle structure and theories of muscle contraction. Progr Biophys Biophys Biochem 7:255–318.

Keener J, Sneyd J. 1998. Mathematical physiology. New York: Springer.

Lee L, LeVeque RJ. 2003. An immersed interface method for incompressible Navier–Stokes equations. SIAM J Sci Comput 25:47–79.

Legaya A, Chessab J, Belytschkoc T. 2006. An Eulerian–Lagrangian method for fluid–structure interaction based on level sets. Comput Methods Appl Mech Eng 195:2070–87.

Miller LA, Peskin CS. 2009. Flexible clap and fling in tiny insect flight. J Exp Biol 212:3076–90.

Mittal R, Iaccarino G. 2005. Immersed boundary methods. Annu Rev Fluid Mech 37:239–61.

Peskin CS. 1972. Flow patterns around heart valves: A numerical method. J Comput Phys 10:252–71.

Peskin CS. 2002. The immersed boundary method. Acta Numer 11:479–517.

Peskin CS, McQueen DM. 1996. Fluid dynamics of the heart and its valves. In: Adler FR, Lewis MA, Dalton JC, Othmer HG, editors. Case studies in mathematical modeling—ecology, physiology, and cell biology. New Jersey: Prentice Hall. p. 309–37.

Press WH, Teukolsky SA, Vetterling WT, Flannery BP. 1992. Numerical recipes in Fortran: The art of scientific computing. New York: Cambridge University Press.

Schmidt-Nielsen K. 1997. Animal physiology: Adaptation and environment. Cambridge: Cambridge University Press.

Shoele QZAK. 2008. Propulsion performance of a skeleton-strengthened fin. J Exp Biol 211:2087–100.

Skorczewski T, Griffith BE, Fogelson AL. 2014. Multi-bond models of platelet adhesion and cohesion. AMS Contemporary Mathematics Series on Biofluids 628:149–72.

Sussman M, Smereka P, Osher S. 1994. A level set approach for computing solutions to incompressible two-phase flows. J Comput Phys 114:146–59.

Tornberg A-K, Engquist B. 2003. The segment projection method for interface tracking. Commun Pure Appl Math 56:47–79.

Tytell ED, Hsu C-Y, Williams TL, Cohen AH, Fauci LJ. 2010. Interactions between body stiffness, muscle activation, and fluid environment in a neuromechanical model of lamprey swimming. Proc Natl Acad Sci USA 107:19832–7.

Unverdi SO, Tryggvason G. 1992. A front-tracking method for viscous, incompressible, multi-fluid flows. J Comput Phys 100:25–37.

Vogel S. 1996. Life in moving fluids: The physical biology of flow. Princeton, NJ: Princeton University Press.

Vogel S. 2013. Comparative biomechanics: Life's physical world. Princeton: Princeton University Press.

Appendix 1

A brief overview of the method used to numerically solve the Navier–Stokes equations (6) and (7) is provided below. More details can be found elsewhere (Peskin and McQueen 1996). The discretization method is implicitly defined as follows:

$$\rho\left(\frac{\mathbf{u}^{k+1}-\mathbf{u}^k}{\Delta t}+S_{\Delta x}\left(\mathbf{u}^k\right)\mathbf{u}^k\right)+D^0\mathrm{p}^{k+1} = \mu\sum_{\alpha=1}^{2}D_\alpha^+D_\alpha^-\mathbf{u}^{k+1}+\mathbf{F}^k. \tag{24}$$

$$D^0\cdot\mathbf{u}^{k+1}. \tag{25}$$

where ρ is the density of the fluid and μ is the viscosity of the fluid. $\mathbf{D}^0$ is the central difference approximation to ∇. It is defined as follows:

$$\boldsymbol{D}^0 = \left(D_1^0, D_2^0\right), \tag{26}$$

$$\left(D_\alpha^0 \phi\right)(\mathbf{x}) = \frac{\phi(\mathbf{x} + \Delta x \mathbf{e}_\alpha) - \phi(\mathbf{x} - \Delta x \mathbf{e}_\alpha)}{2\Delta x}, \tag{27}$$

where $[e_1, e_2]$ is the standard basis of $\mathbf{R}^2$. The operators $D_\alpha^\pm$ are forward and backward difference-approximations of $\partial/\partial x_\alpha$. They are defined as follows:

$$\left(D_\alpha^+ \phi\right)(\mathbf{x}) = \frac{\phi(\mathbf{x} + \Delta x \mathbf{e}_\alpha) - \phi(\mathbf{x})}{\Delta x}, \tag{28}$$

$$\left(D_\alpha^- \phi\right)(\mathbf{x}) = \frac{\phi(\mathbf{x}) - \phi(\mathbf{x} - \Delta x \mathbf{e}_\alpha)}{\Delta x}. \tag{29}$$

Thus, the viscous term given as $\sum_{\alpha=1}^{2} D_\alpha^+ D_\alpha^-$ is a difference-approximation to the Laplace operator.

Finally, $S_{\Delta x}$ $(\mathbf{u})$ is a skew-symmetric difference operator that serves as a difference-approximation of the nonlinear term $\mathbf{u} \cdot \nabla \mathbf{u}$. This skew-symmetric difference operator is defined as follows:

$$S_{\Delta x}(\mathbf{u})\phi = \frac{1}{2}\mathbf{u} \cdot \mathbf{D}_{\Delta x}^0 \phi + \frac{1}{2}\mathbf{D}_{\Delta x}^0 \cdot (\mathbf{u}\phi). \tag{30}$$

Since the equations are linear in the unknowns $\mathbf{u}^{k+1}$ and p^{k+1}, the Fast Fourier Transform (FFT) algorithm was used to solve for $\mathbf{u}^{k+1}$ and p^{k+1} from $\mathbf{u}^k$, p^k, and $\mathbf{f}^k$ (Press et al. 1992; Peskin and McQueen 1996).

Appendix 2

The examples provided above and some additional examples are available on github in the "Examples" folder. Each example contains a file named *input2d*. This file contains the parameters and flags necessary to run a simulation, e.g., properties of the fluid, temporal information, the structure of the Eulerian grid, flags for the mathematical model(s) used in constructing the IB, and flags for printing and displaying output. A user manual is also available on github with additional details and updates.

The user has the option to change the dynamic viscosity, μ, and density, ρ, of the fluid using the *input2d* file. Changing these parameters allows one to easily vary the Re and explore scaling effects in the system. It is important to note that changing Re may require the user to modify the simulation's time-step, as the stiffness of the equations of the model may demand better temporal refinement for stability of the numerical solver. The user also has the ability to change the time-step and final time of the simulation. The grid parameters pass the resolution and dimensions of the computational domain to the IB solver. The resolution, Nx and Ny, set the number of uniformly spaced fluid points in the *x* and *y* cardinal directions, respectively, within the domain of dimensions [0, Lx] × [0, Ly]. Note that Lx/Nx should equal Ly/Ny.

The input2d file also contains saving data and plotting options. The user can specify the interval between successive saves of the Eulerian and Lagrangian data in the simulation, i.e., *print_dump*. The data are saved in *.vtk* format, which can be visualized with open-source visualization tools such as VisIt and Paraview. The option is also available to plot the data in MATLAB using the *plot_Matlab* flag. Moreover, the user can choose which dynamic quantities MATLAB is to plot while running the simulation. In this case, MATLAB will plot the desired quantities during the same time-steps in which the dynamical data are being saved. Please be aware that plotting during the simulation may significantly increase the simulation time.

The Lagrangian structure flags indicate the model boundaries to be used in the simulation. Boolean logic indicates whether an elastic model is to be used. Note that if the simulation does not have the necessary input files for a particular elastic model but such an elastic model is indicated, the code will throw an error. An example would be the case in which there is no *struct_name.beam* file, but beams are indicated in the *input2d* file.