Cartesian Grid Methods for Fluid Flow in Complex Geometries [1]

Randall J. LeVeque[2] and Donna Calhoun[3]

Abstract. Biological fluid dynamics typically involves geometrically complicated structures which are often deforming in time. We give a brief overview of some approaches based on using fixed Cartesian grids instead of attempting to use a grid which conforms to the boundary. Both finite-difference and finite-volume methods are discussed, as well as a combined approach which has recently been used for computing incompressible flow using the streamfunction-vorticity formulation of the incompressible Navier-Stokes equations.

1 Introduction

Biological fluid dynamics typically involves geometrically complicated structures which are often deforming in time. Solving fluid dynamics equations in such geometries can be extremely challenging, and methods based on body-fitted grids which conform to the geometry can be difficult to implement. We give a brief overview of some approaches based on using fixed Cartesian grids instead of forcing the grid to conform to the boundary. Peskin's *immersed boundary method* outlined in Section 4 is the best-known example of this class in the field of biological fluid dynamics and has been successfully used by many researchers. The *immersed interface method* discussed in Section 5 is a related approach which can often achieve better accuracy near the boundary. These are both finite-difference methods.

For advection-diffusion equations, finite-volume methods may be better suited, as described in Section 6. These allow exact conservation on grid cells to be maintained and flux boundary conditions to be easily imposed. High-resolution "shock-capturing" methods developed for conservation laws are also quite useful for problems with discontinuities or steep gradients in the solution.

In Section 7 we describe a hybrid approach for computing incompressible flow using the streamfunction-vorticity formulation of the Navier-Stokes equations. Vorticity is advected and diffused on a finite-volume grid, with flux at solid wall boundaries as required to satisfy the no-slip boundary conditions. The streamfunction is computed from the vorticity on a finite-difference grid and then differenced to obtain velocities needed in the finite-volume method.

2 Finite-difference and finite-volume methods

We will discuss both finite-difference and finite-volume approaches, and first review the essential ideas of each approach.

[1] To appear in Proc. IMA Workshop on *Computational Modeling in Biological Fluid Dynamics*, January, 1999.

[2] Department of Applied Mathematics and Department of Mathematics, University of Washington, Box 352420, Seattle, WA 98195-2420. rjl@amath.washington.edu

[3] Department of Applied Mathematics, University of Washington, Box 352420, Seattle, WA 98195-2420. calhoun@amath.washington.edu

A finite-difference discretization of a differential equation is based on replacing derivatives by differences of values defined on a discrete set of *grid points*. The approximate solution generated consists of values U_{ij}^n (in two dimensions) representing pointwise approximations to the solution at grid points (x_i, y_j) at time t_n,

$$U_{ij}^n \approx u(x_i, y_j, t_n). \tag{1}$$

A finite-volume method, on the other hand, is based on approximating cell averages of the solution over grid cells,

$$U_{ij}^n \approx \frac{1}{\Delta x \Delta y} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} u(x, y, t_n)\, dx\, dy. \tag{2}$$

For differential equations which arise from conservation laws and fluxes (including advection-diffusion equations and most equations in fluid dynamics), this is a very natural approach. The differential equation often arises from a more fundamental integral form stating that the integral of u over any finite volume changes with time only due to the fluxes across the boundary of the volume. If the volume is a single grid cell, then this leads to an updating formula for the cell average based on numerical flux formulas. This has some advantages over more familiar finite-difference formulations. For problems where some quantity should be conserved it is often easy to guarantee exact conservation numerically, This is particularly important in nonlinear wave-propagation problems with shock waves, where the conservation form must be respected in order to obtain correct solutions (see [42], for example). It can also be important in other problems, such as when we wish to exactly conserve the total mass of some chemical advecting in a flow. Another advantage of finite-volume methods is that physical boundary conditions are often given in terms of fluxes at the boundary. These are often more naturally discretized in the context of finite-volume methods than with finite differences.

In some contexts, however, finite-difference methods are more natural and may be easier to use. We have used a mixture of the two approaches in developing Cartesian grid methods, using each where it is most appropriate. (Another standard approach for partial differential equations is the finite element method, based on variational principles. We don't discuss this class of methods here.)

3 Cartesian grids

Figure 1 shows a comparison of body-fitted grids with Cartesian grids, in both the finite-difference and finite-volume frameworks. The top row shows body-fitted grids near a curved wall. A finite-difference method might update the values at one grid point based on values at the 4 nearest neighbors, as indicated by the 5-point stencil shown. A finite-volume method, on the other hand, would update the cell average of $u(x, y, t)$ over the shaded grid cell based on fluxes through the four edges indicated. If the fluxes are based on values in this cell and the four neighboring cells, then again a 5-point stencil is obtained.

The middle row of Figure 1 shows a Cartesian grid at the same boundary. With a finite-difference method one might develop a modified 5-point method based on the unequal distances from the grid point shown to the points which lie on the wall. With the finite-volume approach one instead has an irregular-shaped grid cell which requires computing a flux at the edge corresponding to the physical boundary as well as at the sides adjacent to other cells. Note

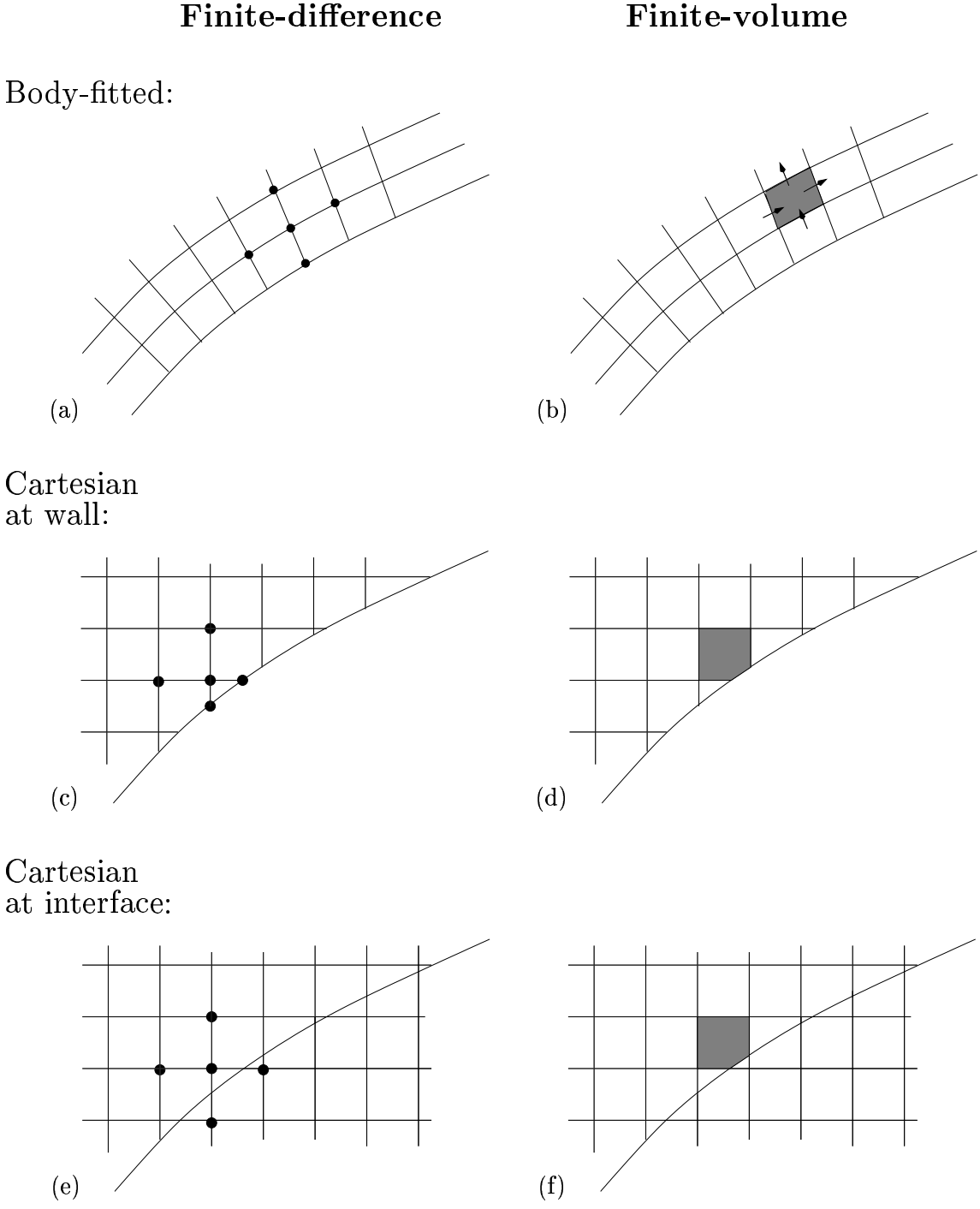

Figure 1: Comparison of different types of grids for boundaries or interfaces.

that the finite-volume method requires a *flux* at the wall (Neumann boundary conditions) whereas the finite-difference method requires a pointwise value at the wall (Dirichlet boundary condition), although with additional work either boundary condition can be imposed on either grid.

The bottom row of Figure 1 shows a different situation in which there is an interface immersed in the domain rather than a wall. This interface might be a membrane immersed in the fluid or an interface between two different fluids, for example. In this case we typically have partial differential equations which must hold on each side of the interface, though perhaps with discontinuous coefficients across the interface, or with an additional source term appearing only at the interface. This generally leads to nonsmoothness in the solution at the interface. With a finite-difference method we might attempt to use a 5-point stencil with equally-spaced points on both sides of the interface as indicated in Figure 1(e). This can be successful as long as the nonsmoothness is properly accounted for. With a finite-volume method one might view grid cells as being confined to one side or the other of the interface, as indicated in Figure 1(f), with special formulas for the flux across the interface, or one could view the uniform Cartesian cells as spanning the interface, with appropriate source terms incorporated in addition to the fluxes.

4 Embedded bodies and the Immersed Boundary Method

For problems with boundaries which cut through the grid, rather than using grids of the type shown in Figure 1(c) or (d) an alternative is to embed the physical domain into a larger computational domain and view the physical boundary as an interface of the type shown in Figure 1(e) or (f). Grid points or cells are introduced which lie outside the physical domain. If the equations can be extended in a suitable way, it may be simpler to solve the extended equations on a purely Cartesian grid. A variety of methods of this type have been introduced, such as fictitious domain methods (*e.g.*, [33], [34]) and Mayo's method for elliptic equations [50], [51].

The *immersed boundary method* is a numerical method of this type developed primarily by Peskin and his co-workers for solving the incompressible Navier-Stokes equations in complicated time-varying geometries. The original application, which is still being studied with this method, was to model a beating heart in order to study the motion of both natural and artificial heart valves [53], [54], [55], [56], [57]. Since then it has been used for many other applications, including aquatic locomotion [23], blood platelet aggregation [26] [27], [28], and wave motion in the cochlea [6], [7]. Similar approaches have also been used outside of biophysics, *e.g.*, to sedimentation [64] and bubble dynamics [68], [69].

For a problem such as heart modeling, the geometry is not only extremely complex but is also time-varying as the heart goes through its pumping cycle and the valves open and close. Attempting to use a grid that conforms to the physical boundary would be extremely difficult. Peskin's idea was to instead view the entire heart as being immersed in a Cartesian box of fluid (a square in the original 2D model or a cube in the current 3D model). The incompressible Navier-Stokes equations are solved on a uniform Cartesian grid in this box and the outer heart wall (as well as interior walls separating chambers, valve leaflets, etc.) are viewed as being immersed within the fluid. These structures are modeled by a separate Lagrangian set of grid points that move with the local fluid velocity in order to impose the requirement that no fluid can flow through the wall.

In each time step, the Navier-Stokes equations are solved on the uniform grid to determine the fluid velocities at each of these grid points. These velocities are then interpolated to the immersed boundary points to determine its motion. Of course the boundary must also have an effect on the fluid, and this is modeled by adding a forcing term to the Navier-Stokes equations to represent the force exerted on the fluid by the immersed boundary. This is a singular (delta function) force with support only along the boundary. The strength of this force depends on the nature and position of the boundary. For example, an elastic material which is stretched exerts a restoring force that can be determined from the configuration of the boundary. The heart wall also exerts force due to muscle contraction that can be determined based on position and phase of the heart cycle.

The incompressible Navier-Stokes equations then take the form

$$\begin{aligned} \vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} + \nabla p &= \nu \Delta \vec{u} + f \\ \nabla \cdot \vec{u} &= 0 \end{aligned} \tag{3}$$

where

$$f(\vec{x}, t) = \int_\Gamma F(s,t)\delta(\vec{x} - \vec{X}(s,t))\, ds \tag{4}$$

Here $\vec{x} = (x, y)$ is the spatial variable in 2D, Γ is the immersed boundary consisting of points $\vec{X}(s,t)$ parameterized by s at time t, δ is the 2D delta function and $F(s,t)$ is the force density at $\vec{X}(s,t)$. These equations are coupled with the equations for the motion of Γ,

$$\frac{\partial}{\partial t}\vec{X}(s,t) = u(\vec{X}(s,t), t) \tag{5}$$

and equations for determining $f(s,t)$ from the current boundary configuration $\vec{X}(s,t)$. We solve these equations in a box Ω that contains Γ, with some convenient boundary conditions on the boundary of Ω, *e.g.*, periodic boundary conditions as used by Peskin[53].

The box is then discretized by a uniform grid with grid points $\vec{x}_{ij} = (x_i, y_j)$ with $x_i = ih$ and $y_j = jh$. The immersed boundary Γ is represented by a set of points $\vec{X}_k = (X_k, Y_k)$, $k = 1, 2, \ldots, M$ with spacing which is $O(h)$. The Navier-Stokes equations (3) are discretized using any standard technique on the uniform grid. The only difficulty is that the equation for updating u_{ij}, the velocity at $\vec{x}_{ij}$, will typically involve f_{ij}, the forcing function evaluated at this point. But since the singular force (4) has support only on Γ, this will typically be zero at each $\vec{x}_{ij}$. We need some method to spread the singular force (4) out to neighboring grid points on the regular grid. This can be done by replacing the delta function by some approximation, a sharply peaked function with unit integral and support that extends only distance $O(h)$ from the origin. In one space dimension a simple choice is the hat function

$$d_h(x) = \begin{cases} \frac{1}{h^2}(h - |x|) & \text{if } |x| < h \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

though other choices are generally used in practice. In two space dimensions, the product $d_h(x)d_h(y)$ can be used. The force f_{ij} used in the discrete equations at $\vec{x}_{ij}$ is then given by

$$f_{ij} = \sum_{k=1}^{M} F_k(t) d_h(x_i - X_k) d_h(y_j - Y_k) \tag{7}$$

where $\vec{X}_k = (X_k, Y_k)$ and $F_k(t)$ is the force at this point.

Moving the boundary requires interpolating the velocities from the grid to Γ. One way to do this is to use the same discrete delta function defined above and set

$$U_k = \sum_{i,j} u_{ij} d_h(x_i - X_k) d_h(y_j - Y_k). \tag{8}$$

Even when a highly accurate Navier-Stokes solver is used on the uniform grid, there remains the interesting and difficult issue of the accuracy of the immersed boundary procedure, *i.e.*, the replacement of the boundary conditions by a singular force that is spread to the uniform grid by discrete delta functions. This has been analyzed in some simple cases by Beyer and LeVeque[8], who considered a 1D model problem of the form

$$u_t = u_{xx} + c(t)\delta(x - \alpha(t)) \qquad 0 \le x \le 1 \tag{9}$$

with the boundary conditions $u(0,t) = u(1,t) = 0$. Here the point $\alpha(t)$ plays the role of the immersed boundary Γ and $c(t)$ is the strength of the singular source at α. In the simplest case $\alpha(t) \equiv \alpha$ is fixed and $c(t)$ is a smooth function that is given *a priori*. A discretization in the spirit of the immersed boundary method would be the Crank-Nicolson method

$$\frac{1}{\Delta t}(U_j^{n+1} - U_j^n) = \frac{1}{2}(D_x^2 U_j^n + D_x^2 U_j^{n+1}) + c(t_{n+1/2}) d_h(x_j - \alpha) \tag{10}$$

where D_x^2 is the second difference operator

$$D_x^2 U_j = \frac{1}{h^2}(U_{j+1} - 2U_j + U_{j-1})$$

and d_h is a discrete delta function such as (6). If $c(t) \equiv 0$ then the Crank-Nicolson method is well known to be second order accurate in both space and time. Beyer and LeVeque [8] show that the method (10) remains second order accurate for smooth functions $c(t)$ provided that the discrete delta function d_h satisfies certain moment conditions.

When $\alpha(t)$ is moving, it was found that a correction term must be added to the natural extension of the Crank-Nicolson method in order to maintain second order accuracy [8]. This is because the time derivative of the solution at a fixed grid point is not continuous in time as the point $\alpha(t)$ passes this grid point. However, again the jump can be determined directly from the equations and used to correct the formulas. This is an indication that there may be difficulties in achieving second order accuracy with the immersed boundary method in fluid dynamics problems, even with an improved delta function.

In two dimensions it appears to be very difficult to obtain second-order accuracy near the boundary by using a simple discrete delta function. Zhilin Li [46] found this to be so even for the simplest case of a steady state Poisson problem

$$u_{xx} + u_{yy} = f \tag{11}$$

where f is again given by a singular source on some curve Γ,

$$f(x,y) = \int_\Gamma F(s)\delta(x - X(s))\delta(y - Y(s))\, ds. \tag{12}$$

The immersed boundary method would correspond to using the discrete equations

$$(D_x^2 + D_y^2)u_{ij} = f_{ij}$$

where $D_x^2 + D_y^2$ is the standard 5-point Laplacian and f_{ij} is given by (7). There does not seem to be a simple way to define d_h so that this yields second order accuracy for arbitrary curves Γ.

It is, however, possible to obtain second order accuracy at all uniform grid points by a different construction of the f_{ij}, as discussed in the next section.

5 The immersed interface method

In an attempt to obtain better accuracy for problems with delta-function source terms on an interface cutting between grid points, a different approach was developed by Li [46] and LeVeque and Li [44], first in the context of elliptic equations such as the Poisson problem mentioned above. This approach was called the *immersed interface method* since it is similar in spirit to the immersed boundary method but turns out to be applicable to a variety of interface problems beyond immersed boundaries in incompressible flow. The basic idea is most easily explained by considering the 1D model problem (9) with $\alpha(t) \equiv \alpha$. Suppose we want to compute $u_t(x_i, t)$ at some grid point x_i at time t. If α lies between grid points then $u_t = u_{xx}$ since the source term vanishes at x_i. We can usually approximate u_{xx} to second order by the second difference

$$D_x^2 u(x_i, t) = \frac{1}{h^2}(u(x_{i+1}, t) - 2u(x_i, t) + u(x_{i+1}, t)).$$

This is second order accurate provided α does not lie between x_{i-1} and x_{i+1}. If it does, then the nonsmoothness of $u(x, t)$ at α destroys the accuracy of this approximation. The solution to (9) is continuous and so is u_{xx}, but the first derivative u_x has a discontinuity at $x = \alpha$ of magnitude $-c(t)$. We can use this information to obtain a good approximation to u_{xx} at grid points near α.

Suppose, for example, that $x_j < \alpha < x_{j+1}$ and we wish to approximate $u_{xx}(x_j, t)$. Expanding in Taylor series about α shows that

$$\begin{aligned} u(x_j, t) &= u(\alpha, t) + (x_j - \alpha)u_x^-(\alpha, t) + \frac{1}{2}(x_j - \alpha)^2 u_{xx}(\alpha, t) + O(h^3) \\ u(x_{j+1}, t) &= u(\alpha, t) + (x_{j+1} - \alpha)u_x^+(\alpha, t) + \frac{1}{2}(x_{j+1} - \alpha)^2 u_{xx}(\alpha, t) + O(h^3). \end{aligned}$$

We use the notation

$$u_x^-(\alpha, t) = \lim_{x \to \alpha -} u_x(x, t), \qquad u_x^+(\alpha, t) = \lim_{x \to \alpha +} u_x(x, t)$$

for the limits as x approaches α from the left and right. (Note that continuity of u_{xx} allows us to simply write $u_{xx}(\alpha, t)$.)

We also have

$$\begin{aligned} u_x^-(\alpha, t) &= u_x(x_j, t) + (\alpha - x_j)u_{xx}(x_j, t) + O(h^2) \\ u_x^+(\alpha, t) &= u_x(x_j, t) + (\alpha - x_j)u_{xx}(x_j, t) + [u_x] + O(h^2) \end{aligned}$$

where $[u_x] = u_x^+(\alpha,t) - u_x^-(\alpha,t)$ is the jump in u_x at α. Combining these expressions gives

$$u(x_{j+1},t) = u(x_j,t) + hu_x(x_j,t) + \frac{1}{2}h^2 u_{xx}(x_j,t) + (x_{j+1}-\alpha)[u_x] + O(h^3).$$

Combining this with the standard Taylor series expansion of $u(x_{j-1},t)$ about $u(x_j,t)$,

$$u(x_{j-1},t) = u(x_j,t) - hu_x(x_j,t) + \frac{1}{2}h^2 u_{xx}(x_j,t) + O(h^3)$$

shows that

$$u_{xx}(x_j,t) = D_x^2 u(x_j,t) - \frac{1}{h^2}(x_{j+1}-\alpha)[u_x] + O(h).$$

Since we know that

$$[u_x] = -c(t),$$

this gives an approximation to $u_{xx}(x_j,t)$:

$$u_{xx}(x_j,t) = \frac{1}{h^2}(u(x_{j+1},t) - 2u(x_j,t) + u(x_{j-1},t)) + \frac{1}{h^2}(x_{j+1}-\alpha)c(t) + O(h).$$

Similarly, at x_{j+1} we find that

$$u_{xx}(x_{j+1},t) = D_x^2 u(x_{j+1},t) + \frac{1}{h^2}(\alpha - x_j)c(t) + O(h).$$

At all other grid points x_i we can use $D_x^2 u(x_i,t)$ alone to obtain a second order accurate approximation.

It is interesting to note that these formulas can be combined in the following general formula valid for all i, including j and $j+1$:

$$u_{xx}(x_i,t) = D_x^2 u(x_i,t) + c(t)d_h(x_i-\alpha) + \begin{cases} O(h) & \text{if } \ i = j,\ j+1 \\ O(h^2) & \text{otherwise} \end{cases} \tag{13}$$

where $d_h(x)$ is the hat function (6) Note that this is a discrete delta function, so using the approximation (13) in a Crank-Nicolson method for (9) gives precisely the method (10) with the particular choice (6) for the discrete delta function. (See [8], [44] for more details.)

It turns out that we only need an $O(h)$ local truncation error at these grid points in order to obtain $O(h^2)$ global errors, since there are only two grid points involved independent of h. It is this type of analysis that was used in [8] to show that the immersed boundary method is second order accurate in some cases for the 1D model problem.

In two dimensions consider the Poisson problem (11). It can be shown that the solution u will be continuous across Γ while the normal derivative of u will have a jump discontinuity of magnitude $F(s)$. To devise a finite difference scheme, we need to approximate $u_{xx} + u_{yy}$ to at least $O(h)$ at each grid point (x_i, y_j) near the interface. The standard 5-point approximation $D_x^2 u_{ij} + D_y^2 u_{ij}$ can be used provided that the interface Γ does not come between any of the points in this stencil. If it does, then we must add correction terms arising from jumps in derivatives across Γ, just as in the 1D expression (13). This correction can be derived by expanding u at each grid point in the stencil about some common nearby point on Γ, *e.g.*, the

closest point on Γ to (x_i, y_j). The resulting formula for approximating $u_{xx} + u_{yy}$ will have the form

$$u_{xx} + u_{yy} = D_x^2 u_{ij} + D_y^2 u_{ij} + C_{ij} + O(h)$$

where the correction term C_{ij} is zero away from Γ and near Γ will have magnitude $O(1/h)$ and depend on the source strength $F(s)$. For details see [44]. If this approximation is used to discretize

$$u_{xx} + u_{yy} = 0$$

then we will obtain the system

$$D_x^2 u_{ij} + D_y^2 u_{ij} = -C_{ij}$$

which can be viewed as a discretization of the original equation (11). But in this case, unlike the 1D model problem, it is not possible to derive the correct C_{ij} simply by using the formula (7) with an appropriate choice of d_h. The expression for C_{ij} can be obtained from the more general case considered in [44]. The resulting formula involves not only the jump in normal derivative of u along the interface but also the tangential derivative of this jump and the curvature of the interface. These can all be obtained from the original problem.

By computing the correct values C_{ij}, we are able to obtain a second order accurate method that is in the spirit of the immersed boundary method, but in a case where the immersed boundary method itself fails to deliver this accuracy. All that is required for this approach is *a priori* knowledge of the jumps in certain derivatives across Γ. This can often be determined from the PDE, as in the above examples.

Returning to the incompressible Navier-Stokes equations, we find that this is also true. In this case the force density $F(s,t)$ is a vector. By decomposing this into components normal and tangential to Γ, it is possible to determine jumps in the pressure p and in derivatives of p and the velocities. It should be possible to use this information to derive appropriate correction terms for a second order Navier-Stokes solver on a uniform grid.

5.1 Stokes flow

So far this has been completely carried out only for the Stokes equations,

$$\begin{aligned} p_x &= \mu \Delta u + f^{(1)} \\ p_y &= \mu \Delta v + f^{(2)} \\ u_x + v_y &= 0 \end{aligned} \tag{14}$$

where $f = (f^{(1)}, f^{(2)})$ has the form (12). This problem can be reduced to a set of Poisson problems ($\Delta p = 0$ and then $\mu \Delta u = p_x$ and $\mu \Delta v = p_y$) with known jumps in the functions and derivatives across Γ.

Development of methods for this problem with a moving interface are developed in [45]. The method is compared with the immersed boundary method for a test problem studied by Tu and Peskin [67] consisting of a two-dimensional "balloon" relaxing to a circular shape in a highly viscous fluid. It is shown that the new approach yields second order accuracy in both pressure and velocity while an immersed boundary approach with the discrete delta function (6) gives only first order accuracy in velocities and $O(1)$ errors in pressure near the discontinuity. Two-dimensional bubbles are also considered in [45], in which the singular force at the interface arises from surface tension rather than from an elastic membrane.

5.2 Stiffness and implicit methods

As in Peskin's immersed boundary method, the moving interface tracked in [45] is modeled by a set of marker points which move with the local fluid velocity as interpolated from the uniform grid on which velocity is computed. An explicit method might be written symbolically as

$$\vec{X}^{n+1} = \vec{X}^n + \Delta t \mathcal{U}(\vec{X}^n), \tag{15}$$

where $\vec{X}^n$ represents a vector of marker positions at time t_n. The nonlinear function $\mathcal{U}(\vec{X})$ is the mapping from marker locations at some time to the resulting velocities at the same points. Evaluating $\mathcal{U}(\vec{X})$ typically requires evaluating forces at the points $\vec{X}$, computing the resulting source terms F_{ij} on the uniform grid, solving the fluid equations, and interpolating velocities back to the marker points $\vec{X}$. Often this problem is very stiff and the time step Δt in the explicit method (15) must be taken very small in order to avoid instabilities in which small oscillations in the interface grow exponentially. Often a semi-implicit approach is used with the immersed boundary method to improve stability, see for example [67], [52], [54], [63].

For the Stokes flow problem in [45] a fully implicit method has been used, based on the trapezoidal method

$$\vec{X}^{n+1} = \vec{X}^n + \frac{1}{2}\Delta t(\mathcal{U}(\vec{X}^n) + \mathcal{U}(\vec{X}^{n+1})). \tag{16}$$

The potential difficulty with this approach is that $\mathcal{U}(\vec{X})$ is expensive to evaluate and it is impossible to explicitly compute the Jacobian of this mapping as would be required for a Newton-type method.

In [45], a quasi-Newton method has been used with good results. With this approach an approximation to the Jacobian is built up in the process of iterating to solve the nonlinear system (16), using rank 1 or rank 2 updates of the previous approximation. In the first time step the procedure takes quite a few iterations to converge, but in future time steps a good initial approximation to the Jacobian is available from the previous time step. For the bulk of the computation only 2 or 3 iterations per time step are required. This approach has also been successfully used for Hele-Shaw flow [35] and for Stefan problems with moving phase boundaries. It could probably be used for many other problems of this type, also in conjunction with the immersed boundary method or other methods for computing the mapping $\mathcal{U}(\vec{X})$.

Another device used in [45] to reduce the expense of the implicit method is to represent the interface by relatively few marker points, with the distance between them much larger than the mesh spacing of the Cartesian grid. Cubic splines are then used to interpolate and compute jump conditions as needed at various points on the interface in implementing the immersed interface method.

5.3 Discontinuous coefficients

Besides providing the possibility of a more accurate version of the immersed boundary method, the immersed interface approach has the advantage that it can be applied to a wide class of interface problems where discontinuities in the solution or its derivatives across Γ arise from factors in the equations other than a singular source term along Γ. For example, the Poisson problem

$$\nabla \cdot (\beta \nabla u) = f$$

with variable coefficients $\beta(x,y)$ which are discontinuous across an interface Γ has a solution u which is continuous but has a discontinuous normal derivative across Γ. This arises from the physical jump condition that the flux $\beta\frac{\partial u}{\partial n}$ must be continuous, which gives the jump condition required to implement the immersed interface method. Problems of this form with discontinuous coefficients arise in many applications, *e.g.*, electrostatic or steady state heat distribution problems in nonhomogeneous media, and also in the Stokes flow problem described above if the fluids inside and outside the balloon or bubble have different viscosities.

LeVeque and Li [44] showed how second-order accuracy could be obtained using the same idea of a Taylor series expansion as described in Section 5 for this case, with the 5-point Laplacian replaced by a 6-point stencil whose weights depend on the coefficients β and the location and curvature of the interface. Unfortunately, choosing the appropriate six points to use and obtaining a method with uniformly good stability properties and smooth behavior as the interface moves turns out to be difficult for some problems. The problems with this approach have been the subject of more recent research, and several improvements and alternatives have been proposed which give more robust results. See for example [9], [25], [47], [49], [72], [74]. Wiegmann and Bube [73] have also extended this approach to a nonlinear equation with discontinuous coefficients and singular sources.

6 Advection-diffusion equations

We now turn to some problems where the finite-volume formulation of Figure 1(d) is quite natural to use. Suppose we wish to model the advection and diffusion of some substance in a given velocity field (*e.g.*, the transport of oxygen or other substances in the blood or tissue, the dispersal of a pheromone). In this section we assume the velocity field is known, and in the next section we will illustrate how this technique can be applied together with immersed interface ideas to compute time-varying velocities in incompressible flow using a streamfunction-vorticity formulation.

The finite-volume framework is natural if we want to insure exact conservation. Also, for the advection portion of the algorithm we can use high-resolution "shock-capturing" finite-volume methods which can deal with steep gradients or even discontinuities in the concentration without introducing spurious numerical oscillations or excessive numerical diffusion. The CLAWPACK software [39], [43], which applies to more general nonlinear hyperbolic systems of equations, is used as the basis for the Cartesian grid method we have developed.

Cartesian grid finite-volume methods for fluid dynamics with embedded boundaries have been the subject of much work recently, see for example [2], [4], [5], [18], [19], [30], [32], [37], [40], [41], [58], [60], [61]. Here we briefly review the method developed in [10]. We solve the advection-diffusion equation

$$\kappa q_t + u q_x + v q_y = (D q_x)_x + (D q_y)_y \tag{17}$$

in two space dimensions, where $q(x,y,t)$ is a density or concentration, $(u(x,y,t), v(x,y,t))$ is the specified velocity field, $D(x,y)$ is the diffusion coefficient, and $\kappa(x,y)$ is a capacity function. This function could represent, for example, heat capacity if q is the temperature, or porosity in a porous medium if q is the concentration of a transported solute. A fundamental feature of the numerical method we have developed is the use of a discrete version of κ which also incorporates information about the fraction of a computational grid cell which lies in the physical domain.

This is used to obtain a method which is conservative and as well as accurate and stable on the cut cells.

The advection-diffusion equation (17) is solved using a fractional step approach in which we alternate between solving the advection equation

$$\kappa q_t + u q_x + v q_y = 0 \tag{18}$$

and the diffusion equation

$$\kappa q_t = (D q_x)_x + (D q_y)_y. \tag{19}$$

This allows a high-resolution explicit method to be used for the advection equation while an implicit method is used for the diffusion equation.

6.1 Advection

We assume the incompressible velocity field is specified via a streamfunction $\psi(x, y)$ with

$$u = \psi_y, \qquad v = -\psi_x.$$

We assume that ψ is known at all grid points in the rectangular computational domain, with ψ identically constant in any "solid body" region where there is no fluid. The points where we need ψ are the corners of the finite-volume grid cells, *e.g.*, $\psi_{i-1/2,j-1/2}$ is the value at the lower left corner of the (i, j) cell. Then differencing ψ between any two corners gives the integral of the normal velocity along the edge in between, so we easily obtain average edge velocities

$$\begin{aligned} U_{i-1/2,j} &= \frac{1}{\Delta y}(\psi_{i-1/2,j+1/2} - \psi_{i-1/2,j-1/2}) \\ V_{i,j-1/2} &= -\frac{1}{\Delta x}(\psi_{i+1/2,j-1/2} - \psi_{i-1/2,j-1/2}). \end{aligned} \tag{20}$$

If we have accurate pointwise values of ψ then the average velocities computed by (20) will be accurate even if the boundary cuts through an edge of the cell. In this case the velocity does *not* approximate the fluid velocity but rather the average over the edge taking into account the fact that the normal velocity is zero over part of the edge. See [10] for more discussion and the relation to "Darcy velocities" in porous media flow. These velocities are exactly what are needed in computing fluxes across the edge in the high-resolution methods for the advection equation. Moreover, these velocities are divergence free on the discrete grid in the sense that

$$\frac{1}{\Delta x}(U_{i+1/2,j} - U_{i-1/2,j}) + \frac{1}{\Delta y}(V_{i,j+1/2} - V_{i,j-1/2}) = 0, \tag{21}$$

as is easily verified.

How ψ is determined depends on the problem. It might be specified analytically for steady-state flow in a simple geometry. Or we might compute it numerically based on some model of the fluid dynamics. In [10] we consider flow in a porous medium with embedded objects, in which case ψ is obtained using Darcy's law by solving a Poisson problem. Accurate pointwise values of ψ on the Cartesian grid can be obtained using a variant of the immersed interface method.

In Section 7 we discuss extensions of this approach to solving the streamfunction-vorticity formulation of the time-dependent incompressible Navier-Stokes equations. In this case the streamfunction must be recomputed in each time step based on the vorticity distribution. Figures 3 through 5 show examples discussed later in which this solution is coupled with an advection equation for a tracer in the fluid. Streamlines of the flow at one instant are shown as contour lines of ψ, and the concentration of the advected tracer at this instant are also shown in these figures. The gray tracer is injected at the left boundary starting at time $t = 0$. In these examples there is no diffusion, $D = 0$ in (17), and the sharp front in the concentration is well captured.

6.2 Diffusion

We also solve the diffusion equation in a finite volume setting, using the capacity function to handle the irregular geometry. The diffusion step of the fractional step scheme is given by

$$\kappa q_t = \nabla \cdot (D(x,y)\ \nabla q) \tag{22}$$

subject to Neumann boundary conditions on the boundary of embedded objects. This equation is discretized by using numerical fluxes at cell edges to update the average value of q in each cell. These fluxes are the approximation at cell edges to the quantity $-D\partial q/\partial n$, where $\partial q/\partial n$ is the derivative normal to the cell edge. At the edges of regular cells, these fluxes are computed numerically by simply differencing the values of q in cells adjacent to the edge. At the edges of irregular cells, we use bilinear interpolation to approximate the normal derivatives at partial cell edges. The details of these approximations can be found in [10].

Using these numerical fluxes, we discretize (22) using the second order, implicit Crank-Nicolson scheme. Non-zero fluxes at the boundary of the embedded objects are incorporated into the discretization as known source terms. This discretization leads to a linear system of equations which is not symmetric, because of the presence of the irregular cells, and so cannot be solved using standard fast solvers. We use instead a linear iterative method designed for non-symmetric systems. In particular, we have found that the stabilized bi-conjugate gradient method (BiCGSTAB) of Van der Vorst [20] works quite well.

Because we only use bilinear interpolation to compute fluxes at irregular cell edges, our method is formally only first order at these edges. However, numerical results show that the method achieves global second order accuracy. Another method developed by Johansen and Collela [37] is formally second order and could be considered as an alternative to our approach.

7 Streamfunction-vorticity equations

In the previous section, we described a numerical method for solving advection-diffusion equation in complex geometries. However, we only considered transport in a steady state velocity field $\vec{u}(x,y) = (u(x,y), v(x,y)) = (\psi_y(x,y), -\psi_x(x,y))$. We now want to consider methods for computing a time dependent velocity field. Also, since fluid viscosity and friction at boundaries plays an important role in many biological fluid dynamics problems, we also want to include the affects of fluid viscosity and impose a no-slip boundary condition on the velocity field at solid boundaries.

We begin by considering the three-dimensional Navier-Stokes equations, given by

$$\begin{aligned} \vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} + \nabla p =& \nu \nabla^2 \vec{u} \\ \nabla \cdot \vec{u} =& 0, \end{aligned} \tag{23}$$

where $\vec{u}(x, y, z, t) = (u(x, y, z, t), v(x, y, z, t), w(x, y, z, t))$ is the velocity, $p(x, y, z, t)$ is the pressure field, and ν the kinematic viscosity. We have assumed that the density of the fluid is constant and equal to 1. This is sometimes called the *primitive* variable formulation since desired quantities such as velocity and pressure are solved for directly. This formulation serves as the basis for the projection method, a computational method developed by Chorin [13] and in common use in the biological fluid dynamics community [21],[23].

One potential drawback to solving the primitive variable equations directly is that we are not given any explicit boundary conditions on pressure. To eliminate the pressure from equations (23), we can take the curl of the momentum equation and obtain an equation for the *vorticity* $\vec{\omega} = \nabla \times \vec{u}$:

$$\vec{\omega}_t + (\vec{u} \cdot \nabla)\, \vec{\omega} - (\vec{\omega} \cdot \nabla)\, \vec{u} = \nu \nabla^2 \omega \tag{24}$$

This equation is most tractable in two space dimensions, where $\vec{\omega} \equiv (0, 0, \omega)$ and the vorticity transport equation (24) reduces to

$$\omega_t + (\vec{u} \cdot \nabla)\omega = \nu \nabla^2 \omega \tag{25}$$

where now, $\vec{u} = (u, v)$ and $\omega = v_x - u_y$ is the scalar vorticity. Because the velocity field is continuous and divergence free, we can introduce a function $\psi(x, y, t)$ for which

$$u = \psi_y, \qquad v = -\psi_x \tag{26}$$

Using the definition of the vorticity, we have that ψ satisfies the elliptic equation $\nabla^2 \psi = -\omega$. The function $\psi(x, y, t)$ has the property that

$$(\vec{u} \cdot \nabla)\psi = 0 \tag{27}$$

so that level curves of ψ trace out instantaneous particle paths or *streamlines* of the flow. For this reason, ψ is called a *streamfunction*.

The streamfunction-vorticity equations for incompressible flow are then given by

$$\begin{aligned} \omega_t + (\vec{u} \cdot \nabla)\omega =& \nu \nabla^2 \omega \\ \nabla^2 \psi =& -\omega \\ u = \psi_y, \quad & v = -\psi_x \end{aligned} \tag{28}$$

subject to boundary conditions which we discuss momentarily. For a more detailed discussion of these equations, one can consult any one of a number of elementary fluid dynamics texts, including [1] and [15].

The streamfunction-vorticity equations have certain computational and modeling advantages over the primitive-variable formulation. First, these equations are naturally decoupled into an advection-diffusion equation for the vorticity and an elliptic equation for the streamfunction. Second, in two space dimensions, the vorticity is a conserved quantity and so we can

take advantage of high resolution finite volume advection algorithms to solve the advection-diffusion equation for vorticity. Finally, the numerically computed velocity field obtained by differencing the streamfunction as in (20) satisfies the discrete divergence free condition (21) exactly. From a modeling point of view, one may actually prefer to compute vorticity and streamlines directly, rather than velocity or pressure. For example, in [70], Wang uses this formulation to compute vortex shedding behind flapping insect wings. In [12] and [11], Cheer and Koehl compute the mass flux between filtering appendages, which they represent as solid objects embedded in a flow field. Such a computation can be done quite easily if the streamfunction for the flow is known. Finally, quantities of aerodynamic interest, such as lift and drag forces are easily computed from a given vorticity field.

Boundary conditions appropriate for the above equations are derived from desired boundary conditions on the primitive variables. A no-flow condition $\vec{u} \cdot \vec{n} = 0$, where $\vec{n}$ is the vector normal to a boundary, is imposed by requiring that the tangential derivative of ψ, $\partial\psi/\partial\tau$, along the boundary be zero, or that ψ be constant along solid boundaries. Mathematically, this corresponds to imposing a Dirichlet condition on ψ at the solid boundaries. To impose the no-slip condition, we must require that $\vec{u} \cdot \vec{\tau} = 0$, or that $\partial\psi/\partial n = 0$. This corresponds to a Neumann condition on ψ. If our domain is multiply-connected, we must also impose a circulation condition on each object to insure the uniqueness of the streamfunction solution. The fact that we must satisfy both $\psi =$ constant and $\partial\psi/\partial n = 0$ on ψ appears to overdetermine the elliptic problem for the streamfunction. However, when we consider the elliptic problem *coupled* with the vorticity transport equation, we see that we have exactly the right number of boundary conditions. This is seen most easily by eliminating ω from the equations and considering the resulting biharmonic equation for ψ.

When considering how to solve the coupled system of equations numerically, however, we are faced with the problem that we do not have any explicit boundary conditions on ω, and seemingly too many for ψ. Mathematically, a shear gradient in the boundary layer at a solid boundary is coincident with the presence of non-zero vorticity there. Although physically the mechanism by which vorticity appears near solid boundaries is not so clear, it is nonetheless common to appeal to the notion of "vorticity generation" near solid boundaries and to consider numerical methods which introduce vorticity into the domain at these boundaries in an amount that exactly cancels the slip velocity, thereby satisfying the boundary condition $\psi_n = 0$. Considerable research has gone into deriving schemes for introducing vorticity into the domain or coming up with conditions on the vorticity that insure that the no-slip boundary condition will be satisfied. See for example [65], [36], [14], [59], [3], [71]. Our approach is in the spirit of vorticity generation, in that we derive a flux condition on the vorticity that introduces vorticity into the domain and cancels the slip velocity at the boundary.

For the remainder of this section, we discuss our approach to solving equations (28) on a Cartesian grid in the presence of embedded complex geometry. A typical example of the geometry we have considered so far is the multiply-connected exterior region shown in Figure 2. Although our methods are applicable to very general geometries, we focus here on the exterior flow problem for a couple of reasons. First, the problem of flow around isolated objects arises in a number of problems in bio-fluid mechanics, including flow around swimming organisms [22], filtering appendages [12], [11], [38], through tissue past capillaries [66] and flapping wings [70]. Second, we wanted to be able to compare our results to standard benchmark problems, of which the classic problem is flow past a cylinder. Finally, we wish to discuss the additional constraints that one must impose on the streamfunction in a multiply-connected domains.

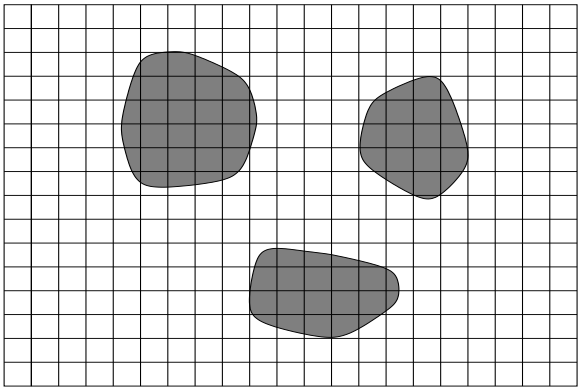

Figure 2: A typical grid on which the streamfunction vorticity equations are solved. The flow domain is the region exterior to the embedded objects. The objects are assumed to be stationery and rigid.

We assume that the embedded objects shown in Figure 2, labeled $\Omega_j, j = 1, \ldots M_{bodies}$ are stationary and rigid, and so we must satisfy both a no-normal flow and a no-slip boundary condition on each object. On the boundary of the computational domain, we impose inflow conditions at the left edge, outflow conditions at the right edge, and free stream conditions on the top and bottom edges. The top and bottom act like free-slip boundaries parallel to the direction of the mean flow.

To solve the advection-diffusion equation in (28) for the vorticity, we use the fractional step approach described in Section 6. First, a given vorticity field is advected using a velocity field computed from the current streamfunction. The irregular geometry is handled using the capacity function, as described in Section 6. In this step, we impose no-flux boundary conditions on the embedded objects. The advection step is followed by the diffusion step described in Section 6.2. In this diffusion step, we impose a non-zero flux condition $\partial\omega/\partial n = g$ at the solid boundaries. This flux g is determined by solving an auxiliary Stokes flow problem, described below. In the process of solving this Stokes flow problem, we also determine a streamfunction ψ which we use to update our velocity field, and, because it satisfies the Stokes flow problem, satisfies $\psi_n = 0$.

The auxiliary Stokes flow problem is given in terms of a node-based quantity $\hat{\omega}$, obtained as a result of averaging the cell-averaged quantity ω onto grid nodes. The Stokes flow problem is

$$\begin{aligned} \hat{\omega}_t &= \nu\nabla^2\hat{\omega}, & \partial\hat{\omega}/\partial n &= g, \quad (x,y) \in \partial\Omega \\ \nabla^2\psi &= -\hat{\omega}, & \psi &= \overline{\psi}_j, \quad (x,y) \in \partial\Omega_j \\ \psi_n &= 0, & (x,y) &\in \partial\Omega \\ \int_{\partial\Omega_j} \frac{\partial\hat{\omega}}{\partial n}\, ds &= 0. & j &= 1, \ldots M_{bodies} \end{aligned} \tag{29}$$

where the flux g of vorticity along the boundaries of the objects is an unknown in the problem. Physically, the vorticity distribution $\hat{\omega}(x, y, t + \Delta t)$ that results from applying the flux g over a single time step Δt has the property that the solution to $\nabla^2\psi = -\hat{\omega}$ satisfies $\psi_n = 0$, in addition to the Dirichlet condition $\psi = \overline{\psi}_j$.

The constants $\overline{\psi}_j, j = 1, \ldots M_{bodies}$ are also unknowns in the problem and are determined so that the circulation Γ_j

$$\Gamma_j = \int_{\partial\Omega_j} \vec{u} \cdot \vec{\tau}\, ds = \int_{\partial\Omega_j} \psi_n\, ds = \int_{\Omega_j} \omega\, dA \tag{30}$$

around each object Ω_j remains constant in time. This condition is satisfied by requiring that the net flux of vorticity into the bodies be zero for all time, or that the last equation in (29) hold. Note that we don't require that the circulation around the objects be zero; the circulation is determined initially by our choice of ψ_n (or, equivalently by the initial vorticity distribution inside each object). Of course, when the bodies are stationary, then the no-slip condition is just $\psi_n = 0$ and inside of each body we have $\omega \equiv 0$, so that for each body, $\Gamma_j = 0$ for all time.

To discretize the equations for the Stokes flow problem in (29), we use a modified version of the Immersed Interface Method described in Section 5 of the sort developed in [9], [75] and [72]. The IIM, as applied to these problems, gives second order accurate solutions to ψ and $\hat{\omega}$, and $\psi_n = 0$ is satisfied to second order. Once we have discretized the equations, we can easily solve them by recognizing that we can set up a linear system for the flux function g evaluated at discrete locations along the solid boundaries. This linear system can either be solved directly, by explicitly forming a matrix equation, or solved iteratively, using for example, the Generalized Minimum Residual algorithm (GMRES). Numerical tests demonstrate that this linear system is quite well conditioned [9].

7.1 Numerical results for the streamfunction-vorticity equations

Comparison of our results with those from the standard benchmark problem, flow around a cylinder, show that algorithm yields results are in excellent agreement with other published computational and experimental results. In [9], Calhoun compared the results of the above algorithm with the experimental results of Coutanceau and Bouard [16] and [17] and the computational results of Fornberg [29] and found that velocity fields and vorticity wakes were quite well reproduced. For low Reynolds number flows ($Re < 40$), the flow remains steady and symmetric and any small perturbations in the wake of the cylinder are damped. At around Reynolds number 50, the the streamline wake begins to oscillate and by Reynolds number 100, vorticies are shed behind the cylinder forming the well known von Karman vortex street.

In the three sets of plots in Figures 3 to 5, we show results of our numerical algorithm for more complicated flows at Reynolds numbers varying from $Re = 20$ for Figure 3 to $Re \approx 100$ for Figure 5. In each case, the Reynolds number is based on the diameter of the object(s). In each set of plots, we show the streamfunction, advection of a tracer (grey shaded material), the horizontal velocity field and the vorticity distribution at some fixed time. For the calculation shown in Figure 3, the flow has reached a steady state, and there is no vortex shedding in the wake behind the bodies. From the plots of the horizontal velocities, one can see that the boundary layers around each object is fairly thick, and from the tracer plots, significantly more tracer is flowing around the objects than through them. In the calculation shown in Figure 4 at $Re = 40$, the shear gradient is steeper (and the boundary layer thinner) and more tracer material flows between the embedded objects. The last set of plots in Figure 5 provides an example of vortex shedding behind a complex bluff body. In this example, the boundary layers are fairly thin, although there are large areas where the velocity is essentially zero due to the shielding affect of the body on the flow.

Our goal in showing these plots is to demonstrate that our algorithm produces realistic results in some biologically relevant regimes. Our choice of examples was motivated to some extent by the work of Cheer and Koehl [12], [11], who considered flow around fixed objects. Wang [70] also considers flow around a fixed wing, and so in principle our scheme could be used for these calculations as well. However, these examples, as well as most other interesting examples in biofliud mechanics, really involve moving and deforming boundaries and more complicated boundary conditions. For these reasons, we are now considering methods for incorporating these features into our finite difference/finite volume methods.

Adding the more complicated boundary conditions should be a simple modification to the code. We would like to consider problems in which the boundaries of the embedded objects are absorbing or releasing some tracer material (with concentration q) into the flow. In general, the flux rate of this tracer will depend on the amount of tracer present near the boundary, and so this will lead to a boundary condition on the tracer that takes the form $\partial q/\partial n = f(q, \mathbf{x}, t)$, where f may be a linear or non-linear function of q. This type of boundary condition appears, for example, in oxygenation of tissue, where the flux of oxygen through capillary walls is proportional to the partial pressure of oxygen in the tissue [66].

We are currently working on extending our method to moving boundaries. Successful Cartesian-grid finite-volume methods for moving geometries have been developed for compressible flow, see [31] for one example. Our goal for incompressible flow is still to solve the streamfunction-vorticity equations for the fluid motion on a uniform grid, along with advection-diffusion equations for any tracer concentrations. The volume of the finite-volume cells will now be time-dependent, however, as the boundaries move through the grid. In the future we also hope to combine this algorithm with an immersed interface method for flexible membranes immersed in the fluid.

8 Summary

We have presented an overview of some recent work on Cartesian grid methods for solving fluid dynamics in complex geometry. The immersed boundary and immersed interface methods are finite-difference methods which can deal with discontinuities in the solution across boundaries or interfaces. These methods are often used for problems with moving boundaries as well, in cases where the motion of the boundary is coupled to the fluid dynamics.

Finite-volume methods have some advantages for problems where conservation is important, and steep gradients in the solution can be well-resolved by high-resolution methods. The finite-volume method described here for advection-diffusion and the streamfunction-vorticity equations can deal with complex geometries. We have developed an approach to determine the flux of vorticity at a no-slip wall based on the behavior of the streamfunction at the wall (as computed by a finite-difference immersed interface method). This gives the required boundary condition for vorticity.

References

[1] D. J. Acheson. *Elementary Fluid Dynamics.* Oxford Applied Mathematics and Computing Science Series. Clarendon Press, 1990.

[2] A. S. Almgren, J. B. Bell, P. Colella, and T. Marthaler. A Cartesian grid projection method for the incompressible Euler equations in complex geometries. *SIAM J. Sci. Comput.*, 18:745–762, 1997.

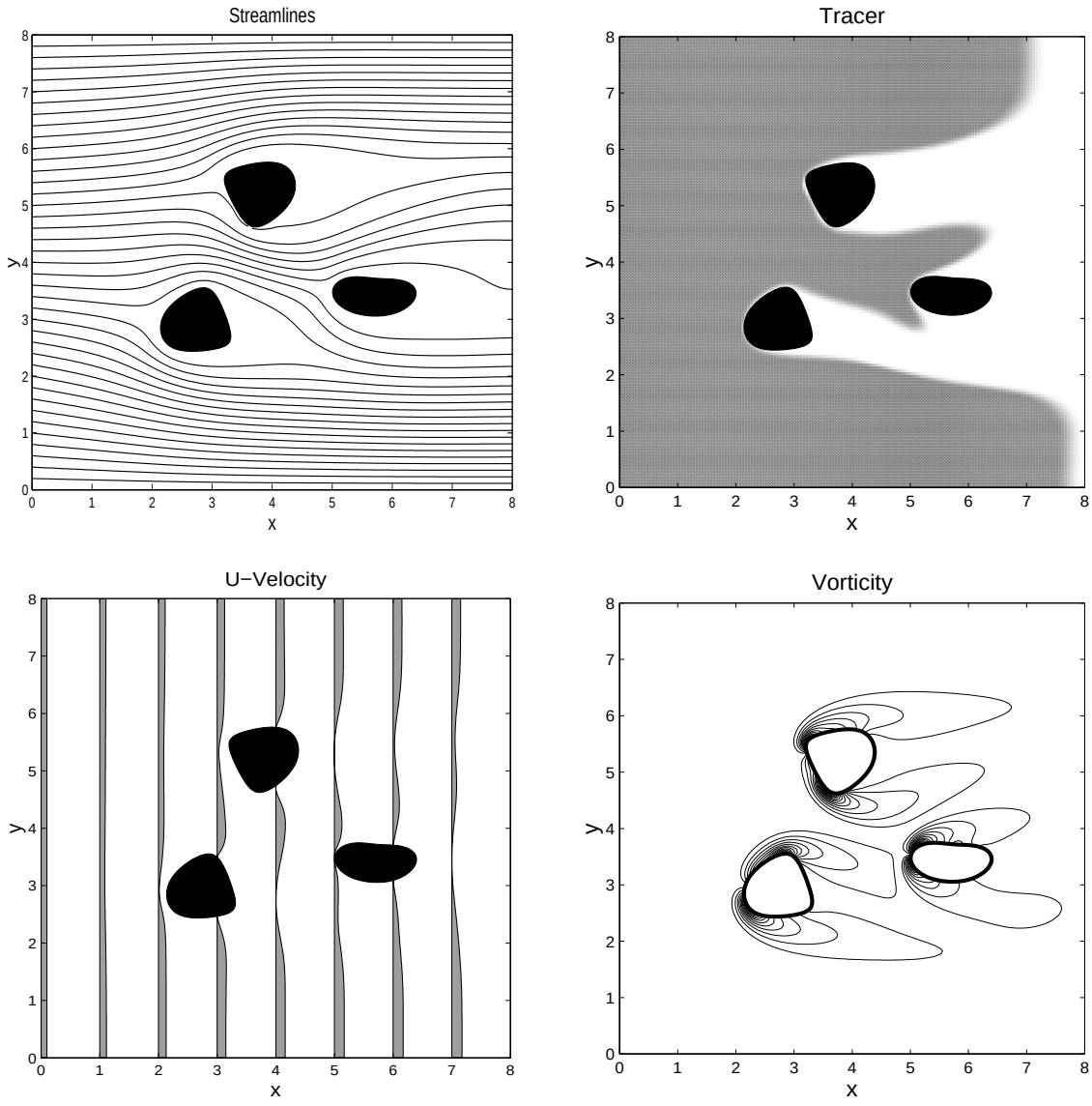

Figure 3: Flow past objects at $Re = 20$ Several quantities are illustrated at the same instant in time.

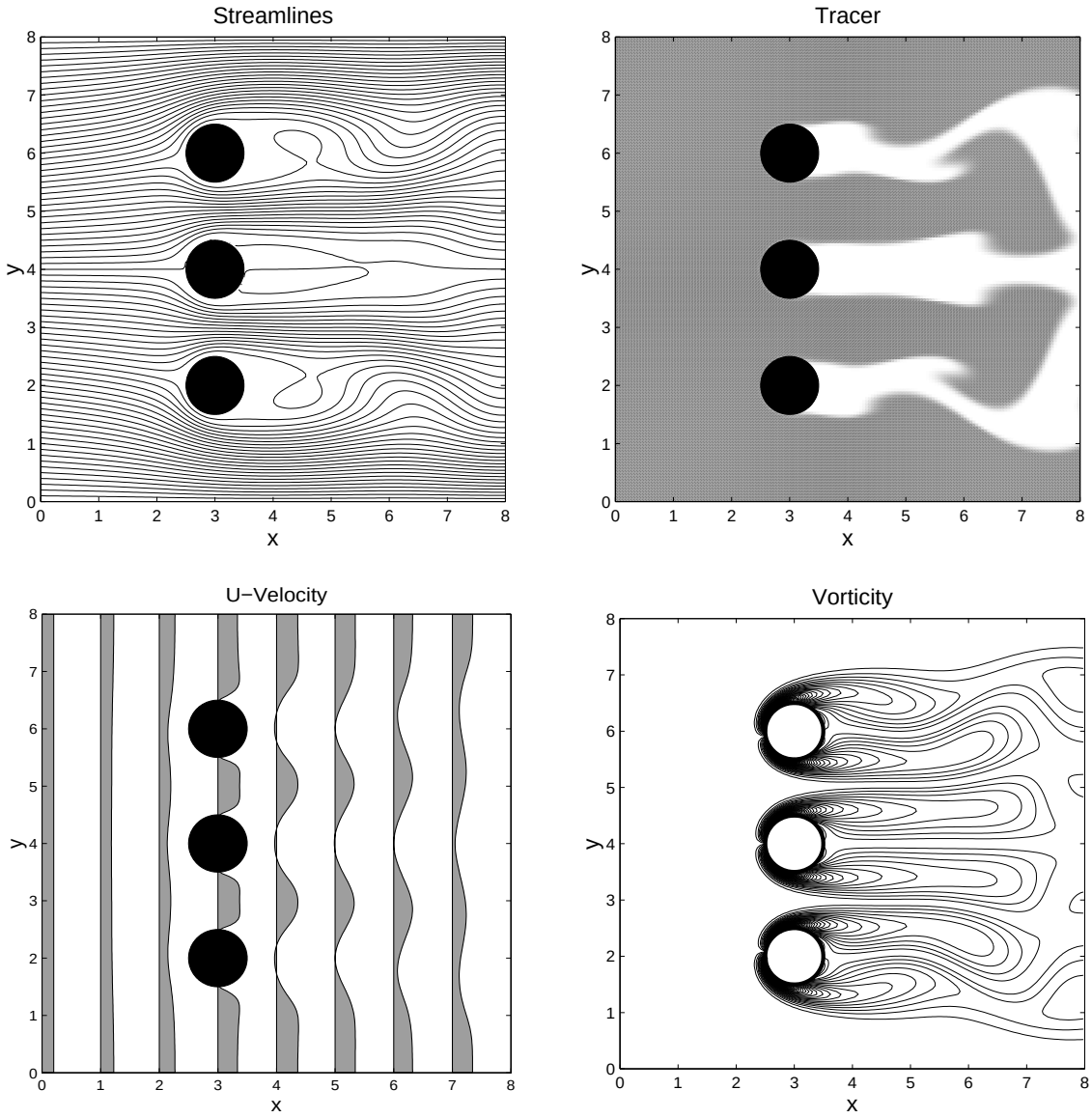

Figure 4: Flow past objects at $Re = 40$. Several quantities are illustrated at the same instant in time.

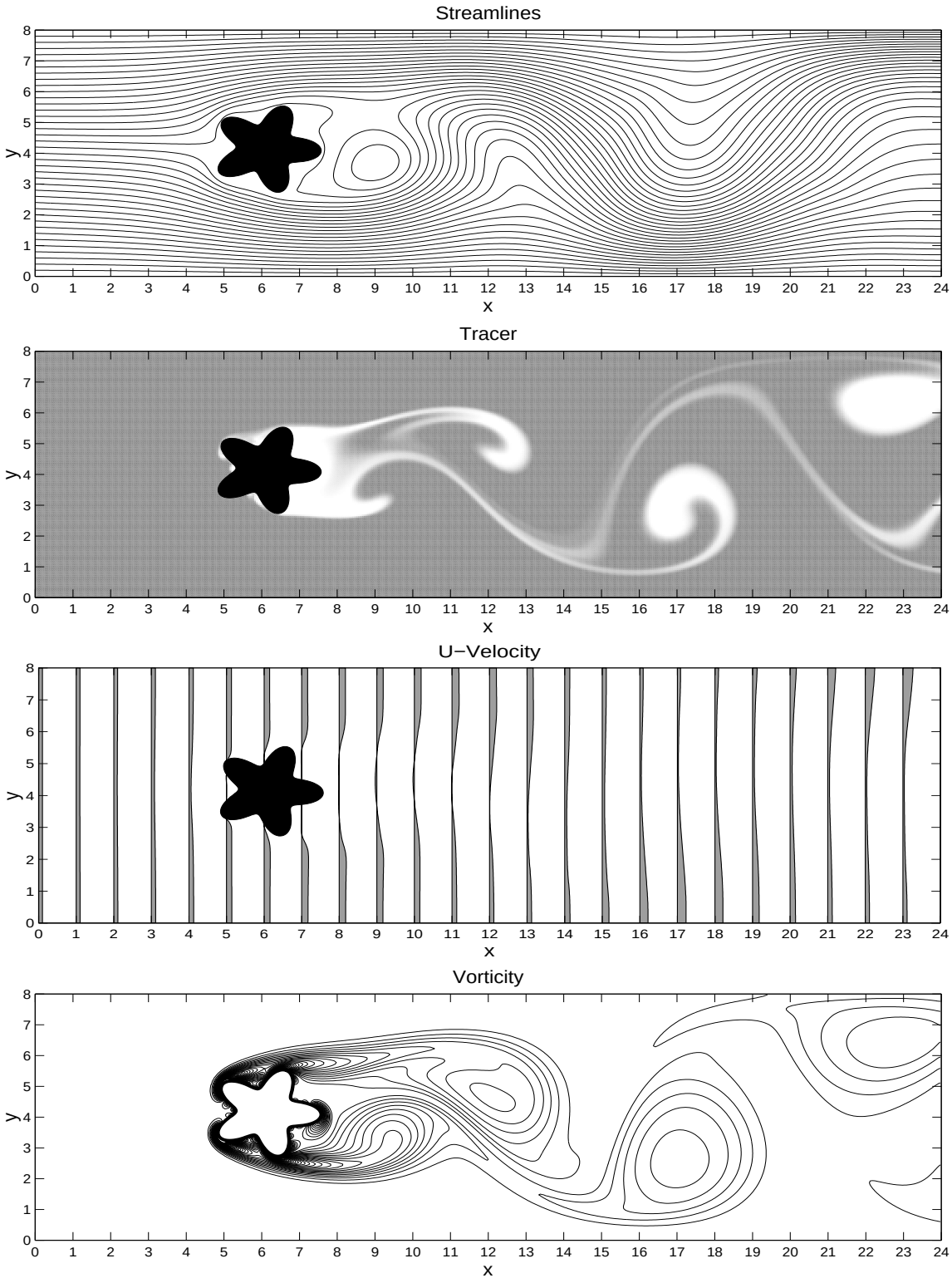

Figure 5: Flow at $Re \approx 100$ with vortex shedding. Several quantities are illustrated at the same instant in time.

[3] C. Anderson. Vorticity Boundary Conditions and Boundary Vorticity Generation for Two-dimensional Viscous Incompressible Flows. *J. Comput. Phys.*, 80:72–97, 1989.

[4] M. Berger and R. J. LeVeque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. AIAA paper AIAA-89-1930, 1989.

[5] M. Berger and R. J. LeVeque. Stable boundary conditions for Cartesian grid calculations. *Computing Systems in Engineering*, 1:305–311, 1990.

[6] R. P. Beyer. *A computational model of the cochlea using the immersed boundary method.* PhD thesis, University of Washington, 1989.

[7] R. P. Beyer. A computational model of the cochlea using the immersed boundary method. *J. Comput. Phys.*, 98:145–162, 1992.

[8] R. P. Beyer and R. J. LeVeque. Analysis of a one-dimensional model for the immersed boundary method. *SIAM J. Num. Anal.*, 29:332–364, 1992.

[9] D. Calhoun. *A Cartesian grid method for solving the streamfunction-vorticity equations in irregular geometries.* PhD thesis, University of Washington, 1999.

[10] D. Calhoun and R. J. LeVeque. Solving the advection-diffusion equation in irregular geometries. submitted to J. Comput. Phys., 1998.
(`ftp://amath.washington.edu/pub/rjl/papers/dc-rjl:advdiff.ps.gz`).

[11] A. Cheer and M. Koehl. Fluid flow through filtering appendages of insects. *IMA Journal of Mathematics Applied in Medicine and Biology*, 4:185–199, 1987.

[12] A. Cheer and M. Koehl. Paddles and Rakes: Fluid Flow through ristled Appendages of Small Organisms. *J. Theor. Biol.*, 129:17–39, 1987.

[13] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[14] A. J. Chorin. Numerical study of slightly viscous flow. *J. Fluid Mech.*, 75(4):785–96, 1973.

[15] A. J. Chorin and J. E. Marsden. *A Mathematical Introduction to Fluid Mechanics.* Springer-Verlag, 1979.

[16] M. Coutanceau and R. Bouard. Experimental determination of the main features of the vicous flow in the wake of a circular cylinder in uniform translation. part 1. steady flow. *J. Fluid Mech.*, 79:231–256, 1977. Part 1.

[17] M. Coutanceau and R. Bouard. Experimental determination of the main features of the vicous flow in the wake of a circular cylinder in uniform translation. part 2. unsteady flow. *J. Fluid Mech.*, 79:257–272, 1977. Part 2.

[18] M. S. Day, P. Colella, M. J. Lijewski, C. A. Rendleman, and D. L. Marcus. Embedded boundary alogirthms for solving the Poisson equation on complex domains. Preprint LBNL-41811, Lawrence Berkeley Lab, 1998.

[19] D. De Zeeuw and K. Powell. An adaptively-refined Cartesian mesh solver for the Euler equations. *J. Comput. Phys.*, 104:56–68, 1993.

[20] H. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13:631–644, 1992.

[21] R. Dillon, L. Fauci, A. Fogelson, and D. Gaver. Modeling Biofilm Processes Using the Immersed Boundary Method. *J. Comput. Phys.*, 129:57–73, 1996.

[22] R. Dillon, L. Fauci, and D. Gaver. A Microscale Model of Bacterial Swimming, Chemotaxis and Substrate Transport. *J. Theor. Biol.*, 177:325–340, 1995.

[23] L. Fauci and C. S. Peskin. A computational model of aquatic animal locomotion. *J. Comput. Phys.*, 77:85–108, 1988.

[24] L. J. Fauci. Interaction of oscillating filaments – a computational study. *J. Comput. Phys.*, 86:294–313, 1990.

[25] A. Fogelson and J. Keener. Immersed interface methods for Neumann and related problems in two and three dimensions. preprint, 1997.

[26] A. L. Fogelson. A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting. *J. Comput. Phys.*, 56:111–134, 1984.

[27] A. L. Fogelson. Mathematical and computational aspects of blood clotting. In B. Wahlstrom, editor, *Proceedings of the 11th IMACS World Congress on System Simulation and Scientific Computation, Vol. 3*, pages 5–8. North Holland, 1985.

[28] A. L. Fogelson and C. S. Peskin. Numerical solution of the three dimensional stokes equations in the presence of suspended particles. In *Proc. SIAM Conf. Multi-phase Flow*. SIAM, June 1986.

[29] B. Fornberg. A numerical study of steady viscous flow past a circular cylinder. *J. Fluid Mech.*, 98:819–855, 1980.

[30] H. Forrer. *Boundary Treatments for Cartesian-Grid Methods*. PhD thesis, ETH-Zurich, 1997.

[31] H. Forrer and M. Berger. Flow simulations on Cartesian grids involving complex moving geometries. In R. Jeltsch, editor, *Proc. 7'th Intl. Conf. on Hyperbolic Problems*, pages 315–324. Birkhäuser Verlag, 1998.

[32] H. Forrer and R. Jeltsch. A higher-order boundary treatment for Cartesian-grid methods. *J. Comput. Phys.*, 140:259–277, 1998.

[33] R. Glowinski, T.-S. Pan, and J. Periaux. A fictitious domain method for Dirichlet problem and applications. *Comp. Meth. Appl. Mech. Eng.*, 111:283–303, 1994.

[34] R. Glowinski, T.-S. Pan, and J. Periaux. A fictitious domain method for external incompressible viscous flow modeled by Navier-Stokes equations. *Comp. Meth. Appl. Mech. Eng.*, 112:133–148, 1994.

[35] T. Y. Hou, Z. Li, H. Zhao, and S. Osher. A hybrid method for moving interface problems with application to the Hele-Shaw flow. *J. Comput. Phys.*, 134:236–252, 1997.

[36] M. Israeli. On the Evaluation of Iteration Parameters for the Boundary Vorticity. *Studies in Applied Mathematics*, LI(1):67–71, March 1972.

[37] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147:60–85, 1998.

[38] M. Koehl. Fluid flow through hair-bearing appendages: Feeding, smelling, and swimming at low and intermediate Reynolds numbers. In C. P. Ellington and T. J. Pedley, editors, *Biological Fluid Dynamics*, volume 49, pages 157–82. Soc. Exp. Biol. Symp, 1995.

[39] R. J. LeVeque. CLAWPACK software. available from `netlib.bell-labs.com` in `netlib/pdes/claw` or from `http://www.amath.washington.edu/~rjl/clawpack.html`.

[40] R. J. LeVeque. Cartesian grid methods for flow in irregular regions. In K. W. Morton and M. J. Baines, editors, *Num. Meth. Fl. Dyn. III*, pages 375–382. Clarendon Press, 1988.

[41] R. J. LeVeque. High resolution finite volume methods on arbitrary grids via wave propagation. *J. Comput. Phys.*, 78:36–63, 1988.

[42] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser-Verlag, 1990.

[43] R. J. LeVeque. Wave propagation algorithms for multi-dimensional hyperbolic systems. *J. Comput. Phys.*, 131:327–353, 1997.

[44] R. J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 31:1019–1044, 1994.

[45] R. J. LeVeque and Z. Li. Immersed interface methods for Stokes flow with elastic boundaries or surface tension. *SIAM J. Sci. Comput.*, 18:709–735, 1997.

[46] Z. Li. *The Immersed Interface Method — A Numerical Approach for Partial Differential Equations with Interfaces.* PhD thesis, University of Washington, 1994.

[47] Z. Li. A fast iterative algorithm for elliptic interface problems. *SIAM J. Numer. Anal.*, 35:230–254, 1998.

[48] Z. Li. The immersed interface method using a finite element formulation. *Applied Numer. Math.*, 27:253–267, 1998.

[49] Xu-Dong Liu, Ronald P. Fedkiw, and Myungjoo Kang. A boundary condition capturing method for poisson's equation on irregular domains. CAM Report 99-15, UCLA Mathematics Department, 1999.

[50] A. Mayo. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Num. Anal.*, 21:285–299, 1984.

[51] A. Mayo and A. Greenbaum. Fast parallel iterative solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Sci. Stat. Comput.*, 13:101–118, 1992.

[52] A. A. Mayo and C. S. Peskin. An implicit numerical method for fluid dynamics problems with immersed elastic boundaries. *Contemp. Math.*, 141:261–277, 1993.

[53] C. S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[54] C. S. Peskin. Lectures on mathematical aspects of physiology. *Lectures in Appl. Math.*, 19:69–107, 1981.

[55] C. S. Peskin and D. M. McQueen. Modeling prosthetic heart valves for numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 37:113–132, 1980.

[56] C. S. Peskin and D. M. McQueen. Computer-assisted design of pivoting-disc prosthetic mitral valves. *J. Thorac. Cardiovasc. Surg.*, 86:126–135, 1983.

[57] C. S. Peskin and D. M. McQueen. Computer-assisted design of butterfly bileaflet valves for the mitral position. *Scand. J. Thorac. Cardiovasc. Surg.*, 19:139–148, 1985.

[58] K. Powell. Solution of the Euler and Magnetohydrodynamic Equations on Solution-Adaptive Cartesian Grids. Von Karman Institute for Fluid Dynamics Lecture Series, 1996.

[59] L. Quartapelle and F. Valz-griz. Projection conditions on the vorticity in viscous incompressible flows. *Int. J. Numer. methods. fluids*, 1:129–144, 1981.

[60] J. J. Quirk. An alternative to unstructured grids for computing gas-dynamic flow around arbitrarily complex 2-dimensional bodies. *Comput. Fluids*, 23:125–142, 1994.

[61] J. J. Quirk. A Cartesian grid approach with hierarchical refinement for compressible flows. ICASE Report No. TR-94-51, NASA Langley Research Center, 1994.

[62] J. M. Stockie and S. I. Green. Simulating the motion of flexible pulp fibres using the immersed boundary method. *J. Comput. Phys.*, 147:147–165, 1998.

[63] J. M. Stockie and B. T. R. Wetton. Stability analysis for the immersed fiber problem. *SIAM J. Appl. Math.*, 55:1577–1591, 1995.

[64] D. Sulsky and J. U. Brackbill. A numerical method for suspension flow. *J. Comput. Phys.*, 96:339–368, 1991.

[65] A. Thom. The flow past circular cylinders at low speeds. *Proc. Roy. Soc. A*, 141:651, 1933.

[66] M. Titcombe and M. J. Ward. An asymptotic study of oxygen transport from multiple capillaries to skeletal muscle tissue. submitted to SIAM J. Appl. Math., 1998.

[67] C. Tu and C. S. Peskin. Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods. *SIAM J. Sci. Stat. Comput.*, 13:1361–1376, 1992.

[68] S. O. Unverdi and G. Tryggvason. Computations of multi-fluid flows. *Physica D*, 60:70–83, 1992.

[69] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *J. Comput. Phys.*, 100:25–37, 1992.

[70] Z. J. Wang. Vortex shedding and frequency selection in flapping flight. Submitted to the J. Fluid Mech., 1999.

[71] E. Weinan and J. Liu. Vorticity Boundary Condition and Related Issues for Finite Difference Schemes. *J. Comput. Phys.*, 124:368–382, 1996.

[72] A. Wiegmann. *The Explicit Jump Immersed Interface Method and Interface Problems for Differential Equations.* PhD thesis, University of Washington, 1998.

[73] A. Wiegmann and K. P. Bube. The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 35:177–200, 1998.

[74] A. Wiegmann and K. P. Bube. The explicit jump immersed interface method: Finite difference methods for pde with piecewise smooth solutions. To appear in SIAM J. Numer. Anal., 1999.

[75] Z. Yang. *A Cartesian grid method for elliptic boundary value problems in irregular regions.* PhD thesis, University of Washington, 1996.