Journal of Computational Physics **161,** 428–453 (2000)
doi:10.1006/jcph.2000.6502, available online at http://www.idealibrary.com on IDEAL®

The Blob Projection Method for Immersed Boundary Problems

R. Cortez*,[1] and M. Minion†,[2]

*Department of Mathematics, Tulane University, 6823 St. Charles Ave., New Orleans, Louisiana 70118; and †Department of Mathematics, Phillips Hall, CB 3250, University of North Carolina, Chapel Hill, North Carolina 27599
E-mail: *cortez@math.tulane.edu and †minion@amath.unc.edu

Received May 12, 1999; revised March 6, 2000

A new finite difference numerical method for modeling the interaction between flexible elastic membranes and an incompressible fluid in a two-dimensional domain is presented. The method differs from existing methods in the way the forces exerted by the membranes on the fluid are modeled. These are described by a collection of regularized point forces, and the velocity field they induce is computed directly on a regular Cartesian grid via a smoothed dipole potential. Comparisons between this method and the immersed boundary method of Peskin and McQueen are presented. The results show that the method proposed here preserves volumes better and has a higher order of convergence.

Key Words: immersed boundaries; projection method.

1. INTRODUCTION

A high-order numerical method for the solution of two-dimensional immersed boundary problems is presented. In this context, immersed boundaries refer to thin, flexible membranes within a constant density, incompressible fluid. The key feature of these problems is that both the fluid and the immersed boundary motions must be computed simultaneously, accounting for the interaction between the forces developed along the boundaries and the motion of the fluid surrounding them. Existing numerical methods for immersed boundary problems can be placed into two general categories: methods that determine the jump in the variables that are discontinuous across the boundaries (see, e.g., [16]) and methods that regularize the same variables to smooth out the jumps. The immersed boundary method introduced by Peskin and developed further by Peskin and McQueen fits into the second category

[1] Supported in part by NSF Grant DMS-9816951.
[2] Supported in part by the Alfred P. Sloan Foundation and by NSF Grant DMS-9973290.

0021-9991/00 $35.00

[17, 19, 18]. This method has been applied to many physiological and biological problems, including blood flow in the heart, flows in collapsible tubes [21], sperm motility and other flagellar motions [12, 9, 11], platelet aggregation, and others [13].

The immersed boundary method represents moving boundaries with Lagrangian markers at which boundary forces are computed. These forces are then transferred to an underlying Cartesian grid upon which the Navier–Stokes equations are solved. This approach has two advantages: the method is relatively simple to implement even in the presence of complicated boundaries and it allows in principle the use of any grid-based fluid solver. However, the immersed boundary method displays only first-order convergence properties near the boundaries in spite of second-order computational procedures for the forces and the fluid motion. The reason for this is the manner in which the membrane forces are regularized and transferred to the grid. An overview of the immersed boundary method is presented in Section 2 as well as a new version of the method designed to improve its accuracy while holding to the same force regularization methodology. This is accomplished by designing higher-order procedures for representing the boundary forces and for modeling the fluid motion.

In Section 3, a new way of dealing with the boundary–grid communication is offered which leads to a practical high-order numerical scheme. The new method, the blob projection method, is based on ideas taken from vortex and impulse methods [3, 4, 6, 7], in which a cutoff function, or blob, is used to regularize the force field. While the numerical parameters must satisfy certain constraints for stability and accuracy, the support of the blob is decoupled from the other numerical parameters in the method. This important property gives it flexibility and allows one to scale the size of the blob support in a way that balances the various errors in the method.

In Section 4, a careful study of the accuracy and convergence rates for the different approaches is presented. We first examine the convergence properties of the immersed boundary method and verify that the improved version of the immersed boundary method does yield somewhat higher convergence rates. For the improvements achieved, however, the cost associated with this method is high, making it impractical in some situations. We then show that the blob projection method yields even better accuracy and convergence rates than the improved immersed boundary method at a reasonable cost. Finally we identify the step in these methods that reduces the convergence rate below what might be expected from considering only the accuracy of the fluid solver.

2. THE PROBLEM STATEMENT

The incompressible Navier–Stokes equations are

$$\mathbf{u}_t = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \nu\Delta\mathbf{u} + \mathbf{F}, \qquad \nabla\cdot\mathbf{u} = 0. \tag{1}$$

For immersed boundary problems in two dimensions, the force $\mathbf{F}$ arises along the curves that define the immersed boundaries. If $\mathbf{z}(\alpha, t)$ is a parametric representation of the immersed boundary at time t, then one can write

$$\mathbf{F}(\mathbf{x}, t) = \int_0^L \mathbf{f}(\alpha, t)\delta(\mathbf{x} - \mathbf{z}(\alpha, t))\, d\alpha, \tag{2}$$

where the force density $\mathbf{f}(\alpha, t)$ is problem-specific but is typically a function of the local curvature of the membrane and includes tensile and bending components. This force field is singular since it is given by the line integral of a two-dimensional delta function.

We are interested in the dynamics of the boundary which is assumed to move with the fluid velocity according to

$$\frac{d\mathbf{z}(\alpha, t)}{dt} = \mathbf{u}(\mathbf{z}(\alpha, t), t) = \int_{\mathbb{R}^2} \mathbf{u}(\mathbf{x}, t)\delta(\mathbf{x} - \mathbf{z}(\alpha, t))\, d\mathbf{x}. \tag{3}$$

2.1. The Projection Formulation

The numerical method proposed in this paper is based on the projection formulation of the Navier–Stokes equations. The following standard decomposition theorem is needed in order to state the formulation.

THEOREM 2.1. *A vector field* $\mathbf{w}$ *in a bounded domain* $\mathcal{D} \in \mathbb{R}^2$ *can be decomposed uniquely into* $\mathbf{w} = \mathbf{u} + \nabla\xi$, *where* $\mathbf{u}$ *satisfies* $\nabla \cdot \mathbf{u} = 0$ *in* $\mathcal{D}$ *and appropriate boundary conditions.*

The proof of this theorem also indicates the numerical procedure used for implementing projections. One is interested in finding $\mathbf{u}$, which plays the role of the fluid velocity. Taking the divergence of $\mathbf{w} = \mathbf{u} + \nabla\xi$ yields

$$\begin{aligned} \Delta\xi &= \nabla \cdot \mathbf{w}, && \text{in } \mathcal{D} \\ B(\xi, \mathbf{w}) &= 0, && \text{on } \partial\mathcal{D}, \end{aligned} \tag{4}$$

where $B(\xi, \mathbf{w})$ represents the appropriate boundary conditions consistent with the definition of the decomposition. The field $\mathbf{u} = \mathbf{w} - \nabla\xi$ is the projection of $\mathbf{w}$ onto the space of zero-divergence vector fields and is denoted by $\mathbf{u} = \mathbb{P}\mathbf{w}$.

In two dimensions the field $\mathbf{u}$ satisfying $\nabla \cdot \mathbf{u} = 0$ can be written in an equivalent manner using the stream function ψ as $\mathbf{u} = (\psi_y, -\psi_x)$. Using this variable the decomposition is $\mathbf{w} = (\psi_y, -\psi_x) + (\xi_x, \xi_y)$, where ψ is the solution of

$$\begin{aligned} -\Delta\psi &= \nabla \times \mathbf{w}, && \text{in } \mathcal{D} \\ B(\psi, \mathbf{w}) &= 0, && \text{on } \partial\mathcal{D}, \end{aligned} \tag{5}$$

where again $B(\psi, \mathbf{w})$ represents the original boundary conditions for $\mathbf{u}$ written in terms of ψ.

Using projections, one can write the incompressible Navier–Stokes Eq. (1) as

$$\mathbf{u}_t = \mathbb{P}[-(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p + \nu\Delta\mathbf{u} + \mathbf{F}]. \tag{6}$$

When periodicity is imposed as the boundary condition, the pressure term on the right does not contribute to the projection and can be eliminated altogether.

2.2. The Immersed Boundary Method

A brief description of the immersed boundary method is presented in this section. More detailed treatments can be found in [18]. Consider a two-dimensional curve that represents a membrane immersed in the fluid. The membrane is discretized by M points, $\mathbf{z}_k$ for

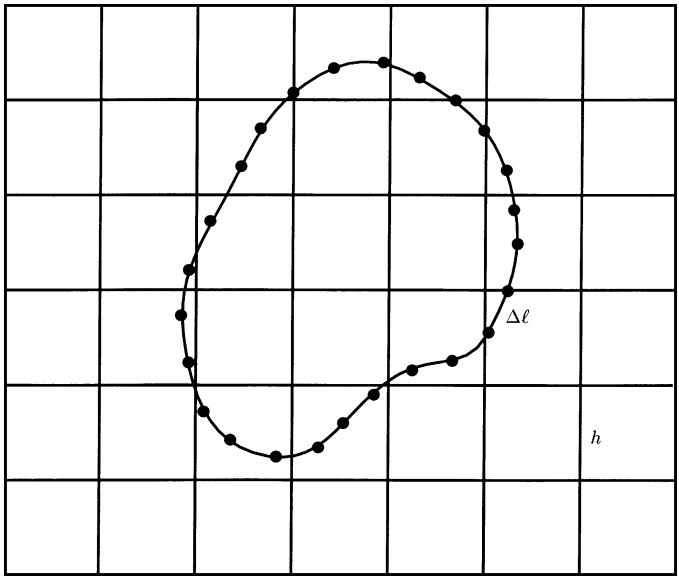

FIG. 1. Immersed boundaries are described by a Lagrangian collection of points embedded in a fluid domain covered by a regular grid. The grid cells are $h \times h$ and the initial point spacing along boundaries is $\Delta\ell$.

$k = 1, \ldots, M$. Let $\Delta\ell$ denote the separation between points along the immersed boundary. Suppose that the fluid occupies the unit square $[0, 1] \times [0, 1]$ covered with a grid of size $h = 1/N$ as in Fig. 1.

Let $\mathbf{u}^n_{i,j}$ represent the velocity at the grid point (i, j) at time step n, and $\tilde{\mathbf{u}}^n_k$ denote the velocity at the kth particle position at time step n. At every time step, given $\mathbf{z}^n_k$, $\mathbf{u}^n_{i,j}$, and $\tilde{\mathbf{u}}^n_k$, the goal is to compute new particle positions $\mathbf{z}^{n+1}_k$ and grid velocities $\mathbf{u}^{n+1}_{i,j}$ and then to interpolate the latter to find $\tilde{\mathbf{u}}^{n+1}_k$ at the new particle locations. For simplicity we describe one step of the time integration based on forward Euler's method, although in practice we use a Runge–Kutta method. The boundary position and grid velocities are updated with

$$\begin{aligned}
\mathbf{z}^{n+1}_k &= \mathbf{z}^n_k + \Delta t \tilde{\mathbf{u}}^n_k \\
\mathbf{u}^{n+1}_{i,j} &= \mathbf{u}^n_{i,j} + \Delta t \mathbb{P}\left[-\left(\mathbf{u}^n_{i,j} \cdot \nabla_h\right)\mathbf{u}^n_{i,j} + \nu \Delta_h \mathbf{u}^n_{i,j} + \mathbf{F}^n_{i,j}\right]
\end{aligned}$$

where ∇_h and Δ_h represent discrete operators acting on grid functions.

The new boundary positions can be computed first from the velocities $\tilde{\mathbf{u}}^n_k$. To update the grid velocities one must evaluate the forces $\mathbf{F}^n_{i,j}$ at each grid point in the domain so that the Navier–Stokes equations can be solved on the grid. This requires first the computation of the force densities $\mathbf{f}(\alpha)$ given the current boundary configuration. A typical feature of immersed boundary problems is that the forces are evaluated at the membrane markers which do not necessarily coincide with grid nodes. To represent the forces at the grid nodes, the immersed boundary method makes use of an approximation of the delta function, D_h, and Eq. (2) to spread the forces to the grid. The result is

$$\mathbf{F}^n_{i,j} = \sum_{k=1}^{M} \mathbf{f}^n_k D_h\left(\mathbf{x}_{i,j} - \mathbf{z}^n_k\right)\Delta\ell, \tag{7}$$

where $\mathbf{x}_{i,j}$ is the grid node (i, j) and $\mathbf{f}_k^n$ is the force density at the boundary point $\mathbf{z}_k^n$. The approximate delta function is the product of one-dimensional discrete delta functions, $D_h(\mathbf{x}) = d_h(x)d_h(y)$. There are various possible choices for $d_h(x)$; a typical one that has been used extensively is

$$d_h(x) = \begin{cases} \frac{1}{4h}\left[1 + \cos\left(\frac{\pi x}{2h}\right)\right], & |x| \le 2h \\ 0, & |x| > 2h. \end{cases} \tag{8}$$

This function is $C^1(\mathbb{R})$, has unit mass, and satisfies $\int_{-\infty}^{\infty} x d_h(x)\, dx = 0$. Note that the support of this function is always an interval of length equal to four grid cells regardless of the grid size. In other words, the support of d_h scales linearly with h.

Once the forces are computed along the immersed boundaries and spread to the grid, one can use any appropriate numerical method to update the grid velocities, since all quantities, including the forces, have now been computed on the grid. Once the boundary position and grid velocities have been updated, the last remaining step is to interpolate the updated velocities from the grid back to the immersed boundary points. This is done by discretizing Eq. (3) using the same approximate delta function

$$\tilde{\mathbf{u}}_k^{n+1} = \sum_{i,j} \mathbf{u}_{i,j}^{n+1} D_h\left(\mathbf{x}_{i,j} - \mathbf{z}_k^{n+1}\right) h^2. \tag{9}$$

A version of the immersed boundary method which uses second-order time integration and finite differencing in space leads to a scheme whose accuracy is formally second order. As will be seen in the numerical results of Section 4 the observed convergence rate of this method is lower than 2. A version which is formally fourth order can also be designed. This is explained next.

2.3. A Formally Fourth-Order Immersed Boundary Method

One might expect that a version of the immersed boundary method which uses fourth-order procedures for the fluid solver, the time integration, the force spreading, and the velocity interpolation would yield higher convergence rates than the method outlined above. To investigate this, a fourth-order approximate delta function is required. Following the design of previous functions one choice is

$$d_h(x) = \begin{cases} \frac{1}{8h}\left[1 + \frac{2}{9}(\pi^2 - 6)\cos\left(\frac{\pi x}{2h}\right) + \frac{1}{9}(2\pi^2 - 3)\cos\left(\frac{\pi x}{4h}\right)\right], & |x| \le 4h \\ 0, & |x| > 4h \end{cases} \tag{10}$$

which is $C^1(\mathbb{R})$, has unit mass, and satisfies $\int_{-\infty}^{\infty} x^p d_h(x)\, dx = 0$ for $p = 1, 2, 3$. Although the support of this function remains a fixed multiple of h in each direction ($8h \times 8h$), the support is wider than the previous function. This form carries a substantial computational cost which is discussed in Section 2.4.

One can also define the order of an approximate delta function based on discrete moment conditions rather than the integrals mentioned above. Examples of second- and fourth-order functions of this type appear in [19, 22], respectively. Appropriately scaled, those functions agree qualitatively with the ones used here. To our knowledge, no significant differences in the results have been reported from the use of those functions as a substitute for the ones defined here.

The discrete delta function described above is coupled with standard fourth-order centered differences for the advective and diffusive derivatives in Eq. (6). The temporal integration for the formally fourth-order immersed boundary method is done with the standard Runge–Kutta method. The discretization for the projection is disussed below.

2.4. The Projection on the Grid

In the immersed boundary method, the discrete projection is implemented via the solution of Eq. (4). This requires a choice of discretization of the Laplacian, divergence, and gradient operators. Since this choice affects the accuracy properties of the method, several different projections are used here for comparison. In each case, the same finite difference approximations used to define the gradient operator, say $\nabla_h = (D_x, D_y)$, are also used to compute the divergence. Moreover, the Laplacian $\Delta_h = D_x^2 + D_y^2$ is used to solve Eq. (4). In order for this procedure to define an exact projection, i.e., one for which the discrete divergence of the projected field is identically zero, the gradient and divergence operators must satisfy an adjoint condition [5]. An approximate projection results if this is not the case.

In this paper the immersed boundary method was implemented with three different types of projections: an exact projection with ∇_h computed using standard centered differences, an approximate projection with ∇_h as suggested by Peskin and Printz [20], and an exact projection with ∇_h defined spectrally as the square root of a compact second-derivative operator [8]. As an example of the latter case, the second-order spectral representation of D_x is

$$\widehat{D_x}(k) = \sqrt{(-2 + 2\cos(2\pi kh))/h^2}.$$

The projection proposed by Peskin and Printz uses a gradient stencil derived from the approximate delta function. The operators D_x and D_y derived from Eq. (8) were developed in [20] and require a 5-by-5 stencil. For the formally fourth-order immersed boundary method a new gradient operator must be derived based on the function in Eq. (10). This new gradient requires a dense 9-by-9 stencil which results in a substantial computational expense since the right-hand side of Eq. (4) and the gradient of ξ in the last step of the projection are computed in physical space at every grid point. For periodic domains, one could alternatively evaluate the divergence and gradient in Fourier space, which would make the cost independent of the stencil size. However, there would be an increased cost associated with performing an additional FFT and inverse FFT.

3. THE BLOB PROJECTION METHOD

The new approach is based on the observation that, depending on the boundary conditions for the velocity $\mathbf{u}$, the projection in Eq. (6) is a linear operator. This is certainly true in the case of periodic boundary conditions so that the linearity of the projection implies

$$\mathbf{u}_t = \mathbb{P}[-(\mathbf{u}\cdot\nabla)\mathbf{u} + \nu\Delta\mathbf{u}] + \mathbb{P}\mathbf{F}. \tag{11}$$

This representation separates the quantities that are evaluated on the grid (the velocities) and those evaluated at the immersed boundary points (the forces). One can use a regularized

form of the forces along the boundary to find an explicit expression that represents the projection of these forces without first transferring the forces to the grid. A recipe for doing this is explained below.

3.1. The Numerical Method

Consider a finite difference numerical method based on Eq. (11) with doubly periodic boundary conditions. Using the notation of previous sections, the equation is then

$$\frac{\mathbf{u}_{i,j}^{n+1}-\mathbf{u}_{i,j}^{n}}{\Delta t}=\mathbb{P}\big[-\big(\mathbf{u}_{i,j}^{n}\cdot\nabla_h\big)\mathbf{u}_{i,j}^{n}+\nu\Delta_h\mathbf{u}_{i,j}^{n}\big]+(\mathbb{P}\mathbf{F}^{n})_{i,j}. \tag{12}$$

All quantities containing $\mathbf{u}^n$ are known on the grid so the first term on the right-hand side can be readily evaluated with any number of existing numerical methods. On the other hand, the forces are given by a discretization of Eq. (2) at the immersed boundary points $\mathbf{z}_k$.

The approach proposed here is motivated by the regularizations used in Lagrangian methods such as vortex and impulse methods. The cutoffs are smooth functions designed to satisfy certain conditions in order to formally give high-order convergence to the method. One advantage of this approach is that the cutoff radius δ is a numerical parameter that is not fixed a priori as a multiple of h but can be chosen separately.

Consider the approximation to Eq. (2) given by

$$\tilde{\mathbf{F}}(\mathbf{x})=\Delta\ell\sum_{k=1}^{M}\mathbf{f}_k\phi_\delta(\mathbf{x}-\mathbf{z}_k), \tag{13}$$

where the cutoff is $\phi_\delta(\mathbf{x})=\delta^{-2}\phi(|\mathbf{x}|/\delta)$ and ϕ is a radially symmetric function satisfying appropriate moment conditions.

The projection of the regularized force field in Eq. (13) evaluated at an arbitrary point $\mathbf{x}$ can be found explicitly to be (see [4, 6])

$$\mathbb{P}\tilde{\mathbf{F}}(\mathbf{x})=\Delta\ell\sum_{k=1}^{M}\frac{1}{2}\,\mathbf{f}_k\phi_\delta(r)+\frac{1}{2\pi}[\mathbf{f}_k-2(\mathbf{f}_k\cdot\hat{\mathbf{x}}_k)\hat{\mathbf{x}}_k]\,\frac{rF'(r)-2F(r)}{2r^2}, \tag{14}$$

where $r=|\mathbf{x}-\mathbf{z}_k|$, $\hat{\mathbf{x}}_k=(\mathbf{x}-\mathbf{z}_k)/r$, and $F(r)=2\pi\int_0^r s\phi_\delta(s)\,ds$. Each term in this sum represents a regularized dipole field which is oriented in the direction $\mathbf{f}_k$. An example of this dipole field is shown in Fig. 2.

In principle, the expression in Eq. (14) can be evaluated at any location $\mathbf{x}$. However, since the method requires the velocity at every grid point, it would be inefficient to use this equation directly. The success of the method depends on the fast evaluation of this projected field, which is explained in the next section. In summary, the algorithm is

1. compute the new forces $\mathbf{f}_k^n$ at the membrane points $\mathbf{z}_k^n$;
2. compute $\mathbf{u}_1=\Delta t\mathbb{P}[-(\mathbf{u}_{i,j}^n\cdot\nabla_h)\mathbf{u}_{i,j}^n+\nu\Delta_h\mathbf{u}_{i,j}^n]$ using finite differences;
3. evaluate $\mathbf{u}_2=\Delta t(\mathbb{P}\tilde{\mathbf{F}})_{i,j}$ at the grid points (see next section);
4. update the grid velocity with $\mathbf{u}^{n+1}=\mathbf{u}^n+\mathbf{u}_1+\mathbf{u}_2$;
5. update the position of boundary points by $\mathbf{z}_k^{n+1}=\mathbf{z}_k^n+\Delta t\tilde{\mathbf{u}}_k^n$; and
6. update the velocity at the membrane points by interpolation $\tilde{\mathbf{u}}_k^{n+1}=I(\mathbf{u}_{i,j}^{n+1})$.

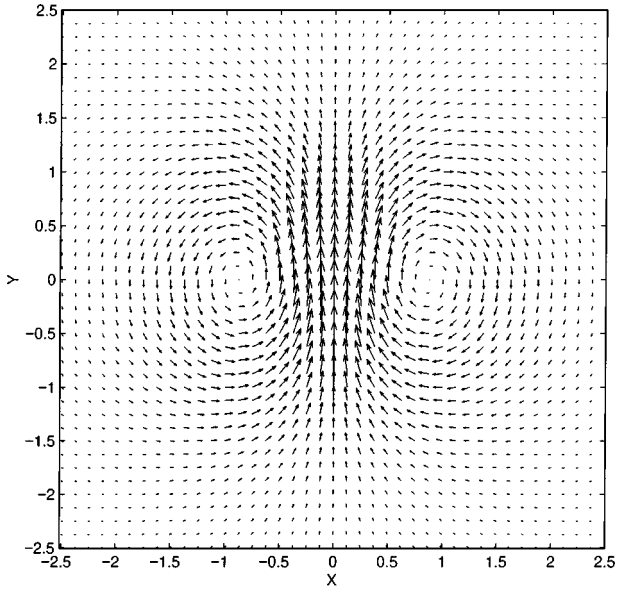

FIG. 2. Velocity field due to a single force blob of strength $(0, 1)$ located at the origin. The value $\delta = 1$ was used for this illustration.

For step 2, centered finite difference operators are used to calculate the advective and diffusive derivative terms in Eq. (12). Second- or fourth-order versions of these operators can be made for this method. While the description above was made using Euler's method for the temporal integration, in practice a standard Runge–Kutta algorithm is used. A formally second-order method uses second-order finite differencing and Runge–Kutta. A formally fourth-order method uses fourth-order versions.

For the problems presented here, the stiffness of the boundary forces is the dominant restriction on the time step; hence the restriction on the time step imposed by the explicit differencing of advective and diffusive terms is not an issue. The implementation of integrators for problems where the stiffness in the boundary makes the time step unreasonably small (see, e.g., [10, 15]) is a topic of future work.

The interpolation procedure in step 6 is not tied to the spreading operator, as is the case in the immersed boundary method. Here, a 4×4 patch of grid surrounding each immersed boundary point is used to compute its velocity with a bi-cubic polynomial interpolant. The resulting approximation is fourth-order accurate.

3.2. Evaluation of $\mathbb{P}\tilde{\mathbf{F}}$ at Grid Points

The third step in the algorithm requires some explanation. The projected field in Eq. (14) does not have compact support; instead, it decays like $1/r^2$ for large r. A direct evaluation of this equation at all grid points would require $O(MN^2)$ amount of work, where M is the number of points defining the immersed boundary and N is the number of grid points in each spatial direction. Additional work would be required to account for the periodicity of the problem. This would be prohibitively expensive for even a moderate value of M; hence a fast procedure, which is similar in spirit to Anderson's method of local corrections [1], is used to effect the evaluation.

The fast evaluation procedure utilizes the curl of $\tilde{\mathbf{F}}(\mathbf{x})$ which is equal to the curl of $\mathbb{P}\tilde{\mathbf{F}}(\mathbf{x})$. Since the latter field is a regularized vortex dipole, the curl is concentrated in a region near the immersed boundary points $\mathbf{z}_k$. The cutoff function in Eq. (13) controls the size of this region and the rate at which the curl decays. In two dimensions Eq. (5) can be solved to find the stream function representing the projection of the boundary forces $\mathbb{P}\tilde{\mathbf{F}}(\mathbf{x})$. The right-hand side is formed by evaluating and summing the curl of all contributions due to the $\mathbf{f}_k$'s on a small patch of grid centered at $\mathbf{z}_k$ and setting it to zero outside this patch. The patch of grid on which the curl is evaluated must be large enough to include the support of the cutoff function. In the case of cutoffs with infinite support, the neighborhood of $\mathbf{z}_k$ must be sufficiently large so that any discontinuities in the projection field at the edge of the grid patch are comparable in magnitude to other error terms. For the cutoff functions used here, a patch of grid of size 4δ is sufficient.

In practice it is not necessary to solve a separate Poisson problem for the stream function representing $\mathbb{P}\tilde{\mathbf{F}}(\mathbf{x})$. By linearity, a single Poisson problem for both the projections in Eq. (12) can be solved simultaneously. The curl of $\tilde{\mathbf{F}}(\mathbf{x})$ (which is given analytically) is simply added to the curl of the sum of the advective and diffusive terms (which is calculated by finite differences) to form the right-hand side of Eq. (5). Standard centered differences are used for the discrete derivatives, resulting in a method readily adaptable to more complicated grid geometries.

In three dimensions the stream function is no longer a scalar; hence this procedure would require the solution of three Poisson problems rather than one. However, a similar procedure based on the solution of Eq. (4) rather than Eq. (5) could also be used. This would require evaluating both the force and the divergence of the force on grid patches near the boundary, but only one Poisson problem would need to be solved in either two or three dimensions. In two dimensions, the method based on Eq. (5) is more economical since only the curl of the force is evaluated on the grid (but not the force itself).

4. NUMERICAL RESULTS

The numerical results presented in this section are for both the immersed boundary method and the new blob projection method proposed in this paper. First, an analysis of the conservation of volume for the methods will be presented. Two convergence studies are also presented. The first one is used to test and compare the convergence properties of the methods, and the second to isolate the cause of the differences in observed convergence rates.

4.1. A Perturbed Ellipse

The initial conditions for the first example consist of a perturbed elliptical membrane immersed in a fluid at rest in the unit square. The curve can be described in polar coordinates by

$$\mathbf{z}(\theta) = (r(\theta)\cos(\theta - \theta_0), r(\theta)\sin(\theta - \theta_0)),$$

where

$$r^2(\theta) = a^2\cos^2(\theta) + b^2\sin^2(\theta) + \epsilon\left(-\tfrac{4}{3}e^{-3(\theta-\pi)^2} - e^{-5(\theta-\theta_1)^2} + e^{-8(\theta-\theta_2)^2}\right)$$

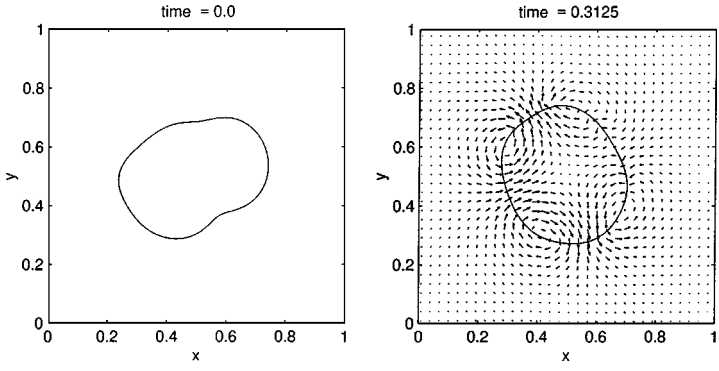

FIG. 3. The initial configuration of the immersed boundary for the perturbed ellipse convergence tests and the solution at time 0.3125.

and $0 \le \theta \le 2\pi$. The parameters used are $a = 0.2$, $b = 0.25$, $\epsilon = 0.012$, $\theta_0 = \pi/3, \theta_1 = 4\pi/5$, and $\theta_2 = 2\pi/3$. The reason for the perturbation is to induce immediate motion of all points of the boundary and to eliminate any symmetries in the problem. The initial particle spacing is approximately equal in arc length, although this will not remain true as the boundary moves. The initial position of the boundary as well as a the solution at a later time are shown in Fig. 3.

The force density for this problem depends on the curvature and is given by

$$\mathbf{f}(\theta) = \sigma \kappa(\theta) \hat{n},$$

where $\hat{n}$ is the outward unit normal of the curve, σ is a fixed stiffness constant, and $\kappa(\theta)$ is the curvature at $\mathbf{z}(\theta)$. In this example the stiffness was set to $\sigma = 1/5$. The curvature is computed in polar coordinates by

$$\kappa(\theta) = \frac{r^2 + 2r'r - (r'')^2}{[r^2 + (r')^2]^{3/2}}.$$

The derivatives of r with respect to θ are approximated by fitting a sixth-order polynomial in θ to seven boundary points and evaluating the derivatives of this polynomial at the middle point.

4.1.1. *Volume Conservation*

The first set of results compares the volume conservation for different versions of the immersed boundary method and the blob projection method. First, results were computed with the immersed boundary method. Second-order projection and time-integration steps were used and the approximate delta function was the one derived from Eq. (8). The method was run using a 128×128 grid and a time step of $\Delta t = 0.1h$. The membrane was discretized using 400 points so that the initial point separation along it was less than the maximum empirical value of $h/2$ (to keep fluid from leaking across the membrane). The fluid viscosity used was $\nu = 0.005$. Figure 4 shows the conservation of the volume inside the membrane using three different projection operators.

The discrete Laplacian operator derived from center differences is a wide 5-point stencil which decouples the grid into four subgrids. This operator has been reported as having

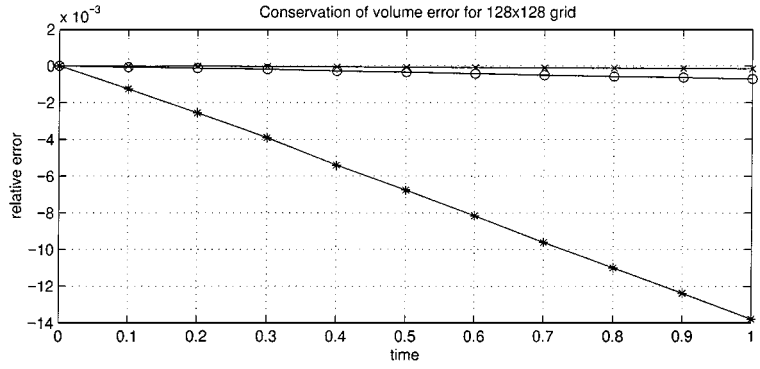

FIG. 4. Volume conservation using the immersed boundary method and three different projection operators with a 128×128 grid and 400 boundary points. The most accurate conservation ($\times$) is achieved with the projection operator in Peskin and Printz. The least accurate conservation ($*$) results from using the decoupled five-point Laplacian operator.

poor volume conservation properties which is confirmed by the steepest curve in the figure. The difference operator for improved volume conservation proposed by Peskin and Printz in [20] is a dense stencil on a 5-by-5 patch of grid and is derived from the approximate delta function. The volume conservation using this operator is represented in the figure by the top curve and gives the best conservation. The figure also shows the results using a second-order version of the spectral operator described in Section 2.4. The application of this operator results in an exact projection with a stencil that does not decouple the grid values of the projected field. It is also unrelated to the approximate delta function used. Although the volume is not preserved as well as the operator derived from the delta function, the improvement over the decoupled operator is significant.

Refining the grid and the membrane discretization reduces the errors and improves the volume conservation. Reducing the numerical parameters by a factor of 2 gives the results in Fig. 5. The volume conservation using any of the three projection operators improves by about a factor of 2, indicating linear dependence on the numerical parameters. This is in spite of using second-order numerical methods in space and time.

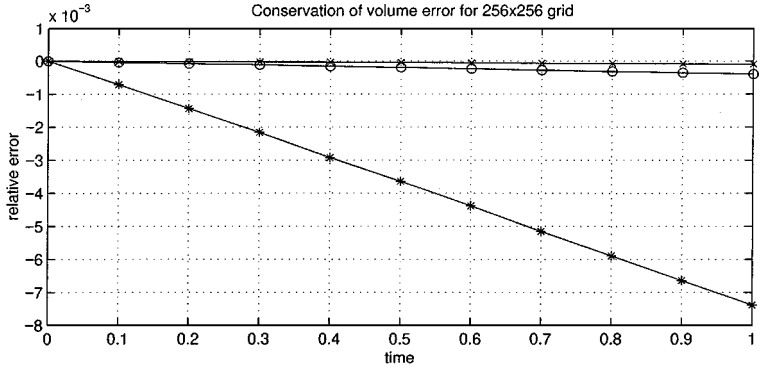

FIG. 5. Volume conservation using the immersed boundary method and three different projection operators with a 256×256 grid and 800 boundary points. The most accurate conservation ($\times$) is achieved with the projection operator in Peskin and Printz. The least accurate conservation ($*$) results from using the decoupled five-point Laplacian operator.

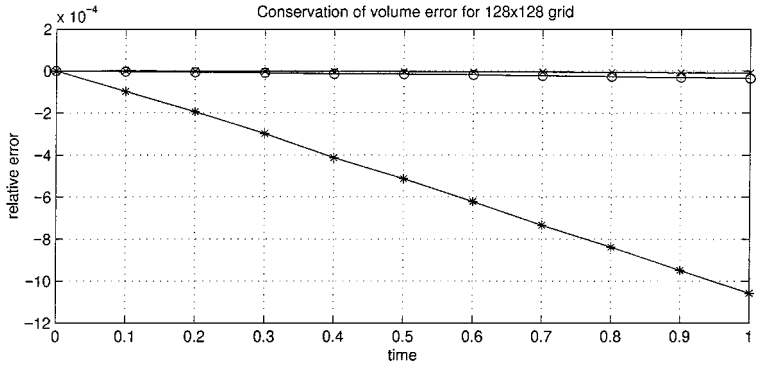

FIG. 6. Volume conservation using the fourth-order immersed boundary method and three different projection operators with a 128×128 grid and 400 boundary points. The most accurate conservation ($\times$) is achieved with the projection operator in Peskin and Printz. The least accurate conservation ($*$) results from using a standard centered projection operator.

Next, the same problem was run using the formally fourth-order method outlined in Section 2.3. The volume conservation results are shown in Fig. 6. The conservation properties have in fact improved by a factor of 10 over the corresponding second-order results of Fig. 4. As in the first case, the worst performance is observed when using a standard centered difference projection despite the fact that the fourth-order stencil for the Laplacian does not decouple.

The same problem was then solved using the blob projection method. Here again, second-order projection and time integration were employed. The regularizations used were based on the fourth-order cutoff

$$\phi_4(r) = \frac{1}{\pi}(2 - r^2)e^{-r^2} \tag{15}$$

and the eighth-order cutoff

$$\phi_8(r) = \frac{1}{6\pi}(24 - 36r^2 + 12r^4 - r^6)e^{-r^2} \tag{16}$$

with $\delta = 4h$. The order of the blob refers to the number of moment conditions it satisfies. The cutoff radius δ is not locked as a fixed multiple of h as in the immersed boundary method so δ can be chosen in different ways. Figure 7 shows a comparison between the best conservation plot obtained with the second-order immersed boundary method and the blob projection method. The results from the blob projection method are at least 10 times better than the immersed boundary result with comparable parameters.

The details of the volume conservation are appreciated better in Fig. 8, which shows the results using a 128×128 grid and 400 boundary points. These results were obtained using the two blobs mentioned earlier and the cutoff parameter set to $\delta = 4h$ and $\delta = 2h$.

While the cutoff functions are formally of high order, the rest of the numerical method being used is second order. In this sense the conservation properties displayed by the method are extremely satisfying. Regardless of the choice of cutoff function, δ must be chosen larger than h in order to regularize the velocities over a region spanning several grid nodes. The results also suggest that there is nothing to be gained in terms of volume conservation by using an eighth-order cutoff versus a fourth-order one.

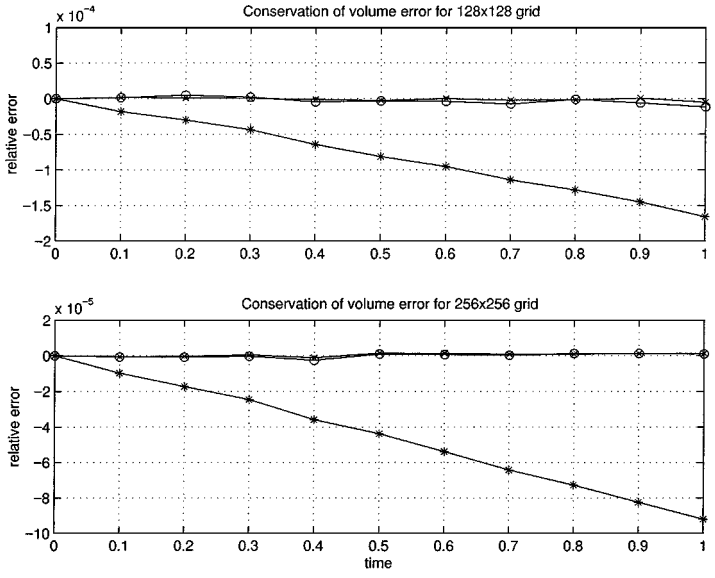

FIG. 7. Volume conservation using the second-order version of the blob projection method and the immersed boundary method. Results for a 128×128 grid with 400 points are shown in the top frame, and those for a 256×256 grid with 800 points in the bottom. The different curves represent the second-order immersed boundary method ($*$), the second-order blob projection method with fourth-order blobs ($\times$), and with eighth-order blobs ($\bigcirc$).

Finally, the blob projection method using fourth-order spatial and temporal accuracy was implemented and used for the same problem. Figure 9 shows a comparison between the immersed boundary and the blob projection methods using fourth-order differencing. For these results the immersed boundary method was run with the improved volume conservation projection which yielded the best results. Looking at the scales along the axes there are two conclusions that one can draw. One is that the blob projection method yields better volume conservation than the immersed boundary method. The second one is that as the grid and boundary are refined, the error in the blob projection method is reduced by a larger factor (about 6) than the error in the immersed boundary method (about 2).

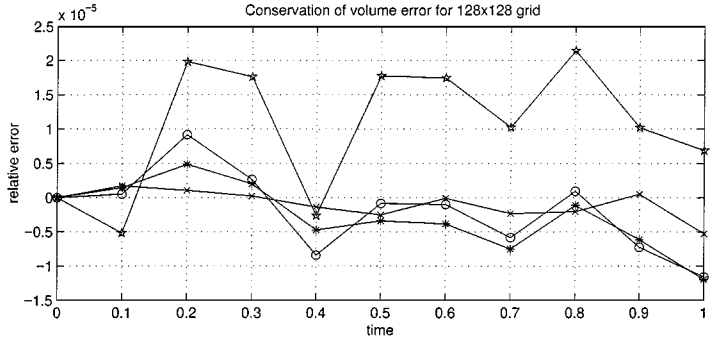

FIG. 8. Volume conservation using the second-order version of the blob projection method on a 128×128 grid with 400 points. The results of four runs are shown. Two of them use a fourth-order cutoff function and cutoff parameter $\delta = 4h$ and $2h$ (denoted by $\times$ and $\bigcirc$). The other two use an eighth-order cutoff function with the same choices of δ (denoted by $*$ and $\star$). These results should be compared to Fig. 4.

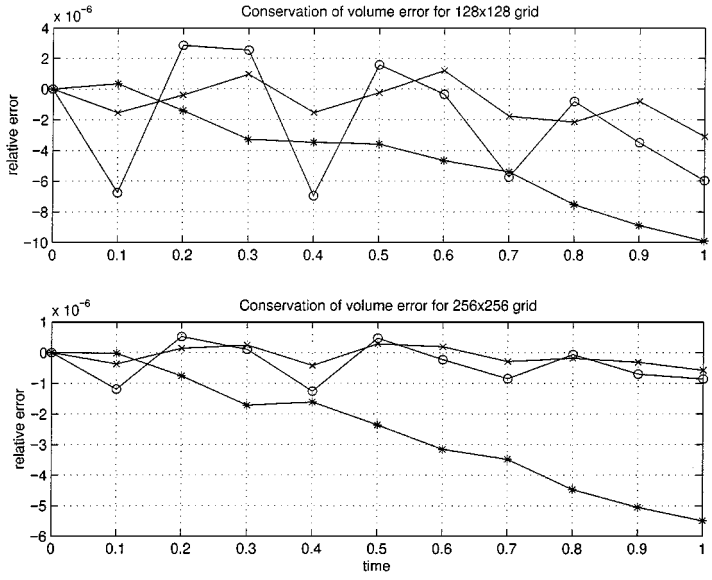

FIG. 9. Volume conservation using the fourth-order version of the blob projection method and the immersed boundary method. Results for a 128×128 grid with 400 points are in the top frame, and those for a 256×256 grid with 800 points in the bottom. The different curves represent the fourth-order immersed boundary method ($*$) and the fourth-order blob projection method with $\delta = 4h$ ($\times$) and with $\delta = 2h$ ($\bigcirc$).

4.1.2. *Convergence Rates*

Next we explore the convergence rates of the immersed boundary and blob projection methods on the perturbed ellipse problem. Convergence rates for immersed boundary problems can be calculated in different ways with varying results. Several procedures are employed here to illustrate this fact.

Estimating convergence rates for immersed boundary problems is more complicated than for standard problems for incompressible flow. In the blob projection method, there are three separate numerical parameters which can be varied in a convergence test: the grid size h, the initial particle separation along the boundary $\Delta\ell$, and the cutoff radius δ. The first two are generally of the same size or with $\Delta\ell$ somewhat smaller than h, while δ must be larger than $\Delta\ell$ but is free to vary. In the immersed boundary method, the analogs of these three parameters are necessarily scaled linearly. Therefore, convergence rates will first be presented for this scaling. Later, convergence tests for the blob projection method where the three numerical parameters are not reduced together will be presented in order to examine different parts of the numerical error.

Several quantities can be examined to estimate convergence rates. First, the position of and velocity at the immersed boundary points can be measured. Convergence rates can be calculated for each individual point yielding a vector of convergence rates of which a norm can be taken. This will be referred to as the *norm of the rates*. Alternatively, the entire boundary can be considered a vector of positions or velocities and a convergence rate can be calculated by computing the norm of errors of this vector. This will be referred to as the *rate of the norm*. Lastly, the velocity on the grid can be used to calculate convergence rates as well. Since the velocities at immersed boundary points are already being considered, convergence rates for the velocity on the grid will only be computed on patches of grid well separated from the immersed boundary.

The convergence rates themselves are computed in two different ways. Richardson extrapolation using three numerical runs of different resolution is one way to estimate rates. Here grids of size $N \times N$ with $N = 256$, 384, and 576 are used with corresponding boundaries of 600, 900, and 1350 points. For example, if U_N represents the velocity of the $N \times N$ run, then one estimate of the convergence rate would be

$$\text{rate} = \frac{\ln(\|U_{256} - U_{384}\|_p / \|U_{384} - U_{576}\|_p)}{\ln(3/2)}.$$

A different way to estimate the error in a given calculation is by comparing the result to a highly resolved solution. For this, a solution was computed on a 1536×1536 grid using 4800 boundary points and a time step of $0.025h$. We will call this solution the reference solution and the reported error in a given calculation is the deviation from the reference solution. Two coarser runs using grids of size $N = 256$ and 512 with 800 and 1600 boundary points respectively were then used to estimate the convergence rate.

For instance, let h be the grid size of the reference solution, i.e., $h = 1/1536$. Then assuming that the error is of the form Ch^p, one can compute

$$\varepsilon = \frac{C(6h)^p - Ch^p}{C(3h)^p - Ch^p} = \frac{6^p - 1}{3^p - 1},$$

from which p can be estimated.

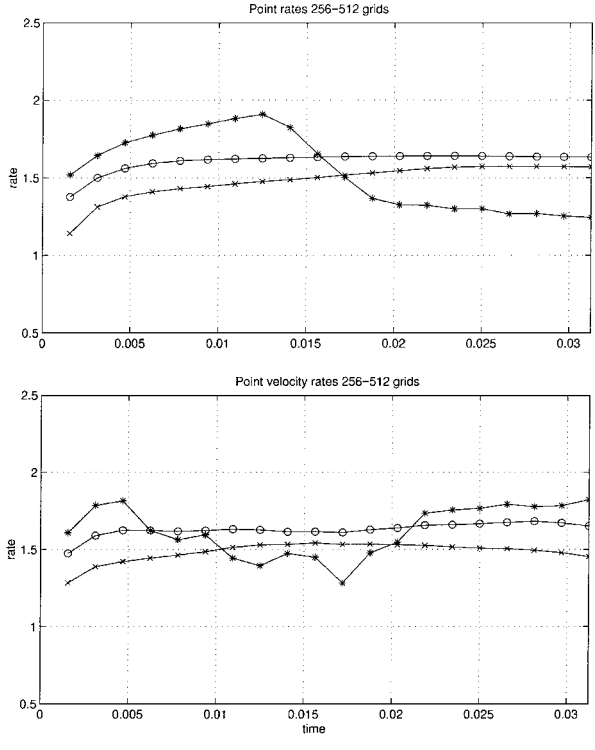

FIG. 10. Convergence rates for the immersed boundary method computed by comparison with a reference solution. The top graph shows the rate of the L_1 ($\times$), L_2 ($\bigcirc$), and L_∞ ($*$) norms of the boundary particle positions. The bottom graph shows the same for the boundary particle velocities.

The first set of results refers to the solution of the perturbed ellipse problem using the (formally second-order) immersed boundary method. The solutions were computed to time $1/32$ using a time step $\Delta t = 0.1h$ and compared to a reference solution at time intervals of $1/640$. Figure 10 shows the computed convergence rates of the velocities using $h = 1/256$ and $h = 1/512$ and reducing $\Delta\ell$ linearly with h. The top plot shows the rate of the norms for the particle positions using the one, two, and infinity norms. The bottom plot shows the rate of the norms for the velocities at the immersed boundary points. Note that the rates are all approximately 1.5.

Figure 11 shows three additional measures of convergence rates. The top graph shows the L_1 and L_2 norms of the rates of individual particle positions, while the middle graph shows the same for the particle velocities. The discrepancy between the two curves in the middle graph indicates that there is a large variation in the velocity rates computed at each point. The velocity at some particles may display a convergence rate higher than 2 while others a rate less than 1. This is perhaps an expression of noise in the solutions. The bottom plot shows the observed convergence rates on a patch of grid away from the immersed boundary. All norms show agreement.

The immersed boundary method shows convergence rates of order about 1.5 for the immersed boundary variables $\mathbf{u}_\alpha$ and $\mathbf{z}$ despite being formally second order. The reason for this is conjectured to be the treatment of the forces. In particular, spreading the forces to the

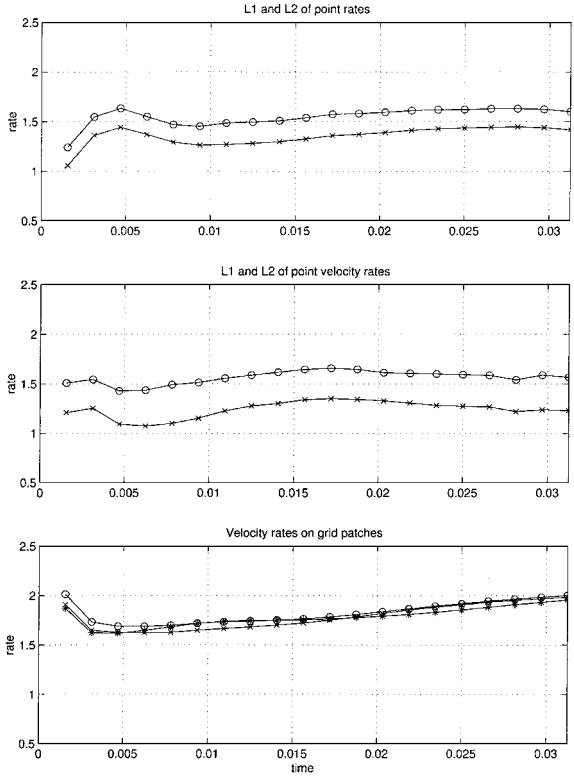

FIG. 11. Convergence rates for the immersed boundary method computed by comparison with a reference solution. The top graph shows the L_1 and L_2 norms of the rates of individual particle positions; the middle graph shows the same for the particle velocities. The bottom graph shows convergence rates on a patch of grid away from the immersed boundary.

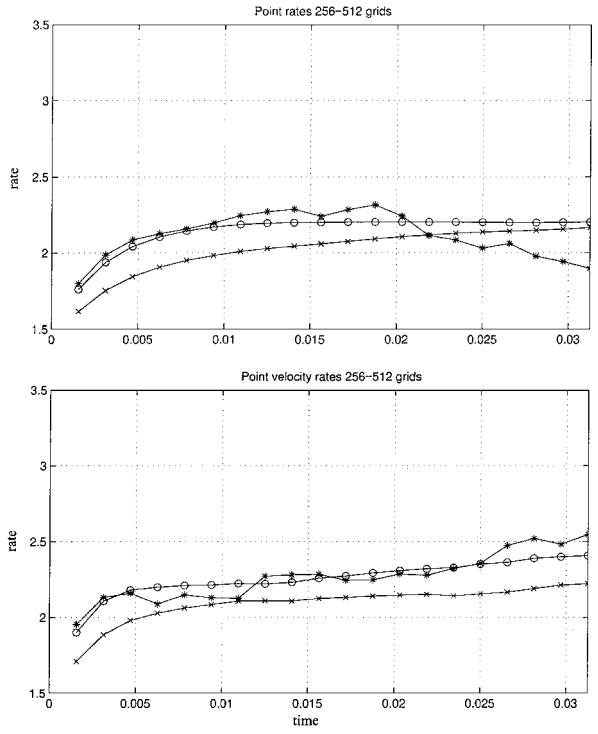

FIG. 12. Convergence rates for the fourth-order immersed boundary method computed by comparison to a reference solution. The top graph shows the rate of the L_1, L_2, and L_∞ norms of the boundary particle positions. The bottom graph shows the same for the boundary particle velocities.

grid with an approximate delta function with small support ($4h \times 4h$) and then performing a projection appear not to resolve the flow accurately enough near the immersed boundaries. The flow away from the boundaries is also affected, showing a convergence rate between 1.5 and 2.

The same experiment was then performed with the version of the immersed boundary method which uses fourth-order differencing and the improved volume conservation operators derived for the new approximate delta function in Eq. (10). Figures 12 and 13 are the analogues of Figs. 10 and 11 for this method. In all cases the convergence rates have improved to a number between 2 and 2.5 for the boundary variables and to 2.5 for the grid velocity.

The next set of figures shows the same information as in the previous set for the fourth-order blob projection method. Figures 14 and 15 should be compared with Figs. 12 and 13. The figures show that the new method is converging at a rate between 2.5 and 3. In these examples, the size of δ was set to $4h$ and hence decreased at the same rate as h. Results using $\delta = 2h$ are very similar and therefore not shown.

Figures 16 and 17 show the convergence rates computed by Richardson extrapolation for the blob projection method. The length of these runs is five times that of the previous runs. Although the convergence rates show increased variability at longer times, the overall size of the rates is roughly the same as previously computed.

Inspection of the norm of the rate plots for the previous two examples shows a much smaller discrepancy between the L_1 and L_2 norms of the individual point rates for the

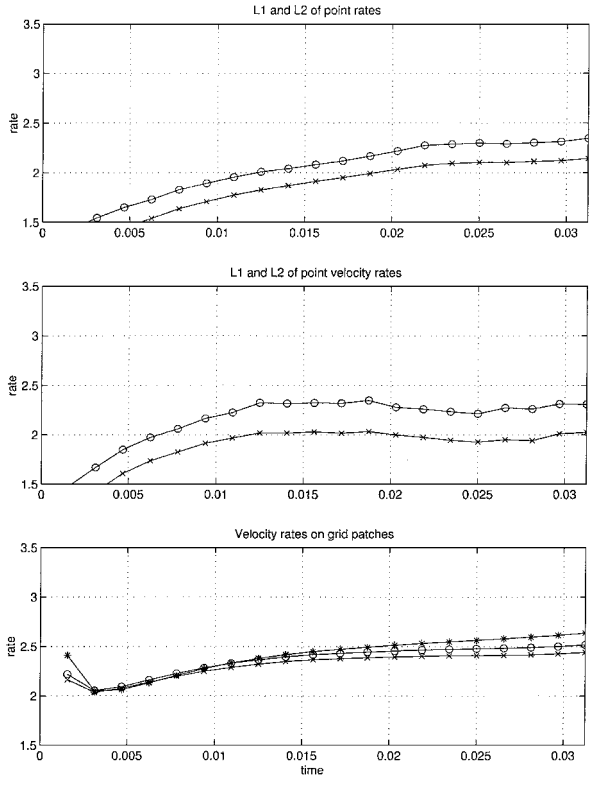

FIG. 13. Convergence rates for the fourth-order immersed boundary method computed by comparison to a reference solution. The top graph shows the L_1 and L_2 norms of the rates of individual particle positions; the middle graph shows the same for the particle velocities. The bottom graph shows convergence rates on a patch of grid away from the immersed boundary.

blob projection method than for the immersed boundary method (see the middle graphs of Figs. 15 and 13). This indicates that the rates at which the individual boundary particle positions converge have less variation in the blob projection scheme. This is seen in Fig. 18 where the convergence rates of individual particle positions from both methods were calculated at time $1/32$ by comparison with the reference solution. Here it is clearly seen that the convergence rates for the blob projection method are much more consistent and less noisy. The largest oscillations are due to cancellation error at points that are nearly stationary at this instant and are not meaningful.

4.1.3. *Convergence Rates for Fixed Regularization*

In the context of Lagrangian methods for Euler flow, such as vortex methods and impulse methods, one defines the cutoff radius as $\delta = Ch^q$, for q in the range $0 \le q \le 1$ (e.g., see [14]). In the present context it is not obvious what the optimal scaling should be since the Lagrangian elements discretize a curve embedded in $\mathbb{R}^2$. One choice of scaling that is easily achieved with the blob projection method is the one associated with $q = 0$. This corresponds to a fixed regularization which may be justified by the physical thickness of the tissue being represented by the immersed boundary in a particular application. As $h, \Delta\ell \to 0$ the convergence rate is expected to reflect the order of the finite difference method used.

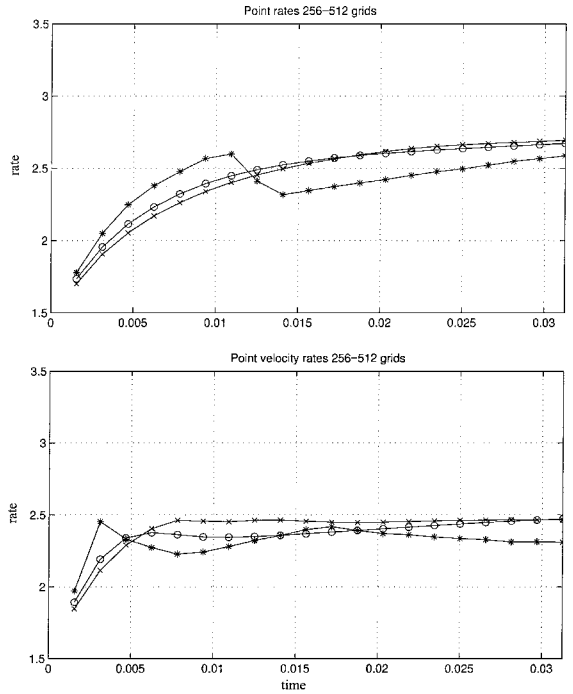

FIG. 14. Convergence rates for the blob projection method computed by comparison with a reference solution. The top graph shows the rate of the L_1, L_2, and L_∞ norms of the boundary particle positions. The bottom graph shows the same for the boundary particle velocities.

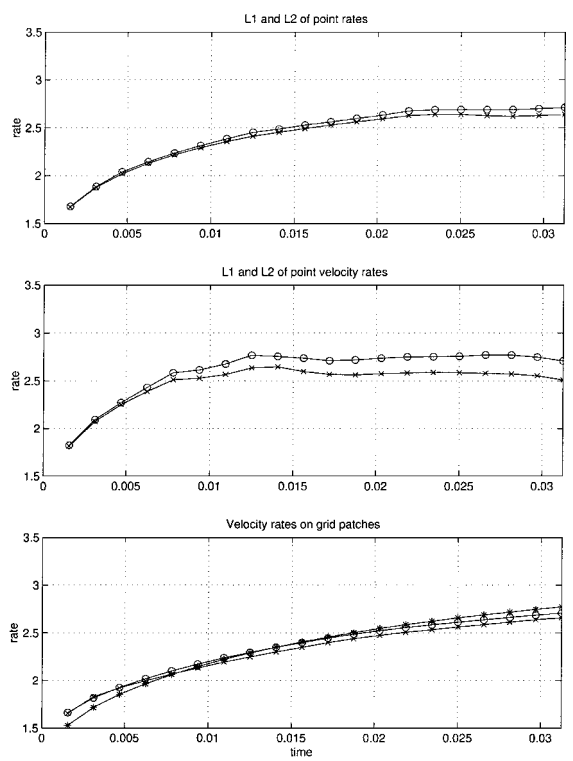

FIG. 15. Convergence rates for the blob projection method computed by comparison with a reference solution. The top graph shows the L_1 and L_2 norms of the rates of individual particle positions; the middle graph shows the same for the particle velocities. The bottom graph shows convergence rates on a patch of grid away from the immersed boundary.

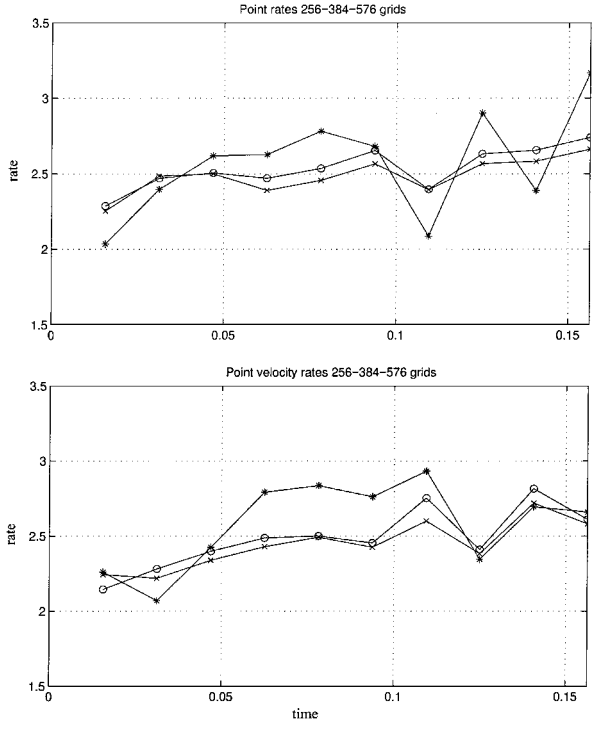

FIG. 16. Convergence rates for the blob projection method computed by Richardson extrapolation. The top graph shows the rate of the L_1, L_2, and L_∞ norms of the boundary particle positions. The bottom graph shows the same for the boundary particle velocities.

A convergence test using the same initial conditions of the perturbed ellipse experiments was run using the fourth-order blob function with δ set to $2/256$ regardless of the grid size. The problem was run on $N \times N$ grids with $N = 256$, 384, and 576 to time $t = 10/64$ using a CFL number of 0.1 and viscosity $\nu = 0.002$. The number of boundary points was again set to 600, 900, and 1350 so that h and $\Delta\ell$ remain proportional. The convergence rates for the position and velocity of membrane points were calculated using Richardson extrapolation at time intervals of $1/64$. Figures 19 and 20 show the results of this experiment. In all measures, the solution of this regularized problem is converging at about a fourth-order rate, the formal accuracy of the fluid solver.

The fact that fourth-order convergence rates are observed when the regularization is fixed, but lower rates are observed when δ is scaled linearly with h and $\Delta\ell$, indicates that there must be a lower-order component of the overall error which depends on δ. This error is identified and discussed in the following section.

4.2. Investigation of Boundary Errors

The convergence rates given in the previous section suggest that there is a source of error in the blob projection method which depends on δ and is responsible for decreasing the observed rates. A numerical experiment designed to illuminate this error will now be presented.

The experiment consists of investigating the error in computing the projection of the boundary forces for a specific boundary configuration. In other words, only the term $\mathbb{P}(\mathbf{F})$

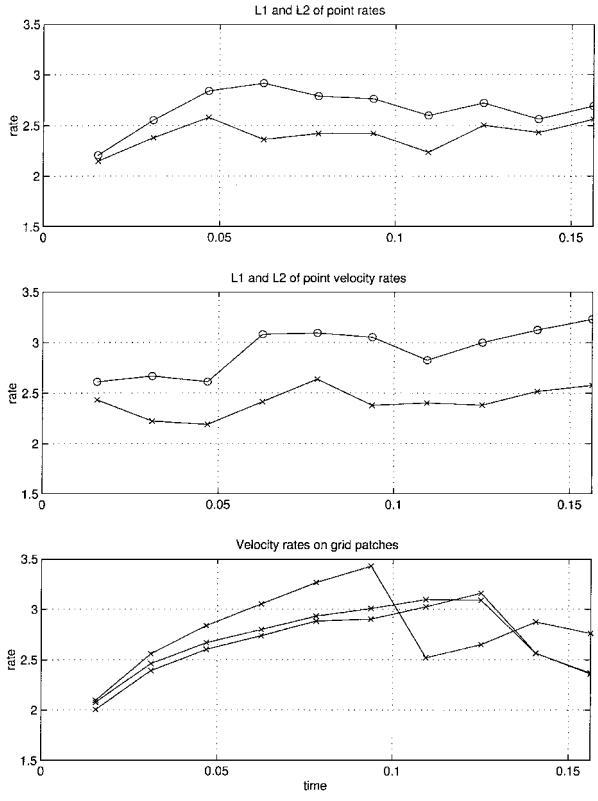

FIG. 17. Convergence rates for the blob projection method computed by Richardson extrapolation. The top graph shows the L_1 and L_2 norms of the rates of individual particle positions; the middle graph shows the same for the particle velocities. The bottom graph shows convergence rates on a patch of grid away from the immersed boundary.

is being computed for a single configuration. No dynamics are included. The membrane shape is an ellipse, for which the curvature-dependent force density can be found exactly. The fluid motion is ignored; hence only the error due to the discretization of the boundary, the projection of the boundary forces, and the interpolation of these forces to the boundary points is relevant.

In order to estimate errors, a reference solution was computed on a 3172×3172 grid using 9600 boundary points and the blob in Eq. (15) with $\delta = 2h$. In the first experiment, the grid size was held fixed at $h = 1/1536$ and the dependence of the error on δ and $\Delta\ell$ was examined. Three sets of runs were done using 400, 800, and 1600 boundary points, respectively. For each boundary discretization, the L_2 norm of the errors at the boundary locations was computed.

The results are shown in Fig. 21 which illustrates several points. For large values of δ the error does not depend on the number of boundary points. This is due to the fact that once the regularization exceeds a threshold, the membrane resolution is enough to reduce the discretization error well below the error due to the regularization, which has now become the leading term. Another observation is that for each boundary configuration, the error has a minimum at a point where δ is about the size of $\Delta\ell$. For smaller values of δ the error increases rapidly. For larger values of δ the error depends approximately linearly on δ. To see this, a line of slope 1 has been superimposed on the graph.

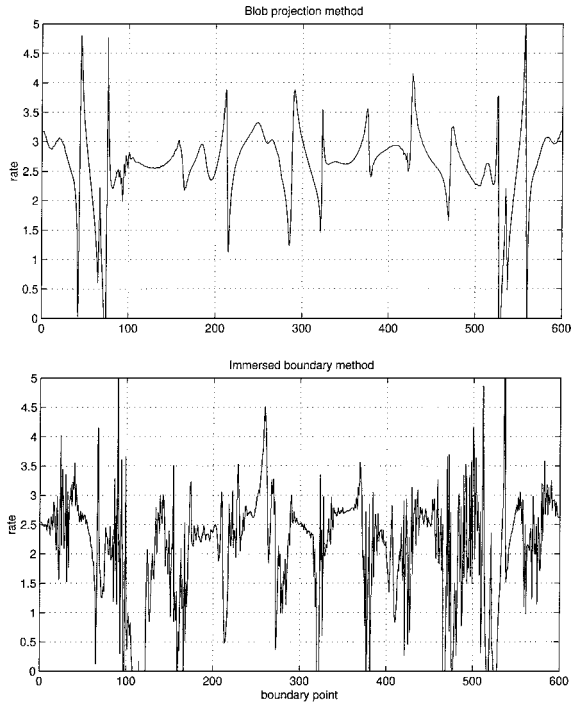

FIG. 18. Individual point convergence rates for the blob projection (top) and formally fourth-order immersed boundary method (bottom) taken at $t = 1/32$.

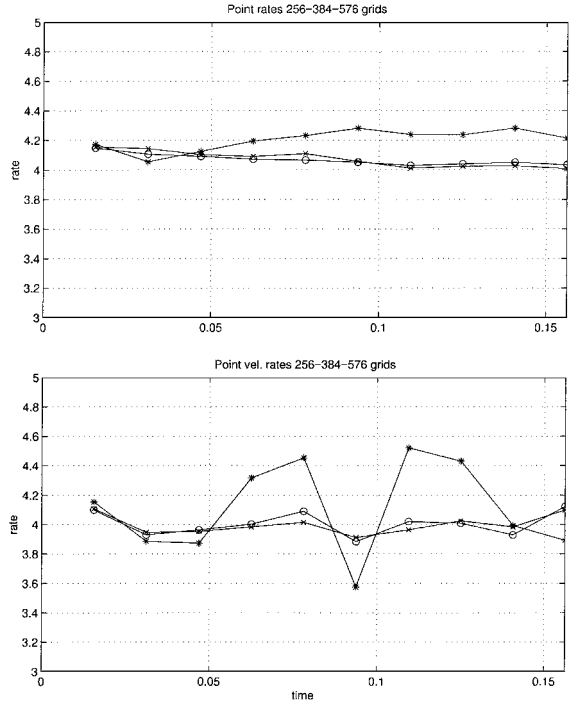

FIG. 19. Convergence rates for the blob projection method with fixed regularization computed by Richardson extrapolation. The top graph shows the rate of the L_1, L_2, and L_∞ norms of the boundary particle positions. The bottom graph shows the same for the boundary particle velocities.

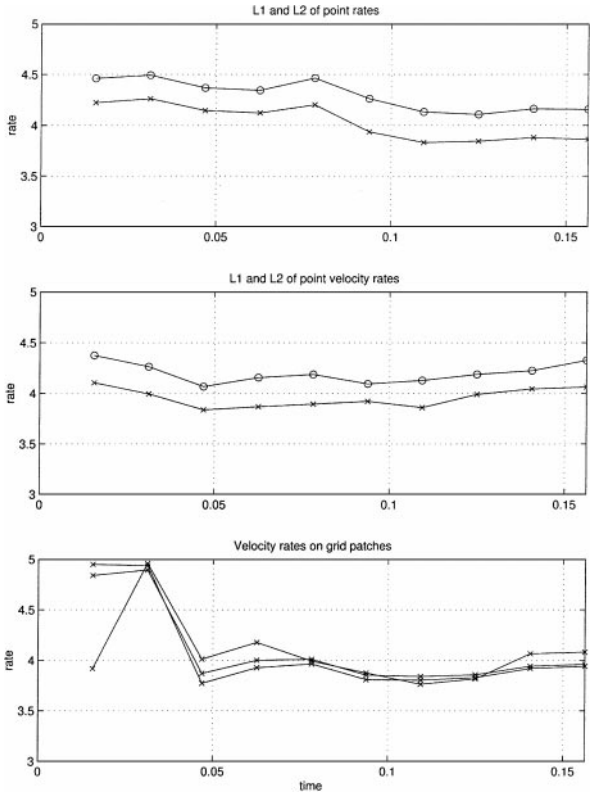

FIG. 20. Convergence rates for the blob projection method with fixed regularization computed by Richardson extrapolation. The top graph shows the L_1 and L_2 norms of the rates of individual particle positions; the middle graph shows the same for the particle velocities. The bottom graph shows convergence rates on a patch of grid away from the immersed boundary.

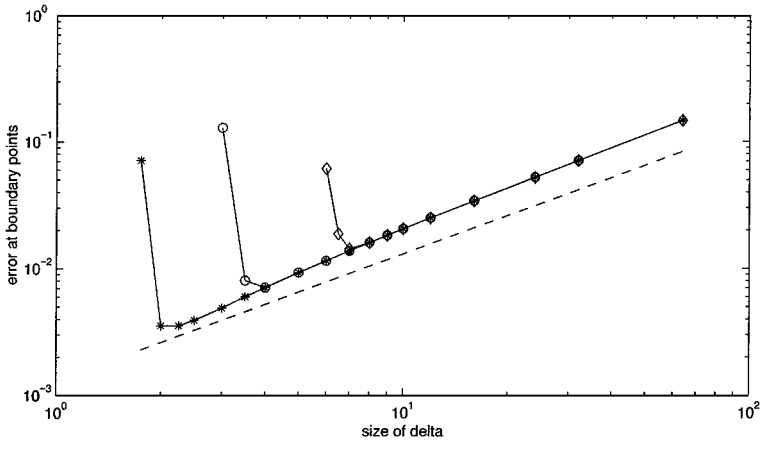

FIG. 21. Errors in computing $\mathbb{P}(\mathbf{F})$ at the immersed boundary for varying δ. Results for 1600 ($*$), 800 ($\bigcirc$), and 400 ($\diamondsuit$) boundary points are shown. The size of delta is in units of $h = 1/1536$.

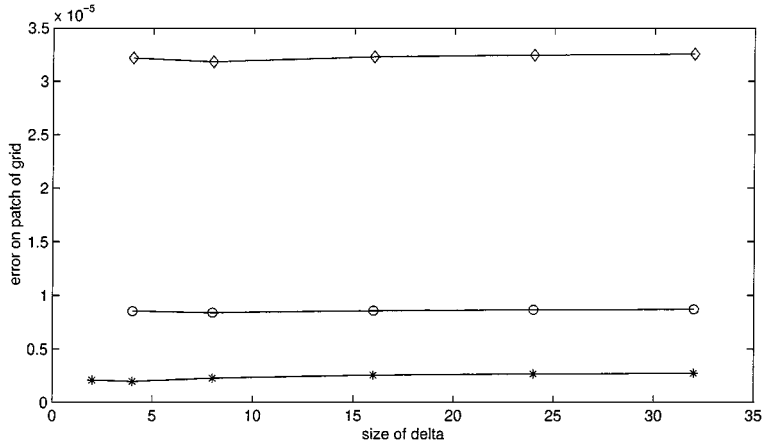

FIG. 22. Errors in computing $\mathbb{P}(\mathbf{F})$ at a patch of grid at the center of the ellipse for varying δ. Results for 1600 ($*$), 800 ($\bigcirc$), and 400 ($\diamondsuit$) boundary points are shown. The size of delta is in units of $h = 1/1536$.

The errors away from the influence of the blobs can also be considered. The errors on a patch of grid located in the center of the ellipse were computed next. The patch covers the square determined by the points $(1/2, 1/2)$ and $(9/16, 9/16)$ and errors are calculated using the L_1 norm. Results are shown in Fig. 22. In this case, for each of the boundary configurations, the error on the grid patch does not depend significantly on the size of δ. This is because the patch of grid is outside of the support of the blob at each point. Note, however, that doubling the number of points on the boundary reduces the error roughly by a factor of 4; i.e., the error away from the immersed boundary depends quadratically on $\Delta\ell$.

Finally, a set of runs was done where h, $\Delta\ell$, and δ are all varied together. Grids of size $N \times N$ were used with $N = 128$, 192, 256, 384, and 512. For each run $\delta = 2h$, and the number of boundary points was also scaled linearly with N from 400 for $N = 128$ to 1600 for $N = 512$. Errors were computed both at the boundary and on the grid patch and are displayed in Fig. 23. The error computed at the grid was multiplied by 1000 for ease of

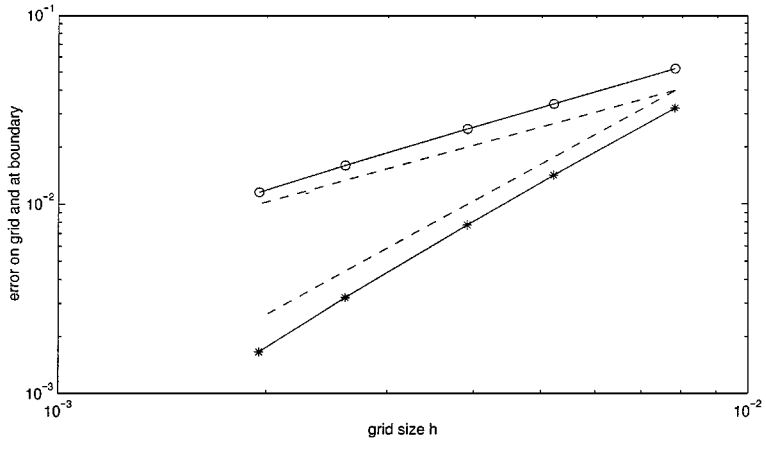

FIG. 23. Errors in computing $\mathbb{P}(\mathbf{F})$ on the boundary and at a patch of grid with all three numerical parameters scaled linearly. The error at the boundary is given by the top line and the error on the grid (multiplied by 1000) by the bottom. Lines of slope 1 and 2 are also superimposed.

comparison and is clearly seen to depend quadratically with h as would be expected from Fig. 22. The error at the boundary decreases linearly as would be expected from Fig. 21.

5. DISCUSSION AND CONCLUSIONS

The last set of results can help explain some features of the convergence properties of the blob projection method. When δ is kept fixed, the method displays fourth-order convergence rates which reflect the order of the finite differencing and the time integration. As δ is decreased there is a source of lower-order error which reduces the observed convergence rates to between 2.5 and 3. This error is concentrated in a small area of the domain near the immersed boundary, the size of which decreases with δ. This is evident in the graphs that show the convergence rates of boundary variables, especially for small values of t. Since at the beginning of all the simulations the fluid is at rest, the error at the first time step is due almost entirely to $\mathbb{P}(\mathbf{F})$. Because this error is of lower order than the other errors in the fluid solver, all of the convergence tests show convergence rates which begin lower than is observed later in the run.

The source of this error (in δ for the blob projection method and in h for the immersed boundary method) is not specific to the numerical methods discussed here. It is due to the nature of the problem and appears as a result of computing a line integral in a two-dimensional domain. The standard moment conditions imposed on the cutoff function used in the blob projection method are unable to improve the computed solution to the degree expected. Recent work by Beale and Lai [2] suggests that more appropriate conditions on the blobs can be used to reduce the errors derived from the regularization of the boundary. In addition, the weights associated with the trapezoid rule in Eq. (14), applied to an integrand with large derivatives, can also be modified to increase the order of this discretization error. The correct procedure for reducing these errors in the context presented here is not known yet and work remains to be done in this direction. However, once these issues are fully understood, the blobs and quadrature corrections would be easily incorporated into the blob projection method.

The results of the numerical experiments indicate that the convergence rates obtained with the blob projection method are higher than those observed previously. In addition the method displays some attractive properties. For instance, the volume conservation is excellent and no special stencils are required to achieve it. The motion of the immersed boundary shows little noise due to the nature of the regularization by the cutoff function. This may be of importance when stresses or other quantities involving derivatives of the flow field are needed at the boundary. The decoupling of the cutoff parameter and the discretization parameters gives the method more flexibility to balance the different error terms. Also, extensions to three dimensions appear to be straightforward.

REFERENCES

1. C. R. Anderson, A method of local corrections for computing the velocity field due to a distribution of vortex blobs, *J. Comput. Phys.* **62**, 111 (1986).
2. J. T. Beale and M.-C. Lai, A method for computing nearly singular integrals, submitted for publication.
3. J. T. Beale and A. Majda, High order accurate vortex methods with explicit velocity kernels, *J. Comput. Phys.* **58**, 188 (1985).

4. T. F. Buttke, Velocity methods: Lagrangian numerical methods which preserve the Hamiltonian structure of incompressible fluid flow, in *Vortex Flows and Related Numerical Methods*, edited by J. T. Beale, G.-H. Cottet, and S. Huberson, NATO ASI Series C, Vol. 395 (Kluwer Academic, Dordrecht/Norwel, MA, 1993), pp. 39–57.
5. A. J. Chorin, On the convergence of discrete approximations to the Navier–Stokes equations, *Math. Comput.* **23**, 341 (1969).
6. R. Cortez, An impulse-based approximation of fluid motion due to boundary forces, *J. Comput. Phys.* **123**, 341 (1996).
7. R. Cortez and D. A. Varela, The dynamics of an elastic membrane using the impulse method, *J. Comput. Phys.* **138**, 224 (1997).
8. W. E and C.-W. Shu, *A Numerical Resolution Study of High Order Essentially Non-oscillatory Schemes Applied to Incompressible Flow*, ICASE Rep. 92-39 (ICASE, 1992).
9. L. J. Fauci, Interaction of oscillating filaments—A computational study, *J. Comput. Phys.* **86**, 294 (1990).
10. L. J. Fauci and A. L. Fogelson, Truncated Newton methods and the modeling of complex immersed elastic structures, *Commun. Pure Appl. Math.* **46**, 787 (1993).
11. L. J. Fauci and A. McDonald, Sperm motility in the presence of boundaries, *Bull. Math. Biol.* **57**, 679 (1995).
12. L. J. Fauci and C. S. Peskin, A computational model of aquatic animal locomotion, *J. Comput. Phys.* **77**, 85 (1988).
13. A. L. Fogelson, A mathematical model and numerical method for studying platelet adhesion and aggregation during blood cloting, *J. Comput. Phys.* **56**, 111 (1984).
14. O. H. Hald, Convergence of vortex methods, in *Vortex Methods and Vortex Motion*, edited by K. E. Gustafson and J. A. Sethian (SIAM, Philadelphia, 1991), pp. 33–58.
15. T. Y. Hou, J. S. Lowengrub, and M. J. Shelley, Removing the stiffness from interfacial flows with surface tension, *J. Comput. Phys.* **114**, 312 (1994).
16. 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 (1994).
17. C. S. Peskin, Numerical analysis of blood flow in the heart, *J. Comput. Phys.* **25**, 220 (1977).
18. C. S. Peskin and D. M. McQueen, A three-dimensional computational method for blood flow in the heart. I. Immersed elastic fibers in a viscous incompressible fluid, *J. Comput. Phys.* **81**, 372 (1989).
19. C. S. Peskin and D. M. McQueen, A general method for the computer simulation of biological systems interacting with fluids, in *Biological Fluid Dynamics*, edited by C. P. Ellington and T. J. Pedley, Symposia of the Society for Experimental Biology, Vol. 19 (Cambridge Univ. Press, Cambridge, UK, 1995), pp. 265–276.
20. C. S. Peskin and B. F. Printz, Improved volume conservation in the computation of flows with immersed elastic boundaries, *J. Comput. Phys.* **105**, 33 (1993).
21. M. E. Rosar, *A Three-Dimensional Computer Model for Fluid Flow through a Collapsible Tube*, Ph.D. thesis (New York University, 1994).
22. J. Stockie, *Analysis and Computation of Immersed Boundaries, with Application to Pulp Fibres*, Ph.D thesis (University of British Columbia, 1997).