A BÉZIER-BASED MOVING MESH FRAMEWORK FOR SIMULATION WITH ELASTIC MEMBRANES*

David E. Cardoze[1] Gary L. Miller[1] Mark Olah[1] Todd Phillips[2]

[1] *Computer Science Department, Carnegie Mellon University*
[2] *Department of Mathematics, Carnegie Mellon University*

ABSTRACT

In this paper we present an application of our Bézier-based approach to moving meshes [1] to Navier-Stokes simulations with several immersed elastic membranes. By a moving mesh we mean one that moves with the material and is adapted to maintain good aspect ratio triangles of minimal size. The adaptations we employ include point insertion and removal, as well as edge smoothing. This work is being done as part of the Sangria project [2] whose goal is to develop geometric and numerical algorithms and software for the simulation of blood flow at the microstructural level. In our approach, we adopt the Lagrangian paradigm where domain boundaries and object interfaces move together with the fluid in which they are immersed. This approach has the advantage that boundaries and object interfaces are easy to track. A moving mesh also poses difficult geometric problems since very distorted elements can be created as the simulation evolves. This can lead to several undesirable or catastrophic situations such as inverted or overlapping elements. From the computational geometry perspective, the challenge presented by the Lagrangian paradigm is the ability to maintain a good quality mesh as the simulation evolves in time. We tackle this problem by using non-linear elements and by locally modifying the mesh using a few primitive operations. The use of non-linear elements allows us to represent the mesh with fewer elements in our simulations, and the use of local operators allows us to avoid remeshing the simulation domain at every time step.

Keywords: Bezier Triangles, B-Splines, Moving Meshes, Immersed Boundary, Navier-Stokes, Blood Flow

1. INTRODUCTION

In this paper we present an application of our Bézier-based approach to moving meshes [1] to Navier-Stokes simulations with several immersed elastic membranes. By a moving mesh we mean one that moves with the material and is adapted to maintain good aspect ratio triangles of minimal size. The adaptations we employ include point insertion and removal, as well as edge smoothing. This work is being done as part of the Sangria project [2] whose goal is to develop geometric and numerical algorithms and software for the simulation of blood flow at the microstructural level. In our approach, we adopt the Lagrangian paradigm where domain boundaries and object interfaces move together with the fluid in which they are immersed. This approach has the advantage that boundaries and object interfaces are easy to track. A moving mesh also poses difficult geometric problems since very distorted elements can be created as the simulation evolves. This can lead to several undesirable or catastrophic situations such as inverted or overlapping elements. From the computational geometry perspective, the challenge presented by the Lagrangian paradigm is the ability to maintain a good quality mesh as the simulation evolves in time. We tackle this problem by using non-linear elements and by locally modifying the mesh using a few primitive operations. The use of non-linear elements allows us to represent the mesh with fewer elements in our simulations, and the use of local operators allows

*SUPPORTED IN PART BY NSF GRANTS CCR–9902091, CCR-9706572, AND ACI-0086093.

us to avoid remeshing the simulation domain at every time step.

One of the main goals of the Sangria project is to gain a better understanding of the behavior of red blood cells in blood flow. Red cells are deformable bodies which consist of fluid contained within a solid membrane. The plasma in which they are immersed is a (mainly) Newtonian fluid. Accordingly, blood flow can be described by means of incompressible Navier-Stokes equations with the addition of elastic forces for cell membranes. In this paper we apply our moving mesh approach to the solution of such systems.

The rest of the paper is organized as follows. In Section 2 we describe the Sangria project and its goals. In Section 3 we describe previous related work. In Section 4 we describe our moving mesh framework in more detail. In Section 5 we describe the mathematical formulation of our physical model. In Section 6 we describe our experimental results, and finally in Section 7 we mention future work.

2. THE SANGRIA PROJECT

The goal of the Sangria project is to develop parallel geometric and numerical algorithms for the simulation of complex flows with dynamic interfaces. Our target application is the simulation of blood flow at the microscopic level where individual cell deformations and their interactions with the surrounding fluid have to be accounted. Due to the complexities involved in the simulation of thousands of individual cells, no simulations of blood flow at this scale exists. However, they are crucial to the development of artificial organs, and better models of macroscopic blood flow.

From a computational perspective the main challenges in microstructural blood flow simulations are the development of numerical algorithms that stably and accurately deal with the interaction of moving and deforming domains and the moving fluids around them. We must maintain the deforming interfaces with a geometric object such as a mesh and must develop efficient geometric algorithms to update the underlying mesh as time evolves. In this paper we concentrate on such geometric algorithms. The target application is to simulate blood flow in three dimensions, here we focus first on two-dimensional simulations, utilizing many procedures that generalize well to higher dimension applications.

3. RELATED WORK

Approaches to the problem of an elastic membrane in a fluid motion have traditionally been one of three types. (as catagorized by Hill in [3]). The first approach is what is known as an Eulerian boundary-capturing method. In this approach, a fixed grid is used to compute the fluid flow, and the location of the mesh boundary is maintained throughout the use of phase field variables (level sets). This approach is described in detail in [3]. The main difficulty with boundary-capturing methods is that very high-order functions must be used for the phase field, and even still the interfaces cannot be sharply resolved. Another problem is that these methods generally have a great deal of difficulty tracking multiple or different types of interfaces.

The second major approach is known as an Eulerian boundary-tracking method. In this method the fluid flow is once again computed on a fixed grid, however, the membranes are tracked using a set of markers or another mesh. The markers are then moved and updated with the flow to track the interface. Central to such methods is the *immersed boundary method* developed by Peskin. This appraoch is discussed in more detail in [4, 5].

Lastly, our type of approach to the problem of an elastic membrane in a fluid motion is the Lagrangian front-tracking approach. In this approach the fluid is solved on a mesh which explicitly contains the boundary. From the meshing perspective, this is the most challenging approach to this problem, since the mesh must be relocated and adapted at every timestep. Antaki et. al. discuss an implementation of this approach in [6]. Their method relied on moving the mesh points but then completely remeshing at every timestep. Ours is the first approach we know of to solving the problem of an elastic membrane in a fluid using a purely Lagrangian moving mesh.

We see many advantages with our method. Most notable is the ability to handle a large number of boundaries or membranes that may be geometrically very close but not touching. These boundaries may be connected to each other in a complicated way, such as membranes forming a honeycomb pattern. Lee and Leveque [5] state that the problem of how to handle different densities and viscosities on each side of a membrane remains open for Eulerian boundary-tracking approaches. Handling different densities and viscosities in our framework is extremely easy. Another major advantage in using moving meshes is the size of the formulation. Far fewer elements are needed to resolve the boundary geometry. In section 6 we present several simulations accomplished with only a few thousand degrees of freedom. Most other simulations attempting to capture similar behavior have relied on hundreds of thousands of variables.

There are several other works in the area of moving meshes, all of which are restricted to moving linear meshes. Kuprat et al. [7] describe a three-dimensional system for moving meshes, X3D. Their system modifies the mesh topology to maintain the Delaunay property as the mesh moves. It also provides mesh smoothing to optimize mesh quality, and mesh refinement by means of point insertions. Other work in moving linear meshes with local modifications is described in [8, 9]. In the case when there is no flow, such as simulating a shock wave, one may coarsen and refine a non-moving mesh. A nice example appears in [10].

4. MESHING FRAMEWORK

In this section we give an overview of our moving mesh framework. For a more complete discussion please refer to [1].

4.1 Element Types and Mesh Hierarchy

Bézier curves and triangles were selected for defining mesh elements for two main reasons. Firstly, using curved instead of linear elements allows us to use meshes with far fewer elements, both for representing geometry and for obtaining accurate numeric solutions. Secondly, Bézier curves and triangles have a number of mathematical properties leading to elegant algorithms.

Bézier curves are completely defined by their control points which form the control polygon. Similarly, Bézier triangles are completely defined by their control points which form a control net. See Figure 1. For more information about Bézier curves and triangles see [11, 12] among others.

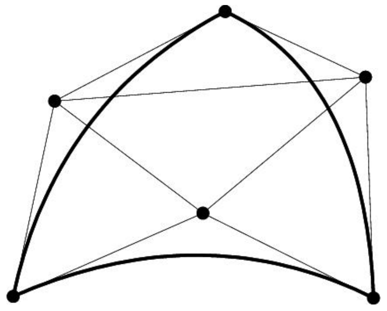

Figure 1: A quadratic Bézier triangle: The boundary of the quadratic triangle is shown in bold. The control net consists of six vertices and four straight triangles.

B-spline curves are a convenient way for us to represent C^1 continuous curves. First, they allow us to represent object boundaries when we want to enforce C^1 continuity. Quadratic B-spline curves are made of a sequence of quadratic Bézier curves connected in such a way that the overall curve is C^1 continuous everywhere. They are completely determined by a control polygon, or de Boor polygon, and a knot sequence. See Figure 2. For more information regarding B-spline curves the reader can refer to [11, 12]. Using B-spline curves to represent mesh interfaces allows us to due computations with membranes that are C^1 continuous. (See Figures 7-11).

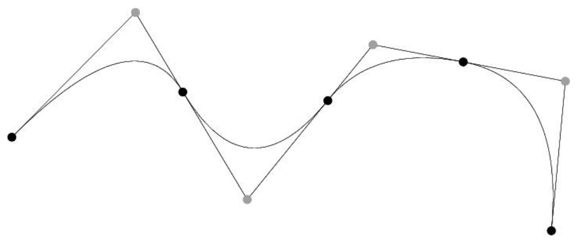

Figure 2: A quadratic B-spline and corresponding control polygon. The black points are the values the curve takes at the knots. The white points are the internal points of the control polygon.

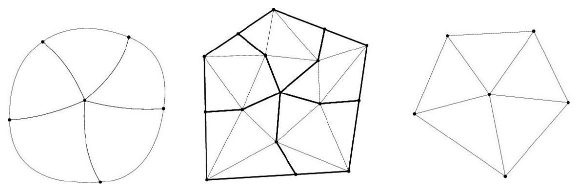

Figure 3: A Bézier mesh (left), its control mesh(center), and logical mesh(right)

4.2 Mesh Adaptation Methods

In our meshing methods we consider three different level of meshes: the Bézier mesh, the control mesh, and the logical mesh. The distinction between these three meshes is very useful in describing and defining mesh improvement methods for curved elements, although all three meshes need not be distinctly represented in the implementation. The curved mesh is the highest level mesh. This is a mesh of the domain, and it is on this mesh that functions are defined. The logical (or linear) mesh is the straight mesh formed by connecting the vertices of the curved mesh and maintaining the same topology. If the curved mesh is not very curved, we expect it will inherit most of the properties of the logical mesh. The control mesh is obtained from the Bézier mesh by replacing every curved triangle by four straight triangles. The vertices of these triangles are the vertices and control points of the curved triangle. By controlling the validity and quality of this control mesh, we can control the validity and quality of our curved elements. Figure 3 shows a Bézier mesh and its logical and control meshes.

As the mesh domain moves and deforms, it becomes necessary to modify the mesh in order to maintain certain quality guarantees. Our algorithms for these tasks are based on linear mesh improvement techniques.

First, we keep the logical mesh Delaunay using well-known incremental Delaunay algorithms based around bistellar edge flips. The edge flip for curved triangles is implemented using edge flips of the control mesh. See Figure 4.

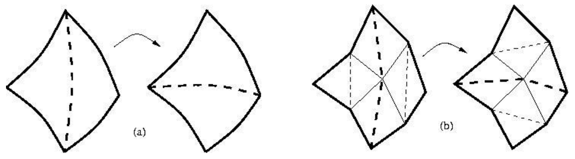

Figure 4: (a) The dashed edge is flipped in the curved mesh. (b) The corresponding dashed edges are flipped in the control mesh.

Second, as the mesh is distorted, elements may be stretched too large to capture the desired properties of the simulation, or elements may develop a poor aspect ratio that cannot be cured by edge flipping. In this case, we refine the mesh by inserting circumcenters as in [13]. Third, as we refine elements as described above, portions of the mesh may end up with many triangles whose sizes are much smaller than actually needed by the simulation. To keep the mesh from becoming too large, we perform mesh coarsening by means of the Douglas-Peucker algorithm [14] for boundaries, and the function-based coarsening paradigm of Talmor et al [15, 16].

Finally, if triangles become very curved, poor numeric solutions may follow. To avoid this situation we reposition the control points of such triangles by means of smoothing techniques. See [1] for details.

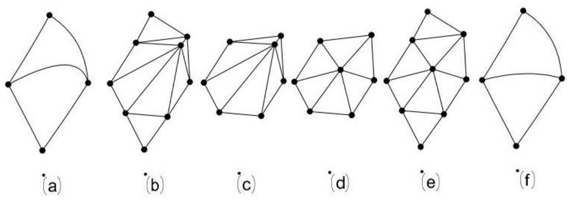

Figure 5: (a) An edge to be smoothed is identified based on its quality. (b) The control mesh (c) Isolate the star of the control point to be move (d) Move control point based on linear mesh improvement techniques (e) Update the control mesh (f) better quality elements

4.3 State-Dependent Functions

Throughout the process of adapting the mesh after it has moved, we must maintain several functions **f** on the mesh domain. The type of function **f** is problem-dependant and user-specified. In our simulations, we use isoparamtric finite elements, so our mesh will carry piecewise quadratic functions. Most important to us is the velocity solution from the previous timestep, since it is necessary for computing the next solution (see Section 5).

Since adapting the mesh changes the domain for **f**, we must project **f** onto the new mesh domain. This is accomplished by making local projections of **f** whenever local mesh modifications are made, and then computing the local reinterpolation error. The choice of a relevant error norm and projection is problem dependant, but the computations are generally easy, since they are done in a local setting with few degrees of freedom. If the calculated re-interpolation error due to a mesh modification is too great, then the modification can be aborted and can be replaced with a mesh refinement.

The most common case of this procedure is when an edge flip would create an undesirable amount of reinterpolation error, and accordingly, the edge is split instead. Strict refinement techniques (point insertions, edge splits), will not introduce any re-interpolation error. Using these ideas, the mesh adaptation methods can be fairly liberal with respect to altering the mesh, and can default to the more conservative refinement schemes as necessary to control error.

In Figure 6, we show an example of one possible reinterpolation. Piecewise quadratic functions are being carried on the mesh. When the mesh domain is changed due to an edge flip, a new piecewise quadratic function is chosen that will minimize the L^2 error between the old function and the new function.

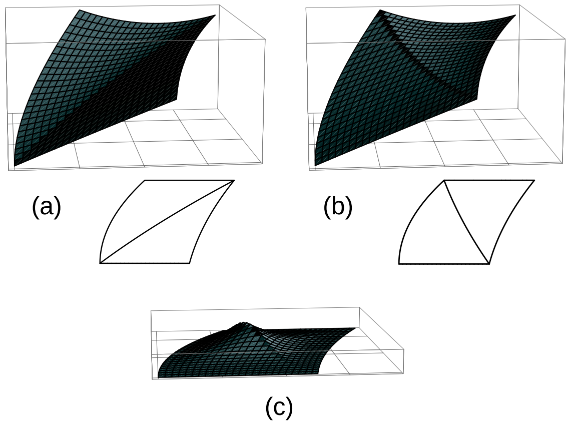

Figure 6: (a) An initial quadratic mesh domain and a function carried on the mesh (b) The mesh domain is changed by an edge flip, the old function is projected to a new function (c) Plot of the reinterpolation error due to the edge flip.

5. PROBLEM FORMULATION

5.1 Lagrangian Formulation of Incompressible Navier-Stokes

We begin with the basic expression of the incompressible Navier-Stokes equations in a Lagrangian framework. Consider a body Ω with a smooth boundary Γ. We denote velocity as $\mathbf{u}$, pressure as p, viscosity and density as μ and ρ. We wish to find a solution (with appropriate boundary conditions) to the system:

$$\rho\dot{\mathbf{u}} + \mu\Delta\mathbf{u} - \nabla p = \mathbf{f} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

where $\dot{\mathbf{u}}$ is the material time derivative of velocity and Δ is the Laplacian operator. We use a Backward-Euler scheme for the time derivative, to arrive at:

$$\rho\mathbf{u} + \tau(\mu\Delta\mathbf{u} - \nabla p) = \tau\mathbf{f} + \rho\mathbf{u}_0 \tag{3}$$

Where $\mathbf{u}_0$ is the velocity solution from the previous timestep and τ is the size of a timestep. The moving mesh framework is what makes $\mathbf{u}_0$ easily computable in the material configuration.

5.2 Membrane Forces

In our simulations we have chosen to model the tension due to the membrane as a body force lagged in time.

This simplifies the numerical simulation by absorbing the membrane forces into $\mathbf{f}$, so that the elasticity only affects the right hand side of the equation. This has shown to be a somewhat stable approximation in practice. If we denote the force due to elasticity as $\mathbf{f}_m$, this effectively makes our system:

$$\rho\mathbf{u} + \tau(\mu\Delta\mathbf{u} - \nabla p) = \tau\mathbf{f}_{m_0} + \rho\mathbf{u}_0 \tag{4}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{5}$$

So that $\mathbf{f}_{m_0}$ is the membrane force computed on the previous iteration. Again, because of the moving mesh, this is easily computable in the material configuration.

Since we have chosen to model the force due to the membrane as a body force, we must find some way to capture the boundary layer that the membrane forces act upon. We approximate the membrane force due to a membrane as follows:

$$\mathbf{f}_m(\mathbf{x}) = \int_\Gamma \mathbf{f}_m(\boldsymbol{\chi}(\hat{s}))\delta(\mathbf{x} - \boldsymbol{\chi}(\hat{s}))\ d\hat{s} \tag{6}$$

Where δ is the discrete delta function and the membrane Γ is arclength parametrized by $\boldsymbol{\chi}(\hat{s})$. This approach is the immersed boundary method and is due to Peskin [4]. When we use this approximation with a moving mesh that is tracking the membrane Γ, the effect is that the spatial approximation given by the mesh spreads the boundary layer over all the elements adjacent to the membrane. This means that as the mesh is refined near the membrane, the approximation to the membrane force is improved.

Traditional Eulerian boundary tracking methods approximate the membrane forces on all the intersected elements of a fixed grid, and thus robust Eulerian methods must continually adapt the mesh for resolution near the moving membrane. By using a moving mesh framework, once we have established the needed mesh resolution near the membrane, the created mesh elements move with the membrane over time and mesh resolution near the membrane is preserved.

5.3 Calculating Elastic Membrane Forces

The remaining problem is the calculation of $\mathbf{f}_m(\boldsymbol{\chi}(\hat{s}))$, which we will simply write as $\mathbf{f}_m(\hat{s})$. Following [5, 4], given any parameterization $\boldsymbol{\chi}$ of the membrane Γ, the elastic tension at s can be written as:

$$t(s) = k_e(|\boldsymbol{\chi}'(s)| - 1) \tag{7}$$

$$\mathbf{f}_m(s) = k_e\frac{d}{ds}\left(t(s)\boldsymbol{\tau}(s)\right) \tag{8}$$

The elastic constant of the membrane is given by k_e and the unit tangent vector of Γ is given by $\boldsymbol{\tau}(s)$. This computation is extremely straightforward when using moving meshes. An initial arclength parametrization of the membrane Γ is given as $\hat{\chi}(\hat{s})$. All the edges and vertices in the mesh corresponding to Γ are endowed with pointers to the membrane Γ and with markers $\hat{s}_0$ (and $\hat{s}_1$ for edges). When ever the mesh is refined or coarsened these boundary markers are appropriately maintained. Then a simple change of variables allows us to calculate (and subsequently integrate) $\mathbf{f}_m$ on any edge of the mesh.

6. EXPERIMENTS

In this section we present the results of three different experiments that exercise our framework.

6.1 Stretched Membrane

In this experiment we have two fluids both with viscosity of 0.0125 inside a square whose sides are of length 1. Initially, one of the fluids occupies a circular region of radius 0.25 and is separated from the other fluid by a membrane. From time $t = 0.0$ to time $t = 0.2$ upward and downward forces are applied to the upper and lower parts of the membrane respectively. After time $t = 0.2$ these forces are removed. We present two different setups for this experiment: one where the membrane elasticity constant is 1.0 and another where it is 10.0. See Figure 7 and Figure 8.

Area preservation is one common gauge of the quality of numeric simulations that attempt to track membranes. Our simulations maintain the same quality with respect to this metric as many other simulations using vastly more elements [5].

It can be seen in these figures that as the tension in the membrane increases as the fluid inside the membrane experiences less deformation. The fact that the fluids do not go back to their initial configurations can be attributed to the effect of viscosity.

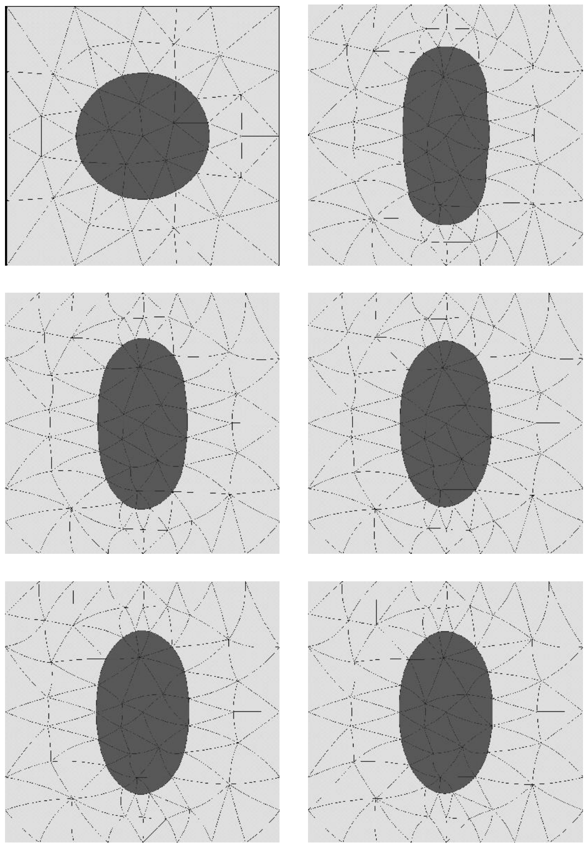

Figure 7: Stretched membrane with elasticity constant of 1.0. From left to rigth and top to bottom, the frames correspond to times $t = 0$, $t = 0.12$, $t = 0.24$, $t = 0.36$, $t = 0.48$ and $t = 0.60$ respectively.

6.2 Falling Membrane

In this experiment we have the same two fluids as before. Again they both have a viscosity of 0.0125 and are separated by a membrane. In this case however a constant downward body force (gravity) is applied over time to the region enclosed by the membrane. We

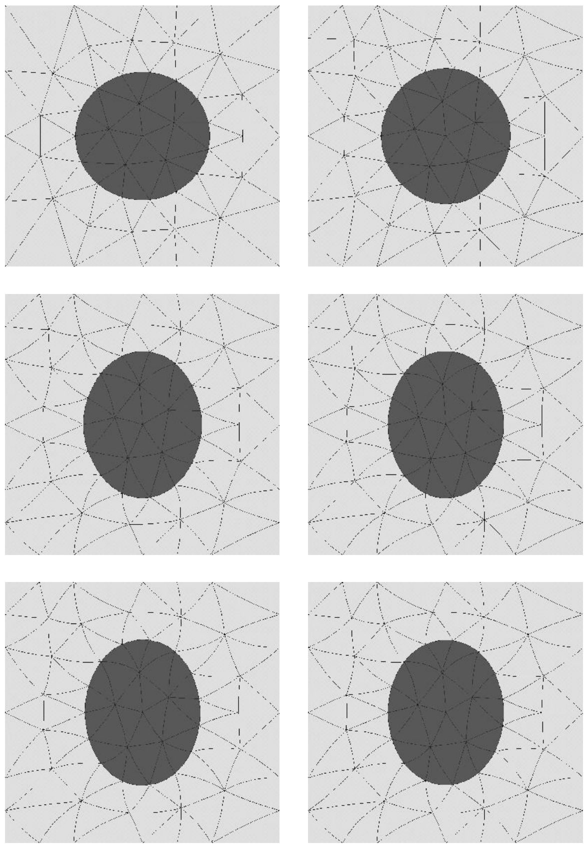

Figure 8: Stretched membrane with elasticity constant of 10.0. From left to right and top to bottom, the frames correspond to times $t = 0$, $t = 0.12$, $t = 0.24$, $t = 0.36$, $t = 0.48$ and $t = 0.60$ respectively.

present the results for two different set ups: one where the constant of elasticity for the membrane is 0.0, and another where the elasticity constant is 5.0. In both cases we start with little to no initial tension on the membrane.

It is observed that as the elasticity constant increases the fluid undergoes less deformation, as expected. See Figures 9 and 10.

6.3 Flow between parallel plates

In this simulation we consider fluid flowing between two parallel plates. The flow in this case establishes a parabolic profile. We have placed an elastic membrane off center in the fluid. As time progresses, the membrane begins to shear, until this is counteracted by the elastic tension, at which point the cell gradually realigns itself with the horizontal fluid flow.

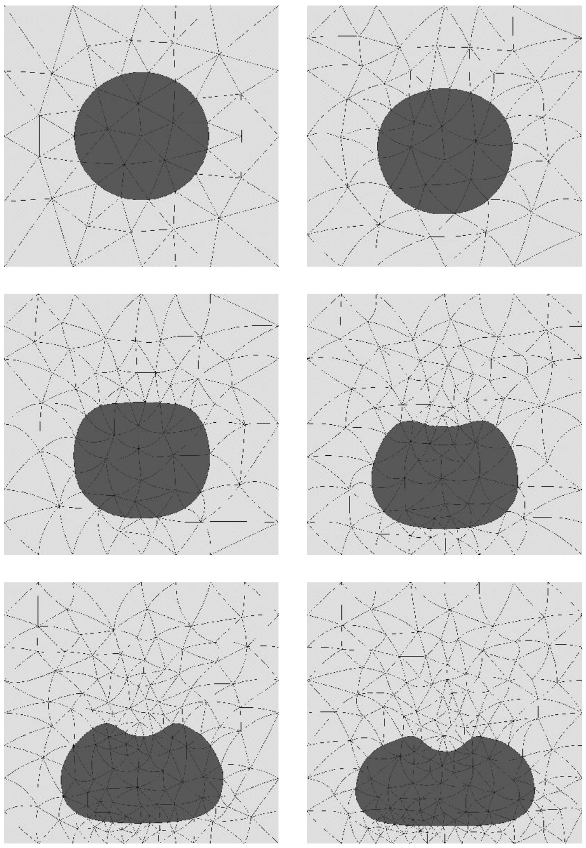

Figure 9: Falling membrane with elasticity constant of 0.0. From left to right and top to bottom, the frames correspond to times $t = 0.0$, $t = 0.30$, $t = 0.60$, $t = 0.90$, $t = 1.20$ and $t = 1.50$ respectively.

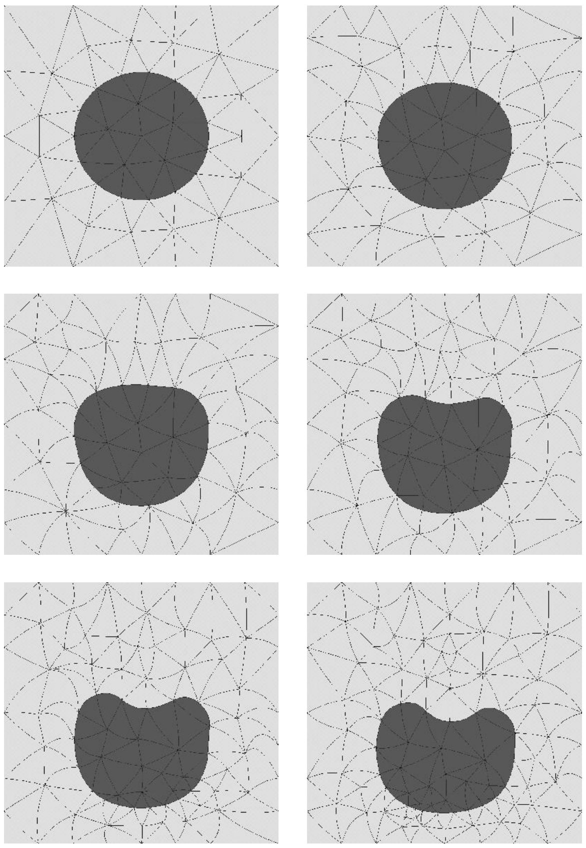

Figure 10: Falling membrane with elasticity constant of 5.0. From left to right and top to bottom, the frames correspond to times $t = 0.0$, $t = 0.30$, $t = 0.60$, $t = 0.90$, $t = 1.20$ and $t = 1.50$ respectively.

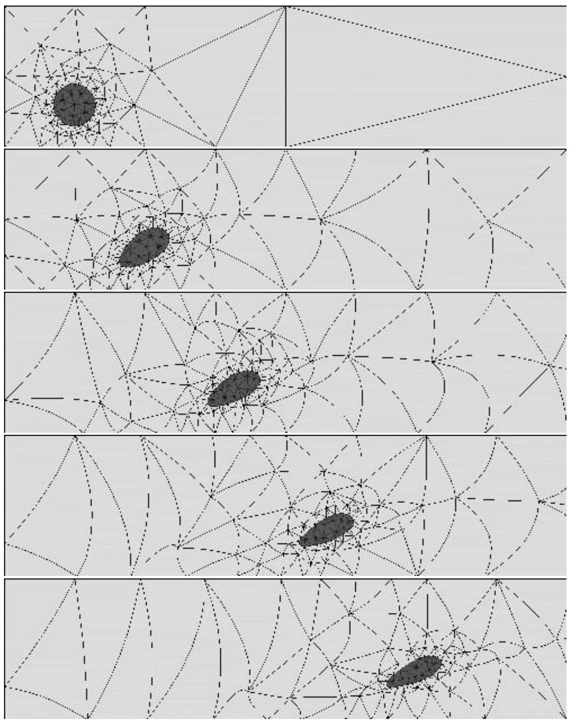

Figure 11: An elastic membrane moving with flow between two plates. Frames are shown at time $t = 0.0$, $t = 2.0$, $t = 4.5$, $t = 7.0$ and $t = 9.5$ from top to bottom.

7. FUTURE WORK

From the meshing perspective the main open research question is the development of three-dimensional simulations of blood flow. We believe that most of the tools we have developed for two-dimensional simulations will readily extend to three dimensions, but there are still several open geometric questions.

Within the two-dimensional realms, most of our future work is in the development of better numerical schemes. The current scheme which lags the elastic forces could be improved to include elasticity in the computation, by using an implicit method. Incorporating elasticity into the solver should yield better results. Another area in need of improvement is the time stepping procedure. The development of a higher-order time stepping scheme would allow for even larger time steps in our simulations. The last area of research is in better ways to approximate the boundary layer. Given that the boundary is precisely tracked by the moving mesh, we could foresee using higher-order elements only in the vicinity of the membrane, and using less accurate elements elsewhere. A mixed element-type approach of this nature would allow for greater accuracy of the solution without dramatically increasing the number of elements in the moving mesh. If we wish to use mixed elements for Navier-Stokes simulations we must be concerned with satisfying the Babuska-Brezzi conditions for compatability between our admissable spaces of velocity and pressure functions, thus some care must be taken when adding new basis elements to the finite element method.

References

[1] Cardoze D., Cunha A., Miller G., Phillips T., Walkington N. "A Bezier-Based Approach to Unstructured Moving Meshes." *20th Symposium on Computational Geometry.* 2004

[2] "The Sangria Project." URL `http://cs.cmu.edu/ sangria`. Supported by NSF ITR ACI-0086093

[3] Hill J. *A phasefield approach to modeling fluid-fluid interfaces in an Eulerian framework.* Ph.D. thesis, Carnegie Mellon, 2004

[4] Peskin C.S. "The immersed boundary method." *Acta Numerica*, 2002

[5] Lee L., LeVeque R. "An Immersed Interface Method for Incompressible Navier-Stokes equations." *SIAM Journal on Scientific Computing*, vol. 25, no. 3, 832–856, 2003

[6] Antaki J.F., Blelloch G.E., Ghattas O., Malcevic I., Miller G.L., Walkington N.J. "A Parallel Dynamic-Mesh Lagrangian Method for Simulation of Flows with Dynamic Interfaces." *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing.* 2000

[7] Kuprat A., George D., Linnebur E., Smith R.K., Trease H.E. "Moving Adaptive Unstructured 3-D Meshes in Semiconductor Process Modeling Applications." *VLSI Journal*, vol. 6(1-4), 373–378, 1998

[8] Wan J., Kocak S., Shephard M.S. "Automated Adaptive Forming Simulations." *Proceedings, 12th International Meshing Roundtable*, pp. 323–334. Sandia National Laboratories, September 14–17 2003

[9] Baker T. "Mesh Movement and Metamorphosis." *Proceedings, 10th International Meshing Roundtable*, pp. 387–396. Sandia National Laboratories, October 7–10 2001

[10] Li X.Y., Teng S.H., Üngör A. "Simultaneous refinement and coarsening: adaptive meshing with moving boundaries." *7th International Meshing Roundtable*, pp. 201–210. Dearborn, Mich., 1998. URL `citeseer.nj.nec.com/li98simultaneous.html`

[11] Farin G. *Curves and Surfaces for CAGD: A Practical Guide.* Morgan Kaufman, 2002

[12] Gallier J. *Curves and Surfaces in Geometric Modeling: Theory and Algorithms.* Morgan Kaufman, 1998

[13] Ruppert J. "A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation." *Journal of Algorithms*, vol. 18(3), 548–585, 1995

[14] Douglas D., Peucker T. "Algorithms for the reduction of the number of points required to represent a line or its caricature." *The Canadian Cartographer*, vol. 10, no. 2, 112–122, 1973

[15] Miller G.L., Talmor D., Teng S.H. "Optimal Coarsening of Unstructured Meshes." *Journal of Algorithms*, vol. 31, no. 1, 29–65, Apr 1999

[16] Talmor D. *Well-Spaced Points for Numerical Methods.* Ph.D. thesis, Carnegie Mellon University, Pittsburgh, August 1997. CMU CS Tech Report CMU-CS-97-164