Contemporary Mathematics
Volume **141**, 1993

An Implicit Numerical Method for Fluid Dynamics Problems with Immersed Elastic Boundaries

ANITA A. MAYO* AND CHARLES S. PESKIN†

ABSTRACT. Problems of biofluid-dynamics often involve the interaction of a viscous incompressible fluid with an immersed elastic boundary. One method of computing solutions of such problems is by using a regular Eulerian mesh for the fluid-dynamics calculation, and a Lagrangian representation of the immersed boundary as a sequence of particles or discrete delta functions. These two representations are coupled in a particle-mesh-type calculation. In this paper we introduce a new implicit method which is significantly less expensive than the fully implicit method, and which is more stable than the approximate implicit method.

1. Introduction

Problems of biological fluid dynamics often involve the interaction of a viscous incompressible fluid with an elastic immersed boundary. One approach to the solution of such problems is to use a regular Eulerian computational lattice for the fluid-dynamics computation together with a Lagrangian representation of the immersed boundary, the Eulerian and Lagrangian representations being coupled by a carefully chosen approximation to the Dirac delta function (Ref. 1). This approach has been applied to problems of blood flow in the heart (Ref. 1), wave propagation in the cochlea (Ref. 2), aquatic animal locomotion (Ref. 3), platelet aggregation during blood clotting (Ref. 4), and the flow of suspensions (Refs. 5, 6).

In all such computations, there is a serious issue of numerical stability. Instability may arise in the following way. Suppose that the force-field applied by the immersed boundary to the fluid at a given time-step is computed from the

1991 *Mathematics Subject Classification.* Primary 65; Secondary 35.

The final version of this paper will be submitted for publication elsewhere. The second author was supported in part by the National Institutes of Health under research grant HL17859.

boundary configuration at the beginning of the time-step (an *explicit* scheme). Then, if the boundary is too stiff or the time-step is too large, the boundary may overshoot equilibrium and arrive, at the end of the time-step, at a configuration for which the force-field is roughly in the opposite direction and of considerably greater magnitude than the force-field computed at the beginning of the time-step. Repetition of this process over several time-steps leads to a complete failure of the computation; the boundary appears to explode.

The difficulty that we have just described may be overcome in principle by using an *implicit* scheme, in which the boundary force is computed from the boundary configuration at the end of the time-step. Note, however, that this configuration is unknown (hence the name "implicit") and depends in a complicated way on the very force which one is trying to compute. Because of the difficulty in actually solving for the boundary force at the end of the time-step, most immersed-boundary computations have used an *approximately implicit* scheme in which, for purposes of the force computation only, an approximation has been introduced to simplify the influence of the boundary force on the boundary configuration at the end of the time-step.

The thesis of Tu (Ref. 7) compares the behavior of an explicit, an approximately implicit, and an implicit scheme (for Stokes flow) in a region containing an immersed elastic boundary. In her work, the approximately-implicit scheme has a greater range of stability than the explicit scheme, but it is not unconditionally stable. The implicit scheme does appear to be unconditionally stable, but it is very expensive to use. Thus, Tu's thesis leads to the challenge of implementing an implicit scheme for the immersed-boundary problem at a reasonable cost. The goal of this paper is to meet that challenge.

2. Equations of motion

For simplicity we consider the model problem of a two-dimensional viscous incompressible fluid containing an immersed massless elastic boundary in the form of a simple closed curve. The fluid is contained in a periodic square domain Ω of side L and is characterized by its mass density ρ and its viscosity μ. The fluid velocity and pressure are to be described in Eulerian form by the functions $\mathbf{u}(\mathbf{x},t)$ and $p(\mathbf{x},t)$.

The immersed boundary is to be described in Lagrangian form by the function $\mathbf{X}(s,t)$, $0 \le s \le 2\pi$, with $\mathbf{X}(s+2\pi,t) = \mathbf{X}(s,t)$. This function gives the position at the t of the material point whose Lagrangian label is s. We assume that the force applied by an arc ds of the boundary to the fluid with which it is in contact is given by the expression $\mathbf{f}(s,t)\,ds$ where $\mathbf{f}(s,t) = K\partial^2\mathbf{X}/\partial s^2$, and where K is a given stiffness constant.

Remark. This particular elasticity law may be derived from the assumption that the elastic energy stored in the immersed boundary at time t is given by

$$E = \frac{1}{2}K\int_0^{2\pi}\left|\frac{\partial \mathbf{X}}{\partial s}\right|^2 ds. \tag{1}$$

Note that this is minimized when the boundary curve has shrunk to a point. Thus we are assuming that the unstressed length of the boundary is zero, a condition which is necessary in order to achieve a linear mapping from $\mathbf{X}(\,,t)$ to $\mathbf{f}(\,,t)$. (As the reader may verify, a linear stress-strain relation with nonzero rest length leads to a nonlinear mapping from $\mathbf{X}$ to $\mathbf{f}$.) This linearity is convenient but not essential for what follows.

With the above notation and assumptions we have the following equations of motion:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla p+\mu\nabla^2\mathbf{u}+\mathbf{F}(\mathbf{x},t), \tag{2}$$

$$\nabla\cdot\mathbf{u}=0, \tag{3}$$

$$\mathbf{F}(\mathbf{x},t)=K\int_0^{2\pi}\frac{\partial^2\mathbf{X}}{\partial s^2}(s,t)\delta^2\left[\mathbf{x}-\mathbf{X}(s,t)\right]ds, \tag{4}$$

$$\frac{\partial \mathbf{X}}{\partial t}(s,t)=\mathbf{u}\left[\mathbf{X}(s,t),t\right]=\int_\Omega \mathbf{u}(x,t)\delta^2\left[\mathbf{x}-\mathbf{X}(s,t)\right]d\mathbf{x}. \tag{5}$$

Note that equations (2) and (3) are the Navier-Stokes equations of a viscous incompressible fluid acted upon by a force density $\mathbf{F}(\mathbf{x},t)$. In our case, this force density arises from the immersed boundary and is given by equation (4). The delta function in equation (4) is two-dimensional, $\delta^2(\mathbf{x})=\delta(x_1)\delta(x_2)$, where $\mathbf{x}=(x_1,x_2)$, and it expresses the local character of the interaction between the immersed boundary and the fluid. It is also worth mentioning that equation (4) is equivalent to the statement that

$$\int \mathbf{F}(\mathbf{x},t)\cdot\mathbf{w}(\mathbf{x},t)\,d\mathbf{x}=K\int_0^{2\pi}\frac{\partial^2\mathbf{X}}{\partial s^2}(s,t)\cdot\mathbf{w}\left[\mathbf{X}(s,t),t\right]ds \tag{6}$$

for all smooth test functions $\mathbf{w}$. If we interpret $\mathbf{w}$ as a velocity, this is equivalent to conservation of energy, since the left-hand side is the work done *on* the fluid and the right-hand side is the work done *by* the immersed boundary. Equation (5) is the no-slip condition which here appears as an equation of motion for the immersed boundary. The second form of equation (5) is written down to emphasize the symmetry with equation (4) and to motivate our numerical scheme.

3. The implicit scheme

The fluid equations are discretized according to the projection method of Chorin (Ref. 8) with the boundary force added in the projection step. The boundary force is determined, however, from the unknown boundary configuration at the end of the time-step. The resulting implicit scheme may be stated as follows:

$$\rho\left(\frac{\mathbf{u}^{n+1,1}-\mathbf{u}^n}{\Delta t}+u_1^n D_1^0\mathbf{u}^{n+1,1}\right)=\mu D_1^+D_1^-\mathbf{u}^{n+1,1}, \tag{7}$$

$$\rho\left(\frac{\mathbf{u}^{n+1,2}-\mathbf{u}^{n+1,1}}{\Delta t}+u_2^n D_2^0 \mathbf{u}^{n+1,2}\right)=\mu D_2^+ D_2^- \mathbf{u}^{n+1,2}, \tag{8}$$

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n+1,2}}{\Delta t}+\mathbf{D}p^{n+1}=\mathbf{F}^{n+1}, \tag{9}$$

$$\mathbf{D}\cdot\mathbf{u}^{n+1}=0, \tag{10}$$

$$\mathbf{F}^{n+1}(\mathbf{x})=K\sum_s\left(D_s^+D_s^-\mathbf{X}^{n+1}\right)(s)\delta_h^2\left[\mathbf{x}-\mathbf{X}^n(s)\right]h_B, \tag{11}$$

$$\frac{\mathbf{X}^{n+1}(s)-\mathbf{X}^n(s)}{\Delta t}=\sum_{\mathbf{x}}\mathbf{u}^{n+1}(\mathbf{x})\delta_h^2\left[\mathbf{x}-\mathbf{X}^n(s)\right]h^2. \tag{12}$$

In these equations, the first superscript on a variable is the time-step index. Thus, $\mathbf{u}^n=\mathbf{u}(\ ,n\Delta t)$, where Δt is the duration of the time-step. When a second superscript appears, it denotes an intermediate quantity computed on the way to determining the main quantity of interest. Thus, $\mathbf{u}^{n+1,1}$ and $\mathbf{u}^{n+1,2}$ are stepping stones in the computation of $\mathbf{u}^{n+1}$. The independent variable $\mathbf{x}$ takes values in the lattice of points $\mathbf{x}=(j_1h,j_2,h)$, where j_1 and j_2 are integers. The sum in equation (12) is a sum over this lattice. Similarly, the independent variable s takes values of the form $s=kh_B, h_B=2\pi/n_B$, where k and n_B are integers. The sum in equation (11) is a sum over one period of s, that is, over those values of s given by $k=0,\dots,n_B-1$. The subscripts in equations (7) and (8) refer to the two space directions. Thus, u_1^n and u_2^n are the two components of the vector $\mathbf{u}^n$.

The spatial difference operators which appear in equations (7) to (10) are defined as follows. For $r=1,2$,

$$\left(D_r^+\phi\right)(\mathbf{x})=\frac{\phi(\mathbf{x}+h\mathbf{e}_r)-\phi(\mathbf{x})}{h}, \tag{13}$$

$$\left(D_r^-\phi\right)(\mathbf{x})=\frac{\phi(\mathbf{x})-\phi(\mathbf{x}-h\mathbf{e}_r)}{h}, \tag{14}$$

$$\left(D_r^0\phi\right)(\mathbf{x})=\frac{\phi(\mathbf{x}+h\mathbf{e}_r)-\phi(\mathbf{x}-h\mathbf{e}_r)}{2h}, \tag{15}$$

$$\mathbf{D}=\left(D_1^0,D_2^0\right)=\sum_{r=1}^{2}\mathbf{e}_rD_r^0, \tag{16}$$

where $\mathbf{e}_r$ is the unit vector in the space direction r. Note that $\mathbf{D}$ is the difference analog of the vector differential operator ∇, and that $\mathbf{D}p$ corresponds to ∇p, the gradient of p, while $\mathbf{D}\cdot\mathbf{u}$ corresponds to $\nabla\cdot\mathbf{u}$, the divergence of $\mathbf{u}$. The difference operators D_s^+ and D_s^- which appear in equation (11) are applied to functions of the boundary variables. Their definitions are similar to equations (13) and (14):

$$\left(D_s^+\psi\right)(s)=\frac{\psi(s+h_B)-\psi(s)}{h_B}, \tag{17}$$

$$\left(D_s^-\psi\right)(s)=\frac{\psi(s)-\psi(s-h_B)}{h_B}. \tag{18}$$

The function δ_h^2 which appears in equations (11) and (12) is defined as follows:

$$\delta_h^2(\mathbf{x}) = \delta_h(x_1)\delta_h(x_2), \tag{19}$$

where $\mathbf{x} = (x_1, x_2)$, and where

$$\delta_h(x) = \begin{cases} \frac{1}{4h}\left(1 + \cos\frac{\pi x}{2h}\right) & x \leq 2h \\ 0 & x \geq 2h \end{cases} \tag{20}$$

This is a smoothed approximation to the Dirac delta function. The motivation for this particular choice of δ_h is discussed in (Ref. 1).

Clearly, equations (10) to (12) are discretizations of equations (3) to (5), respectively. To see the connection between equations (7) to (9) and equation (2), just add equations (7) to (9) and note the cancellation of $\mathbf{u}^{n+1,1}$ and $\mathbf{u}^{n+1,2}$ in the time-difference terms. The result is

$$\begin{aligned} \rho\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + u_1^n D_1^0 \mathbf{u}^{n+1,1} + u_2^n D_2^0 \mathbf{u}^{n+1,2}\right) \\ = -\mathbf{D}p^{n+1} + \mu\left(D_1^+ D_1^- \mathbf{u}^{n+1,1} + D_2^+ D_2^- \mathbf{u}^{n+1,2}\right) + \mathbf{F}^{n+1}. \end{aligned} \tag{21}$$

This is a discretization of equation (2), since $\mathbf{u}^{n+1,1}$ and $\mathbf{u}^{n+1,2}$ are within $O(\Delta t)$ of $\mathbf{u}^n$ (or $\mathbf{u}^{n+1}$).

The structure of the implicit scheme, equations (7) to (12), may now be summarized as follows. In equation (7), the unknowns are $\mathbf{u}^{n+1,1}(\mathbf{x})$. The difference operators in equation (7) involve coupling in the x_1-direction only, and points are only coupled to their nearest neighbors. Thus, equation (7) amounts to collection of separate tridiagonal systems, one for each row of the computational lattice. Similarly, equation (8) is a collection of separate tridiagonal systems for the unknowns $\mathbf{u}^{n+1,2}(\mathbf{x})$, and there is one such tridiagonal system for each column of the computational lattice. Note that equations (7) and (8) do not involve the immersed boundary at all.

Once equations (7) and (8) have been solved, it is necessary to solve equations (9) to (12) simultaneously for the unknowns $\mathbf{u}^{n+1}$, p^{n+1}, $\mathbf{X}^{n+1}$, $\mathbf{F}^{n+1}$. How to do so is described in the next section.

4. Implicit equations for boundary configuration at end of time-step

This section is concerned with the solution of equations (9) to (12). We begin by introducing a more succinct notation. First, note that the subsystem of equations (9) and (10) defines $\mathbf{u}^{n+1}$ as the orthogonal projection of $\mathbf{u}^{n+1,2} + (\Delta t/\rho)\mathbf{F}^{n+1}$ onto the space of (discretely) divergence-free vector fields. Thus, we may write

$$\begin{aligned} \mathbf{u}^{n+1} &= P(\mathbf{u}^{n+1,2} + \frac{\Delta t}{\rho}\mathbf{F}^{n+1}) \\ &= P\mathbf{u}^{n+1,2} + \frac{\Delta t}{\rho} P\mathbf{F}^{n+1}, \end{aligned} \tag{22}$$

where the operator P has the characteristic properties of an orthogonal projection:

$$P^2 = P = P^*. \tag{23}$$

(Here and in the sequel we use the superscript * to denote the adjoint of an operator.)

Next we introduce notation for the operations which appear in equations (11) and (12). First, let S^n be the interpolation operator defined as follows:

$$(S^n\phi)(s) = \sum_{\mathbf{x}} \phi(\mathbf{x})\delta_h^2(\mathbf{x} - \mathbf{X}^n(s))h^2. \tag{24}$$

Its adjoint, $(S^n)^*$, is given by

$$((S^n)^*\phi)(\mathbf{x}) = \sum_{s} \phi(s)\delta_h^2(\mathbf{x} - \mathbf{X}^n(s))h_B. \tag{25}$$

The statement that $(S^n)^*$ is the adjoint of S^n is justified by the following identity:

$$\begin{aligned}\sum_{s} \phi(s)(S^n\phi)(s)h_B &= \sum_{\mathbf{x},s} \phi(s)\phi(\mathbf{x})\delta_h^2\left[\mathbf{x} - \mathbf{X}^n(s)\right]h^2h_B \\ &= \sum_{\mathbf{x}} \phi(\mathbf{x})\left[(S^n)^*\phi\right](\mathbf{x})h^2. \end{aligned}\tag{26}$$

In terms of S^n and $(S^n)^*$, equations (11) and (12) become

$$\mathbf{F}^{n+1} = (S^n)^*KD_s^+D_s^-\mathbf{X}^{n+1}, \tag{27}$$

$$\mathbf{X}^{n+1} = \mathbf{X}^n + (\Delta t)S^n\mathbf{u}^{n+1}. \tag{28}$$

Combining equations (22), (27), and (28), we get the following linear system for $\mathbf{X}^{n+1}$:

$$\mathbf{X}^{n+1} = \left[\mathbf{X}^n + (\Delta t)S^nP\mathbf{u}^{n+1,2}\right] + \frac{(\Delta t)^2K}{\rho}S^nP(S^n)^*D_s^+D_s^-\mathbf{X}^{n+1}. \tag{29}$$

This system is of the form

$$\mathbf{X} = \mathbf{Z} + SPS^*A\mathbf{X}, \tag{30}$$

where we have dropped the time-step indices, and where we have introduced the notation $\mathbf{Z}$ for the known vector $\mathbf{X}^n + (\Delta t)S^nP\mathbf{u}^{n+1,2}$, and the notation A for the operator $\left[(\Delta t)^2K/\rho\right]D_s^+D_s^-$. It is easy to show that A is symmetric and negative semidefinite:

$$A^* = A \leq 0. \tag{31}$$

(We use the shorthand $A \leq 0$ to mean $\mathbf{X}^*A\mathbf{X} \leq 0$ for all $\mathbf{X}$.) Moreover, the matrix A has the following useful property:

$$\mathbf{X}^*A\mathbf{X} = 0 \Rightarrow A\mathbf{X} = 0. \tag{32}$$

To see this, recall that $D_s^+ = (D_s^-)^*$. Hence, if $\mathbf{X}^*A\mathbf{X} = 0$, then $\mathbf{X}^*(D_s^-)^*D_s^-\mathbf{X} = 0$, which implies that $D_s^-\mathbf{X} = 0$ and, hence, that $A\mathbf{X} = 0$.

We can use this property of A to show that $(I - SPS^*A)$ is invertible (and hence that equation (30) has a unique solution). Suppose

$$(I - SPS^*A)\mathbf{X} = 0. \tag{33}$$

We want to show that $\mathbf{X} = 0$. To do this, multiply both sides by $-\mathbf{X}^*A$. The result is

$$-\mathbf{X}^*A\mathbf{X} + \mathbf{X}^*ASPS^*A\mathbf{X} = 0. \tag{34}$$

Since both terms on the left are nonnegative, they must be separately zero. Thus, $\mathbf{X}^*A\mathbf{X} = 0$, and this implies that $A\mathbf{X} = 0$ (as shown above). Once we know that $A\mathbf{X} = 0$ equation (33) shows that $\mathbf{X} = 0$. This completes the proof that $I - SPS^*A$ is invertible. It follows that equation (30) [and hence equations (9) to (12)] have a unique solution. The implicit scheme is well-defined.

We have solved the linear system described by equation (30) by several different methods.

The simplest method we used was by the iteration of the form

$$(I - \lambda A)(\mathbf{x}^{m+1} - \mathbf{X}^m) = \mathbf{Z} - (I - SPS^*AA)\mathbf{X}^m, \tag{35}$$

where the superscripts here denote iteration number (within a given time-step), not the time-step index. The operator λ which appears in equation (35) is a multiplication operator (diagonal matrix) consisting of multiplication by the function $\lambda(s)$ defined as follows:

$$\lambda = SS^*1, \tag{36}$$

where $1(s)$ is the function that takes the value 1 for every s. More concretely,

$$\lambda(s) = \sum_{\mathbf{X},s'} \delta_h^2\left[\mathbf{x} - \mathbf{X}(s)\right]\delta_h^2\left[\mathbf{x} - \mathbf{X}(s')\right]h^2 h_B. \tag{37}$$

Note that $\lambda(s) \geq 0$ for all s. [Recall the definition of δ_h^2, equations (19) and (20)] Thus, the same form of proof that was used to establish the invertibility of $(I - SPS^*A)$ can be used again to show that $(I - \lambda A)$ is invertible (we skip the details). It follows that the iteration described by equation (35) is well-defined.

Let $\mathbf{X}$ be the solution of equation (30). Then $\mathbf{X}$ (trivially) satisfies

$$(I - \lambda A)(\mathbf{X} - \mathbf{X}) = \mathbf{Z} - (I - SPS^*A)\mathbf{X}. \tag{38}$$

Subtracting this from equation (35) we find

$$(I - \lambda A)(\mathbf{E}^{m+1} - \mathbf{E}^m) = -(I - SPS^*A)\mathbf{E}^m, \tag{39}$$

where

$$\mathbf{E}^m = \mathbf{X}^m - \mathbf{X} \tag{40}$$

is the error at the mth iteration.

To study the behavior of the error, we consider the eigenvectors of this process. That is, we seek nonzero vectors $\mathbf{E}$ such that $\mathbf{E}^m = \mu^m \mathbf{E}$ (for some number μ) is a solution of equation (39). The equations for $\mathbf{E}$ and μ are as follows:

$$(\mu - 1)(I - \lambda A)\mathbf{E} = -(I - SPS^* A)\mathbf{E}, \tag{41}$$

or

$$\mu(I - \lambda A)\mathbf{E} = -(\lambda A - SPS^* A)\mathbf{E}. \tag{42}$$

If $A\mathbf{E} = 0$, then $\mu = 0$. Otherwise, we multiply both sides by $-\mathbf{E}^* A$ and solve for μ as follows:

$$\mu = \frac{\mathbf{E}^* A\lambda A\mathbf{E} - \mathbf{E}^* ASPS^* A\mathbf{E}}{(-\mathbf{E}^* A\mathbf{E}) + \mathbf{E}^* A\lambda A\mathbf{E}}, \tag{43}$$

$$1 - \mu = \frac{(-\mathbf{E}^* A\mathbf{E}) + \mathbf{E}^* ASPS^* A\mathbf{E}}{(-\mathbf{E}^* A\mathbf{E}) + \mathbf{E}^* A\lambda A\mathbf{E}}. \tag{44}$$

These formulas show that μ is real. We shall prove that $0 \le \mu < 1$ (and hence that the iteration converges).

For the various terms appearing in the formulae for μ and $1 - \mu$, we have the following inequalities:

$$\mathbf{E}^* A\lambda A\mathbf{E} \ge 0, \tag{45}$$

$$\mathbf{E}^* ASPS^* A\mathbf{E} \ge 0, \tag{46}$$

$$-\mathbf{E}^* A\mathbf{E} \ge 0. \tag{47}$$

In the last case, we have equality *only* if $A\mathbf{E} = 0$ [recall this useful property of the operator A, equation (32)], and we have already shown that $A\mathbf{E} = 0 \Rightarrow \mu = 0$. Otherwise, we may assume that

$$-\mathbf{E}^* A\mathbf{E} > 0. \tag{48}$$

From these inequalities it follows at once that $1 - \mu > 0$ and hence that $\mu < 1$. To show that $\mu \ge 0$, we need one further result, namely,

$$\mathbf{E}^* A\lambda A\mathbf{E} \ge \mathbf{E}^* ASPS^* A\mathbf{E}. \tag{49}$$

This will be proved by showing that

$$\lambda \ge SPS^*, \tag{50}$$

for which it is sufficient to show that

$$\lambda \ge SS^*, \tag{51}$$

since the projection operator P satisfies $I \ge P$.

To establish the inequality $\lambda \ge SS^*$, we note that the operation described by SS^* is of the form

$$(SS^* \psi)(s) = \sum_{s'} m(s, s')\psi(s')h_B, \tag{52}$$

where

$$m(s, s') = m(s', s) \geq 0. \tag{53}$$

In this notation

$$\lambda(s) = \sum_{s'} m(s, s') \cdot 1 h_B. \tag{54}$$

Therefore, for arbitrary ψ we have

$$\begin{aligned}
\psi^* S S^* \psi &= \sum_{ss'} m(s, s') \psi(s') \psi(s) h_B^2 \\
&\leq \frac{1}{2} \sum_{ss'} m(s, s')(\psi^2(s) + \psi^2(s')) h_B^2 \\
&= \sum_{ss'} m(s, s') \psi^2(s) h_B^2 \\
&= \sum_{s} \lambda(s) \psi^2(s) h_B = \psi^* \lambda \psi.
\end{aligned} \tag{55}$$

This shows that $\lambda \geq SS^*$ and hence that $\mu \geq 0$.

In summary, we have reduced the equations for the boundary configuration at the end of the time-step to a system of the form

$$(I - SPS^* A)\mathbf{X} = \mathbf{Z}, \tag{56}$$

and we have shown that this system can be solved by an iterative scheme of the form

$$(I - \lambda A)(\mathbf{X}^{m+1} - \mathbf{X}^m) = \mathbf{Z} - (I - SPS^* A)\mathbf{X}^m. \tag{57}$$

Note that the matrix $I - \lambda A$, which is inverted at each step of the ieration, is tridiagonal, whereas the matrix $(I - SPS^* A)$ is dense. When we use the iterative scheme defined by equation (57), there is, of course, no need to compute the elements of the dense matrix $(I - SPS^* A)$. Instead, the operators that appear in it are applied one at a time. The cost of each step of the ieration is therefore quite small, $O(n_b)$.

We implemented this method on several different problems with different time-steps, stiffness, and initial configuration. In practice, we found that although iteration (57) was always convergent, and often much more stable than the approximately implicit and explicit methods, it was not always stable.

Part of the reason for this lack of stability is that the method is not completely implicit. The reason it is not completely is that the arguments of the interpolating operator S and its adjoint S^* involve the position of the boundary at the beginning of the time-step. Thus, one way of possibly increasing the stability is to replace the operators S^n and S^{n*} by S^m and S^{m*}, where $(S^m \phi)(s) = \sum_{\mathbf{x}} \phi(\mathbf{x}) \delta_h^2(\mathbf{x} - \mathbf{X}^m(s)) h^2$.

That is, instead of using equation (35), we can use the iteration

$$(I - \lambda A)(\mathbf{X}^{m+1} - \mathbf{X}^m) = \mathbf{Z} - (I - S^m P S^{m*} A)\mathbf{X}^m. \tag{58}$$

Essentially the same argument as given above for equation (35) shows that this iteration is also convergent. [The argument uses the fact that

$$\mathbf{X}^{m+1} = (I - (I - \lambda A)^{-1}(I - S^m P S^{m*} A))\mathbf{X}^m + (I - \lambda A)^{-1}\mathbf{Z},$$

and the fact that $(I - (I - \lambda A)^{-1}(I - S^m P S^{m*} A))$ is a contraction.]

The solution of this iteration gives a solution of the nonlinear equation

$$\mathbf{X} = \mathbf{Z} + S(\mathbf{X}) P S^*(\mathbf{X}) A \mathbf{X}, \tag{59}$$

where now $\mathbf{Z} = \mathbf{X}^n + (\Delta t) S^{n+1} P \mathbf{u}^{n+1,2}$.

The solution of equation (59) is also, of course, an approximation to the position of the boundary at the end of the time-step. However, instead of equations (27) and (28) we have

$$\mathbf{F}^{n+1} = (S^{n+1})^* K D_s^+ D_s^- \mathbf{X}^{n+1}, \tag{60}$$

$$\mathbf{X}^{n+1} = \mathbf{X}^n + (\Delta t) S^{n+1} \mathbf{u}^{n+1}. \tag{61}$$

Thus, the force and final velocity values used are more nearly equal to their values at the end of the time-step.

We also implemented this method and, as expected, found that the iteration defined by equation (58) was always convergent. Furthermore, this method was stable more often than the previous one.

Although both of the above iterations (57) and (58) were always convergent, they often converged rather slowly. One can, of course, try to accelerate the convergence of the iterations. We tried to do this by using Aitken extrapolation. Aitken extrapolation is a standard method that can be used to accelerate the convergence of a linearly convergent sequence. It uses the fact that if the sequence x_n converges linearly, successive error vectors satisfy equations of the form

$$e^k = x^* - x^k \approx \lambda(x^* - x^{k-1}),$$

and

$$e^{k+1} = x^* - x^{k+1} \approx \lambda(x^* - x^k).$$

Dividing the first equation by the second gives the following extrapolated results for the ith component:

$$x_i^* = x_i^* + \frac{x_i^{k+1} - x_i^k}{1 - \frac{x_i^{k+1} - x_i^k}{x_i^k - x_i^{k-1}}}.$$

Although this technique worked well for small values of m, its behavior was sometimes erratic for large values. (This type of behavior is typical and well known). We found, in general, that both mothods converged most rapidly when we only used one step of Aitken extrapolation per time-step.

We also solved equation (30) by the preconditioned conjugate-gradient-squared method (Ref. 9) with preconditioning matrix $I - \lambda A$. The conjugate-gradient-squared method is a bi-orthogonalization iterative method for solving nonsymmetric linear systems of equations adapted from the conjugate-gradient iteration.

It makes use of a three-term recurrence, and each iteration requires two matrix vector multiplies. Each iteration is, therefore, about twice as expensive as the above iterations (57) and (58). Although, in order to get the method to converge we had to use the position of the boundary at the beginning of the time-step in the arguments of S and S^*, we found that on the problems we tested this method was no less stable than when we used the iteration (58). (However, this may not always be the case, and the method is not guaranteed to converge.) Furthermore, we found that this method converged so quickly that it was usually less expensive than the others. Note that the preconditioning matrix we used is symmetric and the matrix equation we solved is normal. It is known that when this is the case the rate of convergence depends on the spectrum of the matrix (Ref. 10).

Results

This section presents the results of our experiments. In all cases we started out with an initial boundary configuration with 192 points on it, and embedded it in a periodic unit square with a uniform 64 x 64 mesh. We always chose $\rho = 1.1$, $\mu = 1.3$, and time-step $dt = 0.7(dx)^2\mu/\rho$ where $dx = 0.9/64$. Since the unstressed length of the boundary was zero, the final position of each boundary should have always been a circle. In each set of experiments we used the approximately implicit method, the method derived from iteration (57) (method 1), the method derived from iteration (58) (method 2), and the conjugate-gradient-squared method. Our convergence tolerance for ending the iterations was always 10^{-4}.

In the approximately implicit method that we used, the force was placed at the right-hand side of equation (7), instead of at the right-hand side of equation (9), and was computed in the following way. The position of the boundary at the $n+1$th time-step was approximated by the solution of the equation

$$(I - \lambda A)\mathbf{X}^{n+1} = \mathbf{X}^n$$

and the force was computed in the usual way at this position. Thus, each step of the approximately implicit method was much less expensive than a step of the iterations (57) and (58).

In the first set of experiments the initial boundary was the ellipse with semi-axes 0.4 and 0.2. We initially choose the stiffness constant $K + 10,000$, and took 12 time-steps. The boundary converged to a circle with all the methods.

In the next set of experiments (problem 2) we used the same initial configuration, but increased the stiffness to $K = 250,000$, multiplied the time-step by 5, and took four time-steps. In Figures 1–3, the final positions of the boundary computed by the conjugate-gradient-squared method, by method 1, and by method 2 are plotted. In Figure 4 the final position of the boundary computed by the approximately implicit method after 20 steps with the original time-step is plotted. The approximately implicit method is clearly unstable, and method 1 also appears to be unstable. To verify that this was indeed the case, we plotted the boundary configuration after five time-steps. (Fig. 5) Both method 2 and the conjugate-gradient method were stable and converged to the correct solution.

In the next set of experiments (problem 3), the initial configuration was the (nonconvex) crescent given parametrically by $x(t) = (\cos t + 0.15\sin^2 t + 1/3)/2.4$, $y(t) = (\sin t + 0.7\cos^2 t + 1.3)/2.4$. (Fig. 6) The stiffness constant was set equal to 200,000, the time-step was multiplied by 4, and we took four time-steps. In Figures 7–9 the final positions of the boundary computed by the conjugate-gradient squared method, by method 1, and by the method 2 are plotted, and in Figure 10 the position computed by the approximately implicit method after 16 steps with the original time-step is plotted. Again, the approximately implicit method is clearly unstable, but this time method 1 appears to be stable. Also, neither the calculation with method 1 nor the one with method 2 had yet converged. (Further calculation showed them to be converging.)

In our last set of experiments (problem 4) the initial configuration was the crescent given parametrically by $x(t) = (\cos t + 0.15\sin^2 t + 1/3)/2.4$, $y(t) = (\sin t + 0.9\cos^2 t + 1.3)/2.4$. (Fig. 11) This crescent has much larger curvature than the crescent of problem 3, and when we used the same stiffness and time-step as in problem 3, none of the methods worked. Instead, we set the stiffness constant equal to 150,000, used the original time-step, and took 24 time-steps. In Figures 12–15 the final positions of the boundary computed by the conjugate-gradient-squared method, by method 1, by method 2, and by the approximately implicit method are plotted. Here too, the approximately implicit method is unstable, and the others appear to be stable. This time, however, the conjugate-gradient-squared method was the only one that had not yet converged. Further calculations showed that the boundary was oscillating around the equilibrium position before converging. (In Fig. 16 the boundary is plotted after 36 time-steps.)

These results show that all of the three new methods we have proposed for solving the implicit equation are much more stable than the approximately implicit method. That is, we could perform calculations where the boundary was stiffer and the time-step was larger than with the approximately implicit method. Method 1, however, was occasionally less stable than the other two. Of course, when it converged the approximately implicit method was much less expensive than the others.

It is also necessary to compare the number of iterations necessary to achieve convergence. In general, we found that the conjugate-gradient-squared method required fewer than half the number of iterations required by the other two. For example, on problem 4, four time-steps took a total of 30 iterations with method 1, 31 with method 2 and only 10 with the conjugate-gradient-squared method. When methods 1 and 2 were accelerated using Aitken extrapolation the iteration counts went down to 23 and 24 respectively.

However, it is also important to note that methods 1 and 2 are guaranteed to converge, whereas the conjugate-gradient-squared method is not. And again, methods 1 and 2 converge monotonically, but this is not always true of the conjugate-gradient-squared method.

REFERENCES

1. Peskin, C. S.: Numerical Analysis of Blood Flow in the Heart. J. Comput. Phys., vol. 25, 1977, pp. 220–252.
2. Beyer, R.: A Computational Model of the Cochlea Using the Immersed Boundary Method. Technical Report No. 89-8, Applied Mathematics, University of Washington, Seattle, Wash., Dec. 1989.
3. Fauci, L.; and Peskin, C.: A Computational Model of Aquatic Animal Locomotion. J. Comput. Phys., vol. 77, 1988, pp. 85–108.
4. Fogelson, A.: A Mathematical Model and Numerical Method for Studying Platelet Adhesion and Aggregation during Blood Clotting. J. Comput. Phys., vol. 56, 1984, pp. 111–134.
5. Fogelson, A.; and Peskin, C.: A Fast Numerical Method for Solving the Three-Dimensional Stokes Equations in the Presence of Suspended Particles. J. Comput. Phys., vol. 79, 1988, pp. 50–69.
6. Sulsky, D.; and Brackbill, J.: A Numerical Method for Suspension Flow. J. Comput. Phys., vol. 96, 1991, pp. 339–368.
7. Tu, C.; and Peskin, C. S.: Stability and Instability in the Computation of Flows with Moving Immersed Boundaries. To appear in J. Comput. Phys..
8. Chorin, A.: On the Convergence of Discrete Approximations to the Navier-Stokes Equations. Math. Comp., vol.23, 1959, pp. 341–353.
9. Sonneveld, P.: CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear Systems. SIAM J. Sci. Stat. Comp., vol. 10, 1989, pp. 36–52.
10. Nachtigal, N.; Reddy, S.; and Tregethan, L.: How Fast Are Nonsymmetric Matrix Iterations? Manuscript.

* WATSON RESEARCH CENTER, IBM, YORKTOWN HEIGHTS, NEW YORK 10598.

† COURANT INSTITUTE OF MATHEMATICAL SCIENCES, NEW YORK UNIVERSITY, 251 MERCER ST, NEW YORK, NEW YORK 10021.

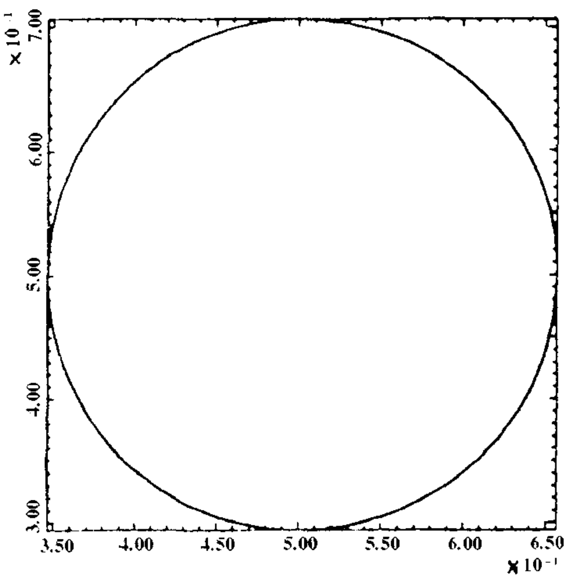

Figure 1

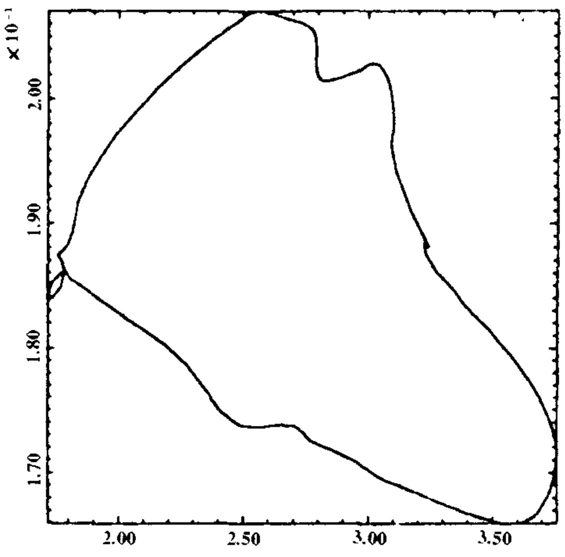

Figure 2

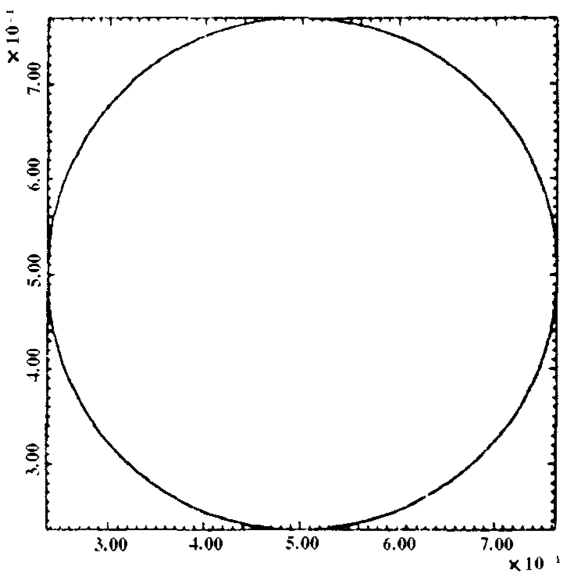

Figure 3

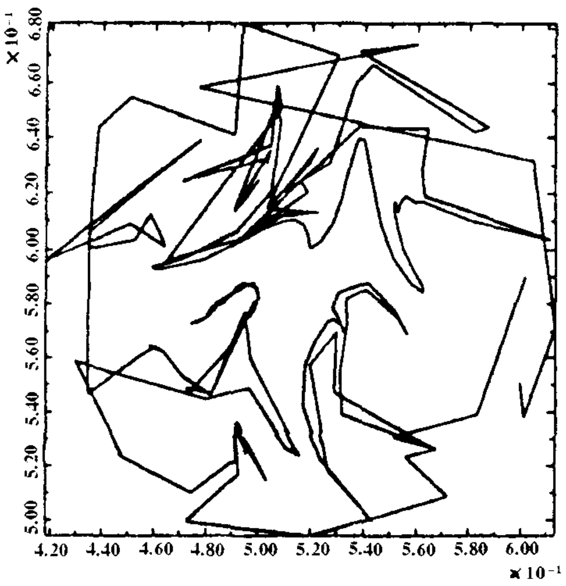

Figure 4

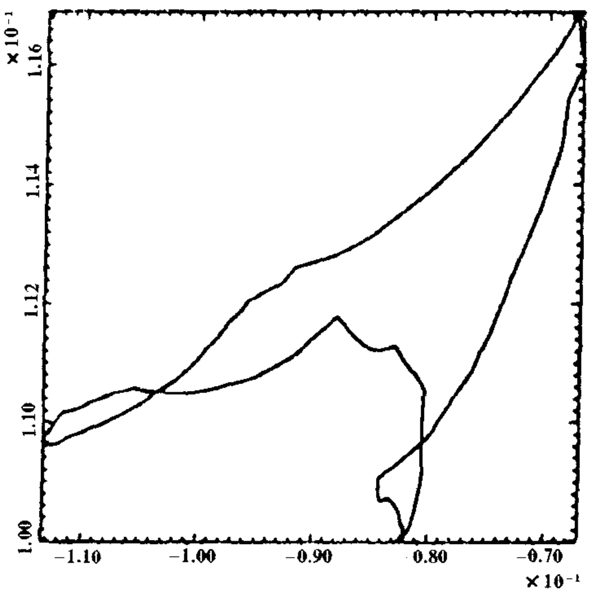

Figure 5

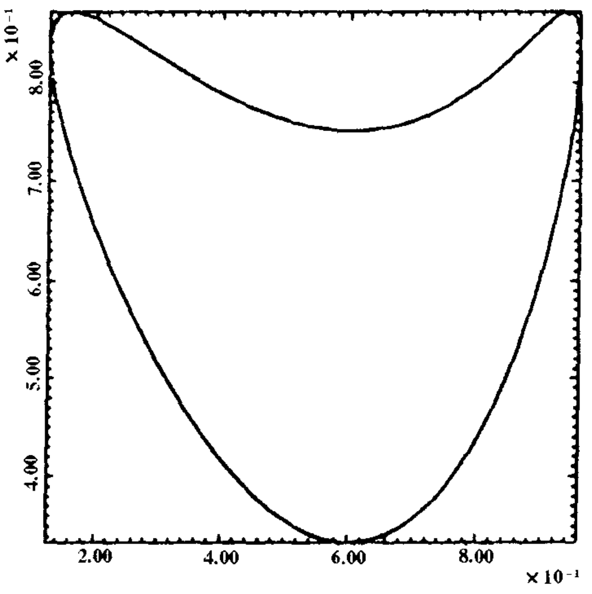

Figure 6

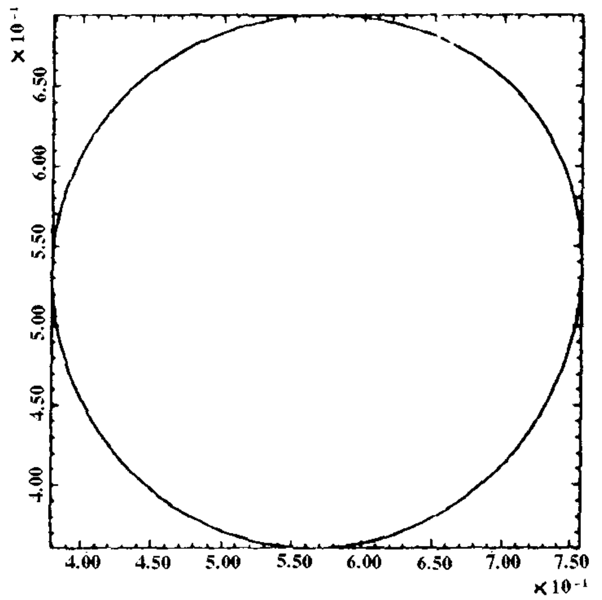

Figure 7

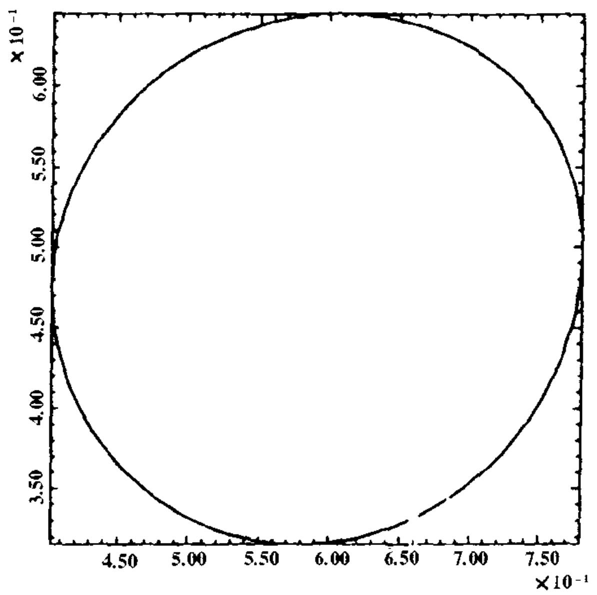

Figure 8

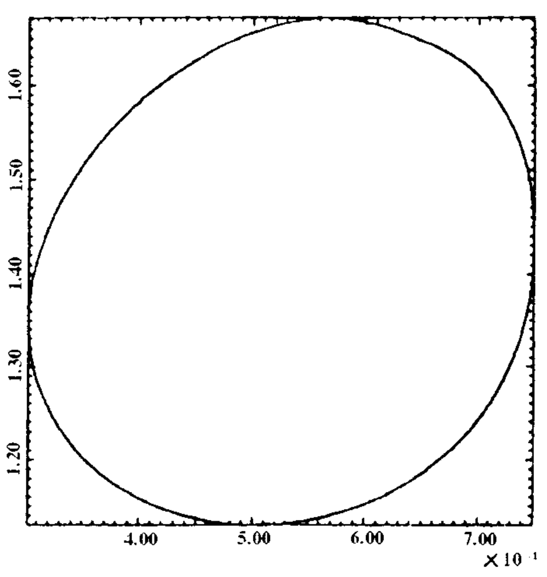

Figure 9

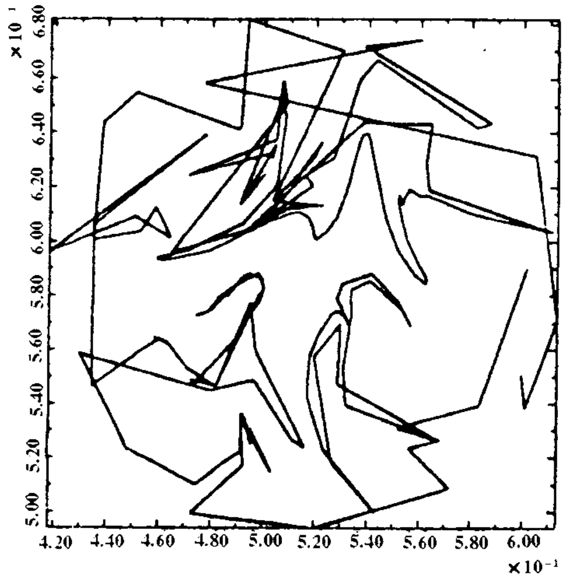

Figure 10

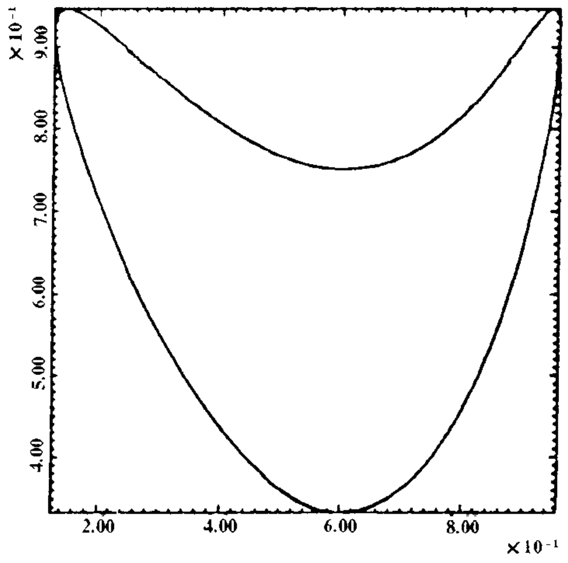

Figure 11

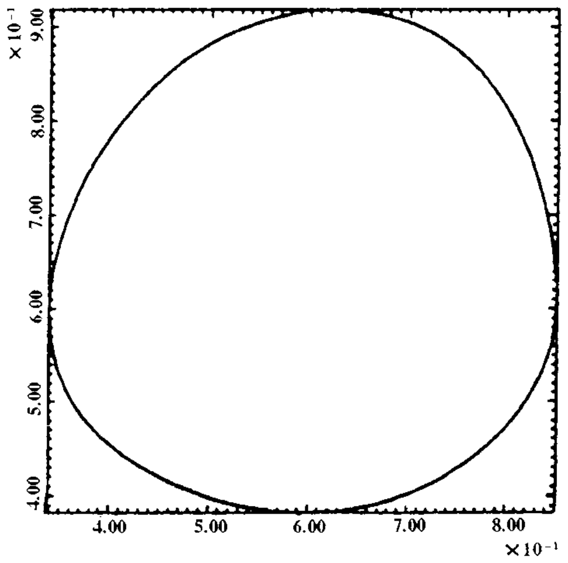

Figure 12

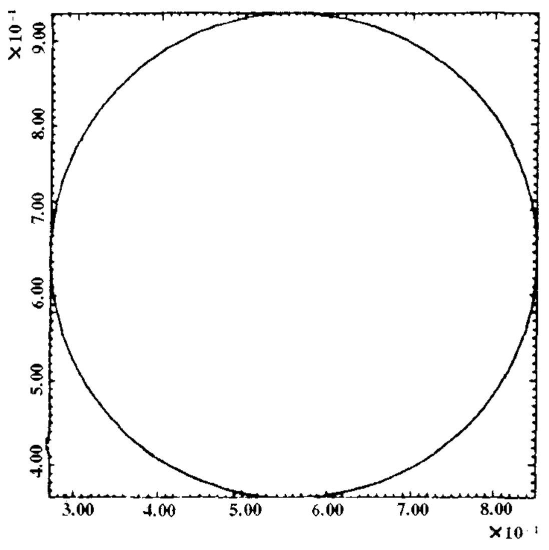

Figure 13

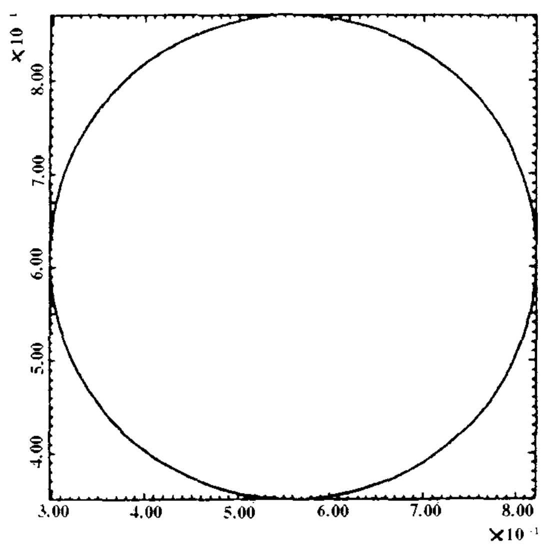

Figure 14

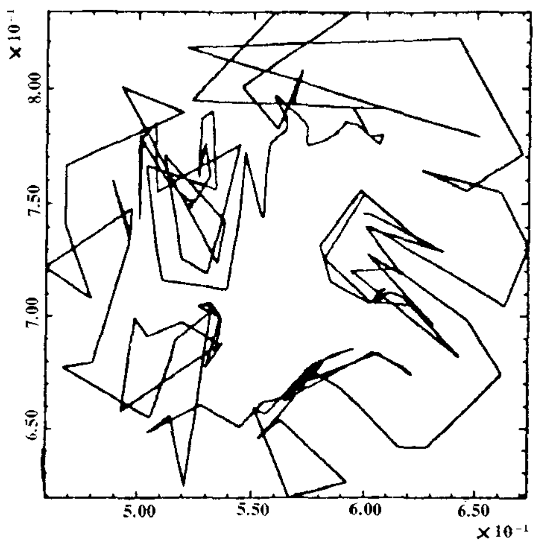

Figure 15

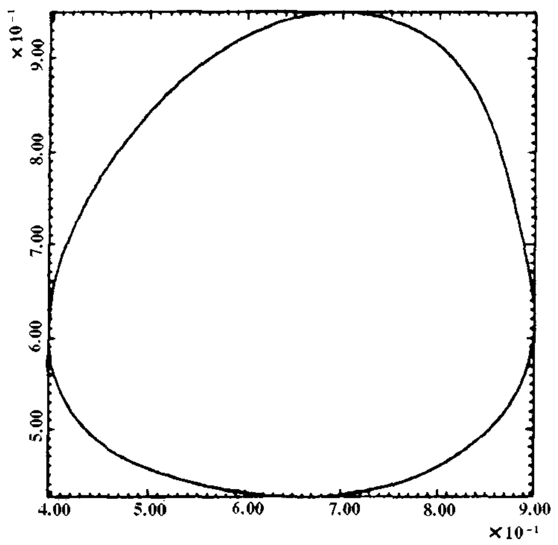

Figure 16