JOURNAL OF COMPUTATIONAL PHYSICS 79, 50–69 (1988)

A Fast Numerical Method for Solving the Three-Dimensional Stokes' Equations in the Presence of Suspended Particles

AARON L. FOGELSON

Department of Mathematics,
University of Utah, Salt Lake City, Utah 84112

AND

CHARLES S. PESKIN

Courant Institute of Mathematical Sciences,
New York University, 251 Mercer Street, New York, New York 10012

Received March 6, 1987; revised December 14, 1987

A new fast numerical method for solving the three-dimensional Stokes' equations in the presence of suspended particles is presented. The fluid dynamics equations are solved on a lattice. A particle is represented by a set of points each of which moves at the local fluid velocity and is not constrained to lie on the lattice. These points are coupled by forces which resist deformation of the particle. These forces contribute to the force density in the Stokes' equations. As a result, a single set of fluid dynamics equations holds at all points of the domain and there are no internal boundaries. Particle size, shape, and deformability may be prescribed. Computational work increases only linearly with the number of particles, so large numbers (500–1000) of particles may be studied efficiently. The numerical method involves implicit calculation of the particle forces by minimizing an energy function and solution of a finite-difference approximation to the Stokes' equations using the Fourier–Toeplitz method. The numerical method has been implemented to run on all CRAY computers: the implementation exploits the CRAY's vectorized arithmetic, and on machines with insufficient central memory, it performs efficient disk I/O while storing most of the data on disk. Applications of the method to sedimentation of one-, two-, and many-particle systems are described. Trajectories and settling speeds for two-particle sedimentation, and settling speed for multiparticle sedimentation from initial distributions on a cubic lattice or at random give good quantitative agreement with existing theories.

1. INTRODUCTION

We present a new fast numerical method for solving the three-dimensional Stokes' equations in the presence of discrete suspended elastic particles (rigid particles are modelled as slightly deformable). This method is an extension of the two-dimensional method for studying flows with immersed boundaries introduced

0021-9991/88 $3.00

by Peskin [12] for studying blood flow in the heart, and previously applied by Fogelson [5] in an investigation of platelet aggregation during blood clotting. The salient feature of this immersed-boundary method is the replacement of material boundaries by suitable contributions to a force density term in the fluid dynamics equations. A single set of fluid dynamics equations then holds in the entire domain and there are no internal boundary conditions. In the present context, this technique permits the study of particles of different sizes and shapes as well as particles that deform. Also, since each particle influences the fluid motion, and hence the motion of other particles, only through its contribution to the fluid force density, and since this contribution is determined independently for each particle, the computational work associated with the particles increases only linearly with the number of particles. Hence, the dynamics of large multiparticle systems, perhaps involving as many as 1000 particles, may be investigated.

The numerical method is designed to study a model fluid–particle system in which each particle is represented by a finite number of fluid points. These points are joined together by a prescribed set of elastic links which generate internal particle forces whose function is to cause the nearby fluid to move as a single unit. (This representation of an elastic particle is motivated by a derivation of the equations of elasticity obtained by considering the continuum limit of ensembles of points connected by spring forces.) The internal particle forces and the applied external forces determine the overall fluid motion. Since the points which make up the particles are fluid points, they move at the local fluid velocity. We note that the use of a distribution of points to represent each particle permits us to study particle rotation as well as translation.

The motion of the fluid is described by the inhomogeneous Stokes' equations. These equations are approximated, using finite differences, at points of a computational lattice, and the resulting discrete Poisson equations are solved by means of the Fourier–Toeplitz method [4]. The motion of the points which make up the particles is tracked independently of the lattice using a simple Euler scheme. *Explicit* evaluation of the internal particle forces produces violent instabilities in the motion of the particles. For this reason, these forces are calculated *implicitly* by minimizing a nonlinear energy function ϕ. The minimization is accomplished using Gill and Murray's modified Newton's algorithm [7, 8] which can handle the indefinite Hessian matrices of ϕ that are frequently encountered. The numerical method has been implemented for use on CRAY computers; the implementation exploits the CRAY's ability to do vector arithmetic, and, on machines with insufficient central memory capacity, it uses asynchronous I/O (input/output) routines to perform efficient disk I/O while storing most of the data on the disk (see [6] for details).

We present promising preliminary results from our application of the numerical method to sedimentation problems. We expect that the method will be a powerful tool for studying many types of suspension flows. We plan, in particular, to incorporate the method into our ongoing study of platelet aggregation in order to extend that study to three dimensions.

2. Model System

We study the interaction of a viscous fluid with a collection of discrete model particles inside a three-dimensional cube B of side L. We assume that inertia is negligible and we describe the fluid motion by the Stokes' equations,

$$0 = -\nabla p + \mu \Delta \mathbf{u} + \mathbf{f}(\mathbf{x}, t) \tag{1}$$

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

Here, $\mathbf{u}$ is the fluid velocity, p the pressure, and μ the viscosity. The force density $\mathbf{f}(\mathbf{x}, t)$ is constructed to have period L in each of the coordinate directions. We seek a solution $\mathbf{u}$, p which is similarly periodic. The Stokes' equations hold for *all* points $\mathbf{x}$ in the cube B.

Each model particle, say particle k, is constructed from a finite collection of points $\mathbf{x}_{kl}$. Each such point is assumed to have no inertial mass and to move at a velocity which is a weighted average of the fluid velocity in the immediate neighborhood:

$$\frac{d\mathbf{x}_{kl}}{dt} = \int_B \mathbf{u}(\mathbf{x}, t)\, \delta_s(\mathbf{x} - \mathbf{x}_{kl}(t))\, d\mathbf{x}. \tag{3}$$

In this formula, the function $\delta_s(\mathbf{x})$ is a smooth function of integral 1 with support in a sphere of radius s. (Here and in Eq. (5), if a point in the support of δ_s is outside the basic periodic box B, its periodic image inside B should be used instead.) The reason for using this local average, instead of just evaluating $\mathbf{u}$ at $\mathbf{x}_{kl}$, will appear below.

The points $\{\mathbf{x}_{kl}\}$ which comprise the kth particle are linked to one another by a prescribed set of elastic links. For example, an octahedral particle can be constructed from seven points, six at the vertices and one at the center, with each vertex joined by elastic links to its four neighboring vertices and by a similar link to the center point. The links generate internal particle forces which act, through the mediation of the fluid, to resist deformation of the particles. It may be helpful to think of the link forces as approximations to the distribution of forces which a real particle exerts on the adjacent fluid. For a rigid particle these forces are just those necessary to make the motion that of a rigid body. In this paper, we consider only the limiting case of very stiff elastic links in order to approximate rigid particles as slightly deformable. For simplicity, we assume that each link acts as a linear spring. Thus, the resultant of the link forces which act on point $\mathbf{x}_{kl}$ is given by

$$\mathbf{f}_{kl}^{\text{int}} = \sum_{m \in L_l} S_{lm}(\|\mathbf{x}_{kl} - \mathbf{x}_{km}\| - r_{lm}) \frac{\mathbf{x}_{km} - \mathbf{x}_{kl}}{\|\mathbf{x}_{km} - \mathbf{x}_{kl}\|}. \tag{4}$$

Here, L_l is the set of indices of the points linked to the point whose index is l, S_{lm} is the stiffness of the link joining points $\mathbf{x}_{kl}$ and $\mathbf{x}_{km}$, and r_{lm} is the resting length of this link. We assume throughout this paper that all of the particles have the same

number of points (M) and the same structure of links and note that this structure is specified by the sets L_l, $l=1, ..., M$. We note also that the sets L_l are defined once and do not change during the course of the calculations. (Generalization to particles constructed from different numbers of points or with different configurations of links is immediate.)

In addition to the internal link forces, application-dependent external forces, such as gravitational forces, as well as interparticle forces, such as electrical forces, may act on the points of each particle. Both the internal and external forces influence particle motion only through the mediation of the fluid. They do this by contributing to the force density $\mathbf{f}(\mathbf{x}, t)$ which appears in the Stokes' equations. More precisely, if we assume that there are N particles each constructed from M points $\mathbf{x}_{kl}$, then we define the force density by

$$\mathbf{f}(\mathbf{x}, t) = \sum_{k=1}^{N} \sum_{l=1}^{M} (\mathbf{f}_{kl}^{\text{int}} + \mathbf{f}_{kl}^{\text{ext}})\, \delta_s(\mathbf{x} - \mathbf{x}_{kl}(t)) - \mathbf{c}. \tag{5}$$

Here, $\mathbf{f}_{kl}^{\text{ext}}$ is the applied external force on point $\mathbf{x}_{kl}$, $\mathbf{f}_{kl}^{\text{int}}$ is the resultant of the spring forces of the links joining $\mathbf{x}_{kl}$ to other points in the kth particle, and $\delta_s(\mathbf{x})$ is the same weight function which is defined in Eq. (3). Thus, each force $\mathbf{f}_{kl}^{\text{ext}}$ or $\mathbf{f}_{kl}^{\text{int}}$ is "felt" by the fluid over a small volume surrounding the point $\mathbf{x}_{kl}$. This gives $\mathbf{x}_{kl}$ an effective radius of order s (see Fig. 1). We emphasize that the force density $\mathbf{f}$ is the *only* way that the fluid feels the presence of the particles. The vector $\mathbf{c}$ in Eq. (5) is a constant vector chosen so that

$$\int_B \mathbf{f}(\mathbf{x}, t)\, d\mathbf{x} = 0. \tag{6}$$

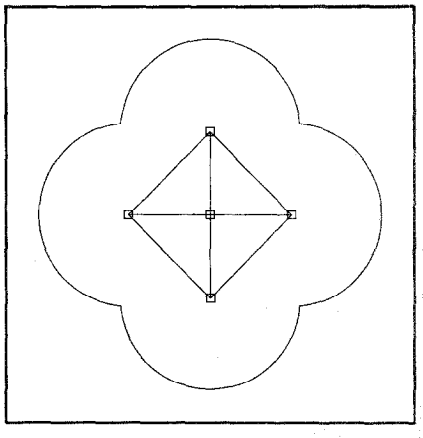

FIG. 1. Cross section of an octahedral particle. The boxes indicate points $\mathbf{x}_{kl}$ and are joined by line segments representing elastic links. The circular arcs enclose the region in which the particle makes a contribution to the force density $\mathbf{f}(\mathbf{x}, t)$. Figures reprinted with the permission of the Society for Industrial and Applied Mathematics from *Advances in Multiphase Flow and Related Problems*, G. Papanicolaou, Ed.

This is a compatibility condition for the Stokes' equations in the periodic box B, as may be seen by integrating Eq. (1) over the cube B and then invoking the periodicity of $\mathbf{u}$ and p. We remark that the compatibility condition $\int_B \mathbf{f}\, d\mathbf{x} = 0$ is automatically satisfied by $\mathbf{f}^{(\text{int})}$. This follows from the translation-invariance of the energy function from which $\mathbf{f}^{(\text{int})}$ is derived. (See below.)

The Stokes' equations may alternatively be written in the form

$$\nabla \cdot (\nabla p) = \nabla \cdot \mathbf{f} \tag{7}$$

$$\mathbf{w} = \mathbf{f} - \nabla p \tag{8}$$

$$\Delta \mathbf{u} = -(1/\mu)\mathbf{w}. \tag{9}$$

Here, $\mathbf{w}$ is the projection of the force density onto the space of divergence-free vector fields. In this formulation (which exploits the simplicity of the periodic domain) the pressure is computed first by solving one Poisson equation and then the velocity is computed by solving a second Poisson equation. Furthermore, in Eq. (9), the components of the velocity $\mathbf{u}$ are uncoupled from each other.

Associated with Eqs. (7) and (9) are the compatibility conditions

$$\int_B \nabla \cdot \mathbf{f}\, d\mathbf{x} = 0 \tag{10}$$

and

$$\int_B \mathbf{w}\, dx = 0, \tag{11}$$

respectively. The divergence theorem and the periodicity of $\mathbf{f}$ imply that Eq. (10) is satisfied, while the compatibility condition on $\mathbf{f}$ (see Eq. (6)) and the periodicity of p ensure that Eq. (11) holds.

To summarize our discussion thus far, we model physical particles by constructing each particle out of a set of points $\{\mathbf{x}_{kl}\}$. These points are subject to external forces as well as to internal forces designed to resist changes in the particle's size and shape. These forces all contribute to the force density $\mathbf{f}$ as described in Eq. (5); $\mathbf{f}$ determines the fluid motion by Eqs. (7)–(9); and the points $\{\mathbf{x}_{kl}\}$ move at the local fluid velocity as described by Eq. (3). We repeat that the main reason for representing particles by collections of localized forces is the relative simplicity of the resulting equations; a single set of fluid dynamics equations is valid everywhere in the domain, and there are no internal boundary conditions.

We conclude this section with a discussion of the energy flow in the model system defined by the foregoing equations. In particular, this discussion explains why it is important to use the same weight function δ_s in Eqs. (3) and (5).

We assume here that the *internal* forces are negative gradients of a translation-invariant function, the elastic internal energy. That is,

$$\mathbf{f}_{kl}^{\text{int}} = -\frac{\partial E}{\partial \mathbf{x}_{kl}}, \tag{12}$$

where $E(\dots \mathbf{x}_{kl} \dots)$ has the property

$$E(\mathbf{x}_{11}+\mathbf{a}, \dots, \mathbf{x}_{NM}+\mathbf{a}) = E(\mathbf{x}_{11}, \dots, \mathbf{x}_{NM}) \tag{13}$$

for every constant displacement $\mathbf{a}$. (In Eq. (12), the expression $\partial/\partial \mathbf{x}_{kl}$ means the gradient with respect to $\mathbf{x}_{kl}$.) It follows directly from Eq. (13) and the arbitrariness of $\mathbf{a}$ that

$$\sum_{kl} \mathbf{f}_{kl}^{\text{int}} = 0. \tag{14}$$

An example of such a function E is given by the elastic energy function

$$E = \sum_{k=1}^{N} \sum_{l=1}^{M} \sum_{m \in L_l} \frac{1}{4} S_{lm} (\|\mathbf{x}_{kl} - \mathbf{x}_{km}\| - r_{lm})^2. \tag{15}$$

(Note that each link appears twice in the sum, hence the factor $\frac{1}{4}$ instead of $\frac{1}{2}$). This function is obviously translation-invariant, and its negative gradient is the system of internal forces given by Eq. (4).

We now consider the rate of change of the elastic internal energy as the points $\mathbf{x}_{kl}$ move according to Eq. (3). According to the chain rule

$$\begin{aligned} \frac{dE}{dt} &= \sum_{kl} \frac{\partial E}{\partial \mathbf{x}_{kl}} \cdot \frac{d\mathbf{x}_{kl}}{dt} \\ &= -\sum_{kl} \mathbf{f}_{kl}^{\text{int}} \cdot \int_B \mathbf{u}(\mathbf{x}, t)\, \delta_s(\mathbf{x} - \mathbf{x}_{kl})\, d\mathbf{x} \\ &= -\int_B \mathbf{u}(\mathbf{x}, t) \cdot \left(\sum_{kl} \mathbf{f}_{kl}^{\text{int}}\, \delta_s(\mathbf{x} - \mathbf{x}_{kl}) \right) d\mathbf{x} \end{aligned} \tag{16}$$

Next, we rewrite Eq. (5) in the form

$$\mathbf{f}(\mathbf{x}, t) = \mathbf{f}^{\text{int}}(\mathbf{x}, t) + \mathbf{f}^{\text{ext}}(\mathbf{x}, t), \tag{17}$$

where

$$\mathbf{f}^{\text{int}}(\mathbf{x}, t) = \sum_{kl} \mathbf{f}_{kl}^{\text{int}}\, \delta_s(\mathbf{x} - \mathbf{x}_{kl}) \tag{18}$$

$$\mathbf{f}^{\text{ext}}(\mathbf{x}, t) = \sum_{kl} \mathbf{f}_{kl}^{\text{ext}}\, \delta_s(\mathbf{x} - \mathbf{x}_{kl}) - \mathbf{c}. \tag{19}$$

Note that the constant $\mathbf{c}$, which is chosen to enforce the compatibility condition $\int_B \mathbf{f}\, d\mathbf{x} = 0$, is only associated with the external forces, since $\int_B \mathbf{f}^{\text{int}}\, d\mathbf{x} = 0$ follows directly from the translation invariance of E.

Because we have used the same function δ_s in Eqs. (3) and (5), we may now substitute Eq. (18) into (16) to obtain

$$\frac{dE}{dt} = -\int_B \mathbf{u}(\mathbf{x}, t) \cdot \mathbf{f}^{\text{int}}(\mathbf{x}, t)\, d\mathbf{x}$$

$$= -\int_B \mathbf{u}(\mathbf{x}, t) \cdot \mathbf{f}(\mathbf{x}, t)\, dx + \dot{W}^{\text{ext}}, \tag{20}$$

where

$$\dot{W}^{\text{ext}} = \int_B \mathbf{u}(\mathbf{x}, t) \cdot \mathbf{f}^{\text{ext}}(\mathbf{x}, t)\, d\mathbf{x} \tag{21}$$

is the rate at which the external forces do work in the system.

It remains to evaluate $\int_B \mathbf{u} \cdot \mathbf{f}\, d\mathbf{x}$. This is done by applying $\int_B \mathbf{u} \cdot (\ )\, d\mathbf{x}$ to all terms in the Stokes' equations. The result is

$$0 = -\int_B \mathbf{u} \cdot \nabla p\, d\mathbf{x} + \mu \int_B \mathbf{u} \cdot \Delta \mathbf{u}\, d\mathbf{x} + \int_B \mathbf{u} \cdot \mathbf{f}\, d\mathbf{x}. \tag{22}$$

Integrating by parts in the periodic domain B, we find

$$-\int_B \mathbf{u} \cdot \nabla p\, d\mathbf{x} = \int_B (\nabla \cdot \mathbf{u})\, p\, d\mathbf{x} = 0, \tag{23}$$

since $\nabla \cdot \mathbf{u} = 0$, and

$$\mu \int_B \mathbf{u} \cdot \Delta \mathbf{u}\, d\mathbf{x} = \mu \int_B \sum_{i,j=1}^{3} u_i \frac{\partial}{\partial x_j} \frac{\partial u_i}{\partial x_j}\, d\mathbf{x}$$

$$= \mu \int_B \sum_{i,j=1}^{3} u_i \frac{\partial}{\partial x_j} \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right) d\mathbf{x}$$

$$= -\mu \int_B \sum_{i,j=1}^{3} \frac{\partial u_i}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) d\mathbf{x}$$

$$= -\frac{\mu}{2} \int_B \sum_{i,j=1}^{3} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)^2 d\mathbf{x} = -\dot{H}, \tag{24}$$

where $\dot{H}$ is the rate at which heat is generated by viscous dissipation [1, p. 153]. (Note that $\dot{H} \geqslant 0$.) Thus, Eq. (22) implies

$$\int_B \mathbf{u} \cdot \mathbf{f}\, d\mathbf{x} = \dot{H}. \tag{25}$$

Substituting this result into Eq. (20), we obtain

$$\frac{dE}{dt} + \dot{H} = \dot{W}^{\text{ext}} \tag{26}$$

which is the appropriate law of conservation of energy for this massless system. In case $\dot{W}^{\text{ext}} = 0$, we have, since $\dot{H} \geqslant 0$, the inequality $dE/dt \leqslant 0$. This shows that the system cannot cycle in the absence of externally applied forces.

3. Numerical Method: Introduction

We now describe the numerical scheme used to study this model system. Because of the periodicity imposed on our functions, we may restrict our attention to any one period. Thus, we take as our computational domain the set of points $B = \{(x_1, x_2, x_3)\colon 0 \leqslant x_i < L,\ i = 1, 2, 3\}$, where we identify opposite sides of the cube, e.g., $x_1 = 0$ and $x_1 = L$. We place a uniform lattice of size h ($h = L/N_h$, $N_h =$ a power of 2) over the cube and divide time into timesteps of size Δt. The Eulerian variables $\mathbf{u}$, p and $\mathbf{f}$ are defined at points of the lattice and we use the usual notation $\mathbf{u}^n_{j_1 j_2 j_3} = \mathbf{u}(j_1 h, j_2 h, j_3 h, n\,\Delta t)$, etc. The particle points are not constrained to coincide with lattice points and we use the notation $\mathbf{x}^n_{kl} = \mathbf{x}_{kl}(n\,\Delta t)$ to denote the position of the lth point of particle k at time $n\,\Delta t$.

In order to move the points $\mathbf{x}_{kl}$ according to Eq. (3), the weighted-average of velocities in the neighborhood of $\mathbf{x}_{kl}$ which appears in that equation is replaced by a similar average of the velocities at lattice points near $\mathbf{x}_{kl}$. To accomplish this, we introduce a discrete approximation $D_{j_1 j_2 j_3}(\mathbf{x})$ to the function $\delta_s(\mathbf{x})$. ($D_{j_1 j_2 j_3}(\mathbf{x})$ is a three-dimensional version of Peskin's discrete approximation to the delta-function [12].) Let

$$d(r) = \begin{cases} \dfrac{1}{4h}\left(1 + \cos\left(\dfrac{\pi r}{2h}\right)\right), & \text{if} \quad |r| \leqslant 2h \\ 0, & \text{if} \quad |r| > 2h. \end{cases}$$

Then, $D_{j_1 j_2 j_3}(\mathbf{x})$ is defined by

$$D_{j_1 j_2 j_3}(\mathbf{x}) = d(x_1 - j_1 h)\, d(x_2 - j_2 h)\, d(x_3 - j_3 h).$$

We note that the function $D_{j_1 j_2 j_3}(\mathbf{x})$ has support consisting of those lattice points within a cube of side $4h$ centered on $\mathbf{x}$; $D_{j_1 j_2 j_3}(\mathbf{x})$ is of order h^{-3} at $\mathbf{x}$; and $\sum_{i_1 j_2 j_3} h^3 D_{j_1 j_2 j_3}(\mathbf{x}) = 1$ for any $\mathbf{x}$. For any point $\mathbf{x}$ in our domain, we define the "interpolated velocity"

$$\mathbf{u}^{n+1}(\mathbf{x}) = \sum_{j_1 j_2 j_3} \mathbf{u}^{n+1}_{j_1 j_2 j_3} D_{j_1 j_2 j_3}(\mathbf{x})\, h^3. \tag{27}$$

(As with the continuous function $\delta_s(\mathbf{x})$, if (j_1, j_2, j_3) in the support of $D_{j_1 j_2 j_3}(\mathbf{x})$ is outside the basic periodic box, its periodic image within the box is used instead.) We take the velocity of the point $\mathbf{x}_{kl}^n$ to be the value $\mathbf{u}^{n+1}(\mathbf{x}_{kl}^n)$ given by Eq. (27) with $\mathbf{x} = \mathbf{x}_{kl}^n$.

We also use $D_{j_1 j_2 j_3}(\mathbf{x})$ to approximate $\delta_s(\mathbf{x})$ in the definition of $f(\mathbf{x}, t)$ given by Eq. (5). Thus, we define values of the force density on the lattice by

$$\mathbf{f}_{j_1 j_2 j_3} = \sum_{k=1}^{N} \sum_{M=1}^{M} (\mathbf{f}_{kl}^{\text{int}} + \mathbf{f}_{kl}^{\text{ext}})\, D_{j_1 j_2 j_3}(\mathbf{x}_{kl}^n) - \mathbf{c}_h. \tag{28}$$

The superscripts denoting time are omitted in Eq. (28) for reasons that will be made clear below. The constant vector $\mathbf{c}_h$ is defined so that a discrete version of the compatibility condition Eq. (6) on $\mathbf{f}$ holds on the lattice, i.e.,

$$\sum_{j_1 j_2 j_3} \mathbf{f}_{j_1 j_2 j_3} = 0. \tag{29}$$

The internal particle forces are constructed in a way that ensures that

$$\sum_{k=1}^{N} \sum_{l=1}^{M} \mathbf{f}_{kl}^{\text{int}} = 0,$$

so the vector $\mathbf{c}_h$ is determined only by the external forces:

$$\mathbf{c}_h = -\frac{1}{(N_h)^3} \sum_{k-1}^{N} \sum_{l-1}^{M} \mathbf{f}_{kl}^{\text{ext}},$$

i.e., by the lattice average of the applied forces. The vector $\mathbf{c}_h$ is typically much smaller than $\mathbf{f}_{kl}^{\text{ext}}$ and $\mathbf{f}_{kl}^{\text{int}}$.

We note that s (as used in the definition of $\delta_s(\mathbf{x})$) is a physical parameter and h is a numerical parameter. Hence, in principle, the ratio $s/h \to \infty$ as $h \to 0$ and the support of $D_{j_1 j_2 j_3}(\mathbf{x})$ should grow to reflect this. In practice, we use a finite mesh size of order s.

The operations performed to advance the model system from time $(n\,\Delta t)$ to time $(n+1)\Delta t$ are as follows: Values $\mathbf{f}_{kl}^{\text{int},*}$ for the internal spring forces are calculated using an approximately implicit scheme (see below) to avoid numerical instabilities in the motion of the points $\mathbf{x}_{kl}$. These values are used in the right-hand side of Eq. (28), along with the prescribed external forces, to define the force density $\mathbf{f}_{j_1 j_2 j_3}^{n+1}$ at points of the lattice. Using these values of the force density, a new velocity field $\mathbf{u}_{j_1 j_2 j_3}^{n+1}$ is determined. Finally, the velocities are interpolated to the locations $\mathbf{x}_{kl}^n$ using Eq. (27), and the points $\mathbf{x}_{kl}$ are moved to new positions

$$\mathbf{x}_{kl}^{n+1} = \mathbf{x}_{kl}^n + \Delta t\, \mathbf{u}^{n+1}(\mathbf{x}_{kl}^n). \tag{30}$$

A description of the implicit force calculation is given below. The solution of the Stokes' equations is also described below; a more detailed description of this portion of the numerical method is given in [6].

4. Internal Particle Forces

The internal spring forces $\mathbf{f}_{kl}^{\text{int}}$ are calculated implicitly so as to avoid inducing instabilities in the motions of the points $\mathbf{x}_{kl}$. An approximately implicit scheme, similar to one introduced by Peskin [12] (and previously used in [5]), proves sufficient to damp out these instabilities. The internal forces which act on the points $\mathbf{x}_{kl}$ of the kth particle are functions only of the points in *that* particle and of the set of links connecting these points. It follows from this and from the approximation introduced below that the implicit calculation of all the internal forces uncouples into the separate determination of each particle's internal forces. The calculation of the internal forces for particle k proceeds as follows: We define points $\mathbf{x}_{k1}^*, ..., \mathbf{x}_{kM}^*$ by the system of equations

$$\mathbf{x}_{kl}^* = \mathbf{x}_{kl}^n + \Delta t\, \lambda \mathbf{f}_{kl}^{\text{int}}(\mathbf{x}_{k1}^*, ..., \mathbf{x}_{kM}^*), \tag{31}$$

for $l = 1, ..., M$, where M is the number of points which make up the kth particle. The parameter λ in Eq. (31) is the magnitude of the velocity induced by a localized unit force and was estimated numerically. The approximation built into Eq. (31) is that the fluid velocity at $\mathbf{x}_{kl}$ is proportional to the internal force $\mathbf{f}_{kl}^{\text{int}}$ and is independent of the forces $\mathbf{f}_{kl}^{\text{int}}$, $m \neq l$. Insofar as this approximation is valid, $\mathbf{x}_{k1}^*, ..., \mathbf{x}_{kM}^*$ is the configuration of points making up particle k at the end of the timestep. (We are ignoring here the motion induced by the external forces $\mathbf{f}_{kl}^{\text{ext}}$.) Equation (31) is *implicit* because the forces are calculated from the (unknown) configuration $\mathbf{x}_{k1}^*, ..., \mathbf{x}_{kM}^*$ rather than from the given configuration $\mathbf{x}_{k1}^n, ..., \mathbf{x}_{km}^n$. Using the solution of Eq. (31), we define

$$\mathbf{f}_{kl}^{\text{int},*} = \mathbf{f}_{kl}^{\text{int}}(\mathbf{x}_{k1}^*, ..., \mathbf{x}_{kM}^*). \tag{32}$$

We use these internal forces, along with the prescribed external forces, in Eq. (28) to define the force density for the Stokes' equations. The new velocity field which is in equilibrium with *all* of the forces is calculated and the points $\mathbf{x}_{kl}$ are moved according to Eq. (30). We emphasize that the points $\mathbf{x}_{kl}^*$ are used only to calculate $\mathbf{f}_{kl}^{\text{int},*}$.

In order to describe the means of solving Eq. (31), we introduce the notation $\mathbf{X} = (\mathbf{x}_{k1}^*, ..., \mathbf{x}_{kM}^*)$; $\mathbf{X}^0 = (\mathbf{x}_{k1}^n, ..., \mathbf{x}_{kM}^n)$; and $\mathbf{F} = (\mathbf{f}_{k1}^{\text{int}}, ..., \mathbf{f}_{kM}^{\text{int}})$ and note that Eq. (31) can be expressed

$$0 = \mathbf{X} - \mathbf{X}^0 - \lambda\, \Delta t\, \mathbf{F}(\mathbf{X}). \tag{33}$$

Although the Jacobian matrix of the (nonlinear) function $\mathbf{X} - \lambda\, \Delta t\, \mathbf{F}(\mathbf{X})$ is symmetric, it is often indefinite in our calculations, so the multidimensional Newton–Raphson method is inappropriate for solving Eq. (33). Instead, we introduce

a differentiable "energy function" $E(\mathbf{E})$ such that the following conditions are satisfied:

(i) $E(\mathbf{X}) \geqslant 0$

(ii) $E(\mathbf{x}) \to \infty$ as $\|x\| \to \infty$

(iii) $\mathbf{F}(\mathbf{x}) = -\text{grad}\, E(\mathbf{X})$.

(In terms of our original notation, a suitable energy function is $E = \frac{1}{4}\sum_{l=1}^{M}\sum_{m \in L_l} S_{lm}(\|\mathbf{x}_{kl} - \mathbf{x}_{km}\| - r_{lm})^2$. This is the same energy function as that defined in Eq. (15) except that here we consider one particle at a time). A minimum point of the function $\phi(\mathbf{X}) = \frac{1}{2}\|\mathbf{X} - \mathbf{X}^0\|^2 + \lambda\, \Delta t\, E(\mathbf{X})$ is then a solution of Eq. (31). We use Gill and Murray's modified Newton's method (MNM) [7, 8] to seek a minimum point of ϕ. This method can handle the symmetric but indefinite Hessian matrices that we encounter (the Hessian matrix of ϕ is identical to the Jacobian of $\mathbf{X} - \lambda\, \Delta t\, \mathbf{F}(\mathbf{X})$ mentioned before.)

In brief, Gill and Murray's method works as follows: Let $\mathbf{g}$ be the gradient of ϕ and G be the Hessian matrix of ϕ. A sequence $\{\mathbf{x}^{(q)}\}$ of approximations to a minimum point of ϕ is defined iteratively by the equations

$$(G^{(q)} + \Lambda^{(q)})\, \mathbf{p}^{(q)} = -\mathbf{g}^{(q)} \tag{34}$$

$$\mathbf{X}^{(q+1)} = \mathbf{X}^{(q)} + \alpha^{(q)}\mathbf{p}^{(q)}. \tag{35}$$

In Eq. (34), $\Lambda^{(q)}$ is a nonnegative diagonal matrix to be discussed below. The vector $\mathbf{p}^{(q)}$ is called a search direction and $\alpha^{(q)}$ is a positive scalar chosen to achieve a "sufficient decrease" in $\phi^{(q+1)}$ relative to $\phi^{(q)}$. The process of choosing $\alpha^{(q)}$ is called linesearch. We employ a safeguarded-cubic-interpolation linesearch algorithm similar to that described in [9].

The solution of Eq. (34) is facilitated by the symmetric Gaussian decomposition

where $U^{(q)}$ is a unit upper-triangular matrix and $D^{(q)}$ is a diagonal matrix. $\Lambda^{(q)}$ is constructed during the row-by-row factorization in such a way that:

(i) $G^{(q)} + \Lambda^{(q)}$ is positive definite;

(ii) diagonal elements of $D^{(q)}$ are bounded away from zero by a positive constant δ;

(iii) the inequality $|d_{rr}u_{rs}^2| \leqslant \beta^2$ holds for $s > r$, where β^2 is a prescribed constant.

Condition (i) guarantees that $\mathbf{p}^{(q)}$ gives a descent direction for ϕ as can be seen by premultiplying Eq. (34) by $\mathbf{p}^{(q)T}$. Conditions (ii) and (iii) ensure the numerical stability of the factorization and the subsequent forward and back substitution used in solving Eq. (34).

Provided β^2 has been suitably chosen, $\Lambda^{(q)}$ is automatically set to zero if $G^{(q)}$ is

itself sufficiently positive definite in a sense defined by Gill and Murray [7]. The modified Newton's method therefore reduces to Newton's method where $G^{(q)}$ is positive definite (e.g., in the neighborhood of a minimum point of ϕ), and thus it has the locally quadratic rate of convergence exhibited by the Newton's method. In our calculations, convergence is usually achieved in 3 or 4 iterations.

Recall that a separate energy minimization is performed for each model particle. In principle, these separate calculations could be performed in parallel. The CRAY architecture does not support fully parallel computations, but it does support vector operations. The latter allows for substantial speed-up of a sequence of independent calculations provided they have *identical* structure. We obtain a code which is successfully vectorized by the CRAY FORTRAN compiler by replacing appropriate assignment statements in a scalar implementation of the MNM with (innermost) DO-loops which run over the indices of particles of identical structure. Thus, the innermost loop runs over different sets of data (in this case each set of data involves information about one particle) to which the *same* algorithm is applied. We illustrate this conversion process with a simple example: One step of the MNM is the calculation of the maximum magnitude GAMMA of a diagonal element of the Hessian matrix $G^{(q)}$. For a scalar version of the algorithm, this calculation is accomplished by the FORTRAN loop

```
      Dimension G(MSIZE, MSIZE)
        ⋮
      GAMMA = 0·0
      DO 2 J = 1, MSIZE
    2 GAMMA = AMAX1 (GAMMA, ABS(G(J, J))).
```

For a vectorizable implementation of the algorithm, this portion of code is replaced by

```
      Dimension GAMMA (N)
      Dimension G (N, MSIZE, MSIZE)
        ⋮
      DO 1 I = 1, N
    1 GAMMA (I) = 0·0
      DO 2 J = 1, MSIZE
      DO 2 I = 1, N
    2 GAMMA (I) = AMAX1(GAMMA(I), G(I, J, J,))).
```

Systematic use of this technique leads to a code in which almost every innermost DO-loop vectorizes. The only exceptions are a few IF statements which could not be rewritten in this way.

We note that in each of our numerical experiments to date, *all* of the particles had the same structure. In general, with particles of differing structures, the set of particles would be partitioned into classes, each consisting of particles of identical structure, and vectorization would be achieved for each class.

The factorization (34) as described by Gill and Murray requires order $n^3/6$ operations and order n^2 storage locations, where n is the order of the Hessian matrix $\mathbf{G}^{(q)}$. For our work, n is three times the number of points $\{\mathbf{x}_{kl}\}$ used to

construct one particle. Factorization of several Hessian matrices for each particle is usually required in each time step. For octahedral particles, $n = 21$, but for more elaborate particles, Hessian matrices of order 100–200 may result. Such matrices would be sparse, as each point $\mathbf{x}_{kl}$ would be linked to only a few neighboring points. In [5], Fogelson described a means of merging the MNM with the sparse symmetric Gaussian elimination algorithm of the Yale sparse matrix package [3] to reduce substantially the computational work and storage required for the factorization (34) of sparse Hessian matrices. This method could be adapted for use in minimizing the energy functions of particles constructed from many points. We note that the complicated linked-list data structure used by the sparse factorization routine to keep track of the nonzero entries of the Hessian would not interfere with successfull vectorization of the code. This is because the innermost FORTRAN loops run over the indices of particles which have the same configuration of elastic links and for which, therefore, the zero/nonzero structure of the Hessian matrices would be the same.

5. Solution of the Stokes' Equations

Next, we briefly describe the numerical solution of the projection form of the Stokes' equations given in Eqs. (7)–(9). We assume that the force density $\mathbf{f}$ has been defined at *all* lattice points and that it satisfies the discrete compatibility condition Eq. (29). To describe our discretization of Eqs. (7)–(9), we need to introduce the usual centered-, forward-, and backward-difference operators, D_i^0, D_i^+, and D_i^-, respectively, for each of the coordinate directions i. For example, $D_i^0\phi(\mathbf{x}) = (\phi(\mathbf{x}+h\mathbf{e}_i) - \phi(\mathbf{x}-h\mathbf{e}_i))/2h$, where $\mathbf{e}_i$ is the unit vector in the i-coordinate direction and h is the lattice spacing. For a scalar function ψ and a vector function $\mathbf{u} = (u_1, u_2, u_3)$, we define discrete gradient, divergence, and Laplacian operators

$$\mathbf{G}\psi = (D_1^0\psi, D_2^0\psi, D_3^0\psi) \tag{37}$$

$$D \cdot \mathbf{u} = D_1^0 u_1 + D_2^0 u_2 + D_3^0 u_3 \tag{38}$$

$$\mathscr{L}\psi = \sum_{i=1}^{3} D_i^+ D_i^- \psi. \tag{39}$$

Each of these discrete operators is a second-order approximation to the corresponding differential operator. Our discrete approximation to Eqs. (7)–(9) is

$$D \cdot \mathbf{G}p = D \cdot \mathbf{f} \tag{40}$$

$$\mathbf{w} = \mathbf{f} - \mathbf{G}p \tag{41}$$

$$\mathscr{L}\mathbf{u} = -(1/\mu)\mathbf{w}. \tag{42}$$

These equations are meant to hold at *all* points of the computational lattice. Discrete analogs of the compatibility conditions (10) and (11) must be satisfied.

That they are indeed satisfied follows from Eq. (29) and the periodicity of $\mathbf{f}$ and p. We note that the operator $D \cdot G$ involves differences on a staggered grid; i.e., each point of the lattice is coupled to its second neighbors, while the operator $\mathscr{L}$ involves nearest-neighbor coupling. It follows from Eq. (40) and (41) that $D \cdot \mathbf{w} = 0$. Equation (42) and the periodicity of $\mathbf{u}$ then imply that $D \cdot \mathbf{u} = 0$ as well. The discrete divergence of $\mathbf{u}$ would not vanish if the operator $\mathscr{L}$ had been used to approximate the Laplacian in the equation for the pressure.

Equations (40)–(42) constitute a set of four discrete Poisson equations that we solve to determine $\mathbf{u}$ and p. Despite the different grids underlying the discrete Laplacian operators in the pressure and velocity equations, essentially the same method of solution is used in both equations. This is the Fourier–Toeplitz method [4]. We outline its application to the pressure equation. Let $q = D \cdot \mathbf{f}$. Then, we wish to solve

$$(D \cdot \mathbf{G}p)_{j_1 j_2 j_3} = q_{j_1 j_2 j_3} \tag{43}$$

at all points $(j_1 h, j_2 h, j_3 h)$ of the lattice. Let

$$q_{j_1 j_2 j_3} = \frac{1}{N_h^2} \sum_{k_1=0}^{N_h-1} \sum_{k_2=0}^{N_h-1} Q_{j_3}(k_1, k_2) \exp\left(\frac{2\pi i}{N_h}(j_1 k_1 + j_2 k_2)\right) \tag{44}$$

and

$$p_{j_1 j_2 j_3} = \frac{1}{N_h^2} \sum_{k_1=0}^{N_h-1} \sum_{k_2=0}^{N_h-1} P_{j_3}(k_1, k_2) \exp\left(\frac{2\pi i}{N_h}(j_1 k_1 + j_2 k_2)\right). \tag{45}$$

Substitution of these into Eq. (43) yields, for each (k_1, k_2), a periodic tridiagonal system for the Fourier coefficients $P_{j_3}(k_1, k_2)$:

$$-P_{j_3-2}(k_1, k_2) + 2\left\{1 + 2\left(\sin^2\frac{2\pi k_1}{N_h} + \sin^2\frac{2\pi k_2}{N_h}\right)\right\} P_{j_3}(k_1, k_2) - P_{j_3+2}(k_1, k_2)$$
$$= -2hQ_{j_3}(k_1, k_2) \tag{46}$$

for $j_3 = 0, 1, 2, \ldots, N_h - 1$ ($N_h = L/h =$ a power of 2), with subscript arithmetic modulo N_h. Because of the staggered differencing present in Eq. (46), we, in fact, have two $N_h/2 \times N_h/2$ periodic tridiagonal systems for each (k_1, k_2), one for odd j_3 and one for even j_3. The solutions of each such periodic tridiagonal system are found by forming a suitable linear combination of the solutions of two related nonperiodic tridiagonal systems.

The data storage requirements for the fluid dynamics calculation exceed the central memory storage capacity of computers even as large as the CRAY X-MP. For this reason, we have implemented an "out-of-core" version of the fluid-dynamics algorithm, in which, at any particular instant during the calculations, only a small subset of the data resides in central memory; the reminder of the data reside on the disk. Thus, data must be read from the disk to central memory and

vice versa. Such input/output operations are slower than arithmetic computations, so we have implemented the fluid-dynamics algorithm in a way that overlaps steps of the algorithm that use the same data. We should point out that, with very minor programming changes, the numerical method can be (and has been) used on the CRAY-2, which has an enormous central memory, with all of the data residing in central memory.

The fluid dynamics algorithm has been implemented in such a way that all innermost loops vectorize. The numerical solution of the Stokes' equations is discussed at greater length in [6].

6. Computational Work

The work required to carry out the solution of the fluid dynamics equations is order $N_h^3(\log N_h)$, where N_h is the number of lattice points in each coordinate direction. The work required for the particle calculations is proportional to the number of particles. This contrasts with other methods that involve direct particle–particle interactions and for which the work is proportional to the square of the number of particles. The relative efficiency of the present method suggests that it will be especially useful for calculations with large numbers of particles. It is worth remarking that the computational time (CPU time) required by our calculations actually grows less rapidly than it would seem from the above discussion. This is a consequence of the improved efficiency of vectorization as the vector length grows.

7. Results

We present simulations of one-, two-, and many-particle systems that involve the sedimentation of the particles under gravity. For all of these simulations, the external force is given by

$$\mathbf{f}_{kl}^{\text{ext}} = -g\mathbf{e}_3, \tag{47}$$

where $\mathbf{e}_3$ is a unit vector in the up direction and g is constant. The stiffnesses of the internal elastic links are large so that we are approximating rigid particles.

The no flux condition at the surface of a particle certainly is satisfied when a large number of points is used to construct the particle, for these points move at the local fluid velocity and the interpolated velocity field is slowly varying. Figure 2 is from a simulation of the settling of an octahedral particle, a particle which consists of only seven points. The picture depicts the situation after the particle has fallen approximately ten particle diameters and shows the *interpolated* velocity field (as defined by Eq. (27)) near the particle in a vertical plane which bisects the octahedron. The frame of reference is one in which the average of the velocities of the left and right vertices is zero. The vertices of the octahedron are separated by

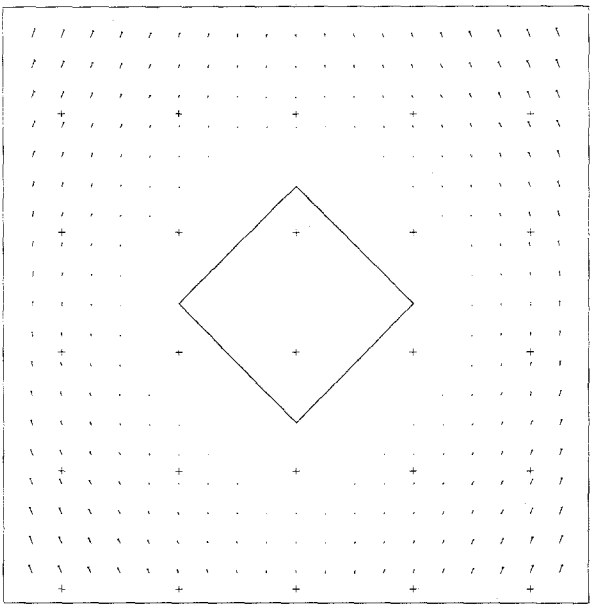

FIG. 2. Vertical section through center of mass of octahedral particle settling under gravity. Arrows show interpolated velocity field in rest frame of particle. Crosses indicate lattice points. Reprinted with permission.

distances that deviate by less than one percent from their initial values. The interpolated velocity field, which is plotted at points separated by $\frac{1}{4}$ of the mesh space used in the fluid dynamics calculations, is nearly zero for the fluid contained within the octahedron and in its immediate vicinity. This calculation shows the effectiveness of using a suitable configuration of point forces (even as few as seven) to cause nearby fluid to move as a rigid body.

Arguments based on symmetry and the linearity of the Stokes' equations can be used to show that two identical rigid spheres which are initially at the same height will settle along parallel vertical lines. We conducted a series of numerical experiments to see how well our method could reproduce this result. Figure 3a shows the settling of two particles each constructed from 21 points distributed at the vertices and center of a dodecahedron. The Stokes' equations were solved on a 64^3 lattice. The two particles fall nearly in parallel; each particle's trajectory deviates from the vertical by an angle of about 0.2°. (The particles rotate in opposite directions as they fall.) We did a similar experiment with two octahedral particles (7 points per particle) and a 64^3 lattice, and another with two "refined octahedra" (19 points per particle) each of which was constructed from an octahedral particle by inserting an additional point at the midpoint of each edge and linking together the new points of each face. This last calculation was performed using a 128^3 lattice. In the Fig. 3b, we plot the distance between the centers of

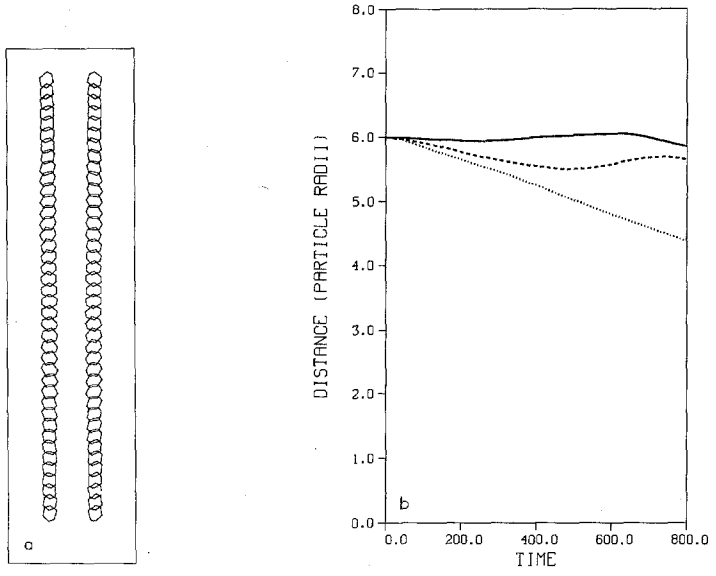

FIG. 3. (a) Sequence of vertical sections through centers of mass of two dodecahedral particles settling under gravity. Vertical extent of panel is one computational period L. (b) Distance between centers of mass versus time for 2 octahedra (dotted curve), 2 dodecahedra (dashed curve), and 2 "refined octahedra" (solid curve) settling under gravity. Reprinted with permission.

mass of the two particles as a function of time for each of these experiments. The theoretical result to which these graphs should be compared is the horizontal line, DISTANCE = 6. We see that as the particles are made "more spherical" through the use of a larger number of points per particle, and also as the computational mesh is refined, the numerical results approach the theoretical result for rigid spheres. (Note, however, that in the "refined octahedra," the additional points are not projected onto the sphere in which the particle is inscribed.) In all of these experiments, the nominal radius of each particle (i.e., the radius of the smallest circumscribing sphere) was 1.0, the initial interparticle distance was 6.0, and the imposed spatial period in the calculation was 64.0.

We also calculated the ratio of the settling speed for two particles to that for a single particle and compared this ratio to that predicted by a calculation using the method of reflections applied to two rigid spheres, as described, for example, in Happel and Brenner [10]. For each of the numerical experiments described above, the ratio was within 9% of that given by the method of reflections calculation; for the "refined octahedra," the ratios differed by less than 3%.

Two identical rigid spheres, which settle from different initial heights, move in parallel, each proceeding along a line that lies between the vertical line through it and the line through the two particles' centers of mass. Figure 4 shows a numerical simulation of this phenomenon. The particles were modeled by dodecahedra and

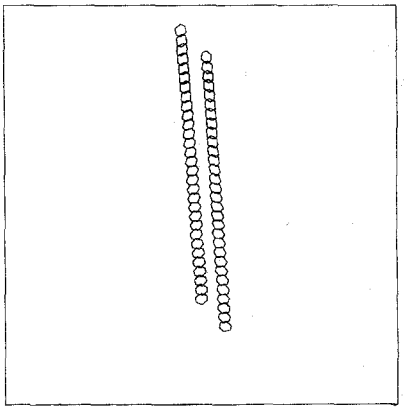

FIG. 4. Sequence of vertical sections through centers of mass of two dodecahedral particles settling under gravity from different initial heights. Reprinted with permission.

the calculation was done using a 64^3 lattice. The angle of fall observed in this simulation was 4.5°, which compares favorably with a prediction of 5.0° based on a calculation using the method of reflections [10].

We next describe two types of multi-particle experiments each of which involved 125 particles. In the first of these experiments, octahedral particles were initially distributed on a cubic lattice which filled the periodic cube B used in our calculations. The observed average settling speed of the 125 particles was compared to a prediction based on Hasimoto's theory for the settling of a periodic cubic array of spheres [11]. For the situation we considered, Hasimoto's theory predicts a settling speed of approximately

$$U_H = \frac{(1.0 - 1.76\beta^{1/3})g}{6\pi\mu a}, \tag{48}$$

where a is the particle radius, g is the gravitational force on each sphere, and $\beta = \frac{4}{3}\pi(a/l)^3$ is the volume fraction for particles spaced a distance l from one another. In order to use Eq. (48) to predict a settling speed for an array of particles, the particle radius must be known. The effective radius of one of our octahedral particles is larger than the radius of the sphere in which its vertices are inscribed, which augments the nominal radius of the octahedron. If we set U_H equal to the settling speed observed in our *single* octahedron experiments, and the interparticle spacing l equal to the imposed period L inherent in our calculations, then we can solve Eq. (48) for an effective radius, a_{eff}, of the octahedral particle. Determining a_{eff} in this way yields a volume fraction $\beta = 0.019$ for our 125-particle experiment. Thus, Hasimoto's theory predicts a settling speed of 0.00339 cm/s for this experiment. This differs by just over 5% from our observed settling speed of 0.00357 cm/s.

Our second multi-particle experiment involved 125 octahedral particles whose random initial positions were chosen from a uniform distribution without overlap. The observed settling speed early in the calculation (while the distribution was still random) was compared to Batchelor's prediction for the settling speed of a random array of spheres [1]. To first order, this prediction is

$$U_B = (1.0 - 6.55\beta) U_1, \tag{49}$$

where U_1 is the single particle settling speed. With volume fraction $\beta = 0.019$, as in the previous experiment, Eq. (49) predicts a settling speeds which differs by 4% from the particle-averaged settling speed, U_{125}, we observed. Since β is small in this experiment, it may be more useful to compare the "correction coefficient" $(U_{125} - U_1)/(-\beta U_1) = 8.48$, based on our computations; with Batchelor's coefficient of 6.55. Of course, many such experiments, with different random choices for the patches' initial positions must be performed in order to make meaningful comparisions between Batchelor's theory and our computational results. Yet, even from these preliminary experiments, we see that the correction in the random case is of the right order of magnitude; and that the numerical method can clearly distinguish between the case of a periodic array of particles, for which the settling speed should be approximately 40% less than for a single particle (for $\beta = 0.019$), and a random array for which the slowdown should be approximately 12%.

The above calculations were performed on CRAY-1 and CRAY-2 computers at the Minnesota Supercomputing Center. For the calculations on the CRAY-1, use of disk storage was essential. The central memory capacity of the CRAY-2 is sufficiently large that all of our data could reside in central memory even when we used a $(128)^3$ lattice. An indication of the CPU time required for the fluid and particle portions of our calculations comes from two runs of 100 timesteps, each of which used a $(64)^3$ grid. In the first of these runs, *no* particles were present, and the average CPU-time/timestep was 1.67 s. The second run involved fluid and 64 octahedral particles. The average CPU-time/timestep here was 2.73 s; hence, the particle calculations for 64 particles took just over 1 s/timestep.

8. Concluding Remarks

We have presented a new fast numerical method for studying in three dimensions the dynamics of particles suspended in a very viscous fluid. The method's behavior compares favorably in a variety of tests with existing theories for sedimentation in two- and many-particle systems. Still, these results are preliminary; more sophisticated comparisons of the method with theory and experiment for a range of physical situations are needed to foster confidence in its predictions. Also, much work remains to be done to characterize better the influence of numerical parameters (e.g., mesh size, number and arrangement of points and links in a particle, stiffness of links) on the results produced by the method. In addition, the

method can be made more efficient, e.g., by exploiting the sparseness of the Hessian matrices in the particle-force calculations. Yet, we think it is not premature to say that the method will prove to be a powerful numerical tool for conducting multiparticle studies of a wise range of suspension flows. Among the special strengths of the numerical methods are its relative efficiency in handling large numbers of particles and its ability to treat particles of different sizes and shapes. While we have thusfar only looked at the rigid-particle limit, we expect that the method will also allow the study of suspensions of deformable particles and polymers. Further, the method is not limited to studying flows in unbounded domains, as walls (e.g., those of a tube) can also be assembled out of arrays of the same kind of localized forces as those used to model the particles.

ACKNOWLEDGMENTS

This work was supported in part by the Multiphase Flow Project (under National Science Foundation Grant DMS-8312229) at the Courant Institute of Mathematical Sciences, in part by the National Science Foundation under Grants DMS-8502339 and DMS-8602166, and in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-765F00098.

REFERENCES

1. G. K. BATCHELOR, *J. Fluid Mech.* **52**, 245 (1972).
2. G. K. BATCHELOR, *An Introduction to Fluid Dynamics* (Cambridge Univ. Press, London, 1974), p. 153.
3. S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ, AND A. H. SHERMAN, Yale University Department of Computer Science Report 112, 1977 (unpublished).
4. D. FISCHER, G. GOLUB, O. HALD, C. LEIVA, AND O. WIDLUND, *Math. Comp.* **28**, 349 (1974).
5. A. L. FOGELSON, *J. Comput. Phys.* **56**, 111 (1984).
6. A. L. FOGELSON AND C. S. PESKIN, in preparation.
7. P. E. GILL AND W. MURRAY, *Math. Programming* **7**, 311 (1974).
8. P. E. GILL AND W. MURRAY, *Numerical Methods for Constrained Optimization*, (Academic Press, New York, 1974), p. 37.
9. P. E. GILL AND W. MURRAY, National Physical Laboratory Report NAC 37, 1974, (unpublished).
10. J. HAPPEL AND H. BRENNER, *Low Reynolds Number Hydrodynamics* (Nijhoff, Boston, 1983), p. 242.
11. H. HASIMOTO, *J. Fluid Mech.* **5**, 317 (1959).
12. C. S. PESKIN, *J. Comput. Phys.* **25**, 220 (1977).