NUMERICAL METHODS FOR THE MOTION OF PARTICLES IN LOW REYNOLDS NUMBER HYDRODYNAMICS

by

Zhifeng Guo

B.Sc., University of Science and Technology of China, 1987

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

Mathematics and Statistics

SIMON FRASER UNIVERSITY

December 1997

0-612-24144-0

APPROVAL

Name:	Zhifeng Guo
Degree:	Master of Science
Title of thesis:	NUMERICAL METHODS FOR THE MOTION OF PARTICLES IN LOW REYNOLDS NUMBER HYDRODYNAMICS

Examining Committee: Dr. A. Lachlan
Chair

Dr. M. C. A. Kropinski,
Senior Supervisor

Dr. T. Tang

Dr. K. Promislow

Dr. B. Russell,
External Examiner

Date Approved: December 8, 1997

Abstract

Numerical methods for calculating the trajectories of solid particles moving under the action of hydrodynamic forces in a low Reynolds number fluid are investigated. Under the assumption of a vanishing Reynolds number, the fluid flow is governed by the Stokes equations and the total forces acting on the particles are zero. This is the basis of the Quasi-Static Approximation in low Reynolds number hydrodynamics: particles adjust their velocity instantaneously to maintain a force-free configuration, and their motion is computed as a sequence of steady-state solutions to the Stokes equations. The Stokes equations are formulated as an integral equation based on complex-variable theory for the biharmonic equation and are solved numerically using a spectrally-accurate discretization scheme. Several methods are investigated for integrating the initial value problem for computing the particle trajectories. These methods include the forward and backward Euler's methods, the Runge-Kutta schemes available in Brankin, Gladwell and Shampine's RKSUITE package, and the Adams-Pece formulae in Shampine and Gordon's ODE package. The performance of these time-integration schemes is tested on examples of computing the motion of neutrally-buoyant ellipses in a shear flow. Our results show that the ODE package is best for high accuracy simulation, and RKSUITE is most efficient when low to moderate accuracy is required.

Acknowledgements

I would like to express my sincere thanks to my supervisor Dr.M. C. A. Kropinski for her patience, guidance and introduced me to the topics in the thesis.

The author is grateful for the continuing support of the Department of Mathematics and Statistics, Simon Fraser University.

Dedication

To my family

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The motion of particles in low Reynolds number hydrodynamics has a wide application in a number of diverse fields such as colloid science, aerosol and hydrosal technology, and blood flow (see, e.g. [10, 21]). In treating unsteady motion of small particles, the Quasi-Static Approximation is often used, which neglects the particles' inertia and assumes the particles adjust their velocity instantaneously to maintain a force-free configuration. Thus their motion is composed of a sequence of steady states.

A number of numerical methods have been developed to study particle motions under the Quasi-Static Approximation, These include Stokesian Dynamics (c.f. [1, 27]), multipole collocation methods (c.f. [7]) and boundary integral equation techniques (c.f. [27]). In general, these methods are either low-order accurate or are limited to studying particles of regular shape. In [9], Greengard *et al.* presented an integral equation method based on complex-variable theory for the biharmonic equation. These methods are highly efficient and accurate. Kropinski [13] extended these methods by developing an integral equation formulation to compute particle trajectories in a slow viscous flow. However, in this paper the performance of various time integration schemes was not investigated. In this thesis, we investigate the forward and backward Euler's methods, various Runge-Kutta schemes and the Adams-Pece methods on calculating the trajectories of elliptical cylinders in a shear flow. We compare their performance in terms of both accuracy and computational efficiency.

In Chapter 2, we start with a review of the equations governing low Reynolds number hydrodynamics and give the basis for the Quasi-Static Approximation. The relevant complex variable theory for the biharmonic equation is discussed, and we conclude this chapter by formulating the Sherman-Lauricella integral equation.

In Chapter 3, we discuss the numerical discretization of the Sherman-Lauricella integral equation. A review of the time-integration schemes for the initial value problems is also given in this chapter.

In Chapter 4, examples of the motion of neutrally-buoyant particles in a shear flow are studied. The numerical results are presented and analyzed.

Finally, in Chapter 5, we present our conclusions on the best time integration scheme and we discuss some areas for future work.

Chapter 2

General Theory

There are many problems in fluid dynamics that deal with situations in which the inertia forces can be neglected. This type of flow is called slow viscous flow or Stokes flow (c.f. [4, 15, 25]). There are many fields in Science and Engineering in which Stokes flows are important, for example colloid chemistry and biology. Many practical devices, such as fluid-lubricated bearings, and many modern industrial processes, such as the manufacture of color film or magnetic recording tape, involve Stokes flows.

In this chapter, we consider the motion of a solid particles in incompressible Stokes flows. First, we derive the dimensionless governing equations and discuss the Quasi-Steady approach. In section 2.2, we discuss the complex variable theory for the biharmonic equation, and in section 2.3, we derive the integral equation based on this theory.

2.1 The Governing Equations and The Quasi-Static Approximation

We consider slow viscous flow in a two-dimensional infinite domain D with boundary Γ which is M-ply connected. The boundary Γ consists of the boundaries of the solid particles which are denoted by $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$ (see Figure 2.1).

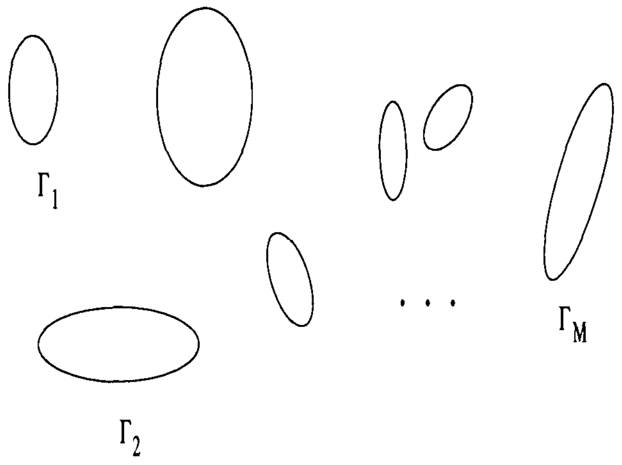

Figure 2.1: An unbounded multiply-connected domain. The solid particle boundaries are denoted by $\Gamma_1, \ldots, \Gamma_M$.

The motion of the fluid is governed by the principles of classical mechanics and thermodynamics for conservation of mass, momentum and energy. Application of these principles to a Newtonian, incompressible, isothermic fluid gives us the Navier-Stokes equations(see [4, 15, 25]):

$$\begin{aligned} \rho(\partial \mathbf{u}/\partial t + \mathbf{u}\cdot\nabla\mathbf{u}) &= -\nabla p + \mu\nabla^2\mathbf{u}, \\ \nabla\cdot\mathbf{u} &= 0, \end{aligned} \tag{2.1}$$

for $\mathbf{x} \in D$, where ρ is the fluid density, $\mathbf{u} = (u, v)$ is velocity field, p is the pressure, and μ is the dynamic coefficient of viscosity.

On the boundary Γ_k of the k-th particle, the no-slip boundary condition is applied:

$$\mathbf{u} = \mathbf{V}_k + \Omega_k(-(y-y_k), x-x_k), \tag{2.2}$$

where $\mathbf{V_k} = (u_k, v_k)$ is the translational velocity, $\Omega_k = d\theta_k/dt$ is the angular velocity and $\mathbf{x}_k = (x_k, y_k), \theta_k$ are respectively the center of mass and the angle of an axis fixed with respect to the particle(see Figure 2.2).

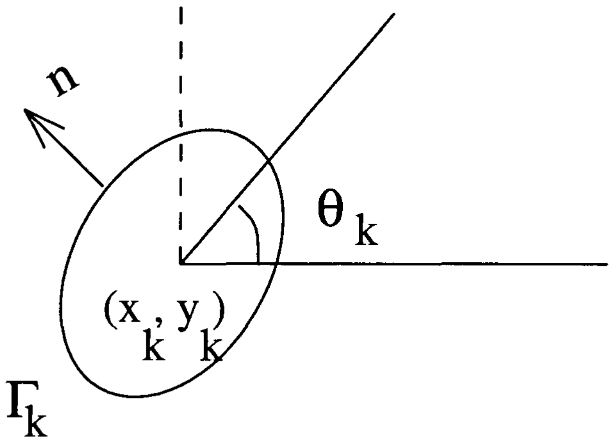

Figure 2.2: The k-th particle rotates in the plane: (x_k, y_k) is its centre, θ_k is its orientation angle, and $\mathbf{n}$ is the unit vector normal to its boundary.

To study the motion of the particles in the flow, we must consider the forces acting on the particles. According to Newton's second Law, equations of motion for each particle are

$$m_k \frac{d\mathbf{V}_k}{dt} = \mathbf{F}_k, \quad I_k \frac{d\Omega_k}{dt} = T_k, \quad k = 1, 2, \cdots, M, \tag{2.3}$$

where m_k is the mass and I_k is the rotational inertia of the k-th particle. Here, the total force $\mathbf{F}_k$ is the sum of the applied force such as gravity and the hydrodynamic force, i.e.

$$\mathbf{F}_k = \mathbf{F}_k^{appl} + \mathbf{F}_k^{hydro}.$$

Similarly, for the total torque we have

$$T_k = T_k^{appl} + T_k^{hydro}.$$

The hydrodynamic forces and torques are given by [13, 15]

$$\mathbf{F}_k^{hydro} = \int_{\Gamma_k} \sigma \mathbf{n} ds, \quad T_k^{hydro} = \int_{\Gamma_k} (-(y - y_k), x - x_k) \cdot \sigma \mathbf{n} ds,$$

where $\mathbf{n}$ is the unit vector normal to the particle's surface (see Figure 2.2), pointing into the fluid, and σ is the stress tensor defined by [13, 15]

$$\sigma = -pI + \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T).$$

The particle trajectories are calculated by solving the equations of motion (2.3) coupled with the Stokes equations (2.1) and boundary conditions (2.2).

In order to measure the magnitude of various physical effects, we write equations (2.1) in terms of dimensionless variables. Let the characteristic length of a particle be d and the characteristic velocity be a certain U. Scaling the velocity and the spatial coordinates by these characteristic values and the pressure by $\mu U/d$, the scaled Navier-Stokes equations are

$$\begin{aligned} R(\partial \mathbf{u}/\partial t + \mathbf{u} \cdot \nabla \mathbf{u}) &= -\nabla p + \nabla^2 \mathbf{u}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{2.4}$$

for $\mathbf{x} \in D$, where $R = \rho d U/\mu$ is the Reynolds number.

The Reynolds number represents the ratio between convective inertia forces and viscous effects. In the case of a Stokes flow, the viscous effects dominate and $R \to 0$. This is the usual consequence of considering the motion of very small particles $d << 1$, for example. We can then approximate the Navier-Stokes equations by the Stokes equations:

$$\nabla p = \nabla^2 \mathbf{u}, \quad \nabla \cdot \mathbf{u} = 0, \quad \mathbf{x} \in D. \tag{2.5}$$

We scale the variables in the dynamic equations (2.3) similarly. If we choose d/U as a reference time and $\mu U d$ as a force, these equations become

$$\frac{m_k}{\rho d^3} R \frac{d\mathbf{V}_k}{dt} = \mathbf{F}_k, \quad \frac{I_k}{\rho d^5} R \frac{d\Omega_k}{dt} = T_k, \quad k = 1, 2, \cdots, M$$

When $R \to 0$, we obtain the approximation

$$\mathbf{F}_k = 0, \quad T_k = 0, \quad k = 1, 2, \ldots, M.$$

The above equation says that the particles adjust their velocities and angular velocities instantaneously to maintain a force-free configuration. This is the basis of the Quasi-Steady Approximation: the particle trajectories are composed of a sequences of steady states. The following initial value problem is used to compute the trajectories of the particles:

$$\frac{d\mathbf{x}_k}{dt} = \mathbf{V}_k, \quad \frac{d\theta_k}{dt} = \Omega_k, \quad k = 1, 2, \ldots, M, \tag{2.6}$$

together with a given initial configuration. Equations (2.6) are coupled with the Stokes equations (2.5) and boundary conditions (2.2).

2.2 The Complex Variable Theory for the Biharmonic Equation

For two-dimensional Stokes flow, equations (2.5) can be rewritten as

$$\frac{\partial p}{\partial x} = \nabla u, \quad \frac{\partial p}{\partial y} = \nabla v, \quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{2.7}$$

These equations can be expressed more compactly by introducing a scalar stream function W(x,y) [4, 15]:

$$u = \frac{\partial W}{\partial y}, \quad v = -\frac{\partial W}{\partial x}. \tag{2.8}$$

Substituting (2.8) into (2.7), gives us

$$\frac{\partial p}{\partial x} = \frac{\partial \triangle W}{\partial y}, \quad \frac{\partial p}{\partial y} = -\frac{\partial \triangle W}{\partial x}, \tag{2.9}$$

and after eliminating the pressure p, the biharmonic equation is obtained:

$$\triangle^2 W = 0.$$

The no-slip boundary condition (2.2) becomes

$$\left(\frac{\partial W}{\partial y}, -\frac{\partial W}{\partial x}\right) = \mathbf{V}_k + \Omega_k(-(y-y_k), x-x_k), \quad \mathbf{x}_k \in \Gamma_k,$$

The vorticity field has only one component, perpendicular to the plane of the flow,

$$\zeta = -\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) = \triangle W. \tag{2.10}$$

Substituting the expression for vorticity (2.10) into (2.9), we get

$$\frac{\partial p}{\partial x} = -\frac{\partial \zeta}{\partial y}, \quad \frac{\partial p}{\partial y} = \frac{\partial \zeta}{\partial x},$$

which are just the Cauchy-Riemann equations ([4, 15]). Thus ζ and p are conjugate harmonic functions, so they are the real and imaginary parts of an analytic function f of a complex variable z, i.e.,

$$f(z) = \zeta + ip,$$

where $z = x + iy$.

We now derive an expression for $W(x,y)$ in terms of complex variables. Let $\phi(z) = g(x,y) + i\tilde{g}(x,y)$ be one of the infinitely many analytic functions whose derivative is $\frac{1}{4}(\zeta - ip)$. It is easy to verify that

$$G(x,y) = -xg(x,y) - y\tilde{g}(x,y) + W(x,y)$$

is also harmonic, since

$$\begin{aligned} \triangle G &= -x\triangle g - y\triangle\tilde{g} - 2\left(\frac{\partial g}{\partial x} + \frac{\partial \tilde{g}}{\partial y}\right) + \triangle W \\ &= -4\frac{\partial g}{\partial x} + \triangle W \\ &= -\zeta + \triangle W \\ &= 0. \end{aligned}$$

If χ is the analytic function whose real part is $G(x,y)$, then we have

$$W(x,y) = -\mathrm{Re}[\chi(z)] + xg(x,y) + y\tilde{g}(x,y).$$

However, $xg(x,y) + y\tilde{g}(x,y) = \text{Re}[\bar{z}\phi(z)]$, so that

$$W(x,y) = \text{Re}[\bar{z}\phi(z) + \chi(z)]. \tag{2.11}$$

Thus the stream function $W(x,y)$ can be expressed in terms of a pair of complex potentials $\phi(z)$ and $\chi(z)$. This follows from the fact that W satisfies the biharmonic equation.

The functions $\phi(z)$ and $\psi(z) = \chi'(z)$ are known as Goursat's functions. The velocity components and stress components can be expressed in terms of Goursat's functions. Calculating the derivative of W from equation (2.11) leads to Muskhelishvili's formula [17, 18, 19]

$$\frac{\partial W}{\partial x} + i\frac{\partial W}{\partial y} = \phi(z) + z\overline{\phi'(z)} + \overline{\psi(z)}. \tag{2.12}$$

This provides an expression for the velocity. In a similar way, the pressure, vorticity, and the stress tensors can be expressed in terms of the Goursat's functions,

$$\zeta + ip = -4\phi'(z),$$

and

$$\begin{aligned}
\sigma_{11} &= 2\text{Im}[2\phi'(z) - \overline{z}\phi''(z) - \psi'(z)] \\
\sigma_{22} &= 2\text{Im}[2\phi'(z) + \overline{z}\phi''(z) + \psi'(z)] \\
\sigma_{12} &= \sigma_{21} = -2\text{Re}[\overline{z}\phi''(z) + \psi'(z)].
\end{aligned}$$

Muskhelishvili's formula (2.12) must satisfy the no-slip boundary conditions. In terms of complex variables this is

$$\phi(\tau) + \tau\overline{\phi'(\tau)} + \overline{\psi(\tau)} = iV_k - \Omega_k(\tau - z_k), \quad \tau \in \Gamma_k, \tag{2.13}$$

where $V_k = u_k + iv_k$, and $z_k = x_k + iy_k$ for $\mathbf{x} \in \Gamma_k$. Thus we have reduced the problems in Stokes flow to a problem in complex variable theory.

We now view the force F_k as a complex variable function with $F_k = (F_x + iF_y)_k$. The hydrodynamic force F_k^{hydro} can also be written in terms of the Goursat functions ϕ and ψ [13]

$$F_k^{hydro} = 2[\phi - \overline{\psi}]_{\Gamma_k}, \tag{2.14}$$

where $[f(z)]$ means the increment in $f(z)$ as the curve Γ_k is traversed in the clockwise direction. An expression for the hydrodynamic torque acting on particle k is

$$T_k^{hydro} = -2\mathrm{Im}\left[[\overline{z_k}\phi]_{\Gamma_k} - \int_{\Gamma_k}(z - z_k)d\psi\right]. \tag{2.15}$$

2.3 Integral Equation Formulation

We now discuss a construction of the analytical functions $\phi(z)$ and $\psi(z)$ which satisfy (2.13) on the boundaries of the domain. Following the discussion by L. Greengard *et al.* [9], we start with the Sherman-Lauricella representations

$$\begin{aligned} \phi(z) &= \frac{1}{2\pi i}\int_\Gamma \frac{\omega(\xi)}{\xi - z}d\xi + \sum_{j=1}^{M} C_j \log(z - z_j) \\ \psi(z) &= \frac{1}{2\pi i}\int_\Gamma \frac{\overline{\omega(\xi)d\xi} + \omega(\xi)d\overline{\xi}}{\xi - z} - \frac{1}{2\pi i}\int_\Gamma \frac{\overline{\xi}\omega(\xi)}{(\xi - z)^2}d\xi \\ &\quad + \sum_{j=1}^{M}\frac{b_j}{z - z_j} + \sum_{j=1}^{M}\overline{C_j}\log(z - z_j) - \sum_{j=1}^{M} C_j \frac{\overline{z}}{z - z_j}, \end{aligned} \tag{2.16}$$

where $\omega(\xi)$ is an unknown complex density, C_j are complex constants and b_j are real constants.

The singularities centred in each particle are directly related to the hydrodynamic force and torque acting on that particle. These relations are given by [13]:

$$\begin{aligned} F_k^{hydro} &= -8\pi i C_k, \\ T_k^{hydro} &= 4\pi i b_k. \end{aligned}$$

According to Quasi-Static approximation, the total force and torque acting on the particle are zero. Thus, we have

$$\begin{aligned} F_k^{appl} &= -F_k^{hydro} = 8\pi i C_k \\ T_k^{appl} &= -T_k^{hydro} = -4\pi i b_k. \end{aligned}$$

Therefore, the singularity strengths C_k and b_k are given by

$$C_k = \frac{F_k^{appl}}{8\pi i}, \quad b_k = -\frac{T_k^{appl}}{4\pi i}.$$

To obtain the Sherman-Laricella integral equation, we substitute the representations (2.16) into Muskhelishvili's formula (2.13) and take the limit as z tends to a point τ on the boundary Γ. This gives [13]

$$\omega(\tau)+\frac{1}{2\pi i}\int_\Gamma \omega(\xi)\mathrm{d}\ln\frac{\xi-\tau}{\bar{\xi}-\bar{\tau}}-\frac{1}{2\pi i}\int_\Gamma \overline{\omega(\xi)}\mathrm{d}\ln\frac{\xi-\tau}{\bar{\xi}-\bar{\tau}}-iV_k+\Omega_k(\tau-z_k)=g(\tau), \quad (2.17)$$

where $g(\tau)$ is

$$g(\tau)=-\sum_{j=1}^{M}\left(2C_j\log|\tau-z_j|+\overline{C_j}\frac{\tau-z_j}{\bar{\tau}-\bar{z_j}}+\frac{b_j}{\bar{\tau}-\bar{z_j}}\right).$$

In (2.17), $\omega(\xi), V_k$ and Ω_k are unknowns. To complete the system, we add some contraints (c.f. [13] for a discussion)

$$\int_{\Gamma_k}\omega(\xi)ds=0, \quad i\int_{\Gamma_j}\omega(\tau)d\bar{t}-\overline{\omega(\tau)}d\tau=0, \quad j=1,2,\cdots,M. \qquad (2.18)$$

Provided that the contours Γ_k are smooth, (2.17) is a Fredholm integral equation of the second kind with a smooth kernel, and therefore the Fredholm alternative applies. A discussion on invertibility for this type of integral equation can be found in [17, 18, 19].

Chapter 3

Numerical Methods

To calculate the trajectories of solid particles in a Stokes flow, we must integrate the equations (2.6) in time. This integration is coupled with the solutions to the integral equation (2.17) which calculate the particle velocities based on the Quasi-Static Approximation.

We introduce the numerical methods in two parts. In section 3.1, we discuss the numerical solution for the integral equation, and in section 3.2, we discuss the methods to solve the initial value problem.

3.1 Numerical Solution of Integral Equations

In order to solve the Sherman-Lauricella integral equation (2.17) and (2.18), we use a Nyström discretization algorithm based on the trapezoidal rule. This quadrature achieves superalgebraic convergence for smooth functions on smooth periodic boundaries, i.e. this algorithm converges with a exponential rate. For this purpose, assume that we are given N points on each particle boundary Γ_k, equispaced in some parameterization $\tau^k : [1, L_k] \to \Gamma_k$. At each point τ_j^k on boundary Γ_k, there is a corresponding unknown value ω_j^k. We denote the derivative $(\tau_k)'$ as d_k and assume that we are given the derivative value d_j^k at the discretization points. The step length in the discretization of Γ_k is defined by $h_k = L_k/N$, where L_k is the length of the curve Γ_k. The total number of discretization points is MN.

The discretized Sherman-Lauricella integral equation (2.17) is [13]

$$\omega_j^k + \sum_{m=1}^{M}\sum_{n=1}^{N} K_1(\tau_j^k,\tau_n^m)\omega_n^m + \sum_{m=1}^{M}\sum_{n=1}^{N} K_2(\tau_j^k,\tau_n^m)\overline{\omega_n^m} - iV_k + \Omega_k(\tau_j^k - z_k) = g_j^k, \quad (3.1)$$

where $g_j^k = g(\tau_j^k)$. The kernels K_1 and K_2 are given by

$$\begin{aligned} K_1(\tau_j^k,\tau_n^m) &= \frac{h_m}{2\pi i}\left(\frac{d_n^m}{\tau_j^k-\tau_n^m} + \frac{\overline{d_n^m}}{\overline{\tau_j^k-\tau_n^m}}\right) \\ K_2(\tau_j^k,\tau_n^m) &= -\frac{h_m}{2\pi i}\left(\frac{d_n^m}{\overline{\tau_j^k-\tau_n^m}} - \frac{\overline{d_n^m}(\tau_j^k-\tau_n^m)}{(\overline{\tau_j^k-\tau_n^m})^2}\right), \end{aligned}$$

when $\tau_j^k \neq \tau_n^m$; when $t_j^k = t_n^m$, K_1 and K_2 are replaced by the appropriate limits (c.f. [9, 13]):

$$\begin{aligned} K_1(\tau_j^k,\tau_n^m) &= \frac{h_k}{2\pi}\kappa_j^k|d_j^k| \\ K_2(\tau_j^k,\tau_n^m) &= -\frac{h_k}{2\pi}\kappa_j^k(d_j^k)^2/|d_j^k|, \end{aligned}$$

where κ_j^k denotes the curvature at the point τ_j^k. Equations (2.17) discretized as

$$h_k\sum_{j=1}^{N_k}\omega_j^k = 0, \quad ih_k\sum_{j=1}^{N_k}\omega_j^k\overline{d_j^k} - \overline{\omega_j^k}d_j^k = 0. \quad (3.2)$$

Using a complex conjugation operator $\mathcal{C}$, we can rewrite equations (3.1) and (3.2) as

$$(I + K_1 + K_2\mathcal{C})\omega + E\mathbf{u} = \mathbf{g},$$

$$(F_1 + F_2\mathcal{C})\omega = \mathbf{0}.$$

where E, F_1, F_2 are coefficient matrices, and

$$\begin{aligned} \omega &= (\omega_1^1,\omega_2^1,\cdots,\omega_N^1,\cdots,\omega_1^M,\cdots,\omega_N^M)^T, \\ \mathbf{u} &= (u_1,v_1,\Omega_1,\cdots,u_M,v_M,\Omega_M)^T, \\ \mathbf{g} &= (g_1^1,g_2^1,\cdots,g_N^1,\cdots,g_1^M,\cdots,g_N^M)^T. \end{aligned}$$

Therefore, the following system of equations is obtained:

$$\begin{pmatrix} I + K_1 + K_2\mathcal{C} & E \\ F_1 + F_2\mathcal{C} & 0 \end{pmatrix} \begin{pmatrix} \omega \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{0} \end{pmatrix}. \tag{3.3}$$

The coefficient matrix is a $(2MN+3M) \times (MN+3M)$ matrix, where the $2MN \times 3M$ matrix E represents the influence of the unknown particle velocities in the discretized Sherman-Lauricella equation and the $3M \times 2MN$ matrix F represents the discrete constraint equations (3.2).

The linear system (3.3) is solved using Gaussian Elimination. The cost of computation for Gaussian Elimination increases with the number of unknowns and is proportional to $(2MN + 3M)^3$. We view M as being fixed, thus the cost is $O(N^3)$. We only consider the cases with small $N(N \leq 256)$ and $M(M \leq 2)$. However, for more complicated problems involving many particles and a more refined discretization, the computational cost of Gaussian Elimination becomes too expensive. In [9, 13], the matrix equation are solved iteratively using the Fast Multipole Method (FMM) (c.f. [5, 8]) to compute the matrix-vector products efficiently. This method reduces the cost to $O(N)$ [5, 8, 9].

3.2 Numerical Methods for the Initial Value Problem

We wish to investigate the performance of various numerical methods to integrate (2.6). Here, we compare the performance of RKSUITE [2], Shampine and Gordon's ODE package [22] and the forward and backward Euler's methods.

We write the initial value problem into vector form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{3.4}$$

where $\mathbf{y}$ is the vector of unknowns and $\mathbf{y}_0$ are the initial conditions. The vector $\mathbf{y}$ contains the particles' orientations,

$$\mathbf{y} = (x_1, y_1, \theta_1, \cdots, x_M, y_M, \theta_M),$$

and the vector $\mathbf{f}(t, \mathbf{y})$ contains the velocities,

$$\mathbf{f}(t, \mathbf{y}) = (u_1, v_1, \Omega_1, \cdots, u_M, v_M, \Omega_M).$$

Thus for M particles, (3.4) is a system of $3M$ equations. We now discuss the various methods in detail.

3.2.1 The Forward and Backward Euler's Methods

One of the most straightforward methods to solve the initial value problem (3.4) is the forward Euler's method. Consider the time interval $[T_0, T_1]$ over which we want the solution $\mathbf{y}(t)$. Subdividing $[T_0, T_1]$ into n equispaced time intervals, our stepsize is

$$\Delta t = \frac{T_1 - T_0}{n}.$$

We find approximations to $\mathbf{y}$ at time t^j, where

$$t^j = T_0 + j\Delta t, \quad j = 1, 2, \cdots, n,$$

and these approximate values are denoted by $\mathbf{y}^j \approx \mathbf{y}(t^j)$ and are found from

$$\frac{\mathbf{y}^{j+1} - \mathbf{y}^j}{\Delta t} = \mathbf{f}^j, \tag{3.5}$$

where $\mathbf{f}^j = \mathbf{f}(t^j, \mathbf{y}^j)$. In general, for one-step method, the local truncation error τ^j is defined by

$$\tau^j = \mathbf{y}(t^j + \Delta t) - \mathbf{y}^{j+1}.$$

So the local truncation error of Euler's Method at t^j is the remainder of the first Taylor approximation

$$\begin{aligned} \mathbf{y}(t^j + \Delta t) &= \mathbf{y}(t^j) + \Delta t \mathbf{y}'(t^j) + \tfrac{1}{2}\mathbf{y}''(\xi)\Delta t^2 \\ &= \mathbf{y}(t^j) + \Delta t \mathbf{f}(t^j, \mathbf{y}(t_j)) + O(\Delta t^2) \end{aligned} \tag{3.6}$$

where $t^j < \xi < t^j + \Delta t$. The local truncation error τ^j in Lagrange's form is

$$\tau^j = \frac{1}{2}\mathbf{y}''(t^j + \zeta\Delta t)(\Delta t)^2 = O(\Delta t^2), \text{ where } 0 < \zeta < 1.$$

Thus the forward Euler's method is a first order method. One disadvantage of this method is that it is not unconditionally stable [16]: a stable time step size depends on the nature of the problem.

The backward Euler's method is similar to the forward method except it uses $\mathbf{f}^{j+1}$ instead of $\mathbf{f}^j$:

$$\mathbf{y}^{j+1} = \mathbf{y}^j + \Delta t \mathbf{f}^{j+1}, \tag{3.7}$$

where $\mathbf{f}^{j+1} = \mathbf{f}(t^{j+1}, \mathbf{y}^{j+1})$.

This scheme requires the value of $\mathbf{f}^{j+1}$, which is not known at step j. However, we can approximate $\mathbf{f}^{j+1}$ and $\mathbf{y}^{j+1}$ with following scheme:

$$\begin{aligned} \mathbf{y}_0^{j+1} &= \mathbf{y}^j \\ \mathbf{f}_0^{j+1} &= \mathbf{f}(t^{j+1}, \mathbf{y}_0^{j+1}), \\ \mathbf{y}_{i+1}^{j+1} &= \mathbf{y}^j + \Delta t \mathbf{f}_i^{j+1}, \qquad i = 1, 2, \cdots \end{aligned}$$

This iterative process continues until for a given small δ, there is a l such that

$$\max\left(\frac{|\mathbf{y}_l^{j+1} - \mathbf{y}_{l-1}^{j+1}|}{|\mathbf{y}_l^{j+1}|}, \frac{|\mathbf{f}_l^{j+1} - \mathbf{f}_{l-1}^{j+1}|}{|\mathbf{f}_l^{j+1}|}\right) < \delta.$$

The number of iterations to obtain appropriate $\mathbf{f}^{j+1}$ and $\mathbf{y}^{j+1}$ depends on the constant δ and time step Δt. A smaller δ or larger Δt will normally need more iterations.

The backward Euler's method has the same order of convergence as the forward method, but it is absolutely stable independent of the stepsize [16].

3.2.2 Runge-Kutta Methods and RKSUITE

The Euler's method was derived from the $O(\Delta t^2)$ Taylor expansion to approximate the solution to the differential equation. This technique can be extended to obtain higher order convergence. The method based on this technique is the nth-order Taylor method

$$\mathbf{y}^{j+1} = \mathbf{y}^j + \Delta t \mathbf{f}^j + \frac{\Delta t^2}{2}(\mathbf{f}')^j + \cdots + \frac{\Delta t^n}{n!}(\mathbf{f}^{(n-1)})^j. \tag{3.8}$$

The Taylor method has the disadvantage of requiring the computation and evaluation of the derivatives of $\mathbf{f}(t, \mathbf{y})$. Runge-Kutta methods use the high order local truncation error of the Taylor method while eliminating the computation and evaluation of the derivatives of $\mathbf{f}(t, \mathbf{y})$, since

$$\mathbf{f}' = \frac{d\mathbf{f}}{dt}(t, \mathbf{y}(t)) = \mathbf{f}_t + \mathbf{f}_\mathbf{y}\mathbf{y}' = \mathbf{f}_t + \mathbf{f}_\mathbf{y}\mathbf{f}.$$

For example, we can derive a second-order Runge-Kutta method by using

$$\mathbf{f}(t_j + p\Delta t, \mathbf{y}_j + q\Delta t\mathbf{f}_j) \approx \mathbf{f}_j + p\Delta t\mathbf{f}_t(t_j, \mathbf{y}_j) + q\Delta t\mathbf{f}_j\mathbf{f}_\mathbf{y}(t_j, \mathbf{y}_j),$$

and approximating $\mathbf{f}^j + \frac{\Delta t}{2}(\mathbf{f}')^j$ with $a_1\mathbf{f}(t_j, \mathbf{y}_j) + a_2\mathbf{f}(t_j + p\Delta t, \mathbf{y}_j + q\Delta t\mathbf{f}_j)$,

$$\mathbf{y}^{j+1} = \mathbf{y}^j + a_1\Delta t\mathbf{f}(t_j, \mathbf{y}_j) + a_2\Delta t\mathbf{f}(t_j + p\Delta t, \mathbf{y}_j + q\Delta t\mathbf{f}_j), \tag{3.9}$$

where the constants a_1, a_2, p and q are determined by

$$a_1 + a_2 = 1, \quad \text{and} \quad a_2 p = a_2 q = \frac{1}{2}.$$

This constraint is set so that the approximation is $O(\Delta t^2)$.

RKSUITE is a suite of codes based on Runge-Kutta formulae. It implements three Runge-Kutta formula pairs, (2,3), (4,5) and (7,8), which are corresponding to 3rd, 5th and 8th order methods, respectively. One of these pairs must be selected for the integration.

Generally, the nth-order Runge-Kutta scheme can be written as

$$\mathbf{y}^{j+1} = \mathbf{y}^j + \Delta t \phi(t^j, \mathbf{y}^j, \Delta t)$$

with a local truncation error τ_{j+1} of order $O(\Delta t^n)$. Suppose there is another method

$$\tilde{\mathbf{y}}^{j+1} = \tilde{\mathbf{y}^j} + \Delta t \tilde{\phi}(t^j, \mathbf{y}^j, \Delta t)$$

with a local truncation error $\tilde{\tau_{j+1}}$ of order $O(\Delta t^{n+1})$; then we have [3]

$$\tau_{j+1} \approx \frac{1}{\Delta t}(\tilde{\mathbf{y}}^{j+1} - \mathbf{y}^{j+1}),$$

and this implies for some constant k

$$k\Delta t^n \approx \frac{1}{\Delta t}(\tilde{\mathbf{y}}^{j+1} - \mathbf{y}^{j+1}). \tag{3.10}$$

These schemes are called an $(n, n+1)$ pair, and (3.10) can be used to control error.

Since in general the global error of the methods cannot be determined, the local truncation error is used to control the global error. The object of error control is to estimate an appropriate step size so that the error is less than a tolerance ε. Changing the step size from Δt to $q\Delta t$, where q is a positive number, and bounding $\tau_{j+1}(q\Delta t)$ by ε give us [3]

$$q \leq \left(\frac{\varepsilon \Delta t}{|\tilde{\mathbf{y}}^{j+1} - \mathbf{y}^{j+1}|}\right)^{1/n}.$$

The value of q determined at the ith step is used to repeat the calculation using $q\Delta t$, if necessary, and predict an appropriate initial choice of Δt for the next step.

RKSUITE uses relative local error control, with the error tolerance being the desired relative accuracy. The code tries to advance the integration as far as possible subject to the specified accuracy. At each step, an appropriate step size for its next step is chosen automatically so that the integration will proceed efficiently while keeping the local error estimate smaller than the tolerance. If the step size is too big for the formula to pass the tolerance control, the code will adjust the step size and try again. Poor choices of the tolerance parameters may lead to low accuracy, inefficient computation or even termination of the program. Thus, it is important to select appropriate tolerances for local error control when RKSUITE is used to solve initial value problems.

3.2.3 Adams Methods and ODE Package

The Euler's methods and Runge-Kutta methods are all one-step methods since the approximations for time t^{j+1} involve the information from the previous time step. Methods which include information from more than one time step are called multistep methods. The Adams method is one example.

Any solution of the differential equation (3.4) can be written as

$$\mathbf{y}(t^{j+1}) = \mathbf{y}(t^j) + \int_{t^j}^{t^{j+1}} \mathbf{y}'(s)ds = \mathbf{y}(t^j) + \int_{t^j}^{t^{j+1}} \mathbf{f}(s, \mathbf{y}(s))ds. \tag{3.11}$$

The Adams method approximates this solution by replacing $\mathbf{f}(s, \mathbf{y}(s))$ with a polynomial interpolating the computed derivative values $\mathbf{f}^i$, and then integrating the polynomial. An Adams formula of order k at t^j uses a polynomial $P_{k,j}(t)$ interpolating the computed derivative at the $k+1$ preceding points,

$$P_{k,j}(t^{j+1-i}) = \mathbf{f}^{j+1-i}, \quad i = 0, 1, \cdots, k.$$

This leads to

$$\mathbf{y}^{j+1} = \mathbf{y}^j + \Delta t \sum_{i=0}^{k} \alpha_i \mathbf{y}^{j+1-i}, \tag{3.12}$$

where the coefficients α_i are determined from the interpolation.

When $\alpha_0 = 0$, the method is explicit; otherwise, the method is implicit. In practice, implicit methods are used to improve approximations obtained by explicit methods. The combination of an explicit and implicit methods is called predictor-corrector (PECE) method (c.f. [3, 24]). The procedure of PECE is: predict P^{j+1}, evaluate $\mathbf{f}_P^{j+1}$, correct to get $\mathbf{y}^{j+1}$ and evaluate $\mathbf{f}^{j+1}$ to complete the step.

If the Euler's method ($k = 1$) is used as the predictor and the trapezoidal rule ($k = 2$) as the corrector, the PECE formulae are

$$\begin{aligned} P^{j+1} &= \mathbf{y}^j + \Delta t \mathbf{f}^j, \\ \mathbf{y}^{j+1} &= \mathbf{y}^j + \frac{\Delta t}{2}[\mathbf{f}(t^{j+1}, P^{j+1} + \mathbf{f}^j)], \\ \mathbf{f}^{j+1} &= \mathbf{f}(t^{j+1}, \mathbf{y}^{j+1}). \end{aligned} \tag{3.13}$$

In general, if the predictor is of order k and the corrector of order $k+1$, then

$$\mathbf{y}^{j+1} = \mathbf{y}^j + \Delta t \beta_k \nabla^k \mathbf{f}_P^{j+1-i}. \tag{3.14}$$

where β_k is a constant, and $\nabla^k \mathbf{f}_P^{j+1-i}$ is defined by

$$\begin{aligned} \nabla^0 \mathbf{f}_P^{j+1-i} &= \mathbf{f}_P^{j+1-i} = \mathbf{f}(t^{j+1-i}, P^{j+1-i}), \\ \nabla^1 \mathbf{f}_P^{j+1-i} &= \nabla \mathbf{f}_P^{j+1-i} = \mathbf{f}_P^{j+1-i} - \mathbf{f}^{j-i}, \\ \nabla^k \mathbf{f}_P^{j+1-i} &= \nabla^{k-1} \mathbf{f}_P^{j+1-i} - \nabla^{k-1} \mathbf{f}^{j-i}. \end{aligned}$$

ODE is a package of codes developed by Shampine *et al.* [24] based on the Adams methods in PECE form. The package includes a integrator (STEP), an interpolation routine, and a driver (DE). The code starts an integration by using (3.13), which requires only the differential equation and the initial condition. Upon completion of the first step, the code stores $\mathbf{y}^1, \mathbf{f}^1$ and $\mathbf{f}^0$, which is exactly the information that is needed for a second order predictor and third order corrector to take the second step. With the results from second step, a third order predictor and fourth order corrector can be estimated. In this way the code can increase the order of accuracy. Thus ODE package uses variable-order methods.

The Adams methods are most advantageous for problems in which function evaluations are expensive, or when moderate to high accuracy is requested. Some additional features are provided in the ODE code, such as the capability to detect and deal with moderate stiffness or detect and return the presence of severe stiffness. The code uses relative and absolute error tolerance to control the local error, but does not control the global error directly. If a constant step size is used in the implementation of the code, the global error depends mainly on the time step. The error tolerances are used to ensure the efficient completion of each step. However, an unreasonable choice of the tolerances will result in wasted computation or error return from the code.

Chapter 4

Numerical Results

We investigate the performance of the numerical methods described in Chapter 3 on examples of particles moving in a shear flow. All of the algorithms have been implemented in FORTRAN using double-precision arithmetic. All CPU times cited are for a SUN ULTRA 1 workstation.

4.1 The Rotation of an Ellipse in Shear Flow

In this example, we first consider the slow motion of a solid elliptical cylinder in a simple shear flow, rotating under the action of hydrodynamic forces. The undisturbed shear flow is given by $(u, v) = (\kappa y, 0)$ (see Figure 4.1).

Under the Quasi-Static Approximation, Jeffrey [12] obtained an analytical solution to describe the rotation of an ellipsoid. For a two-dimensional ellipse with an initial orientation given by $x^2/a^2+y^2/b^2 = 1$, the ellipse rotates with variable angular velocity (see Fig 4.1) given by

$$\Omega = \frac{d\theta}{dt} = \frac{-\kappa}{a^2+b^2}(a^2\sin^2\theta + b^2\cos^2\theta) \tag{4.1}$$

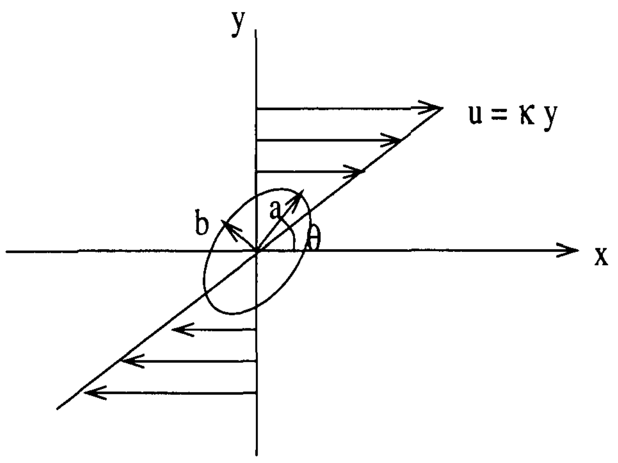

Figure 4.1: *A neutrally buoyant ellipse rotating in a simple shear flow* $(u,v) = (\kappa y, 0)$. *The ellipse is initially at* $\theta = 0$.

and the angle θ changes with time according to

$$\tan\theta = -\frac{b}{a}\tan\frac{ab\kappa t}{a^2+b^2}. \tag{4.2}$$

Note that the angular velocity and the angle depend only on the length ratio of the two semi-axes. We use these formulas to check the errors associated with various numerical methods mentioned in Chapter 3.

In [13], Kropinski checked the spatial accuracy of the integral equation method to obtain the instantaneous angular velocity of the ellipse. The relevant ratio used is $b/a = 10$, and the results showed that the solution converges very rapidly according to the spectral accuracy of the discretization. Here, we compute the case with the ratio $b/a = 1/7$. The superalgebraic convergence is seen in the results (Table 4.1): the solution is exact to machine precision with $N = 256$.

The number of discretization points N needed to adequately resolve the spatial discretization depends on the geometry of the ellipse. As the aspect ratio increases, more points are needed. This is demonstrated in Table 4.2.

We use Jeffrey's solution (4.1) and (4.2) to investigate the performance of FE

N	Ω_c	$\lvert\Omega_c - \Omega\rvert/\lvert\Omega\rvert$
16	+0.002766911227806	$1.1383455613903 \times 10^{-0}$
32	-0.019603558200636	$1.9822089968221 \times 10^{-2}$
64	-0.019999923765098	$3.8117451021946 \times 10^{-6}$
128	-0.019999999999998	$7.7021722333370 \times 10^{-14}$
256	-0.020000000000000	$5.2041704279305 \times 10^{-16}$

Table 4.1: *The instantaneous angular velocity of an ellipse with $b/a = 1/7$. N denotes the number of discretization points on the ellipse. The exact value is $\Omega = -0.02$.*

Ratio	$\lvert\Omega_c - \Omega\rvert/\lvert\Omega\rvert$
2	$5.5511151231260 \times 10^{-16}$
4	$1.9892976155233 \times 10^{-12}$
8	$4.1727766043054 \times 10^{-5}$
16	$2.0814887720058 \times 10^{-1}$
32	$2.2231407762913 \times 10^{+1}$

Table 4.2: *The error in the instantaneous angular velocity of an ellipse with different aspect ratios b/a . $N = 64$ is the number of discretization points on the ellipse.*

(the forward Euler's method), BE (the backward Euler's method), RKSUITE and the ODE package in calculating the particle's rotation. The aspect ratio of the ellipse we use is $b/a = 2$, and the number of discretization points on the ellipse is 64, thus the spatial discretization error is negligible (see Table 4.2).

We begin with a constant time step of $\Delta t = 0.1$, and integrate (2.6) with FE and BE over time from 0 to 16, which corresponds to two complete rotations of the ellipse. We find the maximum error in the angle and angular velocity is approximately 10^{-2}. We use this number as the relative error tolerance of RKSUITE (in this case, the (2,3) Runge-Kutta pair is used) and ODE package. Figure 4.2 shows the error in the axis angles $\Delta\theta$ and the angular velocity $\Delta\Omega/\lvert\Omega\rvert$. The exact solutions are shown in Figure 4.2 (a) and (b).

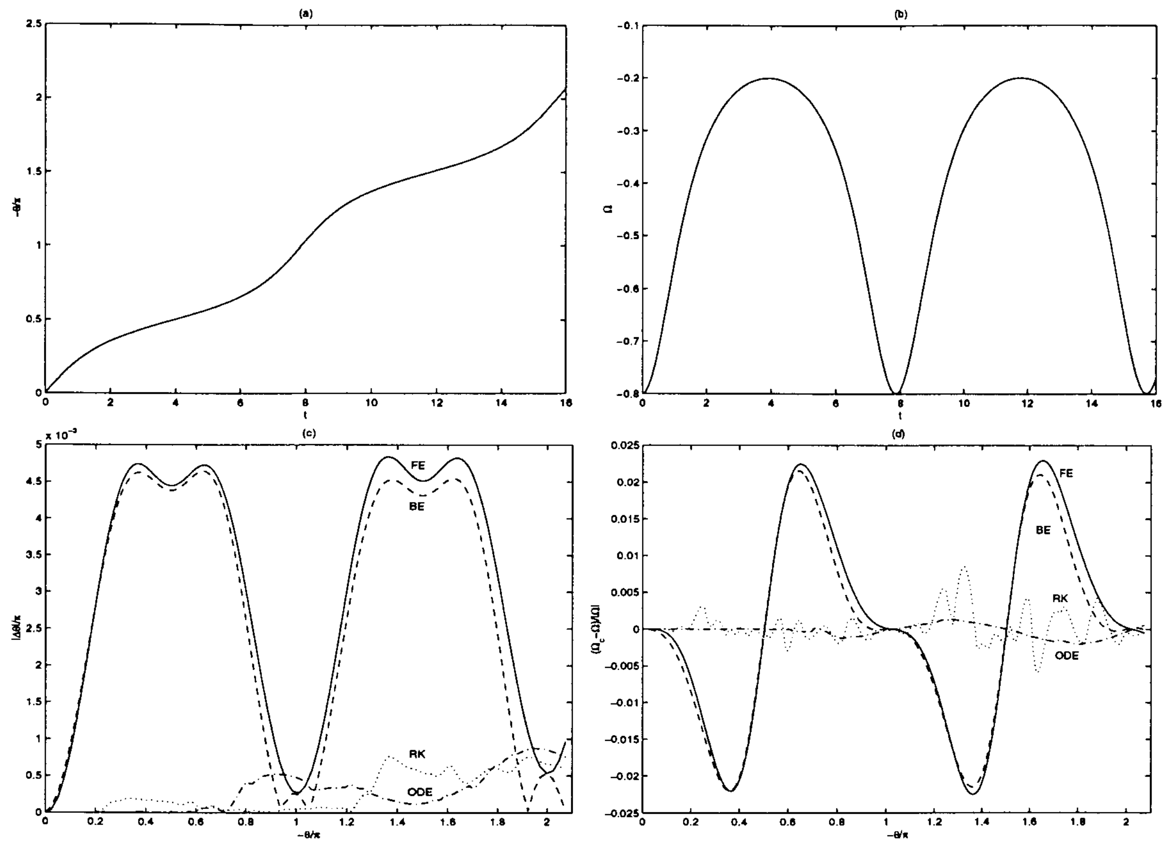

Figure 4.2: The results for time step $\Delta t = 0.1$, for an ellipse with $b/a = 2$. Plot (a) and (b) show the exact values for θ and Ω from Jeffrey's solution. Plot (c) and (d) show the errors for various time integration schemes.

The errors for FE and BE methods are qualitatively similar and the minimum errors occur when the ellipse returns to its initial configuration, i.e. the axis angle is $\pi, 2\pi, \cdots$. Smaller errors are seen for RKSUITE and ODE, but the error distribution shows no observable pattern. The comparison of CPU time and maximum errors for these methods are shown in Table 4.3.

We show similar comparison for $\Delta t = 0.01$ and $\Delta t = 0.001$ in Figures 4.3 and 4.4 and Tables 4.4 and 4.5. As can be seen, FE and BE show a convergence rate of Δt, as predicted. The ODE package achieves the smallest error (10^{-9} and 10^{-11} for time step of 0.01 and 0.001 with a tolerance of 10^{-4}), and converges very fast. RKSUITE is the

Method	$\max(\lvert\Delta\theta\rvert/\pi)$	$\max(\lvert\frac{\Delta\Omega}{\Omega}\rvert)$	CPU time
FE	$1.521379955 \times 10^{-2}$	$2.299314193 \times 10^{-2}$	39.2
BE	$1.459203235 \times 10^{-2}$	$2.198248979 \times 10^{-2}$	240.9
RKSUITE	$2.411457801 \times 10^{-3}$	$8.470804508 \times 10^{-3}$	10.5
ODE	$2.745405955 \times 10^{-3}$	$2.024073351 \times 10^{-3}$	18.1

Table 4.3: *The comparison of maximum errors and CPU time among various initial value methods. The time step is* $\Delta t = 0.1$. *RKSUITE uses (2,3) pair. Relative error tolerance for RKSUITE and ODE is* 10^{-2}.

Method	$\max(\lvert\Delta\theta\rvert/\pi)$	$\max(\lvert\frac{\Delta\Omega}{\Omega}\rvert)$	CPU time
FE	$1.476411103 \times 10^{-3}$	$2.210983255 \times 10^{-3}$	473.5
BE	$1.470237034 \times 10^{-3}$	$2.201142879 \times 10^{-3}$	1720.8
RKSUITE	$2.194438446 \times 10^{-3}$	$8.906136417 \times 10^{-3}$	24.0
ODE	$3.721113195 \times 10^{-9}$	$2.753160012 \times 10^{-9}$	163.1

Table 4.4: *The comparison of maximum errors and CPU time among various initial value methods. The time step is* $\Delta t = 0.01$. *RKSUITE uses (2,3) pair. Relative error tolerance for RKSUITE and ODE is* 10^{-3}.

Method	$\max(\lvert\Delta\theta\rvert/\pi)$	$\max(\lvert\frac{\Delta\Omega}{\Omega}\rvert)$	CPU time
FE	$1.472006184 \times 10^{-4}$	$2.202343256 \times 10^{-2}$	4694.2
BE	$1.471655672 \times 10^{-4}$	$2.201385026 \times 10^{-4}$	13953
RKSUITE	$7.987708795 \times 10^{-4}$	$4.351326924 \times 10^{-4}$	48.9
ODE	$6.400131076 \times 10^{-11}$	$4.797014671 \times 10^{-11}$	1889.7

Table 4.5: *The comparison of maximum errors and CPU time among various initial value methods. The time step is* $\Delta t = 0.001$. *RKSUITE uses (4,5) pair. Relative error tolerance for RKSUITE and ODE is* 10^{-4}.

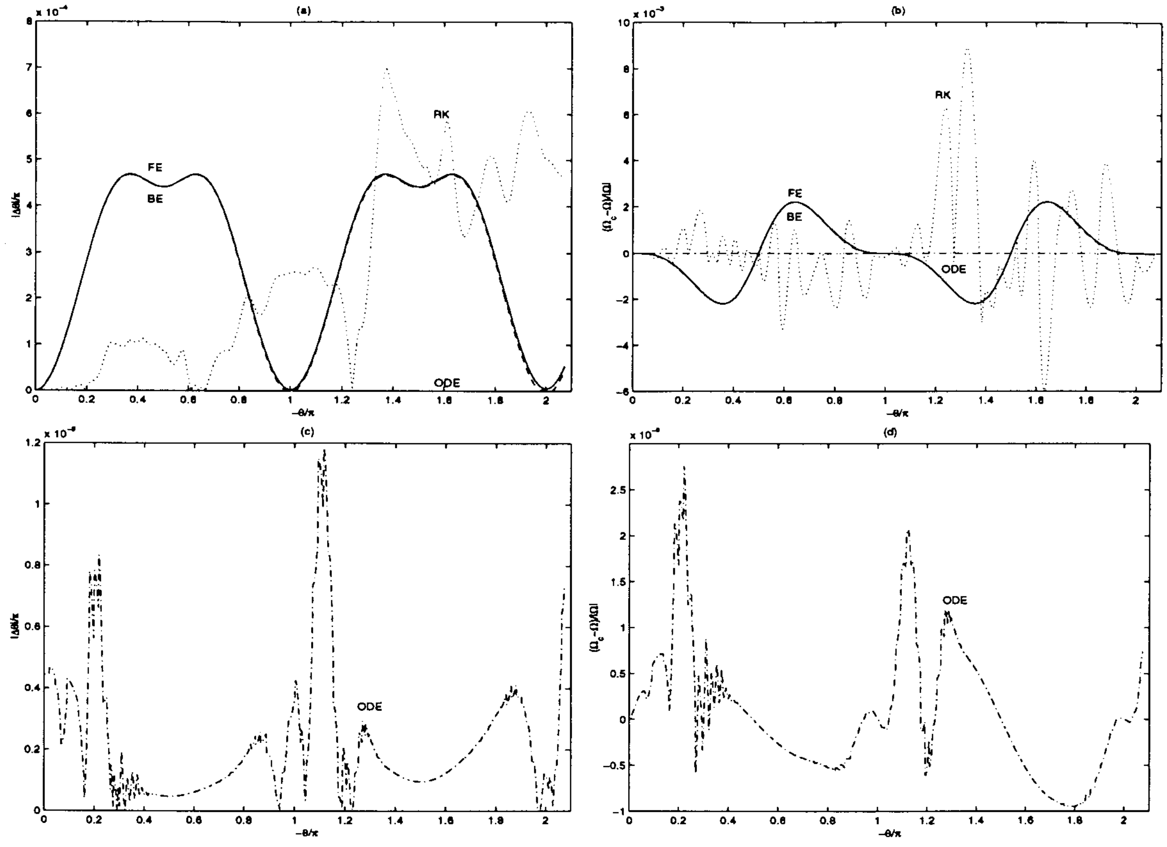

Figure 4.3: The results for time step $\Delta t = 0.01$, for an ellipse with $b/a = 2$. Plot (a) and (b) show the errors for various time integration schemes. Plot (c) and (d) show the errors for ODE.

fastest method, but the accuracy it achieved is similar to that for FE and BE. This is mainly due to the relatively large tolerance we used. For instance, if we use the same time step $\Delta t = 0.01$, integrate (2.6) with the (4,5) Runge-Kutta pair, with tolerance of $10^{-4}, 10^{-6}$ and 10^{-8}, the relative velocity error $(\Omega_c - \Omega)/|\Omega|$ improves from 10^{-3} to 10^{-8} (see Table 4.6). In comparison, to achieve an accuracy of 10^{-6} using the Forward Euler's method, a time step of $\Delta t = 0.00001$ is needed and approximate 108 hours CPU time is required.

With a smaller error tolerance, the CPU time for RKSUITE increases. From Table 4.4 and Table 4.6, to achieve a similar level of accuracy (10^{-9} for ODE and

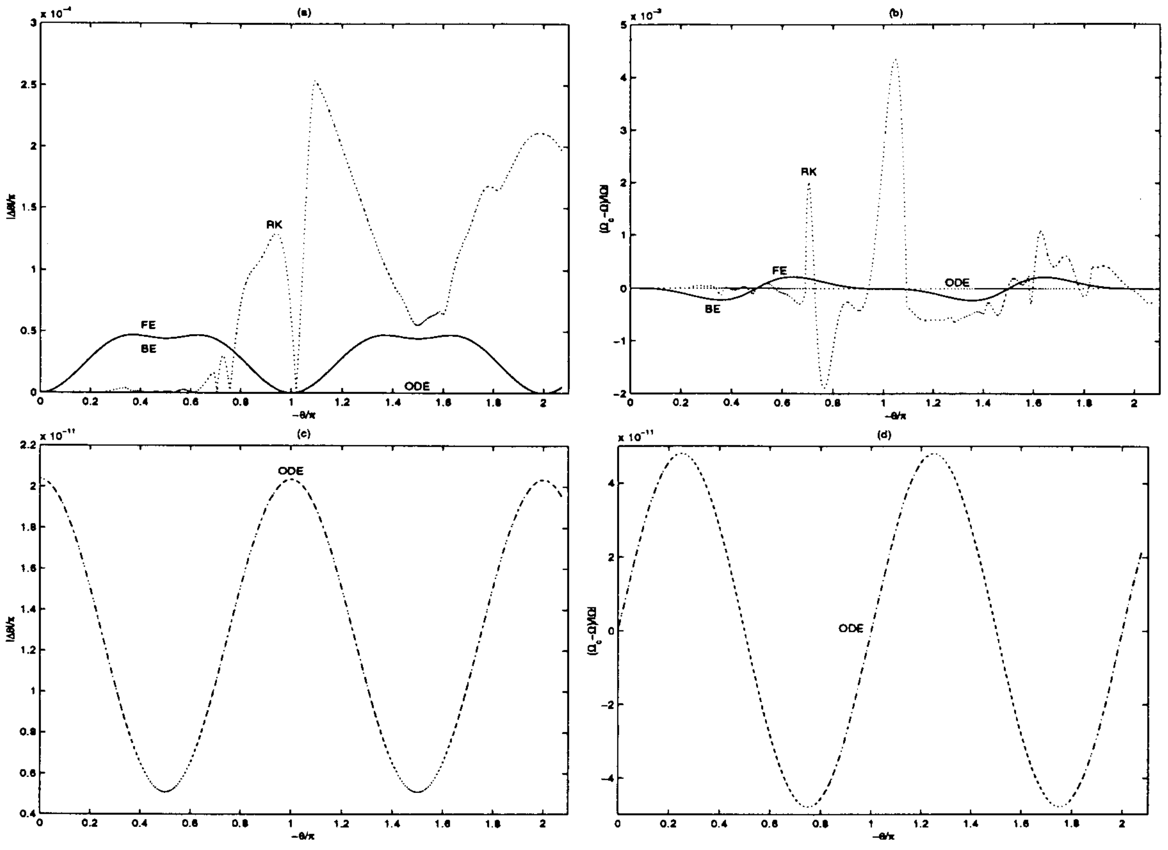

Figure 4.4: The results for time step $\Delta t = 0.001$, for an ellipse with $b/a = 2$. Plot (a) and (b) show the errors for various time integration schemes. Plot (c) and (d) show the errors for ODE.

10^{-8} for RKSUITE), RKSUITE requires 576.7 seconds of CPU time with 2370 function evaluations, while ODE requires 163.1 seconds of CPU time and 687 function evaluations.

Thus, among the initial value methods we test, the ODE package and RKSUITE give satisfactory performance for coupling with the integral equation method to simulate the motion of particles in shear flow. More specifically, if a high accuracy is required, ODE is best; if only low to moderate accuracy is required, RKSUITE gives fastest performance.

Tolerance	$\max(\lvert\frac{\Delta\Omega}{\Omega}\rvert)$	CPU time	Function Evaluation
10^{-4}	$1.228097477 \times 10^{-3}$	35.7	145
10^{-6}	$1.121866347 \times 10^{-5}$	66.9	273
10^{-8}	$2.119796121 \times 10^{-8}$	576.7	2370

Table 4.6: *The comparison of different tolerances for RKSUITE. The time step is* $\Delta t = 0.01$. $(4,5)$ *Runge-Kutta pair is used.*

4.2 The Motion of Two Cylinders in a Shear Flow

In this example, we study the motion of two neutrally buoyant cylinders in a shear Stokes flow. The circles are initially located in the top half of the plane. We investigate the change in motion of these two particles as the initial separation decreases. As particles come into close contact, an increase in mesh refinement is needed to resolve the integral equation. In [13], Kropinski studied this problem and showed for adequate resolution, the arc length spacing of mesh h_1 and h_2 of these two particles in close contact must satisfy

$$h_1, h_2 < \frac{1}{4}\varepsilon, \tag{4.3}$$

where ε is the distance of closest approach.

We consider two cylindrical particles in a simple shear flow. The cylinders have a radius of 1, and their centres are located initially at $(0, 1.5)$ and $(0, 3.6)$, respectively. Thus the distance of closest approach between them is $\varepsilon = 0.1$. By formula (4.3), h_1 and h_2 should be less than 0.025. If we use equal mesh spacing in arc length, then the number of discretization points must be greater than or equal to 256. We can verify this by computing the trajectory of the centers and the axis angles with 256, 128, 64, 32 discretization points on each cylinder until $t = 16$. The relative differences with $N = 256$ are shown in Table 4.7.

N	$\max(\lvert\frac{\Delta x}{x}\rvert)$	$\max(\lvert\frac{\Delta y}{y}\rvert)$	$\max(\lvert\frac{\Delta\theta}{\theta}\rvert)$
128	$4.324851693 \times 10^{-4}$	$7.689614014 \times 10^{-3}$	$6.783282906 \times 10^{-4}$
64	$7.120146271 \times 10^{-3}$	$1.253204325 \times 10^{-1}$	$1.115733623 \times 10^{-2}$
32	$7.157086083 \times 10^{-2}$	$4.471192934 \times 10^{-1}$	$9.659807766 \times 10^{-1}$

Table 4.7: *The difference in trajectories in comparison with $N = 256$. The ODE package is used with $\Delta t = 0.1$ and integration continues until $t = 16$.*

Figure 4.5 shows that the maximum differences occur when the distance between the particles becomes near its minimum.

The trajectories of the particles are computed by integrating (2.6) until $t = 32$ with the ODE code. The relative error tolerance is set to be 10^{-4}, and each particle is discretized with 256 points and the time step is $\Delta t = 0.1$. The particle positions at 12 different times are shown in Figure 4.6. The total CPU time needed to reach $t = 32$ is approximately 14 hours.

As can be seen in Figure 4.6, under the action of the hydrodynamic forces, each particle rotates about its center and moves to the right. Because their initial separation is close, the two particles tend to rotate as a single body, held together by the lubrication forces of their interaction [13]. This motion is periodic, as shown in Figure 4.7, with the time of the period being approximately 28.

When ε is larger, the interaction between the two particles is weaker, and their motion is qualitatively different. For instance, we use the same example as above, but the particles are initially located at $(0, 1.5)$ and $(0, 3.8)$, so $\varepsilon = 0.3$. According to (4.3), each particle is discretized with 128 points. The same time step $\Delta t = 0.1$ and relative error tolerance of 10^{-4} are used. The particle positions at five different times are shown in Figure 4.9. For this example, the total CPU time is approximately 2 hours. From Figure 4.9, it is clear to see that the particles move apart from each other. This tendency is shown by the trajectories of their centres in Figure 4.8.

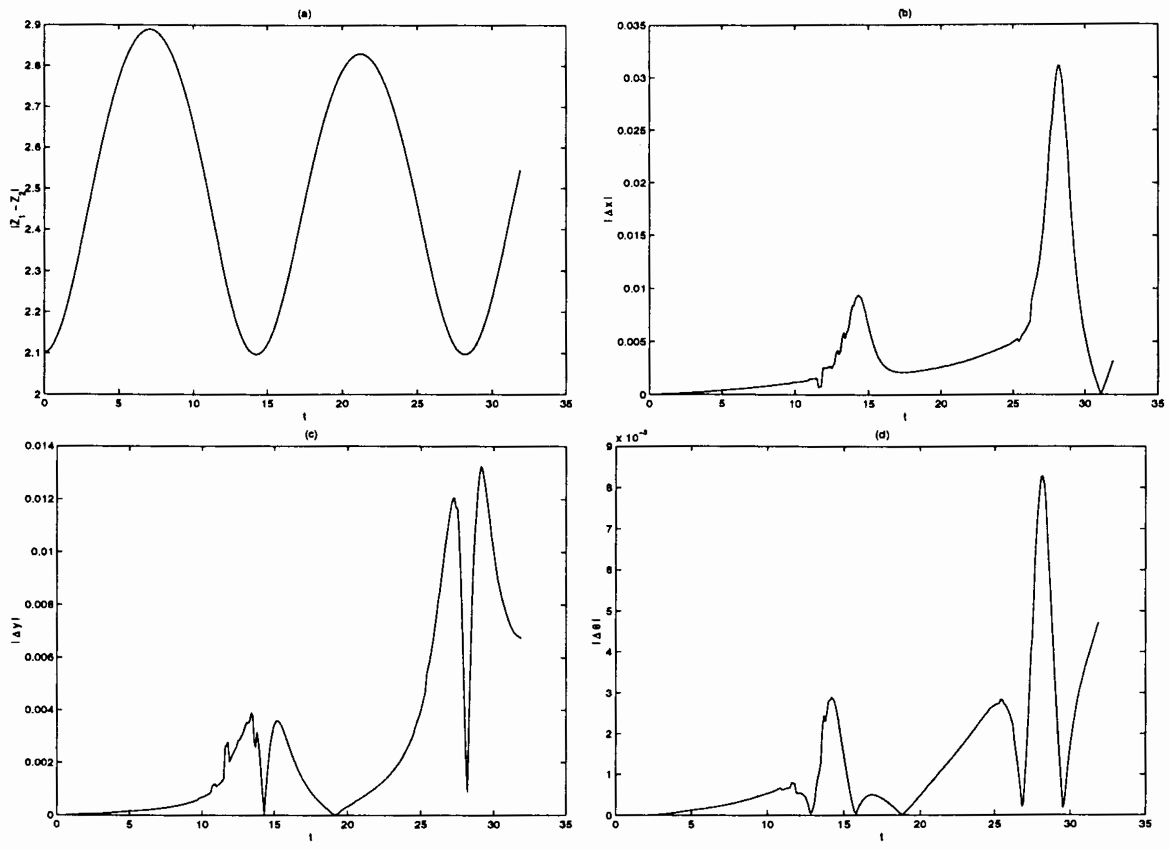

Figure 4.5: The differences of trajectory computed with $N = 256$ and $N = 128$. ODE is used with time step $\Delta t = 0.1$: (a) the distance between two centers of the circles from $t = 0$ to $t = 32$, (b) the difference of x-axis distance, (c) the difference of y-axis distance, (d) the difference of the axis angle.

An important factor in selecting a time-integration scheme is to determine whether or not the problem under consideration is stiff. Generally, the system (3.4) is considered stiff if the Jacobian matrix

$$J = \left(\frac{\partial f_i}{\partial y_j}\right)$$

has all negative real part eigenvalues, and the eigenvalues satisfy [14, 16]

$$0 < \min(|\mathrm{Re}(\lambda_j)|) << \max(|\mathrm{Re}(\lambda_j)|).$$

At a time t, from the position of a particle (x, y, θ), the velocities (u, v, Ω) can be

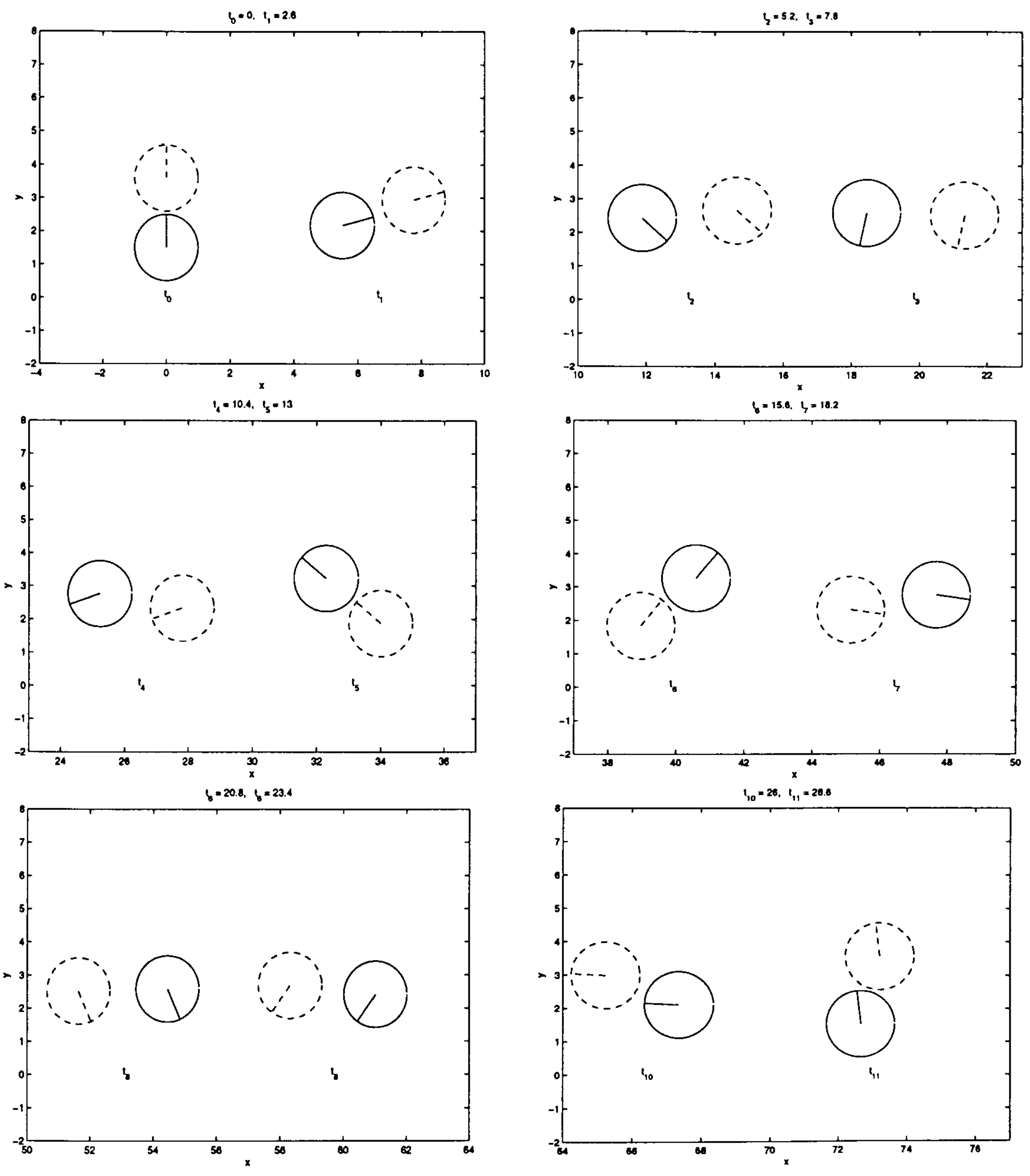

Figure 4.6: The trajectories of two cylindrical particles in shear flow. ODE is used with time step $\Delta t = 0.1$. $\varepsilon = 0.1, N = 256$.

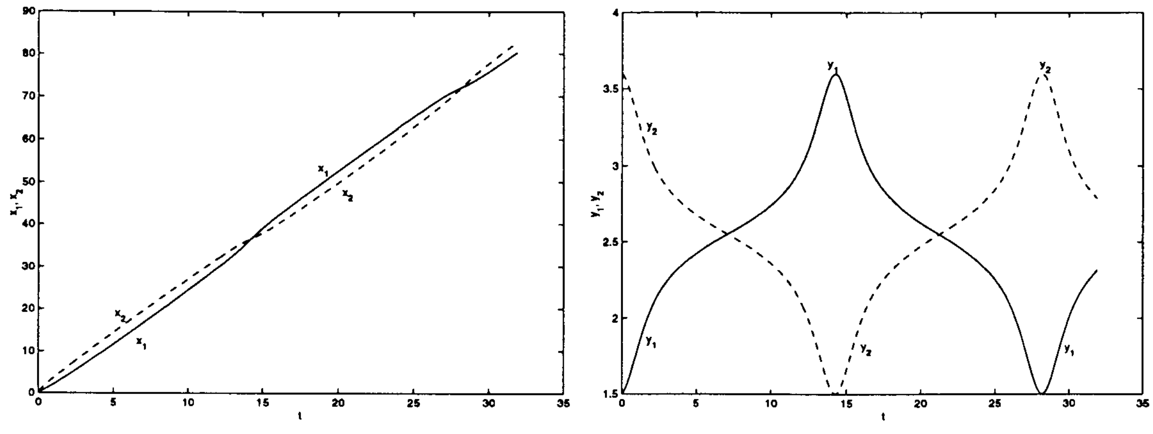

Figure 4.7: The trajectories of two cylindrical particles in shear flow in x-axis and y-axis. ODE is used with time step $\Delta t = 0.1$. $\varepsilon = 0.1, N = 256$.

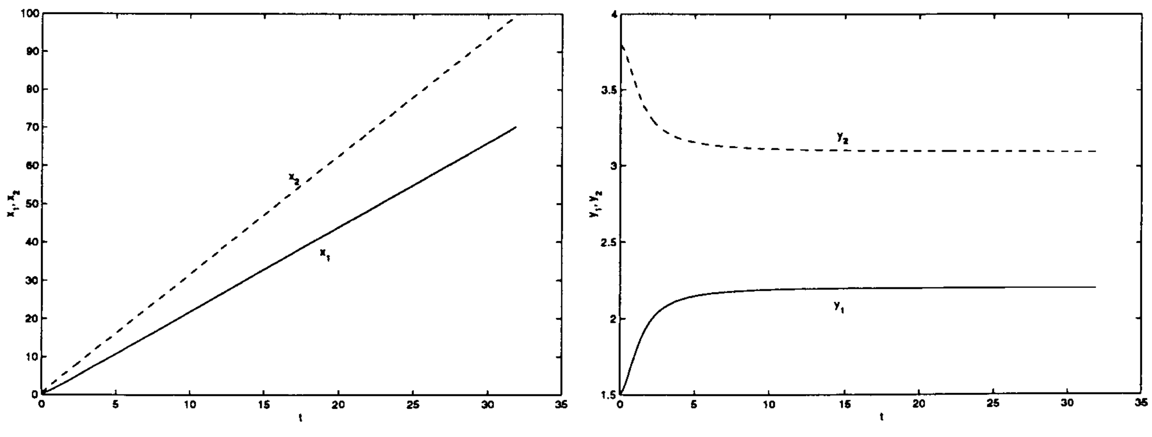

Figure 4.8: The trajectories of two cylindrical particles in shear flow in x-axis and y-axis. ODE is used with time step $\Delta t = 0.1$. $\varepsilon = 0.3, N = 256$.

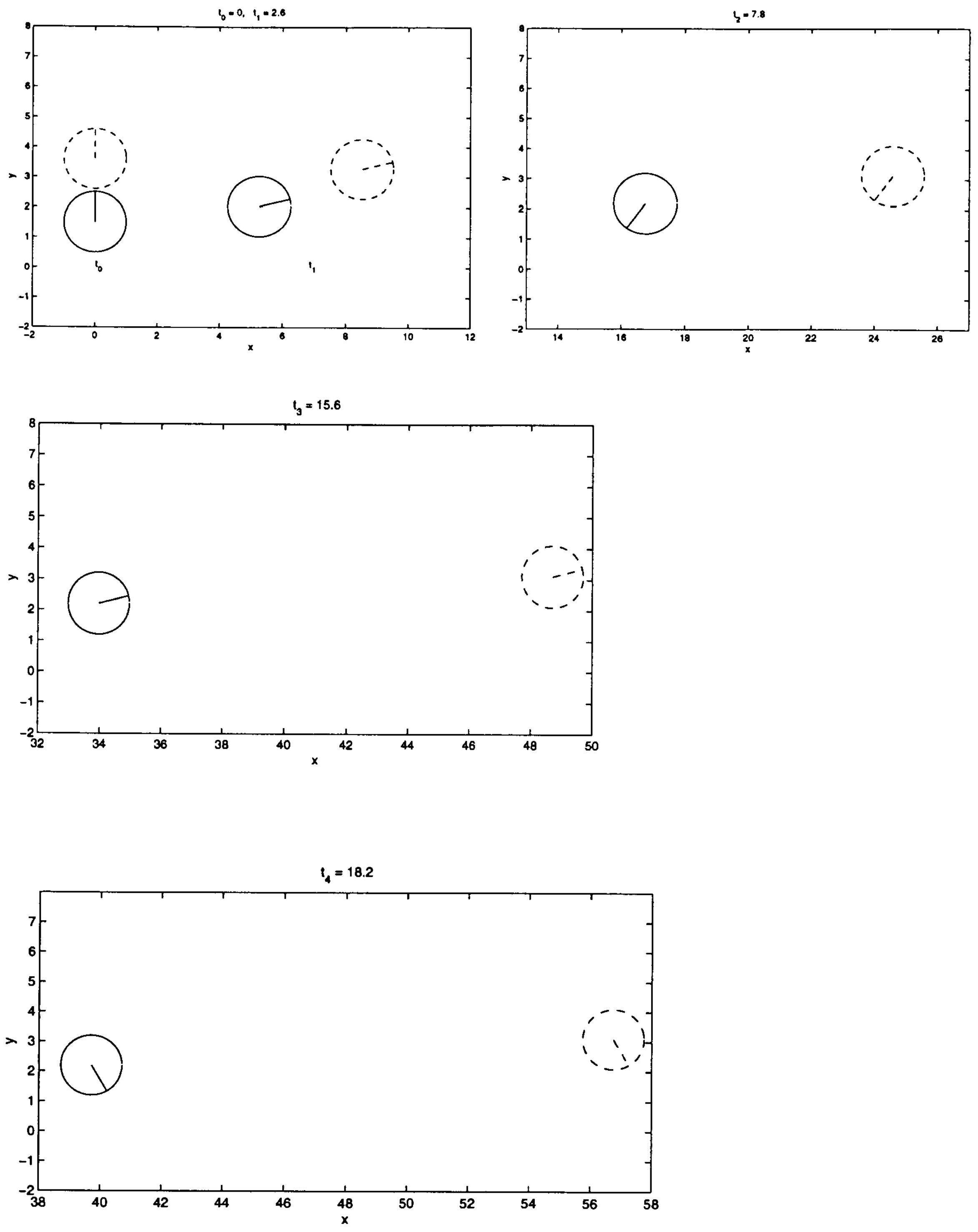

Figure 4.9: The trajectories of two cylindrical particles in shear flow. ODE is used with time step $\Delta t = 0.1$. $\varepsilon = 0.3, N = 128$.

found using the integral equation method. Now by deviating the position element x with a small value δ, the numerical derivatives in the Jacobian matrix

$$\frac{\partial u}{\partial x} \approx \frac{\tilde{u}-u}{\delta}, \quad \frac{\partial v}{\partial x} \approx \frac{\tilde{v}-v}{\delta}, \quad \frac{\partial \Omega}{\partial x} \approx \frac{\tilde{\Omega}-\Omega}{\delta},$$

can be computed. Similarly, we can differentiate with respect to y and θ to compute the Jacobian matrix, from which the eigenvalues can be obtained.

For instance, this approach can be used to investigate the problem discussed in this section. Tables 4.8 and 4.9 list the maximum and minimum real parts of the eigenvalues for the first and second problem at $t = 0, t = 5, t = 10$, and $t = 15$.

time	$\max(\mathrm{Re}(\lambda_j))$	$\min(\mathrm{Re}(\lambda_j))$
0	$4.302616747 \times 10^{-5}$	$-4.322004377 \times 10^{-5}$
5	$2.479031009 \times 10^{-1}$	$-2.330991216 \times 10^{-1}$
10	$1.260935172 \times 10^{-1}$	$-1.245055093 \times 10^{-1}$
15	$9.456182806 \times 10^{-2}$	$-9.416972676 \times 10^{-2}$

Table 4.8: *The maximum and minimum values of the eigenvalues for Jacobian matrix at various time.* $\varepsilon = 0.1$, *and* $\delta = 10^{-6}$.

time	$\max(\mathrm{Re}(\lambda_j))$	$\min(\mathrm{Re}(\lambda_j))$
0	$1.213992448 \times 10^{-6}$	$-2.433470114 \times 10^{-6}$
5	$1.339177877 \times 10^{-1}$	$-1.314937713 \times 10^{-1}$
10	$3.997479799 \times 10^{-2}$	$-3.997479799 \times 10^{-2}$
15	$1.884250079 \times 10^{-2}$	$-1.882483389 \times 10^{-2}$

Table 4.9: *The maximum and minimum values of the eigenvalues for Jacobian matrix at various time.* $\varepsilon = 0.3$, *and* $\delta = 10^{-6}$.

Since the real part of the eigenvalues are not all negative, it can be concluded that the problems under consideration are nonstiff.

Our results showed that the ODE package coupled with the integral equation methods worked well with the problems. But for the Forward Euler's method, the first order method, only a smaller stepsize can give a reasonable result. A bigger stepsize leads to a wrong solution (see Figure 4.10).

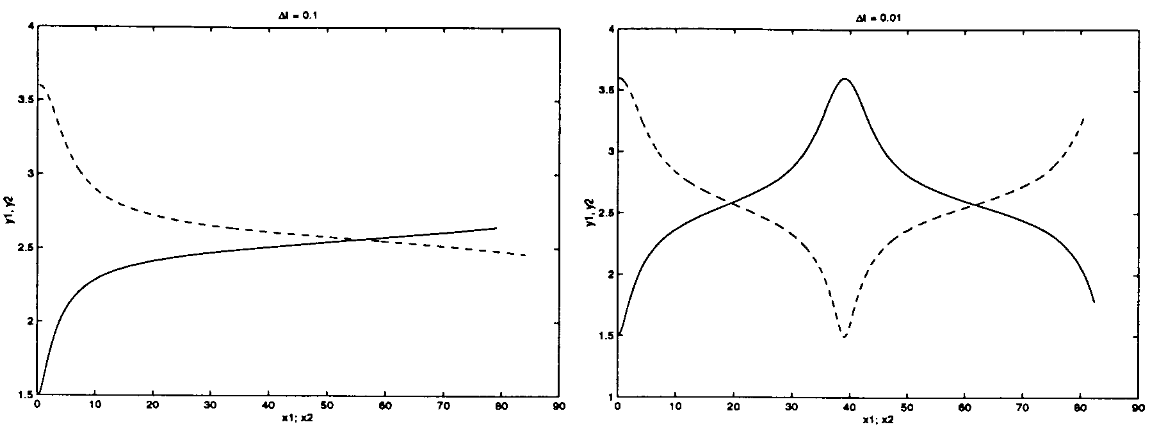

Figure 4.10: The comparison of trajectories of two cylindrical particles in shear flow in x-axis and y-axis by FE with time step $\Delta t = 0.1$ and $\Delta t = 0.01$. $\varepsilon = 0.1, N = 256$.

Chapter 5

Concluding Remarks

We have investigated numerical methods for computing the motion of neutrally-buoyant solid particles in a Stokes flow. The solid particles satisfy the Quasi-Static Approximation in low Reynolds number hydrodynamics: particles adjust their velocity instantaneously to maintain a force-free configuration, and their motion is computed as a sequence of steady-state solutions to the Stokes equations.

In this thesis, the performance of several time-integration schemes is tested for integrating the initial value problem for computing the particle trajectories. We use the Forward and Backward Euler's methods, the Runge-Kutta schemes (implementation by RKSUITE package) and Adams-Pece formulae (by ODE package). The ODE package achieved, by several orders of magnitude, the highest level of accuracy. ODE uses variable-order schemes, this result is consistent with the conclusion by T. E. Hull *et al.* [11] that variable-order methods are generally better than those whose orders are fixed. RKSUITE provides three different order pairs, but still uses a fixed order when the pair is set. RKSUITE can achieve high accuracy if an appropriate relative error tolerance is selected, however it is computationally expensive. When only moderate to low accuracy is required, this code is significantly faster than others.

The hydrodynamic interactions among solid particles in a shear flow is investigated in Section 4.2, and it was shown that these problems are not stiff. Our results show that the integral equation method discussed in Chapter 3 coupled with the ODE solver works well for small simulations. However, when the particles come into close contact or the particles have complicated shape, refined mesh spacing is needed.

Searching for more efficient time-integration schemes for particle simulation is worthy of further study. Another area of future research is to develop an appropriate mesh-adaption scheme for more complex simulation problems.

Bibliography

[1] John F. Brady and Georges Bossis (1988), *Stokesian dynamics*, Ann. Rev. Fluid Mech. **20**, pp. 111-157.

[2] R. W. Brankin, I. Gladwell, and L. F. Shampine (1991), *RKSUITE: a Suite of Runge-Kutta Codes for the Initial Value Problem for ODEs*, Softreport 91-1, Math. Dept., Southern Methodist University,, Dallas, Texas, U.S.A.

[3] R. L. Burden and J. D. Faires (1989), *Numerical Analysis*, PWS-KENT Publishing Co., Boston, 4th Edition.

[4] V. N. Constantinescu (1995), *Laminar Viscous Flow*, Springer-Verlag, New York.

[5] J. Carrier, L. Greengard, and V. Rokhlin (1988), *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., **9**, pp. 669-686.

[6] J. Feng and D. D. Joseph (1995), *The unsteady motion of solid bodies in creeping flows*, J. Fluid Mech., **303**, pp. 88-102.

[7] P. Ganatos, R. Pfeffer, and S. Weinbaum (1978), *A numerical-solution technique for three-dimensional Stokes flows, with application to the motion of strongly interacting spheres in the plane*, J. Fluid Mech., **84**, pp. 79-111.

[8] L. Greengard (1988), *The Rapid Evaluation of Potential Field Particle Systems*, MIT press, Cambridge.

[9] L. Greengard, M. C. Kropinski and A. Mayo (1996), *Integral equation method for Stokes flow and isotropic elasticity in the plane*, J. Comp. Phys., **125**, 403-414.

[10] J. Happel and H. Brenner (1983), *Low Reynolds Number Hydrodynamics*, Kluwer Academic Publishers, Dordrecht.

[11] T. E. Hull, W. H. Enright, B. M. Fellen and A. E. Sedgwick (1972), *Comparing numerical methods for ordinary differential equations*, SIAM J. Numer. Anal., **9**, pp. 603-637.

[12] G. B. Jeffrey (1922), *The motion of ellipsoidal particles immersed in a viscous fluid*, Proc. R. Soc. Lond. A., **8**, pp. 161-179.

[13] M. C. A. Kropinski, *Integral equation methods for particle simulations in creeping flows*, Submitted to Journal of Fluid Mechanics.

[14] J. D. Lambert (1991), *Numerical Methods for Ordinary Differential Systems*, John Wiley and Sons, Chichest.

[15] W. E. Langlois (1964), *Slow Viscous Flow*, MacMillan, New York.

[16] L. Lapidus and J. H. Seinfeld (1971), *Numerical Solution of Ordinary Differential Equations*, Academic Press, New York and London.

[17] S. G. Mikhlin (1957), *Integral Equations*, Pergamon Press, London.

[18] S. G. Muskhelishvili (1953), *Some Basic Problems of the Mathematical Theory of Elasticity*, P. Noordhoff, Groningen.

[19] V. Z. Parton and P. I. Perlin (1982), *Integral Equation Methods in Elasticity*, MIR, Moscow.

[20] H. Power (1993), *The completed double layer boundary integral equation method for two-dimensional Stokes flow*, IMA J. Appl. Math., **51**, pp. 123-145.

[21] W. B. Russel, D. A. Savilee and W. R. Schowalter (1989), *Colloidal Dispersions*, Cambridge University Press, Cambridge.

[22] L. F. Shampine (1994), *Numerical Solution of Ordinary Differential Equations*, Chapman and Hall, New York and London.

[23] L. F. Shampine (1984), *Stiffness and the automatic selection of ODE codes*, J. Comp. Phys. **54**, 74-86.

[24] L. F. Shampine and M. K. Gordon (1975), *Computer Solution of Ordinary Differential Equation: The Initial Value Problem*, W. H. Freeman and Company, San Francisco.

[25] F. S. Sherman (1990), *Viscous Flow*, McGraw-Hill, New York.

[26] I. S. Sokolnikoff (1956), *Mathematical Theory of Elasticity*, McGraw-Hill, New York.

[27] S. Weinbaum, P. Ganatos and Z. Yan (1990), *Numerical multipole and boundary integral equation techniques in Stokes flow*, Annu. Rev. Fluid Mech. **22**, pp. 275-316