JOURNAL OF COMPUTATIONAL PHYSICS 120, 1–14 (1995)

A BEM-BDF Scheme for Curvature Driven Moving Stokes Flows

G. A. L. VAN DE VORST AND R. M. M. MATTHEIJ*

Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Received May 31, 1994

A backward differences formulae (BDF) scheme, is proposed to simulate the deformation of a viscous incompressible Newtonian fluid domain in time, which is driven solely by the boundary curvature. The boundary velocity field of the fluid domain is obtained by writing the governing Stokes equations in terms of an integral formulation that is solved by a boundary element method. The motion of the boundary is modelled by considering the boundary curve as material points. The trajectories of those points are followed by applying the Lagrangian representation for the velocity. Substituting this representation into the discretized version of the integral equation yields a system of non-linear ODEs. Here the numerical integration of this system of ODEs is outlined. It is shown that, depending on the geometrical shape, the system can be *stiff*. Hence, a BDF-scheme is applied to solve those equations. Some important features with respect to the numerical implementation of this method are high-lighted, like the approximation of the Jacobian matrix and the continuation of integration after a mesh redistribution. The usefulness of the method for both two-dimensional and axisymmetric problems is demonstrated.

1. INTRODUCTION

In the last decade, the usage of integral formulations for the numerical solution of Stokes flows with moving boundaries has become a well-known and powerful technique. The method consists of rewriting the problem in terms of an integral equation based on so-called *hydrodynamical single- and double-layer potentials*. The motion of the boundary is described by a kinematic constraint that relates the change of the boundary curve to the velocity at the curve; i.e., a *quasi-static* approach is employed to model the movement.

To date, most work has been done on axisymmetric and two-dimensional flow problems which ranges from the study of rising (deformable) drops towards an interface, to problems such as the deformation of cells and the deformation of a liquid film along a wall. Studies of three-dimensional arbitrarily shaped surface deformations are still in their early stages and only modest distortions have been computed. More about the application of this solution technique for free-creeping Stokes flows can be found in the reviews of Tanzosh *et al.* [17] and Stone [16]. The recent books of both Kim and Karrila [12] and Pozrikidis [14] extensively outline the theoretical derivation and the practical application of such formulations. A general review of the state-of-the-art of numerical solution methods that can be used for viscous flows with moving boundaries are to be found in Floryan and Rasmussen [5].

The governing integral formulation is solved numerically by applying a boundary element method (BEM); cf. Brebbia *et al.* [3] or Becker [2]. This yields the boundary velocity field of the fluid domain at a fixed time. Then a time step has to be performed to obtain the boundary deformation at a next time level. The difficulties which have to be taken care of during the evolution of such a fluid domain are the following: (1) the boundary condition involves the computation of the curvature in order to describe surface tension effects; this requires an accurate approximation of the curvature. Furthermore, the boundary may undergo large distortions during the deformation; this causes (2) the collocation points to become unevenly distributed. This will lead to numerical inaccuracies in the computed boundary conditions as well as the unknowns. Moreover, (3) non-analytic cusp-like curves may evolve and (4) the connectivity of the domain can change by breaking up or touching of boundaries.

As mentioned before, the boundary movement is modelled by a kinematic constraint. Generally, three kinds of constraints are distinguished, viz., from a so-called Eulerian, Lagrangian, or the mixed Eulerian–Lagrangian point of view.

In the Eulerian viewpoint, the collocation points at the boundary move in the direction normal to the boundary and so the normal component of the boundary velocity field is used. An advantage of this approach is that the nodal points tend to remain evenly distributed for a smooth boundary. However, in the neighbourhood of cusp-like regions the collocation points need to be redistributed since otherwise these nodes come to close to each other leading to numerical instabilities. A disadvantage of this approach is the difficulty of the implementation of higher order time integration schemes for the governing kinematic constraint. Therefore, only a simple forward Euler method is employed to obtain the next time level geometry.

In the Lagrangian viewpoint, the nodes are considered to be material points and moved according to the actual boundary velocity; i.e., the characteristic curves are followed. Actually, one obtains a system of nonlinear ODEs which can be solved

* E-mail: wsangv@win.tue.nl; wstanw10@win.tue.nl.

0021-9991/95 $12.00

easily by employing a multistep integrator. However, in literature, usually only the Forward Euler scheme seems to have been applied for solving these ODEs. Yet, some higher order explicit time integrators were implemented by Longuet-Higgins and Cokelet [13], Haack *et al.* [8], for example; they applied an Adams–Bashforth–Moulton method. A Runge Kutta scheme was applied by Ramsden and Holloway [15]. When the system of ODEs appears to be stiff such explicit methods will not be very efficient, however. The disadvantage of the Lagrangian description is that nodal points seem to move quite a bit along the boundary even for small deformations (in a certain time interval). Hence, the collocation points have to be redistributed frequently.

In the mixed Eulerian–Lagrangian viewpoint the path line of a fluid particle is followed by using a higher order Taylor expansion in terms of the material time derivative. This explicit time stepping method has not been applied yet in the case of Stokes flows. However, it has been used successfully in the simulation of water waves when the velocity field can be modelled by Laplace's equation; see, for instance, Grilli *et al.* [7] and Cooker *et al.* [4]. In order to obtain the coefficients of the Taylor expansion in the latter problem, one has to solve a successive number of Laplace problems for the velocity potential and its time derivatives (depending on the number of terms applied).

In this paper we present a numerical integration technique to solve the motion of a certain class of Stokes flow problems, viz., fluids driven solely by the boundary curvature. Such problems arise, for example, by modelling the deformation of glass heated to a sufficiently high temperature such that it becomes a viscous fluid. In particular this phenomenon occurs during the process called *viscous sintering* (cf. Van de Vorst *et al.* [18, 20] and Van de Vorst [21, 22], and Subsection 6.3). To obtain a kinematic constraint we use the Lagrangian viewpoint, since this enables the application of higher order time integration schemes. In order to keep the nodal points evenly distributed, an efficient node redistribution scheme was developed (Van de Vorst and Mattheij [20]).

After substituting the Lagrangian representation for the velocity of the collocation points into the system of algebraic equations derived by the BEM, we obtain a system of nonlinear ODEs. It turns out that for most of the geometrical shapes under consideration, the system of ODEs is *stiff*. Because of this, the time step will be carried out by a more sophisticated time integrator: a variable step, variable order backward differences formulae (BDF) scheme. Thus, in contrast to all earlier mentioned studies, we will use an *implicit* multistep method. The numerical implementation of this integration method will be outlined in Sections 4 and 5. In particular, we will highlight some important features like the approximation of the Jacobian matrix and the continuation of integration after a redistribution of the collocation points. Finally, the usefulness of the proposed numerical scheme is demonstrated for some typical geometries in Section 6. We start with a derivation of the mathematical model in the next section.

2. MATHEMATICAL FORMULATION

We consider a two-dimensional, incompressible Newtonian viscous fluid domain, which is surrounded by a "smooth" boundary curve, say Γ. The motion of the fluid is governed by the following: first we require conservation of momentum, i.e., the *Stokes' equation* (equation of motion) applies, which in dimensionless form reduces to

$$\Delta \mathbf{v} - \operatorname{grad} p = 0, \tag{1}$$

where $\mathbf{v}$ denotes the velocity and p is the pressure of the fluid. Second, the fluid is assumed to be incompressible, i.e.,

$$\operatorname{div} \mathbf{v} = 0, \tag{2}$$

which expresses the *convervation of mass.* Since we assume the fluid to be Newtonian, this yields the following constitutive equation for the *stress tensor* $\mathcal{T}$,

$$\mathcal{T}_{ij} = -p\delta_{ij} + \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{3}$$

Moreover, the surface tension, say $\mathbf{b}$, is proportional to the boundary curvature in the normal direction,

$$b_j = -\kappa\, n_j, \tag{4}$$

where $\mathbf{n}$ is the outer normal and κ denotes the boundary curvature.

The time-dependence does not appear explicitly in the above equations. Because of this we use a *kinematic constraint* that describes the motion of the fluid domain (quasi-static approach). Here the movement of the boundary is modelled by considering the boundary curve as a set of material points. Then the trajectories of those points can be followed by using the Lagrangian representation of the velocity. In particular, for each collocation point it holds that

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}). \tag{5}$$

Since we are only interested in the deformation of the fluid domain, the Stokes equations are transformed into an integral equation over the boundary. Then the unknown variable is the boundary velocity only. A Fredholm integral formulation of the second kind can be deduced that relates the velocity of a point $\mathbf{x}$ on Γ to the applied surface tension (cf. Pozrikidis [14]). In dimensionless form this equation reads

$$\frac{1}{2}v_j(\mathbf{x}) + \int_\Gamma q_{ij}(\mathbf{x}-\mathbf{y})v_j(\mathbf{y})\,d\Gamma_y = \int_\Gamma u_{ij}(\mathbf{x}-\mathbf{y})b_j(\mathbf{y})\,d\Gamma_y. \tag{6}$$

Note that, for the sake of simplicity, we have written down the basic formulation, viz., when the fluid region is a simply connected domain. Moreover, we have ignored the terms originated from deflating the integral operator which are required to obtain a fully determined problem (cf. Van de Vorst [21]). In Eq. (6) the coefficients q_{ij}, u_{ij} are equal to respectively

$$q_{ij}(\mathbf{r}) = \frac{r_i r_j}{\pi R^4} r_k n_k,$$
$$u_{ij}(\mathbf{r}) = \frac{1}{4\pi}\left[-\delta_{ij}\log R + \frac{r_i r_j}{R^2}\right], \tag{7}$$

where $r_i = x_i - y_i$, $R = \sqrt{r_1^2 + r_2^2} = |\mathbf{x} - \mathbf{y}|$.

The integral equation (6) is solved by applying a BEM. Hence the boundary is discretized into a set of nodal points, say N; the boundary curve Γ is replaced by a polygon through these nodal points. Moreover, the integral formulation is enforced on the polygon for each of the collocation points. This results into a square full rank system of 2N linear algebraic equations with a 2N unknowns which will be denoted by

$$\mathcal{H}(\mathbf{x})\ \mathbf{v} = \mathcal{G}(\mathbf{x})\ \mathbf{b}\ (\mathbf{x}), \tag{8}$$

where $\mathbf{x}$ is a vector of length 2N that consists of all successive collocation points, whereas the vectors $\mathbf{v}$ and $\mathbf{b}$ represents the corresponding boundary velocity and tension, respectively. The element matrix coefficients of the matrices $\mathcal{G}$ and $\mathcal{H}$ typically consist of the following integrals (Van de Vorst *et al.* [18]),

$$\int_{-1}^{1} \phi_k(s) u_{mn}(\mathbf{x}^p - \mathbf{y}(s))\,|\mathbf{y}'(s)|\,ds;$$
$$\int_{-1}^{1} \phi_k(s) q_{mn}(\mathbf{x}^p - \mathbf{y}(s))|\mathbf{y}'(s)|\,ds, \tag{9}$$

where $'$ denotes the derivative with respect to s; $\mathbf{y}(s)$ is the interpolant of the element boundary; i.e., $\mathbf{y}(s) = \phi_k(s)\mathbf{y}^k$ and $\mathbf{y}^k$ are the element nodes. Here $\phi_k(s)$ $(k = 1, \ldots, M+1)$ are the Lagrangian finite element type polynomials where the degree of the polynomial approximation is equal to M.

After solving the system (8) we have to perform a time step. Using the kinematic constraint (5) for each collocation point together with Eq. (8), yields the following 2N non-linear system of ODEs,

$$\frac{d\mathbf{x}}{dt} = \mathcal{H}(\mathbf{x})\mathcal{G}(\mathbf{x})\mathbf{b}(\mathbf{x}). \tag{10}$$

In the next sections we will consider the numerical integration of the above system.

3. STIFFNESS

There are various definitions of "stiffness" in the literature. Before defining the concept of stiffness used here, we first introduce some notation. Let $\lambda_i = \lambda_i(\mathbf{x})$ be an eigenvalue of the Jacobian matrix $\mathcal{J}(\mathbf{x})$ of the system (10), where

$$\mathcal{J}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}}(\mathcal{H}^{-1}(\mathbf{x})\mathcal{G}(\mathbf{x})\mathbf{b}(\mathbf{x})), \tag{11}$$

which is taken at relevant nodal points $\mathbf{x}$. Furthermore, let ρ denote the spectral radius of $\mathcal{J}$; i.e.,

$$\rho = \max_{i=1}^{2N} |\lambda_i|.$$

Then, we will call the system of ODEs (10) *stiff* on an interval $[a, a + T]$ if

$$\max_{t\in[a,a+T]} \rho(\mathcal{J}(\mathbf{x}(t)))\,T \gg 1, \tag{12}$$

(cf. Ascher *et al.* [1]).

From a physical point of view, an arbitrary fluid geometry will deform to a steady state as time increases. In particular for the case of a curvature driven two-dimensional flow, the domain transforms itself into a circular disk with an area that remains constant during the deformation. This because a circle minimizes the curve length of the boundary that surrounds a two-dimensional region.

According to definition (12), the system of ODEs (10) will be stiff on an interval if the spectral radius ρ is large and the interval length is substantially larger than $1/\rho$. It is impossible to derive an analytical expression for the Jacobian (11) or the spectral radius in this particular case. Because of this fact, the appearance of stiffness is demonstrated by a couple of simple but typical examples that represent the basic evolution features.

First, we consider the evolution of the coalescence of two equal cylinders that already make a line contact with each other initially. These coalescing cylinders demonstrate the deformation phenomena very well. In the early stage of the coalescence, the boundary curvature is very large in the region where both cylinders are touching (almost a *cusp*); in later stages the shape is becoming "smoother," i.e., the curvature is only moderately varying everywhere.

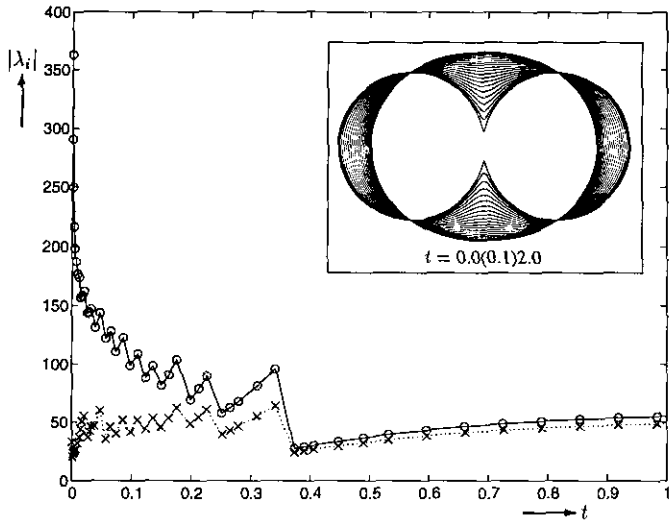


FIG. 1. The spectral radius ρ development (solid line) compared to the absolute eigenvalue second in magnitude (dotted line) of the numerically obtained (exact) Jacobian of two equal coalescing cylinders is showing stiffness in the initial stage of the coalescence. The jumps in the spectral radius are caused by node redistributions.

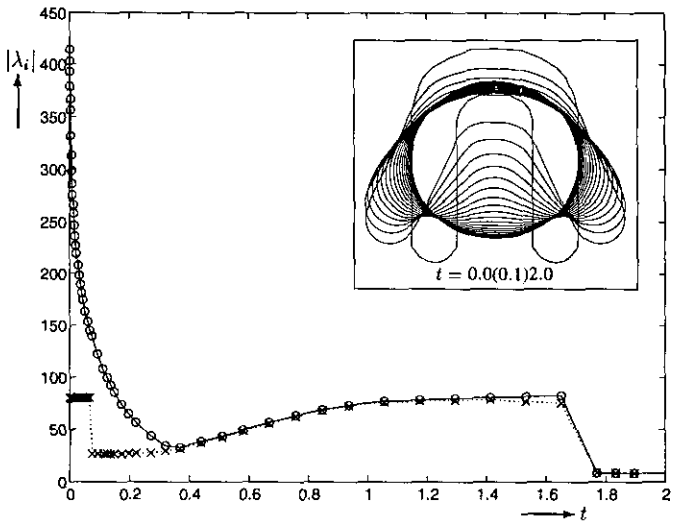


FIG. 2. The spectral radius ρ development (solid line) compared to the second largest absolute eigenvalue (dotted line) of the Jacobian of an n-shaped region is showing that the ODEs are stiff when larger parts of the shape are moving fast (during the initial stage).

In Fig. 1 we have plotted the spectral radius (solid line) of the numerically obtained (exact) Jacobian computed after every successful integration time step, when the fluid is transforming itself into a circle as time evolves. The spectral radius is compared to the eigenvalue second in magnitude (dotted line). From the difference in size at the initial stage of the coalescence, we conclude that there are smoother modes (of the linearized problem) that ask for larger steps on an interval that is substantially larger than $\mathbb{O}(\rho^{-1})$; hence the ODEs are *stiff* in that region. Here, the problem was solved using *linear* boundary elements (the jumps in the spectral radius are caused by the node redistribution algorithm). After performing such a redistribution, the trajectories of a different set of particles may be being followed. As can be seen from the figure, the stiffness of the ODEs can change drastically. From this figure, we also observe that as time increases the stiffness is disappearing, because the boundary is almost becoming a circle. This can be concluded from the time step that is used during later stages; apparently both eigenvalues are not relevant for the evolution anymore.

However, the ODEs for a smooth curved geometry can be stiff too. This occurs when larger parts of the shape are moving fast. To illustrate this behaviour we show an n-shaped region in Fig. 2. The development of the two largest (absolute) eigenvalues of the Jacobian are also plotted. Again, we observe a large difference between these two values during the initial time stage when the shape is stretching itself. Later, the magnitude of these eigenvalues become very close (and to the rest as well).

Another typical phenomenon is the shrinkage and vanishing of holes inside the fluid region. In order to investigate whether stiffness is present in such a type of problem we consider the most simple example that covers all effects: the shrinkage of a circular annulus. For this example we have taken the initial outer radius equal to 1 and the initial inner radius equal to 0.5. In Fig. 3 we show the spectral radius development (solid line) and the second largest eigenvalue (dotted line of the Jacobian computed after every time step for this particular simulation:

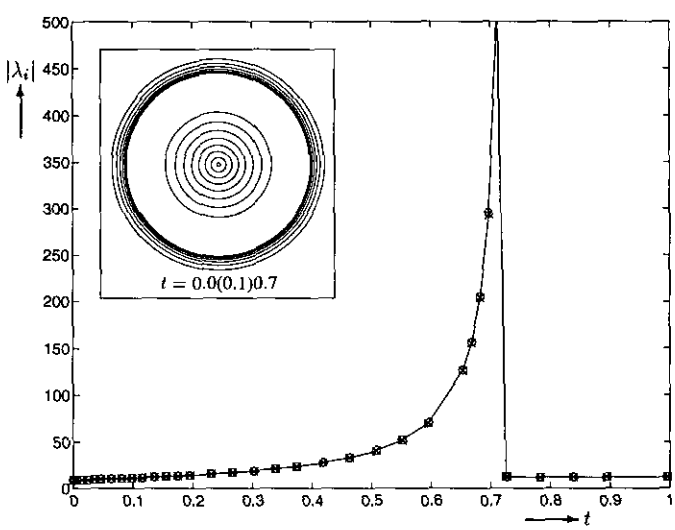


FIG. 3. The two largest eigenvalues of the Jacobian during the shrinkage of a circular annulus are almost identical as time evolves. The increase of both values as the inside hole is nearly vanished is due to the curvature increase of the hole. The vanishing of the hole can still be called *stiff* since both the boundaries evolute at different time scales.

both values remain almost identical as time evolves. The large increase of the values when the inside hole has nearly vanished is due to the curvature of the hole boundary since this is asymptotically tending to infinity. We observed that there were more such larger eigenvalues (depending on the number of nodes that discretized the hole). As can be obtained from the exact analytical solution of this problem (cf. Van de Vorst [21]), the hole is completely vanished at a finite time, viz., $t = \sqrt{3} - 1$. There is not any activity of the fluid afterwards (see Fig. 3). Near to the vanishing of such a hole again *stiffness* occurs as there are two time scales: the evolution of the inside hole curve and the outer boundary curve.

As the examples above illustrate, the ODEs (10) which have to be integrated can have widely differing time constants. Therefore, we have used the variable step, variable order BDF-method, as is implemented in the solver LSODE; cf. Hindmarsh [9–10], for obtaining the solution of the examples above. We also observed from those simulations, that nearly all the eigenvalues are real and negative; only a few eigenvalues have a small imaginary part (order 10^{-3}). Because of this, we do not expect to have any efficiency problems for the higher order BDF-methods, since those methods tend to take very small steps in order to provide stability when the Jacobian has eigenvalues with large imaginary parts but small negative real parts; cf. Hindmarsh [11].

4. APPROXIMATION OF THE JACOBIAN

Since the BDF-method is an implicit linear multistep method, the corrector equation has to be solved by some iteration method. In LSODE this is performed by applying a modified Newton iteration; cf. Hindmarsh [10]. This implies that the code requires the Jacobian (11) of the system of ODEs.

As we remarked in the previous section, it is practically impossible to derive an analytical expression for the Jacobian. A numerical approximation of the exact Jacobian (11) would be extremely time consuming: one Jacobian evaluation requires the assembling and solution of the system of equations 2N times. However, it is not necessary to have exact Jacobians anyway, since the BDF-solver is using a modified Newton method; i.e., the same Jacobian is used in subsequent (Newton) iterations and for several time integration steps. Therefore, we will use an approximate Jacobian. The derivation of this approximation is outlined for a simply connected domain in the remaining part of this section.

Denote by $\hat{\mathbf{x}}_{j,k}$ the vector of all boundary nodes where the nodal point "j" is perturbed by a small value, say ε ($\varepsilon \ll 1$) in the kth-direction ($k = 1, 2$), i.e.,

$$\hat{\mathbf{x}}_{j,k} = \mathbf{x} + \varepsilon \mathbf{e}_{2j-k} = (x_1^1, x_2^1, \ldots, x_1^j + \varepsilon, x_2^j, \ldots, x_1^N, x_2^N)^T.$$

Furthermore, assume that $\hat{\mathbf{v}}_{j,k}$ is the solution of the system (10) for this perturbed boundary, thus

$$\mathcal{H}(\hat{\mathbf{x}}_{j,k})\hat{\mathbf{v}}_{j,k} = \mathcal{G}(\hat{\mathbf{x}}_{j,k})\mathbf{b}(\hat{\mathbf{x}}_{j,k}).$$

By Taylor expansion in ε of these quantities, up to first order, we find

$$(\mathcal{H} + \varepsilon\delta\mathcal{H}_{j,k})(\mathbf{v} + \varepsilon\delta\mathbf{v}_{j,k}) \doteq (\mathcal{G} + \varepsilon\delta\mathcal{G}_{j,k})(\mathbf{b} + \varepsilon\delta\mathbf{b}_{j,k}). \tag{13}$$

Here, both $\delta\mathcal{H}_{j,k}$ and $\delta\mathcal{G}_{j,k}$ are sparse matrices containing derivative of the integrals (9) with respect to x_k^j. The non-zero elements of these matrices have row indices "$2j - 1$" and "$2j$" and column indices from "$2j - 1 - p$" to "$2j + p$," where p is equal to 2 in the case of linear elements or when node "j" is the mid-point of a quadratic element; p is equal to 4 when quadratic elements are applied and node "j" is one of the corners of the element. The vector $\delta\mathbf{b}_{j,k}$ has non-zero elements for the same indices as the columns of the above matrices in the case of a mid-point when quadratic elements are used. We remark that the vector $\delta\mathbf{v}_{j,k}$ approximates the $(2j - k)$th column of the Jacobian $\mathcal{J}$. Thus, using the exact solution (8) and omitting the higher order terms in (13), we obtain the following first-order approximation for the $(2j - k)$th column of $\mathcal{J}$,

$$\delta\mathbf{v}_{j,k} \doteq \mathcal{H}^{-1}(\mathcal{G}\,\delta\mathbf{b}_{j,k} + \delta\mathcal{G}_{j,k}\mathbf{b} - \delta\mathcal{H}_{j,k}\,\mathbf{v}). \tag{14}$$

The above Jacobian approximation is not expensive to compute, compared to the numerical exact Jacobian: when a new Jacobian evaluation is required, LSODE is asking for this Jacobian *after* a call which solves the system of Eqs. (8) for this boundary. Thus the matrix $\mathcal{G}$, the LU-decomposition of the matrix $\mathcal{H}$, and the vectors $\mathbf{b}$ and $\mathbf{v}$ are already available. And because of the sparsity of the derivative matrices $\delta\mathcal{H}_{j,k}$ and $\delta\mathcal{G}_{j,k}$, and the vector $\delta\mathbf{b}_{j,k}$, the computational costs to approximate the total Jacobian $\mathcal{J}$ will be of the order of four times the costs of assembling the system of equations where we note that the system assembling is the most expensive part of the solution process. Furthermore, we have to perform a forward and backward substitution of 2N right-hand sides with respect to $\mathcal{H}$ ($\sim 8N^3$). Therefore, we are still not satisfied with the required computational effort to obtain this approximate Jacobian.

We note that the computing costs of that Jacobian will be reduced considerably when the terms $\delta\mathcal{G}_{j,k}\mathbf{b} - \delta\mathcal{H}_{j,k}\mathbf{v}$ may be omitted; besides 2N forward and backward substitutions of $\mathcal{G}\delta\mathbf{b}_{j,k}$, that the right-hand side will require six matrix-vector operations only ($\sim 16N^2$). Moreover, the computation of the Jacobian will become very simple too. Therefore, we briefly analyze and quantify the contributions of the different terms in the vector

$$\mathcal{G}\,\delta\mathbf{b}_{j,k} + \delta\mathcal{G}_{j,k}\mathbf{b} - \delta\mathcal{H}_{j,k}\mathbf{v} \tag{15}$$

in order to reduce the computational effort further.

The sparse matrices $\delta\mathcal{G}_{j,k}$ and $\delta\mathcal{H}_{j,k}$ consist of the derivative

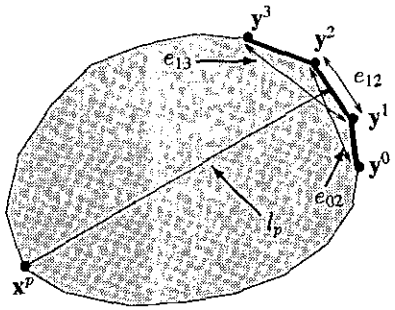


FIG. 4. Notation of defined lengths when the collocation point $\mathbf{x}^p$ is not a node of the considered linear element.

of the coefficient integrals (9) with respect to x_k^j, which can be distinguished as one of three different types. If row derivatives are considered then x_k^j is equal to a component of $\mathbf{x}$; in the case of a column derivative, x_k^j is equal to a coordinate of a nodal point from the boundary interpolant $\mathbf{y}(s)$; or at the intersection of a row and column, x_k^j is equal to both. The contribution of these various derivatives can be estimated. In Van de Vorst [22] estimates are derived for the case where linear elements are applied; here we only summarize the results of this analysis. The estimates which follow below are often also useful for the quadratic element case, since the interpolation functions will cause in the integrals minor changes only.

The contribution of the column integrals of the matrices involved in the vector (15), i.e., taking $x_k^j = y_k^j$, can be estimated by considering those integrals with respect to, say, $\mathbf{x}^p$ for both the element defined by the collocation points, say, $\mathbf{y}^1$ and $\mathbf{y}^2$, and the two neighbouring elements; cf. Fig. 4. Moreover, let $e_{mn} = |\mathbf{y}^m - \mathbf{y}^n|$ $(m, n = 0, ..., 3)$ and let l_p be the distance between the point $\mathbf{x}^p$ and the midpoint of the element through $\mathbf{y}^1$ and $\mathbf{y}^2$ (Fig. 4). The vectors $\mathbf{b}$ and $\delta\mathbf{b}_{j,k}$ can be quantified for this particular element in a similar way. Furthermore, we observed for a large number of simulations that the maximum values of the boundary velocity $\mathbf{v}$ is nearly almost of order 1 or smaller. Using the latter observation and the approximations that can be derived for the various integrals and vectors involved (cf. Van de Vorst [22]), the column contribution of the vector $\mathcal{G}\delta\mathbf{b}_{j,k} + \delta\mathcal{G}_{j,k}\mathbf{b} - \delta\mathcal{H}_{j,k}\mathbf{v}$ can be estimated as

$$\|\delta\mathcal{H}_{j,k}\mathbf{v}\| = \mathbb{O}\left(\frac{1}{l_p}\right), \quad \|\delta\mathcal{G}_{j,k}\mathbf{b}\| = \|\mathcal{G}\|\mathbb{O}\left(\frac{1}{e_{02}^2} + \frac{1}{e_{13}^2}\right),$$
$$\|\mathcal{G}\,\delta\mathbf{b}_{j,k}\| = \|\mathcal{G}\|\mathbb{O}\left(\frac{e_{12}}{e_{02}^3} + \frac{e_{12}}{e_{13}^3}\right). \tag{16}$$

Note that $\|\delta\mathcal{H}_{j,k}\mathbf{v}\|$ is small compared to both other terms. When the element is a piece of a "smooth" part of the boundary, both e_{02} and e_{13} are $\mathbb{O}(e_{12})$ so that $\|\delta\mathcal{G}_{j,k}\mathbf{b}\|$ and $\|\mathcal{G}\delta\mathbf{b}_{j,k}\|$ are of the same order. However, when that piece of boundary is curved or situated in the neighborhood of a cusp-like region then $e_{02} \ll e_{12}$ and/or $e_{13} \ll e_{12}$ so that $\|\mathcal{G}\delta\mathbf{b}_{j,k}\|$ will be the leading term of this vector.

In the case of a row derivative, i.e., $x_k^j = x_k$, the term $\delta\mathcal{G}_{j,k}\mathbf{b} - \delta\mathcal{H}_{j,k}\mathbf{v}$ is equal to the discretized version of the derivative of the original integral formulation with respect to x_k. Consequently, $\delta\mathcal{G}_{j,k}\mathbf{b} - \delta\mathcal{H}_{j,k}\mathbf{v}$ is equal to a linear combination of $\partial v_i^j/\partial x_k$. Since the latter derivatives are occurring in the stress tensor too, we assume that these derivatives at the boundary can be quantified to be of the order of the local boundary curvature. Here, we used the fact that the stress in the normal direction at the boundary is proportional to the curvature. From this assumption and the estimate that can be found for $\delta\mathbf{b}_{j,k}$, it follows that in the case of a row derivative

$$\|\delta\mathcal{G}_{j,k}\mathbf{b} - \delta\mathcal{H}_{j,k}\mathbf{v}\| = \mathbb{O}\left(\frac{e_{12}}{e_{02}^2} + \frac{e_{12}}{e_{13}^2}\right), \tag{17}$$

which is similar in size as we obtained for the case of a column derivative.

From the analysis above and equation (14), it follows that we can approximate the $(2j - k)$th columns $\delta\mathbf{v}_{j,k}$ of the Jacobian $\mathcal{J}$ as

$$\delta\mathbf{v}_{j,k} \doteq \mathcal{H}^{-1}\mathcal{G}\,\delta\mathbf{b}_{j,k}. \tag{18}$$

This approximation is meaningful even when the ODEs are not stiff since then a rough approximation of the Jacobian will be sufficient. Anyhow, in Section 6 we will show that accurate results are obtained for two model examples of the previous section using this approximate Jacobian for both the linear and quadratic element solution.

5. RESTART OF INTEGRATION AFTER A NODE REDISTRIBUTION

In Van de Vorst and Mattheij [20] we presented an algorithm for an optimal node redistribution based on equidistributing the curvature of the boundary. After a node redistribution, the (material) points of which the trajectories were being followed also change, i.e., the set of ODEs can completely change its character. This is also illustrated by Fig. 1, where the "jumps" in the spectral radius are due to this node redistribution. After such a redistribution the time integration is started without any information of the previous time step. The spectral radius development of both other examples does not show such a jumping behaviour after a node redistribution. This can be explained from the similarity of the particle trajectories in the neighbourhood of a particular node for such smooth shapes. Therefore in this case, it is not necessary to redistribute the nodal points frequently, thus a restart of the time integration with these new mesh will not yield much; however, the latter does make sense for shapes with an evolving "cusp," since

the position of nodes near such a cusp-like region have to be controlled well in order to obtain a realistic value for the curvature there.

When we like to restart the time integration, LSODE has to be started without further information; i.e., the order of the method is equal to 1 and the initial step size is given by the program. However, we want the integrator to continue with the order and step-size equal to the latest value before the node redistribution was carried out. We will show below that under certain conditions it is possible to perform such a restart. Before doing this, we first have to dwell on some aspects of the implementation of the BDF-method in LSODE which is also discussed in Gear [6] and Hindmarsh [9, 10].

The code LSODE is based on the *Nordsieck representation* of the fixed step size BDF-methods. For the solution of the ODEs at time $t = t_{i+1}$ the original pth-order BDF-method needs the actual values of the boundary nodes at previous times $t_i, ..., t_{i-p+1}$ and the velocity of the boundary at t_i as well. When this pth-order BDF-method is expressed in so-called Nordsieck representation, the boundary at $t = t_i$ and the first till the pth derivative (with respect to t) of this boundary are required. Thus, the Nordsieck vector, say $\mathbf{z}^i$, can be expressed as

$$\mathbf{z}^i = \left(\mathbf{x}(t_i), h\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}(t_i), ..., \frac{h^p}{p!}\frac{\mathrm{d}^p\mathbf{x}}{\mathrm{d}t^p}(t_i) \right)^{\mathrm{T}},$$

where h is the step size that will be applied. The advantage of this representation is that when the step size h is changed, the Nordsieck vector for this new step size is easy to find. The Nordsieck vector is also used to predict the solution at the next time level $t_{i+1} = t_i + h$, i.e.,

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \mathbf{h}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}(t_i) + \cdots + \frac{h^p}{p!}\frac{\mathrm{d}^p\mathbf{x}}{\mathrm{d}t^p}(t_i). \qquad (19)$$

This then is the starting vector for the Newton iteration for solving the corrector BDF. LSODE is using this particular initial guess of the next time level geometry to determine the velocity field at that level and to update the Jacobian eventually.

Furthermore, this predictor is used for estimating the local integration error; indeed LSODE applies the following automatic step size and order selection strategy (see also Hindmarsh [10]): at each step, an estimate of the local error for the present method of order p is obtained from the difference between the predictor (19) and the finally corrected values of $\mathbf{x}(t_{i+1})$. From this local error vector a step size h is deduced which satisfies the required tolerance. When this tolerance is not reached, the computed step size will be used to redo this particular step. After $p + 1$ steps with the same value for h, the order and/or step size will be updated; besides, for the method of order p, error estimates are formed for methods of order $p - 1$ and $p + 1$, too. From these three error estimates new step sizes h are computed. Based solely on efficiency considerations, the new-order method is chosen so that the new step size will be maximum in magnitude. For the simulation of sintering problems, we observed that with very small tolerances, LSODE tried to solve the ODEs with the largest possible order method; by lowering the tolerance to more practical applied values, it had the tendency to use a method of order two or three.

In order to continue with LSODE after a node redistribution, with the same order and step size as before the redistribution, the Nordsieck vector for those new nodes is required, i.e., the first until the pth derivative (with respect to t) of these nodal points. We now outline the procedure for finding the higher order derivatives.

In principle, we have the Nordsieck vector, e.g., the derivatives, for the old nodal points. The boundary is found by a Lagrangian polynomial interpolation through these points; i.e., in the notation of Section 2,

$$\mathbf{y}(s) = \phi_j(s)\mathbf{y}^j,$$

where $-1 \leq s \leq 1$ and $j = 1, ..., M + 1$. Since the interpolation polynomials ϕ_j are independent of time, the kth derivative with respect to t of the above equation is equal to

$$\frac{\mathrm{d}^k\mathbf{y}}{\mathrm{d}t^k}(s) = \phi_j(s)\frac{\mathrm{d}^k\mathbf{y}^j}{\mathrm{d}t^k}.$$

In this way, we see that the problem of finding the new Nordsieck vector can be reduced to an interpolation problem using the old Nordsieck vector.

We do not want the interpolation error which is introduced by this interpolation, to influence the new Nordsieck vector. By taking into consideration the multiple usage of the predictor (19) for the next level geometry as mentioned before, we have to avoid that the accuracy of this initial guess is affected by the interpolation. So the degree of the Lagrangian polynomials has to be large enough to ensure that the resulting error be smaller than the smallest component of the Nordsieck vector. Because of this we apply a polynomial interpolation of degree five.

Note that also the *spatial* discretization error induced by the BEM will affect the actually found velocity $\mathbf{v}$ at the new nodal points. In particular during the initial stage of the evolution of a cusp-like region, one can observe that the velocity field as obtained from interpolating the old Nordsieck vector can differ in all digits and for all points from the actual velocity field of the new discretized boundary. This large difference is caused by the ill-conditioning of the boundary value problem for such kind of shapes (cf. Van de Vorst and Mattheij [20]. Consequently, the initial guess that is obtained from the interpolated Nordsieck vector will have a large error too, irrespective of the quality of that interpolation. Note, that also interpolation problems might occur in the neighbourhood of such a cusp-like point since the redistribution algorithm is effectively

smoothing out this part of the boundary somewhat. Through this larger error in the predictor, the step size and order selection strategy of LSODE can easily be disturbed on top of possible problems of the convergence of the Newton iteration. From this point of view, it does not seem quite meaningful to restart with the same order and step size. However, by improving the initial guess, it may be possible to continue the integration by a second order method. Since we observed that LSODE does have serious difficulties when restarting the integration without any further information has been given (see also Section 6), this can sometimes mean a considerable saving in computer time.

A satisfactory choice for the predictor is solving the problem for this new grid first and then replacing the first derivative in the new Nordsieck vector by the thus obtained exact $h\mathbf{v}$. In this way we obtain a predictor that is much closer to the corrected value after the Newton iteration. This requires an extra assembling and solution of the system of equations; however, we also observed that this investment was decreasing the *total* computing time considerably (see Section 6), since there was less overhead in restarting and the error control was not affected either.

For a smooth geometry, the difference between the velocity field as obtained from the interpolated old Nordsieck vector and the exact velocity of the new boundary is not so dramatically large; this error is of the same order as the anticipated discretization error. In order to build in an extra safeguard, we replace the first derivative in the Nordsieck vector by the exact computed $h\mathbf{v}$ for the new boundary too.

6. NUMERICAL EXPERIMENTS AND DISCUSSION

Here, we will first demonstrate how well the approximated Jacobian of Section 4 is performing when used by the time integrator. Next, we show the advantages of the algorithm to restart LSODE after a node redistribution, as described in the previous section. In these simulations, all the program parameters that occur in the redistribution algorithm and the time integrator will be the same for each test problem. Finally, we demonstrate the applicability of this numerical solution method to simulate viscous sintering problems.

6.1. *Jacobian Approximation*

To illustrate that the approximation derived for $\mathcal{J}$ is sufficient, we compare the numerical solutions of the example problems of Section 3 for a numerically exact Jacobian and the approximation (18) in LSODE. These solutions are compared to the exact analytical solution at some characteristic boundary points and at a fixed number of times, which will not necessarily be the time levels of the integrator. The solution at such an intermediate point in time is found by applying the interpolation facility of LSODE. Since we compare the numerical solutions to the exact analytical solution in relation to the Jacobian that is used, both the actual error size and source (spatial and/or time) are not important for the moment. Here, the only interest is the question whether this error is *behaving in a similar way* for both Jacobians; if so we can conclude that the approximated Jacobian is sufficient. Furthermore, we will compare other important properties like the size and total number of time steps, the number of Jacobian evaluations and the order of the BDF-method. We also show that this approximate Jacobian can be used successfully in the case of the quadratic element solution.

We consider the two equal coalescing cylinders first. For the error control in the time integrator LSODE we use a global absolute error tolerance parameter equal to 10^{-4}; the relative error parameter is taken component-wise. This relative error is set equal to 10^{-3} for the "smooth" parts of the boundary and equal to 10^{-5} for the nodes in the touching region of both cylinders. A node redistribution is carried out when the nodal points are becoming too close to each other (10^{-3}) and after each 15 consecutive steps at most. The minimum and maximum distance between two successive nodes (used in the redistribution algorithm) is set equal to 5×10^{-3} and 0.15, respectively.

We have compared the difference of both the coalescence rates (ε_r) and the shrinkage (ε_s) between the analytical and numerical solutions. These particular two boundary points are chosen since the coalescing rate represents the point where the boundary is undergoing the largest deformation on one hand and the point that denotes the shrinkage rate represents a part of the boundary which is hardly deforming on the other hand. In Fig. 5 we have plotted those mentioned absolute differences at a time sequence $t = 0.0(0.5)2.0$, respectively. The solid lines in the plots indicate that the exact Jacobian is used by the simulation; otherwise (dotted lines) the approximation (18) is used. The difference in the coalescence rate is marked by circles and the shrinking error is denoted by "**x**"-marks.

As can be observed from this figure, the error behaviour of the numerical solution obtained by employing the approximated Jacobian is matching quite well with the error behaviour in the case of exact Jacobians. We see that this is valid when the shape has parts with large varying curvature as well as when the boundary is smoother. In Section 4 we found that for the latter case the approximate Jacobian may have a larger error. However, we observe that this error does not influence the numerical solution. Furthermore, this similar error behaviour is valid for both the linear and quadratic element implementations. The reason that for the initial stage the error in the coalescence rate is much larger than during the rest of the simulation, is due to the difficulty of quadratic elements to follow sharply curved regions; there is a tendency to smooth such a cusp-like region much faster so that the coalescence is proceeding quicker too. However, this is not influencing the error at later time stages through the conditioning of the problem as can be seen from the figure.

In Fig. 6 the step size development of LSODE is plotted for both Jacobians, showing a similar behaviour also. The order development of the BDF-method (not shown) is also behaving similarly. Here, LSODE is started after a redistribution without any further information from the previous time level. The de-

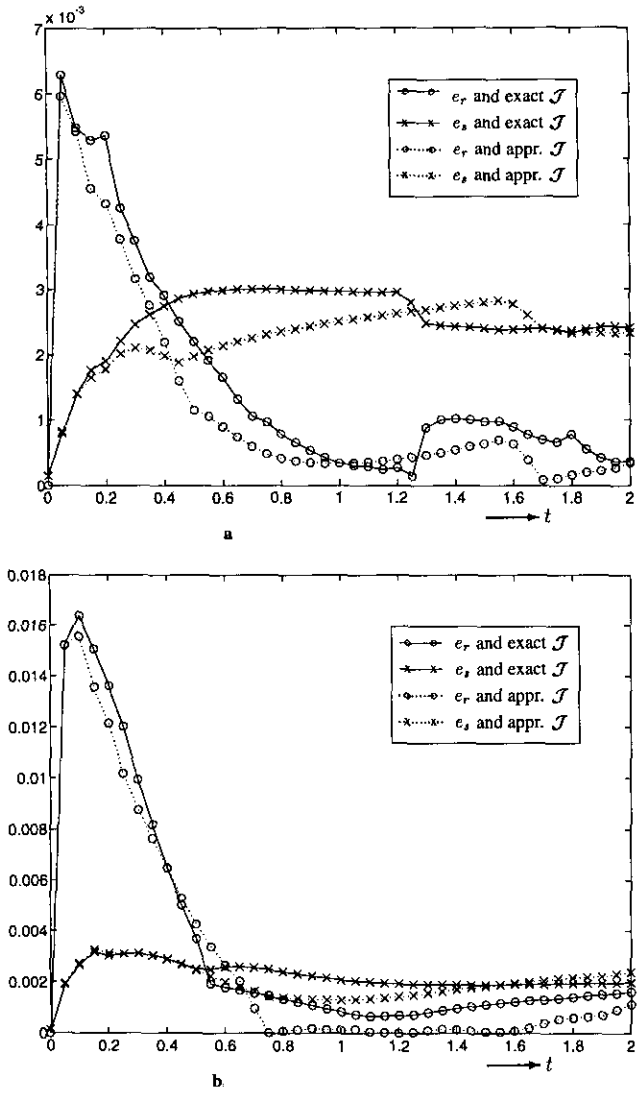


FIG. 5. For two coalescing cylinders is shown the similar error behaviour between the exact analytical solution and the numerical solutions obtained by using the exact (solid line) and the approximated (dotted line) Jacobian for both linear elements (a) and quadratic elements (b); e_r is the absolute error in the coalescing rate and e_s is the error in particle shrinkage rate.

crease in step size for the linear element solution with the approximate Jacobian at the later stage is due to a restart after a node redistribution (see also Table I further on, where all characteristics of this particular evolution are printed). The tables printed in this figure show a similar behaviour of the total number integration steps (#steps) and the total Jacobian updates (#$\mathcal{J}$). All the similarities mentioned above appear to indicate that the Jacobian approximation (18) is sufficient for all kinds of simply connected domains.

Next, we show that this approximation is also valid in the case of multiply connected domains, by considering a shrinking circular annulus example. For the error control in LSODE we use a global absolute and relative tolerance equal to 10^{-6}. The node redistribution is carried out when nodes are becoming too close to each other (10^{-3}) and at most after each 25 consecutive steps. The minimum and maximum distance between two successive nodes is taken to be the same as in the previous example. In Fig. 7 we have plotted the absolute error development between the exact analytical solutions, found by applying both Jacobians when linear elements are applied at a sequence of time points $t = 0.0(0.02)0.72$. The reason that the error in both numerical solutions is identical is caused by the fact that we have taken the time tolerance much smaller than the spatial

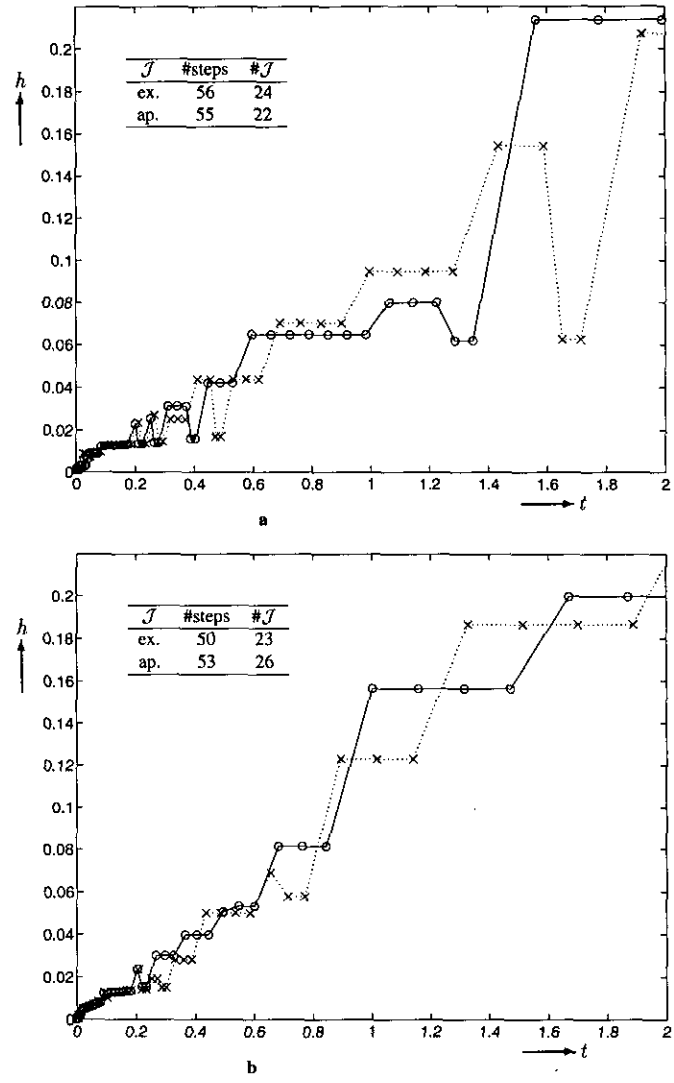


FIG. 6. The time step (h) development of LSODE using the approximate Jacobian (×-marks and dotted line) shows a similar behaviour as when the exact Jacobian is applied (circles and solid line) for two coalescing cylinders in both the linear (a) and the quadratic element (b) solution. The tables show the similar behaviour in the total number of time steps (#steps) and Jacobian updates (#$\mathcal{J}$) required.

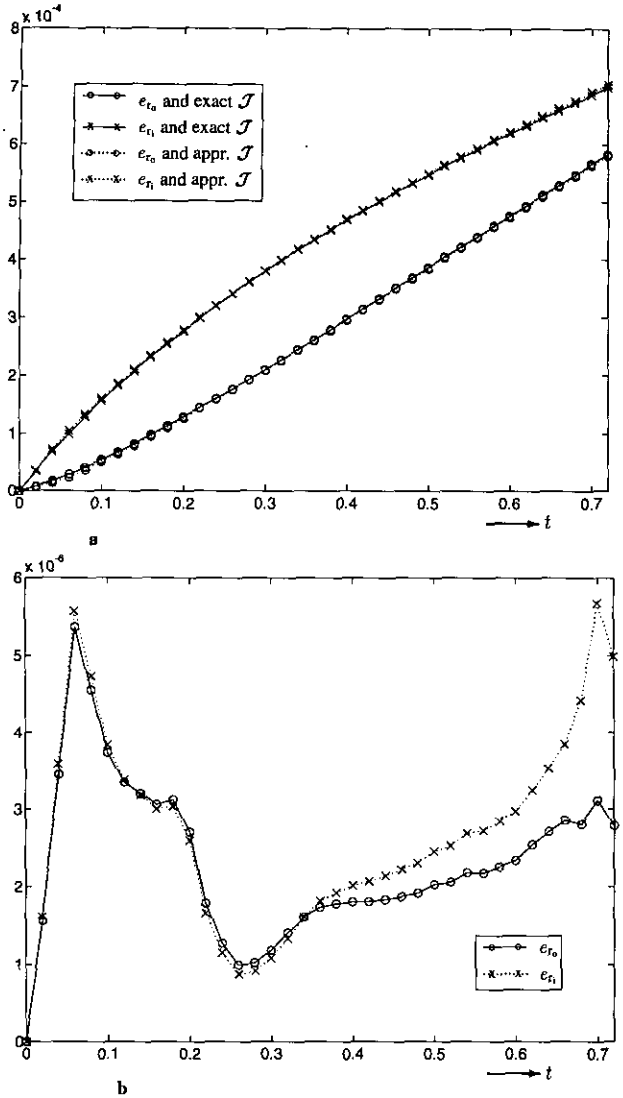


FIG. 7. For the case of the circular annulus shrinkage is shown in a the identical error behaviour of the exact analytical solutions of both the inner (e_{r_i}) and outer (e_{r_o}) radius compared to the numerical solutions when using both the Jacobians and linear elements. Figure b shows that the difference between both the numerical solutions is of the order of the time integration tolerance (10^{-6}).

error tolerance; thus the actual error this plot shows is due to the spatial discretization. In order to show that the approximated Jacobian for this particular case is performing well too, we have plotted the absolute difference between these numerical solutions. As can be seen this difference is of the order of the tolerance of the time integrator. Again, we observe that the step size and order development of LSODE, using either Jacobian exhibits a similar behaviour too. However, the exact Jacobian solution needs a lower number of time steps (33) and Jacobian updates (11) than the other solution method (40 time steps and 15 Jacobian updates). These values do still not justify the use of the better and much more expensive approximation (14). We obtained similar results for the quadratic element implementation. From this, we can conclude that the approximate Jacobian is sufficient for multiply connected domains too.

6.2. *Restarting the Time Integration*

The savings in computational costs that can be achieved by applying the restarting algorithm of Section 5 after a node redistribution in the case that a cusp-like region is involved during the evolution will be shown by an example of two equal coalescing cylinders. In Table I the subsequent time steps (t_i) are printed for the case that after a node redistribution (*rd*) the restart of the time integration has been carried out without further information; i.e., the starting order of the method is equal to 1 and the initial step size will be set by LSODE. Here, we use the linear element implementation. N is the total number of points and $\#\mathcal{H}^{-1}\mathcal{G}$ is the *total* number of times that the assembling and solving of the system of equations is carried out up to and including the time t_i. By *area* we denote the total area of the fluid region (which has to be preserved during the evolution). The (relative) change of this total area, compared to the area of the original shape, is also printed. These numbers in the table show that the relative error in the area is caused by the node redistribution algorithm only.

From Table I we also observe that the order of the BDF-method is nearly equal to 1 during the evolution the cusp-like region; i.e., a backward Euler method is used. Furthermore, this table shows the computational costs of a *restart* caused by a node redistribution: 2-7 times the assembling and solution of the system of equations and 1-2 Jacobian updates. These large numbers are caused by wrong choices of the initialization when using LSODE as a "black box." Thus, LSODE has serious difficulties to restart the integration during this evolution period. Note that this table also shows the behaviour of the order and step size selection strategy of LSODE as we described in Section 5.

In Table II we have printed the integration steps for the same problem when the order and step size are the same as before the node redistribution. Here the Nordsieck vector for the new nodes is found by interpolating the old Nordsieck vector using Lagrangian polynomials with degree five, as is outlined in Section 5.

Now, we observe that the order of the BDF-method is equal to two (or more) during the evolution and that the total number of integration steps is smaller (~20%). Further, we see a considerable reduction (~40%) of the total number of the assemblies and system solves that have to be carried out and of the number of Jacobian updates as well. This gives a justification for the restarting method as we described in Section 5.

6.3. *Simulation of Viscous Sintering*

In this subsection we consider some typical evolution problems which may occur during viscous sintering simulations to

TABLE I

The Time Steps for the Coalescence of Two Equal Cylinders when Linear Elements Are Used

i		N	t_i	h	p	$\#\mathcal{H}^{-1}\mathcal{G}$	$\#\mathcal{G}$	Area	Error (%)
1	rd	92	0.0011	0.00114	1	6	2	3.1404	0.0006
2		92	0.0023	0.00114	1	7	2	3.1404	0.0011
3		92	0.0036	0.00136	2	8	2	3.1404	0.0015
4		92	0.0050	0.00136	2	9	2	3.1404	0.0018
5		92	0.0064	0.00136	2	10	2	3.1404	0.0021
6		92	0.0089	0.00250	3	11	2	3.1404	0.0027
7		92	0.0114	0.00250	3	12	2	3.1403	0.0033
8		92	0.0139	0.00250	3	13	2	3.1403	0.0040
9		92	0.0164	0.00250	3	14	2	3.1403	0.0046
10	rd	100	0.0249	0.00858	1	21	3	3.1402	0.0188
11		100	0.0335	0.00858	1	24	3	3.1402	0.0169
12	rd	96	0.0405	0.00697	1	30	5	3.1404	0.0058
13		96	0.0475	0.00697	1	32	5	3.1404	0.0064
14	rd	100	0.0561	0.00858	1	39	6	3.1414	0.1050
15		100	0.0646	0.00858	1	42	6	3.1414	0.1055
3		3	3	3	3	3	3	3	3
50		76	1.4343	0.15420	3	111	20	3.1371	0.3250
51		76	1.5885	0.15420	3	112	20	3.1370	0.3338
52	rd	48	1.6510	0.06253	1	114	21	3.1369	0.3431
53		48	1.7136	0.06253	1	115	21	3.1370	0.3381
54		48	1.9207	0.20709	2	116	22	3.1368	0.3596
55		48	2.1277	0.20709	2	118	22	3.1365	0.3841

Note. After a node redistribution (*rd*), the time integration is restarted *without* further information.

TABLE II

The Time Steps for the Coalescence of Two Equal Cylinders when Linear Elements Are Used

i		N	t_i	h	p	$\#\mathcal{H}^{-1}\mathcal{G}$	$\#\mathcal{G}$	Area	Error (%)
1	rd	92	0.0011	0.00114	1	6	2	3.1404	0.0006
2		92	0.0023	0.00114	1	7	2	3.1404	0.0011
3		92	0.0036	0.00136	2	8	2	3.1404	0.0015
4		92	0.0050	0.00136	2	9	2	3.1404	0.0018
5		92	0.0064	0.00136	2	10	2	3.1404	0.0021
6		92	0.0089	0.00250	3	11	2	3.1404	0.0027
7		92	0.0114	0.00250	3	12	2	3.1403	0.0033
8		92	0.0139	0.00250	3	13	2	3.1403	0.0040
9		92	0.0164	0.00250	3	14	2	3.1403	0.0046
10	rd	100	0.0240	0.00765	2	17	3	3.1403	0.0083
11		100	0.0317	0.00765	2	19	3	3.1403	0.0096
12	rd	100	0.0393	0.00765	2	21	4	3.1407	0.0340
13		100	0.0470	0.00765	2	23	4	3.1407	0.0314
14	rd	100	0.0582	0.01126	2	25	5	3.1406	0.0214
15		100	0.0695	0.01126	2	27	5	3.1406	0.0174
3		3	3	3	3	3	3	3	3
39		72	1.2887	0.15532	3	66	14	3.1358	0.4593
40		72	1.4440	0.15532	3	68	14	3.1358	0.4603
41		72	1.5994	0.15532	3	69	14	3.1358	0.4607
42		72	1.7547	0.15532	3	70	14	3.1357	0.4639
43		72	1.9883	0.23360	3	72	15	3.1358	0.4622
44		72	2.2219	0.23360	3	73	15	3.1358	0.4590

Note. After a node redistribution (*rd*), the time integration is restarted with the same order and step size as before the redistribution.

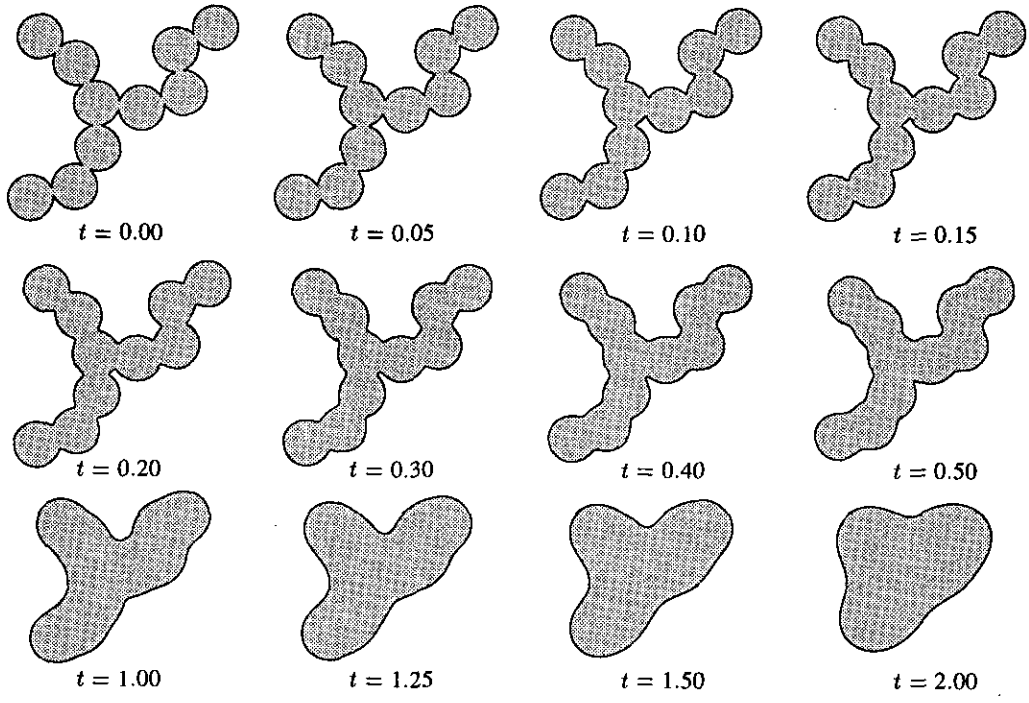


FIG. 8. The rounding of an irregular chain of cylindrical particles.

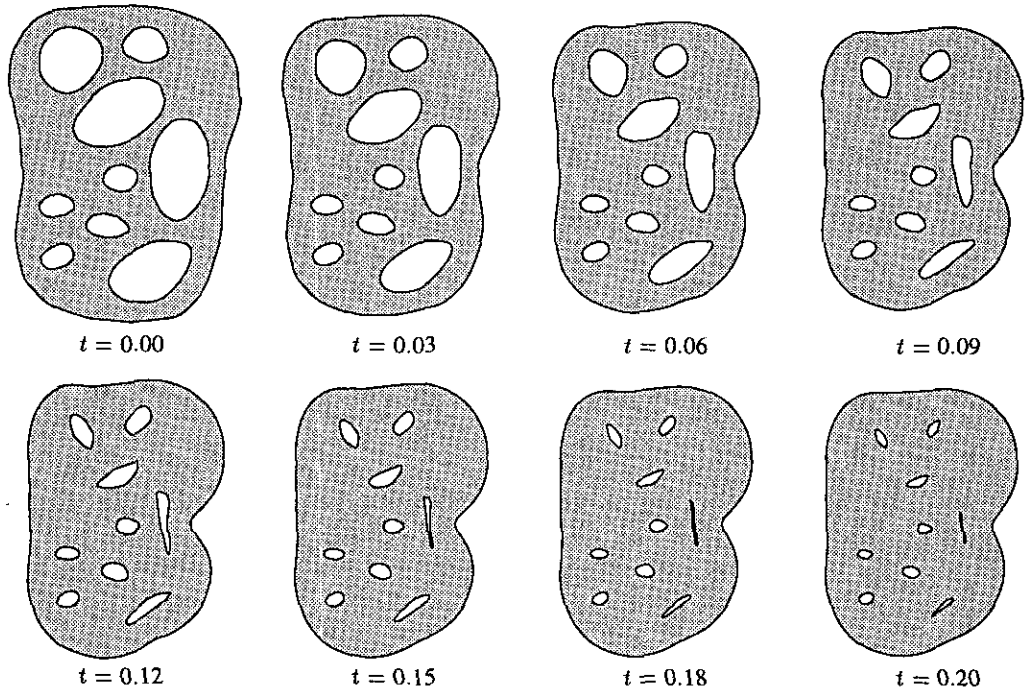


FIG. 9. The densification of a fluid region with nonuniform sized pores shows both the faster shrinking of the larger pores and the touching of boundaries inside the pores.

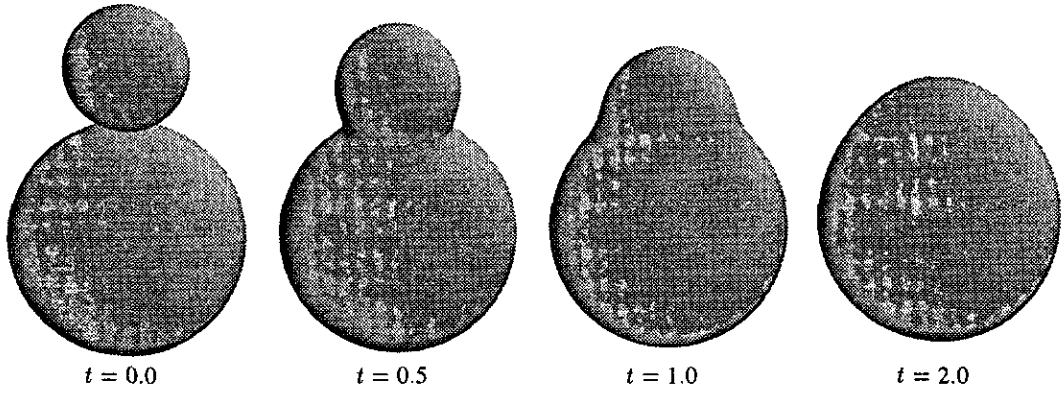
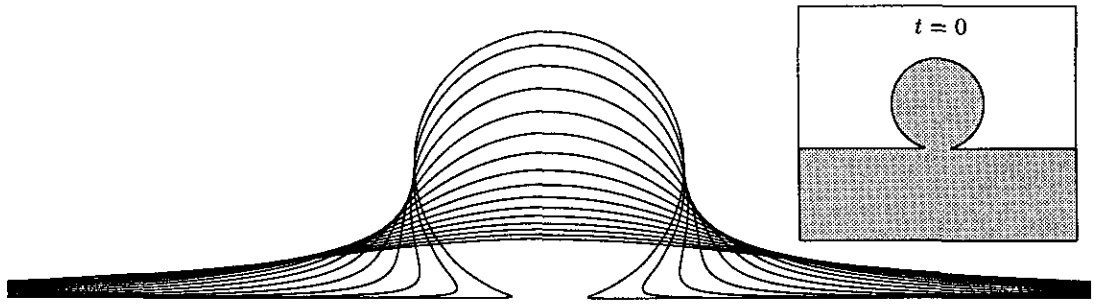
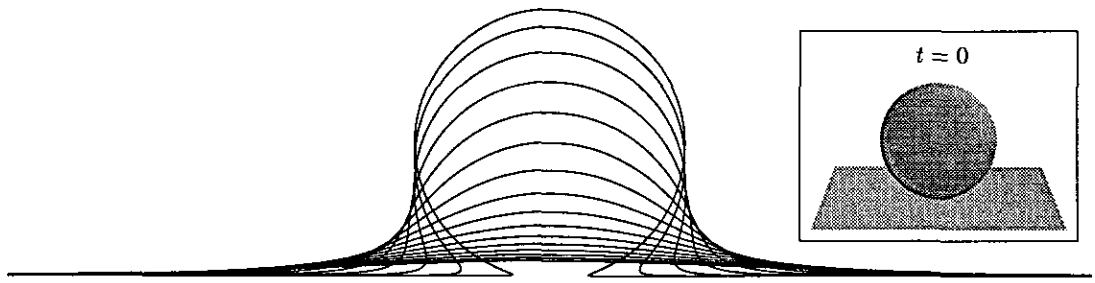


FIG. 10. The coalescence of a sphere with initial radius 0.5 on a sphere with radius 1. The smaller sphere is gradually "eaten up" by the larger one. This phenomenon is usually referred to as *Ostwald ripening*.

demonstrate the time integration method presented above. As we already mentioned in the Introduction, the viscous sintering process occurs when a porous pure glass is heated to a sufficiently high temperature so that the glass becomes a highly viscous fluid: the flow causes densification of the glass. The driving force for this phenomenon is the excess of free surface energy of the porous glass compared to a same quantity of a fully dense glass. Ideally, one wants to produce a dense and homogeneous glass, free from voids and impurities, which, for example, can be used to produce glass fibres for the telecommunication industry. Therefore, a good theoretical understanding is needed of the densification kinetics of the porous glass, i.e., the viscous sintering phenomenon. In particular, one is interested in the shrinkage rate of the glass as a function of the viscosity and particle size, which reflects how time, temperature, and microstructure influence the development of the densification process. Another question is what kind of structural configuration leads to a higher densification rate. A simple approach of describing the sintering phenomenon is to



Coalesence of a cylinder with radius 0.5 on a halfspace



Coalesence of a sphere with radius 0.5 on a halfspace

FIG. 11. A comparison between the coalescence of a cylinder on a plate and the coalescence of a sphere with both equal radius of 0.5 for $t = 0.0(0.2)3.0$.

consider the behaviour of simple systems only, like the coalescence of two spheres, which can be used to understand the behaviour of macroscopic systems. A more sophisticated approach is the determination of a representative unit cell within the porous glass and to consider the densification of it. This unit cell has to be chosen so that it reflects the sintering of the porous glass as a whole realistically.

A first typical behaviour that we consider here is the evolution of nearly cusp-like regions into, eventually, a cylindrical shape. The deformation of an arbitrary chain of equally sized cylindrical particles is shown in Fig. 8. At the time stage $t = 0.4$ we observe the development of a new sharply curved region. Moreover, during the initial stage of the coalescence the particles are (a little) rearranging.

A second typical problem is the densification of a fluid region with nonuniform sized pores. In Fig. 9 we simulated the shrinkage of such a fluid domain. One remarkable phenomenon that can be observed is that the larger pores are shrinking significantly faster as compared to the smaller ones as time evolves. Another effect to be taken care of is the touching of the boundary inside a pore; this phenomenon can be seen in the final shape of Fig. 9.

So far, we have concentrated on two-dimensional problems; however, the solution method may also be applied to other type of problems like axisymmetrically shaped fluid domains. This requires a change of the kernels in the integral formulation and the approximation of, now, the *surface* curvature. As a consequence of the latter remark, the approximate Jacobian (18) has to be updated too. In order to demonstrate the applicability of the method, we have plotted the coalescence of two unequal spheres in Fig. 10. As can be observed from these pictures, the smaller sphere is gradually vanishing into the larger one. This phenomenon of growth of large particles at the expense of smaller ones is usually referred to as *Ostwald ripening* or *cannibalism* in sintering literature.

Another type of problem where we successfully used this solution technique is the deformation of a particle on a halfspace, in both the two-dimensional and axisymmetric case. In Fig. 11 we show both the coalescence of a cylinder on a halfspace (fiber on a plate) and a sphere on a halfspace. Recently, these latter two problems have received a physical application in the sense that these solutions can be used to determine the surface tension of a certain type of glass at relatively low temperatures (600°C); cf. De With and Corbijn [23]. Note that this halfspace implementation may also be applied to investigate the smoothing of a pressed profile in a hot piece of glass or the smoothing of cracks on a glass surface.

REFERENCES

1. U. M. Ascher, R. M. M. Mattheij, and R. D. Russel, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* (Prentice–Hall, Englewood Cliffs, NJ, 1988).
2. A. A. Becker, *The Boundary Element Method in Engineering* (McGraw–Hill, London, 1992).
3. C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel, *Boundary Element Techniques* (Springer-Verlag, Berlin, 1984).
4. M. J. Cooker, D. H. Peregrine, C. Vidal, and J. W. Dold, *J. Fluid Mech.* **215,** (1990).
5. J. M. Floryan and H. Rasmussen, *Appl. Mech. Rev.* **42,** 323 (1989).
6. C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations* (Prentice–Hall, Englewood Cliffs, NY, 1971).
7. S. T. Grilli, J. Skourup, and I. A. Svendsen, *Engng. Anal.* **6,** 97 (1989).
8. C. Haack, P. Gravert, and V. Schlegel, "The Modelling of Extreme Gravity Waves: An Approach towards a Numerical Wave Channel," in *Proceedings, Computational Modelling of Free and Moving Boundary Problems. Vol. I. Fluid Flow, Southampton,* 1991, edited by L. C. Wrobel and C. A. Brebbia (Comput. Mech., Southampton, 1991), p. 91.
9. H. C. Hindmarsh, *ACM-SIG Newsletter* **15** (1980).
10. H. C. Hindmarsh, "ODEPACK, a Systematized Collection of ODE Solvers," in *Scientific Computing,* edited by R. Stepleman *et al.* (North-Holland, Amsterdam, 1983), p. 55.
11. H. C. Hindmarsh, "Decline Stability Barriers in BDF Solvers," in *Computational Ordinary Differential Equations,* edited by J. Cash and I. Gladwell (Clarendon, Oxford, 1992), p. 87.
12. S. Kim and S. J. Karrila, *Microhydrodynamics: Principles and Selected Applications* (Academic Press, London, 1991).
13. M. S. Longuet-Higgins and E. D. Cokelet, *Proc. R. Soc. London A* **350,** 1 (1976).
14. C. Pozrikidis, *Boundary Integral and Singularity Methods for Linearized Viscous Flow* (Cambridge Univ. Press, Cambridge, 1992).
15. D. Ramsden and G. Holloway, *J. Comput. Phys.* **94,** 101 (1992).
16. H. A. Stone, *Ann. Rev. Fluid Mech.* **26,** 65 (1994).
17. J. Tanzosh, M. Manga, and H. A. Stone, "Boundary Integral Methods For Viscous Free-Boundary Problems: Deformation of Single and Multiple Fluid-Fluid Interfaces," in *Proceedings, Boundary Element Technology VII,* edited by C. A. Brebbia and M. S. Ingber (Comput. Mech., Southampton, 1992), p. 19.
18. G. A. L. van de Vorst, R. M. M. Mattheij, and H. K. Kuiken, *J. Comput. Phys.* **100,** 50 (1992).
19. G. A. L. van de Vorst and R. M. M. Mattheij, "A BDF-BEM Scheme for Modelling Viscous Sintering," in *Proceedings, Boundary Element Technology VII,* edited by C. A. Brebbia and M. S. Ingber (Comput. Mech., Southampton, 1992), p. 59.
20. G. A. L. van de Vorst and R. M. M. Mattheij, *Computing* **49,** 239 (1992).
21. G. A. L. van de Vorst, *J. Fluid Mech.* **257,** 667 (1993).
22. G. A. L. van de Vorst, Ph.D. thesis, Eindhoven University of Technology, 1994.
23. G. de With and A. J. Corbijn, *J. Appl. Phys.,* submitted.