A BDF-BEM scheme for modelling viscous sintering

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

RANA 92-05
March 1992

A BDF-BEM SCHEME FOR MODELLING VISCOUS SINTERING

by
G.A.L. van de Vorst
R.M.M. Mattheij

Reports on Applied and Numerical Analysis
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands
ISSN: 0926-4507

A BDF-BEM Scheme for Modelling Viscous Sintering

G.A.L. van de Vorst
R.M.M. Mattheij

Abstract

By viscous sintering it is meant the process of bringing a granular compact to a temperature at which the viscosity of the material becomes low enough for surface tension to cause the particles to deform and coalesce, whereby the material transport can be modelled as a viscous incompressible newtonian volume flow. Here a two-dimensional model is considered. A Boundary Element Method (BEM) is applied to solve the governing Stokes creeping flow equations for a *fixed* fluid region. The movement of the boundary is obtained by following the trajectories of the (material) boundary points. It appears that the system of ordinary differential equations (ODEs), which describe this movement can be *stiff*. Because of this, a Backward Differences Formulae (BDF) is used for the solution of those ordinary differential equations. The BDF-method is modified so that the time integrator does not have to restart when a node redistribution is performed. Some numerical examples illustrate the usefulness of the method.

A.M.S. Classifications: 65R99, 76D07
Keywords : sintering, viscosity, boundary element method, moving boundaries.

1 Introduction

Sintering is the process of bringing a powder of metals, ionic crystals, or glasses (a compact) to such a high temperature that sufficient mobility is present to release the excess free energy of the surface of the powder, thereby joining the particles together. For a survey of the most important papers about sintering we refer to the book edited by Sōmiya and Moriyoshi [8].

We consider the case of sintering glasses; than the material transport can be modelled as a viscous incompressible Newtonian volume flow, driven solely by surface tension (viscous sintering), i.e. the Stokes creeping flow equations hold, see also [5] or [9]. This sintering is in effect a three-dimensional problem, but because of the complexity of such three-dimensional geometries, we restrict our self to simple geometries; to start with in two-dimensions only.

In [9] and [10] is described the approach for solving the viscous sintering problem numerically (in the sections 2 and 3 we summarize this solution). A Boundary Element Method (BEM) is applied to solve the Stokes equations for a fixed fluid region. The movement in time of the boundary is obtained by following the trajectories of the (material) boundary particles, i.e. the Lagrangian formulation is used. In the above mentioned papers, the time integration of the derived system of ordinary differential equations (ODEs) was carried out by a simple forward Euler scheme. However, it appears that this system of ODEs can be stiff: than the time step in the forward Euler scheme has to be taken very small for obtaining a stable method.

A review of some time stepping schemes for Lagrangian particles in two dimensional Eulerian flow fields is recently discussed by Ramsden and Holloway [7]. They show that for the problems they were considering, a fourth order Runge-Kutta method was giving the best results. However, this time stepping scheme is only performing well for non-stiff problems.

In this paper, in particular section 4, we shall outline the implementation of a more sophisticated time integrator: a variable step, variable order Backward Differences Formulae (BDF) scheme as is implemented in the package LSODE, cf. Hindmarsh [3]. This code is requiring the Jacobian of the system of ODEs, because some Newton method is used for the solution of the corrector equation of this implicit linear multistep method. An approximation of this Jacobian is given in subsection 4.2. Furthermore, we show in subsection 4.3, how to resume the time integration after a node redistribution, using the step size and order that was employed by the time integrator before the node redistribution was carried out. Finally, some numerical examples are given which illustrate the usefulness of the method.

2 Problem Formulation

We model the viscous sintering problem by the Stokes creeping flow equations, i.e. the flow is a viscous incompressible Newtonian fluid, see also Kuiken [5]. The *two dimensional* fluid region is assumed to be *simply connected* and is defined by a closed curve Γ with interior domain Ω. We denote the dimensionless velocity field of the fluid by $\mathbf{v}$ and the dimensionless pressure by p.

The Stokes creeping flow equations in dimensionless form read,

$$\begin{aligned} \Delta\mathbf{v} - \text{grad } p &= 0 \\ \text{div } \mathbf{v} &= 0, \end{aligned} \tag{1}$$

with stress tensor $\mathcal{T}$, given by

$$\mathcal{T}_{ij} = -\delta_{ij}p + \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{2}$$

The driving force of the deformation of the boundary is a tension, proportional to the local boundary curvature in the normal direction on the boundary. This boundary condition can be described as

$$\mathbf{b} = \mathcal{T}\mathbf{n} = \kappa\,\mathbf{n}, \tag{3}$$

where κ is the boundary curvature and $\mathbf{n}$ is the outward unit normal vector of the boundary.

In principle, the above equations can be solved for a *fixed* boundary from which we obtain the velocity field $\mathbf{v}$ of this boundary. The displacement of the boundary is derived from this boundary velocity field, in the following way,

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}) \qquad (\mathbf{x} \in \Gamma), \tag{4}$$

where t is the dimensionless time. The above equation is expressing the movement of the *material* particles, i.e. we are following the trajectories of those particles.

Our only interest is the shape evolution of the boundary when time is increasing. Hence only the boundary velocity is required, from which we can calculate the movement of the fluid region directly. Therefore this problem is ideally suited to be solved numerically by the BEM. To do this, we have to reformulate the problem as an integral equation over the boundary. This is done in terms of boundary distributions of hydrodynamical single- and double-layer potentials, see also Ladyzhenskaya [6].

The equations (1)-(3) which have to be solved for a fixed boundary, do not ensure a unique solution $\mathbf{v}$. It can be seen that a superposition of an arbitrary rigid-body *translation* or *rotation* upon any particular solution of these equations, is also a solution those equations and will not alter the stress field at the boundary. Hence, in total we need to add three extra conditions (equations) for obtaining a uniquely defined boundary velocity field. This is performed by adding three additional variables w_i to the integral equation which prescribe the translation and rotation.

When the boundary is sufficiently "smooth", the integral formulation that can be derived for the Stokes equations at a point, say $\mathbf{x}$ reads in matrix notation (see also [9],[10])

$$\mathcal{C}\,\mathbf{v}(\mathbf{x}) + \int_\Gamma \mathcal{Q}(\mathbf{x},\mathbf{y})\,\mathbf{v}\,d\Gamma_y + \mathcal{V}(\mathbf{x})\,\mathbf{w} = \int_\Gamma \mathcal{U}(\mathbf{x},\mathbf{y})\,\mathbf{b}\,d\Gamma_y. \tag{5}$$

Here $\mathcal{C}$,$\mathcal{Q}(\mathbf{x},\mathbf{y})$ and $\mathcal{U}(\mathbf{x},\mathbf{y})$ are 2×2 matrices with coefficients c_{ij}, q_{ij} and u_{ij} respectively:

$$c_{ij} = \begin{cases} \delta_{ij} & \mathbf{x} \in \Omega \\ \frac{1}{2}\delta_{ij} & \mathbf{x} \in \Gamma, \end{cases} \quad q_{ij} = \frac{r_i r_j}{\pi R^2} r_k n_k, \qquad u_{ij} = \frac{1}{4\pi}\Big[-\delta_{ij}\tfrac{1}{2}\log R + \frac{r_i r_j}{R}\Big], \tag{6}$$

where δ_{ij} is the Kronecker delta, $r_i = x_i - y_i$, $R = r_1^2 + r_2^2$, and the Einstein summation convention is used. Furthermore, $\mathcal{V}$ is a 2×3 matrix defined by

$$\mathcal{V}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & x_2 \\ 0 & 1 & -x_1 \end{bmatrix}, \tag{7}$$

and $\mathbf{x} = (x_1, x_2)$. We add the following three equations for obtaining a well-proposed problem.

In order to prescribe the translation freedom, we formulate the problem to be stationary at a (reference) point in the fluid, say $\mathbf{x}^r$. With regard to this reference point the velocity of the boundary points is computed. The most natural choice for this reference point is the centre of mass: the point where the gravity forces would grip the body, thus:

$$\mathbf{v}(\mathbf{x}^r) = 0. \tag{8}$$

Using this, we derive from the integral formulation (5) and $\mathbf{x} = \mathbf{x}^r$ the following equation

$$\int_\Gamma \mathcal{Q}(\mathbf{x}^r,\mathbf{y})\,\mathbf{v}\,d\Gamma_y = \int_\Gamma \mathcal{U}(\mathbf{x}^r,\mathbf{y})\,\mathbf{b}\,d\Gamma_y. \tag{9}$$

Furthermore we assume the tangential component of the velocity at the boundary to be zero, i.e.

$$\int_\Gamma (\mathbf{v},\tau)\,d\Gamma = 0, \tag{10}$$

where τ is the tangential vector of the boundary.

3 Boundary Element Solution

As we already mentioned in section 2, the viscous sintering problem is ideally suited to be solved by the BEM. Therefore the boundary is discretized into a sequence of N nodes and the boundary velocity and tension are written in terms of their values at those points. From the discretized form of equation (5), together with the relations (9) and (10), we derive a system of 2N linear algebraic equations with 2N unknowns, which is outlined briefly in this section.

Following Brebbia [1], we define the polynomial functions $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{b}}$, which apply at a typical element "j",

$$\tilde{\mathbf{v}} = \Phi \mathbf{v}^j \qquad \text{and} \qquad \tilde{\mathbf{b}} = \Phi \mathbf{b}^j, \tag{11}$$

where $\mathbf{v}^j$ and $\mathbf{b}^j$ are vectors which consist of the velocity and tension vectors of all successive nodes from element j. The interpolation function Φ is a 2×2M matrix of shape functions, i.e.

$$\Phi = \begin{bmatrix} \phi_1 & 0 & \phi_2 & 0 & \dots & \phi_M & 0 \\ 0 & \phi_1 & 0 & \phi_2 & \dots & 0 & \phi_M \end{bmatrix}. \tag{12}$$

The functions ϕ_m are the standard (Lagrangian) finite-element-type polynomials (cf. [1]). The number M is equal to the degree of the polynomial approximation plus one. Let the boundary consist of, say, L elements. Further, define the following types of integrals (in matrix notation) which have to be evaluated for every element Γ_j and every nodal point $\mathbf{x}^i$,

$$\begin{aligned} \hat{H}^{ij} &= \int_{\Gamma_j} \mathcal{Q}(\mathbf{x},\mathbf{y})\Phi \, d\Gamma_y, \qquad G^{ij} = \int_{\Gamma_j} \mathcal{U}(\mathbf{x},\mathbf{y})\Phi \, d\Gamma_y, \\ H^{ij} &= \begin{cases} \hat{H}^{ij} & i \neq j \\ \hat{H}^{ij} + C^i & i = j. \end{cases} \end{aligned} \tag{13}$$

After substituting the approximation polynomial (11) into integral equation (5) with a discretized boundary, and using the above abbreviations, we obtain the following equation for an arbitrary node i:

$$\sum_{j=1}^{L} H^{ij} \mathbf{v}^j + \mathcal{V}(\mathbf{x}^i)\,\mathbf{w} = \sum_{j=1}^{L} G^{ij} \mathbf{b}^j. \tag{14}$$

If we now let i vary from 1 to L, together with the discretized form of the extra relations (9) and (10), we derive the following system of 2N+3 linear algebraic equations with 2N+3 unknowns

$$\begin{bmatrix} H_1 & H_2 \\ H_3 & I_3 \end{bmatrix} \begin{bmatrix} \underline{\mathrm{v}} \\ \mathrm{w} \end{bmatrix} = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \underline{\mathrm{b}}. \tag{15}$$

Here $\underline{\mathrm{v}}$ and $\underline{\mathrm{b}}$ are the velocity, cq. tension, of the successive nodal points; H_1 and G_1 are 2N×2N matrices derived from the discretized integral equation (5). Furthermore, H_2 is a 2N×3 matrix equal to

$$H_2 = \left[\mathcal{V}(\mathbf{x}^1) \,|\, \mathcal{V}(\mathbf{x}^2) \,|\, \dots \,|\, \mathcal{V}(\mathbf{x}^N)\right]^T;$$

H_3 and G_2 are 3×2N matrices obtained from the discretized form of the relations (9)-(10), and I_3 is the identity matrix.

The system (15) can be reduced to a system of 2N linear algebraic equations; we note that the additional variables w_i are equal to

$$\mathrm{w} = G_2 \underline{\mathrm{b}} - H_3 \underline{\mathrm{v}}. \tag{16}$$

From this equation and (15), we obtain for $\underline{\mathrm{v}}$ the following system of equations,

$$\left(H_1 - H_2 H_3\right) \underline{\mathrm{v}} = \left(G_1 - H_2 G_2\right) \underline{\mathrm{b}}, \tag{17}$$

which we shall denote as

$$\mathcal{H}\underline{\mathrm{v}} = \mathcal{G}\,\underline{\mathrm{b}}. \tag{18}$$

This system is uniquely solvable.

When we include the movement of the boundary, cf. (4), in to the above equations, we derive the following system of ODEs,

$$\dot{\underline{\mathrm{x}}} = \mathcal{H}^{-1}\mathcal{G}\,\underline{\mathrm{b}}, \tag{19}$$

where $\underline{\mathrm{x}}$ is the vector of all successive nodes, and the dot denote, the derivative with respect to the time.

4 Time Integration

In this section we consider the numerical integration of the ODEs derived in the previous section, which are describing the movement of the boundary (in particular the trajectories of the nodes when they are considered as material points). First, we shall show that this system of ODEs is *stiff*. Hence, we propose to use a BDF-scheme for solving those equations.

4.1 Stiffness

The mathematical definition of "stiffness" is varying in the literature, here we say that the (non-linear) ODE (19) is stiff in an interval $[a,a+\mathrm{T}]$ if

$$\max_{t\in[a,a+\mathrm{T}]} \rho\big(\mathcal{J}(\underline{\mathrm{x}}(t))\big)\,\mathrm{T} \gg 1, \tag{20}$$

where the spectral radius ρ is defined to be

$$\rho = \max_{i=1}^{2\mathrm{N}} |\lambda_i|, \tag{21}$$

and $\lambda_i = \lambda_i(\underline{\mathrm{x}})$ are the local eigenvalues of the Jacobian matrix $\mathcal{J}(\underline{\mathrm{x}})$ of the system (19), i.e.

$$\mathcal{J}(\underline{\mathrm{x}}) = \frac{\partial}{\partial\underline{\mathrm{x}}}\big(\mathcal{H}^{-1}(\underline{\mathrm{x}})\,\mathcal{G}(\underline{\mathrm{x}})\,\underline{\mathrm{b}}(\underline{\mathrm{x}})\big), \tag{22}$$

and $\underline{\mathrm{x}}$ are relevant nodal points.

It is impossible to derive an analytical expression for the Jacobian or the spectral radius. Because of this, we show the stiffness of the sintering problem on the basis of a simple but typical example, viz. the evolution of the coalescence of two equal circles, cf. [10] section 6.

In figure 1 we have plotted the spectral radius of the numerically obtained (exact) Jacobian at various time steps, when the fluid is transforming itselfs into a circle when time is increasing. Here, the problem was solved using *quadratic* boundary elements. The jumps in the spectral radius are caused by a node redistribution, i.e. the trajectories of other particles are followed then. When

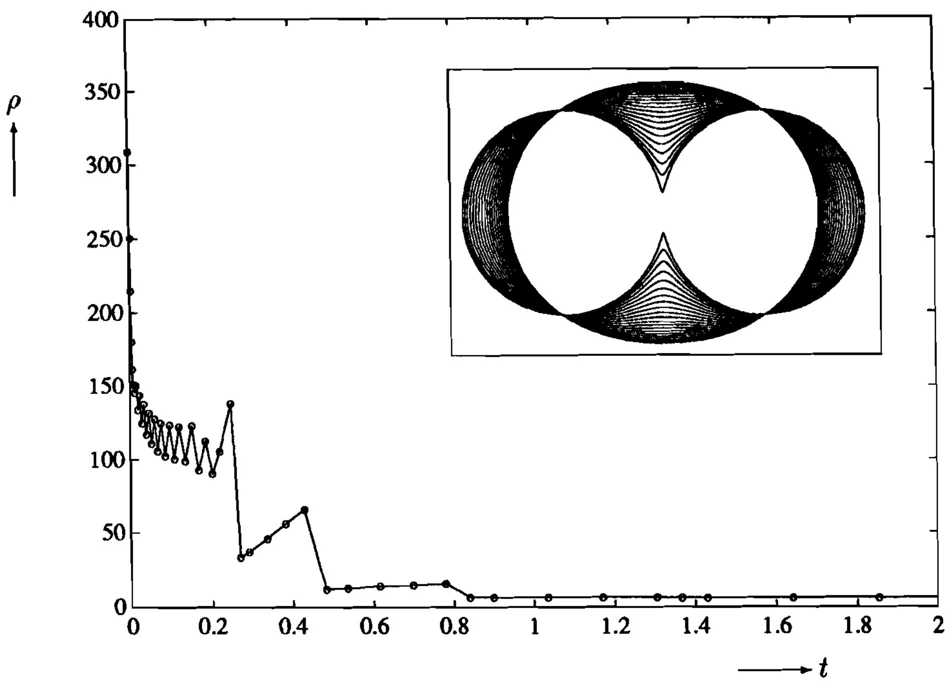

Figure 1: The spectral radius ρ of the numerically obtained (exact) Jacobian during the evolution when quadratic elements are used. The jumps in the spectral radius are caused by node redistributions.

time is increasing, we see that the stiffness is disappearing; because the boundary is becoming almost a circle.

We applied the variable step, variable order BDF-method, as is implemented in the solver LSODE, cf. Hindmarsh [3], for obtaining the solution of those ODEs. For solving the corrector equation of this implicit linear multistep method, LSODE is using some Newton method. This implies that the code requires the Jacobian (22) of the system of ODEs.

4.2 Approximation of the Jacobian

As we remarked in the previous subsection, it is practically impossible to derive an analytical expression for the Jacobian. A numerical approximation of the exact Jacobian is also out the question because of the excessive computational costs: one Jacobian evaluation requires the assembling and solution of the system of equations 2N times. However, it is not necessary to approximate the Jacobian exactly, because the BDF-solver is using, more precisely, a modified Newton method, i.e. the same Jacobian is used in subsequent (Newton) iterations and for several time integration steps. Therefore, we use only a first order approximation of the Jacobian. The derivation of this approximation is outlined in the remaining part of this subsection.

Denote by $\underline{x}_{j,l}$ the vector of all boundary nodes whereby the l-direction ($l =$

1, 2) of the nodal point 'j' is perturbed with a small value, say ε, ($\varepsilon \ll 1$) i.e.

$$\underline{\check{\mathbf{x}}}_{j,l} \;=\; \underline{\mathbf{x}} + \varepsilon \mathbf{e}_{2*j-l} \;=\; \left(x_1^1, x_2^1, \ldots, x_1^j + \varepsilon, x_2^j, \ldots, x_1^N, x_2^N\right)^T. \tag{23}$$

Furthermore, we assume that $\underline{\check{\mathbf{v}}}_{j,l}$ is the solution of the system (18) for this perturbed boundary, thus

$$\mathcal{H}(\underline{\check{\mathbf{x}}}_{j,l})\,\underline{\check{\mathbf{v}}}_{j,l} \;=\; \mathcal{G}(\underline{\check{\mathbf{x}}}_{j,l})\,\underline{\mathbf{b}}(\underline{\check{\mathbf{x}}}_{j,l}). \tag{24}$$

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

$$\left(\mathcal{H} + \varepsilon\,\delta\mathcal{H}_{j,l}\right)(\underline{\mathbf{v}} + \varepsilon\,\delta\underline{\mathbf{v}}_{j,l}) \doteq \left(\mathcal{G} + \varepsilon\,\delta\mathcal{G}_{j,l}\right)(\underline{\mathbf{b}} + \varepsilon\,\delta\underline{\mathbf{b}}_{j,l}). \tag{25}$$

Here, both $\delta\mathcal{H}_{j,l}$ and $\delta\mathcal{G}_{j,l}$ are sparse matrices which are containing the derivatives of the integrals (13), i.e. H^{ij} and G^{ij}, with respect to x_l^j. The non-zero elements of these matrices are the rows '$2j-l$' and '$2j$' and between the columns '$2j-k-l$' and '$2j-k$', where k is equal to 2 in the case of linear elements or when node 'j' is the mid-point of a quadratic element; k is equal to 4 when quadratic elements are applied and node 'j' is one of the corners of the element. The vector $\delta\underline{\mathbf{b}}_{j,l}$ has also the same structure as those matrices. Furthermore, we remark that the vector $\delta\underline{\mathbf{v}}_{j,l}$ is the $(2j-l)^{\text{st}}$ column of the Jacobian $\mathcal{J}$.

Using the exact solution (18), and omitting the higher order terms in (25), we obtain the following first order approximation for the $(2j-l)^{\text{st}}$ column of $\mathcal{J}$,

$$\delta\underline{\mathbf{v}}_{j,l} \doteq \mathcal{H}^{-1}\left(\mathcal{G}\,\delta\underline{\mathbf{b}}_{j,l} + \delta\mathcal{G}_{j,l}\,\underline{\mathbf{b}} - \delta\mathcal{H}_{j,l}\,\underline{\mathbf{v}}\right). \tag{26}$$

The above approximation is not expensive to compute, because when a new Jacobian evaluation is required, LSODE is asking for this Jacobian *after* a call which solves the system of equations (18) for this boundary. Thus the matrix $\mathcal{G}$, the LU-decomposition of the matrix $\mathcal{H}$, and the vectors $\underline{\mathbf{b}}$ and $\underline{\mathbf{v}}$ are already known. And because of the sparsity of the derivative matrices $\delta\mathcal{H}_{j,l}$ and $\delta\mathcal{G}_{j,l}$, and the vector $\delta\underline{\mathbf{b}}_{j,l}$, the computational costs of the approximate $(2j-l)^{\text{st}}$ column of $\mathcal{J}$ are not high.

To illustrate that this approximation is correct, we compare the solutions from the newton iteration, which are obtained by using the approximate Jacobian and the numerical approximation of the exact Jacobian. For the coalescence of two equal circles, using quadratic elements, we have plotted in figure 2 the maximum relative error between both vectors of the result of the newton iteration at the (successful) time integration steps.

As can be seen, the approximated Jacobian is matching in four or more digits after a small period of time. In the initial period, the differences are somewhat larger, which is due to the large curvature and the density of nodes in the region where both circles are touching. This is causing a larger error in the approximated Jacobian. However, we observed no influence of this error in the time stepping or order of the time integrator.

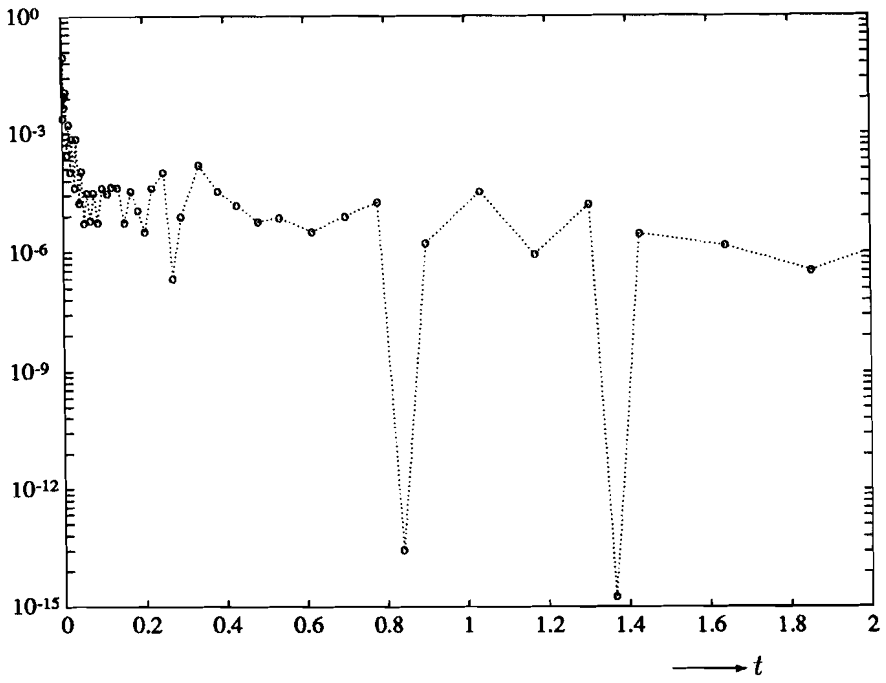

Figure 2: The maximum relative difference between the solutions of the newton iteration at the (successful) time steps, when the approximate Jacobian and the numerical approximation of the exact Jacobian are used.

4.3 Starting the BDF-method after a Node Redistribution

In [10], we presented an algorithm for an optimal node redistribution based on equidistributing the curvature of the boundary. The aim of that algorithm is twofold. Firstly, the number and position of the discretization points are optimized, which is important because the computational costs per time step are proportional to $(2N)^3$, where N is the number of points. Secondly, the algorithm treats regions, where a neck (or cusp) is occurring, in a special way.

After a node redistribution, the (material) points of which the trajectories were being followed, are also changed; i.e. the set of ODEs can completely change its character. This is also illustrated by figure 1, where the "jumps" in the spectral radius are due to this node redistribution.

When we like to resume 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 set by the program. However, we want the BDF-solver to continue with the order and step-size equal to the latest value before the node redistribution was carried out. Fortunately, this is possible and we shall show this below. Before doing this, we first have to dwell on some aspects of the implementation of the BDF-method in LSODE.

The code LSODE is based on the Nordsieck representation of the fixed step size

BDF-methods, see also Gear [2]. For the solution of the ODEs at time $t = t_n$ the original p^{th} order BDF-method needs the actual values at the boundaries at previous times $t_{n-1}, \ldots, t_{n-p}$ and the velocity of the boundary at t_{n-1} as well. When this p^{th} order BDF-method is expressed in the so-called Nordsieck representation, the boundary at $t = t_{n-1}$ and the first till the p^{th} derivative (with respect to t) of this boundary are required.

For the Nordsieck vector $\mathbf{z}_i$ we have,

$$\mathbf{z}_i = \left(\underline{\mathbf{x}}(t_i),\, h\,\underline{\mathbf{x}}^{(1)}(t_i), \ldots,\, h^p\,\underline{\mathbf{x}}^{(p)}(t_i)/p!\right)^{\mathrm{T}}, \tag{27}$$

where (.) is denoting the derivative with respect to time and 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 used as an initial guess for the solution at the next time step, i.e.

$$\underline{\mathbf{x}}(t_i + h) = \underline{\mathbf{x}}(t_i) + h\,\underline{\mathbf{x}}^{(1)}(t_i) + \ldots + h^p\,\underline{\mathbf{x}}^{(p)}(t_i)/p!. \tag{28}$$

This is the starting vector for the Newton iteration.

In order to continue with LSODE after a node redistribution, with the same order and step size as before the redistribution, we have to supply the Nordsieck vector for those new nodes, i.e. the first till the p^{th} derivative (with respect to the t) of these nodal points. We also have to give an approximation of the Jacobian, cf. subsection 4.2, for this new boundary: we have to assemble and solve the system of equations for these new nodes. Because of this, we obtain automatically the first derivative of those nodes for the Nordsieck vector, i.e. the velocity $\underline{\mathbf{v}}$.

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 3,

$$\tilde{\mathbf{x}}(s) = \Phi(s)\mathbf{x}^j, \tag{29}$$

where $-1 \leq s \leq 1$. Since the interpolation matrix Φ is independent of t, the p^{th} derivative with respect to t of the above equation is equal to

$$\tilde{\mathbf{x}}^{(p)}(s) = \Phi(s)\mathbf{x}^{(p)j}. \tag{30}$$

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 polynomial matrix Φ, to influence the new Nordsieck vector. So the degree of the interpolation polynomials has to be so large that the error is smaller than the smallest component of the Nordsieck vector. Because of this we applied a polynomial interpolation with degree three or five, depending on the order of the integrator.

The algorithm used for determining the Nordsieck vector for the new nodes is the following. First, for every new nodal point obtained by the node redistribution algorithm, we seek two successive points from the old grid which are the neighbouring points of this new node, i.e. the new node is lying between those two old points. Then we obtain an approximation of the boundary in this particular region by interpolation through two or three old points at both sides of this new node. Next, we *replace* the new node, by a point that is closest to this node and that is lying on the approximated boundary region, i.e. for a certain value of s, say $s = \tilde{s}$. This point is the new grid point and for this point we compute the Nordsieck vector, i.e. using the old Nordsieck vector and this particular interpolation polynomial for $s = \tilde{s}$.

After this procedure is carried out for the complete grid, we compute the approximate Jacobian and we replace the first derivative of the Nordsieck vector by the (exact) calculated $h\,\underline{v}$. Note that this calculated $\underline{v}$ has a *spatial* discretization error which is induced by the BEM. But we assume that when we are applying quadratic elements for the BEM and the nodal points are distributed "nice", this spatial discretization error is smaller than the smallest component of the Nordsieck vector. Then this spatial discretization error will not influence the initial guess of the nodes at the next time step, which is used for the newton iteration, cf. (28).

5 Numerical Results and Discussion

In this section we shall illustrate the restart of the BDF-method after a node redistribution using the same order and time step as before the redistribution. Again, we consider the evolution of two equal circles and apply quadratic elements.

For the error control in the time integrator LSODE we used a global absolute error tolerance parameter equal to 10^{-4}; the relative error parameter was taken component wise. This relative error was set equal to 10^{-3} for the "smooth" parts of the boundary and equal to 10^{-4} for the nodes in the touching region of both circles. A node redistribution was carried out when the nodal points were coming too close to each other ($=10^{-3}$) and in general after each five consecutive steps.

The coalescence can be described analytically, cf. Hopper [4]. Because of this, a comparison can be made between the derived numerical results and the analytical solution. We observed that those results did agree with the analytical solution. More details are given in [10].

In table 1 the subsequent time steps (t_i) printed for the case that after a node redistribution (nd) the time integration is started without further information, i.e. the order (p) of the method is equal to 1 and the step size (h) is set by LSODE. Here N is the total number of points and $\#\ \mathcal{H}^{-1}\mathcal{G}$ is giving the *total* number of assembling and solving of the system of equations that is carried out *till* the time $t_i + h$; $\#\ \mathcal{J}$ is equal to the total number of Jacobian updates till this time step. By *surface* is denoted the total surface of the fluid region which has to be preserved during the evolution. The (relative) change of this total surface, compared to the surface of the original shape, is also printed. These numbers are showing that the relative error in the surface is caused by the node redistribution algorithm only.

i		N	$t_i + h$	h	p	$\# \mathcal{H}^{-1}\mathcal{G}$	$\# \mathcal{J}$	*surface*	error (%)
1	nd	84	0.0012	0.00118	1	6	2	3.1411	0.0002
2		84	0.0024	0.00118	1	7	2	3.1411	0.0002
3		84	0.0040	0.00163	2	8	2	3.1411	0.0002
4		84	0.0056	0.00163	2	9	2	3.1411	0.0002
5		84	0.0073	0.00163	2	10	2	3.1411	0.0003
6	nd	84	0.0109	0.00369	1	16	4	3.1400	0.1059
7		84	0.0146	0.00369	1	17	4	3.1401	0.1049
8	nd	84	0.0190	0.00432	1	23	6	3.1400	0.1119
9		84	0.0233	0.00432	1	24	6	3.1400	0.1107
10	nd	84	0.0283	0.00505	1	30	8	3.1399	0.1150
11		84	0.0334	0.00505	1	31	8	3.1400	0.1135
12	nd	84	0.0393	0.00589	1	37	10	3.1399	0.1240
13		84	0.0452	0.00589	1	38	10	3.1399	0.1221
14	nd	84	0.0521	0.00693	1	44	12	3.1407	0.0443
15		84	0.0590	0.00693	1	45	12	3.1407	0.0421
16	nd	84	0.0672	0.00821	1	51	14	3.1407	0.0412
17		84	0.0754	0.00821	1	52	14	3.1407	0.0384
18	nd	84	0.0855	0.01007	1	57	16	3.1407	0.0444
19		84	0.0956	0.01007	1	58	16	3.1407	0.0405
20	nd	84	0.1075	0.01193	1	62	17	3.1405	0.0572
21		84	0.1194	0.01193	1	64	17	3.1406	0.0519
22	nd	84	0.1349	0.01545	1	67	18	3.1402	0.0885
23		84	0.1503	0.01545	1	68	18	3.1403	0.0802
24	nd	84	0.1675	0.01721	1	71	19	3.1399	0.1240
25		84	0.1848	0.01721	1	72	19	3.1400	0.1147
26	nd	84	0.2020	0.01729	1	75	20	3.1405	0.0593
27		84	0.2193	0.01729	1	76	20	3.1406	0.0502
28		84	0.2493	0.02995	2	77	20	3.1406	0.0522
29	nd	68	0.2703	0.02102	1	80	21	3.1409	0.0196
30		68	0.2913	0.02102	1	81	21	3.1410	0.0075
31		68	0.3371	0.04578	2	82	22	3.1410	0.0101
32		68	0.3829	0.04578	2	84	22	3.1410	0.0138
33		68	0.4287	0.04578	2	87	22	3.1409	0.0170
34	nd	64	0.4820	0.05336	1	90	23	3.1416	0.0507
35		64	0.5354	0.05336	1	91	23	3.1421	0.0988
36		64	0.6169	0.08149	2	92	23	3.1420	0.0881
37		64	0.6984	0.08149	2	93	23	3.1419	0.0764
38		64	0.7799	0.08149	2	94	23	3.1418	0.0665
39	nd	64	0.8394	0.05958	1	97	24	3.1420	0.0914
40		64	0.8990	0.05958	1	98	24	3.1423	0.1218
41		64	1.0342	0.13518	2	99	25	3.1420	0.0914
42		64	1.1694	0.13518	2	101	25	3.1417	0.0562
43		64	1.3046	0.13518	2	102	25	3.1414	0.0287
44	nd	56	1.3666	0.06201	1	105	26	3.1417	0.0587
45		56	1.4286	0.06201	1	106	26	3.1418	0.0690
46		56	1.6412	0.21265	2	107	27	3.1415	0.0363
47		56	1.8539	0.21265	2	109	27	3.1411	0.0009
48		56	2.0665	0.21265	2	110	27	3.1408	0.0278

Table 1: The time steps for the coalescence of two equal circles when quadratic elements are used. After a node redistribution (*nd*), the time integration is started without further information.

From table 1 we also observe that the order of the BDF-method is almost everywhere equal to one, i.e. a backward Euler method. Furthermore, this table is showing the computational costs of a *restart* caused by a node redistribution: 3-6 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 carelessly.

Table 2 is showing the integration steps for the same problem when the order and step size are set the same as before the node redistribution. Here the Nordsieck vector for the new nodes was found by interpolating the old Nordsieck vector using Lagrangian polynomials with degree five, as was outlined in subsection 4.3.

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. Further, we see a considerable reduction of the total number of assembling and solution of the system of equations and the number of Jacobian updates as well. This is giving a justification for the restarting method we described in subsection 4.3.

Another example to illustrate our numerical scheme is plotted in figure 3. The (initial) fluid region is taken from Hopper [4], figure 6. The caption by this figure is "Does the globe extract itself from the mouth without hitting the walls?" Our numerical results are showing that the globe is extracting itself indeed. However, at later (not plotted) time steps the boundaries were coming too close each other, which caused various kind of errors; this lead to a break-down of the algorithm.

Acknowledgment

This research was supported by the Technology Foundation (STW).

References

[1] Brebbia C.A. and Dominguez J., Boundary Elements An Introductory Course, Computational Mechanics Publications, Southampton, 1989.

[2] Gear C.W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, 1971.

[3] Hindmarsh A.C., LSODE and LSODI, two new initial value ordinary differential equation solvers, ACM-SIGNUM newsletter 15, no. 4, 1980

[4] Hopper R.W., Plane Stokes flow driven by capillarity on a free surface, J. Fluid Mech., Vol. 213, pp. 349-375, 1990.

[5] Kuiken H.K., Viscous sintering: the surface-tension-driven flow of a liquid form under the influence of curvature gradients at its surface, J. Fluid Mech., Vol. 214, pp. 503-515, 1990.

[6] Ladyzhenskaya O.A., The Mathematical Theory of Viscous Incompressible Flow, Gordon and Beach., New York-London, 1963.

i		N	$t_i + h$	h	p	$\# \mathcal{H}^{-1}\mathcal{G}$	$\# \mathcal{J}$	*surface*	error (%)
1	*nd*	84	0.0012	0.00118	1	6	2	3.1411	0.0002
2		84	0.0024	0.00118	1	7	2	3.1411	0.0002
3		84	0.0040	0.00163	2	8	2	3.1411	0.0002
4		84	0.0056	0.00163	2	9	2	3.1411	0.0002
5		84	0.0073	0.00163	2	10	2	3.1411	0.0003
6	*nd*	84	0.0107	0.00342	3	12	3	3.1412	0.0094
7		84	0.0141	0.00342	3	14	3	3.1412	0.0090
8	*nd*	84	0.0175	0.00342	3	17	4	3.1409	0.0210
9		84	0.0210	0.00342	3	18	4	3.1409	0.0212
10		84	0.0277	0.00680	2	19	5	3.1409	0.0217
11	*nd*	84	0.0345	0.00680	2	21	6	3.1409	0.0224
12		84	0.0413	0.00680	2	23	6	3.1409	0.0228
13	*nd*	84	0.0508	0.00944	2	26	7	3.1404	0.0688
14		84	0.0602	0.00944	2	27	7	3.1404	0.0687
15	*nd*	84	0.0697	0.00944	2	29	8	3.1405	0.0615
16		84	0.0791	0.00944	2	31	8	3.1405	0.0617
17	*nd*	84	0.0941	0.01498	2	34	9	3.1406	0.0514
18		84	0.1091	0.01498	2	36	9	3.1406	0.0522
19	*nd*	84	0.1240	0.01498	2	39	10	3.1405	0.0613
20		84	0.1390	0.01498	2	40	10	3.1405	0.0612
21	*nd*	84	0.1604	0.02143	2	43	11	3.1406	0.0512
22		84	0.1819	0.02143	2	46	11	3.1406	0.0520
23	*nd*	84	0.2033	0.02143	2	49	12	3.1406	0.0461
24		84	0.2247	0.02143	2	50	12	3.1406	0.0466
25	*nd*	76	0.2582	0.03348	2	53	13	3.1402	0.0941
26		76	0.2917	0.03348	2	54	13	3.1402	0.0940
27	*nd*	68	0.3252	0.03348	2	57	14	3.1406	0.0535
28		68	0.3586	0.03348	2	58	14	3.1405	0.0550
29		68	0.4116	0.05298	2	60	15	3.1405	0.0603
30		68	0.4646	0.05298	2	61	15	3.1405	0.0636
31		68	0.5176	0.05298	2	62	15	3.1404	0.0685
32	*nd*	72	0.5706	0.05298	2	65	16	3.1403	0.0824
33		72	0.6236	0.05298	2	66	16	3.1402	0.0868
34		72	0.6765	0.05298	2	67	16	3.1402	0.0909
35		72	0.7769	0.10037	2	69	17	3.1400	0.1106
36		72	0.8773	0.10037	2	70	17	3.1398	0.1297
37	*nd*	56	0.9776	0.10037	2	72	18	3.1393	0.1753
38		56	1.0780	0.10037	2	73	18	3.1392	0.1911
39		56	1.2600	0.18201	2	75	19	3.1386	0.2496
40		56	1.4420	0.18201	2	76	19	3.1381	0.3042
41		56	1.6240	0.18201	2	77	19	3.1376	0.3458
42	*nd*	64	1.8645	0.24046	3	80	20	3.1377	0.3371
43		64	2.1050	0.24046	3	81	20	3.1381	0.2952

Table 2: The time steps for the coalescence of two equal circles when quadratic elements are used. After a node redistribution (*nd*), the time integration is restarted with the same order and step size as before the redistribution.

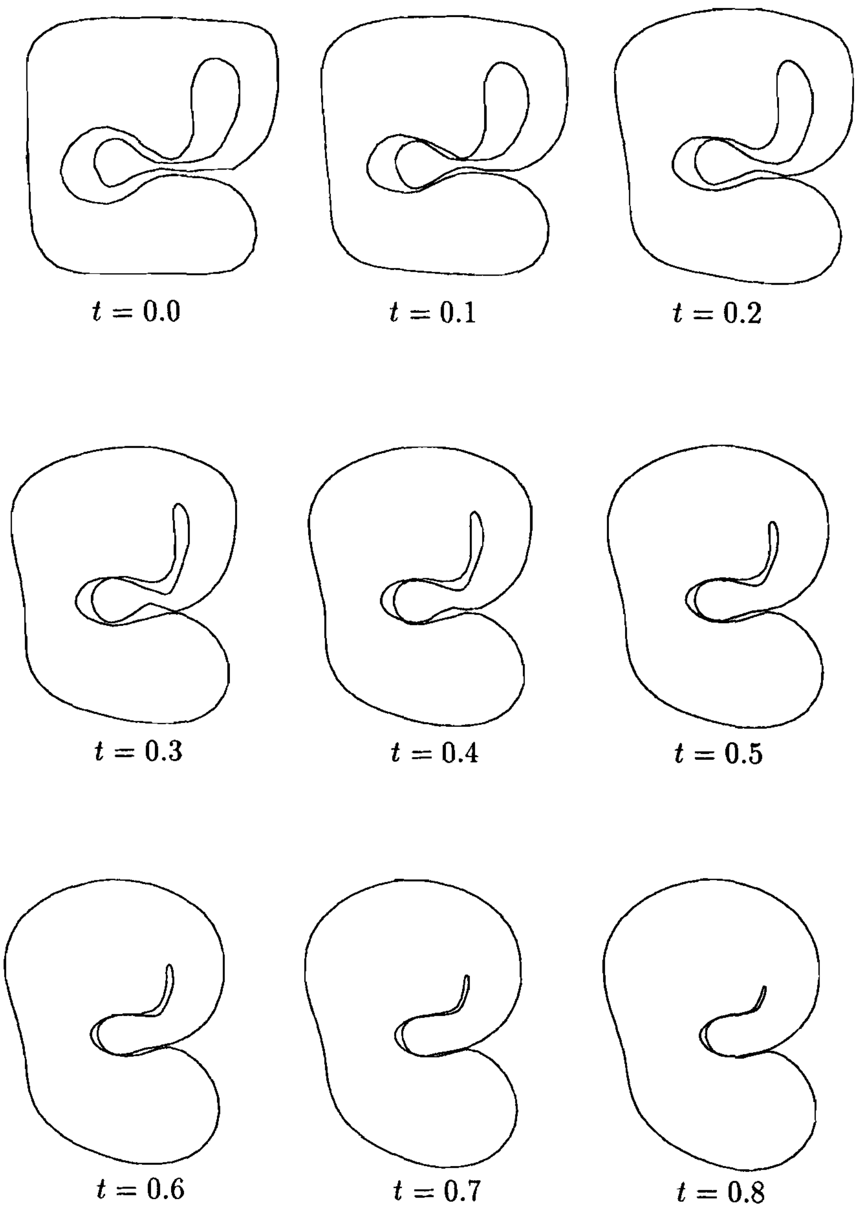

Figure 3: Another example (from Hopper [4], figure 6) of the transformation of a fluid region in time. Hopper plotted this figure with the caption: "Does the globe extract itself from the mouth without hitting the walls?"

[7] Ramsden D., Holloway, G., Timestepping Lagrangian Particles in Two Dimensional Eulerian Flow Fields, J. Comput. Phys., Vol. 94, pp. 101-116, 1992.

[8] Sōmiya S., Moriyoshi Y. (Eds.), Sintering Key Papers, Elsevier Applied Science, London, 1990.

[9] Vorst G.A.L. van de, Mattheij R.M.M., Kuiken H.K., A Boundary Element Solution for Two-Dimensional Viscous Sintering, To appear in J. Comput. Phys., 1992.

[10] Vorst G.A.L. van de, Mattheij R.M.M., Numerical Analysis of a 2-D Viscous Sintering Problem with Non Smooth Boundaries, Submitted to Computing, 1992.