DEVELOPMENT OF A NUMERICAL OPTIMISATION METHOD FOR BLOWING GLASS PARISON SHAPES

J. A. W. M. Groot  
Dept. of Math. and Computer Science  
Eindhoven University of Technology  
PO Box 513, 5600 MB Eindhoven  
The Netherlands  
Email: j.a.w.m.groot@tue.nl

C. G. Giannopapa  
Email: c.g.giannopapa@tue.nl

R. M. M. Mattheij  
Email: r.m.m.mattheij@tue.nl

DEPARTMENT OF COMMERCE AND COMPUTER SCIENCE  
Eindhoven University of Technology  
PO Box 513, 5600 MB Eindhoven  
The Netherlands

ABSTRACT

Industrial glass blowing is an essential stage of manufacturing glass containers, i.e. bottles or jars. An initial glass preform is brought into a mould and subsequently blown into the mould shape. Over the last few decades, a wide range of numerical models for forward glass blow process simulation have been developed. A considerable challenge is the inverse problem: to determine an optimal preform from the desired container shape. A simulation model for blowing glass containers based on finite element methods has previously been developed [14, 15]. This model uses level set methods to track the glass-air interfaces. The model described in a previous paper of the authors showed how to perform the forward computation of a final bottle from the given initial preform without using optimisation. This paper introduces a method to optimise the shape of the preform combined with the existing simulation model. In particular, the new optimisation method presented aims at minimising the error in the level set representing the glass-air interfaces of the desired container. The number of parameters used for the optimisation is restricted to a number of control points for describing the interfaces of the preform by parametric curves, from which the preform level set function can be reconstructed. Numerical applications used for the preform optimisation method presented are the blowing of an axi-symmetrical ellipsoidal container and an axi-symmetrical jar.

KEY WORDS: optimisation, level set methods, finite element methods, glass forming
NOMENCLATURE

\( \alpha \) [-] tolerance
\( c_p \) [J kg\(^{-1}\) K\(^{-1}\)] specific heat
\( g \) [m s\(^{-1}\)] gravitational acceleration
\( h \) [m] height of preform
\( I \) [-] identity matrix
\( J \) [m\(^{-1}\)] Jacobian matrix of residuals
\( k_c \) [W m\(^{-1}\) K\(^{-1}\)] effective conductivity
\( \lambda \) [m\(^{-2}\)] Levenberg-Marquardt parameter
\( N \) [-] number of control points minus two
\( p \) [Pa] pressure
\( \mathbf{p} \) [m] parameter vector
\( \mathbf{r} \) [-] residual vector
\( s \) [m] parameter increment vector
\( \mathbf{u} \) [m s\(^{-1}\)] flow velocity
\( t \) [s] time
\( T \) [K] temperature
\( \mathbf{x} \) [m] position
\( \Gamma_0 \) [-] initial glass-air interfaces
\( \phi \) [-] level set function
\( \Phi \) [-] objective function
\( \Phi^\sim \) [-] weighted objective function
\( \mu \) [kg m\(^{-1}\) s\(^{-1}\)] dynamic viscosity
\( \rho \) [kg m\(^{-3}\)] density
\( \zeta \) [-] sum of the penalty functions
INTRODUCTION

Nowadays glass containers, such as bottles and jars, are an essential part of everyday life. As a consequence, glass manufacturers have to meet the ever growing demand of the consumer society. Optimisation of quality aspects, such as strength, smoothness, weight, shape and cooling conditions, is strived for, together with an increased production speed. Therefore, understanding, controlling and optimising the glass container forming process in glass manufacturing is of crucial importance.

In a glass container forming process the glass melt supplied straight from the furnace is cut into uniform gobs, which are successively brought into a forming machine. Two widely used glass container forming techniques are the blow-blow method and the press-blow method. In both methods first a preform is constructed in a mould by either a press stage or a blow stage. Subsequently, the resulting preform is carried by a ring to another mould for the final blow stage (Fig. 1). During this blow stage the preform first sags under the influence of gravity and then pressurised air is blown into the mould to force the glass into the mould shape. This paper is concerned with the final blow stage of such forming processes.

Computer simulation models have played an important role in gaining more insight in glass forming processes over the last few decades. Whereas measurements are often complicated, for instance because of high temperatures, computer simulation models may offer a good alternative or can be used for comparison of results. Furthermore, computer simulations can be used instead of expensive and time consuming trial-and-error methods with glass forming equipment. The growing interest of glass industry for computer simulation has been a motivation for a fair number of papers on glass blow simulation models [6,7,10,16]. A suitable simulation model allows the user to adapt a range of input data, such as the initial temperature distribution, the inlet air pressure or the preform shape, and subsequently computes the evolution of the relevant physical phenomena in time.

A simulation model for blowing glass containers, based on finite element methods, has been previously developed and tested for industrial applications [14,15]. The model uses level set methods to track the glass-air interfaces. A major advantage of level set methods is that no re-meshing is required to distinguish between the glass domain and the air domain in time; the location of the interface can be marked by the so-called level set function instead [1,8,29,30].

In general glass blow simulation models are able to perform merely the direct computation of the final container from the given initial preform. Consequently, finding the desired glass distribution over the mould wall at
the end of a simulation is usually a trial-and-error procedure of finding the corresponding input settings, such as the shape of the preform and the initial temperature distribution. However, this approach is often not sufficiently efficient, particularly when innovative processes are involved in which practical expertise is no longer useful. In such cases it is interesting to consider an inverse problem corresponding to the blow process, that is to determine the initial conditions, given a final container shape. It should be mentioned that this class of inverse problems is quite challenging, as in general coupled, highly nonlinear systems and complicated geometries are involved. Usually, numerical optimisation methods are employed in order to find a solution to this kind of problems.

Nowadays, optimisation techniques are not unusual in glass forming simulation models. For instance a numerical optimisation method has been introduced in a glass press model to find the optimum tool geometry [24]. Optimisation methods have also been used to estimate heat transfer coefficients or the initial temperature distribution in glass forming simulation models [9, 23]. An engineering approach to find the optimum parison shape for glass blowing have been presented by P. Moreau, D. Lochegnies et al [18, 25]. They combined a computer aided design (CAD) model, a 3D thermomechanical finite element model with adaptive mesh techniques and an optimisation technique based on the Levenberg-Marquardt method. The algorithm attempts to optimise the geometry of the mould for the first blow stage of the blow-blow process, given the glass distribution at the end of the second blow stage. More recently, T. Bernard and E. Ebrahimi Moghaddam developed an optimisation algorithm for predictive control over a class of rheological forming processes [2]. They used a so-called Trouton model for the forming process combined with a gradient based method for optimisation. The aforementioned examples involve highly complicated, coupled, nonlinear problems and consequently reduction of the computational effort is a main issue.

This paper is concerned with an inverse problem corresponding to the final blow stage of a glass container forming process: to determine an optimal shape of the preform, given the container shape. A numerical optimisation method for the inverse problem is introduced in the simulation model. The method aims to find the preform for which the error in the zero level set of the corresponding container is minimal with respect to the desired container. Given a model container and an end time, as well as the height of the preform, the model attempts to find a preform for which the model container can be blown within the stated end time.

PROBLEM FORMULATION

Both the forward problem and the inverse problem for blowing glass parison shapes are formulated. The forward problem considered is to find the position of the glass-air interfaces during a glass blow process. The inverse problem is to find the shape of the preform, given the final container shape.

The Forward Problem

In the mathematical model used for the forward problem, the glass melt is modelled as an incompressible, Newtonian fluid. Since viscous forces dominate, the flow of glass can be described by a Stokes flow problem:

\[
\begin{align*}
\nabla \cdot (\mu \nabla \mathbf{u}) - \nabla p + \rho g &= 0, \\
\nabla \cdot \mathbf{u} &= 0.
\end{align*}
\]

(1)

Because of the strong temperature dependency of the glass viscosity, the flow problem is coupled to an energy
problem involving the heat equation:

\[ \rho c_p \left( \frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T \right) - \nabla \cdot \left( k_c \nabla T \right) = 0. \]  

(2)

Different material properties for glass and air are distinguished by the sign of a level set function \( \phi \). The level set function is a solution of the evolution equation

\[ \frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0. \]  

(3)

So, the glass-air interfaces at time \( t \) are implicitly given by \( \phi(\mathbf{x}, t) = 0 \) [1, 8, 29, 30].

Together (1)-(3) form a fully coupled system of partial differential equations that describe the glass flow during a blow process. Boundary conditions include:

- an inflow pressure at the mould entrance,
- free-stress conditions for air and no-slip conditions for glass on the mould wall,
- 2D axial symmetry conditions on the symmetry axis.

Further details and analysis regarding the problem formulation of the glass blow simulation model are given in [14, 15].

**The Inverse Problem**

The inverse problem can be formulated as follows: find the location of the initial glass-air interfaces, given the glass-air interfaces at time \( t = t^* \). More specifically, find the initial condition for the level set problem \( \phi(\mathbf{x}, 0) = \phi_0(\mathbf{x}) \), such that

\[ \begin{cases} \frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \\ \phi(\mathbf{x}, t^*) = \phi^*(\mathbf{x}), \end{cases} \]  

(4)

where \( \phi^*(\mathbf{x}) \) is the level set function corresponding to the model container. The end time \( t^* \) can be given as any reasonable upper bound of the actual process time. After all, due to the no-slip conditions for glass on the mould wall, the location of the glass-air interfaces does not change anymore from the moment the mould wall is completely covered with glass. However, this would mean that the initial condition is not unique for the problem as it is formulated above, as the level set function at any time \( t > 0 \) represents a possible preform for the model container. Therefore, also the height \( h \) of the preform is given, such that bottom of the preform at the symmetry axis is located at \( z = -h \) with respect to the mould entrance.

**DISCRETISATION METHOD**

A Galerkin finite element method has been used for the discretisation of the forward problem. The 2D axi-symmetrical finite element model has been implemented in COMSOL 3.3 with Matlab.
A level set method is used to capture the glass-air interfaces. This approach allows for the computation of two-fluid flow solutions while using a fixed finite element mesh. Moreover, the level set function naturally deals with topological changes.

A triangulated fast marching method is used as a re-initialisation algorithm to maintain the level set function as a signed distance function [17, 29]. Re-initialisation basically means that the computation of the level set function is stopped at some point in time to rebuild a new level set function. This is necessary, because the signed distance representation will be deformed by the flow velocity, which will lead to additional numerical difficulties. Fast marching methods build the signed distance function outward starting from the zero level set. Further details about the fast marching algorithm used can be found in [14, 15].

**OPTIMISATION STRATEGY**

Subject to optimisation is the location of the glass-air interfaces corresponding to the preform, 

\[ \Gamma_0 := \{ \mathbf{x} \in \Omega \mid \phi_0(\mathbf{x}) = 0 \} \tag{5} \]

where \( \Omega \) is the flow domain. For efficient optimisation it is desirable to restrict the number of parameters to be optimised to a minimum, while still being able to trace the complete shape of the preform. Therefore, the interfaces are described by parametric curves, e.g. splines or Bezier curves. Suppose the inner interface and outer interface are described by parametric curves with control points \( P_0, P_1, \ldots, P_{N_p+1} \) and \( Q_0, Q_1, \ldots, Q_{N_q+1} \), for \( N_p, N_q \in \mathbb{N} \), respectively, as illustrated in Fig. 2 for \( N_p = 1 \) and \( N_q = 4 \). Points \( P_0 \) and \( Q_0 \) are fixed to the ring, and points \( P_{N_p+1} \) and \( Q_{N_q+1} \) are positioned on the symmetry axis. The coordinates of the other points, as well as the \( z \)-coordinate of \( P_2 \), can be adjusted to control the shape of the preform. The \( z \)-coordinate of \( Q_5 \) is fixed to maintain the height of the preform. So the optimisation problem can be reduced to finding optimal positions of the variable control points. A vector \( \mathbf{p} \) comprising the parameters to be optimised is defined by

\[ \mathbf{p} = (r_{P_1}, z_{P_1}, \ldots, z_{P_{N_p}}, r_{Q_1}, z_{Q_1}, \ldots, z_{Q_{N_q}})^T. \tag{6} \]

A similar idea is presented in [18], where the geometry of the mould for the first blow stage is optimised.

The objective of the optimisation method is to fit the computed zero level set \( \hat{\Gamma}_0 \) at end time \( t = T^* \) to the zero level set \( \Gamma^*_0 \) of the model container in the least squares sense. Let \( \hat{\phi}(\mathbf{p}) \) be the solution vector of the forward level set problem and let \( \phi^* \) comprise the corresponding level set values of the model container. Subsequently, let \( \hat{\phi}_\delta(\mathbf{p}) \) and \( \phi^*_\delta \) only comprise those nodal values for which \( \phi^* < \delta \), given a small value \( \delta > 0 \). Then the nonlinear least squares problem aims for minimisation of the objective function

\[ \Phi(\mathbf{p}) = \frac{1}{2} \mathbf{r}(\mathbf{p})^T \mathbf{r}(\mathbf{p}), \tag{7} \]

where \( \mathbf{r}(\mathbf{p}) \) is the residual vector, given by

\[ \mathbf{r}(\mathbf{p}) = \hat{\phi}_\delta(\mathbf{p}) - \phi^*_\delta. \tag{8} \]
Given the parameters \( p \), the solution vector \( \hat{\phi}(p) \) can be computed as follows. First, for each interface, a parametric curve \( z(\alpha; p) \) is defined, where the curve parameter \( \alpha \) varies from 0 to 1. Then a vector \( \hat{\phi}_0 \equiv \hat{\phi}_0(p) \) comprising the initial level set values is constructed. The magnitude of initial level set value \( \hat{\phi}_{0,i} \) equals the distance between nodal point \( x_i \) and the nearest parametric curve. This distance can be computed efficiently by finding the parameter \( \alpha = \alpha_i \) of the Bezier curve, such that the distance to \( x_i \) from the corresponding point on the curve is minimal. For this purpose, a safeguarded method combining Newton’s method and Golden Section Search is applied. In order to determine the sign of \( \hat{\phi}_{0,i} \), the normal vector from \( z_i := z(\alpha_i; p) \) is computed:

\[
n_i := n(\alpha_i; p) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \frac{z'(\alpha_i; p)}{\|z'(\alpha_i; p)\|_2}.
\]

The sign of \( \hat{\phi}_{0,i} \) is \( \pm \text{sign}\left( (x - z_i) \cdot n_i \right) \), where the \( \pm \)-sign is determined by whether the parametric curve represents the inner or the outer interface. Finally, \( \hat{\phi}_0 \) is used as an initial condition in the finite element model to compute \( \hat{\phi}(p) \).

Constraints are imposed to ensure that the preform satisfies reasonable physical conditions. Firstly, the control points of the outer interface should be located at some distance from the mould, except for a small region around the ring. Secondly, the control points of the outer interface should be located further away from the centre of the mould entrance than nearby control points of the inner interface, so that a minimum glass thickness is maintained. Thirdly, all control points should be located at one side of the symmetry axis and the mould entrance. An example is given in the results section. To incorporate the constraints in the nonlinear least squares formulation, a weighted objective function is introduced, given by

\[
\tilde{\Phi}(p) = \Phi(p) + \zeta(p).
\]
Here function \( \zeta \) is the sum of the so-called weighted penalty functions

\[
\zeta(p) = \sum_l \zeta_l(p),
\]

where the weighted penalty functions \( \zeta_l \) are given by

\[
\zeta_l(p) = \frac{w_l}{c_l(p)},
\]

with constraint functions \( c_l(p) > 0 \) and non-negative weights \( w_l [5, 13, 18, 28] \).

The Levenberg-Marquardt method is used to solve the constrained nonlinear least squares problem. A modified Levenberg-Marquardt method is introduced in [28] to account for the weighted penalty functions. In the \( i \)th iteration of this modified Levenberg-Marquardt method, the following system of equations is solved:

\[
\left( J^T J + \lambda^i I + H^i \right) s^i = -J^T r^i + f^i,
\]

where the matrices \( J, f \) and \( H \) comprise the elements

\[
J_{jk} = \frac{\partial r_j}{\partial p_k}, \quad f_k = \frac{\partial \zeta}{\partial p_k}, \quad H_{kl} = \frac{\partial^2 \zeta}{\partial p_k \partial p_l},
\]

for \( j = 1, \ldots, n, \ k, l = 1, \ldots, m \), with \( n \) the number of nodes for which \( \phi^* < \delta \) holds and \( m = 2N_p + 2N_q + 1 \) the number of parameters. The solution \( s^i \) is a parameter increment, which is used to adjust the control points. So, the new parameters are determined by

\[
p^{(i+1)} = p^{(i)} + s^i.
\]

The algorithm stops if

\[
\frac{\|s^i\|_2}{\|p^{(i)}\|_2} < \alpha,
\]

for some tolerance \( \alpha \). The parameter \( \lambda^{(i)} \) in (13) denotes the non-negative Levenberg-Marquardt parameter. It acts as a regularisation parameter in case the Jacobian matrix \( J^{(i)} \) is nearly singular and interpolates between a steepest descent and a Gauss-Newton iteration. If \( \Phi(p^{(i+1)}) > \Phi(p^{(i)}) \), the Levenberg-Marquardt parameter \( \lambda^{(i)} \) is increased and (13) is solved again. It can be proven that the condition \( \Phi(p^{(i+1)}) < \Phi(p^{(i)}) \) is always satisfied for sufficiently large \( \lambda^{(i)} [19] \).

While the gradient vector \( f \) and the Hessian matrix \( H \) can be calculated analytically, the Jacobian matrix \( J \) is not readily available, but has to be approximated each iteration. A finite difference approximation of \( J \) requires...
\( m \) times solving a forward problem, which takes a considerable amount of computational time, if a fair number of control points is needed to represent the interfaces. Therefore, a secant method using a Broyden update of the Jacobian is applied [3, 4, 11, 22]. To this end, a hybrid method is used, which updates the Jacobian matrix after each iteration of the modified Levenberg-Marquardt method by either one of the formulas:

\[
J^{(i+1)} = J^{(i)} + \frac{(\Delta r^{(i)} - J^{(i)} s^{(i)}) \otimes s^{(i)}}{s^{(i)} \cdot s^{(i)}},
\]

if \( \frac{|s^{(i)} \cdot s^{(i-1)}|}{s^{(i)} \cdot s^{(i)}} < \frac{|\Delta r^{(i)} \cdot \Delta r^{(i-1)}|}{|\Delta r^{(i)} \cdot (J^{(i)} s^{(i)})|}, \) \hspace{1cm} (17)

\[
J^{(i+1)} = J^{(i)} + \frac{(\Delta r^{(i)} - J^{(i)} s^{(i)}) \otimes \Delta r^{(i)}}{\Delta r^{(i)} \cdot (J^{(i)} s^{(i)})}, \quad \text{otherwise},
\]

with \( \Delta r^{(i)} = r^{(i+1)} - r^{(i)} \). Here formula (17) is known as Broyden’s good method and formula (18) is called Broyden’s bad method [20, 21]. Secant methods may considerably reduce the computational time, if this is greatly determined by the amount of function evaluations. A drawback of secant methods is that they do not provide an accurate approximation of the Jacobian, which is a requirement for a reliable descent direction and stop criteria in minimisation [11]. As a result, the objective function (10) may fail to converge. B. Ruan suggests to compute the Jacobian by finite differences in the first few iterations and use a secant method in the subsequent iterations [27]. If no convergence is reached after some iterations with the secant method, it is still possible to approximate the Jacobian matrix by finite differences in the optimisation algorithm, if \( \lambda \) becomes too large.

**RESULTS**

As a first example a simple axi-symmetrical glass container with elliptical cross section is considered, e.g. a bowl or a vase. The mould is 10 cm wide and 10 cm high, so that it is still reasonable to consider creep flow (\( \text{Re}_{\text{glass}} \sim 10^{-3} \)). Typical values for this example are given in Tab. 1. Quantities \( R_1 \) and \( R_2 \) denote the inner and outer radius of the ring respectively. For simplicity a constant typical value is chosen for the glass viscosity. The energy problem (2) is not solved in this example. Furthermore, the force term in (1) is disregarded. For the simulations air is replaced by a fictitious fluid with a larger viscosity, e.g. 1 kg m\(^{-1}\) s\(^{-1}\). The viscosity of the fictitious fluid remains much smaller than the glass viscosity, but is sufficiently large to neglect the influence of the inertia term. Hence the flow of the fictitious fluid is described by Stokes flow equations (1).

A non-uniform mesh for the elliptic mould is generated that consists of 17328 elements with 336 boundary elements. Second order Lagrange elements are used for the discretisation of the level set function. Figure 3 shows a typical mesh representation.
<table>
<thead>
<tr>
<th>Quantity</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$</td>
<td>5.0 cm</td>
</tr>
<tr>
<td>$R_2$</td>
<td>5.0 cm</td>
</tr>
<tr>
<td>$R_1$</td>
<td>4.38 cm</td>
</tr>
<tr>
<td>$\mu_{\text{glass}}$</td>
<td>$5 \cdot 10^3$ kg m$^{-1}$ s$^{-1}$</td>
</tr>
<tr>
<td>$\rho_{\text{glass}}$</td>
<td>$2.5 \cdot 10^3$ kg m$^{-3}$</td>
</tr>
<tr>
<td>$\mu_{\text{air}}$</td>
<td>$10^{-5}$ kg m$^{-1}$ s$^{-1}$</td>
</tr>
<tr>
<td>$\rho_{\text{air}}$</td>
<td>1.0 kg m$^{-3}$</td>
</tr>
<tr>
<td>$p_{\text{in}}$</td>
<td>$2.5 \cdot 10^3$ Pa</td>
</tr>
</tbody>
</table>

Table 1. **TYPICAL VALUES FOR BLOWING 2D AXI-SYMMETRICAL ELLIPSOIDAL GLASS CONTAINERS**

Suppose one would like to have the container in Fig. 4 (air is blue, glass is purple). A preform is searched for, for which the model container can be blown with an inlet pressure of $p_{\text{in}} = 2.5 \cdot 10^3$ Pa and within a time duration of $t^* = 0.55$ s.

To parameterise the interfaces cubic splines and Bezier curves are used [12,26] and results are compared with each other. Splines pass through the control points, so that the control points are always located on the interfaces and each change in a parameter exactly represents a displacement of the interface in the corresponding point. Bezier curves do not necessarily pass through the control points, but merely follows a trajectory within the convex hull of the control points. As a result Bezier curves tend to become less wiggly than splines as the arrangement of the control points becomes disorganised. Thus, it can be expected that the shape of a Bezier curve will be less sensitive to changes in the coordinates of the control points. For the preform equal numbers of control points are chosen for the curves, i.e. $N = N_p = N_q$.

First an initial guess of the preform is required for the Levenberg-Marquardt method. For the initial guess the control points of the inner interface and the outer interface are positioned on circles in the cross section of the mould with radius $R_1$ and radius $R_2$, respectively. Initial guesses of the preform are shown in Fig. 5, 6 for Bezier curves and cubic splines, respectively. In the figures the air domain is light blue, the glass domain is purple, the curves and control points are dark blue and the flow velocity field is depicted by arrows.

The constraints are successively given by

$$R_2 - \sqrt{r_{Q_j}^2 + (\gamma_1 z_{Q_j})^2} + \eta_1 > 0,$$

$$\sqrt{r_{Q_j}^2 + z_{Q_j}^2} - \sqrt{r_{P_i}^2 + z_{P_i}^2} - \eta_2 > 0, \quad h + z_{P_N p} - \eta_2 > 0,$$

$$r_{P_i}, r_{Q_j} > 0,$$

$$-(z_{P_i} + \eta_3), -(z_{P_N p} + \eta_3), -(z_{Q_j} + \eta_3) > 0,$$

for $i = 1, \ldots, N_p$, $j = 1, \ldots, N_q$, where $\gamma_1 = 0.5$ is an elliptic constant and $\eta_1 = 0.65$ cm, $\eta_2 = 0.25$ cm, $\eta_3 = 0.5$ cm are constant margins. Constraints (20) do not necessarily include all pairs $(i,j)$, but merely those pairs for...
which points $P_i$ and $Q_j$ are nearest to each other in the initial guess of the preform. Figure 7 gives a sketch of the constraints. The control points are marked as dark dots on the curves. Increments in the control points may not exceed the indicated boundaries.

Each iteration the control points are adjusted and the new glass container is computed from the corresponding preform. The difference between the obtained glass container and the model container in Fig. 4 is computed to determine the new value of the objective function. Figure 8-9 show final results of the optimisation method. Figure 8 shows results for glass-air interfaces described by Bezier curves and Figure 9 for glass-air interfaces described by cubic splines. Subfigures (a)-(c) show the preforms and subfigures (d)-(f) show the resulting glass containers, for three, four and five control points, respectively. It can be observed that all obtained glass containers, particularly those in Fig. 8 (f), 9 (e), 9 (f), are fair approximations of the model container in Fig. 4. Results tend to be slightly better for cubic splines than for Bezier curves for more than two control points. The diversity in
Figure 6. (a) INITIAL GUESS OF PREFORM FOR CUBIC SPLINES WITH FIVE CONTROL POINTS PER INTERFACE (b) RESULTING GLASS CONTAINER.

Figure 7. CONSTRAINTS ON THE CONTROL POINTS.

preform shapes for similar containers indicates how sensitive the inverse problem can be to small changes in the glass container, which may considerably complicate the search for an optimal preform.

Figure 10 shows convergence results for Bezier curves and cubic splines for different numbers of control points. The Euclidean norm of the residual vector (8) is plotted as a function of the iterations. It appears that if four or five control points are involved, better convergence is obtained for Bezier curves than for cubic splines. An explanation for this may be that the shape of a Bezier curve is less sensitive to increments in the parameters. This can for instance be seen by comparing the curves in Fig. 8(c) and Fig. 9(c). Furthermore, for Bezier curves better convergence is obtained as more control points are involved. For Bezier curves with four or five control points only 5 iterations are required to reduce the residual norm to thirty percent of the initial residual norm. For cubic splines there is no obvious difference in convergence rate for different numbers of control points.

Figure 11 shows bar graphs with the final Euclidean norm of the residuals obtained for successively three, four and five control points, for Bezier curves (left) and cubic splines (right). For the case of cubic splines with three control points the final residuals are not those for which the norm is minimal, since the stop criterium is satisfied.
at a later stage when the residuals have started growing again (see Fig. 10). This explains the large residuals obtained for this case. For four and five control points better results are obtained for cubic splines than for Bezier curves, although faster convergence is obtained for Bezier curves.

<table>
<thead>
<tr>
<th>Curves</th>
<th>Points</th>
<th>SM or FD</th>
<th>$n_f$</th>
<th>$n_i$</th>
<th>$\Phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bezier curves</td>
<td>4</td>
<td>SM/FD</td>
<td>32</td>
<td>8</td>
<td>1.75</td>
</tr>
<tr>
<td>Bezier curves</td>
<td>5</td>
<td>SM/FD</td>
<td>13</td>
<td>5</td>
<td>1.44</td>
</tr>
<tr>
<td>Cubic splines</td>
<td>4</td>
<td>SM/FD</td>
<td>66</td>
<td>17</td>
<td>0.65</td>
</tr>
<tr>
<td>Cubic splines</td>
<td>5</td>
<td>SM/FD</td>
<td>62</td>
<td>16</td>
<td>1.05</td>
</tr>
<tr>
<td>Bezier curves</td>
<td>4</td>
<td>FD</td>
<td>98</td>
<td>9</td>
<td>1.36</td>
</tr>
</tbody>
</table>

Table 2. PERFORMANCE OF THE OPTIMISATION ALGORITHM
Figure 9. (a)-(c) OPTIMAL PREFORM; (d)-(f) RESULTING GLASS CONTAINER; CUBIC SPLINES ARE USED WITH THREE \{(a), (d)\}, FOUR \{(b), (e)\} AND FIVE \{(c), (f)\} CONTROL POINTS PER INTERFACE.

Figure 10. CONVERGENCE OF THE OPTIMISATION METHOD

The computational effort of the optimisation method is mainly determined by the amount of function evaluations, i.e. the number of times the glass container shape is computed. For this example a finite difference approximation of the Jacobian costs $m = 4N + 1$ function evaluations, which means that the optimisation algo-
Figure 11. BAR GRAPH OF THE RESIDUALS

The algorithm is likely to take more computational time for more parameters. Table 2 shows that this is often not the case if a secant method with only occasionally a finite difference approximation (SM/FD) is applied. Here \( n_f \) is the number of function evaluations and \( n_i \) is the number of iterations. By way of illustration, also a result for finite difference (FD) approximations is given. Here Bezier curves with four control points are used. Although in comparison to SM/FD the numbers of iterations required do not differ much, FD requires many more function evaluations.

Finally, the optimisation method is applied to find the preform of an axi-symmetrical jar. Consider the model jar in Fig. 12. The mould for the jar has a diameter of 10 cm at its entrance and is 15.6 cm high. For the simulations a mesh is used that consists of 5420 elements with 228 boundary elements (see Fig. 13). A preform is constructed such that the glass form does not break during the blow process, since this may result in a significant initial difference between the resulting jar and the model jar and moreover lead to additional numerical difficulties. Such a preform can easily be found by ensuring the glass is sufficiently thick, even though in this way the absolute difference in mass between the preform and the model jar may increase. For the parametrisation of the interfaces cubic splines are chosen with four control points for the inner interface and five control points for the outer interface. Horizontal tangents are enforced at the symmetry axis. Figure 14(a) shows the initial guess of the preform and the resulting jar.

Suppose the jar is obtained from the preform by first letting the preform sag for 1 s and then blowing air in the mould with a pressure of \( 3.3 \cdot 10^3 \) Pa for 3.6 s. In this way the jar in Fig. 14(b) is obtained by blowing the initial guess of the preform. Note that the glass does not completely cover the mould wall at \( t = 4.6 \) s because the thick preform brings about a relatively high resistance against flow during the blow process.

Application of the optimisation method leads to the preform in Fig. 15(a). Only 12 iterations are required until convergence if \( \alpha = 0.05 \) in stop criterium (16). The mean distance between the glass-air interfaces of the jar resulting from the optimal preform in Fig. 15(b) and the glass-air interfaces of the model jar in Fig. 12 is 0.093 cm and the maximum distance, which is obviously found at the symmetry axis, is 0.519 cm. The thickness near the symmetry axis can be further reduced by putting aside the condition that the tangent of the splines at the symmetry axis should be horizontal or by increasing the number of control points. The mean distance obtained is less than
one percent of the diameter of the mould entrance, which is an encouraging result.

Figure 12. **THE MODEL JAR**

Figure 13. **A TYPICAL MESH REPRESENTATION FOR A JAR**

Figure 14. (a) **INITIAL GUESS OF PREFORM**  (b) **RESULTING JAR.**
CONCLUSIONS

A method to optimise the preform in blowing glass containers (e.g. jars or bottles) has been presented. The optimisation method uses a modified Levenberg-Marquardt algorithm combined with a secant method. A 2D axi-symmetrical simulation model previously developed is applied to compute the shape of the glass container. The simulation model is based on finite element methods and uses a level set method to track the glass-air interfaces. Numerical examples for the preform optimisation method presented are the blowing of a 2D axi-symmetrical ellipsoidal glass container, e.g. a bowl or a vase, and of a 2D axi-symmetrical jar.

The number of parameters used for optimisation is restricted to a number of control points for describing the interfaces of the preform by parametric curves. Bezier curves and cubic splines have been tested in the optimisation algorithm. Bezier curves tend to lead to faster convergence than cubic splines as more control points are involved, although in the examples cubic splines eventually give a smaller residual.

The significant diversity in preform shapes for similar glass containers gives a good indication of the sensitivity of the inverse problem.

The computational effort does in general not increase significantly if the number of control points is increased up to five. This is partly because the secant method avoids time consuming computations of the Jacobian matrices by finite differences, which is a considerable saving on the amount of function evaluations in case of many parameters.

By means of the optimisation method a jar has been obtained with a mean distance from the desired jar of less than one percent of the diameter of the mould entrance, which gives confidence that the optimisation method can be used for practical applications.
REFERENCES


