On the usage of NURBS as interface representation in free-surface flows

S. Elgeti*.,†, H. Sauerland, L. Pauli and M. Behr

Chair for Computational Analysis of Technical Systems, CCES, RWTH Aachen University, 52056 Aachen, Germany

SUMMARY

When simulating free-surface flows using the finite element method, there are many cases where the governing equations require information which must be derived from the available discretized geometry. Examples are curvature or normal vectors. The accurate computation of this information directly from the finite element mesh often requires a high degree of refinement—which is not necessarily required to obtain an accurate flow solution. As a remedy and an option to be able to use coarser meshes, the representation of the free surface using non-uniform rational B-splines (NURBS) curves or surfaces is investigated in this work. The advantages of a NURBS parameterization in comparison with the standard approach are discussed. In addition, it is explored how the pressure jump resulting from surface tension effects can be handled using doubled interface nodes. Numerical examples include the computation of surface tension in a two-phase flow as well as the computation of normal vectors as a basis for mesh deformation methods. For these examples, the improvement of the numerical solution compared with the standard approaches on identical meshes is shown. Copyright © 2011 John Wiley & Sons, Ltd.

KEY WORDS: free-surface flows; non-uniform rational B-splines; surface tension; normal computation; discontinuous pressure; finite element method

1. INTRODUCTION

Many applications suitable for finite element simulations require the handling of free-surface and two-phase flows. The main tasks in this context are to account for the interface, which is generally in motion, the surface tension effects, and to represent discontinuities in field variables across the interface.

One can distinguish between two approaches for the description of the interface: interface-capturing and interface-tracking. Examples of interface-capturing are level-set and volume of fluid (VOF) methods [1, 2]. Here, the computational mesh remains fixed and the interface is implicitly described by an additional scalar field. This flexible interface description has the advantage of allowing topological changes, but presents difficulties when resolving discontinuous field variables across the interface. Typically, adaptive mesh refinement approaches are therefore applied close to the interface. Interface-tracking, on the other hand, describes the interface explicitly with an aligned mesh adapting the motion of the interface [3, 4]. The treatment of topological changes is not straightforward, but besides this issue the method can accurately describe the interface and interfacial jump conditions if required.

*Correspondence to: S. Elgeti, Chair for Computational Analysis of Technical Systems, CCES, RWTH Aachen University, 52056 Aachen, Germany.
†E-mail: elgeti@cats.rwth-aachen.de

Copyright © 2011 John Wiley & Sons, Ltd.
With regard to the consideration of surface tension effects, the difficult part is the determination of the interface curvature. In interface-tracking, the curvature cannot be obtained directly from a piecewise linear interface mesh. One option is to approximate the interface with a sufficiently smooth curve that enables the calculation of the curvature. Another possibility is the reformulation of the surface tension term using the Laplace–Beltrami operator [5]. Ganesan et al. [6] compared different approaches and proposed to use a cubic spline approximation of the interface to calculate the curvature. In [7], a discontinuous-Galerkin interface-tracking method is introduced in the context of two-phase flows. Although higher order polynomials are used for the interface representation in the mesh, B-Spline interpolation is additionally applied in order to accurately obtain the interface curvature. Similar approaches can also be found in previous works on front-tracking or interface-capturing methods [8–11]. Hughes et al. [12] introduced an approach representing the geometry and the solution space in terms of NURBS functions.

The pressure jump across the interface due to the surface tension effects is another difficulty in two-phase flows. In interface-capturing methods and interface-tracking using continuous approximation spaces, jumps in the field variables are smeared out as they cannot be represented appropriately (see e.g. [6]). The use of discontinuous approximation spaces can be advantageous [6, 13, 14].

In this paper we present a robust interface-tracking finite element method, designed for two-phase incompressible flows including surface tension effects. We use P1P1 finite elements with least-squares stabilization. For mesh deformation, the DSD/SST (Deforming Spatial Domain/Stabilized Space–Time) procedure is used [15]. This approach is based on the discontinuous-Galerkin method in time (space–time elements), details of which can be found in [16, 17]. The variational formulation of the problem is written over the associated space–time domain, accounting for possible deformations of the computational domain. The interface is approximated using non-uniform rational B-splines (NURBS) that allow a straightforward calculation of the interface curvature and normal vectors, enabling the evaluation of the surface tension forces and the treatment of the interface mesh movement in the normal direction. When surface tension effects are considered, the pressure field across the interface is discontinuous. This pressure jump is accounted for by equipping the interface nodes with an additional degree of freedom. Jumps across the interface can thereby be represented accurately. Numerical examples show the accurate evaluation of the surface tension forces. A die swelling test case is given as an example for an industrially relevant application of the method. The numerical test cases compare the performance of the NURBS approach with the standard approach.

The paper is structured as follows: In Section 2, the governing equations are introduced. Their discretization is described in Section 3. Section 4 deals with the NURBS representation of the interface. In Section 5, the treatment of pressure discontinuities is described. Numerical examples are given in Section 6 and final conclusions are drawn in Section 7.

2. GOVERNING EQUATIONS

In general, the governing equations for incompressible fluid flow are the instationary, incompressible Navier–Stokes equations. Consider an instationary fluid flow problem with any number (in this case one or two) of immiscible Newtonian phases. The computational domain, denoted by \( \Omega \), is a subset of \( \mathbb{R}^n_{ad} \), where \( n_{ad} \) is the number of space dimensions. Then at each instant \( t \in (0, T) \), the velocity, \( \mathbf{u}(\mathbf{x}, t) \), and the pressure, \( p(\mathbf{x}, t) \), in each phase are governed by the following equations:

\[
\rho_i \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \mathbf{f} \right) - \nabla \cdot \mathbf{\sigma}_i = 0 \quad \text{on } (\Omega_i) \quad \forall t \in (0, T) \tag{1}
\]

\[
\nabla \cdot \mathbf{u} = 0 \quad \text{on } (\Omega_i) \quad \forall t \in (0, T) \tag{2}
\]
for $i = 1, \ldots$, number of phases. $\rho_i$ is the density of the respective fluid and the stress tensor $\sigma_i$ is defined as

$$\sigma_i(u, p) = -pI + 2\nu_i \varepsilon(u) \quad \text{on } (\Omega_i)_t, \quad i = 1, 2,$$

(3)

$$\varepsilon(u) = \frac{1}{2}(\nabla u + (\nabla u)^T),$$

(4)

where $\nu_i$ is the kinematic viscosity.

In the actual implementation, the computational domain is divided into parts $(\Omega_i)_t$, each occupied by fluid $i$. Note that the spatial domain is time-dependent, which is indicated by the subscript $t$. A subdomain may never be occupied by more than one fluid. The interface of two subdomains, $\partial\Omega_1 \cap \partial\Omega_2$, is denoted by $\Gamma^{\text{int}}_t$.

At the interface between two phases, we impose a boundary condition which is based on the Laplace–Young equation:

$$\sigma n = \gamma kn.$$

(5)

Here, $n$ is the outward unit normal vector on $\Gamma^{\text{int}}_t$, $\gamma$ the surface tension coefficient and $k$ the local curvature of $\Gamma^{\text{int}}_t$. Furthermore, the velocities are assumed to be continuous across the interface.

For creeping flows (i.e. Reynolds number $\ll 1$), the advective term in the Navier–Stokes equations is often neglected, giving rise to the Stokes equations. In the stationary case, the Stokes equations can be written as follows:

$$-\nabla \cdot \sigma_i = f \quad \text{on } \Omega_i,$$

(6)

$$\nabla \cdot u = 0 \quad \text{on } \Omega_i.$$

(7)

The constitutive equations for the Newtonian fluids remain the same as above.

3. SOLUTION TECHNIQUE

For the discretization of Equations (1) and (2), we use P1P1 finite elements, i.e. linear interpolation for both the velocity and pressure degrees of freedom. This combination of interpolation functions is known to violate the Ladyzhenskaya–Babuska–Brezzi (LBB) compatibility condition. Consequently without an appropriate stabilization, the pressure field is likely to present spurious and oscillatory results. The stabilization technique used here is Galerkin/least-squares (GLS) stabilization. In the GLS method, the stabilization term consists of an element-by-element weighted least-squares form of the original differential equation [18].

Since the considered computational domains are intended to be deformable over time, this also needs to be included into the formulation. For the presented applications, the domain deformation is lead by the necessity to track the unknown position of the free surface. In the past, two different approaches have emerged to deal with such problems:

interface-tracking (IT): The computational mesh is aligned with the moving interface and follows the movement of the interface.

interface-capturing (IC): The computational mesh remains stationary, and a special indicator specifies which elements are filled with fluid, empty, or contain the interface.

In the presented cases, the deformation of the free surface is sufficiently moderate and does not involve irregular motion or topological changes. Hence, interface-tracking as an accurate way to describe the interface has been chosen. In this context, a very successful representation of deforming domain problems is the arbitrary Lagrangian–Eulerian (ALE) formulation. It was initially stated in the finite difference context by Hirt [19], and later adopted also in the finite element community—see [3, 20] and the references contained therein. In the modern ALE approaches, velocities of the nodes of the computational mesh explicitly enter the momentum equation, which is written over
a reference domain. The ALE philosophy found another expression in the DSD/SST procedure [15], which is used here. Its basis is the space–time method, details of which can be found in [16, 17].

In order to construct the finite element function spaces for the space–time method, the time interval is divided into subintervals $I_n = (t_n, t_{n+1})$, with $t_n$ and $t_{n+1}$ representing an ordered series of time levels $0 = t_0 < t_1 < \cdots < t_N = T$. Now, if $\Omega_n = \Omega_{n+1}$, the space–time slab $Q_n$ is defined as the domain enclosed by the surfaces $\Omega_n, \Omega_{n+1}$ as well as the surface described by $\partial\Omega_n$ as $t$ traverses $I_n$, which shall be named $P_n$. For each space–time slab, the following finite element interpolation and weighting function spaces are defined for the velocity and the pressure:

\[
(\mathcal{V}_u^h)_n = \{ u^h | u^h \in [H^{1h}(Q_n)]^{n,d}, u^h = \mathbf{g}^h \quad \text{on} \quad (P_n)_g \},
\]

\[
(\mathcal{V}_w^h)_n = \{ w^h | w^h \in [H^{1h}(Q_n)]^{n,d}, w^h = 0 \quad \text{on} \quad (P_n)_g \},
\]

\[
(\mathcal{V}_p^h)_n = \{ p^h | p^h \in H^{1h}(Q_n) \}.
\]

The interpolation functions in the elements are first-order polynomials in space as well as in time. For each fluid domain, the interpolation functions are continuous in space, but discontinuous in time.

### 3.1. Variational form

The stabilized space–time formulation of the incompressible Navier–Stokes equations (1) and (2) for deforming domains can be expressed as follows: Given $(u^h)_n^-$, find $u^h \in (\mathcal{V}_u^h)_n$ and $p^h \in (\mathcal{V}_p^h)_n$ such that $\nabla w^h \in (\mathcal{V}_w^h)_n$, $\forall q^h \in (\mathcal{V}_p^h)_n$:

\[
\int_{Q_n} w^h \cdot \rho_1 \left( \frac{\partial u^h}{\partial t} + u^h \cdot \nabla u^h - f \right) dQ + \int_{Q_n} \varepsilon(w^h) \cdot \sigma_i(p^h, u^h) dQ \]
\[
+ \int_{Q_n} q^h \nabla \cdot u^h dQ + \int_{\Omega_n} (w^h)_n^+ \cdot \rho_1 ((u^h)_n^+ - (u^h)_n^-) d\Omega \]
\[
+ \sum_{e=1}^{(n_e)_h} \int_{Q_n} \tau_{\text{MOM}} \left[ \rho_1 \left( \frac{\partial w^h}{\partial t} + u^h \cdot \nabla w^h \right) - \nabla \cdot \sigma_i(q^h, w^h) \right] \cdot \rho_1 \left( \frac{\partial u^h}{\partial t} + u^h \cdot \nabla u^h - f \right) - \nabla \cdot \sigma_i(p^h, u^h) \right] dQ + \sum_{e=1}^{(n_e)_h} \int_{Q_n} \tau_{\text{CONT}} \nabla \cdot w^h \rho_1 \nabla \cdot u^h dQ
\]
\[
= \gamma \int_{(P_h)_{\text{int}}} \kappa w^h \cdot \mathbf{n} dP.
\]

In the above equation, the following notation is used:

\[
(u^h)_n^\pm = \lim_{\varepsilon \to 0} u(t_n \pm \varepsilon),
\]

\[
\int_{Q_n} \ldots dQ = \int_{t_n} \int_{\Omega_n} \ldots d\Omega dt,
\]

\[
\int_{P_n} \ldots dP = \int_{t_n} \int_{\Omega_n} \ldots d\Gamma dt.
\]

The problem is solved sequentially for each space–time slab, starting with:

\[
(u^h)_0^+ = u_0.
\]
Details on the method and its parameters $\tau_{\text{MOM}}$ and $\tau_{\text{CONT}}$ can be found in [21]. The variational form of the steady Stokes equations (6) is a straightforward adaption of (11).

### 3.2. Mesh movement rules

In DSD/SST, the deformation of the spatial domain is taken into account automatically by writing the variational formulation of the problem over the associated space–time domain. The only restriction on the movement is imposed by the boundary nodes being related to the interface motion. The inner nodes can be moved in an arbitrary way. The condition that has to be met at the interface is

$$ v \cdot n = u \cdot n $$

with $v$ being the velocity of the nodes on the boundary. This condition leaves several possible options for the choice of $v$, as the tangent components can be chosen freely. Here, $v$ is chosen to be equal to the normal component of $u$. Moving the boundary nodes in this way satisfies the kinematic condition, but at the same time, node clustering due to strong movement in the tangential direction is avoided.

As long as the movement of the boundary nodes stays within a certain range, the mesh can be equipped with the necessary flexibility to account for the mesh motion without the need for remeshing. For this purpose, we use a moving elastic mesh technique, see e.g. [22].

### 4. NURBS REPRESENTATION OF THE INTERFACE

Key aspects in two-phase and free-surface flows strongly related to an accurate interface representation are surface tension and normal computation. The surface tension force (5) is proportional to the curvature of the interface. It is a delicate task to calculate the curvature explicitly, due to the discretization of the interface with the finite element mesh. One possibility to circumvent this problem is the reformulation of the surface tension term using the Laplace–Beltrami operator as proposed in e.g. [5]. This formulation does not require the interface curvature, but can also drastically reduce the convergence order [6]. Another issue is the calculation of the normal vectors at the nodes, required for the kinematic condition of the mesh movement as introduced in Section 3.2. The normal vectors are not uniquely defined at the nodes. One common possibility is to simply calculate the normal vector at a node as the mean value of the normal vectors of the element faces sharing that node [23] or calculate an averaged normal vector by taking into account the size of the neighboring element faces [24].

We are proposing a different approach for the calculation of the interface curvature and normal vectors. The interface is approximated using a NURBS. The curvature and normal vectors can then be calculated directly from the NURBS. The following section recalls the basics of NURBS and describes how the interface approximation is realized.

#### 4.1. NURBS curves and surfaces

Initially, NURBS curves were mainly of interest in the computer-aided design community. Here, they became the standard for curve description. Yet, NURBS have also become an industry standard when it comes to the representation or exchange of geometrical information.

A NURBS curve of degree $K$ in $\mathbb{R}^2$ is defined as [25]

$$ C(\tau) = \frac{\sum_{i=0}^{n} N_{i,K}(\tau) w_i P_i}{\sum_{i=0}^{n} N_{i,K}(\tau) w_i}, \quad 0 \leq \tau \leq 1 $$

with $P_i \in \mathbb{R}^2$ denoting the control points, $w_i$ the weights of each control point and $\tau$ the parameterization of the curve. The $K$th degree B-spline basis functions $N_{i,K}(\tau)$ are recursively defined.
as [25]

\[ N_{i,0}(\tau) = \begin{cases} 
1 & \text{if } t_i \leq \tau < t_{i+1}, \\
0 & \text{otherwise}, 
\end{cases} \]  

(18)

\[ N_{i,K}(\tau) = \frac{\tau-t_i}{t_{i+K}-t_i}N_{i,K-1}(\tau) + \frac{t_{i+K+1}-\tau}{t_{i+K+1}-t_{i+1}}N_{i+1,K-1}(\tau), \]  

(19)

on a so-called knot vector \( T = \{ t_0, \ldots, t_m \} \), \( t_i \leq t_{i+1} \) for \( i = 0, \ldots, m-1 \). A closed NURBS curve can be obtained by overlapping \( K \) control points at the ends \( (P_{n-K+1} = P_0, \ldots, P_n = P_{K-1}) \).

The extension to NURBS surfaces is straightforward. A NURBS surface of degree \((K, L)\) in \( \mathbb{R}^3 \) is given by Piegl and Tiller [25]

\[ S(\tau, \varphi) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{o} N_{i,K}(\tau)N_{j,L}(\varphi)w_{i,j}P_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{o} N_{i,K}(\tau)N_{j,L}(\varphi)w_{i,j}}, \quad 0 \leq \tau, \varphi \leq 1. \]  

(20)

\( N_{i,K}(\tau) \) and \( N_{j,L}(\varphi) \) are defined on different knot vectors and parameterizations in the direction \( \tau \) and \( \varphi \). \( n \times o \) number of control points \( P_{i,j} \in \mathbb{R}^3 \) are required.

### 4.2. Curve/surface fitting

As mentioned before, we aim to obtain a NURBS representation of the fluid interface. Global gradient-based algorithms are used in this work to obtain a NURBS approximation of the interface. We will restrict the discussion to the two-dimensional case for simplicity. Given are: the interface grid nodes \( G_j \), \( 1 \leq j \leq L \), and a number of control points \( P_{i,j} \in \mathbb{R}^3 \) are required.

We will restrict the discussion to the two-dimensional case for simplicity. Given are: the interface grid nodes \( G_j \), \( 1 \leq j \leq L \), and a number of control points \( P_{i,j} \in \mathbb{R}^3 \) are required.

As initial conditions a uniform knot vector

\[ T_i = \frac{i - K}{n + 2 - K}, \quad i = 0, \ldots, n + K + 1 \]  

(22)

a uniform parameterization

\[ \tau_0 = 0, \quad \tau_m = 1, \quad \tau_j = \frac{j}{m}, \quad j = 1, \ldots, m - 1 \]  

(23)

and weights \( \{ w_i \}_{i=0}^{n} = 1 \) are chosen. It turned out that a uniform knot vector (22), generally leads to a faster convergence of the fitting algorithm in comparison with, e.g. a chord length approach [25, 27]. The Polak–Ribiere conjugate gradient algorithm consists of two steps, where in the first step, the positions of the control points are determined. This step does not require initial conditions.

This fitting procedure leads to a definite relation between the interface grid nodes and the NURBS curve—the NURBS curve at the local coordinates \( \tau_j \), \( C(\tau_j, P, w) \), is fitted to the interface grid nodes \( G_j \). Hence, quantities calculated from the NURBS representation (curvature, normal vectors) can be used directly in the mesh-based finite element calculations.
4.3. Curvature and normal vectors of NURBS

The curvature and normal vectors of the interface nodes can be calculated directly from the NURBS approximation of the interface. From differential geometry, the curvature of a parametric curve \( C(\tau) \) can be calculated with [28]:

\[
\kappa(\tau) = \frac{|C'(\tau) \times C''(\tau)|}{|C'(\tau)|^3}.
\]  

(24)

\( C'(\tau) \) and \( C''(\tau) \) denote the first and second derivatives of the curve with respect to parameter \( \tau \). For the sake of brevity, \( N_{i,K}(u) \) is written as \( N_{i,K} \) and \( \sum_{i=0}^{n} \) as \( \sum_i \) in the following equations:

\[
C'(\tau) = \frac{\sum_i N_{i,K}' w_i P_i \sum_i N_{i,K} w_i - \sum_i N_{i,K} w_i P_i \sum_i N_{i,K}' w_i}{(\sum_i N_{i,K} w_i)^2}.
\]  

(25)

\[
C''(\tau) = \frac{\sum_i N_{i,K}'' w_i P_i \sum_i N_{i,K} w_i - \sum_i N_{i,K} w_i P_i \sum_i N_{i,K}'' w_i}{(\sum_i N_{i,K} w_i)^2} - \frac{2 \sum_i N_{i,K}' w_i (\sum_i N_{i,K}' w_i P_i \sum_i N_{i,K} w_i - \sum_i N_{i,K} w_i P_i \sum_i N_{i,K}' w_i)}{(\sum_i N_{i,K} w_i)^3}.
\]  

(26)

\( N_{i,K}' \) and \( N_{i,K}'' \) are the first and second derivatives of the basis functions with respect to the parameter \( \tau \). Furthermore, the calculation of the surface tension force and the mesh movement, cf. Section 3.2, requires the normal vectors of the NURBS curve. They can easily be obtained from the tangent vector \( t \), given by the normalized derivative of the curve \( C'(\tau) \) (here: in \( \mathbb{R}^2 \)) [28]:

\[
n(\tau) = \begin{pmatrix} -t_y \\ t_x \end{pmatrix} \quad \text{with} \quad t(\tau) = \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \frac{C'(\tau)}{|C'(\tau)|}.
\]  

(27)

The curvature of NURBS surfaces is calculated as the sum of the principal curvatures in directions \( \tau \) and \( \zeta \), each evaluated with Equation (24). The normal vectors on a surface can easily be calculated as the cross-product of a binormal and a tangent vector [28]:

\[
n(\tau, \zeta) = \frac{b(\tau, \zeta) \times S'(\tau, \zeta)}{|b(\tau, \zeta) \times S'(\tau, \zeta)|} \quad \text{with} \quad b(\tau, \zeta) = \frac{S'(\tau, \zeta) \times S''(\tau, \zeta)}{|S'(\tau, \zeta) \times S''(\tau, \zeta)|}.
\]  

(28)

\( S'(\tau, \zeta) \) and \( S''(\tau, \zeta) \) denote the first and second derivatives of the NURBS surface with respect to \( \tau \) or \( \zeta \).

5. PRESSURE DISCONTINUITY

In immiscible two-phase flows, due to the density and viscosity differences, a kink in the velocity and pressure fields occurs across the interface. If furthermore surface tension effects are considered, the pressure field may be discontinuous. An inappropriate handling of this pressure jump gives rise to spurious velocities [6]. Standard finite element shape functions cannot account for the discontinuous pressure field appropriately, as they themselves are continuous. The pressure jump is therefore spread out over several elements, see Figure 1(a). A solution used, e.g. by Ganesan et al. [6] in this context is the enrichment of the pressure approximation space with discontinuous functions, handling the pressure jump. This approach is also known from the field of interface-capturing methods, see e.g. [13, 14]. However, in many cases it is known a priori from physical considerations where the pressure jump occurs. In particular, in case of two-phase flows, we can conclude that a pressure jump will only occur across the fluid–fluid interface. Within each subdomain occupied by a single phase, the continuous pressure interpolation space suffices for
representing the solution. Therefore, the nodes on the interface can simply be equipped with an extra degree of freedom, so that the pressure solutions for the domains $\Omega_1$ and $\Omega_2$ can be represented separately (see Figure 1(b)). From the implementational point of view, this is achieved by doubling the nodes at the interface. The velocity degrees of freedom at the doubled nodes are then coupled by applying periodic boundary conditions.

6. NUMERICAL EXAMPLES

6.1. Stationary bubble test case

In the first test case, a two-dimensional, steady Stokes flow is considered with a circular interface. The computational domain is depicted in Figure 2.

No-slip boundary conditions are prescribed at the domain boundary and no gravitational forces are considered. For this case, the analytical solution of the pressure jump across the interface is given by the Laplace equation [29]:

$$\Delta p = \gamma \kappa,$$

(29)

where the surface tension coefficient $\gamma$ is a chosen constant and the curvature $\kappa$ of the circular interface is known by construction. Here, the radius is 1 m and therefore the curvature $\kappa = 1/r = 1 \text{ m}^{-1}$. The analytical velocity field is $u = 0 \text{ m/s}$ and thereby no interface movement occurs. In order to account for the pressure jump, the approach described in the preceding section is applied in this test case.

Arbitrary physical parameters, shown in Table I, are chosen for the simulation. Hence, a pressure jump of $10 \text{ N/m}^2$ is to be expected (cf. Equation (29)). The calculation is performed on a triangular mesh consisting of 2840 nodes and 2654 elements (Figure 3). The interface is then described by 38 nodes.
Table I. Stationary bubble: Physical parameters.

<table>
<thead>
<tr>
<th>Phase</th>
<th>( \rho ) (kg/m²)</th>
<th>( \nu ) (m²/s)</th>
<th>( \gamma ) (N/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000</td>
<td>0.903</td>
<td>102</td>
</tr>
<tr>
<td>2</td>
<td>0.783</td>
<td>0.681</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 3. Stationary bubble: Mesh.

We compare the numerical results using our NURBS approximation approach with the results obtained by using the approach proposed in [5], where the Laplace–Beltrami operator is used to reformulate the surface tension term:

\[
\int_{(P_n)_{int}} \kappa w^h \cdot n \, dP = \int_{(P_n)_{int}} \Delta \mathbf{d}_{P_{int}} \cdot \mathbf{w}^h \, dP = \int_{(P_n)_{int}} \nabla \mathbf{d}_{P_{int}} : \nabla \mathbf{w}^h \, dP
\]

(30)

with

\[
\nabla f := \nabla f - (n \cdot \nabla f)n.
\]

(31)

In this case, the curvature of the interface is not required explicitly.

Given a NURBS with 20 control points and a degree of 4, the fitting algorithm requires only 1 iteration to reach the specified residual criteria of \( 5 \times 10^{-7} \). The calculated pressure jump of 10.00160N/m² is in good agreement with the analytical solution of 10N/m² and much better compared with the solution with the Laplace–Beltrami approach on the same mesh: 10.0346N/m².

We carried out a mesh refinement study for the Laplace–Beltrami implementation (uniform refinement). A mesh with more than 5 times the number of interface nodes (123,900 total nodes) is required to obtain a comparable pressure accuracy using the Laplace–Beltrami approach: 10.00130N/m².

As expected, the proposed approach provides accurate representation of the interface curvature and normal vectors, already on coarse meshes.

6.2. Pulsating drop

The second test case is taken from [4]. In contrast to the previous test case, the interface moves at every time step. Furthermore, only one phase is modeled while the surrounding phase is assumed to have negligible density and viscosity and is thereby neglected. A drop with an initial elliptic shape is considered. The initial horizontal and vertical axes are 1.25 and 0.80m (cf. Figure 4) and a triangular mesh with 1819 nodes and 3518 elements is used (Figure 5).

The density of the fluid is given as \( \rho = 1 \) kg/m³, the dynamic viscosity \( \nu = 0.001 \) m²/s and the surface tension coefficient \( \gamma = 0.001 \) N/m. Gravity is neglected again. The time step size for this unsteady simulation is chosen to be 0.1 s. Since mechanical systems try to find a state of minimal potential energy, the drop tends to approach the shape with the smallest circumference, i.e. the circle (cf. [29]). Hence, uniform, flattening oscillations in the horizontal and vertical axis length

Copyright © 2011 John Wiley & Sons, Ltd.

are expected. The NURBS approximation uses a degree 4 curve with 50 control points. In Figure 6 the plot of the principal axis of the drop over the simulation time of 480s is shown. The result agrees with the result in [4]. Figure 7 shows the mesh, pressure contour and velocity vectors at the 4 marked time values in Figure 6.

The NURBS approximation method again requires only one iteration in all the time steps. The velocity vector plots nicely visualize the pulsating movement of the drop, which leads to circulations inside the drop. As the amplitude of the oscillations decrease, the pressure differences inside the drop decrease as well. This test case shows that the accuracy of the proposed method also holds for moving interfaces.

6.3. Swelling

A more realistic free-surface flow example is a fluid that exits a shape-giving channel such as a nozzle or a die. In this case, the parabolic velocity profile, due to wall adhesion, transitions to a plug profile and undergoes swelling. For a Newtonian fluid, the swell ratio is at most 13%. For viscoelastic fluids this ratio can be more than 100% [30], resulting in a very steep rise of the free surface. Here, the example of a cylindrical nozzle is used. As this case is symmetric, only one quarter of the nozzle is represented by the finite element mesh; refer to Figure 8 for the exact
dimensions. The fluid is considered to be governed by a Newtonian constitutive model with a density of 1.0 g/mm$^3$ and a viscosity of 8 Pas.

The governing equations for this problem are the steady Stokes equations (6), as well as the mesh kinematic condition (16). If these equations are solved in a coupled manner, the problem is steady and the velocity, pressure and position of the free surface can be solved for simultaneously. However, one can also iterate between the flow equation and the mesh deformation equation, until a steady state is reached, which is the approach used for this work.
In any case, the computation of the normal vectors at the nodes of the free surface which are needed for the mesh deformation poses problems for unstructured 3D meshes. Averaging the normals of the adjacent element faces is computationally cheap and easy to implement, but suffers from low accuracy. With the usual grid generation techniques, the mesh is likely to suffer from defects as depicted in Figure 9. These defects complicate the computation of the normal vectors for the neighboring nodes. Owing to this corrupt normal computation, the defects are likely to become even more pronounced during the simulation, leading to oscillatory free surfaces (cf. Figure 10). Furthermore, at boundary edges, the normals cannot be represented correctly, as the slope of the actual free surface is not represented by the adjacent element faces.

The NURBS representation of the free surface results in vast improvement. The procedure is as follows: a NURBS surface, which has the exact dimensions of the initial free surface, is created. To establish a connection from the NURBS to the finite element mesh which is generated in a separate step, the local NURBS coordinates of each node of the finite element mesh need to be found. This is accomplished using an optimization procedure as described in Section 4.2. From the appropriate NURBS representing the free surface, the normals can be computed very accurately. Subsequently, the nodes on the free surface are moved as described in Section 3.2. After the nodes have been moved, the NURBS surface is fitted to the new position of the nodes. Since the deformation is very moderate in this case, a fitting of only the control point positions can be considered to be sufficient. This simplifies the fitting procedure, which can now be based on a damped steepest decent algorithm as described for example in [31]. The previously determined local coordinates of the nodes of the finite element mesh are considered fixed. This is an assumption which is only valid if the deformation of the free surface is sufficiently small. Even after the fitting, the NURBS surface will in general not touch every node of the finite element mesh. Therefore, in a final step all nodes are moved back into their corresponding point on the NURBS surface in order to keep the free surface smooth.

Regarding the standard approach to normal computation, we used three different refinement levels (average mesh size $h$ of 0.5, 0.25 and 0.125) in order to generate the reference results for the NURBS-based simulations, using the computed swell ratio at steady state as reference. The values for the different meshes are given in Table II. All NURBS-based simulations were performed on the coarsest grid with $h = 0.5$. Different numbers of control points were used, namely, 10 in the flow direction and 4 along the quarter circle (referenced to as $10 \times 4$), 6 in the flow direction and 6 along the quarter circle ($6 \times 6$), as well as 6 in the flow direction and 4 along the quarter circle ($6 \times 4$). All NURBS used have a degree $K = 3$. 
Table II. Swell ratios for the simulations performed.

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Swell ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>(h = 0.5)</td>
<td>1.2104</td>
</tr>
<tr>
<td>(h = 0.25)</td>
<td>1.1701</td>
</tr>
<tr>
<td>(h = 0.125)</td>
<td>1.0871</td>
</tr>
<tr>
<td>(10 \times 4)</td>
<td>—</td>
</tr>
<tr>
<td>(6 \times 6)</td>
<td>1.1749</td>
</tr>
<tr>
<td>(6 \times 4)</td>
<td>1.1627</td>
</tr>
</tbody>
</table>

Figure 11. Swelling: Free surface shape obtained using a \(6 \times 6\) NURBS.

For the \(10 \times 4\), no steady solution could be obtained, which is why no swell ratio could be computed. The control points were so close in this case, that the solution became oscillatory. The other results suggest, that the accuracy of the result using the NURBS interpolation is comparable to the solution on a mesh with twice the resolution if the normal vectors are computed from the finite element mesh. Figure 11 illustrates the smoothness of the free surface if a NURBS approximation is used as compared with Figure 10. The underlying finite element mesh is the same in both figures.

7. CONCLUSION

Showing several conceptually different free-surface flow test cases, we have illustrated that the use of NURBS curves and surfaces for the representation of the free surface can increase the accuracy of the computation. Depending on the complexity of the fitting task, we have introduced a possible approach to obtain a connection between the nodes on the finite element mesh and the local coordinates on the NURBS. In all of our test cases, the NURBS representation has increased the quality of the solution on a mesh of a particular resolution. However, the easiest to implement and computationally fastest solution is still to use the finite element mesh for the computation of information which is needed in addition to the actual degrees of freedom (in our case, curvature and normal vectors). Nonetheless, if the resolution needed to compute this additional information with sufficient accuracy is much higher than the one needed for the actual degrees of freedom, a more advanced solution might be worth the computational effort.

In this case, NURBS curves and surfaces can be recommended as a very complete and flexible solution. NURBS have the capability to represent general curves and surfaces accurately and in one single and global representation. Special attention needs to be paid to how the nodes on the finite element mesh can be related to the NURBS. Depending on the exact scenario, i.e. whether both the position of the control points and the local coordinates need to be fitted or only the control points or only the local coordinates, different algorithms can be recommended based on our experience.

In addition, a very accessible method for handling pressure jumps in connection with interface-tracking schemes has been introduced. The interface nodes are equipped with two pressure degrees of freedom in order to represent the jump accurately even if only continuous interpolation functions are available.

Acknowledgements

This work was supported by the German Research Foundation under CoE ‘Integrative production technology for high-wage countries’ program and the Emmy-Noether-research group ‘Numerical methods for discontinuities in continuum mechanics’. Computing resources were provided by the AICES graduate school.

References