# Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes

by

Tormod Bjøntegaard, Einar M. Rønquist

PREPRINT
NUMERICS NO. 3/2008

# Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes

Tormod Bjøntegaard, Einar M. Rønquist

March 31, 2008

We present a new method for tracking an interface immersed in a given velocity field. The method is particularly relevant to the simulation of unsteady free surface problems using the arbitrary Lagrangian-Eulerian (ALE) framework. The new method has been constructed with two goals in mind: (i) to be able to accurately follow the interface; and (ii) to maintain a good point distribution for the grid points along the interface. The method combines information from a pure Lagrangian approach with information from an ALE approach. No integration backwards in time is needed; instead, the new method relies of the solution of several pure convection problems along the interface in order to obtain the relevant information. The new method offers flexibility in terms of how an "optimal" point distribution should be defined. We have been able to verify first, second, and third order temporal accuracy for the new method by solving two-dimensional model problems with known solutions.

## 1 Introduction

The ability to accurately follow time-dependent surfaces is very important in many areas of computational science and engineering. An important class of such problems is free surface flows, with the free surface representing the interface between two fluids, e.g., air and water. Computational methods for solving such problems can typically be classified into two categories: methods which explicitly track the free surface (interface-tracking methods; e.g., [16]) and methods where the interface is more implicitly defined (e.g., level set methods [15, 17, 14] or volume-of-fluid methods [6]); we will here focus on the former class.

Interface-tracking methods (or sometimes also referred to as front-tracking methods) comprise a few essential steps. At any particular point in time, a velocity field is typically determined from the governing equations within the fluid(s), e.g., by solving the Navier-Stokes equations. By integrating this velocity field, it is possible to obtain a new position of the interface.

A pure Lagrangian approach applied to an evolving interface is simply based on integrating the velocity of the fluid particles along the surface to obtain the position of the surface at a later point in time. However, in the context of a numerical approximation (e.g., using finite-element-based methods), a pure Lagrangian approach is often not a very practical approach since it typically results in large deformations of the computational domain.

In the context of free surface flows, the arbitrary Lagrangian-Eulerian (ALE) formulation of the governing equations has been very successful as a point of departure for a numerical approximation [7, 3, 10]. A typical approach to updating the free surface is to enforce a kinematic condition along the surface. This condition has its origin in a continuum description, and says that the normal fluid velocity has to be equal to the normal domain velocity at any point along the surface. An important consequence of this condition is the fact that a fluid particle which is present somewhere along the free surface at a particular time will also be present at the free surface at a later time.

While the kinematic condition enforces a normal condition, a tangential domain velocity also needs to be specified along the surface; a common choice is to enforce a homogeneous Dirichlet condition for the tangential component [18, 1]. This choice typically reduces the deformation of the computational domain compared to a pure Lagrangian approach, however, it offers limited control over the quality of the grid used to represent the free surface. In particular, the *distribution* of the grid points along the free surface may deteriorate over time, which may ultimately result in severe loss of accuracy (or even breakdown of the simulation). This latter issue may be dealt with in various ways, e.g., through remeshing or other mesh update strategies [11, 4]. However, the temporal accuracy will typically suffer using such a strategy.

The issue of a non-optimal evolution of the surface representation is particularly acute in the context of using high order finite elements or spectral elements. The reason for this is related to the fact that such methods depend on locally regular mappings between a reference domain and the corresponding physical element. If the distribution of the surface points along the free surface becomes very distorted, this mapping may not be so regular anymore, resulting in a loss of spatial accuracy. This will again affect the calculation of tangent and normal vectors, as well as the local curvature, since the computation of these quantities depends on the coupling between many surface points [9, 21].

One could also imagine enforcing the kinematic condition together with a tangential component of the domain velocity in such a way that the integration of the total domain velocity would: (i) result in an accurate representation of the free surface; and (ii) maintain a good distribution of the grid points along the surface [4, 2]. An obvious challenge with this approach is how to define the overall domain velocity in such a way that not only good spatial accuracy is achieved (with no need for remeshing), but in a way that will also ensure good temporal accuracy (better than first order). The goal of this paper is to propose a way to achieve these two objectives at the same time.

The paper is organized as follows. We first discuss some key aspects associated with the two-dimensional problems we will focus on, including the notation we will use. We will only discuss the evolution of a surface when it is "immersed" in a known two-dimensional velocity field; no partial differential equation will be solved to obtain this velocity field. We will let the surface evolve in time, and different computational strategies for predicting the surface evolution will be tested and compared. In particular, we will compare two well-known methods with the new approach proposed in this work. Numerical tests will illustrate the similarities and differences between the methods, and conclusions will be made based on these. This study is part of an ongoing research project on solving partial differential equations in time-dependent geometries.

## 2 Two-dimensional interface-tracking

Consider first the front depicted in Figure 1. Assume that we know the front at time $t^n$. Assume also that a numerical approximation of the front is used, something which requires a surface parameterization. A typical way to achieve this is to use piecewise polynomial approximations (e.g., finite-element-based methods), which typically introduces grid points.

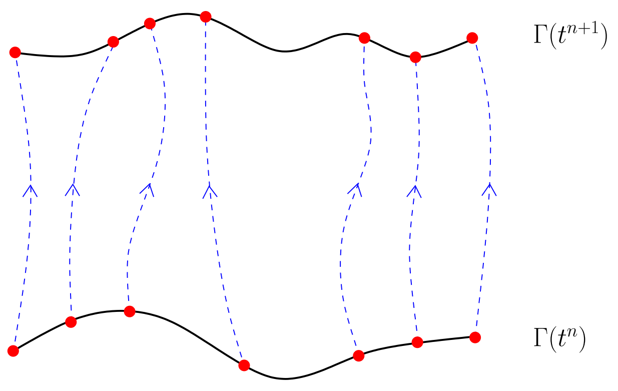


Figure 1: A front at time $t^n$ with an “optimal” point distribution. For example, the points can be the nodes along an edge of a deformed spectral element. The front is "immersed" in a velocity field and the particles follow the path of the dashed lines.

Assume now that we have an interface with “optimally” distributed grid points at time level $t^n$. At each point $\mathbf{x}$ along the surface there is an associated velocity field $\mathbf{u}$. This velocity field can be explicitly known (as in our study here), or it can be given as the solution of an underlying partial differential equation (e.g., the solution of the Navier-Stokes equations in a free surface problem).

We assume that the velocity at a point along the surface represents the velocity of the corresponding "fluid particle". If we integrate the velocity of all the fluid particles along the surface, we obtain the position of the surface at a later time. This is what a pure Lagrangian description will give us; the motion of a particle is simply governed by the equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}(\mathbf{x}, t). \tag{1}$$

In a computational setting, we can limit the integration of (1) to the grid points, and then use the underlying surface parameterization to represent the entire surface at a later time; see Figure 1. A severe problem with this approach is obvious: we have no control over the distribution of the grid points at a later time $t^{n+1}$. This will again result in a loss of accuracy in the calculation of surface quantities, e.g., tangent and normal vectors, as well as the local curvature.

A great advantage with the ALE formulation is that it introduces a separate domain velocity $\mathbf{w}$ (also referred to as the grid velocity in the context of the discrete problem), which limits the deformation of the computational domain. In an ALE-framework, the position of the interface is advanced according to

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{w}(\mathbf{x}, t) \tag{2}$$

instead of the pure Lagrangian approach (1).

A continuum description dictates that $\mathbf{w} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}$ (the kinematic condition), where $\mathbf{n}$ is the unit normal along the interface. However, no particular condition is required for the tangential component $\mathbf{w} \cdot \mathbf{t}$ of the domain velocity ($\mathbf{t}$ is unit tangent vector). A common choice is to set $\mathbf{w} \cdot \mathbf{t} = 0$ along the surface, although this is often not an optimal choice; see Figure 2 and Figure 3.

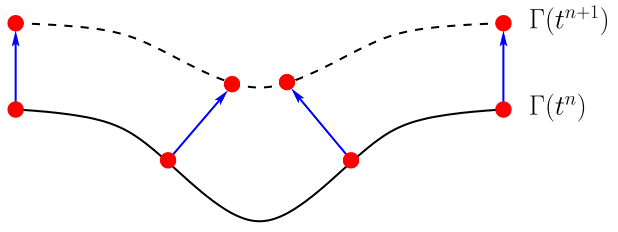


Figure 2: A front at time $t^n$ with an "optimal" point distribution. This interface is advanced by honoring the kinematic condition, while imposing a zero tangential grid velocity. The resulting point distribution at a later time $t^{n+1}$ is obviously no longer optimal.

Let us also comment on the issue of temporal accuracy. Integration of (1) and (2) can be done using an explicit method; often an explicit multi-step method (e.g., Adams-Bashforth) is used [8, 1]. The choice of an explicit method is often motivated by the wish to compute the velocity field separately from the treatment of the geometry. If the velocity fields ($\mathbf{u}$ and $\mathbf{w}$) are sufficiently regular, we expect to achieve higher order temporal accuracy (second and third) in terms of the *location* of individual points along the front. As mentioned above, this approach may yield limited control over the *distribution* of the points along the front. If the point distribution is non-optimal, the resulting loss of spatial accuracy *will*Ê affect the accuracy of surface quantities such as normal and tangent vectors, local curvature, and length/area, and this again may affect the accuracy of the interface tracking.

In order to construct a computational approach which will yield both high order temporal accuracy (e.g., second or third order), as well as good spatial accuracy in the calculation of surface quantities, we need to solve the problem of automatically obtaining a good point distribution in a satisfactory way. This problem is particularly acute in the context of using high order methods. Despite the importance of this issue, very limited discussion or results appear to be available in the literature.

We also mention a complicating factor in the development and assessment of various approaches for interface tracking: the lack of analytic solutions. In Section 4, we propose a few examples of test problems which we will use for verification purposes.

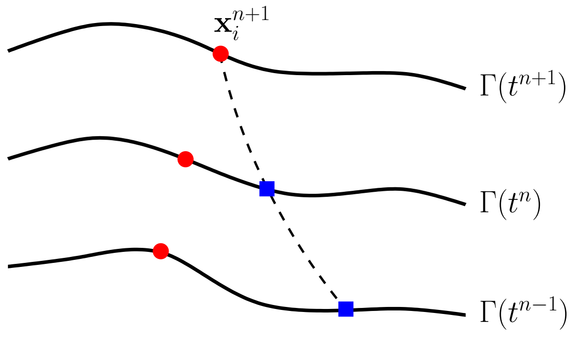


Figure 3: The plot depicts the position of a single point along the front at three different time levels (red circles). The point moves according to (2), with $\mathbf{w}\cdot\mathbf{n} = \mathbf{u}\cdot\mathbf{n}$ and $\mathbf{w}\cdot\mathbf{t} = 0$. The position at time level $t^{n+1}$ is $\mathbf{x}_i^{n+1}$. Note that the corresponding positions at time levels $t^n$ and $t^{n-1}$ do *not* correspond to the same fluid particle; the path of the fluid particle (e.g., a particle which moves according to (1)) ending up at position $\mathbf{x}_i^{n+1}$ at time $t^{n+1}$ follows the dashed line.

## 3 The new approach

We will now propose a method which gives an accurate representation of the interface (i.e., the individual grid points end up on the correct surface), as well as the flexibility of specifying a user defined point distribution. This method will be "automatic" in the sense of fulfilling both goals in an "integrated" fashion, i.e., with no need for remeshing etc.

Our strategy is based on a combination of a pure Lagrangian approach and an ALE approach, in which both (1) and (2) are integrated in order to find the new grid points. One major feature of the new algorithm is that we search for the fluid particle at time $t^n$ which ends up along some prescribed direction at time $t^{n+1}$. Thus, this will invole an iteration process for each grid point.

In order to make the algorithm concrete, we will limit our discussion to the case where the entire front $\Gamma(t)$ corresponds to an edge in a single spectral element. However, we remark that the ideas behind the proposed method is equally applicable to the case where the front is composed of a number of low order finite elements.

Following a standard spectral element discretization [12], the front $\Gamma(t)$ is parameterized as follows: for a given $\xi \in \hat{\Gamma} = [-1, 1]$, the corresponding point $\mathbf{x}$ on $\Gamma(t)$ is given as $\mathbf{x} = (x_1, x_2) = (x_1(\xi), x_2(\xi))$. In general, any field variable $\varphi$ associated with the front can be represented in terms of the reference variable $\xi$. In particular, an $N$th order polynomial approximation $\varphi_N$ of $\varphi$ at time $t^n$ can be expressed in terms of the following nodal basis:

$$\varphi_N^n(\xi) = \sum_{j=0}^{N} \varphi_j^n \ell_j(\xi). \tag{3}$$

Here, $\varphi_j^n$ represents an approximation of $\varphi(\xi_j, t^n)$, $\xi_j$ is the $j$-th Gauss-Lobatto Legendre (GLL) point, and $\ell_j(\xi)$ is the $N$th order Lagrangian interpolant through the GLL points; as usual, $\ell_j(\xi_i) = \delta_{ij}$.

Without presenting all the details at once, we first discuss the key ingredients in a second order temporal scheme. We will later return to discuss all the details, as well as the extension to a third order temporal scheme.

We assume that the following surface variables are known:

$$\underline{x}_1^n, \underline{x}_2^n, \underline{u}_1^n, \underline{u}_2^n, \underline{u}_1^{n-1}, \underline{u}_2^{n-1}.$$

Here, $(x_i^n)_j$ is the $i$'th coordinate of the $j$'th point at time $t^n$ and $(u_i^n)_j$ is the $i$'th velocity component of the $j$'th point at time $t^n$. We use underscore to denote a vector comprising *all* the nodal values associated with a field variable, i.e., $j = 0, \ldots, N$; see (3).

In Algorithm 1 we present the first version of the new algorithm. A few comments are required at this stage. First, *convergence* in this setting is related to the point *distribution.* The position computed in Step 4 represents the integration of (1) using a second order Adams-Bashforth scheme, however, we don't know if the computed grid point on $\Gamma(t^{n+1})$ (grid point $j$) is in a good position relative to its neighbors. Thus, we obtain convergence when we have chosen the "correct" particle, which amounts to choosing the "correct" $\xi$ at time $t^n$ (which we denote as $\xi^n$). However, we need to quantify this and we will return to this issue shortly.

Second, Step 2 in Algorithm 1 involves finding $\xi^{n-1}$, which is the reference coordinate at time $t^{n-1}$ for the particle which at time $t^n$ has the reference coordinate $\xi^n$; see also Figure 3. Hence, the computation of $\xi^{n-1}$ appears to involve integration backwards in time. However, what we actually need is the *velocity* the particle in question had at time $t^{n-1}$, and not necessarily the position it had at time $t^{n-1}$. In Section 3.1 we will propose a way to compute $\hat{u}_i^2$, $i = 1, 2$, which does not involve integration backwards in time.

Third, when we have converged to the correct position $(x_i^{n+1})_j$ of a grid point $j$ on $\Gamma(t^{n+1})$, see Step 4, we also compute the corresponding grid velocity $(w_i^n)_j$ such that an integration of (2) using a second order Adams-Bashforth method will result in the same position; this is done in Step 6. Here, $(x_i^n)_j$, $(x_i^{n+1})_j$, and $(w_i^{n-1})_j$ are known, while $(w_i^n)_j$ is unknown. The reason for computing the grid velocity in this way is that: (i) it gives consistency with a pure Lagrangian approach; (ii) we are indirectly satisfying the kinematic condition; and (iii) we are indirectly and "automatically" able to specify a tangential grid velocity such that we obtain a good point distribution.

**Algorithm 1** First version of a second order temporal scheme

```
for $j = 0$ to $N$ do
  1. Guess $\xi^n$.
  repeat
    2. Find $\xi^{n-1}(\xi^n)$.
    3. Compute
         $\hat{x}_i^1 = x_i^n(\xi^n) \qquad i = 1, 2,$
         $\hat{u}_i^1 = u_i^n(\xi^n) \qquad i = 1, 2,$
         $\hat{u}_i^2 = u_i^{n-1}(\xi^{n-1}) \quad i = 1, 2.$
    4. Compute $(x_i^{n+1})_j = \hat{x}_i^1 + \Delta t \left(\frac{3}{2}\hat{u}_i^1 - \frac{1}{2}\hat{u}_i^2\right), \quad i = 1, 2.$
    5. Update $\xi^n$.
  until convergence

  6. Compute $(w_i^n)_j$ from $(x_i^{n+1})_j = (x_i^n)_j + \Delta t \left(\frac{3}{2}w_i^n - \frac{1}{2}w_i^{n-1}\right)_j, \quad i = 1, 2.$
end for
```

Let us now discuss more precisely what we mean by convergence in Algorithm 1. Essentially, convergence is related to quantifying a good point distribution. We will consider two alternative strategies. The first strategy is illustrated in Figure 4. Starting from the front at time $t^n$, we move the end points to time level $t^{n+1}$. How these points are moved will be problem dependent, but it may for instance be a pure Lagrangian motion where the two end points follow the path of the fluid particles from $t^n$ to $t^{n+1}$ (see (1)), or the end points may move according to a standard ALE-formulation (see (2)). When this is done, we construct the chord between the two end points, and distribute grid points along this chord according to the desired distribution (e.g., a GLL distribution). Next, the normal from the chord is constructed at each of these points. Thus, using this strategy we search for the particle at time $t^n$ which, when advanced through a pure Lagrangian motion, is located along this normal up to a given tolerance. The advantage of this approach is that we are in full control of the inner grid-points at each time-step. The disadvantage is that it may not be so easy to extend the approach to three dimensions.

The second strategy is illustrated in Figure 5. Again, the end points are first advanced to $t^{n+1}$. Next, for each end point, a vector which connects the end point at $t^n$ to the end point at $t^{n+1}$ is constructed. Then, directional vectors for the interior nodes are constructed through a linear interpolation on the reference domain, $\hat{\Gamma}$, of the end point vectors. Finally, the requirement is to find the particle at time $t^n$ which, when advanced to $t^{n+1}$, is located along this interpolated vector starting at the grid point at time $t^n$. This strategy has the advantage that it is more easily extended to three dimensions. However,

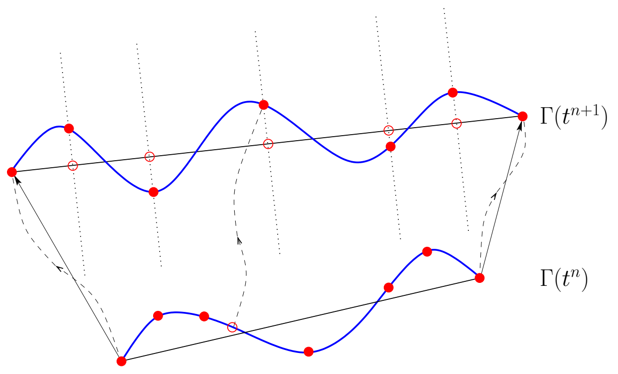


Figure 4: Strategy 1.

we have less explicit control over the interior grid points at each time level.

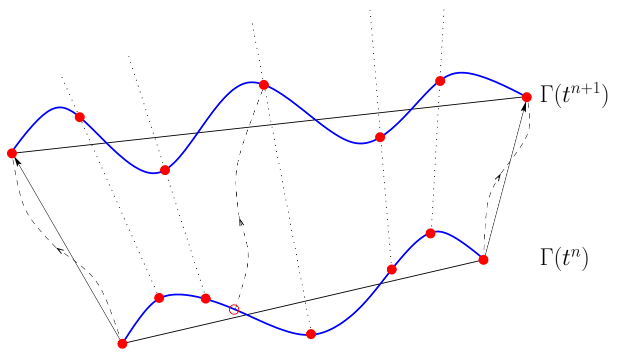


Figure 5: Strategy 2.

There will probably be other strategies for addressing the issue of grid distribution as well, however, this will only change the convergence condition. The rest of the algorithm which will be presented in the subsequent sections will remain the same.

## 3.1 Finding a fluid particle's earlier velocities

The second order scheme presented in Algorithm 1 requires information about the velocities at time levels $t^n$ and $t^{n-1}$ for those fluid particles along the interface which end up at the grid points $\mathbf{x}_j^{n+1}$, $j = 0, \ldots, N$, along $\Gamma(t^{n+1})$. We would like to avoid tracking the characteristics backwards in time since this requires interpolation and can be expensive for high order methods [19, 5]. In addition, we need to deal with the fact that the interface $\Gamma$ changes shape as a function of time.

Since we assume that the same particles remain on the surface at all time, it is sufficient to solve a one-dimensional convection problem *forward* in time. In order to explain the procedure, consider the following pure time-dependent convection problem along the interface $\Gamma(\tau)$: Find $\varphi(s, \tau)$ such that

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + u_s \frac{\partial \varphi}{\partial s} &= 0, && \text{on } \Gamma(\tau), \\ \varphi(s, \tau = 0) &= \varphi_0(s), && \text{on } \Gamma(\tau = 0). \end{aligned}$$

Here, $s$ is an arc-length variable and $u_s = \mathbf{u} \cdot \mathbf{t}$ is the tangential component of the fluid velocity; in all our model problems $u_s = 0$ on $\partial\Gamma(\tau)$. We can easily derive the ALE formulation of this problem: Find $\varphi(s;\tau) \in X$ such that

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\int_{\hat{\Gamma}} v\varphi J_s \,\mathrm{d}\xi + \int_{\hat{\Gamma}} v(u_s - w_s)\frac{\partial\varphi}{\partial\xi}\,\mathrm{d}\xi - \int_{\hat{\Gamma}} v\varphi\frac{\partial w_s}{\partial\xi}\,\mathrm{d}\xi = 0, \quad \forall v \in X, \qquad \varphi(\xi;\tau=0) = \varphi_0(\xi). \tag{4}$$

Here, $X$ is an appropriate function space, e.g., $X = H^{1/2}(\Gamma)$ if all the field variables correspond to the trace of $H^1$-functions in the adjacent fluid domains. Furthermore, $J_s = \left(\left(\frac{\partial x_1}{\partial\xi}\right)^2 + \left(\frac{\partial x_2}{\partial\xi}\right)^2\right)^{1/2}$ is the surface Jacobian, and $w_s = \mathbf{w}\cdot\mathbf{t}$ is the tangential component of the domain velocity; note that both $u_s$ and $w_s$ are zero on $\partial\Gamma$ for all the model problems in our study. We also remark that we have used the same symbol $\varphi$ for a field variable expressed both in the arc-length variable and in the reference coordinate.

We discretize the convection problem (4) using a standard spectral method in space based on high order polynomials, and arrive at the following set of ordinary differential equations:

$$\frac{\mathrm{d}(\underline{B}^s\,\underline{\varphi})}{\mathrm{d}\tau} = -\underline{C}^s(\underline{\mathbf{u}},\underline{\mathbf{w}})\,\underline{\varphi}, \qquad \underline{\varphi}(\tau=0) = \underline{\varphi}_0. \tag{5}$$

Here, $\underline{B}^s$ is the surface mass matrix, $\underline{C}^s$ is the discrete convection operator along the surface (including the "surface divergence" of the domain velocity); note that both $\underline{B}^s$ and $\underline{C}^s$ are time-dependent.

If we integrate (5) from 0 to $\Delta t$ with $\underline{\varphi}_0 = \underline{u}_i^{n-1}$, $i = 1,2$, we observe that $\underline{\varphi}(\tau = \Delta t)$ will be an approximation to the $i$'th velocity component at time $t^{n-1}$ of the fluid particles which at time $t^n$ are located at the grid points along $\Gamma(t^n)$. This approach is inspired by the ideas presented in [13] in the context of constructing convection-Stokes splitting schemes.

Note that $\hat{u}_i^2$ in Algorithm 1 generally represents the velocity of a fluid particle at time $t^{n-1}$ which does *not* coincide with a grid point along $\Gamma(t^n)$. However, by computing the velocities at time $t^{n-1}$ of the fluid particles which coincide with the grid points of $\Gamma(t^n)$ (i.e., by solving (5)), we can use the polynomial expansion (3) to find the velocity at *any* value of the parameter $\xi$. Moreover, *one* particular value of $\xi$ will now give us information about the velocity of a fluid particle at two different time levels ($t^n$ and $t^{n-1}$). This is also exactly the type of information we need in our algorithm; in fact, access to this information avoids entirely the need to find $\xi^{n-1}(\xi^n)$ in Step 2 of Algorithm 1 and thus represents a significant simplification.

The approach is readily extended to velocities at earlier time levels as well. For instance, if we choose $\underline{\varphi}_0 = \underline{u}_i^{n-2}$ in (5), $\underline{\varphi}(\tau = 2\Delta t)$ will be an approximation to the $i$'th velocity component at time $t^{n-2}$ of the fluid particles which at time $t^n$ are located at the grid points along $\Gamma(t^n)$.

For the integration of (5) we have chosen the classical explicit fourth order Runge-Kutta scheme. Similar to the convection subproblem treated in [13], the convection velocities $\underline{\mathbf{u}}$ and $\underline{\mathbf{w}}$, as well as the surface geometry $\underline{\mathbf{x}}$, are each approximated as a polynomial in time of one order lower than the Adams-Bashforth scheme used in Step 4 and Step 6 of Algorithm 1. Thus, for a second order temporal scheme, a first order polynomial interpolation/extrapolation in time is used for these quantities when solving (5), while the extension to a third order scheme will use a second order polynomial approximation in time for $\underline{\mathbf{u}}$, $\underline{\mathbf{w}}$ and $\underline{\mathbf{x}}$.

To summarize this section: in Algorithm 1 we are interested in the velocities that particular fluid particles had at time level $t^n$ and $t^{n-1}$. We choose $\underline{\varphi}_0 = \underline{u}_i^{n-1}$ in (5) and set $\underline{\tilde{u}}_i^n = \underline{\varphi}(\tau = \Delta t)$, $i = 1, 2$. We now have six sets of nodal values, $\underline{x}_i^n$, $\underline{u}_i^n$, and $\underline{\tilde{u}}_i^n$, $i = 1, 2$, all associated with the interface $\Gamma(t^n)$. By using the polynomial expansion (3), we have thus six polynomial approximations, and all of these are defined along $\Gamma(t^n)$. Hence, one particular value of $\xi$ corresponds to the *same* particle. Thus, there is no longer need to compute $\xi^{n-1}(\xi^n)$ in Step 2 of Algorithm 1. We may now only focus on finding the appropriate $\xi^n$ which will allow us to achieve convergence in Step 4.

## 3.2 Finding the "correct" particle

Assume that we are searching for the position of a fluid particle at time level $t^n$, as well as its velocities at time levels $t^n$ and $t^{n-1}$, such that we can compute the position of the fluid particle at time level $t^{n+1}$ according to Step 4 in Algorithm 1. As explained in the previous section, the problem can be reduced to finding one particular value of $\xi \in \hat{\Gamma}$.

Assume that we are currently interested in updating the information about the grid point in the middle of the reference domain $\hat{\Gamma}$ (i.e., this point is associated with the index $j$ in Algorithm 1), and that we are searching for information about a particle located somewhere in $\hat{\Gamma}$. Finding this particle is certainly possible, but will be somewhat cumbersome if the interface $\Gamma$ comprises several elements (either low order finite elements or high order spectral elements). The reason for this is that we do not *a priori* then know which element the particle belongs to at the different time levels, and some kind of search algorithm needs to be used.

Instead of finding the coordinate of this fluid particle (or the equivalent value of $\xi$), we propose to find a corresponding artificial time $\tilde{\tau}$ which convects this particle to our grid point (index $j$) using an artificial convecting velocity $U$; see Figure 6. To this end, we consider the one-dimensional convection problem

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + U \frac{\partial \varphi}{\partial \xi} &= 0, \qquad \text{on } \hat{\Gamma}, \\ \varphi(\tau = 0) &= \varphi_0. \end{aligned} \tag{6}$$

We assume that $U = 0$ on $\partial\hat{\Gamma}$ (a similar condition was assumed for $u_s$ in the previous section), and we will thus not specify any particular boundary condition for $\varphi$. We discretize (6) using a spectral method based on high order polynomials and arrive at the following set of ordinary differential equations

$$\begin{aligned} \underline{\hat{B}} \frac{\mathrm{d}\underline{\varphi}}{\mathrm{d}\tau} + \underline{\hat{C}}\, \underline{\varphi} &= 0, \\ \underline{\varphi}(\tau = 0) &= \underline{\varphi}_0. \end{aligned} \tag{7}$$

In (7) , $\underline{\hat{B}}$ and $\underline{\hat{C}}$ are the mass matrix and discrete convection operator, respectively; the hat indicates that both matrices are associated with $\hat{\Gamma}$. The idea is that, instead of finding a suitable $\xi$ in searching for the suitable particle, we search for a $\tilde{\tau}$ such that we obtain the information we need when we integrate (7) from $\tau = 0$ to $\tau = \tilde{\tau}$. Hence, instead of moving along the surface searching for a particle, we will sit at a fixed point and convect the pertinent information about the particle to us. Thus, the new procedure requires no backward time integration, and eliminates the need for a search and interpolation algorithm; see [20, 5] in the context of semi-Lagrangian schemes.

Note that the particular value of $\tilde{\tau}$ is actually of no interest to us; all we need is $\underline{\varphi}(\tilde{\tau})$. For this reason we only need to consider a fixed domain (in our case, $\hat{\Gamma}$). There is also

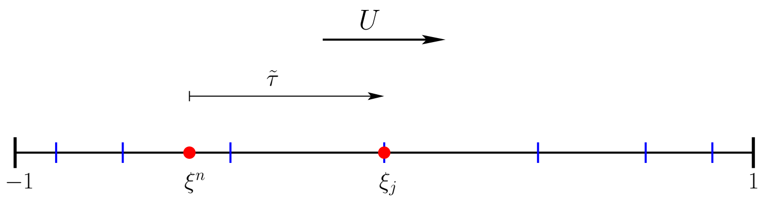


Figure 6: Six sets of nodal values, $\underline{x}_i^n$, $\underline{u}_i^n$, and $\underline{\tilde{u}}_i^n$, $i = 1, 2$, are associated with the reference domain $\hat{\Gamma}$. By integrating (7) from $\tau = 0$ to $\tau = \tilde{\tau}$, we can artificially convect the information associated with a particular $\xi \in \hat{\Gamma}$ (i.e., associated with a particular fluid particle) and check if Step 4 in Algorithm 1 will give us a "valid" new position for a grid point along $\Gamma(t^{n+1})$.

great flexibility in the choice of the convective velocity, $U$. This is because, for a specific particle somewhere along the surface, and for a reasonable choice of $U$, there will always be a corresponding $\tilde{\tau}$ (perhaps negative) which convects this particle to the current grid point. An easy and reasonable choice is $U(\xi) = 1 - \xi^2$ since this velocity is very smooth, it does not change sign, and it is also compatible with the condition $u_s = 0$ at $\partial\Gamma$ (which is the case for our numerical examples).

Note that $\underline{\varphi}$ in (7) is a vector with nodal values from the entire surface (corresponding to a single spectral element as in our study, or perhaps multiple finite elements). We need to carry the entire vector in the computation in order to evaluate the spatial derivative. On the other hand, we are only interested in the information convected to our current grid point. Thus, we need to solve one problem of the type (7) for each grid point.

## 3.3 Final version of a second order temporal scheme

The discussion from the preceding sections leads us to an improved algorithm for a second order temporal scheme; see Algorithm 2Ê below. Let us say a few words about how we achieve convergence in the new grid positions. In Step 1 we guess a $\tilde{\tau}$. For each grid point, we then solve (7) six times. From the resulting information, we update the new grid position as indicated in Step 3. We can check how this position compares with a "valid" position according to our chosen convergence strategy, see Figure 4 and Figure 5. A Newton-iteration can then be used to find the correct $\tilde{\tau}$, which again implies the correct $\xi$ (or choosing the correct fluid particle). In our case, we obtain derivative information for the Newton iteration from the last stage in the explicit Runge-Kutta scheme that we use to integrate (7). For the test problems considered in this study, convergence is achieved in 2 or 3 Newton iterations.

Finally, we remark that the loop $j = 0, \ldots, N$ over the grid points includes the two end points of the interface. These two end points may be treated differently compared to the inner points; this depends on the particular information which is available for the end points. For instance, for our numerical test problems, the end points are moved in a pure Lagrangian fashion.

## 3.4 Extension to a third order temporal scheme

Algorithm 3 represents a third order version of this method. We see that we now have to first solve two problems of the type (5) for each velocity component. During the integration of these problems, we need to use a second order polynomial approximation in time of the

**Algorithm 2** Final version of a second order temporal scheme

```
Solve (5) with $\underline{\varphi}_0 = \underline{u}_i^{n-1}$ and set $\underline{\tilde{u}}_i^n = \varphi(\tau = \Delta t)$ for $i = 1, 2$.
for $j = 0$ to $N$ do
  1. Guess $\tilde{\tau}$.
  repeat
    2. Integrate (7) from 0 to $\tilde{\tau}$ six times with $\underline{\varphi}_0 = \underline{x}_i^n$, $\underline{\varphi}_0 = \underline{u}_i^n$, $\underline{\varphi}_0 = \underline{\tilde{u}}_i^n$, for $i = 1, 2$.
    Define the results as $\underline{\hat{x}}_i$, $\underline{\hat{u}}_i^1$, $\underline{\hat{u}}_i^2$, respectively.
    3. Compute $(x_i^{n+1})_j = (\hat{x}_i)_j + \Delta t \left(\frac{3}{2}\hat{u}_i^1 - \frac{1}{2}\hat{u}_i^2\right)_j$ for $i = 1, 2$.
    4. Update $\tilde{\tau}$.
  until convergence
  5. Compute $(w_i^n)_j$ from $(x_i^{n+1})_j = (x_i^n)_j + \Delta t \left(\frac{3}{2}w_i^n - \frac{1}{2}w_i^{n-1}\right)_j$ for $i = 1, 2$.
end for
```

**Algorithm 3** Final version of a third order temporal scheme

```
Solve (5) with $\underline{\varphi}_0 = \underline{u}_i^{n-1}$ and set $\underline{\tilde{u}}_i^n = \varphi(\tau = \Delta t)$,   $i = 1, 2$.
Solve (5) with $\underline{\varphi}_0 = \underline{u}_i^{n-2}$ and set $\underline{\tilde{\tilde{u}}}_i^n = \varphi(\tau = 2\Delta t)$,   $i = 1, 2$.
for $j = 0$ to $N$ do
  1. Guess $\tilde{\tau}$.
  repeat
    2. Integrate (7) from 0 to $\tilde{\tau}$ eight times with $\underline{\varphi}_0 = \underline{x}_i^n$, $\underline{\varphi}_0 = \underline{u}_i^n$, $\underline{\varphi}_0 = \underline{\tilde{u}}_i^n$, $\underline{\varphi}_0 = \underline{\tilde{\tilde{u}}}_i^n$,
    $i = 1, 2$. Define the results as $\underline{\hat{x}}_i$, $\underline{\hat{u}}_i^1$, $\underline{\hat{u}}_i^2$, $\underline{\hat{u}}_i^3$, respectively.
    3. Compute $(x_i^{n+1})_j = (\hat{x}_i)_j + \Delta t \left(\frac{23}{12}\hat{u}_i^1 - \frac{4}{3}\hat{u}_i^2 + \frac{5}{12}\hat{u}_i^3\right)_j$,   $i = 1, 2$.
    4. Update $\tilde{\tau}$.
  until convergence
  5. Compute $(w_i^n)_j$ from $(x_i^{n+1})_j = (x_i^n)_j + \Delta t \left(\frac{23}{12}w_i^n - \frac{4}{3}w_i^{n-1} + \frac{5}{12}w_i^{n-2}\right)_j$, $i = 1, 2$.
end for
```

grid, the fluid velocity, and the grid velocity in order to maintain a third order temporal convergence rate. Obviously, the overall integration scheme for integrating (5) must also be at least of third order in time. As before, we use a fourth order explicit Runge-Kutta scheme, so this will not cause any problems. The solution of (7) will be the same as for the second order scheme. In Step 3 and Step 5, we use a third order Adams-Bashforth scheme for computing the new position and the new grid velocity of each grid point, respectively.

## 4 Numerical results

In this section we will perform numerical experiments in order to validate and compare the different algorithms for tracking the interface. For all the test problems we will define a two-dimensional, time-dependent velocity field, and at time $t = 0$ we will specify an initial interface. With the interface "immersed" in this two-dimensional velocity field, we will then monitor:

1. how accurately we are able to follow the exact interface; and

2. the quality of the corresponding point distribution.

The velocity fields will be prescribed in such a way that we are able to obtain analytic solutions for the exact interfaces at all times.

## 4.1 Error computation

The way we compute the error, $E_1$, in following the interface is illustrated in Figure 7. We first compute the chord between the end points of our numerical solution, and then compute the normal, $\mathbf{n}_c$, to this chord. For each grid point, $(\mathbf{x}^n)_j$, $j = 0, \dots, N$, along our numerically approximated interface, we find the intersection between the analytical front and the line originating from $(\mathbf{x}^n)_j$ and moving in the $\mathbf{n}_c$-direction. We call this intersection $(\mathbf{x}^e)_j$ and compute

$$e_j = \sqrt{((x_1^e)_j - (x_1^n)_j)^2 + ((x_2^e)_j - (x_2^n)_j)^2}.$$

Finally, we define the error $E_1$ as

$$E_1 = \frac{1}{N+1} \sum_{j=0}^{N} e_j. \tag{8}$$

The way we will measure the grid quality is by computing the error, $E_2$, in the length of the interface. For a high order approximation, this measure will generally give an indication of the quality of the point distribution (although there are special situations when this is not the case). Thus, we first compute the length, $S_n$, of our numerically computed interface. Using GLL quadrature we compute

$$S_n = \sum_{\alpha=0}^{N} \rho_\alpha (J_s^n)_\alpha = \sum_{\alpha=0}^{N} \rho_\alpha \left( \left( \frac{\partial x_1^n}{\partial \xi} \right)^2 + \left( \frac{\partial x_2^n}{\partial \xi} \right)^2 \right)_\alpha^{1/2}, \tag{9}$$

where $J_s^n$ is the surface Jacobian at time $T = t^n$, and $\rho_\alpha$, $\alpha = 0, \dots, N$ are the GLL quadrature weights. We then define the error

$$E_2 = |S_n - S_e|, \tag{10}$$

where $S_e$ is the length of our exact interface.

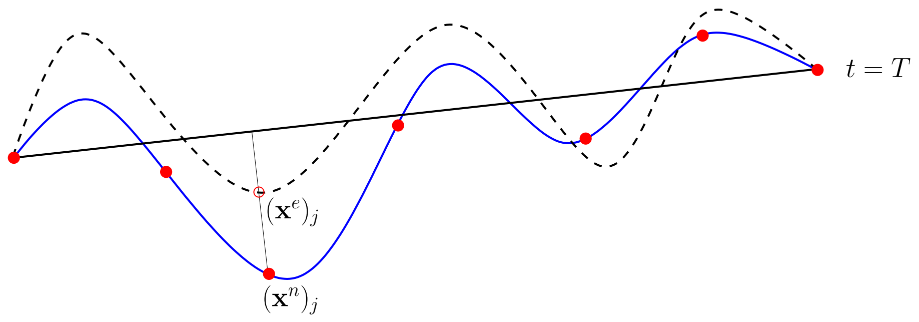


Figure 7: The solid line corresponds to our numerically computed interface at time $T > 0$, while the dashed line represents the corresponding exact interface.

## 4.2 Convergence tests

In order to more clearly see the strengths and the weaknesses of the different approaches, we will first consider three examples where the velocity field is chosen such that the front keeps it shape. For these three tests, we will compare four different approaches:

(i) the new proposed method using Strategy 1 (see Figure 4);
(ii) the new proposed method using Strategy 2 (see Figure 5);
(iii) enforcing the kinematic condition as well a zero tangential grid velocity (denoted as the "Normal" method in the following);
(iv) a pure Lagrangian approach.
The fourth and last test will involve a more complex interface and a more complex velocity field; in this test we will only compare (i) and (iv).

## 4.3 Test 1

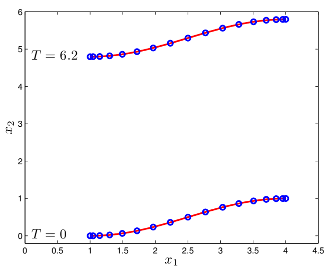
In the first example, the interface motion is only in the $x_2$-direction. The initial front given by

$$x_2(x_1) = \frac{1}{2}\left(1 - \cos\left(\frac{\pi(x_1 - 1)}{3}\right)\right), \qquad 1 \leq x_1 \leq 4,$$

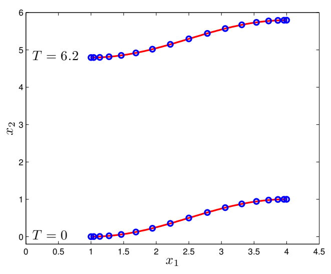
while the prescribed velocity field is given as

$$\begin{aligned} u_1(x_1, t) &= 0, \\ u_2(x_1, t) &= -\frac{1}{2} + \frac{1}{2}e^{t/4}(1 + \cos(\pi t)). \end{aligned}$$

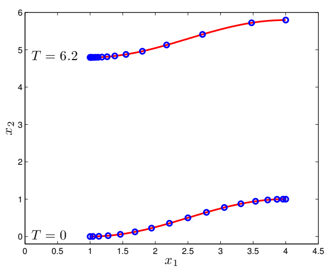
Thus, each particle on the initial front will only move in the $x_2$-direction. Note that, for the Normal approach, we compute the normals *analytically* (which we can do since we know the analytical expression of the front at all times), and not numerically. A numerical computation of the normals will lead to the Normal algorithm breaking down due to the bad interpolation properties when the point distribution becomes poor. Figure 8 shows the initial point distribution and the point distribution at the final time $T = 6.2$ for the four different methods. In Figures 9 and 10 we report the errors $E_1$ and $E_2$, respectively. We observe that all four methods perform well in terms of “hitting” the front. However, for the Normal approach, the point distribution is poor at $T = 6.2$ due to the shape of the front; this again leads to a poor approximation of the length of the front. For the three other methods, the error $E_2$ reaches machine precision since $u_2$ does not depend on $x_1$, and we use a sufficiently large polynomial degree, $N$.



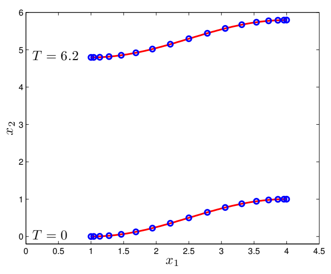
(a) Strategy 1



(b) Strategy 2

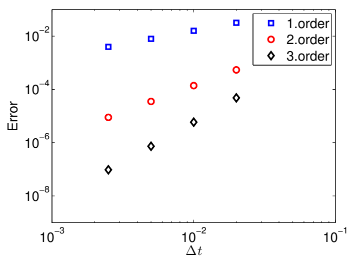


(c) Normal

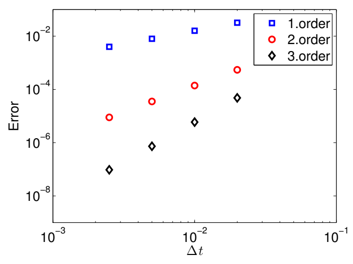


(d) Lagrangian

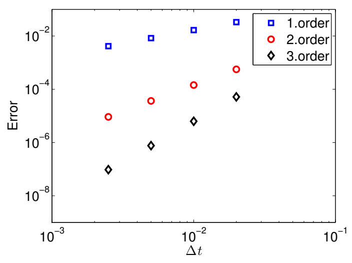
Figure 8: The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 6.2$ for Test 1 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



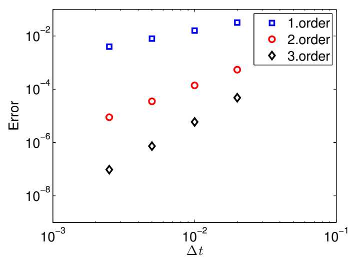
(a) Strategy 1



(b) Strategy 2

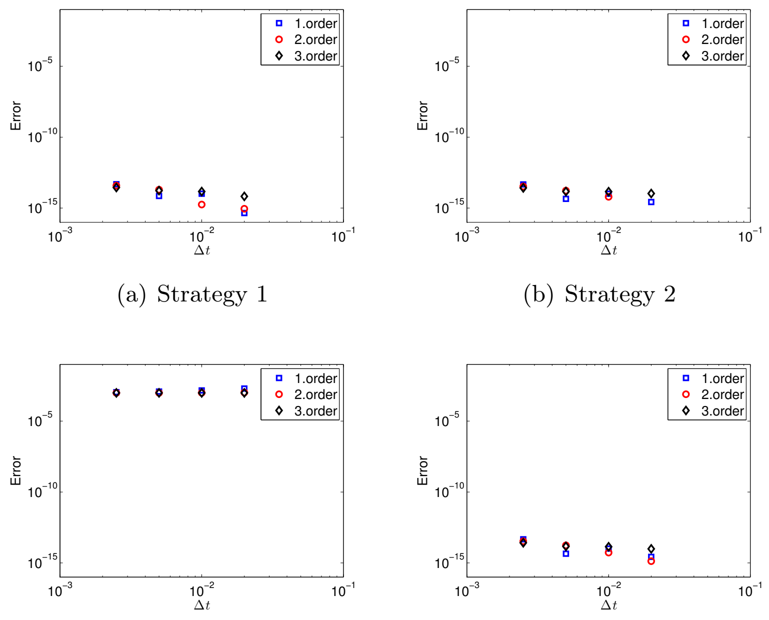


(c) Normal



(d) Lagrangian

Figure 9: The error $E_1$ at time $T = 6.2$ for Test 1 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



(a) Strategy 1 (b) Strategy 2

(c) Normal (d) Lagrangian

Figure 10: The error $E_2$ at time $T = 6.2$ for Test 1 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation. The computed value for the length of the interface, $S_n$, does not converge for the Normal method due to an incorrect point distribution.

## 4.4 Test 2

Next, we will consider a trivial interface, but a velocity field with a more substantial tangential contribution. The initial interface is given by

$$x_2(x_1) = 0, \qquad 1 \leq x_1 \leq 4,$$

while the prescribed velocity field is

$$\begin{aligned} u_1(x_1, t) &= \frac{2}{5} \sin\left(\frac{\pi(x_1 - 1)}{3}\right), \\ u_2(x_1, t) &= -\frac{1}{2} + \frac{1}{2} e^{t/4}(1 + \cos(\pi t)). \end{aligned}$$

Hence, the front has no curvature, and all the normals point in the $x_2$-direction. Thus, $u_1(x, t)$ corresponds to a pure tangential component. Figure 11 shows the initial point distribution and the point distribution at the final time $T = 6.2$ for the four different methods. In Figures 12 we report the error $E_1$. In this example, Strategy 1 and Strategy 2 give identical results. We get first, second and third order temporal convergence in capturing the interface for all four methods.

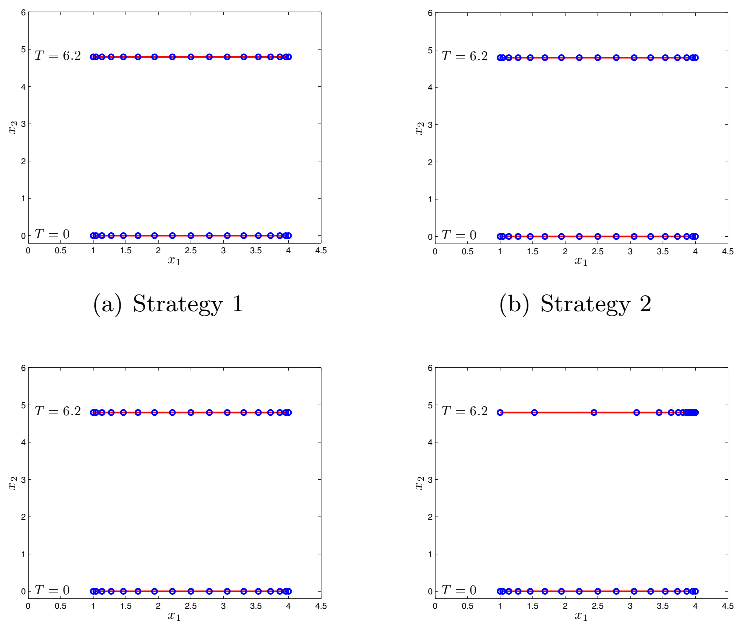
In Figure 11 we see that the pure Lagrangian approach leads to a poor point distribution. However, this is not reflected in the computation of the length due to the simplicity of the front. For this simple interface, (9) reduces to

$$S_n = \sum_{\alpha=0}^{N} \rho_\alpha \left(\frac{\partial(x_1)_N}{\partial \xi}\right)_\alpha = \int_{-1}^{1} \frac{\partial(x_1)_N}{\partial \xi} \, \mathrm{d}\xi,$$

since $\frac{\partial(x_1)_N}{\partial \xi}$ is an $(N-1)$'th degree polynomial which is integrated exactly using GLL quadrature. In addition, since

$$\int_{-1}^{1} \frac{\partial(x_1)_N}{\partial \xi} \, \mathrm{d}\xi = (x_1)_N(1) - (x_1)_N(-1),$$

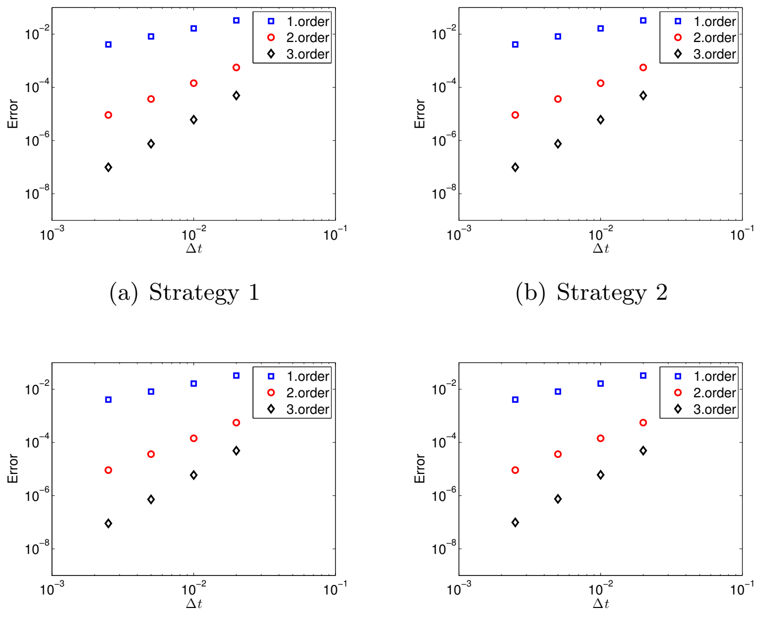
and since the end points are the same for all strategies, we achieve machine precision for all four methods regardless of the quality of the grid. It should be emphasized that this is, indeed, a very special case.



(a) Strategy 1 (b) Strategy 2

(c) Normal (d) Lagrangian

Figure 11: The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 6.2$ for Test 2 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



(a) Strategy 1 (b) Strategy 2

(c) Normal (d) Lagrangian

Figure 12: The error $E_1$ at time $T = 6.2$ for Test 2 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.

## 4.5 Test 3

We now consider a combination of the previous two tests. The initial front is the same as in Test 1, but we prescribe a velocity field with non-zero components in both the $x_1$- and the $x_2$-direction. In particular, we choose the same velocity field as in Test 1, but with the modification that we also add another tangential component. Hence, the interface will still keep its shape during the simulation. Our initial interface is then given by

$$x_2(x_1) = \frac{1}{2}\left(1 - \cos\left(\frac{\pi(x_1 - 1)}{3}\right)\right), \qquad 1 \leq x_1 \leq 4,$$

while the prescribed velocity field is given as

$$\begin{aligned} u_1(x_1, t) &= u_1^t, \\ u_2(x_1, t) &= -\frac{1}{2} + \frac{1}{2}e^{t/4}(1 + \cos(\pi t)) + u_2^t, \end{aligned}$$

with

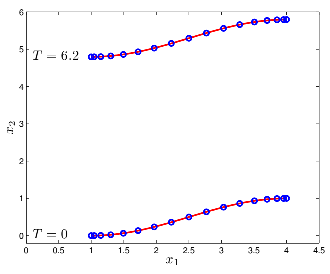
$$\begin{aligned} u_1^t &= \frac{2}{5}\sin\left(\frac{\pi(x_1 - 1)}{3}\right), \\ u_2^t &= \frac{\pi}{6}\sin\left(\frac{\pi(x_1 - 1)}{3}\right)\frac{2}{5}\sin\left(\frac{\pi(x_1 - 1)}{3}\right). \end{aligned}$$

Here, $\mathbf{u}^t = u_1^t\mathbf{e}_1 + u_2^t\mathbf{e}_2$ is a vector which points in the tangential direction of the interface. Figure 13 shows the initial point distribution and the point distribution at the final time $T = 6.2$ for the four different methods.

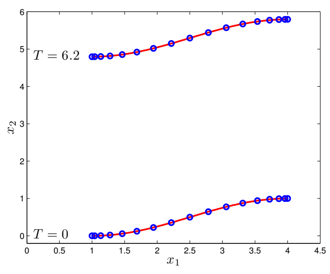
In Figures 14 and 15 we report the errors $E_1$ and $E_2$, respectively. These results are in agreement with the two previous numerical experiments. We observe that all four methods perform well in terms of "hitting" the front.

Strategy 1 and 2 still perform well with respect to both error measures. The Normal approach gives the same results as for Test 1 since the only difference with this example is the addition of a tangential velocity component. The grid quality for a pure Lagrangian approach is poor, which is what we would expect.

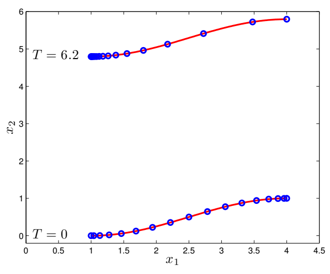
Another thing we observe is that the error level for $E_2$ (the length computation) is about a factor $10^2 - 10^3$ smaller than the error level for $E_1$ (the interface error). The reason for this is that the interface error is rather uniform, which again is due to the simplicity of the problem. This makes the error in the spatial *derivative* of the interface substantially smaller than the interface error, which again leads to a better approximation of the length of the front.



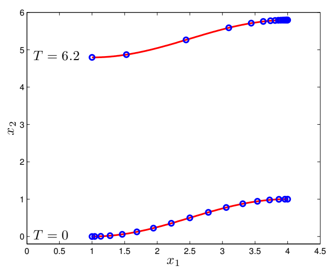
(a) Strategy 1



(b) Strategy 2

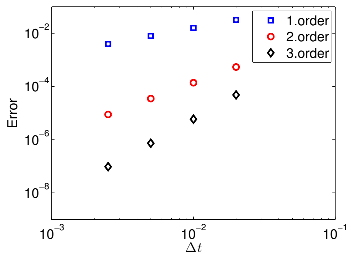


(c) Normal

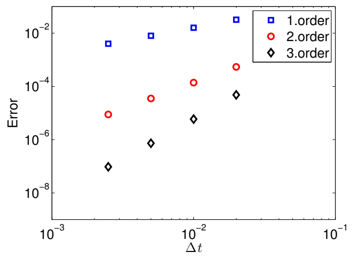


(d) Lagrangian

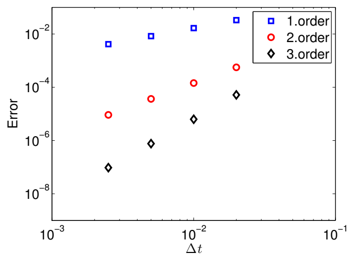
Figure 13: The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 6.2$ for Test 3 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



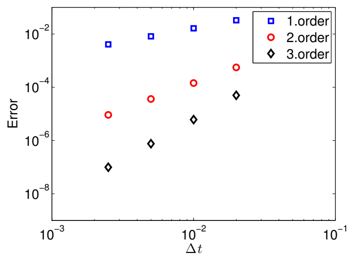
(a) Strategy 1



(b) Strategy 2

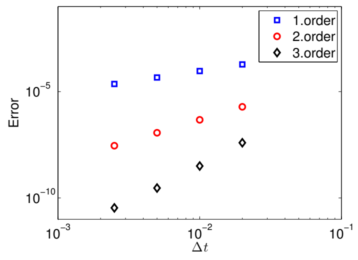


(c) Normal

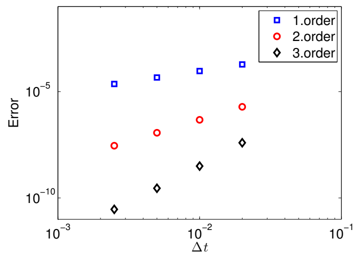


(d) Lagrangian

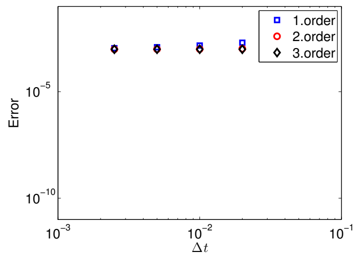
Figure 14: The error $E_1$ at time $T = 6.2$ for Test 3 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



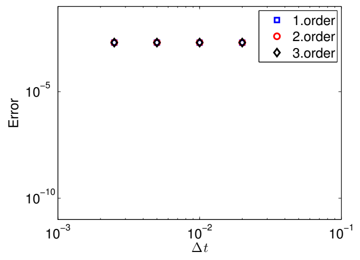
(a) Strategy 1



(b) Strategy 2



(c) Normal



(d) Lagrangian

Figure 15: The error $E_2$ at time $T = 6.2$ for Test 3 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation. The computed value for the length of the interface, $S_n$, does not converge for the Normal method and for the Lagrangian method due to an incorrect point distribution.

### 4.6 Test 4

The three previous test cases where all constructed to illuminate some of the strengths and weaknesses of the different methods for tracking an interface; for this reason they were chosen to be rather simple. We now consider a more complicated numerical example in order to demonstrate the applicability of the new strategy to solve more general problems. We still choose a velocity field which depends on the initial shape of the front, and in such a way that we are able to derive an analytical expression for the shape of the front at all times. A major difference from the previous test cases is that we now choose a time dependent front. In particular, we demand that the *shape* of the front is given by

$$x_2(x_1,t) = \frac{1}{4}\cos(\pi t)\left(\cos\left(\frac{\pi(x_1-1)}{3+0.4t}\right) - \cos\left(\frac{3\pi(x_1-1)}{3+0.4t}\right)\right). \tag{11}$$

Hence, the front will have a time-dependent amplitude and a time-dependent wavelength. We also wish to rotate the front in a circular motion, and in order to achieve this, the velocity field must be chosen in a careful manner. In particular, it consists of four contributions:

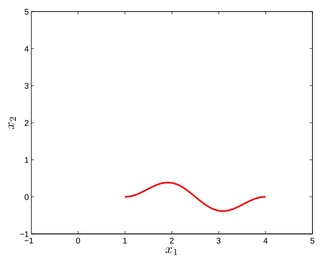
- an angular velocity which is responsible for a pure rotation of the initial front. The angle is computed with respect to a circle with center $(-4, 0)$, and a constant angular velocity $u_\theta = 0.1$ is imposed;
- a velocity field which accounts for the time-dependent amplitude in (11);
- a velocity field which "stretches" the front in accordance with the time-dependent wavelength in (11);
- an additional velocity field which points in the tangential direction on the front.

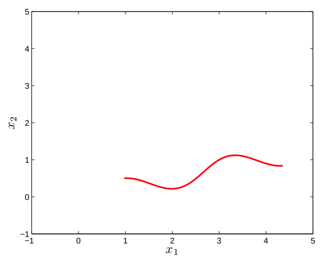
By adding these four contributions, we obtain a smooth, two-dimensional, time-dependent velocity field. Apart from spatial and temporal discretization errors, the initial front $x_2(x_1, t = 0)$ will keep the *shape* given by (11) when "immersed" in our velocity field; see Figure 16.

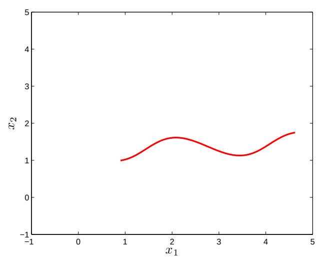
In Figure 17 we show the initial point distribution and the point distribution at the final time $T = 5$ using Strategy 1 and a Lagrangian approach. In Figures 18 and 19 we report the errors $E_1$ and $E_2$ for the two methods, respectively. We see that Strategy 1 performs well both in terms of "capturing" the front and in terms of giving the correct length. The Lagrangian approach is also able to "capture" the front, but the point distribution is poor such that the error in the length of the front is large. Also, compared with Test 3, the difference between the error levels for $E_1$ and $E_2$ is now much smaller; this is due to the time-dependent amplitude in (11), which makes the interface error much less uniform.

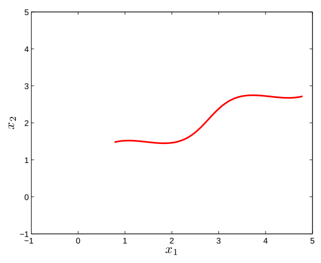
## 5 Conclusions

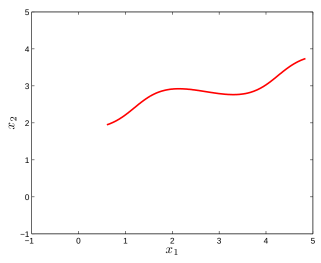
We have presented a new approach for tracking an interface immersed in a given velocity field. The method is particularly relevant to the simulation of unsteady free surface problems using the arbitrary Lagrangian-Eulerian framework. The new method has been constructed with two goals in mind: (i) to be able to accurately follow the interface; and (ii) to maintain a good point distribution for the grid points along the interface. The method combines information from a pure Lagrangian approach with information from an ALE approach. No interpolation of data is needed; instead, we have been able to obtain


(a) $T = 0$


(b) $T = 1$


(c) $T = 2$


(d) $T = 3$


(e) $T = 4$

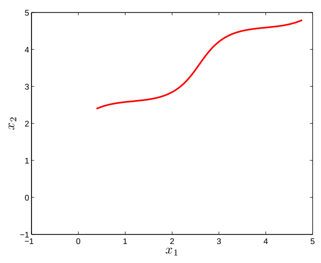
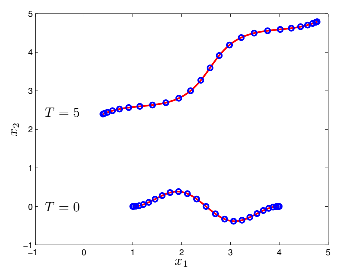
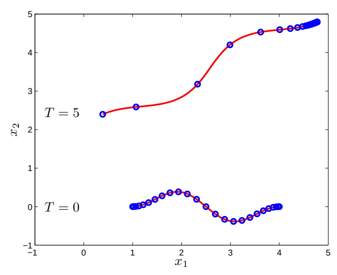

(f) $T = 5$

Figure 16: The front $x_2(x_1, t)$ in (11) at different times when "immersed" in the prescribed velocity field.

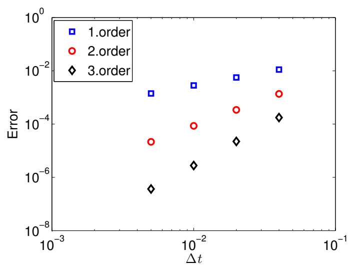


(a) Strategy 1

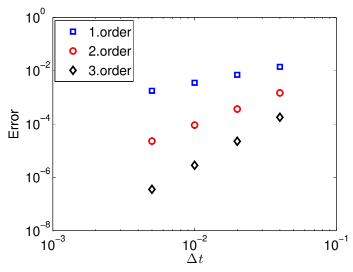


(b) Lagrangian

Figure 17: The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 5$ for Test 4 using Strategy 1 and the Lagrangian approach. A polynomial degree $N = 24$ is used for the spectral approximation.

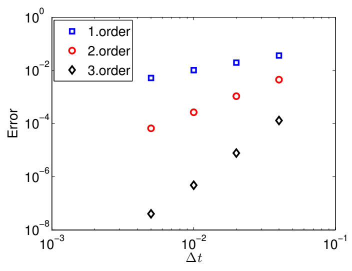


(a) Strategy 1

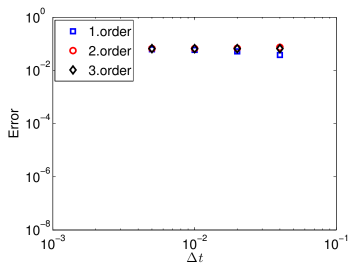


(b) Lagrangian

Figure 18: The error $E_1$ at time $T = 5$ for Test 4 using Strategy 1 and the Lagrangian approach. A polynomial degree $N = 24$ is used for the spectral approximation.



(a) Strategy 1



(b) Lagrangian

Figure 19: The error $E_2$ at time $T = 5$ for Test 4 using Strategy 1 and the Lagrangian approach. A polynomial degree $N = 24$ is used for the spectral approximation. The computed value for the length of the interface, $S_n$, does not converge for the Lagrangian method due to an incorrect point distribution.

the relevant information by solving several pure convection problems along the interface. In this respect, the new approach represents a semi-Lagrangian method applied to surface information.

We have been able to construct two-dimensional model problems offering analytical expressions for both the interface as well as the prescribed velocity field in which the interface (or front) is "immersed". This has allowed us to verify and compare the temporal accuracy of different methods: the new approach, a pure Lagrangian approach, and an approach honoring the kinematic condition in the normal direction, but imposing a homogeneous Dirichlet condition for the tangential component of the grid velocity (called the Normal approach).

Using the new approach we have been able to achieve both of our primary objectives; in particular, we have verified first, second, and third order temporal accuracy for all four model problems. The new method is particularly important in the context of using high order spatial discretization schemes.

Both the Lagrangian approach and the Normal approach generally give a non-optimal point distribution along the interface, something which again may result in large errors in the computation of important surface quantities (e.g., normal and tangent vectors, local curvature, length etc). Such errors may, in worst case, result in a complete breakdown of the interface tracking.

The new method should be extended to, and tested in, more general situations, in particular, by solving real free surface problems using an ALE approach, and by extending the approach to three dimensions.

## Acknowledgment

The authors would like to thank the Norwegian University of Science and Technology for the financial support of this project.

## References

[1] R. Bouffanais and M.O. Deville. Mesh update techniques for free-surface flow solvers using spectral elements. *Journal of Scientific Computing*, 27(1-3):137–149, 2006.

[2] W. Dettmer and D. Perić. A computational framework for free surface fluid flows accounting for surface tension. *Computer Methods in Applied Mechanics and Engineering*, 195:3038–3071, 2006.

[3] J. Donea S. Giuliani and J.P Halleux. An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33:689–723, 1982.

[4] F. Duarte, R. Gormaz, and S. Natesan. Arbitrary Lagrangian-Eulerian method for Navier-Stokes equations with moving boundaries. *Computer Methods in Applied Mechanics and Engineering*, 193:4819–4836, 2004.

[5] F.X. Giraldo. The Lagrange-Galerkin Spectral Element Method on Unstructured Quadrilateral Grids. *Journal of Computational Physics*, 147:114–146, 1998.

[6] C. W. Hirt and B.D. Nichols. Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[7] C.W. Hirt, A.A. Amsden, and J.L Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14:227–253, 1974.

[8] L.W. Ho and A.T. Patera. A Legendre spectral element method for simulation of unsteady incompressible viscous free-surface flows. *Computer Methods in Applied Mechanics and Engineering*, 80:355–366, 1990.

[9] L.W. Ho and A.T. Patera. Variational formulation of three-dimensional viscous free-surface flows: Natural imposition of surface tension boundary conditions. *International Journal for Numerical Methods in Fluids*, 13:691–698, 1991.

[10] A. Huerta and A. Rodríguez-Ferran (eds.). The Arbitrary Lagrangian-Eulerian Formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(39-41):4073–4456, 2004.

[11] A.A. Johnson and T.E. Tezduyar. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering*, 119:73–94, 1994.

[12] Y. Maday and A.T. Patera. Spectral element methods for the Navier-Stokes equations. *in: A.K. Noor, J. T. Oden (Eds.), State of the Art Surveys in Computational Mechanics, ASME, New York*, pages 71–143, 1989.

[13] Y. Maday, A.T. Patera, and E.M. Rønquist. An Operator-Integration-Factor Splitting Method for Time-Dependent Problems: Application to Incompressible Fluid Flow. *Journal of Scientific Computing*, 5(4):263–292, 1990.

[14] S. Osher and R.P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[15] S. Osher and J.A. Sethian. Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[16] B. Ramaswamy and M. Kawahara. Arbitraty Lagrangian-Eulerian finite element method for unsteady convective incompressible viscous free surface flow. *International Journal for Numerical Methods in Fluids*, 7:1053–1074, 1987.

[17] J.A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge University Press, 1999.

[18] M.A. Walkey, P.H. Gaskell, P.K. Jimack, M.A. Kelmanson, and J.L. Summers. Finite Element Simulation of Three-Dimensional Free-Surface Flow Problems. *Journal of Scientific Computing*, 24(2):147–162, 2005.

[19] D. Xiu and G.E. Karniadakis. A Semi-Lagrangian High-Order Method for Navier-Stokes Equations. *Journal of Computational Physics*, 172:658–684, 2001.

[20] J. Xu, D. Xiu, and G.E. Karniadakis. A Semi-Lagrangian Method for Turbulence Simulations Using Mixed Spectral Discretizations. *Journal of Scientific Computing*, 17(1-4):585–597, 2002.

[21] H. Zhou and J.J. Derby. An assessment of a parallel, finite element method for three-dimensional, moving-boundary flows driven by capillarity for simulation of viscous sintering. *International Journal for Numerical Methods in Fluids*, 36:841–865, 2001.