COMMUN. MATH. SCI. © 2009 International Press
Vol. 7, No. 2, pp. 365–398

FAST INTERFACE TRACKING VIA A MULTIRESOLUTION REPRESENTATION OF CURVES AND SURFACES[*]

OLOF RUNBORG[†]

Abstract. We consider the propagation of an interface in a velocity field. The initial interface is described by a normal mesh [Guskov, et al, SIGGRAPH Proc., 259-268, 2000] which gives us a multiresolution decomposition of the interface and the related wavelet vectors. Instead of tracking marker points on the interface we track the wavelet vectors, which like the markers satisfy ordinary differential equations. We show that the finer the spatial scale, the slower the wavelet vectors evolve. By designing a numerical method which takes longer time steps for finer spatial scales we are able to track the interface with the same overall accuracy as when directly tracking the markers, but at a computational cost of $O(\log N/\Delta t)$ rather than $O(N/\Delta t)$ for N markers and timestep Δt. We prove this rigorously and give numerical examples supporting the theory. We also consider extensions to higher dimensions and co-dimensions.

Key words. Interface tracking, multiresolution analysis, normal meshes, fast algorithms.

AMS subject classifications. 65L20, 42C40, 65D10.

1. Introduction

Tracking the evolution of interfaces or fronts is important in many applications, for instance, wave propagation, multiphase flow, crystal growth, melting, epitaxial growth and flame propagation. The interface in these cases is a manifold of co-dimension one which moves according to some physical law that depends on the shape and location of the interface. We suppose for convenience that it can be parameterized, so that for a fixed time t the interface is described by the function $x(t,s):\mathbb{R}^+\times\mathbb{R}^q\to\mathbb{R}^d$, with the parameterization $s\in\Omega\subset\mathbb{R}^q$ and $q=d-1$. In this paper we consider the simplified case when the interface is moving in a time-varying velocity field that does not depend on the shape of the front, only its location. Then $x(t,s)$ satisfies the parameterized ordinary differential equation (ODE)

$$\frac{\partial x(t,s)}{\partial t}=F(t,x(t,s)),\qquad x(0,s)=\gamma(s),\qquad s\in\Omega,\tag{1.1}$$

where $F(t,x):\mathbb{R}^+\times\mathbb{R}^d\to\mathbb{R}^d$ is a given function representing the velocity field and $\gamma(s):\mathbb{R}^q\to\mathbb{R}^d$ is the initial interface. We will mostly discuss curves in two dimensions, $d=2$, $q=1$, but we will also discuss extensions to higher dimensions $d=3$, $q=2$ and co-dimensions $d=3$, $q=1$. Applications could include the tracking of physically motivated interfaces, like wavefronts in high frequency wave propagation problems, or "artificial" fronts of propagation paths parameterized by initial data, where a problem has the structure of (1.1) even though the front has no direct physical interpretation. This could be, for instance, iso-distance curves on a surface (front of geodesics), fiber tract bundles in brain imaging, or the method of characteristics for the solution graph of hyperbolic partial differential equations (PDEs). In many of these problems it is better numerically to consider a front rather than a set of individual paths, since the connectivity between paths is then maintained, which for example simplifies interpolation between them.

[*]Received: August 19, 2008; accepted (in revised version): February 1, 2009. Communicated by Peter Smereka.

[†]Department of Numerical Analysis, CSC, KTH, 100 44 Stockholm, Sweden (olofr@nada.kth.se). URL: www.nada.kth.se/~olofr

Numerical methods for this problem include the Lagrangian front tracking method [15], which has been used extensively in e.g. multiphase flow [14, 40] and geophysics [41, 27]. There are also Eulerian approaches like the level set [32, 31] and segment projection [39, 12] methods. Related algorithms have been proposed for computing invariant manifolds in dynamical systems [26, 17]. For flow problems we should also mention the marker-and-cell (MAC) [42] and volume of fluid (VOF) [21] methods.

We focus here on front tracking, in which the interface is described by a set of marker points that are connected in a known topology. In one dimension one would approximate $x_j(t) \approx x(t,s_j)$ and use a numerical method for ODEs to solve

$$\frac{dx_j(t)}{dt} = F(t,x_j(t)), \qquad x_j(0) = \gamma(s_j),$$

where $s_0 < s_1 < \ldots < s_N$ is a discretization of Ω. For surfaces in three dimensions, the markers on the interface are typically held together in a triangulation. Propagating one marker numerically with a timestep Δt to a fixed time costs $O(1/\Delta t)$ operations. Hence, if the interface is represented by N points the cost of standard front tracking is $O(N/\Delta t)$. As a comparison, the standard level set method costs $O(N^2/\Delta t)$ if the full domain is discretized in 2D. There are, however, several clever versions that localize the computations around the interface, e.g. local level set methods [33] and tree methods [36, 37]. These bring the complexity down to $O(N \log N/\Delta t)$, which is almost the same as front tracking.

In this article we describe an interface tracking algorithm that uses a multiresolution representation of the interface instead of point values. The representation is based on *normal meshes* [6, 18], which is an efficient way to describe curves and surfaces. Our main result is a proof, backed up by numerical experiments, that for fixed small enough accuracy the cost of our method is only $O(\log N/\Delta t)$ or even $O(1/\Delta t)$. Thus, asymptotically the cost to propagate the whole curve is of the same order as the cost of propagating just one point, i.e., an order better than for front tracking and level sets. In the method, the interface is represented by wavelet vectors which correspond to the details of the interface on different scale levels. It is well-known that for a fixed curve or surface, the size of these vectors decays rapidly as the scale becomes finer. In the dynamic setting we show that the *time derivatives* of the wavelet vectors decay in a similar manner. This means that the fine scales evolve slower than the coarse scales of the interface. Our method exploits this by taking shorter timesteps for the coarse scales than for the fine scales. This greatly reduces computational cost without affecting the overall accuracy.

Adaptivity is usually an important feature of front tracking algorithms. Marker points on the interface will typically spread out which results in a badly resolved interface. One then needs to adaptively add new points in between the old ones, when the marker points gets too wide apart. This can be done via interpolation. We will, however, not discuss adaptation strategies for the new method in this paper. We just note in the conclusion that the multiresolution framework offers many opportunities for such improvements. Let us also add that for level set and segment projection methods, adaptivity is less of an issue since they use Eulerian grids. They are also much better suitable for handling topological changes in the interface; like with other front tracking based algorithms, this would be difficult with the proposed method.

For time-dependent differential equations there are a number of results similar in style to the ones in this paper. They are related to what Demanet and Ying [7] call *time-upscaling:* For a problem spatially discretized with N points in each coordinate direction, time upscaled methods are able to compute the solution at $O(1)$ time levels at a $O(N^d)$ cost, for fixed accuracy, instead of $O(N^{d+1})$ needed by standard method where the time step must be of the same order as the spatial mesh size for stability and accuracy reasons. In our methods, if we discretize the interface with a mesh size of the same order as the time step, $N \sim 1/\Delta t$, then we can reduce the cost of tracking a curve from $O(N^2)$ to $O(N)$, and following this terminology we could call our results time upscaling for interface tracking. Time-upscaled methods were proposed for the advection and parabolic equations in one [11] and higher dimensions [28, 16], for the wave equation in one dimension [35] and two dimensions [7] as well as for the computation of the phase map for dynamical systems in the phase flow method [43]. Techniques used in these methods to accelerate the computations include transforming the problem into wavelet like bases [11, 35, 7], sparse grids [45, 1, 28, 16] and repeated squaring of the solution operator [11, 7, 43]. This last idea has also been used to speed up the time-independent problem of the Helmholtz equation in waveguides with periodic structures [19, 44]. The technique introduced by Stolk in [35] is closer to the one in this paper. After transforming one-way formulations of the wave equation into wavelet bases he is able to reduce computational cost by using *multiscale timestepping;* in much the same way as in the present paper he integrates the fine (spatial) scales with longer time steps than the coarse scales. This also bears a resemblance to a recent result by Giles [13] who uses a hierarchy of solutions obtained with different time steps to reduce the computational cost for weak approximations of stochastic ordinary differential equations.

This article is organized as follows. In section 2 the multiresolution representation of the interface is presented and the governing ODEs are derived. The numerical methods used to solve those ODEs are given in some detail in section 3 where also a preliminary analysis and explanation of the advantages of those methods are given. Precise analysis of the methods is carried out in section 4. Numerical experiments with the basic methods are presented in section 5. Extensions to higher order, dimensions and co-dimensions with additional numerical experiments are the topics of section 6. section 7 concludes the paper and discusses some open problems.

2. Multiresolution description of the interface

In standard front tracking algorithms marker points are used to represent the interface. We will instead consider a multiresolution representation, which is often a more efficient way to describe curves and surfaces. *Multiresolution meshes* are a popular tool used to approximate static curves and surfaces. They consist of a hierarchy of increasingly detailed meshes. Each new mesh level is computed from the previous one by first predicting a new point, for instance by using so-called subdivision schemes, and then correcting the predicted point by a wavelet (or detail) vector. Only the wavelet vectors need to be stored and because of the surface smoothness most wavelet vectors will be small, lending the representation well to compression.

In our case we will use a special type of multiresolution mesh for the initial data. These are called *normal meshes*. In the time-evolution of the interface, we relax the constraints in the description but still use the basic multiresolution mesh setting.

2.1. Static case.

Let us first consider a static curve in $\mathbb{R}^2$ given by $\gamma(s): [0,1] \to \mathbb{R}^2$. We assume it to be twice continuously differentiable in s and non-self-intersecting. In the normal approximation procedure proposed in [18] the original curve γ is described by successively finer approximations, in the form of piecewise linear curves γ_j which connect the vertices $x_{j,k}$, $k=0,\ldots,2^j$, on γ. The process is illustrated in figure 2.1. The initial approximation γ_0 is the line connecting the edge points $x_{0,0}$ and $x_{0,1}$. To construct the vertices at level $j+1$ from those at level j, we first set $x_{j+1,2k}=x_{j,k}$; this makes the construction interpolating. We next compute a point $x_{j+1,2k+1}$ that lies in between the two old points $x_{j,k}$ and $x_{j,k+1}$. This is done by first computing a *predicted point* $x^*_{j+1,2k+1}$ as an average of the neighboring points,

$$x^*_{j+1,2k+1}=\frac{x_{j,k}+x_{j,k+1}}{2}. \tag{2.1}$$

We then add a detail offset by drawing a line from $x^*_{j+1,2k+1}$ in the direction orthogonal to the line segment $(x_{j,k},x_{j,k+1})$. This line is guaranteed[1] to cross the curve segment between $x_{j,k}$ and $x_{j,k+1}$ and we call this *corrected* point $x_{j+1,2k+1}$. As this procedure continues, the polyline γ_j comes closer and closer to γ. We can interpret this as a wavelet transformation similar to the notion of lifting [38], where $x^*_{j+1,2k+1}$ is a prediction of the real point $x_{j+1,2k+1}$ computed based only on coarser information. Then the detail offset

$$w_{j,k}:=x_{j+1,2k+1}-x^*_{j+1,2k+1} \tag{2.2}$$

is a wavelet vector, which is what we will call it henceforth.

In order to reconstruct the polyline γ_j we only need to store the edge points $x_{0,0}$, $x_{0,1}$, and the wavelet vectors $w_{j',k}$ for $j'<j$, since at each level the predicted point is based only on coarser information. In fact, since also the normal to the segment $(x_{j,k},x_{j,k+1})$ again only depends on coarser data we just need to store the *length* $|w_{j,k}|$ and one sign bit to characterize γ_j completely. The normal approximation thus allows a purely *scalar* representation of the curve. We will call $\{w_{j,k}\}$ and $\{x_{j,k}\}$ a normal multiresolution representation of $\gamma(s)$. The construction also defines the break point values $\{s_{j,k}\}$ in parameter space by the relation $\gamma(s_{j,k})=x_{j,k}$.

As was shown in [6], the size of the wavelet vectors decays exponentially with the level j. The prediction in (2.1) can be seen as the application of the simplest of subdivision schemes, the "midpoint scheme." The prediction can be improved by using higher order subdivision schemes which take into account more neighboring points, resulting in a faster decay rate of $w_{j,k}$. We will need this when we construct higher order schemes in section 6.1.

The fast decay of the wavelet vectors and the fact that one just needs to store one floating point number for each vertex, instead of the standard 2- or 3-vector, give normal meshes good compression properties [23, 29, 24]. The normal representation permits the use of standard scalar compression codes. It also indirectly improves compression rates since it contains little redundant parameterization information [24].

2.2. Dynamic case. We now consider the dynamic case when the curve moves in a velocity field. The curve is then given by $x(t,s)$ at time t, and satisfies the parameterized ODE

$$\frac{\partial x(t,s)}{\partial t}=F(t,x(t,s)), \qquad 0\le s\le 1, \quad t>0, \qquad x(0,s)=\gamma(s).$$

We assume that there is a normal approximation of the initial curve $\gamma(s)$ given by the wavelet vectors $\{w_{j,k}\}$ and parameter space break points $\{s_{j,k}\}$. We then define the time-dependent vertices on the curve

$$x_{j,k}(t):=x(t,s_{j,k}), \qquad 0\le k\le 2^j,$$

[1]There may be several points where the line segment crosses the curve. In that case we can take any one of them.

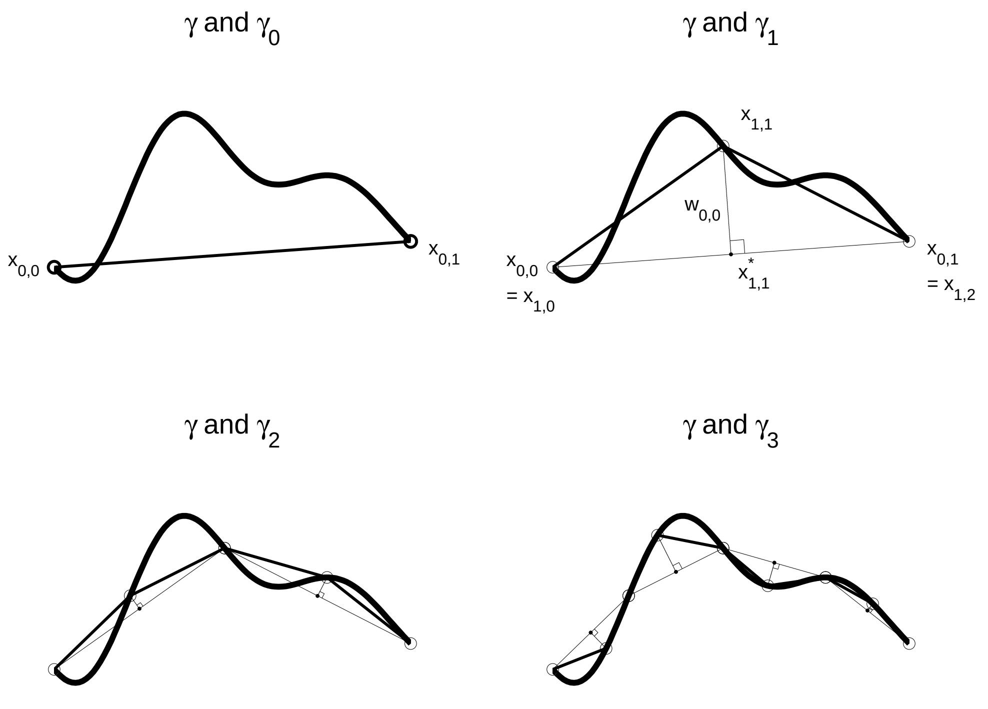


FIG. 2.1. *Example of the normal mesh algorithm using the mean value of adjacent points as predictor.*

such that

$$\frac{dx_{j,k}(t)}{dt} = F(t, x_{j,k}(t)).$$

At $t = 0$ we have by the definitions in (2.1) and (2.2) that

$$x_{j+1,2k} = x_{j,k}, \qquad w_{j,k} = x_{j+1,2k+1} - \frac{1}{2}(x_{j,k} + x_{j,k+1}), \tag{2.3}$$

where $w_{j,k}$ is normal to $x_{j,k+1} - x_{j,k}$. We take this expression as the definition of $w_{j,k}$ also for $t > 0$. We thus relax the strict normality condition and in general $w_{j,k} \not\perp x_{j,k+1} - x_{j,k}$ for $t > 0$. We obtain

$$\begin{aligned}\frac{dw_{j,k}}{dt} &= F(t, x_{j+1,2k+1}) - \frac{1}{2}(F(t, x_{j,k}) + F(t, x_{j,k+1}))\\ &= F\left(t, \frac{x_{j,k} + x_{j,k+1}}{2} + w_{j,k}\right) - \frac{1}{2}(F(t, x_{j,k}) + F(t, x_{j,k+1})).\end{aligned} \tag{2.4}$$

Setting

$$G(t, y, z, w) = F\left(t, \frac{y+z}{2} + w\right) - \frac{F(t,y) + F(t,z)}{2},$$

we thus have the following alternative system of ODEs

$$\frac{dw_{j,k}}{dt} = G(t, x_{j,k}, x_{j,k+1}, w_{j,k}), \qquad \frac{dx_{0,0}}{dt} = F(t, x_{0,0}), \qquad \frac{dx_{0,1}}{dt} = F(t, x_{0,1}), \tag{2.5}$$

which together with (2.3) describe the dynamics of the system.

REMARK 2.1. When F is constant, corresponding to rigid motion, then all of the $\{w_{j,k}\}$ are constant and the dynamics of the system is completely determined by the motion of the edge points $x_{0,0}$ and $x_{0,1}$. When $F = Ax$ is linear, then $w'_{j,k} = Aw_{j,k}$ for all j,k and the equations for the wavelet coefficients decouple completely.

REMARK 2.2. Fine scales depend on coarser scales, but there is no dependence in the other direction. This means that we can compute the different scale levels sequentially from coarse to fine, one after the other.

REMARK 2.3. It is not possible to write a closed ODE system for $w_{j,k}(t)$ which maintains the normality of the wavelet vectors. To do this, more information about the initial curve is needed then the location of the initial marker points $x_{j,k}(0)$.

3. Basic methods

In this section we will describe two basic methods for tracking the interface in a rapid way, by solving (2.5) together with (2.3). The methods will be carefully analyzed in section 4. It is well-known that for the normal approximation of curves described above the wavelet coefficients $w_{j,k}$ decay as 2^{-2j} when the curve is at least C^2, see Proposition 4.2 which was proved in [6]. The main motivation for the methods given below is Theorem 4.3 in section 4.1 which shows that if the initial curve is described by a normal approximation, then also the time derivatives of the wavelet vectors $\{w_{j,k}\}$ decay as 2^{-2j} over a fixed time interval. An interpretation of this is that the evolution of the system takes place on different time scales. Fine spatial scales change slower than coarse spatial scales. Since the rate of change in $w_{j,k}$ is much smaller at finer than at coarser levels, we will use longer timesteps for the fine scales.

Let us define the methods more precisely. We denote the numerical approximations to be

$$x^n_{j,k} \approx x_{j,k}(t_n), \qquad w^n_{j,k} \approx w_{j,k}(t_n), \qquad t_n = n\Delta t,$$

where Δt is the reference timestep. We are interested in computing the solution up to time T and we assume that there is a positive integer M such that $T = M\Delta t$. We also assume that the number of points is an even power of two, $N = 2^J$ for some integer $J > 0$. Letting I_j be the index set $0,\dots,2^j$ and $\bar{I}_j$ the index set $0,\dots,2^j - 1$, then $x_{j,k}$ is defined for all $k \in I_j$ and $w_{j,k}$ for all $k \in \bar{I}_j$. These values are related via the reconstruction

$$x^n_{j+1,2k+1} = \frac{x^n_{j,k} + x^n_{j,k+1}}{2} + w^n_{j,k}, \quad k \in \bar{I}_j, \qquad x^n_{j+1,2k} = x^n_{j,k}, \qquad k \in I_j. \tag{3.1}$$

At different refinement levels j we will use different timesteps, denoted $\Delta t_j := m_j \Delta t$, where $m_j \in \mathbb{Z}^+$ and $m_0 = 1$. We also require that $m_{j+1} = q_j m_j$ for some positive integer q_j. (Typically in this method we take $q_j = 2$ to double the time step in each level, and hence have $m_j = 2^j$.) We then consider two methods.

Method 1: forward Euler. On the zeroth level, for $t_{n+1} \le T$,

$$x^{n+1}_{0,0} = x^n_{0,0} + \Delta t F(t_n, x^n_{0,0}), \qquad x^{n+1}_{0,1} = x^n_{0,1} + \Delta t F(t_n, x^n_{0,1}), \tag{3.2}$$

and on level $j > 0$, for $t_{(n+1)m_j} \le T$ and $k \in \bar{I}_j$,

$$w^{(n+1)m_j}_{j,k} = w^{nm_j}_{j,k} + \Delta t_j G\left(t_{nm_j}, x^{nm_j}_{j,k}, x^{nm_j}_{j,k+1}, w^{nm_j}_{j,k}\right), \tag{3.3}$$

where $\{x_{j,k}\}$ are computed from $\{w_{j-1,k}\}$ and $\{x_{j-1,k}\}$ using (3.1).

Method 2: Runge-Kutta 2. This method is also known as Heun's method. On the zeroth level, for $t_{n+1} \le T$,

$$\begin{aligned} k_1 &= F\left(t_n, x_{0,0}^n\right), & k_1 &= F\left(t_n, x_{0,1}^n\right), \\ k_2 &= F\left(t_{n+1}, x_{0,0}^n + \Delta t k_1\right), & k_2 &= F\left(t_{n+1}, x_{0,1}^n + \Delta t k_1\right), \\ x_{0,0}^{n+1} &= x_{0,0}^n + \frac{\Delta t}{2}(k_1 + k_2), & x_{0,1}^{n+1} &= x_{0,1}^n + \frac{\Delta t}{2}(k_1 + k_2). \end{aligned} \tag{3.4}$$

At level $j > 0$, for $t_{(n+1)m_j} \le T$ and $k \in \bar{I}_j$,

$$\begin{aligned} k_1 &= G\left(t_{nm_j}, x_{j,k}^{nm_j}, x_{j,k+1}^{nm_j}, w_{j,k}^{nm_j}\right), \\ k_2 &= G\left(t_{(n+1)m_j}, x_{j,k}^{(n+1)m_j}, x_{j,k+1}^{(n+1)m_j}, w_{j,k}^{nm_j} + \Delta t_j k_1\right), \\ w_{j,k}^{(n+1)m_j} &= w_{j,k}^{nm_j} + \frac{\Delta t_j}{2}(k_1 + k_2). \end{aligned} \tag{3.5}$$

where, as in Forward Euler, $\{x_{j,k}\}$ are computed from results at the coarser level using (3.1).

The methods above define the values of $w_{j,k}^n$ for $n = 0, m_j, 2m_j, \ldots, n'm_j \le M$. If we want to compute the shape of the curve at other time levels we need to add one more reconstruction step. For $nm_j + r \le M$, with $1 \le r < m_j$ we define

$$w_{j,k}^{nm_j+r} = w_{j,k}^{nm_j} + \frac{r}{m_j}\Delta t_j G\left(t_{nm_j}, x_{j,k}^{nm_j}, x_{j,k+1}^{nm_j}, w_{j,k}^{nm_j}\right) \tag{3.6}$$

for both methods. In particular, this is the only step taken for large enough j, where $m_j > M$.

3.1. Complexity and accuracy. Let us now derive the time complexity of the methods above and make a non-rigorous analysis of their approximation errors. For the complexity we suppose that we want to find the solution at the fixed time T which is of order $O(1)$. We start by noting that the complexity of a standard method for the front tracking problem is $O(N/\Delta t)$, since there are N unknowns that are each moved $O(T/\Delta t)$ time steps. The accuracy is $O(\Delta t^p)$ for a p-th order method.

As was remarked above, in the fast methods the data for each level only depends on the results from the previous level. Hence, once level j has been computed, the results enter as a variable coefficient in level $j+1$. Method 1 and 2 can be implemented in this way and the computational cost of the methods is therefore simply the sum of the costs on each level. The number of unknowns on level j is 2^j and the number of timesteps needed is first $\lfloor T/\Delta t_j \rfloor$ of the type (3.3) or (3.5) and then for $j > 0$ at most one step with (3.6). Supposing $N = 2^J$, the total cost is then proportional to

$$\sum_{j=0}^{J} 2^j \left(\left\lfloor \frac{T}{\Delta t_j} \right\rfloor + 1 \right) \sim \frac{T}{\Delta t} \sum_{j=0}^{J} \frac{2^j}{m_j} + N. \tag{3.7}$$

We now consider the accuracy of the methods disregarding the final step (3.6). Letting $\tau_{j,k}^n$ denote the local truncation error in time step n for wavelet coefficient k at level j, we assume that $\tau_{j,k}^n$ is given by the $(p+1)$-th order derivative of the exact solution for a p-th order method,

$$\tau_{j,k}^n \sim \Delta t_j^{p+1} \frac{d^{p+1} w_{j,k}}{dt^{p+1}}.$$

We define ε_j as the global truncation error at level j,

$$\varepsilon_j = \sup_{\substack{k\in I_j \\ 0\le t_n\le T}} |x^n_{j,k} - x_{j,k}(t_n)|.$$

Assuming stability, ε_j is bounded by the sum of all local truncation errors,

$$\varepsilon_j \le \sum_{j'=0}^{j} \sum_{n'=0}^{\lfloor T/\Delta t_{j'}\rfloor} \max_{k\in \bar I_{j'}} |\tau^{n'}_{j',k}|.$$

If we use the result from Theorem 4.3 and still assume $N=2^J$, the global error can then be estimated as

$$\varepsilon_J \le \sum_{j=0}^{J} \sum_{n=0}^{\lfloor T/\Delta t_j\rfloor} \max_{k\in \bar I_j} \left|\Delta t_j^{p+1} \frac{d^{p+1}w_{j,k}}{dt^{p+1}}\right| \le T\sum_{j=0}^{J} \Delta t_j^p \left|\frac{d^{p+1}w_{j,k}}{dt^{p+1}}\right|_\infty \le CT\Delta t^p \sum_{j=0}^{J} m_j^p 2^{-2j}. \tag{3.8}$$

This is in fact essentially what is proved rigorously in Theorem 4.1 in section 4 with $p=1$ for Method 1 (Forward Euler) and $p=2$ for Method 2 (Runge-Kutta 2).

In the standard configuration of the methods we will double the timestep in each level, giving $m_j = 2^j$. Then, if $T=O(1)$ and $N=2^J$, by (3.7) the complexity for Method 1 is $O(J/\Delta t + N) = O(\log_2 N/\Delta t + N)$ and by (3.8) the accuracy is $O(\Delta t)$, independent of N since the sum in (3.8) is convergent. Another possible choice is to take $m_j = 4^j$, quadrupling the timestep in each level. By the same arguments the complexity is then a little better, $O(1/\Delta t + N)$, and the accuracy is a little worse, $O(\log_2 N\Delta t)$.

We note that it is in fact possible to turn both (3.7) and (3.8) into convergent sums for Method 1. We can pick m_j such that $2^j < m_j < 2^{2j}$; for instance $m_j = 3^j$, corresponding to tripling the timestep in each level. This is not the only possibility though. We recall the restriction that $m_{j+1} = q_j m_j$ with $q_j \in \mathbb{Z}$. One can thus use different factors in each level to obtain an arbitrary growth rate of m_j, e.g. if $q_j = 2,3,2,3,\ldots$ then $m_j \le 6^{j/2} \approx 2^{1.29j}$, etc. These choices all give a complexity of $O(1/\Delta t + N)$ and an accuracy of $O(\Delta t)$. Although asymptotically yielding better accuracy for a given cost, it is not clear that these choices are superior in practice to the simple doubling $m_j = 2^j$ strategy, however, see section 5.1.

We would like to stress that the fast methods above have a complexity that is significantly improved as compared to the standard methods, while still achieving the same order of accuracy. In fact, when $\Delta t < T/N$ the asymptotic cost of the fast methods is of the same order as when computing the trajectory of only *one* marker on the interface, despite the fact that we compute the dynamics of the whole curve.

Remark 3.1. Disregarding the final time step (3.6) the fast algorithms compute the wavelet vectors $w^n_{j,k}$ for

$$j=0,\ldots,J \qquad k\in \bar I_j, \qquad n=0,m_j,\ldots,n'_j m_j,$$

with $n'_j = \lfloor T/\Delta t_j \rfloor$. In total, if we take $m_j = 2^j$, then

$$\sum_{j=0}^{J} 2^j \left\lfloor \frac{T}{\Delta t_j}\right\rfloor \sim O(\log_2 N/\Delta t)$$

wavelet vectors are computed. Given these, we want to construct an approximation of the curve at a time level t_n. We first reconstruct all $x^n_{j,k}$ corresponding to the computed $w^n_{j,k}$, then time step with (3.6) wherever necessary and finally reconstruct the curve at the chosen time level. Each of these three steps has an $O(N)$ complexity. We can hence conclude that if we use $O(\log_2 N/\Delta t)$ memory to store the computed values, we can, a posteriori, get the computed interface for any time level t_n at a $O(N)$ cost, which is the optimal cost since there are N marker points on the interface. We can therefore also think of the computed wavelet vectors as an efficient description of the whole surface that the interface sweeps out in time, $\{x(t,s): 0\leq t\leq T,\ s\in\Omega\}\subset \mathbb{R}^+\times\mathbb{R}$ (the Lagrangian submanifold of phase space). In fact, the wavelet vectors are obtained precisely on a sparse grid in (t,s)-space, which is known to be an efficient format [45, 1].

4. Analysis of the method

In this section we analyze the errors in the methods proposed in section 3 and derive an error estimate which confirms the simple analysis in section 3.1 when the velocity field and initial curve is sufficiently smooth and bounded. The initial curve $x(0,s)=\gamma(s)$ thus has the normal multiresolution representation $\{w_{j,k}(0)\}$, and $\{x_{j,k}(0)\}$. The total number of points N is an even power of two, and we define J as the finest level so that $N=2^J$. The time-evolution of $\{w_{j,k}\}$ and the node points $\{x_{j,k}\}$ are approximated by Forward Euler (Method 1) or Runge-Kutta 2 (Method 2). In the analysis we use the function spaces $C^p(\Omega_1;\Omega_2)$, which denote all measurable functions from Ω_1 to Ω_2 with continuous derivatives up to order p, and $C^p_b(\Omega_1;\Omega_2)$, where all derivatives are also bounded. We can then prove the following theorem.

THEOREM 4.1. *Suppose there are positive integers M and q_j such that $T=M\Delta t$ and $m_{j+1}=q_j m_j$ with $m_0=1$. We assume that there is a constant C independent of J such that*

$$\sum_{j=0}^{J} m_j^p 2^{-2j}\leq C\,J^r,$$

for some $r\geq 0$, where $p=1$ for Forward Euler and $p=2$ for Runge-Kutta 2. If $F\in C_b^{p+2}(\mathbb{R}^+\times\mathbb{R}^2;\mathbb{R}^2)$ and $\gamma\in C^2([0,1];\mathbb{R}^2)$ then there exist unique solutions $w_{j,k}(t)$ and $x_{j,k}(t)$ in $C^{p+3}(\mathbb{R}^+)$ to (2.5) together with (2.3) for all times. Moreover, there are constants C',D independent of J and Δt such that

$$\sup_{\substack{k\in I_J\\ 0\leq t_n\leq T}} |x^n_{J,k}-x_{J,k}(t_n)|\leq C'\Delta t^p \sum_{j=0}^{J} m_j^p 2^{-2j}, \qquad 0\leq J\leq \frac{D}{\Delta t^{\frac{p}{r+1}}}. \tag{4.1}$$

In the proof of the theorem we will first derive some estimates of the exact solution in section 4.1. These lead up to Theorem 4.3 that shows the fast decay of the wavelet vectors and their time derivatives, as well as a Lipschitz type bound on the function $G(t,x,y,w)$. With this result and a lemma on growth in recursions, Lemma 4.8, we can subsequently prove Theorem 4.1.

REMARK 4.1. The condition on J in (4.1) is necessary in the proof to control the effect of higher order spatial error terms in the reconstruction of the interface. These could in principle become large when the numerical time stepping is underresolved. It is, however, a mild restriction. We can take on the order $O(2^{D\Delta t^{\frac{-p}{r+1}}})$ points before violating it. Arguably the restriction is technical; we never see any problems related to it in our numerical computations.

4.1. Estimates of the exact solution. In this section we will derive estimates of the exact solutions to (2.5) together with (2.3). The estimates are based on a result from [6, 34] on the decay of the wavelet coefficients for a fixed normal multiresolution description of a smooth curve:

PROPOSITION 4.2. *Suppose $\{x_{j,k}\}$ and $\{w_{j,k}\}$ is a normal multiresolution representation of the non-selfintersecting curve $\gamma(s)$ based on the prediction (2.1). If $\gamma \in C^2([0,1];\mathbb{R}^2)$ then*

$$|w_{j,k}| \le C2^{-2j}, \qquad |x_{j,k+1} - x_{j,k}| \le C'2^{-j}, \qquad \forall k \in \bar{I}_j,$$

where the constants C and C' depend on $\gamma(s)$ but are independent of j,k.

Proof. This is essentially the result from Lemma 1 and Theorem 2 in [34]. We just need to note that the shortest distance $|x_{j,k+1} - x_{j,k}|$ is majorized by the arclength between the points, which is considered in [34]. ∎

Our main result is that the time derivative of $w_{j,k}$ satisfies the same estimate in j as the wavelet coefficient itself, and that these estimates can be extended to a finite time interval. From this we also get a Lipschitz type bound on the function $G(t,x,y,w)$. The results are summarized in the following theorem.

THEOREM 4.3. *Suppose $\{x_{j,k}\}$ and $\{w_{j,k}\}$ is a normal multiresolution representation of the non-selfintersecting curve $\gamma(s)$ based on the prediction (2.1). If $\gamma \in C^2([0,1];\mathbb{R}^2)$ and $F \in C_b^{p+1}(\mathbb{R}^+ \times \mathbb{R}^2;\mathbb{R}^2)$, then there exist unique solutions $w_{j,k}(t)$ and $x_{j,k}(t)$ in $C^{p+2}(\mathbb{R}^+)$ to (2.5) together with (2.3) for all times. For $0 \le t \le T$ and $k \in \bar{I}_j$,*

$$\left|\frac{d^\ell w_{j,k}}{dt^\ell}\right| \le C(T)2^{-2j}, \qquad 0 \le \ell \le p. \tag{4.2}$$

Moreover, if $F \in C_b^2(\mathbb{R}^+ \times \mathbb{R}^2;\mathbb{R}^2)$, then for $0 \le t \le T$, and $k \in \bar{I}_j$,

$$\begin{aligned} &|G(t,x_{j,k}+\varepsilon_0, x_{j,k+1}+\varepsilon_1, w+\varepsilon_w) - G(t,x_{j,k},x_{j,k+1},w)| \\ &\le C'(T)\left[(|w|+2^{-j})(|\varepsilon_0|+|\varepsilon_1|) + |\varepsilon_w| + (|\varepsilon_0|+|\varepsilon_1|)^2\right]. \end{aligned} \tag{4.3}$$

The constants $C(T)$ and $C'(T)$ do not depend on j, k and ℓ, but may depend on T, p, and bounds on the derivatives of F and γ.

We first present a lemma on the time derivatives of the wavelet coefficients.

LEMMA 4.4. *Suppose $F \in C_b^{p+1}(\mathbb{R}^+ \times \mathbb{R}^2;\mathbb{R}^2)$. Then there exist unique solutions $w(t)$, $x(t)$ and $y(t)$ in $C^{p+2}(\mathbb{R}^+)$ for all times to the system of ODEs*

$$\frac{dx}{dt} = F(t,x), \qquad \frac{dy}{dt} = F(t,y), \qquad \frac{dw}{dt} = G(t,x,y,w).$$

Moreover,

$$\left|\frac{d^\ell w}{dt^\ell}\right| \le C_\ell(|w| + |x-y|^2), \qquad 1 \le \ell \le p,$$

where the constant C_ℓ only depends on bounds on the derivatives of F.

Proof. We note first that by the assumptions on F, standard ODE theory ensures the stated existence, uniqueness and regularity of solutions. Next, let $F_1(t,x) = F(t,x)$ and recursively define for $\ell \ge 1$

$$F_{\ell+1}(t,x) = D_x F_\ell(t,x) \cdot F(t,x) + \partial_t F_\ell(t,x).$$

Then $F_\ell \in C_b^{2+p-\ell}(\mathbb{R}^+ \times \mathbb{R}^2;\mathbb{R}^2)$. Moreover, we claim that

$$\frac{d^\ell w}{dt^\ell} = F_\ell\left(t, \frac{x+y}{2} + w\right) - \frac{F_\ell(t,x)+F_\ell(t,y)}{2}, \qquad 1 \le \ell \le p.$$

This is true for $\ell = 1$ and, if it is true for $\ell \le r < p$, then

$$\begin{aligned}\frac{d^{r+1}w}{dt^{r+1}} &= \frac{d}{dt}\left(F_r\left(t, \frac{x+y}{2} + w\right) - \frac{F_r(t,x)+F_r(t,y)}{2}\right)\\ &= D_x F_r\left(t, \frac{x+y}{2} + w\right) \cdot \left(\frac{\frac{dx}{dt} + \frac{dy}{dt}}{2} + \frac{dw}{dt}\right) - \frac{D_x F_r(t,x)\cdot \frac{dx}{dt} + D_x F_r(t,y)\cdot\frac{dy}{dt}}{2}\\ &\quad + \partial_t F_r\left(t, \frac{x+y}{2} + w\right) - \frac{\partial_t F_r(t,x) + \partial_t F_r(t,y)}{2}.\end{aligned}$$

Using the fact that

$$\frac{\frac{dx}{dt} + \frac{dy}{dt}}{2} + \frac{dw}{dt} = \frac{F(t,x)+F(t,y)}{2} + G(t,x,y,w) = F\left(t, \frac{x+y}{2} + w\right),$$

we obtain

$$\begin{aligned}\frac{d^{r+1}w}{dt^{r+1}} &= D_x F_r\left(t, \frac{x+y}{2} + w\right) \cdot F\left(t, \frac{x+y}{2} + w\right)\\ &\quad - \frac{D_x F_r(t,x)\cdot F(t,x) + D_x F_r(t,y)\cdot F(t,y)}{2}\\ &\quad + \partial_t F_r\left(t, \frac{x+y}{2} + w\right) - \frac{\partial_t F_r(t,x) + \partial_t F_r(t,y)}{2}\\ &= F_{r+1}\left(t, \frac{x+y}{2} + w\right) - \frac{F_{r+1}(t,x)+F_{r+1}(t,y)}{2},\end{aligned}$$

which shows the claim by induction. For $1 \le \ell \le p$ we note that $F_\ell \in C_b^2(\mathbb{R}^+ \times \mathbb{R}^2;\mathbb{R}^2)$. Setting $\Delta = (x-y)/2$ we then obtain

$$\begin{aligned}\left|F_\ell\left(t, \frac{x+y}{2}\right) - \frac{F_\ell(t,x)+F_\ell(t,y)}{2}\right| &= \left|F_\ell\left(t, \frac{x+y}{2}\right) - \frac{F_\ell\left(t, \frac{x+y}{2} + \Delta\right) + F_\ell\left(t, \frac{x+y}{2} - \Delta\right)}{2}\right|\\ &\le \frac{1}{2}|\Delta|^2 |D_x^2 F_\ell(t,\cdot)|_\infty.\end{aligned}$$

This means that

$$\begin{aligned}\left|F_\ell\left(t, \frac{x+y}{2} + w\right) - \frac{F_\ell(t,x)+F_\ell(t,y)}{2}\right| &\le \left|F_\ell\left(t, \frac{x+y}{2} + w\right) - F_\ell\left(t, \frac{x+y}{2}\right)\right|\\ &\quad + \left|F_\ell\left(t, \frac{x+y}{2}\right) - \frac{F_\ell(t,x)+F_\ell(t,y)}{2}\right|\\ &\le |D_x F_\ell(t,\cdot)|_\infty |w| + \frac{1}{2}|\Delta|^2 \left|D_x^2 F_\ell(t,\cdot)\right|_\infty.\end{aligned}$$

This shows the lemma with $C_\ell = \sup_{t \ge 0} \max(\left|D_x^2 F_\ell(t,\cdot)\right|_\infty/8, |D_x F_\ell(t,\cdot)|_\infty)$. ∎

We next show the Lipschitz type bound on $G(t,x,y,w)$, which also includes a quadratic term.

Lemma 4.5. *Suppose $F\in C_b^2(\mathbb{R}^+\times\mathbb{R}^2;\mathbb{R}^2)$. Then,*

$$\begin{aligned}&|G(t,x+\varepsilon_x,y+\varepsilon_y,w+\varepsilon_w)-G(t,x,y,w)|\\ &\le C\left[(|w|+|x-y|)(|\varepsilon_x|+|\varepsilon_y|)+|\varepsilon_w|+(|\varepsilon_x|+|\varepsilon_y|)^2\right],\end{aligned}$$

where the constant C only depends on bounds on the derivatives of F.

Proof. Throughout this proof the time t is held constant; we will drop it from most of the notation and simply let $D=D_x$ denote spatial differentiation. We start by introducing $\tilde{x}(s):=x+s\varepsilon_x$, $\tilde{y}(s):=y+s\varepsilon_y$ and $\tilde{w}(s):=w+s\varepsilon_w$. Furthermore, let

$$\bar{x}(s)=\frac{\tilde{x}(s)+\tilde{y}(s)}{2},\quad \Delta_x(s)=\frac{\tilde{x}(s)-\tilde{y}(s)}{2},\quad \bar{\varepsilon}=\frac{\varepsilon_x+\varepsilon_y}{2},\quad \Delta_\varepsilon=\frac{\varepsilon_x-\varepsilon_y}{2}.$$

Then

$$\begin{aligned}&G(x+\varepsilon_x,y+\varepsilon_y,w+\varepsilon_w)-G(x,y,w)\\ &=\int_0^1\frac{d}{ds}G(\tilde{x}(s),\tilde{y}(s),\tilde{w}(0))ds+\int_0^1\frac{d}{ds}G(\tilde{x}(1),\tilde{y}(1),\tilde{w}(s))ds\\ &=\int_0^1 DF(\bar{x}(s)+w)\cdot\bar{\varepsilon}-\frac{DF(\tilde{x}(s))\cdot\varepsilon_x+DF(\tilde{y}(s))\cdot\varepsilon_y}{2}+DF(\bar{x}(1)+\tilde{w}(s))\cdot\varepsilon_w ds\\ &=\int_0^1 DF(\bar{x}(s)+w)\cdot\bar{\varepsilon}-DF(\bar{x}(s))\cdot\bar{\varepsilon}ds\\ &\quad+\int_0^1 DF(\bar{x}(s))\cdot\bar{\varepsilon}-\frac{DF(\bar{x}(s)+\Delta_x(s))+DF(\bar{x}(s)-\Delta_x(s))}{2}\cdot\bar{\varepsilon}ds\\ &\quad+\int_0^1\frac{DF(\bar{x}(s)-\Delta_x(s))-DF(\bar{x}(s)+\Delta_x(s))}{2}\cdot\Delta_\varepsilon ds\\ &\quad+\int_0^1 DF(\bar{x}(1)+\tilde{w}(s))\cdot\varepsilon_w ds\\ &=:E_1+E_2+E_3+E_4.\end{aligned}$$

We now note that when $F\in C_b^2$, then

$$|DF(x_1+\delta_1)\cdot\delta_2-DF(x_1)\cdot\delta_2|=\left|\int_0^1 D^2F(x_1+t\delta_1)(\delta_1,\delta_2)dt\right|\le\left|D^2F\right|_\infty|\delta_1||\delta_2|,$$

for any vectors x_1,δ_1,δ_2. Applying this to $E_1,\dots,E_3$ we obtain the bounds

$$|E_1|\le C_1|\bar{\varepsilon}||w|,\qquad |E_2|\le C_1\sup_{s\in[0,1]}|\bar{\varepsilon}||\Delta_x(s)|,$$

$$|E_3|\le C_1\sup_{s\in[0,1]}|\Delta_\varepsilon||\Delta_x(s)|,\qquad |E_4|\le C_2|\varepsilon_w|,$$

where $C_1=\sup_{t\ge0}\left|D_x^2F(t,\cdot)\right|_\infty$ and $C_2=\sup_{t\ge0}|D_xF(t,\cdot)|_\infty$. The result then follows from the additional estimates

$$|\Delta_\varepsilon|\le\frac{|\varepsilon_x|+|\varepsilon_y|}{2},\qquad |\bar{\varepsilon}|\le\frac{|\varepsilon_x|+|\varepsilon_y|}{2},\qquad \sup_{s\in[0,1]}|\Delta_x(s)|\le\frac{1}{2}|x-y|+|\Delta_\varepsilon|,$$

and taking $C = \max(C_1/2, C_2)$. ∎

We can now finally extend the result of Proposition 4.2 to a fixed time interval $[0,T]$.

LEMMA 4.6. *Suppose $F \in C_b^2(\mathbb{R}^+ \times \mathbb{R}^2; \mathbb{R}^2)$. Then there exist unique solutions $w(t)$, $x(t)$ and $y(t)$ in $C^3(\mathbb{R}^+)$ for all times to the system of ODEs*

$$\frac{dx}{dt} = F(t,x), \qquad \frac{dy}{dt} = F(t,y), \qquad \frac{dw}{dt} = G(t,x,y,w).$$

Moreover, there are constants $\bar{C}_1$ and $\bar{C}_2$, which only depend on bounds on the derivatives of F, such that

$$|x(t)-y(t)| \le |x(0)-y(0)| e^{t\bar{C}_1}, \qquad |w(t)| \le \left(|w(0)| + |x(0)-y(0)|^2\right) e^{t\bar{C}_2}. \tag{4.4}$$

Proof. Existence, uniqueness, and regularity follows from Lemma 4.4. For the difference $x-y$ we have

$$\frac{d|x-y|^2}{dt} = 2(x-y)^T (F(t,x) - F(t,y)) \le 2C'(t)|x-y|^2,$$

with $C' = \sup_{t \ge 0} |D_x F(t,\cdot)|_\infty$. By Grönwall's lemma,

$$|x(t)-y(t)|^2 \le |x(0)-y(0)|^2 e^{2C't}.$$

This shows the left estimate in (4.4) with $\bar{C}_1 = C'$. For $w(t)$, from Lemma 4.4,

$$\frac{d|w|^2}{dt} = 2w^T G(t,x,y,w) \le 2C_1(|w|^2 + |w||x-y|^2) \le 3C''|w|^2 + C''|x-y|^4,$$

where we can take $C'' = \max(C_1, 2C')$ with C_1 being as in Lemma 4.4. Again by Grönwall,

$$\begin{aligned}
|w(t)|^2 &\le e^{3C''t}|w(0)|^2 + C'' e^{3C''t} \int_0^t e^{-3C''s} |x(s)-y(s)|^4 ds \\
&\le e^{3C''t}|w(0)|^2 + C''|x(0)-y(0)|^4 e^{3C''t} \int_0^t e^{(4C'-3C'')s} ds \\
&\le e^{3C''t}|w(0)|^2 + C''|x(0)-y(0)|^4 e^{3C''t} \int_0^t e^{-C''s} ds \\
&= \Big(|w(0)|^2 + |x(0)-y(0)|^4 (1 - e^{-C''t})\Big) e^{3C''t}.
\end{aligned}$$

By taking square roots of both sides we obtain the right estimate in (4.4) with $\bar{C}_2 = 3C''/2$. ∎

We can now conclude the proof of Theorem 4.3. Since $F \in C_b^{p+1}(\mathbb{R}^+ \times \mathbb{R}^2; \mathbb{R}^2)$ there are unique solutions $x_{j,k}(t)$ and $w_{j,k}(t)$ in $C^{p+2}(\mathbb{R}^+)$ for all times by Lemma 4.4. From Lemma 4.6 combined with Proposition 4.2 we get the estimates

$$|w_{j,k}(t)| \le C 2^{-2j}, \qquad |x_{j,k+1}(t) - x_{j,k}(t)| \le C' 2^{-j}, \qquad \forall k \in \bar{I}_j, \quad 0 \le t \le T,$$

since $p+1 \ge 2$ and $\{x_{j,k}(0)\}$ and $\{w_{j,k}(0)\}$ are a normal multiresolution representation of the non-selfintersecting initial curve $\gamma \in C^2([0,1]; \mathbb{R}^2)$. The estimate (4.2) then follows from Lemma 4.4 and (4.3) from Lemma 4.5.

4.2. Proof of Theorem 4.1. For the remaining proof we note that Theorem 4.3 ensures the stated existence, uniqueness and regularity of solutions $x_{j,k}(t)$ and $w_{j,k}(t)$. We also introduce some additional notation. Throughout this section C will denote an arbitrary constant independent of j, k and n, although it may depend on for instance T, p as well as bounds on F and the initial curve. The error in the wavelet vectors is denoted by

$$\delta_{j,k}^n := w_{j,k}(t_n) - w_{j,k}^n, \qquad \delta_j^n = \sup_{k \in \bar{I}_j} |\delta_{j,k}^n|,$$

and we recall that the maximum error in the node points on level j is called

$$\varepsilon_j = \sup_{\substack{k \in I_j \\ 0 \le t_n \le T}} |x_{j,k}(t_n) - x_{j,k}^n|.$$

The relation between wavelet errors and node point error is given by the following lemma.

LEMMA 4.7. *Under the assumptions of Theorem 4.1 there are constants c, c' and c'', depending on T and p, but independent of j and n, such that*

$$\delta_j^{(n+1)m_j} \le (1 + c\Delta t_j)\delta_j^{nm_j} + c'\Delta t_j \left(2^{-j}\varepsilon_j + \varepsilon_j^2 + \Delta t_j^p 2^{-2j}\right), \qquad 0 \le t_{(n+1)m_j} \le T, \tag{4.5}$$

and, when $1 \le r < m_j$,

$$\delta_j^{nm_j + r} \le c''(\delta_j^{nm_j} + 2^{-j}\varepsilon_j + \varepsilon_j^2) + c'\Delta t_j^p 2^{-2j}, \qquad 0 \le t_{nm_j + r} \le T, \tag{4.6}$$

for both methods.

Proof. We use the following shorthand notation:

$$G_{j,k}(t) = G(t, x_{j,k}(t), x_{j,k+1}(t), w_{j,k}(t)), \qquad G_{j,k}^n = G(t_n, x_{j,k}^n, x_{j,k+1}^n, w_{j,k}^n).$$

Forward Euler. Suppose $1 \le r \le m_j$. For the Forward Euler scheme and the reconstruction step (3.6) we obtain

$$\begin{aligned} \delta_{j,k}^{nm_j + r} &= w_{j,k}\left(t_{nm_j} + \frac{r}{m_j}\Delta t_j\right) - w_{j,k}^{nm_j + r} \\ &= \delta_{j,k}^{nm_j} + \frac{r}{m_j}\Delta t_j \left[G_{j,k}(t_{nm_j}) - G_{j,k}^{nm_j}\right] + \tau_{j,k}^{nm_j}, \end{aligned} \tag{4.7}$$

where $\tau_{j,k}^{nm_j}$ is the local truncation error for Forward Euler. It is well-known that this can be estimated by the second order Taylor remainder term of the solution, and therefore by Theorem 4.3,

$$\sup_{\substack{k \in \bar{I}_j \\ 0 \le t_n \le T}} |\tau_{j,k}^n| \le \frac{1}{2}\left(\frac{r}{m_j}\Delta t_j\right)^2 \sup_{\substack{k \in \bar{I}_j \\ 0 \le t \le T}} \left|\frac{d^2 w_{j,k}}{dt^2}\right| \le C\left(\frac{r}{m_j}\Delta t_j\right)^2 2^{-2j}.$$

Moreover, also using the second result in Theorem 4.3 we obtain

$$\begin{aligned} &|G_{j,k}(t_{nm_j}) - G_{j,k}^{nm_j}| \\ &\le C(|w_{j,k}(t_{nm_j})| + 2^{-j})(|x_{j,k}(t_{nm_j}) - x_{j,k}^{nm_j}| + |x_{j,k+1}(t_{nm_j}) - x_{j,k+1}^{nm_j}|) \\ &\qquad + |\delta_{j,k}^{nm_j}| + (|x_{j,k}(t_{nm_j}) - x_{j,k}^{nm_j}| + |x_{j,k+1}(t_{nm_j}) - x_{j,k+1}^{nm_j}|)^2 \\ &\le C\left(2^{-j}\varepsilon_j + \delta_j^{nm_j} + \varepsilon_j^2\right). \end{aligned} \tag{4.8}$$

Taking the supremum over k in (4.7) with $r=m_j$ now gives (4.5) with $p=1$. After noting that $r\Delta t_j/m_j=r\Delta t\le T$ and that $(r\Delta t_j/m_j)^2\le\max(1,T)\Delta t_j^p$ for $p=1,2$ we also prove (4.6).

Runge-Kutta 2 . For the Runge-Kutta 2 scheme we obtain

$$\delta_{j,k}^{(n+1)m_j}=w_{j,k}(t_{nm_j}+\Delta t_j)-w_{j,k}^{(n+1)m_j}=\delta_{j,k}^{nm_j}+\Delta t_j\frac{\Delta k_1+\Delta k_2}{2}+\tau_{j,k}^{nm_j},$$

where $\tau_{j,k}^{nm_j}$ is the local truncation error for Runge-Kutta 2, and

$$\begin{aligned}\Delta k_1&=G_{j,k}(t_{nm_j})-G_{j,k}^{nm_j},\\ \Delta k_2&=G\left(t_{(n+1)m_j},x_{j,k}(t_{(n+1)m_j}),x_{j,k+1}(t_{(n+1)m_j}),w_{j,k}(t_{nm_j})+\Delta t_jG_{j,k}(t_{nm_j})\right)\\ &\quad-G\left(t_{(n+1)m_j},x_{j,k}^{(n+1)m_j},x_{j,k+1}^{(n+1)m_j},w_{j,k}^{nm_j}+\Delta t_jG_{j,k}^{nm_j}\right).\end{aligned}$$

The truncation error can be estimated as follows (see Appendix A):

$$\sup_{\substack{k\in\bar I_j\\0\le t_n\le T}}|\tau_{j,k}^n|\le\Delta t_j^3\sup_{\substack{k\in\bar I_j\\0\le t\le T}}\left(\frac{|w'''_{j,k}(t)|}{12}+\frac{|w''_{j,k}(t)|}{4}\left[\sup_{0\le t\le T}|D_wG_{j,k}(t)|_\infty\right]\right)\le C\Delta t_j^32^{-2j},$$

again using Theorem 4.3 and the boundedness of D_xF,

$$|D_wG_{j,k}(t)|=\left|D_xF\left(t,\frac{x_{j,k}(t)+x_{j,k+1}(t)}{2}+w_{j,k}(t)\right)\right|\le C.$$

For Δk_2 we have from Theorem 4.3

$$\begin{aligned}|\Delta k_2|\le C\Big[&(|w_{j,k}(t_{nm_j})+\Delta t_jG_{j,k}(t_{nm_j})|+2^{-j})\\ &\times\left(|x_{j,k}(t_{(n+1)m_j})-x_{j,k}^{(n+1)m_j}|+|x_{j,k+1}(t_{(n+1)m_j})-x_{j,k+1}^{(n+1)m_j}|\right)\\ &+|\delta_{j,k}^{nm_j}+\Delta t_j\Delta k_1|\\ &+\left(|x_{j,k}(t_{(n+1)m_j})-x_{j,k}^{(n+1)m_j}|+|x_{j,k+1}(t_{(n+1)m_j})-x_{j,k+1}^{(n+1)m_j}|\right)^2\Big]\\ \le C&\left[(|w_{j,k}(t_{nm_j})|+T|G_{j,k}(t_{nm_j})|+2^{-j})\varepsilon_j+\delta_j^{nm_j}+T|\Delta k_1|+\varepsilon_j^2\right],\end{aligned}$$

since $\Delta t_j=t_{m_j}\le T$. Now, using the fact that $|G_{j,k}(t)|=|w'_{j,k}(t)|\le C2^{-2j}$ by Theorem 4.3, we have

$$|\Delta k_2|\le C\left(2^{-j}\varepsilon_j+\delta_j^{nm_j}+\varepsilon_j^2\right)+|\Delta k_1|.$$

We already estimated $|\Delta k_1|$ in (4.8) above and the result (4.5) with $p=2$ follows. ◻

For the remaining part of the proof we need a lemma on the growth in recursions.

LEMMA 4.8. *Suppose*

$$y_{n+1}\le(1+a_n)y_n+\alpha y_n^2+b_n,\qquad n\ge0.$$

Then, if $\alpha=0$,

$$y_{n+1}\le b_n+\sum_{j=0}^{n-1}b_j\prod_{k=j+1}^n(1+a_k)+y_0\prod_{k=0}^n(1+a_k).\tag{4.9}$$

If $a_n = a$, $b_n = b$ for all n and $\alpha = 0$, then

$$y_{n+1} \le b \sum_{j=0}^{n} (1+a)^j + y_0 (1+a)^{n+1} = b \frac{(1+a)^{n+1} - 1}{a} + y_0 (1+a)^{n+1}. \qquad (4.10)$$

Suppose $\{a_n\}$ and $\{b_n\}$ are positive real numbers whose sums have a bounded growth rate,

$$A_n := \sum_{k=0}^{n} a_k \le A, \qquad B_n := \sum_{k=0}^{n} b_k \le B(n+1)^r, \qquad q \ge 0,$$

for all $n \ge 0$. Then if $\alpha > 0$

$$y_{n+1} \le e^{A+1}(B_n + y_0), \qquad 0 \le n \le \left(\frac{e^{-(A+1)}}{\alpha(B+y_0)} \right)^{\frac{1}{r+1}}. \qquad (4.11)$$

Proof. Suppose (4.9) holds for $n < p$. Then

$$y_{p+1} \le (1+a_p) y_p + b_p \le (1+a_p) \left(b_{p-1} + \sum_{j=0}^{p-2} b_j \prod_{k=j+1}^{p-1} (1+a_k) + y_0 \prod_{k=0}^{p-1} (1+a_k) \right) + b_p$$
$$= \sum_{j=0}^{p-1} b_j \prod_{k=j+1}^{p} (1+a_k) + y_0 \prod_{k=0}^{p} (1+a_k) + b_p.$$

Since (4.9) obviously holds for $n = 0$ the first result follows by induction. The second result (4.10) is a direct consequence of (4.9).

For the last estimate, we define $B_{-1} = 0$ and let $N(\beta)$, with $\beta \ge 1$, denote the largest n for which $y_j \le \beta(B_{j-1} + y_0)$ whenever $0 \le j \le n$. Then

$$y_{n+1} \le (1 + a_n + (B_{n-1} + y_0)\beta\alpha) y_n + b_n, \qquad 0 \le n \le N(\beta).$$

Consequently, from (4.9), when $0 \le n \le N(\beta)$,

$$y_{n+1} \le b_n + \sum_{j=0}^{n-1} b_j \prod_{k=j+1}^{n} (1 + a_k + (B_{k-1} + y_0)\beta\alpha) + y_0 \prod_{k=0}^{n} (1 + a_k + (B_{k-1} + y_0)\beta\alpha).$$

Since $1 + x \le e^x$,

$$y_{n+1} \le b_n + \sum_{j=0}^{n-1} b_j e^{\sum_{k=j+1}^{n} (a_k + (B_{k-1} + y_0)\beta\alpha)} + y_0 e^{\sum_{k=0}^{n} (a_k + (B_{k-1} + y_0)\beta\alpha)}$$
$$\le (B_n + y_0) e^{\sum_{k=0}^{n} (a_k + (B_{k-1} + y_0)\beta\alpha)} \le (B_n + y_0) e^{A + \beta\alpha \sum_{k=0}^{n} (B_{k-1} + y_0))}.$$

Moreover,

$$\sum_{k=0}^{n} (B_{k-1} + y_0) \le B \sum_{k=0}^{n} k^r + y_0(n+1) \le (B + y_0)(n+1)^{r+1},$$

and we finally obtain

$$y_{n+1} \le e^{A + \beta\alpha(B + y_0)(n+1)^{r+1}} (B_n + y_0).$$

Thus, by construction,

$$\beta(B_{N(\beta)}+y_0)<y_{N(\beta)+1}\le e^{A+\beta\alpha(B+y_0)(N(\beta)+1)^{r+1}}(B_{N(\beta)}+y_0).$$

Taking $\beta=\exp(A+1)$ and abbreviating $N=N(\exp(A+1))$ we obtain

$$1<(B+y_0)(N+1)^{r+1}e^{A+1}\alpha \quad\Rightarrow\quad N>\left(\frac{e^{-(A+1)}}{\alpha(B+y_0)}\right)^{\frac{1}{r+1}}-1.$$

Since, by construction, $y_n\le\exp(A+1)(B_{n-1}+y_0)$ for $0\le n\le N(\exp(A+1))$ the estimate (4.11) follows. ∎

We are now ready to finish the proof of Theorem 4.1. Taking y_n as $\delta_j^{nm_j}$ in Lemma 4.8 we obtain from (4.5), (4.10) and the fact that $\delta_{j,k}^0=0$,

$$\begin{aligned}\delta_j^{(n+1)m_j}&\le c'\Delta t_j\left(2^{-j}\varepsilon_j+\varepsilon_j^2+\Delta t_j^p2^{-2j}\right)\frac{(1+c\Delta t_j)^{n+1}-1}{c\Delta t_j}\\&\le c'\left(2^{-j}\varepsilon_j+\varepsilon_j^2+\Delta t_j^p2^{-2j}\right)\frac{e^{c\Delta t_j(n+1)}-1}{c}\\&\le C\left(2^{-j}\varepsilon_j+\varepsilon_j^2+\Delta t_j^p2^{-2j}\right),\end{aligned}$$

for $(n+1)\Delta t_j\le T$. Thus, for any $t_n:=t_{n'm_j+r}\le T$, with $0\le r<m_j$ we then get from (4.6),

$$\delta_j^n\le c''(\delta_j^{n'm_j}+2^{-j}\varepsilon_j+\varepsilon_j^2)+c'\Delta t_j^p2^{-2j}\le C\left(2^{-j}\varepsilon_j+\varepsilon_j^2+\Delta t_j^p2^{-2j}\right).$$

For odd points we have

$$\begin{aligned}&\sup_{\substack{k\in\bar I_j\\0\le t_n\le T}}|x_{j+1,2k+1}(t_n)-x_{j+1,2k+1}^n|\\&=\sup_{\substack{k\in\bar I_j\\0\le t_n\le T}}\left|\frac{x_{j,k}(t_n)-x_{j,k}^n+x_{j,k+1}(t_n)-x_{j,k+1}^n}{2}+w_{j,k}(t_n)-w_{j,k}^n\right|\\&\le\sup_{\substack{k\in\bar I_j\\0\le t_n\le T}}\frac{|x_{j,k}(t_n)-x_{j,k}^n|+|x_{j,k+1}(t_n)-x_{j,k+1}^n|}{2}+|w_{j,k}(t_n)-w_{j,k}^n|\\&\le\varepsilon_j+\sup_{0\le t_n\le T}\delta_j^n.\end{aligned}$$

Since even points are the same on consecutive levels, $x_{j+1,2k}^n=x_{j,k}^n$ for all $k\in I_j$, the same holds for the error, $\varepsilon_{j+1,2k}^n=\varepsilon_{j,k}^n$ and the above estimate therefore trivially extends to all points,

$$\varepsilon_{j+1}\le\varepsilon_j+\sup_{0\le t_n\le T}\delta_j^n\le\varepsilon_j+C\left(2^{-j}\varepsilon_j+\varepsilon_j^2+\Delta t_j^p2^{-2j}\right).$$

We can then use (4.11) in Lemma 4.8 with $a_n=C2^{-n}$, $\alpha=C$, and $b_n=C\Delta t_n^p2^{-2n}$. Since then

$$A=\sum_{n=0}^{\infty}C2^{-n}=2C.$$

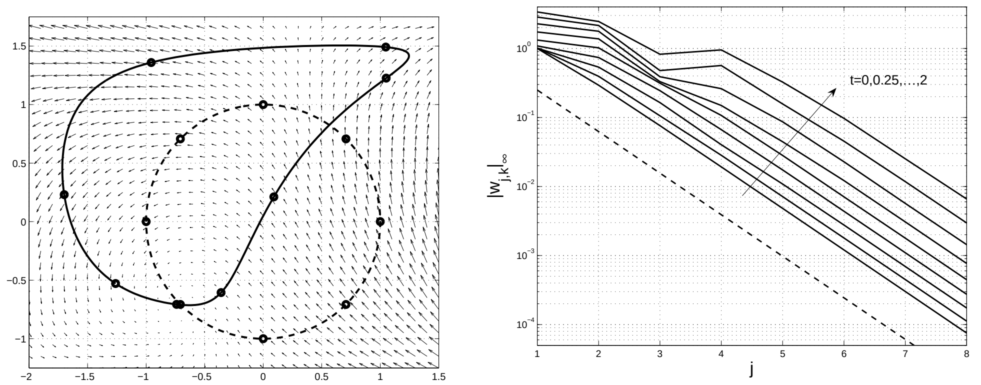


FIG. 5.1. *Left: Solution example; curve plotted at $t=0$ and $t=1$. Right: Decay of wavelet coefficients at $t=0,0.25,\dots,2$. Dashed line: 2^{-2j}.*

and

$$B_n = \sum_{k=0}^{n} C\Delta t_k^p 2^{-2k} = \Delta t^p C \sum_{k=0}^{n} m_k^p 2^{-2k} \le C\Delta t^p n^r,$$

then

$$\varepsilon_{n+1} \le e^{2C+1}\left(\Delta t^p C \sum_{k=0}^{n} m_k^p 2^{-2k} + \varepsilon_0\right), \qquad 0 \le n \le \left(\frac{e^{-(2C+1)}}{C(C\Delta t^p + \varepsilon_0)}\right)^{\frac{1}{r+1}}.$$

For both methods the edge points are solved with a standard p-th order method and therefore $\varepsilon_0 \le C\Delta t^p$. The result in the theorem then follows.

5. Numerical examples

In this section we will verify the theoretical results obtained above with a few numerical examples. We consider the following test case. We take the velocity field given by

$$F(x,y) = \begin{pmatrix} y\sin(x) - \frac{1}{2} \\ (x+0.2)\cos(y) + 0.4 \end{pmatrix}, \tag{5.1}$$

and we let the initial curve be a circle,

$$x_{j,k}(0) = \begin{pmatrix} \cos(2\pi k 2^{-j}) \\ \sin(2\pi k 2^{-j}) \end{pmatrix}, \qquad w_{j,k}(0) = \left[1 - \cos\left(2^{-j}\pi\right)\right] x_{j+1,2k+1}(0). \tag{5.2}$$

The result from a well resolved direct simulation of this problem at $t=1$ is shown in the left frame of figure 5.1, where the vector field (5.1) is overlaid. In the right frame the decay of the wavelet coefficients are plotted for various times, up to $t=2$. It confirms the theoretical decay rate of 2^{-2j} at every fixed time, and also indicates that the constant in the estimate grows with time.

All computations below were done in MATLAB with little effort spent on optimizing the implementation of the fast method. Timings shown are wall clock timings obtained by the `tic` and `toc` commands in Matlab on a standard desktop computer.

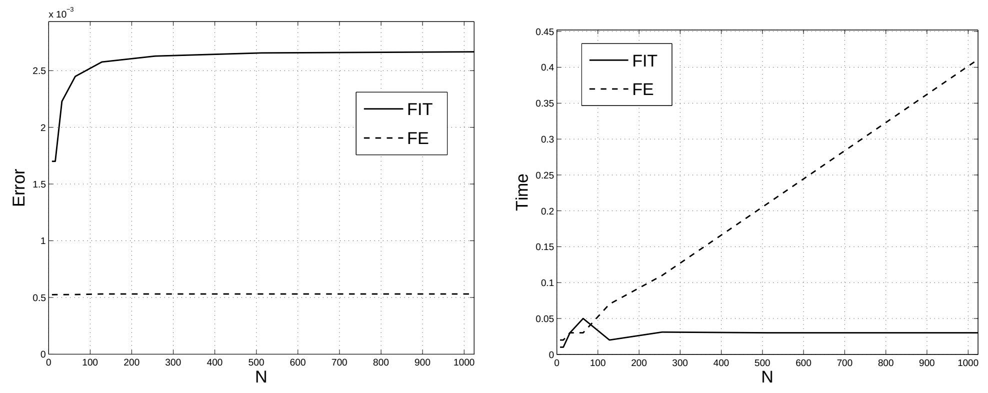


FIG. 5.2. *Error and timing for a fixed time $t=1$ and reference timestep $\Delta t = 5\cdot 10^{-4}$, doubling in each refinement, $m_j = 2^j$. Error (left) and wall clock timing (right) as a function of N, for the Fast Interface Tracking (FIT) and basic Forward Euler (FE).*

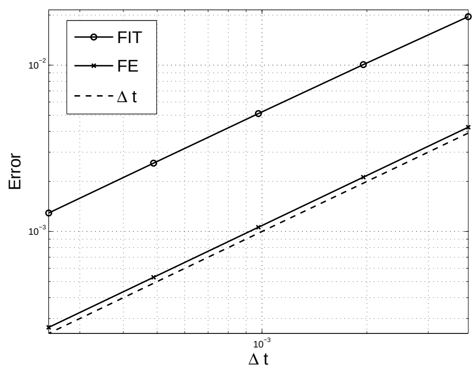


FIG. 5.3. *Error for a fixed time $t=1$ and number of points $N=512$, doubling the timestep in each refinement, $m_j = 2^j$. Error as a function of Δt, for the Fast Interface Tracking (FIT) and basic Forward Euler (FE).*

5.1. First order method. We begin to test the first order Method 1, based on Forward Euler (3.2, 3.3). In figure 5.2 we show the error and wall clock timing as a function of N, the number of marker points on the interface. The plots compare the fast method with a direct Forward Euler simulation. In the left frame of figure 5.2 one can see that the error of the fast method is around five times larger than the direct method, but importantly, it is bounded as N grows. Meanwhile, in the right frame the the timings of the methods are compared showing that the fast method has almost constant execution time while it grows linearly with N for the direct method. In figure 5.3 the first order accuracy of the fast method is confirmed for a fixed problem size $N=512$.

In figure 5.4 the error as a function of time is studied for varying N and Δt. In each frame N is constant and the result for different Δt is plotted. As expected from the theory, for sufficiently large N the fast interface tracking will generate an approximation with smaller error in shorter time compared to the direct Forward Euler method. For this example the break even point is around $N=256$. This is of

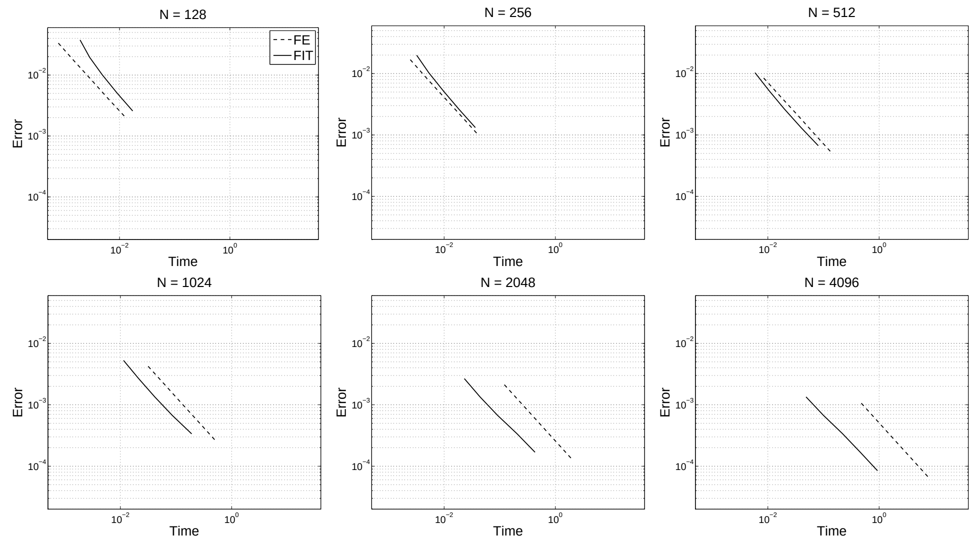


FIG. 5.4. *Error and timings for $m_j = 2^j$ with different N and Δt, comparison between Fast Interface Tracking (FIT, solid line) and Forward Euler (FE, dashed line). In each frame N is constant and the result for different Δt is plotted.*

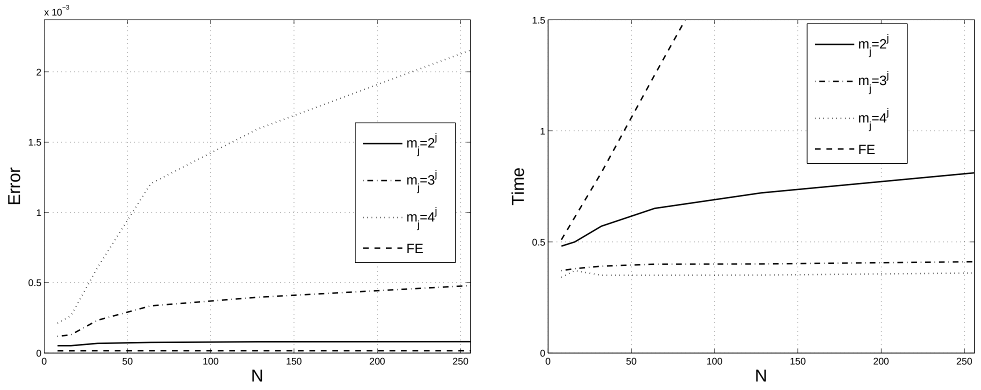


FIG. 5.5. *Error and timings for different timestep ratios m_j with fixed reference timestep* $\Delta t = 1.5 \cdot 10^{-5}$.

course a rather problem dependent figure, but still shows that the fast method can also beat standard methods for reasonably small problems, and that the constant in the complexity estimate is not prohibitively large. Finally, in figure 5.5 we examine the effect of varying the time step ratio between levels, i.e. the value of m_j. For $m_j = 3^j$, corresponding to a tripling of the time step in each level, we should in principle have an error and complexity that is independent of N, while for $m_j = 2^j$ and $m_j = 4^j$ the complexity and the error, respectively, should grow logarithmically with N. In practice these differences are hard to discern, since the effect of the variations in prefactors in front of the complexity is more significant when problems of moderate sizes are studied. In the example in figure 5.5 the choice $m_j = 2^j$ is probably the best

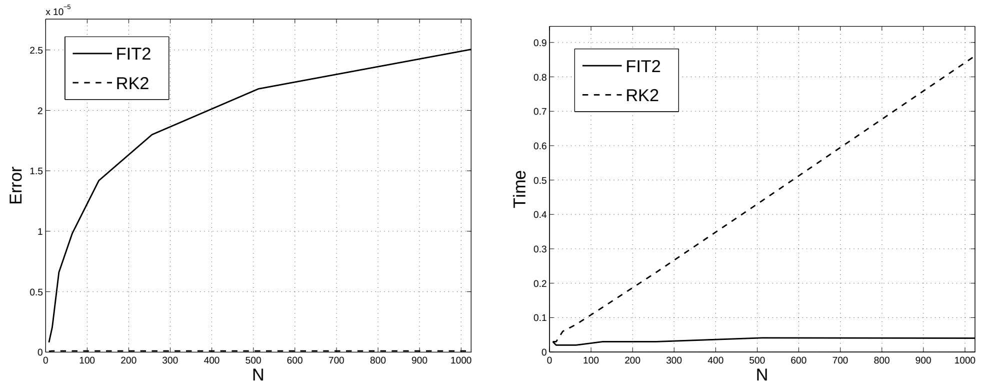


FIG. 5.6. *Error and timing for a fixed time $t=1$ and reference timestep $\Delta t=5\cdot 10^{-4}$, doubling in each refinement, $m_j=2^j$. Error (left) and wall clock timing (right) as a function of N, for the second order Fast Interface Tracking (FIT2) and basic Runge-Kutta 2 (RK2).*

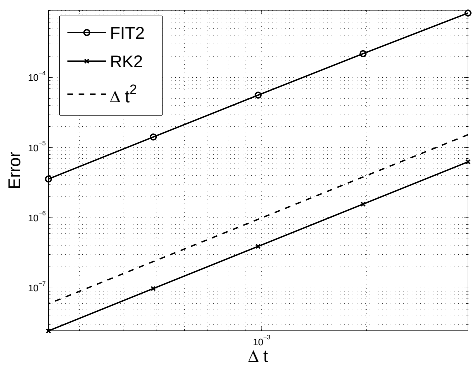


FIG. 5.7. *Error for a fixed time $t=1$ and number of points $N=512$, doubling the timestep in each refinement, $m_j=2^j$. Error as a function of Δt, for the second order Fast Interface Tracking (FIT2) and basic Runge-Kutta 2 (RK2).*

because of the small error constant, even though it is a bit slower than the choices $m_j=3^j$ and $m_j=4^j$.

5.2. Second order method. For the second order method (3.4, 3.5) we make similar experiments and compare with a direct solution with Runge-Kutta 2. The results corresponding to figure 5.2 are indicated in figure 5.6. In this case the theoretical error estimate gives $O(\Delta t^2 \log_2 N)$ and the error thus grows with N. In fact, at $N=1000$, the error is almost 100 times as large as with the direct solver. This prefactor is considerably larger than in the first order case above, where it was around five for the same problem. The execution time, on the other hand, is almost constant in N as seen in the right frame of figure 5.6. The second order accuracy is confirmed in figure 5.7. In all, the large prefactor makes this method less practical. We will get back to this problem in section 6.1 below where better high order methods are constructed.

6. Extensions

We will now consider a few extensions to the basic methods in section 3. We do not analyze these extensions in detail as in section 4, but rely on a simplified analysis like the one in section 3.1. Here we generalize the setting and assume that the estimate of the wavelet coefficients and their time derivatives satisfy $|\partial_t^p w_{j,k}| \leq C 2^{-Qj}$ (instead of 2^{-2j}) and that the number of unknowns at level j is N_j (previously $N_j = 2^j$). The finest level is denoted J and we always assume in this section that $T/\Delta t_j$ is an integer for all $j \leq J$ so that we do not need to make extra time steps of the type (3.6) at the end (thus avoiding the "$+N$" term in (3.7)). The same steps as in section 3.1 then give that

$$\text{cost} \sim \frac{1}{\Delta t} \sum_{j=0}^{J} \frac{N_j}{m_j}, \qquad \text{error} \sim \Delta t^p \sum_{j=0}^{J} m_j^p 2^{-Qj}, \tag{6.1}$$

for a p-th order time stepping scheme. While the left estimate is straightforward, rigorous justification of the right estimate for the extensions requires different techniques than in section 4. This will not be done here, but will be considered in future publications. We just note that numerical experiments confirm its validity.

6.1. Higher order methods.

Higher order methods require an improved prediction strategy compared to (2.1). Let us consider a p-th order time-stepping scheme, where we still let $N_j = 2^j$ and suppose $Q = 2$. The cost estimate (6.1) then shows that we need $m_j \geq C 2^j$ to have a significantly lower cost than $N/\Delta t$. But this implies that

$$\text{error} \geq C \Delta t^p \sum_{j=0}^{J} 2^{(p-2)j},$$

and there is therefore a "barrier" at $p = 2$ beyond which the iteration becomes unstable and the error will start to grow rapidly with N. Hence, it is apparent that the methods used so far cannot be generalized to higher order than two if the wavelet decay rate is fixed at $Q = 2$. To overcome this barrier we must have wavelets that decay faster than 2^{-2j} with $Q > 2$. Fortunately, this can be accomplished rather easily by using more general *subdivision* schemes as predictor instead of the simple averaging of neighboring points (2.1) used so far.

Subdivision is a procedure to iteratively create smooth curves and surfaces from an initial coarse mesh. Consider the sequence $\{y_{j,k}\}$, which for fixed j can be interpreted as the sample values of a piecewise linear function $y_j(t)$ on a grid of size 2^{-j}, hence $y_j(k2^{-j}) = y_{j,k}$. We introduce the subdivision operator S acting on infinite sequences, $S : \ell^\infty \to \ell^\infty$ such that

$$\{y_{j+1,k}\} = S\{y_{j,k}\}.$$

We limit ourselves here to linear interpolatory subdivision for which S is defined as follows. For even points,

$$y_{j+1,2k} = y_{j,k}$$

and for odd points,

$$y_{j+1,2k+1} = \sum_{\ell} s_\ell y_{j,k+\ell},$$

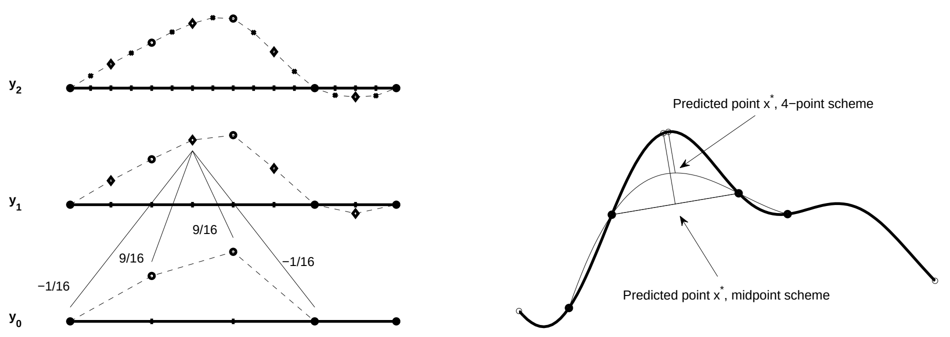

y2
y1
y0
9/16
9/16
−1/16
−1/16
Predicted point x*, 4−point scheme
Predicted point x*, midpoint scheme

FIG. 6.1. *Example of higher order subdivision: The "4-point scheme" [9] used to construct a smooth curve from a coarse mesh (left) and as predictor for a normal mesh construction (right). While midpoint subdivision corresponds to a straight line between the two closest neighboring points, the 4-point scheme corresponds to a third order polynomial interpolating the four closest neighboring points.*

where $\{s_\ell\}$ is a sequence with a finite number of non-zero entries. For well-chosen coefficients $\{s_\ell\}$ we obtain that $y_j(t)$ converges to a continuous function $y(t)$ as j grows, see e.g. [2, 5]. The procedure is exemplified in the left frame of figure 6.1. The smoothness of the limit function $y(t)$ is an important feature of the subdivision process and depends crucially on S. As an example we can consider the averaging of neighboring points we have used so far which corresponds to $\{s_\ell\}=\{1/2,\ 1/2\}$ and results in piecewise linear (Lipschitz) limit functions. This is usually called the "midpoint" or "2-point" scheme.

In our case we will use the subdivision operator as a predictor in the normal mesh procedure. We let $\{x^*_{j+1,k}\}=S\{x_{j,k}\}$ be the predicted points, replacing (2.1), and then as before define $x_{j+1,2k+1}$ to be an intersection point between the curve $\gamma(s)$ and the line passing through $x^*_{j+1,2k+1}$ that is orthogonal to the segment $(x_{j,k},x_{j,k+1})$. The wavelet vector is the difference $x_{j+1,2k+1}-x^*_{j+1,2k+1}=w_{j,k}$, and $w_{j,k}\perp x_{j,k+1}-x_{j,k}$. We still set $x_{j+1,2k}=x^*_{j+1,2k}=x_{j,k}$ for even points. This is illustrated in the right frame of figure 6.1. It is thus a generalization of the previous simpler procedure based on (2.1).

Examples of subdivision schemes include the Lagrange interpolation subdivision schemes [8], which are natural generalizations of the midpoint scheme. The $\{s_\ell\}$ sequences for these schemes are

- "4-point",
$$\{s_\ell\}=\frac{1}{16}\{-1,\ 9,\ 9,\ -1\},$$
- "6-point",
$$\{s_\ell\}=\frac{1}{256}\{3,\ -25,\ 150,\ 150,\ -25,\ 3\},$$
- "8-point",
$$\{s_\ell\}=\frac{1}{2048}\{-5,\ 49,\ -245,\ 1225,\ 1225,\ -245,\ 49,\ -5\}.$$

Prediction with higher order subdivision schemes will give faster decay of wavelet vectors. It was for instance shown in [6] that if the curve $\gamma(s)$ is in $C^{Q+\varepsilon}([0,1];\mathbb{R}^2)$

for $\varepsilon > 0$ then

$$|w_{j,k}| \le C 2^{-(Q-\varepsilon)j}, \tag{6.2}$$

where Q depends on the subdivision operator in a nontrivial way, for example

- "4-point:" $Q = 3$,
- "6-point:" $Q \approx 3.83$,
- "8-point:" $Q \approx 4.55$.

In the dynamic case we get an alternative description of the curve movement by writing down the ODEs for the wavelet vectors $\{w_{j,k}\}$ as before. We have

$$x_{j+1,2k} = x_{j,k}, \qquad x_{j+1,2k+1} = x^*_{j+1,2k+1} + w_{j,k} = \sum_\ell s_\ell x_{j,k+\ell} + w_{j,k}.$$

Consequently,

$$\frac{dw_{j,k}}{dt} = \frac{dx_{j+1,2k+1}}{dt} - \sum_\ell s_\ell \frac{dx_{j,k+\ell}}{dt} = F\left(t, \sum_\ell s_\ell x_{j,k+\ell} + w_{j,k}\right) - \sum_\ell s_\ell F(t, x_{j,k+\ell}).$$

We conjecture that, as in the midpoint subdivision case, the estimate (6.2) also holds for time derivatives of $w_{j,k}$ in a fixed time interval, and that the error estimate in (6.1) can be rigorously proved for higher order subdivision operators. The technique used in section 4 needs to be refined, however, since the max norms of those subdivision operators are typically not bounded by one, as it is for the midpoint scheme. This will be the topic of a future publication.

From this analysis we then see that the $p = 2$ barrier can be removed by taking a higher order subdivision scheme with $Q > 2$ as predictor. We exemplify the gain by repeating the numerical experiment in section 5.1 and section 5.2, with velocity field and initial curve as in (5.1) and (5.2). Now, however, we use the 4-point scheme, with $Q = 3$, instead of the midpoint scheme, with $Q = 2$. The results are indicated by FIT2–4 in figure 6.2 and figure 6.3, while the previous second order method from section 5.2 is indicated by FIT2–mid. The improvement in accuracy is substantial; the error is bounded in N and the prefactor has dropped to around ten. The method is marginally slower.

6.2. Problems with higher codimension. We can also consider the case when the interface has higher co-dimension than one. In particular we look at one-dimensional curves in three-dimensional space, so that $x(t,\cdot) : \mathbb{R} \to \mathbb{R}^3$. The approach for this case is a simple generalization of the algorithm used so far. Instead of a line normal to the vector $x_{j,k+1} - x_{j,k}$, we consider the *plane* normal to this vector. We then let $x_{j+1,2k+1}$ be the point where the normal plane that passes through the predicted point $x^*_{j+1,2k+1}$ pierces the interface. The wavelet vector is the difference $x_{j+1,2k+1} - x^*_{j+1,2k+1}$. The theory for this construction is very similar to the previous case and the proof of (6.1) can in fact be done in exactly the same way as in section 4 since the proof of Proposition 4.2 in [34] goes through almost verbatim for curves in higher dimensions as well. In particular, the wavelet decay is still 2^{-2j}, giving the same cost and accuracy formulas as before,

$$\text{cost} \sim \frac{1}{\Delta t} \sum_{j=0}^{J} \frac{2^j}{m_j}, \qquad \text{error} \sim \Delta t \sum_{j=0}^{J} m_j 2^{-2j}.$$

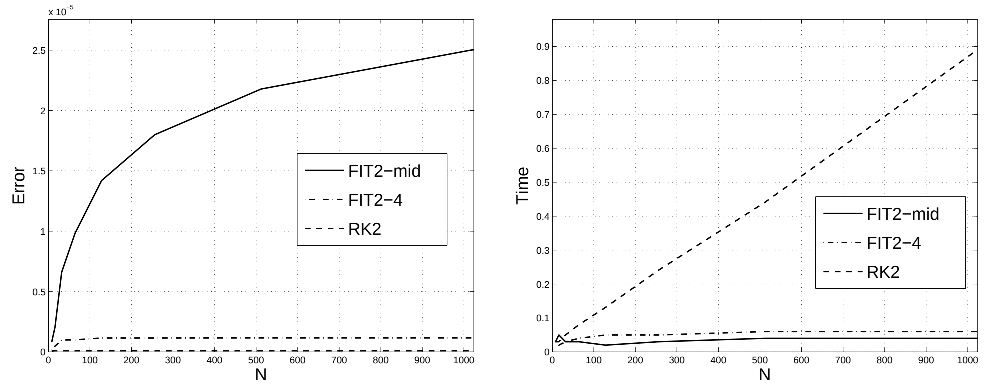


FIG. 6.2. *Error and timing for a fixed time $t=1$ and reference timestep $\Delta t = 5\cdot 10^{-4}$, doubling in each refinement, $m_j = 2^j$. Error (left) and wall clock timing (right) as a function of N, for the second order Fast Interface Tracking with midpoint prediction (FIT2–mid), with 4-point prediction (FIT2–4) and basic Runge-Kutta 2 (RK2).*

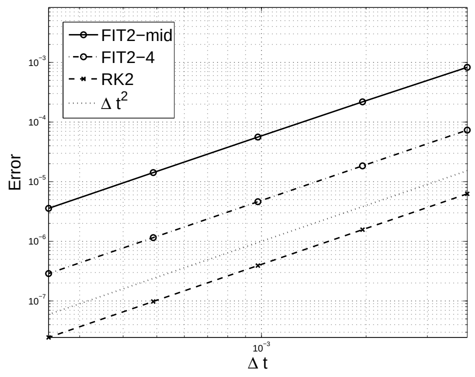


FIG. 6.3. *Error for a fixed time $t=1$ and number of points $N=512$, doubling the timestep in each refinement, $m_j = 2^j$. Error as a function of Δt, for the second order Fast Interface Tracking with midpoint prediction (FIT2–mid), with 4-point prediction (FIT2–4) and basic Runge-Kutta 2 (RK2).*

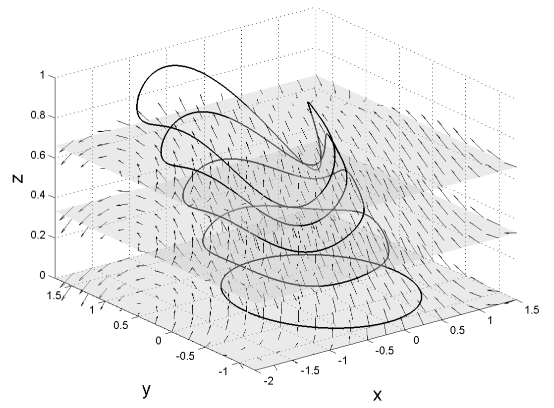
As an example we take the velocity field

$$F(x,y,z) = \begin{pmatrix} y\sin(x) - \frac{1}{2} \\ (x+0.2)\cos(y) + 0.4 \\ \cos(z+xy) \end{pmatrix}, \tag{6.3}$$

and let the initial curve be a circle in the $z=0$ plane,

$$x_{j,k}(0) = \begin{pmatrix} \cos(2\pi k 2^{-j}) \\ \sin(2\pi k 2^{-j}) \\ 0 \end{pmatrix}, \qquad w_{j,k}(0) = \left[1 - \cos\left(2^{-j}\pi\right)\right] x_{j+1,2k+1}(0). \tag{6.4}$$

The solutions at different times are plotted in the left frame and the decay of wavelet vectors are plotted in the right frame of figure 6.4, respectively. The error and wall



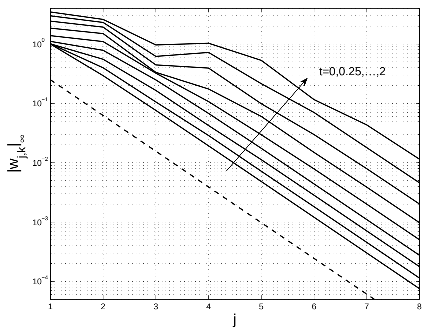
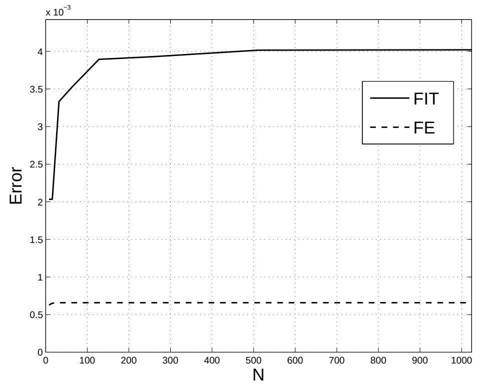


FIG. 6.4. *Left: Solution example; curve plotted at $t=0,0.25,\ldots,1$. Right: Decay of wavelet coefficients at $t=0,0.25,\ldots,2$. Dashed line: 2^{-2j}.*



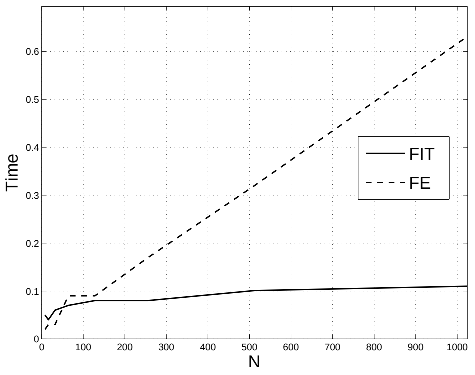


FIG. 6.5. *Error and timings for higher co-dimension problem, with same parameters as in figure 5.2.*

clock timings of Method 1 compared to direct Forward Euler behave in the same was as before. They are plotted in figure 6.5.

6.3. Two-dimensional problems. Normal meshes can be generalized to two dimensions in several ways. See [18] and [22] for some examples. Here we will approximate the surface by a triangulation and use the following simple approach based on face-splitting for refinements. To construct the normal wavelet vector and to go from one level to the next we take the steps given below, illustrated in figure 6.6.

1. Start from two adjacent triangles.
2. Construct normals to the triangles, n_1 and n_2.
3. Compute an average normal:

$$n_{\text{aver}} = \frac{n_1+n_2}{|n_1+n_2|}.$$

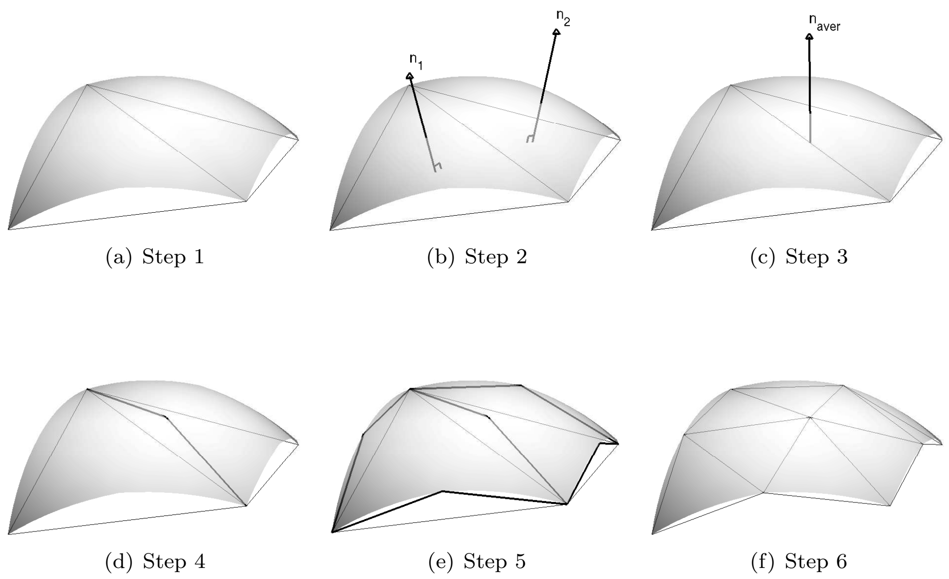


FIG. 6.6. *Steps in normal mesh construction. The surface to be approximated is indicated as a transparent sheet.*

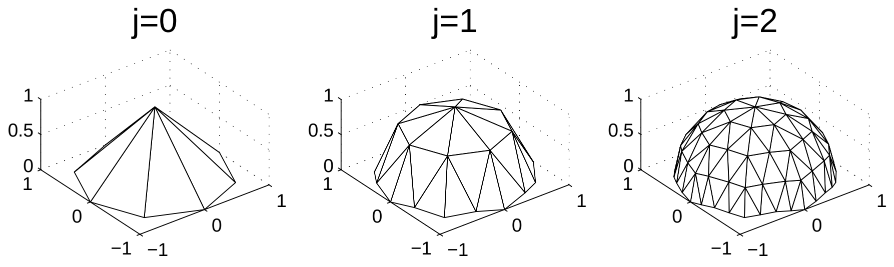


FIG. 6.7. *Example of the first levels in a normal mesh construction, with* $N_0=9$, $E_0=16$ *and* $T_0=8$.

4. Let the predicted point be the middle of the connecting edge. Apply n_{aver} there and find the point where it pierces the surface. This gives the normal wavelet vector.
5. Do the same thing for all edges.
6. Connect the new points to form a refined triangulation.

An example of the first steps in the normal construction of mesh for a half sphere is shown in figure 6.7.

There is no rigorous result available that establishes the general decay rate of the wavelet vectors for this approach. Since the predicted point is obtained in a way similar to the midpoint predictor in the one-dimensional case, one can, however, expect that the generic rate would be the same 2^{-2j} here also, if the triangulation is refined uniformly. This is also what we see in numerical experiments, c.f. the right frame of figure 6.8. Given this decay rate, the accuracy of the method would also be the same as in one dimension,

$$\text{error} \sim \Delta t \sum_{j=0}^{J} m_j 2^{-2j}.$$

As for the complexity, let the number of points, edges and triangles on level j be denoted N_j, E_j and T_j respectively. From the normal construction above it follows that in each level the number of triangles is quadrupled. For each new triangle there will be three new edges, and each old edge will be cut in two. Thus, the number of new points equals the number of old edges. We obtain the recursions

$$T_{j+1} = 4T_j, \qquad E_{j+1} = 2E_j + 3T_j, \qquad N_{j+1} = N_j + E_j.$$

Solving this system of difference equations yields

$$N_j = N_0 - E_0 + T_0 + 2^j \left(E_0 - \frac{3}{2} T_0 \right) + 4^j \frac{T_0}{2}.$$

Hence, with this normal construction the number of points N_j grows as $O\left(2^{2j}\right)$ and the computational cost is

$$\text{cost} \sim \frac{1}{\Delta t} \sum_{j=0}^{J} \frac{2^{2j}}{m_j}.$$

It is therefore more difficult to maintain a low cost than before, when we had the term 2^j instead of 2^{2j}. We need to take $m_j = 4^j$ to avoid essential growth of the computational cost with N_J. Then

$$\text{cost} \sim \frac{J}{\Delta t} \sim \frac{\log N_J}{\Delta t}.$$

However, as seen in the one-dimensional case the accuracy for this choice of m_j is rather bad compared to smaller m_j, see figure 5.5. We can still beat the $O(N_J/\Delta t)$ complexity of a direct method by taking smaller m_j, however. For instance, if we use $m_j = 2^j$ as before,

$$\text{cost} \sim \frac{1}{\Delta t} \sum_{j=0}^{J} \frac{2^{2j}}{2^j} \sim \frac{2^J}{\Delta t} \sim \frac{\sqrt{N_J}}{\Delta t}. \tag{6.5}$$

It would also be possible to construct refinement schemes with $m_j = 3^j$. That gives

$$\text{cost} \sim \frac{(4/3)^J}{\Delta t} \sim \frac{N_J^{0.21}}{\Delta t}.$$

It should be noted that, as was proposed for the one-dimensional case, the situation could be improved by using higher order interpolatory subdivision schemes like Butterfly [10, 46] for the predicted point.

For the numerical experiments, we use the same velocity field as in (6.3) and let the initial surface be a half sphere with radius one centered at the origin $x^2 + y^2 + z^2 = 1$,

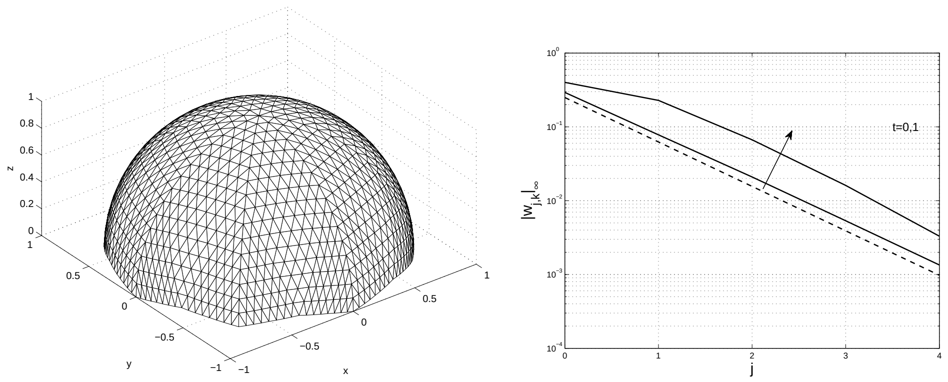


FIG. 6.8. *Left: Triangulation of initial front with $J=4$ and $N_J=1089$. Right: Decay of wavelet coefficients at $t=0,1$. Dashed line: 2^{-2j}.)*

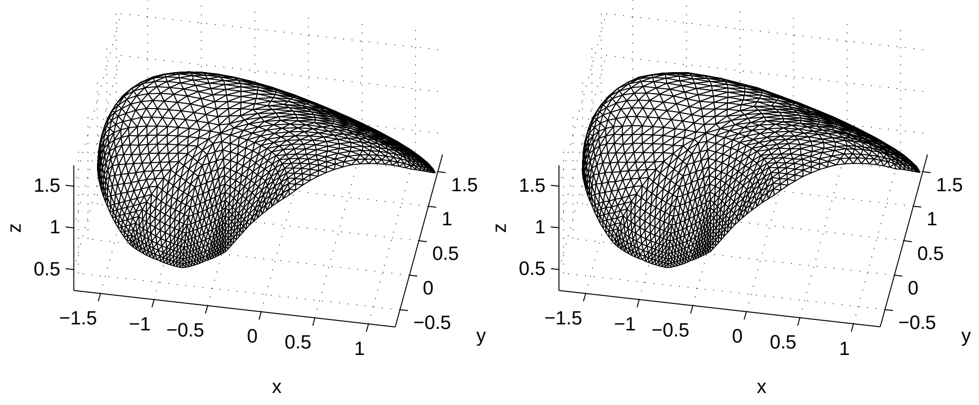


FIG. 6.9. *Solution at $t=1$ with $\Delta t=0.004$, $J=4$ and $N_J=1089$ using Forward Euler (left) and Fast Interface Tracking with $m_j=4^j$ (right).*

$z\geq 0$. The initial surface is triangulated as in figure 6.8. The solution computed by the fast method, with $m_j=4^j$, and basic Forward Euler is shown in figure 6.9. There is no discernible difference in the results. A more detailed look at the errors and timings of the method is given in figure 6.10; the error curves are similar to the one-dimensional case in figure 5.5, as expected, while the timings differ for $m_j=2^j$, where the computational cost now grows with N as predicted by (6.5). The first order accuracy of the methods for a fixed N is confirmed in figure 6.11.

7. Conclusions and open problems

We have constructed fast methods for tracking the evolution of one- and two-dimensional interfaces in a time-varying velocity field. When we are interested in the location of the interface at a fixed number of points in time, the proposed methods are much faster than standard methods, while still having the same order of accuracy. To construct higher order methods more accurate prediction schemes are needed. We have shown how this can be done in one dimension. It will be needed also in two dimensions.

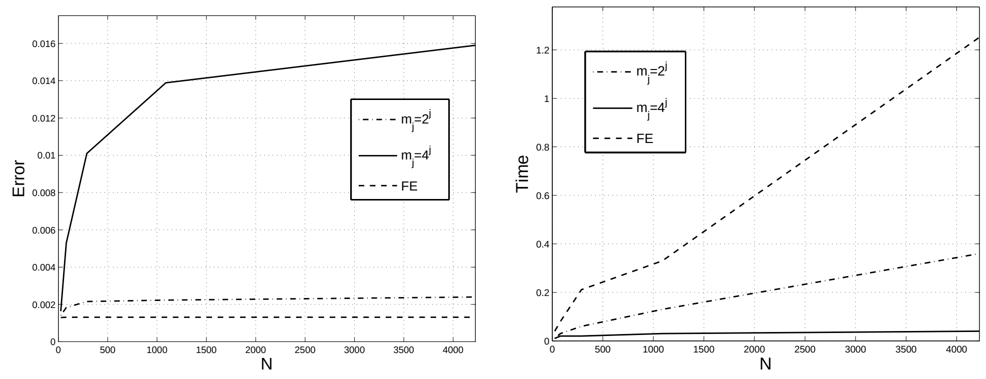


Fig. 6.10. *Error and timing for a fixed time $t=1$ and reference timestep $\Delta t=0.001$. Error (left) and wall clock timing (right) as a function of N, for the Fast Interface Tracking with doubling ($m_j=2^j$) and quadrupling ($m_j=4^j$) of time step in each level, compared with basic Forward Euler (FE).*

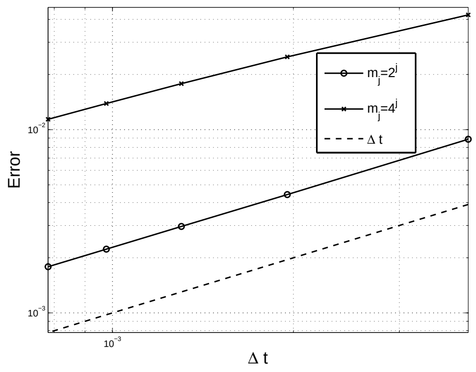


Fig. 6.11. *Error for a fixed time $t=1$ and number of points $N_J=1089$, $J=4$ Error as a function of Δt, for the Fast Interface Tracking with doubling ($m_j=2^j$) and quadrupling ($m_j=4^j$) of time step in each level, compared with basic Forward Euler (FE).*

When computing over longer times the proposed methods will need to be improved and augmented with additional steps. As in standard front tracking the representation of the interface can deteriorate since the length or area of the interface can expand quickly and the number of marker points used initially may not be enough to resolve it. An adaptive mechanism which adds and removes marker points as the resolution of the interface changes is therefore usually necessary in front tracking algorithms. Such adaptivity will also be needed for the methods in this paper. In the multiresolution setting we expect that this can be done in a natural way, given that the representation is already dealing with scales. In fact, the ability of wavelets to detect local regularity and singularities have made them particulary useful in adaptive schemes for PDEs, where better resolution in non-smooth regions is obtained by using finer scale level of the wavelets there, see e.g. [20, 4] for hyperbolic problems and [3] for elliptic problems. In the front tracking setting it is the parameterization of the curve that becomes non-smooth, not the curve itself. Finer levels would be added at the places on the interface where derivatives of $x(t,s)$ with respect to s are large. This could follow the steps of standard adaptive front tracking, adding new levels when the wavelet vectors or the distances between marker points become too large.

The multiresolution representation can also deteriorate in another way; as time evolves the wavelet vectors shift away from the locally defined normal direction and $w_{j,k}$ is not normal to the line $(x_{j,k},x_{j,k+1})$ for $t>0$, in general. Although Theorem 4.1 shows that for fixed times, the lack of orthogonality does not destroy the error estimates, clearly the quick growth in time of the wavelet decay prefactors seen in Fig. 5.1, 6.4 and 6.8 also implies growth in the error constant C' in (4.1). Eventually, for large enough times, the method will not be useful anymore. Therefore, when solving the interface tracking problem for long times, reinitialization of the mesh to a better form will be necessary at regular time intervals to bring down the size of those prefactors. For one-dimensional curves, it would not be too difficult to return to an exactly normal representation using interpolation of the fully reconstructed curve (an $O(N)$ operation). For higher dimensional manifolds this is a greater challenge, although an $O(N\log N)$ algorithm was presented in [18] for surfaces in 3D. An exactly normal representation may not be necessary, however, and one could for instance look at ideas from computer graphics, where *remeshing* is a standard problem for finding improved, but not necessarily exactly normal, representations of static meshes. See for example [30]. There is also some existing work on remeshing for deformable surfaces [25]; in fact, updating the representation continuously for a dynamically changing mesh should be somewhat easier than doing it for a static mesh. It should be noted that the deterioration of normality is a property of the continuous problem, not the discretization, so, in principle, the time between reinitializations could be chosen independent of Δt and N, and the reinitializations would then not affect the overall complexity.

It would be natural to try to apply the proposed method also for more complicated interface tracking problems, like geometric motions where the velocity of the interface also depends on the local shape of the interface, e.g. the curvature. Then F in (1.1) would be of the form $F=F(t,x,x_s,x_{ss},\cdots)$. In its present form, the method is, however, not suitable for these cases. Computing the velocity from just the wavelet coefficients and the marker points on the previous level is no longer possible. One also needs an approximation of the derivatives x_s, x_{ss}, etc. This can be obtained by reconstructing the whole interface and differentiating numerically along the interface, but then all unknowns couple and one can no longer compute level by level, which is necessary to obtain the speedup of the present method. Moreover, the special relation between time and space scales, which is the fundamental reason why the method works, may be different for these problems, i.e., Theorem 4.3 may no longer be true. Thus, currently the method is primarily for passive transport and will not work for geometric motion unless some new ideas come up.

Appendix A. Local truncation error for Runge-Kutta 2. The local truncation errors for Runge-Kutta schemes are not as straightforward to derive as for multistep methods. For completeness we therefore show here the error estimate used in section 4 for Runge-Kutta 2. Let $y(t)\in C^3(\mathbb{R}^+)$ be the exact solution satisfying $y'=f(t,y)\in C_b^2(\mathbb{R}^+,\mathbb{R}^2)$. The local truncation error τ^n is defined as the residual when the exact solution is entered into the scheme, hence

$$\tau^n=y(t_n)+\frac{\Delta t}{2}\left[f\Big(t_n,y(t_n)\Big)+f\Big(t_{n+1},y(t_n)+\Delta t f(t_n,y(t_n))\Big)\right]-y(t_{n+1}).$$

By using the ODE we obtain

$$\begin{aligned}\tau^n = & y(t_n)+\frac{\Delta t}{2}[y'(t_n)+y'(t_{n+1})]-y(t_{n+1})\\ & +\frac{\Delta t}{2}\Big[f\Big(t_{n+1},y(t_n)+\Delta t y'(t_n)\Big)-f\Big(t_{n+1},y(t_{n+1})\Big)\Big].\end{aligned}$$

The error has two parts which can be rewritten as follows. After expanding y and y' around t_n we get from Taylor's formula with an integral remainder term that

$$E_1 := y(t_n)+\frac{\Delta t}{2}[y'(t_n)+y'(t_{n+1})]-y(t_{n+1}) = \frac{\Delta t^3}{2}\int_0^1 s(1-s)y'''(t_n+s\Delta t)ds.$$

Moreover,

$$\begin{aligned}E_2 := & \frac{\Delta t}{2}\Big[f\Big(t_{n+1},y(t_n)+\Delta t y'(t_n)\Big)-f\Big(t_{n+1},y(t_{n+1})\Big)\Big]\\ = & \frac{\Delta t}{2}\int_0^1 D_y f\Big(t_{n+1},\bar{y}(s))\Big)\Big(y(t_n)+\Delta t y'(t_n)-y(t_{n+1})\Big)ds,\end{aligned}$$

where $\bar{y}(s) := s(y(t_n)+\Delta t y'(t_n))+(1-s)y(t_{n+1})$. Hence,

$$\begin{aligned}|\tau^n| \le |E_1|+|E_2| \le & \frac{\Delta t^3}{2}\int_0^1 |s(1-s)|ds \sup_{t_n\le t\le t_{n+1}}|y'''(t)|\\ & +\frac{\Delta t}{2}|D_y f(t_{n+1},\cdot)|_\infty |y(t_n)+\Delta t y'(t_n)-y(t_{n+1})|\\ \le & \frac{\Delta t^3}{12}\sup_{t_n\le t\le t_{n+1}}|y'''(t)|+\frac{\Delta t^3}{4}|D_y f(t_{n+1},\cdot)|_\infty \sup_{t_n\le t\le t_{n+1}}|y''(t)|.\end{aligned}$$

REFERENCES

[1] H.J. Bungartz and M. Griebel, *Sparse grids*, Acta Numerica, 13, 1–121, 2004.

[2] A.S. Cavaretta, W. Dahmen and C.A. Micchelli, *Stationary subdivision*, Memoirs Amer. Math. Soc., 93(453), 1991.

[3] A. Cohen, W. Dahmen and R.A. DeVore, *Adaptive wavelet methods for elliptic operator equations: convergence rates*, Math. Comp., 70, 27–75, 2001.

[4] A. Cohen, S. M Kaber, S. Müller and M. Postel, *Fully adaptive multiresolution finite volume schemes for conservation laws*, Math. Comp., 72(241), 183–225, 2003.

[5] I. Daubechies and J.C. Lagarias, *Two-scale difference equations I. Existence and global regularity of solutions*, SIAM J. Math. Anal., 22(5), 1388–1410, 1991.

[6] I. Daubechies, O. Runborg and W. Sweldens, *Normal multiresolution approximation of curves*, Constr. Approx., 20, 399–463, 2004.

[7] L. Demanet and L. Ying, *Wave atoms and time upscaling of wave equations*, preprint, 2007.

[8] G. Deslauriers and S. Dubuc, *Symmetric iterative interpolation processes*, Constr. Approx., 5(1), 49–68, 1989.

[9] N. Dyn, D. Levin and J. Gregory, *A 4-point interpolatory subdivision scheme for curve design*, Comput. Aided Geom. Des., 4, 257–268, 1987.

[10] N. Dyn, D. Levin and J. Gregory, *A butterfly subdivision scheme for surface interpolation with tension control*, ACM Trans. on Graphics, 9(2), 160–169, 1990.

[11] B. Engquist, S. Osher and S. Zhong, *Fast wavelet based algorithms for linear evolution equations*, SIAM J. Sci. Comput., 15(4), 755–775, 1994.

[12] B. Engquist, O. Runborg and A.K. Tornberg, *High frequency wave propagation by the segment projection method*, J. Comput. Phys., 178, 373–390, 2002.

[13] M.B. Giles, *Multilevel Monte–Carlo path simulation*, Oper. Res., 56(3), 607–617, 2008.

[14] J. Glimm, J.W. Grove, X.L. Li, K.M. Shyue, Y. Zeng and Q. Zhang, *Three-dimensional front tracking*, SIAM J. Sci. Comput., 19(3), 703–727, 1998.

[15] J. Glimm, E. Isaacson, D. Marchesin and O. McBryan, *Front tracking for hyperbolic systems*, Adv. Appl. Math, 2, 91–119, 1981.

[16] M. Griebel and D. Oeltz, *A sparse grid space-time discretization scheme for parabolic problems*, Computing, 81(1), 1–34, 2007.

[17] J. Guckenheimer and A. Vladimirsky, *A fast method for approximating invariant manifolds*, SIAM J. Appl. Dyn. Syst., 3(3), 232–260, 2004.

[18] I. Guskov, K. Vidimce, W. Sweldens and P. Schröder, *Normal meshes*, Computer Graphics (SIGGRAPH '00 Proceedings), 259–268, 2000.

[19] H. Han, M. Ehrhardt and C. Zheng, *Numerical simulation of waves in periodic structures*, Comm. Comp. Phys., 2008.

[20] A. Harten, *Adaptive multiresolution schemes for shock computations*, J. Comput. Phys., 115(2), 319–338, 1994.

[21] C.W. Hirt and B.D. Nichols, *Volume of fluid (VOF) method for the dynamics of free boundaries*, J. Comput. Phys., 39, 201–225, 1981.

[22] M. Jansen, H. Choi, S. Lavu and R. Baraniuk, *Multiscale image processing using normal triangulated meshes*, IEEE International Conference on Image Processing, Thessaloniki, Greece, 2, 229–232, 2001.

[23] A. Khodakovsky and I. Guskov, *Compression of Normal Meshes*, Geometric Modeling for Scientific Visualization, Springer Verlag, 2004.

[24] A. Khodakovsky, P. Schröder and W. Sweldens, *Progressive geometry compression*, Computer Graphics (SIGGRAPH '00 Proceedings), 271–278, 2000.

[25] S. Kircher and M. Garland, *Progressive multiresolution meshes for deforming surfaces*, Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation, 191–200, 2005.

[26] B. Krauskopf, H.M. Osinga, E.J. Doedel, M.E. Henderson, J. Guckenheimer, A. Vladimirsky, M. Dellnitz and O. Junge, *A survey of methods for computing (un)stable manifolds of vector fields*, Int. J. Bifurcat. Chaos, 15, 763–792, 2005.

[27] G. Lambaré, P.S. Lucio and A. Hanyga, *Two-dimensional multivalued traveltime and amplitude maps by uniform sampling of ray field*, Geophys. J. Int., 125, 584–598, 1996.

[28] B. Lastdrager, B. Koren and J. Verwer, *The sparse-grid combination technique applied to time-dependent advection problems*, Appl. Numer. Math., 38(4), 377–401, 2001.

[29] S. Lavu, H. Choi and R. Baraniuk, *Geometry Compression of Normal Meshes using Rate-Distortion Algorithms*, Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, Aachen, Germany, 52–61, 2003.

[30] A. Lee, W. Sweldens, P. Schröder, L. Cowsar and D. Dobkin, *MAPS: Multiresolution Adaptive Parametrization of Surfaces*, Computer Graphics (SIGGRAPH '98 Proceedings), 95–104, 1998.

[31] S.J. Osher, L.T. Cheng, M. Kang, H. Shim and Y.H. Tsai, *Geometric optics in a phase-space-based level set and Eulerian framework*, J. Comput. Phys., 179(2), 622–648, 2002.

[32] S.J. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79(1), 12–49, 1988.

[33] D. Peng, B. Merriman, S. Osher, H. Zhao and M. Kang, *A PDE based fast local level set method*, J. Comput. Phys., 155(2), 410–438, 1999.

[34] O. Runborg, *Introduction to normal multiresolution analysis*, B. Engquist, P. Lötstedt and O. Runborg (eds), Multiscale Methods in Science and Engineering, Lecture Notes in Computational Science and Engineering, Heidelberg, Springer Verlag, 44, 205–224, 2005.

[35] C.C. Stolk, *A fast method for linear waves based on geometrical optics*, preprint, 2007.

[36] J. Strain, *Tree methods for moving interfaces*, J. Comput. Phys., 151(2), 616–648, 1999.

[37] J. Strain, *A fast modular semi-Lagrangian method for moving interfaces*, J. Comput. Phys., 161(2), 512–536, 2000.

[38] W. Sweldens, *The lifting scheme: a construction of second generation wavelets*, SIAM J. Math. Anal., 29(2), 511–546, 1997.

[39] A.K. Tornberg and B. Engquist, *The segment projection method for interface tracking*, Comm. Pure Appl. Math., 56(1), 47–79, 2003.

[40] S.O. Unverdi and G. Tryggvason, *A front tracking method for viscous, incompressible, multi-fluid flows*, J. Comput. Phys., 100, 25–37, 1992.

[41] V. Vinje, E. Iversen and H. Gjøystdal, *Traveltime and amplitude estimation using wavefront construction*, Geophysics, 58(8), 1157–1166, 1993.

[42] J.E. Welch, F.W. Harlow, J.P. Shannon and B.J. Daly, *The MAC method: a computing technique for solving viscous, incompressible, transient fluid flow problems involving free sur-*

faces, Los Alalmos Scientific Laboratory Report LA, 3425, 1966.

[43] L. Ying and E. Candes, *The phase flow method*, J. Comput. Phys., 220(1), 184–215, 2006.

[44] L. Yuan and Y.Y. Lu, *A recursive doubling Dirichlet-to-Neumann map method for periodic waveguides*, J. Lightwave Technol., 25, 3649–3656, 2007.

[45] C. Zenger, *Sparse grids*, W. Hackbusch, (ed.), Parallel Algorithms for Partial Differential Equations, Notes on Numerical Fluid Mechanics, Vieweg, 31, 241–251, 1991.

[46] D. Zorin, P.Schröder and W. Sweldens, *Interpolating subdivision for meshes with arbitrary topology*, Computer Graphics (SIGGRAPH '96 Proceedings), 189–192, 1996.