Journal of Computational Mathematics
Vol.33, No.6, 2015, 576–586.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1503-m4532

AN ADAPTIVE FAST INTERFACE TRACKING METHOD*

Yana Di
LSEC, NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China
Email: yndi@lsec.cc.ac.cn
Jelena Popovic
Department of Numerical Analysis, CSC, KTH, 100 44 Stockholm, Sweden
Email: jelenap@csc.kth.se
Olof Runborg
Department of Mathematics and Swedish e-Science Research Center (SeRC), KTH, 100 44 Stockholm, Sweden
Email: olofr@nada.kth.se

Abstract

An adaptive numerical scheme is developed for the propagation of an interface in a velocity field based on the fast interface tracking method proposed in [2]. A multiresolution stategy to represent the interface instead of point values, allows local grid refinement while controlling the approximation error on the interface. For time integration, we use an explicit Runge-Kutta scheme of second-order with a multiscale time step, which takes longer time steps for finer spatial scales. The implementation of the algorithm uses a dynamic tree data structure to represent data in the computer memory. We briefly review first the main algorithm, describe the essential data structures, highlight the adaptive scheme, and illustrate the computational efficiency by some numerical examples.

Mathematics subject classification: 42C40, 65D10.
Key words: Interface tracking, Multiresolution, adaptivity, Fast algorithms.

1. Introduction

Tracking the evolution of interfaces or fronts is important in many application, 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 $\boldsymbol{x}(t,s) : \mathbb{R}^+ \times \mathbb{R}^q \to \mathbb{R}^d$, 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 front itself. Then $\boldsymbol{x}(t,s)$ satisfies the parameterized ordinary differential equation (ODE)

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

where $F(t,\boldsymbol{x}) : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d$ 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 treat curves in two dimensions, $d = 2$,

* Received June 3, 2014 / Revised version received March 20, 2015 / Accepted March 26, 2015 / Published online November 17, 2015 /

$q = 1$, but 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 (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 PDEs. In many of these problems it is better to numerically 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. Numerical methods for this problem include the Lagrangian front tracking method [4]. There are also Eulerian approaches like the level set method [5] and segment projection [6]. For flow problems we should also mention the marker-and-cell (MAC) [7] and volume of fluid (VOF) [8] 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 $\boldsymbol{x}_j(t) \approx \boldsymbol{x}(t, s_j)$ and use a numerical method for ODEs to solve

$$\frac{\mathrm{d}\boldsymbol{x}_j(t)}{\mathrm{d}t} = F(t, \boldsymbol{x}_j(t)), \qquad \boldsymbol{x}_j(0) = \gamma(s_j), \tag{1.2}$$

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 time step length 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 $O(N/\Delta t)$. In [1–3], wavelet vectors were used to describe the interface, which correspond to the details of the interface on different scale levels. It was shown that the time derivatives of the wavelet vectors, just as the wavelet vectors themselves, decay exponentially with level of detail. By taking multiscale time steps, i.e. longer time steps for the fine scales than for the coarse scales, the computational cost is reduced to only $O(\log N/\Delta t)$ or even $O(1/\Delta t)$ without affecting the overall accuracy. We should emphasize that this is different from standard wavelet based adaptive schemes where shorter time steps are often used for the fine details, which is the opposite of the method in [2]. With such strategy the cost will be reduced, but it will only be the constant in the complexity estimate that is improved; the complexity itself remains the same order. The reason is that there are comparatively few coarse scale wavelet vectors, where efficiency improvement is achieved, and many fine scale wavelet vectors, where there is little gain.

Adaptivity is usually an important feature of front tracking algorithms. Since the length or area of the interface can grow 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 becomes necessary. For multiresolution methods, an advantage to define adaptive techniques is an efficient data representation with an accurate estimation of the local approximation error. Based on the details, or wavelet coefficients, between two consecutive grid-refinement levels, multiresolution methods provide a rigorous regularity analysis [14], while for adaptive mesh refinement methods rigorous error estimators are quite difficult to be derived. In the past, adaptive wavelet-based multiresolution methods have been introduced to improve the computational efficiency and to reduce the memory requirement of the algorithms, e.g., [9–13]. According to error estimates from different resolution levels, numerical schemes have been developed for adjusting grid resolution locally and dynamically. To obtain additional speed-up, space-time adaptive methods [15] are introduced, where the size of

each cell and its time step are adjusted dynamically.

In the current paper we develop an adaptive multiresolution scheme with multi-scale time stepping for interface tracking. Our starting point is the fast interface tracking method proposed in [2,3], which reduced the computational cost using the multi-scale time steps, and the adaptive versions in [1]. We employ a tree data structure to achieve high rates of data compression. To be able to track the interface as accurately as possible, we update the adaptive data structure by wavelet coefficients and arc lengths to ensure all elements are below the reference tolerance. A standard multiresolution analysis is performed successively from the coarsest to the finest level to complete the local approximation error control. As a result, the adaptive fast interface tracking method can adjust the mesh dynamically as time evolves and improve computational efficiency significantly.

The remainder of this article is organized as follows. In Section 2 the multiresolution representation of the interface is presented and the governing ODEs are derived. In Section 3 the fully adaptive fast interface tracking scheme is described. Details on the data structure and the implementation are also given. Numerical results to demonstrate the accuracy of the algorithm and its efficiency are in Section 4. Finally, we conclude and present some perspectives for future work.

2. Multiresolution Description of the Interface Propagation

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 mesh* is 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 curve or surface smoothness most wavelet vectors will be small, lending the representation well to compression.

In our case the curve $\boldsymbol{x}(t,s)$ with $0 \le s \le 1$, will be described as follows. We introduce the parameter indices $s_{j,k} = k2^{-j}$ and define

$$\boldsymbol{x}_{j,k}(t) := \boldsymbol{x}(t, k2^{-j}), \qquad 0 \le k \le 2^j. \tag{2.1}$$

Note that $\boldsymbol{x}_{j+1,2k} = \boldsymbol{x}_{j,k}$ and that for a fixed j the markers $\{\boldsymbol{x}_{j,k}\}$ will be a discretization of the interface with a level of detail that increases with the fixed j-value. We assume that we start from a given fine discretization with 2^J points on the interface, which thus corresponds to level J. We let $\boldsymbol{x}_j(t) = \{\boldsymbol{x}_{j,k}(t)\}_{k=0}^{2^j-1}$ and next define the wavelet vectors

$$\boldsymbol{w}_{j+1}(t) = \boldsymbol{x}_{j+1}(t) - S\boldsymbol{x}_j(t), \tag{2.2}$$

where S is the so-called subdivision scheme. A special example of a subdivision scheme is the midpoint interpolating scheme where $(S\boldsymbol{x}_j)_{2k+1} = (\boldsymbol{x}_{j,k} + \boldsymbol{x}_{j,k+1})/2$ with second order. This is done recursively and gives an alternative description of the interface in terms of $\boldsymbol{x}_0(t)$ together with $\boldsymbol{w}_j(t)$ for $j = 1, \ldots, J$. The wavelet sequences $\boldsymbol{w}_j(t) := \{\boldsymbol{w}_{j,k}(t)\}_{k=0}^{2^j-1}$ can be computed from the original discretization $\boldsymbol{x}_J(t)$ at an $O(N)$ cost, where $N = 2^J$ is the number of discretization points. Similarly, with an inverse wavelet transform based on reversing the recursion in (2.2), the points $\boldsymbol{x}_J(t)$ can be computed from $\{\boldsymbol{w}_j\}$ and $\boldsymbol{x}_0$ at an $O(N)$ cost. Moreover, for smooth $\boldsymbol{x}(t,s)$ the wavelets decay exponentially in j with a rate determined

by the order of the subdivision scheme S. The fast decay of the wavelet vectors gives the representation good compression properties, and as was shown in [2, 3] it also allows us to construct a fast interface tracking algorithm.

For the dynamic case we insert (2.2) in (1.1) and get

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{w}_{j+1}}{\mathrm{d}t} =& F(t, \boldsymbol{x}_{j+1}(t)) - SF(t, \boldsymbol{x}_j(t)) \\ =& F(t, S\boldsymbol{x}_j(t) + \boldsymbol{w}_{j+1}(t)) - SF(t, \boldsymbol{x}_j(t)).\end{aligned} \tag{2.3}$$

Setting

$$G(t, \boldsymbol{x}, \boldsymbol{w}) = F(t, S\boldsymbol{x} + \boldsymbol{w}) - SF(t, \boldsymbol{x}), \tag{2.4}$$

we thus have the following alternative system of ODEs

$$\frac{\mathrm{d}\boldsymbol{w}_{j+1}(t)}{\mathrm{d}t} = G(t, \boldsymbol{x}_j(t), \boldsymbol{w}_{j+1}(t)), \qquad \frac{\mathrm{d}\boldsymbol{x}_0(t)}{\mathrm{d}t} = F(t, \boldsymbol{x}_0(t)), \tag{2.5}$$

which together with (2.2) describe the dynamics of the system.

Remark 2.1. Fine scales depend on coarser scales, but there is no dependence in the other direction. This makes us able to compute the different scale levels sequentially from coarse to fine, one after the other. This is the same idea that is used in the inverse wavelet transform.

3. Numerical Method

In this section we describe the adaptive fast interface tracking scheme, i.e. we describe the numerical method for solving (2.5) together with (2.2).

3.1. Time integration

Due to the adaptive space discretization, the mesh is changing in time, and therefore we first discretize in time and then in space. At different refinement levels j we will use different time steps, denoted by $\Delta t_j := m_j \Delta t$, where $m_j \in \mathbb{Z}^+$ and $m_0 = 1$. Typically we double the time step in each level, and hence have $m_j = 2^j$. The motivation is shown in [2] that if the initial curve is described by a normal approximation, then also the time derivatives of the wavelet vectors $\boldsymbol{w}_{j,k}$ decay as 2^{-2j} over a fixed time interval. An interpretation 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 $\boldsymbol{w}_{j,k}$ is much smaller at finer than at coarser levels, we will use longer time steps for the fine scales. Here we use an explicit second-order accurate Runge-Kutta (RK2) scheme. We denote the numerical approximations to be

$$\boldsymbol{x}_{j,k}^n \approx \boldsymbol{x}_{j,k}(t^n), \qquad \boldsymbol{w}_{j,k}^n \approx \boldsymbol{w}_{j,k}(t^n), \qquad t^n = n\Delta t,$$

where Δt is the reference time step. Since we are interested in computing the solution up to time T we also define integer M such that $T = M\Delta t$. Let I_j be the index set $0, \ldots, 2^j$ and $\bar{I}_j$ the index set $0, \ldots, 2^j - 1$, then $\boldsymbol{x}_{j,k}$ is defined for all $k \in I_j$ and $\boldsymbol{w}_{j,k}$ for all $k \in \bar{I}_j$. Given the initial values

$$\{\boldsymbol{x}_{0,k}^0\}, \;\; k \in I_0, \qquad \{\boldsymbol{w}_{j,k}^0\}, \;\; j \in \{0, \ldots, J\}, \;\; k \in \bar{I}_j,$$

where $J > 0$ is the highest level, the RK2 scheme used here has the form at level $j > 0$,

$$k_1 = G(t_{nm_j}, \boldsymbol{x}_{j,k}^{nm_j}, \boldsymbol{x}_{j,k+1}^{nm_j}, \boldsymbol{w}_{j,k}^{nm_j}), \tag{3.1}$$

$$k_2 = G(t_{(n+1)m_j}, \boldsymbol{x}_{j,k}^{(n+1)m_j}, \boldsymbol{x}_{j,k+1}^{(n+1)m_j}, \boldsymbol{w}_{j,k}^{nm_j} + \Delta t_j k_1), \tag{3.2}$$

$$\boldsymbol{w}_{j,k}^{(n+1)m_j} = \boldsymbol{w}_{j,k}^{nm_j} + \frac{\Delta t_j}{2}(k_1 + k_2), \tag{3.3}$$

where $x_{j,k}$ are computed from results at the coarser level,

$$\boldsymbol{x}_{j+1,2k+1}^n = \frac{\boldsymbol{x}_{j,k}^n + \boldsymbol{x}_{j,k+1}^n}{2} + \boldsymbol{w}_{j,k}^n, k \in \bar{I}_j, \quad \boldsymbol{x}_{j+1,2k}^n = \boldsymbol{x}_{j,k}^n, k \in I_j. \tag{3.4}$$

On the zeroth level,

$$k_1 = F(t_n, \boldsymbol{x}_{0,0}^n), \qquad k_1 = F(t_n, \boldsymbol{x}_{0,1}^n), \tag{3.5a}$$

$$k_2 = F(t_{n+1}, \boldsymbol{x}_{0,0}^n + \Delta t k_1), \qquad k_2 = F(t_{n+1}, \boldsymbol{x}_{0,1}^n + \Delta t k_1), \tag{3.5b}$$

$$\boldsymbol{x}_{0,0}^{n+1} = \boldsymbol{x}_{0,0}^n + \frac{\Delta t}{2}(k_1 + k_2), \qquad \boldsymbol{x}_{0,1}^{n+1} = \boldsymbol{x}_{0,1}^n + \frac{\Delta t}{2}(k_1 + k_2). \tag{3.5c}$$

3.2. Error estimate

Estimates of the local truncation error are used to mark those elements with unacceptably large errors for refinement. 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,

$$\tau_{j,k}^n \sim \Delta t_j^{p+1} \frac{\mathrm{d}^{p+1} \boldsymbol{w}_{j,k}}{\mathrm{d}t^{p+1}} = \Delta t^{p+1} \frac{\mathrm{d}^{p+1} \boldsymbol{w}_{j,k}}{\mathrm{d}t^{p+1}} 2^j. \tag{3.6}$$

The *goals* for our adaptivity is to make all local truncation errors under control, precisely, smaller than a prescribed tolerance on the finest level. Runborg [2] showed that, for smooth enough velocity field $F \in C_b^{p+1}(\mathbb{R}^+ \times \mathbb{R}^2; \mathbb{R}^2)$, the time derivatives of the wavelet coefficients satisfy

$$\left| \frac{\mathrm{d}^l \boldsymbol{w}_{j,k}}{\mathrm{d}t^l} \right| \le C_l(|\boldsymbol{w}_{j,k}| + |\boldsymbol{x}_{j,k} - \boldsymbol{x}_{j,k+1}|^2), \qquad 1 \le l \le p+1, \tag{3.7}$$

where the constant C_l only depends on the bound of the derivatives of F. Additionally, from the definition of the wavelet vector in (2.2), the size of the wavelet vector $|\boldsymbol{w}_j(t)|$ indicates the the error of the interpolation precisely.

Hence, the indicator, or estimator for the adaptive scheme will be set to

$$\epsilon_{j,k} = |\boldsymbol{w}_{j,k}^n| + |\boldsymbol{x}_{j,k}^n - \boldsymbol{x}_{j,k+1}^n|^2, \tag{3.8}$$

which uses the length of a wavelet vector plus the square of arc length as an indicator for space adaptivity. If the reference tolerance, the tolerance on the finest level is set to TOL, the tolerance TOL_j at the level j will be set to

$$\epsilon_{j,k} \le \mathrm{TOL}_j = 2^{(J-j)q} \mathrm{TOL}. \tag{3.9}$$

Then the overall truncation error ε_J is bounded by the sum of all local truncation errors,

$$\begin{aligned}\varepsilon_J &= \sup_{\substack{k\in I_j \\ 0\le t_n\le T}} |x^n_{J,k} - x_{J,k}(t_n)| \\ &\le \sum_{j=0}^{J}\sum_{n=0}^{\lfloor T/\Delta t_j\rfloor} \tau^n_{j,k} \le \sum_{j=0}^{J} C\Delta t^{p+1}\mathrm{TOL}2^{-j} \\ &\le C\Delta t^{p+1}\mathrm{TOL}. \end{aligned} \tag{3.10}$$

3.3. Implementation

The principle of the multiresolution method is to represent a set of data given on a fine mesh as values on a coarser mesh plus a series of differences at different levels of nested dyadic meshes. In order to equip the scheme with a dynamically evolving adaptive mesh, a sequence of nested meshes are constructed that each element is twice finer than the previous one. The differences contain the information of the interface when going from a coarse mesh to a finer one. Small wavelet coefficients on fine levels of resolution indicate the regions where the interface is smooth. The hierarchical data structure serves the simplicity of programming and allows fast algorithms for mesh refinement. By hierarchical we mean that the structures of the mesh are described hierarchically (points, lines, quadrilaterals etc.) and the refinement has to be made hierarchically as well. It is obvious that we should not construct a uniform hierarchy geometry in the adaptive algorithm. An element in the hierarchy geometry tree will be refined only when a mesh adaption is necessary based on the indicators in our implementation.

A mesh is a cluster of elements which can cover the whole domain but have no common interior parts. The elements of the meshes are coming from the nodes of the hierarchy geometry tree. To take out some of the nodes from the hierarchy geometry tree, we cut off the branches of the hierarchy geometry tree to get a finite sub-tree. If a node in the sub-tree has no further descendents, it is called a leaf node. The considered mesh will be composed by the leaf nodes of this finite sub-tree. The mesh adaption algorithm is based on the indicators (3.8) to modify the mesh to a new one. Therefore, if the indicator is great enough, then the element will be refined, and if on a patch of elements, the indicators are all very small, then these elements will be coarsened into a bigger element.

First, depending on the initial condition, an initial graded tree is created. Given the graded tree structure, a time evolution is made. Then we remesh the tree according to the indicators (3.8). In more detail, our algorithm includes the following steps:

1. Initialization
   - Create the root
   - Split elements and compute the wavelets in the children elements
   - If the wavelet in a child element is greater than the prescribed tolerance, split this child
   - Repeat the same procedure until all the children have smaller wavelets or the maximum level is reached
2. Time evolution
   - Compute Runge-Kutta steps in Section 3.1

3. Remesh
   - For the whole tree from the leaves to the root, if the indicator in this node and in its brothers are smaller than the prescribed tolerance, the element and its brothers are marked as deletable.
   - For the whole tree from the leaves to the root,
     - delete children if the node and all its children nodes are deletable
     - add one more level for each undeletable leaf if not at the maximum level.
   - Deallocate the tree

4. Numerical Examples

In this section, we will validate the computational efficiency with a few numerical examples. Bearing in mind that adaptivity is expected to pay off in the presence of interface expansions, in section 4.1 we consider first a problem where the length of the interface expands quickly. The second example in section 4.2 is concerned with more sophisticated problems. To our knowledge, such examples have not been studied before in the wavelet context. Therefore, the quantitative performance of the scheme should be very instructive. In the experiments, spatial approximations of second order, $q = 2$. For time discretization, we test first order method, Forward Euler scheme, and then the Runge-Kutta method of second order. Time step is, $\Delta t_j = \Delta t m^{j-J}$ with $m = 2$. All errors are measured averagely on all N leaf nodes and compared with a well resolved direct numerical simulation, $\frac{1}{N}\sum_i |\boldsymbol{x}_i(t_n) - \boldsymbol{x}_i^n|$.

4.1. Simple example

We consider the following test case. The velocity field is given by

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{pmatrix} x_2 \sin(x_1) - 0.5 \\ (x_1 + 0.2)\cos(x_2) + 0.4 \end{pmatrix}, \tag{4.1}$$

and the initial curve is given as a circle $|\boldsymbol{x}| = 1$. The numerical solution of (4.1) at $t = 2.0$ is given in Fig. 4.1 for TOL= 0.01, which includes $N = 105$ elements and corresponds to a maximum of $2^9 = 512$ elements with finest scale $J = 10$. We can notice that, the highest level is reached around the quick expansion region of the interface, which shows that the adaptive fast interface tracking method automatically detects the region where small scales are necessary and tracks the propagation phenomenon. In the right part of Fig. 4.1, we have plotted the sets of wavelet indices that correspond to the adaptively chosen leaf wavelets. We have compared the adaptive scheme with uniform refinement in order to get an impression of the effect of the different sizes of markers. The results of these numerical tests are shown in the left part of Fig. 4.2. We clearly see that the adaptive method is always cheaper in CPU time and memory requirements than the multiresolution method on the finest mesh. The error of the multiresolution method decreases almost linearly with degrees of freedom, which agrees to the theoretical result for the second-order method. The error of the adaptive method decreases a little faster than linear rate. They confirm indeed the gain of efficiency for adaptive schemes for this example. As the standard configuration of the methods we double the time step in each level. The timings of the methods are compared in the right part of Fig. 4.2. First it confirms

that the fast method has almost constant execution time, which doesn't grow linearly with N as the direct method, while the adaptive method preserves such property.

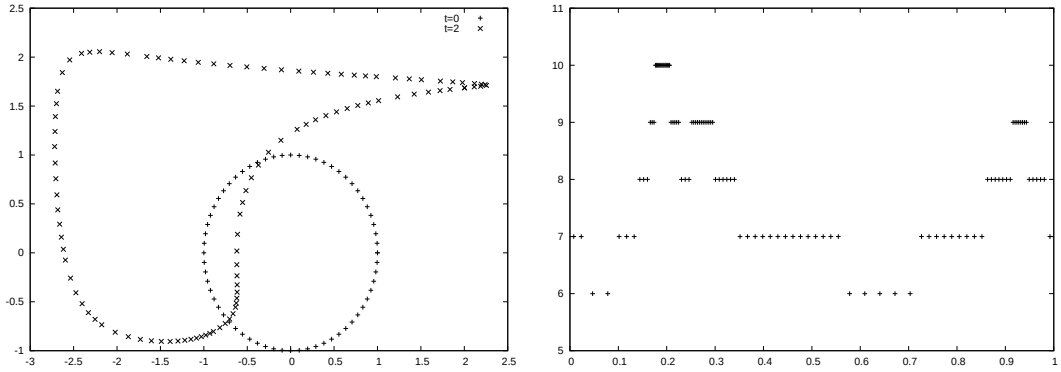

Fig. 4.1. Left: Initial interface (plus +) and computed interface (cross ×) by adaptive multiresolution at $t = 2$ with $TOL = 0.01$. Right: corresponding tree structure.

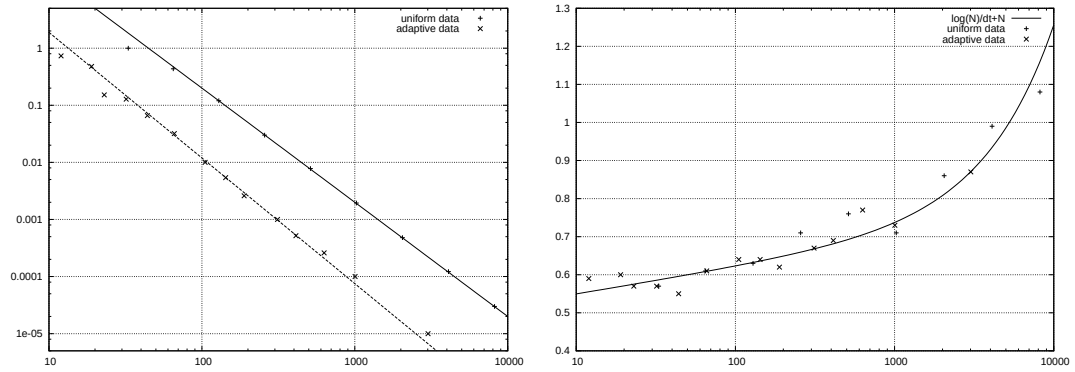

Fig. 4.2. Left: Errors $\inf_i |\boldsymbol{x}_i(t_n) - \boldsymbol{x}_i^n|$ at $t = 2$. Right: Comparisons of the CPU time with different DOFs and same overall error.

4.2. Vertex example

We have tested our algorithm for vertex flows, which are interesting because the interface may exhibit singularities solely caused by the vertex. The vertex velocity field is taken to be

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{pmatrix} \tanh(-x_2^2 + 0.25) \\ \sin(2\pi x_1) \end{pmatrix}. \tag{4.2}$$

The evolutions of the interface at $T = 2$ are shown in Fig. 4.3 for TOL $= 0.01$, which includes $N = 183$ leaf elements and corresponds to a maximum of $2^{11} = 2048$ elements with finest scale $J = 12$. Again, we have tested the performance of the adaptive algorithm in this regard. In the right part of Fig. 4.3, we have plotted the sets of leaf wavelet indices that correspond to the adaptively chosen wavelets. These wavelets are sometimes called active. One observes that the adaptive algorithm in fact detects the nearly singular gradient of the solution according

to high curvatures and adds wavelet coefficients locally in these regions. The highest level is reached around the steep gradient region, which shows that the adaptive multiresolution method automatically detects the region where more wavelets are necessary and tracks the interface.

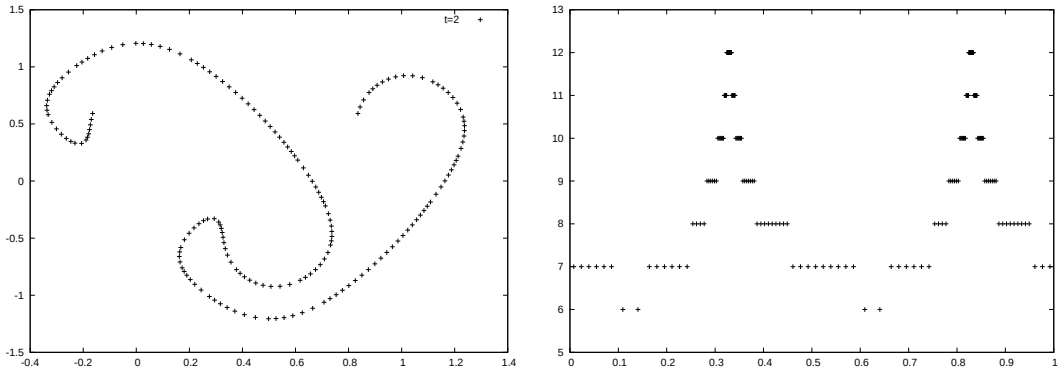

Fig. 4.3. Left: The computed points by adaptive multiresolution at $t = 2$ with $TOL = 0.001$. Right: corresponding tree structure.

In Fig. 4.4, the $L_\infty-$ errors have been displayed in a logarithmic scale for both the fast interface tracking method and the adaptive fast interface tracking method as the degree of freedom increases. Not only the efficiency of the adaptive scheme is improved as expected, but also both errors show almost the same decreasing behavior. Again CPU time are compared in the right part of Fig. 4.4.

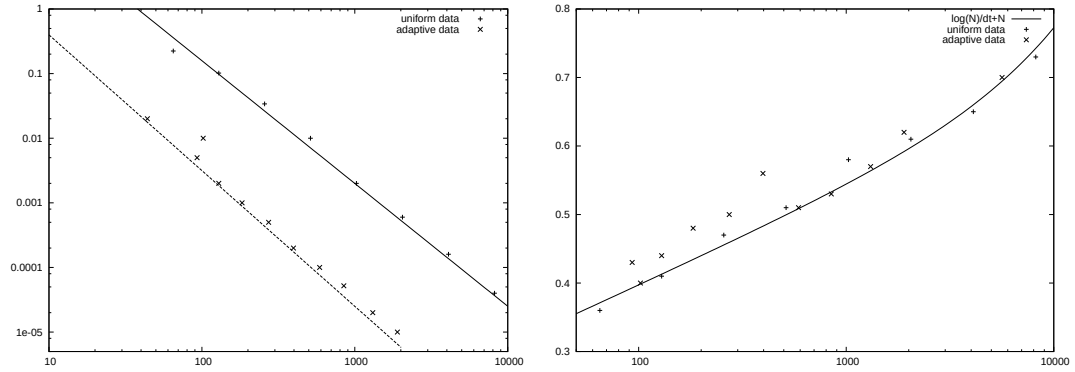

Fig. 4.4. Left: Errors $\inf_i |\boldsymbol{x}_i(t_n) - \boldsymbol{x}_i^n|$ at $t = 2$. Right: Comparisons of the CPU time with different DOFs and same overall error.

4.3. A wave front propagating in a heterogeneous medium

Finally, the adaptive fast interface tracking method is applied for more complicated interface tracking problem, a wave with a sharp front (which contains high frequencies) that propagates in a heterogeneous medium. Here the tracking has been done by a pure Lagrangian front tracking method. It is related to ray tracing, but instead of individual rays, the location of many rays coming from one source is computed, at fixed times. Those points form a wavefront and its evolution is tracked in the physical or the phase space in Fig. 4.5.

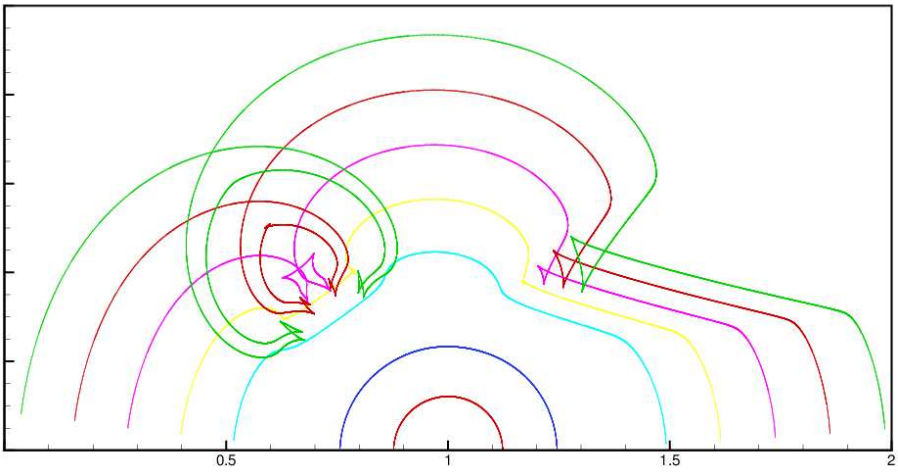

Fig. 4.5. A wave propagates from a point source through a heterogeneous medium.

5. Conclusion

In the present paper we developed a new adaptive numerical scheme to speed up interface tracking at a computational cost of $O(\log N/\triangle t)$ rather than $O(N/\triangle t)$ for N markers and timestep $\triangle t$. We demonstrated its computational efficiency and the numerical accuracy by computing several test-cases of interface tracking with given velocity fields.

Instead of tracking marker points on the interface we track the wavelet vectors, which like the markers satisfy ordinary differential equations. A dynamical adaption strategy which exploits the multiscale representation of the interface by adding finer scales allows us to advance the interface in time. The adaptive algorithm is implemented using a graded tree data structure to represent the adaptive interface in the computer memory. By taking 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, which is verified numerically. We have compared the performance in terms of CPU time and memory requirements to a direct tracking method using the same numerical schemes on the finest regular mesh with a static data structure.

The current work is dealing with the second order schemes. The developed adaptive scheme could be extended to higher order. To construct higher order methods more accurate prediction schemes are needed. In [3], a thorough analysis has been given that with higher order spatial averaging using general subdivision schemes, high order accuracy can be combined with low computational cost. We also plan to extend the developed scheme using high order Runge-Kutta schemes for the time stepping to match with sufficiently high order subdivision schemes.

Acknowledgments. This research is supported by the Dahlquist Research Fellowship in KTH. Y. Di is supported in part by the NSFC (11271358).

References

[1] J. Popovic, Fast Adaptive Numerical Methods for High Frequency Waves and Interface Tracking, PhD thesis, KTH Royal Institute of Technology, 2012.

[2] O. Runborg, Fast interface tracking via a multiresolution representation of curves and surfaces, *Commun. Math. Sci.*, **7** (2009), 365-398.
[3] O. Runborg, Analysis of high order fast interface tracking methods, *Numer. Math.*, **128**:2 (2014), 339-375.
[4] J. Glimm, E. Isaacson, D. Marchesin and O. McBryan, Front tracking for hyperbolic systems, *Adv. Appl. Math.*, **2** (1981), 91-119.
[5] S.J. Osher and J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, *J. Comput. Phys.*, **79**:1 (1988), 12-49.
[6] A.K. Tornberg and B. Engquist, The segment projection method for interface tracking, *Comm. Pure Appl. Math.*, **56**:1 (2003), 47-79.
[7] 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 surfaces, *Los Alalmos Scientific Laboratory Report LA*, **3425** (1966).
[8] C.W. Hirt and B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, *J. Comput. Phys.*, **39** (1981), 201-225.
[9] A. Harten, Adaptive multiresolution schemes for shock computations, *J. Comput. Phys.*, **115** (1994), 319-338.
[10] A. Cohen, S. M. Kaber, S. Müller and M. Postel, Fully adaptive multiresolution finite volume schemes for conservation laws, *Math. Comput.*, **72** (2003), 183-225.
[11] O. Roussel, K. Schneider, A. Tsigulin and H. Bockhorn, A conservative fully adaptive multiresolution algorithm for parabolic PDEs, *J. Comput. Phys.*, **188** (2003), 493-523.
[12] M.O. Domingues, S.M. Gomes, O. Roussel and K. Schneider, Adaptive multiresolution methods, *ESAIM Proc.*, **34** (2011), 1-96.
[13] K. Schneider and O.V. Vasilyev, Wavelet methods in computational fluid dynamics, *Annu. Rev. Fluid Mech.*, **42** (2010), 473-503.
[14] A. Cohen, Wavelet methods in numerical analysis, *Handb. Numer. Anal.*, **7** (2000), 417-711.
[15] M.O. Domingues, S.M. Gomes, O. Roussel and K. Schneider, An adaptive multiresolution scheme with local time stepping for evolutionary PDEs, *J. Comput. Phys.*, **227** (2008), 3758-3780.
[16] I. Daubechies, O. Runborg and W. Sweldens, Normal multiresolution approximation of curves, *Constr. Approx.*, **20** (2004), 399-463.
[17] S. Harizanov, P. Oswald and T. Shingel, Normal multi-scale transforms for curves, *Found. Comput. Math.*, **11**:6, (2011), 617-656.