Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 209 (2005) 391–405

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

An adaptive method with rigorous error control for the Hamilton–Jacobi equations. Part II: The two-dimensional steady-state case ☆

Bernardo Cockburn *, Bayram Yenikaya

School of Mathematics, University of Minnesota, 206 Church Street S.E., Minneapolis, MN 55455, USA

Received 8 October 2004; received in revised form 7 February 2005; accepted 8 February 2005
Available online 21 June 2005

Abstract

In this paper, we devise and study an adaptive method for finding approximations to the viscosity solution of Hamilton–Jacobi equations. The method, which is an extension to two space dimensions of a similar method previously proposed for one space dimension, is studied in the framework of steady-state Hamilton–Jacobi equations with periodic boundary conditions. It seeks numerical approximations whose L^{∞}-distance to the viscosity solution is no bigger than a prescribed tolerance. A thorough numerical study is carried out which shows that a strict error control is achieved and that the method exhibits an optimal computational complexity which does not depend on the value of the tolerance or on the type of Hamiltonian.
© 2005 Elsevier Inc. All rights reserved.

Keywords: Adaptivity; A posteriori error estimate; Hamilton–Jacobi equations

1. Introduction

In this paper, we continue the study of adaptive methods for Hamilton–Jacobi equations began in [4] for the one-dimensional case and extend it to two-dimensional problems of the same type. We present an efficient adaptive method for approximating the *viscosity* solution of the following model steady-state Hamilton–Jacobi equation:

☆ Partially supported by the National Science Foundation (Grant DMS-0411254) and by the University of Minnesota Supercomputing Institute.
* Corresponding author. Tel.: +1 612 625 2587; fax: +1 612 626 2017.
E-mail addresses: cockburn@math.umn.edu (B. Cockburn), yenikaya@math.umn.edu (B. Yenikaya).

$$u + H(\nabla u) = f \quad \text{in } \Omega = (0,1) \times (0,1), \tag{1}$$

with *periodic* boundary conditions. For any given positive parameter τ, the method seeks an approximation u_h satisfying the quality constraint

$$\|u - u_h\|_{L^\infty(\Omega_h)} \leqslant \tau, \tag{2}$$

where Ω_h is the set of nodes on which the approximate solution is computed. We show the efficiency and the reliability of the method through extensive numerical experiments.

Let us briefly describe its main components. To find an approximation to the viscosity solution, we use the monotone scheme introduced in [1] for unstructured triangular meshes. To verify whether or not the quality of the approximation is acceptable, that is, to see whether or not the constraint (2) is satisfied, we use an approximate version of the rigorous a posteriori error estimate obtained in [2], namely,

$$\|u - u_h\|_{L^\infty(\Omega)} \leqslant \Phi(u_h),$$

where Φ is a suitably defined non-linear functional which, roughly speaking, depends on the residual. The last component is how to use the information of the residual, the above error estimate and the tolerance to generate the mesh. Our method is not based on local mesh refinement; instead, it generates a new mesh at every iteration step. In our experience, this seems to work better than the local mesh refinement strategies that are usually used in adaptive algorithms. This approach enables us to generate meshes for which the quality constraint (2) is satisfied with optimal complexity; a rigorous proof of these properties remains an open problem.

To the knowledge of the authors, the method presented in this paper is the only adaptive algorithm for Hamilton–Jacobi equations with general Hamiltonians based on a rigorous a posteriori estimate of its error. In [4], the reader can find the study of the method in the simpler setting of one space dimension. An adaptive method for the Hamilton–Jacobi–Bellman equation, based on rigorous error estimates, was obtained by Grüne [8] and later extended to the time-dependent case in [10]; see also [9]. Adaptive methods for the time-dependent Hamilton–Jacobi equations have been developed by Tang et al. [16]. Even though their adaptive strategy is not based on a rigorous error estimate, their results are very impressive. The approximate a posteriori error estimator we use to drive our adaptivity could be easily incorporated into their approach.

The rest of the paper is organized as follows. In Section 2, we describe the details of each component of the main features of the method, namely the monotone numerical scheme that we use, the use of approximate a posteriori error estimate, and the way we update the mesh. In Section 3, we carry out a thorough numerical study of its performance on several test problems. Finally, in Section 4 we end with some concluding remarks.

2. The adaptive method

For any given tolerance $\tau > 0$, the adaptive method proceeds as follow:

(0) Construct an initial mesh Ω_h.
(1) Compute an approximate solution u_h on Ω_h.
(2) If $\Phi_h(u_h) \leqslant \tau$ then take u_h as the approximation sought and stop.
(3) Otherwise, compute a new mesh Ω_h, and go to (1).

In the rest of this section we describe all these steps in details, namely we explain the numerical method that we use to compute the approximate solution, the approximate a posteriori error estimate $\Phi_h(u_h)$, and most importantly how we compute a new mesh. But first, we begin by defining the viscosity solution of the problem under consideration.

2.1. The viscosity solution

To state the definition of the viscosity solution of the model equation (1), we need the definition of *semi-differentials* of a function. We follow the same notation as in [4] for this section and for the section on a posteriori error estimate. The *super-differential* of a function u at a point $x \in \mathbb{R}^2$, $D^+u(x)$, is the set of all vectors p in $\mathbb{R}^2$ such that

$$\limsup_{y\to x}\left(\frac{u(y)-\{u(x)+(y-x)\cdot p\}}{\|y-x\|}\right)\leqslant 0,$$

and the *sub-differential* of a function u at a point $x \in \mathbb{R}^2$, $D^-u(x)$, is the set of all vectors p in $\mathbb{R}^2$ such that

$$\liminf_{y\to x}\left(\frac{u(y)-\{u(x)+(y-x)\cdot p\}}{\|y-x\|}\right)\geqslant 0.$$

Also we need the following quantity before we define the viscosity solution:

$$R(u;x,p)=u(x)+H(p)-f(x)$$

which is just the residual of u at x if $p = \nabla u(x)$.

Definition 2.1 (*The viscosity solution* [5]). A viscosity solution of the Hamilton–Jacobi equation (1) is a continuous periodic function on $\mathbb{R}^2$ such that, for all x in $\mathbb{R}^2$,

$$+R(u;x,p)\leqslant 0 \quad \forall p\in D^+u(x), \qquad \text{and} \qquad -R(u;x,p)\leqslant 0 \quad \forall p\in D^-u(x).$$

If both the Hamiltonian H and the right-hand side f of (1) are Lipschitz, then the viscosity solution exists. See [5] and the references therein for more general results.

2.2. A monotone scheme

We use Lax–Friedrich's type monotone scheme on triangular meshes introduced by Abgrall [1]. Here we define the numerical scheme, but we leave out its derivation and convergence properties. We refer interested reader to [1] for more details.

Let T_h be a triangulation of (0, 1) × (0, 1) and $\{M_i\}_{i=1}^n$ be the set of nodes. For a fixed i, let $\{T_l\}_{l=0}^{k_i}$ be the set of triangles having M_i as a vertex, see Fig. 1. To this set of triangles is associated a family of angular sectors $\{\mathcal{O}_l\}_{l=0}^{k_i}$ defined as the inner sectors at M_i of all the triangles in $\{T_l\}_{l=0}^{k_i}$. Let θ_l denote the angle of $\mathcal{O}_l$, and let us assume that the indexing of the angular sectors is done clock-wise so that $\theta_l > 0$, and $\sum_{l=0}^{k_i}\theta_l = 2\pi$. We denote the unit vector of the half-line $\mathcal{O}_l \cap \mathcal{O}_{l+1}$ by $\vec{n}_{l+1/2}$, and we define $\beta_{l+1/2}$ as

$$\beta_{l+1/2}=\tan\left(\frac{\theta_l}{2}\right)+\tan\left(\frac{\theta_{l+1}}{2}\right), \quad l=0,\ldots,k_i,$$

where $k_i + 1$ is associated with 0. For any function ϕ let $\phi_i := \phi(M_i)$, $i = 1, \ldots, n$. We consider the function u, the solution we seek, to be piecewise linear on T_h, therefore its gradient is piecewise constant. Let ∇u_{T_l} be the constant value of the gradient of u on the triangle T_l.

The numerical scheme is defined as follows:

$$u_i+\mathscr{H}(u_i)=f_i, \quad i=1,\ldots,n, \tag{3}$$

where

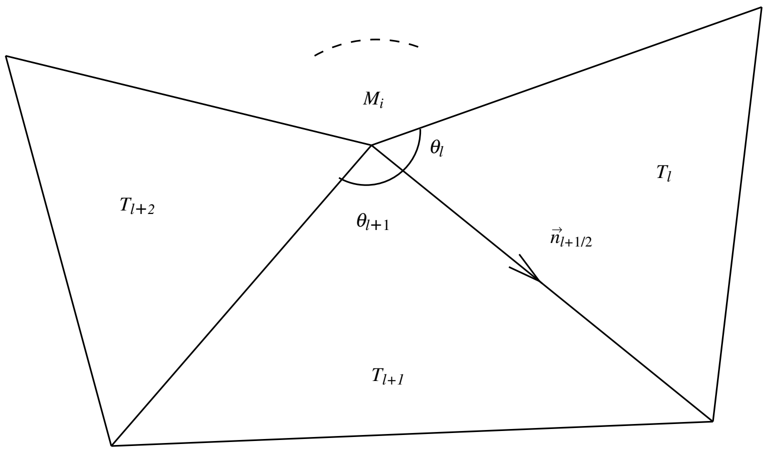

Fig. 1. Structure of the triangular mesh around a node M_i.

$$\mathscr{H}(u_i) = H\left(\frac{\sum_{l=0}^{k_i}\theta_l \nabla u_{T_l}}{2\pi}\right) - \omega \sum_{l=0}^{k_i} \beta_{l+1/2}\left(\frac{\nabla u_{T_l} + \nabla u_{T_{l+1}}}{2}\right) \cdot \vec{n}_{l+1/2}.$$

Here the artificial viscosity term ω is defined as

$$\omega = \frac{C(L)}{\pi}, \tag{4}$$

where $C(L)$ is a Lipschitz constant for H on the set $\{p \in \mathbb{R}^2 : \|p\| < 1\}$.

Since H is non-linear for our test problems, the scheme is implicit. Therefore we use Newton's Method to solve the non-linear system iteratively. To solve the linear equations, we used the GMRES method as implemented in the package SPARSKIT2, created by Youcef Saad [14].

2.3. The approximate a posteriori error estimate

In this subsection, we present the rigorous a posteriori error estimate obtained in [2], and then the approximate a posteriori error estimate that we use. We start by introducing several quantities that we need to state the error estimate. First, we introduce the following semi-norms:

$$|u - v|_- = \sup_{x\in\Omega}(u(x) - v(x))^+, \quad |u - v|_+ = \sup_{x\in\Omega}(v(x) - u(x))^+,$$

where $w^+ \equiv \max\{0, w\}$, and $\Omega = \mathbb{R}^2$. We consider $\Omega = (0, 1) \times (0, 1)$ because of the periodicity.

Then we define a generalization of the residual

$$R_\epsilon(u; x, p) = u(x) + H(p) - f(x - \epsilon p)\frac{1}{2}\varepsilon|p|^2, \tag{5}$$

which we call the *shifted residual.*

And finally, we introduce the paraboloid,

$$P_v(x, p, k; y) = v(x) + (y - x) \cdot p + \frac{k}{2}|y - x|^2, \quad y \in \mathbb{R}^2, \tag{6}$$

where x is a point in $\mathbb{R}^2$, p is a vector of $\mathbb{R}^2$, and k is a real number.

The rigorous a posteriori error estimate is stated in the following theorem.

Theorem 2.2 (A posteriori error estimate [2]). *Let u be the viscosity solution of the model equation (1) and let v be any continuous function on $\mathbb{R}^2$ periodic in each coordinate with period* 1. *Then, for* $\sigma \in \{-, +\}$,

$$|u - v|_\sigma \leqslant \inf_{\epsilon \geqslant 0} \Phi_\sigma(v;\epsilon), \tag{7}$$

where

$$\Phi_\sigma(v;\epsilon) = \sup_{(x,p)\in \mathscr{A}_\sigma(v;\epsilon)} (\sigma R_{\sigma\varepsilon}(v;x,p))^+. \tag{8}$$

The set $A_\sigma(v;\epsilon)$ is the set of elements (x, p) satisfying

$$x \in \Omega,$$
$$\sigma\{v(y) - P_v(x,p,\sigma/\epsilon;y)\} \leqslant 0 \quad \forall y \in \mathbb{R}^2.$$

(*For* $\epsilon = 0$, *we only require the first condition and that* $p \in D^\sigma v(x)$.)

Note that the above result implies that

$$\|u - v\|_{L^\infty(\Omega)} \leqslant \Phi(v),$$

where

$$\Phi(v) = \max\left\{\inf_{\epsilon\geqslant 0} \Phi_+(v,\epsilon), \inf_{\epsilon\geqslant 0} \Phi_-(v,\epsilon)\right\}.$$

In practice, we do not use the above functional Φ, but an approximation Φ_h, also introduced in [2]. The functional Φ_h is given by

$$\Phi_h(v) = \max\left\{\inf_{\epsilon\in\varepsilon_h} \Phi_{h,+}(v,\epsilon), \inf_{\epsilon\in\varepsilon_h} \Phi_{h,-}(v,\epsilon))\right\}.$$

Rather than taking the parameter ϵ in the interval $[0, \infty)$, we take it in the discrete set ε_h, where

$$\varepsilon_h = \{i \cdot E/N, 0 \leqslant i \leqslant N\},$$

where

$$E = 2h\omega|\ln(h\omega)| \quad \text{and} \quad N = 4|\ln(h\omega)|.$$

Here ω is the artificial diffusion coefficient used in the numerical scheme (4), and h is the size of the largest element in the triangulation. This choice of ε_h is motivated in [2].

It remains to describe the functional $\Phi_{h,\sigma}(v, \epsilon)$. These functional are obtained from the functional $\Phi_\sigma(v, \epsilon)$ by simply replacing the set $\mathscr{A}_\sigma(v;\epsilon)$ by the set $\Phi_{h,\sigma}(v; \epsilon)$ which is defined as the set of elements (x, p) satisfying

$$x \in \Omega_c,$$
$$\sigma\{(v(y) - P_v(x,p,\sigma/\epsilon;y)\} \leqslant 0 \quad \forall y \in \Omega_c : |y - x| \leqslant 2\|v\|_{\mathrm{Lip}(\Omega)}\epsilon,$$

where Ω_c is the set of barycenters of the elements in the triangulation. Note that since we assume u_h to be piecewise linear, when $v = u_h$ and x is the center of a triangle T_l, then the only element in $D^\sigma v(x)$ is $p = \Delta u_h(x)$, the gradient of u_h on the element T_l. Hence we have a very simple way to both compute the shifted residual, and find the set $\mathscr{A}_{h,\sigma}(v;\epsilon)$. Note also that if we replace Ω_c by Ω, the above test is *equivalent* to the original test in Theorem 2.2; see [2].

2.4. Computing a new mesh

Given a mesh Ω_h^k, we compute an approximate solution u_h^k by using the numerical scheme (3). If $\Phi_h(u_h^k) \geqslant \tau$, we must compute a new mesh, Ω_h^{k+1}. In [4], where we considered the 1-D case, we computed a new grid by properly denning a positive function Γ in terms of the old mesh, solving the equation

$$\frac{1}{\Gamma}\frac{\mathrm{d}}{\mathrm{d}x}\mathcal{N} = 1 \text{ in } (0,1), \quad \text{where } \mathcal{N}(0) = 0, \tag{9}$$

and, finally, setting

$$\{x_j = \mathcal{N}^{-1}(j)\}_{j=1}^{n},$$

where n is the smallest natural number greater or equal to $\mathcal{N}(1)$. A natural extension of such procedure would be to replace the above boundary value problem with using the Eikonal equation

$$\frac{1}{\Gamma}|\nabla\mathcal{N}| = 1 \text{ in } \Omega, \quad \text{where } \mathcal{N} = 0 \text{ on } \partial\Omega.$$

Mesh-generation algorithms using (more sophisticated versions of) this idea had already been proposed and tested by Hoch and Rascle [11] with very impressive results. To give a rough idea of how to do this, take Ω to be $\mathbb{R}^2 \setminus P$. Then

$$\mathcal{N}(Q) = \int_P^Q \Gamma\,\mathrm{d}\gamma,$$

where the integral is evaluated along the segment joining P and Q. If $\Gamma = 1/h$, the points Q for which $\mathcal{N}(Q) = 1$ form nothing but the circle of radius h centered at the point P. Then, six equilateral triangles could be easily formed with one of their vertices being the point P. Repeating this process, we could mesh $\mathbb{R}^2$ with equilateral triangles. If we take Γ equal to $1/h^k$, where the mesh-size function h^k is a piecewise constant function equal to the size of the longest edge of the triangle T of the mesh Ω_h^k, then we can generate a new mesh with a similar technique. If we want to modify the mesh Ω_h^k according to the mesh modification function μ, we could take Γ equal to μ/h^k. In this case, the quantity

$$l(P,Q) = \int_P^Q \frac{\mu(s)}{h^k(s)}\mathrm{d}\sigma$$

can be thought of as being a *normalized* distance P and Q and what we want is to generate a mesh on which the normalized length of each edge of elements is equal to 1. As an example, consider the case when the function μ is identically equal to an integer m. Then the normalized lengths of the edges of the old mesh, Ω_h^k, are equal to m. Therefore the new mesh should have elements m times smaller than the elements of the old one. Since the function μ determines how the old mesh is to be modified, it is called the mesh modification function.

In this paper, we do not follow the strategy proposed in [11]. Instead, we use a somewhat related, but different, algorithm which is implemented in the mesh generation package **Gmsh**, created by Christophe Geuzaine and Jean-François Remacle [7]. First, we generate the mesh of the boundary of the domain with the same algorithm used for 1-D problems in [4]. Then, we generate a coarse initial mesh of the domain with no interior points by a constrained Delaunay algorithm [15], which keeps the boundary mesh unchanged. We add new points, namely, the centroid of the elements of the current mesh whose *normalized* size is bigger than 1 by using the Bowyer-Watson algorithm [13]. We do this until the *normalized* lengths of all the edges are less than or equal to one. See [6] for more details on this algorithm as implemented in Gmsh.

It only remains to describe how to define the mesh modification function. We take $\mu(s) = \Psi(\gamma^k(s))$, where

$$\gamma^k(s) = \frac{1}{\tau}\min\left\{\max_{B_l}|R_{l_m}|, \Phi_h(u_h^k)\right\} \tag{10}$$

for $s \in T_l$, where the set B_l consists of the element T_l, three immediate neighboring elements of T_l, and the neighboring elements of those three. Here, the value of the residual on any triangle T_{l_m} is denoted by R_{l_m}. Moreover, Ψ is given by

$$\Psi(t) = \begin{cases} t(1+\sqrt{t-1}), & t > 1, \\ \frac{t+3}{4}, & 0 \leqslant t \leqslant 1. \end{cases}$$

See [4] for a thorough discussion of this function. This choice works equally well for both 1-D and 2-D problems as indicated by the numerical results in the next section. We use this choice of Ψ after the second step. For the first two steps, we use $\Psi(t) = t$ since, typically, the corresponding mehses we use are very coarse the function γ^k is already quite large.

If we start the algorithm with a very coarse mesh, then the mesh generated after the first iteration is not regular. This irregularity propagates into the next iterations and affects the quality of the meshes. Therefore we find it helpful to replace the adaptive irregular mesh at the second step with a uniform mesh with the same number of nodes. This improves the quality of the adaptive meshes in the successive iterations, however it does not alter the convergence properties of the method.

3. Numerical results

3.1. The test problems

We test our algorithm on four test problems; see Table 1. Problem 1 is one of the 1-D problems tested in [4], now extended to 2-D. We consider this problem to see whether or not the one-dimensional mesh structure is actually preserved when we use the two-dimensional version of the adaptive method. Then, to study the performance of the algorithm on two-dimensional problems, we consider three other problems. We consider a problem with a smooth solution, Problem 2, as well as two problems with kinks in the viscosity solution, Problems 3 and 4. Note that our problem set contains problems with both convex and non-convex Hamiltonians.

3.2. Results with the adaptive method

We begin by testing the quality of the a posteriori error estimate. To do that, we study the behavior of the effectivity index

$$\mathrm{ei}_h(u, u_h) = \frac{\Phi_h(u_h)}{\|u - u_h\|_{L^\infty(\Omega_h)}}$$

Table 1
Test problems

Problem	$H(p)$	Right-hand side $f(x, y)$	Viscosity solution $u(x, y)$
(1)	$-p_1^2/4\pi^2$	$\cos^4(\pi x)$	$\cos^2(\pi x)$
(2)	$p_1^2 + p_2^2$	$(\sin(2\pi x) + \cos(2\pi y))/2\pi + \cos^2(2\pi x) + \sin^2(2\pi y)$	$(\sin(2\pi x) + \cos(2\pi y))/2\pi$
(3)	$(p_1^2 + p_2^2)/\pi^2$	$-\lvert\cos(\pi x)\rvert - \lvert\cos(\pi y)\rvert + \sin^2(\pi x) + \sin^2(\pi y)$	$-\lvert\cos(\pi x)\rvert - \lvert\cos(\pi y)\rvert$
(4)	$-\lvert p\rvert^4 + 2\lvert p\rvert^2 - 1$	$u(x, y) - \sin^4(\pi(x + y))/4 + \sin^2(\pi(x + y)) - 1$	$-\lvert\cos(\pi(x + y))\rvert/2\pi$

on approximate solution obtained with uniform meshes. In Table 2, we see that, as expected, the order of the scheme is one, and that the a posteriori error estimate is reasonable given that its values remain close to 1 and do not increase, or increase very slowly, when the mesh is refined.

Next, we compare the performance of the adaptive method in the 1-D and the 2-D cases. In Fig. 2, we show the results for the first test problem on Table 1 with $\tau = 2.5\text{E}{-2}$ for both cases. We see that the results for 2-D case are qualitatively similar to those for 1-D. Indeed, they exhibit a 1-D structure since their main features are essentially constant along any line parallel to the y-axis. This gives a strong indication that it is reasonable to expect that the remarkable results obtained for the 1-D case will also hold in the 2-D case, in spite of the use of triangular unstructured meshes and a different algorithm to compute them.

Finally, in Table 3, we display the results obtained by applying the adaptive method to the four test problems. The method is tested for the tolerance values $\tau = 10\text{E}{-2}$, $5\text{E}{-2}$, and $2.5\text{E}{-2}$. We measure the success of the method by considering two effectivity indexes. The first is

$$\mathrm{ei}_\tau(\tau; u, u_h) = \frac{\tau}{\|u - u_h\|_{L^\infty(\Omega_h)}},$$

and shows whether the strict error control is achieved. From Table 3, we see that this index is always bigger than 1. In other words, the method enforces a strict error control and is thus reliable. The second effectivity index we consider is

Table 2
History of convergence for uniform grids

Problem	n	$\|u - u_h\|_{L^\infty(\Omega_h)}$	Order	$\mathrm{ei}_h(u, u_h)$
(1)	20^2	9.3E−2	0.00	1.08
	40^2	4.8E−2	0.94	1.05
	80^2	2.5E−2	0.98	1.04
	160^2	1.2E−2	0.99	1.03
	320^2	0.2E−3	0.99	1.03
	640^2	3.1E−3	1.00	1.03
	1280^2	1.6E−3	1.00	1.03
(2)	20^2	3.8E−1	0.00	1.82
	40^2	1.9E−1	0.98	1.92
	80^2	1.0E−1	0.98	1.95
	160^2	5.0E−2	0.99	1.92
	320^2	2.5E−2	0.99	1.86
	640^2	1.2E−2	0.99	1.80
	1280^2	6.0E−3	1.00	1.75
(3)	20^2	3.8E−1	0.00	1.46
	40^2	1.9E−1	0.97	1.46
	80^2	1.0E−1	0.98	1.67
	160^2	5.0E−2	0.99	2.11
	320^2	2.5E−2	0.99	2.13
	640^2	1.2E−2	1.00	2.15
	1280^2	6.0E−3	1.00	2.60
(4)	20^2	2.3E−1	0.00	1.07
	40^2	1.2E−1	0.92	1.91
	80^2	6.1E−2	0.97	1.55
	160^2	3.1E−2	0.98	1.96
	320^2	1.6E−2	0.99	2.52
	640^2	7.8E−3	1.00	3.13
	1280^2	3.9E−3	1.00	3.35

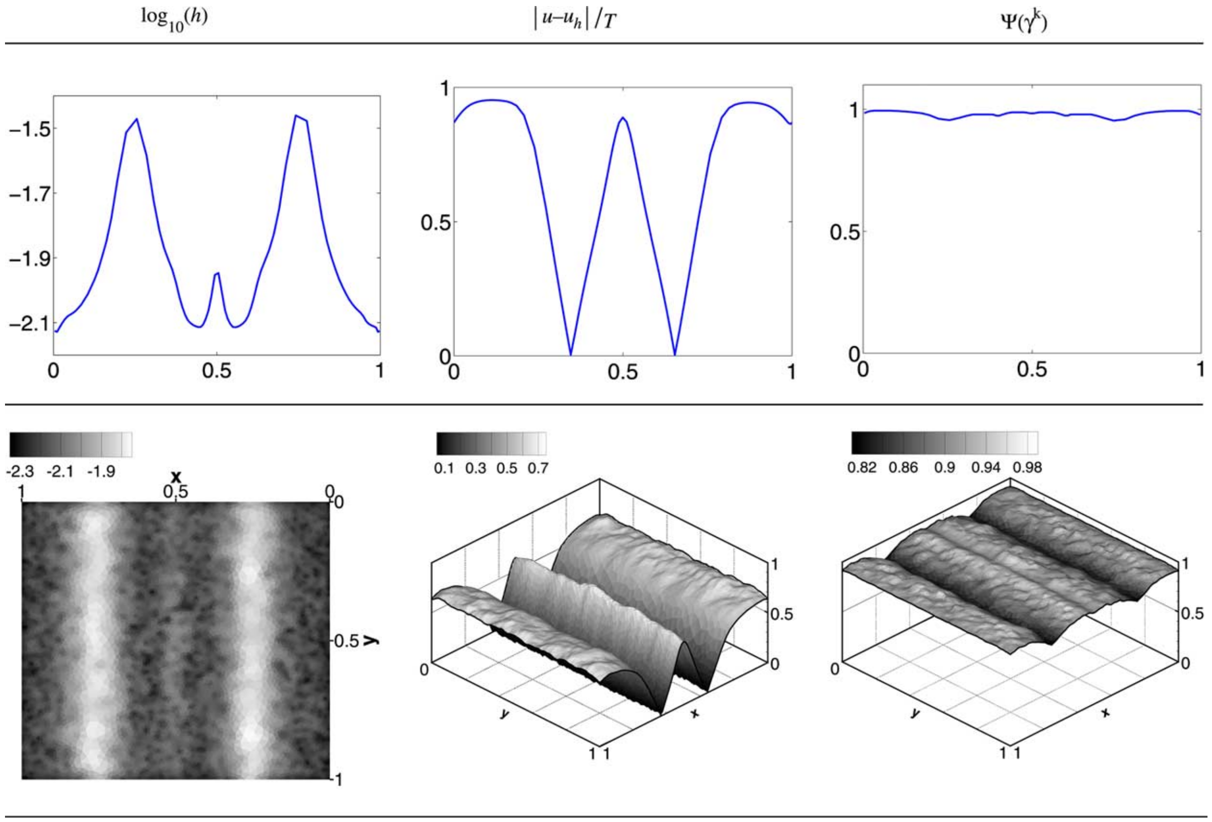

Fig. 2. The final step of Problem 1 for τ = 2.5E−2: one- (top) and two-dimensional (bottom) algorithms.

Table 3
Results of the adaptive method

Problem	τ	n	$\|u-u_h\|_{L^\infty(\Omega_h)}$	Order	ei_τ	$\mathrm{ei}_{\mathrm{adap}}$	Steps	cmplxr
(1)	10.0E−2	392	8.2E−2	–	1.20	1.01	3	1.96
	5.0E−2	1475	4.1E−2	1.06	1.21	1.00	3	1.82
	2.5E−2	7032	1.9E−2	1.01	1.34	1.06	4	2.43
(2)	10.0E−2	17499	4.6E−2	–	2.16	1.10	6	3.78
	5.0E−2	64830	2.3E−2	1.06	2.16	1.06	5	2.99
	2.5E−2	292324	1.0E−2	1.02	2.33	1.20	5	2.84
(3)	10.0E−2	14483	6.9E−2	–	1.44	1.08	4	2.23
	5.0E−2	65241	3.0B-2	1.09	1.64	1.16	5	2.60
	2.5E−2	186660	1.6E−2	1.16	1.51	1.03	4	2.30
(4)	10.0E−2	3690	6.5E−2	–	1.53	1.22	4	2.03
	5.0E−2	11863	3.4E−2	1.07	1.44	1.07	3	1.42
	2.5E−2	41569	1.7E−2	1.14	1.48	1.08	3	1.66

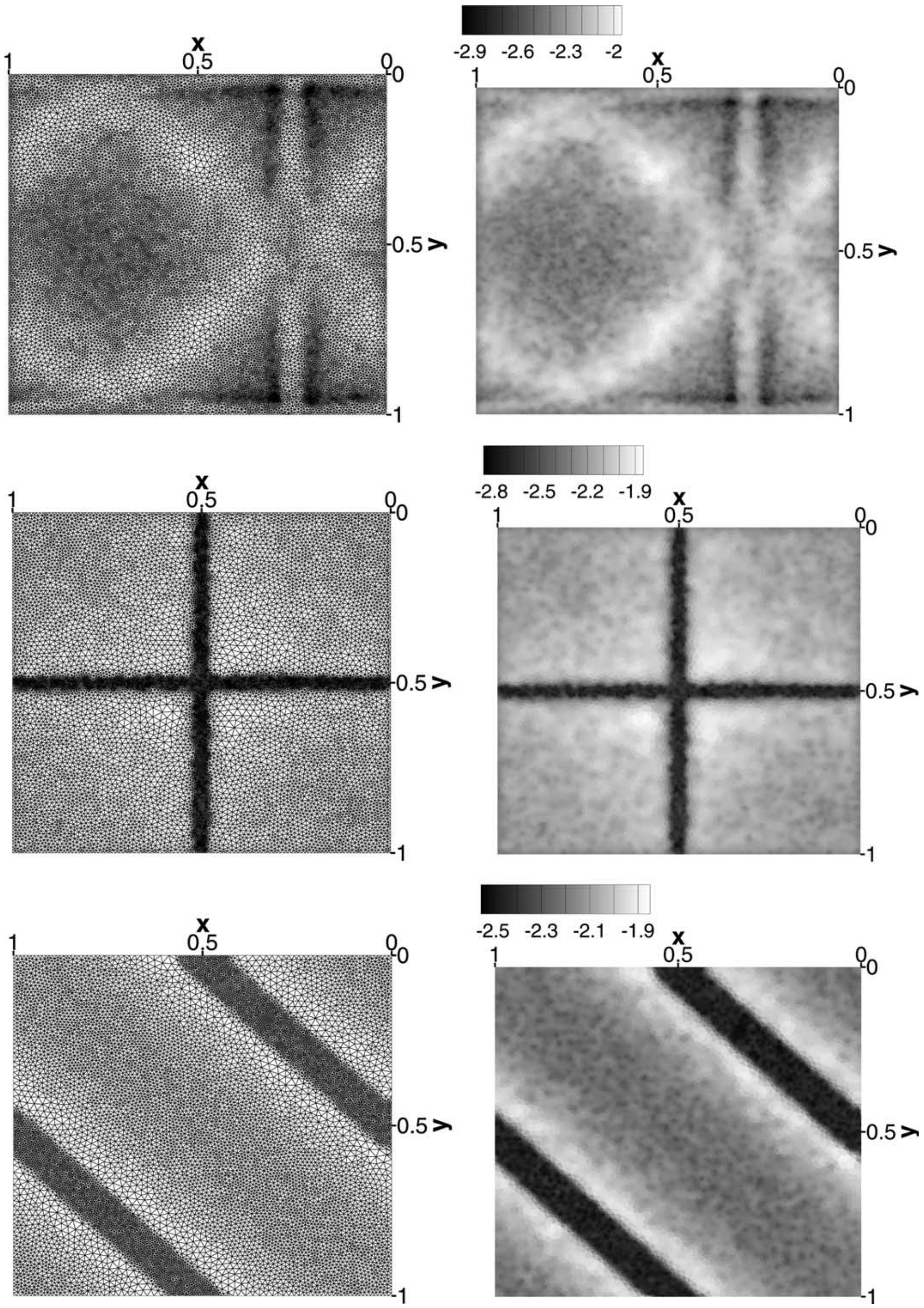

Fig. 3. Actual adaptive meshes (left column) and their mesh-size functions $\log_{10}(h)$ (right column) for Problem 2 (top), Problem 3 (middle), and Problem 4 (bottom).

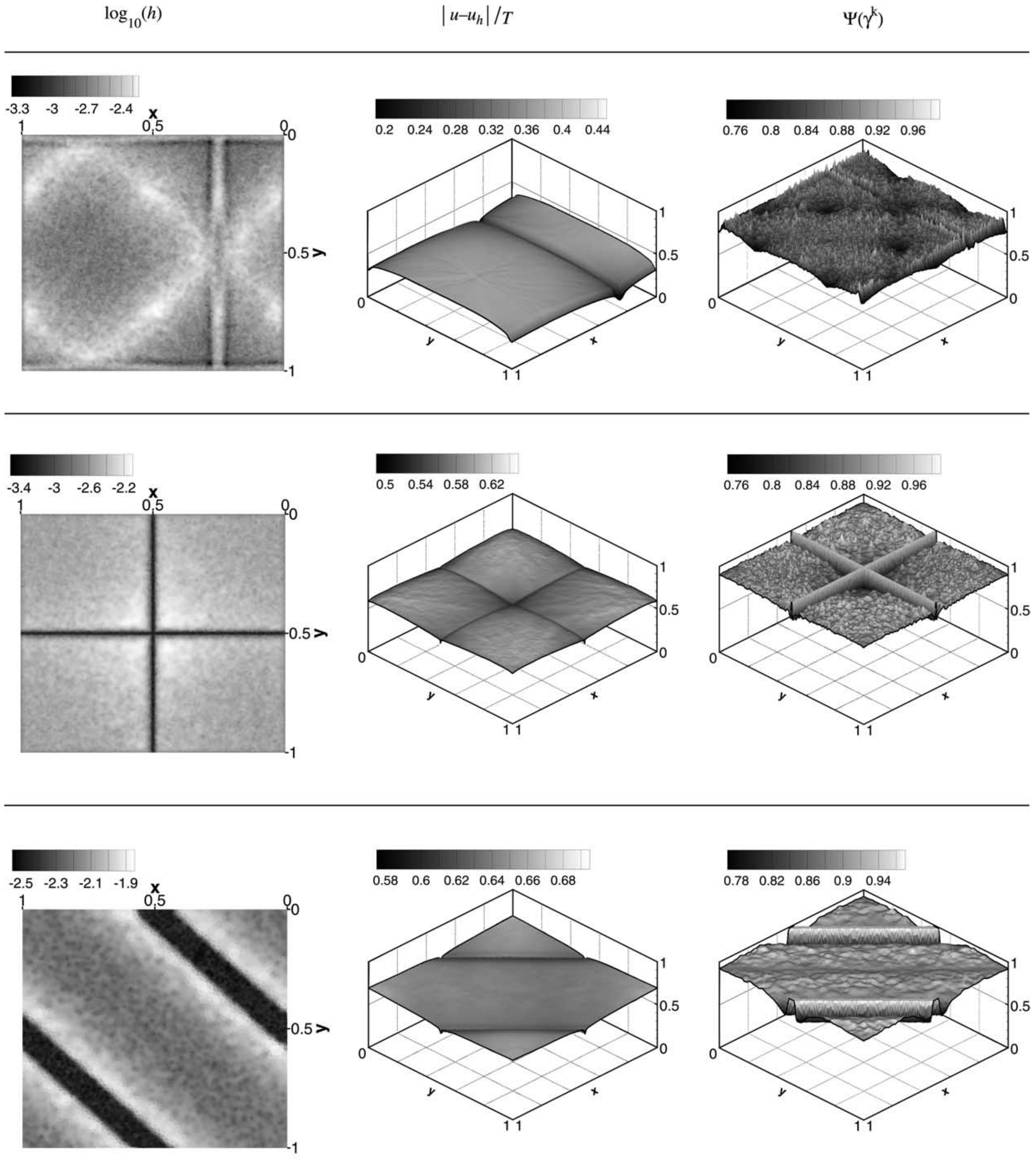

Fig. 4. The final step for $\tau = 5.0\mathrm{E}{-2}$, for Problem 2 (top), Problem 3 (center), and Problem 4 (bottom).

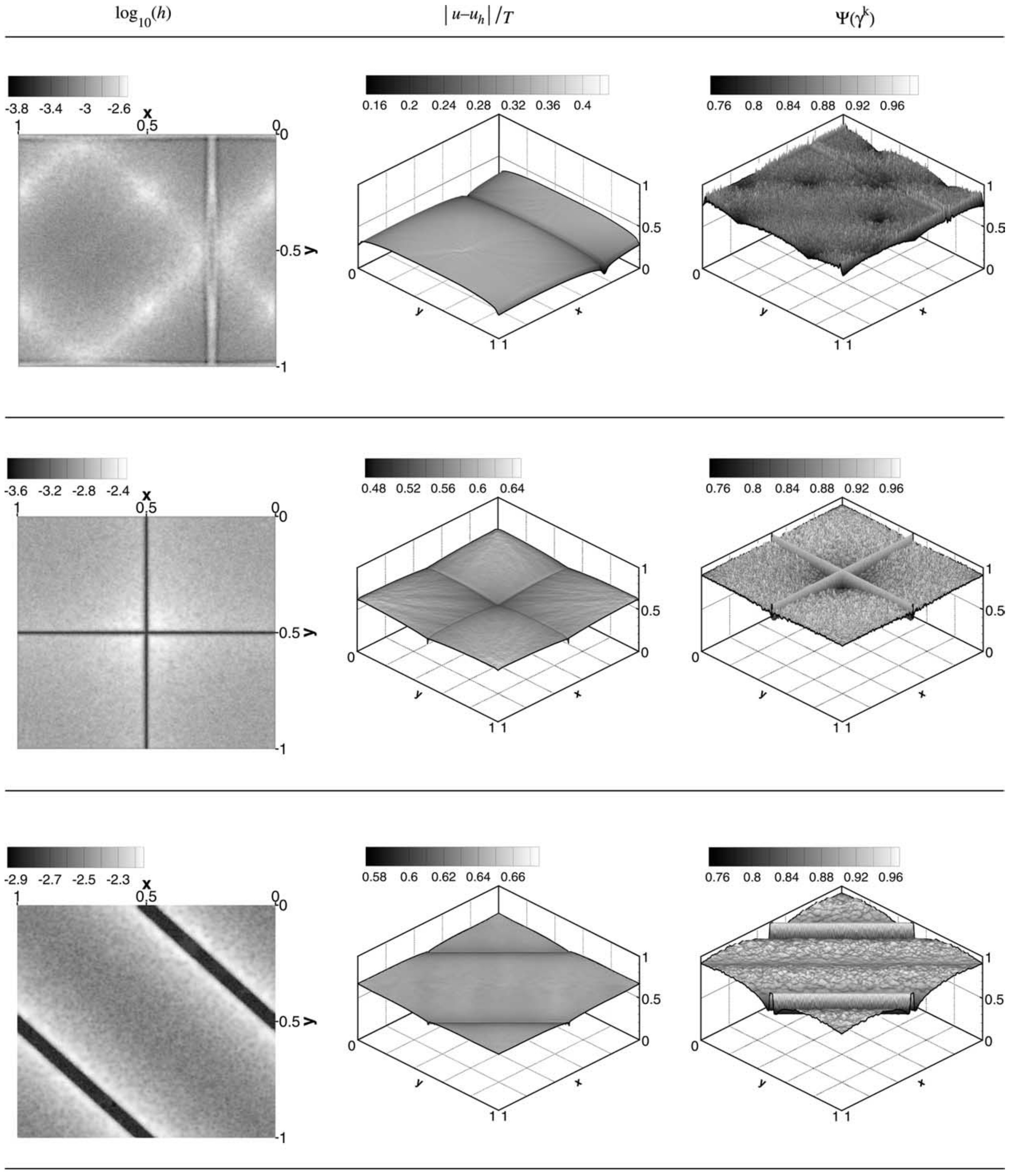

Fig. 5. The final step for $\tau = 2.5E{-}2$, for Problem 2 (top), Problem 3 (center), and Problem 4 (bottom).

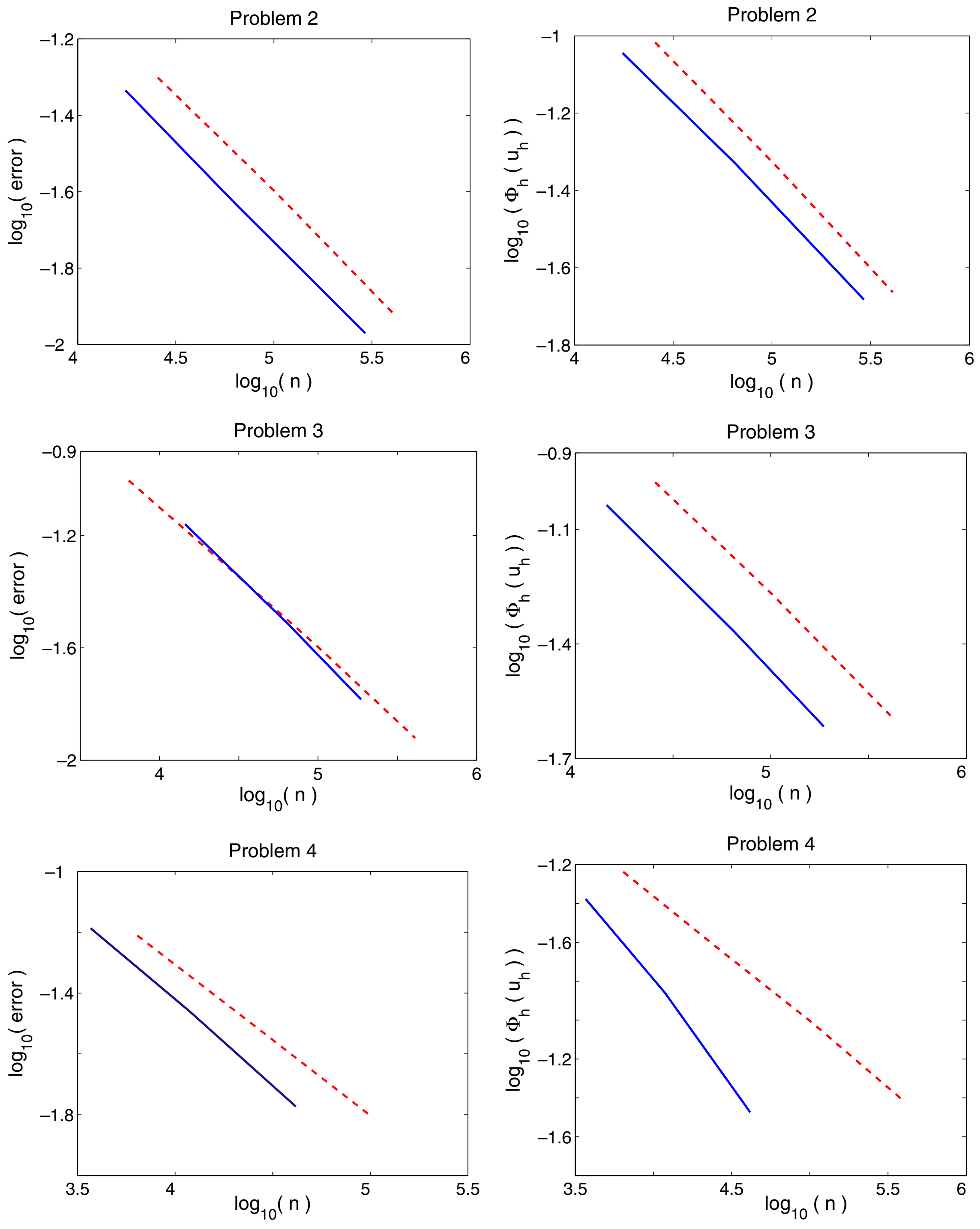

Fig. 6. Comparison of convergence rates: uniform (dashed line) and adaptive (solid line) refinement.

$$\mathrm{ei}_{\mathrm{adap}}(\tau; u_h) = \frac{\tau}{\Phi_h(u_h)},$$

which is a measure of the quality of the adaptive method. We see that this index is remarkably close to the ideal value 1, in spite of the variation of the value of the tolerance. This is an indication that the method is efficient. To further explore this issue, we need to observe the distribution of the error and the residual on the entire domain. We plot these in Figs. 4 and 5. (In Fig. 3, we show some meshes and the logarithm of their respective mesh size functions to give an idea of the meaning of such plots.) We see that the mesh modification function $\Psi(\gamma)$ is very close to the ideal value 1 almost everywhere. This indicates that any further modification of the mesh would result in very insignificant changes.

The above discussion refers only to the last step of the adaptive method. To measure the efficiency of the whole method we need to consider the computational complexity necessary to carry out all the steps. Since, in our case, the computational complexity is proportional to the number of nodes in the mesh, we introduce the quantity

$$\mathrm{cmplxr} = \frac{\sum_{k=1}^{K} n^K}{n^K},$$

where n^k is the number of nodes of the mesh of the step k, and K is the number of steps needed for convergence. We use this quantity to compare the computational complexity needed to carry out all the steps of the method to the computational complexity needed to carry out the last one. A bigger complexity ratio indicates a less efficient adaptive method. In the last two columns of Table 3, we display the numbers of steps needed for convergence and the complexity ratio. We see that the complexity ratio is always less than 4, independently of the problem and the value of the tolerance. In other words, the observed complexity of the method is optimal.

In our implementation, the CPU time to compute the approximate solution was smaller than that needed to compute the a posteriori error estimate. This, in turn, was *significantly* smaller than the computation of the new mesh.

3.3. Adaptivity versus uniform refinement

In this subsection we compare the performance of our adaptive method and that of using uniform meshes. In all our computations uniform triangular meshes are generated by drawing the diagonals of uniform rectangular meshes.

In Fig. 6, we make this comparison by considering both the exact error $\|u - u_h\|_{L^\infty(\Omega_h)}$ and the approximate a posteriori error estimate, $\Phi_h(u_h)$. We see that the adaptive method is more efficient in all the cases. For Problem 3, when considering the exact error for large tolerances both methods seem to perform similarly. However, when the tolerance becomes smaller the adaptive method becomes more efficient. This slight disadvantage at the beginning is actually compensated by a significant success when considering the approximate a posteriori error estimate. Hence over all the adaptive method is considerably more efficient.

4. Conclusions and extensions

In this paper, we have extended the adaptive method proposed in [4] to two-dimensional steady-state Hamilton–Jacobi equations. The method has been shown to guarantee a rigorous error control and to be extremely reliable and efficient for a wide variation of the tolerance parameter even in the presence of kinks in the viscosity solution with non-convex Hamiltonians. From these results, it is reasonable to expect that the extension of the algorithm to higher-dimensional problems will display a similar performance. A similar remark can be made about extending these results to Hamiltonians depending on both x and u.

The extension of this work to bounded domains could be done when the corresponding a posteriori error estimate becomes available. Such an estimate can actually be obtained in some simple cases, but the general case still remains an open problem. In the time-dependent case, straightforward extensions of the monotone scheme (3) can be used. However, in this case the time-space grids must be allowed to vary locally and the schemes must be properly modified to handle them. The a posteriori error estimate obtained in [3] can be used. Although we have focused on monotone schemes, the adaptive method proposed here can be easily applied to high-order accurate methods like the discontinuous Galerkin method developed in [12]. The above extensions constitute the subject of ongoing work.

Acknowledgments

The authors would like to give special thanks to Christophe Geuzaine for providing a source code for **Gmsh** and for his help in the installation and execution of the code. They would also thank Lars Grüne and Tao Tang for bringing their work on Hamilton–Jacobi equations to their attention.

References

[1] R. Abgrall, Numerical discretization of the first-order Hamilton–Jacobi equations on triangular meshes, Comm. Pure Appl. Math. 49 (1996) 1339–1377.

[2] S. Albert, B. Cockburn, D. French, T. Peterson, A posteriori error estimates for general numerical methods for Hamilton–Jacobi equations. Part I: The steady state case, Math. Comput. 71 (2002) 49–76.

[3] S. Albert, B. Cockburn, D. French, T. Peterson, A posteriori error estimates for general numerical methods for Hamilton–Jacobi equations. Part II: The time-dependent case, in: R. Herbin, D. Kröner (Eds.), Finite Volumes for Complex Applications, vol. III, Hermes Penton Science, 2002, pp. 17–24.

[4] B. Cockburn, B. Yenikaya, An adaptive method with rigorous error control for the Hamilton–Jacobi equations. Part I: The one-dimensional steady state case, Appl. Numer. Math. 52 (2005) 175–195.

[5] M.G. Crandall, L.C. Evans, P.-L. Lions, Some properties of viscosity solutions of Hamilton–Jacobi equations, Trans. Amer. Math. Soc. 282 (1984) 487–502.

[6] P.L. George, Tet meshing: construction, optimization, and adaptation, in: Proceedings of the 8th International Meshing Rountable, 1999, pp. 133–141.

[7] C. Geuzaine, J.-F. Remacle, Gmsh reference manual: the documentation for Gmsh, a finite element mesh generator with built-in pre- and post-processing facilities. Available from: <http://www.geuz.org/gmsh/>.

[8] L. Grüne, An adaptive grid scheme for the discrete Hamilton–Jacobi–Bellman equation, Numer. Math. 75 (1997) 319–337.

[9] L. Grüne, Adaptive grid generation for evolutive Hamilton–Jacobi–Bellman equations, in: M. Falcone, Ch. Makridakis (Eds.), Numerical Methods for Viscosity Solutions and Applications, World Scientific, 2001, pp. 153–172.

[10] L. Grüne, Error estimation and adaptive discretization for the discrete stochastic Hamilton–Jacobi–Bellman equation, Numer. Math. 99 (2004) 85–112.

[11] PH. Hoch, M. Rascle, Hamilton–Jacobi equations on a manifold and applications to grid generation or refinement, SIAM J. Sci. Comput. 23 (2002) 2055–2073.

[12] C. Hu, C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM J. Sci. Comput. 21 (1999) 666–690.

[13] S. Rebay, Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm, J. Comput. Phys. 106 (1) (1993) 125–138.

[14] Y. Saad, Sparskit: a basic tool kit for sparse matrix computations. Available from <http://www-users.cs.umn.edu/saad/software/SPARSKIT/sparskit.html>.

[15] J.R. Shewchuk, Constraint Delaunay tetrahedralization and provably good boundary recovery, in: Proceedings of the 11th International Meshing Roundtable, 2002, pp. 193–204.

[16] H.-Z. Tang, T. Tang, P.-W. Zhang, An adaptive mesh redistribution method for nonlinear Hamilton–Jacobi equations in two- and three-dimensions, J. Comput. Phys. 188 (2003) 543–572.