Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 226 (2007) 1027–1058

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

An adaptive high-order discontinuous Galerkin method with error control for the Hamilton–Jacobi equations. Part I: The one-dimensional steady state case

Yanlai Chen *, Bernardo Cockburn [1]

School of Mathematics, University of Minnesota, 127 Vincent Hall, 206 Church St. SE, Minneapolis, MN 55455, United States

Received 30 October 2006; received in revised form 30 April 2007; accepted 3 May 2007
Available online 13 May 2007

Abstract

We propose and study an adaptive version of the discontinuous Galerkin method for Hamilton–Jacobi equations. It works as follows. Given the tolerance and the degree of the polynomial of the approximate solution, the adaptive algorithm finds a mesh on which the approximate solution has an L^∞-distance to the viscosity solution no bigger than the prescribed tolerance. The algorithm uses three main tools. The first is an iterative solver combining the explicit Runge–Kutta discontinuous Galerkin method and the implicit Newton's method that enables us to solve the Hamilton–Jacobi equations efficiently. The second is a new a posteriori error estimate based on the approximate resolution of an approximate problem for the actual error. The third is a method that allows us to find a new mesh as a function of the old mesh and the ratio of the a posteriori error estimate to the tolerance. We display extensive numerical evidence that indicates that, for any given polynomial degree, the method achieves its goal with optimal complexity independently of the tolerance. This is done in the framework of one-dimensional steady-state model problems with periodic boundary conditions.
© 2007 Elsevier Inc. All rights reserved.

MSC: 65N15; 65N30; 65N50; 68W40

Keywords: Hamilton–Jacobi equations; Adaptivity; A posteriori error estimates; Discontinuous Galerkin methods

1. Introduction

We introduce and study an adaptive method that can efficiently compute an approximation, with a guaranteed precision set beforehand by the practitioner, to the viscosity solution of the model steady-state Hamilton–Jacobi equation,

$$u + H(u_x) = f \quad \text{in } \mathbb{R}, \tag{1.1}$$

* Corresponding author. Tel.: +1 612 625 3395; fax: +1 612 626 2017.
E-mail addresses: ylchen@math.umn.edu (Y. Chen), cockburn@math.umn.edu (B. Cockburn).
[1] Author was supported in part by the National Science Foundation (Grant DMS-0411254) and by the University of Minnesota Supercomputing Institute. Tel.: +1 612 625 2587; fax: +1 612 626 2017.

where the function f is periodic of period one. The method finds, for any given tolerance τ and any fixed polynomial degree k, a grid G_h such that

$$\|u - u_h\|_{L^\infty(\boldsymbol{G}_h)} \leqslant \tau,$$

where the grid $\boldsymbol{G}_h$ is a refinement of the grid G_h. Here, u_h is the approximation given by the discontinuous Galerkin (DG) method using polynomials of degree k proposed by Hu and Shu [7]. In this paper, we display extensive numerical evidence indicating that this adaptive algorithm actually reaches its goal with optimal complexity.

This method can be considered a nontrivial extension to the DG method of the adaptive method introduced by Cockburn and Yenikaya [4,5] for monotone schemes. Indeed, our algorithm has a similar structure, namely,

(i) Construct an initial grid G_h.
(ii) Compute the DG approximate solution u_h on the grid G_h.
(iii) Compute the estimate of the error, $\Phi_h(u_h)$.
(iv) If $\|\Phi_h(u_h)\|_{L^\infty(\boldsymbol{G}_h)} \leqslant \tau$, **stop**.
(v) If not, compute a new grid G_h and go to Step (ii).

On the other hand, the fact that the approximate solution u_h is not given by a monotone scheme forced us to replace the a posteriori error estimate used in [4,5] by a *new* one. The a posteriori error estimate used therein was obtained by Albert et al. [1] and worked very well on monotone schemes. It also worked well on DG methods *provided* the mesh was uniform and a suitable post-processing was applied to the solution *before* evaluating the a posteriori error. In fact, if such a post-processing is not used, the ratio of the a posteriori error estimate to the true error grows like $(\Delta x)^{-1}$ where Δx is the maximum mesh size, instead of staying close to the optimal value of one. In contrast, for the new a posteriori error estimate, that ratio remains constant and fairly close to one, uniformly with respect to the grid, the polynomial degree and the tolerance.

The adaptive method we study here is able to achieve a strict error control with optimal complexity uniformly in the tolerance τ and the polynomial degree k of the approximation even in the presence of kinks in the viscosity solution. It is the first adaptive method with these properties for the Hamilton–Jacobi equations.

Let us illustrate how the adaptive method works by displaying its approximate solution for the cases:

$$H(p) = \frac{p^3}{8\pi^3}, \quad f(x) = \sin(2\pi x) + \cos^3(2\pi x)$$

and

$$H(p) = \frac{p^2}{\pi^2}, \quad f(x) = -\left|\sin\left(\pi\left(x - \frac{\pi}{4}\right)\right)\right| + \cos^2\left(\pi\left(x - \frac{\pi}{4}\right)\right).$$

The first problem has a smooth solution $u(x) = \sin(2\pi x)$ while the viscosity solution of the second, $u(x) = -|\sin(\pi(x - \frac{\pi}{4}))|$, has a kink at $\frac{\pi}{4}$.

The results for the first problem with a tolerance $\tau = 10^{-6}$ and polynomial degree $k = 1, 2, 3, 4$ are displayed in Fig. 1 where we plot the approximate solutions on the meshes generated by the method. We see that the number of intervals we need to achieve this accuracy decreases dramatically as we increase the polynomial degree, as expected. If we recall that it takes 120, 749 intervals for a monotone scheme (which corresponds to taking $k = 0$ in the DG method) to obtain an accuracy of 10^{-4} for the same problem, see [4], we see that the use of high-degree polynomials is highly advantageous.

The results for the second problem are shown in Fig. 2. Note that in this case, when the solution has kinks, the adaptive method automatically identifies a region around the kink where it uses piecewise constant approximations; elsewhere, it uses polynomials of degree k. We see that, as it is typical of DG methods, the increase of the degree of polynomial approximation does not result in the degradation of the overall approximation; the reasonably good resolution of the kink is also maintained. Unfortunately, we see that

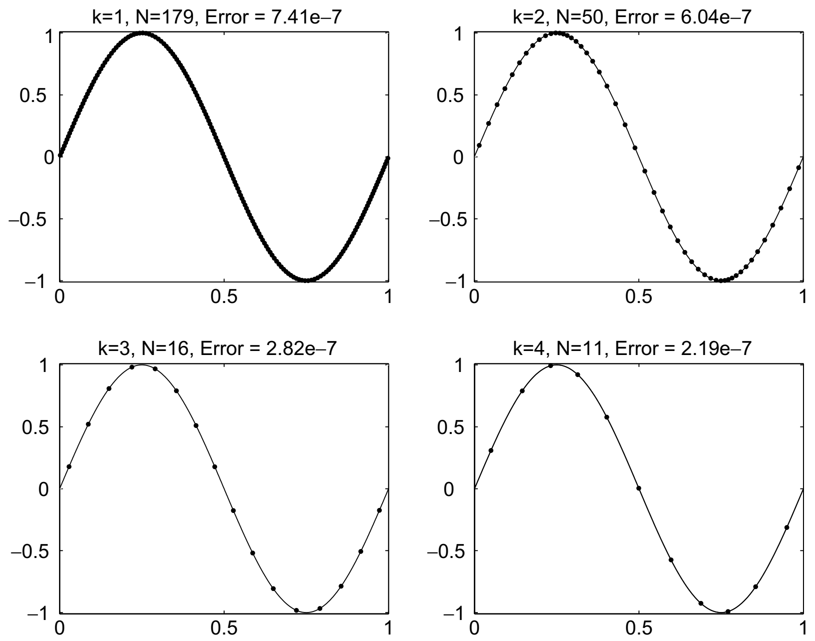

Fig. 1. The approximate solution u_h (solid line) generated by the method with $\tau = 10^{-6}$ for the first test problem. The polynomial degree for the approximation of u_x is denoted by k and the number of elements by N. The value of the approximate solution at the midpoints of the elements is plotted with dark circles.

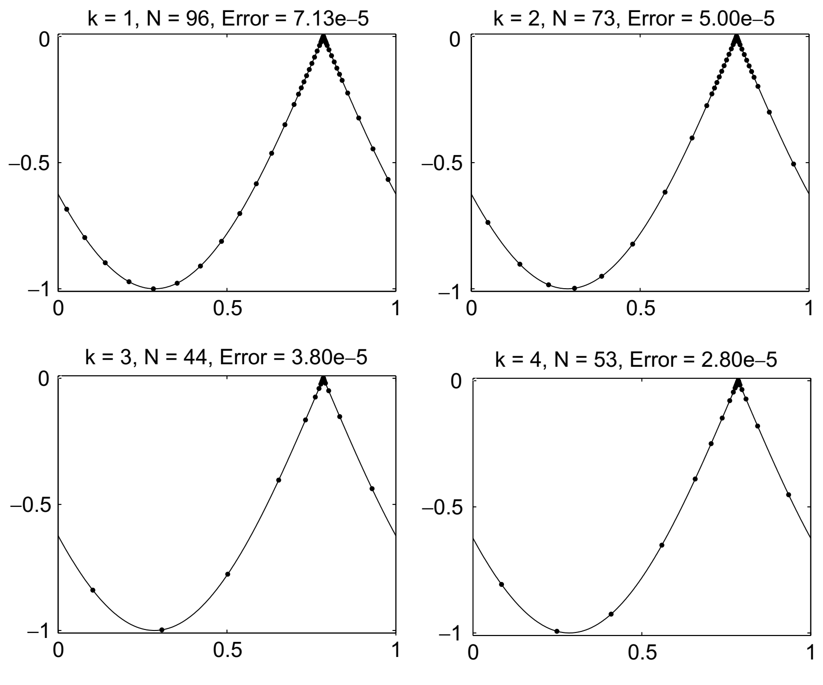

Fig. 2. The approximate solution u_h (solid line) generated by the method with $\tau = 10^{-4}$ for the second test problem. The polynomial degree for the approximation of u_x is denoted by k and the number of elements by N. The value of the approximate solution at the midpoints of the elements is plotted with dark circles.

the number of intervals does not monotonically decrease as the polynomial degree k increases. In particular, we need more elements for the $k = 4$ case than the $k = 3$ case. This differs from what happens when the solution is smooth. Indeed, as we see in Fig. 1, the higher the order of the scheme, the fewer the number of elements we need. This might be improved by using a better, rigorous a posteriori error estimate. Nonetheless, note that, with the current strategy, we do capture the kink better when the polynomial degree is higher. In contrast, when monotone schemes are used, an enormous number of intervals is required to achieve a given same accuracy. For example, in [4], 40, 148 intervals were needed when working with a tolerance of $\tau = 10^{-4}$.

The paper is organized as follows. In Section 2, we describe each of the main three ingredients of the method, namely, the iterative solver to solve the nonlinear system resulting from the DG method of Hu and Shu [7], the new a posteriori error estimate, and the way of computing a new grid in terms of the old grid and the information given by the ratio of the a posteriori error estimate to the tolerance. In Section 3, we carry out a thorough numerical study on six, qualitatively different test problems; we show that the new a posteriori error estimate is sharp and the adaptive method is effective and robust. Finally, we end in Section 4 with some concluding remarks.

2. The adaptive method

In this section, we briefly describe the exact solution we seek to approximate, namely, the viscosity solution of the Hamilton–Jacobi equations. We then describe the adaptive algorithm in full detail: First, we display the DG method, then the a posteriori error estimate and finally, how to compute a new mesh. We end with a brief summary of the adaptive method.

2.1. The viscosity solution

To state the definition of the viscosity solution of (1.1), we need the notion of *semi-differentials* of a function. The *superdifferential* of a function u at a point x, $D^+u(x)$, is the set of all vectors p in $\mathbb{R}$ such that

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

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

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

We also need to define the following quantity:

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

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

We are now ready to define the viscosity solution of (1.1).

Definition 2.1 [6]. *The viscosity solution of the Hamilton–Jacobi equation* (1.1) is a continuous periodic function on $\mathbb{R}$ such that, for all x in $\mathbb{R}$,

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

2.2. Discontinuous Galerkin schemes

Next, we adapt the discontinuous Galerkin Runge–Kutta method of Hu and Shu [7], originally devised for transient Hamilton–Jacobi problems, to our steady state setting. Roughly speaking, the idea is first to obtain $\varphi = u_x$, which solves the scalar hyperbolic conservation law

$$\varphi + H(\varphi)_x = f_x$$

and then compute u by the formula

$$u(x) = f(0) - H(\varphi(0)) + \int_0^x \varphi(s).$$

Notice that we are using the equation $u = f - H(u_x)$ to define the value of u at $x = 0$.

To obtain the approximation φ_h to φ, we proceed as follows. By the periodicity of the problem (1.1), we only need to solve it on the interval $(0,1)$. Given any grid $G_h = \{x_i\}_{i=0}^N$, where $0 = x_0 < x_1 < \cdots < x_N = 1$, we set $I_j = (x_j, x_{j+1})$, $h_j = x_{j+1} - x_j$, $j = 0,\ldots,N-1$. Then, on the interval I_j, the approximate solution φ_h is defined as the element of $P^{k_j}(I_j)$, the space of polynomials of degree at most k_j defined on I_j, such that

$$\int_{x_j}^{x_{j+1}} (\varphi_h v - H(\varphi_h)v_x - f_x v)\,\mathrm{d}x + \widehat{H}_{j+1}v(x_{j+1}^-) - \widehat{H}_j v(x_j^+) = 0, \tag{2.1a}$$

$\forall v \in P^{k_j}(I_j)$, where

$$\widehat{H}(a,b) = \frac{1}{2}[H(a) + H(b) - C(b-a)], \tag{2.1b}$$

and

$$C = \max_{\{s\in[0,1]:u_x(s)\ \text{exists}\}} |H'(u_x(s))|. \tag{2.1c}$$

Finally, our approximate solution is given by

$$u_h(x) = f(0) - \widehat{H}_0 + \int_0^x \varphi_h(s)\,\mathrm{d}s.$$

Next, we describe the iterative solver of Eq. (2.1).

2.3. The iterative solver

The iterative method to solve our equations has three major components. The first is the Newton's method with which we are going to generate a sequence converging to the solution. It is well known that the convergence of Newton's method takes place only if the initial guess should be close enough to the solution. To provide such initial guess, we are going to use the second component of the iterative solver which is nothing but the RKDG method. The third component is a special procedure that will allow us to deal with the high-order nature of the solution. This solver is a nontrivial extension of the solver used in [1] and in [4,5] for monotone schemes.

To be able to describe the solver, we need to introduce some notation. We are going to describe the form that the approximate solution φ_h has by simply saying that $\varphi_h \in V_{h,\boldsymbol{k}}(G_h)$, where, for $G_h = \{x_i\}_{i=0}^N$ and $I_j = (x_j, x_{j+1})$,

$$V_{h,\boldsymbol{k}}(G_h) := \{v \in L^2(0,1) : v|_{I_j} \in P^{k_j}(I_j) \quad \forall j = 0,\ldots,N-1\}, \tag{2.2}$$

where $\boldsymbol{k} := (k_0, k_1, \ldots, k_{N-1})$. In what follows, we use the notation $\boldsymbol{k}_\ell := (\ell, \ell, \ldots, \ell)$ and we use $V_{h,\boldsymbol{k}}$ instead of $V_{h,\boldsymbol{k}}(G_h)$ for simplicity if there is no confusion.

We can now introduce the projection $\mathbb{P}^\ell$, from $V_{h,\boldsymbol{k}}$ with $k_j \leqslant \ell$ for $j = 0,\ldots,N-1$ to $V_{h,\boldsymbol{k}_\ell}$: for any function $\phi \in V_{h,\boldsymbol{k}}$, we define $\mathbb{P}^\ell\phi \in V_{h,\boldsymbol{k}_\ell}$ by requiring that, for any $j = 0,\ldots,N-1$ and any polynomial v of degree ℓ

$$\int_{I_{j-1}\cup I_j\cup I_{j+1}} (E_j(\mathbb{P}^\ell\phi) - \phi)v\,\mathrm{d}x = 0,$$

where the function $E_j(\mathbb{P}^\ell\phi)$ is the natural extension of $\mathbb{P}^\ell\phi|_{I_j}$ to $I_{j-1} \cup I_j \cup I_{j+1}$. Notice that, by periodicity, $I_{-1} := I_{N-1}$ and $I_N := I_0$.

We denote the degrees of freedom of $\varphi_h \in V_{h,k}$ by $\vec{\varphi}_h := (\vec{\varphi}_{h,0}, \vec{\varphi}_{h,1}, \ldots, \vec{\varphi}_{h,N-1})$, where $\vec{\varphi}_{h,j} := (\varphi_0^j, \varphi_1^j, \ldots, \varphi_{k_j}^j)$ are the degrees of freedom of $\varphi|_{I_j}$. Then, Eq. (2.1a) can be rewritten as the following nonlinear system of equations for $\vec{\varphi}_h$:

$$\mathbb{F}(\vec{\varphi}_h) = 0.$$

It is well known that the DG method described above with $\boldsymbol{k} = \boldsymbol{k}_0$ gives rise to a well defined monotone scheme; see [2]. In this paper, we do not address the issue of the existence and uniqueness of the approximate solution for arbitrary $\boldsymbol{k}$. However, we devise an iterative algorithm that does converge in practice; it is a combination of Newton's method and the RKDG method.

The iteration given by Newton's method is

$$N(\vec{\varphi}_h^{(m)}) := \vec{\varphi}_h^{(m)} - \left(\mathbb{F}'(\vec{\varphi}_h^{(m)})\right)^{-1} \cdot \mathbb{F}(\vec{\varphi}_h^{(m)}).$$

It is reasonable to believe that the Newton's method will converge to the desired solution provided we use piecewise constant approximations wherever the exact solution is not smooth. The identification of such a region is a crucial part of our iterative method. Such a region is determined by information provided by the slope limiter of the RKDG method. We use the RKDG method exactly as described in [7]; see [3] for a detailed description. In what follows, we denote the degrees of freedom obtained by applying one RKDG step by $\mathrm{RKDG}(\vec{\varphi}_h)$.

Finally, we are going to use two operators with which we are going to determine the space of the solution we are seeking. The introduction of these operators is motivated by the well known fact that the use of piecewise-constant approximations around the kinks can ensure the absence of unphysical spurious oscillation in the approximate solution. Both operators are related to the slope limiter used by the RKDG method, $\Lambda\Pi_h$ which is going to be described later, and make sure that those unphysical spurious oscillations are damped. The image of the first operator $\boldsymbol{K} := \mathrm{SLD}(\varphi_h, \boldsymbol{k})$, where $\boldsymbol{k} := (k_0, k_1, \ldots, k_{N-1})$, is defined by

$$K_j := \begin{cases} 0 & \text{if} \quad \Lambda\Pi_h \varphi_h|_{I_j} = \bar{\varphi}_h|_j, \\ k_j & \text{otherwise.} \end{cases} \tag{2.3}$$

The image of the second operator $\boldsymbol{K} := \mathrm{PD}(\varphi_h, \boldsymbol{k}, \ell)$, where $\boldsymbol{k} := (k_0, k_1, \ldots, k_{N-1})$ and ℓ is an integer, is defined by

$$K_j := \begin{cases} 0 & \text{if } \Lambda\Pi_h \varphi_h|_{I_j} = \bar{\varphi}_h|_j \quad \text{or there exist } j_1 \leqslant j \quad \text{and } j_2 \geqslant j \quad \text{such that} \\ & k_\ell = 0 \quad \text{for } \ell \in \{j_1, \ldots, j_2\} \quad \text{and} \\ & \Lambda\Pi_h \varphi_h|_{I_\ell} = \bar{\varphi}_h|_\ell \quad \text{for } \ell \in \{j_1 - 1, j_2 + 1\}, \\ \ell & \text{otherwise.} \end{cases} \tag{2.4}$$

We are now ready to describe the iterative solver.

Given a polynomial degree k and a grid G_h, the iterative solver finds a space $V_{h,\boldsymbol{k}}$ and a solution $\varphi_h \in V_{h,\boldsymbol{k}}$ of Eq. (2.1) as follows:

Algorithm 1. (Iterative solver)

(i) **Initialization**:
(α) Set $\ell = 0$.
(β) Initialize $\vec{\varphi}_h^{(0)} \in V_{h,\boldsymbol{k}_\ell}$ by zero.
(γ) Set the flow control parameter ϵ to 10^{-3}.

(ii) **Computation of an initial guess for Newton's method**:
(α) Set $\boldsymbol{k}_{\mathrm{RK}}^{(0)} := \boldsymbol{k}_\ell$ and $m := 0$.
(β) Given the function $\vec{\varphi}_h^{(m)} \in V_{h,\boldsymbol{k}_{\mathrm{RK}}^{(m)}}$, compute the next iterate $\vec{\varphi}_h^{(m+1)} \in V_{h,\boldsymbol{k}_{\mathrm{RK}}^{(m+1)}}$, where $\boldsymbol{k}_{\mathrm{RK}}^{(m+1)} := \mathrm{SLD}(\varphi_h^{(m)}, \boldsymbol{k}_{\mathrm{RK}}^{(m)})$ by using the RKDG method:

$$\vec{\varphi}_h^{(m+1)} := \mathrm{RKDG}(\vec{\varphi}_h^{(m)}).$$

(γ) Evaluate $e := \|\vec{\varphi}_h^{(m+1)} - \vec{\varphi}_h^{(m)}\|_{\ell^\infty}$ and proceed as follows:
(1) If $e \leqslant \epsilon$, the initial guess for Newton's method is

$$\vec{\varphi}_h^{(1)} := \mathbb{P}^\ell \vec{\varphi}_h^{(m+1)},$$

and the associated vector of polynomial degrees is

$\boldsymbol{k}_{\mathrm{RK}}^{(1)} := \mathrm{PD}(\vec{\varphi}_h^{(1)}, \boldsymbol{k}_{\mathrm{RK}}^{(m+1)}, \ell)$.

(2) Otherwise, go to step (β).

(iii) **Computation of the iterates of Newton's method**:

(α) Set $\boldsymbol{k}_{\mathrm{NT}} := \boldsymbol{k}_\ell$ and $m := 1$.

(β) Given the function $\vec{\varphi}_h^{(m)} \in V_{h,\boldsymbol{k}_{\mathrm{NT}}}$, compute the next iterate $\vec{\varphi}_h^{(m+1)} \in V_{h,\boldsymbol{k}_{\mathrm{NT}}}$ by using Newton's method:

$\vec{\varphi}_h^{(m+1)} := N(\vec{\varphi}_h^{(m)})$.

(γ) Evaluate $e := \|\mathbb{F}(\vec{\varphi}_h^{(m+1)})\|_{\ell^\infty}$ and proceed as follows:

(1) If $e \leqslant 10^{-14}$, we consider $\vec{\varphi}_h^{(m+1)}$ to be the solution of the nonlinear system with polynomials of degree ℓ. Then

(a) If $\ell = k$, we consider $\vec{\varphi}_h^{(m+1)}$ to be the solution we seek and **stop**.

(b) Otherwise, we consider $\vec{\varphi}_h^{(m+1)}$ as the initial guess for the Newton's method for solving the nonlinear system with polynomials of degree $\ell + 1$. Thus, we set $\vec{\varphi}_h^{(1)} := \mathbb{P}^{\ell+1}\vec{\varphi}_h^{(m+1)}$, $\epsilon := 2 \times 10^{-3}$, $\ell := \ell + 1$ and go to step (α).

(2) If $e > 10^{-14}$ and $m < 10$, set $m := m + 1$ and go to step (β) to compute another iterate.

(3) If $e > 10^{-14}$ and $m = 10$, then

(a) If $\boldsymbol{k}_{\mathrm{NT}} = \boldsymbol{k}_\ell$ and $\ell > 0$, we consider that the space $V_{h,\boldsymbol{k}_{\mathrm{NT}}}$ was not suitable for the computations; we change it and begin anew. Thus, we set $\boldsymbol{k}_{\mathrm{NT}} := \boldsymbol{k}_{\mathrm{RK}}^{(1)}$, $m := 1$ and go to step (β).

(b) Otherwise, we consider that the initial guess $\vec{\varphi}_h^{(1)}$ obtained by using the RKDG method in step (ii) is not suitable and we go back to that step to obtain a better guess. Thus, we set $\vec{\varphi}_h^{(0)} := \vec{\varphi}_h^{(1)}$, $\epsilon := \epsilon/2$, and go to step (ii).

We can see that, the iterative solver uses the explicit RKDG method to advance to a point that is close enough to the solution of (2.1a) and then switches to using Newton's method to try to achieve a much faster convergence. We can also see that, to find the approximate solution associated with the polynomial degree k, we do obtain the solutions associated with the degree ℓ for $\ell = 0, \ldots, k$. The reason is that the solution associated with the degree ℓ provides a good initial guess to compute the solution associated with the degree $\ell + 1$. In our experience, this algorithm is very efficient and it only takes a few Newton iterations to converge. In the presence of kinks with nonconvex Hamiltonian and for meshes with elements of very different sizes, the algorithm takes significantly longer time to converge as the Newton's method requires an initial guess that is very close to the solution. Obtaining such an initial guess via the RKDG method is time-consuming since the size of the time step is very small when the mesh is extremely fine on part of the domain around the kink.

To complete the description of the iterative solver, we still have to describe the slope limiter we use. It is similar to the generalized slope limiter $\Lambda\Pi_h$ considered in [3]. To describe it, we need the following notation. For any function $\varphi_h \in V_{h,\boldsymbol{k}}$, we set

$$(\Lambda\Pi_h v_h)_{j+1}^- = \bar{v}_j + \frac{1}{2}h_j m\left(\frac{v_{j+1}^- - \bar{v}_j}{h_j/2}, \frac{\bar{v}_{j+1} - \bar{v}_j}{(h_j + h_{j+1})/4}, \frac{\bar{v}_j - \bar{v}_{j-1}}{(h_{j-1} + h_j)/4}\right), \tag{2.5a}$$

$$(\Lambda\Pi_h v_h)_j^+ = \bar{v}_j - \frac{1}{2}h_j m\left(\frac{\bar{v}_j - v_j^+}{h_j/2}, \frac{\bar{v}_{j+1} - \bar{v}_j}{(h_j + h_{j+1})/4}, \frac{\bar{v}_j - \bar{v}_{j-1}}{(h_{j-1} + h_j)/4}\right), \tag{2.5b}$$

where $\bar{v}_j := \frac{1}{h_j}\int_{x_j}^{x_{j+1}} v\,\mathrm{d}x$ and the *minmod* function m is defined by

$$m(a_1, a_2, a_3) = \begin{cases} a_1, & \text{if } |a_1| \leqslant 2Mh_j, \\ s \min\limits_{1 \leqslant n \leqslant 3} |a_n|, & \text{else if } s = \mathrm{sign}(a_1) = \mathrm{sign}(a_2) = \mathrm{sign}(a_3), \\ 0, & \text{else.} \end{cases}$$

Assuming that the parameter M is known, the slope limiter is defined as follows.

Algorithm 2. (*Slope limiter operator $\Lambda\Pi_h$*).
Given φ_h, for $j = 0, 1, \ldots, N-1$, we define $\Lambda\Pi_h\varphi_h|_{I_j}$ by

(i) Compute $(\Lambda\Pi_h\varphi_h)^-_{j+1}$ and $(\Lambda\Pi_h\varphi_h)^+_j$ by using (2.5a) and (2.5b).
(ii) If $\varphi^-_{hj+1} = (\Lambda\Pi_h\varphi_h)^-_{j+1}$ and $\varphi^+_{hj} = (\Lambda\Pi_h\varphi_h)^+_j$, set $\Lambda\Pi_h\varphi_h|_{I_j} = \varphi_h|_{I_j}$. If not, set $\Lambda\Pi_h\varphi_h|_{I_j} = \bar{\varphi}_h|_j$.

This completes the description of step (ii) of Algorithm 1 for the iterative solver.

It remains to show how to compute the parameter M. It is not difficult to see that M is nothing but an upper bound of the absolute value of the second-order derivative of the solution at local extrema. Given an approximation to the exact solution, $\varphi_h^\star$, we compute $M = M(\varphi_h^\star)$ as follows.

Algorithm 3. (*Computation of $M(\varphi_h)$*).

(i) For each interval I_{i+1}, $i = 0, 1, \ldots, N-1$, compute the parabola $p^i_{\varphi_h}(x)$ determined by the conditions

$$p^i_{\varphi_h}(x_{i-1/2}) = \bar{\varphi}_{hi-1},$$
$$p^i_{\varphi_h}(x_{i+1/2}) = \bar{\varphi}_{hi},$$
$$p^i_{\varphi_h}(x_{i+3/2}) = \bar{\varphi}_{hi+1},$$

where $x_{j+1/2}$ is the midpoint of the interval I_j.
(ii) For each interval I_{i+1}, $i = 0, 1, \ldots, N-1$, set $M_i = 0$. If any of the following conditions holds:
(1) $\bar{\varphi}_{hi} > \mathrm{Max}\{\bar{\varphi}_{hi-1}, \bar{\varphi}_{hi+1}\}$ and $\left|1 - \frac{M_{i-1}+M_{i+1}}{2M_i}\right| < 0.05$,
(2) $\bar{\varphi}_{hi} < \mathrm{Min}\{\bar{\varphi}_{hi-1}, \bar{\varphi}_{hi+1}\}$ and $\left|1 - \frac{M_{i-1}+M_{i+1}}{2M_i}\right| < 0.05$,
set $M_i = \left|{p^i_{\varphi_h}}''\right|$.
(iii) Set $M(\varphi_h) = \mathrm{Max}\{M_0, M_1, \ldots, M_{N-1}\}$.

The second part of the condition (1) or (2) is a practical way to differentiate critical points from discontinuities. It is motivated by the following computation. Assuming $\varphi(x)$ is smooth, we have the following Taylor expansions:

$$\varphi(x_{j+1}) = \varphi(x_j) + h\varphi'(x_j) + \frac{h^2}{2}M_j + \frac{h^3}{6}\mathcal{N} + \mathrm{O}(h^4)$$
$$\varphi(x_j) = \varphi(x_{j+1}) - h\varphi'(x_{j+1}) + \frac{h^2}{2}M_{j+1} - \frac{h^3}{6}\mathcal{N} + \mathrm{O}(h^4)$$
$$\varphi(x_{j-1}) = \varphi(x_j) - h\varphi'(x_j) + \frac{h^2}{2}M_j - \frac{h^3}{6}\mathcal{N} + \mathrm{O}(h^4)$$
$$\varphi(x_j) = \varphi(x_{j-1}) + h\varphi'(x_{j-1}) + \frac{h^2}{2}M_{j-1} + \frac{h^3}{6}\mathcal{N} + \mathrm{O}(h^4)$$

Here, $M_j = \varphi''(x_j)$, $M_{j+1} = \varphi''(x_{j+1})$, $M_{j-1} = \varphi''(x_{j-1})$ and we assume $\varphi'''(x_{j-1}) = \varphi'''(x_j) = \varphi'''(x_{j+1}) = \mathcal{N}$. Adding the four equations expanding $\varphi''(x_{j+1})$ and $\varphi''(x_{j-1})$ at x_j to obtain

$$1 - \frac{M_{j+1} + M_{j-1}}{2M_j} = \mathrm{O}(h^2).$$

Of course, the question is now how to obtain such an approximation $\varphi_h^\star$. In our experience, it is enough to take $\varphi_h^\star$ as the solution associated to a uniform grid of 20 elements and a space of piecewise linear approximations. Of course, we cannot use Algorithm 1 to do that, but we can use a very similar procedure we describe next.

Algorithm 4. (*Computation of* $\varphi_h^\star$).

(i) **Initialization**:
- (α) Set $\ell = 0$.
- (β) Initialize $\vec{\varphi}_h^{(0)} \in V_{h,\boldsymbol{k}_\ell}$ by zero.
- (γ) Set the flow control parameter ϵ to 10^{-3}.

(ii) **Computation of an initial guess for Newton's method**:
- (α) Set $\boldsymbol{k}_{\mathrm{RK}}^{(0)} := \boldsymbol{k}_\ell$ and $m := 0$.
- (β) Given the function $\vec{\varphi}_h^{(m)} \in V_{h,\boldsymbol{k}_{\mathrm{RK}}^{(m)}}$, compute the next iterate $\vec{\varphi}_h^{(m+1)} \in V_{h,\boldsymbol{k}_{\mathrm{RK}}^{(m+1)}}$, where $\boldsymbol{k}_{\mathrm{RK}}^{(m+1)} := \mathrm{SLD}(\varphi_h^{(m)}, \boldsymbol{k}_{\mathrm{RK}}^{(m)})$ by using the RKDG method:
  $$\vec{\varphi}_h^{(m+1)} := \mathrm{RKDG}\left(\vec{\varphi}_h^{(m)}\right).$$
- (γ) Evaluate $e := \left\|\vec{\varphi}_h^{(m+1)} - \vec{\varphi}_h^{(m)}\right\|_{\ell^\infty}$ and proceed as follows:
  - (1) If $e \leqslant \epsilon$, the initial guess for Newton's method is
    $$\vec{\varphi}_h^{(1)} := \mathbb{P}^\ell \vec{\varphi}_h^{(m+1)},$$
    and the associated vector of polynomial degrees is
    $$\boldsymbol{k}_{\mathrm{RK}}^{(1)} := \mathrm{PD}\left(\vec{\varphi}_h^{(1)}, \boldsymbol{k}_{\mathrm{RK}}^{(m+1)}, \ell\right).$$
  - (2) Otherwise, go to step (β).

(iii) **Computation of the iterates of Newton's method**:
- (α) Set $\boldsymbol{k}_{\mathrm{NT}} := \boldsymbol{k}_\ell$ and $m := 1$.
- (β) Given the function $\vec{\varphi}_h^{(m)} \in V_{h,\boldsymbol{k}_{\mathrm{NT}}}$, compute the next iterate $\vec{\varphi}_h^{(m+1)} \in V_{h,\boldsymbol{k}_{\mathrm{NT}}}$ by using Newton's method
  $$\vec{\varphi}_h^{(m+1)} := N(\vec{\varphi}_h^{(m)}).$$
- (γ) Evaluate $e := \|\mathbb{F}(\vec{\varphi}_h^{(m+1)})\|_{\ell^\infty}$ and proceed as follows:
  - (1) If $e \leqslant 10^{-14}$, we consider $\vec{\varphi}_h^{(m+1)}$ to be the solution of the nonlinear system with polynomials of degree ℓ. Then
    - (a) If $\ell = 1$, set $\vec{\varphi}_h^\star := \vec{\varphi}_h^{(m+1)}$ and **stop**.
    - (b) Otherwise, we consider $\vec{\varphi}_h^{(m+1)}$ as the initial guess for the Newton's method for solving the nonlinear system with polynomials of degree $\ell + 1$. Thus, we set $\vec{\varphi}_h^{(1)} := \mathbb{P}^{\ell+1}\vec{\varphi}_h^{(m+1)}$, $\epsilon := 2 \times 10^{-3}$, $\ell := \ell + 1$, $M := M(\vec{\varphi}_h^{(m+1)})$ and go to step (α).
  - (2) If $e > 10^{-14}$ and $m < 10$, set $m := m + 1$ and go to step (β) to compute another iterate.
  - (3) If $e > 10^{-14}$ and $m = 10$, then
    - (a) If $\boldsymbol{k}_{\mathrm{NT}} = \boldsymbol{k}_\ell$ and $\ell > 0$, we consider that the space $V_{h,\boldsymbol{k}_{\mathrm{NT}}}$ was not suitable for the computations; we change it and begin anew. Thus, we set $\boldsymbol{k}_{\mathrm{NT}} := \boldsymbol{k}_{\mathrm{RK}}^{(1)}$, $m := 1$ and go to step (β).
    - (b) Otherwise, we consider that the initial guess $\vec{\varphi}_h^{(1)}$ obtained by using the RKDG method in step (ii) is not suitable and we go back to that step to obtain a better guess. Thus, we set $\vec{\varphi}_h^{(0)} := \vec{\varphi}_h^{(1)}$, $\epsilon := \epsilon/2$, and go to step (ii).

Table 1 shows that the M approximated by our method is indeed very close to the exact value for the six qualitatively different test problems, see Tables 2 and 3.

Let us summarize the algorithm. For any given problem, we first estimate M by using Algorithm 3 with $\varphi_h^\star$ computed with Algorithm 4. Let us remind the reader that the grid we are using here is a uniform grid of 20 intervals and that we are working with piecewise linear approximations. Thus, the computation of M is extremely efficient. We can now apply Algorithm 1 to find the approximate solution of the DG method

Table 1
The approximate M and the exact M for 6 test problems

Problem	Approximate M	Exact M	Problem	Approximate M	Exact M
LS	120.987	124.025	CNS	0	0
CCS	120.927	124.025	NCNS	0	0
NCS	241.941	248.05	CNS2	0	0

Table 2
Smooth solution test problems

Problem	Hamiltonian $H(p)$	Right-hand side $f(x)$	Viscosity solution $u(x)$
LS	p (Linear)	$\cos^2(\pi x) - \pi\sin(2\pi x)$	$\cos^2(\pi x)$
CCS	$-\frac{p^2}{4\pi^2}$ (Concave)	$\cos^4(\pi x)$	$\cos^2(\pi x)$
NCS	$\frac{p^3}{8\pi^3}$ (Nonconvex)	$\sin(2\pi x) + \cos^3(2\pi\, x)$	$\sin(2\pi x)$

Table 3
Nonsmooth solution test problems

Problem	Hamiltonian $H(p)$	Right-hand side $f(x)$	Viscosity solution $u(x)$
CNS	$\frac{p^2}{\pi^2}$ (Convex)	$-\lvert\sin(\pi(x-\frac{\pi}{4}))\rvert + \cos^2(\pi(x-\frac{\pi}{4}))$	$-\lvert\sin(\pi(x-\frac{\pi}{4}))\rvert$
NCNS	$-p^4 + 3p^2 - 1$ (Nonconvex)	$u(x) + H(u'(x))$	$-\frac{\lvert\sin(\pi(x-\frac{\pi}{4}))\rvert}{\pi}$
CNS2	$\frac{p^2}{\pi^2}$ (Convex)	$-\frac{\lvert\sin(\pi(2x-\frac{\pi}{4}))\rvert}{2} + \cos^2(\pi(2x-\frac{\pi}{4}))$	$-\frac{\lvert\sin(\pi(2x-\frac{\pi}{4}))\rvert}{2}$

associated to any grid G_h and any polynomial degree k, where the slope limiter used by the RKDG method is given by Algorithm 2.

2.4. The a posteriori error estimate $\Phi_h(u_h)$

Here we describe our new a posteriori approximate error estimate $\Phi_h(u_h)$. This estimate is obtained by combining an a posteriori error estimate obtained by assuming that the viscosity solution is *smooth* with a *localized* version of an a posteriori error estimate for general viscosity solutions. To describe it, we proceed in several steps. We begin by introducing the error estimate for smooth exact solutions. Then we describe a localized version of the error estimate obtained in [1]. Finally, we combine these two estimates to obtain $\Phi_h(u_h)$.

2.4.1. The a posteriori error estimate $\Phi_h(u_h)$ when u is smooth

Assume that the exact solution u is a smooth function and set $e_u = u_h - u, e_{u_x} = u_{hx} - u_x$, and $e_{H(u_x)} = H(u_{hx}) - H(u_x)$; here u_{hx} denotes $(u_h)_x$. Then, since u is smooth, $u + H(u_x) - f = 0$, and we have that

$$\begin{aligned} R &= u_h + H(u_{hx}) - f \\ &= u_h + H(u_{hx}) - f - (u + H(u_x) - f) \\ &= (u_h - u) + (H(u_{hx}) - H(u_x)) \\ &= e_u + H'(u_{hx})(u_{hx} - u_x) + \Theta_{H(u_x)} \\ &= e_u + H'(u_{hx})(e_u)_x + \Theta_{H(u_x)}, \end{aligned}$$

where

$$\Theta_{H(u_x)} = H(u_{hx}) - H(u_x) - H'(u_{hx})(u_{hx} - u_x). \tag{2.6}$$

Now, since $\Theta_{H(u_x)} = \mathcal{O}((e_{u_x})^2)$, it is reasonable to assume that, when e_{u_x} is small enough,

$$H'(u_{hx})(e_u)_x + e_u - R \approx 0.$$

In this case, it is reasonable to take the estimate of the error, $\Phi_h(u_h)$, as a formally second-order accurate approximation to the solution $\Phi = \Phi(u_h)$ of

$$H'(u_{hx})\Phi_x + \Phi - R = 0,$$

that we compute as follows.

Algorithm 5. (*Computation of* $\Phi_h(u_h)$ for smooth u).

- Step 1. Given the grid $G_h = \{x_i\}_{i=0}^{N}$, compute the grid $\boldsymbol{G}_h = \{y_j\}_{j=1}^{nN}$ by
  $$y_{(i-1)*n+\ell} = x_{i-1} + (\ell-1)(x_i - x_{i-1})/(n-1),$$
  where $n = \mathsf{N}*(k+1)$, for $\ell = 1,\ldots,n$ and $i = 1,\ldots,N$. Then set
  $$h_j := x_i - x_{i-1} \quad \text{for } j = (i-1)*n+\ell \quad \text{and } \ell = 1,\ldots,n,$$
  $$R_j := R(y_j^-),$$
  $$H'_j := H'(u_{hx}(y_j^-)),$$
  $$a_{j-1/2} := \begin{cases} 0 & \text{if} \begin{cases} \frac{1}{2}|H'_j + H'_{j-1}| \leqslant h_j^{k+0.5}, \\ \text{or } |H'_j| \leqslant h_j^{k+0.5}, \\ \text{or } |H'_{j-1}| \leqslant h_j^{k+0.5}, \end{cases} \\ e^{-\frac{2(y_j - y_{j-1})}{H'_j + H'_{j-1}}} & \text{otherwise.} \end{cases}$$
  $$b_{j-1/2} := \frac{1}{2}(R_j + R_{j-1})(1 - a_{j-1/2}),$$
  for $j = 2,\ldots,nN$. Let $a_{j-1/2} := 1$ for $j = in+1$, any i. Here, y_j^- means the limit from inside of the element.
- Step 2. Set
  $$\Phi_h(0) = \frac{\sum_{m=2}^{nN}\left(\prod_{\ell=m+1}^{nN} a_{\ell-1/2}\right) b_{m-1/2}}{1 - \prod_{\ell=2}^{nN} a_{\ell-1/2}}.$$
- Step 3. Define $\Phi_h(y_j)$ by
  $$\Phi_h(y_j) = b_{j-1/2} + a_{j-1/2}\Phi_h(y_{j-1}),$$
  for $j = 2,\ldots,nN$.
- Step 4. Set
  $$a_{j-1/2} := \begin{cases} 0 & \text{if } a_{j-1/2} = 0, \\ 1/a_{j-1/2} & \text{otherwise.} \end{cases}$$
  $$b_{j-1/2} := \frac{1}{2}(R_j + R_{j-1})(1 - a_{j-1/2}),$$
  for $j = 2,\ldots,nN$.
- Step 5. If $\left|\frac{\sum_{m=2}^{nN}\left(\prod_{\ell=2}^{m-1} a_{\ell-1/2}\right) b_{m-1/2}}{1-\prod_{\ell=2}^{nN} a_{\ell-1/2}}\right| \leqslant |\Phi_h(1)|$, set
  $$\Phi_h(1) = \frac{\sum_{m=2}^{nN}\left(\prod_{\ell=2}^{m-1} a_{\ell-1/2}\right) b_{m-1/2}}{1 - \prod_{\ell=2}^{nN} a_{\ell-1/2}}.$$
- Step 6. If $|b_{j-1/2} + a_{j-1/2}\Phi_h(y_j)| \leqslant |\Phi_h(y_{j-1})|$, define $\Phi_h(y_{j-1})$ by
  $$\Phi_h(y_{j-1}) = b_{j-1/2} + a_{j-1/2}\Phi_h(y_j)$$
  for $j = nN,\ldots,2$.
- Step 7. Set $\Phi_h := 1.5\Phi_h$.

In all our numerical experiments, we take the number N to be equal to 15. Notice that, since we expect the error to be equal to zero at most $(k+1)$ times per element, we need to properly resolve those oscillations; this is why we compute it at $n = \mathsf{N} * (k+1)$ points in each interval.

Let us justify the fact that the function Φ_h defined in the second and third steps is in fact a second-order accurate approximation of the exact solution of the equation under consideration. The exact solution is

$$\Phi(x) = \mathrm{e}^{-\int_0^x \frac{1}{H'(u_{hx}(s))}\mathrm{d}s}\Phi(0) + \int_0^x \mathrm{e}^{-\int_s^x \frac{1}{H'(u_{hx}(t))}\mathrm{d}t}\,\frac{R(s)}{H'(u_{hx}(s))}\,\mathrm{d}s,$$

provided, for example, that $H'(u_{hx})$ is uniformly away from zero and that $1/H'(u_{hx})$ and R are integrable. As a consequence, the periodicity condition $\Phi(1) = \Phi(0)$ is fulfilled if and only if

$$\Phi(0) = \frac{\int_0^1 \mathrm{e}^{-\int_s^1 \frac{1}{H'(u_{hx}(t))}\mathrm{d}t}\,\frac{R(s)}{H'(u_{hx}(s))}\,\mathrm{d}s}{1 - \mathrm{e}^{-\int_0^1 \frac{1}{H'(u_{hx}(s))}\mathrm{d}s}}.$$

We then see that, indeed, the expression for $\Phi_h(0)$ given in the second step of the algorithm and $\Phi_h(1)$ in the fifth are second-order accurate approximations to this quantity.

We also know that the exact solution satisfies:

$$\Phi(y_j) = \mathrm{e}^{-\int_{y_{j-1}}^{y_j} \frac{1}{H'(u_{hx}(s))}\mathrm{d}s}\Phi(y_{j-1}) + \int_{y_{j-1}}^{y_j} \mathrm{e}^{-\int_s^x \frac{1}{H'(u_{hx}(t))}\mathrm{d}t}\,\frac{R(s)}{H'(u_{hx}(s))}\,\mathrm{d}s$$

and we see that the formulae given in the third and sixth steps do define a second-order approximation, as claimed.

It is also worth mentioning that the two-way sweeping performed in steps 2, 3 and steps 5, 6 gives more accurate results than any one-way sweeping. Obviously, these sweepings are exactly the same when none of the $a_{j-1/2}$'s is zero. Let us show how any single sweep will fail and combining them would succeed to give accurate error estimates by applying the algorithm on the following problem:

$$H(p) = -\frac{p^2}{16\pi^2}, \quad f(x) = \cos^4(2\pi x)$$

which has exact solution $\cos^2(2\pi x)$. We use DG($P4$) method to solve the problem on a uniform mesh of 40 elements. The results are displayed in Fig. 3. We see that, with one sweeping, the error estimate is too big when H' changes sign.

Finally, notice that in the last step, we are setting the approximate estimate of the error $\Phi_h(u_h)$ to be 1.5 times the function e_u. Notice that if e_u were exact, our error estimate $\Phi_h(u_h)$ would be 50% bigger than the optimal value and, as a consequence, the best effectivity index we can obtain is 1.5. However, in our numerical experiments, we see that effectivity indexes that are smaller than 1.5 are obtained. Without a doubt, this is due to the fact that e_u is the approximate solution of an equation that approximates the error equations. It is thus reasonable to multiply it by the factor 1.5 to compensate for these approximations and make sure that the error is actually smaller. Although we do not have a proof of this fact, the extensive numerical results we present, both with uniform and with nonuniform grids, show that we always achieve effectivity indexes that are never smaller than one. This justifies the last step of our algorithm.

2.4.2. The a posteriori error estimate $\Phi_h(u_h)$ when u is not smooth

In practice, when the exact solution u is smooth, the use of the above estimate produces very good results. However, its use cannot be justified whenever the viscosity solution u has kinks. To deal with this case, we introduce a localized version of the rigorous a posteriori error estimate obtained in [1]. The estimate *assumes* that the error is *known* at the border of an arbitrary subset Ω and gives an estimate of the error inside Ω. As the reader might be expecting, in our setting the set Ω is nothing but a small region surrounding the kinks of the viscosity solution. It is important to emphasize that the construction of such a region can be done extremely well by using monotone schemes, as shown in [4,5]. Since the DG methods we are considering automatically behave essentially like monotone schemes around the kinks (thanks to the use of the generalized slope limiter),

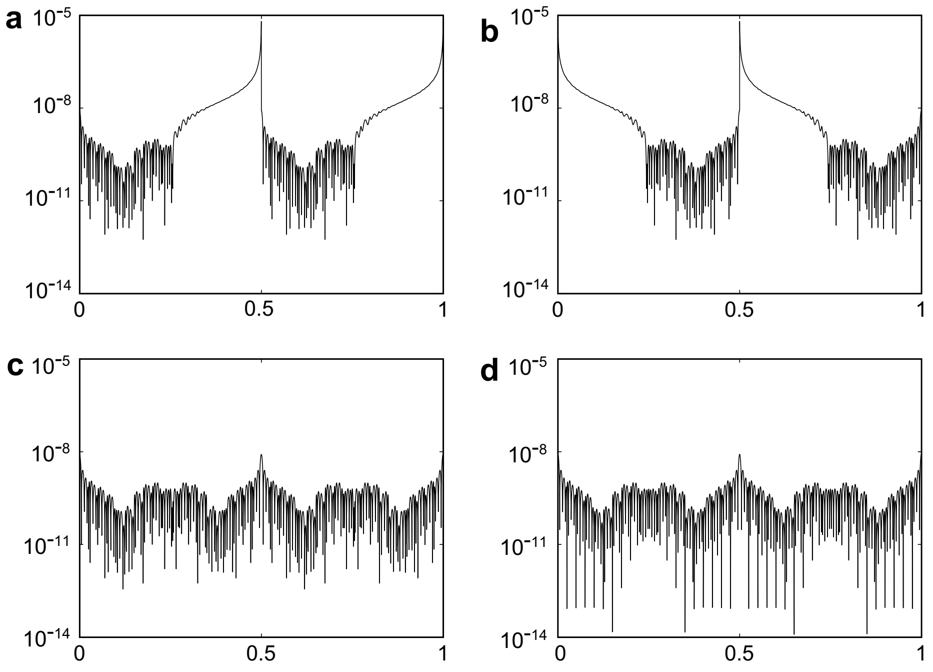

Fig. 3. (a) The error estimate after one sweeping from 0 to 1. (b) The error estimate after one sweeping from 1 to 0. (c) The error estimate after both sweepings as in the algorithm. (d) The exact error.

it is reasonable to believe that the construction of Ω should also be done as efficiently regardless of the degree of the polynomial approximation. Our numerical experiments show that this is actually the case.

Thus, given the set Ω and the error e_u at $\partial\Omega$, the estimate gives an upper bound for the following semi-norms:

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

where $w^+ \equiv \max\{0,w\}$. To state the estimate, we need to introduce the *shifted residual*,

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

and the paraboloid P_v,

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

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

We are now ready to describe the a posteriori error estimate.

Theorem 2.2 (A posteriori error estimate). *Let u be the viscosity solution of Eq.* (1.1) *and let v be any continuous function on* $\mathbb{R}$ *periodic with period* 1. *Let* Ω *be a closed subset of* $\mathbb{R}$ *and suppose the error* $e_u(x) \equiv u_h(x) - u(x)$ *is given on* $\partial\Omega$. *Then we have that*

$$\psi_{h\Omega} := \max_{\sigma\in\{-,+\}} |u-v|_{\sigma,\Omega} \leqslant \max_{\sigma\in\{-,+\}} \inf_{\epsilon\geqslant 0} \max\{\Psi_\sigma(v;\epsilon), \Phi_\sigma(v;\epsilon)\},$$

where

$$\Psi_\sigma(v;\epsilon) = \sup_{x\in\partial\Omega,(x+\sigma\epsilon p,p)\in A_\sigma(v;\epsilon)} \sigma[v(x+\sigma\epsilon p) - v(x) + e_u(x)] - \frac{1}{2}\epsilon|p|^2,$$

$$\Phi_\sigma(v;\epsilon) = \sup_{(x,p)\in A_\sigma(v;\epsilon)} (\sigma R_{\sigma\epsilon}(v;x,p))^+.$$

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

$$(x,p) \in (\Omega \setminus \partial\Omega, D^\sigma v(x)) \cup (\partial\Omega, B_\sigma(v,x)), \tag{2.7}$$

$$\sigma\{v(y) - P_v(x,p,\sigma/\epsilon;y)\} \leqslant 0 \quad \forall y \in \mathbb{R}, \tag{2.8}$$

where $B_\sigma(v,x) = \left\{p|\sigma p \cdot n(x) \leqslant \sigma \frac{\mathrm{d}}{\mathrm{d}x}^- v(x) \cdot n(x)\right\}$, and $n(x)$ is the unit outward normal vector of $\partial\Omega$ at x, $\frac{\mathrm{d}}{\mathrm{d}x}^- v(x)$ is the gradient of $v(x)$ at x from inside of Ω.

Proof. We prove the result for $\sigma = -$; the proof for the case $\sigma = +$ is similar. Given $\epsilon > 0$, define the auxiliary function

$$\psi(x,y) = u(x) - v(y) - \frac{|x-y|^2}{2\epsilon},$$

and let $(\hat{x},\hat{y}) \in \Omega \times \Omega$ be such that

$$\psi(\hat{x},\hat{y}) \geqslant \psi(x,y) \quad \forall x,y \in \Omega.$$

Such a point exists since Ω is compact and ψ is continuous. Set $\hat{p} = \frac{\hat{x}-\hat{y}}{\epsilon}$.

We assume $|u-v|_{-,\Omega} > 0$, otherwise the result is trivial. In this case, we have

$$\begin{aligned}
|u-v|_{-,\Omega} &= \sup_{x\in\Omega}\{u(x) - v(x)\} \\
&= \sup_{x\in\Omega}\psi(x,x) \\
&\leqslant \sup_{x,y\in\Omega} \psi(x,y) \\
&= \psi(\hat{x},\hat{y}) = u(\hat{x}) - v(\hat{y}) - \frac{|\hat{x}-\hat{y}|^2}{2\epsilon}.
\end{aligned}$$

First, we characterize $(\hat{y},\hat{p})$. Since $\psi(\hat{x},\hat{y}) \geqslant \psi(\hat{x},y)$ for all $y \in \Omega$, we have that

$$\begin{aligned}
v(y) &\geqslant v(\hat{y}) + \frac{|\hat{x}-\hat{y}|^2}{2\epsilon} - \frac{|\hat{x}-y|^2}{2\epsilon} \\
&= v(\hat{y}) + \hat{p}\cdot(y-\hat{y}) - \frac{|y-\hat{y}|^2}{2\epsilon}. \\
&= P_v(\hat{y},\hat{p},-\frac{1}{\epsilon};y).
\end{aligned}$$

Note that this implies that $\hat{p} \in D^- v(\hat{y})$ when $\hat{y} \in \Omega\backslash\partial\Omega$, and $-\hat{p}\cdot n(x) \leqslant -q\cdot n(x)$ for any $q = \frac{\mathrm{d}}{\mathrm{d}x}^- v(\hat{y})$ when $\hat{y} \in \partial\Omega$.

We use $A_-(v;\epsilon)$ to denote the set of elements (x,p) satisfying

$$(x,p) \in (\Omega \setminus \partial\Omega, D^- v(x)) \cup (\partial\Omega, B_-(v,x)) \text{ and } -\{v(y) - P_v(x,p,\sigma/\epsilon;y)\} \leqslant 0 \quad \forall y \in \mathbb{R},$$

where $B_-(v,x) = \{p| -p\cdot n(x) \leqslant -\frac{\mathrm{d}}{\mathrm{d}x}^- v(x)\cdot n(x)\}$, and $n(x)$ is the unit outward normal vector of $\partial\Omega$ at x, $\frac{\mathrm{d}}{\mathrm{d}x}^- v(x)$ is the gradient of $v(x)$ at x from inside of Ω.

We proceed by considering the following two cases.

Case 1. $\hat{x} \in \partial\Omega$:

$$\psi(\hat{x},\hat{y}) = -e_u(\hat{x}) + v(\hat{x}) - v(\hat{x}-\epsilon\hat{p}) - \frac{1}{2}\epsilon|\hat{p}|^2.$$

We thus have

$$|u-v|_{-,\Omega} \leqslant \psi(\hat{x},\hat{y}) \leqslant \sup_{x\in\partial\Omega,(x-\epsilon p,p)\in A_-(v;\epsilon)} -[v(x-\epsilon p) - v(x) + e_u(x)] - \frac{1}{2}\epsilon|p|^2.$$

Case 2. $\hat{x} \in \Omega \setminus \partial\Omega$:

$$\psi(\hat{x},\hat{y}) = [u(\hat{x}) + H(\hat{p}) - f(\hat{x})] - \left[v(\hat{y}) + H(\hat{p}) - f(\hat{y}) + f(\hat{y}) - f(\hat{x}) + \frac{|\hat{x}-\hat{y}|^2}{2\epsilon} \right] = R(u;\hat{x},\hat{p}) - R_{-\epsilon}(v;\hat{y},\hat{p}).$$

Since the mapping $x \mapsto \psi(x,\hat{y})$ has a maximum at $x = \hat{x}$, we have that

$$0 \in D_x^+\psi(\hat{x},\hat{y}) = D^+u(\hat{x}) - \hat{p},$$

and so $\hat{p} \in D^+u(\hat{x})$. Since u is the viscosity solution, this implies that $R(u;\hat{x},\hat{p}) \leqslant 0$ and hence

$$\psi(\hat{x},\hat{y}) \leqslant -R_{-\epsilon}(v;\hat{y},\hat{p}).$$

We thus have $|u-v|_{-,\Omega} \leqslant \sup_{(y,p)\in A_-(v;\epsilon)}\{-R_{-\epsilon}(v;y,p)\}$. Since these results are true for any ϵ, this completes the proof. □

To apply Theorem 2.2, we take v to be u_h and discretize the nonlinear functionals of the a posteriori error estimate. The set $\Omega = [s,e]$ over which we evaluate the functionals $\Phi_\sigma(\cdot,\cdot)$ and $\Psi_\sigma(\cdot,\cdot)$, is replaced by the following finite number of points:

$$\Omega_h := \{s,e\} \cup \{c_i, 0 \leqslant i < N_{[s,e]}\},$$

where c_i's are the midpoints of the intervals contained in the set $[s,e]$ and $N_{[s,e]}$ is the number of intervals therein.

Another modification to be made in practice is on the values of the auxiliary parameter ϵ. The evaluation of the functionals in the theorem requires optimization over the set $\epsilon \in [0,\infty)$. However, we replace $[0,\infty)$ in practice by the set

$$\varepsilon_h = \left\{ i \cdot \frac{E}{N_E}, \quad 0 \leqslant i \leqslant N_E \right\},$$

where $E = 50\omega|\ln(1/\omega)|$ and $N_E = 100|\ln(1/\omega)|$.

Here ω is an upper bound for the artificial diffusion coefficient of the numerical scheme under consideration. We take $\omega = Ch_{\min}$ where C is as defined by (2.1c) and $h_{\min} = \mathrm{Min}\{h_0,\ldots,h_{N-1}\}$.

As to the motivation of these discretizations and the fast evaluation of the paraboloid test, that is, the evaluation of (2.8), we refer to [1].

2.4.3. *The a posteriori error estimate $\Phi_h(u_h)$*

We define our a posteriori error estimate by means of the following algorithm:

Algorithm 6 (*Computation of $\Phi_h(u_h)$*).

- Step 1. Given the approximate solution u_h, Algorithm 1 splits the interval $[0,1]$ as the union of two disjoint sets B_{u_h} and $A_{u_h} = [0,1] - B_{u_h}$ with $B_{u_h} = \cup_{i=1}^{\ell}[s_i,e_i]$ where $P0$-elements are used on $[s_i,e_i]$.
- Step 2. Compute $a_{j-1/2}$ and $b_{j-1/2}$ as in Algorithm 5.
- Step 3. Set $a_{j-1/2} = 1$ and $b_{j-1/2} = 0$ on the B_{u_h}.
- Step 4. Compute $\Phi_h(y_j)$ by using Algorithm 5.
- Step 5. Set $\Phi_h := 1.5\Phi_h$.
- Step 6. On each $[s_i,e_i]$, $i = 1,\ldots,\ell$ in B_{u_h}, compute $\psi_{h[s_i,e_i]}$ given by Theorem 2.2.
- Step 7. Set

$$\Phi_h(u_h)(x) := \begin{cases} \Phi_h(u_h)(x) & \text{if } x \in A_{u_h}, \\ \psi_{h[s_i,e_i]} & \text{if } x \in [s_i,e_i] \subset B_{u_h}. \end{cases}$$

In practice, when the viscosity solution u is smooth, the generalized slope limiter is not used and the set B_{u_h} is empty. This implies that the a posteriori error estimate computed by using Algorithm 6 is exactly the one given by Algorithm 5. On the other hand, if the viscosity solution has kinks and the approximation is close enough to it, the slope limiter will be turned on and the set B_{u_h} will not be empty anymore. In this case, Step 3 will force the equality $e_u(s_i) = e_u(e_i)$ and Step 7 will take the a posteriori error estimate on $[s_i, e_i]$ to be the number $\psi_{[s_i,e_i]}$ given by Theorem 2.2. Forcing such equality is somewhat arbitrary, but we feel that the fact that both quantities should be roughly of the same order justifies this choice. Our numerical experiments show that the method works well.

2.5. Computing a new grid

Next, we present the technique proposed in [4] to compute a mesh and show how to adapt it to our case.

To describe the way we compute a new grid, we need to introduce an auxiliary mapping that associates a grid of the domain $[0, 1]$ to any given bounded and integrable function $\Gamma : [0, 1] \to \mathbb{R}^+$. It is defined as follows. First, we solve the equation

$$\frac{\mathrm{d}}{\mathrm{d}x}\mathcal{N} = \Gamma \quad \text{in } (0,1) \quad \text{with } \mathcal{N}(0) = 0. \tag{2.9}$$

Then, we set

$$\mathcal{G}(\Gamma) = \left\{ x_j = \mathcal{N}^{-1}\left(j \frac{\mathcal{N}(1)}{n} \right) \right\}_{j=0}^{n}, \tag{2.10}$$

where n is the smallest natural number no less than $\mathcal{N}(1)$; this guarantees that $x_n = 1$. Let us point out that in [4], the mapping is defined as

$$\mathcal{G}(\Gamma) = \left\{ x_j = \mathcal{N}^{-1}(j) \right\}_{j=0}^{n},$$

tacitly assuming that if $n > \mathcal{N}(1)$, then $x_n = 1$. With this definition, an extremely small interval $(x_{n-1}, 1)$ can be created which, in our setting, has undesirable consequences.

The modification of a given grid of mesh-size function h is the grid $\mathcal{G}(\Gamma = \mu/h)$, where the function μ is the so-called *grid-size modification function*. We take

$$\mu := \Psi(\gamma),$$

where, for $s \in [x_{i-1}, x_i]$.

$$\gamma(s) = \left(\frac{\| \Phi_h(u_h)(s) \|_{L^\infty(\{y_j\}_{y_j \in [x_{i-1}, x_i]})}}{\tau} \right)^{\frac{1}{\mathcal{O}(s)}},$$

$$\Psi(x) = \begin{cases} x, & \text{if } x \geqslant 1.4, \\ 1 + 0.1\sqrt[4]{10x - 10}, & \text{if } 1 < x < 1.4, \\ \frac{x+3}{4}, & \text{if } 0 \leqslant x \leqslant 1. \end{cases}$$

Moreover, we set

$$\mu := \max\{1, \Psi(\gamma)\},$$

whenever we consider that our grid is *reasonable*, i.e., whenever we have that

$$\| (1 - \Psi(\gamma))^+ \|_{L^1(0,1)} \leqslant 0.035.$$

Let us briefly mention that whenever μ is bigger or smaller than one, the modified grid will have smaller or bigger elements, respectively, than those of the original grid. A detailed discussion of a very similar choice is provided in [4]. The main difference between what was done in [4] and what we do here is the definition of $\gamma(s)$. Here, to take into account that the numerical scheme converges formally, at s, with order $\mathcal{O}(s)$ that is not necessarily 1, we take the absolute value of the ratio of the a posteriori error estimate to the tolerance to the power $1/\mathcal{O}(s)$. Obviously, $\mathcal{O}(s) = 1$ in [4]. In our numerical examples, we set

$$\mathcal{O}(s) = \begin{cases} k + 2, & \text{if } s \in A_{u_h} \\ k + 2 - \min\left\{1, \frac{\text{step}}{6}\right\} \cdot (k+1), & \text{Otherwise.} \end{cases}$$

Where *step* is the current number of step of the adaptive method. We see that, close to the kink, $\mathcal{O}(s)$ is a number between $k+2$ and 1 that goes to 1 gradually as we identify the kink better and better during our adaptive procedure. This particular choice can avoid overrefinement during the first several steps when the position of the kinks has not been well recognized.

2.6. Summary of the method

To end this section, we give a precise description of our adaptive method. We are given the tolerance τ and the polynomial degree of the approximation for Φ_h, k.

Algorithm 7 (*Adaptive algorithm*).

- Step 1. Construct the initial grid $G_h = \{j/20\}_{j=0}^{20}$.
- Step 2. Compute the DG approximation u_h on the grid G_h.
- Step 3. Compute the estimate of the error $\Phi_h(u_h)$.
- Step 4. If $\|\Phi_h(u_h)\|_{L^\infty} \leqslant \tau$, **stop**.
- Step 5. Compute the new grid $G_h = \mathcal{G}(h/\mu)$ and go to Step 2.

In step 2, we initialize the DG approximation u_h by zero when computing it on the grid G_h in Algorithm 1. In fact, we only need to do it on the initial grid. On the following grids, we can initialize it by the Π^0-projection of the solution on the previous mesh.

The projection Π^0, from $V_{h,\boldsymbol{k}_\ell}(G_h^1)$ with $G_h^1 = \{x_i^1\}_{i=0}^{N_1}$ to $V_{h,\boldsymbol{k}_0}(G_h^2)$ with $G_h^2 = \{x_i^2\}_{i=0}^{N_2}$, is defined as follows: for any function $\phi \in V_{h,\boldsymbol{k}_\ell}(G_h^1)$, we define $\Pi^0\phi \in V_{h,\boldsymbol{k}_0}(G_h^2)$ as, for any $j = 0, \ldots, N_2 - 1$,

$$\Pi^0\phi|_{\boldsymbol{I}_j} = \frac{1}{|\Omega_j|}\int_{\Omega_j} \phi \mathrm{d}x,$$

where Ω_j is the smallest union of elements in G_h^1 that contains $I_j = (x_j^2, x_{j+1}^2)$.

3. Numerical results

3.1. The test problems

We test our adaptive method on problems similar to those used in [4] for the study of the first-order adaptive method; see Tables 2 and 3. The solutions of the problems displayed in Table 2 are smooth, and the solutions of the problems in Table 3 have kinks at $x = \pi/8$, $x = \pi/4$ or $x = \pi/8 + 0.5$.

3.2. The effectivity index of the a posteriori error estimate

We display the behavior of the discrete effectivity index

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

as we refine the grid uniformly in Tables 4 and 5. We see that, independently of the polynomial degree of the approximation or the problem, the effectivity index remains remarkably constant and fairly close to the best value of 1.5 when the solution is smooth. We can thus consider that the estimate is reliable.

It is interesting to note that the approximate solutions converges exponentially to the exact solution, see Fig. 4.

3.3. Results with the adaptive method

Here we display and discuss the performance of the adaptive method. In Tables 6–11, we show the history of convergence together with

$$\mathrm{ei}_\tau(\tau;u,u_h)=\frac{\tau}{\|u-u_h\|_{L^\infty(\boldsymbol{G}_h)}},\quad \mathrm{ei}_{ad}(\tau;u_h)=\frac{\tau}{\Phi_h(u_h)},$$

the numbers of steps needed for convergence and,

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

where n^k is the number of intervals of the grid of the step k and K is the number of steps needed for convergence. We call this quantity the *complexity ratio* since the complexity of each step is proportional to the number of elements of the grid; it is a measure of the efficiency of the adaptive method.

In Figs. 5–10, we plot $\log_{10}h$ as a measure of how the grid points are distributed on the domain, $\frac{\text{error}}{\tau}$ and the mesh modification function $\mu=\Psi(\gamma)$. In these figures, by the error at the point x we mean

$$\|u-u_h\|_{L^\infty(\{y_j\}_{y_j\in[x_{i-1},x_i]})},$$

whenever $x\in(x_{i-1},x_i)$. If we plot the actual error at each point $y_j\in\boldsymbol{G}_h$, we would see a highly oscillatory function, of course.

From the above mentioned tables and figures we see the following:

- The method enforces a strict control on the error $\|u-u_h\|_{L^\infty(\boldsymbol{G}_h)}$ since the effectivity index $\mathrm{ei}_\tau(\tau;u,u_h)$ is always bigger than one. Moreover, error/τ (see the second column of the figures) stays around 0.5 in most of the domain.
- The effectivity index $\mathrm{ei}_{ad}(\tau;u_h)$, which is a measure of the quality of the adaptive method, is very close to the ideal value of one, independently of the huge variations in the value of the tolerance. This means that the method is reliable.
- The complexity remains of order one uniformly in the size of the tolerance and the polynomial degree. This means that the method has optimal complexity.

Table 4
History of convergence for the smooth solution test problems LS, CCS and NCS with uniform meshes

Deg	$\frac{1}{\Delta x}$	LS			CCS			NCS		
		$\|u-u_h\|$	ei_h	Order	$\|u-u_h\|$	ei_h	Order	$\|u-u_h\|$	ei_h	Order
1	20	3.77e−04	1.50	–	3.15e−04	1.95	–	8.65e−04	1.45	–
	40	4.73e−05	1.50	3.00	4.00e−05	1.78	2.98	1.09e−04	1.49	2.99
	80	5.92e−06	1.51	3.00	4.99e−06	1.85	3.00	1.37e−05	1.52	3.00
	160	7.40e−07	1.50	3.00	6.23e−07	2.08	3.00	1.71e−06	1.57	3.00
	320	9.25e−08	1.53	3.00	7.79e−08	2.44	3.00	2.13e−07	1.67	3.00
	640	1.16e−08	1.63	3.00	9.73e−09	3.29	3.00	2.67e−08	1.86	3.00
	1280	1.45e−09	1.68	3.00	1.22e−09	3.97	3.00	3.33e−09	2.25	3.00
2	20	5.39e−06	1.50	–	3.61e−05	1.50	–	5.25e−05	1.50	–
	40	3.41e−07	1.51	3.98	3.59e−06	1.50	3.33	5.09e−06	1.50	3.37
	80	2.14e−08	1.51	3.99	3.50e−07	1.50	3.36	4.88e−07	1.50	3.38
	160	1.34e−09	1.53	4.00	3.36e−08	1.50	3.38	3.79e−08	1.50	3.69
	320	8.40e−11	1.57	4.00	3.20e−09	1.50	3.39	3.25e−09	1.50	3.54
	640	5.54e−12	1.62	3.92	3.05e−10	1.50	3.39	2.75e−10	1.51	3.56
3	20	7.27e−08	1.49	–	7.21e−08	3.45	–	1.42e−07	1.53	–
	40	2.29e−09	1.48	4.99	2.28e−09	3.98	4.98	4.56e−09	1.52	4.96
	80	7.18e−11	1.48	5.00	7.17e−11	4.10	4.99	1.43e−10	1.59	4.99
	160	2.30e−12	1.52	4.96	2.25e−12	4.50	5.00	4.50e−12	1.76	4.99
4	20	9.35e−10	1.49	–	8.28e−09	1.48	–	1.52e−08	1.50	–
	40	1.48e−11	1.48	5.98	2.16e−10	1.48	5.26	3.40e−10	1.50	5.48
	80	2.59e−13	1.41	5.84	5.59e−12	1.48	5.27	7.48e−12	1.50	5.50

Here $\|u-u_h\|:=\|u-u_h\|_{L^\infty(\boldsymbol{G}_h)}$.

Table 5
History of convergence for the nonsmooth solution test problems CNS, NCNS and CNS2 with uniform meshes

Deg	$\frac{1}{\Delta x}$	CNS			NCNS			CNS2		
		$\|u-u_h\|$	ei$_h$	Order	$\|u-u_h\|$	ei$_h$	Order	$\|u-u_h\|$	ei$_h$	Order
1	20	6.43e−02	2.01	–	2.20e−02	4.96	–	3.81e−02	5.98	–
	40	3.71e−02	1.99	0.79	1.26e−02	3.41	0.80	3.21e−02	2.43	0.25
	80	1.09e−02	2.79	1.77	5.35e−03	5.91	1.24	1.85e−02	1.99	0.79
	160	8.70e−03	1.55	0.33	3.21e−03	4.52	0.74	5.45e−03	2.85	1.77
	320	4.29e−03	1.90	1.02	1.60e−03	5.36	1.01	4.35e−03	1.95	0.33
	640	2.15e−03	2.34	1.00	7.92e−04	4.96	1.01	2.15e−03	3.95	1.02
	1280	1.02e−03	4.03	1.08	3.88e−04	4.87	1.03	1.07e−03	2.39	1.00
2	20	9.59e−02	10.6	–	2.21e−02	4.93	–	7.80e−02	2.97	–
	40	3.76e−02	1.97	1.35	1.28e−02	3.37	0.79	4.87e−02	20.2	0.68
	80	2.32e−02	1.95	0.69	5.30e−03	5.97	1.27	1.88e−02	1.97	1.37
	160	8.72e−03	1.25	1.41	3.21e−03	4.51	0.72	1.17e−02	2.20	0.68
	320	4.24e−03	1.22	1.04	1.58e−03	5.42	1.02	4.36e−03	1.37	1.42
	640	2.19e−03	1.15	0.95	8.07e−04	4.87	0.97	2.12e−03	1.28	1.04
	1280	1.59e−03	2.35	0.47	3.93e−04	4.80	1.04	1.10e−03	2.08	0.95
3	20	6.43e−02	2.22	–	2.19e−02	4.96	–	7.34e−02	3.37	–
	40	3.82e−02	1.93	0.75	1.30e−02	3.32	0.76	3.21e−02	2.46	1.19
	80	2.34e−02	2.07	0.71	5.35e−03	5.91	1.28	1.91e−02	1.93	0.75
	160	8.72e−03	2.39	1.42	3.21e−03	4.52	0.74	5.46e−03	3.19	1.81
	320	4.30e−03	2.39	1.02	1.60e−03	5.36	1.01	4.36e−03	3.86	0.33
	640	2.21e−03	2.27	0.96	8.10e−04	4.85	0.98	2.15e−03	2.50	1.02
	1280	1.04e−03	2.39	1.09	3.94e−04	4.79	1.04	1.11e−03	2.33	0.96
4	20	6.45e−02	2.22	–	2.20e−02	4.94	–	7.81e−02	5.84	–
	40	5.08e−02	1.18	0.35	1.29e−02	3.33	0.77	4.78e−02	3.07	0.71
	80	1.09e−02	2.14	2.22	5.33e−03	5.93	1.28	2.58e−02	2.31	0.89
	160	8.78e−03	2.36	0.31	3.22e−03	4.49	0.73	5.45e−03	2.48	2.24
	320	4.30e−03	3.94	1.03	1.60e−03	5.36	1.01	4.39e−03	3.84	0.31
	640	2.21e−03	1.16	0.96	8.08e−04	4.86	0.98	2.15e−03	3.94	1.03
	1280	1.05e−03	2.37	1.07	3.97e−04	4.75	1.02	1.10e−03	1.21	0.96

Here $\|u-u_h\| := \|u-u_h\|_{L^\infty(G_h)}$.

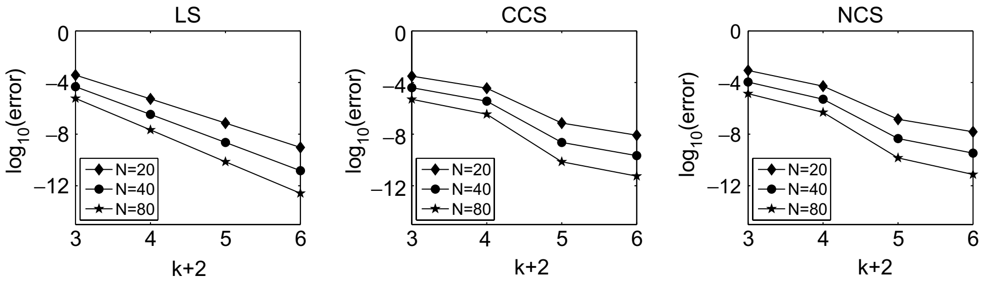

Fig. 4. Exponential convergence of the method on uniform grid.

- The first column of the figures shows that the grid points have a similar distribution on the domain independently of the tolerance.
- The mesh modification function $\mu = \Psi(\gamma)$ is very close to be identically equal to 1.

Therefore, we conclude that the adaptive method is reliable and extremely efficient uniformly in the polynomial degree k, which varied from 1 to 4, and in the tolerance τ, which varied from 10^{-1} to 10^{-11} in the tables and from 10^{-5} to 10^{-10} in the figures.

Table 6
History of convergence of the adaptive method for the problem LS

Prob	Deg	τ	n	$\|u-u_h\|$	Order	$\mathrm{ei}_\tau(\tau;u,v)$	$\mathrm{ei}_{ad}(\tau;v)$	Steps	Complxr
LS	1	1.0e−04	32	6.47e−05	–	1.55	1.04	3	2.56
		1.0e−05	72	6.05e−06	2.92	1.65	1.10	4	3.14
		1.0e−06	151	6.71e−07	2.97	1.49	1.01	3	2.03
		1.0e−07	329	7.91e−08	2.75	1.27	1.01	4	2.94
		1.0e−08	708	8.96e−09	2.84	1.12	1.00	5	3.88
	2	1.0e−05	20	5.39e−06	–	1.86	1.24	1	1.00
		1.0e−06	31	6.44e−07	4.85	1.55	1.05	2	1.65
		1.0e−07	56	6.13e−08	3.98	1.63	1.08	3	2.30
		1.0e−08	102	6.72e−09	3.69	1.49	1.00	3	2.11
		1.0e−09	185	6.94e−10	3.81	1.44	1.00	3	2.00
		1.0e−10	334	7.10e−11	3.86	1.41	1.01	4	2.93
		1.0e−11	613	7.16e−12	3.78	1.40	1.02	5	3.87
	3	1.0e−07	21	6.09e−08	–	1.64	1.09	2	1.95
		1.0e−08	30	6.63e−09	6.22	1.51	1.00	2	1.67
		1.0e−09	50	6.70e−10	4.49	1.49	1.01	3	2.32
		1.0e−10	81	6.68e−11	4.78	1.50	1.01	3	2.15
		1.0e−11	133	6.50e−12	4.70	1.54	1.03	3	2.02
	4	1.0e−08	20	9.35e−10	–	10.7	7.20	1	1.00
		1.0e−09	23	5.05e−10	4.40	1.98	1.35	2	1.87
		1.0e−10	31	5.68e−11	7.32	1.76	1.19	3	2.58
		1.0e−11	43	6.63e−12	6.57	1.51	1.09	3	2.44

Here $\|u-u_h\| := \|u-u_h\|_{L^\infty(\boldsymbol{G}_h)}$.

Table 7
History of convergence of the adaptive method for problem CCS

Prob	Deg	τ	n	$\|u-u_h\|$	Order	$\mathrm{ei}_\tau(\tau;u,v)$	$\mathrm{ei}_{ad}(\tau;v)$	Steps	Complxr
CCS	1	1.0e−04	31	6.61e−05	–	1.51	1.02	3	2.61
		1.0e−05	64	6.28e−06	3.25	1.59	1.04	4	3.31
		1.0e−06	142	6.10e−07	2.93	1.64	1.04	4	3.11
		1.0e−07	300	6.08e−08	3.08	1.64	1.02	5	4.03
		1.0e−08	669	6.33e−09	2.82	1.58	1.02	9	8.02
	2	1.0e−05	22	6.22e−06	–	1.61	1.07	4	3.91
		1.0e−06	39	5.63e−07	4.19	1.78	1.19	5	4.31
		1.0e−07	70	5.89e−08	3.86	1.70	1.14	6	5.04
		1.0e−08	124	6.20e−09	3.94	1.61	1.08	7	5.92
		1.0e−09	223	6.23e−10	3.92	1.61	1.08	8	6.84
		1.0e−10	393	6.40e−11	4.01	1.56	1.05	10	8.91
		1.0e−11	713	6.29e−12	3.90	1.59	1.06	9	7.80
	3	1.0e−07	22	4.87e−08	–	2.05	1.39	3	2.91
		1.0e−08	31	6.19e−09	6.02	1.62	1.12	2	1.65
		1.0e−09	50	5.80e−10	4.95	1.73	1.15	5	4.28
		1.0e−10	80	6.51e−11	4.65	1.54	1.02	4	3.16
		1.0e−11	127	6.57e−12	4.96	1.52	1.01	4	3.07
	4	1.0e−08	19	5.88e−09	–	1.70	1.15	2	2.05
		1.0e−09	25	5.92e−10	8.37	1.69	1.14	4	3.68
		1.0e−10	37	5.50e−11	6.06	1.82	1.23	6	5.22
		1.0e−11	53	5.73e−12	6.29	1.75	1.18	6	5.09

Here $\|u-u_h\| := \|u-u_h\|_{L^\infty(\boldsymbol{G}_h)}$.

Table 8
History of convergence of the adaptive method for problem NCS

Prob	Deg	τ	n	$\|u - u_h\|$	Order	$\mathrm{ei}_{\tau}(\tau; u, v)$	$\mathrm{ei}_{ad}(\tau; v)$	Steps	Complxr
NCS	1	1.0e−04	39	7.16e−05	–	1.40	1.11	4	3.41
		1.0e−05	86	6.94e−06	2.95	1.44	1.11	4	3.08
		1.0e−06	179	7.41e−07	3.05	1.35	1.05	4	3.02
		1.0e−07	385	7.06e−08	3.07	1.42	1.05	7	6.05
		1.0e−08	904	6.26e−09	2.84	1.60	1.02	7	5.82
	2	1.0e−05	28	6.56e−06	–	1.52	1.02	5	4.68
		1.0e−06	50	6.04e−07	4.11	1.66	1.11	4	3.26
		1.0e−07	89	6.57e−08	3.85	1.52	1.02	6	5.07
		1.0e−08	162	6.52e−09	3.86	1.53	1.02	4	2.96
		1.0e−09	293	6.45e−10	3.90	1.55	1.02	4	2.89
		1.0e−10	536	6.27e−11	3.86	1.60	1.02	4	2.82
		1.0e−11	992	6.21e−12	3.76	1.61	1.03	4	2.75
	3	1.0e−07	25	5.73e−08	–	1.74	1.17	4	3.76
		1.0e−08	34	6.07e−09	7.30	1.65	1.10	3	2.56
		1.0e−09	56	5.98e−10	4.65	1.67	1.12	4	3.25
		1.0e−10	91	6.63e−11	4.53	1.51	1.01	4	3.09
		1.0e−11	149	5.26e−12	5.14	1.90	1.16	5	3.96
	4	1.0e−08	21	6.19e−09	–	1.62	1.08	3	2.95
		1.0e−09	29	6.36e−10	7.05	1.57	1.05	5	4.62
		1.0e−10	43	5.51e−11	6.21	1.82	1.21	6	5.19
		1.0e−11	61	6.49e−12	6.12	1.54	1.03	4	3.20

Here $\|u - u_h\| := \|u - u_h\|_{L^{\infty}(\boldsymbol{G}_h)}$.

Table 9
History of convergence of the adaptive method for problem CNS

Prob	Deg	τ	n	$\|u - u_h\|$	Order	$\mathrm{ei}_{\tau}(\tau; u, v)$	$\mathrm{ei}_{ad}(\tau; v)$	Steps	Complxr
CNS	1	1.0e−01	17	4.78e−03	–	20.9	1.25	2	2.18
		1.0e−02	20	2.68e−03	3.55	3.72	1.07	5	5.25
		1.0e−03	39	2.60e−04	3.50	3.85	1.08	5	4.33
		1.0e−04	96	7.13e−05	1.43	1.40	1.12	6	4.58
		1.0e−05	224	6.16e−06	2.89	1.62	1.09	11	9.03
	2	1.0e−01	17	1.94e−02	–	5.15	1.98	3	3.29
		1.0e−02	20	5.78e−04	21.6	17.3	1.10	5	5.50
		1.0e−03	42	7.24e−04	−0.30	1.38	1.29	5	4.14
		1.0e−04	73	5.00e−05	4.84	2.00	1.01	7	5.89
		1.0e−05	197	8.70e−06	1.76	1.15	1.00	6	4.50
	3	1.0e−01	17	2.71e−03	–	36.9	1.61	2	2.18
		1.0e−02	14	3.96e−03	1.96	2.52	1.17	8	9.64
		1.0e−03	32	2.54e−04	3.32	3.93	1.73	9	7.41
		1.0e−04	44	3.80e−05	5.97	2.63	1.19	9	8.45
		1.0e−05	112	4.10e−06	2.38	2.44	2.18	7	4.64
	4	1.0e−01	17	4.17e−02	–	2.40	1.39	3	3.24
		1.0e−02	21	1.57e−03	15.5	6.38	3.06	8	7.48
		1.0e−03	37	2.76e−04	3.07	3.62	1.07	7	5.68
		1.0e−04	53	2.80e−05	6.36	3.57	1.01	7	5.13
		1.0e−05	127	3.37e−06	2.43	2.97	1.10	7	3.94

Here $\|u - u_h\| := \|u - u_h\|_{L^{\infty}(\boldsymbol{G}_h)}$.

Table 10
History of convergence of the adaptive method for problem NCNS

Prob	Deg	τ	n	$\|u-u_h\|$	Order	$\mathrm{ei}_\tau(\tau;u,v)$	$\mathrm{ei}_{ad}(\tau;v)$	Steps	Complxr
NCNS	1	1.0e−01	19	1.18e−02	–	8.48	1.07	2	2.05
		1.0e−02	29	2.53e−03	3.64	3.95	1.21	6	5.69
		1.0e−03	70	2.21e−04	2.77	4.52	1.38	7	5.81
		1.0e−04	195	6.10e−05	1.26	1.64	1.05	7	5.03
		1.0e−05	1015	6.18e−06	1.39	1.62	1.07	7	4.46
	2	1.0e−01	19	1.25e−02	–	8.00	1.05	2	2.05
		1.0e−02	28	9.53e−04	6.64	10.5	1.20	5	4.50
		1.0e−03	69	2.80e−04	1.36	3.57	1.13	8	6.19
		1.0e−04	186	1.92e−05	2.70	5.20	1.17	8	5.17
		1.0e−05	588	8.26e−07	2.73	12.1	1.20	8	4.37
	3	1.0e−01	18	1.87e−02	–	5.33	1.24	2	2.11
		1.0e−02	25	2.00e−03	6.81	4.99	1.19	7	6.72
		1.0e−03	49	6.59e−05	5.07	15.2	1.17	9	7.67
		1.0e−04	116	6.33e−06	2.72	15.8	1.15	8	5.72
		1.0e−05	248	2.01e−06	1.51	4.97	2.38	7	4.35
	4	1.0e−01	19	1.29e−02	–	7.75	1.05	2	2.05
		1.0e−02	25	1.45e−03	7.98	6.92	1.20	7	6.36
		1.0e−03	57	2.22e−04	2.27	4.50	1.03	7	4.81
		1.0e−04	105	1.17e−05	4.82	8.55	1.21	9	6.13
		1.0e−05	186	1.66e−06	3.41	6.02	1.29	9	6.28

Here $\|u-u_h\| := \|u-u_h\|_{L^\infty(G_h)}$.

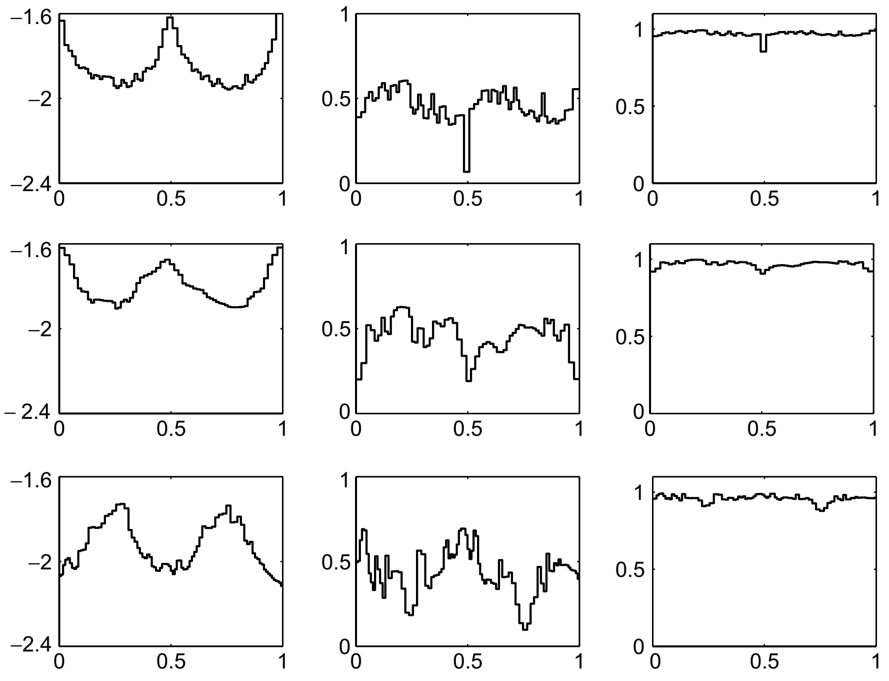

Fig. 5. Convergence step for the smooth test problems: $\tau = 10^{-5}$ and P^1 elements. Plotted from left to right are $\log_{10}h$, $\frac{\text{error}}{\tau}$ and the mesh modification function $\Psi(\gamma)$. The top row is for problem LS, the middle row for CCS and the bottom row for NCS.

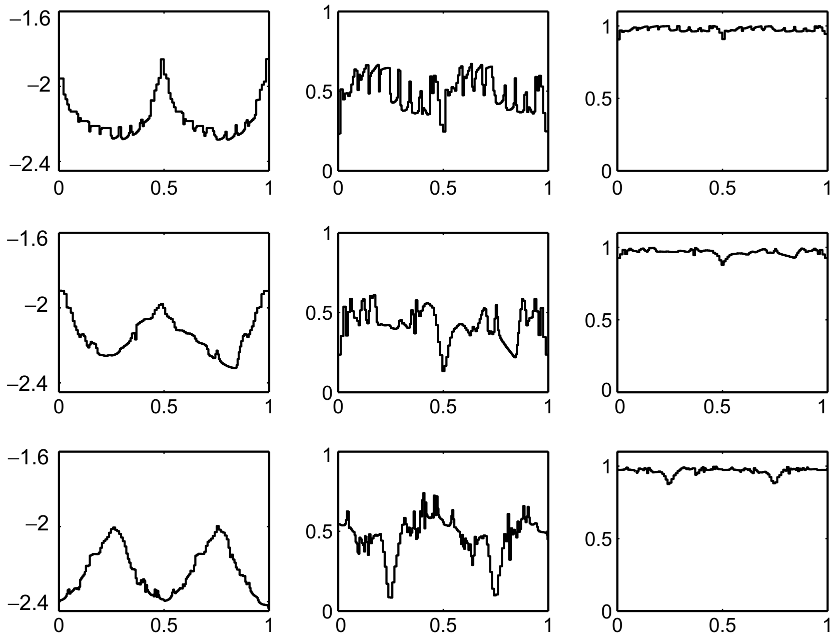

Fig. 6. Convergence step for the smooth test problems: $\tau = 10^{-6}$ and P^1 elements. Plotted from left to right are $\log_{10}h$, $\frac{\text{error}}{\tau}$ and the mesh modification function $\Psi(\gamma)$. The top row is for problem LS, the middle row for CCS and the bottom row for NCS.

Let us point out that the grids produced by the method are very smooth in most of the domain. Indeed, in Figs. 11 and 12, where we display the logarithm of the ratio of the size of two consecutive intervals, we can see that it is very close to zero in most of the domain.

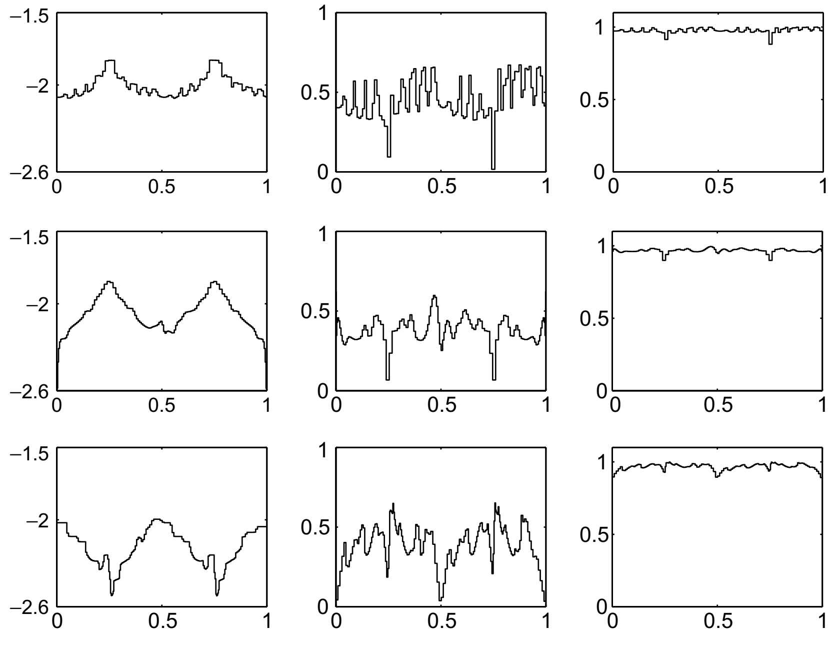

Fig. 7. Convergence step for the smooth test problems: $\tau = 10^{-8}$ and P^2 elements. Plotted from left to right are $\log_{10}h$, $\frac{\text{error}}{\tau}$ and the mesh modification function $\Psi(\gamma)$. The top row is for problem LS, the middle row for CCS and the bottom row for NCS.

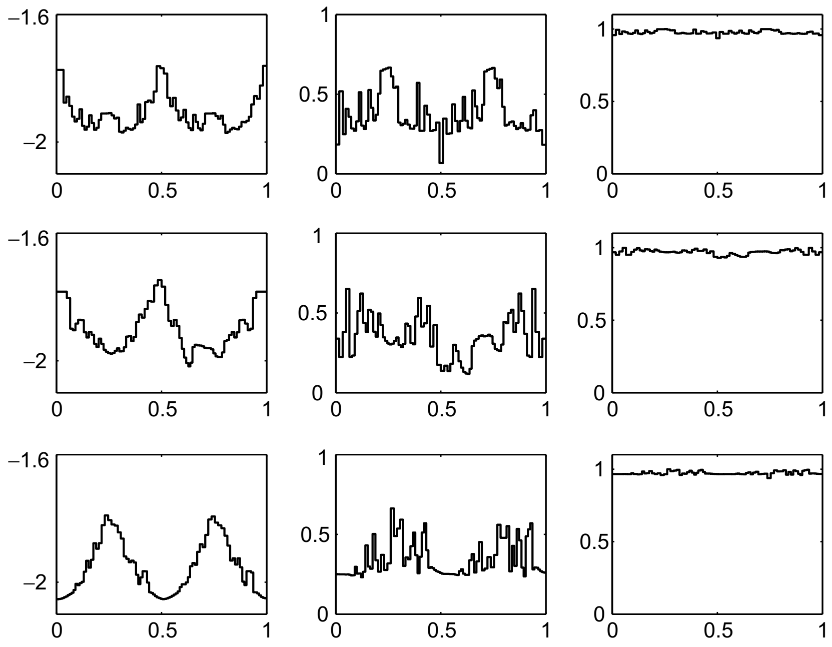

Fig. 8. Convergence step for the smooth test problems: $\tau = 10^{-10}$ and P^3 elements. Plotted from left to right are $\log_{10}h$, $\frac{\text{error}}{\tau}$ and the mesh modification function $\Psi(\gamma)$. The top row is for problem LS, the middle row for CCS and the bottom row for NCS.

Finally, in Tables 12–14, We show that the method can *capture* the kinks efficiently. The distance between the approximate position of the kink and the exact value is, roughly speaking, of the same order as the tolerance. So is size of the union of the intervals that capture the kinks.

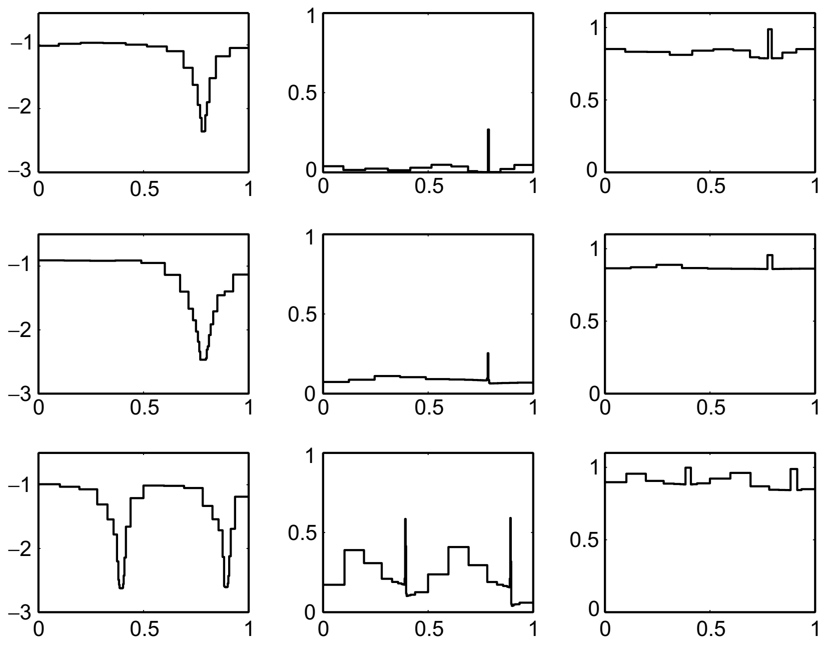

Fig. 9. Convergence step for the nonsmooth test problems: $\tau = 10^{-2}$ and P^1 elements. Plotted from left to right are $\log_{10}h$, $\frac{\text{error}}{\tau}$ and the mesh modification function $\Psi(\gamma)$. The top row is for problem CNS, the middle row for NCNS and the bottom row for CNS2.

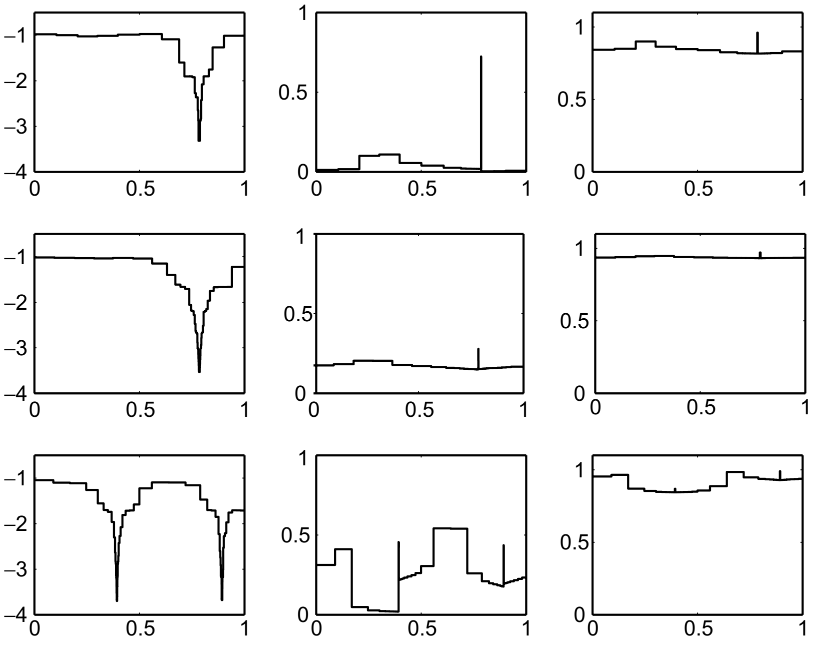

Fig. 10. Convergence step for the nonsmooth test problems: $\tau = 10^{-3}$ and P^2 elements. Plotted from left to right are $\log_{10}h$, $\frac{\text{error}}{\tau}$ and the mesh modification function $\Psi(\gamma)$. The top row is for problem CNS, the middle row for NCNS and the bottom row for CNS2.

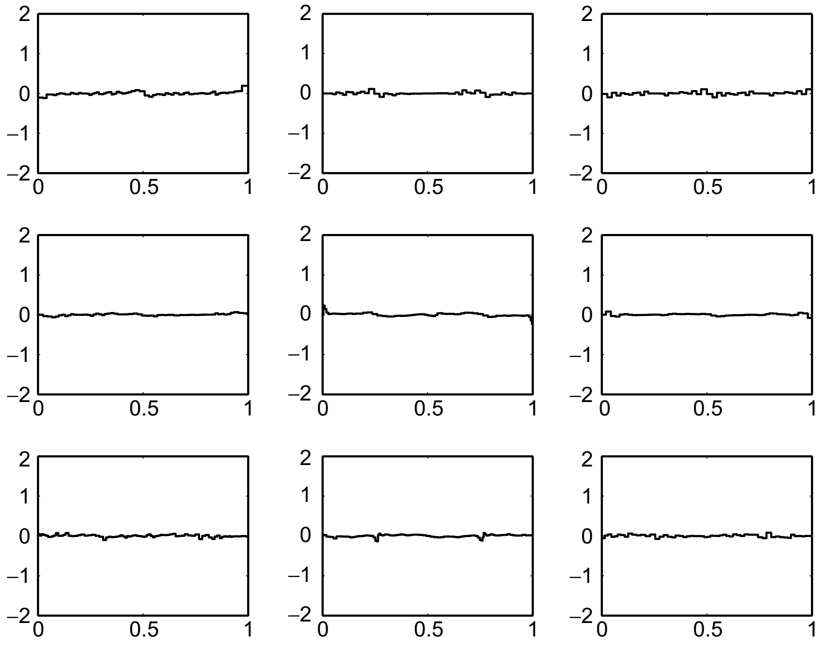

Fig. 11. Logarithm of the ratio of the mesh size of two consecutive elements. The tolerance $\tau = 10^{-5}$, 10^{-7}, 10^{-9} and the polynomial degree $k = 1, 2, 3$ for the columns from left to right respectively. The top row is for problem LS, the middle row for CCS and the bottom row for NCS.

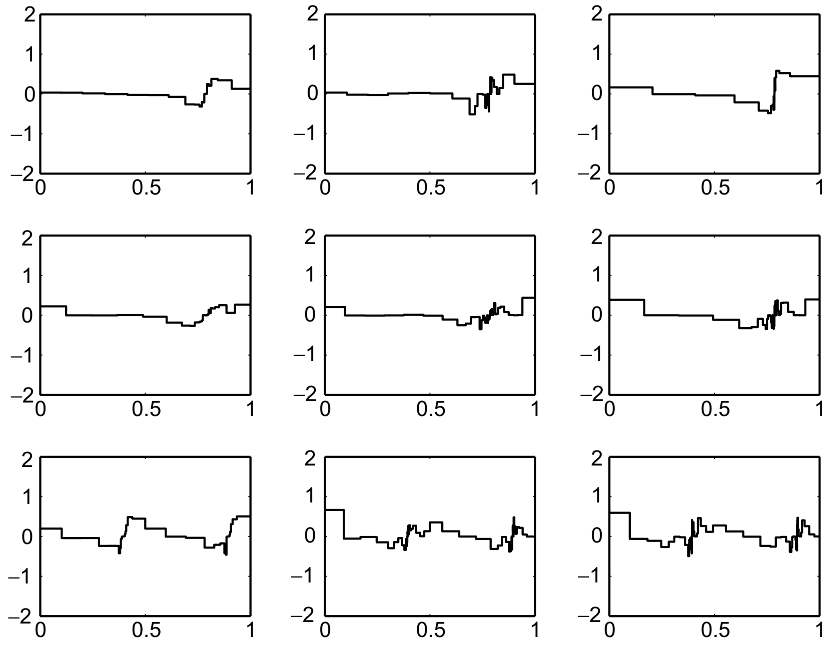

Fig. 12. Logarithm of the ratio of the mesh size of two consecutive elements. The tolerance $\tau = 10^{-2}$, 10^{-3}, 10^{-4} and the polynomial degree $k = 1$, 2, 3 for the columns from left to right respectively. The top row is for problem CNS, the middle row for NCNS and the bottom row for CNS2.

Table 11
History of convergence of the adaptive method for problem CNS2

Prob	Deg	τ	n	$\|u - u_h\|$	Order	$\mathrm{ei}_\tau(\tau; u, v)$	$\mathrm{ei}_{ad}(\tau; v)$	Steps	Complxr
CNS2	1	1.0e−01	19	3.86e−02	–	2.59	1.05	3	3.11
		1.0e−02	40	5.91e−03	2.52	1.69	1.01	6	4.85
		1.0e−03	88	7.87e−04	2.56	1.27	1.09	9	8.55
		1.0e−04	200	6.62e−05	3.02	1.51	1.24	8	7.99
		1.0e−05	444	6.86e−06	2.84	1.46	1.00	7	6.66
	2	1.0e−01	21	4.23e−02	–	2.36	1.07	3	2.95
		1.0e−02	43	4.66e−03	3.08	2.15	1.06	5	4.33
		1.0e−03	88	5.42e−04	3.00	1.85	1.04	7	5.36
		1.0e−04	144	5.82e−05	4.53	1.72	1.35	8	6.34
		1.0e−05	352	3.82e−06	3.05	2.62	1.17	9	7.91
	3	1.0e−01	25	4.23e−02	–	2.37	1.01	3	2.68
		1.0e−02	46	5.05e−03	3.48	1.98	1.08	6	4.89
		1.0e−03	63	4.13e−04	7.96	2.42	1.08	8	6.75
		1.0e−04	117	4.60e−05	3.55	2.18	1.01	8	6.97
		1.0e−05	238	3.94e−06	3.46	2.54	1.23	7	4.94
	4	1.0e−01	25	3.12e−02	–	3.21	1.09	5	4.52
		1.0e−02	43	4.08e−03	3.75	2.45	1.34	8	5.74
		1.0e−03	105	4.54e−04	2.46	2.20	1.10	9	6.90
		1.0e−04	115	4.96e−05	24.3	2.02	1.37	8	6.13
		1.0e−05	195	4.49e−06	4.55	2.23	1.27	8	5.15

Here $\|u - u_h\| := \|u - u_h\|_{L^\infty(G_h)}$.

Table 12
Kink-capturing for the nonsmooth solution test problems CNS

τ	Deg	$\frac{\lvert e_i-s_i\rvert}{\tau}$	$\frac{s_i+e_i}{2}$	$\frac{\text{error}}{\tau}$	N_i	Deg	$\frac{\lvert e_i-s_i\rvert}{\tau}$	$\frac{s_i+e_i}{2}$	$\frac{\text{error}}{\tau}$	N_i
1e−1	1	1.64	0.790	4.26e−02	4	3	1.64	0.789	3.90e−02	4
1e−2		1.76	0.785	4.64e−02	4		1.16	0.783	2.14e−01	4
1e−3		2.16	0.785	9.32e−03	6		1.24	0.785	2.68e−02	7
1e−4		1.51	0.785	1.38e−02	6		1.25	0.785	1.60e−01	7
1e−5		1.12	0.785	8.77e−02	6		1.21	0.785	3.24e−01	6
1e−1	2	0.88	0.806	2.06e−01	4	4	1.17	0.767	1.88e−01	4
1e−2		2.50	0.783	2.49e−01	5		1.02	0.784	1.24e−01	5
1e−3		2.41	0.785	5.46e−02	5		1.56	0.785	1.02e−01	5
1e−4		1.47	0.785	2.46e−02	7		2.21	0.785	1.04e−01	7
1e−5		1.69	0.785	2.23e−03	8		1.73	0.785	8.62e−02	9

Here, $[s_i, e_i]'s$ are the sets containing kinks, error := $\lvert\frac{s_i+e_i}{2}$ – exact position of the kink$\rvert$ and N_i is the number of elements covering $[s_i, e_i]$.

Table 13
Kink-capturing for the nonsmooth solution test problems NCNS

τ	Deg	$\frac{\lvert e_i-s_i\rvert}{\tau}$	$\frac{s_i+e_i}{2}$	$\frac{\text{error}}{\tau}$	N_i	Deg	$\frac{\lvert e_i-s_i\rvert}{\tau}$	$\frac{s_i+e_i}{2}$	$\frac{\text{error}}{\tau}$	N_i
1e−1	1	2.35	0.786	9.24e−03	6	3	2.52	0.774	1.15e−01	6
1e−2		2.38	0.785	1.69e−02	7		4.07	0.784	1.13e−01	10
1e−3		3.06	0.785	4.83e−01	10		3.76	0.786	1.32e−01	13
1e−4		6.99	0.785	7.17e−01	24		4.58	0.785	3.19e−01	18
1e−5		46.1	0.785	1.15e+00	177		3.17	0.785	1.00e+00	12
1e−1	2	2.39	0.783	1.98e−02	6	4	2.41	0.783	2.25e−02	6
1e−2		4.87	0.785	4.62e−03	8		3.99	0.786	2.14e−02	10
1e−3		3.54	0.785	5.08e−02	12		5.33	0.785	1.33e−02	13
1e−4		4.73	0.785	8.85e−02	20		4.61	0.785	2.60e−01	19
1e−5		3.12	0.785	4.85e−02	16		4.14	0.785	1.09e−01	20

Here, $[s_i, e_i]'s$ are the sets containing kinks, error := $\lvert\frac{s_i+e_i}{2}$ – exact position of the kink$\rvert$ and N_i is the number of elements covering $[s_i, e_i]$.

Table 14
Kink-capturing for the nonsmooth solution test problem CNS2

Deg	τ	CNS2 kink1				CNS2 kink2			
		$\frac{\lvert e_i-s_i\rvert}{\tau}$	$\frac{s_i+e_i}{2}$	$\frac{\text{error}}{\tau}$	N_i	$\frac{\lvert e_i-s_i\rvert}{\tau}$	$\frac{s_i+e_i}{2}$	$\frac{\text{error}}{\tau}$	N_i
1	1e−1	0.83	0.383	9.29e−02	3	0.83	0.897	4.49e−02	3
	1e−2	2.58	0.396	3.49e−01	10	3.15	0.899	6.43e−01	10
	1e−3	0.98	0.393	7.69e−02	5	1.43	0.893	3.24e−01	4
	1e−4	1.55	0.393	1.26e−01	5	1.38	0.893	1.12e−01	6
	1e−5	13.2	0.393	1.21e+00	71	10.8	0.893	4.52e−01	47
2	1e−1	1.16	0.392	3.33e−03	4	0.81	0.892	6.70e−03	3
	1e−2	1.24	0.394	1.22e−01	5	1.55	0.892	6.94e−02	6
	1e−3	1.88	0.393	3.12e−01	9	1.68	0.893	5.11e−02	8
	1e−4	1.18	0.393	2.23e−02	6	1.21	0.893	5.40e−02	6
	1e−5	1.46	0.393	1.54e−01	8	1.12	0.893	1.43e−01	8
3	1e−1	0.82	0.391	1.32e−02	3	0.82	0.905	1.19e−01	3
	1e−2	2.13	0.395	1.96e−01	7	2.28	0.893	1.89e−02	8
	1e−3	2.16	0.393	5.66e−02	10	2.55	0.893	2.11e−01	10
	1e−4	2.60	0.393	4.30e−01	12	2.49	0.893	5.11e−01	10
	1e−5	1.46	0.393	2.82e−01	7	1.20	0.893	2.10e−01	7
4	1e−1	1.02	0.397	4.24e−02	6	0.95	0.888	5.13e−02	6
	1e−2	1.24	0.392	4.30e−02	9	1.91	0.892	7.68e−02	9
	1e−3	1.56	0.393	2.43e−02	7	1.50	0.893	1.34e−01	8
	1e−4	1.81	0.393	6.05e−02	5	0.81	0.893	7.98e−02	5
	1e−5	1.51	0.393	8.05e−02	5	0.91	0.893	9.98e−02	5

Here, $[s_i, e_i]'s$ are the sets containing kinks, error := $\lvert\frac{s_i+e_i}{2}$ – exact position of the kink$\rvert$ and N_i is the number of elements covering $[s_i, e_i]$.

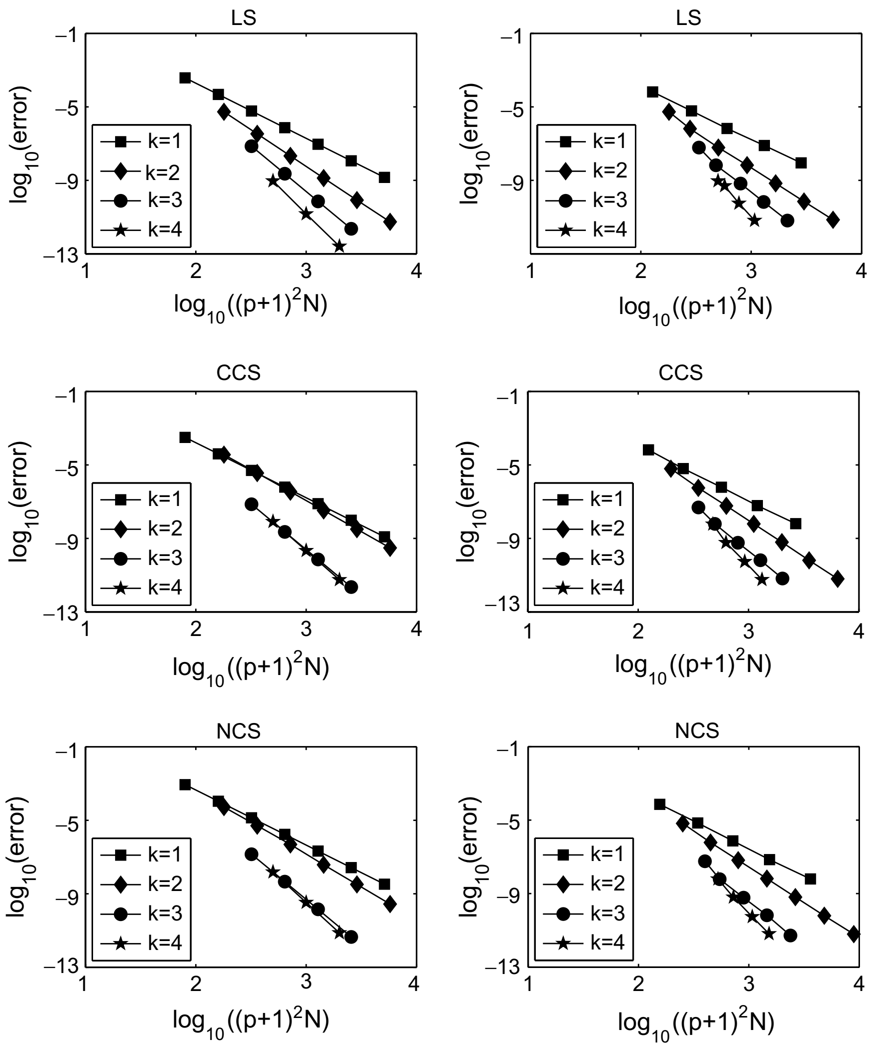

Fig. 13. Comparison of computational complexity: Uniform (left column) and adaptive (right column) refinement.

3.4. Adaptivity versus uniform refinement

Here we compare the uniform and the adaptivity refinements. In Fig. 13, we see that it is better to use higher degree polynomials than to use finer mesh. This is especially the case for adaptive refinement since the lines are more spread out. This shows that the adaptive method is very efficient.

In Figs. 14–19, we see that the adaptive refinement is obviously better than uniform refinement, since, to obtain the approximate solution with a given accuracy, the adaptive refinement always uses less intervals. Furthermore, the adaptive refinement method converges significantly faster than the uniform refinement for CCS and NCS. Let us end by pointing out that it is remarkable that, in the cases when the scheme converges sub-optimally with uniform refinement, it converges *optimally* with adaptive refinement.

4. Concluding remarks

In this paper, we have proposed a new a posteriori error estimate for approximations to the viscosity solution of a model steady state Hamilton–Jacobi equation which we used to devise an adaptive method for computing these approximations. The error estimate has been shown to be sharp and the adaptive method based

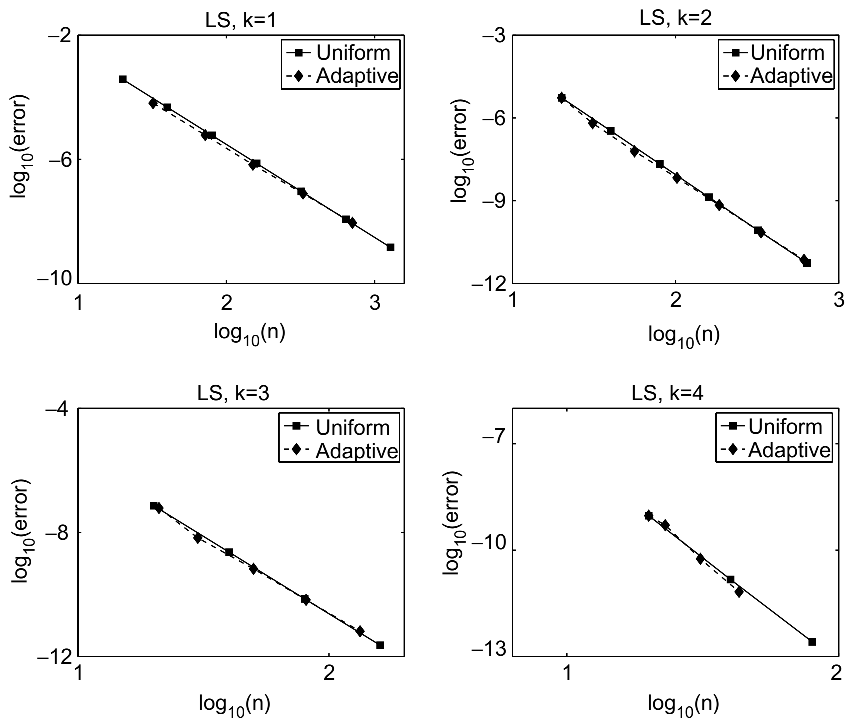

Fig. 14. Comparison of convergence rates: Uniform (solid line) and adaptive (dashed line) refinement for smooth solution test problem LS.

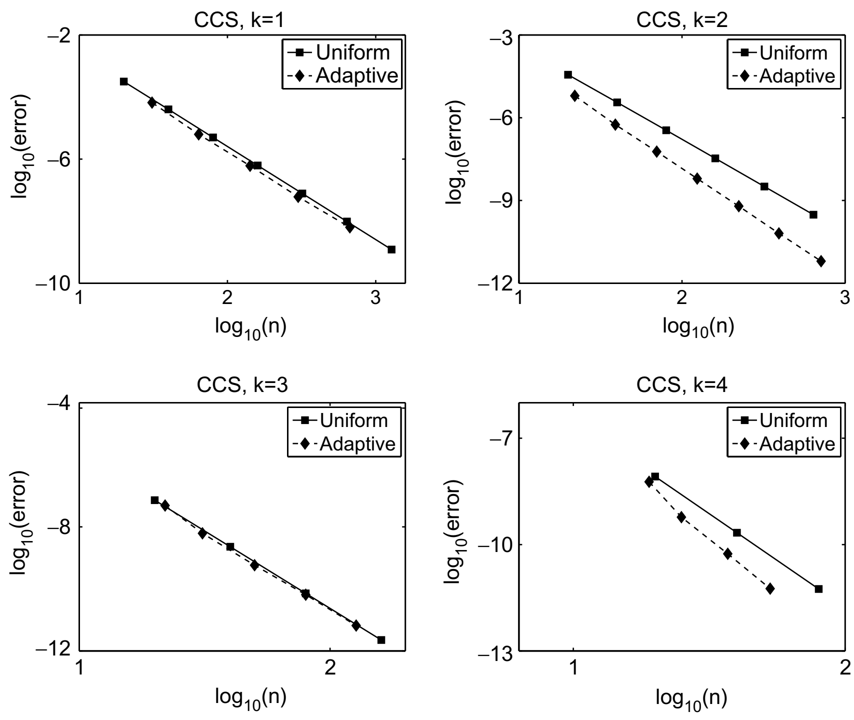

Fig. 15. Comparison of convergence rates: Uniform (solid line) and adaptive (dashed line) refinement for smooth solution test problem CCS.

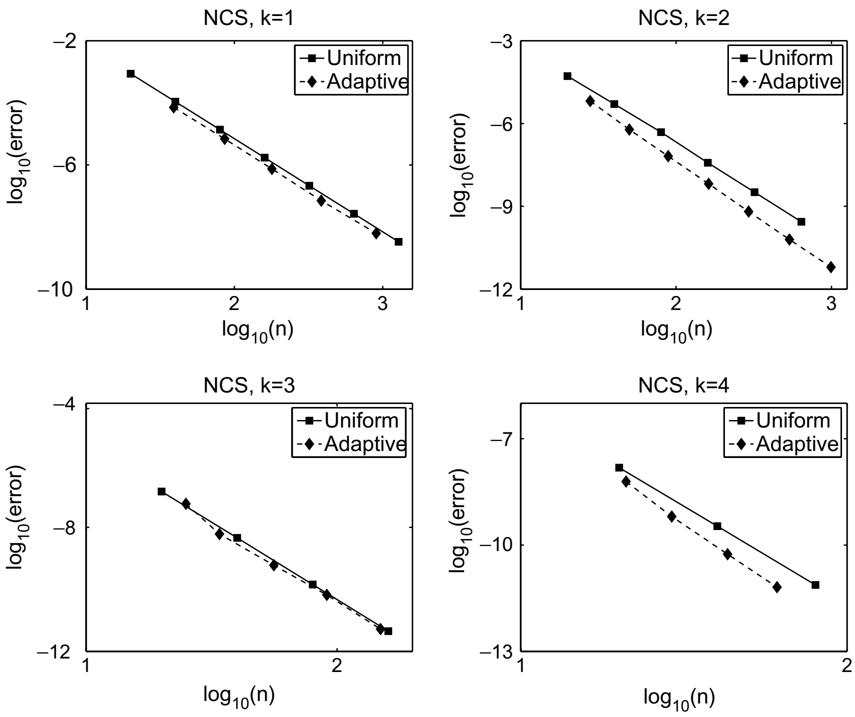

Fig. 16. Comparison of convergence rates: Uniform (solid line) and adaptive (dashed line) refinement for smooth solution test problem NCS.

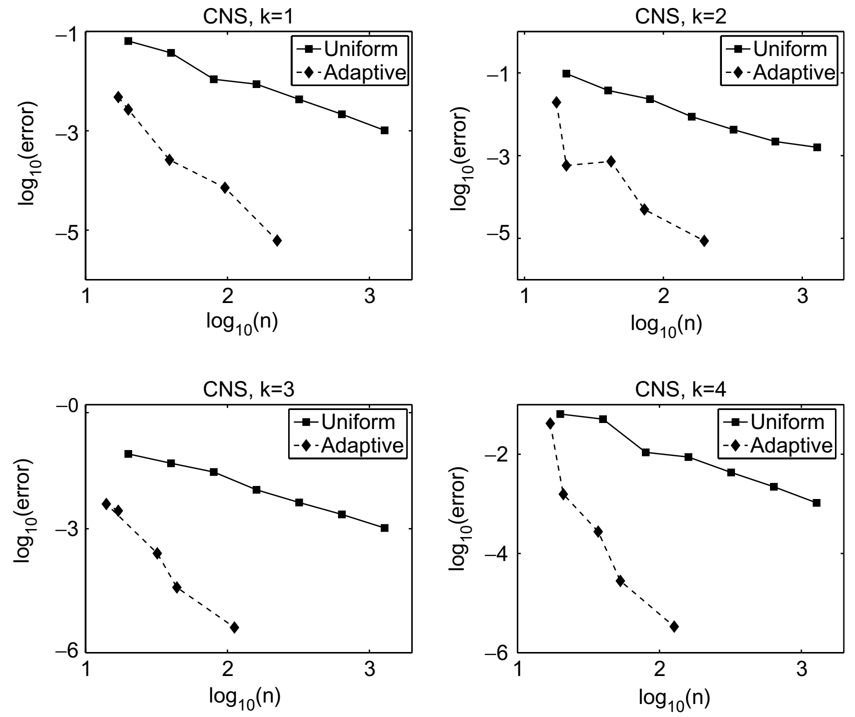

Fig. 17. Comparison of convergence rates: Uniform (solid line) and adaptive (dashed line) refinement for nonsmooth solution test problem CNS.

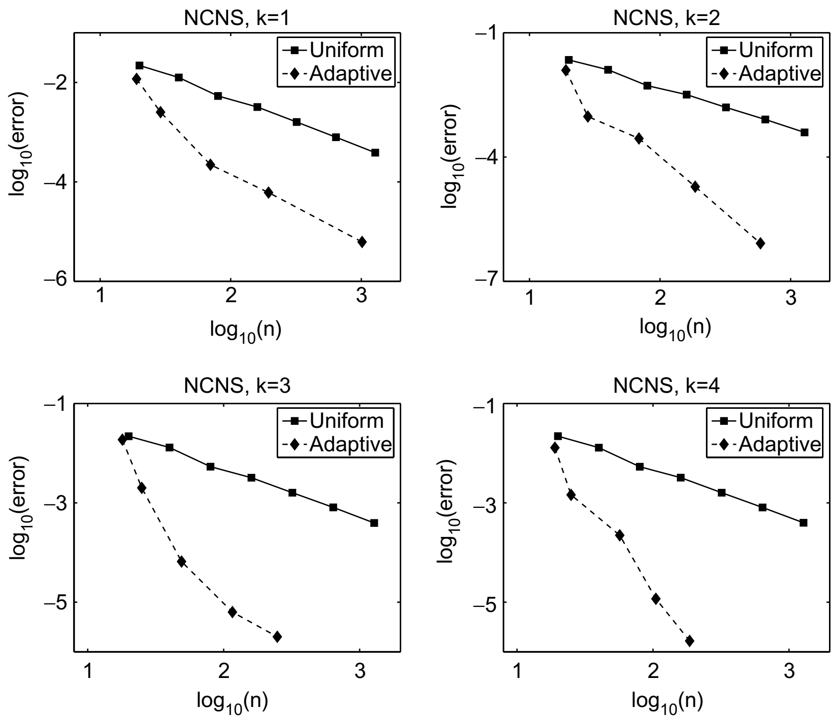

Fig. 18. Comparison of convergence rates: Uniform (solid line) and adaptive (dashed line) refinement for nonsmooth solution test problem NCNS.

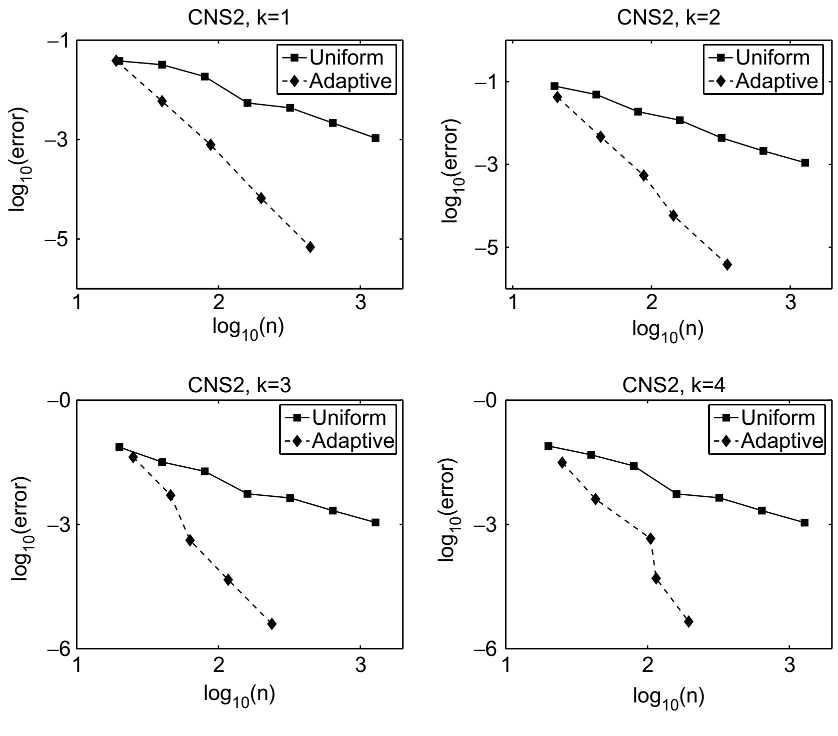

Fig. 19. Comparison of convergence rates: Uniform (solid line) and adaptive (dashed line) refinement for smooth solution test problem CNS2.

on it to be able to achieve a rigorous error control with optimal complexity uniformly in the tolerance τ and the polynomial degree k of the approximation even in the presence of kinks in the viscosity solution.

The extension of this adaptive algorithm to the multi-dimensional problems constitutes the subject of ongoing work.

References

[1] 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. Comp 71 (2002) 49–76.

[2] B. Cockburn, J. Qian, Continuous dependence results for Hamilton–Jacobi equations, in: D. Estep, S. Tavener (Eds.), Collected Lectures on the Preservation of Stability under Discretization, SIAM, 2002, pp. 67–90.

[3] B. Cockburn, C.W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (2001) 173–261.

[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] B. Cockburn, B. Yenikaya, An adaptive method with rigorous error control for the Hamilton–Jacobi equations. Part II: The two-dimensional steady state case, J. Comput. Phys. 209 (2005) 391–405.

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

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