

A MULTISCALE DOMAIN DECOMPOSITION ALGORITHM FOR BOUNDARY VALUE PROBLEMS FOR EIKONAL EQUATIONS*

LINDSAY MARTIN[†] AND YEN-HSI R. TSAI[‡]

Abstract. In this paper, we present a new multiscale domain decomposition algorithm for computing solutions of static Eikonal equations. In our new method, the decomposition of the domain does not depend on the slowness function in the Eikonal equation or the boundary conditions. The novelty of our new method is a coupling of coarse grid and fine grid solvers to propagate information along the characteristics of the equation efficiently. The method involves an iterative parareal-like update scheme in order to stabilize the method and speed up convergence. One can view the new method as a general framework where an effective coarse grid solver is computed “on the fly” from coarse and fine grid solutions that are computed in previous iterations. We study the optimal weights used to define the effective coarse grid solver and the stable update scheme via a model problem. To demonstrate the framework, we develop a specific scheme using Cartesian grids and the fast sweeping method for solving Eikonal equations. Numerical examples are given to show the method’s effectiveness on Eikonal equations involving a variety of multiscale slowness functions.

Key words. Eikonal equation, parallel algorithms, domain decomposition, multiscale algorithms

AMS subject classifications. 65N22, 65N55, 65N12, 35F30, 65Y05

DOI. 10.1137/18M1186927

1. Introduction. The Eikonal equation has many applications in optimal control, path planning, seismology, geometrical optics, etc. The equation is fully nonlinear and classified as a Hamilton–Jacobi equation. Usually, classical solutions do not exist, and the unique viscosity solution is sought after. Our goal is to numerically solve the following boundary value problem for the static Eikonal equation:

$$(1.1) \quad |\nabla u(x)| = r_\epsilon(x), \quad x \in \Omega \subset \mathbb{R}^d,$$

$$(1.2) \quad u(x) = g(x), \quad x \in \Gamma \subset \partial\Omega.$$

In particular, we are interested in the case where

$$r_\epsilon(x) = r_0(x) + a_\epsilon(x),$$

where r_0 is smooth and a_ϵ describes multiscale features in which the scales cannot be separated easily.

Many serial algorithms exist for computing numerical solutions to Eikonal equations. However, these algorithms have limitations when applied to large scale discretized systems. Since we are interested in Eikonal equations that have multiscale features, a very fine grid discretization is needed in order to accurately capture the fine scale features. This creates a large system of coupled nonlinear equations to

*Received by the editors May 14, 2018; accepted for publication (in revised form) March 19, 2019; published electronically April 25, 2019.

<http://www.siam.org/journals/mms/17-2/M118692.html>

Funding: This research is partially supported by National Science Foundation grants DMS-1620396 and DMS-1720171.

[†]Department of Mathematics, The University of Texas at Austin, Austin, TX 78712 (lmartin@math.utexas.edu).

[‡]Department of Mathematics and Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, and KTH Royal Institute of Technology, Sweden (ytsai@math.utexas.edu).

solve. Therefore, the numerical solutions are expensive to compute and speed-up is desired. The most popular serial algorithms are the fast sweeping method (FSM) [24, 27] and the fast marching method (FMM) [25, 22], which have complexity $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$, respectively. Here, N is the total number of grid points. Hidden in the $\mathcal{O}(N)$ complexity of FSM is a constant that corresponds to the number of times a characteristic curve of (1.1) “turns around.”

There are several approaches to reducing the computational cost of numerically solving Eikonal equations. For certain periodic functions r_ϵ , one approach is homogenization [18, 20]. The goal of homogenization is to derive an effective function, $\bar{r}$, that accurately describes the effective properties of r_ϵ in the solution. Once $\bar{r}$ is known, the homogenized equation can be solved on the coarse grid which is independent of the small parameter ϵ . For more general r_ϵ , we consider domain decomposition methods. The development of domain decomposition algorithms for Eikonal equations is nontrivial because of the causal nature of the equations. Standard domain decomposition methods can be difficult to apply because information may not be known at the boundaries of subdomains a priori. Furthermore, the causal relations among the subdomains may change depending on the solutions.

Our new algorithm works on general Eikonal equations, i.e., equations with multiple scales with or without scale separation. The algorithm combines features from parareal methods and standard Eikonal solvers in order to achieve speed-up and maintain accuracy. A set of coarse grids is used to set up boundary conditions for each subdomain. Since the structure of the characteristics of the equation are generally less complex in a subdomain compared to the overall domain, we expect that FSM on the subdomain grids will require less sweeping iterations than FSM on the overall fine grid. After each subdomain is processed in parallel, the method uses a parareal-like update in order to speed up the accuracy of the solution on the coarse grids.

Next we give an overview of the discretization of (1.1) and FSMs, followed by a review of current parallel methods for Eikonal equations. The paper is organized as follows. In section 2, we give an overview of parareal methods. Our new algorithm is presented in section 3. The stability analysis, complexity, and speed-up are given in section 4, experimental results are in section 5, and the summary and conclusion follow in section 6.

1.1. Upwind discretization and FSM. The Eikonal equation (1.1) can be derived from an optimal control problem. Suppose a particle travels at speed $F : \Omega \rightarrow \mathbb{R}$ and its direction of travel is the control of the system. Let $g : \Gamma \rightarrow \mathbb{R}$ be the penalty charged once the particle reaches Γ . Then the value function $u(x)$ is defined to be the minimum time it takes to travel from x to Γ . In [10], it is shown that the viscosity solution to (1.1) coincides with the value function of the optimal control problem and the characteristics of the PDE coincide with the optimal paths for moving through Ω .

In our case $F(x) = 1/r_\epsilon(x)$. Thus, we refer to r_ϵ as the slowness function. For this paper, we choose the following first-order upwind discretization on a uniform Cartesian grid. Let $u_{i,j}$ denote the numerical solution at $x_{i,j}$. For the sake of notation, we will omit the numerical solution’s dependence on the grid size h . We use a Godunov upwind scheme to discretize the Eikonal equation at points in the interior of the computational domain [21]:

$$(1.3) \quad \sqrt{\max(a^+, b^-)^2 + \max(c^+, d^-)^2} = r_{i,j},$$

where

$$\begin{aligned} a &= D_x^- u_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h}, \\ b &= D_x^+ u_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{h}, \\ c &= D_y^- u_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{h}, \\ d &= D_y^+ u_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{h}, \end{aligned}$$

for $i = 1, \dots, I-1$ and $j = 1, \dots, J-1$. Here, we have $x^+ = \max(x, 0)$ and $x^- = \max(-x, 0)$. On the boundary nodes, we will use a one-sided difference, i.e., in (1.3) we use b^- in place of $\max(a^+, b^-)$ if $i = 0$, a^+ in place of $\max(a^+, b^-)$ if $i = I$, d^- in place of $\max(c^+, d^-)$ if $j = 0$, and c^+ in place of $\max(c^+, d^-)$ if $j = J$.

This discretization is consistent and monotone and converges to the viscosity solution as $h \rightarrow 0$ [5]. The upwind scheme is also causal, i.e., $u_{i,j}$ depends only on the neighboring grid values that are smaller. After discretization, we have a system of $N = (I+1)(J+1)$ coupled nonlinear equations. A simple approach is to solve the system iteratively [21]. However, it is important to take advantage of the causality of the solution. In the FMM [25, 22], the solution is updated one grid node at a time and the ordering of grid nodes is given by whichever grid node has the smallest value at the time of updating. Because a heapsort algorithm is needed, the complexity is $\mathcal{O}(N \log N)$. Next we describe the FSM [24, 27], which we have chosen to use in our method. FSM uses Gauss–Seidel updates following a predetermined set of grid node orderings. For simplicity, we will describe the algorithm in two dimensions.

Initialization. Set $u_{i,j} = g_{i,j}$ for $x_{i,j}$ on or near the computational boundary. These values are fixed in later iterations. For all the other grid nodes, assign a large positive value.

Sweeping iterations. A compact way of writing the grid orderings in C/C++ is

```
for(s1=-1; s1<=1; s1+=2)
  for(s2=-1; s2<=1; s2+=2)
    for(i=(s1<0?I:0); (s1<0?i>=0:i<=I); i+=s1)
      for(j=(s2<0?J:0); (s2<0?j>=0:j<=J); j+=s2)
```

Update formula. For each grid node $\mathbf{x}_{i,j}$ whose value is not fixed during the initialization, compute the solution to (1.3) using the current values at the neighboring grid nodes. Denote the solution by $\tilde{u}$; then the update formula is as follows:

$$(1.4) \quad u_{i,j}^{new} = \min(u_{i,j}^{curr}, \tilde{u}).$$

The alternating ordering of sweeping ensures that all the directions of characteristics are captured. In [27] it is shown that with the first-order Godunov upwind scheme, 2^d sweeps are sufficient to compute the numerical solution to first order in h . The exact number of sweeps needed is related to the number of times characteristics change directions. Thus in general the computational complexity of the FSM is $\mathcal{O}(N)$ with the caveat that the constant in front of N can be very large depending on the characteristics of the equation. In section 3, we will describe how we use the FSM as the Eikonal equation solver in our method.

1.2. Review of current parallel methods. Here we give a brief overview of existing parallel approaches. In [28], the author proposes two parallelizations of FSM. The first performs the 2^d sweeps of the domain on different processors and after each iteration information is shared by taking the minimum value at each grid node from each sweep. The second is a domain decomposition method that performs FSM on each subdomain in parallel. The information is shared along mutual boundaries after each iteration. The drawbacks to this method are that subdomains have to wait to be updated until the information propagates to that part of the domain and the number of sweeping iterations may be more than the number needed in serial FSM.

In [12], a method is introduced that takes advantage of the following fact: for the upwind scheme (1.3), certain slices of the grid nodes do not directly depend on each other. The method uses FSM where the sweeping ordering is designed to allow these sets of grid nodes to be updated simultaneously. The advantage of this method is that the number of iterations needed in the parallel implementation is equal to the serial FSM. In a later paper [11], the domain decomposition approach of [28] is used to couple the subdomain computations done in parallel; furthermore, in each subdomain, the parallel algorithm of [12] is applied.

Several algorithms have been developed to parallelize FMM. In [6], a domain decomposition method for FMM is proposed. The main idea is to split the boundary among different processors, which leads to an equation dependent method. In [26], another domain decomposition algorithm is presented for FMM. In this method, the computational domain is split among different processors and a novel restarted narrow band approach which coordinates the communications among the boundaries of the domains is used.

Domain decomposition methods that utilize two scales can be found in [7, 9]. In [7], the method takes advantage of the optimal control formulation of Eikonal equations. First, the algorithm computes the solution of (1.1) on a coarse grid. Next, the domain decomposition is determined by the feedback optimal control. Last, the solution of the equation is computed on a fine grid in each subdomain. However, the algorithm can lead to complex division of the domain. The method in [9] is a parallelization of the heap cell method (HCM) [8]. HCM maintains a list of cells to be processed. The order of processing is determined by an assigned cell value that is given by an estimate of the likelihood that that cell influences other cells. If it is determined that a cell highly influences other cells it should be processed first. The method mimics FMM on the coarse level, and FSM is used at the cell level. This choice is motivated by the observation that the characteristics of the equation typically do not “wind” around as much in each cell due to the smaller domain size, and FSM needs fewer iterations. On the coarse level, the characteristics are expected to be followed more efficiently by the FMM. The parallelization of HCM divides the cells evenly among p heaps and performs HCM among each individual heap. If a cell is tagged for reprocessing, then it is added to the heap with the current lowest number of cells. This method was found to achieve the best speed-up on problems where the amount of work per cell is high.

In the above methods, only adjacent subdomains exchange information. Thus, if an iterative approach such as the one in [28] is used, the subdomains further downwind of the characteristics will get the correct information update only after a sufficient number of iterations. If a fast marching type strategy is used, those subdomains will have to wait until the information is propagated there. Our method employs an entirely different way of coupling the subdomains. In our method, an effective coarse grid solver (see (2.6), (4.3), and (4.7)–(4.10)) computes solutions on the fly using

information from previous coarse and fine grid computations and allows information to propagate efficiently throughout the domain. In this regard, our method looks more like a heterogeneous multiscale method [13], with the difference that no wide scale separation is assumed and (therefore) larger subdomains are used. An example of a different scheme under this framework is given in the first example in section 5.3. Last, the decomposition of the domain in our new method is simple since it does not rely on the specific characteristics of the given equation as in some previous methods.

2. Overview of parareal methods. Parareal methods [17, 4] were developed to parallelize numerical computations of the solutions to ODEs of the form

$$(2.1) \quad \frac{d}{dt}u = f(u), \quad u(0) = u_0$$

on bounded time interval $[0, T]$. Let u_n^k be the computed solution at iteration k at time $t_n = nH$. Let C_H and F_H be the numerical coarse and fine integrators, over time step H . The idea is that C_H is less accurate and inexpensive to compute, and F_H is very accurate and expensive to compute. The parareal update scheme is then defined as

$$(2.2) \quad u_{n+1}^{k+1} = C_H(u_n^{k+1}) + F_H(u_n^k) - C_H(u_n^k), \quad n, k = 0, 1, 2, \dots, N,$$

with initial conditions

$$(2.3) \quad u_0^k = u_0, \quad k = 0, 1, 2, \dots, N.$$

The zeroth iteration is given by

$$(2.4) \quad u_{n+1}^0 = C_H(u_n^0), \quad n = 0, 1, 2, \dots, N.$$

The integrations $F_H(u_n^k)$ are independent for each n and can be computed in parallel. If C_H is of order 1, then under certain assumptions, the error after k iterations of the parareal scheme is of order $o(H^k + e^f)$, where e^f is the global error from solving (2.1) with the fine integrator F_H [19]. The method provides speed-up only if k is much smaller than N .

The method is generally unstable for hyperbolic problems and problems with imaginary eigenvalues [23, 3]. Parareal methods for highly oscillatory ODEs can be found in [1, 15]. In [14], analysis of the parareal method on a class of ODEs originating in Hamiltonian dynamical systems is presented, and in [16] the parareal method is applied to stiff dissipative ODEs. Recently, a “weighted” parareal scheme, called θ -parareal, was proposed in [2]. Following the scheme in [2], let

$$(2.5) \quad u_{n+1}^{k+1} = \theta C_H(u_n^{k+1}) + (1 - \theta)C_H(u_n^k) + F_H(u_n^k) - C_H(u_n^k),$$

which simplifies to

$$(2.6) \quad u_{n+1}^{k+1} = \theta C_H(u_n^{k+1}) + F_H(u_n^k) - \theta C_H(u_n^k).$$

In [2], the “weight” θ is generalized to an operator which maps $C_H u$ to a small neighborhood of $F_H u$. In this paper, we only let θ be a real number which may vary for each grid node, i.e., $\theta = \theta_n^k$.

Several properties of the parareal scheme are appealing when solving Eikonal equations.

- Parareal methods use communications between the two scales in order to propagate information quickly through time. Because the fine integrations can be computed in parallel, the method is able to deal with a large number of unknowns.
- The characteristics of Eikonal equations also have a “time-like” structure which makes parareal methods attractive.

The main challenge in applying the parareal scheme to Eikonal equations is that we are now dealing with an infinite number of characteristics simultaneously. We also must be able to handle the collision of characteristics which should be captured accurately in the numerical solution in order to compute the viscosity solution. We adapt the θ -parareal scheme in order to stabilize the new method.

3. The new method. The method is a domain decomposition method that uses two scales to resolve the fine scale features in r_ϵ and propagate information through the computational domain. We use FSM as the Eikonal equation solver on the coarse and the fine grid. An adapted version of the θ -parareal method is used to propagate information along the characteristics efficiently where the weight θ stabilizes the method. The optimal choice of weights for stability is studied in section 4. First, we will demonstrate the method on a one-dimensional problem and then explain how to set up the method in two dimensions, which can be generalized to higher dimensions.

3.1. One-dimensional example. Consider the following one-dimensional Eikonal equation:

$$(3.1) \quad |u_x| = r(x), \quad 0 < x < 1,$$

$$(3.2) \quad u(0) = u(1) = 0,$$

where

$$(3.3) \quad r_\epsilon(x) = 1 + 10e^{\frac{-(x-.75)^2}{2(.01)^2}}.$$

Figure 1 shows the plot of the slowness function r_ϵ . Let the coarse grid be defined by

$$\Omega^H := \{jH : j = 0, 1, \dots, N\},$$

where $H = 1/N$ and for $i = 0, 1, \dots, N - 1$. Define the fine grids by

$$\Omega_i^h := \{iH + mh : m = 0, 1, \dots, M\},$$

where $h = 1/(MN)$. Define $\Omega^h := \bigcup_{i=0}^{N-1} \Omega_i^h$. The solution to the upwind Godunov scheme in one dimension is given by

$$(3.4) \quad C_H(U_{i-1}, U_{i+1}) := \min(U_{i-1}, U_{i+1}) + r(X_i)H.$$

We see that if we only solve (3.1) on the coarse grid, the bump in the slowness function is not seen and the solution is very inaccurate. There are also points in Ω^H where the flow of characteristics is incorrect. Therefore, we keep track of wind direction, i.e., which neighboring grid node gives the minimum in (3.4). Let $X_i = iH$. We denote the numerical solution at the k th iteration at the coarse grid node X_i by U_i^k . For grid nodes on the subintervals, Ω_i^h , let $X_{i_m} = iH + mh$ and $u_{i_m}^k$ be the numerical solution at the k th iteration at the fine grid node X_{i_m} . For each coarse grid node, X_i , $i = 1, \dots, N - 1$, we will get two values from the fine grid computations. One value is from Ω_{i-1}^h and another from Ω_i^h . Let u_i^k be the fine grid solution at the k th iteration at X_i which we will define in Step 3. The method is as follows.

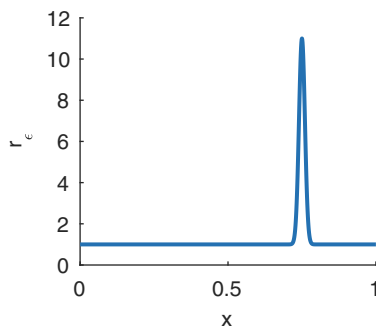


FIG. 1. $r_\epsilon(x) = 1 + 10e^{\frac{-(x-.75)^2}{2(.01)^2}}$.

Step 1: Initialization. Solve (3.1) with boundary conditions (3.2) via FSM on the coarse grid Ω^H , and denote the solution U^0 . If the left-hand neighboring grid node is used to compute U_i^0 , denote the wind direction at X_i by $W_i^0 = 1$. If the right-hand neighboring grid node is used, define $W_i^0 = -1$.

Step 2: Update boundary conditions for the subintervals. Once the coarse grid has been initialized, we use the coarse grid values, U^k , as boundary values for Ω_i^h . The characteristics may flow into or out of Ω_i^h . Thus, when setting the boundary conditions, we check the wind direction to see if the coarse grid value should be used as a boundary value. Intuitively, if a characteristic at a coarse grid node, x_{i_0} or x_{i_M} , is arriving into the subinterval, then we set the boundary value to U_i or U_{i+1} at x_{i_0} or x_{i_M} , respectively. Otherwise, we set the boundary value to be ∞ .

Step 3: Solve for u^k in parallel. In parallel for each $i = 0, 1, \dots, N-1$, we solve via FSM on Ω_i^h

$$(3.5) \quad |u_x| = r(x), \quad x \in (iH, (i+1)H)$$

with the boundary conditions described in Step 2. Denote the solutions after sweeping by $u_{i_m}^k$ for $m = 0, \dots, M$. We keep track of the fine wind directions, $w_{i_m}^k$, in the same manner as in Step 1. For each coarse grid node, X_i , $i = 1, \dots, N-1$, we will get two values from the fine grid computations. One value is from Ω_{i-1}^h and another from Ω_i^h . Consider a coarse grid point, X_i :

- If $w_{i-1_M}^k = w_{i_0}^k = 1$, then we choose u_i^k to be $u_{i-1_M}^k$ since the wind is flowing from left to right.
- If $w_{i-1_M}^k = w_{i_0}^k = -1$, then we choose u_i^k to be $u_{i_0}^k$ since the wind is flowing from right to left.
- Otherwise, we take the minimum of $u_{i-1_M}^k$ and $u_{i_0}^k$.
- We set w_i^k to be the wind value corresponding to the fine grid point used to define u_i^k .

For the given example, U^0 is plotted in Figure 2(a) and u^0 is plotted in Figure 2(b).

Step 4: Coarse grid updates. Now we compute U^{k+1} . We use the previous coarse and fine wind directions to determine whether we will use a weighted correction. We sweep the grid as in FSM and the update formula is as follows:

- Let $\tilde{U} = C_H(U_{i-1}^{k+1}, U_{i+1}^{k+1})$. If the left-hand neighboring grid node was used to compute $\tilde{U}$, then denote the current wind direction $\tilde{W} = 1$. If the right-hand neighboring grid node was used, define $\tilde{W} = -1$.

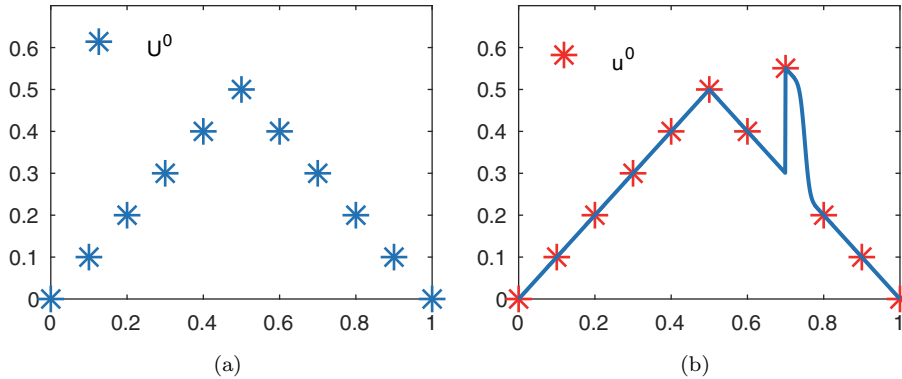


FIG. 2. (a) Plot of U^0 . (b) Plot of u^0 using U^0 as boundary conditions as defined in Step 2.

- If $W_i^k = w_i^k = \tilde{W}$, then we use a weighted correction update, i.e.,

$$(3.6) \quad U_i = \theta \tilde{U} + u_i^k - \theta C_H(U_{i-1}^k, U_{i+1}^k)$$

and $W_i = w_i^k$.

- Otherwise we set $U_i = u_i^k$ and $W_i = w_i^k$.
- After the weighted corrections, the solutions may have the wrong causality because information on the fine grid that was not seen previously has now been propagated to the coarse level. To correct this, we implement a causal sweep after each coarse grid update. Sweeping the coarse grid in both directions, the causal update is as follows:
 - If $W_i = 1$ and $U_i < U_{i-1}$, then $U_i = U_{i-1}$.
 - If $W_i = -1$ and $U_i < U_{i+1}$, then $U_i = U_{i+1}$.

After the causal sweep on the coarse grid, denote the solution by U^{k+1} and the wind directions by W^{k+1} . Repeat Steps 2–4 until convergence.

In Figure 3(a), we see that at $X_7 = 0.7$ the effect of the Gaussian bump in r_ϵ has been propagated to the coarse level. Before the causal sweep, $U_6 < U_7$, but $W_6 = -1$. Therefore, after the casual sweep, $U_6^1 = U_7^1$. Figures 4 and 5 show the next two iterations of the method which converges at $k = 3$.

The advantage of the coarse and fine grid coupling is clear in this example. Our method is equivalent to the method in [28] if we let $\theta = 0$ and there is no coarse grid initialization. When $\theta = 0$, it would take six iterations to converge since there are six subintervals between the left boundary node and the subinterval where the characteristics intersect in the overall fine solution. This is because there is no propagation of the information at the coarse level after the fine grid simulations. The number of iterations it takes to converge increases as the number of subintervals increases when $\theta = 0$.

Next, we introduce the method in two dimensions in more detail.

3.2. New method in two dimensions. In two dimensions we solve

$$(3.7) \quad |\nabla u(x)| = r(x), \quad x \in [0, 1]^2 \setminus \Gamma,$$

$$(3.8) \quad u(x) = 0, \quad x \in [0, 1]^2 \Gamma.$$

One of the main challenges of setting up the method in two dimensions and higher is setting up the boundary conditions of the subdomains. We approach this by setting

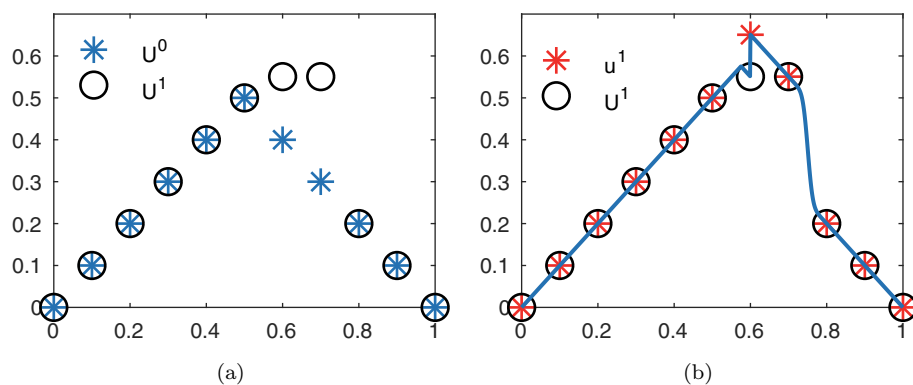


FIG. 3. (a) Plot of U^0 and U^1 . (b) Plot of u^1 using U^1 as boundary conditions as defined in Step 2.

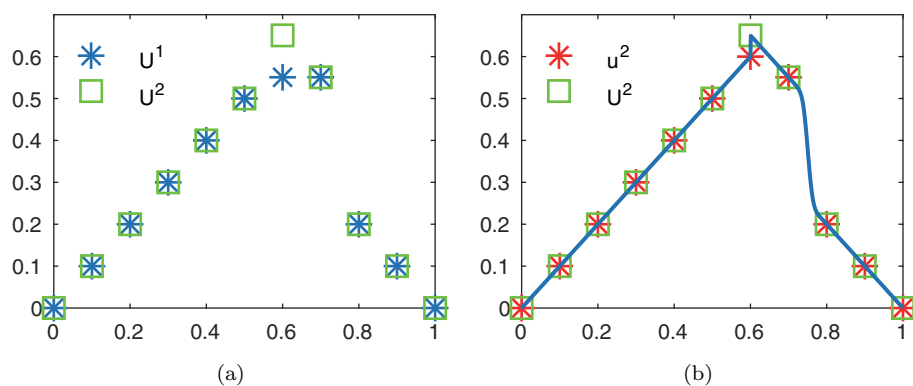


FIG. 4. (a) Plot of U^1 and U^2 . (b) Plot of u^2 using U^2 as boundary conditions as defined in Step 2.

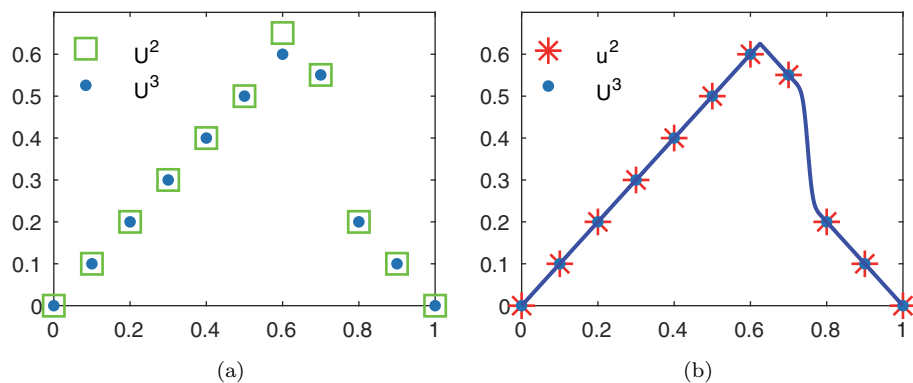


FIG. 5. (a) Plot of U^2 and U^3 . (b) Plot of u^3 using U^3 as boundary conditions as defined in Step 2. We see that the method has converged and $U^3 = u^3$.

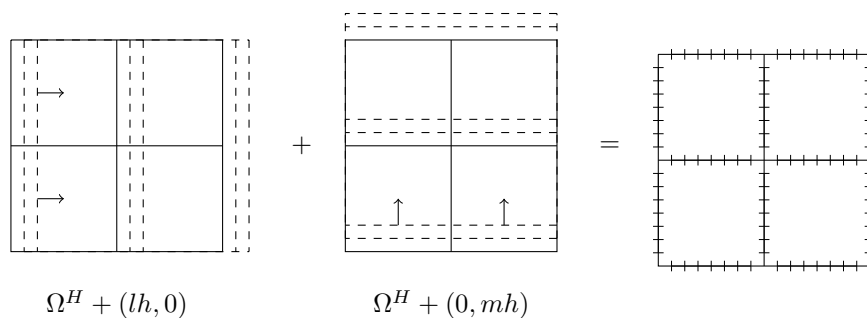


FIG. 6. Shifted coarse grids in two dimensions.

up a coarse grid and shifting it vertically and horizontally $M - 1$ times each. Let

$$\Omega^H = \{(iH, jH) : i, j = 0, 1, \dots, N\},$$

and $H = 1/N$. Then the horizontally shifted coarse grids are defined by

$$\Omega^H + (x_0, 0) = \{(iH + x_0, jH) : i = 0, 1, \dots, N - 1, j = 0, 1, \dots, N\},$$

where $x_0 = lh$ for $l = 1, \dots, M - 1$ where $h = 1/(MN)$. The vertically shifted coarse grids are defined by

$$\Omega^H + (0, y_0) = \{(iH, jH + y_0) : i = 0, 1, \dots, N, j = 0, 1, \dots, N - 1\},$$

where $y_0 = mh$ for $m = 1, \dots, M - 1$. The shifted grids are demonstrated in Figure 6. Next we define the fine grids on the subdomains for $i, j = 0, 1, \dots, N - 1$:

$$\Omega_{i,j}^h = \{(lh + iH, mh + jH) : 0 \leq l, m \leq M\}.$$

The notation for the two-dimensional problem is as follows:

$$\begin{aligned} X_{i,j} &= (iH, jH) \in \Omega_H, \\ X_{i_l,j} &= (iH + lh, jH) \in \Omega_H + (lh, 0), \\ X_{i,j_m} &= (iH, jH + mh) \in \Omega_H + (0, mh), \\ x_{i_l,j_m} &= (iH + lh, jH + mh) \in \Omega_{i,j}^h, \\ U_{i,j}^k &\text{ denotes the coarse solution at } X_{i,j} \text{ in the } k\text{th iteration,} \\ U_{i_l,j}^k &\text{ denotes the coarse solution at } X_{i_l,j} \text{ in the } k\text{th iteration,} \\ U_{i,j_m}^k &\text{ denotes the coarse solution at } X_{i,j_m} \text{ in the } k\text{th iteration,} \\ u_{i,j}^k &\text{ denotes the fine solution at } X_{i,j} \text{ in the } k\text{th iteration,} \\ u_{i_l,j_m}^k &\text{ denotes the fine solution at } x_{i_l,j_m} \text{ in the } k\text{th iteration.} \end{aligned}$$

Now that we have the grids set up we begin the description of the method. The coarse grid solver is given by the solution to (1.3):

$$(3.9) \quad C_H(\text{nbrs}^H(U_{i,j})) = \begin{cases} \frac{1}{2} \left(a + b + \sqrt{2r_{i,j}^2 H^2 - (a - b)^2} \right) & \text{if } |a - b| < r_{i,j} H, \\ \min(a, b) + r_{i,j} H & \text{if } |a - b| \geq r_{i,j} H, \end{cases}$$

where $\text{nbrs}^H(U_{i,j}) = \{U_{i-1,j}, U_{i+1,j}, U_{i,j-1}, U_{i,j+1}\}$, $a = \min(U_{i-1,j}, U_{i+1,j})$, and $b = \min(U_{i,j-1}, U_{i,j+1})$.

The steps are the same as in the one-dimensional case, except we also have a causal sweep in the initialization step. Step 1 is the initialization of the coarse grids with a causal sweep, Step 2 is to update the boundary conditions for the subdomains, and Step 3 is to compute the fine solutions on the subdomains in parallel. Step 4 is to perform weighted corrections on the coarse grids where we allow θ to vary for each coarse grid node. The weighted update will be

$$(3.10) \quad U_{i,j}^{k+1} = \theta_{i,j}^{k+1} C_H(\text{nbrs}^H(U_{i,j}^{k+1})) + u_{i,j}^k - \theta_{i,j}^{k+1} C_H(\text{nbrs}^H(U_{i,j}^k)).$$

Note that the weight function $\theta_{i,j}^k$ acts as a multiplicative correction to the coarse operator C_H so that $\theta_{i,j}^{k+1} C_H(\text{nbrs}^H(U_{i,j}^k))$ ideally approximates $u_{i,j}^k$.

Step 1: Initialize coarse grids in parallel. Since the shifted coarse grids are independent of each other, the values $\{U_{i,j}^0\}_{i,j}$ and $\{U_{i,j_m}^0\}_{i,j}$ are computed in parallel for each l and m . We solve (3.7) with boundary conditions (3.8) on each of the coarse grids. Keeping track of the flow of characteristics is more complex than in the one-dimensional problem. In two dimensions, the set of eight distinct wind direction vectors is $\{(\pm 1, \pm 1), (\pm 1, 0), (0, \pm 1)\}$. The wind direction at a coarse grid node is determined by the solution to (1.3). We describe how to initialize the grid Ω^H . The initialization on $\Omega^H + (lh, 0)$ and $\Omega^H + (0, mh)$ for $l = 1, \dots, M-1$ and $m = 1, \dots, M-1$ is the same.

- Initialize U as described in section 1.1.
- Sweep the grid as described in section 1.1. Algorithm 3.1 explains the update formula as well as how to compute the wind directions $\mathbf{W}_{i,j}^0$. At each $X_{i,j}$, we input $\text{nbrs}^H(U_{i,j})$, $U_{i,j}$, and H , using one sided differences if $X_{i,j}$ is a boundary grid node.

The solutions on the coarse grids may have the wrong causality because small scale features in r_ϵ may be sampled on some shifted coarse grids and not others. To correct this, we implement a causal sweep. We must sweep the coarse grids sequentially in order to capture the right causality. We sweep all the coarse grids in each of the four directions just once. The update is given by inputting $U_{i,j_m-1}, U_{i,j_m+1}, U_{i,j_m}, \mathbf{W}_{i,j_m}$ into Algorithm 3.2, which describes the update for a vertically shifted grid node. The updates for the other coarse grid nodes are defined analogously. Note that since we are sweeping the coarse grids sequentially, U_{i,j_m-1}, U_{i,j_m+1} , and U_{i,j_m} belong to three different vertically shifted coarse grids. Denote the solutions after sweeping by U^0 and $\mathbf{W}^0$.

Step 2: Update boundary conditions for subdomains. Now that we have computed the solutions on all the coarse grids, we can set the boundary conditions for each $\Omega_{i,j}^h$. Intuitively, if a characteristic at a coarse grid point is arriving into the boundary of the subdomain, $\partial\Omega_{i,j}^h$, then we set u at that node to be the value from the coarse grid computations, U^k . Otherwise, we set u to be ∞ at the coarse grid point. To describe this mathematically for a vertically shifted coarse grid point, define $\mathbf{n}_{w,r_m}$ to be the inward normal vector to the subdomain $\Omega_{i,j}^h$ at $X_{w,r_m} \in \partial\Omega_{i,j}^h$. Then define

$$g_{w,r_m}(U_{w,r_m}, \mathbf{W}_{w,r_m}) := \begin{cases} U_{w,r_m} & \text{if } \mathbf{W}_{w,r_m} \cdot \mathbf{n}_{w,r_m} > 0, \\ \infty & \text{otherwise.} \end{cases}$$

When $X_{w_l,r}$ is a horizontally shifted grid point, the definition of $g_{w_l,r}$ is the same as above. For $X_{w,r}$, a nonshifted coarse grid point on $\partial\Omega_{i,j}^h$, the inward normal vector of

Algorithm 3.1 Update and wind formula for initialization.

Input: $\text{nbrs}^H(U_{i,j}), U_{i,j}, H$
Output: $U_{i,j}, \mathbf{W}_{i,j}$
 {Compute wind in x direction.}
if $U_{i-1,j} < U_{i+1,j}$ **then**
 $W_x = 1$
else
 $W_x = -1$
end if
 {Compute wind in y direction.}
if $U_{i,j-1} < U_{i,j+1}$ **then**
 $W_y = 1$
else
 $W_y = -1$
end if
 {Compute in solution to (1.3) and define $\tilde{\mathbf{W}}$.}
 $\tilde{U} = C_H(\text{nbrs}^H(U_{i,j}))$
 $a = \min(U_{i-1,j}, U_{i+1,j})$
 $b = \min(U_{i,j-1}, U_{i,j+1})$
if $\tilde{U} < b$ **then**
 $\tilde{\mathbf{W}} = (W_x, 0)$
else if $\tilde{U} < a$ **then**
 $\tilde{\mathbf{W}} = (0, W_y)$
else
 $\tilde{\mathbf{W}} = (W_x, W_y)$
end if
 {Take minimum.}
if $\tilde{U} < U_{i,j}$ **then**
 $U_{i,j} = \tilde{U}$
 $\mathbf{W}_{i,j} = \tilde{\mathbf{W}}$
end if

Algorithm 3.2 Causal sweep update formula for vertically shifted coarse grid node.

Input: $U_{i,j_{m-1}}, U_{i,j_{m+1}}, U_{i,j_m}, \mathbf{W}_{i,j_m}$
Output: U_{i,j_m}
if $\mathbf{W}_{i,j_m} \cdot (0, -1) > 0$ and $U_{i,j_m} < U_{i,j_{m+1}}$ **then**
 $U_{i,j_m} = U_{i,j_{m+1}}$
end if
if $\mathbf{W}_{i,j_m} \cdot (0, 1) > 0$ and $U_{i,j_m} < U_{i,j_{m-1}}$ **then**
 $U_{i,j_m} = U_{i,j_{m-1}}$
end if

$\Omega_{i,j}^h$ is not unique since the coarse grid point is a corner of the subdomain. There are two possibilities for the inward normal vector. Denote them by $\mathbf{n}_{w,r}^1$ and $\mathbf{n}_{w,r}^2$; then

$$g_{w,r}(U_{w,r}^k, \mathbf{W}_{w,r}^k) = \begin{cases} U_{w,r}^k & \text{if } \mathbf{W}_{w,r}^k \cdot \mathbf{n}_{w,r}^1 > 0 \text{ or } \mathbf{W}_{w,r}^k \cdot \mathbf{n}_{w,r}^2 > 0, \\ \infty & \text{otherwise.} \end{cases}$$

Step 3: Solve for u^k in parallel. In parallel for $i, j = 0, \dots, N-1$, we solve

$$(3.11) \quad |\nabla u(x)| = r(x), \quad x \in (iH, (i+1)H) \times (jH, (j+1)H),$$

$$(3.12) \quad u = g, \quad \text{on } \partial([iH, (i+1)H] \times [jH, (j+1)H])$$

via FSM on the grid $\Omega_{i,j}^h$ and g is defined in Step 2.

- Initialize u as described in section 1.1.
- Sweep the grid. To update the solution at each x_{i_l, j_m} , input $\text{nbrs}^h(u_{i_l, j_m})$, u_{i_l, j_m} , and h into Algorithm 3.3. Use one sided differences if x_{i_l, j_m} is a boundary grid node. Here, $\text{nbrs}^h(u_{i_l, j_m}) = \{u_{i_l-1, j_m}, u_{i_l+1, j_m}, u_{i_l, j_m-1}, u_{i_l, j_m+1}\}$.

Denote the solutions after sweeping by u_{i_l, j_m}^k and $\mathbf{w}_{i_l, j_m}^k$ for $l, m = 0, \dots, M$.

After the computations on each subdomain, we will have two or four values for each coarse grid node, depending on whether the point is in a shifted or nonshifted coarse grid. Intuitively, we define the value $u_{i,j}^k$ by the following:

- If the coarse wind and the fine wind flow into the same subdomain $\Omega_{s,t}^h$ from $\Omega_{s',t'}^h$, then we set the value $u_{i,j}^k$ to be the fine grid solution from the subdomain $\Omega_{s',t'}^h$.
- Otherwise we set $u_{i,j}^k$ to be the minimum of the fine grid solutions at the coarse grid point.

A vertically shifted coarse grid node, X_{i,j_m} , is on the boundary of the two subdomains, $\Omega_{i-1,j'}^h$ and $\Omega_{i,j'}^h$. Denote the two possibilities of an inward normal vector by $\mathbf{n}^1 = (-1, 0)$ and $\mathbf{n}^2 = (1, 0)$. Algorithm 3.4 explains how to compute u_{i,j_m}^k at a vertically shifted coarse grid node, X_{i,j_m} . The computations at a horizontally shifted and a nonshifted coarse grid point are similar.

Step 4: Coarse grid updates. Now we compute the coarse grid updates, U^{k+1} . Again since the shifted coarse grids are independent of each other, the values $\{U_{i,j}^{k+1}\}_{i,j}$ and $\{U_{i,j_m}^{k+1}\}_{i,j_m}$ can be computed in parallel for each l and m .

- Initialize U as described in section 1.1.
- Sweep the grid and the update formula at a vertically shifted coarse grid node is given by Algorithm 3.5. The computations at a horizontally shifted and nonshifted coarse grid point are similar. Let $\mathbf{n}^1$ and $\mathbf{n}^2$ be the inward normal vectors as defined in Step 3. We input $\text{nbrs}^H(U_{i,j_m}), \text{nbrs}^H(U_{i,j_m}^k), \mathbf{W}_{i,j_m}^k, \mathbf{w}_{i,j_m}^k$, and H into Algorithm 3.5.

Again we must implement a sequential causal sweep to make sure the coarse grids respect the causality of the solution. Sweep the coarse grids sequentially in each of the four directions once. The update formula is given by inputting $U_{i,j_m-1}, U_{i,j_m+1}, U_{i,j_m}, \mathbf{W}_{i,j_m}$ into Algorithm 3.2 for a vertically shifted coarse grid point. The updates for other coarse grid nodes are defined similarly. Denote the solutions after sweeping by U^{k+1} and $\mathbf{W}^{k+1}$. Repeat Steps 2–4 until convergence.

The method is demonstrated in Figure 7, which shows the contours for the fine grid solution patched together for $r_\epsilon^1 = 1 + .99 \sin(2\pi x) \sin(2\pi y)$ for $k = 0, 2, 4$, and 6. We see the solution contours begin to smooth out after a few iterations.

4. Analysis of the new method. We choose the following model problem to study the choice of weight θ . Let $\Omega = [0, 1] \times [0, H]$. Then we numerically solve via our method

$$(4.1) \quad |\nabla u(x, y)| = 1, \quad (x, y) \in \Omega \setminus \Gamma,$$

$$(4.2) \quad u(x, y) = \sqrt{x^2 + y^2}, \quad (x, y) \in \Gamma,$$

Algorithm 3.3 Update and wind formula for fine grid computations.

Input: $\text{nbrs}^h(u_{i_l, j_m}), u_{i_l, j_m}, h$
Output: $u_{i_l, j_m}, \mathbf{w}_{i_l, j_m}$
 {Compute wind in x direction.}
if $u_{i_{l-1}, j} < u_{i_{l+1}, j}$ **then**
 $a = u_{i_{l-1}, j}$
 $w_x = -1$
else
 $a = u_{i_{l+1}, j}$
 $w_x = 1$
end if
 {Compute wind in y direction.}
if $u_{i, j_{m-1}} < u_{i, j_{m+1}}$ **then**
 $b = u_{i, j_{m-1}}$
 $w_y = -1$
else
 $b = u_{i, j_{m+1}}$
 $w_y = 1$
end if
 {Solve (1.3).}
if $|a - b| < r_{i,j} h$ **then**
 $\tilde{u} = \frac{1}{2} \left(a + b + \sqrt{2r_{i,j}^2 h^2 - (a - b)^2} \right)$
 $\tilde{\mathbf{w}} = (w_x, w_y)$
else
 $\tilde{u} = \min(a, b) + r_{i,j} h$
 if $a < b$ **then**
 $\tilde{\mathbf{w}} = (w_x, 0)$
 else
 $\tilde{\mathbf{w}} = (0, w_y)$
 end if
end if
 {Take minimum.}
if $\tilde{u} < u_{i_l, j_m}$ **then**
 $u_{i_l, j_m} = \tilde{u}$
 $\mathbf{w}_{i_l, j_m} = \tilde{\mathbf{w}}$
end if

where $\Gamma = \{(x, 0) : 0 \leq x \leq 1\} \cup \{(0, y) : 0 \leq y \leq H\}$. The coarse grids can be defined in one set by

$$\Omega^H := \{(iH, jh) : i = 0, 1, \dots, N \text{ and } j = 0, 1, \dots, M\}$$

with $X_{i,j} = (iH, jh)$. The overall fine grid is given by

$$\Omega^h = \{(lh, mh) : l = 0, 1, \dots, NM \text{ and } m = 0, 1, \dots, M\}.$$

The advantage of this problem is that the characteristics can be captured in one sweep of FSM, i.e., an upward right sweep. This fact means we can use a weighted correction

Algorithm 3.4 Update formula for u_{i,j_m}^k and $\mathbf{w}_{i,j_m}^k$ for a vertically shifted coarse grid node.

Input: $u_{i-1M,j_m}^k, u_{i0,j_m}^k, \mathbf{w}_{i-1M,j_m}^k, \mathbf{w}_{i0,j_m}^k, \mathbf{W}_{i,j_m}^k$
output: $u_{i,j_m}^k, \mathbf{w}_{i,j_m}^k$
if $\mathbf{w}_{i-1M,j_m}^k \cdot \mathbf{n}^1 \geq 0, \mathbf{w}_{i0,j_m}^k \cdot \mathbf{n}^1 \geq 0$, and $\mathbf{W}_{i,j_m}^k \cdot \mathbf{n}^1 \geq 0$ **then**
 $u_{i,j_m}^k = u_{i0,j_m}^k$
 $\mathbf{w}_{i,j_m}^k = \mathbf{w}_{i0,j_m}^k$
else if $\mathbf{w}_{i-1M,j_m}^k \cdot \mathbf{n}^2 \geq 0, \mathbf{w}_{i0,j_m}^k \cdot \mathbf{n}^2 \geq 0$, and $\mathbf{W}_{i,j_m}^k \cdot \mathbf{n}^2 \geq 0$ **then**
 $u_{i,j_m}^k = u_{i-1M,j_m}^k$
 $\mathbf{w}_{i,j_m}^k = \mathbf{w}_{i-1M,j_m}^k$
else
 if $u_{i-1M,j_m}^k \leq u_{i0,j_m}^k$ **then**
 $u_{i,j_m}^k = u_{i-1M,j_m}^k$
 $\mathbf{w}_{i,j_m}^k = \mathbf{w}_{i-1M,j_m}^k$
 else
 $u_{i,j_m}^k = u_{i0,j_m}^k$
 $\mathbf{w}_{i,j_m}^k = \mathbf{w}_{i0,j_m}^k$
 end if
end if

Algorithm 3.5 Update formula for weighted corrections for a vertically shifted coarse grid node.

Input: $\text{nbrs}^H(U_{i,j_m}), \text{nbrs}^H(U_{i,j_m}^k), W_{i,j_m}^k, w_{i,j_m}^k, H$
Output: $U_{i,j_m}, \tilde{W}_{i,j_m}$
Compute $\tilde{U}$ and $\tilde{W}$ as in Algorithm 3.1
if $\mathbf{w}_{i,j_m}^k \cdot \mathbf{n}^1 \geq 0, \mathbf{W}_{i,j_m}^k \cdot \mathbf{n}^1 \geq 0$, and $\tilde{\mathbf{W}} \cdot \mathbf{n}^1 \geq 0$ **then**
 $U_{i,j_m} = \theta_{i,j_m}^{k+1} \tilde{U} + u_{i,j_m}^k - \theta_{i,j_m}^{k+1} C_H(\text{nbrs}^H(U_{i,j_m}^k))$
 $\mathbf{W}_{i,j_m} = \mathbf{w}_{i,j_m}^k$
else if $\mathbf{w}_{i,j_m}^k \cdot \mathbf{n}^2 \geq 0, \mathbf{W}_{i,j_m}^k \cdot \mathbf{n}^2 \geq 0$, and $\tilde{\mathbf{W}} \cdot \mathbf{n}^2 \geq 0$ **then**
 $U_{i,j_m} = \theta_{i,j_m}^{k+1} \tilde{U} + u_{i,j_m}^k - \theta_{i,j_m}^{k+1} C_H(\text{nbrs}^H(U_{i,j_m}^k))$
 $\mathbf{W}_{i,j_m} = \mathbf{w}_{i,j_m}^k$
else
 $U_{i,j_m} = u_{i,j_m}^k$
 $\mathbf{W}_{i,j_m} = \mathbf{w}_{i,j_m}^k$
end if

update for every coarse grid node. Let u^f be the overall fine solution on Ω^h . Suppose we allow θ to vary for each coarse grid node and iteration and denote it by $\theta_{i,j}^k$. Then for $i = 1, \dots, N$ and $j = 1, \dots, M$, we have the following coarse grid solver:

$$C_H(U_{i-1,j}, U_{i,j-M}) = \begin{cases} \frac{U_{i-1,j} + U_{i,j-M} + \sqrt{2H^2 - (U_{i-1,j} - U_{i,j-M})^2}}{2}, & j = M, \\ U_{i-1,j} + H & \text{otherwise,} \end{cases}$$

where if $j = 1, \dots, M-1$ we ignore the second argument of the coarse grid solver. Let

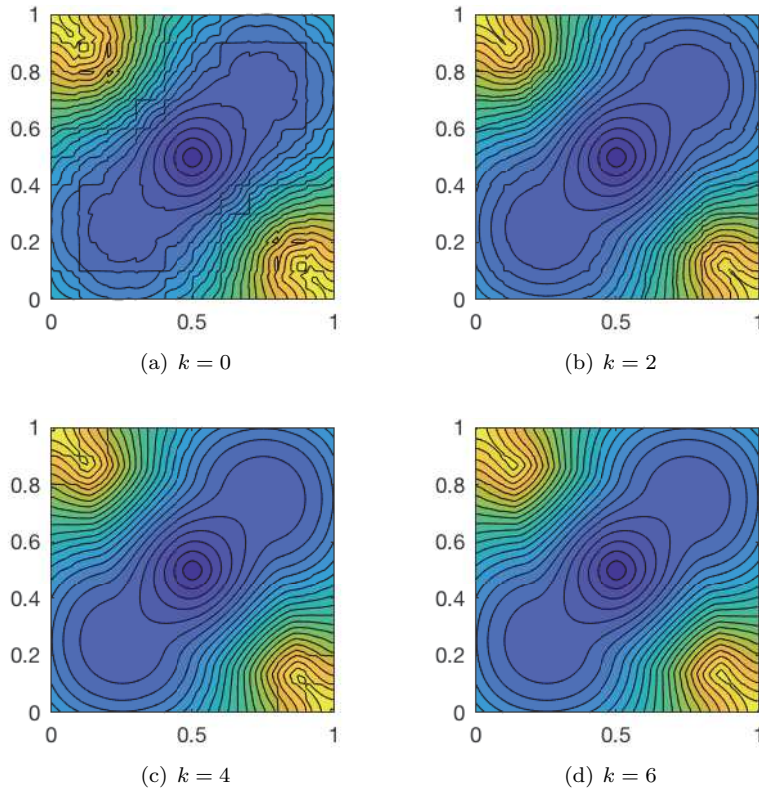


FIG. 7. Contours for fine grid solutions patched together for the slowness function $r_e^1(x, y) = 1 + .99 \sin(2\pi x) \sin(2\pi y)$ where in (a) $k = 0$, (b) $k = 2$, (c) $k = 4$, and (d) $k = 6$.

$U_{i,j}^0 = C_H(U_{i-1,j}^0, U_{i,j-M}^0)$. The weighted update for this problem for $j = 1, \dots, M$ is

$$(4.3) \quad U_{i,j}^{k+1} = \theta_{i,j}^{k+1} \left[C_H(U_{i-1,j}^{k+1}, U_{i,j-M}^{k+1}) - C_H(U_{i-1,j}^k, U_{i,j-M}^k) \right] + u_{i,j}^k$$

with initial conditions

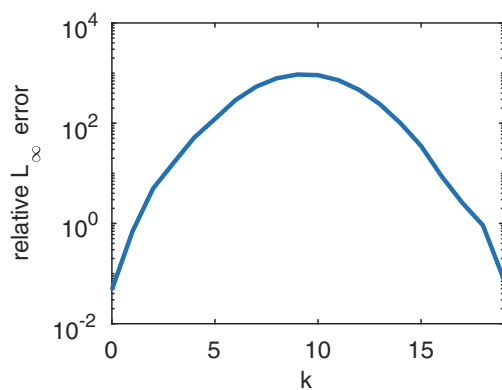
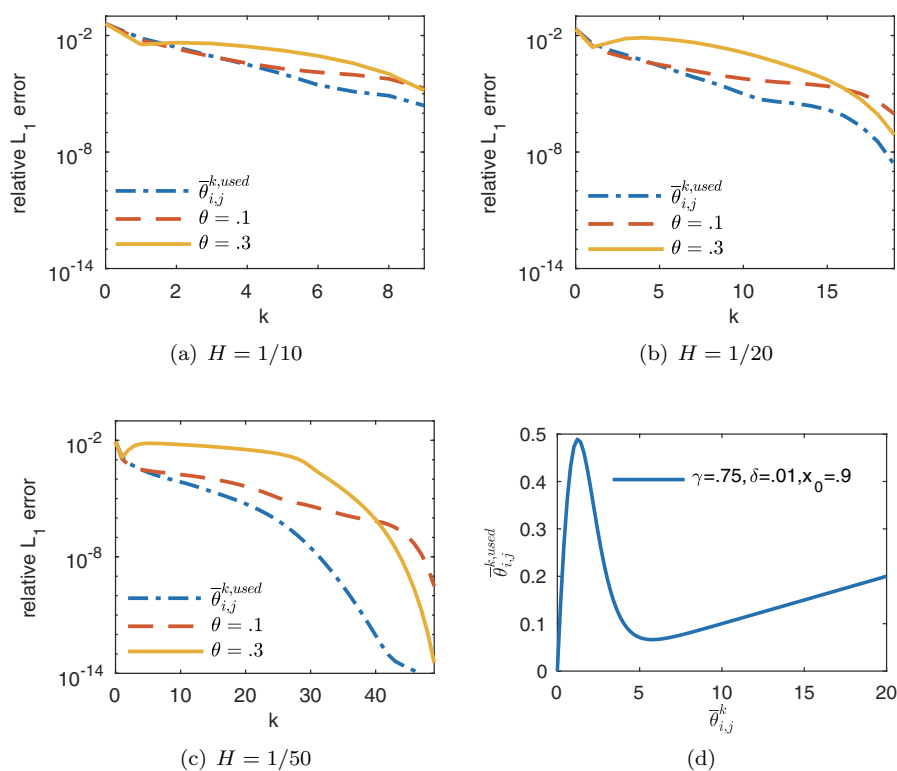
$$(4.4) \quad U_{0,j}^{k+1} = u_{0,j}^f \text{ for } j = 1, \dots, M \text{ and } k = 0, 1, 2, \dots$$

and

$$(4.5) \quad U_{i,0}^{k+1} = u_{i,0}^f \text{ for } i = 1, \dots, N \text{ and } k = 0, 1, 2, \dots$$

Figure 8 shows the L_∞ error plot for $\theta = 1$ which is analogous to the standard parareal method. Note the error is large from the first iteration and increases for later iterations. The error peaks around $k = 10$. This is because as k increases the solutions in each successive subdomain converge to the exact solution, which then allows the maximum error to begin to decrease. If we choose a small value for θ , the solutions converge as seen in Figure 9. However, the convergence may be slow. Next, we study how to choose $\theta_{i,j}^k$ for the coarse grid updates.

4.1. Analysis of θ on model problem. First we prove a theorem that gives an exactness property for the method on this model problem. Let u^f be the overall fine solution on Ω^h .

FIG. 8. $\|U^k - u^f\|_{L_\infty}$ for $\theta = 1$, $H = 1/20$, $h = 1/1000$.FIG. 9. Error plots of $\|U^k - u^f\|_{L_1}$ for specified values of H . In all three examples $h = 1/1000$. For (a), (b), and (c), $\|u^f - u^{exact}\|_{L_1} = 3.79 \times 10^{-4}$. (d) Parameters γ , δ , and x_0 used to estimate θ . For (a)–(c), $\gamma = .75$, $\delta = .01$, $x_0 = .9$.

THEOREM 4.1. Let $U_{i,j}^k$ be given by (4.3). Then for each $j = 1, \dots, M$,

$$U_{i,j}^k = u_{i,j}^f \text{ for } k \geq i.$$

Proof. First note $U_{0,j}^1 = U_{0,j}^0$ and $U_{1,0}^1 = U_{1,0}^0 = u_{1,0}^f$. Now,

$$\begin{aligned} U_{1,j}^1 &= \theta_{1,j}^1 \left[C_H(U_{0,j}^1, U_{1,j-M}^1) - C_H(U_{0,j}^0, U_{1,j-M}^0) \right] + u_{1,j}^0 \\ &= u_{1,j}^0 \\ &= u_{1,j}^f. \end{aligned}$$

The second equality comes from the fact that $u_{1,j}^0$ was computed using only the boundary values.

Now assume

$$(4.6) \quad U_{i,j}^k = u_{i,j}^f \text{ for } k \geq i.$$

Let $k \geq i + 1$. We have

$$U_{i+1,j}^k = \theta_{i+1,j}^k \left[C_H(U_{i,j}^k, U_{i+1,j-M}^k) - C_H(U_{i,j}^{k-1}, U_{i+1,j-M}^{k-1}) \right] + u_{i+1,j}^{k-1}.$$

Now $k \geq i + 1$ and (4.6) imply $U_{i,j}^k = U_{i,j}^{k-1} = u_{i,j}^f$. Also, (4.5) implies

$$U_{i+1,0}^k = U_{i+1,0}^{k-1} = u_{i+1,0}^f.$$

Therefore,

$$U_{i+1,j}^k = u_{i+1,j}^{k-1} = u_{i+1,j}^f,$$

where the second equality comes from the fact that $u_{i+1,j}^{k-1}$ is computed using the values $U_{i,j}^{k-1}$ and (4.6) implies $U_{i,j}^{k-1} = u_{i,j}^f$ for $j = 0, \dots, M - 1$. Thus, we have our desired result. $\square$

Now that the exactness property for the method is proven, we have the following theorem that proves existence of $\theta_{i,j}^k$ for each k such that the sequence of solutions is monotonically decreasing for the model problem. The following fact is used in the proof of the theorem: If $a \leq b \leq c$, then $C_H(a, c) \leq C_H(b, c)$.

THEOREM 4.2. *For $j = 1, \dots, M$ and $i = 1, \dots, N$, there exists $\theta_{i,j}^k$ such that $U_{i,j}^k < U_{i,j}^{k-1}$ and $U_{i,j}^k > u_{i,j}^f$ for all $i > k$.*

Proof. First we note $U_{i,j}^0 > u_{i,j}^f$ and $u_{i,j}^0 \geq u_{i,j}^f$ for $i = 1, \dots, N$ and $j = 1, \dots, M$. We will proceed by induction on k . Let $k = 1$. We will show the theorem holds for all $i > 1$. Either $u_{2,j}^0 > U_{2,j}^0$ or $u_{2,j}^0 \leq U_{2,j}^0$. If $u_{2,j}^0 \leq U_{2,j}^0$, choose $\theta_{2,j}^1 > 0$. Then

$$\begin{aligned} U_{2,j}^1 &= \theta_{2,j}^1 \left[C_H(U_{1,j}^1, U_{2,j-M}^1) - C_H(U_{1,j}^0, U_{2,j-M}^0) \right] + u_{2,j}^0 \\ &< u_{2,j}^0 \\ &\leq U_{2,j}^0, \end{aligned}$$

where the first inequality comes from the fact that $U_{1,j}^1 = u_{1,j}^f < U_{1,j}^0$. If $u_{2,j}^0 > U_{2,j}^0$, then define

$$\overline{m}_{2,j}^1 = \frac{U_{2,j}^0 - u_{2,j}^0}{C_H(U_{1,j}^1, U_{2,j-M}^1) - C_H(U_{1,j}^0, U_{2,j-M}^0)}.$$

Now $\overline{m}_{2,j}^1 > 0$. Thus, if $\theta_{2,j}^1 > \overline{m}_{2,j}^1$, then $U_{2,j}^1 < U_{2,j}^0$. Now let

$$\overline{M}_{2,j}^1 = \frac{u_{2,j}^f - u_{2,j}^0}{C_H(U_{1,j}^1, U_{2,j-M}^1) - C_H(U_{1,j}^0, U_{2,j-M}^0)}.$$

Note $\overline{M}_{2,j}^1 > 0$. If we choose $\theta_{2,j}^1 < \overline{M}_{2,j}^1$, then $U_{2,j}^1 > u_{2,j}^f$.

Now assume $U_{i,j}^1 > U_{i,j}^0$ and $U_{i,j}^1 > u_{i,j}^f$ for all $i > 1$. If $u_{i+1,j}^0 \leq U_{i+1,j}^0$, choose $\theta_{i+1,j}^1 > 0$. Then

$$\begin{aligned} U_{i+1,j}^1 &= \theta_{i+1,j}^1 \left[C_H(U_{i,j}^1, U_{i+1,j-M}^1) - C_H(U_{i,j}^0, U_{i+1,j-M}^0) \right] + u_{i+1,j}^0 \\ &< u_{i+1,j}^0 \\ &\leq U_{i+1,j}^0, \end{aligned}$$

where the first inequality comes from the induction assumption. If $u_{i+1,j}^0 > U_{i+1,j}^0$, then let

$$\overline{m}_{i+1,j}^1 = \frac{U_{i+1,j}^0 - u_{i+1,j}^0}{C_H(U_{i,j}^1, U_{i+1,j-M}^1) - C_H(U_{i,j}^0, U_{i+1,j-M}^0)}.$$

Now $\overline{m}_{i+1,j}^1 > 0$. Thus, if $\theta_{i+1,j}^1 > \overline{m}_{i+1,j}^1$, then $U_{i+1,j}^1 < U_{i+1,j}^0$. Next let

$$\overline{M}_{i+1,j}^1 = \frac{u_{i+1,j}^f - u_{i+1,j}^0}{C_H(U_{i,j}^1, U_{i+1,j-M}^1) - C_H(U_{i,j}^0, U_{i+1,j-M}^0)}.$$

Note $\overline{M}_{i+1,j}^1 > 0$. If we choose $\theta_{i+1,j}^1 < \overline{M}_{i+1,j}^1$, then $U_{i+1,j}^1 > u_{i+1,j}^f$. So if $0 < \theta_{i+1,j}^1 < \overline{M}_{i+1,j}^1$, the theorem holds for $k = 1$.

Assume the theorem holds for k , i.e., $U_{i,j}^k < U_{i,j}^{k-1}$ and $U_{i,j}^k > u_{i,j}^f$ for $i > k$. We want to show it holds $i > k + 1$. Let $i = k + 2$. The induction hypothesis implies $U_{k+1,j}^k > u_{k+1,j}^f$ and Theorem 4.1 implies $U_{k+1,j}^{k+1} = u_{k+1,j}^f$. Thus,

$$C_H(U_{k+1,j}^{k+1}, U_{k+2,j-M}^{k+1}) - C_H(U_{k+1,j}^k, U_{k+2,j-M}^k) < 0.$$

If $u_{k+2,j}^k \leq U_{k+2,j}^k$, choose $\theta_{k+2,j}^{k+1} > 0$. Then

$$\begin{aligned} U_{k+2,j}^{k+1} &= \theta_{k+2,j}^{k+1} \left[C_H(U_{k+1,j}^{k+1}, U_{k+2,j-M}^{k+1}) - C_H(U_{k+1,j}^k, U_{k+2,j-M}^k) \right] + u_{k+2,j}^k \\ &< u_{k+2,j}^k \\ &\leq U_{k+2,j}^k. \end{aligned}$$

If $u_{k+2,j}^k > U_{k+2,j}^k$, let

$$\overline{m}_{k+2,j}^{k+1} = \frac{U_{k+2,j}^k - u_{k+2,j}^k}{C_H(U_{k+1,j}^{k+1}, U_{k+2,j-M}^{k+1}) - C_H(U_{k+1,j}^k, U_{k+2,j-M}^k)}.$$

Note $\overline{m}_{k+2,j}^{k+1} > 0$. If $\theta_{k+2,j}^{k+1} > \overline{m}_{k+2,j}^{k+1}$, then $U_{k+2,j}^{k+1} < U_{k+2,j}^k$.

Next we need $U_{k+2,j}^{k+1} > u_{k+2,j}^f$. Let

$$\overline{M}_{k+2,j}^{k+1} = \frac{u_{k+2,j}^f - u_{k+2,j}^k}{C_H(U_{k+1,j}^{k+1}, U_{k+2,j-M}^{k+1}) - C_H(U_{k+1,j}^k, U_{k+2,j-M}^k)}.$$

Note $\overline{M}_{k+2,j}^{k+1} > 0$. Then if $\theta_{k+2,j}^{k+1} < \overline{M}_{k+2,j}^{k+1}$, $U_{k+2,j}^{k+1} > u_{k+2,j}^f$. So if $u_{k+2,j}^k \leq U_{k+2,j}^k$, choose $0 < \theta_{k+2,j}^{k+1} < \overline{M}_{k+2,j}^{k+1}$. If $u_{k+2,j}^k > U_{k+2,j}^k$, choose $\overline{m}_{k+2,j}^{k+1} < \theta_{k+2,j}^{k+1} < \overline{M}_{k+2,j}^{k+1}$. Therefore, the theorem holds. $\square$

The proof of Theorem 4.2 provides insight on the stability of the method and the optimal choice for the weights $\theta_{i,j}^k$. Let

$$\tilde{m}_{i,j}^k = \begin{cases} 0 & \text{if } \overline{m}_{i,j}^k \leq 0, \\ \overline{m}_{i,j}^k & \text{otherwise.} \end{cases}$$

If $\tilde{m}_{i,j}^k < \theta_{i,j}^k < \overline{M}_{i,j}^k$, we have a monotonically convergent sequence of solutions. The closer we choose $\theta_{i,j}^k$ to $\overline{M}_{i,j}^k$ the more accurate $U_{i,j}^k$ is.

If we analyze the values for $\overline{M}_{i,j}^k$ we see in early iterations that $\overline{M}_{i,j}^k$ can be very small. For example, if $k = 1, h = 1/20, h = 1/1000$, $\min_{X_{i,j} \in \Omega^H} (\overline{M}_{i,j}^k) = 5.6 \times 10^{-3}$. This is a reason why we cannot use the standard parareal method where $\theta = 1$. In practice, we do not know $\overline{M}_{i,j}^k$ a priori since it relies on knowing $u_{i,j}^f$. Therefore, we estimate $\overline{M}_{i,j}^k$ in order to choose $\theta_{i,j}^k$ and create a sequence $U_{i,j}^k$ that converges very quickly to $u_{i,j}^f$. Next, we explain how we estimate $\overline{M}_{i,j}^k$ in practice.

4.2. Estimating $\overline{M}_{i,j}^k$. Recall

$$\overline{M}_{i,j}^k = \frac{u_{i,j}^f - u_{i,j}^{k-1}}{C_H(U_{i-1,j}^k, U_{i,j-M}^k) - C_H(U_{i-1,j}^{k-1}, U_{i,j-M}^{k-1})}$$

and we would like $\tilde{m}_{i,j}^k \leq \theta_{i,j}^k < \overline{M}_{i,j}^k$. Since we do not know $u_{i,j}^f$ a priori, we estimate $\overline{M}_{i,j}^k$ by the following:

$$(4.7) \quad \overline{\theta}_{i,j}^k = \frac{u_{i,j}^{k-1} - u_{i,j}^{k-2}}{C_H(U_{i-1,j}^{k-1}, U_{i,j-M}^{k-1}) - C_H(U_{i-1,j}^{k-2}, U_{i,j-M}^{k-2})}.$$

However, upon implementation this estimation produces very unstable solutions. The values $\overline{\theta}_{i,j}^k$ become extremely large and create sequences of solutions where $U_{i,j}^k \ll u_{i,j}^f$ or $U_{i,j}^k \gg U_{i,j}^{k-1}$. This occurs when the denominator of (4.7) is much smaller than the numerator. We overcome this issue by using a weighted sum in the denominator, i.e.,

$$(4.8) \quad \overline{\theta}_{i,j}^k = \frac{u_{i,j}^{k-1} - u_{i,j}^{k-2}}{\left[\sum_{s=0}^2 \omega_s C_H(U_{i-1,j}^{k-s}, U_{i,j-M}^{k-s}) - C_H(U_{i-1,j}^{k-1-s}, U_{i,j-M}^{k-1-s}) \right] / (\omega_0 + \omega_1 + \omega_2)}.$$

To further ensure that the estimated value $\overline{\theta}_{i,j}^k$ does not become too large we dampen the values if they are beyond a threshold and apply a smooth approximation function. Let

$$(4.9) \quad \sigma(\overline{\theta}_{i,j}^k) = \frac{1}{1 + e^{(\overline{\theta}_{i,j}^k - x_0)/\gamma}}.$$

Then

$$(4.10) \quad \overline{\theta}_{i,j}^{k,used} = \left[\sigma(\overline{\theta}_{i,j}^k) \overline{\theta}_{i,j}^k + (1 - \sigma(\overline{\theta}_{i,j}^k)) \delta \overline{\theta}_{i,j}^k \right]^+,$$

where again $a^+ = \max(0, a)$ and x_0, γ , and δ are parameters chosen experimentally.

Figure 9(d) shows the plot of $\overline{\theta}_{i,j}^k$ versus $\overline{\theta}_{i,j}^{k,used}$, and Figures 9(a) to 9(c) show the error plots for various values of H . In all three examples $h = 1/1000$. We see the advantage of using $\overline{\theta}_{i,j}^{k,used}$ over a fixed value of θ .

4.3. Complexity and speed-up. Let $N = 1/H$ and $M = 1/(Nh)$. Define $A[N, d] := C(2^d(N+1)^d)$ to be the number of flops for FSM where C depends on the characteristics of the given Eikonal equation. Then the number of flops for the computations on all of the coarse grids, the causal sweep, and all of the subdomains is

$$(dM)A[N, d] + 2^d M(N+1)^2 + (N^d)A[M, d].$$

If we solve the Eikonal equation on Ω_h , then the number of flops is given by $A(NM, d)$. Theoretically suppose we have enough processors to compute the solution on each coarse grid and each subdomain in parallel. Then after k iterations the computational time is proportional to $k(A[N, d] + 2^d M(N+1)^2 + A[M, d])$. For our method to achieve speed-up via parallelization, we need

$$k \ll \frac{A[NM, d]}{A[N, d] + 2^d M(N+1)^2 + A[M, d]}.$$

For example, suppose $N = 20$, $M = 100$, and $d = 2$ and we perform 10 sweeping iterations on each coarse grid as well as on each subdomain; then with ideal parallelization we need $k \ll 266$ in order to achieve speed up.

5. Numerical results.¹ Next we present some numerical results computed by the method. Every example is computed on $\Omega = [0, 1]^2$ and Γ is a set of source points chosen in each example. The focus of our examples is demonstrating the reduction in error in a few iterations and the ability to handle some stereotypes of r_ϵ . We report the L_1 relative error in each example, i.e., $\|u^k - u^f\|_{L_1}$, where u^f is the overall fine solution. In every example, $\bar{\theta}_{i,j}^{k,used}$ is chosen so the solution is stable and converges to the overall fine solution.

We choose the examples in section 5.1 to demonstrate the method on smooth slowness functions, where the coarse grids do not sufficiently resolve the oscillations. The point is to show that the coupling scheme is able to correct the coarse grid computation using the fine grid computations. The maze examples in section 5.2 are chosen because the solutions have large changes in the direction of the characteristics. This example highlights that when the initial coarse grid solve captures the direction of the characteristics incorrectly, the scheme is able to correct those directions in later iterations.

The examples in section 5.3 demonstrate the method in the case of multiscale slowness functions with and without scale separation. The goal is to show that, at the expense of parallelization, the method can be used to replace numerical homogenization for a much wider class of multiscale problems.

Finally, we report some simple parallel computations where the method is tested on the Marmousi velocity model and a simple three-dimensional example. All parallel computations were performed on Stampede2 at the Texas Advanced Computing Center using a common shared memory hardware architecture. We used the OpenMP default for loop scheduling to divide the work among the threads. For more details on the specifications of Stampede2 visit <https://www.tacc.utexas.edu/systems/stampede2>.

Nevertheless, we emphasize that the focus of our paper is not on parallel implementation but rather to present a framework for a domain decomposition method with the benefit that the method then can be easily parallelized. We believe that

¹MATLAB/C++ code used to produce all numerical results can be found at <https://github.com/lindsmart/MartinTsayEikonal>.

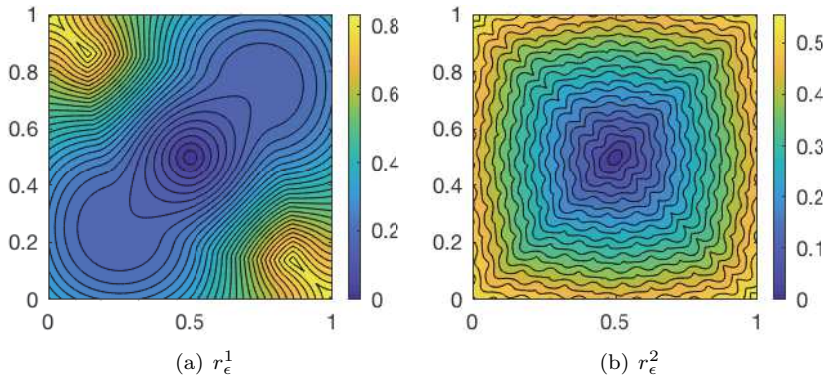


FIG. 10. (a) *Solution contour for $r_\epsilon^1(x, y) = 1 + .99 \sin(2\pi x) \sin(2\pi y)$.* (b) *Solution contour for $r_\epsilon^2(x, y) = 1 + .5 \sin(20\pi x) \sin(20\pi y)$.*

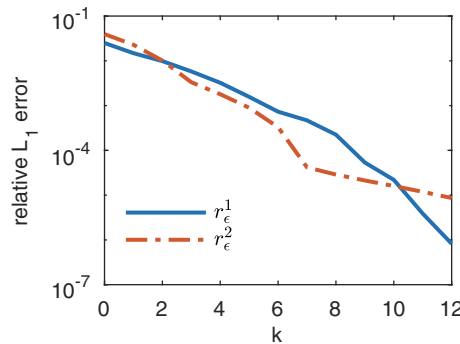


FIG. 11. *Relative L_1 error plots for r_ϵ^1 and r_ϵ^2 for $H = 1/10$ and $h = 1/500$.*

the major addition of our method to this area of research is the stable coupling of the coarse and fine solvers where the stabilization is achieved through the use of data computed “on the fly.”

5.1. Smooth slowness functions. We test the method on two smooth oscillatory continuous slowness functions. Figures 10(a) and 10(b) show the contour plots of the overall fine solution where

$$r_\epsilon^1(x, y) = 1 + .99 \sin(2\pi x) \sin(2\pi y)$$

and

$$r_\epsilon^2(x, y) = 1 + .5 \sin(20\pi x) \sin(20\pi y).$$

Figure 11 shows the error plots for $H = 1/10$ and $h = 1/500$. The method is able to handle small and large changes in direction of the characteristics. We see that the performance is worse in earlier iterations for r_ϵ^1 . This is because the solutions in the upper left and bottom right corners depend on more subdomains than in the r_ϵ^2 case.

5.2. Mazes and obstacles. We show the method’s performance on examples that model optimal paths through a maze. Here, we define $r_\epsilon(x, y) = 1000$ inside the barriers so that all optimal paths choose to avoid them. We also test the method

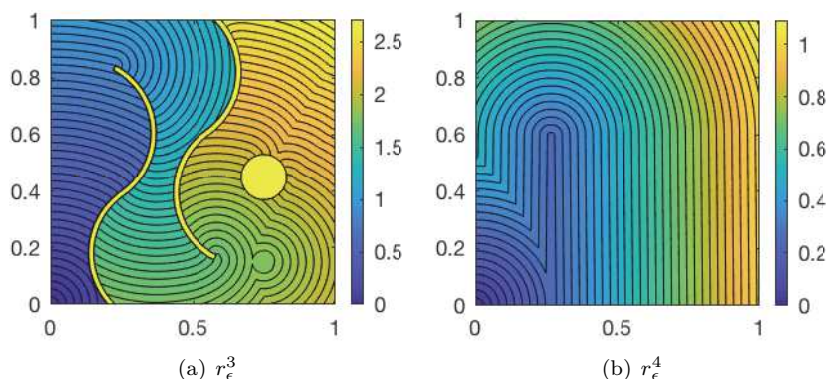


FIG. 12. (a) *Solution contour for r_ϵ^3 .* (b) *Solution contour for r_ϵ^4 .*

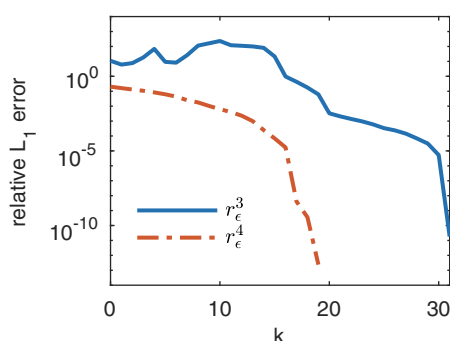


FIG. 13. *Relative L_1 error plots for the slowness functions r_ϵ^3 and r_ϵ^4 for $H = 1/10$ and $h = 1/500$.*

on the case where an obstacle may be a “fast obstacle,” i.e., $r_\epsilon(x, y) = 0.01$ inside and optimal paths near the obstacle choose to go through it. We set $r_\epsilon(x, y) = 1$ everywhere else and let the source point be given by $\Gamma = \{(0, 0)\}$. These examples show the performance of the method on problems when the coarse grid captures the flow of characteristics in the opposite direction. The causal sweeps are critical in order to capture the right flow of characteristics. This is because it is possible that in some of the coarse grids the causality may never be computed correctly. The solution contours for r_ϵ^3 and r_ϵ^4 are shown in Figures 12(a) and 12(b), respectively.

For r_ϵ^3 , there are coarse grid points which coincide with the obstacles as well as points in the obstacles that do not coincide with a coarse grid point. The circle barrier in Figure 12(a) contains an entire subdomain and the other circle is a fast obstacle that is contained entirely in a subdomain. The nonmonotonicity of error is due to the causal sweeps. The method provides speed-up only once the right characteristics have been captured around the barriers. This is seen in Figure 13, where the error starts to decrease monotonically around 20 iterations. For r_ϵ^3 , it takes around $2/H$ iterations for the coarse grid to “see” around the two curved barriers.

For r_ϵ^4 , the fast obstacle is located at $[0.26, 0.27] \times [0, 0.6]$, and it affects the characteristics throughout the majority of the domain, i.e., almost every optimal path in Figure 12(b) must go through the obstacle. This example demonstrates that

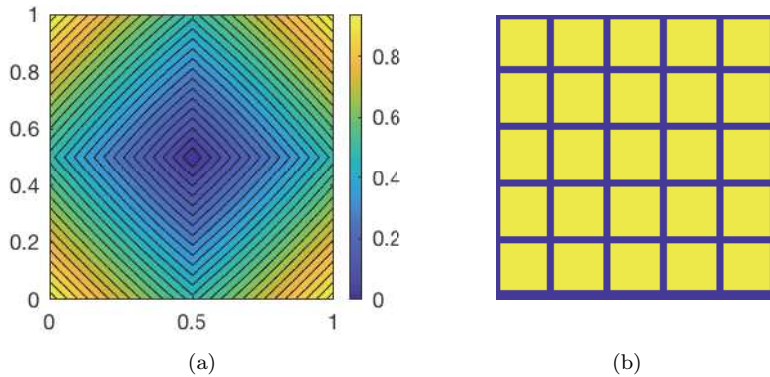


FIG. 14. (a) *Solution contour for the squares slowness function where $h = 1/1400$ and $\epsilon = 1/200$.* (b) *Plot of squares slowness function for $\epsilon = 1/5$.*

the method performs well when there is a large collision of characteristics that occurs in several subdomains.

5.3. Multiscale slowness functions. We show the advantage of the method on multiscale slowness functions. These examples arise in front propagation in multiscale media problems. In our computations, we let the scale epsilon be seven fine grid points, i.e., $\epsilon = 7h$, in order for the fine grid to fully capture the microscale behavior. As mentioned in section 1, one approach for numerically resolving the multiscale behavior in r_ϵ is homogenization. We will first demonstrate our method on an example where the homogenized slowness function, $\bar{r}$, can be computed.

Let the source point be given by $\Gamma = \{(0.5, 0.5)\}$, and define the slowness function as follows: let

$$r(x, y) = \begin{cases} 1 & \text{if } x = 0 \text{ or } y = 0, \\ 2 & \text{otherwise} \end{cases}$$

and define r_ϵ by extending r by periodicity ϵ . Figure 14(b) shows the slowness function for $\epsilon = 1/5$. The homogenized slowness function is anisotropic and is equal to $\bar{r}(\alpha) = (\alpha_1 + \alpha_2)$, where $\alpha = (\alpha_1, \alpha_2)$ and $|\alpha| = 1$. This is due to the optimal paths moving only vertically or horizontally [20].

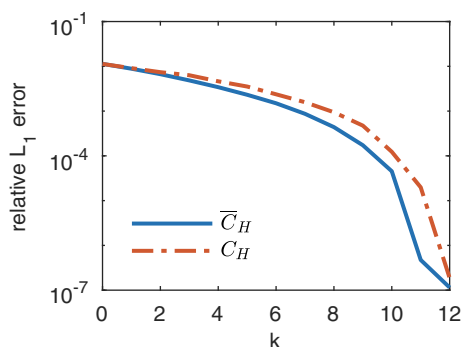
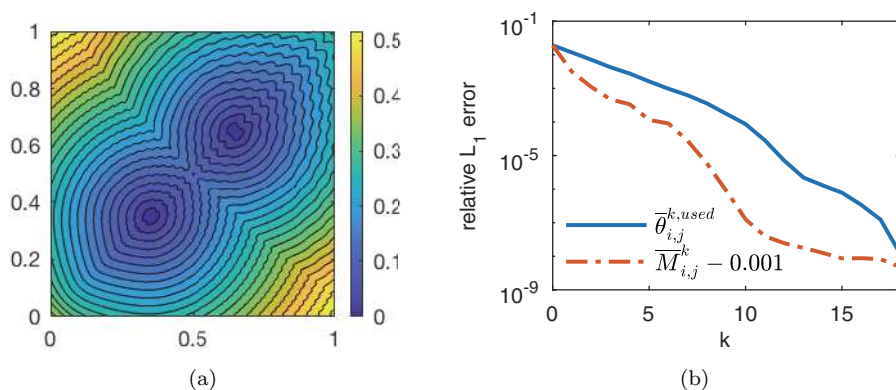
In our computations, we chose $H = 1/14$, $h = 1/1400$, and $\epsilon = 1/200$. The value of r_ϵ on the coarse grid points is always equal to 1. Thus, the coarse grid solver is always solving the equation

$$|\nabla u| = 1.$$

This equation is inaccurate as seen by the shape of the solution contour in Figure 14(a), which is a diamond and not a circle. Suppose in the method we have the coarse solver solve an equation that better describes the macroscale behavior of the solution. Since in this example we know the homogenized equation, on the coarse grid we can solve the homogenized equation

$$(5.1) \quad \frac{1}{\bar{r}(\frac{\nabla u}{|\nabla u|})} |\nabla u| = 1.$$

Denote the homogenized equation coarse solver by $\bar{C}_H$. Figure 15 shows the relative L_1 error plots for both the method that uses the C_H as described in section 3 and

FIG. 15. Relative error L_1 error plots for C_H and $\overline{C}_H$.FIG. 16. Generalization of Figure 10(b): (a) Solution contour for $r_\epsilon(x, y) = 1 + .5 \sin(\frac{\pi x}{\epsilon}) \sin(\frac{\pi y}{\epsilon})$ where $\epsilon = \frac{|x|+|y|+0.001}{50}$. (b) Relative L_1 error plot for the given r_ϵ where the estimated $\overline{\theta}_{i,j}^{k,used}$ and $\overline{M}_{i,j}^k - 0.001$ are used.

the method that uses $\overline{C}_H$ in place of C_H . As expected, we can see the method that uses $\overline{C}_H$ performs better.

Next, we demonstrate the method on a generalization of r_ϵ^2 as defined in section 5.1. Notice in Figure 10(b) that we can see the rough shape of the contours of the solution to the homogenized equation. Let

$$r_\epsilon = 1 + .5 \sin\left(\frac{\pi x}{\epsilon}\right) \sin\left(\frac{\pi y}{\epsilon}\right).$$

For r_ϵ^2 , $\epsilon = 1/20$. Now suppose we let ϵ vary throughout the domain, i.e., the problem cannot be solved via homogenization. Define

$$\epsilon = \frac{|x| + |y| + 0.001}{50}$$

and $\Gamma = \{(0.35, 0.35), (0.65, 0.65)\}$. Then ϵ is very small near $(0, 0)$ and increases as we move diagonally up and right through the domain. Figure 16(a) shows the solution contour for this given r_ϵ . Since u^f can be computed a priori, we compare the error plots of our method where we use formula (4.10) and $\overline{M}_{i,j}^k - 0.001$ as the choice of

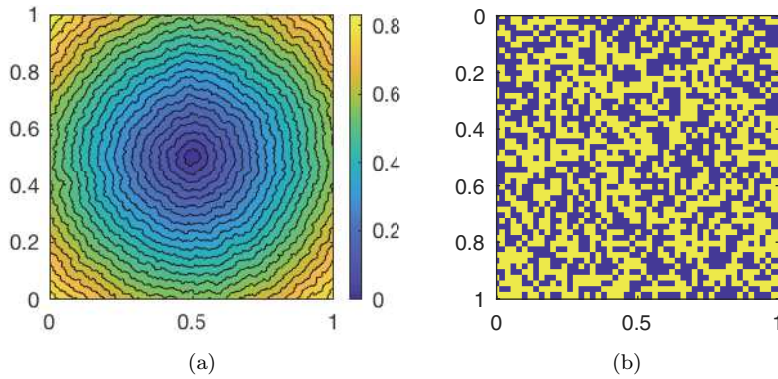


FIG. 17. (a) *Solution contour plot of random periodic checkboard of scale ϵ .* (b) *Plot of random slowness function.*

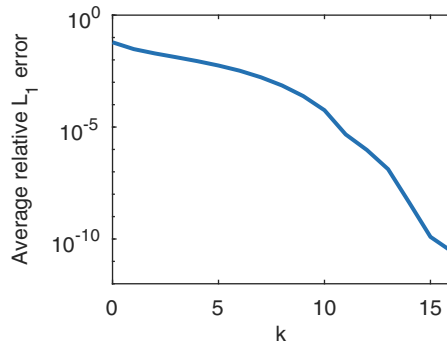


FIG. 18. *Relative L_1 error plot for random r_ϵ .*

weights in the method. In this example, $H = 1/14$ and $h = 1/1400$. The error plots in Figure 16(b) suggest that the proposed formula (4.10) has room for improvement in estimating $\overline{M}_{i,j}^k$. For example, if $X_{i,j} = (0, 0)$, we have

$$\min_k (|\overline{M}_{i,j}^k - 0.001 - \overline{\theta}_{i,j}^{k,used}|) = 0.1053,$$

but

$$\max_k (|\overline{M}_{i,j}^k - 0.001 - \overline{\theta}_{i,j}^{k,used}|) = 2.508 \times 10^5.$$

Finally, we show the results of our method in a case where the values of the slowness function are random. We follow the setup of the random slowness function in [20]. Consider a periodic checkerboard where the slowness function is either 1 or 2 with probability $1/2$. Let the scale of the periodicity be ϵ . A solution contour and the plot of a random slowness function are shown in Figures 17(a) and 17(b), respectively. In [20], the authors showed experimentally the homogenized slowness function, $\overline{r}$, is isotropic and its value is a little less than 1. Figure 18 shows the plot of the average error over 20 trials where $H = 1/14$ and $h = 1/1400$.

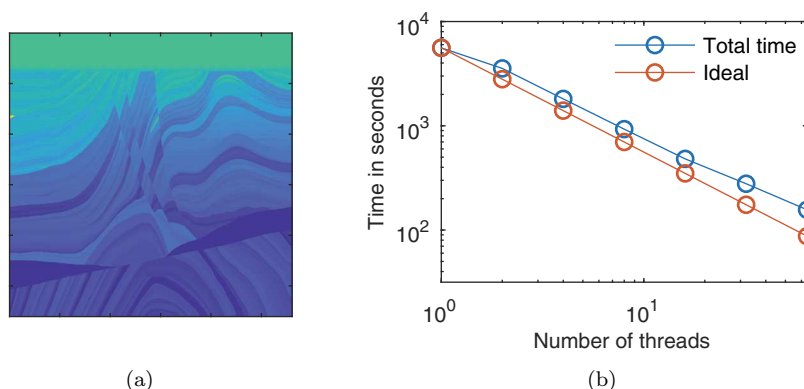


FIG. 19. (a) Reciprocal of the Marmousi velocity displayed as an image. (b) Execution time as a function of the number of threads on an 11201×11201 grid with 14^2 fine subdomains.

TABLE 1

Time for each method on an 11201×11201 grid with 14^2 subdomains using 64 threads where the tolerance prescribed in the new method is the relative L_1 error.

	FSM	New method (1×10^{-5})	New method (1×10^{-16})
Time (sec)	1016.08	132.42	169.20

TABLE 2

Comparison of the number of sweeping iterations in the FSM and the new method. Each sweeping iteration contains four sweeps in each sweeping direction.

FSM	Average number of coarse sweeps per coarse grid	Average number of fine sweeps per fine subdomain
14	2.56	3.13

5.4. Parallel and three-dimensional simulations. We study the performance of a parallelization of the method on the Marmousi velocity model.² This model is a data set used in seismic imaging and has different scales and many discontinuities. It is widely used to test wave propagation algorithms in the seismic imaging community. We define r_ϵ as the reciprocal of the Marmousi velocity. Figure 19(a) displays r_ϵ as an image. We test the scaling of the method on an 11201×11201 grid with 14^2 fine subdomains. Figure 19(b) shows the execution time of the method as a function of the number of threads.

Table 1 reports the time comparisons with the FSM and our new method. We report the times of the new method with respect to L_1 difference with the overall fine solution computed with FSM. With relative L_1 error of 1×10^{-5} , we get a speed up factor close to 7.7. Finally, we report the number of sweeps needed for convergence of FSM on the coarse grids and fine subdomains during each iteration of the method in Table 2. For the fine subdomains, the characteristic structure is less complex than that of the overall fine grid. Thus, we do not need the same amount of sweeps that the original FSM needs on the entire fine grid.

Finally, we verify the implementation of the method in three dimensions. Note that in order to obtain the boundary conditions for each subdomain, we need $(K+1)^2$

²Marmousi velocity data set source: <http://www.agl.uh.edu/downloads/downloads.htm>.

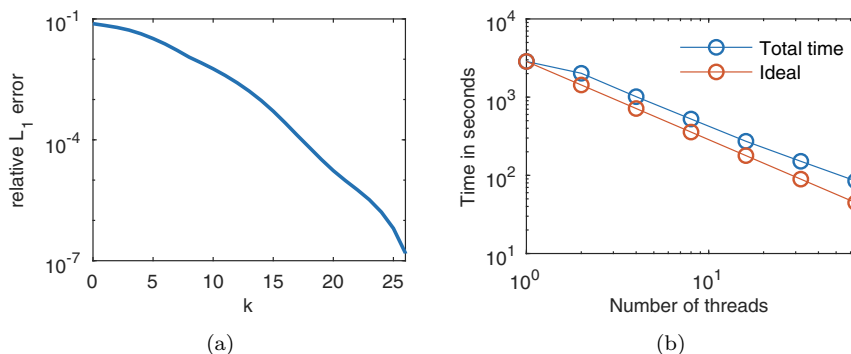


FIG. 20. (a) Relative L_1 error plot for three-dimensional problem with slowness function r_ϵ^{3D} . (b) Execution time as a function of the number of threads on a 301^3 grid with 10^3 fine subdomains.

coarse grid nodes on each face of a subdomain. This gives us a total of $3K(K-1)+1$ shifted coarse grids. In this example, we place the source point at the origin, and we use the slowness function, $r_\epsilon^{3D}(x, y, z) = 1$, except for four spherical obstacles. Centered at the points $(.5, .4, .4)$ and $(.3, .3, .3)$ are “fast” spheres of radius 0.025 where $r_\epsilon^{3D}(x, y, z) = 0.001$ and centered at the points $(.7, .7, .5)$ and $(.6, .3, .6)$ are “slow” spheres of radius 0.025 where $r_\epsilon^{3D}(x, y, z) = 10$. Figure 20(a) shows the error plot for this slowness function when $H = 1/10$ and $h = 1/300$. We also test the scaling of the method in three dimensions on a 301^3 grid with 10^3 fine subdomains. Figure 20(b) shows the execution time of the method as a function of the number of threads.

6. Summary and conclusion. In this paper, we presented a new domain decomposition algorithm for solving boundary value Eikonal equations. We develop a framework that couples a coarse and a fine grid solver to propagate information from the subdomains into the coarse level. The use of the coarse solver distinguishes our method from other existing domain decomposition algorithms. The coarse grid is initialized using FSM, and the values and wind directions are used to define the boundary conditions for the subdomains. Next, we perform fine grid computations in each subdomain in parallel. In our coarse grid updates we apply an adapted weighted parareal scheme to speed up convergence. A causality sweep is performed after each coarse grid update in order to ensure the wind directions are captured correctly.

By clever choice of the weight, it is possible to stabilize parareal-like iterative methods. The weight function $\theta_{i,j}^k$ acts as a multiplicative correction to the coarse operator C_H so that θC_H ideally approximates the fine scale solution operator over a subdomain. At each coarse grid node, $\theta_{i,j}^k$ is computed using previous coarse and fine grid solutions. It is an estimate of $\overline{M}_{i,j}^k$ which is defined to be the upper bound for $\theta_{i,j}^k$ to create a monotonically decreasing sequence of solutions for the model problem. We show via numerical examples on a model problem that the choice of $\theta_{i,j}^k$ stabilizes the method and using a variable θ has advantages over a fixed value. By improving the estimate of $\overline{M}_{i,j}^k$, it is possible to further increase the speed-up of the method.

We demonstrated the method on several classes of slowness functions showing that it performs well on general types of r_ϵ including multiscale slowness functions where homogenization cannot be applied. The errors decrease to an acceptable tolerance well within the limit of theoretical speed-up. Thus, we can solve efficiently through

parallelization multiscale problems beyond the conventional multiscale methods. The example in Figure 14 gives us a direction for future work. We would like to use the coarse and fine grid computations to estimate the effective slowness function “on the fly,” which could further speed up the method based on the evidence in Figure 15.

Acknowledgments. The authors acknowledge the Texas Advanced Computing Center at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper (<http://www.tacc.utexas.edu>).

REFERENCES

- [1] G. ARIEL, S. J. KIM, AND R. TSAI, *Parareal multiscale methods for highly oscillatory dynamical systems*, SIAM J. Sci. Comput., 38 (2016), pp. A3540–A3564, <https://doi.org/10.1137/15M1011044>.
- [2] G. ARIEL, H. NGUYEN, AND R. TSAI, *θ -Parareal Schemes*, arXiv:1704.06882, 2017.
- [3] G. BAL, *On the convergence and the stability of the parareal algorithm to solve partial differential equations*, in Domain Decomposition Methods in Science and Engineering, Lect. Notes Comput. Sci. Eng. 40, Springer, Berlin, 2005, pp. 425–432, https://doi.org/10.1007/3-540-26825-1_43.
- [4] G. BAL AND Y. MADAY, *A “parareal” time discretization for non-linear PDE’s with application to the pricing of an American put*, in Recent Developments in Domain Decomposition Methods (Zürich, 2001), Lect. Notes Comput. Sci. Eng. 23, Springer, Berlin, 2002, pp. 189–202, https://doi.org/10.1007/978-3-642-56118-4_12.
- [5] G. BARLES AND P. E. SOUGANIDIS, *Convergence of approximation schemes for fully nonlinear second order equations*, Asymptot. Anal., 4 (1991), pp. 271–283.
- [6] M. BREUSS, E. CRISTIANI, P. GWOSDEK, AND O. VOGEL, *An adaptive domain-decomposition technique for parallelization of the fast marching method*, Appl. Math. Comput., 218 (2011), pp. 32–44, <https://doi.org/10.1016/j.amc.2011.05.041>.
- [7] S. CACACE, E. CRISTIANI, M. FALCONE, AND A. PICARELLI, *A patchy dynamic programming scheme for a class of Hamilton-Jacobi-Bellman equation*, SIAM J. Sci. Comput., 34 (2012), pp. A2625–A2649, <https://doi.org/10.1137/110841576>.
- [8] A. CHACON AND A. VLADIMIRSKY, *Fast two-scale methods for Eikonal equations*, SIAM J. Sci. Comput., 34 (2012), pp. A547–A578, <https://doi.org/10.1137/10080909X>.
- [9] A. CHACON AND A. VLADIMIRSKY, *A parallel two-scale method for Eikonal equations*, SIAM J. Sci. Comput., 37 (2015), pp. A156–A180, <https://doi.org/10.1137/12088197X>.
- [10] M. G. CRANDALL AND P.-L. LIONS, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.
- [11] M. DETRIXHE AND F. GIBOU, *Hybrid massively parallel fast sweeping method for static Hamilton-Jacobi equations*, J. Comput. Phys., 322 (2016), pp. 199–223, <https://doi.org/10.1016/j.jcp.2016.06.023>.
- [12] M. DETRIXHE, F. GIBOU, AND C. MIN, *A parallel fast sweeping method for the Eikonal equation*, J. Comput. Phys., 237 (2013), pp. 46–55, <https://doi.org/10.1016/j.jcp.2012.11.042>.
- [13] W. E AND B. ENGQUIST, *The heterogeneous multiscale methods*, Commun. Math. Sci., 1 (2003), pp. 87–132.
- [14] M. J. GANDER AND E. HAIRER, *Analysis for parareal algorithms applied to Hamiltonian differential equations*, J. Comput. Appl. Math., 259 (2014), pp. 2–13, <https://doi.org/10.1016/j.cam.2013.01.011>.
- [15] T. HAUT AND B. WINGATE, *An asymptotic parallel-in-time method for highly oscillatory PDEs*, SIAM J. Sci. Comput., 36 (2014), pp. A693–A713, <https://doi.org/10.1137/130914577>.
- [16] F. LEGOLL, T. LELIÈVRE, AND G. SAMAËY, *A micro-macro parareal algorithm: Application to singularly perturbed ordinary differential equations*, SIAM J. Sci. Comput., 35 (2013), pp. A1951–A1986, <https://doi.org/10.1137/120872681>.
- [17] J.-L. LIONS, Y. MADAY, AND G. TURINICI, *A “parareal” in time discretization of PDE’s*, C. R. Math. Acad. Sci. Ser. I, 332 (2001), pp. 661–668, [https://doi.org/10.1016/S0764-4442\(00\)01793-6](https://doi.org/10.1016/S0764-4442(00)01793-6).
- [18] S. LUO, Y. YU, AND H. ZHAO, *A new approximation for effective Hamiltonians for homogenization of a class of Hamilton-Jacobi equations*, Multiscale Model. Simul., 9 (2011), pp. 711–734, <https://doi.org/10.1137/100799885>.

- [19] Y. MADAY, *The parareal in time algorithm*, in Substructuring Techniques and Domain Decomposition Methods, Saxe-Coburg Publications, Stirlingshire, UK, 2010, pp. 19–44.
- [20] A. M. OBERMAN, R. TAKEI, AND A. VLADIMIRSKY, *Homogenization of metric Hamilton-Jacobi equations*, Multiscale Model. Simul., 8 (2009), pp. 269–295, <https://doi.org/10.1137/080743019>.
- [21] E. ROUY AND A. TOURIN, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884, <https://doi.org/10.1137/0729053>.
- [22] J. A. SETHIAN, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 1591–1595, <https://doi.org/10.1073/pnas.93.4.1591>.
- [23] G. A. STAFF AND E. M. RØ NQUIST, *Stability of the parareal algorithm*, in Domain Decomposition Methods in Science and Engineering, Lect. Notes Comput. Sci. Eng. 40, Springer, Berlin, 2005, pp. 449–456, https://doi.org/10.1007/3-540-26825-1_46.
- [24] Y.-H. R. TSAI, L.-T. CHENG, S. OSHER, AND H.-K. ZHAO, *Fast sweeping algorithms for a class of Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 41 (2003), pp. 673–694, <https://doi.org/10.1137/S0036142901396533>.
- [25] J. N. TSITSIKLIS, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538, <https://doi.org/10.1109/9.412624>.
- [26] J. YANG AND F. STERN, *A highly scalable massively parallel fast marching method for the Eikonal equation*, J. Comput. Phys., 332 (2017), pp. 333–362, <https://doi.org/10.1016/j.jcp.2016.12.012>.
- [27] H. ZHAO, *A fast sweeping method for Eikonal equations*, Math. Comp., 74 (2005), pp. 603–627, <https://doi.org/10.1090/S0025-5718-04-01678-3>.
- [28] H. ZHAO, *Parallel implementations of the fast sweeping method*, J. Comput. Math., 25 (2007), pp. 421–429.