ICES REPORT 18-07

May 2018

A multiscale domain decomposition algorithm for boundary value problems for Eikonal equations

by

Lindsay Martin, Richard Tsai

The Institute for Computational Engineering and Sciences
The University of Texas at Austin
Austin, Texas 78712

Reference: Lindsay Martin, Richard Tsai, "A multiscale domain decomposition algorithm for boundary value problems for Eikonal equations," ICES REPORT 18-07, The Institute for Computational Engineering and Sciences, The University of Texas at Austin, May 2018.

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

LINDSAY MARTIN† AND RICHARD TSAI ‡

Abstract. In this paper, we present a new multiscale domain decomposition algorithm for computing solutions of static Eikonal equations. The new method is an iterative two-scale method that uses a parareal-like update scheme in combination with standard Eikonal solvers. The purpose of the two scales is to accelerate convergence and maintain accuracy. We adapt a weighted version of the parareal method for stability and the optimal weights are studied via a model problem. Numerical examples are given to demonstrate the method.

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:

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

$$u(x) = g(x),\ x \in \Gamma \subset \partial\Omega \tag{1.2}$$

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. 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 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) [20, 22] and the Fast Marching Method (FMM) [21, 18] 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) "turn 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 [14, 16]. The goal of homogenization is to derive an effective function, $\overline{r}$, that accurately describes the effective properties of r_ϵ in the solution. Once $\overline{r}$ 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

*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 (lmartin@math.utexas.edu).

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

nontrival 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 combines features from parareal methods and standard Eikonal solvers in order acheive speed up and maintain accuracy. A set of coarse grids is used to set up boundary conditions for the subdomains. After each subdomain is processed in parallel, the method uses are 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 Fast Sweeping Methods, 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 $F : \Omega \to \mathbb{R}$ and its direction of travel is the control of the system. Let $g : \partial\Omega \to \mathbb{R}$ be the exit-time penalty charged at the boundary. Then the value function $u(x)$ is defined to the minimum time it takes to travel from x to $\partial\Omega$. In [8], it is shown 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 $\mathbf{x}_{i,j}$. For the sake of notation, we will omit the numerical solution's dependence on the grid size h. We use a Godonuv upwind scheme to discretize the Eikonal equation at points in the interior of the computational domain [17]:

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

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) 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 $h \to 0$ [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 [17]. However, it is important to take advantage of the causality of

the solution. In the fast marching method (FMM) [21, 18], 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 fast sweeping method (FSM) [20, 22] 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 the all 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:

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

The alternating ordering of sweeping ensures that all the directions of characteristics are captured. In [22] it is shown that with the first order Godonuv upwind scheme, 2^d sweeps is 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 fast sweeping method is $O(N)$ 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 fast sweeping method 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 [23], 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 [9] the authors introduce a method 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.

One of the most recent domain decomposition algorithm is presented in [7]. The method is a parallelization of the Heap Cell method (HCM) [6]. 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. 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.

2. Overview of parareal methods. Parareal methods [13, 4] were developed to parallelize numerical computations of the solutions to ODEs of the form

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

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

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

with initial conditions

$$u_0^k = u_0,\ k = 0, 1, 2, \dots N \tag{2.3}$$

The zeroth iteration is given by

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

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 [15] . The method only provides speed up if k is much smaller than N.

The method is generally unstable for hyperbolic problems and problems with imaginary eigenvalues [19, 3]. Parareal methods for highly oscillatory ODEs can be found in [1, 11]. In [10], analysis of the parareal method on a class of ODEs originating in Hamiltonian dynamical systems is presented, and in [12] 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

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

which simplifies to

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

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 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:

$$|u_x| = r(x),\ 0 < x < 1, \tag{3.1}$$

$$u(0) = u(1) = 0, \tag{3.2}$$

where

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

Figure 3.1 shows the plot of the slowness function r_ϵ. Let the coarse grid be defined by

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

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

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

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

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

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 kth 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 kth iteration at the fine grid node X_{i_m}. For each coarse grid node , X_i, $i = 1, \ldots, N-1$, 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 kth iteration at X_i which we will define in step 3. The method is as follows:

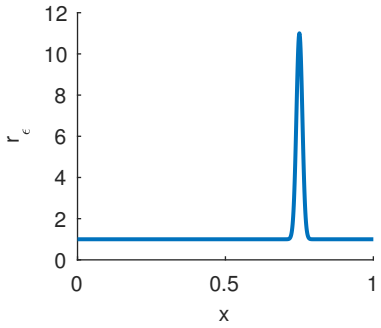


FIG. 3.1. $r_\epsilon(x) = 1 + 10e^{\frac{(x-.75)^2}{2(.01)^2}}$.

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 $i = 0, 1, \ldots, N-1$, we solve via FSM on Ω_i^h

$$|u_x| = r(x), \; x \in [iH, (i+1)H] \tag{3.5}$$

with the boundary conditions described in step 2 . Denote the solutions after sweeping by $u_{i_m}^k$ for $m = 0, \ldots, M$. 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, $i = 1, \ldots, N-1$, 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 3.2a and u^0 is plotted in Figure 3.2b .

Step 4: Coarse grid updates. Now we compute U^{k+1}. We use the previous coarse and fine wind directions to determine whether or not 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 $\tilde{W} = 1$.

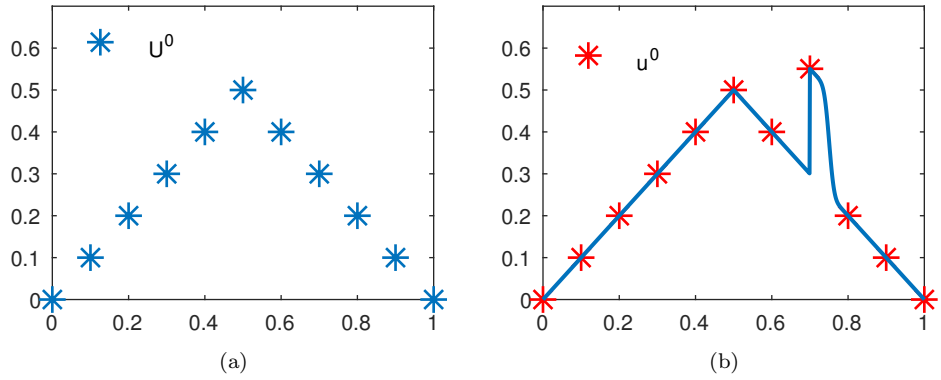


FIG. 3.2. *(a) Plot of U^0. (b) Plot of u^0 using U^0 as boundary conditions as defined in step 2.*

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

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

  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 3.3a, 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 3.4 and 3.5 show the next two iterations of the method which converges at $k = 3$. Next, we introduce the method in two dimensions in more detail.

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

$$|\nabla u(x)| = r(x), \quad x \in \Omega = [0,1]^2 \tag{3.7}$$

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

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

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

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

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

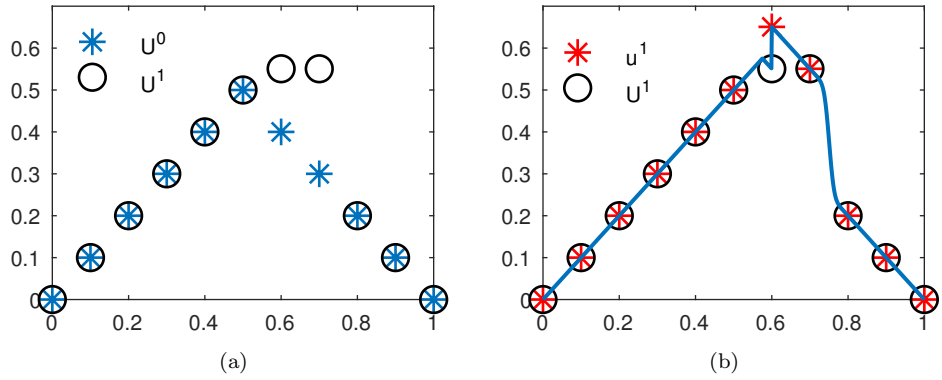


FIG. 3.3. *(a) Plot of U^0 and U^1. (b) Plot of u^1 using U^1 as boundary conditions as defined in step 2.*

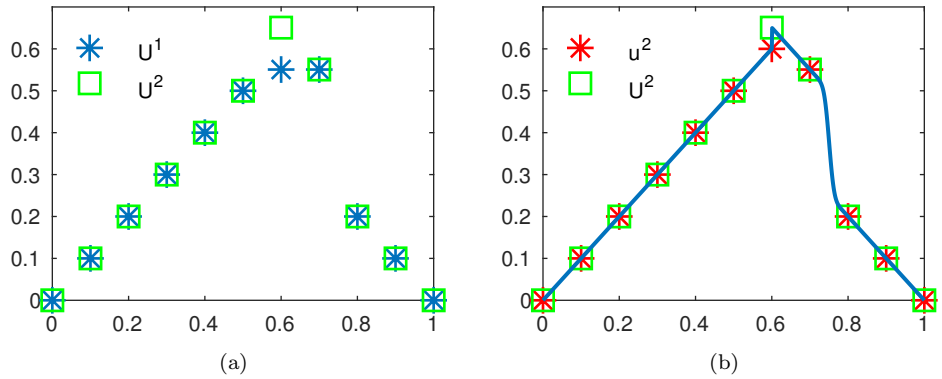


FIG. 3.4. *(a) Plot of U^1 and U^2. (b) Plot of u^2 using U^2 as boundary conditions as defined in step 2.*

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

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

where $y_0 = mh$ for $m = 1, \ldots, M-1$. The shifted grids are demonstrated in Figure 3.6. Next we define the fine grids on the subdomains for $i, j = 0, 1, \ldots, N-1$:

$$\Omega^h_{i,j} = \{(lh + iH, mh + jH) : 0 \le l, m \le M\}.$$

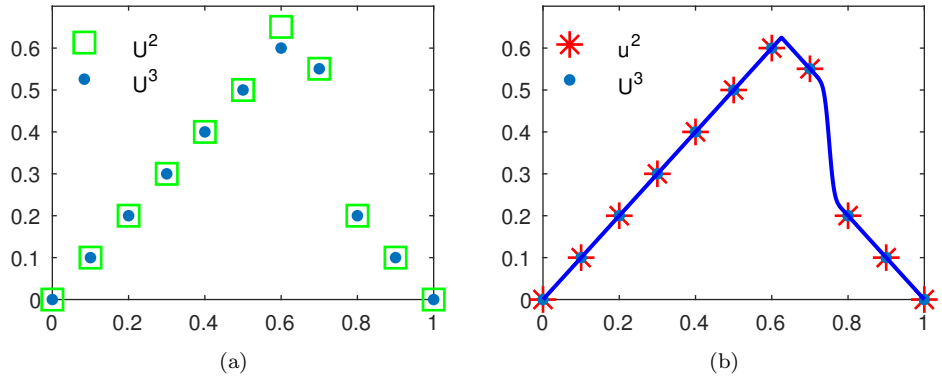


FIG. 3.5. *(a) Plot of U^2 and U^3. (b) Plot of u^3 using U^3 as boundary conditions as defined in step 2. We see that the method has converged and $U^3 = u^3$.*

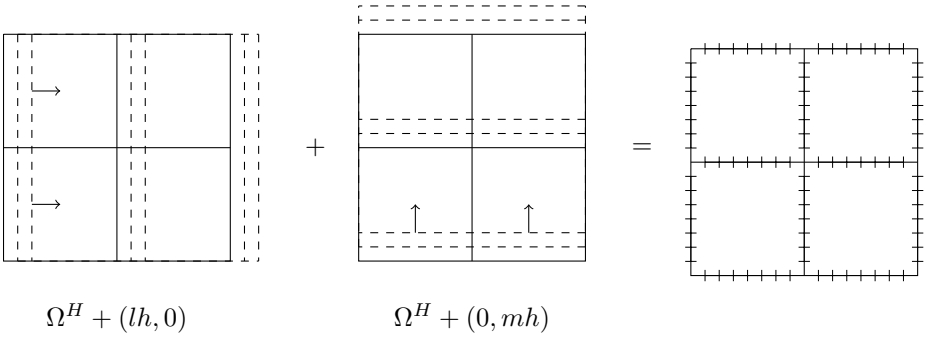


FIG. 3.6. *Shifted coarse grids in two dimensions*

The notation for the two dimensional problem is as follows:

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

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):

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

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:

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

Step 1: Initialize coarse grids in parallel. Since the shifted coarse grids are independent of each other, the values $\{U_{i_l,j}^0\}_{i,j}$ 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 subsection 1.1.
- Sweep the grid as described in subsection 1.1. Algorithm 3.1 explains the update formula as well as how to compute $\mathbf{W}_{i,j}^0$. 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 inputing $U_{i,j_{m-1}}, U_{i,j_{m+1}}, U_{i,j_m}, \mathbf{W}_{i,j_m}$ 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 non-shifted coarse grid point on $\partial\Omega_{i,j}^h$, the inward normal vector of $\Omega_{i,j}^h$ is not unique since the coarse grid point is a corner of the subdomain. There

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

are two possibilities for the inward normal vector. Denote them by $\mathbf{n}^1_{w,r}$ and $\mathbf{n}^2_{w,r}$, then

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

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

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

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

via FSM on the grid $\Omega^h_{i,j}$ and g is defined in step 2.

- Initialize u as described in subsection 1.1
- Sweep the grid. To the 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 $u^k_{i_l,j_m}$ and $\mathbf{w}^k_{i_l,j_m}$ 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 non-shifted coarse grid. Intuitively, we define the value $u^k_{i,j}$ by the following:

- If the coarse wind and the fine wind flow into the same subdomain $\Omega^h_{s,t}$ from $\Omega^h_{s',t'}$, then we set the value $u^k_{i,j}$ to be the fine grid solution from the subdomain $\Omega^h_{s',t'}$.
- Otherwise we set $u^k_{i,j}$ 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, $\Omega^h_{i-1,j'}$ and $\Omega^h_{i,j'}$. 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^k_{i,j_m} at a vertically shifted coarse grid node, X_{i,j_m}. The computations at a horizontally shifted and non shifted 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 $\{U^{k+1}_{i_l,j}\}_{i,j}$ and $\{U^{k+1}_{i,j_m}\}_{i,j}$ can be computed in parallel for each l and m.

- Initialize U as described in subsection 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 non shifted 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 $\text{nbrs}^H(U_{i,j_m}), \text{nbrs}^H(U^k_{i,j_m}), \mathbf{W}^k_{i,j_m}, \mathbf{w}^k_{i,j_m}$ 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 inputing $U_{i,j_{m-1}}, U_{i,j_{m+1}}, U_{i,j_m}$, $\mathbf{W}_{i,j_m}$ 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 3.7 which shows the contours for the fine grid solution patched together for $r^1_\epsilon = 1 + .99 \sin(2\pi x) \sin(2\pi y)$ 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

$$|\nabla u(x, y)| = 1, \quad (x, y) \in \Omega \backslash \Gamma \tag{4.1}$$

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

Algorithm 3.3 Update and wind formula for fine grid compuations

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**
  $\tilde{u} = \frac{1}{2}\left(a + b + \sqrt{2r_{i,j}^2h^2 - (a-b)^2}\right)$
  $\tilde{\mathbf{w}} = (w_x, w_y)$
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 $\Gamma = \{(x,0) : 0 \le x \le 1\} \cup \{(0,y) : 0 \le y \le H\}$. The coarse grids can be defined in one set by

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

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

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

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^k_{i,j_m} and $\mathbf{w}^k_{i,j_m}$ for a vertically shifted coarse grid node

Input: $u^k_{i-1_M,j_m}, u^k_{i_0,j_m}, \mathbf{w}^k_{i-1_M,j_m}, \mathbf{w}^k_{i_0,j_m}, \mathbf{W}^k_{i,j_m}$
output: $u^k_{i,j_m}, \mathbf{w}^k_{i,j_m}$
if $\mathbf{w}^k_{i-1_M,j_m}\cdot\mathbf{n}^1 \geq 0, \mathbf{w}^k_{i_0,j_m}\cdot\mathbf{n}^1 \geq 0$, and $\mathbf{W}^k_{i,j_m}\cdot\mathbf{n}^1 \geq 0$ **then**
  $u^k_{i,j_m} = u^k_{i_0,j_m}$
  $\mathbf{w}^k_{i,j_m} = \mathbf{w}^k_{i_0,j_m}$
else if $\mathbf{w}^k_{i-1_M,j_m}\cdot\mathbf{n}^2 \geq 0, \mathbf{w}^k_{i_0,j_m}\cdot\mathbf{n}^2 \geq 0$, and $\mathbf{W}^k_{i,j_m}\cdot\mathbf{n}^2 \geq 0$ **then**
  $u^k_{i,j_m} = u^k_{i-1_M,j_m}$
  $\mathbf{w}^k_{i,j_m} = \mathbf{w}^k_{i-1_M,j_m}$
else

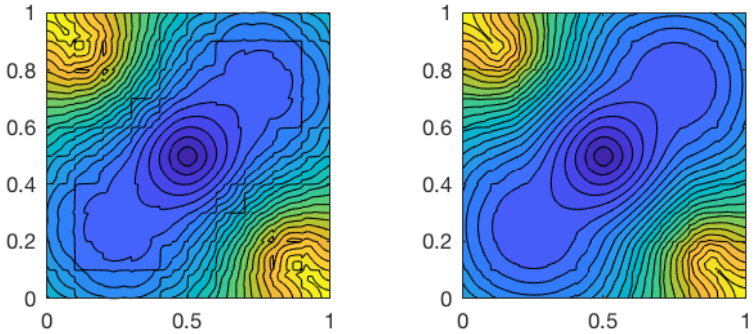
  if $u^k_{i-1_M,j_m} \leq u^k_{i_0,j_m}$ **then**
    $u^k_{i,j_m} = u^k_{i-1_M,j_m}$
    $\mathbf{w}^k_{i,j_m} = \mathbf{w}^k_{i-1_M,j_m}$
  else
    $u^k_{i,j_m} = u^k_{i_0,j_m}$
    $\mathbf{w}^k_{i,j_m} = \mathbf{w}^k_{i_0,j_m}$
  end if
end if

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

Input: $\text{nbrs}^H(U_{i,j_m}), \text{nbrs}^H(U^k_{i,j_m})$, $W^k_{i,j_m}, w^k_{i,j_m}, H$
Output: U_{i,j_m}, W_{i,j_m}
Compute $\tilde{U}$ and $\tilde{W}$ as in Algorithm 3.1
if $\mathbf{w}^k_{i,j_m}\cdot\mathbf{n}^1 \geq 0, \mathbf{W}^k_{i,j_m}\cdot\mathbf{n}^1 \geq 0$, and $\tilde{\mathbf{W}}\cdot\mathbf{n}^1 \geq 0$ **then**
  $U_{i,j_m} = \theta^{k+1}_{i,j_m}\tilde{U} + u^k_{i,j_m} - \theta^{k+1}_{i,j_m}C_H(\text{nbrs}^H(U^k_{i,j_m}))$
  $\mathbf{W}_{i,j_m} = \mathbf{w}^k_{i,j_m}$
else if $\mathbf{w}^k_{i,j_m}\cdot\mathbf{n}^2 \geq 0, \mathbf{W}^k_{i,j_m}\cdot\mathbf{n}^2 \geq 0$, and $\tilde{\mathbf{W}}\cdot\mathbf{n}^2 \geq 0$ **then**
  $U_{i,j_m} = \theta^{k+1}_{i,j_m}\tilde{U} + u^k_{i,j_m} - \theta^{k+1}_{i,j_m}C_H(\text{nbrs}^H(U^k_{i,j_m}))$
  $\mathbf{W}_{i,j_m} = \mathbf{w}^k_{i,j_m}$
else
  $U_{i,j_m} = u^k_{i,j_m}$
  $\mathbf{W}_{i,j_m} = \mathbf{w}^k_{i,j_m}$
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 $\theta^k_{i,j}$. Then for $i = 1,\ldots,N$ and $j = 1,\ldots,M$, we have the following coarse grid solver:

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



(a) $k = 0$ (b) $k = 2$

(c) $k = 4$ (d) $k = 6$

FIG. 3.7. *Contours for fine grid solutions patched together for the slowness function* $r_\epsilon^1(x, y) = 1 + .99\sin(2\pi x)\sin(2\pi y)$ *where in (a)* $k = 0$*, (b)* $k = 2$*,(c)* $k = 4$ *and (d)* $k = 6$.

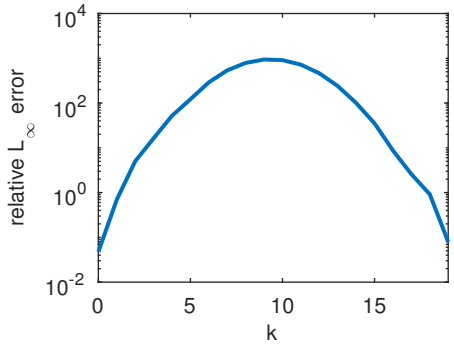


FIG. 4.1. $||U^k - u^f||_{L_\infty}$ *for* $\theta = 1, H = 1/20, h = 1/1000$.

where if $j = 1, \ldots, M-1$ we ignore the second argument of the coarse grid solver. Let $U^0_{i,j} = C_H(U^0_{i-1,j}, U^0_{i,j-M})$. The weighted update for this problem for $j = 1, \ldots M$:

$$U^{k+1}_{i,j} = \theta^{k+1}_{i,j}\left[C_H(U^{k+1}_{i-1,j}, U^{k+1}_{i,j-M}) - C_H(U^k_{i-1,j}, U^k_{i,j-M})\right] + u^k_{i,j} \tag{4.3}$$

with initial conditions

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

and

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

Figure 4.1 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 4.2. 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.

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,

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

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

Now assume

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

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

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

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) imply

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

Therefore,

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

where 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. ∎

Now that we the exactness property for the method is proven, we have the following theorem that proves existence of $\theta^k_{i,j}$ 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 $a \le b \le c$, then $C_H(a,c) \le C_H(b,c)$.

THEOREM 4.2. *For $j = 1,\dots,M$ and $i = 1,\dots,N$, there exists $\theta^k_{i,j}$ such that $U^k_{i,j} < U^{k-1}_{i,j}$ and $U^k_{i,j} > u^f_{i,j}$ for all $i > k$.*

Proof. First we note $U^0_{i,j} > u^f_{i,j}$ and $u^0_{i,j} \ge u^f_{i,j}$ 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 $u^0_{2,j} > U^0_{2,j}$ or $u^0_{2,j} \le U^0_{2,j}$. If $u^0_{2,j} \le U^0_{2,j}$, choose $\theta^1_{2,j} > 0$. Then

$$\begin{aligned} U^1_{2,j} &= \theta^1_{2,j}\left[C_H(U^1_{1,j}, U^1_{2,j-M}) - C_H(U^0_{1,j}, U^0_{2,j-M})\right] + u^0_{2,j} \\ &< u^0_{2,j} \\ &\le U^0_{2,j} \end{aligned}$$

where the first inequality comes from the fact that $U^1_{1,j} = u^f_{1,j} < U^0_{1,j}$. If $u^0_{2,j} > U^0_{2,j}$, then define

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

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

$$\overline{M}^1_{2,j} = \frac{u^f_{2,j} - u^0_{2,j}}{C_H(U^1_{1,j}, U^1_{2,j-M}) - C_H(U^0_{1,j}, U^0_{2,j-M})}.$$

Note $\overline{M}^1_{2,j} > 0$. If we choose $\theta^1_{2,j} < \overline{M}^1_{2,j}$, then $U^1_{2,j} > u^f_{2,f}$.

Now assume $U^1_{i,j} > U^0_{i,j}$ and $U^1_{i,j} > u^f_{i,j}$ for all $i > 1$. If $u^0_{i+1,j} \le U^0_{i+1j}$, choose $\theta^1_{i+1,j} > 0$. Then

$$\begin{aligned} U^1_{i+1,j} &= \theta^1_{i+1,j}\left[C_H(U^1_{i,j}, U^1_{i+1,j-M}) - C_H(U^0_{i,j}, U^0_{i+1,j-M})\right] + u^0_{i+1,j} \\ &< u^0_{i+1,j} \\ &\le U^0_{i+1,j}, \end{aligned}$$

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

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

Now $\overline{m}^1_{i+1,j} > 0$. Thus, if $\theta^1_{i+1,j} > \overline{m}^1_{i+1,j}$, then $U^1_{2,j} < U^0_{2,j}$. Next let

$$\overline{M}^1_{i+1,j} = \frac{u^f_{i+1,j} - u^0_{i+1,j}}{C_H(U^1_{i,j}, U^1_{i+1,j-M}) - C_H(U^0_{i,j}, U^0_{i+1,j-M})}.$$

Note $\overline{M}^1_{i+1,j} > 0$. If we choose $\theta^1_{i+1,j} < \overline{M}^1_{i+1,j}$, then $U^1_{i+1,j} > u^f_{i+1,j}$. So if $0 < \theta^1_{i+1,j} < \overline{M}^1_{i+1,j}$, the theorem holds for $k = 1$.

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

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

If $u^k_{k+2,j} \le U^k_{k+2,j}$, choose $\theta^{k+1}_{k+2,j} > 0$. Then

$$\begin{aligned} U^{k+1}_{k+2,j} &= \theta^{k+1}_{k+2,j}\Big[C_H(U^{k+1}_{k+1,j}, U^{k+1}_{k+2,j-M}) - C_H(U^k_{k+1,j}, U^k_{k+2,j-M})\Big] + u^k_{k+1,j} \\ &< u^k_{k+2,j} \\ &\le U^k_{k+2,j} \end{aligned}$$

If $u^k_{k+2,j} > U^k_{k+2,j}$, let

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

Note $\overline{m}^{k+1}_{k+2,j} > 0$. If $\theta^{k+1}_{k+2,j} > \overline{m}^{k+1}_{k+2,j}$, then $U^{k+1}_{k+2,j} < U^k_{k+2,j}$. Next we need $U^{k+1}_{k+2,j} > u^f_{k+2,j}$. Let

$$\overline{M}^{k+1}_{k+2,j} = \frac{u^f_{k+2,j} - u^k_{k+2,j}}{C_H(U^{k+1}_{k+1,j}, U^{k+1}_{k+2,j-M}) - C_H(U^k_{k+1,j}, U^k_{k+2,j-M})}.$$

Note $\overline{M}^{k+1}_{k+2,j}$. Then if $\theta^{k+1}_{k+2,j} < \overline{M}^{k+1}_{k+2,j}$, $U^{k+1}_{k+2,j} > u^f_{k+2,j}$. So if $u^k_{k+2,j} \le U^k_{k+2,j}$, choose $0 < \theta^{k+1}_{k+2,j} < \overline{M}^{k+1}_{k+2,j}$. If $u^k_{k+2,j} > U^k_{k+2,j}$, choose $\overline{m}^{k+1}_{k+2,j} < \theta^{k+1}_{k+2,j} < \overline{M}^{k+1}_{k+2,j}$. Therefore, the theorem holds. ∎

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

$$\widetilde{m}^k_{i,j} = \begin{cases} 0 & \text{if } \overline{m}^k_{i,j} \le 0 \\ \overline{m}^k_{i,j} & \text{otherwise} \end{cases}.$$

If $\widetilde{m}^k_{i,j} < \theta^k_{i,j} < \overline{M}^k_{i,j}$, we have a monotonically convergent sequence of solutions. The closer we choose $\theta^k_{i,j}$ to $\overline{M}^k_{i,j}$ the more accurate $U^k_{i,j}$ is.

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

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

$$\overline{M}^k_{i,j} = \frac{u^f_{i,j} - u^{k-1}_{i,j}}{C_H(U^k_{i-1,j}, U^k_{i,j-M}) - C_H(U^{k-1}_{i-1,j}, U^{k-1}_{i,j-M})}$$

and we would like $\tilde{m}^k_{i,j} \le \theta^k_{i,j} < \overline{M}^k_{i,j}$. Since we do not know $u^f_{i,j}$ a priori, we estimate $\overline{M}^k_{i,j}$ by the following

$$\overline{\theta}^k_{i,j} = \frac{u^{k-1}_{i,j} - u^{k-2}_{i,j}}{C_H(U^{k-1}_{i-1,j}, U^{k-1}_{i,j-M}) - C_H(U^{k-2}_{i-1,j}, U^{k-2}_{i,j-M})}. \tag{4.7}$$

However, upon implementation this estimation produces very unstable solutions. The values $\overline{\theta}^k_{i,j}$ become extremely large and creates sequences of solutions where $U^k_{i,j} \ll u^f_{i,j}$ or $U^k_{i,j} \gg U^{k-1}_{i,j}$. 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.,

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

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

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

Then

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

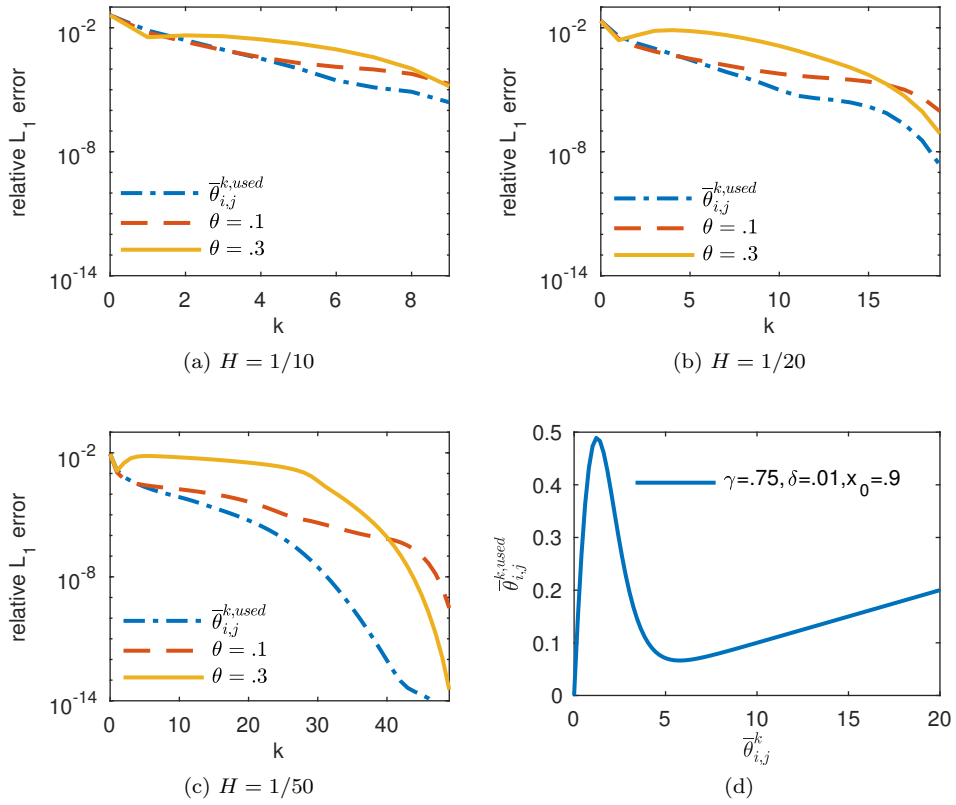
where x_0, γ, and δ are parameters chosen experimentally.

Figure 4.2d shows the plot of $\overline{\theta}^k_{i,j}$ versus $\overline{\theta}^{k,used}_{i,j}$, and Figures 4.2a to 4.2c show the error plots for various values of H. In all three examples $h = 1/1000$. We see the advantage of using $\overline{\theta}^{k,used}_{i,j}$ over a fixed value of θ.

4.3. Complexity and speed up. Let $N = 1/H$ and $M = 1/(Nh)$. Define $A(N,d) := C(2^d(N+1)^d)$ 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 and the causal sweep is $(dM)A(N,d) + 2^dM(N+1)^2$ and the number of flops for all the subdomains combined is $(N^d)A(M,d)$. 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 number of flops will be $k[A(N,d) + A(M,d) + 2^dM(N+1)^2]$. For our method to achieve speed up via parallelization, we need

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

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 we need $k \ll 266$ in order to achieve speed up.



(a) $H = 1/10$ (b) $H = 1/20$

(c) $H = 1/50$ (d)

FIG. 4.2. *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) shows the parameters* γ, δ, *and* x_0 *used to estimate* θ. *For (a)- (c),* $\gamma = .75, \delta = .01, x_0 = .9$.

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, $\overline{\theta}^{k,used}_{i,j}$ is chosen so the solution is stable and converges to the overall fine solution.

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

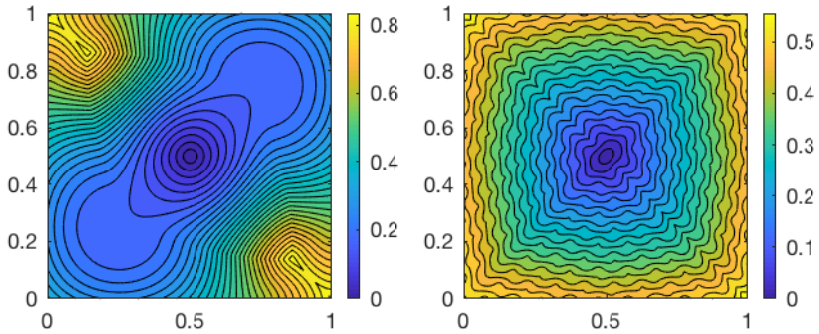
$$r^1_\epsilon(x,y) = 1 + .99\sin(2\pi x)\sin(2\pi y)$$

and

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

[1]Matlab/C++ code used to produce all numerical results can be found at https://github.com/lindsmart/MartinTsaiEikonal

Figure 5.2 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 corner depend on more subdomains than in the r_ϵ^2 case.



(a) r_ϵ^1 (b) r_ϵ^2

FIG. 5.1. *(a) Solution contour for* $r_\epsilon^1(x,y) = 1 + .99\sin(2\pi x)\sin(2\pi y)$. *(b) Solution contour for* $r_\epsilon^2(x,y) = 1 + .5\sin(20\pi x)\sin(20\pi y)$.

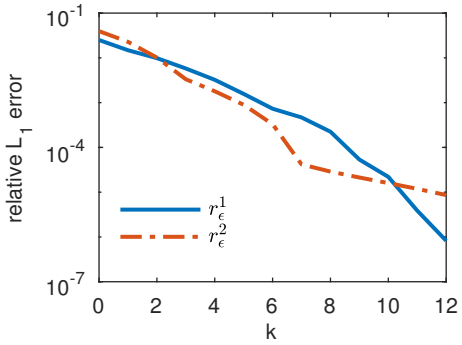


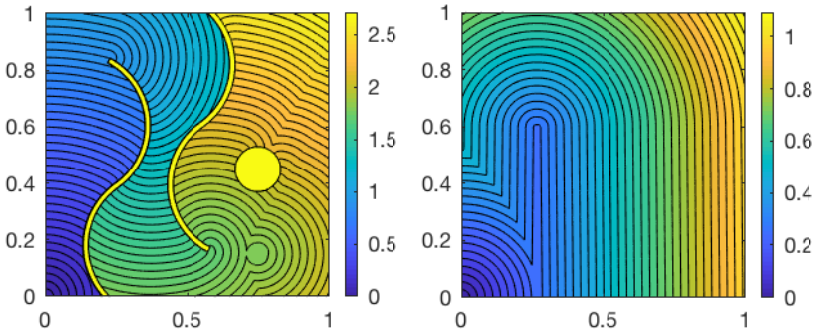
FIG. 5.2. *Relative* L_1 *error plots for* r_ϵ^1 *and* r_ϵ^2 *for* $H = 1/10$ *and* $h = 1/500$.

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 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 characters in the opposite direction. The causal sweeps are critical in order to capture the right flow of characteristics because some coarse grids the right causality will never be computed. The solution contours for r_ϵ^3 and r_ϵ^4 is shown in Figures 5.3a and 5.3b, 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 5.3a contains an entire subdomain and the other circle is a fast obstacle that is contained entirely in a subdomain. The non-monotonicity of error is due to the causal sweeps. The method only provides speed up once the right characteristics have been captured around the barriers. This is seen in Figure 5.4 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 in $[0.26, 0.27] \times [0, 0.6]$. This example demonstrates that the method performs well when there is large collision of characteristics that goes through several subdomains, and there is a fast obstacle affects the characteristics throughout the majority of the domain. i.e., almost every optimal path in Figure 5.3b must go through the obstacle.



(a) r_ϵ^3 (b) r_ϵ^4

FIG. 5.3. *(a) Solution contour for r_ϵ^3. (b) Solution contour for r_ϵ^4.*

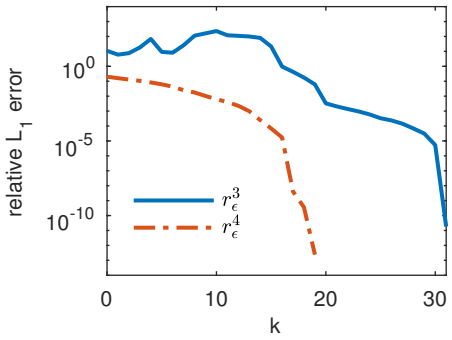
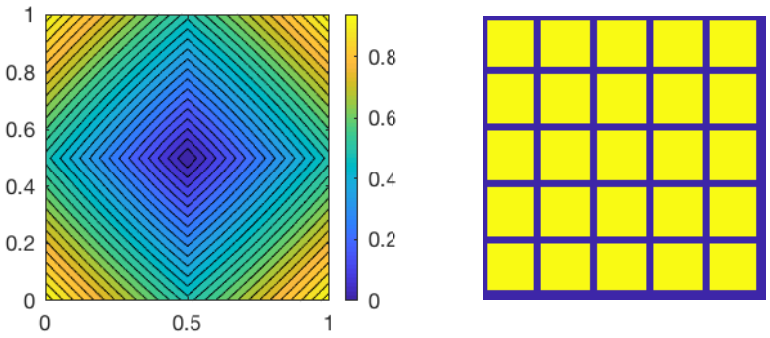


FIG. 5.4. *Relative L_1 error plots for the slowness functions r_ϵ^3 and r_ϵ^4 for $H = 1/10$ and $h = 1/500$.*

5.3. Multiscale slowness functions. Finally, 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 7 fine grid points, i.e., $\epsilon = 7h$, in order for the fine grid to fully capture the micro scale 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



(a) (b)

FIG. 5.5. *(a) Solution contour for the squares slowness function where* $h = 1/1400$ *and* $\epsilon = 1/200$. *(b) Plot of squares slowness function for* $\epsilon = 1/5$.

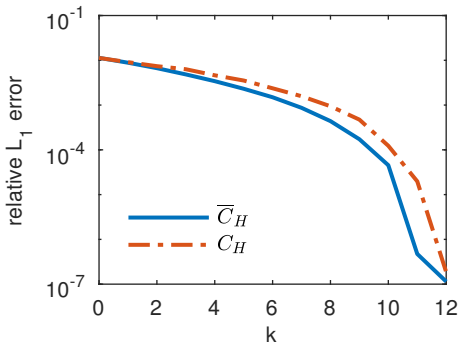


FIG. 5.6. *Relative error* L_1 *error plots for* C_H *and* $\overline{C}_H$.

on an example where the homogenized slowness function, $\overline{r}$, can be computed.

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

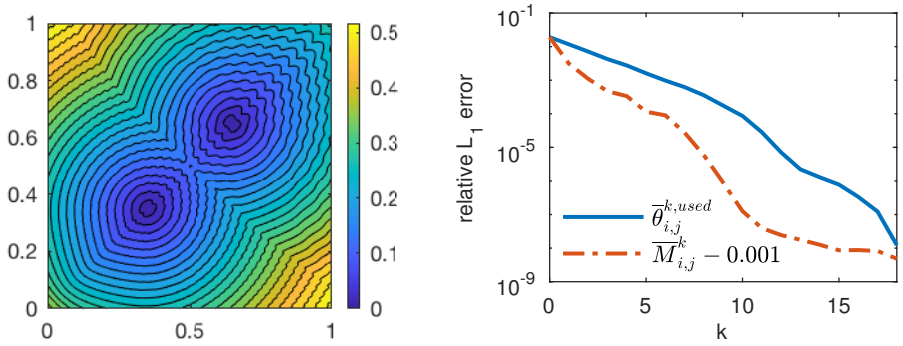
$$r(x,y) = \begin{cases} 1 & \text{if } x = 0 \text{ or } y = 0 \\ 2 & \text{otherwise} \end{cases}$$

and define r_ϵ by extending r by periodicity ϵ. Figure 5.5b shows the slowness function for $\epsilon = 1/5$. The homogenized slowness function is anisotropic and is equal to $\overline{r}(\alpha) = (\alpha_1 + \alpha_2)$ where $\alpha = (\alpha_1, \alpha_2)$ and $|\alpha| = 1$. This is due to the optimal paths moving only vertically or horizontally [16].

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 5.5a which is a diamond and not a circle. Suppose in the method we have the coarse solver solve an equation that better describes the macro scale behavior of the solution. Since in this example we know the homogenized equation, on the coarse grid we can solve



(a) (b)

FIG. 5.7. *Generalization of Figure 5.1b: (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* $\overline{\theta}^{k,used}_{i,j}$ *and* $\overline{M}^k_{i,j} - 0.001$ *are used.*

the homogenized equation

$$\frac{1}{\overline{r}(\frac{\nabla u}{|\nabla u|})}|\nabla \overline{u}| = 1. \tag{5.1}$$

Denote the homogenized equation coarse solver by $\overline{C}_H$. Figure 5.6 shows the relative L_1 error plots for both the method that uses the C_H as described in section 3 and the method that uses $\overline{C}_H$ in place of C_H. As expected, we can see that method that uses $\overline{C}_H$ performs better.

Next, we demonstrate the method on a generalization of r^2_ϵ as defined in subsection 5.1. Notice in Figure 5.1b, we can see the rough shape of the contours of the solution to the homogenized equation. Let

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

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

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

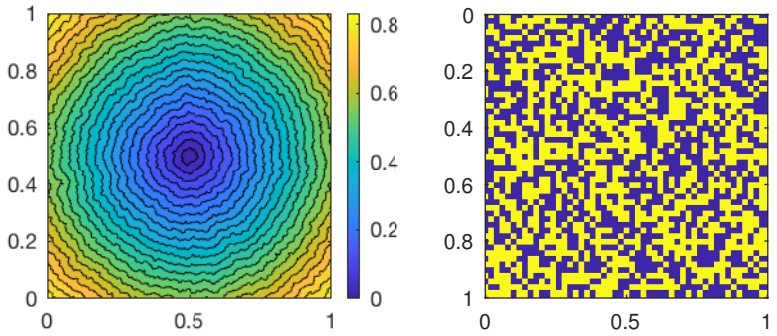
and $\Gamma = \{(0.35, 0.35), (0.65, 0.65)\}$. Then ϵ is very small near $(0,0)$ and increases as we move diagonally up and right through the domain. Figure 5.7a shows the solution contour for this given r_ϵ. Since u^f can be computed a priori, we compare the error plots of our method where we use formula (4.10) and $\overline{M}^k_{i,j} - 0.001$ as the choice of weights in the method. In this example, $H = 1/14$ and $h = 1/1400$. The error plots Figure 5.7b suggest that the proposed formula (4.10) has room for improvement in estimating $\overline{M}^k_{i,j}$. For example, if $X_{i,j} = (0,0)$, we have

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

but

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

Finally, we show the results of our method on a case where the values of the slowness function are random. We follow the set up of the random slowness function in [16]. 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 is shown in Figures 5.8a and 5.8b, repsectively. In [16], the authors showed experimentally the homogenized slowness function, $\overline{r}$, is isotropic and its value is a little less than 1. Figure 5.9 shows the plot of the average error over 20 trials where $H = 1/14$ and $h = 1/1400$.



(a) (b)

FIG. 5.8. *(a) Solution contour plot of random periodic checkboard of scale ϵ. (b) Plot of random slowness function.*

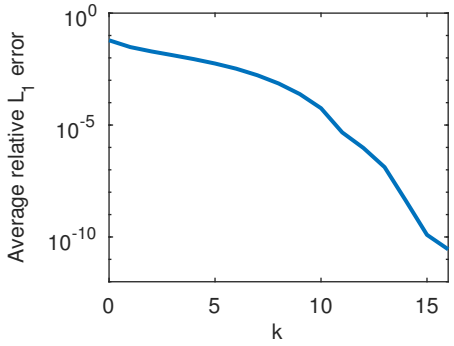


FIG. 5.9. *Relative L_1 error plot for random r_ϵ.*

6. Summary and conclusion. In this paper, we presented a new domain decomposition algorithm for solving boundary value Eikonal equations. Traditional domain decomposition algorithms are difficult to apply due to the nonlinear casual nature of the Eikonal equation. We overcome this difficulty by using coarse and fine grids to propagate information from the subdomains into the coarse level. The parallelization of our method is simple. 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. At each coarse grid node, $\theta_{i,j}^k$ is computed by estimating $\overline{M}_{i,j}^k$ 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. We speculate that improving the estimate of $\overline{M}_{i,j}^k$, it would be possible further increase the stability and speed up of the method.

We demonstrated the method on several classes of slowness functions showing that the 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 5.5 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 5.6.

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.

[2] G. Ariel, H. Nguyen, and R. Tsai, *θ-parareal schemes*, ArXiv e-prints, (2017), https://arxiv.org/abs/1704.06882.

[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, vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 425–432.

[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), vol. 23 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2002, pp. 189–202.

[5] G. Barles and P. E. Souganidis, *Convergence of approximation schemes for fully nonlinear second order equations*, Asymptotic Anal., 4 (1991), pp. 271–283.

[6] A. Chacon and A. Vladimirsky, *Fast two-scale methods for eikonal equations*, SIAM J. Sci. Comput., 34 (2012), pp. A547–A578.

[7] A. Chacon and A. Vladimirsky, *A parallel two-scale method for eikonal equations*, SIAM J. Sci. Comput., 37 (2015), pp. A156–A180.

[8] M. G. Crandall and P.-L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.

[9] M. Detrixhe, F. Gibou, and C. Min, *A parallel fast sweeping method for the Eikonal equation*, J. Comput. Phys., 237 (2013), pp. 46–55.

[10] M. J. Gander and E. Hairer, *Analysis for parareal algorithms applied to Hamiltonian differential equations*, J. Comput. Appl. Math., 259 (2014), pp. 2–13.

[11] T. Haut and B. Wingate, *An asymptotic parallel-in-time method for highly oscillatory PDEs*, SIAM J. Sci. Comput., 36 (2014), pp. A693–A713.

[12] F. Legoll, T. Lelièvre, and G. Samaey, *A micro-macro parareal algorithm: application to singularly perturbed ordinary differential equations*, SIAM J. Sci. Comput., 35 (2013), pp. A1951–A1986.

[13] J.-L. Lions, Y. Maday, and G. Turinici, *A "parareal" in time discretization of PDE's*, Comptes Rendus de l'Acadmie des Sciences - Series I - Mathematics, 332 (2001), pp. 661–668.

[14] 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.

[15] Y. Maday, *The parareal in time algorithm*, in Substructuring Techniques and Domain Decomposition Methods, Saxe-Coburg Publications, Stirlingshire, UK, 2010, pp. 19–44.

[16] A. M. Oberman, R. Takei, and A. Vladimirsky, *Homogenization of metric Hamilton-Jacobi equations*, Multiscale Model. Simul., 8 (2009), pp. 269–295.
[17] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.
[18] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Nat. Acad. Sci. U.S.A., 93 (1996), pp. 1591–1595.
[19] G. A. Staff and E. M. Rø nquist, *Stability of the parareal algorithm*, in Domain decomposition methods in science and engineering, vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 449–456.
[20] 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.
[21] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538.
[22] H. Zhao, *A fast sweeping method for eikonal equations*, Math. Comp., 74 (2005), pp. 603–627.
[23] H. Zhao, *Parallel implementations of the fast sweeping method*, J. Comput. Math., 25 (2007), pp. 421–429.