JET MARCHING METHODS FOR SOLVING THE EIKONAL EQUATION

SAMUEL F. POTTER AND MARIA K. CAMERON

Abstract. We develop a family of compact high-order semi-Lagrangian label-setting methods for solving the eikonal equation. These solvers march the total 1-jet of the eikonal, and use Hermite interpolation to approximate the eikonal and parametrize characteristics locally for each semi-Lagrangian update. We describe solvers on unstructured meshes in any dimension, and conduct numerical experiments on regular grids in two dimensions. Our results show that these solvers often yield both third-order accurate values for the eikonal and its gradient. Along these lines, we prove that one of our solvers is consistent, in a manner that suggests that the solver is at worst quadratically accurate. We additionally show how to march the second partials of the eikonal using cell-based interpolants. Second derivative information computed this way is second-order accurate if the gradients are third-order accurate, suitable for locally solving the transport equation. This provides a means of marching the prefactor coming from the WKB approximation of the Helmholtz equation. These solvers are designed specifically for computing a high-frequency approximation of the Helmholtz equation in a complicated environment with a slowly varying speed of sound, and, to the best of our knowledge, are the first solvers with these properties.

1. Introduction.

Our goal is to develop a family of high-order semi-Lagrangian eikonal solvers which use compact stencils. This is motivated by problems in high-frequency room acoustics, although the eikonal equation arises in a tremendous variety of modeling problems [32].

In multimedia, virtual reality, and video games, precomputing room impulse responses (RIRs) or transfer functions (RTFs) enables convincing spatialized audio, in combination with binaural or surround sound formats. Such an approach, usually referred to as *numerical acoustics*, involves computing pairs of RIRs by placing probes at different locations in a voxelized domain, numerically solving the acoustic wave equation, and capturing salient perceptual parameters throughout the domain using a streaming encoder [25, 26]. These parameters are later decoded using signal processing techniques in real time as the listener moves throughout the virtual environment. Assuming that the encoded parameters can comfortably fit into memory, a drawback of this approach is that the complexity of the simulation depends intrinsically on the highest frequency simulated. In practice, simulations top out at around 1 kHz. The hearing range of humans is roughly 20 Hz to 20 kHz, which requires these methods to either implicitly or explicitly extrapolate the bandlimited transfer functions to the full audible spectrum.

An established alternative to this approach is *geometric acoustics*, where methods based on raytracing are used [27]. Contrary to methods familiar from computer graphics, the focus of geometric acoustics is different. Acoustic waves are mechanical and have macroscopic wavelengths. This means that subsurface scattering, typically modeled using BRDFs in raytracing for computer graphics [20], is less relevant, and is limited to modeling macroscopic scattering from small geometric features, since reflections from flat surfaces are specular in nature. What's more, accurately modeling diffraction effects is crucial [28]: e.g., we can hear a sound source occluded by an obstacle, but we can't see it. **TODO**: *say something about other geometric acoustics methods (image source method, frustum tracing, etc.), and discuss their drawbacks.*

Geometric acoustics and optics both assume a solution to the wave equation based on an asymptotic high-frequency (WKB) approximation to the Helmholtz equation [21]. In this approximation, the eikonal plays the role of a spatially varying phase function, whose level sets describe propagating wavefronts. The prefactor of this ap-

proximation describes the amplitude of these wavefronts. The WKB approximation assumes a ray of "infinite frequency", suitable for optics, since the effects of diffraction are limited. A variety of mechanisms for augmenting this approximation with frequency-depenendent diffraction effects have been proposed, the most successful of which is Keller's *geometric theory of diffraction* [14] (including the later *uniform theory of diffraction* [16]). **TODO**: *mention other diffraction approaches.*

The complete geometric acoustic field of multiply reflected and diffracted rays can be parametrized by repeatedly solving the eikonal equation, using boundary conditions derived from the WKB approximation to patch together successive fields. A related approach is Benamou's *big raytracing* (BRT) [2, 3]. This approach requires one to be able to accurately solve the transport equation describing the amplitude, e.g. using paraxial raytracing [21]. In order to do this, the first and second order partial derivatives of the eikonal must be computed. High-order accurate iterative schemes for solving the eikonal equation exist (**TODO**: *cite*), but their performance deteriorates in the presence of complicated obstacles. Direct solvers for the eikonal equation allow one to locally parametrize the characteristics (rays) of the eikonal equation, which puts one in a position to simultaneously march the amplitude. This enables the design of work-efficient algorithms, critical if a large number of eikonal problems must be solved. **TODO**: *cite my poster and Benamou's work that talks about dynamically monitoring the amplitude, discuss caustics, etc.*

We mention here that for our particular application, we can safely ignore shocks in the eikonal; i.e. the curves (in 2D) or surfaces (in 3D), where there are multiply arriving wavefronts. However, for applications where only computing the first arrival is of interest, it is straightforward to augment our solver with a shock detector, which will be left for future work. **TODO:** *check to see if Benamou did anything related to this.*

Benamou's line of research related to BRT seems to have stalled due to difficulties faced with caustics [4]. This is reasonable considering that the application area was in seismic exploration, where the eikonal equation is used to model first arrival times of P-waves. In this case, the speed of sound is extremely complicated, resulting in a large number of caustics [38]. On the other hand, in room acoustics, the speed of sound varies slowly. The main challenge is a geometric one—the domain is potentially filled with obstacles. This provides another motivation for compact stencils: such stencils can be adapted for use with unstructured meshes, and the sort of complicated boundary conditions that arise when using finite differences are avoided entirely.

The solvers developed in this work are sufficiently high-order, have optimally local/compact stencils, and are label-setting methods (much like Sethian's *fast marching method* [31] or Tsitsiklis's semi-Lagrangian algorithm for solving the eikonal equation [37]). Additionally, being semi-Lagrangian, they locally parametrize characteristics (acoustic rays), making them suitable for use with paraxial raytracing [21], the method of choice for locally computing the amplitude. To the best of our knowledge, these are the first eikonal solvers with this collection of properties.

We refer to our solvers as *jet marching methods* to reflect the fact that the key idea is to march the *jet* of the eikonal (the eikonal and its partial derivatives up to a particular order [35]) in a principled fashion. Sethian and Vladimirsky developed a fast marching method that additionally marched the gradient of the eikonal in a short note, but did not prove convergence results or provide detailed numerical experiments [33]. Related methods exist in the level set method community and are referred to as *gradient augmented level set methods* or *jet schemes* [18, 29].

In the rest of this work we lay out these methods, provide detailed numerical

experiments, and establish some theoretical performance guarantees. Our presentation is done for unstructured grids in n-dimensions, and our numerical experiments are done in 2D. We are currently developing solvers on structured and unstructured meshes in 3D and will report on these later in the context of room acoustics applications.

1.1. Problem setup. Let $\Omega \subseteq \mathbb{R}^n$ be a domain, let $\partial\Omega$ be its boundary, and let $\Gamma \subseteq \Omega$. The *eikonal equation* is a nonlinear first-order hyperbolic partial differential equation given by:

$$\begin{aligned} \|\nabla\tau(\boldsymbol{x})\| &= s(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \\ \tau(\boldsymbol{x}) &= g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma. \end{aligned} \tag{1.1}$$

Here, $\tau : \Omega \to \mathbb{R}$ is the *eikonal*, a spatial phase function that encodes the first arrival time of a wavefront propagating with pointwise *slowness* specified by $s : \Omega \to (0, \infty)$, which can be thought of as an index of refraction. The function $g : \Gamma \to \mathbb{R}$ specifies the boundary conditions, and is subject to certain compatibility conditions [5].

One way of arriving at the eikonal equation is by approximating the solution u of the *Helmholtz equation*:

$$\Big(\Delta + \omega^2 s(\boldsymbol{x})^2\Big) u(\boldsymbol{x}) = 0 \tag{1.2}$$

with the WKB ansatz:

$$u(\boldsymbol{x}) \sim A(\boldsymbol{x}) e^{i\omega\tau(\boldsymbol{x})}, \tag{1.3}$$

where ω is the frequency [21]. As $\omega \to \infty$, this asymptotic approximation is $O(\omega^{-1})$ accurate. This is referred to as the *geometric optics* approximation [4]. The level sets of τ denote the arrival times of bundles of rays, and the amplitude A, which satisfies the transport equation:

$$A(\boldsymbol{x})\Delta\tau(\boldsymbol{x}) + 2\nabla\tau(\boldsymbol{x})^\top \nabla A(\boldsymbol{x}) = 0, \tag{1.4}$$

describes the attenuation of the amplitude of the wavefront due to the propagation and geometric spreading of rays. The characteristics (*rays*) of the eikonal equation satisfy the raytracing ODEs.

The solution of the eikonal equation can also be described using Fermat's principle:

$$\tau(\boldsymbol{x}) = \min_{\substack{\boldsymbol{y}\in\Gamma \\ \boldsymbol{\psi}:[0,1]\to\Omega \\ \boldsymbol{\psi}(0)=\boldsymbol{y},\boldsymbol{\psi}(1)=\boldsymbol{x}}} \left\{ \tau(\boldsymbol{y}) + \int_0^1 s(\boldsymbol{\psi}(\sigma))\|\boldsymbol{\psi}'(\sigma)\| d\sigma \right\}. \tag{1.5}$$

Observe that this equation is recursive, suggesting a connection with dynamic programming and Bellman's principle of optimality. Indeed, the path $\boldsymbol{\psi}$ connecting $\boldsymbol{x}$ and $\boldsymbol{y}$ is a ray, the Hamiltonian of the eikonal equation is:

$$\mathcal{H}(\boldsymbol{x}, \nabla\tau(\boldsymbol{x})) = \frac{\|\nabla\tau(\boldsymbol{x})\|^2 - s(\boldsymbol{x})^2}{2} = 0, \tag{1.6}$$

and its Lagrangian is:

$$\mathcal{L}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = s(\boldsymbol{x})\|\dot{\boldsymbol{x}}\|. \tag{1.7}$$

This provides the connection between the Eulerian perspective given by the eikonal equation, Fermat's principle, and the Lagrangian view provided by raytracing.

2. Related work. The quintessential numerical method for solving the eikonal equation is the *fast marching method* [30]. We discretize Ω into a grid of nodes Ω_h, where $h > 0$ is the characteristic length scale of elements in Ω_h. Let $T : \Omega_h \to \mathbb{R}$ be the numerical eikonal. To compute T, equation (1.1) is discretized using first-order finite differences and the order in which individual values of T are relaxed is determined using a variation of Dijkstra's algorithm for solving the single source shortest paths problem [31, 32]. If $N = |\Omega_h|$, then the fast marching method solves (1.1) in $O(N \log N)$ with $O(h \log \frac{1}{h})$ worst-case accuracy [41]. The logarithmic factor only appears when rarefaction fans are present: e.g., point source boundary data, or if the wavefront diffracts around a singular corner or edge. In these cases, full $O(h)$ accuracy can be recovered by proper initialization near rarefaction fans, or by employing a variety of factoring schemes [24].

We can also use a variety of semi-Lagrangian algorithms to solve the eikonal equation, in which the ansatz (1.5) is discretized and applied locally [37]. For instance, at a point $\hat{\boldsymbol{x}} \in \Omega_h$, we consider a neighborhood of points $\texttt{nb}(\boldsymbol{x}) \subseteq \Omega_h$, assume that τ is fixed over the "surface" of this neighborhood, and approximate (1.5). As an example, if $\texttt{nb}(\hat{\boldsymbol{x}})$ consists of its $2n$ nearest neighbors, if we linearly interpolate τ over the facets of its $\text{conv}(\texttt{nb}(\hat{\boldsymbol{x}}))$ $\boldsymbol{x} \in \Omega_h$, and discretize the integral in (1.5) using a right-hand rule, the resulting solver is equivalent to the fast marching method [34].

The Eulerian approach has generally been favored when developing higher-order solvers for the eikonal equation [40]. The eikonal equation is discretized using higher-order finite difference schemes and solved in the same manner as the fast marching method or using a variety of appropriate iterative schemes. Unfortunately, these approaches presuppose a regular grid and require wide stencils.

Our goal is to develop solvers for the eikonal equation that are *high order*, are *optimally local* (only use information from the nodes in $\texttt{nb}(\hat{\boldsymbol{x}})$ to update $\hat{\boldsymbol{x}}$), and are flexible enough to work on *unstructured meshes*. Using a semi-Lagrangian approach based on a high-order discretization of (1.5) allows us to do this.

Our approach is influenced by two lines of research. First, are *gradient-augmented level set methods* (or *jet schemes*) [18, 29]. Although developed for solving time-dependent advection problems, the setting is similar enough that we are able to borrow a variety of ideas and apply them in the *time-dependent* (or *static*) setting. In particular, the incipient idea for this approach was motivated by the idea of marching cell-based interpolants in an optimally local fashion. Second, we borrow the idea of using a semi-Lagrangian solver to construct a finite element solution to the eikonal equation incrementally [5].

3. The jet marching method. Label-setting algorithms [7], such as the fast marching method, compute an approximation to τ by marching a numerical approximation $T : \Omega_h \to \mathbb{R}^n$ throughout the domain. The boundary data g is not always specified at the nodes of Ω_h. Once T is computed at $\Gamma_h \subseteq \Omega_h$, and approximation of Γ, with sufficiently high accuracy, the solver begins to operate. To drive the solver, a set of states $\{\texttt{far}, \texttt{trial}, \texttt{valid}\}$ is used for bookkeeping. We initially set:

$$\texttt{state}(\boldsymbol{x}) = \begin{cases} \texttt{trial}, & \text{if } \boldsymbol{x} \in \Gamma_h, \\ \texttt{far}, & \text{otherwise.} \end{cases} \tag{3.1}$$

The `trial` nodes are sorted by their T value into a heap (typically a binary array-based heap, although alternatives have been explored [11]). At each step of the iteration, the node $\boldsymbol{x}$ with the minimum T value is removed from the heap, $\texttt{state}(\boldsymbol{x})$ is set to

`valid`, the `far` nodes in $\texttt{nb}(\boldsymbol{x})$ have their state set to `trial`, and each `trial` node in $\texttt{nb}(\boldsymbol{x})$ is subsequently updated.

From this, we can see that the value $T(\boldsymbol{x})$ depends on the values of T at the nodes of a directed graph leading from $\boldsymbol{x}$ back to Γ_h, noting that $T(\boldsymbol{x})$ can—and in general does—depend on multiple nodes in $\texttt{nb}(\boldsymbol{x})$. This means that the error in T accumulates as the solution propagates downwind from Γ_h. We generally assume that the depth of the directed graph of updates connecting each $\boldsymbol{x} \in \Omega_h$ to Γ_h is $O(h^{-1})$. The error due to each update comes from two sources: the running error accumulated in T, and the error incurred by approximating the integral in (1.5). Hence, to obtain $O(h^p)$ accuracy in T globally, it is necessary to compute updates with $O(h^{p+1})$ accuracy. This means that we need to:

- initialize $T(\boldsymbol{x})$ with $O(h^{p+1})$ accuracy for each $\boldsymbol{x} \in \Gamma_h$,
- be able to represent $\partial^{\boldsymbol{\alpha}} T(\boldsymbol{x})$ locally with at least $O(h^{p-k+1})$ accuracy, where $k = \|\boldsymbol{\alpha}\|_1$ is the order of the derivative,
- and approximate the integral with $O(h^{p+1})$ accuracy at each step.

Formally, at least, this allows us to compute $T(\boldsymbol{x})$ with $O(h^p)$ error globally.

Next, we assume that we only know the values of the eikonal and some of its derivatives at the nodes $\boldsymbol{x} \in \Omega_h$. In particular, we do not consider a finite element approximation of τ based on piecewise Lagrange elements[1] [23]. To obtain higher-order accuracy locally, we make use of piecewise Hermite elements. In particular, at each node $\boldsymbol{x}$, we approximate the *jet* of the eikonal; i.e., τ and a number of its derivatives [35]. If we compute the jet with sufficiently high accuracy when we set $\texttt{state}(\boldsymbol{x}) \leftarrow \texttt{valid}$, we will be in a position to approximate τ using Hermite interpolation locally over $\text{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. We consider several variations on this idea.

3.1. The general cost function. Fix a point $\hat{\boldsymbol{x}} \in \Omega_h$, thinking of it as the *update point*. To compute $T(\hat{\boldsymbol{x}})$, we consider sets of `valid` nodes $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d\} \subseteq \texttt{nb}(\hat{\boldsymbol{x}})$, where $1 \leq d \leq n$. The tuple of nodes $(\hat{\boldsymbol{x}}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ is an *update* of dimension d, and the collection of updates a *stencil*. We refer to the nodes $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d\}$ as the vertices of the *base of the update*. In some cases, such as on an unstructured mesh, stencils may vary with $\hat{\boldsymbol{x}}$.

Necessary conditions on the updates and stencils for monotonic convergence have begun to be studied, and come in the form of *causality* conditions [15]. In particular, the cone spanned by $\{\boldsymbol{x}_1 - \hat{\boldsymbol{x}}, \ldots, \boldsymbol{x}_n - \hat{\boldsymbol{x}}\}$ should fit inside the nonnegative orthant after being rotated [33, 34]. It is not clear that causal stencils are also sufficient for monotone convergence. It appears to be necessary for the union of the cones spanned by each update to cover $\mathbb{R}^n$, but this union need not partition $\mathbb{R}^n$. Furthermore, for $O(h)$ solvers that do not make use of gradient information, a variety of stencils lead to monotone convergence. However, in section 9.2, we present a simple counterexample leading to a reduced order of convergence.

In previous works, we have explored a variety of ways of building stencils, but a simple approach is to mesh the surface of $\text{conv}(\texttt{nb}(\hat{\boldsymbol{x}}))$, letting the stencil be comprised of the faces of the mesh [22, 39].

To describe a general update, without loss of generality we assume $d = n$, and assume that the update nodes are in general position. That is, if we choose n nodes from $\{\hat{\boldsymbol{x}}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the remaining node does not lie in their convex hull. We assume that we have access to a sufficiently accurate approximation of τ over $\text{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, call it T. We distinguish between T and T in the following way: T denotes the local

[1] Developing such a solver may be of interest. However, updating the weights of each Lagrange element would be somewhat involved, making the development of an efficient solver challenging.

numerical approximation of τ used by a particular update, while T denotes the global numerical approximation of τ. The two may not be equal to each other. Indeed, T is in general only defined on Ω_h, while T is only defined on $\mathrm{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

Let $\boldsymbol{x}_{\boldsymbol{\lambda}} \in \mathrm{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, and let $L = L_{\boldsymbol{\lambda}} = \|\hat{\boldsymbol{x}} - \boldsymbol{x}_{\boldsymbol{\lambda}}\|$. We approximate $\boldsymbol{\psi}$ with a cubic parametric curve $\boldsymbol{\varphi} : [0, L] \to \Omega$ such that:

$$\boldsymbol{\varphi}(0) = \boldsymbol{x}_{\boldsymbol{\lambda}}, \qquad \boldsymbol{\varphi}(L) = \hat{\boldsymbol{x}}, \qquad \boldsymbol{\varphi}'(0) \sim \boldsymbol{t}_{\boldsymbol{\lambda}}, \qquad \boldsymbol{\varphi}'(L) \sim \hat{\boldsymbol{t}}, \tag{3.2}$$

and where $\boldsymbol{t}_{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{t}}$ are tangent vectors which enter as parameters. Note that $\boldsymbol{\varphi}'(0)$ and $\boldsymbol{\varphi}'(L)$ may not be exactly equal to $\boldsymbol{t}_{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{t}}$.

We approximate the integral in (1.5) over $\boldsymbol{\varphi}$ using Simpson's rule. This gives the cost functional:

$$F(\boldsymbol{\varphi}) = \mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}}) + \frac{L}{6}\Big[s(\boldsymbol{x}_{\boldsymbol{\lambda}})\|\boldsymbol{\varphi}'(0)\| + 4s\big(\boldsymbol{\varphi}_{1/2}\big)\big\|\boldsymbol{\varphi}'_{1/2}\big\| + s(\hat{\boldsymbol{x}})\|\boldsymbol{\varphi}'(L)\|\Big], \tag{3.3}$$

where $\boldsymbol{\varphi}_{1/2} = \boldsymbol{\varphi}(L/2)$ and $\boldsymbol{\varphi}'_{1/2} = \boldsymbol{\varphi}'(L/2)$. We have not yet made this well-defined. To do so, we must specify T, $\boldsymbol{t}_{\boldsymbol{\lambda}}$, and $\hat{\boldsymbol{t}}$. We describe several different ways of doing this in the following sections.

3.2. Computing $\nabla T(\hat{\boldsymbol{x}})$. A minimizing extremal $\boldsymbol{\psi}$ of Fermat's integral is a characteristic of the eikonal equation. A simple but important consequence of this is that its tangent vector is locally parallel to $\nabla\tau$. Hence:

$$s(\boldsymbol{\psi}(\sigma))\frac{\boldsymbol{\psi}'(\sigma)}{\|\boldsymbol{\psi}'(\sigma)\|} = \nabla\tau(\boldsymbol{\psi}(\sigma)). \tag{3.4}$$

After minimizing F, we will have found an optimal value of $\hat{\boldsymbol{t}}$. We can then set:p

$$\nabla T(\hat{\boldsymbol{x}}) \leftarrow s(\hat{\boldsymbol{x}})\hat{\boldsymbol{t}}. \tag{3.5}$$

This puts us in a position to march the gradient of the eikonal locally along with the eikonal itself.

3.3. Parametrizing $\boldsymbol{\varphi}$. We consider two methods of choosing $\boldsymbol{\varphi}$. Define:

$$\boldsymbol{\ell}(\sigma) = \boldsymbol{x}_{\boldsymbol{\lambda}} + \sigma\boldsymbol{\ell}', \qquad \boldsymbol{\ell}' = \frac{\hat{\boldsymbol{x}} - \boldsymbol{x}_{\boldsymbol{\lambda}}}{L_{\boldsymbol{\lambda}}}. \tag{3.6}$$

Here, $\boldsymbol{\ell}$ is the arc length parametrized straight line running from $\boldsymbol{x}_{\boldsymbol{\lambda}}$ to $\hat{\boldsymbol{x}}$, and $\boldsymbol{\ell}'$ is its unit tangent vector.

Using a cubic parametric curve. For one approach, we define:

$$\boldsymbol{\varphi}(\sigma) = \boldsymbol{\ell}(\sigma) + \delta\boldsymbol{\varphi}(\sigma), \tag{3.7}$$

where $\delta\boldsymbol{\varphi} : [0, L] \to \Omega$ is a perturbation away from $\boldsymbol{\ell}$ that satisfies:

$$\delta\boldsymbol{\varphi}(0) = \boldsymbol{x}_{\boldsymbol{\lambda}}, \quad \delta\boldsymbol{\varphi}(L) = \hat{\boldsymbol{x}}, \quad \delta\boldsymbol{\varphi}'(0) = \boldsymbol{t}_{\boldsymbol{\lambda}} - \boldsymbol{\ell}', \quad \delta\boldsymbol{\varphi}'(L) = \hat{\boldsymbol{t}} - \boldsymbol{\ell}'. \tag{3.8}$$

We can explicitly write $\delta\boldsymbol{\varphi}$ as:

$$\delta\boldsymbol{\varphi}(\sigma) = \big(\boldsymbol{t}_{\boldsymbol{\lambda}} - \boldsymbol{\ell}'\big)K_0(\sigma) + \big(\hat{\boldsymbol{t}} - \boldsymbol{\ell}'\big)K_1(\sigma), \tag{3.9}$$

where $K_0, K_1 : [0, L] \to \mathbb{R}$ are Hermite basis functions such that:

$$\begin{aligned} K_0(0) &= 0 = K_0(L), \qquad & K_1(0) &= 0 = K_1(L), \\ K_0'(0) &= 1 = K_1'(L), \qquad & K_1'(0) &= 0 = K_0'(L). \end{aligned} \tag{3.10}$$

Let $\boldsymbol{t}_{\boldsymbol{\lambda}}, \hat{\boldsymbol{t}} \in \mathbb{S}^{n-1}$ so that $\|\boldsymbol{t}_{\boldsymbol{\lambda}}\| = 1 = \|\hat{\boldsymbol{t}}\|$. As $L \to 0$, this results in a curve that is approximately parametrized by arc length: i.e., $\|\boldsymbol{\varphi}'(\sigma)\| \to 1$ for all σ such that $0 \leq \sigma \leq L$ [9]. This simplifies the general cost function given by (3.3) to:

$$F(\boldsymbol{\varphi}) = \mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}}) + \frac{L}{6}\Big[s(\boldsymbol{x}_{\boldsymbol{\lambda}}) + 4s\big(\boldsymbol{\varphi}_{1/2}\big)\big\|\boldsymbol{\varphi}'_{1/2}\big\| + s(\hat{\boldsymbol{x}})\Big]. \tag{3.11}$$

Parametrizing $\boldsymbol{\varphi}$ as the graph of a function. We can also define the perturbation away from $\boldsymbol{\ell}$ as the graph of a function; i.e., we assume that the perturbation is orthogonal to $\boldsymbol{\ell}'$. Letting $\boldsymbol{Q} \in \mathbb{R}^{n \times n-1}$ be an orthogonal matrix such that $\boldsymbol{Q}^\top \boldsymbol{\ell}' = 0$, and letting $\boldsymbol{\zeta} : [0, L] \to \mathbb{R}^{n-1}$ be a curve specifying the components of the perturbation in this basis, we choose $\delta\boldsymbol{\varphi}(\sigma) = \boldsymbol{Q}\boldsymbol{\zeta}(\sigma)$ so that:

$$\boldsymbol{\varphi}(\sigma) = \boldsymbol{\ell}(\sigma) + \boldsymbol{Q}\boldsymbol{\zeta}(\sigma). \tag{3.12}$$

Attempting to set $\boldsymbol{\varphi}'(0) = \boldsymbol{t}_{\boldsymbol{\lambda}}$ and $\boldsymbol{\varphi}'(L) = \hat{\boldsymbol{t}}$ leads to:

$$\boldsymbol{\zeta}'(0) = \boldsymbol{Q}^\top\big(\boldsymbol{t}_{\boldsymbol{\lambda}} - \boldsymbol{\ell}'\big), \qquad \boldsymbol{\zeta}'(L) = \boldsymbol{Q}^\top\big(\hat{\boldsymbol{t}} - \boldsymbol{\ell}'\big). \tag{3.13}$$

Since we set the $\boldsymbol{\ell}'$-direction component to unity, we cannot exactly fix the tangent vectors at $\sigma = 0$ and $\sigma = L$ as we please, e.g.:

$$\|\boldsymbol{\varphi}'(0) - \boldsymbol{t}_{\boldsymbol{\lambda}}\| = \left\|\mathrm{proj}_{\boldsymbol{\ell}'}\left(\boldsymbol{\ell}' - \boldsymbol{t}_{\boldsymbol{\lambda}}\right)\right\|. \tag{3.14}$$

However, if the wavefront passing through $\hat{\boldsymbol{x}}$ is well-approximated by a plane wave[2], as $L \to 0$, the characteristics generally approximate straight lines, so this difference tends to zero; that is, $\|\hat{\boldsymbol{t}}\| \sim 1$ and $\|\boldsymbol{t}_{\boldsymbol{\lambda}}\| \sim 1$. Now, noting that:

$$\|\boldsymbol{\varphi}'(\sigma)\| = \sqrt{\|\boldsymbol{\ell}'\|^2 + \|\boldsymbol{Q}\boldsymbol{\zeta}'(\sigma)\|^2} = \sqrt{1 + \|\boldsymbol{\zeta}'(\sigma)\|^2}. \tag{3.15}$$

We can rewrite the cost functional F as:

$$\begin{aligned} F(\boldsymbol{\varphi}) = \mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}}) + \frac{L}{6}\Bigg[&s(\boldsymbol{x}_{\boldsymbol{\lambda}})\sqrt{1 + \|\boldsymbol{Q}^\top \boldsymbol{t}_{\boldsymbol{\lambda}}\|^2} \\ &+ 4s(\boldsymbol{\varphi}_{1/2})\sqrt{1 + \|\boldsymbol{\zeta}'_{1/2}\|^2} + s(\hat{\boldsymbol{x}})\sqrt{1 + \|\boldsymbol{Q}^\top \hat{\boldsymbol{t}}\|^2}\Bigg]. \end{aligned} \tag{3.16}$$

Trade-offs between the two parametrizations of $\boldsymbol{\varphi}$. When $\boldsymbol{\varphi}$ is a cubic parametric curve, we run into an interesting problem described in more detail by Floater [9]. In particular, the order of accuracy of $\boldsymbol{\varphi}$ in approximating $\boldsymbol{\psi}$ is limited by our parametrization of $\boldsymbol{\varphi}$. If we parametrize $\boldsymbol{\varphi}$ over $\sigma \in [0, 1]$ (that is *uniformly*), then the interpolant is at most $O(h^2)$ accurate. If we parametrize it using a *chordal parametrization*, i.e. $\sigma \in [0, L]$, then it is at most $O(h^4)$ accurate. Indeed, any Hermite spline using a chordal parametrization over each of its segment is at most $O(h^4)$ accurate globally. To design a higher order solver than this requires us to parametrize $\boldsymbol{\varphi}$ using a more accurate approximation of the arc length of $\boldsymbol{\psi}$ (consider, e.g., using a quintic parametric curve).

On the other hand, if we parametrize $\boldsymbol{\varphi}$ as the graph of a function, we can directly apply Hermite interpolation theory [36], and there is no such obstacle. However, we can no longer exactly set $\boldsymbol{\varphi}'(0)$ and $\boldsymbol{\varphi}'(L)$ (see equation (3.14)).

[2]This will generally be the case if $\hat{\boldsymbol{x}}$ is reasonably distant from a point source or rarefaction fan, and if s varies slowly. These are reasonable assumptions for problems in room acoustics.

We can also consider the likely complexity of the resulting cost functions in terms of the number of floating point operations required. Using a parametric curve, the cost functions are simpler. However, for a higher order polynomial curve, the form for L would become more complicated, since it would be the result of using a quadrature rule, such as Simpson's rule, to approximate the arc length of the cubic approximation of $\boldsymbol{\psi}$. With $\boldsymbol{\varphi}$ as the graph of a function, F is more complicated, owing to fewer arithmetic simplifications and the need to multiply with $\boldsymbol{Q}$. For a higher order method, parametrizing $\boldsymbol{\varphi}$ as a graph may be a more natural choice.

4. Different types of minimization problems. In this section, we consider four different ways of using F to pose a minimization problem which would allow us to compute $T(\hat{\boldsymbol{x}})$. We note that each of these formulations is compatible with the version of F where we take $\boldsymbol{\varphi}$ to be a parametric curve *and* where we define it as the graph of a function orthogonal to $\boldsymbol{\ell}(\sigma)$. Altogether, this leads to eight different JMMs.

4.1. Determining $\boldsymbol{t_\lambda}$ by minimizing Fermat's integral. Since the optimal $\boldsymbol{\varphi}$ is a characteristic of the eikonal equation, one approach to setting $\boldsymbol{t_\lambda}$ and $\hat{\boldsymbol{t}}$ is to simply let them enter into the cost function as free parameters to be optimized over. This leads to the optimization problem:

$$\begin{aligned} &\text{minimize} && F(\boldsymbol{x_\lambda}, \boldsymbol{t_\lambda}, \hat{\boldsymbol{t}}) \\ &\text{subject to} && \boldsymbol{x_\lambda} \in \operatorname{conv}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n), \\ & && \boldsymbol{t_\lambda}, \hat{\boldsymbol{t}} \in \mathbb{S}^{n-1}. \end{aligned} \tag{4.1}$$

For a d-dimensional update, the domain of this minimization problem has dimension $(d-1)(n-1)^2$, since $\dim(\operatorname{conv}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_d)) = d-1$.

4.2. Determining $\boldsymbol{t_\lambda}$ from the eikonal equation. When we compute updates, we only require high-order accurate jets over $\operatorname{conv}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$. This set, crucially, is a subset of Ω of codimension one; e.g., a subset of a line in 2D, or plane in 3D. This presents a complication. If we have T and ∇T at the vertices of a facet, then we can use Hermite interpolation to determine T on the facet. However, we can only approximate directional derivatives of T in the linear span of this set. We note that this restricts us to only using this method for "full-dimensional" ($d = n$) updates. To compute $\nabla \mathsf{T}$, we need to recover the directional derivative normal to the facet.

Let $\tilde{\boldsymbol{V}} \in \mathbb{R}^{n \times n-1}$ be an orthogonal matrix such that:

$$\operatorname{range}(\tilde{\boldsymbol{V}}) = \operatorname{range}\left(\begin{bmatrix} \boldsymbol{x}_2 - \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_n - \boldsymbol{x}_1 \end{bmatrix}\right), \tag{4.2}$$

and let $\boldsymbol{v} \in \mathbb{R}^n$ be a unit vector such that $\tilde{\boldsymbol{V}}^\top \boldsymbol{v} = 0$. Let $\nabla_{\tilde{\boldsymbol{V}}}$ be the gradient restricted to the range of $\tilde{\boldsymbol{V}}$, and likewise let $d_{\boldsymbol{v}}$ denote the $\boldsymbol{v}$ directional derivative. Then, using the eikonal equation, we have:

$$s(\boldsymbol{x})^2 = \|\nabla \tau(\boldsymbol{x})\|^2 = |d_{\boldsymbol{v}} \tau(\boldsymbol{x})|^2 + \|\nabla_{\tilde{\boldsymbol{V}}} \tau(\boldsymbol{x})\|^2. \tag{4.3}$$

To recover $\nabla \tau(\boldsymbol{x})$, first note that $\nabla \tau(\boldsymbol{x})$ should point in the same direction as $\boldsymbol{\ell}'$. Choosing $\boldsymbol{v}$ so that $\boldsymbol{v}^\top \boldsymbol{\ell}' > 0$, we get:

$$d_{\boldsymbol{v}} \tau(\boldsymbol{x}) = \sqrt{s(\boldsymbol{x})^2 - \|\nabla_{\tilde{\boldsymbol{V}}} \tau(\boldsymbol{x})\|^2}. \tag{4.4}$$

Letting $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v} & \tilde{\boldsymbol{V}} \end{bmatrix}$, equation (4.4) combined with $\nabla \tau(\boldsymbol{x}) = \boldsymbol{V} \nabla_{\boldsymbol{V}} \tau(\boldsymbol{x})$ gives us a means of recovering $\nabla \tau(\boldsymbol{x})$ from $\nabla_{\tilde{\boldsymbol{V}}} \tau(\boldsymbol{x})$.

Using this technique, we can pose the following optimization problem:

$$\begin{aligned} &\text{minimize} && F(\boldsymbol{x}_{\boldsymbol{\lambda}}, \hat{\boldsymbol{t}}) \\ &\text{subject to} && \boldsymbol{x}_{\boldsymbol{\lambda}} \in \operatorname{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n), \\ & && \hat{\boldsymbol{t}} \in \mathbb{S}^{n-1}, \end{aligned} \tag{4.5}$$

where, for each $\boldsymbol{x}_{\boldsymbol{\lambda}}$, we set:

$$\boldsymbol{t}_{\boldsymbol{\lambda}} \leftarrow \frac{\boldsymbol{V}\nabla_{\boldsymbol{V}}\mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}})}{\|\boldsymbol{V}\nabla_{\boldsymbol{V}}\mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}})\|}. \tag{4.6}$$

The dimension of a d-dimensional update based on this minimization problem is $(d-1)(n-1)$

TODO: *talk about lower-dimensional updates.*

4.3. Determining $\boldsymbol{t}_{\boldsymbol{\lambda}}$ by marching cell-based interpolants. Another approach is to march cells that approximate the jet of the eikonal at each point. For example, if we have constructed a finite element interpolant using `valid` data on a cell whose boundary contains $\operatorname{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ then we can evaluate its gradient to obtain:

$$\boldsymbol{t}_{\boldsymbol{\lambda}} \leftarrow \frac{\nabla\mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}})}{\|\nabla\mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}})\|}. \tag{4.7}$$

We can combine this approach with the cost functional given by (4.5), albeit with a modified $\boldsymbol{t}_{\boldsymbol{\lambda}}$. We will elaborate on how we march cells in a later section. An advantage of this approach is that we can use the interpolation theory for a particular element to, at least formally, ensure a sufficiently high order of accuracy for $\nabla\mathsf{T}(\boldsymbol{x}_{\boldsymbol{\lambda}})$.

4.4. A simplified method using a quadratic curve. In some cases, in particular if the speed of sound is linear, i.e.:

$$c(\boldsymbol{x}) = \frac{1}{s(\boldsymbol{x})} = c_0 + \boldsymbol{c}^\top \boldsymbol{x}, \qquad c_0 \in \mathbb{R}, \qquad \boldsymbol{c} \in \mathbb{R}^n, \tag{4.8}$$

the characteristic $\boldsymbol{\psi}$ is well-approximated by a quadratic. In this case, we set:

$$\hat{\boldsymbol{t}} = \left(\boldsymbol{I} - 2\boldsymbol{\ell}'\boldsymbol{\ell}'^\top\right)\boldsymbol{t}_{\boldsymbol{\lambda}}. \tag{4.9}$$

That is, we take $\hat{\boldsymbol{t}}$ to be the reflection of $\boldsymbol{t}_{\boldsymbol{\lambda}}$ across $\boldsymbol{\ell}'$. In this case, we again have a cost functional of the form (4.5). However, we note that F simplifies, since:

$$\|\boldsymbol{\varphi}'(0)\| = \|\boldsymbol{t}_{\boldsymbol{\lambda}}\| = \|\hat{\boldsymbol{t}}\| = \|\boldsymbol{\varphi}'(L)\|, \tag{4.10}$$

Letting $\boldsymbol{t} = \boldsymbol{t}_{\boldsymbol{\lambda}} = \hat{\boldsymbol{t}}$, **TODO**: *determine nice simplified cost functions for parametric and graph cases.*

4.5. Other approaches. We tried two other approaches which failed to provide satisfactory results:

- A combination of the quadratic simplification in subsection 4.4 with the methods in subsections 4.2 or 4.3. In this case, we use our knowledge of $\nabla T(\boldsymbol{x}_{\boldsymbol{\lambda}})$ along the base of the update to choose $\hat{\boldsymbol{t}}$ and $\boldsymbol{t}_{\boldsymbol{\lambda}}$. This reduces the dimensionality of the cost function to $d-1$. However, except for in special cases (e.g. $s \equiv 1$), this propagates errors in a manner that degrades the overall order of convergence. We note that if $s \equiv 1$, still simpler methods can be used, so this combination of approaches does not seem to be useful.
- **TODO**: *Masha's 3 + 1 method, can't remember how it works... and Masha doesn't want to mention it after all...?*

4.6. Optimization algorithms. We do not dwell on the details of how to numerically solve the minimization problems in the preceding sections. We make some general observations:

- These optimization problems are very easy to solve—what's costly is that we have to solve $O(N)$ of them. As $h \to 0$, they are strictly convex and well-behaved. Empirically, Newton's method converges in $O(1)$ steps (typically fewer than 5 with a well-chosen warm start). We leave a detailed comparison of different approaches to numerically solving these optimization problems for future work.
- The gradients and Hessians of these cost functions are somewhat complicated. Programming them can be tricky and tedious, suggesting that automatic differentiation may be a worthwhile approach [19, 12].
- The constraint $\boldsymbol{x}_{\boldsymbol{\lambda}} \in \text{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ corresponds to a set of linear inequality constraints, which are simple to incorporate. Because of the form of these constraints, checking the KKT conditions at the boundary is cheap and easy [22, 39]. See the next section on skipping updates.
- The constraints $\boldsymbol{t}_{\boldsymbol{\lambda}}, \hat{\boldsymbol{t}} \in \mathbb{S}^{n-1}$ are nonlinear equality constraints. However, these constraints can be eliminated. If $n = 2$, then we can simply let, e.g., $\hat{\boldsymbol{t}} = (\cos(\theta), \sin(\theta))$, letting $\theta \in \mathbb{R}$. For $n > 2$, we can use a Riemannian Newton's method for minimization on $\mathbb{S}^{n-1}$, which is simple to implement and known to converge superlinearly [1].

5. Hierarchical update algorithms. Away from shocks, where multiple wavefronts collide, exactly one characteristic will pass through a point $\hat{\boldsymbol{x}}$. When we minimize F over each update in the stencil, the characteristic will pass through the base of the minimizing update, or possibly through the boundary of several adjacent updates. We can use this fact to sequence the updates that are performed to design a work-efficient solver. In our previous work on *ordered line integral methods* (*OLIMs*), we explored variations of this idea [22, 39]. A approach that works well is the *bottom-up* update algorithm.

To fix the idea in 3D, consider $\texttt{nb}(\boldsymbol{x})$ as shown in Figure **??**, for which $|\texttt{nb}(\boldsymbol{x})| = 26$. There are 26 "line" updates, where $d = 1$. To start with, each `valid` line update is done, and $\boldsymbol{x}_1$ for the minimizing line update is recorded. Next, we fix $\boldsymbol{x}_1$ and perform "triangle" updates ($d = 2$) where $\boldsymbol{x}_2$ is varying. In this case, we can restrict the number of triangle updates that are done by assuming either that $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is an edge of mesh discretizing the surface of the 3D stencil shown in Figure **??**, or that $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ is small enough (measuring the distance of these two points in different norms leads to a different number of triangle updates—we find the ℓ_1 norm to work well). Finally, we fix $\boldsymbol{x}_2$ corresponding to the minimizing triangle update, and do tetrahedron updates containing $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Throughout this process, $\boldsymbol{x}_1, \boldsymbol{x}_2$, and $\boldsymbol{x}_3$ must all be `valid`.

We emphasize that our work-efficient OLIM update algorithms work equally well for the class of algorithms developed here. The main differences between the JMMs studied here and the earlier OLIMs are the cost functionals and the we way approximate T.

6. Initialization methods. A common problem with the convergence of numerical methods for solving the eikonal equation concerns how to treat rarefaction fans. Our numerical tests consist of point source problems, for which this problem arises. A standard approach is to introduce the factored eikonal equation [10]. If a point source is located at $\boldsymbol{x}^\circ \in \Omega_h$ and if we set $\Gamma_h = \{\boldsymbol{x}^\circ\}$, then we let $d(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\circ\|$

and use the ansatz:

$$\tau(\boldsymbol{x}) = z(\boldsymbol{x}) + d(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega. \tag{6.1}$$

We insert this into the eikonal equation, modifying our numerical methods as necessary, and solve for $z(\boldsymbol{x})$ instead. This is not complicated—see our previous work on OLIMs for solving the eikonal equation to see how the cost functions should generally be modified [22, 39].

It has been observed that only factoring the eikonal equation locally gives the best results [24]. In particular, for $\boldsymbol{x}$ such that $d(\boldsymbol{x}) < r^\circ$, where $r^\circ = O(1)$, we solve:

$$\|\nabla z(\boldsymbol{x}) + \nabla d(\boldsymbol{x})\| = s(\boldsymbol{x}), \tag{6.2}$$

and the normal eikonal equation outside of this ball. We note that a method for higher-order local factoring has been developed [17]. We leave combining this with our algorithm for future work.

Yet another approach would be to solve the characteristic equations to high-order for each $\boldsymbol{x}$ in such a ball. This would require solving $O(N)$ boundary value problems, each discretized into $O(N^{1/3})$ intervals, resulting in an $O(N^{4/3})$ cost overall (albeit with a very small constant). One issue with this approach is that it only works well if the ball surrounding $\boldsymbol{x}^\circ$ is contained in the interior of Ω.

For our numerical experiments, we simply initialize T and ∇T to the correct, ground truth values in a ball or box of constant size centered at $\boldsymbol{x}^\circ$.

7. Cell marching. Of particular interest is solving the transport equation governing A while simultaneously solving the eikonal equation. Equation (1.4) can be solved using upwind finite differences [2] or paraxial raytracing [21]. We prefer the latter approach since it can be done locally, using the characteristic path $\boldsymbol{\varphi}$ recovered when computing $T(\hat{\boldsymbol{x}})$. Either approach requires accurate second derivative information (we need ΔT for upwind finite differences, or $\nabla^2 T$ for paraxial raytracing).

For the purposes of explanation and our numerical tests, we consider a rectilinear grid with square cells in $\mathbb{R}^2$. On each cell, our goal is to build a bicubic interpolant, approximating $T(\boldsymbol{x})$. This requires knowing $T, \nabla T$, and T_{xy} at each cell corner. If we know these values with $O(h^{4-p})$ accuracy, where p is the order of the derivative, then the bicubic is $O(h^{4-p})$ accurate over the cell. So far, we have described an algorithm that marches T and ∇T, which are sometimes termed the *total 1-jet*. We now show how T_{xy} can also be marched, allowing us to march the *partial 1-jet*.[3]

Let $\boldsymbol{x}_{ij}$ with $(i,j) \in \{0,1\}^2$ denote the corners of a square cell with sides of length h, and assume that we know $\nabla T(\boldsymbol{x}_{ij})$ with $O(h^3)$ accuracy. We can use the following approach to estimate $T_{xy}(\boldsymbol{x}_{ij})$ at each corner:

- First, at the midpoints of the edges oriented in the x direction (resp., y direction), approximate T_{xy} using the central differences involving T_y (resp., T_x) at the endpoints. This approximation is $O(h^2)$ accurate at the midpoints.
- Use bilinear extrapolation to reevaluate T_{xy} at the corners of the cell, yielding $T_{xy}(\boldsymbol{x}_{ij})$, also with $O(h^2)$ accuracy.

This procedure is illustrated in Figure 7.1.

One issue with this approach is that it results in a piecewise interpolant that is only C^1 globally. That is, if we estimate the value of T_{xy} at a corner from each of the cells which are incident upon it, we will get different values in general. To compute

[3] The total k-jet of a function f is the set $\{\partial^{\boldsymbol{\alpha}} f\}_{\boldsymbol{\alpha}}$, where $\|\boldsymbol{\alpha}\|_1 \leq k$; the partial k-jet is $\{\partial^{\boldsymbol{\alpha}} f\}_{\boldsymbol{\alpha}}$ where $\|\boldsymbol{\alpha}\|_\infty \leq k$.

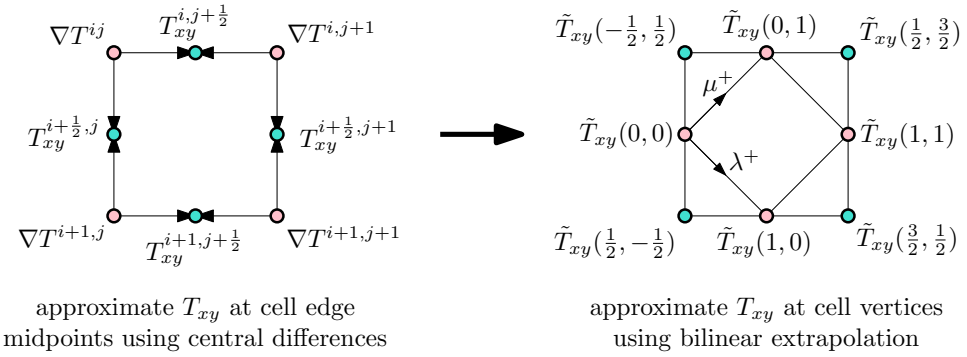

FIG. 7.1. Cell-based interpolation. *To approximate the mixed second partials of a function with $O(h^2)$ accuracy from $O(h^3)$ accurate gradient values available at the corners of a cell, the following method of using central differences to approximate the mixed partials at the midpoints of the edges of the cell, followed by bilinear extrapolation, can be used.*

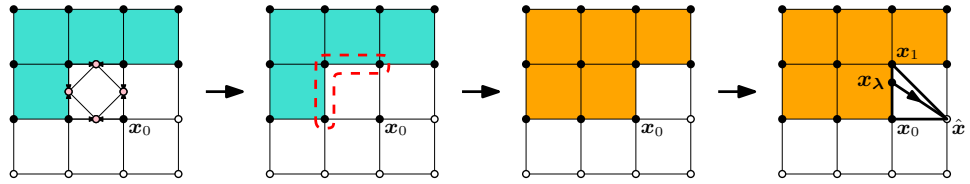

FIG. 7.2. Local cell marching. *After computing values of T_{xy} as shown in Figure 7.1 (*left*), to ensure continuity of the global interpolant, nodal values incident on the newly `valid` cell (containing x_0) can be recomputed by averaging over T_{xy} values taken from incident `valid` cells (*middle*). Finally, a bicubic cell-based interpolant is constructed (*right*).*

a globally C^2 piecewise interpolant, we can average T_{xy} values over incident `valid` cells, where we define a `valid` cell to be a cell whose vertices are all `valid`. How to do this is shown in Figure 7.2.

The idea of approximating the partial 1-jet from the total 1-jet in an optimally local fashion by combining central differences with bilinear extrapolation, and averaging nodal values over adjacent cells to increase the degree of continuity of the interpolant, is borrowed from Seibold et al. [29]. However, applying this idea in this context, and doing the averaging in an upwind fashion is novel

The scheme arrived at in this way is no longer optimally local. However, the sequence of operations described here can be done on an unstructured triangle or tetrahedron mesh. This makes this approach suitable for use with an unstructured mesh that conforms to a complicated boundary.

We should mention here that our approach to estimating T_{xy} is referred to as *twist estimation* in the *computer-aided design* (CAD) community [8], where other approaches have been proposed [6, 13]. We leave the adaptation of these ideas to the present context for future work.

8. Numerical experiments. In this section, we first present a variety of test problems which differ primarily in the choice of slowness function s. The choices of s range from the simple, such as $s \equiv 1$ (a reasonable first choice for speed of sound in room acoustics), to more strongly varying. We then present experimental results for our different JMMs as applied to these different slowness functions, demonstrating the significant effect the choice of s has on solver accuracy. We note that s does not significantly affect the runtime of any of our solvers—our solvers run in $O(|\Omega_h| \log |\Omega_h|)$ time, where the constant factors are essentially insensitive to the choice of s.

8.1. Test problems. Before presenting our experimental results, we collect the details of each problem setup. These test problems primarily differ by the choice of s. We generally consider point source problems on $\Omega = [-1, 1] \times [-1, 1]$. In some cases, we use $\Omega = [0, 1] \times [0, 1]$, due to the behavior of the characteristics of τ (i.e., some choices of s result in shadow zones that are unreachable by rays).

Constant slowness with a point source. For this problem, the slowness and solution are given by:

$$s \equiv 1, \qquad \tau(\boldsymbol{x}) = \|\boldsymbol{x}\|. \tag{8.1}$$

We take the domain to be $\Omega = [-1, 1] \times [-1, 1] \subseteq \mathbb{R}^2$. To control the size of the discretized domain, we let $M > 0$ be an integer and set $h = 1/M$, from which we define Ω_h accordingly. We place a point source at $\boldsymbol{x}^\circ = (0, 0) \in \Omega_h$. The set of initial boundary data locations given by is $\Gamma_h = \{\boldsymbol{x}^\circ\}$, with boundary conditions given by $g(\boldsymbol{x}^\circ) = 0$.

Linear speed with a point source (#1). Our next test problem has a linear velocity profile. This might model the variation in the speed of sound due to a linear temperature gradient (e.g., in a large room). The slowness is given by:

$$s(\boldsymbol{x}) = \left[\frac{1}{s_0} + \boldsymbol{v}^\top \boldsymbol{x}\right]^{-1}, \tag{8.2}$$

where $s_0 > 0$, and $\boldsymbol{v} \in \mathbb{R}^2$ are parameters. The solution is given by:

$$\tau(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{v}\|} \cosh^{-1}\left(1 + \frac{1}{2} s_0 s(\boldsymbol{x}) \|\boldsymbol{v}\|^2 \|\boldsymbol{x}\|^2\right). \tag{8.3}$$

For our first test with a linear speed function, we take $s_0 = 1$ and $\boldsymbol{v} = (0.133, -0.0933)$. The domain, discretized domain, and boundary data are the same as for the constant slowness point source problem described previously.

Linear speed with a point source (#2). For our second linear speed test problem, we set $s_0 = 2$ and $\boldsymbol{v} = (0.5, 0)$. For this problem, we let $\Omega = [0, 1] \times [0, 1]$, discretize into M nodes along each axis, and define Ω_h accordingly (i.e., $|\Omega_h| = M^2$, with $M = h^{-1}$). We take $\boldsymbol{x}^\circ$, Γ_h, and g to be same as in the previous two test problems.

A nonlinear slowness function involving a sine function. For $\boldsymbol{x} = (x_1, x_2)$, we set:

$$\tau(\boldsymbol{x}) = \frac{x_1^2}{2} + 2 \sin\left(\frac{x_1 + x_2}{2}\right)^2. \tag{8.4}$$

This eikonal has a unique minimum, $\tau(0, 0) = 0$, and is strictly convex in $\Omega = [-1, 1]^2$. This lets us determine the slowness from the eikonal equation, giving:

$$s(\boldsymbol{x}) = \sqrt{\sin(x_1 + x_2)^2 + \big(x_1 + \sin(x_1 + x_2)\big)^2}. \tag{8.5}$$

For this test problem, we take Γ_h and Ω_h as in the constant slowness point source problem.

	JMM	$E_{\max}(T)$	$E_{\mathrm{RMS}}(T)$	$E_{\max}(\nabla T)$	$E_{\mathrm{RMS}}(\nabla T)$
Constant	#1	2.92	2.92	2.49	2.82
	#2	2.92	2.92	2.49	2.82
	#3	2.92	2.92	2.49	2.82
Linear #1	#1	2.72	2.83	1.76	2.33
	#2	2.69	2.83	1.43	2.26
	#3	2.81	2.92	1.81	2.55
Linear #2	#1	2.33	2.35	1.47	1.88
	#2	2.24	2.35	1.07	1.74
	#3	2.79	3.02	1.72	2.42
Sine	#1	2.68	2.69	1.70	1.99
	#2	2.53	2.63	1.48	1.89
	#3	2.32	2.36	1.54	1.78
Sloth	#1	2.23	2.32	1.08	1.78
	#2	2.22	2.32	0.86	1.65
	#3	2.07	2.13	1.25	1.75

TABLE 8.1

The order of convergence p for each combination of test problems and solvers, computed for different types of errors and fit as Ch^p.

Sloth. **TODO**: *finishing adding explanation.*

$$s(\boldsymbol{x}) = \sqrt{s_0^2 + \boldsymbol{v}^\top \boldsymbol{x}}. \tag{8.6}$$

For our test with this slowness function, we set $s_0 = 2$, and $\boldsymbol{v} = (0, -3)$. In this case, to avoid caustics, we take $\Omega = [0, \frac{1}{2}]^2$. The discretized domain and boundary data are determined analogously to the earlier cases.

8.2. Experimental results. TODO: *add discussion*

9. Theoretical results. In this section we present two theoretical results of interest:

- We prove consistency for the JMM presented in 4.4. This results suggests that $O(h^2)$-accurate values of T and $O(h)$ accurate ∇T values are to be expected in general, with certain choices of s giving rise to $O(h^3)$ values of T and $O(h^2)$, or even $O(h^3)$, values of ∇T.
- We demonstrate a counterexample that shows how using a 4-point stencil can degrade the overall order of convergence of a jet marching method, and how the use of an 8-point stencil overcomes this problem.

9.1. Consistency of the quadratic curve JMM. As we have seen, our numerical results indicate that we can expect between $O(h^2)$ and $O(h^3)$ accuracy for $T(\boldsymbol{x})$ and between $O(h)$ and $O(h^2)$ accuracy for $\nabla T(\boldsymbol{x})$. Our goal is to determine the conditions under which $O(h^3)$-accurate T and $O(h^2)$-accurate ∇T obtains. Establishing theoretical lower bounds on the order of convergence for T and ∇T requires tedious calculations and nuanced considerations, making the numerical analysis of this solver an interesting and challenging problem in its own right. We defer an extensive

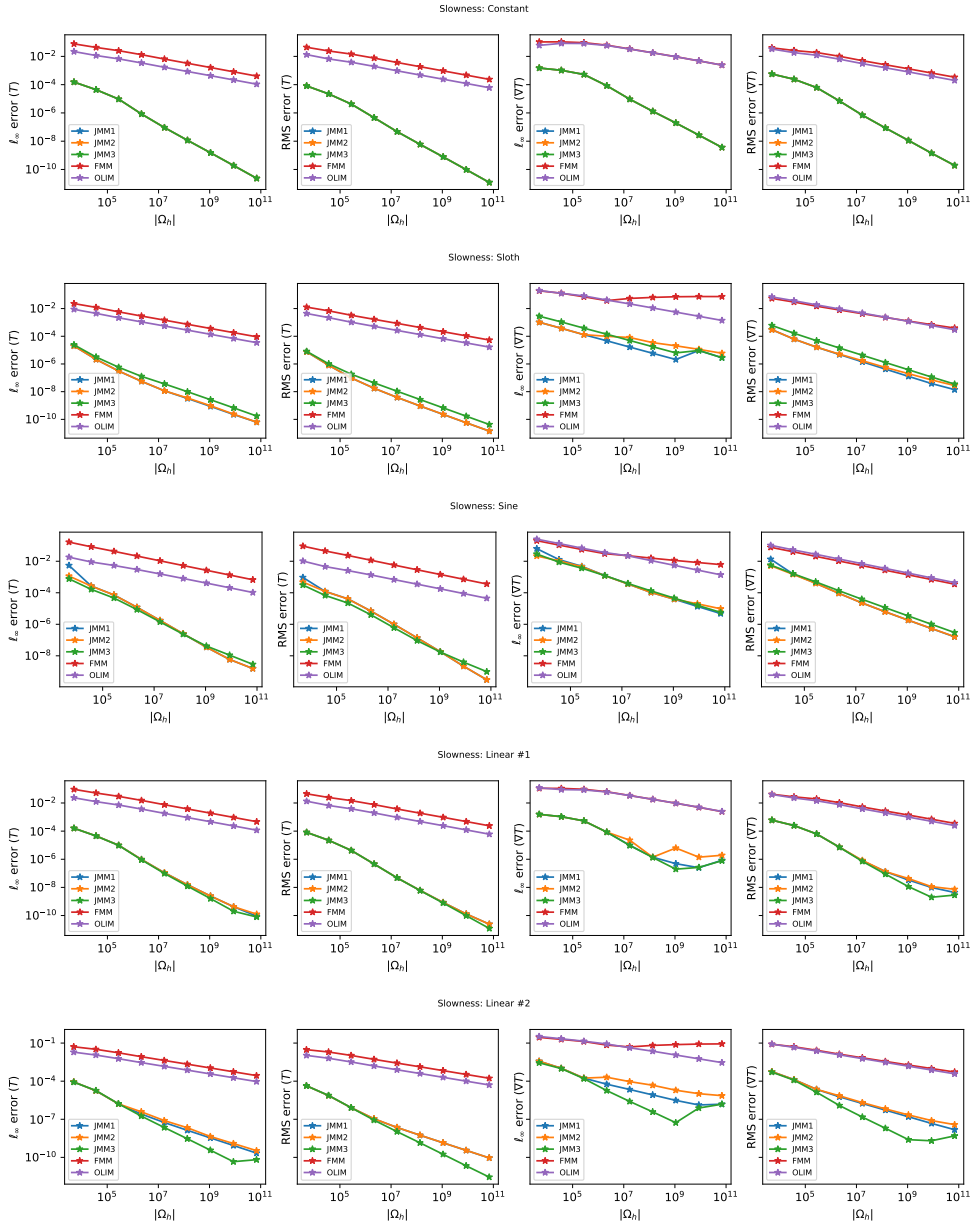

FIG. 8.1. *Plots comparing domain size ($|\Omega_h|$) and errors.*

study of this problem to future work. Here, we simply present consistency results for the simplified JMM described in section 4.4 for the case $\Omega \subseteq \mathbb{R}^2$.

THEOREM 9.1. *Let $\Omega \subseteq \mathbb{R}^2$, assume that $s \in C^2(\Omega)$, and let $(\hat{\boldsymbol{x}}, \boldsymbol{x}_1, \boldsymbol{x}_2)$ be an update triangle where $\texttt{state}(\hat{\boldsymbol{x}}) = \texttt{trial}$, $\texttt{state}(\boldsymbol{x}_1) = \texttt{valid}$, and $\texttt{state}(\boldsymbol{x}_2) = \texttt{valid}$. Assume that no caustic is incident on the update triangle, and that the ray*

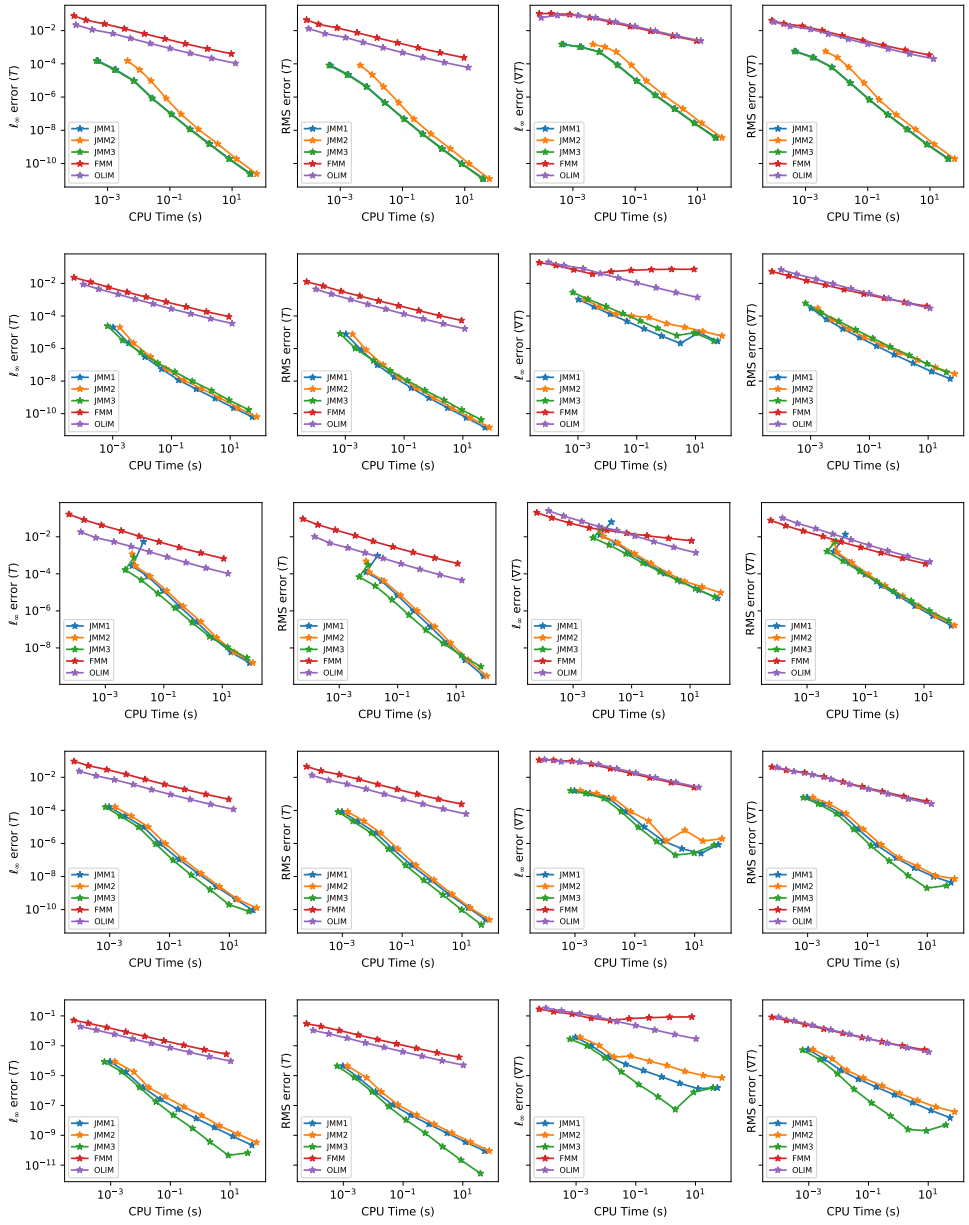

FIG. 8.2. *Plots comparing CPU runtime in seconds and errors.*

arriving at $\hat{\boldsymbol{x}}$ *crosses the segment* $[\boldsymbol{x}_1, \boldsymbol{x}_2]$. *Then:*

$$|\tau(\hat{\boldsymbol{x}}) - T(\hat{\boldsymbol{x}})| = O(h^4), \qquad \|\nabla\tau(\hat{\boldsymbol{x}}) - \nabla T(\hat{\boldsymbol{x}})\| = O(h^2). \tag{9.1}$$

Proof. A detailed proof this theorem requires lengthy but straightforward calculations, and will be presented elsewhere; so, instead, we give an outline of the proof.

Let $\boldsymbol{x}_\lambda = (1-\lambda)\boldsymbol{x}_1 + \lambda + \boldsymbol{x}_2$, we define:

$$f(\lambda) = \tau(\boldsymbol{x}_\lambda) + \int_0^{L_\lambda} s(\boldsymbol{\varphi}(\sigma))\|\boldsymbol{\varphi}'(\sigma)\| d\sigma, \tag{9.2}$$

where $L_\lambda = \|\hat{\boldsymbol{x}} - \boldsymbol{x}_\lambda\|$, and let $F(\boldsymbol{x}_\lambda, \boldsymbol{t})$ be the cost functional given by (**??**). Here, we parametrize the line integral in (9.2) using a chordal parametrization. Note that:

$$\tau(\hat{\boldsymbol{x}}) = \min_{0\le\lambda\le 1} f(\lambda), \qquad T(\hat{\boldsymbol{x}}) = \min_{0\le\lambda\le 1} F(\boldsymbol{x}_\lambda, \boldsymbol{t}(\boldsymbol{x}_\lambda)), \tag{9.3}$$

and that the gradients can recovered via:

$$\nabla\tau(\hat{\boldsymbol{x}}) = s(\hat{\boldsymbol{x}})\frac{\boldsymbol{\varphi}'(L_\lambda)}{\|\boldsymbol{\varphi}'(L_\lambda)\|}, \qquad \nabla T(\hat{\boldsymbol{x}}) = s(\hat{\boldsymbol{x}})\boldsymbol{t}(\boldsymbol{x}_\lambda). \tag{9.4}$$

From these expressions, using multivariable calculus and classical interpolation theory [36], we show that:

$$|f(\lambda) - F(\lambda)| \le Ch^4, \qquad 0 \le h \le 1, \tag{9.5}$$

where $C > 0$ is a constant. Then, using this fact, we can show:

$$\left| \min_{0\le\lambda\le 1} f(\lambda) - \min_{0\le\lambda\le 1} F(\lambda) \right| \le Ch^4. \tag{9.6}$$

This establishes that $|\tau(\hat{\boldsymbol{x}}) - T(\hat{\boldsymbol{x}})| = O(h^4)$. To bound the error in the gradients, we first show that $|f'(\lambda) - F'(\lambda)| = O(h^3)$. Then, we prove that:

$$F''(\lambda) \ge C'h, \qquad 0 \le \lambda \le 1, \tag{9.7}$$

where $C' > 0$ is another positive constant. It follows from the intermediate value theorem that:

$$\left| \arg\min_{0\le\lambda\le 1} f(\lambda) - \arg\min_{0\le\lambda\le 1} F(\lambda) \right| = O(h^2), \tag{9.8}$$

which we use to establish $\|\nabla\tau(\hat{\boldsymbol{x}}) - \nabla T(\hat{\boldsymbol{x}})\| = O(h^2)$. ∎

9.2. A counterexample that demonstrates the need for the 8-point stencil. On a regular grid in $\mathbb{R}^2$, first-order eikonal solvers that use either a 4-point or 8-point stencil are known to work reliably. The use of an 8-point stencil typically improves the error constant by an order of magnitude; in $\mathbb{R}^3$, the differences between the 6-point and 26-point stencils are roughly the same [22]. We are not aware of any thorough studies of the effect of stencils on higher-order solvers.

In this section, we provide a counterexample that demonstrates that using the 4-point stencil can limit the rate of convergence of a JMM to quadratic. Consider the following linear speed function:

$$s : \mathbb{R}^2 \to \mathbb{R}, \qquad \boldsymbol{x} = (x, y) \mapsto 2/(1+x). \tag{9.9}$$

For this choice of s, rays arriving at points upper half-plane ($y > 0$) are circular arcs passing through the origin, whose centers are located on the vertical line $x = -1$.

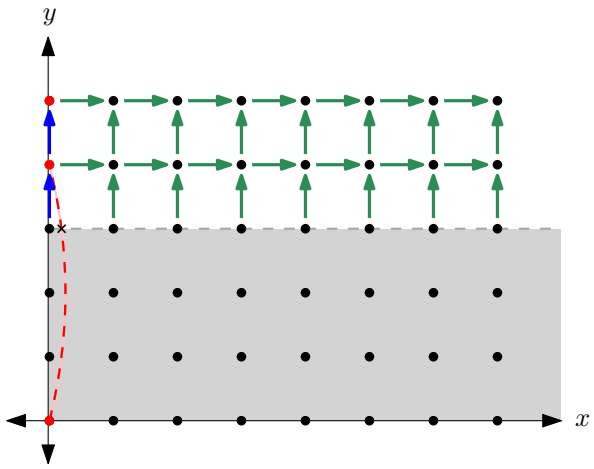

FIG. 9.1. ***TODO***

Indeed, if we differentiate (8.3), fix y, and minimize with respect to x, we find that the x-coordinate satisfies:

$$x = -1 + \sqrt{1 + y^2}. \tag{9.10}$$

Note that the straight line $y = 0$ is also a ray. (**TODO**: *not sure this equation is correct—it isn't a circle.*)

Now, imagine that we solve (1.1) on a grid:

$$\Omega_h = \left\{(h \cdot i, h \cdot j) : (i, j) \in \{0, 1, \ldots\}^2\right\}. \tag{9.11}$$

where $h > 0$. Let $n_{\texttt{slab}} = \lfloor \frac{C}{h} \rfloor$, where C is a small, positive constant (e.g., $C = 0.1$), and initialize each $\boldsymbol{x}$ in the horizontal slab:

$$\{(h \cdot i, h \cdot j) : (i, j) \in \{0, 1, \ldots\} \times [0, n_{\texttt{slab}}]\} \subseteq \Omega_h, \tag{9.12}$$

with the correct values of $\tau(\boldsymbol{x})$ and $\nabla\tau(\boldsymbol{x})$.

TODO: *finish this.*

As a final observation, we can see that the values of ∇T computed by the FMM and `olim8_mp0` in our numerical results are roughly $O(h)$-accurate in most cases. We can see that in some cases, the gradients computed by the FMM diverge. We posit that this difference may be a result of the same phenonemon demonstrated in this section, noting that the FMM uses a 4-point stencil while `olim8_mp0` uses an 8-point stencil. It has often been observed that first-order direct solvers for the eikonal equation compute $O(h)$ accurate gradients. We believe this counterexample sheds some light on the inconsistency in the observed results.

10. Conclusion. We have presented a family of semi-Lagrangian label-setting methods (à la the fast marching method) which are high-order and compact, which we refer to as jet marching methods (JMMs). We examine a variety of approaches to formulating one of these solvers, and in 2D, provide extensive numerical results demonstrating the efficacy of these approaches. We show how a form of "adaptive" cell-marching can be done which is compatible with our stencil compactness requirements, although this scheme no longer displays optimal locality. We also prove preliminary convergence gaurantees for a particular case in 2D.

Our solvers are motivated by problems involving repeatedly solving the eikonal equation in complicated domains where:

- time and memory savings via the use of high-order solvers,
- high-order local knowledge of characteristic directions,
- and compactness of the solver's "stencil" (the neighborhood over which the semi-Lagrangian updates require information)

is paramount. In particular, our goal is to parametrize the multipath eikonal in a complicated polyhedral domain in a work-efficient manner. This solver is a necessary ingredient for carrying out this task.

Apart from this application, we will be continuing to work along the following directions:

- Application of these solvers to regular grids in 3D, which should be straightforward and yield considerable savings over existing approaches, and extension to unstructured simplex meshes in 2D and 3D. Especially in 3D, this problem is more complicated, requiring the computation of "causal stencils" [15, 33].
- Proofs of convergence for each of our approaches in the n-dimensional setting. This requires an extension of the proofs used for this work, but otherwise the idea is the same.
- A more careful characterization of the conditions under which cubic convergence of T and ∇T is obtained.

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, 2009.

[2] J.-D. Benamou, *Big ray tracing: Multivalued travel time field computation using viscosity solutions of the eikonal equation*, Journal of Computational Physics, 128 (1996), pp. 463–474.

[3] J.-D. Benamou, *Multivalued solution and viscosity solutions of the eikonal equation*, (1997).

[4] J.-D. Benamou, *An introduction to eulerian geometrical optics (1992–2002)*, Journal of scientific computing, 19 (2003), pp. 63–93.

[5] F. Bornemann and C. Rasch, *Finite-element discretization of static hamilton-jacobi equations based on a local variational principle*, Computing and Visualization in Science, 9 (2006), pp. 57–69.

[6] P. Brunet, *Increasing the smoothness of bicubic spline surfaces*, Computer Aided Geometric Design, 2 (1985), pp. 157–164.

[7] A. Chacon and A. Vladimirsky, *Fast two-scale methods for eikonal equations*, SIAM Journal on Scientific Computing, 34 (2012), pp. A547–A578.

[8] G. Farin, *Curves and surfaces for computer-aided geometric design: a practical guide*, Elsevier, 2014.

[9] M. S. Floater, *Chordal cubic spline interpolation is fourth-order accurate*, IMA Journal of Numerical Analysis, 26 (2006), pp. 25–33.

[10] S. Fomel, S. Luo, and H. Zhao, *Fast sweeping method for the factored eikonal equation*, Journal of Computational Physics, 228 (2009), pp. 6440–6455.

[11] J. V. Gómez, D. Alvarez, S. Garrido, and L. Moreno, *Fast methods for eikonal equations: an experimental survey*, IEEE Access, (2019).

[12] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, vol. 105, Siam, 2008.

[13] H. Hagen and G. Schulze, *Automatic smoothing with geometric surface patches*, Computer Aided Geometric Design, 4 (1987), pp. 231–235.

[14] J. B. Keller, *Geometrical theory of diffraction*, JOSA, 52 (1962), pp. 116–130.

[15] R. Kimmel and J. A. Sethian, *Computing geodesic paths on manifolds*, Proceedings of the national academy of Sciences, 95 (1998), pp. 8431–8435.

[16] R. G. Kouyoumjian and P. H. Pathak, *A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface*, Proceedings of the IEEE, 62 (1974), pp. 1448–1461.

[17] S. Luo, J. Qian, and R. Burridge, *High-order factorization based high-order hybrid fast sweeping methods for point-source eikonal equations*, SIAM Journal on Numerical Analysis, 52 (2014), pp. 23–44.
[18] J.-C. Nave, R. R. Rosales, and B. Seibold, *A gradient-augmented level set method with an optimally local, coherent advection scheme*, Journal of Computational Physics, 229 (2010), pp. 3802–3827.
[19] R. D. Neidinger, *Introduction to automatic differentiation and matlab object-oriented programming*, SIAM review, 52 (2010), pp. 545–563.
[20] F. E. Nicodemus, *Directional reflectance and emissivity of an opaque surface*, Applied optics, 4 (1965), pp. 767–775.
[21] M. M. Popov, *Ray theory and Gaussian beam method for geophysicists*, EDUFBA, 2002.
[22] S. F. Potter and M. K. Cameron, *Ordered line integral methods for solving the eikonal equation*, Journal of Scientific Computing, 81 (2019), pp. 2010–2050.
[23] P. M. Prenter et al., *Splines and variational methods*, Courier Corporation, 2008.
[24] D. Qi and A. Vladimirsky, *Corner cases, singularities, and dynamic factoring*, Journal of Scientific Computing, 79 (2019), pp. 1456–1476.
[25] N. Raghuvanshi and J. Snyder, *Parametric wave field coding for precomputed sound propagation*, ACM Transactions on Graphics (TOG), 33 (2014), p. 38.
[26] N. Raghuvanshi and J. Snyder, *Parametric directional coding for precomputed sound propagation*, ACM Transactions on Graphics (TOG), 37 (2018), p. 108.
[27] L. Savioja and U. P. Svensson, *Overview of geometrical room acoustic modeling techniques*, The Journal of the Acoustical Society of America, 138 (2015), pp. 708–730.
[28] C. Schissler, R. Mehra, and D. Manocha, *High-order diffraction and diffuse reflections for interactive sound propagation in large environments*, ACM Transactions on Graphics (TOG), 33 (2014), p. 39.
[29] B. Seibold, J.-C. Nave, and R. R. Rosales, *Jet schemes for advection problems*, arXiv preprint arXiv:1101.5374, (2011).
[30] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proceedings of the National Academy of Sciences, 93 (1996), pp. 1591–1595.
[31] J. A. Sethian, *Fast marching methods*, SIAM review, 41 (1999), pp. 199–235.
[32] J. A. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3, Cambridge University Press, 1999.
[33] J. A. Sethian and A. Vladimirsky, *Fast methods for the Eikonal and related Hamilton–Jacobi equations on unstructured meshes*, Proceedings of the National Academy of Sciences, 97 (2000), pp. 5699–5703.
[34] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton–Jacobi equations: theory and algorithms*, SIAM Journal on Numerical Analysis, 41 (2003), pp. 325–363.
[35] G. E. Shilov and R. A. Silverman, *Linear Algebra*, Prentice-Hall, 1971.
[36] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, vol. 12, Springer Science & Business Media, 2013.
[37] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Transactions on Automatic Control, 40 (1995), pp. 1528–1538.
[38] R. Versteeg, *The marmousi experience: Velocity model determination on a synthetic complex data set*, The Leading Edge, 13 (1994), pp. 927–936.
[39] S. Yang, S. F. Potter, and M. K. Cameron, *Computing the quasipotential for nongradient SDEs in 3D*, Journal of Computational Physics, 379 (2019), pp. 325–350.
[40] Y.-T. Zhang, H.-K. Zhao, and J. Qian, *High order fast sweeping methods for static Hamilton-Jacobi equations*, Journal of Scientific Computing, 29 (2006), pp. 25–56.
[41] H. Zhao, *A fast sweeping method for eikonal equations*, Mathematics of computation, 74 (2005), pp. 603–627.