PyKonal: A Python package for solving the Eikonal equation in spherical and Cartesian coordinates using the Fast Marching Method

Malcolm C. A. White*[1], Hongjian Fang†[2], Nori Nakata‡[2], and Yehuda Ben-Zion§[1]

[1] *University of Southern California, Department of Earth Sciences*
[2] *Massachusetts Institute of Technology, Department of Earth, Atmosphere and Planetary Sciences*

December 14, 2020

Corresponding author:

Malcolm White
University of Southern California
Department of Earth Sciences
Zumberge Hall of Science (ZHS)
3651 Trousdale Pkwy
Los Angeles, CA 90089-0740

*malcolm.white@usc.edu
†hfang@mit.edu
‡nnakata@mit.edu
§benzion@usc.edu

Abstract

This article introduces *PyKonal*: a new open-source Python package for computing traveltimes and tracing raypaths in 2D or 3D heterogeneous media using the Fast Marching Method for solving the Eikonal equation in spherical and Cartesian coordinates. Compiled with the Cython compiler framework, *PyKonal* offers a Python application program interface (API) with execution speeds comparable to C or Fortran codes. Designed to be accurate, stable, fast, general, extensible, and easy to use, *PyKonal* offers low- and high-level API functions for full control and convenience, respectively. A scale-independent implementation allows problems to be solved at micro, local, regional, and global scales, and precision can be improved over existing open-source codes by combining different coordinate systems. The resulting code makes state-of-the-art computational capabilities accessible to novice programmers and is efficient enough for modern research problems in seismology.

Introduction

How long do seismic waves generated at one point take to reach another? What path does the energy take to get there? These basic questions are key for seismologists' ability to locate earthquakes, image subsurface structure, and pursue many other fundamental studies. The vast literature documenting different approaches to answering these questions dwarfs the number of corresponding practical tools freely available to seismologists. Here, we present *PyKonal*: a new open-source Python package for computing traveltimes and tracing raypaths using the Fast Marching Method (FMM) for solving the Eikonal equation.

Most numerical approaches for computing traveltimes and raypaths can be classified into one of two main categories: ray-based methods, which solve the *kinematic ray equations*; and grid-based methods, which usually solve the *Eikonal equation* using finite differences or use Dijkstra-like network algorithms (Dijkstra, 1959) to determine the shortest path between two points. Ray-based methods compute traveltimes along individual raypaths and provide no explicit information about traveltimes along alternative raypaths. They are favorable for their speed when the number of raypaths that the user is interested in is small, but they suffer from inaccuracy and instability in regions with significant velocity heterogeneity. Grid-based methods, on the other hand, compute the entire traveltime field—i.e. from the source to every point on a grid—and then use the gradient of the traveltime field to determine individual raypaths. While less efficient for determining individual raypaths, they are preferable for their stability and accuracy in complex media and are more efficient when traveltimes are needed at a large number of points.

The kinematic ray equations can be formulated as an initial-value problem, which can be solved using *shooting methods* (e.g., White, 1989), or as a boundary-value problem, which can be solved using *bending methods* (e.g., Julian & Gubbins, 1977; Thurber & Ellsworth, 1980; Um & Thurber, 1987). See Rawlinson et al. (2008) for a thorough review of ray- and grid-based methods.

A plethora of grid-based methods for solving the Eikonal equation have been proposed since the late 1980s, each with different computational complexity and stability properties, originating primarily in the fields of computational mathematics and physics. Most grid-based Eikonal solvers implement the concept of a *narrow-band* computational front (a limited region of the computational domain where information is strategically propagated from regions where the solution is known to regions where it is unknown). *Sweeping methods* are one class of methods, stemming from the work of Zhao (2004), that omit the concept of a narrow band, which, despite their computational efficiency, find little use in seismological applications because of their instability in heterogeneous media.

Grid-based Eikonal solvers implementing a narrow band generally use either an expanding box or the expanding wavefront as the computational front. Reshef and Kosloff (1986) were probably the first to propose a grid-based method for solving the Eikonal equation in seismological applications, and Vidale (1988) contributed significantly to the expanding-box formulation. The most severe limitation of Vidale's method is that it breaches the causality of the Eikonal equation under certain conditions and is thus unstable. A number of efforts have been directed at resolving this instability and improving the efficiency and generality of that expanding-box formulation (e.g. Vidale, 1990; van Trier & Symes, 1991; Podvin & Lecomte, 1991; Schneider et al., 1992; Hole & Zelt, 1995; Afnimar & Koketsu, 2000). High-order *essentially non-oscillatory* (ENO)

finite-difference schemes (Harten & Osher, 1987) and their weighted extensions (WENO; Liu et al., 1994) have been implemented in the expanding-box framework for seismological applications (e.g. Kim & Cook, 1999; Qian & Symes, 2002b, 2002a; Buske & Kastner, 2004); however the post-processing proposed to overcome the instability inherent in expanding-box methods (e.g. Kim & Cook, 1999) increases algorithmic complexity.

Qin et al. (1992) were apparently the first to replace the expanding box with a computational front that conforms approximately to the shape of the advancing wavefront. This expanding-wavefront formulation always honors the causality of the Eikonal equation, and is thus unconditionally stable, but does so at the cost of increased computational expense because a sorted list of narrow-band values must always be maintained. In an independent effort, Sethian (1996) combined the stability of the expanding-wavefront formulation with the computational efficiency of heap-sort technology to develop the FMM, which simultaneously offers unconditional stability and computational efficiency. Sethian and Popovici (1999) introduced the FMM to the seismological community and Rawlinson and Sambridge (2004a) generalized the method for multiple reflection and transmission phases.

While previous work established a solid practical foundation for estimating traveltimes and raypaths used to advance the understanding of the Earth's interior, improving precision over and providing greater flexibility than existing software can lead to better results. We implement the FMM here because it is relatively easy to code (reducing the likelihood of bugs), unconditionally stable, and computationally efficient. The presented new Python package, *PyKonal*, can improve on various seismic imaging applications. *PyKonal* can solve the Eikonal equation in two or three dimensions using Cartesian or spherical coordinates and combinations thereof. With possible applications to locating earthquakes, performing seismic tomography, computing raypath parameters, and Kirchhoff depth migration, *PyKonal* is designed to be accurate, stable, fast, general, extensible, and easy to use.

In the following sections we outline the theory and implemented methodology (**Method**), present the performance of the implementation with respect to the design criteria above (**Results**), and compare with existing tools and discuss potential extensions and use cases (**Discussion**). Concluding remarks, including discussion of ongoing work, are finally offered (**Conclusions**).

Method

In this section, we review the derivation of the Eikonal equation following Rawlinson et al. (2008), state the problem solved by the FMM, and describe how we implement it.

Deriving the Eikonal equation

The homogeneous scalar wave equation,

$$\nabla^2 \psi\left(\mathbf{r}, t\right) = \frac{1}{v^2\left(\mathbf{r}\right)} \frac{\partial^2 \psi\left(\mathbf{r}, t\right)}{\partial t^2}, \tag{1}$$

is a second-order linear partial differential equation in space and time with general solutions of the form

$$\psi\left(\mathbf{r}, t\right) = A\left(\mathbf{r}\right) e^{\pm i\omega(\tau(\mathbf{r}) - t)}, \tag{2}$$

where ψ, $\mathbf{r}$, t, v, A, ω, and τ represent the wavefunction, position vector, time, wave velocity, wave amplitude, angular frequency, and the spatially dependent traveltime, respectively.

Substituting (2) into (1) gives

$$-\frac{1}{v^2}\omega^2\psi = \left(\nabla^2 A - \omega^2 \left|\nabla\tau\right|^2 \pm i\omega\left(2\nabla A\cdot\nabla\tau + A\nabla^2\tau\right)\right)\psi, \tag{3}$$

which, after factoring out ψ and separating the real and imaginary parts, yields a pair of equations:

$$\nabla^2 A - \omega^2\left|\nabla\tau\right|^2 = -\frac{\omega^2}{v^2}, \tag{4}$$

and

$$2\nabla A\cdot\nabla\tau + A\nabla^2\tau = 0. \tag{5}$$

Dividing (4) by ω^2 and taking the high-frequency limit $\omega\to\infty$ gives the Eikonal equation:

$$\left|\nabla\tau\right|^2 = \frac{1}{v^2} \tag{6}$$

The high-frequency approximation made to obtain (6) is conventionally held to be valid when the wavelength of the propagating wave is much shorter than the scale of velocity structure heterogeneity, although discontinuities are permitted (Rawlinson et al., 2008).

Expression (5) is called the *transport equation* and can be used to compute the amplitude of the propagating wave as a function of space if the traveltime field is known (e.g. Buske & Kastner, 2004; Qian & Symes, 2002a).

Statement of the problem

Given the velocity field, $v(\mathbf{r})$, we aim to determine the traveltime field, $\tau(\mathbf{r})$, by solving (6) under various boundary conditions.

Consider a parametric surface in $\mathbb{R}^3$, $\Gamma(t)$, representing a zero-phase wavefront that propagates perpendicular to itself with velocity v assumed to be a strictly positive function of space, and let $\tau(\mathbf{r})$ be the time it takes to reach a point with position vector $\mathbf{r}$ from an arbitrary boundary value, $\Gamma(t=0)$. The Eikonal equation (6) approximates the relationship between $\tau(\mathbf{r})$ and $v(\mathbf{r})$ at high frequencies, and $\Gamma(t)$ is the surface defining the wavefront at time t—i.e. the set of $\mathbf{r}$ such that $\tau(\mathbf{r}) = t$. The evolution of $\Gamma(t)$ can be tracked by solving (6) for $\tau(\mathbf{r})$, given $v(\mathbf{r})$ and appropriate boundary conditions. The problem is to solve (6) for $\tau(\mathbf{r})$, given $v(\mathbf{r})$ and $\Gamma(t=0)$.

Solution

Sethian (1996) developed an efficient numerical method—the FMM—for approximating a solution to (6), subject to an entropy condition: *the wavefront can only cross each point once.* Imposing this condition implies that the solution obtained comprises only first arrivals; the wavefront propagates along the minimal path in terms of time between any two points.

The next section presents the FMM as implemented herein. See the original paper by Sethian (1996) for a detailed derivation and proof that the method produces a valid solution of the Eikonal equation.

The Fast Marching Method

Evaluating (6) at a discrete number of points gives

$$\left|(\nabla\tau)_{ijk}\right|^2 = \frac{1}{v_{ijk}^2}, \tag{7}$$

where

$$f_{ijk} \equiv f\left(i\Delta\xi_1, j\Delta\xi_2, k\Delta\xi_3\right), \tag{8}$$

with ijk being indices belonging to the set of integers, and $\Delta\xi_1$, $\Delta\xi_2$, and $\Delta\xi_3$ being discretization intervals along the ξ_1, ξ_2, and ξ_3 axes of a general orthogonal coordinate system spanning $\mathbb{R}^3$, respectively.

To satisfy the entropy condition imposed on the approximate solution to (6) obtained by the FMM, the gradient operator must be approximated in such a way that each component of the traveltime gradient is always non-negative. Doing so ensures that information always flows in the same direction as the propagating first-arrival wavefront. Different entropy-satisfying approximations have been proposed (e.g. Osher & Sethian, 1988; Rouy & Tourin, 1992) and, as in Sethian and Popovici (1999), we choose the scheme from Rouy and Tourin (1992):

$$\left|(\nabla\tau)_{ijk}\right| \approx \sum_{a=1}^{3} max\left(\frac{1}{h_{\xi_a}}D_{ijk}^{-\xi_a}, -\frac{1}{h_{\xi_a}}D_{ijk}^{+\xi_a}, 0\right) \tag{9}$$

where h_{ξ_a} represents the scale factor along the ξ_a axis for the used coordinate system, and $D_{ijk}^{-\xi_a}$ and $D_{ijk}^{+\xi_a}$ represent, respectively, backward and forward finite-difference approximations to the derivative of τ at ijk along the ξ_a axis. Substituting (9) into (7) gives

$$\sum_{a=1}^{3} max\left(\frac{1}{h_{\xi_a}}D_{ijk}^{-\xi_a}, -\frac{1}{h_{\xi_a}}D_{ijk}^{+\xi_a}, 0\right)^2 - \frac{1}{v_{ijk}^2} \approx 0, \tag{10}$$

Equation (10) has an *upwind* difference structure—a causality relationship implying that the value of τ_{ijk} is fully determined by neighbouring values, τ_{lmn}, such that $\tau_{lmn} < \tau_{ijk}$. In other words, information propagates in one direction—from lesser values of τ to greater—and unknown values of τ can be derived from neighboring known values by solving (10). Thus, starting with known values $\tau_{ijk} = 0$, the domain of known values can be expanded until it includes all values by iteratively solving (10). This amounts to solving a quadratic equation in τ_{ijk} after expansion. The efficiency of the FMM rests on carefully choosing the order in which to update the unknown values, as presented in the following paragraph and in Figure 1.

```
1  foreach ijk in grid do
2  |  τ_ijk ← ∞;
3  |  Append ijk to Unknown;
4  end
5  foreach ijk ∈ Γ (t = 0) do
6  |  τ_ijk ← 0;
7  |  Transfer ijk from Unknown to
   |   Trial;
8  end
9  while length(Trial) > 0 do
10 |  ijk~ ← index of node with smallest
   |   value of τ in Trial;
11 |  Transfer ijk~ from Trial to Known;
12 |  foreach Neighbour, lmn, of ijk~
   |   do
13 |  |  τ*_lmn ← solution of Eqn (10);
14 |  |  if τ*_lmn < τ_lmn then
15 |  |  |  τ_lmn ← τ*_lmn;
16 |  |  |  if lmn ∈ Unknown then
17 |  |  |  |  Transfer lmn from
   |  |  |  |   Unkown to Trial;
18 |  |  |  end
19 |  |  end
20 |  end
21 end
```

Figure 1: Essential elements of the FMM algorithm for solving the Eikonal equation (6) implemented by *PyKonal*.

Initialize three empty sets: *Known*, *Trial*, and *Unknown*. Assign all nodes to *Unknown*. Set $\tau_{ijk} = 0$ for all nodes on the initial wavefront and transfer them from *Unknown* to *Trial*. Observe that the upwind structure of (10) implies that the node in *Trial* with the smallest value of τ cannot change (if multiple values are equally the smallest, choose one at random—this will not affect the solution); transfer it from *Trial* to *Known* and temporarily label it *Active*. Update the value of each neighbor of *Active* that is not in *Known*

by using values in *Known* to solve (10) and transfer any of them that are in *Unknown* to *Trial*. Iteratively transfer the node in *Trial* with the smallest value of τ to *Known* and update its neighbors until all nodes are in *Known*.

Any sorting algorithm can be used to determine the node in *Trial* with the smallest value of τ. Following Sethian (1996), we use a heap-sort algorithm, which has $\mathcal{O}(N \log N)$ worst-case performance where N is the total number of grid nodes. In practice, the computational efficiency of the FMM is nearly linear in N.

Coordinate systems

The method has been presented in a general coordinate system thus far, but a specific choice must be made in practice and each coordinate system has strengths and weaknesses, which are illustrated in **Results**. Choosing a particular set of coordinates is tantamount to defining the scale factors, h_{ξ_n}, in (10). In Cartesian coordinates, the scale factors are $h_x = h_y = h_z = 1$ and in spherical coordinates they are $h_\rho = 1$, $h_\theta = \rho$, and $h_\phi = \rho \sin\theta$, where the International Organization for Standardization's (ISO) convention is adopted for spherical coordinates (Figure 2).

Spherical coordinates are ideal for tracking spherical wavefronts, whereas Cartesian coordinates are ideal for tracking planar wavefronts (**Results**). Seismologists often treat sources of seismic energy as point sources, making spherical coordinates suitable at small and intermediate distances. Wavefronts from sufficiently distant sources are often assumed to be planar, for which Cartesian coordinates are ideal. When wavefronts must be tracked over great distances and the sphericity of the Earth must be accounted for, spherical coordinates are again naturally suited to the problem. It is often desirable to combine different coordinate systems when solving a particular problem.

Implementing the FMM solution of the Eikonal equation in a general coordinate system makes integrating different systems easy. To combine different coordinate systems, we first solve the Eikonal equation in one, then map the solution into a second and solve the Eikonal equation in the remaining unknown region. This procedure can be iterated *ad infinitum*, but combining two or three coordinate systems is likely sufficient for most problems in seismology.

The unconditional stability of the FMM owes to a proper entropy-condition satisfying approximation for the gradient and has only been proven in the case of first-order-accurate finite-difference operators. Strictly speaking, transitioning between coordinate systems may destabilize the FMM by violating the entropy condition. Transitioning from the first coordinate system to the second assumes that the wavefront does not propagate from the second back into the first anywhere. This is likely of little concern in the vast majority of practical use cases; however, users should be aware of this aspect and use appropriate caution.

One further caveat users should be aware of is the singularity in the gradient operator at the origin of spherical coordinate systems. The Eikonal equation is unsolvable there because the gradient operator is undefined. Thus, PyKonal will raise an error if the user attempts to place a node at the origin of a spherical grid. Grid nodes can be placed sufficiently close to the origin that this has little effect in practice, but any wavefront propagating through the origin will be distorted. Future updates to PyKonal may include implementing an unstructured-mesh update scheme (Sethian, 1999) at the origin of spherical coordinate systems to mitigate the singularity there.

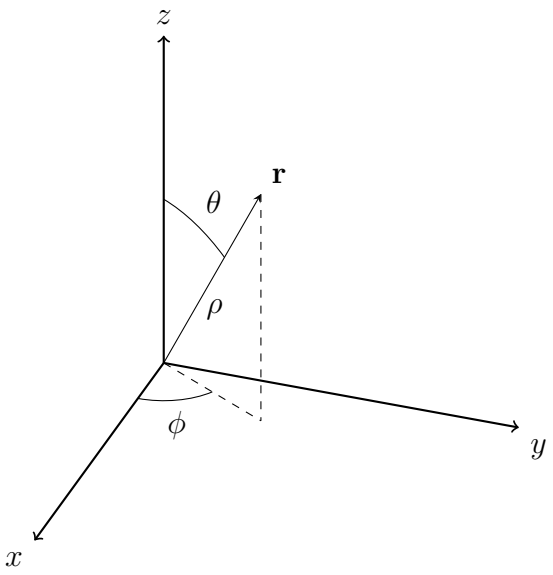

Figure 2: ISO coordinate-system convention adopted for spherical coordinates, where x, y, and z are standard Cartesian coordinates, and ρ, θ, and ϕ are radial, polar, and azimuthal coordinates, respectively, with $\rho \in [0, \infty]$, $\theta \in [0, \pi]$, and $\phi \in [0, 2\pi)$.

Order of the finite-difference operators

The accuracy of the FMM depends partially on the order of the finite-difference operators used where generic operators are specified in (10). The simplest but most inaccurate scheme uses only first-order operators:

$$D_{ijk}^{-\xi_1} = D_{ijk}^{-\xi_1^1} \equiv \frac{\tau_{ijk} - \tau_{i-1jk}}{\Delta\xi_1}, \tag{11}$$

$$D_{ijk}^{+\xi_1} = D_{ijk}^{+\xi_1^1} \equiv \frac{\tau_{i+1jk} - \tau_{ijk}}{\Delta\xi_1}, \tag{12}$$

and similar definitions in ξ_2 and ξ_3, where ξ_m^n indicates an n-th order finite-difference operator along the ξ_m axis (e.g. $D_{ijk}^{+\xi_1^1}$ and $D_{ijk}^{-\xi_3^2}$ represent the finite-difference approximation of the derivative of τ computed using a first-order forward operator along the ξ_1 axis and a second-order backward operator along the ξ_3 axis, respectively). More accurate approximations can be obtained by using higher-order operators, e.g., second-order operators:

$$D_{ijk}^{-\xi_1} = D_{ijk}^{-\xi_1^2} \equiv \frac{\tau_{i+2jk} - 4\tau_{i+1jk} + 3\tau_{ijk}}{2\Delta\xi_1}, \tag{13}$$

$$D_{ijk}^{+\xi_1} = D_{ijk}^{+\xi_1^2} \equiv -\frac{\tau_{i-2jk} - 4\tau_{i-1jk} + 3\tau_{ijk}}{2\Delta\xi_1}, \tag{14}$$

and similar definitions in ξ_2 and ξ_3.

The causality of (10) implies that using higher-order operators is only permissible when the traveltime field at all nodes involved in an update is monotonically decreasing away from the node being updated. For example, a second-order update along the x-axis using a backward finite-difference operator is only permissible when $\tau_{i-2jk} \leq \tau_{i-1jk} \leq \tau_{ijk}$. As in Sethian (1999), we implement second-order operators whenever known values satisfy the causality condition and first-order operators otherwise, which means that the update scheme is of mixed order.

Ray tracing

Energy propagates perpendicular to the wavefronts in isotropic media, so the gradient of the traveltime field is always tangent to the raypath between two points. Thus the raypath can be expressed as a parametric curve, $\mathbf{R}(s)$, satisfying

$$\frac{d\mathbf{R}(s)}{ds} = \nabla\tau, \tag{15}$$

where s represents distance along the curve. Equation (15) is a first-order ordinary differential equation, which can be solved by integrating over ds using the Runge-Kutta method. PyKonal implements a simple first-order Runge-Kutta method (Euler's method).

Because the source location necessarily coincides with the global minimum of the traveltime field, the gradient of the traveltime vanishes there and we cannot integrate (15) from source to receiver. Instead, we integrate (15) in the reverse direction, from receiver to source, which means that we find the path of steepest descent (in terms of traveltime) between the receiver and source.

The fact that the global minimum of the traveltime field coincides with the source location presents a convenient condition for stopping integration of (15). Because we are following the path of steepest traveltime descent, the traveltime at each point along the raypath should be lower than at every point that precedes it. Thus, the ray will satisfy $\tau(\mathbf{R}(s_1)) < \tau(\mathbf{R}(s_2))$ for all $s_1 > s_2$. If the ray overshoots the source location, it will start traveling "uphill" and will violate this condition. Thus, we stop integrating as soon as the ray encounters a point associated with a traveltime value exceeding that of the immediately preceding point.

Results

We now demonstrate that five of our six aims in building this tool are achieved: The code is accurate, stable, fast, general, and extensible. Users may judge whether our tool satisfies our sixth aim: easy to use.

Accuracy

Consider the trivial case of waves emanating from a point source with $v = 1\,\mathrm{km/s}$ everywhere (Figure 3). Spherical wavefronts expand outward from the source, reaching a given point after an amount of time that is in direct proportion to the distance from the source. Comparing this analytical solution to a numerical solution computed using Cartesian coordinates reveals numerical anisotropy. As discussed by Alkhalifah and Fomel (2001), errors are greatest in the near-source region—where wavefront curvature is large relative to the node spacing—and in regions where the wavefront is oblique to the coordinate axes. In spherical coordinates, however, the numerical solution is exact up to computational precision because the radial axis is always orthogonal to the wavefront and the partial derivative of the traveltime field with respect to azimuth (and polar angle in 3D) is always zero. This simple test suggests that, even for more complicated velocity fields, spherical coordinates centered on a point source are more accurate than Cartesian coordinates in the near-field region where wavefront curvature is high.

If the point source in the above example is replaced by a line (in the 2D case) or a plane (in the 3D case) source, the opposite holds and Cartesian coordinates are more accurate (Figure 4). In this case, the y-axis is always orthogonal to the wavefronts leading to an accurate solution, whereas spherical coordinates suffer from numerical anisotropy.

Much work has been devoted to reduc-

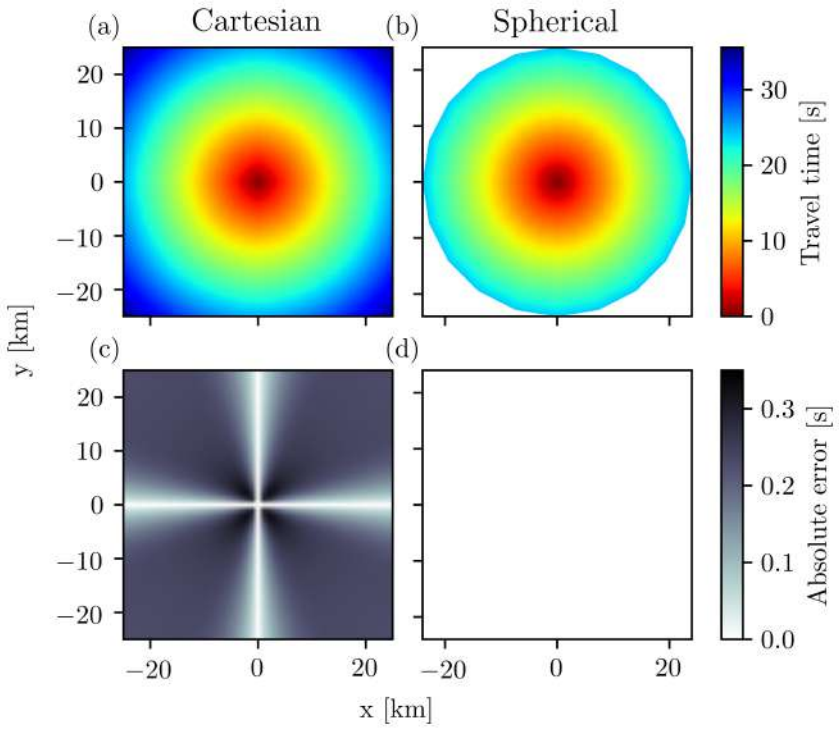

Figure 3: (a,b) Traveltime fields and (c,d) absolute errors for a point source at $(x, y) = (0, 0)$ computed in (a,c) Cartesian coordinates and (b,d) spherical coordinates. (b,d) Portions of the plots are blank because the radial axis of the spherical grid used extends to a maximum of 25 km.

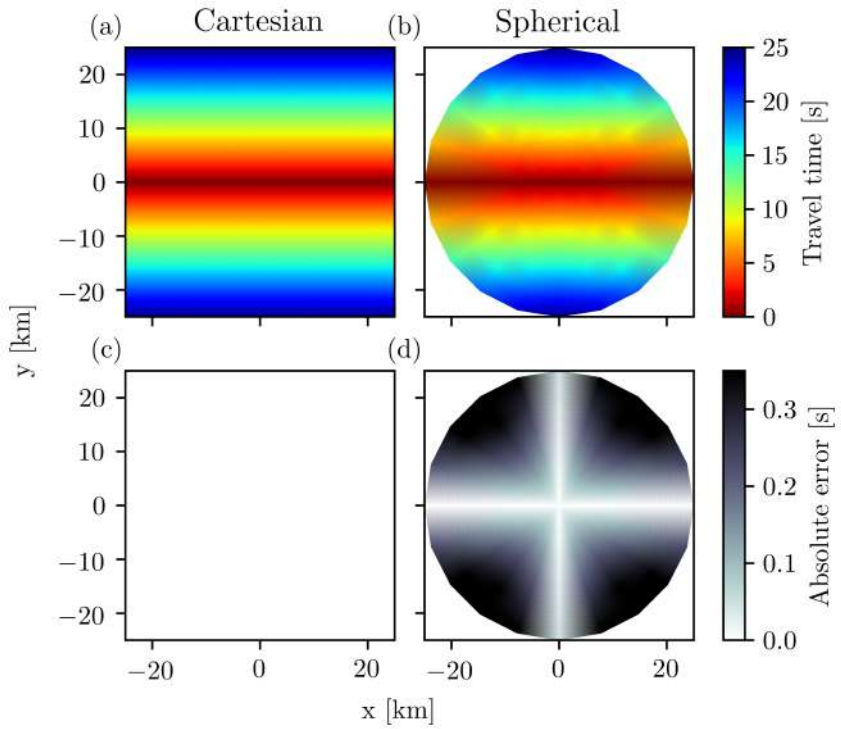

Figure 4: (a,b) Traveltime fields and (c,d) absolute errors for a line source at $y = 0$ computed in (a,c) Cartesian coordinates and (b,d) spherical coordinates. (b,d) Portions of the plots are blank because the radial axis of the spherical grid used extends to a maximum of 25 km.

ing numerical anisotropy introduced by finite-difference schemes, particularly in the field of hyperbolic partial differential equations (see Sescu (2015) for a thorough review). Adapting such approaches to our context seems plausible, however the preceding two examples elucidate an important feature of the FMM for solving the Eikonal equation: judicious design of the computational grid can yield highly accurate solutions while effectively minimizing numerical anisotropy. Because numerical anisotropy can be effectively reduced in this manner, we avoid endeavoring here to implement complicated finite-difference schemes to further reduce it.

Stability

To be useful for solving modern problems in seismology, any ray-tracing tool must be numerically stable in highly complex velocity structures. The FMM is unconditionally stable (Sethian & Popovici, 1999)—meaning that solutions converge to a stable solution that is consistent with the exact solution in the limit that the node intervals go to zero—making it well-suited to any problem in seismology where the Eikonal equation is valid. To demonstrate this convergent behavior, we consider a point source at the surface of the *Marmousi2* velocity model (Versteeg, 1994)—a standard model in exploration seismology comprising complex structures with lateral and vertical gradients, discontinuities, inversions, folded layers, pinch outs, and lenses (Figure 5). Solutions converge towards a stable solution as the node intervals are successively halved. Unconditional stability results from proper choice of an entropy-satisfying gradient approximation (Expression 9), and although it has only been proven in the case of first-order accurate finite-difference operators, these results suggest that the algorithm remains stable with higher-order operators for complex practical use cases.

The unconditional stability of the FMM implies that raypaths obtained by integrating (15) can be made arbitrarily accurate by making the computational grid sufficiently dense (Figure 6). Errors accumulate along the integration path, so they are least at the beginning of the integration path (near the receiver) and greatest at the end (near the source). This error accumulation is exacerbated by relatively large traveltime errors in the source region. Additionally, the criteria used for terminating integration permits inaccurate raypaths to overshoot the source location; however, the amount of overshoot decreases as grid density increases.

Speed

Having established the accuracy of our FMM implementation for the simplest cases (**Accuracy**) and its stability in the complex case of the Marmousi2 model (**Stability**), we turn to its execution speed. We consider a suite of 3D problems of various sizes with a random velocity model and a source at a corner of the grid. The execution time scales nearly linearly with grid size (Table 1) making large problems ($> 10^6$ nodes) tractable and small problems ($< 10^6$ nodes) trivial.

Generality

In **Stability**, we demonstrated that our implementation of the FMM in Cartesian coordinates is applicable at local scales where the curvature of the Earth is negligible. Here we demonstrate that our implementation in spherical coordinates works well at global scales and the property of stable convergence holds. Note that it is the exact same code base that solves the Eikonal equation in both Cartesian and spherical coordinates, eliminating the need for multiple implementations—only the scaling factors in

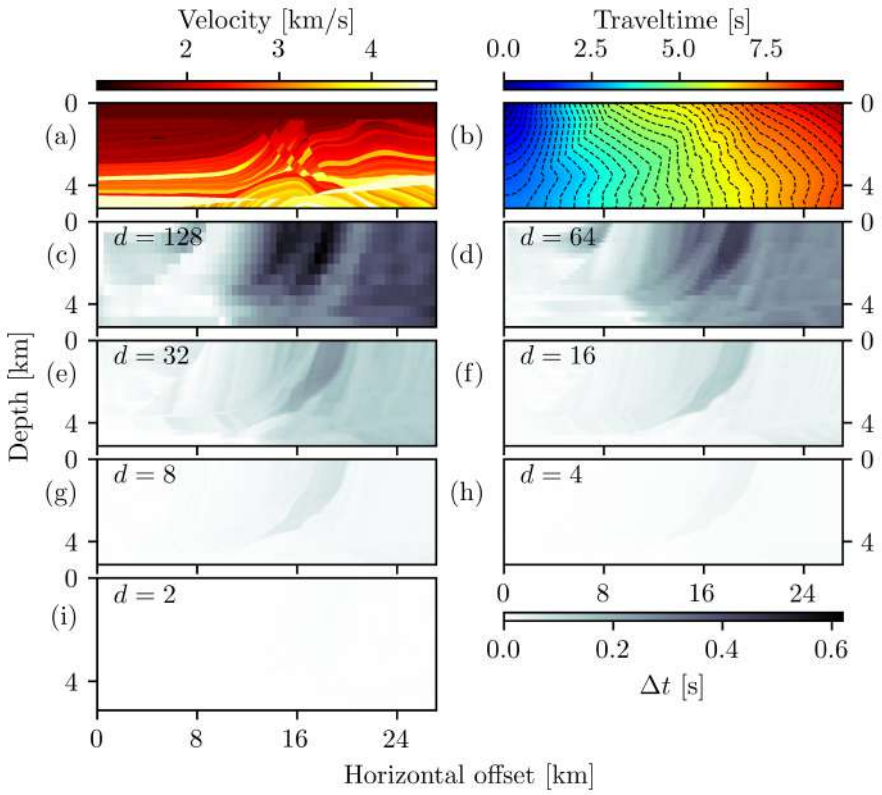

Figure 5: (a) The Marmousi2 velocity model. (b) Traveltime field computed on a Cartesian grid with 6801 and 1401 nodes along the x and y axis, respectively, for a source located at the top left corner $(x, y) = (0, 0)$. Black dashed lines indicate wavefronts at 0.25 s intervals. (c-i) Difference, Δt, at coincident nodes between traveltime field in (b) and traveltime field computed on a grid decimated by a factor, d, specified in the top left corner of each plot.

Table 1: Execution time for problems of various grid sizes. Simulations were run on a Dell Optiplex 9020 workstation with a quad-core 3.4 GHz Intel Core i7-4770 CPU.

Grid size	Number of nodes	Execution time [s]
$2 \times 2 \times 2$	2^3	1.4×10^{-4}
$4 \times 4 \times 4$	2^6	1.1×10^{-4}
$8 \times 8 \times 8$	2^9	6.5×10^{-4}
$16 \times 16 \times 16$	2^{12}	2.8×10^{-3}
$32 \times 32 \times 32$	2^{15}	2.8×10^{-2}
$64 \times 64 \times 64$	2^{18}	2.5×10^{-1}
$128 \times 128 \times 128$	2^{21}	2.6
$256 \times 256 \times 256$	2^{24}	3.7×10^{1}
$512 \times 512 \times 512$	2^{27}	4.4×10^{2}

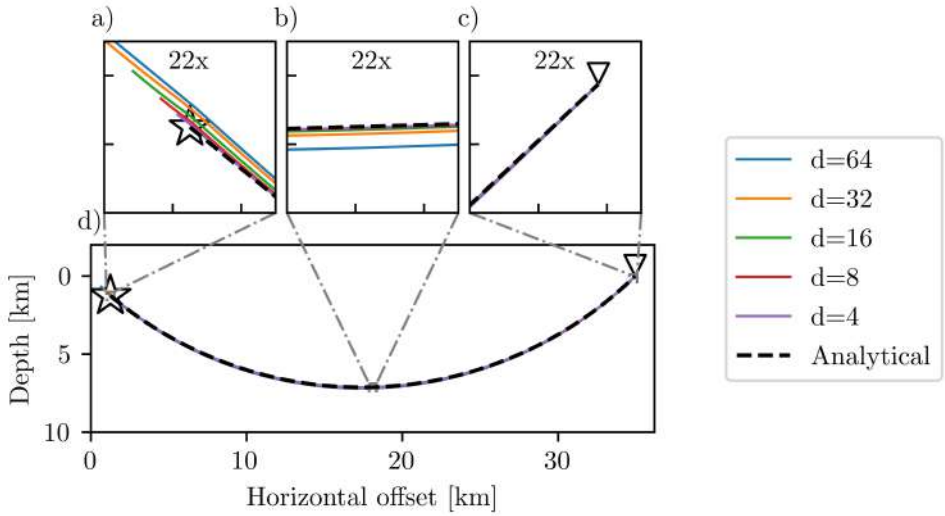

Figure 6: Rays traced through a medium with linear velocity gradient $v(z) = (4.5 + 0.25z)$ [km/s] using different grid densities. The densest grid contains 4096 and 1024 nodes in the x and y directions, respectively, and the grid density is iteratively decimated by a factor of 2 where d represents the decimation factor relative to the densest grid (e.g. the grid for $d = 8$ has $\frac{4096}{8} = 512$ and $\frac{1024}{8} = 128$ nodes in the x and y directions, respectively). Decimation factors corresponding to each ray are given in the legend at right. The analytical solution for the raypath is shown as a dashed black line. The black stars and inverted black triangles represent the source and receiver locations, respectively. (a-c) Zoomed in regions (22x magnification; axes ticks are at 0.2 km intervals) near the (a) source, (b) midpoint, and (c) receiver. (d) The entire raypath.

(10) change between operating modes, and this is done automatically at run time.

We repeat the exercise from Figure 5, but this time we use a 2D slice from the Li et al. (2008) global mantle P-wave model (*MITP2008*) in spherical coordinates. As in the case of Cartesian coordinates, the traveltime field converges to a stable solution as the node intervals decrease (Figure 7).

Extensibility

Tracking secondary phases

The FMM is limited to tracking first arrivals, but secondary reflected arrivals contain much useful information for seismologists. These secondary arrivals can be tracked using the multi-stage approach developed by Rawlinson and Sambridge (2004b) and extended to 3D spherical coordinates by de Kool et al. (2006). We now show that our code can readily be extended to implement such a multi-stage approach for tracking secondary arrivals.

To show this, we track waves reflected from the lower boundary of a rectangular section (Figure 8). First, we compute the downgoing traveltime field from the source to all points. Then, we re-initialize the traveltime field and set the traveltime at each node along the bottom edge to the traveltime of the incident downgoing wave. The upgoing reflected waves are then tracked to all points. This relatively simple example serves as a proof of concept: PyKonal can be extended to solve more complex problems than tracking first arrivals. Tracking multiple reflections and refractions from complex undulating surfaces, however, requires additional functionality that PyKonal does not currently offer. *FM3D* (Rawlinson & Sambridge, 2004b; de Kool et al., 2006) is a more flexible and mature software package with respect to this class of problems.

Locating earthquakes

Seismic analysts locate earthquakes on a daily basis, and typically use 1D velocity models to represent the 3D Earth structure because simple algorithms that can be incorporated into relevant workflows are not available. Research seismologists are left to determine more accurate locations that account for 3D structure, which could be obtained in the first place with the necessary tools. As a final demonstration, we show that it is straightforward to extend the *PyKonal* algorithm to locate earthquakes.

To show this, we relocate a set of 148 well-constrained events in Southern California, taken from the catalog of White et al. (2019), using a simple grid search followed by differential evolution optimization (Storn & Price, 1997) in a small neighborhood around the optimal grid point. Differential evolution is a genetic algorithm that performs a stochastic global search to optimize complex multivariate functions. Each of the chosen events is in the focus region of the study by White et al. (2019), is associated with at least 128 arrivals with root-mean-square residual (RMS) $< 1\,\mathrm{s}$, and was located using the *NonLinLoc* software package (Lomax et al., 2009) for probabilistic non-linear earthquake location in 3D heterogeneous Earth models and a 3D velocity model derived from the results of Fang et al. (2016).

To relocate events we use the source-receiver reciprocity and compute traveltime fields for each receiver by treating them as sources. We use the same velocity model as in White et al. (2019) and a spherical grid with 64, 128, and 128 nodes in the radial, polar, and azimuthal directions, respectively. The origin time is estimated for each node-arrival pair by subtracting the node-to-station traveltime from the observed arrival time, and the node that minimizes the standard deviation of origin time estimates for

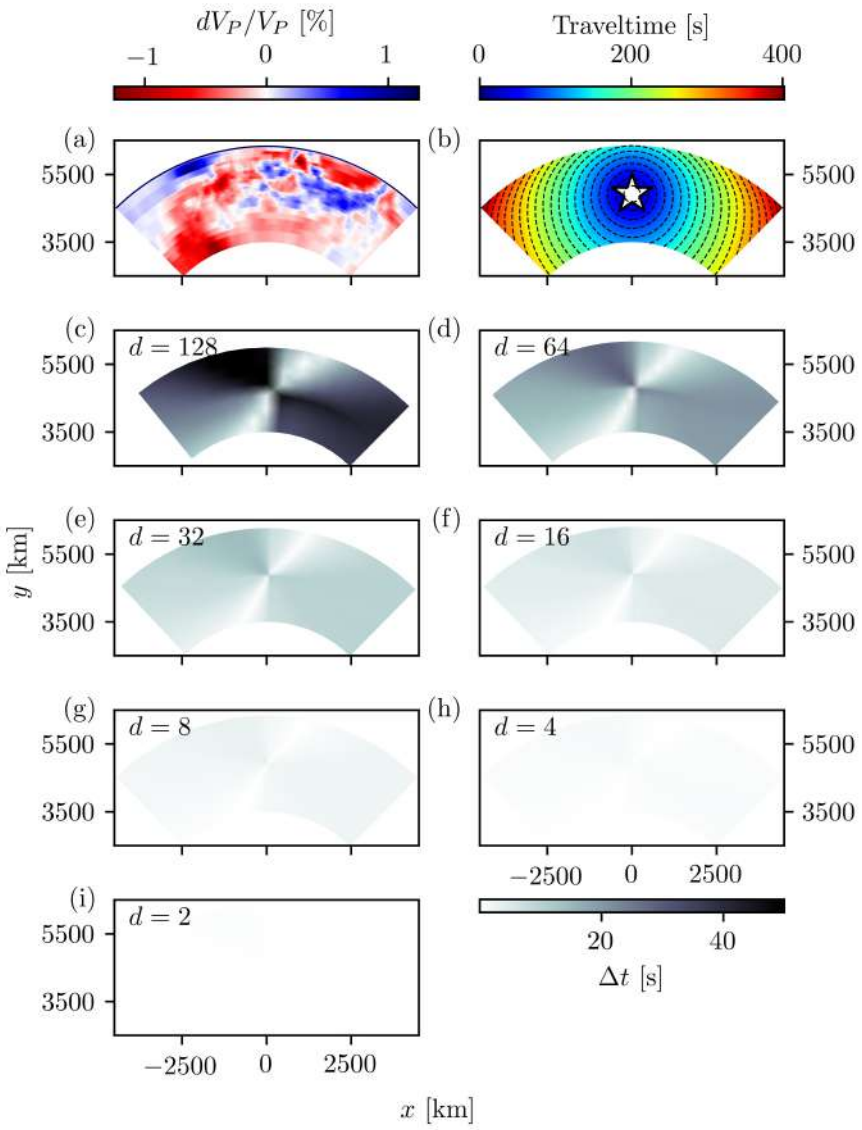

Figure 7: (a) A 2D slice of the MITP2008 velocity model shown as percent P-wave deviation from the *ak135* reference model (Kennett et al., 1995). (b) Traveltime field computed on a spherical grid with 1024 and 2048 nodes along the ρ and ϕ axis, respectively, for a source located at 1444 km depth. The white star with black outline and black dashed lines indicate the source location and wavefronts at 20 s intervals, respectively. (c-i) Difference, Δt, at coincident nodes between traveltime field in (b) and traveltime field computed on a grid decimated by a factor, d, specified in the top left corner of each plot.

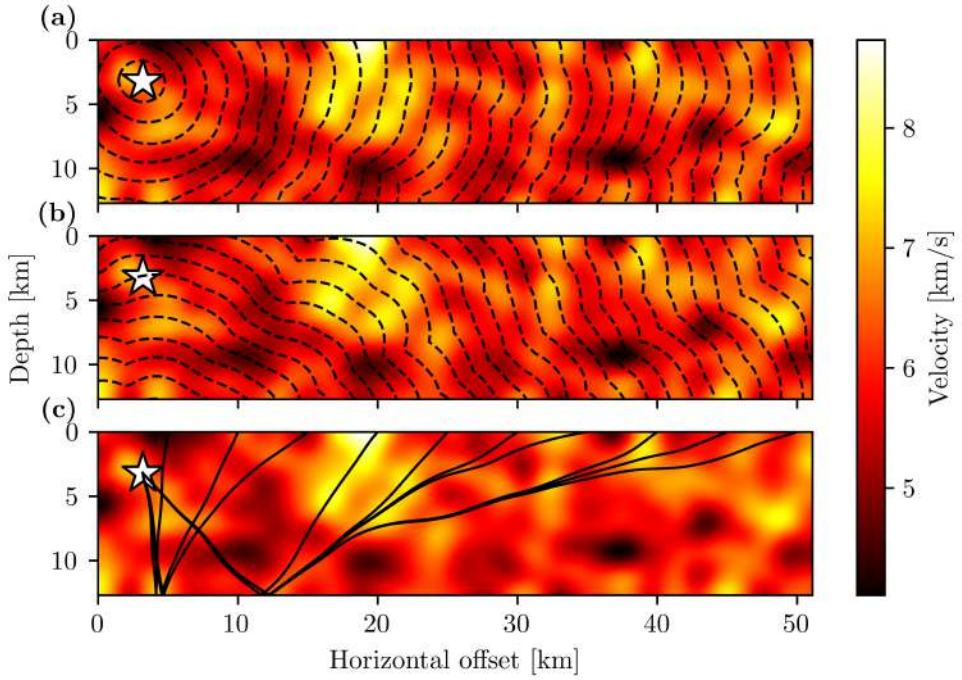

Figure 8: Reflected waves propagating through a velocity model with random velocity perturbations. The white star with black outline indicates the source location. (a,b) The black dashed lines represent wavefronts at 0.25 s intervals for (a) downgoing wavefronts and (b) wavefronts reflected by the lower boundary near 12 km depth. (c) The black lines represent raypaths reflected by the lower boundary and arriving at 5 km intervals at the surface.

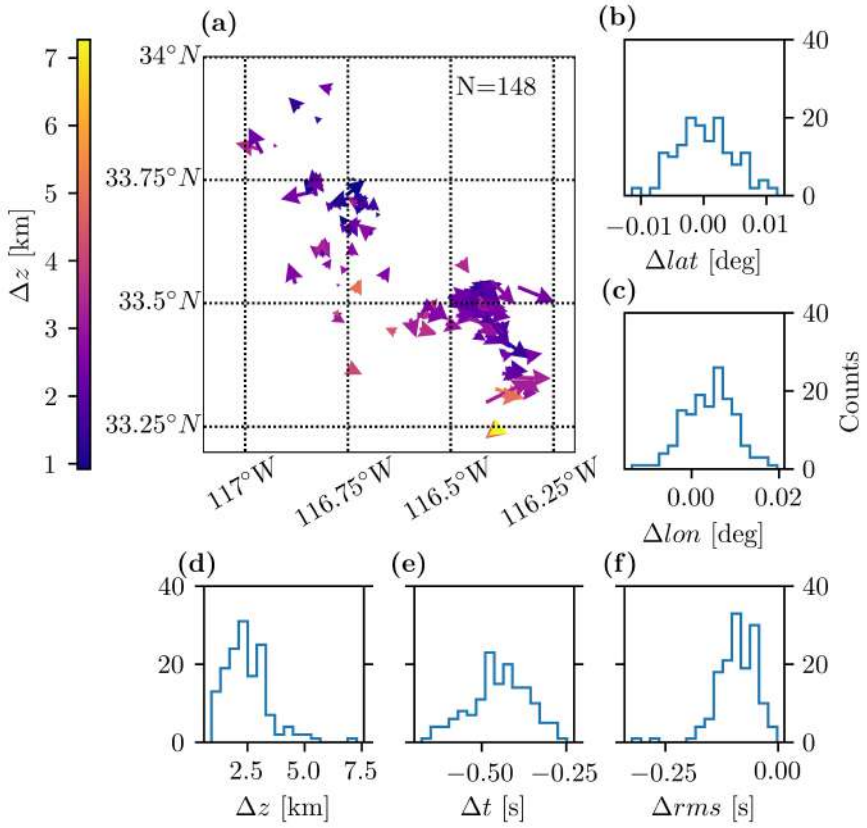

Figure 9: Comparing event locations obtained using NonLinLoc with locations obtained using the algorithm outlined in **Locating earthquakes**. Changes in all quantities, ΔQ, are calculated as $\Delta Q = Q_{NonLinLoc} - Q_{PyKonal}$ where $Q_{NonLinLoc}$ and $Q_{PyKonal}$ are the quantities associated with NonLinLoc and PyKonal locations, respectively. (a) Map of displacement vectors pointing from locations computed using NonLinLoc to locations computed as above. (b-f) Histograms of location parameter changes after relocating events: (b) change in event latitude; (c) change in event longitude; (d) change in event depth; (e) change in event origin time; and (f) change in RMS residual between observed and synthetic arrival times.

all arrivals provides an initial estimate of the hypocenter location. The average of the origin times estimated for the best fitting node provides an initial origin time value. The differential evolution algorithm implemented by the **scipy.optimize.differential_evolution** function in the SciPy Python package (Jones et al., 2001) then refines this initial location by searching a small region around the initial location for the value that minimizes the RMS between observed and synthetic arrival times.

Relocated events are consistent with NonLinLoc locations (Figure 9), but have lower RMS in all 148 cases suggesting that this rudimentary algorithm consistently finds small adjustments that better fit the data. More sophisticated location algorithms can be designed using PyKonal to solve the forward problem.

Discussion

Comparing with FM3D

PyKonal builds on the foundation laid by *FM3D* (de Kool et al., 2006) in four important ways: (a) it can improve accuracy by using hybrid coordinate systems; (b) it accounts for the azimuthal periodicity inherent in spherical coordinates to allow wavefronts to propagate across the $\phi = 0$ plane, making many global-scale problems easier to solve; (c) it solves both 2D and 3D problems; and (d) it offers a simple Python API to accommodate diverse use cases.

Repeating the simple point-source error analysis from **Accuracy** for identical problems solved using *FM3D* and *PyKonal* demonstrates the improved accuracy achieved by *PyKonal* (Figure 10). We consider a point source in homogeneous velocity structure and compare errors associated with *FM3D* and two different configurations of *PyKonal*. In all three cases, the computational grid comprises 256, 5, and 256 nodes along the radial, polar, and azimuthal axes, with 10 km, 0.125°, and 0.125° node intervals, respectively. Both FM3D and PyKonal can significantly reduce the error throughout the computational domain by using a refined source grid. The refined source grid for FM3D in this test spans 20 coarse-grid nodes along each coordinate axis and the node intervals are decreased by a factor of 10; we chose these parameters because they provide a reasonable tradeoff between execution time and accuracy. The refined source grid for PyKonal consists of a spherical grid centered on the source with a grid refinement factor of 5 and extending 40 coarse grid nodes in the radial direction. The accuracy of FM3D in this test approaches that of PyKonal in the region far from the source, but required over an order of magnitude more execution time (41.80 s compared to 1.12 s) to do so on our workstation (a Dell Optiplex 9020 workstation with a quad-core 3.4 GHz Intel Core i7-4770 CPU). Although PyKonal can achieve better accuracy with less execution time than FM3D, and is thus preferable in certain use cases, we note that FM3D offers more flexibility as it can track multiple reflection and transmission phases as well as teleseismic phases propagating through local 3D structure.

Wavefronts that cross over the $\phi = 0$ plane need to be carefully treated to solve problems at global scales. If the periodicity across this plane is neglected, the obtained solutions will, in the best-case scenario, only be valid in a hemisphere centered on the source. By properly accounting for this periodicity, our tool accommodates global-scale problems without any extra effort required of the user.

Python is a popular high-level interpreted programming language that suffers from slow execution as a result of the fact that it checks data types at run time. Languages that

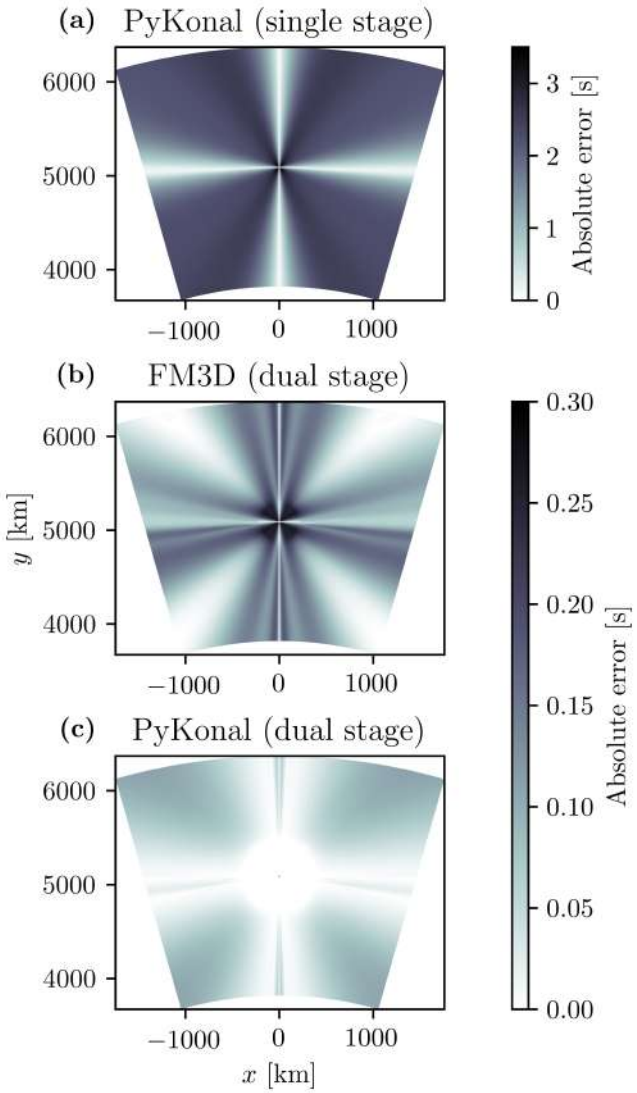

Figure 10: Absolute errors for traveltime fields computed through a homogeneous velocity model ($v = 1\,\mathrm{km/s}$) using three different methods: (a) *PyKonal* with a single coarse grid; (b) *FM3D* with a refined source grid and a coarse far-field grid; and (c) *PyKonal* with a refined source grid and coarse far-field grid.

check data types during compilation, such as C, C++, and Fortran, execute quickly but are slow for developments. We leverage the high-level scripting capabilities of Python while maintaining C-like speeds by producing compiled C extensions for Python using the Cython compiler framework. The code functions as any pure Python package would, but with the speed of C. This makes the presented tool user friendly while standing up to computationally-demanding tasks.

Applications

A robust ray tracer has many applications; we highlight the four uses that we had in mind while developing this tool: locating earthquakes, body-wave tomography, computing take-off angles, and Kirchhoff depth migration.

Locating earthquakes typically involves finding the set of spacetime coordinates that minimizes a misfit function between observed and synthetic arrival times. In seismically active fault zones where highly damaged rocks are juxtaposed against one or two different host rocks (e.g., Ben-Zion & Sammis, 2003; Fang et al., 2016) and other areas where wavespeed varies significantly, 3D heterogeneities need to be considered to accurately locate events. In **Locating earthquakes**, we showed that developing an inversion framework to locate earthquakes using *PyKonal* to solve the forward problem is straightforward. More sophisticated algorithms than the one presented here can be developed. One potential application of PyKonal is to incorporate it into a probabilistic location algorithm that uses a Markov Chain Monte Carlo approach to include *a priori* information about the data and model. Such an algorithm may provide more reliable location estimates and associated uncertainties.

Similar to locating earthquakes, body-wave tomography typically involves minimizing a misfit function between observed and synthetic arrival times. In addition to the optimal spacetime coordinates of the sources, the optimal velocity structure is sought in body-wave tomography. In active-source surveys, where the source locations are known, only the velocity model is sought. Again, being able to accurately synthesize traveltimes in the presence of 3D heterogeneity is crucial for deriving accurate velocity models.

Inverting first-motion polarity data for source mechanisms depends on the take-off angles at which observed rays leave the focal sphere, and small differences in take-off angles can produce significantly different responses at distant observation points. Because PyKonal can reduce traveltime errors in the near-source region (Figure 10), more accurate raypaths, and thus take-off angles, can be computed. More accurate take-off angles should yield more accurate focal mechanisms and lead to better resolution of rupture kinematics.

Kirchhoff depth migration (Schneider, 1978) is a method for transforming seismic data from the time domain to the depth domain that finds wide use in seismic image processing as a means of resolving structural complexities by backpropagating scattered energy to the scatterer's position in the image. Migrating data in this way requires computing traveltime fields that account for 3D structural heterogeneity. *PyKonal* can be integrated easily into imaging workflows to facilitate clearer and more accurate images of subsurface structures.

Jupyter Notebook Examples and Documentation

To promote reproducibility and provide examples of how to use the code, we include as supplementary material Jupyter note-

books that can be used to recreate Figures 3-8. We encourage interested readers to begin familiarizing themselves with the tool by reproducing the figures in this article. After running the example notebooks, users will benefit from API documentation, which is being actively developed and is available at `https://malcolmw.github.io/pykonal-docs`.

Conclusions

PyKonal is a new open-source Python package for computing traveltimes and tracing raypaths in 2D and 3D based on the Fast Marching Method for solving the Eikonal equation (Sethian, 1996). It is designed to be easy enough for novice programmers to use and efficient enough to solve complex research problems in seismology without sacrificing generality or extensibility. The most recent release of the code can be downloaded from `https://github.com/malcolmw/pykonal/releases`. Users are encouraged to submit bug reports via GitHub or via email to the first author at malcolm.white@usc.edu.

We are now using PyKonal in a flexible tomographic method requiring minimal a priori information based on Fang et al. (2020), and are extending that method to include a nonlinear event relocation step similar to the one in **Locating earthquakes**. We are applying the new methods to complementary data sets (Hutton et al., 2010; White et al., 2019) to derive integrated hierarchical images of the crust in Southern California with focus on the San Jacinto Fault Zone and region surrounding the 5 July 2019 M_w7.1 Ridgecrest earthquake sequence.

Data and Resources

The *MITP2008* velocity model used in this paper is available via the cited reference (Li et al., 2008). Figures 3-10 were made using *Matplotlib* (Hunter, 2007). Supplemental Material for this article includes six Jupyter Notebooks to recreate Figures 3-8.

Acknowledgements

The study was supported by the U.S. Department of Energy (award DE-SC0016520). The manuscript benefitted from useful comments by N. Rawlinson and an anonymous referee.

References

Afnimar, & Koketsu, K. (2000). Finite difference traveltime calculation for head waves travelling along an irregular interface. *Geophysical Journal International*, *143*(3), 729–734. doi: 10.1046/j.1365-246X.2000.00269.x

Aldridge, D. F., & Abdelsalifah, T., & Fomel, S. (2001). Implementing the Fast Marching Eikonal Solver: Spherical Versus Cartesian Coordinates. *Geophysical Prospecting*, *49*(2), 165–178. doi: 10.1046/j.1365-2478.2001.00245.x

Ben-Zion, Y., & Sammis, C. G. (2003). Characterization of Fault Zones. *Pure and Applied Geophysics*, *160*(3), 677–715. doi: 10.1007/PL00012554

Buske, S., & Kastner, U. (2004). Efficient and accurate computation of seismic traveltimes and amplitudes. *Geophysical Prospecting*, *52*(4), 313–322. doi: 10.1111/j.1365-2478.2004.00417.x

de Kool, M., Rawlinson, N., & Sambridge, M. (2006). A Practical Grid-Based Method for Tracking Multiple Refraction and Reflection Phases in Three-Dimensional Heterogeneous Media. *Geophysical Journal International*, *167*(1), 253–270. doi: 10.1111/j.1365-246X.2006.03078.x

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. doi: 10.1007/BF01386390

Fang, H., van der Hilst, R. D., de Hoop, M. V., Kothari, K., Gupta, S., & Dokmanić, I. (2020). Parsimonious Seismic Tomography with Poisson Voronoi Projections: Methodology and Validation. *Seismological Research Letters*, *91*(1), 343–355. doi: 10.1785/0220190141

Fang, H., Zhang, H., Yao, H., Allam, A., Zigone, D., Ben-Zion, Y., ... van der Hilst, R. (2016). A new algorithm for three-dimensional joint inversion of body wave and surface wave data and its application to the Southern California plate boundary region. *Journal of Geophysical Research: Solid Earth*, *121*(5), 3557–3569. doi: 10.1002/2015JB012702

Harten, A., & Osher, S. (1987). Uniformly High-Order Accurate Nonoscillatory Schemes. I. *SIAM Journal on Numerical Analysis*, *24*(2), 279–309. doi: 10.1137/0724022

Hole, J. A., & Zelt, B. C. (1995). 3-D finite-difference reflection travel times. *Geophysical Journal International*, *121*(2), 427–434. doi: 10.1111/j.1365-246X.1995.tb05723.x

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

Hutton, K., Woessner, J., & Hauksson, E. (2010). Earthquake Monitoring in Southern California for Seventy-Seven Years (1932-2008). *Bulletin of the Seismological Society of America*, *100*(2), 423–446. doi: 10.1785/0120090130

Jones, E., Oliphant, T., & Peterson, P. (2001). *SciPy: Open Source Scientific Tools for Python.* Retrieved from `http://www.scipy.org/`

Julian, B. R., & Gubbins, D. (1977). Three-dimensional seismic ray tracing. *Journal of Geophysics*, *43*, 95–114.

Kennett, B. L. N., Engdahl, E. R., & Buland, R. (1995). Constraints on Seismic Velocities in the Earth from Traveltimes. *Geophysical Journal International*, *122*(1), 108–124. doi: 10.1111/j.1365-246X.1995.tb03540.x

Kim, S., & Cook, R. (1999). 3-D traveltime computation using second-order ENO scheme. *Geophysics*, *64*(6), 1867–1876. doi: 10.1190/1.1444693

Li, C., van der Hilst, R. D., Engdahl, E. R., & Burdick, S. (2008). A New Global Model for P Wave Speed Variations in Earth's Mantle. *Geochemistry, Geophysics, Geosystems*, *9*(5). doi: 10.1029/2007GC001806

Liu, X.-D., Osher, S., & Chan, T. (1994). Weighted Essentially Non-oscillatory Schemes. *Journal of Computational Physics*, *115*(1), 200–212. doi: 10.1002/fld.3889

Lomax, A., Michelini, A., & Curtis, A. (2009). *Earthquake Location, Direct, Global-Search Methods*. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-27737-5

Osher, S., & Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, *79*(1), 12–49. doi: 10.1016/0021-9991(88)90002-2

Podvin, P., & Lecomte, I. (1991). Finite difference computation of traveltimes in very contrasted velocity models: a massively parallel approach and its associated tools. *Geophysical Journal International*, *105*(1), 271–284. doi: 10.1111/j.1365-246X.1991.tb03461.x

Qian, J., & Symes, W. W. (2002a). An adaptive finite-difference method for traveltimes and amplitudes.

Geophysics, *67*(1), 167–176. doi: 10.1190/1.1451472

Qian, J., & Symes, W. W. (2002b). Finite-difference quasi- P traveltimes for anisotropic media. *Geophysics*, *67*(1), 147–155. doi: 10.1190/1.1451438

Qin, F., Luo, Y., Olsen, K. B., Cai, W., & Schuster, G. T. (1992). Finite-difference solution of the eikonal equation along expanding wavefronts. *Geophysics*, *57*(3), 478–487. doi: 10.1190/1.1443263

Rawlinson, N., Hauser, J., & Sambridge, M. (2008). Seismic ray tracing and wavefront tracking in laterally heterogeneous media. *Advances in Geophysics*, *49*(07), 203–273. doi: 10.1016/S0065-2687(07)49003-3

Rawlinson, N., & Sambridge, M. (2004a). Multiple reflection and transmission phases in complex layered media using a multistage fast marching method. *Geophysics*, *69*(5), 1338–1350. doi: 10.1190/1.1801950

Rawlinson, N., & Sambridge, M. (2004b). Wave Front Evolution in Strongly Heterogeneous Layered Media Using the Fast Marching Method. *Geophysical Journal International*, *156*(3), 631–647. doi: 10.1111/j.1365-246X.2004.02153.x

Reshef, M., & Kosloff, D. (1986). Migration of common-shot gathers. *Geophysics*, *51*(2), 324–331. doi: 10.1190/1.1442091

Rouy, E., & Tourin, A. (1992). A Viscosity Solutions Approach to Shape-From-Shading. *SIAM Journal on Numerical Analysis*, *29*(3), 867–884. doi: 10.1137/0729053

Schneider, W. A. (1978). Integral Formulation for Migration in Two and Three Dimensions. *Geophysics*, *43*(1), 49–76. doi: 10.1190/1.1440828

Schneider, W. A., Ranzinger, K. A., Balch, A. H., & Kruse, C. (1992). A dynamic programming approach to first arrival traveltime computation in media with arbitrarily distributed velocities. *Geophysics*, *57*(1), 39–50. doi: 10.1190/1.1443187

Sescu, A. (2015). Numerical anisotropy in finite differencing. *Advances in Difference Equations*, *2015*(1), 9. doi: 10.1186/s13662-014-0343-0

Sethian, J. A. (1996). A Fast Marching Level Set Method for Monotonically Advancing Fronts. *Proceedings of the National Academy of Sciences*, *93*(4), 1591–1595. doi: 10.1073/pnas.93.4.1591

Sethian, J. A. (1999). Fast Marching Methods. *SIAM Review*, *41*(2), 199–235. doi: 10.1137/S0036144598347059

Sethian, J. A., & Popovici, A. M. (1999). 3-D Traveltime Computation Using the Fast Marching Method. *Geophysics*, *64*(2), 516–523. doi: 10.1190/1.1444558

Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341–359. doi: 10.1023/A:1008202821328

Thurber, C. H., & Ellsworth, W. L. (1980). Rapid Solution of Ray Tracing Problems in Heterogeneous Media. *Bulletin of the Seismological Society of America*, *70*(4), 1137–1148.

Um, J., & Thurber, C. (1987). A Fast Algorithm for Two-Point Seismic Ray Tracing. *Bulletin of the Seismological Society of America*, *77*(3), 972–986.

van Trier, J., & Symes, W. W. (1991). Upwind finite-difference calculation of traveltimes. *Geophysics*, *56*(6), 812–821. doi: 10.1190/1.1443099

Versteeg, R. (1994). The Marmousi Experience: Velocity Model Determination on a Synthetic Complex Data Set. *The Leading Edge*, *13*(9), 927–936. doi:

10.1190/1.1437051

Vidale, J. E. (1988). Finite-Difference Calculation of Travel Times. *Bulletin of the Seismological Society of America*, *78*(6), 2062–2076.

Vidale, J. E. (1990). Finite-Difference Calculation of Traveltimes in Three Dimensions. *Geophysics*, *55*(5), 521–526. doi: 10.1190/1.1442863

White, D. J. (1989). Two-Dimensional Seismic Refraction Tomography. *Geophysical Journal International*, *97*(2), 223–245. doi: 10.1111/j.1365-246X.1989.tb00498.x

White, M. C. A., Ben-Zion, Y., & Vernon, F. L. (2019). A Detailed Earthquake Catalog for the San Jacinto Fault-Zone Region in Southern California. *Journal of Geophysical Research: Solid Earth*, *124*, 6908–6930. doi: 10.1029/2019JB017641

Zhao, H. (2004). A fast sweeping method for Eikonal equations. *Mathematics of Computation*, *74*(250), 603–628. doi: 10.1090/S0025-5718-04-01678-3

Authors' Addresses

1. Malcolm White
   University of Southern California
   Department of Earth Sciences
   Zumberge Hall of Science (ZHS)
   3651 Trousdale Pkwy
   Los Angeles, CA 90089-0740

2. Hongjian Fang
   Massachusetts Institute of Technology,
   Department of Earth, Atmosphere and Planetary Sciences
   Green Bldg
   77 Massachusetts Ave
   Cambridge, MA 02139

3. Nori Nakata
   Massachusetts Institute of Technology,
   Department of Earth, Atmosphere and Planetary Sciences
   Green Bldg
   77 Massachusetts Ave
   Cambridge, MA 02139

4. Yehuda Ben-Zion
   University of Southern California
   Department of Earth Sciences
   Zumberge Hall of Science (ZHS)
   3651 Trousdale Pkwy
   Los Angeles, CA 90089-0740