# Fast Marching Method implementation to solve the 2D isotropic eikonal equation

Jonathas S. Maciel; Victor K. Ramalho; Diego F. B. Pacheco (SENAI CIMATEC) and Reynam C. Pestana (UFBA & INCT-GP)



## Abstract

**The Fast Marching Method (FMM) is widely used to solve static Hamilton-Jacobi equations (Eikonal equations), presented in many applications like robotics, medical scans, optimal design, and geophysics exemplified by Kirchhoff migration and traveltime tomography. This abstract introduces the reader to the single-pass methods, showing the 2D FMM implementation and its High Accuracy version HAFMM. It is a good start point to understand the method's limitations, for further developments in traveltime precision and high dimension extension.**

## Introduction

The high frequency approach of the wave equation plays an important role in geophysics, in the sense of simplifying wave phenomena in terms of the direct arrival events, with that, it is possible to build the time boundaries for each geophysical experiment (traveltime maps). Ray tracing methods have a high degree of travetime accuracy along of rays paths, but could have interpolation problems if the media is very complex, mainly because this kind of media tend to diverge the rays from each other, causing shadow and caustics zones (Cerveny, 2001). A finite-difference scheme was proposed by Vidale (1988) to overcome the ray-tracing issues, however, Vidale's solution may violate the causality condition in moderate to large impedance contrasts (Qin et al., 1992). Shortest path ray tracing (SPR) is a grid based method to calculate the shortest distances of networks (Moser, 1991). It is specified a grid of nodes inside of velocity model and the minimum traveltime need to be found only through the nodes connections. SPR is usually solved by Dijkstra-like algorithms.

The Fast Marching Method (FMM) is one the most common eikonal solvers. It was developed by Sethian (1996) in a similar solving process of Dijkstra's method. Dijkstra's method aims to compute the shortest distance path on a network, but has the limitation to follow the distance in the connection paths. No matter how refined the model for the eikonal problem, Dijkstra's method will always present staircase patterns on it (Tsitsiklis, 1995). In order to find a properly a continuous solution, Sethian (1996) used the upwind finite-difference to approximate the gradient operator in the eikonal equation, at the same time retain the one-pass idea of Dijkstra's algorithm.

The main goal of this work is to introduce the reader to the single-pass method as FMM. We start contextualizing the eikonal approach coming from 2D isotropic acoustic wave equation with constant density, then we introduce upwind finite-difference schemes, as an approach for the gradient operator. We detail how FMM ensures the causality during wave propagation using upwind finite-difference, and a heap data structure to order the grid points update. We show the method limitations and how would overcome these issues with the high accuracy version of FMM. Finally, we have included the Fortran 90 code as a numerical receipt.

## Method

*Isotropic 2D eikonal equation*

The eikonal equation can be obtained directly from the acoustic wave equation with constant density in the frequency domain,

$$\nabla^2 U(\mathbf{x},\omega) + \omega^2 s^2(\mathbf{x}) U(\mathbf{x},\omega) = 0, \tag{1}$$

where $s(\mathbf{x})$ is the slowness field, $U(\mathbf{x},\omega)$ is the pressure field at the position $\mathbf{x} = (z,x)$ and $\omega$ is the angular frequency component.

For instance assuming an impulse wavefield $u(\mathbf{x},t) = A(\mathbf{x})\delta[t-\tau(\mathbf{x})]$, in a non-dispersive medium, written in frequency domain as

$$U(\mathbf{x},\omega) = A(\mathbf{x}) e^{[i\omega\tau(\mathbf{x})]}, \tag{2}$$

where the constant time surface $t = \tau(\mathbf{x})$ means the wavefront, we can substitute 2 in 1 to get the following expression:

$$\left[\omega^2 A(s^2 - \nabla\tau\cdot\nabla\tau) + i\omega(A\nabla^2\tau + 2\nabla A\cdot\nabla\tau) + \nabla^2 A\right] e^{i\omega\tau} = 0. \tag{3}$$

For equation 3 to be valid, each $\omega$ coefficient must be independently null. Therefore, we have three differential equations:

$$\nabla^2 A = 0, \tag{4}$$

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

$$\nabla\tau\cdot\nabla\tau = s^2. \tag{6}$$

Equation 4 ensures the amplitude decays up to the values of boundary conditions. Equation 5 is known as the transport equation and describes the dynamics of amplitude decay, and 6 is known as the eikonal equation which describes the kinematics of the wavefield.

*Fast Marching Method*

The Fast Marching Method (FMM) is a technique for modeling the evolution of closed surfaces such as wavefronts. The result is a temporal distance map, very useful in fluid simulation, robotics, visual medical computing and in geophysics to determine the traveltime of head waves (Gómez et al., 2019).

The method is based on a data structure to store and sequence the updating of nodes (heap data structure), and on the upwind finite-difference scheme to ensure the causality of wavefront evolution.

The upwind scheme for the eikonal equation 6 is given by

$$\max\left(D^{-z}\tau, -D^{+z}\tau, 0\right)^2 + \max\left(D^{-x}\tau, -D^{+x}\tau, 0\right)^2 = s^2, \quad (7)$$

where

$$\begin{aligned} D^{\pm z}\tau &= p_z\left(\tau_{i_z,i_x} - \tau_{i_z\pm1,i_x}\right), \\ D^{\pm x}\tau &= p_x\left(\tau_{i_z,i_x} - \tau_{i_z,i_x\pm1}\right), \end{aligned} \quad (8)$$

are the finite-difference scheme for forward $(+)$ or backward $(-)$ derivatives, $p_z = \mp 1/\Delta_z$ and $p_x = \mp 1/\Delta_x$ store the finite-difference coefficient and the grid spacing $\Delta_z$, $\Delta_x$ (Figure 1).

Substituting 8 in 7, aiming to take the minimum neighbor grid point to give the maximum derivative, and defining the vertical and horizontal neighboring grid point to be chosen as

$$\begin{aligned} \tau_v &= \min(\tau_{i_z+1,i_x}, \tau_{i_z-1,i_x}), \\ \tau_h &= \min(\tau_{i_z,i_x+1}, \tau_{i_z,i_x-1}). \end{aligned} \quad (9)$$

We can rewrite equation 7 in the form

$$\max\left[p_z\left(\tau - \tau_v\right), 0\right]^2 + \max\left[p_x\left(\tau - \tau_h\right), 0\right]^2 = s^2 \quad (10)$$

with $\tau = \tau_{i_z,i_x}$.

Once the slowness is always positive ($s > 0$), $\tau$ must be greater than $\tau_v$ and $\tau_h$, then it is safe to simplify equation 10 to

$$\left[p_z\left(\tau - \tau_v\right)\right]^2 + \left[p_x\left(\tau - \tau_h\right)\right]^2 = s^2. \quad (11)$$

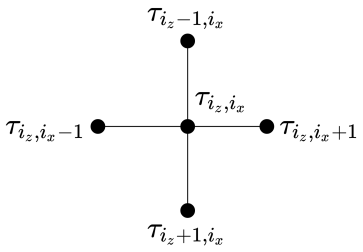


Figure 1: Updating grid point.

The eikonal equation in 11 has a quadratic form $a\tau^2 + b\tau + c = 0$ with coefficients

$$\begin{aligned} a &= p_z^2 + p_x^2 \\ b &= -2(p_z^2\tau_v + p_x^2\tau_h) \\ c &= p_z^2\tau_v^2 + p_x^2\tau_h^2 - s^2 \end{aligned} \quad (12)$$

The large solution is the one we are looking for (equation 7), which means

$$\tau^{zx} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}. \quad (13)$$

It is known as two-side update scheme since $\tau_v$ and $\tau_h$ is taking into account. This quadratic form is only valid if $\tau^{zx} \geq \min(\tau_v, \tau_h)$. Some of the surrounding grid points may have infinity values (the points where the wavefront has not reached yet). In this situation, the causality condition ($b^2 > 4ac$) fails, and becomes necessary to use the one-side update scheme, where equation 11 is solved for each direction independently

$$\tau^z = \tau_v + \frac{s}{|p_z|}, \quad (14)$$

$$\tau^x = \tau_h + \frac{s}{|p_x|}. \quad (15)$$

The final solution is the minimum value between two-side and one-side update schemes

$$\tau = \min(\tau^{zx}, \tau^z, \tau^x). \quad (16)$$

*High Accuracy Fast Marching Method*

It is easy to see that FMM has low accuracy. Figure 2 illustrates the problem when compared to the exact solution of the distance map

$$\tau(z,x) = \sqrt{(z - z_s)^2 + (x - x_s)^2}, \quad (17)$$

where $(z_s, x_s)$ means the source position, $\tau(z_s, x_s) = 0$, $\Delta_z = \Delta_x = 1$m and constant slowness $s = 1$ s/m.

The calculated value (in red) represents an error of $20\%$ more than the exact solution. This is a limitation of discretization in an attempt to capture the exact wavefront curvature. It is possible to mitigate this problem by increasing the accuracy of the finite-difference operator. A High Accuracy version of FMM (HAFMM) changes only in how to choose $\tau_v$, $\tau_h$ and $p_z$, $p_x$ to select the first or second order approach in finite-difference operators.

The second order of the finite-difference approach is represented by

$$D_2^{\pm z}\tau = p_z\left(\tau_{i_z,i_x} - \tau_v\right), \quad (18)$$

$$D_2^{\pm x}\tau = p_x\left(\tau_{i_z,i_x} - \tau_h\right), \quad (19)$$

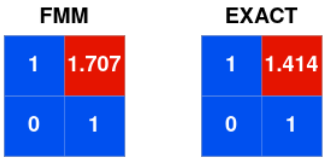


Figure 2: FMM curvature issue.

where

$$\tau_v = \frac{1}{3}\left(4\tau_{i_z\pm1,i_x} - \tau_{i_z\pm2,i_x}\right), \quad (20)$$

$$\tau_h = \frac{1}{3}\left(4\tau_{i_z,i_x\pm1} - \tau_{i_z,i_x\pm2}\right), \quad (21)$$

$$p_z = \mp\frac{3}{2\Delta_z}, \quad (22)$$

$$p_x = \mp\frac{3}{2\Delta_x}. \quad (23)$$

The criteria for choosing between forward or backward derivative is initially the same for the first order (equation 9). Once decided we need to check if there are finite values in the next two grid points to obey the causality condition, $\tau_{i\pm1} > \tau_{i\pm2}$, to proceed with second order. If this is not true, we keep updating using the first order.

The HAFMM method will not be used in the first iterations because there are not enough points for the calculation. However, the error accumulation tends to decrease as its wavefront propagates. Figure 3 illustrates the error between FMM and HAFMM for distance fronts (equation 17). It is observed that the HAFMM has a much more rounded circle and FMM produce flattening along $\pm45°$ axis (Bærentzen, 2001). Figure 4 shows an application for a larger and more complex model (Marmousi), where we can see the FMM can significantly change from HAFMM if the wavefront passes for high velocity values. The finite-difference accuracy increase allows to reduce the accumulation error, however, it does not avoid the error caused by the curvature near the source point. According to Luo, S., Qian, J. (2012), a factored solution of the eikonal equation allows mitigating this problem. In the context of this work, we will not address this method.

*Implementation*

The key of the FMM algorithm is the separation of regions already computed from regions in computing and from regions to be computed, organized by causality criteria in which establishes that the information propagates from small to large $\tau$ values.

Figure 5 illustrates the separation of these regions in a $10\times10$ grid by indentation: "ALIVE", "CLOSE", and "FAR". In the "ALIVE" region we have the points already computed, where the source position, $\tau(z_s,x_s)=0$, is the first point to have this tag.

The points tagged "CLOSE" are in the computing process, whose information is updated according to "ALIVE" points in its neighborhood. This region is implemented in the priority queue data structure (heap structure). The most common implementation is the binary heap (Sethian, 1999) because of its high computational efficiency. We use a non

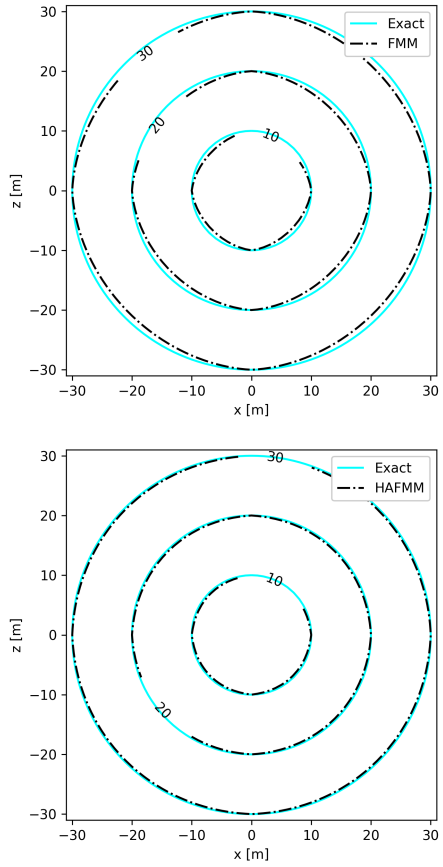


Figure 3: Distance circles for 10m, 20m and 30m radius. The exact solution from equation 17 in solid cyan line, FMM (top) and HAFMM (bottom) in black dashed dot line.

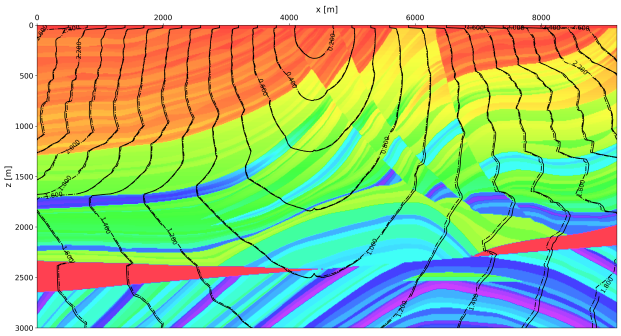


Figure 4: FMM (dashed dot line) and HAFMM (solid line) application in the Marmousi model. Source position at $z=0$m and $x=4396$m.

regular $d$-ary heap structure ($d$-ary is a generalization of the binary heap where each parent node has $d$ children). The heap property (the son leaf has a greater value than the father leaf min-heap) is kept by the causality constraint of the upwind finite-difference scheme. Figure 6 shows the heap structure for $5\times5$ grid model with source injection at the center. Each tree layer sequences the update priority as represented by different colors. In each sequence, the grid points are updated by equations 13, 14, 15, 16, and after that, each grid point is re-tagged as "ALIVE". If there are "FAR" tags in its neighborhood, we re-tag them as

"CLOSE" to build the next tree layer. The tree is build by storing an array with the size of the model minus one (we do not need to store the source position), and each layer is differentiated by pointing out the memory location of the initial and final sample of the sub-array.

The region tagged with "FAR" indicates the points still to be counted. These are allocated with maximum machine floating point representation (infinity), so as, to be disregarded in eikonal traveltime calculation. As the wavefront moves forward, the "CLOSE" tags will be updated for "ALIVE", and "FAR" tags for "CLOSE" until every grid point be "ALIVE" thus ending the modeling process. Figure 7 shows all this process.

*The algorithm*

We make available the complete code in Fortran 90 of the HAFMM in Figure 8. In this section, we are going to detail the main points of the code.

Initialization:

- Set the time in source position, $T(izs, ixs)$ as zero.
- All other grid points set $T(iz, ix) = +\infty$.
- Use $Tag(izs, ixs) =$"**A**LIVE" to freeze the source from update.
- Tag the closest grid points to the source as $Tag(iz, ix) =$"**C**LOSE".
- Tag all other grid points as $Tag(iz, ix) =$"**F**AR".

Modeling:

1. For each $Tag(iz, ix) =$"**C**LOSE" calcule the eikonal distance (T).
2. Fix the new value in $T(iz, ix)$ tagging it as $Tag(iz, ix) =$"**A**LIVE" and if their neighbor is "**F**AR" set as "**C**LOSE".
3. Repeat $1$ and $2$ until there are no "**C**LOSE" tags.

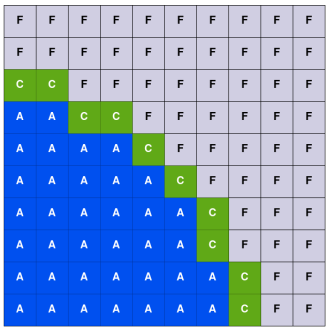

Figure 5: FMM regions: "ALIVE" for computed (blue), "CLOSE" for computing (green) and "FAR" to be computed (gray).

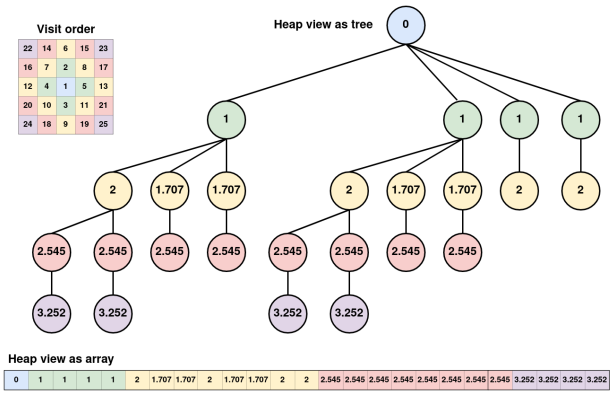


Figure 6: Non regular $d$-ary heap structure. Tree and array display.

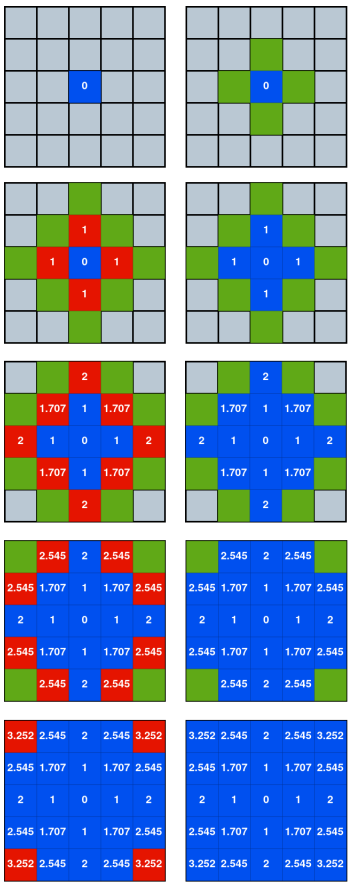

Figure 7: Eikonal modeling in a $5 \times 5$ grid. From top left to the bottom right. The algorithm initializes with the source at position $(3,3)$ with $\tau(z_s, x_s) = 0$. Next tag the closest point (green cross) as "CLOSE". Update the next heap layer and calculate $\tau$ (red) and tag them as "ALIVE" (blue). Find new closest point, re-flagged them from "FAR" (gray) to "CLOSE" and update the new heap layer. Repeat until only "ALIVE" points remain.

## Conclusions

We briefly presented one of the most popular single-pass methods called the Fast Marching Method. The

2D isotropic eikonal equation was the focus as an initial view of the method. The extension for high dimensions is straightforward. We also show that some of the FMM limitations may be handled by the High Accuracy version of FMM (HAFMM). The method was coded and also tested in the complex Marmousi velocity model. In addition, the Fortran code of our implementation is available here.

### Acknowledgments

This work was supported by CENPES/Petrobras and ANP through the Marchenko project at SENAI CIMATEC. Special acknowledgment to the geophysical team at SENAI CIMATEC for discussions and contributions.

### References

Bærentzen, J. A. On the implementation of fast marching methods for 3D lattices:, 2001.

Cerveny, V., 2001, Seismic ray theory: Cambridge University Press.

Gómez, J. V., Álvarez, D., Garrido, S., and Moreno, L., 2019, Fast methods for eikonal equations: An experimental survey: IEEE Access, **7**, 39005–39029.

Luo, S., Qian, J., 2012, Fast sweeping methods for factored anisotropic eikonal equations: Multiplicative and additive factors.: Journal of Scientific Computing, **52**, 360–382.

Moser, T. J., 1991, Shortest path calculation of seismic rays: Geophysics, **56**, no. 1, 59–67.

Qin, F., Luo, Y., Olsen, K. B., Cai, W., and Schuster, G. T., 1992, Finite-difference solution of the eikonal equation along expanding wavefronts: Geophysics, **57**, no. 3, 478–487.

Sethian, J. A., 1996, A fast marching level set method for monotonically advancing fronts: Proceedings of the National Academy of Sciences, **93**, no. 4, 1591–1595.

Sethian, J. A., 1999, Fast marching methods: SIAM Review, **41**, no. 2, 199–235.

Tsitsiklis, J. N., 1995, Efficient algorithms for globally optimal trajectories: IEEE Transactions on Automatic Control, **40**, no. 9, 1528–1538.

Vidale, J., 1988, Finite-difference calculation of travel times: Bulletin of the Seismological Society of America, **78**, no. 6, 2062–2076.

```
SUBROUTINE Isotropic_Eikonal_HAFMM(izs,ixs,nz,nx,dz,dx,slowness,Tmap,ierr)
 IMPLICIT NONE
 INTEGER(4),INTENT(IN)          :: izs,ixs          !Source position indexes
 INTEGER(4),INTENT(IN)          :: nz,nx            !Model size
 REAL(4),INTENT(IN)             :: dz,dx            !Sampling
 REAL(4),INTENT(IN)             :: slowness(nz,nx) !Slowness
 REAL(4),INTENT(OUT)            :: Tmap(nz,nx)      !Travel time map
 INTEGER(4),INTENT(OUT)         :: ierr             !Error flag
 REAL(8),ALLOCATABLE            :: T(:,:)           !Time
 CHARACTER(LEN=1),ALLOCATABLE :: Tag(:,:)         !Grid flags
 INTEGER(4)                     :: iz,ix            !Grid indexes
 INTEGER(4)                     :: nclose,iclose    !Close counter and index
 INTEGER(4)                     :: min_iclose       !Min close point index
 INTEGER(4)                     :: max_iclose       !Max close point index
 REAL(8)                        :: a,b,c,delta      !Quadratic coffiecients
 REAL(8)                        :: twoside          !Twoside time update
 REAL(8)                        :: onesidez         !Oneside time update z
 REAL(8)                        :: onesidex         !Oneside time update x
 REAL(8)                        :: pz,px            !FD coeficient
 REAL(8)                        :: Th,Tv            !Neighbor time distance
 TYPE heap_struc
   INTEGER(4)                   :: hsiz             !Z index
   INTEGER(4)                   :: hsix             !X index
 END TYPE heap_struc
 TYPE(heap_struc),ALLOCATABLE :: heap(:)          !Data structure

 ALLOCATE(T(-1:nz+2,-1:nx+2),Tag(-1:nz+2,-1:nx+2),heap(1:nz*nx-1),STAT=ierr)
 IF(ierr/=0) RETURN

 !------ Initialization ---------
 T(:,:)         = HUGE(1.0e0)
 Tag(:,:)       = 'O' !Out
 Tag(1:nz,1:nx) = 'F' !Far
 Tag(izs,ixs)   = 'A' !Alive
 T(izs,ixs)     = 0.0 !First time

 ! Tag the first close grid points
 nclose = 0
 IF(izs-1 >= 1 )THEN
    Tag(izs-1,ixs) = 'C'
    nclose = nclose + 1
    heap(nclose)%hsiz = izs-1
    heap(nclose)%hsix = ixs
 END IF
 IF(izs+1 <= nz)THEN
    Tag(izs+1,ixs) = 'C'
    nclose = nclose + 1
    heap(nclose)%hsiz = izs+1
    heap(nclose)%hsix = ixs
 END IF
 IF(ixs-1 >= 1 )THEN
    Tag(izs,ixs-1) = 'C'
    nclose = nclose + 1
    heap(nclose)%hsiz = izs
    heap(nclose)%hsix = ixs-1
 END IF
 IF(ixs+1 <= nx)THEN
    Tag(izs,ixs+1) = 'C'
    nclose = nclose + 1
    heap(nclose)%hsiz = izs
    heap(nclose)%hsix = ixs+1
 END IF
 max_iclose = nclose
 min_iclose = 1

 !------ Modeling ---------
 DO WHILE(min_iclose <= max_iclose)
    DO iclose = min_iclose,max_iclose

         iz = heap(iclose)%hsiz
         ix = heap(iclose)%hsix

         !Forward or backward derivative
         IF( T(iz+1,ix) < T(iz-1,ix) )THEN
           pz = -1.0/dz
           Tv = T(iz+1,ix)
           IF(T(iz+2,ix) < T(iz+1,ix) )THEN
              Tv = (4.0*T(iz+1,ix)-T(iz+2,ix))/3.0
              pz = -3.0/(2.0*dz)
           END IF
         ELSE
           pz = 1.0/dz
           Tv = T(iz-1,ix)
           IF(T(iz-2,ix) < T(iz-1,ix) )THEN
              Tv = (4.0*T(iz-1,ix)-T(iz-2,ix))/3.0
              pz = 3.0/(2.0*dz)
           END IF
         END IF
         IF( T(iz,ix+1) < T(iz,ix-1) )THEN
           px = -1.0/dx
           Th = T(iz,ix+1)
           IF(T(iz,ix+2) < T(iz,ix+1) )THEN
              Th = (4.0*T(iz,ix+1)-T(iz,ix+2))/3.0
              px = -3.0/(2.0*dx)
           END IF
         ELSE
           px = 1.0/dx
           Th = T(iz,ix-1)
           IF(T(iz,ix-2) < T(iz,ix-1) )THEN
              Th = (4.0*T(iz,ix-1)-T(iz,ix-2))/3.0
              px = 3.0/(2.0*dx)
           END IF
         END IF

         !Quadratic parameters
         a = pz**2 + px**2
         b = -2.0*(pz*pz*Tv + px*px*Th)
         c = (pz*Tv)**2 + (px*Th)**2 - slowness(iz,ix)**2
         delta = b**2 - 4.0*a*c

         !Isotropic Eikonal solution
         twoside = HUGE(1.0d0)
         IF(delta > 0.0) twoside = (-b+SQRT(delta))/(2.0*a)
         onesidez = Tv + slowness(iz,ix)/ABS(pz)
         onesidex = Th + slowness(iz,ix)/ABS(px)

         !Update the minimum time and tag as alive
         T(iz,ix)   = MIN(twoside,onesidez,onesidex)
         Tag(iz,ix) = 'A'

         !Now if the neighbohr of (iz,ix) is FAR set as close
         IF(Tag(iz-1,ix) == 'F')THEN
            Tag(iz-1,ix) = 'C'
            nclose = nclose + 1
            heap(nclose)%hsiz = iz-1
            heap(nclose)%hsix = ix
         END IF
         IF(Tag(iz+1,ix) == 'F')THEN
            Tag(iz+1,ix) = 'C'
            nclose = nclose + 1
            heap(nclose)%hsiz = iz+1
            heap(nclose)%hsix = ix
         END IF
         IF(Tag(iz,ix-1) == 'F')THEN
            Tag(iz,ix-1) = 'C'
            nclose = nclose + 1
            heap(nclose)%hsiz = iz
            heap(nclose)%hsix = ix-1
         END IF
         IF(Tag(iz,ix+1) == 'F')THEN
            Tag(iz,ix+1) = 'C'
            nclose = nclose + 1
            heap(nclose)%hsiz = iz
            heap(nclose)%hsix = ix+1
         END IF
    END DO
    min_iclose = max_iclose + 1
    max_iclose = nclose
 END DO

 Tmap(1:nz,1:nx) = REAL(T(1:nz,1:nx),4)  !Output

 DEALLOCATE(T,Tag,heap,STAT=ierr)
 IF(ierr/=0) RETURN
 ierr=0
END SUBROUTINE Isotropic_Eikonal_HAFMM
```

Figure 8: HAFMM Fortran 90 code.