Stanford Exploration Project, Report 72, December 18, 1997, pages 1–142

Short Note

Stability of 2D finite difference traveltime algorithms

Alexander M. Popovici[1]

keywords: *algorithm, finite difference, traveltime*

INTRODUCTION

The original finite difference traveltime calculations implemented after Van Trier and Symes (1990) did not have an adaptive step determination. As a result, numerical instabilities appeared in unpredictable cases, for example the simple constant gradient velocity model in Figure 2. Such instability does not present a problem for research algorithms because numerical instabilities can be controlled by decreasing the grid spacing. However, for production programs this is impractical as users may not have complete control of each step of the finite difference calculation.

THE COURANT-FRIEDRICHS-LEWY CONDITION

The Courant-Friedrichs-Lewy (CFL) condition states that a necessary condition for the convergence of an explicit finite difference scheme is that the **domain of dependence** of the discrete problem includes the domain of dependence of the differential equation in the limit as the length of the finite difference steps goes to zero.

The CFL condition is easy to understand when examining the two cases in Figure 1. In the first case, the plane wave propagates with an angle α_1 given by the arctangent of the two vectors $\frac{|\vec{v}|}{|\vec{u}|}$. The explicit upwind finite difference stencil will use the traveltime values in points **A** and **B** to calculate the value in point **C**. When the plane wave propagates with an angle α_2 as in the second case, the information from points **A** and **B** is propagated to the right and above point **C**. In this case the point **C** is outside the domain of dependence and numerical instabilities can appear. To ensure the stability of the algorithm I use an adaptive step algorithm which will calculate the position of the point **C'** inside the domain of dependence.

[1]**email:** not available

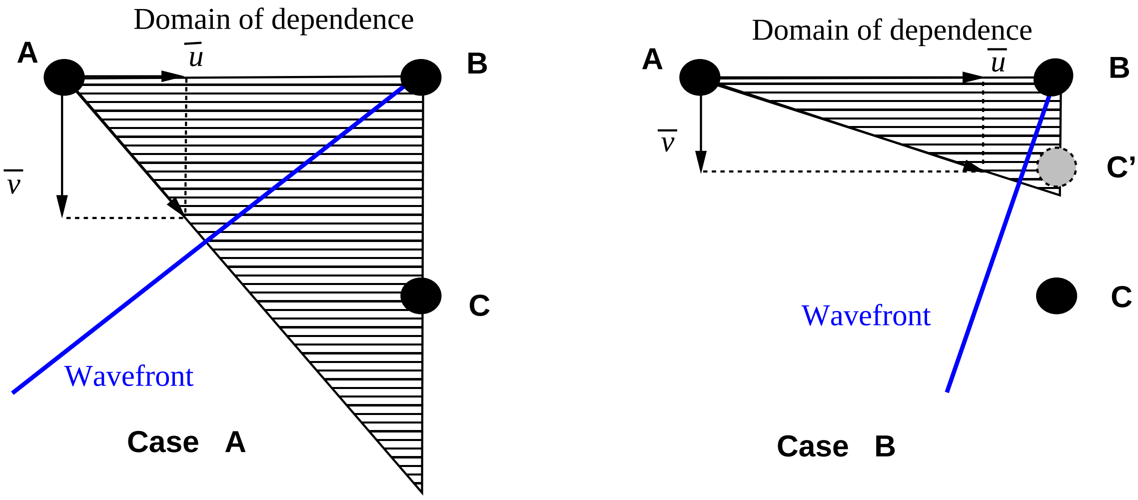

Figure 1: Two cases in plane wave propagation for the explicit finite difference traveltime algorithm. a) The domain of dependence includes the new stencil point. b) The domain of dependence doesn't include the stencil point C, requiring an adaptive step at C'. mihai1-stencil [NR]

THE ADAPTIVE STEP IMPLEMENTATION

The finite difference algorithm described by Van Trier and Symes (1991) is based on the eikonal equation in cartesian coordinates

$$u^2 + v^2 = s^2 \tag{1}$$

where

$$\begin{cases} u = \frac{\partial t(x,z)}{\partial x} \\ v = \frac{\partial t(x,z)}{\partial z} \end{cases}$$

and where $s(x, z)$ is the 2-dimensional slowness model and $t(x, z)$ is the travel time field. In cylindrical coordinates the eikonal equation becomes

$$\frac{u^2}{r^2} + v^2 = s^2 \tag{2}$$

where

$$\begin{cases} u = \frac{\partial t(r,\theta)}{\partial \theta} \\ v = \frac{\partial t(r,\theta)}{\partial r} \end{cases}$$

The traveltime is propagated one depth step at a time in cartesian coordinates, or one radial step at a time in cylindrical coordinates. The values of $u = \frac{\Delta t}{\Delta x}$ and $v = \frac{\Delta t}{\Delta z}$ are calculated on each computational front. The domain of dependence is given by the length ratio of the two vectors $\vec{u}$ and $\vec{v}$, as seen in Figure 1. The fixed grid algorithm is faster than the adaptive algorithm, sometimes by a factor of 10-20, depending on the velocity model considered. However numerical instabilities like the one in Figure 2 can not be predicted. The slower adaptive step algorithm will offer robust results.

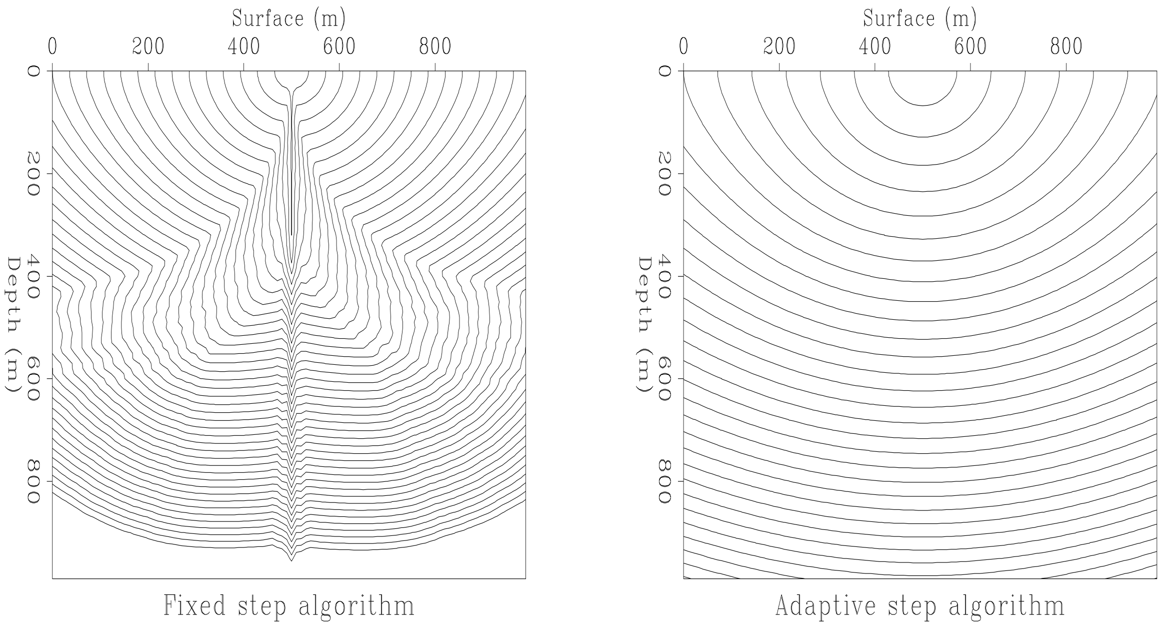

Figure 2: Two cases for a constant gradient velocity model. a) Numerical instabilities appear in the fixed grid algorithm. b) Given the same initial grid the adaptive step algorithm provides the correct solution. mihai1-TravelTimes [ER]

My initial adaptive step calculation in polar coordinates, based on the geometrical arguments presented in Figure 1, is listed below.

```
TINY=1.0e-3
.
.
.
ds=radius*dtheta
DRTINY=ds*TINY
thetaCFL=atan2(ddr,ds)
do it=0,ntheta+1
  thetamin=atan2(abs(v(it,ir1)),abs(u(it,ir1)/radius))
  thetamin=min(thetamin,thetaCFL)
enddo
ddr=min(ddr,ds*tan(thetamin))
ddr=max(ddr,DRTINY)
```

The calculations are done for each new radial computational front. The variable `ddr` is the initial radial increment which is modified adaptively. The arrays `v` and `u` store the values of the functions defined in equation (2). The minimum angle for the domain of dependence is calculated for each computational front and the minimum radial increment is used to advance the front.

A more elegant approach was suggested by Dave Hale. It monitors the same domain of dependence condition avoiding all the time consuming trigonometric relations. It also avoids the division by zero that can occur in either the case of $u = 0$ or $v = 0$. Essentially, the trick is to check for the ratio between v and $\frac{u}{r}$ in the form `v * r <= TINY * u` which increases the dynamic range of the variables compared.

```
TINY=1.0e-3
RTINY=1./TINY
CFL=0.98
.
.
.
cmax=TINY
ds=radius*dtheta
do it=0,ntheta+1
  rv=ABS(v(it,ir1)*radius)
  if(rv.lt.TINY*ABS(u(it,ir1))) then
    cmax=RTINY
  else
    cmax=max( cmax,abs(u(it,ir1)/rv) )
  endif
enddo
ddr=min(dr,CFL/cmax*ds)
```

CONCLUSIONS

Though an adaptive step finite difference traveltime algorithm is slower than a fixed grid algorithm, it has the important advantage of numerical stability. For programs where the user has little understanding of the underlying algorithm, an adaptive step approach is recommended. For research purposes, depending on the velocity model considered, a fixed grid algorithm may be preferred for the speed advantage.

ACKNOWLEDGMENTS

I would like to thank Dave Hale for suggesting an improved alternative to my adaptive code.

REFERENCES

Van Trier, J., and Symes, W., 1990, Upwind finite difference calculation of traveltimes: SEP–**65**, 41–58.

Van Trier, J., and Symes, W.W., 1991, Upwind finite-difference calculation of traveltimes: Geophysics, **56**, 812–821.