

Finite difference modeling of acoustic waves in Matlab

Carrie F. Youzwishen and Gary F. Margrave

ABSTRACT

A Matlab toolkit, called the AFD package, has been written to model waves using acoustic finite differences. It uses central finite difference schemes to approximate derivatives to the scalar wave equation. Both a second order or 5 point approximation, and a fourth order or 9 point approximation, to the Laplacian are included. The fourth order approximation is slower, but is more accurate, and results in a broader temporal bandwidth. The AFD package is also equipped with absorbing boundary conditions to suppress reflections from the edges of the grid. The toolkit is able to create velocity models, shot records, exploding reflector models, as well as snapshots and movies of the wavefield propagating in depth.

THEORY

The basis of the forward modeling algorithm is second order central difference approximations to the scalar wave equation. Recalling the scalar wave equation:

$$\frac{\partial^2 \phi(x, z, t)}{\partial t^2} = v^2(x, z) \nabla^2 \phi(x, z, t) \quad (1)$$

where the Laplacian, ∇^2 , is given by:

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial z^2} \quad (2)$$

The Laplacian operator can be approximated with central difference operators. The two approximations used within the AFD software are a second and a fourth order approximation. The approximations use five and nine points of the grid respectively. The second order approximation to the Laplacian operator is:

$$\nabla^2 \phi_j^n \approx \frac{\phi_j^{n+1} - 2\phi_j^n + \phi_j^{n-1}}{\Delta x^2} + \frac{\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n}{\Delta z^2} \quad (3)$$

where n is the x coordinate and j is the z coordinate of the grid, as illustrated in Figure 1.

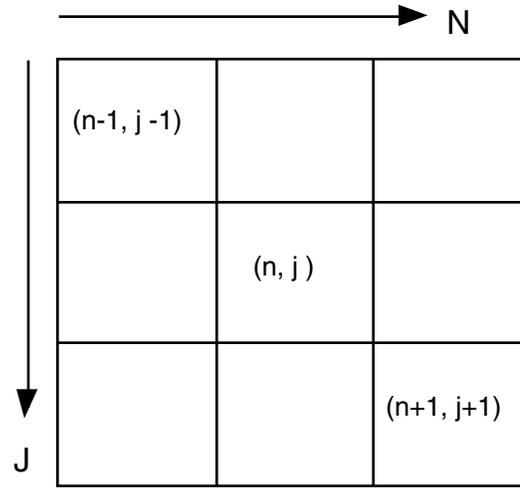


Fig. 1. The computational grid of the approximations to the Laplacian operator

The fourth order approximation is:

$$\nabla^2 \phi_j^n \approx \frac{1}{\Delta x^2} \left(\frac{-1}{12} \phi_j^{n+2} + \frac{16}{12} \phi_j^{n+1} - \frac{30}{12} \phi_j^n + \frac{16}{12} \phi_j^{n-1} - \frac{1}{12} \phi_j^{n-2} \right) + \frac{1}{\Delta z^2} \left(\frac{-1}{12} \phi_{j+2}^n + \frac{16}{12} \phi_{j+1}^n - \frac{30}{12} \phi_j^n + \frac{16}{12} \phi_{j-1}^n - \frac{1}{12} \phi_{j-2}^n \right) \quad (4)$$

In order to reduce the computing time, the AFD toolkit requires the grid spacing to be equal in the horizontal and vertical directions: $\Delta x = \Delta z$. As one would expect, the fourth order approximation is more accurate, but is slower. Its main advantage, though, is an increased bandwidth. Finally, each finite difference scheme has a stability condition (Lines, Slawinsky, and Bording, 1998). The stability conditions for the second order and fourth order approximations are respectively:

$$\frac{v_{\max} \Delta t}{\Delta x} \leq \sqrt{\frac{1}{2}} \quad (5)$$

$$\frac{v_{\max} \Delta t}{\Delta x} \leq \sqrt{\frac{3}{8}} \quad (6)$$

where the velocity, spatial sampling rate, and grid spacing are in consistent units.

The time derivative is calculated by a second order finite difference scheme:

$$\frac{\partial^2 \phi(t)}{\partial t^2} \approx \frac{\phi(t + \Delta t) - 2\phi(t) + \phi(t - \Delta t)}{\Delta t^2} \quad (7)$$

By substituting equation (7) and a Laplacian approximation into the scalar wave equation, one can solve for the wavefield at time $t + \Delta t$.

$$\phi_j^n(t + \Delta t) \approx (\Delta t^2 (v_j^{n,2}) \nabla^2 + 2) \phi_j^n(t) - \phi_j^n(t - \Delta t) \quad (8)$$

Equation 8 shows that the wavefield at time $t + \Delta t$ can be created by knowing the wavefield at time t and $t - \Delta t$. This process is called time-stepping and each wavefield is called a snapshot. Note that the Laplacian is applied to the wavefield at time t while the wavefield at $t - \Delta t$ is simply subtracted. To use equation 8 in a wavefield time-stepping scheme requires the prescription of two initial snapshots at time 0 and Δt . Usually, we simply prescribe these as identical fields of a simple source. It can be shown that this creates a source with equal amounts of upgoing and downgoing waves.

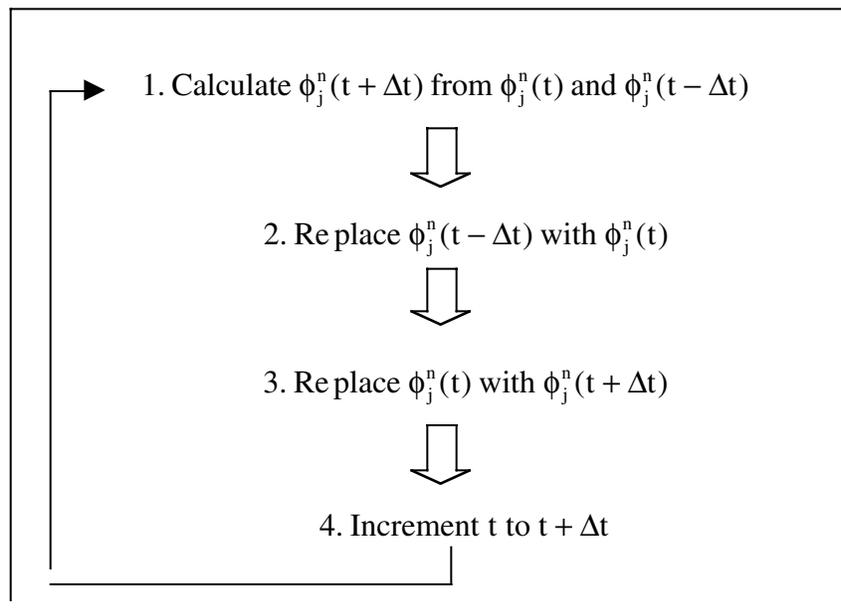


Fig. 2. The time-stepping finite difference algorithm.

Absorbing boundary conditions are included in order to reduce reflections from the grid edges. The absorbing boundary conditions are constructed from paraxial approximations of the wave equation (Clayton and Enquist, 1977). It is important to note that the corners of the absorbing boundaries are calculated with a less robust approximation, and are therefore less accurate. As well, the boundary conditions are applied as one 'layer' of the outside row and column of the grid for the second order finite difference scheme. However, because the fourth order scheme computes the

derivative from the surrounding two rows and columns on either side, it is necessary to apply the absorbing boundary conditions to two 'layers' of outside rows and columns. Therefore, the absorbing boundary conditions are less accurate for the fourth order approximation.

The absorbing boundary conditions for the corners are optimal for a wavefield travelling along a 45 degree diagonal into the corner. This limitation causes artifacts from the corner boundary if a wave travels directly into a corner at 0 or 90 degrees. This has been compensated for in some of the programming, but should be kept in mind. Finally, the absorbing boundary conditions will produce artifacts if a line or point source is positioned too close to the boundary. Because of this, most of the programs in the AFD toolkit do not have an absorbing boundary on the top of the model, so sources may be positioned at the surface. For all the above reasons, it is best to keep interesting features towards the middle of the matrices, so that no desired effects will be missed or cause artifacts.

MATLAB PROGRAMS

The AFD software package includes eight different functions to make it as versatile as possible. These functions all have a number of common variables required as input. The common input variables are:

- `xmax` – the maximum horizontal extent of the grid (in consistent units)
- `zmax` – the maximum depth of the grid (in consistent units)
- `delx` – the grid spacing for the horizontal and vertical directions (in consistent units)
- `delt` – the temporal sampling rate (in seconds)
- `velocity` – the geologic model: the velocity matrix in consistent units
 - must have a size of $\text{floor}(z_{\text{max}}/\text{delx})+1$ by $\text{floor}(x_{\text{max}}/\text{delx})+1$
- `field1` – snapshot of the wavefield at time = $t - \Delta t$
 - must be same size as velocity matrix
- `field2` – snapshot of the wavefield at time = t
 - must be same size as velocity matrix
- `laplacian` – specifies which approximation to the Laplacian operator you desire
 - '1' indicates the second order finite difference scheme (5 point)
 - '2' indicates the fourth order finite difference scheme (9 point)

- boundary – specifies the number of absorbing boundaries for the grid
 - ‘1’ indicates all four sides are to be absorbing (recommended for an exploding reflector model)
 - ‘2’ indicates both sides and the bottom to be absorbing, and the top to not be (recommended for shot records)

The common output variables are:

- z – the vector of the depth coordinates (consistent units)
- x – the vector of the horizontal coordinates (consistent units)
- t – the vector of the time coordinates (seconds)

The eight functions of the AFD toolkit are described below. The input variables are in the vector on the right hand side, and the output variables are in the vector on the left hand side.

1. `afd_snap`

`[snapshotn,z,x]=afd_snap(xmax,zmax,delt,delt,velocity,field1,field2,laplacian, boundary);`

`snapshotn` – the wavefield time-stepped one step forward

This function is the basis for all of the other functions. It time-steps the wavefield one step, and returns its snapshot. This is of minimal use when not embedded in other functions.

2. `afd_snapn`

`[snapshotn,z,x]=afd_snapn(xmax,zmax,delt,delt,velocity,field1,field2,toutput, laplacian,boundary);`

`toutput` – the desired output time in seconds

`snapshotn` – the wavefield time-stepped to the output time

The ‘`afd_snapn`’ function will time-step the wavefield from time zeros to the desired output time. The snapshot of the wavefield at the output time will then be returned. This function can be used for teaching, demonstrations, and troubleshooting.

3. `afd_moviesnapn`

`[M]=afd_moviesnapn(xmax,zmax,delt,delt,velocity,field1,field2,toutput, maxframes, laplacian,boundary);`

toutput – the output time in seconds

maxframes – the maximum number of frames in the movie (it is suggested that there should be no more than 40)

M – the movie of the propagating wavefield

This function is based on the same principles as 'afd_snapn', but instead of returning one snapshot of the wavefield at the output time, it will return a movie of snapshots at regular intervals throughout the propagation of the wavefield. In order to conserve time and memory, it is recommended that the number of frames or snapshots in the movie be no more than 40. However, one must keep in mind that a minimal number of frames means that the number of snapshots between start and finish will be fewer, and less information included. If the number of frames desired is greater than the number of iterations (there will be one snapshot every 'delt' time increment) the function will default to the maximum possible number of frames. To play the movie, simply type the command "movie(M)". For more information on movies type "help movie".

4. afd_seismo

[seis,filtseis,t,x,z]=afd_seismo(xmax,zmax,delx,delt,tmax,velocity,field1,field2,
nrec, xrec,zrec,filt,laplacian);

nrec – the number of receivers

xrec – a vector of the x-coordinates of the receivers in consistent units

zrec – a vector of the z-coordinated of the receivers in consistent units
z=0 will position receivers on the surface

filt – a four component vector specifying the gaussian filter to filter the data with
filt = [fmin wmin fmax wmax] where
fmin - is the 3 dB down point of the filter on the low end
wmin – is the gaussian width of the filter on the low end
fmax – is the 3 dB down point of the filter on the high end
wmax – is the gaussian width of the filter on the high end

seis – the unfiltered shot record

filtseis – the filtered shot record

This function will return a shot record. The number of receivers, their x-coordinates, and their z-coordinates are input in consistent units as 'nrec', 'xrec', and 'zrec' respectively. There is no limit to the number or position of the receivers, so this function can be used to create VSPs as well. The boundary variable defaults to absorbing boundaries on both sides and the bottom, so sources may be placed on the surface (top boundary).

The source array is specified within field1 and field2. These input variables are matrices of zeros the same size as the input velocity matrix. The position and strength of sources are indicated by non-zeros values within the matrices. As mentioned within the theory section, the two fields are usually set to be equal in order to create a source with equal amounts of upgoing and downgoing waves. See `afd_source` for more information on how to create these matrices.

Each time the wavefield is stepped forward by one time increment (using `afd_snap`), the function grabs the seismic response at each receiver location. Thus, only the seismic shot record can be filtered, as the wavefield is in units of depth, not time. Once the shot record has been created, filtering takes place using the gaussian function specified in 'filt'. The program returns the unfiltered ('seis') and filtered data ('filtseis'). There will be one trace per receiver in the output shot records.

5. `afd_exreflector`

```
[seismogram,seis,t,x]=afd_exreflector(xmax,zmax,delt,delt,tmax,velocity,nrec, xrec, zrec,wavelet,tw,nzero,laplacian,clipn);
```

`nrec`, `xrec`, and `zrec` - are described above

`wavelet` – the wavelet with which to convolve the output seismogram
- the temporal sampling rate of the wavelet MUST be the same as that of the seismogram (must =delt)

`tw` – the time vector of the wavelet

`nzero` – the sample number of the wavelet that occurs at zero time value
- this allows the user to create a casual or non-casual wavelet as desired

`clipn` – the number of bins with which to clip the reflectivity matrix to prevent instability in the corners

`seis` – the unconvolved exploding reflector seismogram

`seismogram` – the convolved exploding reflector seismogram

This function will return the convolved seismogram of an exploding reflector model. The program will automatically divide the velocity model by 2 to compensate for one way travel time, and compute its reflectivity. The reflectivity matrix becomes the input wavefield, and is time-stepped forward. Like the `afd_seismo` function, the seismogram is created by grabbing seismic responses at each receiver location for each iteration. The seismogram is then convolved with the input wavelet. It is recommended that the wavelet be created using other CREWES functions such as 'ricker' or 'ormsby' because the time vector is created at the same time. As well, it has been found that an ormsby wavelet is the most appropriate because of the limited bandwidth associated with finite difference models.

The boundary condition will automatically default to absorbing boundaries on all sides. As aforementioned, one of the limitations of the absorbing boundary conditions occurs when a line source is parallel to a boundary, or a wavefield propagates into a corner at any angle other than approximately 45 degrees diagonal to the corner. Because an exploding reflector model frequently violates both of these conditions, a 'clipn' variable has been added. This variable effectively clips the reflectivity data on the outside layer of rows and columns for the specified number of bins. The appropriate number of bins will span not less than 100 meters. Due to time constraints, this number has been roughly estimated, and the problem of artifacts in the corners will be illustrated in the next section. Keeping this in mind, it is best to keep all interesting phenomena to the center of the grid. As well, if the edges are suitably uniform, one always has the option of setting 'clipn' to zero.

6. `afd_reflect`

```
[velocity,field1]=afd_reflect(xmax,zmax,dex,velocity,clipn);
```

velocity – the velocity matrix divided by 2 to compensate for 1 way travel time

field1 – the reflectivity matrix of the velocity, which will be used as the initial wavefield in the exploding reflector model

The function 'afd_reflect' is meant for exploding reflector models. It will divide the velocity matrix by 2 to create velocities appropriate for one way travel time, and will calculate the reflectivity of the velocity model. This function is already embedded in `afd_exreflector`, but is necessary to create the initial wavefield ('field1') to be able to use other functions within this package. In order to prescribe equal amounts of upgoing and downgoing waves, field1 and field2 are set to be identical. This allows the use of 'afd_snapn', and 'afd_moviesnapn'.

7. `afd_source`

```
[wavefield,z,x]=afd_source(xmax,zmax,dex,nsources,xsources,zsources,sz,default,smatrix);
```

nsources – the number of sources, or source arrays

xsources – a vector of the x-coordinates of the sources in consistent units

zsources – a vector of the z-coordinates of the sources in consistent units

sz – the vector of horizontal span of the sources is consistent units (a span of less than the grid spacing is considered a point source)

- if a custom source matrix is entered, the size of this matrix is entered bin numbers

- a custom source matrix must be a square matrix where the size in bins is odd, so that the matrix may be centered

default – indicates whether a custom source matrix is entered, or whether the source is to be built within the program

- '0' indicates that a custom source matrix will be entered
- '1' indicates that the source is to be built within the program

smatrix – the custom source matrix

- if the 'default' variable is set to '1', set smatrix =0
- the source matrix must be a square matrix where the size in bins is an odd number, so that the matrix may be centered

wavefield – the initial wavefield as specified by the source array

This function is to assist in the creation of the initial wavefields field1 and field 2. Within these fields, a source will be represented by a '1', and all other positions will be zeros. There are two options when using this program: one may enter a custom source matrix, or have one built within the program. If you choose not to enter a source matrix, the positions and horizontal extent of the sources are specified with 'xsource', 'zsource', and 'sz'. All variables are entered in consistent units. Finally, 'default' is set to '1', and 'smatrix' to '0'. The function will center the sources at the 'xsource' and 'zsource' locations.

If one wants to enter a custom source, first build the source within a square matrix where the size is an odd number of bins. The matrix will be centered on the position 'xsource' and 'zsource', ensuring that the source matrix is always contained within the grid. If the 'zsource' value is set to '0' (the surface), the program will adjust the centering algorithm so that the source matrix does not protrude above the surface. The positioning variables are entered in consistent units, but the size of the source matrix, 'sz' is entered in bins. Finally, 'default' is set to '0', and 'smatrix' is equal to the source matrix. In this way, the source matrix can be as complicated as desired.

At this time, because the AFD software package is limited to shot records, it is assumed that only one source or source matrix will be needed. However, the program is equipped to deal with more than one source or source matrix. The variables 'xsource', 'zsource', 'sz', and 'smatrix' will then become vectors.

8. afd_vmodel

[velfinal]=afd_vmodel(xmax,zmax,deltx,velocity,vpoly,ypoly,conversion);

vpoly – the velocity within the polygon in consistent units

xpoly – a vector of the x-coordinates of the polygon in consistent units, or in bin numbers

ypoly – a vector of the y-coordinates of the polygon in consistent units, or in bin numbers

conversion – indicates whether ‘xpoly’ and ‘ypoly’ need be converted to bin numbers from consistent units

- ‘1’ turns the conversion on

- ‘0’ turns the conversion off

velfinal – the initial velocity matrix with the polygon of velocity ‘vpoly’ superimposed upon it

The ‘afd_vmodel’ function is meant to assist in the creation of velocity models. The function will superimpose a polygon on a background velocity matrix. The background velocity may be homogeneous, layered, or as complicated as desired. The polygon is created in the order of the coordinates entered, so be careful when assigning ‘xpoly’ and ‘ypoly’. For more complicated shapes, the ‘ginput’ function is helpful. It will return the coordinates of the points as you click on them with the mouse. Because these coordinates will already be in bin numbers (from the initial velocity matrix), the ‘conversion’ variable should be set to ‘0’ to turn it off. The algorithm works by checking to see if each point of the grid is within the polygon, so it is slow. As well, it is limited because only one polygon may be built at a time. It is recommended that the matrices created with this program be saved, so that you will not have to regenerate them.

MODELLING WITH THE AFD PACKAGE

To test and demonstrate the AFD software package, two different velocity models were used. The first is the Marmousi velocity model (Versteeg and Grau, 1991). The second, a thrust belt model, was created using the ‘ginput’ and the afd_vmodel functions. The Marmousi velocity model had an original bin spacing of 12 meters vertically, and 24 meters horizontally. In order to use the velocity model in our software, we made the assumption that the bins are 12 meters square, and essentially shortened the model.

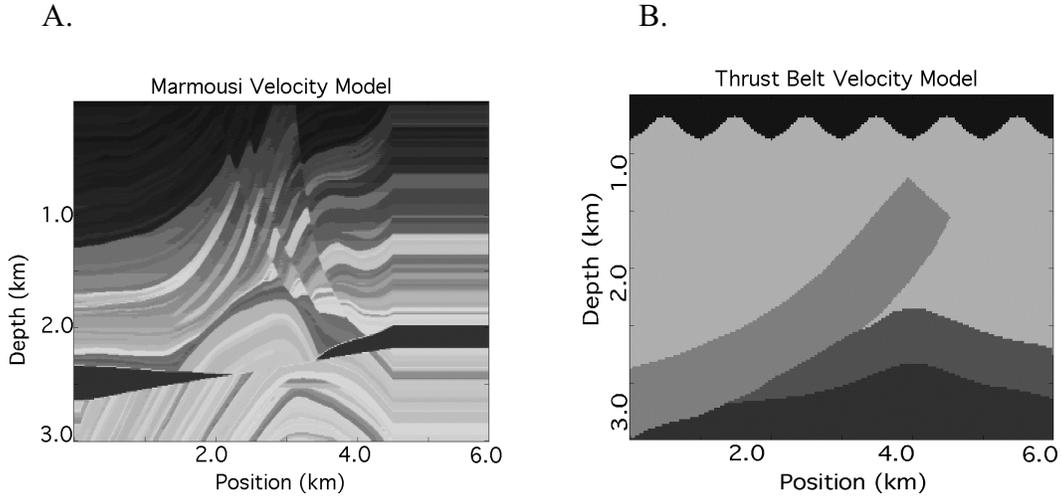


Fig. 2. (a) The Marmousi velocity model and (b) the thrust belt velocity model.

Four different variations of the velocity models were used in generating models.

Table 1: The parameters of the variations of the four different models.

Parameter Set	Velocity Model	Bin Spacing (m)	Temporal Sampling Rate (s)
1	Marmousi	12	0.002
2	Marmousi	6	0.001
3	Marmousi	4	0.0005
4	Thrust Belt	20	0.002

The first parameter set is used to illustrate the `afd_snapn` and the `afd_seismo` functions. The velocity model with the source position, and its models are shown are illustrated in Figure 3.

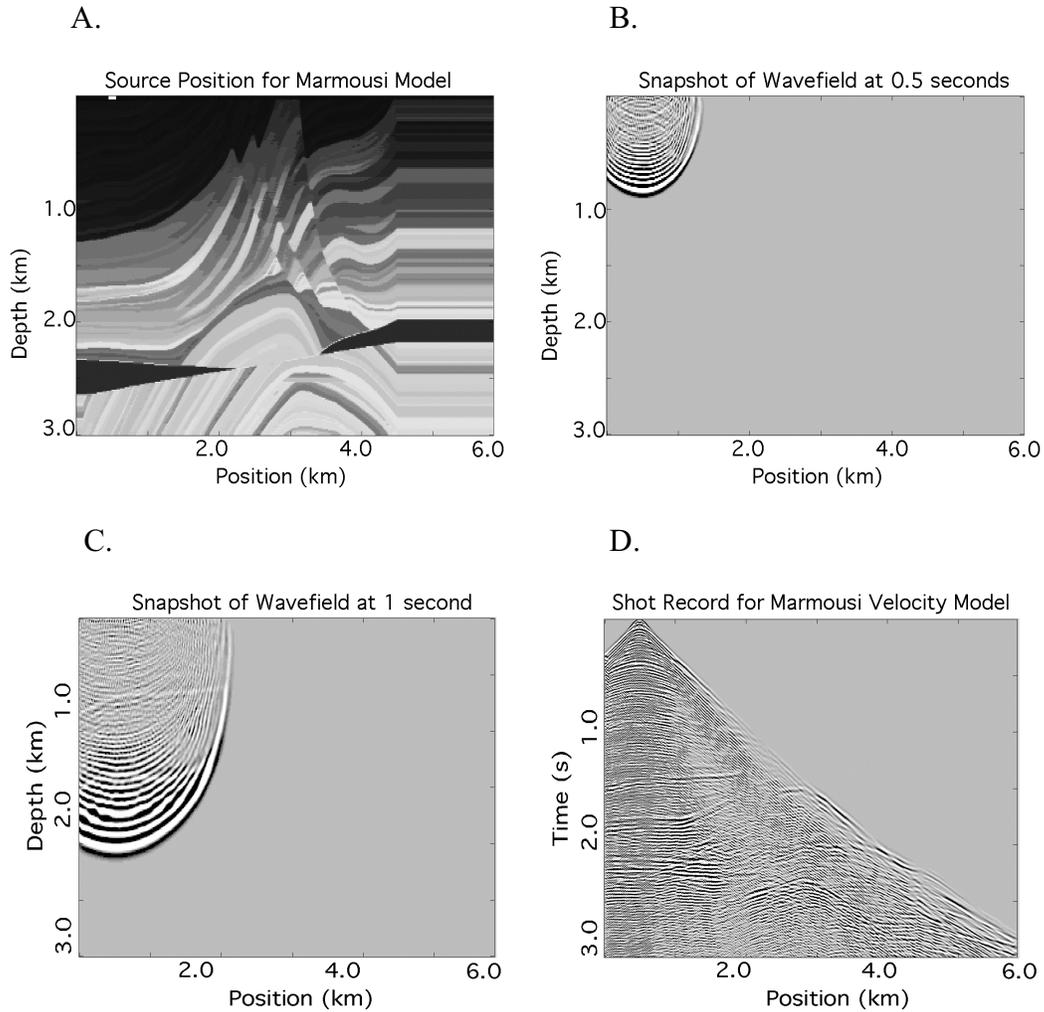


Fig. 3. The velocity model and forward models for parameter set 1: (a) The velocity model where the source is represented by the white block (b) A snapshot of the wavefield at 0.5 seconds (c) A snapshot of the wavefield at 1 second (d) The shot record of the velocity model.

The `afd_seismo` and `afd_exreflector` functions allow the receivers to be placed at any location. This enables us to model VSPs as well as standard shot records. VSP models for both parameter sets 1 and 4 are illustrated in Figures 4 and 5.

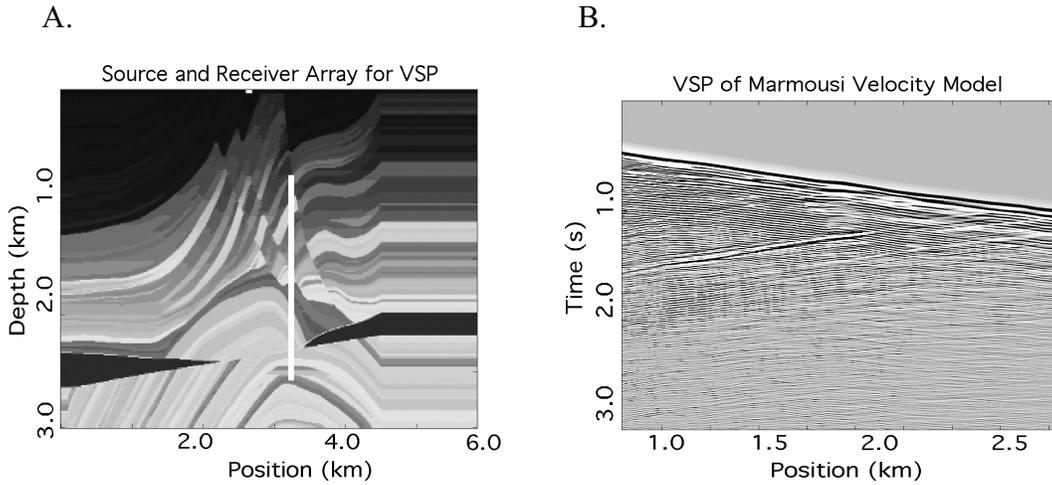


Fig. 4. (a) The shot and borehole position for the Marmousi VSP. (b) The shot record for the VSP.

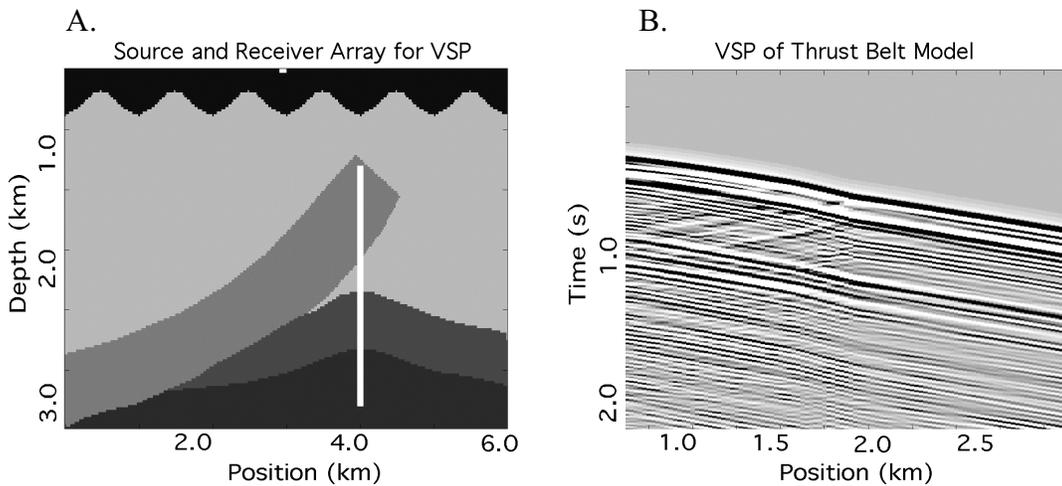


Fig. 5. (a) The shot and borehole position for the thrust belt VSP. (b) The shot record for the VSP.

As one can see, it is almost impossible to distinguish events on the Marmousi VSP because of the complexity of the velocity model. The thrust belt model is simpler, and gives a clearer record of individual events.

The last part of this section deals with the exploding reflector model. The three different parameter sets of the Marmousi velocity model are used to determine the effect that temporal and spatial sampling have on bandwidth. Parameter sets 2 and 3 indicate that the grid spacing has been changed from the original model. This was done using the 'interp2' function, and changed only the number of bins, not the horizontal or vertical extent of the velocity model.

The most important variable in the exploding reflector model is the ‘clipn’ variable, or the number of bins from the edges of the reflectivity model that have been clipped to prevent artifacts. Artifacts are produced because the absorbing boundary conditions are optimal for wavefields travelling at a diagonal of 45 degrees into the corners. When the wavefields are closer to 0 or 90 degrees, the corners become unstable. To prevent this, the ‘clipn’ variable must be set for a number of bins that will span at least 100 meters.

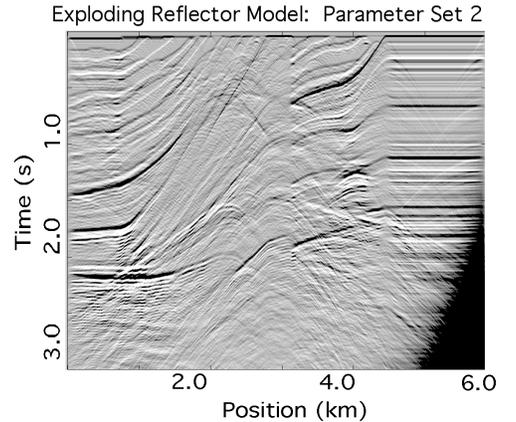


Fig. 6. An exploding reflector model for parameter set 2 with a ‘clipn’ spanning only 30 meters.

Due to time constraints, the optimum value for the ‘clipn’ variable has not been found, and evidence of artifacts will be seen in the following examples!

All of the models included within this paper have been computed with the fourth order, or nine point approximation to the Laplacian. A comparison of the second order and fourth order approximations is shown in Figure 7 and 8.

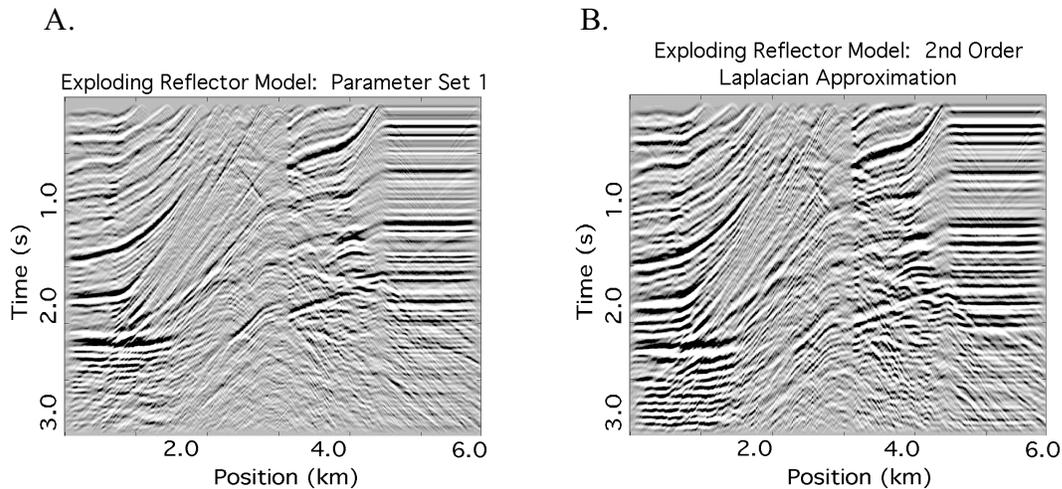


Fig. 7. (a) The exploding reflector model for parameter set 1 with a fourth order approximation to the Laplacian operator. (b) The exploding reflector model for parameter set1 with a second order approximation to the Laplacian operator.



Fig. 8. A comparison of the dB spectrum of the second and fourth order approximations to the Laplacian operator. One can see that the fourth order approximation (solid line) has an increased bandwidth.

The fourth order Laplacian approximation does increase the bandwidth, but not significantly. Next we will compare the effect temporal and spatial sampling rates have on bandwidth using the exploding reflector model. Figure 9 demonstrates the effect temporal sampling rate has on bandwidth by comparing two different sampling rates for parameter set 1. It becomes apparent that though increasing the time sampling rate does increase bandwidth, it also does not have a dramatic effect.

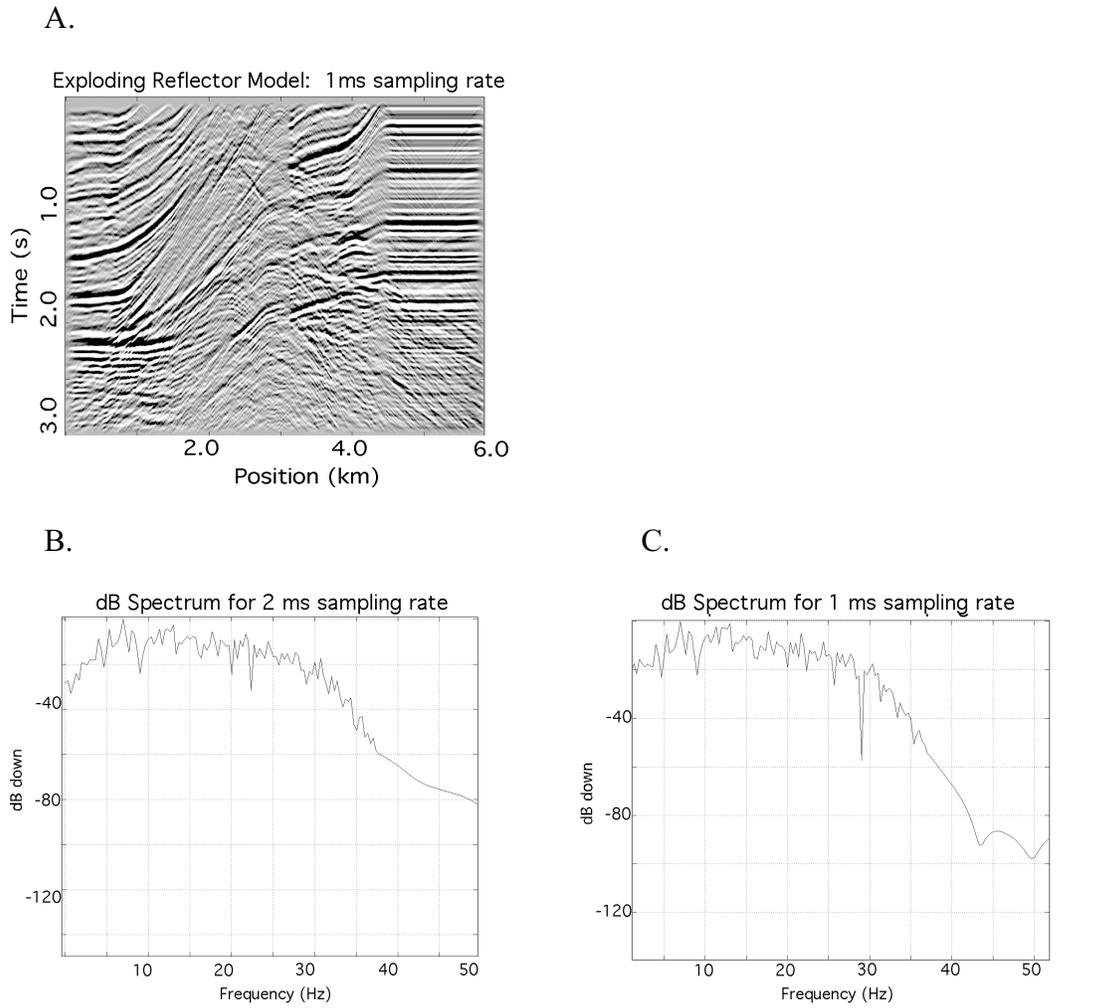


Fig. 9. (a) The exploding reflector model for parameter set 1, a fourth order approximation to the Laplacian operator, at a 1 ms sampling rate. (b) The dB spectrum for parameter set 1 at a 2 ms sampling rate. (c) The dB spectrum for parameter set 1 at a 1 ms sampling rate.

The next figures demonstrate the effect that spatial sampling rate has on bandwidth. The first models illustrate the effect of parameter set 2 with the grid spacing halved, and a time sampling rate of 1 millisecond. The second model illustrates the effect of parameter set 3 which has a grid spacing of 1/3 the original, and a time sampling rate of 0.5 milliseconds. The effects are dramatic, and it becomes apparent that spatial sampling rate has a definite influence on bandwidth.

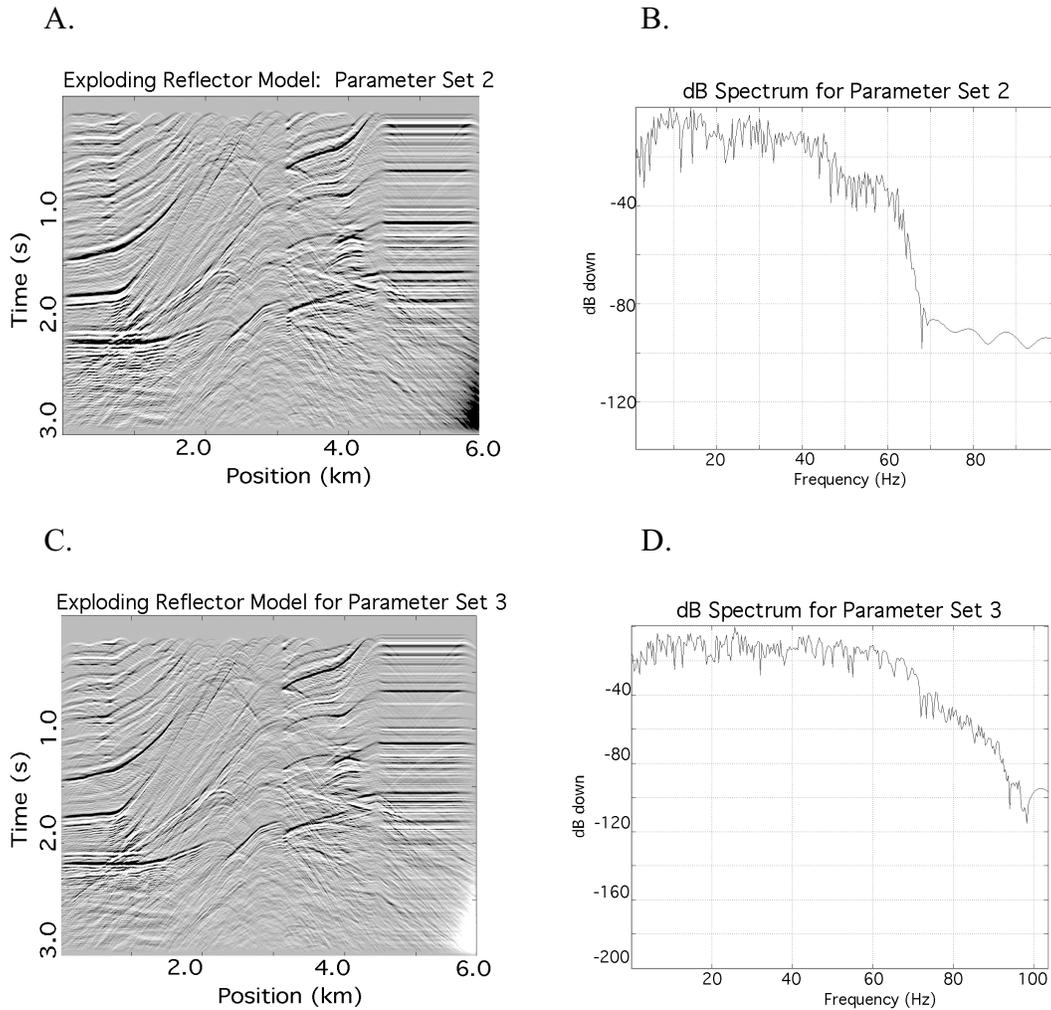


Fig. 10. (a) The exploding reflector model for parameter set 2. (b) The dB spectrum for parameter set 2. (c) The exploding reflector model for parameter set 3. (d) The dB spectrum for parameter set 3.

Figure 10 illustrates the problem of artifacts within the lower right hand corner. The polarity of these artifacts seems to be determined by the polarity of the offending wavefront. Figure 10 (a) has a ‘clipn’ span of 120 meters, and (c) has a ‘clipn’ span of 140 meters. Obviously these are not large enough. It is of interest, however, to note that parameter set 1 has a ‘clipn’ span of 60 meters, and this seems sufficient. It may be that the presence of artifacts is dependent upon not only the distance of the wavefield from the boundary, but that time and spatial sampling have an effect as well.

It is apparent that increased spatial and temporal sampling rate, as well as a more robust approximation to the Laplacian operator, increases bandwidth. This must be balanced with time constraints. Table 2 summarizes the run times for the different exploding reflector models.

Table 2: The summary of the different parameter sets. With the exception of parameter set 1(a) all use the fourth order approximation to the Laplacian operator.

Parameter Set	Velocity Model	Bin Spacing (m)	Temporal Sampling Rate (s)	Run Time
1 (a)	Marmousi **2 nd order Laplacian approximation	12	0.002	15 minutes
1 (b)	Marmousi	12	0.002	35 minutes
1 (c)	Marmousi	12	0.001	50 minutes
2	Marmousi	6	0.001	4 hours
3	Marmousi	4	0.0005	8 hours

DISCUSSION AND FUTURE WORK

The AFD toolkit is a flexible as a modeling tool. Unfortunately, because it uses the finite difference algorithm, it tends to have limited bandwidth. The only way to circumvent this is to increase temporal and spatial sampling, and correspondingly increase the run time.

The absorbing boundary conditions work well for limited conditions. When a line source is parallel and close to a boundary, or a wave travels at 0 or 90 degrees into a corner, the boundary generates artifacts. These problems are especially prevalent in an exploding reflector model. In an effort to circumvent these problems, the ‘clipn’ variable has been introduced. This variable clips the edges of the reflectivity model a specified number of bins. However, the optimum value of the ‘clipn’ variable has not been found. It is possible that this variable is dependent upon temporal and sampling rates.

Overall, the AFD software package has shown to be a useful and flexible tool. Our future plans include: research on the absorbing boundary condition problems, variations on functions to allow users to create topography in their models, and further testing to determine the full capabilities of modeling using acoustic finite difference.

ACKNOWLEDGEMENTS

We would like to thank the CREWES sponsors for their continued support and feedback.

REFERENCES

- Clayton, R. and Enquist, B., 1977, Absorbing boundary conditions for acoustic and elastic wave equations: *Bull. Seis. Soc. Am.*, **67**, 1529-1540.
- Dablain, M. A., 1985, The application of high-order differencing to the scalar wave equation: *Geophysics*, **51**, 54-66.
- Lines, L. R., Slawinski, R., and Bording, R. P., 1998, A recipe for stability analysis of finite-difference wave equation computations: 1998 Annual Research Report of the CREWES Project.
- Versteeg, R. and Grau, G., Editors., 1991, *The Marmousi Experience: Proc. Of 1990 EAEG Workshop on practical aspects of seismic data inversion.*