3-D grid with GOCAD

Arnaud Berlioux [1]

ABSTRACT

The aim of this article is two-fold. First, a small program has been developed to build a 3-D grid and fill it from geological or geophysical models using the GOCAD library of functions. This 3-D grid program has been tested in many cases and has produced good results even with very complex structural models. Second, a proposal is made to elaborate a new implementation of the `GRID3_t` object of GOCAD in a manner more convenient for geophysical and/or geological purposes.

INTRODUCTION

In many scientific endeavors of a computational nature, the representation or the simulation of natural phenomena requires the use of grids. As a matter of fact, such processes are often described by complex sets of equations like partial differential equations that rarely have simple analytical solutions when it becomes necessary to mimic reality. In particular, in geophysics, 2-D or 3-D grids are commonly used for such purposes as finite-difference schemes applied to wave-propagation analysis or traveltime computation.

GOCAD is a software for handling surfaces in 3-D and therefore allows the representation and the definition of sophisticated 3-D models. The definition of 3-D grids is possible in GOCAD via the `GRID3_t` object (Mallet, 1993), but once the empty grid has been allocated, it is necessary to assign to each node of the grid a corresponding value of the variables extracted from the model.

In this article I try to verify the feasibility of designing such a grid especially for complex models, and then I propose a new implementation of the `GRID3_t` object in a manner more convenient for geophysical and geological purposes.

[1] **email:** arnaud@sep.stanford.edu

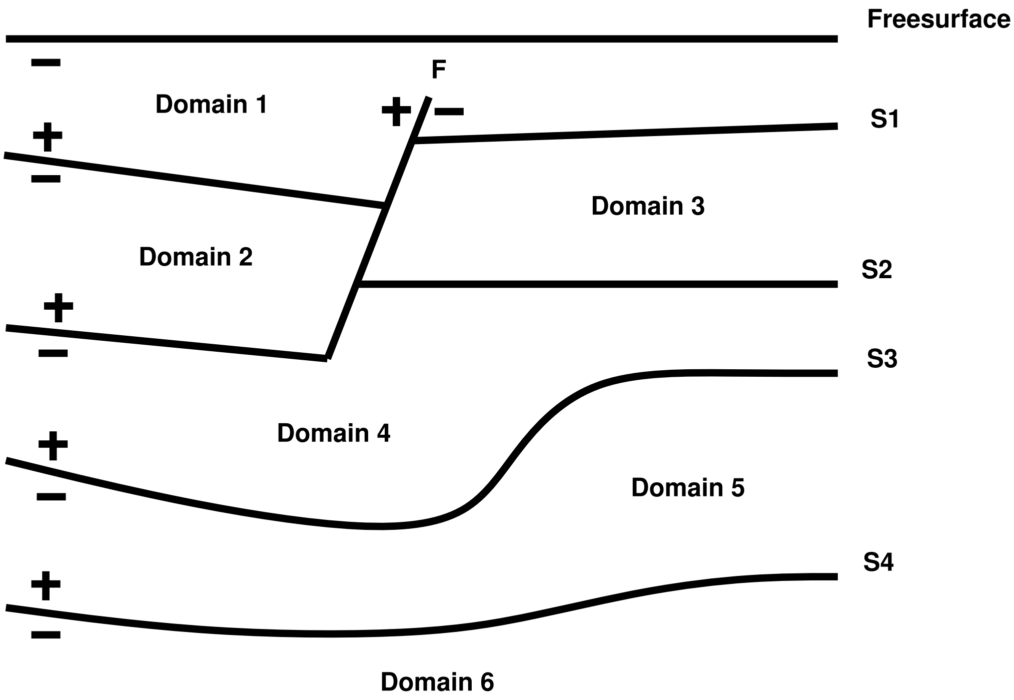

Figure 1: A 2-D slice of an example of a GOCAD model. model [NR]

```
% real V  ; /* Velocity */
% real d0 ; /* Density  */

% function density()   ;
% function K_compute() ;


Domain_3 : S1-, F-, S2+ ;       /* Portlandian */
{
 V  = 3100.0 ;
 d0 = 2.75   ;
}


Domain_4 : S2-, F-, S3+ ;       /* Kimeridgian : clay */
{
 V  = 2700.0 ;

 density( input1, output1 ) ;

 real porosity = 0.37 ;
}


Domain_5 : S3-, S4+ ;           /* Oxfordian : salt */
{
 V  = 4200.0 ;

 real K0_x = 3300 ; /* Porosity */
 real K0_y =  456 ;

 K_compute( point, result ) ;
}
```

HOW TO BUILD A 3-D GRID WITH GOCAD

Many problems in geophysics and other sciences in general are solved using grids extracted from a model. A 3-D grid is a volume composed of small cells, usually rectangular parallelepipedic, at the nodes of which some values of parameters are attached.

Notion of GOCAD model

In GOCAD (Mallet, 1993), a model is a set of domains bounded by interfaces (triangulated surfaces) and in which some properties are defined such as velocities, porosities, permeabilities, and densities. These properties can be constant and attached to a whole domain or variable and computed at each point inside the domain.

Figure 1 shows a 2-D slice of such a model composed of six domains delimited by their boundaries. Each boundary is characterized by an interface and a side (+ or −). On the same page lies an excerpt of an example of a file used in GOCAD to describe the partition of the 3-D space associated with the set of interfaces. For example, the domain named `Domain_4` is defined by the boundary `S2-`, `F-`, and `S3+`, and the properties available in this domain are a constant velocity (2700.0 m/s) and a function named `density( input1, output1)` giving the density of the layer (ouput1) with respect to some parameters (input1). In the GOCAD library, there are functions to retrieve the properties attached to a domain from the location of a point in the 3-D space.

Notion of a 3-D grid inside GOCAD

In the GOCAD software, the class of object GRID3 is aimed at building 3-D grids. As is shown in Figure 2, the 3-D grid is defined by its origin, three vectors $\vec{U}$, $\vec{V}$, $\vec{W}$ giving the three directions of the axes of the grid, and three integer values nu, nv, nw corresponding to the sampling on each of the axes.

When a new grid is created, the value at each node of the grid is set to an initial value. The user has to update the values inside the grid with respect to a given model.

How to fill a GRID3_t object from a given model

The small program I designed fills a 3-D grid with velocity values extracted from a GOCAD model file. The main idea is to make a loop over all the nodes of the grid located on the top row. From each point so defined, a shoot is performed vertically inside the model to locate the intersection points with all the interfaces of the model. Then each node of the column is filled with the value of the velocity attached to the domain corresponding to the location of the node.

The executable produced from this program receives a parameters file as input and generates an SEP output file. The syntax to run this program is

```
grid par=input.par > output.H
```

The parameters file is written in a kind of SEP format. The script text below is an example of such a file:

```
project="ProjDiap.PJ"
model="Diapir.pa"

o1=7000.0
o2=3000.0
o3=0.0

d1=100
d2=100
d3=25

n1=150
n2=110
n3=200

vout=50
nout=60
```

The parameters file contains basically

- the name of a GOCAD project, containing the geometry and the topology of each surface composing the model,
- the name of a GOCAD model, where the domains are defined with the value of the velocity attached to each domain, and
- the parameters defining the 3-D grid: origin, step, and sampling on each axis (the suffixes 1,2,3 in this file replace respectively the suffixes u,v,w of the `GRID3_t` object).

The last two parameters describe the format of the output. Martin Karrenbach and I have chosen to view the 3-D grid slice by slice. `vout` is the value on the v axis corresponding to the first slice to be displayed. `vout` can vary from 0 to $n2$. `nout` is the number of slice that will be shown. `nout` can vary from 1 to $n2$. If `nout` $= 1$, the output will be the slice corresponding to the value `v` $=$ `vout`; if `nout` > 1, `nout` slices of the grid will be displayed one after the other, as a movie, the first one being the frame numbered `vout`. The rule `nout` $+$ `vout` $\leq$ `n2` should always be observed.

An example of the results produced by the algorithm implemented is given in the next section. We have chosen this output arbitrarily. It is obvious that there exist many other possibilities for viewing the whole grid or part of it.

AN EXAMPLE OF A COMPLEX GEOLOGICAL MODEL

Figure 3 represents a model of a salt dome that pierces the two interfaces situated above the salt layer.

A model with 200 gridpoints in depth and 464 gridpoints in horizontal distance is written out of GOCAD and fed into a finite-difference modeling program (Karrenbach, 1992). Figure 4 shows the subsurface model. It consists of homogeneous

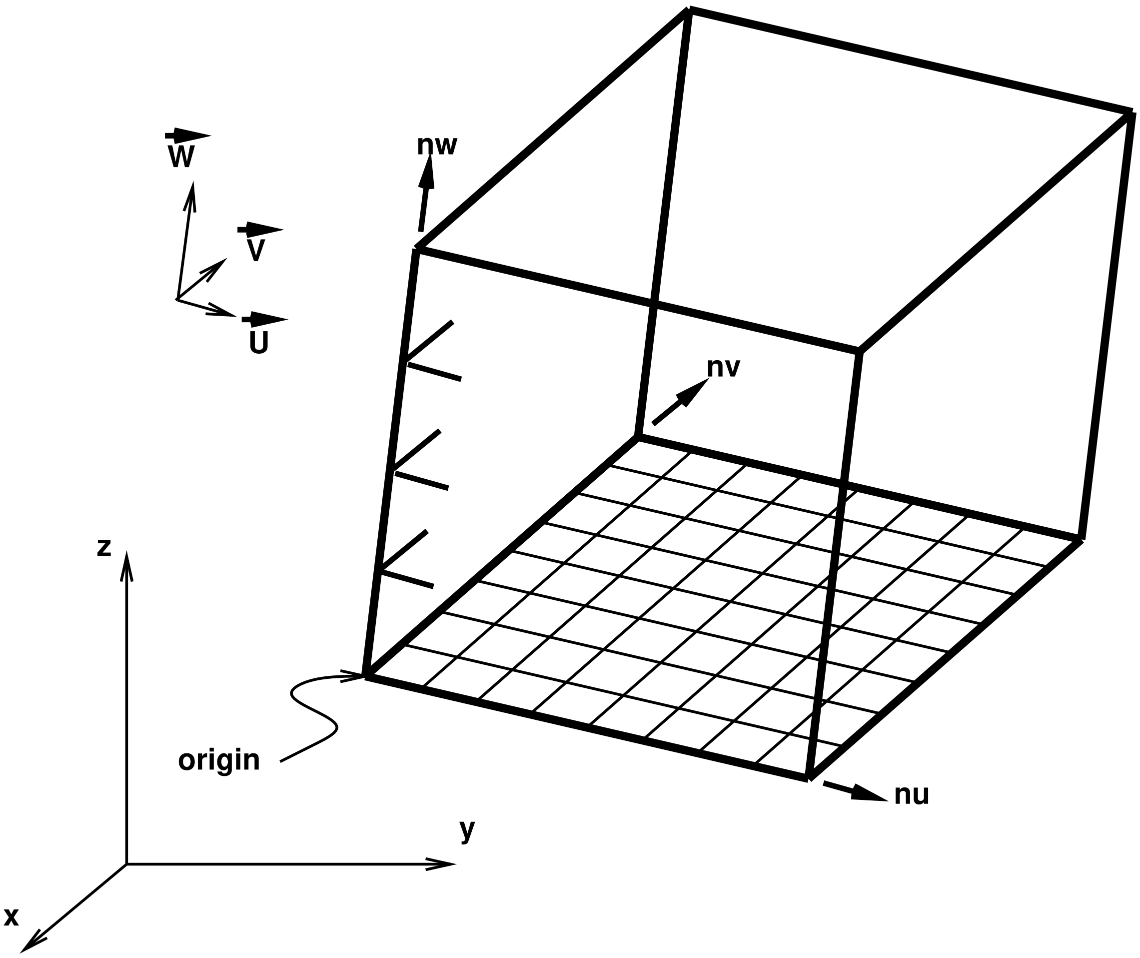

Figure 2: Notion of a GRID3 object in GOCAD. A 3-D grid in GOCAD is defined by its origin, the number of samples on each axis (nu, nv, nw), and three vectors $\vec{U}$, $\vec{V}$, $\vec{W}$ giving the directions of each of the axes. The length of these three vectors gives the step on each axis. grid [NR]

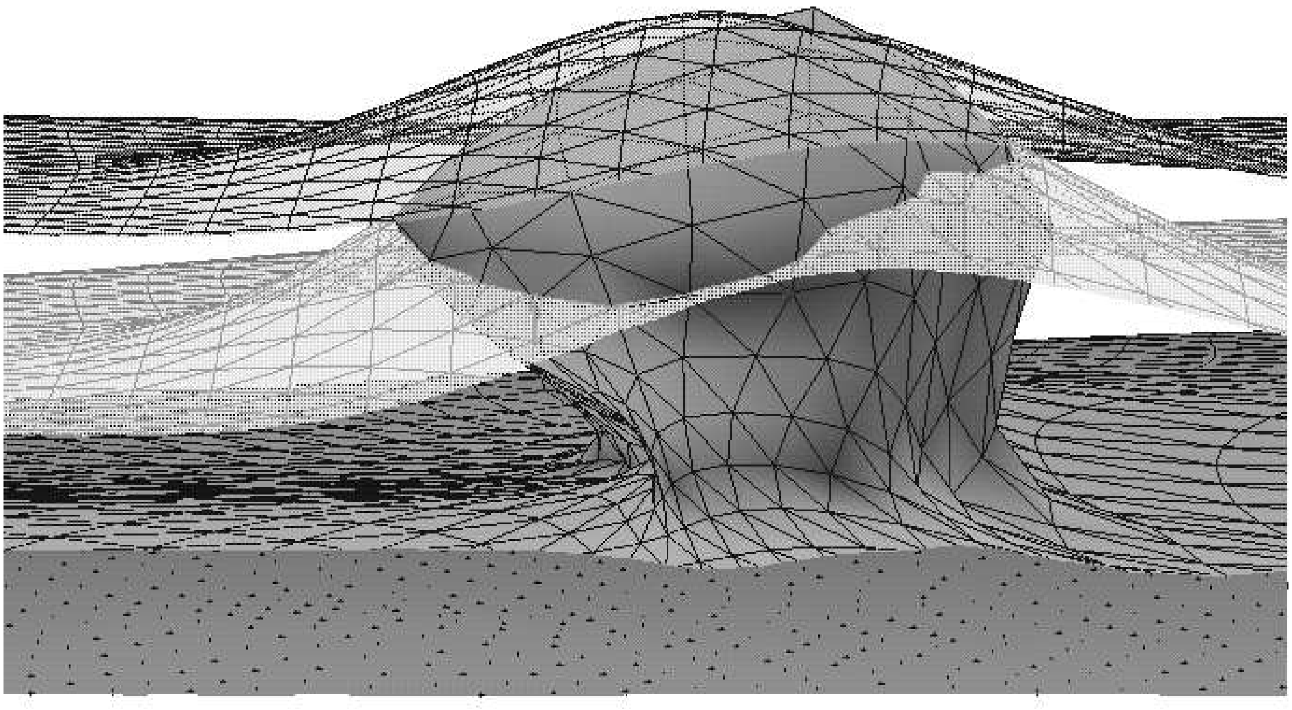

Figure 3: Diapir model used to build the 3-D grid. diapcut [NR]

sedimentary layers that are uplifted and intruded by a salt body, a classic diapir. We use a pressure source in an acoustic medium and collect the z-component of the displacement on the free surface of the model. The wavelet is a derivative of a Gaussian function with a fundamental frequency of 40Hz. Figure 5 shows z-component seismograms collected left, on top and right of the diapir.

As explained previously, the gridding quality depends partly on the smoothness of the model interfaces. We can see some diffractions caused by stair-stepping of the first layer interface in Figure 4. During the wave propagation, the corners of those steps will act as diffraction points. In Figure 6, the snapshot of the z-component of the wave field, those outward propagating diffraction circles are clearly visible. On the seismograms, those diffraction points manifest themselves as hyperbolae suspended from the main reflection hyperbola. To watch the movie of those snapshots press the button at the end of the figure caption in Figure 6.

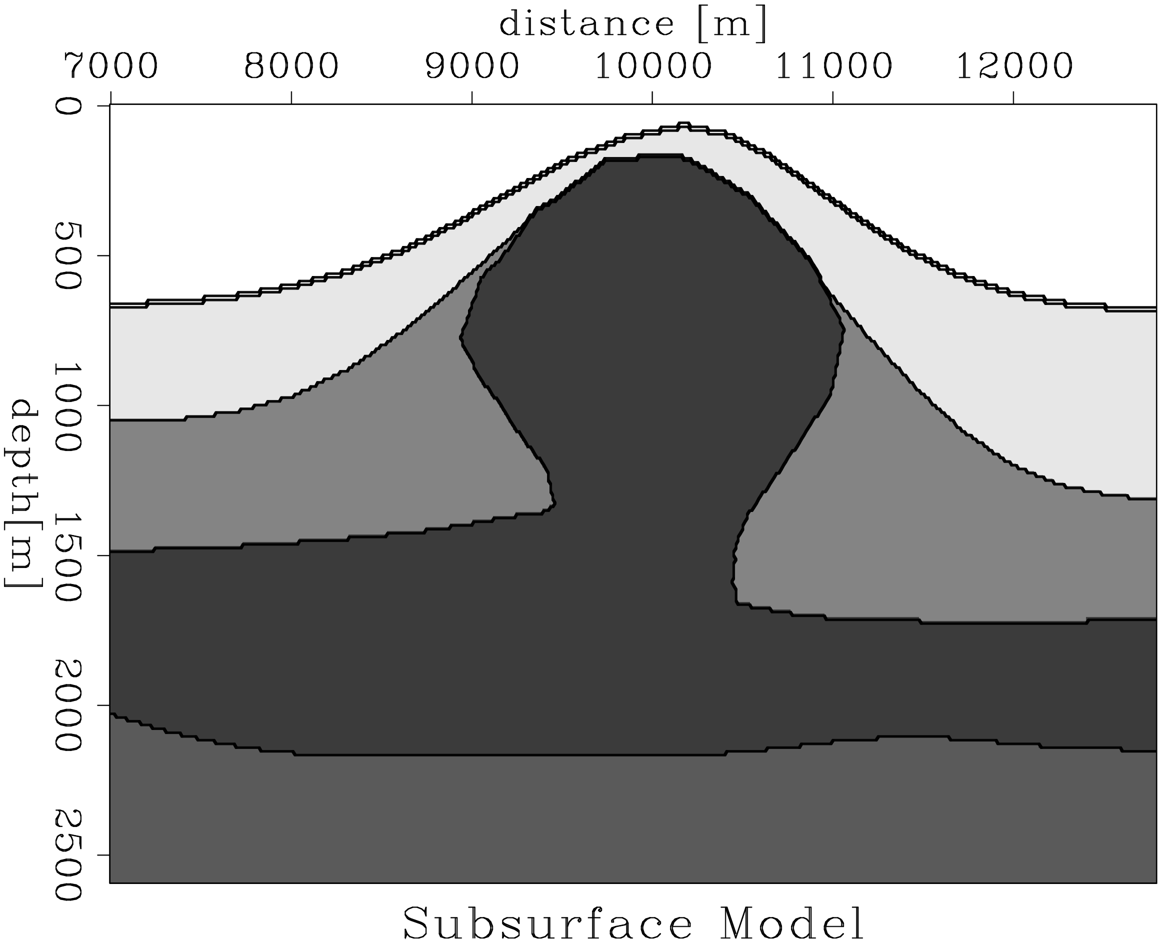

Figure 4: Gridded output of the diapir model. Press button to sweep through the 3-D model cube. diap [NR]

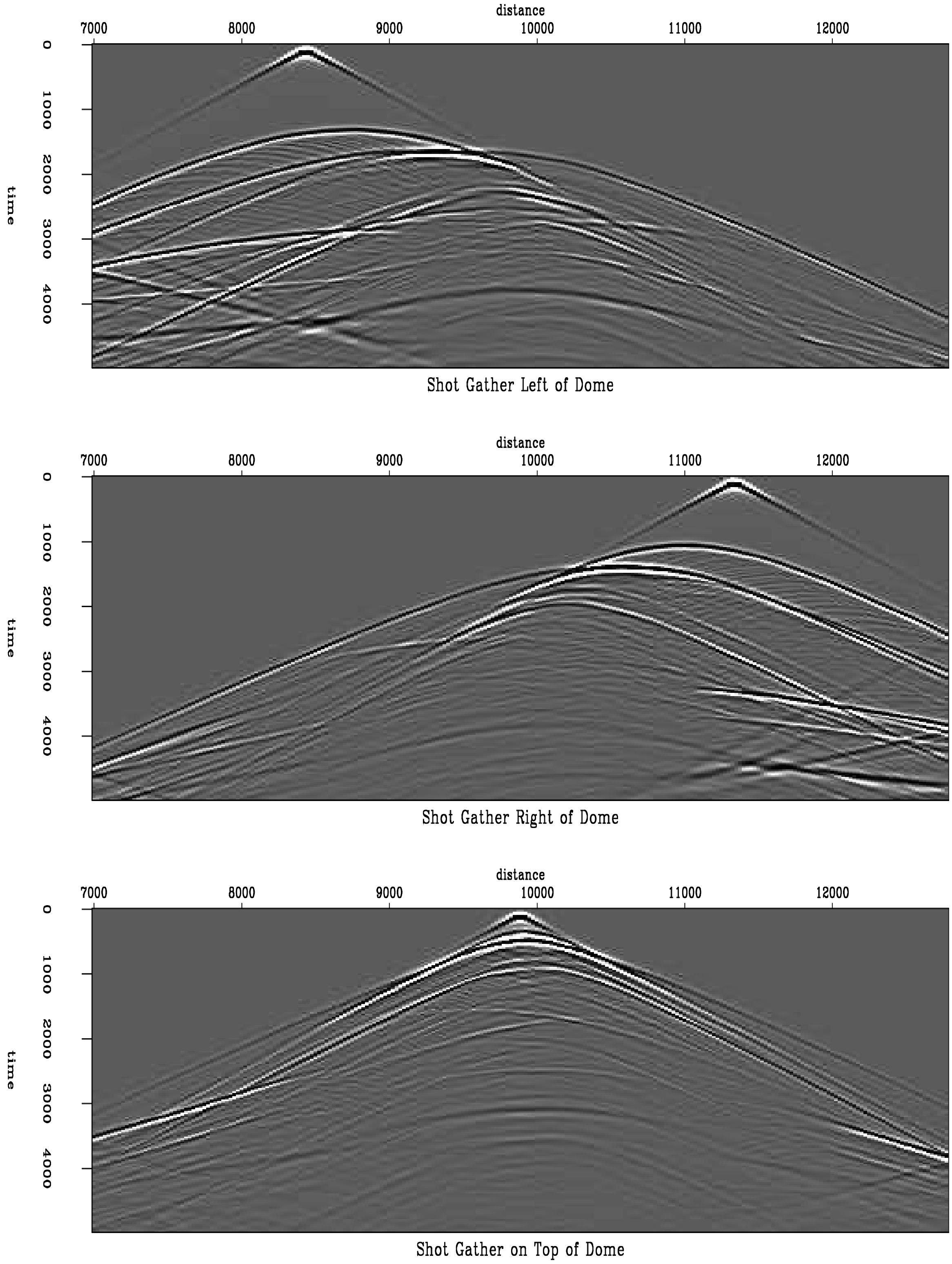

Figure 5: Three shot gathers collected over the diapir model. zseis [NR]

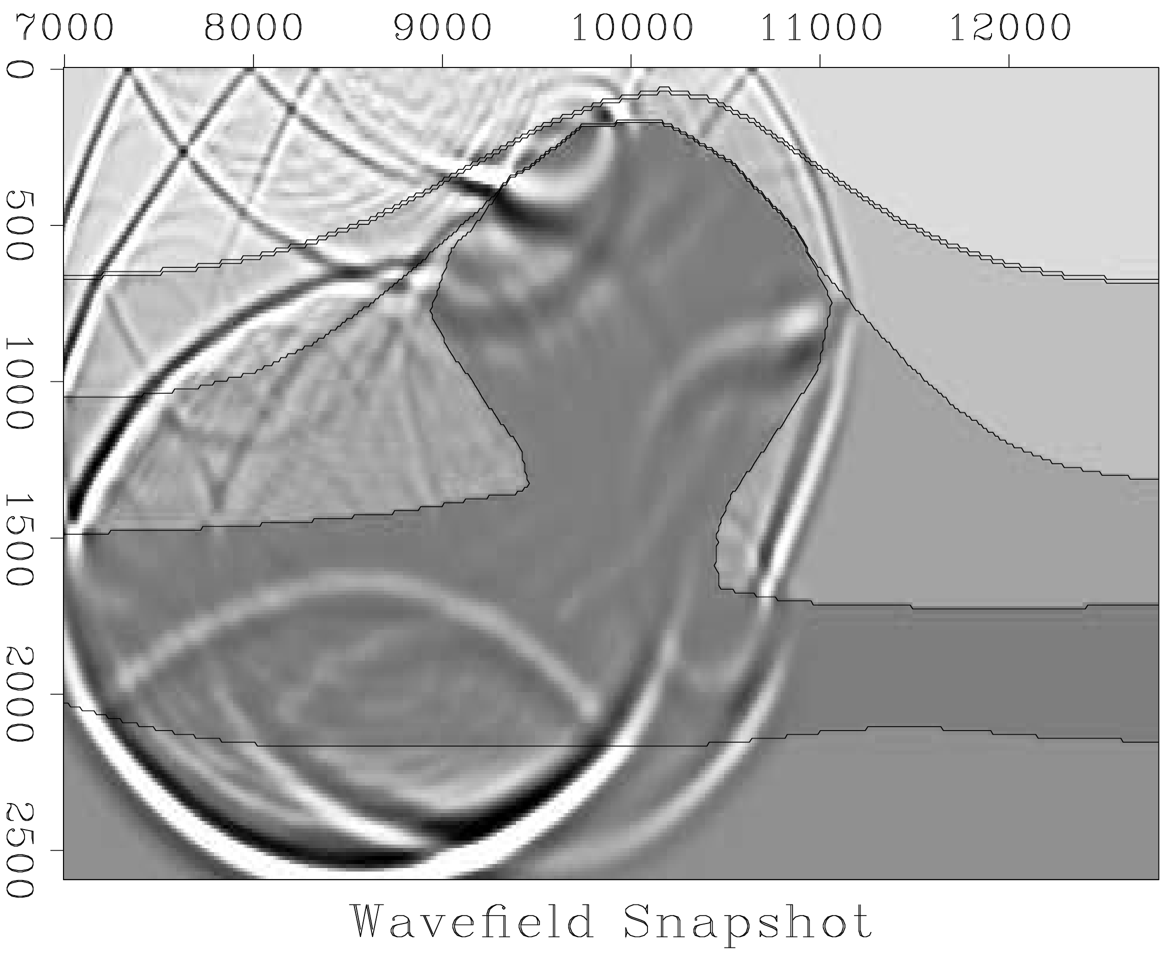

Figure 6: Snapshot of the wavefield at elapsed time 1.5 sec. Press button for a complete movie. over.left [NR]

A PROPOSAL FOR A NEW IMPLEMENTATION OF THE GRID3_t OBJECT

The aim of this section is not to determine which implementation of the 3-D grid object is the best, but rather to make a proposal of a new definition of a grid object more convenient to use for geological and/or geophysical purposes.

A quick look at the parameters file given as an example in the previous section readily shows that I have deviated a little from the original definition of the `GRID3_t` object as it is defined in GOCAD version 8.1. The reason for this is that Martin and I wanted to use a kind of SEP input format, since we had decided to use the SEP library to produce the output.

The first modification concerns the definition of the grid itself. Because in most of cases, the grids have a vertical axis corresponding to the depth or the time variable, it is not necessary any more to specify a vector for this direction. On the other hand, it is useful to allow the horizontal directions to be a linear combination of the usual vectors $\vec{i}$ and $\vec{j}$, respectively unit vectors for the x and y axes. Instead of computing the length of the generator vectors of the grid to obtain the sample on each axis, it seems more convenient to give directly these values in the input.

Another important feature is that it is often necessary to keep more than one value at the nodes of a grid, for example, densities and porosities. It is convenient for the efficiency of the finite-difference algorithms and for problems of memory size, to have one grid with several variables attached to each node rather than several grids, each containing the values of only one variable.

CONCLUSIONS

I obtained very interesting results using the existing structures of GOCAD to build a grid and fill it with velocity values. Many improvements can be added to the small program I implemented. I think that the interface produced here with GOCAD could be very useful since it allows the building of grids and especially 3-D grids from very complex models.

Some modifications could be brought to the definition of the grid object for geophysical purposes to make it more efficient and more convenient to use. Currently the maximum number of bytes that can be dynamically allocated limits the size of the output grid.

ACKNOWLEDGMENTS

I would like to thank Martin Karrenbach who gave me ideas for designing the user interface for my program. He also ran one of his finite-differences programs on a grid I produced.

REFERENCES

Karrenbach, M., 1992, “Plug ’n Play” wave equation modules: SEP–**75**, 273–288.

Mallet, J. L., 1993, GOCAD report, version 8.1, Confidential: Association Scientifique pour la Géologie et ses Applications.