of the statics problem. We will be interpreting and generalizing that approach. Our intuitive model for the data $P(s,g,\omega)$ is

$$P(s,g,\omega) \quad \approx \quad \frac{1}{1\ +\ c_s\, e^{i\omega\tau(s)}}\ \frac{1}{1\ +\ c_g\, e^{i\omega\tau(g)}} \quad \times \tag{11.5}$$
$$e^{i\frac{\omega}{v}\ \sqrt{z^2+4h^2}}\ H(h,\omega)\, Y(y,\omega)\, F(\omega)$$

The first two factors represent the split Backus filter. The next factor is the normal moveout. The factor $H(h,\omega)$ is the residual moveout. The factor $Y(y,\omega)$ is the depth-dependent earth model beneath the midpoint y. The last factor $F(\omega)$ is some average filter that results from both the earth and the recording system.

One problem with the split Backus filter is a familiar one—that the time delays $\tau(s)$ and $\tau(g)$ enter the model in a nonlinear way. So to linearize it the model is generalized to

$$P'(s,g,\omega) \quad \approx \quad S(s,\omega)\ G(g,\omega)\ H(h,\omega)\ Y(y,\omega)\ F(\omega) \tag{11.6}$$

Now S contains all water reverberation effects characteristic of the shot location, including any erratic behavior of the gun itself. Likewise, receiver effects are embedded in G. Moveout correction was done to P, thereby defining P'.

Theoretically, taking logarithms gives a linear, additive model:

$$\ln P'(s,g,\omega)\ \approx\ \ln S(s,\omega)+\ln G(g,\omega)+\ln H(h,\omega)+\ln Y(y,\omega)+\ln F(\omega) \tag{11.7}$$

The phase of P', which is the imaginary part of the logarithm, contains the *travel-time* information in the data. This information loses meaning when the data consists of more than one arrival. The phase function becomes discontinuous, even though the data is well behaved. In practice, therefore, attention is restricted to the real part of (11.7), which is really a statement about power spectra. The decomposition (11.7) is a linear problem, perhaps best solved by iteration because of the high dimensionality involved. In reconstructing S and G from power spectra, Morley used the Wiener-Levinson technique, explicitly forcing time-domain zeroes in the filters S and G to account for the water path. He omitted the explicit moveout correction in (11.5), which may account for the fact that he only used the inner half of the cable.

11.2.8 Replacement-medium concept of multiple suppression

In seismology wavelengths are so long that we tend to forget it is physically possible to have a directional wave source and a directional receiver. Suppose we had, or were somehow able to simulate, a source that radiated only down and a receiver that received only waves coming up. Then suppose that we were somehow able to downward continue this source and receiver beneath the sea floor. This would eliminate a wide class of multiple reflections. Sea-floor multiples and peglegs would be gone. That would be a major achievement. One minor problem would remain, however. The data might now lie along a line that would not be flat, but would follow the sea floor. So there would be a final step, an easy one, which would be to upward continue through a replacement medium that did not have the strong disruptive sea-floor reflection coefficient. The process just described would be called impedance replacement. It is analogous to using a replacement medium in gravity data reduction. It is also analogous to time shifting seismograms for some *replacement velocity.*

The migration operation downward continues an upcoming wave. This is like downward continuing a geophone line in which the geophones can receive only upcoming waves. In reality, buried geophones see both upcoming and downgoing waves. The directionality of the source or receiver is built into the sign chosen for the square-root equation that is used to extrapolate the wavefield. With the reciprocal theorem, the shots could also be downward continued. Likewise shots physically radiate both up and down, but we can imagine shots that radiate either up or down, and mathematically the choice is a sign. So the results of four possible experiments at the sea floor, all possibilities of upward and downward directed shots and receivers, can be deduced.

Extrapolating all this information across the sea-floor boundary requires an estimate of the sea-floor reflection coefficient. This coefficient enters the calculation as a scaling factor in forming linear combinations of the waves above the sea floor. The idea behind the reflection-coefficient estimation can be expressed in two ways that are mathematically equivalent:

- The waves impinging on the boundary from above and below should have a cross-correlation that vanishes at zero lag.
- There should be minimum power in the wave that impinges on the boundary from below.

After the geophones are below, you must start to think about getting the shots below. To invoke reciprocity, it is necessary to invert the directionality of the shots and receivers. This is why it was necessary to include the auxiliary experiment of upward-directed shots and receivers.

EXERCISES:

1 Refer to Figure 11.14.

  a. What graphical measurement shows that the interval velocity for simple sea-floor multiples equals the interval velocity for peglegs?

  b. What graphical measurements determine the sediment velocity?

  c. With respect to the velocity of water, deduce the numerical value of the (inverse) Snell parameter p.

d. Deduce the numerical ratio of the sediment velocity to the water velocity.

2 Consider the upcoming wave U observed over a layered medium of layer impedances given by $(I_1, I_2, I_3, ...)$, and the upcoming wave U' at the surface of the medium $(I_2, I_2, I_3, ...)$. Note that the top layer is changed.

a. Draw raypaths for some multiple reflections that are present in the first medium, but not in the second.

b. Presuming that you can find a mathematical process to convert the wave U to the wave U', what multiples are removed from U' that would not be removed by the Backus operator?

c. Utilizing techniques in FGDP, chapter 8, derive an equation for U' in terms of U, I_1, and I_2 that does not involve I_3, I_4 ....

11.3 PROFILE IMAGING

A field profile consists of the seismograms of one shot and many receivers along a line. Migration of a single profile, or of many widely separated profiles, demands a conceptual basis that is far removed from anything discussed so far in this book, namely, exploding-reflector and survey-sinking concepts. Such a conceptual basis exists, predates (Claerbout [1970]), and seems more basic than that of exploding reflectors or survey sinking. I call this older imaging concept the U/D imaging concept.

The sinking concept seems to demand complete coverage in shot-geophone space. Exploding reflectors requires many closely spaced shots. On the other hand, profile imaging with the U/D concept has no requirement for density along the shot axis. An example of a dataset that could only be handled by the older concept is a sonobuoy. A sonobuoy is a hydrophone with a radio transmitter. It is thrown overboard, and a ship with an air gun sails away, repeatedly firing until the range is too great. The principle of reciprocity says that the data is equivalent to a single source with a *very* long line of geophones.

While improving technology is leading to greater sampling density on the geophone axis, we are unlikely to see increasing density in shot space. There are only twenty-four hours in a day, and we must wait ten seconds between shots for the echoes to die down. So, given a certain area to survey and a certain number of months to work, we end out with an irreducible shotpoint density. Indeed, with three-dimensional geometries proving their worth, we may see *less* spatial sampling density. Poor sample density in shot space is a small impediment to profile methods.

Unlike the exploding-reflector method and the survey-sinking method, U/D concepts readily incorporate modeling and analysis of multiple reflections. Indeed, an ingenious algorithm for simultaneous migration and de-reverberation is found in FGDP. In principle it can be applied to either field profiles or slant stacks.

Wave equation methods have been suggestive of new ways of making weathering-layer corrections. Yet none have yet become widely accepted in practice, and it is too early to tell whether a DSR approach or a profile approach will work better.

All these considerations warrant a review of the profile migration method and the U/D imaging concept. We could easily see a revival of these in one form or another.

11.3.1 The U/D imaging concept

The U/D imaging concept says that *reflectors exist in the earth at places where the onset of the downgoing wave is time-coincident with an upcoming wave.* Figure 11.17 illustrates the concept.

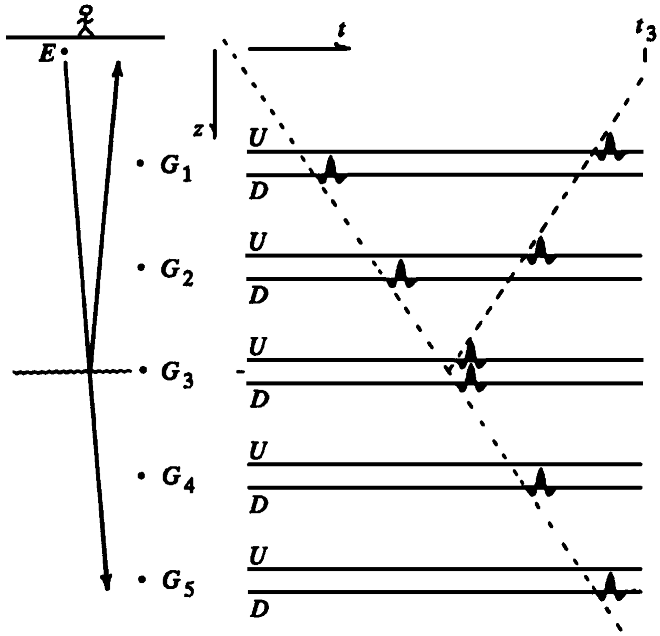

Figure 11.17: Upcoming and downgoing waves observed with buried receivers. A disturbance leaves the surface at $t = 0$ and is observed passing the buried receivers $G_1 ... G_5$ at progressively later times. At the depth of a reflector, z_3, the G_3 receiver records both the upcoming and downgoing waves in time coincidence. Shallower receivers also record both waves. Deeper receivers record only D. The fundamental principle of reflector mapping states that reflectors exist where U and D are time-coincident. (Riley) mltp-udimage [NR]

It is easy to confuse the *survey-sinking* concept with the U/D concept because of the similarity of the phrases used to describe them: "downward continue the shots" sounds like "downward continue the downgoing wave." The first concept refers to computations involving only an upcoming wavefield $U(s, g, z, t)$. The second concept refers to computations involving both upcoming $U(x, z, t)$ and downgoing $D(x, z, t)$ waves. No particular source location enters the U/D concept; the source could be a downgoing plane wave.

In profile migration methods, the downgoing wave is usually handled theoretically, typically as an impulse whose travel time is known analytically or by ray tracing. But this is not important: the downgoing wave could be handled the same way as the upcoming wave, by the Fourier or finite-difference methods described in previous chapters. The upcoming wave could be expressed in Cartesian coordinates, or in the moveout coordinate system to be described below.

The time coincidence of the downgoing and upcoming waves can be quantified in several ways. The most straightforward seems to be to look at the zero lag of the cross-correlation of the two waves. The image is created by displaying the zero-lagged cross-correlation everywhere in (x, z)-space.

The time coincidence of the upcoming wave and the earliest arrival of a downgoing wave gives evidence of the existence of a reflector, but in principle, more can be learned from the two waves. The amplitude ratio of the upcoming to the downgoing wave gives the reflection coefficient.

In the Fourier domain, the product $U(\omega, x, z)\,\bar{D}(\omega, x, z)$ represents the zero lag of the cross-correlation. The reflection coefficient ratio is given by $U(\omega, x, z)/D(\omega, x, z)$. This ratio has many difficulties. Not only may the denominator be zero, but it may have zeroes in the wrong part of the complex plane. This happens when the downgoing wave is causal but not minimum phase. The phase of the complex conjugate of a complex number equals the phase of the inverse of the number. Thus the ratio U/D and the product $\bar{U}$ both have the same phase. It seems you can invent other functional forms that compromise the theoretical appeal of U/D with the stability of $\bar{U}$.

Don C. Riley [1974] proposed another form of the U/D principle, namely, that the upcoming waves must vanish for all time before the first arrival of the downgoing wave. Riley's form found use in wave-equation dereverberation.

11.3.2 Migration with moveout correction

If the earth were truly inhomogeneous in all three dimensions, we could hardly expect the data of a single seismic line to make any sense at all. But reflection seismology usually seems to work, even when it is restricted to a single line. This indicates that the layered model of the earth is a reasonable starting point. Thus normal-moveout correction is usually a good starting process. Mathematically, NMO is an excellent tool for dealing with depth variation in velocity, but its utility drops in the presence of steep dip or a wide dip spectrum.

My early migration programs were based on concepts derived from single profiles. The data and the wave equation were transformed to a moveout-corrected coordinate system. This approach to migration is well suited to data that is sparsely sampled on the geophone axis. When steepness of dip becomes the ground on which migration is evaluated, then moveout correction offers little advantage; indeed, it introduces unneeded complexity. Whatever its merits or drawbacks, NMO commands our attention by its nearly universal use in the industrial world.

11.3.3 Moveout/radial coordinates in geophone space

Our theoretical analysis will abandon the geophone axis g in favor of a radial-like axis characterized by a Snell parameter p. (This really says nothing about the implied data processing itself, since it would be simple enough to transform final equations back to offset). The coordinate system being defined will be called a retarded, moveout-corrected, Snell trace, coordinate system. Ideal data in this coordinate system in a zero-dip earth is unchanged as it is downward continued. Hence the amount of work the differential equations have to do is proportional to the departure of the data from the ideal. Likewise the necessity for spatial sampling of the data increases in proportion to the departure of data from the ideal. Define

p	$(\sin\theta)/v$, the Snell ray parameter
t_p	any one-way time from the surface along a ray with parameter p
g	the surface separation of the shot from the geophone
t'	one-way time, surface to reflector, along a ray
τ	travel-time depth of buried geophones, one-way time along a ray
t	travel time seen by buried geophones
$v(p, t_p)$	a stratified velocity function $v'(z)$, in the new coordinates

The coordinate system is based on the following simple statements: (1) travel time from shot to geophone is twice the travel time from shot to reflector, less the time-depth of the geophone; and (2) the horizontal distance traveled by a ray is the time integral of $v \sin\theta = pv^2$; (3) the vertical distance traveled by a ray is computed the same way as the horizontal distance, but with a cosine instead of a sine.

$$t(t', p, \tau) = 2\, t' - \tau \tag{11.8}$$

$$g(t', p, \tau) = 2\, p \int_0^{t'} v(p, t_p)^2 \, dt_p - p \int_0^{\tau} v(p, t_p)^2 \, dt_p \tag{11.9}$$

$$z(t', p, \tau) = \int_0^{\tau} v(p, t_p) \sqrt{1 - p^2 v^2} \, dt_p \tag{11.10}$$

Surfaces of constant t' are reflections. Surfaces of constant p are rays. Surfaces of constant τ are datum levels. Unfortunately, it is impossible to invert the above system explicitly to get (t', p, τ) as a function of (t, g, z). It is possible, however, to proceed analytically with the differentials. Form the Jacobian matrix

$$\begin{bmatrix} \partial_{t'} \\ \partial_p \\ \partial_\tau \end{bmatrix} = \begin{bmatrix} t_{t'} & g_{t'} & z_{t'} \\ t_p & g_p & z_p \\ t_\tau & g_\tau & z_\tau \end{bmatrix} \begin{bmatrix} \partial_t \\ \partial_g \\ \partial_z \end{bmatrix} \tag{11.11}$$

Performing differentiations only where they lead to obvious simplifications gives the transformation equation for Fourier variables:

$$\begin{bmatrix} -\omega' \\ k_p \\ k_\tau \end{bmatrix} = \begin{bmatrix} 2 & g_{t'} & 0 \\ 0 & g_p & z_p \\ -1 & g_\tau & z_\tau \end{bmatrix} \begin{bmatrix} -\omega \\ k_g \\ k_z \end{bmatrix} \tag{11.12}$$

It should be noted that (11.12) is a linear relation involving the Fourier variables, but the coefficients involve the original time and space variables. So (11.12) is in both domains at once. This is useful and valid so long as it is assumed that second derivatives neglect the derivatives of the coordinate frame itself. This assumption is often benign, amounting to something like spherical divergence correction.

Here we could get bogged down in detail, were we to continue to attack the nonzero offset case. Specializing to zero offset, namely, $p = 0$, we get

$$\begin{bmatrix} -\omega' \\ k_p \\ k_\tau \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2t' - \tau & 0 \\ -1 & 0 & v \end{bmatrix} \begin{bmatrix} -\omega \\ k_g \\ k_z \end{bmatrix} \tag{11.13}$$

Equation (11.13) may be substituted into the single-square-root equation for downward continuing geophones, thereby transforming it to a retarded equation in the new coordinate system.

11.3.4 Historical notes on a mysterious scale factor

My first migrations of reflection seismic data with the wave equation were based on the U/D concept. The first wave-equation migration program was in the frequency domain and worked on synthetic profiles. Since people generally ignored such work I resolved to complete a realistic test on field data. Frequency-domain methods were deemed "academic." I found I could use the bilinear transformation of Z-transform analysis to convert the 15° wave equation to the time domain. As a practical matter, it was apparent that a profile migration program could be used on a section. But the theoretical justification was not easy. At that time I thought of the exploding-reflector concept as a curious analogy, not as a foundation for the derivation.

The actual procedure by which the first zero-offset section was migrated with finite differences was more circuitous and complicated than the procedure later introduced by Sherwood (Loewenthal et al [1976]) and adopted generally. The equation for profile migration in moveout-corrected coordinates has many terms. Neglecting all those with offset as a coefficient (since you are trying to migrate a zero-offset section), you are left with an equation that resembles the retarded, 15° extrapolation equation. But there is one difference. The $v\,\partial_{gg}$ term is scaled by a mysterious coefficient, $[t'/(2t' - \tau)]^2$. This is the equation I used. As the travel-time depth τ increases from zero to the stopping depth t', the mystery coefficient increases slowly from 1/4 to 1.

Unfortunately my derivation was so complicated that few people followed it. (You notice that I do not fully include it here). My 1972 paper includes the derivation but by way of introduction it takes you through a conceptually simpler case, namely, the seismic section that results from a downgoing plane-wave source. This simpler case brings you quickly to the migration equation. But the mystery coefficient is absent. Averaged over depth the mystery coefficient averages to a half. (The coefficient multiplies the second x-derivative and arises from Δx decreasing as geophones descend along a coordinate ray path toward the shot). Sherwood telephoned me one day and challenged me to explain why the coefficient could not be replaced by its average value, 1/2. I could give no practical reason, nor can I today. So he abandoned my convoluted derivation and adopted the exploding-reflector model as an assumption, thereby easily obtaining the required 1/2. I felt more comfortable about the mystery coefficient later when the survey-sinking concept emerged from my work with Doherty, Muir, and Clayton.

My first book, FGDP, describes how the U/D concept can be used to deal with the three problems of migration, velocity analysis, and multiple suppression. In only one of these three applications, namely, zero-offset migration (really CDP-stack migration), has the wave-equation methodology become a part of routine practice. None-the-less, the U/D concept has been generally forgotten and replaced by Sherwood's exploding-reflector concept.

11.4 PREDICTIONS FOR THE NEXT DECADE

In the 1960s seismologists learned how to apply time-series optimization theory to seismic data—see FGDP for that. Eventually time series reached the point of diminishing returns because its approach to spatial relations was oversimplified. In the 1970s seismologists learned to apply the wave equation. That's what this book has been about. You can see that the job of applying the wave equation is not yet complete, but we have come a long way. Perhaps we have solved most of our "first-order" problems, and the problems that remain are mainly "second order." For second-order effects to be significant, all the first-order phenomena must be reasonably accounted for.

Some first-order effects that this book has touched on only lightly relate to obvious, as well as subtle, imperfections in seismic data.

11.4.1 Problems in the database

We often have a problem of *truncation.* The recording cable is of course finite in length, and perceptible waves generally travel well beyond it. The seismic survey itself has finite dimensions. We also have the problem of *gaps.* Gaps in seismic data may occur unpredictably, as when a gun misfires or surveyors are denied access to parcels of land in the midst of their survey. In addition we have the problem of spatial *aliasing.* Because of improving technology, we can expect a substantial reduction in aliasing on the geophone axis, but aliasing on the shot axis will remain. There are only twenty-four hours in a day, and we must wait ten seconds between shots for the echoes to die down. So, given a certain area to survey and a certain number of months to survey it in, we end out with a certain number of shotpoints per square kilometer. With marine data, the spacing in the line of the path of the ship presents no problems compared to the problems presented by data spacing off the line.

Migration provides a mapping from a data space to a model space. This transformation is invertible (in the nonevanescent subspace). When data is missing, the transformation matrix gets broken into two parts. One part operates on the known data values, and the other part operates on the missing values. This book mostly ignores the missing part.

Noisy data can be defined as data that doesn't fit our model. If the missing data were replaced by zeroes, for example, the data would be regarded as complete, but noisy. Data is missing where the signal-to-noise ratio is known to be zero. More general noise models are also relevant, but statistical treatment of partially coherent multidimensional wave *fields* is poorly developed in both theory and practice.

My prediction is that a major research activity of the next decade will be to try to learn to simultaneously handle both the physics and the statistics of wavefields.

11.4.2 Reuniting optimization theory and wave theory

Let's take a quick peek beyond this book into the future. A seismic image is typically a 1000×1000 plane, derived from a volume of about 1000^3 interrelated data points. There are unknowns present everywhere, not only in the earth model, but also in the data, as noise, as gaps, and as insufficient spatial density and extent of data recording. To assemble an interpretation we must combine principles from physics with principles from statistics. Presumably this could be done in some monster optimization formulation. A look at the theory of optimization shows that solution techniques converge in a number of iterations that is greater than the number of unknowns. Thus the solution to the problem, once we learn how to pose the problem properly, seems to require about a million times as much computing power as is available. What a problem!

But the more you look at the problem, the more interesting it becomes. First we have an optimization problem. Since we are constrained to make only a few iterations, say, three, we must go as far as we can in those three steps. Now, not only do we have the original optimization problem, but we also have the new problem of solving it in an optimum way. First we have correlated randomness in the raw data. Then, during optimization, the earth model changes in a correlated random way from one iteration to the next. Not only is the second optimization problem the practical one—it is deeper at the theoretical level.

11.4.3 Throw away your paper sections.

Current seismic interpretation often amounts to taking colored pencils and enhancing aspects of a computer-generated image. Seismic interpretation is entering an era in which the interpretation will all be done on a video screen. The basic reason is that a sheet of paper is only two-dimensional, while most reflection data is three-dimensional. Modern 3-D surveys really record four-dimensional data. A video screen can show a movie. The operator/interpreter can interact with the movie. There are things I would like to show you, but I cannot show you in a book. Seismic data or even a blank sheet of paper has *texture.* When a textured object moves, you immediately recognize it. But I couldn't show it to you with pictures in this book.

(Imagine a sequence of pictures of a blank sheet of paper, each one shifted some way from the previous one). The perception of small changes is blocked by any eye movement between pictures. Astronomers look for changes in the sky by rapidly blinking between looking at photographs taken at different times. Our eyes are special computers. Movies often show "where something comes from," enabling us to notice the unexpected in the general ambiance.

Most seismic interpretation is done on *stacked sections.* The original data is three-dimensional, but one dimension is removed by summation. Theoretically, the summation removes only redundancy while it enhances the signal-to-noise ratio. In reality, things are much more complicated. And much more will be perceptible when summations are done by the human eye (just by increasing the speed of a movie). There will be two generations of seismic interpreters—those who can interpret the prestacked data they see on their video screens—and those who interpret only stacked sections.

Appendix A

RATional FORtran == Ratfor

Bare-bones **Fortran** is our most universal computer language for computational physics. For general programming, however, it has been surpassed by **C**. "**Ratfor**" is Fortran with C-like syntax. I believe Ratfor is the best available expository language for mathematical algorithms. Ratfor was invented by the people who invented C. Ratfor programs are converted to Fortran with the Ratfor **preprocessor**. Since the preprocessor is publicly available, Ratfor is practically as universal as Fortran.[1]

You will not really need the Ratfor preprocessor or any precise definitions if you already know Fortran or almost any other computer language, because then the Ratfor language will be easy to understand. Statements on a line may be separated by ";". Statements may be grouped together with braces { }. Do loops do not require statement numbers because { } defines the range. Given that `if( )` is true, the statements in the following { } are done. `else{ }` does what you expect. We may *not* contract `else if` to `elseif`. We may always omit the braces { } when they contain only one statement. `break` will cause premature termination of the enclosing { }. `break 2` escapes from {{ }}. `while( ) { }` repeats the statements in { } while the condition ( ) is true. `repeat { ... } until( )` is a loop that tests at the bottom. A looping statement more general than `do` is `for`(initialize; condition; reinitialize) { }. An example of one equivalent to `do i=0,n-1` is the looping statement `for(i=0;i<n;i=i+i)`. The statement `next` causes skipping to the end of any loop and a retrial of the test condition. `next` is rarely used, but when it is, we must beware of an inconsistancy between Fortran and C-language. Where Ratfor uses `next`, the C-language uses `continue` (which in Ratfor and Fortran is merely a place holder for labels). The Fortran relational operators `.gt.`, `.ge.`, `.ne.`, etc. may be written >, >=, !=, etc. The logical operators `.and.` and `.or.` may be written & and |. Anything from a # to the end of the line is a comment. Anything that does not make sense to the Ratfor preprocessor, such as Fortran input-output, is passed through without change. (Ratfor has a `switch` statement but we never use it because

[1]Kernighan, B.W. and Plauger, P.J., 1976, Software Tools: Addison-Wesley. Ratfor was invented at AT&T, which makes it available directly or through many computer vendors. The original Ratfor transforms Ratfor code to **Fortran** 66. A public-domain Ratfor transforms it to **Fortran** 77.

it conflicts with the `implicit undefined` declaration. Anybody want to help us fix the `switch` in public domain Ratfor?)

Indentation in `Ratfor` is used for readability. It is not part of the `Ratfor` language. Choose your own style. I have overcondensed. There are two **pitfall**s associated with indentation. The beginner's pitfall is to assume that a `do` loop ends where the indentation ends. The loop ends after the first statement. A larger scope for the `do` loop is made by enclosing multiple statements in braces. The other pitfall arises in any construction like `if() ... if() ... else`. The `else` goes with the last `if()` regardless of indentation. If you want the `else` with the earlier `if()`, you must use braces like `if() { if() ... } else ...`.

The most serious limitation of **Fortran**-77 is its lack of ability to allocate temporary memory. This should be remedied in Fortran-8x. The "x" in 8x refers, unfortunately, to the year in the 1980s that the language was intended to become available. Now it is officially called Fortran 90, though I call it Fortran 9x since in 1994 it is still not on any of our workstations. Meanwhile, with Biondo Biondi and Dave Nichols, I have written a **preprocessor** to **Ratfor** or **Fortran** to overcome Fortran's memory-allocation limitation. This program, named **sat**, allows subroutines to include the declaration `temporary real data(n1,n2)`, so that memory is allocated during execution of the subroutine where the declaration is written. Fortran-77 forces us to accomplish something like this only with predetermined constants or parameters. If the **sat** preprocessor is not available on your system, you can modify the subroutines in this book by putting the appropriate numerical constants into the memory arrays being allocated, and that should suffice until the better days ahead.

Appendix B

Seplib and SEP software

At the time of writing, this book can be run on a variety of computers. You will have noticed that each figure caption contains a box enclosing a label. In the electronic book, this box is a pushbutton that generally activates a rebuilding of the figure, sometimes after program or parameter changes and sometimes interactively. The label in the box points to the location of the underlying software. My associates and I have worked through complete cycles of "**burn**ing" and building all the figures on various computers. To enable you to do the same, and to further enable you to rapidly build on my work, I have released electronic copies of my books.[1] This short appendix describes the utility software that is used extensively in the electronic book.

Most of the seismic utility software at **SEP**[2] (**SEP**) handles seismic data as a rectangular lattice or "cube" of numbers. Each cube-processing program appends to the history file for the cube. Preprocessors extend **Fortran** (or **Ratfor**) to enable it to allocate memory at run time, to facilitate input and output of data cubes, and to facilitate self-documenting programs.

At the **SEP** a library of subroutines known as **seplib** evolved for routine operations. These subroutines mostly handle data in the form of cubes, planes, and vectors. A cube is defined by 14 parameters with standard names and two files: one the data cube itself, and the other containing the 14 parameters and a history of the life of the cube as it passed through a sequence of cube-processing programs. Most of these cube-processing programs have been written by researchers, but several nonscientific cube programs have become highly developed and are widely shared. Altogether there are (1) a library of subroutines, (2) a library of main programs, (3) some naming conventions, and (4) a graphics library called **vplot**. The subroutine library has good manual pages. The main programs rarely have manual pages, their documentation being supplied by the on-line self-documentation that is extracted from the comments

[1]My previous book, "Earth soundings analysis: processing versus inversion," is available on CD-ROM titled **SEP**-CD-4 from Stanford University Press, telephone 415-723-1593 for $15. My current book, "Basic Earth Imaging" (BEI) has been released in incomplete form to sponsors of the **Stanford Exploration Project (SEP)** on disk SEP-CD-3.

[2]Old reports of the Stanford Exploration Project can be found in the library of the Society of Exploration Geophysicists in Tulsa, Oklahoma.

at the beginning of the source file. Following is a list of the names of popular main programs:

`Byte`	Scale floats to brightness bytes for raster display.
`Cat`	Concatenate conforming cubes along the 3-axis.
`Contour`	Contour plot a plane.
`Cp`	Copy a cube.
`Dd`	Convert between ASCI, floats, complex, bytes, etc.
`Dots`	Plot a plane of floats.
`Ft3d`	Do three-dimensional Fourier transform.
`Graph`	Plot a line of floats.
`In`	Check the validity of a data cube.
`Merge`	Merge conforming cubes side by side on any axis.
`Movie`	View a cube with Rick Ottolini's cube viewer.
`Noise`	Add noise to data.
`Reverse`	Reverse a cube axis.
`Spike`	Make a plane wave of synthetic data.
`Ta2vplot`	Convert a byte format to raster display with `vplot`.
`Tpow`	Scale data by a power of time t (1-axis).
`Thplot`	Make a hidden line plot.
`Transpose`	Transpose cube axes.
`Tube`	View a `vplot` file on a screen.
`Wiggle`	Plot a plane of floats as "wiggle traces."
`Window`	Find a subcube by truncation or subsampling.

To use the cube-processing programs, read this document, and then for each command, read its on-line self-documentation. To write cube-processing programs, read the manual page for `seplib` and the subroutines mentioned there and here. To write `vplot` programs, see the references on `vplot`.

B.1 THE DATA CUBE

The data cube itself is like a Fortran three-dimensional matrix. Its location in the computer file system is denoted by `in=PATHNAME`, where `in=` is the literal occurrence of those three characters, and `PATHNAME` is a directory tree location like `/sep/professor/pvi/data/western73.F`. Like the Fortran cube, the data cube can be real, complex, double precision, or byte, and these cases are distinguished by the element size in bytes. Thus the history file contains one of `esize=4`, `esize=8`, or `esize=1`, respectively. Embedded blanks around the "=" are always forbidden. The cube values are binary information; they cannot be printed or edited (without the intervention of something like a Fortran "format"). To read and write cubes, see the manual pages for such routines as `reed, sreed, rite, srite, snap`.

A cube has three axes. The number of points on the 1-axis is `n1`. A Fortran declaration of a cube could be `real mydata(n1,n2,n3)`. For a plane, `n3=1`, and for a line, n2=1. In addition, many programs take "1" as the default for an undefined value of n2 or n3. The physical location of the single data value mydata(1,1,1), like a mathematical origin (o_1, o_2, o_3), is denoted by the three real variables o1, o2, and o3. The data-cube values are presumed to be uniformly spaced along these axes like the mathematical increments $(\Delta_1, \Delta_2, \Delta_3)$, which may be negative and are denoted by the three real variables d1, d2, and d3.

Each axis has a label, and naturally these labels are called label1, label2, and label3. Examples of labels are kilometers, sec, Hz, and "offset, km". Most often, label1="time, sec". Altogether that is $2 + 3 \times 4$ parameters, and there is an optional title parameter that is interpreted by most of the plot programs. An example is title="Yilmaz and Cumro Canada profile 25". We reserve the names n4,o4,d4, and label4 (a few programs support them already), and please do not use n5 etc. for anything but a five-dimensional cubic lattice.

B.2 THE HISTORY FILE

The 15 parameters above, and many more parameters defined by authors of cube-processing programs, are part of the "**history file**" (which is ASCI, so we can print it). A great many cube-processing programs are simple filters—i.e., one cube goes in and one cube comes out—and that is the case I will describe in detail here. For other cases, such as where two go in and one comes out, or none go in and one comes out (synthetic data), or one goes in and none come out (plotting program), I refer you to the manual pages, particularly to subroutine names beginning with aux (as in "auxiliary").

Let us dissect an example of a simple cube-processing program and its use. Suppose we have a seismogram in a data cube and we want only the first 500 points on it, i.e., the first 500 points on the 1-axis. A utility cube filter named Window will do the job. Our command line looks like

```
< mygiven.H Window n1=500 > myshort.H
```

On this command line, mygiven.H is the name of the history file of the data we are given, and myshort.H is the history file we will create. The moment Window, or any other seplib program, begins, it copies mygiven.H to myshort.H; from then on, information can only be appended to myshort.H. When Window learns that we want the 1-axis on our output cube to be 500, it does call putch('n1','i',500), which appends n1=500 to myshort.H. But before this, some other things happen. First, seplib's internals will get our log-in name, the date, the name of the computer we are using, and Window's name (which is Window), and append these to myshort.H. The internals will scan mygiven.H for in=somewhere to find the input data cube itself, and will then figure out where we want to keep the output cube. Seplib will guess that someone named professor wants to keep his data cube at some place like /scr/professor/_Window.H@. You should read the manual page for datapath to see how you can set up the default location for your datasets. The reason datapath exists is to facilitate isolating data from text, which is usually helpful for archiving.

When a cube-processing filter wonders what the value is of `n1` for the cube coming in, it makes a subroutine call like `call hetch("n1","i",n1)`. The value returned for `n1` will be the *last* value of `n1` found on the history file. `Window` also needs to find a different `n1`, the one we put on the command line. For this it will invoke something like `call getch("n1","i",n1out)`. Then, so the next user will know how big the output cube is, it will `call putch("n1","i",n1out)`. For more details, see the manual pages.

If we want to take input parameters from a file instead of from the command line, we type something like `<in.H Window par=myparfile.p > out.H`. The `.p` is my naming convention and is wholly optional, as is the `.H` notation for a history file.

`Sepcube` programs are self-documenting. When you type the name of the program with no input cube and no command-line arguments, you should see the self-documentation (which comes from the initial comment lines in the program).

SEP software supports "pipelining." For example, we can slice a plane out of a data cube, make a contour plot, and display the plot, all with the command line

```
<in.H Window n3=1 | Contour | Tube
```

where, as in UNIX pipes, the "`|`" denotes the passage of information from one program to the next. Pipelining is a convenience for the user because it saves defining a location for necessary intermediate files. The history files do flow down UNIX pipes. You may not have noticed that some location had to be assigned to the data at the intermediate stages, and when you typed the pipeline above, you were spared that clutter. To write `seplib` programs that allow pipelining, you need to read the manual page on `hclose()` to keep the history file from intermingling with the data cube itself.

A sample history file follows: this was an old one, so I removed a few anachronisms manually.

```
 # Texaco Subduction Trench: read from tape by Bill Harlan
n1=1900 n2=2274
o1=2.4 it0=600 d1=.004 d2=50.  in=/d5/alaska

Window:   bill   Wed Apr 13 14:27:57 1983
input() :    in ="/d5/alaska"
output() : sets next in="/q2/data/Dalw"
Input:  float Fortran (1900,2274,1)
Output: float Fortran (512,128,1)
 n1=512 n2=128 n3=1

Swab:   root@mazama   Mon Feb 17 03:23:08 1986
# input history file /r3/q2/data/Halw
input() :    in ="/q2/data/Dalw"
output() : sets next in="/q2/data/Dalw_002870_Rcp"
#ibs=8192 #obs=8192

Rcp: paul Mon Feb 17 03:23:15 PST 1986
Copying from mazama:/r3/q2/data/Halw
```

```
to hanauma:/q2/data/Halw
in="/q2/data/Dalw"

Cp:   jon@hanauma   Wed Apr  3 23:18:13 1991
input() :    in ="/q2/data/Dalw"
output() : sets next in="/scr/jon/_junk.H@"
```

B.3 MEMORY ALLOCATION

Sepcube programs can be written in Fortran, Ratfor, or C. A serious problem with **Fortran**-77 (and hence Ratfor) is that memory cannot be allocated for arrays whose size is determined at run time. We have worked around this limitation by using two home-grown preprocessors, one called **saw** (Stanford Auto Writer) for main programs, and one called **sat** (Stanford Auto Temporaries) for subroutines. Both preprocessors transform either Fortran or Ratfor.

B.3.1 Memory allocation in subroutines with sat

The **sat** preprocessor allows us to declare temporary arrays of arbitrary dimension, such as

```
temporary real*4 data(n1,n2,n3), convolution(j+k-1)
```

These declarations must follow other declarations and precede the executable statements.

B.3.2 The main program environment with saw

The **saw** preprocessor also calls an essential initialization routine `initpar()`, organizes the self-doc, and simplifies data-cube input. See the on-line self-documentation or the manual pages for full details. Following is a complete **saw** program for a simple task:

```
# <in.H  Scale scaleval=1. > out.H
#
# Copy input to output and scale by scaleval
# keyword generic scale
#%
integer n1, n2, n3, esize
from history: integer n1, n2, n3, esize
if (esize !=4) call erexit('esize != 4')
allocate: real x(n1,n2)

subroutine scaleit( n1,n2, x)
integer i1,i2, n1,n2
```

```
real x(n1,n2), scaleval
from par: real scaleval=1.
call hclose() # no more parameter handling.
call sreed('in', x, 4*n1*n2)
do i1=1,n1
do i2=1,n2
x(i1,i2) = x(i1,i2) * scaleval
call srite( 'out', x, 4*n1*n2)
return; end
```

REFERENCES

Claerbout, J., 1990, Introduction to `seplib` and SEP utility software: **SEP–70**, 413–436.

Claerbout, J., 1986, A canonical program library: **SEP–50**, 281–290.

Cole, S., and Dellinger, J., Vplot: **SEP**'s plot language: SEP-**60**, 349–389.

Dellinger, J., 1989, Why does **SEP** still use Vplot?: SEP–**61**, 327–335.

B.4 BASIC SUBROUTINES

A few trivial subroutines used in this book are below:

```
subroutine null(     xx, n)
integer i, n;   real xx( n)
do i= 1, n
        xx(i) = 0.
return; end
```

```
subroutine scale( factor, n, data)
integer i,                n
real factor, data(n)
do i= 1, n
        data(i) = factor * data(i)
return; end
```

```
subroutine copy( n, xx, yy)
integer i, n;   real xx(n), yy(n)
do i= 1, n
        yy(i) = xx(i)
return; end
```