PSPI: STREAMLINING 3D ECHO-RECONSTRUCTIVE IMAGING

E.BONOMI, L.BRIEGER, E.PIERONI * AND M.T.ARIENTI, L.CAZZOLA, P.MARCHETTI †

1. Introduction. Echo-reconstruction techniques for subsurface imaging, widely used in oil exploration, are based on experiments in which short acoustic impulses, emitted at the surface, illuminate a certain volume and are backscattered by inhomogeneities of the medium. The inhomogeneities act as reflecting surfaces which cause signal echoing; the echoes are then recorded at the surface and processed through a propagation model (which acts as a "computational lens") to yield an image of those very inhomogeneities.

Migration, based on the scalar wave equation, is the standard imaging technique for seismic applications [1]. In the migration process, the recorded pressure waves (called the seismic traces or the seismic section) are used as initial conditions for a wave field governed by the scalar wave equation in an inhomogeneous medium. Any migration technique begins with an *a priori* estimate of the velocity field, obtained from well logs and an empirical analysis of the seismic traces, to yield a subsurface image. By comparing imaged interfaces with the discontinuities of the estimated velocity model, insufficiencies of the velocity field can be detected and velocity estimates improved [2], allowing the next migration step to image more accurately. The turnaround process, the iterative process of correcting to a velocity model consistent with the migrated data, can last several computing weeks and is particularly crucial for imaging complex geological structures, including those which are interesting for hydrocarbon prospecting.

Subsurface depth imaging, being as it is the outcome of repeated steps of 3D seismic data migration, requires Gbytes of data which must be reduced, transformed, visualized and interpreted to obtain meaningful information. Severe performance requirements oblige high performance computing hardware and techniques; enormous effort has also historically gone into simplifying the migration model so as to reduce the cost of the operation while retaining the essential features of the signal propagation. One such model leads to the phase-shift-plus-interpolation (PSPI) algorithm, a high-quality method for seismic migration. By optimizing the algorithm to reduce calculation requirements and exploiting its decoupling in the frequency domain for concurrency in parallel implementations, we have created in PSPI a cost-effective method.

The PSPI code developed in our collaboration was first implemented in Portland Group's HPF (PGHPF), for use on an IBM SP2 and an SGI Power Challenge. Porting to an SGI Origin2000 resulted in substantial loss of performance of the HPF code and motivated the decision to re-implement it in OpenMP for use on the shared-memory Origin2000. This should not have posed particular problems, even though HPF is centered around data distribution, while the philosophy behind OpenMP is distribution of work; for PSPI, task distribution and data distribution are practically synonymous, since the concurrent tasks correspond to different data (frequencies). It is natural and should be equally efficient to distribute frequency information among the processors on a distributed-memory machine (HPF-style) or to distribute frequency-related tasks among the processors on a shared-memory machine (OpenMP-style).

*Geophysics Group, CRS4, Italy
†Geophysical Research, AGIP Division, ENI, Italy

So far, our experience with OpenMP on the Origin2000 has been disappointing. The OpenMP code suffers somewhat from the immaturity of the standard. There are currently no efficient array-level reduction operations. Data placement is not part of the OpenMP standard, even though data position can influence performance on shared-memory machines, such as the Origin2000, which have non-uniform memory access times. This obliges the use of non-portable, machine-dependent directives or conventions for data placement. Even using these, on the Origin, such directives cannot override operating system (OS) control of thread and page handling; as a result, the user exerts much less direct control over data placement than might be desirable. Performance seems more dependent on the OS and state of the machine than on programming technique.

On the other hand, improvements in OS and compilers have restored performance of the HPF code on the Origin2000 back to acceptable levels. HPF does not suffer excessively from OS interference nor from competition with other applications running on the machine. Running the HPF implementation on the Origin, we have a fast, scalable, and stable code.

2. The Phase Shift Formula. In a zero-offset model, the seismic section $P(x,y,0,t)$ is used as a surface boundary condition for solving the scalar wave equation

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} + \frac{\partial^2 P}{\partial z^2} - \frac{1}{v(x,y,z)^2}\frac{\partial^2 P}{\partial t^2} = 0 \tag{2.1}$$

in reverse time with zero initial conditions. The exploding reflector model on which zero-offset is based allows us to apply the Claerbout principle [3],[4] and interpret the migrated section $P(x,y,z,t=0)$ as a map of the local reflectivity, $R(x,y,z)$, yielding an acoustic “picture” of the reflectors in what is often called the imaging condition:

$$R(x,y,z) = P(x,y,z,t=0). \tag{2.2}$$

The original phase shift migration method was formulated by J. Gazdag [5] as a fast and simple implementation of zero-offset data migration. It is based on the fact that, as long as velocity v is constant, Eq.(2.1) can be solved, and thus the acoustic image produced, using depth z as the advancing variable along which to propagate the seismic section $P(x,y,0,t)$; this is known as depth extrapolation or depth continuation and is outlined in the following.

Eq.(2.1) written in the wavenumber-frequency domain (k_x, k_y, ω) is the second-order ordinary differential equation

$$\frac{d^2\hat{P}(k_x,k_y,z,\omega)}{dz^2} = -k_z^2\hat{P}(k_x,k_y,z,\omega)\ , \tag{2.3}$$

in which

$$k_z = \frac{\omega}{v}\sqrt{1-\left(\frac{v}{\omega}\right)^2(k_x^2+k_y^2)}\ . \tag{2.4}$$

Eq.(2.3) has two characterisic solutions relating the field at level z with that at level $z+\Delta z$ by a phase shift, but since we want to follow downward displacement of the signals in reverse time, for depth extrapolation we are interested only in the characteristic solution

$$\hat{P}(k_x,k_y,z+\Delta z,\omega) = \hat{P}(k_x,k_y,z,\omega)e^{ik_z\Delta z}\ . \tag{2.5}$$

This is also a solution of the paraxial or one-way wave equation

$$\frac{d\hat{P}}{dz}(k_x, k_y, z, \omega) = ik_z \hat{P}(k_x, , k_y, z, \omega) \ , \tag{2.6}$$

which provides a hierarchy of other migration methods based on approximations of one-way propagation in the space-frequency domain (x, y, ω).

If we use the inverse Fourier transform to map the solution $\hat{P}(k_x, k_y, z + \Delta z, \omega)$ back to the space-frequency domain, we can then apply imaging condition (2.2) and generate the image $R(x, y, z + \Delta z)$ of migrated data at level $z + \Delta z$ by noting that

$$P(x, y, z + \Delta z, t = 0) = \sum_{\omega} \hat{P}(x, y, z + \Delta z, \omega) \ . \tag{2.7}$$

Eqs.(2.4), (2.5) and (2.7), along with imaging condition (2.2), form the basis for the phase shift migration algorithm. They allow the exact inverse extrapolation of seismic data inside a homogeneous layer $]z, z + \Delta z]$ with constant velocity.

Since the power spectrum of the seismic source is band-limited with a cutoff frequency far below the temporal Nyquist, mapping data into the space-frequency domain allows significant data compression. In addition, the phase shift formulation of migration leads to an elegant parallel implementation: the depth extrapolation is composed of entirely concurrent operations, and only the imaging condition sum requires any inter-processor communication or remote memory access.

3. Phase Shift Plus Interpolation. Whereas seismic imaging in a stratified medium can be handled piecewise with the simple phase shift formula, the case with lateral velocity variations requires more attention. In this context the Fourier representation (2.3) of the scalar wave equation is meaningless and no straightforward representation of the solution as with the phase shift formula is possible. To overcome this difficulty and yet keep the computational complexity of the migration to a minimum, the wave propagation model is modified in order to construct a pure spectral method for downward extrapolation in an inhomogeneous medium.

The starting point is the phase shift formula (2.5), split into vertical and horizontal components and then modified to handle wave propagation inside the layer $]z, z+\Delta z]$ which has a laterally variable velocity field. The resulting first term governs vertically-travelling waves through the layer:

$$\hat{P}_0(x, y, z, \omega) = \hat{P}(x, y, z, \omega) e^{i\frac{\omega}{v}\Delta z} \ , v = v_z(x, y) \ . \tag{3.1}$$

The second term governs the horizontal correction for a reference velocity $v_z^{(j)}$, one of $v_z^{(1)} < v_z^{(2)} < \cdots < v_z^{(n_z)}$:

$$\hat{P}^{(n)}(k_x, k_y, z + \Delta z, \omega) = \hat{P}_0(k_x, k_y, z, \omega) \exp\left(i(k_z^{(n)} - \frac{\omega}{v_z^{(n)}})\Delta z \right) \ , \tag{3.2}$$

with $k_z^{(n)}$, $n = 1, 2, \cdots, n_z$, given by Eq.(2.4) evaluated for reference velocity $v_z^{(n)}$.

Fourier-transformed back to the space-frequency domain, the fields $\hat{P}^{(n)}(x, y, z + \Delta z, \omega)$ then serve as reference data from which the final result is obtained by interpolation. Using linear interpolation, the depth-continued wave field is given by

$$\begin{aligned} \hat{P}(x, y, z + \Delta z, \omega) \quad &= \quad \hat{P}^{(n)}(x, y, z + \Delta z, \omega) \times \frac{v_z^{(n+1)} - v_z(x, y)}{v_z^{(n+1)} - v_z^{(n)}} \\ &\quad + \hat{P}^{(n+1)}(x, y, z + \Delta z, \omega) \times \frac{v_z(x, y) - v_z^{(n)}}{v_z^{(n+1)} - v_z^{(n)}} \ , \end{aligned} \tag{3.3}$$

for all points (x, y, z) with $v_z^{(n)} \leq v_z(x, y) \leq v_z^{(n+1)}$. Finally, imaging of the reflectors at $z + \Delta z$ is obtained by the usual condition, Eq.(2.2). This is the essence of the PSPI method, as introduced by J. Gazdag and P. Sguazzero [6].

The PSPI algorithm defined by Eqs.(3.1), (3.2) and (3.3) gives correct depth continuation for vertically-travelling plane waves and maintains high accuracy for small dips characterized by $\sqrt{k_x^2 + k_y^2} \ll |\omega|/v_z^{(n)}$. Although the introduction of laterally-variable velocities and the use of interpolating reference solutions do not have a well established mathematical basis, computer experiments show that this approach is in fact very reliable. PSPI is a practical alternative to other forms of migration – as long as the set of reference velocities is well-chosen. Its implementational advantage is that each reference solution comes from a constant velocity extrapolation whose implementation inherits the parallel structure of the phase shift algorithm [7].

4. Optimal Reference Velocities. The velocities $v_z^{(n)}$ play a crucial role in PSPI migration of seismic data – for accuracy and cost of the method. Using statistical arguments, representative velocities can be optimally chosen for the velocity field of a given layer $]z, z + \Delta z]$. The task is to highlight, for each layer, a minimal set of velocity values that predominate statistically in the propagation process. The manner in which this is done, outlined here, is described in detail in [8].

Using some number $N + 1$ of reference velocities, the velocity interval $[v_m, v_M]$ (where v_m and v_M are the minimal and maximal velocity values on the entire field) is uniformly discretized into N subintervals, and a distribution F_z is defined over the discretization: $F_z^{(k)}$, $k = 1, \ldots, N$, is the fraction of velocities (from the velocity field in that layer) contained in the k^{th} interval. In order to optimize the set of reference velocities, we define the number N_z ($N_z \leq N$) of intervals over which the velocities will have uniform distribution with dispersion S_z, where $S_z[F] = -\sum_{k=1}^{N} F_z^{(k)} \log F_z^{(k)}$ is the "statistical entropy" of the distribution F_z. It follows that $N_z = \lfloor \exp(S_z[F]) + \frac{1}{2} \rfloor$. Then the $N_z + 1$ new reference velocities are chosen so that the resulting distribution of velocities over this discretization is uniform.

Not only does this optimization significantly reduce the operation count for PSPI, it also concentrates more reference velocities where the distribution of velocities is large and results in fewer where the distribution is small. With this adaptive mechanism, the accuracy of PSPI must indeed increase on average because of the statistical importance conferred to those velocities that contribute massively to the downward propagation of the wave field.

5. Reconstruction of a Subsurface Model. We study a synthetic example, for which we know the velocity field to be correct, in order to examine the effect of PSPI simplifications in imaging a complex model. Fig.5.1 illustrates the example velocity field for a vertical slice of a 3D subsurface model: $\Delta x = \Delta y = 23.3$ m, $\Delta z = 10$ m. Velocities vary from 800 to 2500 m/s.

A synthetic seismic section was generated using an acoustic wave propagation model to simulate the surface signals that would be received during a seismic acquisition over this velocity field. For the depth extrapolation of the seismic data, the velocity field of Fig.5.1 was initially discretized using 40 reference velocities. The actual number finally used for a given layer, after the optimization procedure described above, varied from 1 to 6, depending on depth; on average only four reference velocities were necessary for a layer. Such an economy in the number of reference velocities necessary for the PSPI algorithm translates into an important reduction in computation time for depth extrapolation and PSPI migration.

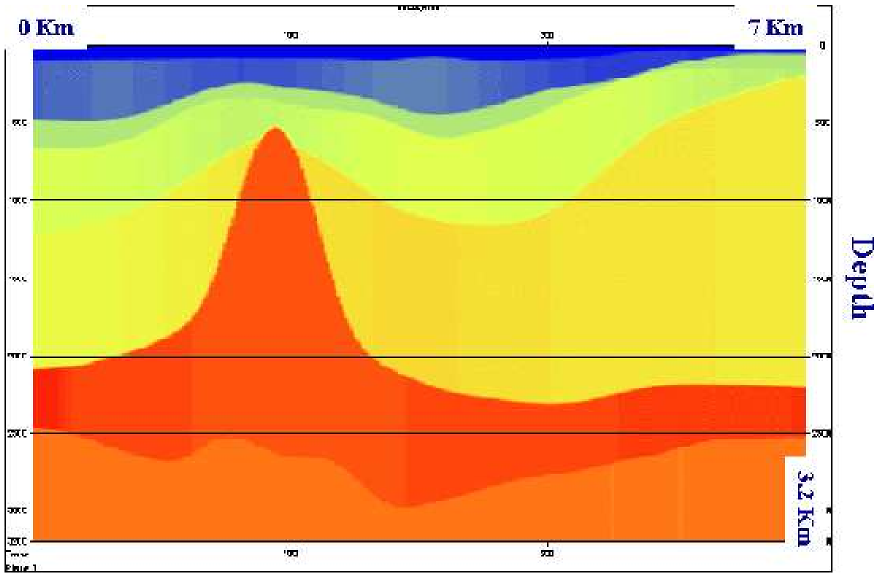

FIG. 5.1. *2D slice of a synthetic 3D subsurface velocity model. This is the field that PSPI migration should reproduce.*

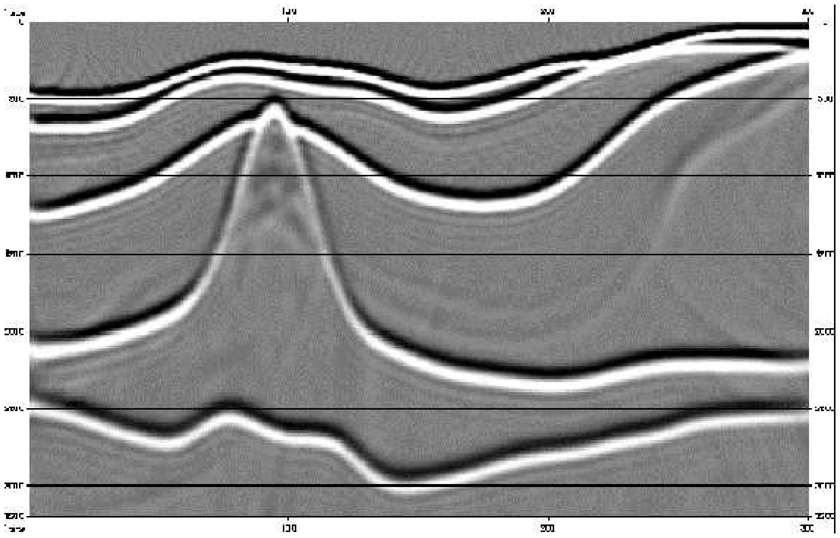

FIG. 5.2. *2D slice of 3D image generated by PSPI migration using velocity model from Fig.5.1.*

The resulting migrated section is shown in Fig.5.2. The complex structure of the subsurface model is largely reconstructed. Agreement between reflecting interfaces picked up by the PSPI migration and discontinuities of the velocity field is very good.

6. Parallel PSPI on the Origin2000. The PSPI algorithm is intrinsically data parallel; thanks to decoupling in the frequency domain, the depth extrapolation of Eqs.(3.1) – (3.3) at each step consists of absolutely concurrent calculations. The imaging of Eq.(2.7) at each depth requires a sum (over frequency) of the results of the depth extrapolation; this is the only point at which inter-processor communications (or remote memory accesses) are required by the algorithm. Implementationally, the required operation is a reduction (global sum) on the elements of a distribution of 2D arrays, once at every depth.

The HPF code had initially appeared to be drastically penalized by the SGI Origin's architecture since it ran much faster on the SGI Power Challenge than on the Origin. Now new compilers and OS have brought impressive performance gains on the Origin2000, and we are back to a fast, scalable HPF code. In the meantime, we have ported the code to native OpenMP on the Origin and can compare performance between the two codes.

One problem posed by the OpenMP conversion was how to carry out a reduction sum element-by-element on a distribution of 2D arrays. Efficient OpenMP reduction is currently only defined for scalars. The reduction sum for an array in OpenMP must be carried out "by hand", using either critical regions or barriers to avoid conflicts between processors. On the other hand, HPF requires only the F90 intrinsic "sum" to define the reduction sum.

A much more important problem with OpenMP on the Origin2000 is that of data placement. Data placement, an intrinsic part of HPF programming, is important on the Origin2000, even when using the shared memory paradigm of OpenMP. While a shared-memory machine guarantees that all data is logically equally accessible to all processors, the cost of that access is not guaranteed to be uniform. In particular, this is the case on the distributed shared memory Origin2000. Its ccNUMA nature gives the cache coherency which assures the single-memory model, and yet the scalable architecture imposes varying memory access costs: less expensive for local access, more expensive for remote access (considered here as communication).

At the same time, OpenMP, used for distributing work on shared-memory machines, was not originally intended to handle data placement, and any data placement directive is currently outside the OpenMP standard. Such directives are furnished by vendors for their specific machines; on the Origin2000, the SGI directives form an extension of OpenMP. While such machine-specific extensions are necessary for optimizing code, their use inhibits the portability that the OpenMP standard was meant to provide.

Because of the data-parallel nature of our problem, the original HPF code was naturally structured for efficiency on distributed-memory machines. The conversion to OpenMP was undertaken so as to preserve the original distributed data structure of the HPF code; on the DSM architecture of the Origin2000 this should serve to guarantee that local memory accesses are favored over the costlier remote accesses and that scalability is thus enhanced.

In the absence of data placement directives in OpenMP, the SGI directives should have provided the means for controlling data placement and imposing the desired structure on the data. A "page" is the minimal granularity of memory space on the Origin, and "page_place", which is supposed to give the user control over placement of pages in memory, was the SGI directive of choice. The page granularity does not allow the fine control over data placement which one generally expects; for example, an array distributed among several processors is distributed by page, not by element. A user can thus be surprised by finding some elements (those which happen to overlap on the page allotted to another processor) residing on processors other than where they were thought to have been placed. This can be avoided by padding the array so as to separate data blocks by at least a page of memory space. However, this requires a level of hand tuning of the array that was left behind long ago on distributed memory machines! Unfortunately, even with this hand tuning, page_place was not operative on the Origin at the time of the conversion of our code from HPF.

So first-touch default placement was utilized. This is a convention which dictates that the processor which first "touches" (carries out some operation on) a datum will take the page containing that datum into its associated local memory. Simply initializing the array via a parallel do-loop whose iterates are distributed conveniently among the processors will achieve a desired distribution – to within the usual constraints imposed by the page granularity. (Again, page overlapping can be avoided between subarray blocks by accordingly padding the array.) Unfortunately, even the first-touch convention is not guaranteed to give the desired data placement if the machine is under heavy use. If the OS is swapping threads among processors while the first-touch initialization is taking place, a subarray block can wind up scattered, page by page, among several memories. On the SGI Origin2000, this can happen even when threads are "locked" to specific processors, because the OS can override, depending on machine use, the user-specified directives and environment variables.

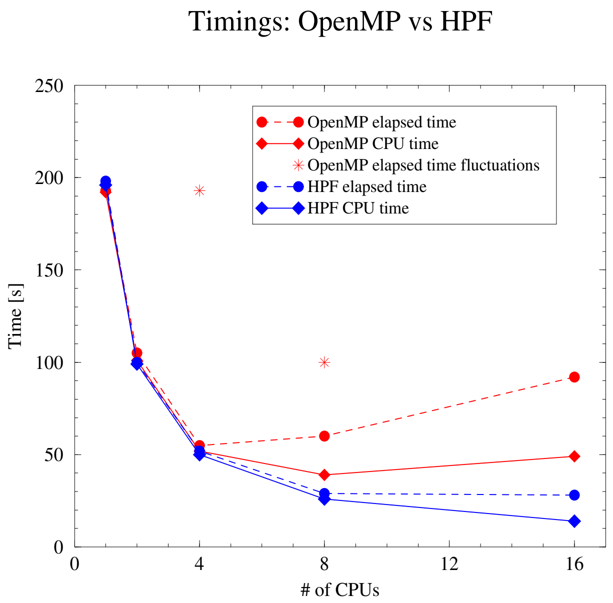

FIG. 6.1. *Timings: OpenMP vs HPF. Fluctuations in elapsed time for OpenMP runs correspond to heavy machine use. No such fluctuations are observed for HPF.*

On the other hand, HPF on the shared memory machine does not suffer from the same problems of data distribution nor of OS interference. Unlike OpenMP, HPF is not constrained to respect Fortran conventions of array storage across processor boundaries. (OpenMP must guarantee that array elements which are contiguous in Fortran remain contiguous in memory – even for distributed arrays and across processor boundaries.) Thus HPF is not influenced by page granularity, and the user can truly control data distribution. HPF procedures are also not subject to the "processor hopping" which characterizes OS control of OpenMP threads and which can penalize OpenMP performance when a machine is under heavy use.

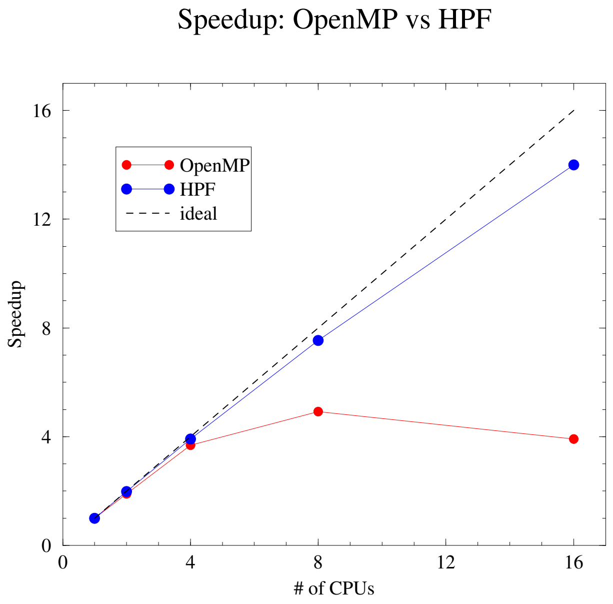

FIG. 6.2. *Speedup: OpenMP vs HPF. The HPF code scales well; the OpenMP code does not.*

Not surprisingly, then, OpenMP performance for this application on the Origin2000 is significantly inferior to HPF performance. One can see in figure 6.1 how the OpenMP version is penalized by the OS; the fluctuations in elapsed time, visible for some of the OpenMP runs, correspond to moments of heavy machine use and accompanying interference from the OS; one run with 16 CPUs which took 15 minutes to finish, is not even included in the figure. The HPF code never manifested fluctuations of this order. Figure 6.2 reports speedup measured using only the favorable runs for OpenMP. The HPF code scales well; the OpenMP code does not.

For the time being, we will continue our code development in PGHPF.

REFERENCES

[1] J. A. Scales, *Theory of Seismic Imaging*, notes for graduate courses, Colorado School of Mines, 1997 (`http://landau.Mines.EDU/~samizdat/imaging/index.html`).

[2] Özdoğan Yilmaz, *Seismic data processing*, Investigation in Geophysics, 2, Soc. Expl. Geophys., 1987.

[3] R. H. Stolt, "Migration by Fourier transform", *Geophysics* **43** (1978) 23–48.

[4] R. H. Stolt and A. K. Benson, "Seismic Migration – Theory and Practice", Handbook of Geophysical Exploration – Section I: Seismic Exploration, **5**, Elsevier Science Ltd., 1985.

[5] J. Gazdag, "Wave migration with the phase shift method", *Geophysics* **43** (1978) 1342–1351.

[6] J. Gazdag and P. Sguazzero, "Migration of seismic data", *Geophysics* **49** (1984) 124–131.

[7] C. Bagaini, E. Bonomi and E. Pieroni, "Data Parallel Implementation of 3D PSPI", in *Proc. 65th SEG Annual Meeting*, Oct. 1995, pp. 188–191.

[8] E. Bonomi, L. Brieger, C. Nardone and E. Pieroni, "PSPI: A Scheme for High-Performance Echo Reconstruction Imaging", *Computers in Physics*, **12** (1998) 126–132 .