Stanford Exploration Project, Report 72, December 18, 1997, pages 1–158

Stanford Exploration Project, Report 72, December 18, 1997, pages 1–158

Phase shift plus interpolation and split-step Fourier migration

Alexander M. Popovici[1]

keywords: *algorithm, velocity, Fourier transform*

ABSTRACT

Migration algorithms based on Fourier methods are naturally parallel. Two Fourier based migration algorithms (Phase Shift Plus Interpolation and Split-step) were implemented on the Connection Machine and tested on variable velocity 2-D and 3-D models. The migration results of the two methods and their run-times on the Convex and the Connection Machine were compared. For the three dimensional case a 33 times improvement in the run-time of the PSPI and a 14 times improvement in the run-time of the Split-step algorithm were obtained on the parallel computer.

INTRODUCTION

A year ago during the time of the SEG conference in San Francisco, Jon Claerbout was looking for a student inside SEP who would volunteer some time and examine the option to bring parallel computing to the Geophysics Department in Stanford. After a month or two of inspecting different vendors and architectures we narrowed the selection to two massively parallel computers: Maspar and the Connection Machine. At the end of December 1990 the final arrangements were made for the delivery of a CM-2. In the same month we installed our new server "Oas" planned to become the front end to the new parallel computer. In January 1991 the Thinking Machine Corporation people were busy installing the Darth Vader look alike Connection Machine.

Many students immediately started to write code for the new parallel machine, porting old programs or writing new algorithms which promised to show improved performance on the Connection Machine as compared to other serial computers. After almost a year in the middle of our hardware ranch, the CM has proved itself and is now routinely used by all the senior SEP students.

The introduction of parallel computers in geophysical data processing opens a new dimension in algorithm research. When designing a new algorithm we are usually

[1]**email:** not available

limited in our approach by the amount of time we estimate it will take the program to run on available resources. Approaches that will take weeks of computer time are usually excluded automatically as the run-time of a program is usually just the exponent of the total time involved in debugging and optimizing a new algorithm. When the total time to run a three dimensional migration program on the Connection Machine is less than the total time to run the two dimensional version on the Convex, our horizons are greatly expanded.

Migration algorithms in the frequency-wavenumber domain are special candidates for implementation on parallel computers as the frequency components are independent, allowing the computations to be done in parallel in surface coordinates. The three dimensional case offers a perfect mapping on the parallel computer memory, by storing the data in a serial fashion along the time dimension and in parallel on different processors in the surface coordinates. After Fourier transforming the data along the time axis, we can process each frequency independently, a slice at a time.

I selected two Fourier domain migration algorithms which handle lateral velocity variations. The Phase Shift Plus Interpolation algorithm presented by Gazdag (1984) and the Split-step Fourier method presented by Stoffa et al. (1990) were natural nominees. The two algorithms are extremely similar, and the three dimensional extension of the two dimensional version is quick and painless. I coded the two migration programs on the Convex first, then I wrote the parallel FORTRAN 90 version of the code to run on the Connection Machine. I compared the migration results of the two programs on two dimensional and three dimensional synthetic models. I studied the execution time of the program kernels on the CM and evaluated the Mflop/s performance and examined the total run-time of the programs on the Convex versus the CM.

THE PHASE SHIFT PLUS INTERPOLATION ALGORITHM

The derivation of the Phase Shift Plus Interpolation (PSPI) algorithm starts with the scalar wave equation

$$\frac{\partial^2 p}{\partial z^2} + \frac{\partial^2 p}{\partial x^2} = \frac{1}{v^2}\frac{\partial^2 p}{\partial t^2} \tag{1}$$

where $p = p(x, z, t)$ is the pressure field and $v = v(x, z)$ is the earth velocity.

The pressure field $p(x, z, t)$ is a finite function and can be therefore expressed as a double Fourier series

$$p(x, z, t) = \sum_{k_x} \sum_{\omega} P(k_x, z, \omega) e^{i(k_x x - \omega t)}. \tag{2}$$

Substituting the equation (2) in equation (1) we obtain

$$\sum_{k_x} \sum_{\omega} [\frac{\partial^2 P(k_x, z, \omega)}{\partial z^2} - k_x^2 P(k_x, z, \omega) + \frac{\omega^2}{v^2(x, z)} P(k_x, z, \omega)] e^{i(k_x x - \omega t)} = 0$$

which after considering only the equation inside the square parenthesis becomes

$$\frac{\partial^2 P(k_x, z, \omega)}{\partial z^2} = (k_x^2 - \frac{\omega^2}{v^2(x,z)}) P(k_x, z, \omega) \tag{3}$$

valid for all the values of k_x and ω. The problem is that in this form, the x coordinate in the pressure field is Fourier transformed and there is no direct correspondence between a point in the medium of coordinates (x, z), the velocity $v(x, z)$, and the corresponding value of $p(x, z, t)$ at that location.

For a laterally invariant velocity and small values of dz, we can consider the velocity constant for a thin depth interval. We write

$$k_z = \pm[\frac{\omega^2}{v^2} - k_x^2]^{\frac{1}{2}} \tag{4}$$

where k_z is constant for two given values of k_x and ω. Equation (3) becomes an ordinary differential equation

$$\frac{\partial^2 P}{\partial z^2} = -k_z^2 P \tag{5}$$

which has the analytic solution

$$P(k_x, z + dz, \omega) = P(k_x, z, \omega) e^{ik_z \Delta z}. \tag{6}$$

Using equation (5) to downward extrapolate the wave field in laterally invariant velocity media $v(z)$ forms the basis of the classic Phase Shift algorithm (Gazdag, 1978).

For laterally variant media, Gazdag (1984) proposes to downward extrapolate the wave field one depth interval at a time with several velocities, then to inverse Fourier transform in x the extrapolated field and finally to interpolate each point along the X axis. In other words starting with equation (3) we consider several constant velocities $(v_1, v_2, ...)$ in the interval $[v_{min}, v_{max}]$ and downward extrapolate the wavefield $P(k_x, z, \omega)$ to $P(k_x, z + \Delta z, \omega)$ with each velocity. The resulting wavefields

$$(P_1(k_x, z + \Delta z, \omega), P_2(k_x, z + \Delta z, \omega), ...)$$

are then inverse Fourier transformed in the direction of the X axis to obtain

$$(P_1^*(x, z + \Delta z, \omega), P_2^*(x, z + \Delta z, \omega), ...).$$

To obtain a single downward extrapolated wavefield, for each point of X coordinates associated with a velocity $v(z, x)$, the value of the resulting wavefield in this point is interpolated between the two wavefields with closest velocities.

In addition to this idea, Gazdag (1984) implements a technique in the PSPI algorithm to ensure that all the zero dips (corresponding to the case $k_x = 0$) are downward continued without distortion. The technique consists of multiplying the wavefield $P(x, z, \omega)$ with

$$e^{-i\frac{\omega}{v(x,z)}\Delta z}$$

prior to the Fourier transformation along the X axis and multiplying the downward extrapolated wavefield $P_j(k_x, z + \Delta z, \omega)$ by

$$e^{i\frac{\omega}{v_j}\Delta z}$$

where the subscript j denotes the index of the constant velocity used in the downward extrapolation step. As it will be seen later in the Split-step Fourier method, this supplemental phase subtractieon and addition improves the accuracy of the results.

THE SPLIT-STEP FOURIER METHOD

Stoffa et al. (1990) present an alternative to the PSPI, replacing the downward continuation step with two or more velocities by a single average velocity followed by a phase shift of the wavefield with a perturbation term to account for lateral velocity variations. The Split-step algorithm is based on splitting the space variant slowness ($s = \frac{1}{v(x,z)}$) into a constant term and a perturbation term,

$$s(x, z) = s_0(z) + \Delta s(x, z),$$

where $s_0(z)$ is a reference slowness defined as the average slowness in a depth interval. The wave equation (1) is Fourier transformed along the time axis to become

$$\frac{\partial^2 P(x, z, \omega)}{\partial z^2} + \frac{\partial^2 P(x, z, \omega)}{\partial x^2} + \omega^2 s^2(x, z) P(x, z, \omega) = 0. \tag{7}$$

After inserting the slowness split into the perturbation term and the average term the equation is transformed into

$$\frac{\partial^2 P(x, z, \omega)}{\partial z^2} + \frac{\partial^2 P(x, z, \omega)}{\partial x^2} + \omega^2 {s_0}^2(z) P(x, z, \omega) = -\omega^2 [2 s_0(z) \Delta s(x, z) + \Delta s^2(x, z)] P(x, z, \omega). \tag{8}$$

Noting the right side of the equation

$$S(x, z, \omega) = \omega^2 (2 s_0(z) \Delta s(x, z) + \Delta s^2(x, z)) P(x, z, \omega)$$

equation (8) becomes

$$\frac{\partial^2 P(x, z, \omega)}{\partial z^2} + \frac{\partial^2 P(x, z, \omega)}{\partial x^2} + \omega^2 {s_0}^2(z) P(x, z, \omega) = -S(x, z, \omega) \tag{9}$$

which is an inhomogeneous wave equation with a source term $S(x, z, \omega)$.

Stoffa et al. show that equation (9) can be integrated over a thin depth layer Δz by ignoring the contribution of the $\Delta s^2(x, z)$. Equation (9) is Fourier transformed in surface coordinates after the second order term of the slowness perturbation is dropped and subsequently integrated over the depth layer Δz. After inverse Fourier transformation back into surface coordinates the solution has the form

$$P(x, z_{n+1}, \omega) = P_0(x, z_{n+1}, \omega) e^{i\omega \Delta s(x,z) \Delta z}, \tag{10}$$

where $P_0(x, z_{n+1}, \omega)$ represents the wavefield downward continued with the average slowness $s_0(z)$. Though the mathematical path is very different from the one Gazdag (1984) followed, the solution is very similar if you consider Gazdag's algorithm with a single velocity.

The phase subtraction and addition trick in Gazdag's algorithm is replaced by multiplication with

$$e^{i(\frac{1}{v_{med}} - \frac{1}{v(x,z)})\omega\Delta z}$$

after the downward extrapolation. Compared with Gazdag's trick to phase shift with

$$e^{-i\frac{\omega}{v(x,z)}\Delta z}$$

followed by

$$e^{i\frac{\omega}{v_j}\Delta z},$$

the only difference is that Gazdag (1984) does the first phase shift before the Fourier transform while Stoffa et al. (1990) do it after the Fourier transform. Though analytically not identical in the general case, the qualitative idea is the same.

The two algorithms are compared in Figure 1.

IMPLEMENTATION ON THE CONNECTION MACHINE

The Fourier domain migration algorithms are easily parallelizable algorithm (after the Fourier transformation) because the computations are done a depth slice at a time, therefore naturally mapping on a parallel computer architecture. Moreover the frequency axis maps into the processor memory, allowing easy independent manipulation of each frequency. For a 3-dimensional survey, the spatial coordinates are mapped in parallel on a 2-D slice of processors, while the time dimension is mapped into the memory of each processor.

The downward extrapolation is done for each frequency independently and the imaging summation is performed inside each processor. The FFT subroutine is part of Connection Machine Scientific Software Library (CMSSL) and therefore presumably highly optimized, saving the programmer the annoyance of coding his own parallel 2-D and 3-D transforms. The code is extremely condensed, adding to the already compressed formulation of the Fortran 90 language.

A simplified version of the actual program is provided to give a flavor of the simplicity of algorithm representation. The subroutine `fft_time` executes a FFT over the time axis, while the subroutines `fft_xy` and `fft_inv_xy` perform forward and inverse FFTs in surface coordinates. All the variables terminated in "CM" are Connection Machine parallel arrays. The phase multiplication is done using the array `phasorCM` and the wave extrapolation field slice for different velocities is stored in the parallel array `PkyzCM`.

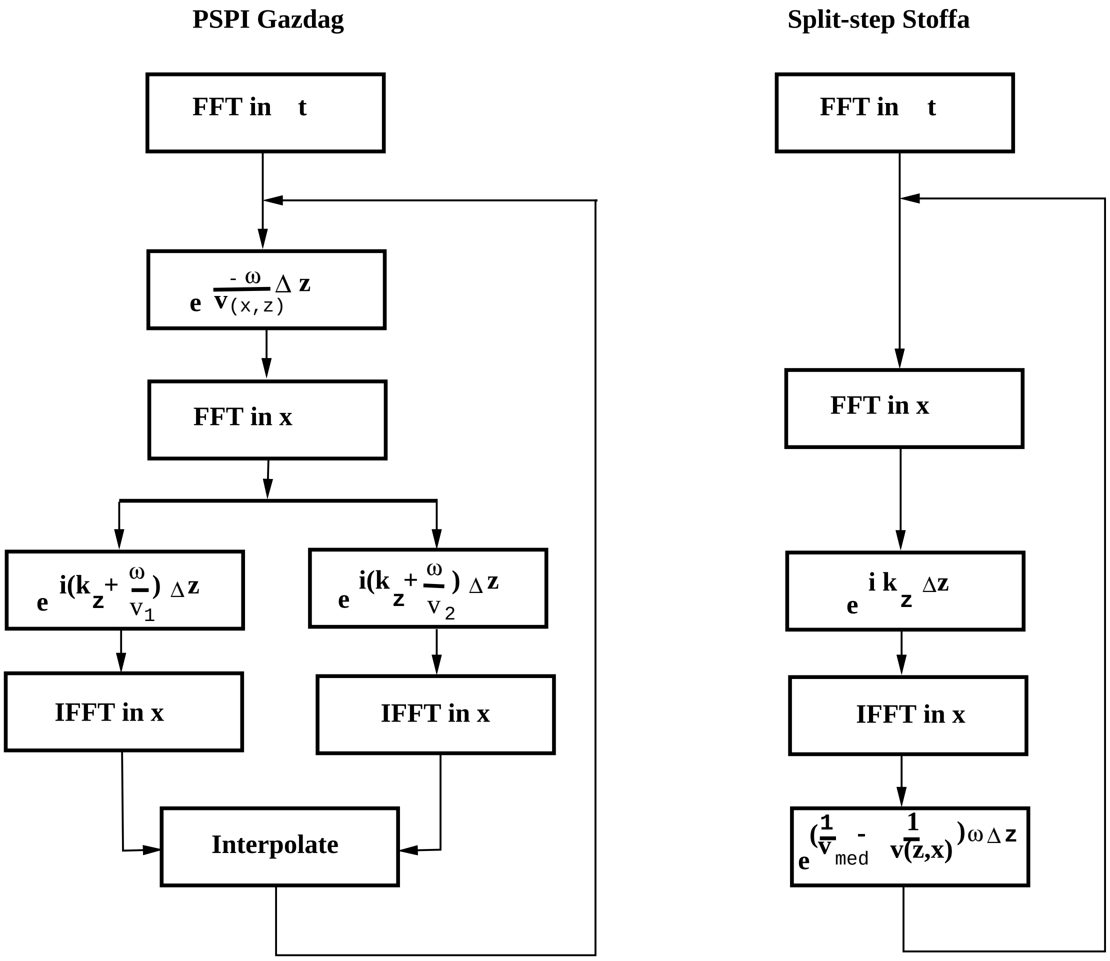

Figure 1: The two Fourier migration algorithms. a) Phase Shift Plus Interpolation (PSPI). b) Split-step Fourier Migration. $\boxed{\text{mihai2-gazsplit}}$ [NR]

```
C Fourier transform the time axis
        call fft_time(nt2,nx2,ny2,PxywCM)

C Start here loop in iz
        do iz=1,nz
           call CMF_fe_array_to_cm(velplaneCM,velmodel(1,1,iz))
C Pick minimum vel(1) and maximum vel(2) velocities
           vel(1)=MINVAL(velplaneCM)
           vel(2)=MAXVAL(velplaneCM)
           deltv=vel(2)-vel(1)
           rdeltv=1./deltv
C Zero the migration table
          migdataCM=0.
C Start loop in frequencies
          do iw=1,nwo2
            w=(iw-1)*dw
            if(iw.eq.1) then
               phasorCM=cmplx(cos((iw-1)*dw),sin((iw-1)*dw))
               akzCM=-dw*dz/velplaneCM
               dphasorCM=cmplx(cos(akzCM),sin(akzCM))
            else
               phasorCM=phasorCM*dphasorCM
            endif
C Multiply by exp(iw*dz/v(x,y,z))
            PxywCM(iw,:,:)=PxywCM(iw,:,:)*phasorCM
          enddo
C FFT over X and Y
          call fft_xy(nt2,nx2,ny2,PxywCM)
          do iw=1,nwo2
            w=(iw-1)*dw
C Start extrapolation loop  (nv=number of velocities)
            do iv=1,nv
              rv=1./vel(iv)
              r2v=rv*rv
              akzCM=w*w*r2v-akxkyCM
              where(akzCM.ge.0.)
                akzCM=-sqrt(akzCM)
                PkyzCM(iv,:,:)=PxywCM(iw,:,:)*cexp(cmplx(0.,(akzCM+w*rv)*dz) )
              elsewhere
                PkyzCM(iv,:,:)=0.
              end where
            enddo
C Inverse FFT in X  for all velocities (nv=number of velocities in layer)
            call fft_inv_xy(nx2,ny2,PkyzCM)
C Use linear interpolation in REAL and IMAGINARY (faster)
            PxywCM(iw,:,:)=(PkyzCM(1,:,:)*(vel(2)-velplaneCM)+
     &      PkyzCM(2,:,:)*(velplaneCM-vel(1)))*rdeltv
C Image for t=0.
            migdataCM=migdataCM+PxywCM(iw,:,:)
C End here loop in iw
          enddo
          call CMF_fe_array_from_cm(mdata,real(migdataCM))
          call rite(outfd,mdata,nx*ny*esize)
      enddo
```

```
C Here ends big loop in iz
```

The Split-step Fourier method can be easily obtained by minimally modifying the PSPI algorithm. The difference is that the downward extrapolation is done with a single velocity and the perturbation slowness phase shift is performed after the downward extrapolation step. Computationally the major difference between PSPI and Split-step is that PSPI performs three Fourier transforms inside the frequency loop while Split-step performs only two. On the Connection Machine this is reflected accurately in the run-time ratio of the two programs. I get a ratio of 1.45 to 1 between the execution time of the kernels of the two programs (158.01 seconds for PSPI and 108.77 seconds for Split-step). Oddly enough, on the Convex I get a ratio of 7.28 to 1 between the two PSPI and Split-step methods.

```
C Fourier transform in time
        call fft_time(nt2,nx2,ny2,PxywCM)
C Start here loop in iz
        do iz=1,nz
          call CMF_fe_array_to_cm(velplaneCM,velmodel(1,1,iz))
C Find average velocity
          vel=SUM(velplaneCM)/(nx2*ny2)
          rv=1./vel
          r2v=rv*rv
C Zero the migration table
          migdataCM=0.
C FFT over X and Y
          call fft_xy(nt2,nx2,ny2,PxywCM)
C Start loop in frequencies
          do iw=1,nwo2
            w=(iw-1)*dw
C Downward extrapolate with average velocity
            akzCM=w*w*r2v-akxkyCM
            where(akzCM.ge.0.)
              akzCM=-sqrt(akzCM)
              PxywCM(iw,:,:)=PxywCM(iw,:,:)*cexp(cmplx(0.,akzCM*dz) )
            elsewhere
              PxywCM(iw,:,:)=0.
            end where
          enddo
C Inverse FFT in X and Y
          call fft_inv_xy(nt2,nx2,ny2,PxywCM)
          do iw=1,nwo2
            w=(iw-1)*dw
C Apply phase shift with slowness perturbation
            PxywCM(iw,:,:)=PxywCM(iw,:,:)*cexp(cmplx(0.,(rv-1./velplaneCM)*w*dz))
C Image for t=0.
            migdataCM=migdataCM+PxywCM(iw,:,:)
C End here loop in iw
          enddo
          call CMF_fe_array_from_cm(mdata,real(migdataCM))
          call rite(outfd,mdata,nx*ny*esize)
```

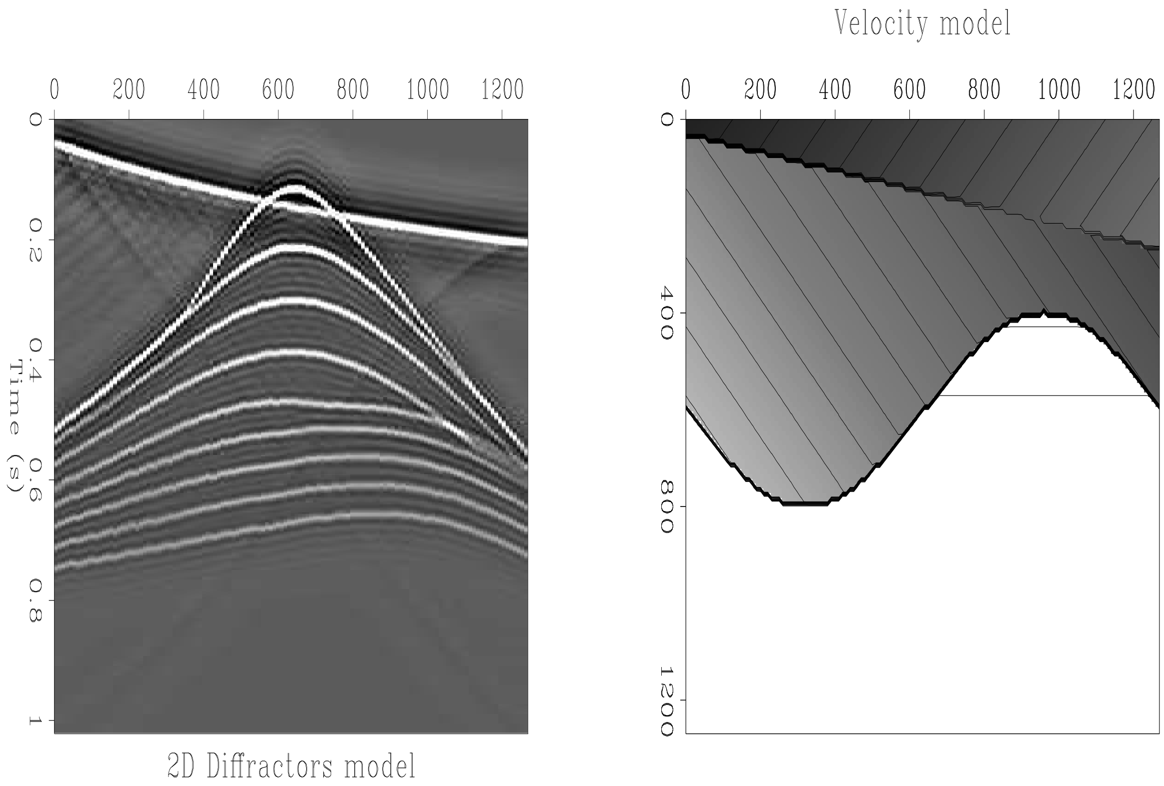

Figure 2: The two dimensional diffraction model and the velocity model. mihai2-diffvel2d [ER]

```
C Here ends big loop in iz
      enddo
```

A 2 dimensional example

The two dimensional velocity model in Figure 2 is composed of a dipping linear interface and a curved interface. Above the first interface, the velocity has a horizontal and a vertical gradient. There is a velocity discontinuity at the location of the dipping event and another velocity jump under the curved boundary. The structural events are a dipping interface at the velocity boundary and nine diffractors positioned at equal depth intervals.

In Figure 2 the diffraction model is shown on a 128x256 grid, together with the velocity model. The model is obtained by simulating the dipping event as an assembly of point scatterers. The travel-times to each point scatterer are calculated using an adaptive step finite-differences travel-time algorithm (Van Trier and Symes, 1991). Running several velocity models and comparing the migration results between the two methods, the PSPI appears to perform better than then Split-step in the presence of lateral velocity discontinuities. This difference is to be expected as the error in the

Split-step algorithm is proportional with the second order slowness perturbation. In media with smooth lateral velocity changes the two methods offer very similar results.

In Figure 3, the second diffractor from the top is less focused in the migration result of the Split-step algorithm, due to the velocity discontinuity along the dipping event. PSPI handles better velocity discontinuities and even though it has the option of splitting the velocity interval in more than two velocities and it appears that very little is gained by using more velocities for the downward extrapolation step.

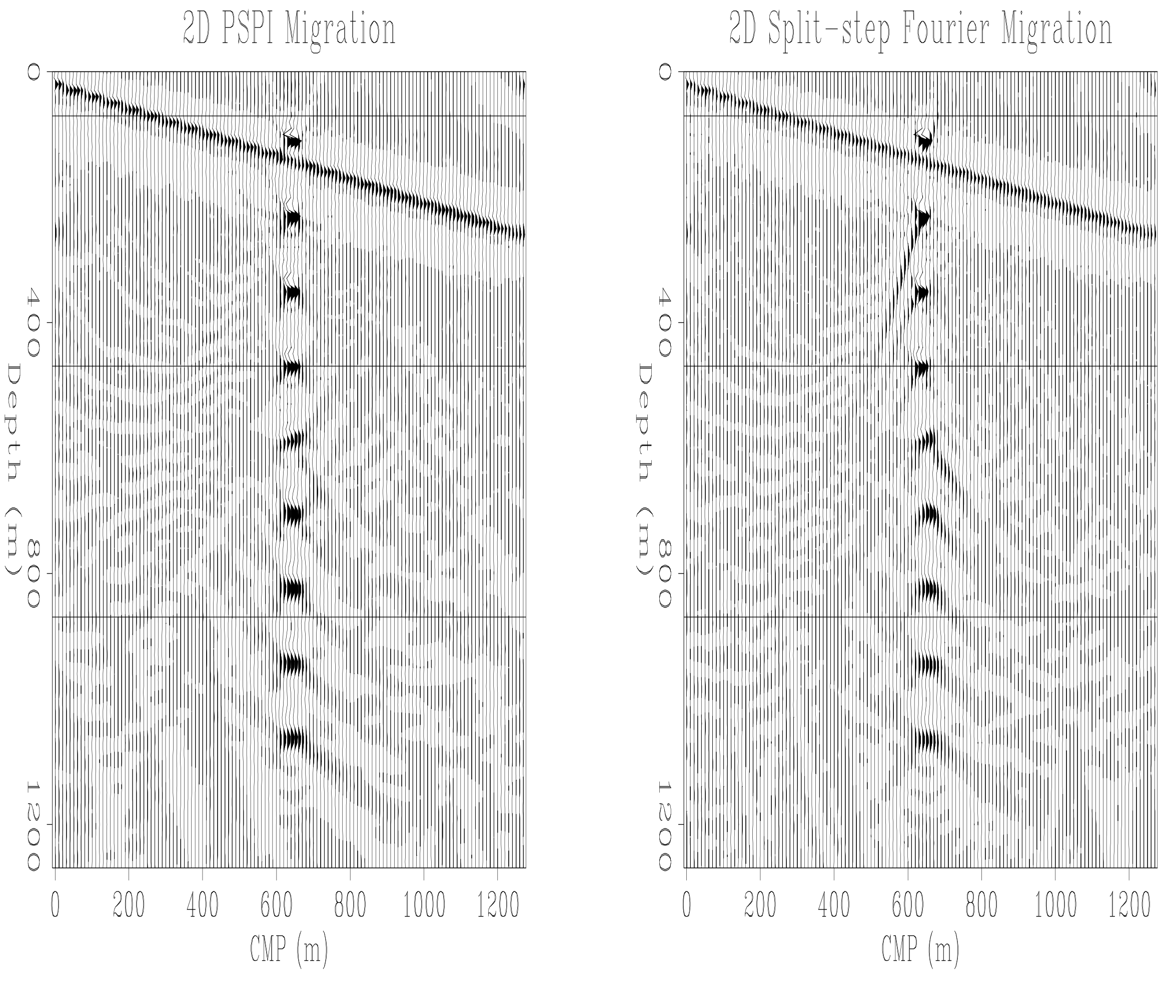

Figure 3: P
hase Shift Plus Interpolation (PSPI). [**RIGHT**] Split-step Fourier Migration.]2-D comparison of the two Fourier migration algorithms. [**LEFT**] Phase Shift Plus Interpolation (PSPI). [**RIGHT**] Split-step Fourier Migration. mihai2-pspisplit [ER]

A 3 dimensional example

The three dimensional model is built in a linearly varying velocity in all three coordinates: $v(x, y, z) = v_0 + ax + by + cz$. The model is composed of three vertical letters,

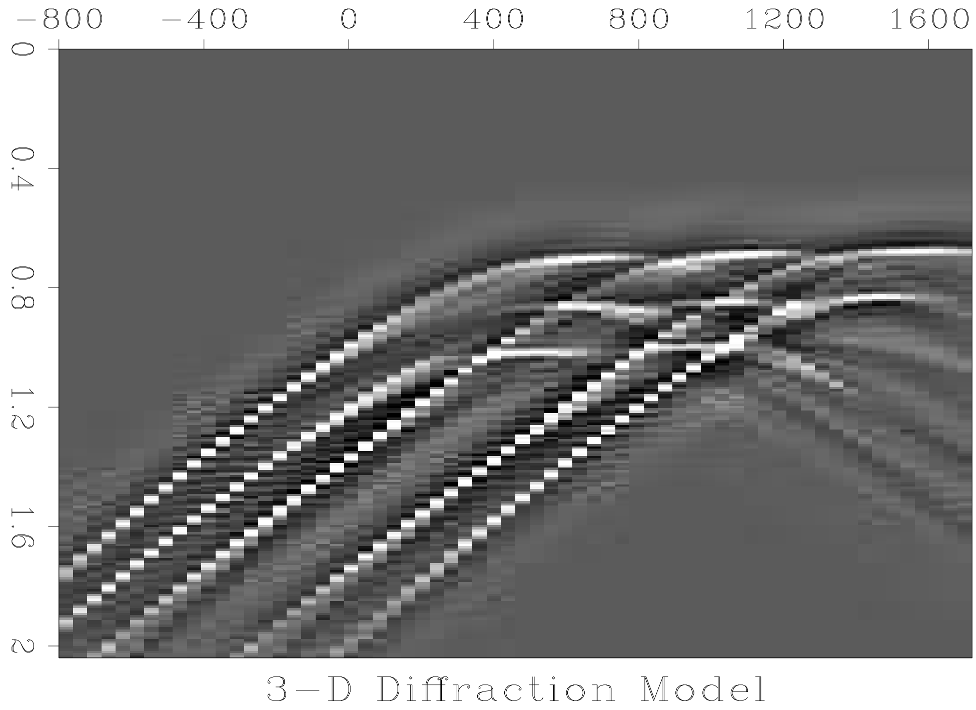

Figure 4: The three dimensional diffraction cube (64x64x256). Each side of the cube displays a section through the middle of the data set. mihai2-cube3Dmodel [ER]

in a plane parallel with the front of the cube. The size of the grid (64x64x256) was imposed by the amount of memory available on the SEP Connection Machine. Figure 4 displays the 3-D cube of diffraction data while Figure 5 illustrates two dimensional sections through the center of the data set.

The results of the 3-D PSPI migration program are displayed in Figure 6. I display the data cube such that on each side of the cube is revealed a section through the center of the data set. On the cube's front side the section shown is passing through the plane where the diffracting letters are positioned. In Figure 7 I show a section through the migrated data passing through the target plane of the letters.

TIMINGS ON THE SEP CONNECTION MACHINE

The 3-D PSPI and Split-step were implamented on the SEP Connection Machine. The SEP CM-2 has 8K serial bit processors, with 64 Kbytes of local memory in each processor. For every 32 serial bit processors there is a Floating Point Unit (FPU) responsible for all the arithmetic operation. The full machine has a total of 256 FPUs, which can be considered the processing nodes of the parallel computer. Both programs ran on the full 8K machine (256 FPU).

The PSPI kernel took 133 seconds to run on a 64x64x256 data model to produce a 64x64x64 depth migration model. The total CM-2 time was 238.4 seconds. The

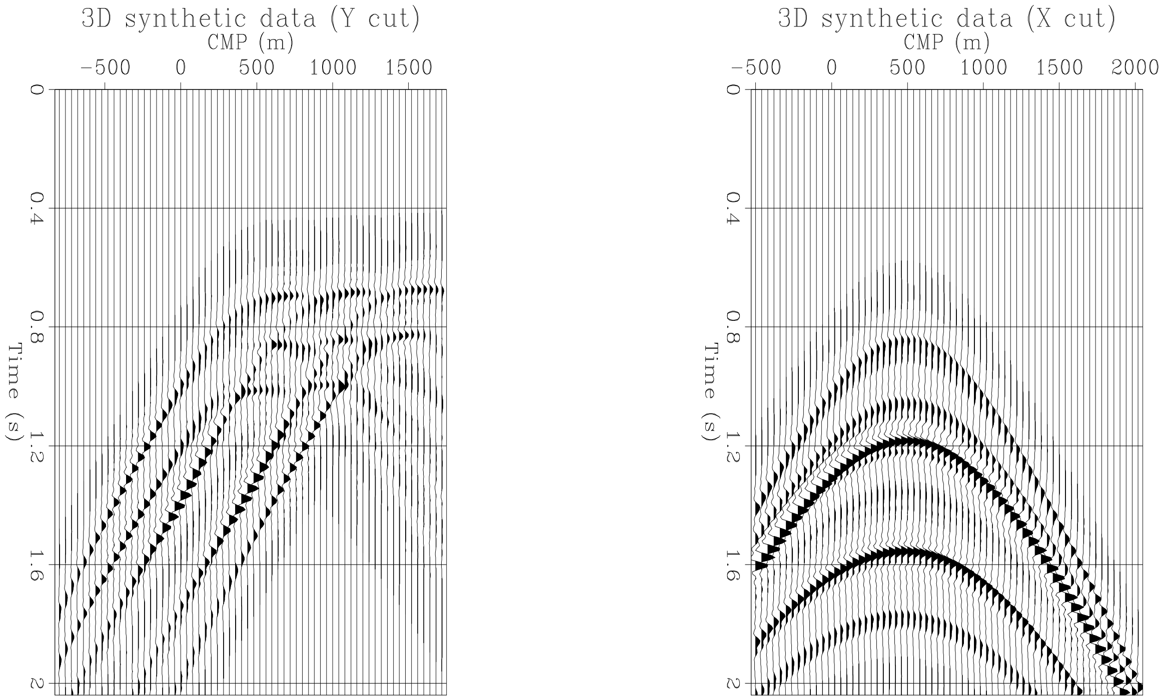

Figure 5: Two cuts through the three dimensional diffraction cube. mihai2-model3Dcuts [ER]

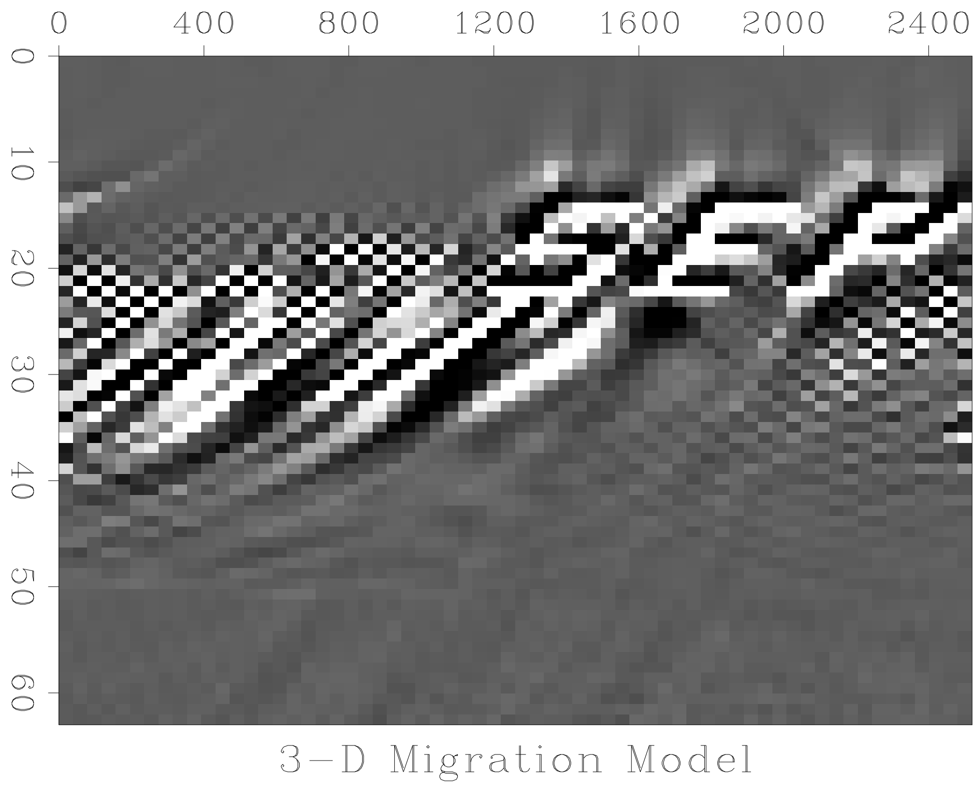

Figure 6: The three dimensional migration cube (64x64x64). mihai2-cube3Dmig [ER]

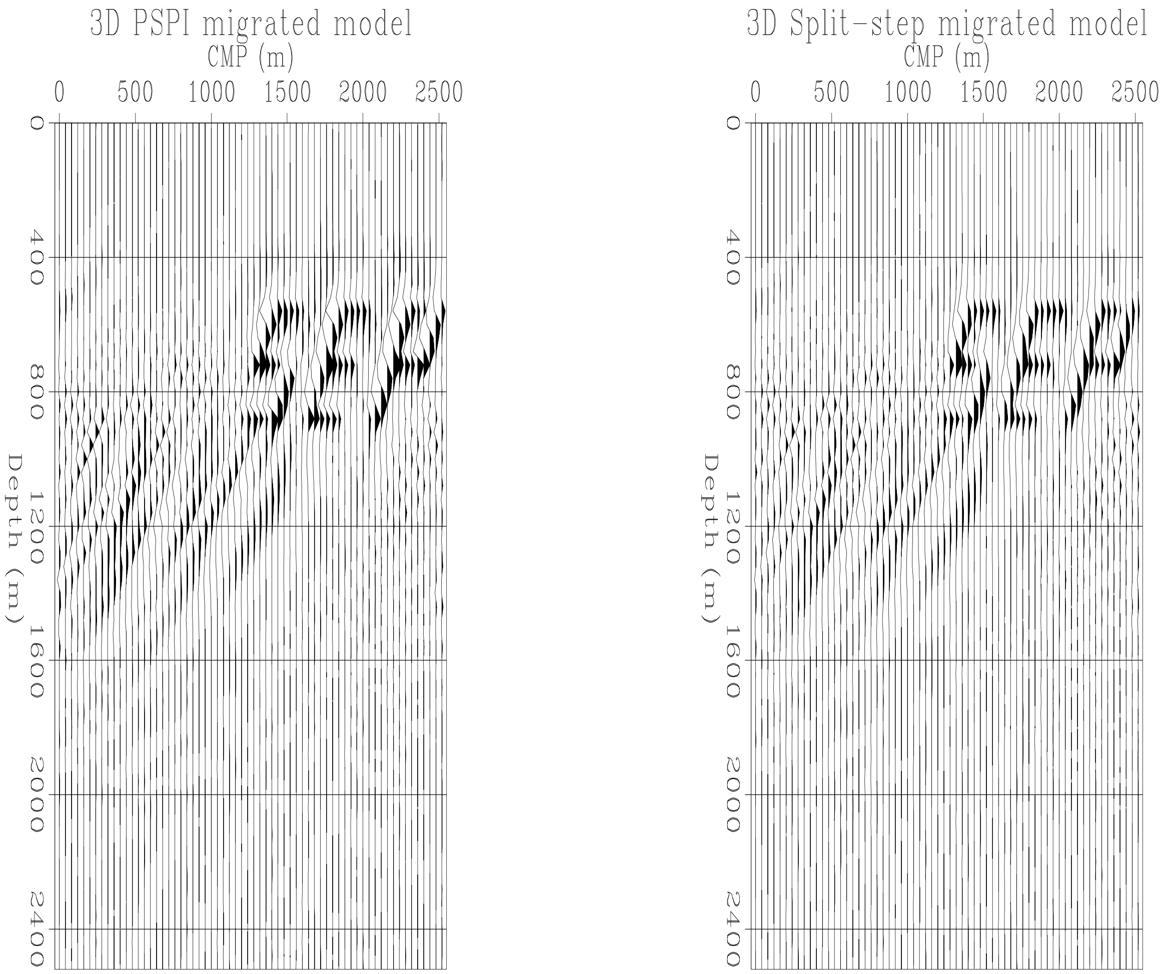

Figure 7: P
SPI migrated data. [RIGHT] Split-step migrated data.]3-D migration comparison: [LEFT] PSPI migrated data. [RIGHT] Split-step migrated data. mihai2-pspisplit3Dcut [ER]

kernel time corresponds to a rate of 177.08 Mflop/s (Mflop = million floating point operations). The Split-step kernel, running on the same data set, takes 108.7 seconds corresponding to a 124.7 Mflop/s rate.

In comparison, on the Convex, the total run time for the PSPI algorithm is 7966.1 seconds single CPU time, or 2:12:46 (2 hours, 12 minutes and 46 seconds), which with a load average of 3 or 4 the normal load average at SEP during report time, translates into more than 6 hours of waiting for the results. The ratio between the total time on the Convex and the total time on the Connection Machine for the PSPI algorithm is around 33 to 1. The Split-step algorithm runs for 1877.1 seconds single CPU time on the Convex, or 31 minutes and 17 seconds. The ratio between the total time on the Convex and the total time on the Connection Machine for the Split-step algorithm is 14 to 1.

Theoretically, the run-time ratio between the PSPI algorithm and the Split-step algorithm should approach 1.5 as PSPI executes three Fourier transforms in the inner loop while Split-step executes only two. The run-time ratio on the Connection Machine approaches this theoretical limit ratio, while on the Convex I have a peculiar ratio of 4.25 between the PSPI and Split-step algorithms.

CONCLUSIONS

I selected two Fourier domain migration algorithms, Phase Shift Plus Interpolation and Split-step, and implemented them on a serial computer (Convex) and a parallel computer (Connection Machine). I compared the migration results of the two methods on 2-D and 3-D laterally varying velocity models. The PSPI handles lateral velocity discontinuities better, but the run time tends to be 1.5 times higher than for the Split-step method. For smoothly varying lateral velocity, the two algorithms produce very similar results and therefore the faster one (Split-step) should be preferred. The run-times on the Convex and the Connection Machine are examined and a 33 to 1 improvement in run-time for the parallel version of the PSPI and a 14 to 1 improvement in run-time for the parallel version of the Split-step algorithm are found for the three dimensional model.

ACKNOWLEDGMENTS

Special thanks to Biondo Biondi, former SEP-er now working for Thinking Machine Corporation, who guided my parallel approach and assisted me in all the phases of developing and running the Connection Machine programs. Special thanks to Karl Schleicher from Halliburton Geophysical Services, who shared with me some of his wide Geophysical knowledge on how to improve migration algorithms.

REFERENCES

Biondi, B., 1991, Wave equation algorithms on massively parallel computers: SEP–**70**, 59–71.

Gazdag, J., 1978, Wave equation migration with the phase shift method: Geophysics, **43**, 1342–1351.

Gazdag, J., and Sguazzero, P., 1984, Migration of seismic data by phase shift plus interpolation: Geophysics, **49**, 124–131.

Stoffa, P. L., Fokkema, J. T., de Luna Freire, R. M., and Kessinger, W. P., 1990, Split-step Fourier Migration: Geophysics, **55**, 410–421.

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