$k_\tau(\tau) \equiv -\omega\sqrt{1-v^2(\tau)(k_x^2+k_y^2)/(2\omega)^2}$. The symbols $\supset$ and $\subset$ follow Bracewell's notations for Fourier transform. The temporal Fourier transform was moved into an inner loop to reduce I/O between the host and the vector processor. The limit on the vector operation is,

$$\omega > \frac{v(\tau)\sqrt{k_x^2+k_y^2}}{2\sqrt{1-\tau^2/t_{\max}^2}} ,$$

(Levin, 1983), to avoid wraparounds in τ, so k_τ is always real.

The weakness of the phase-shift method is its inability to deal with lateral velocity variations. The $f-k$ migration methods (of Stolt (1978) and Gazdag) are attractive enough that a considerable effort has been made to improve their handling of velocity variations: Fabio Rocca and other authors proposed the use of residual migration (Rocca and Salvador, 1982; Rothman, Levin and Rocca, 1984). Gazdag and Sguazzero (1984) proposed multiple velocity migration. Coordinate stretch may also be applicable.

Program (1) spends most of its computation time doing vector operations: *VMUL* and $\sum$. The phase shift costs $C \times nx \times ny \times n\tau \times nt$ floating point operations, where C is a constant, (about 10, because of the complex multiplication and the generation of the exponent-square-root). The FFT's have the relatively low cost of $nt \times nx \times ny \times log\,(nt \times nx^2 \times ny^2)$. Each *VMUL* requires about nt multiplications and $3nt$ memory accesses. Our FPS (120B) at Stanford can do up to 12 megaflops (12 million floating point operations per second) with its 167 nsec cycle and two arithmetic units. For this program, (as usual) the speed is memory limited and we might get only up to 1 megaflop with our 333 nsec memory and 3 memory accesses per operation. A full size problem ($nx=ny=256, n\tau=512, nt=1024$) is about 300 giga operations. It will take about 100 hours on our good old AP†.

PHASE-SHIFT MIGRATION WITH MATRIX PROCESSING

By putting many adders and multipliers together (systolic system), one may push the 12 megaflops limit up to, say 1 gigaflop. If 1 gigaflop could be reached, then the 3-D phase-shift migration could be done in 5 minutes, but can we feed that systolic monster? we couldn't use even the 12 megaflops of our FPS because of the I/O, let alone 1 gigaflop. The systolic system will be useless if we perform program (1) as it is.

† Actually, our system has two I/O bottle-necks outside the AP, (tape/disk-host and host-AP), so the time may be one or two orders of magnitude longer than 100 minutes.

The problem with program 1 is that it uses matrix-vector operations: the wave-field at the depth τ is

$$\begin{aligned} P_\tau(k_x, k_y, \omega) &= e^{i\Delta\tau k_\tau(\tau)} P_{\tau-\Delta\tau}(k_x, k_y, \omega) \\ &= e^{i\Delta\tau k_\tau(\tau)} e^{i\Delta\tau k_\tau(\tau-\Delta\tau)} \cdots e^{i\Delta\tau k_\tau(\Delta\tau)} P_0(k_x, k_y, \omega) \\ &= e^{i\Delta\tau \sum_{\theta=\Delta\tau}^{\tau} k_\tau(\theta)} P_0(k_x, k_y, \omega) \ . \end{aligned}$$

The image is

$$\begin{aligned} Q(k_x, k_y, \tau) &= \sum_\omega P_\tau(k_x, k_y, \omega) \\ &= \sum_\omega e^{i\Delta\tau \sum_{\theta=\Delta\tau}^{\tau} k_\tau(\theta)} P_0(k_x, k_y, \omega) \ . \end{aligned}$$

In a matrix form, for each k_x and k_y ,

$$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ Q(\tau) \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot\ \cdot & \cdot & \cdot\ \cdot \\ \cdot\ \cdot & \cdot & \cdot\ \cdot \\ \cdot\ \cdot & \cdot & \cdot\ \cdot \\ \cdot\ \cdot & \exp[i\Delta\tau \sum_{\theta=\Delta\tau}^{\tau} k_\tau(\theta)] & \cdot\ \cdot \\ \cdot\ \cdot & \cdot & \cdot\ \cdot \\ \cdot\ \cdot & \cdot & \cdot\ \cdot \\ \cdot\ \cdot & \cdot & \cdot\ \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ P_0(\omega) \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

The matrix-vector multiplication requires $N^2+2\times N$ memory accesses and N^2 operations if $N\times N$ is the size of the square matrix; the ratio of floating operations to I/O is about 1. A matrix-matrix multiplication would require N^3 operations and $3\times N^2$ memory accesses; The ratio is $N/3$. The object is therefore to matricize program 1: to do it by matrix-matrix multiplications. It can be done in the following manner,

Program (2):input:Zero-offset data, $p(x, y, t)$. output:Exploding reflectors, $q_2(x, y, \tau)$.

```
    $p(x, y, t) \supset P(k_x, k_y, \omega)$                    /* 3-D FFT */
    For all $k_x$                                              /* First-pass */
        $q_1(k_x, k_y, \tau') = \sum_\omega e^{-i\omega\Delta\tau \sum_{\theta=\Delta\tau}^{\tau'} \sqrt{1-v^2(\theta)k_x^2/\omega^2}} P(k_x, k_y, \omega)$
    For all $k_x$ and $k_y$                                    /* An extra 1-D FFT */
        $Q_1(k_x, k_y, k_{\tau'}) \subset q_1(k_x, k_y, \tau')$
    For all $k_y$                                              /* Second-pass */
```

$$Q_2(k_x, k_y, \tau) = \sum_{k_{\tau'}} e^{-ik_{\tau'} \Delta\tau \sum_{\theta=\Delta\tau}^{\tau} \sqrt{1-v^2(\theta) k_y^2 / k_{\tau'}^2}} \, Q_1(k_x, k_y, k_{\tau'})$$

$Q_2(k_x, k_y, \tau) \subset q_2(x, y, z)$ /* 2-D FFT */

The number of operations in program (2) is about twice that of program (1). The point is, however, that each of the passes is a series of matrix-matrix multiplications. For example, the first pass is a matrix-matrix multiplication for every k_x because the operator is independent of k_y: $q_1(k_x, k_y, \tau')$ is a $n\tau'$ by nk_y matrix (k_x is kept constant), the phase-shift matrix ($\exp[-i\,\omega\Delta\tau\sum\sqrt{1-v^2 k_x^2/\omega^2}]$), is a $n\tau'$ by $n\omega$ matrix, and $P(k_x, k_y, \omega)$ is a $n\omega$ by nk_y matrix, (again, k_x is kept constant). So the phase shift is done by $nk_x + nk_y$ matrix-matrix multiplications in program (2) instead of $nk_x \times nk_y$ matrix-vector multiplications in program (1). The cost of building the phase-shift matrix requires fewer operations ($n\tau' \times n\omega$) than the multiplication itself ($n\tau' \times nk_y \times n\omega$), but building that matrix might take a comparable time if it has to be done outside the matrix processor. (It may be vectorized via lookup tables).

I have not yet shown that programs (1) and (2) are equivalent. I could claim that program (2) is actually an implementation of 3-D migration by two-pass 2-D migration as suggested by Gibson et. al. (1983), and the proof was already given by Jakubowicz and Levin, (1983). But since the proof is short and simple I'll give it here.

For constant velocity, $v = 2$,

$$Q(\tau) = \int d\omega e^{i\tau\sqrt{\omega^2 - k_x^2 - k_y^2}} P(\omega) ,$$

from program (1). And

$$q_1(\tau') = \int d\omega e^{i\tau' \sqrt{\omega^2 - k_x^2}} P(\omega) , \qquad \text{/* First pass */}$$

$$Q_1(k_{\tau'}) = \int d\tau' \, e^{-i\tau' k_{\tau'}} q_1(\tau') , \qquad \text{/* Extra FFT */}$$

$$Q_2(\tau) = \int dk_{\tau'} \, e^{i\tau\sqrt{k_{\tau'}^2 - k_y^2}} Q_1(k_{\tau'}) , \qquad \text{/* Second pass */}$$

from program (2). Now, $Q = Q_2$ because,

$$Q_2(\tau) = \int dk_{\tau'} \, e^{i\tau\sqrt{k_{\tau'}^2 - k_y^2}} \int d\tau' \, e^{-i\tau' k_{\tau'}} \int d\omega e^{i\tau' \sqrt{\omega^2 - k_x^2}} P(\omega)$$

$$= \int d\omega \int dk_{\tau'} \, e^{i\tau\sqrt{k_{\tau'}^2 - k_y^2}} \int d\tau' \, e^{i\tau' [\sqrt{\omega^2 - k_x^2} - k_{\tau'}]} P(\omega) .$$

But,

$$\int d\tau' \; e^{i\tau' a} = \delta(a) \; .$$

So,

$$Q_2(\tau) = \int d\omega \int dk_{\tau'} \; e^{i\tau\sqrt{k_{\tau'}^2 - k_y^2}} \delta(\sqrt{\omega^2 - k_x^2} - k_{\tau'}) P(\omega)$$

$$= \int d\omega e^{i\tau\sqrt{\omega^2 - k_x^2 - k_y^2}} P(\omega)$$

$$= Q(\tau) \; .$$

For varying velocities there is an error but the error is small for typical velocity functions; Gibson et. al. (1983) found that,

> *As applied to examples of synthetic data from inhomogeneous media, the scheme introduces errors well below those attributable, in practice, to uncertainties in migration velocity.*

CONCLUSIONS

3-D migration by two-pass phase-shift can be done as a sequence of matrix-matrix multiplications. The two passes take twice as many operations as one pass takes, and there is an error in reflector positioning, but the error in reflector positioning is believed to be acceptable and the factor of two in the number of operation is compensated by the ability to use the full power of systolic matrix processing units: a full size problem that would take 100 hours on a vector processor should take 10 minutes on a matrix processor.

REFERENCES

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.

Gibson, B., Larner, K., and Levin, S., 1983, Efficient 3-D migration in two steps: Geophysical Prospecting, **31**, 1-33.

Jakubowicz, H., and Levin, S., 1983, A simple exact method of 3-D migration - theory, Geophysical Prospecting, **31**, 34-56.

Levin, S., 1983, Avoiding artifacts in phase shift migration: SEP, **37**, 27-35.

Levin, S., 1984, Parallel space-time migration: SEP, **38**, 207-214.

Rocca, F., and Salvador, L., 1982, Residual migration: Presented at the 52nd Annual International SEG Meeting, Dallas.

Rothman, D., Levin, S., and Rocca, F., 1985, Residual migration: Applications and limitations: Geophysics, **50**, 110-126.

Stolt, R.H., 1978, Migration by Fourier transform: Geophysics, **43**, 23-48.