A fast butterfly algorithm for the hyperbolic Radon transform

Jingwei Hu, Sergey Fomel, The University of Texas at Austin, Laurent Demanet, Massachusetts Institute of Technology, and Lexing Ying, The University of Texas at Austin*

SUMMARY

We introduce a fast butterfly algorithm for the hyperbolic Radon transform commonly used in seismic data processing. For two-dimensional data, the algorithm runs in complexity $O(N^2 \log N)$, where N is representative of the number of points in either dimension of data space or model space. Using a series of examples, we show that the proposed algorithm is significantly more efficient than conventional integration.

INTRODUCTION

In seismic data processing, the Radon transform (RT) (Radon, 1917) is a set of line integrals that maps mixed and overlapping events in seismic gathers to a new transformed domain where they can be separated (Gardner and Lu, 1991). The line integrals can follow different curves; straight lines (linear RT or slant stack), parabolas (parabolic RT), or hyperbolas (hyperbolic RT or velocity stack) are most commonly used. A major difference between these transforms is that the former two are time-invariant whereas the latter is time-variant. When the curves are time-invariant, the transform can be performed efficiently in the frequency domain by the Fourier transform shift theorem. On the contrary, the hyperbolic Radon transform has to be computed in the time domain, which is not feasible in general due to the large size of seismic data. Nevertheless, the hyperbolic transform is often preferred as it better matches the true seismic events. Based on the special properties of the Radon operator, many approaches were proposed to speed up the computation in the time domain (Thorson and Claerbout, 1985; Sacchi, 1996; Cary, 1998; Trad et al., 2002).

In this work, we construct a fast butterfly algorithm to efficiently evaluate the hyperbolic Radon transform. As opposed to the conventional, expensive *velocity scan* (i.e., direct integration + interpolation), our method provides an accurate approximation in only $O(N^2 \log N)$ operations for 2D data. Here N depends only on the range of the parameters and can often be chosen small compared to the problem size. The adjoint of the hyperbolic transform can be implemented similarly without extra difficulty.

The Radon transform has been widely used to separate and attenuate multiple reflections (Hampson, 1986; Yilmaz, 1989; Foster and Mosher, 1992; Herrmann et al., 2000; Moore and Kostov, 2002; Hargreaves et al., 2003; Trad, 2003). By introducing the fast solver, our hope is to improve the inversion process either iteratively or directly.

The rest of the paper is organized as follows. We first introduce the low-rank approximation and the butterfly structure; then using these building elements, we construct our fast algorithm. Numerical examples of both synthetic and field data are presented next to illustrate the accuracy and efficiency of the proposed algorithm.

ALGORITHM

Assume $d(t,h)$ is a function in the data domain, then a hyperbolic Radon transform R maps d to function $(Rd)(\tau,p)$ in the model domain (Thorson and Claerbout, 1985),

$$(Rd)(\tau,p) = \int d(\sqrt{\tau^2+p^2h^2},h)\,dh. \tag{1}$$

Here t is the time, h is the offset, τ is the intercept, and p is the slowness. Fixing (τ,p), the hyperbola $t=\sqrt{\tau^2+p^2h^2}$ describes the traveltime for the event; hence integration along these curves can be used to identify different reflections.

We adopt a different point of view to construct the algorithm by reformulating the transform (1) as a double integral,

$$(Rd)(\tau,p) = \iint \hat{d}(\omega,h)e^{2\pi i\omega\sqrt{\tau^2+p^2h^2}}\,d\omega\,dh, \tag{2}$$

where $\hat{d}(\omega,h)$ is the Fourier transform of $d(t,h)$ in the t variable. Discretizing (2) in the (ω,h) domain, one obtains

$$(Rd)(\tau,p) = \sum_{\omega,h} e^{2\pi i\omega\sqrt{\tau^2+p^2h^2}}\hat{d}(\omega,h). \tag{3}$$

For simplicity, we first perform a linear transformation to map (τ,p) to $\mathbf{x}=(x_1,x_2)\in X=[0,1]^2$, and (ω,h) to $\mathbf{k}=(k_1,k_2)\in K=[0,1]^2$: $\tau=(\tau_{\max}-\tau_{\min})x_1+\tau_{\min}$, $p=(p_{\max}-p_{\min})x_2+p_{\min}$; $\omega=(\omega_{\max}-\omega_{\min})k_1+\omega_{\min}$, $h=(h_{\max}-h_{\min})k_2+h_{\min}$. If we define input (source) $f(\mathbf{k})=\hat{d}(\omega(k_1),h(k_2))$, output (target) $u(\mathbf{x})=(Rd)(\tau(x_1),p(x_2))$, and the phase function $\Phi(\mathbf{x},\mathbf{k})=\omega(k_1)\sqrt{\tau(x_1)^2+p(x_2)^2h(k_2)^2}$, then (3) becomes

$$u(\mathbf{x}) = \sum_{\mathbf{k}\in K} e^{2\pi i\Phi(\mathbf{x},\mathbf{k})}f(\mathbf{k}), \quad \mathbf{x}\in X, \tag{4}$$

which falls into the general discretized form of Fourier integral operators. Our algorithm for computing the summation in equation (4) follows that of Candès et al. (2009). Readers are referred there for detailed mathematical exposition.

Low-rank approximations

The magnitude of the phase $\Phi(\mathbf{x},\mathbf{k})$ determines the degree of oscillation of the kernel $e^{2\pi i\Phi(\mathbf{x},\mathbf{k})}$. Let N be an integer power of two, which is on the order of the maximum of $|\Phi(\mathbf{x},\mathbf{k})|$ for $\mathbf{x}\in X$ and $\mathbf{k}\in K$. The exact choice of N depends on the desired efficiency and accuracy of the algorithm, and several examples will be given in the numerical results. The design of the fast algorithm relies on the key observation that this kernel, properly restricted to subdomains of $\mathbf{x}$ and $\mathbf{k}$, admits accurate and low-rank separated approximations; i.e., if A and B are two

DOI http://dx.doi.org/10.1190/segam2012-0462.1

square boxes in X and K, with sidelengths $w(A)$, $w(B)$ obeying $w(A)w(B) \le 1/N$, then

$$e^{2\pi i\Phi(\mathbf{x},\mathbf{k})} \approx \sum_{t=1}^{r_\varepsilon} \alpha_t^{AB}(\mathbf{x})\beta_t^{AB}(\mathbf{k}), \quad \mathbf{x}\in A,\ \ \mathbf{k}\in B,$$

where the number r_ε of expansions is independent of N for fixed error ε. Furthermore, it is shown that this low-rank approximation can be constructed by a tensor-product Chebyshev interpolation of $e^{2\pi i\Phi(\mathbf{x},\mathbf{k})}$ in the $\mathbf{x}$ variable when $w(A) \le 1/\sqrt{N}$ and in the $\mathbf{k}$ variable when $w(B) \le 1/\sqrt{N}$. Specifically, when $w(B) \le 1/\sqrt{N}$, α_t^{AB} and β_t^{AB} are given by

$$\alpha_t^{AB}(\mathbf{x}) = e^{2\pi i\Phi(\mathbf{x},\mathbf{k}_t^B)}, \tag{5}$$

$$\beta_t^{AB}(\mathbf{k}) = e^{-2\pi i\Phi(\mathbf{x_0}(A),\mathbf{k}_t^B)} L_t^B(\mathbf{k}) e^{2\pi i\Phi(\mathbf{x_0}(A),\mathbf{k})}; \tag{6}$$

when $w(A) \le 1/\sqrt{N}$, α_t^{AB} and β_t^{AB} are given by

$$\alpha_t^{AB}(\mathbf{x}) = e^{2\pi i\Phi(\mathbf{x},\mathbf{k}_0(B))} L_t^A(\mathbf{x}) e^{-2\pi i\Phi(\mathbf{x}_t^A,\mathbf{k}_0(B))}, \tag{7}$$

$$\beta_t^{AB}(\mathbf{k}) = e^{2\pi i\Phi(\mathbf{x}_t^A,\mathbf{k})}. \tag{8}$$

Here $\mathbf{x_0}(A)$ and $\mathbf{k_0}(B)$ denote the center of the boxes A and B. $L_t^B(\mathbf{k})$ is the 2D Lagrange interpolation on the Chebyshev grid $\mathbf{k}_t^B$:

$$L_t^B(\mathbf{k}) = \left(\prod_{s_1=0,s_1\neq t_1}^{q_{k_1}-1} \frac{k_1-k_{s_1}^B}{k_{t_1}^B-k_{s_1}^B}\right)\left(\prod_{s_2=0,s_2\neq t_2}^{q_{k_2}-1} \frac{k_2-k_{s_2}^B}{k_{t_2}^B-k_{s_2}^B}\right),$$

with $\mathbf{k}_t^B = \{(k_{t_1}^B, k_{t_2}^B) \mid \mathbf{k}_0(B) + w(B)(z_{i_1}, z_{i_2}),\ 0 \le i_1 \le q_{k_1} - 1,\ 0 \le i_2 \le q_{k_2}-1,\ r_\varepsilon = q_{k_1}q_{k_2}\}$, and $z_i = \frac{1}{2}\cos(\frac{i\pi}{q-1})$ is the 1D Chebyshev grid of order q on $[-1/2, 1/2]$. $L_t^A(\mathbf{x})$ and $\mathbf{x}_t^A$ are defined accordingly.

Butterfly structure

To realize the above idea, the butterfly algorithm (Michielssen and Boag, 1996; O'Neil and Rokhlin, 2007) turns out to be an appropriate tool. The main data structure underlying the algorithm is a pair of dyadic trees T_X and T_K. The tree T_X has $X = [0,1]^2$ as its root box (level 0) and is built by recursive, dyadic partitioning of X until level $L = \log N$, where the finest boxes are of sidelength $1/N$. The tree T_K is built similarly but in the opposite direction. Figure 1 shows such a partition for $N = 4$. A crucial property of this structure is that at arbitrary level l, the sidelengths of a box A in T_X and a box B in T_K always satisfy $w(A)w(B) = 1/N$. Thus, a low-rank approximation of the kernel $e^{2\pi i\Phi(\mathbf{x},\mathbf{k})}$ is available.

Fast butterfly algorithm

Our goal is to approximate the partial sum generated by the sources $\mathbf{k}$ inside any fixed box B: $u^B(\mathbf{x}) := \sum_{\mathbf{k}\in B} e^{2\pi i\Phi(\mathbf{x},\mathbf{k})} f(\mathbf{k})$.

1. *Initialization.* At level $l = 0$, let A be the root box of T_X. For each leaf box $B \in T_K$, equations (5, 6) are valid since $w(B) \le 1/\sqrt{N}$. Then for $\mathbf{x} \in A$, $u^B(\mathbf{x}) \approx \sum_{t=1}^{r_\varepsilon} e^{2\pi i\Phi(\mathbf{x},\mathbf{k}_t^B)} \delta_t^{AB}$, where $\delta_t^{AB} := \sum_{\mathbf{k}\in B} \beta_t^{AB}(\mathbf{k}) f(\mathbf{k})$ is given by

$$\delta_t^{AB} = e^{-2\pi i\Phi(\mathbf{x}_0(A),\mathbf{k}_t^B)} \sum_{\mathbf{k}\in B} \left(L_t^B(\mathbf{k}) e^{2\pi i\Phi(\mathbf{x}_0(A),\mathbf{k})} f(\mathbf{k})\right).$$

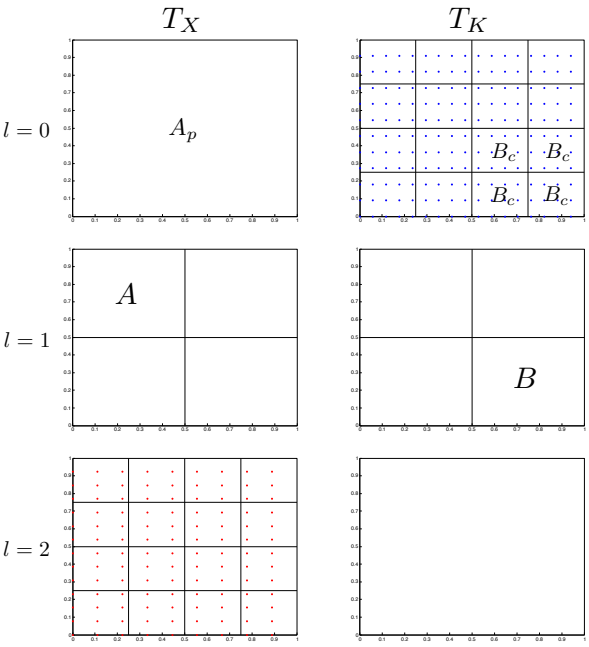

Figure 1: Butterfly structure for the special case of $N = 4$.

Due to the special form of α_t^{AB}, δ_t^{AB} can be treated as *equivalent sources* located at $\mathbf{k}_t^B$. We next aim at updating δ_t^{AB} until the end level L. This is done as follows.

2. *Recursion.* At $l = 1, 2, ..., L/2$, for each pair (A, B), let A_p be A's parent and $B_c, c = 1,2,3,4$ be B's children (see Figure 1). For each child, we have available from the previous level an approximation of the form

$$u^{B_c}(\mathbf{x}) \approx \sum_{t'=1}^{r_\varepsilon} e^{2\pi i\Phi(\mathbf{x},\mathbf{k}_{t'}^{B_c})} \delta_{t'}^{A_pB_c}, \quad \text{for } \mathbf{x}\in A_p.$$

Summing over all children gives

$$u^B(\mathbf{x}) \approx \sum_{c=1}^{4}\sum_{t'=1}^{r_\varepsilon} e^{2\pi i\Phi(\mathbf{x},\mathbf{k}_{t'}^{B_c})} \delta_{t'}^{A_pB_c}, \quad \text{for } \mathbf{x}\in A_p.$$

Since $A \subset A_p$, this is of course true for any $\mathbf{x} \in A$. On the other hand, $e^{2\pi i\Phi(\mathbf{x},\mathbf{k})}$ also has a low-rank approximation of equivalent sources at the current level,

$$u^B(\mathbf{x}) \approx \sum_{t=1}^{r_\varepsilon} e^{2\pi i\Phi(\mathbf{x},\mathbf{k}_t^B)} \delta_t^{AB}, \quad \text{for } \mathbf{x}\in A.$$

An easy calculation suggests

$$\delta_t^{AB} = \sum_{c=1}^{4}\sum_{t'=1}^{r_\varepsilon} \beta_t^{AB}(\mathbf{k}_{t'}^{B_c}) \delta_{t'}^{A_pB_c}.$$

Substituting β_t^{AB} in (6) yields

$$\delta_t^{AB} = e^{-2\pi i\Phi(\mathbf{x}_0(A),\mathbf{k}_t^B)} \sum_{c=1}^{4}\sum_{t'=1}^{r_\varepsilon} \left(L_t^B(\mathbf{k}_{t'}^{B_c}) e^{2\pi i\Phi(\mathbf{x}_0(A),\mathbf{k}_{t'}^{B_c})} \delta_{t'}^{A_pB_c}\right).$$

DOI http://dx.doi.org/10.1190/segam2012-0462.1

3. *Switch.* A switch of the representation to (7, 8) is needed at $l = L/2$ since (5, 6) is no longer valid as $l > L/2$. Indeed, we can set

$$\delta_t^{AB} = \sum_{s=1}^{r_\varepsilon} e^{2\pi i \Phi(\mathbf{x}_t^A, \mathbf{k}_s^B)} \delta_s^{AB},$$

where $\{\delta_t^{AB}\}$ denotes the new set of coefficients and $\{\delta_s^{AB}\}$ the old set.

4. *Recursion.* The rest of the recursion is analogous. For $l = L/2+1, ..., L$, we have

$$\delta_t^{AB} = \sum_{c=1}^{4} \sum_{t'=1}^{r_\varepsilon} \alpha_{t'}^{A_p B_c}(\mathbf{x}_t^A) \delta_{t'}^{A_p B_c}.$$

Substituting α_t^{AB} in (7) gives

$$\delta_t^{AB} = \sum_{c=1}^{4} e^{2\pi i \Phi(\mathbf{x}_t^A, \mathbf{k}_0(B_c))} \sum_{t'=1}^{r_\varepsilon} \left(L_{t'}^{A_p}(\mathbf{x}_t^A) e^{-2\pi i \Phi(\mathbf{x}_{t'}^{A_p}, \mathbf{k}_0(B_c))} \delta_{t'}^{A_p B_c} \right).$$

5. *Termination.* Finally we reach $l = L$, and B is the entire domain K. For each $\mathbf{x} \in A$,

$$u(\mathbf{x}) = u^B(\mathbf{x}) \approx \sum_{t=1}^{r_\varepsilon} \alpha_t^{AB}(\mathbf{x}) \delta_t^{AB}.$$

We thus set (after plugging in α_t^{AB} in (7))

$$u(\mathbf{x}) = e^{2\pi i \Phi(\mathbf{x}, \mathbf{k}_0(B))} \sum_{t=1}^{r_\varepsilon} \left(L_t^A(\mathbf{x}) e^{-2\pi i \Phi(\mathbf{x}_t^A, \mathbf{k}_0(B))} \delta_t^{AB} \right), \quad \mathbf{x} \in A.$$

Discussion

The main workload for the fast butterfly algorithm is in steps 2 and 4. For each level, there are N^2 pairs of boxes (A, B), and the operations between each A and B is a constant small number (depends on r_ε). Since there are $\log N$ levels, the total cost is $O(N^2 \log N)$. It is not difficult to see that step 3 takes $O(N^2)$, and steps 1 and 5 take $O(N_\omega N_h)$ and $O(N_\tau N_p)$ operations. Considering the initial Fourier transform of preparing data in the (ω, h) domain, we conclude that the overall complexity of the algorithm is roughly $O(N_t N_h \log N_t + C(r_\varepsilon) N^2 \log N + N_\omega N_h + N_\tau N_p)$. Normally $N_\omega = N_t$, but by symmetry and compactness of the Fourier transform $\hat{d}(\omega, h)$ (typical for most seismic data), we can significantly shorten the domain for ω, hence further reduce the computational load.

In comparison, the conventional *velocity scan* requires at least $O(N_\tau N_p N_h)$ computations, which quickly become a bottleneck as the problem size increases. Yet the efficiency of our algorithm is mainly controlled by $O(N^2 \log N)$, where N is roughly determined by the degree of oscillation of $e^{2\pi i \omega \sqrt{\tau^2 + p^2 h^2}}$, i.e., the range of τ, p, h, and ω. In practice, N can often be chosen smaller than N_τ, N_p, N_h, and N_ω.

There is no general rule to select N and the number of Chebyshev points $q_{k_1}, q_{k_2}, q_{x_1}, q_{x_2}$ (recall that $r_\varepsilon = q_{k_1} q_{k_2}$ or $r_\varepsilon = q_{x_1} q_{x_2}$). For numerical implementation, these parameters are tuned to achieve the best efficiency and accuracy.

NUMERICAL EXAMPLES

2D synthetic data

We start with a simple 2D example. Figure 2 shows a synthetic CMP gather sampled on $N_t \times N_h = 1000^2$. Figure 3 shows the result by the fast butterfly algorithm obtained on $N_\tau \times N_p = 1000^2$. For this problem, our method provides a reasonable result with $N = 32$, $q_{k_1} = q_{x_1} = 7$, $q_{k_2} = q_{x_2} = 5$ in only 1 second of CPU time, while the traditional *velocity scan* takes about 60 s, whose result is also shown here for comparison (Figure 4).

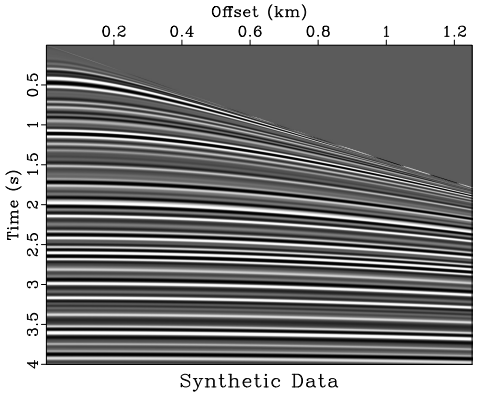

Figure 2: 2D synthetic CMP gather, $N_t \times N_h = 1000^2$.

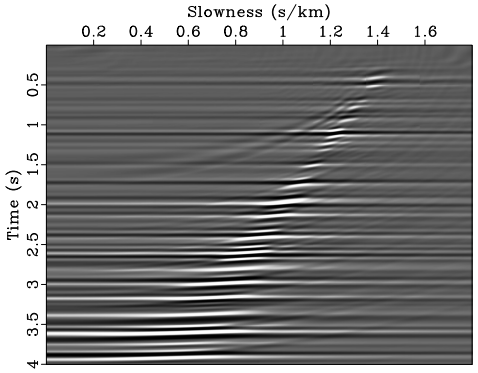

Figure 3: Output of the fast butterfly algorithm, $N_\tau \times N_p = 1000^2$, $N = 32$, $q_{k_1} = q_{x_1} = 7$, $q_{k_2} = q_{x_2} = 5$; CPU time: **1.26 s**.

2D field data

We now consider a real 2D seismic gather in Figure 5. The sampling sizes are $N_t = N_\tau = 1500$, $N_h = 240$, and $N_p = 2000$. Parameters are chosen as $N = 128$, $q_{k_1} = q_{x_1} = 7$, $q_{k_2} = q_{x_2} = 5$. Although the small N_h makes these data not very suitable for testing the fast algorithm, our method (~ 7 s) still outperforms the *velocity scan* (~ 43 s) with acceptable accuracy (Figure 6).

3D synthetic data

Since the above discussion does not require the input data to be uniform, our algorithm can be easily extended to handle the

DOI http://dx.doi.org/10.1190/segam2012-0462.1

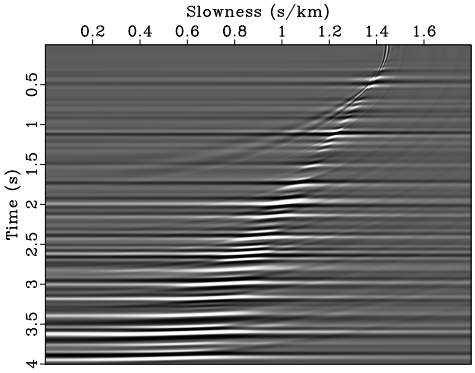

Figure 4: Output of the *velocity scan*, $N_\tau \times N_p = 1000^2$; CPU time: **61.36 s**.

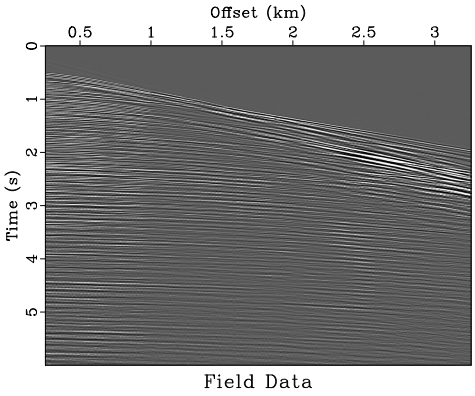

Figure 5: 2D real CMP gather, $N_t = 1500$, $N_h = 240$.

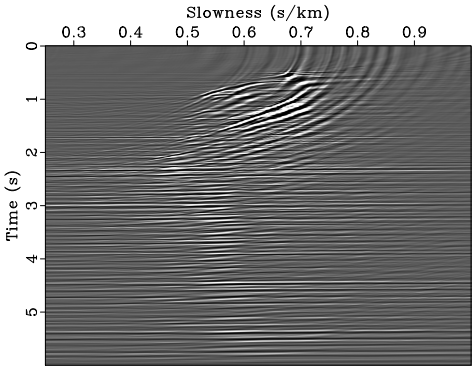

Figure 6: Output of the fast butterfly algorithm, $N_\tau = 1500$, $N_p = 2000$, $N = 128$, $q_{k_1} = q_{x_1} = 7$, $q_{k_2} = q_{x_2} = 5$; CPU time: **7.23 s** (ref: CPU time of *velocity scan* is **42.58 s**).

following problem:

$$(Rd)(\tau,p) = \iint d(\sqrt{\tau^2 + p^2(h_1^2+h_2^2)}, h_1, h_2)\, dh_1\, dh_2, \quad (9)$$

where $d(t,h_1,h_2)$ is a function in 3D rather than 2D. All we need is to introduce a new variable $h = \sqrt{h_1^2+h_2^2}$, and reorder the values $d(t,h_1,h_2)$ according to h. Figure 7 is such synthetic data sampled on $N_t \times N_{h_1} N_{h_2} = 1000 \times 128^2$. The output is obtained for $N_\tau \times N_p = 1000 \times 512$. The fast algorithm (Figure 8) runs in only 7 s for $N = 64$, $q_{k_1} = q_{x_1} = 5$, $q_{k_2} = q_{x_2} = 3$, while the *velocity scan* takes more than 500 s.

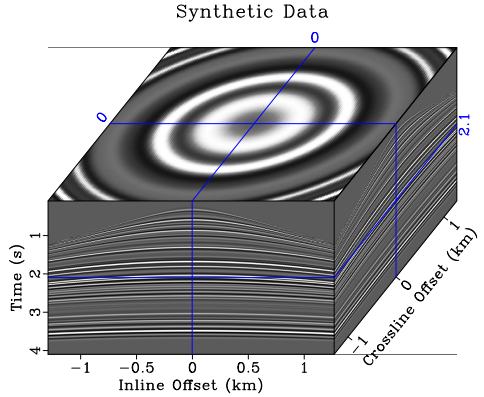

Figure 7: 3D synthetic CMP gather, $N_t = 1000$, $N_{h_1} = N_{h_2} = 128$.

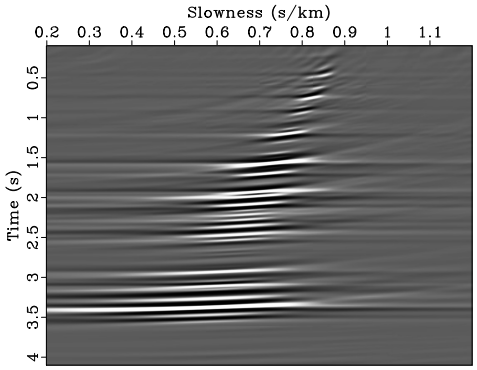

Figure 8: Output of the fast butterfly algorithm, $N_\tau = 1000$, $N_p = 512$, $N = 64$, $q_{k_1} = q_{x_1} = 5$, $q_{k_2} = q_{x_2} = 3$; CPU time: **7.34 s** (ref: CPU time of *velocity scan* is **515.25 s**).

CONCLUSIONS

We constructed a fast butterfly algorithm for the hyperbolic Radon transform. Compared with the time-consuming integration in the time domain, our method runs in only $O(N^2 \log N)$ operations, where N is representative of the number of points in either dimension of data space or model space. An ongoing work is to study the performance of this fast solver for the sparse iterative inversion of the Radon transform in application to multiple attenuation.

DOI http://dx.doi.org/10.1190/segam2012-0462.1

http://dx.doi.org/10.1190/segam2012-0462.1

EDITED REFERENCES

Note: This reference list is a copy-edited version of the reference list submitted by the author. Reference lists for the 2012 SEG Technical Program Expanded Abstracts have been copy edited so that references provided with the online metadata for each paper will achieve a high degree of linking to cited sources that appear on the Web.

REFERENCES

Candès, E., L. Demanet, and L. Ying, 2009, A fast butterfly algorithm for the computation of Fourier integral operators: Multiscale Modeling and Simululation, **7**, 1727–1750.

Cary, P. W., 1998, The simplest discrete Radon transform: 68th Annual International Meeting, SEG, Expanded Abstracts, 1999–2002.

Foster, D. J., and C. C. Mosher, 1992, Suppression of multiple reflections using the Radon transform: Geophysics, **57**, 386–395.

Gardner, G. H. F., and L. Lu, eds., 1991, Slant-stack processing: SEG Geophysics Reprint Series No. 14.

Hampson, D., 1986, Inverse velocity stacking for multiple elimination: Journal of the Canadian SEG, **22**, 44–55.

Hargreaves, N., B. verWest, R. Wombell, and D. Trad, 2003, Multiple attenuation using an apex-shifted Radon transform: 65th Conference and Exhibition, EAGE, Extended Abstracts.

Herrmann, P., T. Mojesky, M. Magesan, and P. Hugonnet, 2000, De-aliased, high-resolution Radon transforms: 70th Annual International Meeting, SEG, Expanded Abstracts, 1953–1956.

Michielssen, E., and A. Boag, 1996, A multilevel matrix decomposition algorithm for analyzing scattering from large structures: IEEE Transactions on Antennas and Propagation, **44**, 1086–1093.

Moore, I., and C. Kostov, 2002, Stable, efficient, high-resolution Radon transforms: 64th Conference and Exhibition, EAGE, Extended Abstracts, F034.

O'Neil, M., and V. Rokhlin, 2007, A new class of analysis-based fast transforms: Yale University Technical report YALEU/DCS/TR-1384.

Radon, J., 1917, Uber die Bestimmung von Funktionen durch ihre Integralwerte Langs Gewisser Mannigfaltigkeiten: Berichte uber die Verhandlungen der Sachsische Akademie der Wissenschaften [Reports on the proceedings of the Saxony Academy of Science], **69**, 262–277.

Sacchi, M., 1996, A bidiagonalization procedure for the inversion of time-variant velocity stack operator: CDSST Report, 73–92.

Thorson, J. R., and J. F. Claerbout, 1985, Velocity-stack and slant-stack stochastic inversion: Geophysics, **50**, 2727–2741.

Trad, D., 2003, Interpolation and multiple attenuation with migration operators: Geophysics, **68**, 2043–2054.

Trad, D., T. Ulrych, and M. Sacchi, 2002, Accurate interpolation with high resolution time-variant Radon transforms: Geophysics, **67**, 644–656.

Yilmaz, O., 1989, Velocity-stack processing: Geophysical Prospecting, **37**, 357–382.