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). By integrating + 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),

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

Here $t$ is the time, $h$ is the offset, $\tau$ is the intercept, and $p$ is the slowness. Fixing ($\tau, p$), the hyperbola $t = \sqrt{\tau^2 + p^2 h^2}$ 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 as a double integral,

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

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

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

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

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

which falls into the general discretized form of Fourier integral operators. Our algorithm for computing the summation in equation (3) follows that of Candès et al. (2009). Readers are referred there for detailed mathematical exposition.

Low-rank approximations

The magnitude of the phase $\Phi(x, k)$ determines the degree of oscillation of the kernel $e^{2\pi i \Phi(x,k)}$. Let $N$ be an integer power of two, which is on the order of the maximum of $|\Phi(x, k)|$ for $x \in X$ and $k \in K$. 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 $x$ and $k$, admits accurate and low-rank separated approximations; i.e., if $A$ and $B$ are two...
square boxes in $X$ and $K$, with sidelengths $w(A), w(B)$ obeying
$w(A)w(B) \leq 1/N$, then

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

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

$$\alpha_t^{AB}(x) = e^{2\pi i \Phi(x,x^t)},$$

$$\beta_t^{AB}(k) = e^{-2\pi i \Phi(x^t_0,k^t_0)} L_t^B(k) e^{2\pi i \Phi(x^t_0,k^t_0)};$$

when $w(A) \leq 1/\sqrt{N}$, $\alpha^{AB}$ and $\beta^{AB}$ are given by

$$\alpha_t^{AB}(x) = e^{2\pi i \Phi(x,k^t_0)} L_t^A(x) e^{-2\pi i \Phi(x,k^t_0)},$$

$$\beta_t^{AB}(k) = e^{2\pi i \Phi(x,k^t_0)}.$$

Here $x^t_0$ and $k^t_0$ denote the center of the boxes $A$ and $B$. $L_t^B(k)$ is the 2D Lagrange interpolation on the Chebyshev grid $k^t_0$.

$$L_t^B(k) = \frac{\lambda_t^{B}(k^t_0)}{\lambda_t^{B}(k^t_0)} \sum_{i=1}^{q_t^B} \frac{1}{\lambda_t^{B}(k^t_0)} \frac{k^t_0 - k^t_i}{k^t_0 - k^t_i},$$

where $k^t_0 = \{ (k^t_0, k^t_1, k^t_2) \mid |k^t_0 + k^t_i, |k^t_0|, 0 \leq i \leq k^t_0 - 1, 0 \leq t < q_t^B, r_x = q_k^B, \}$, and $z_i = \frac{1}{2} \cos(\frac{\pi}{q_t^B})$ is the 1D Chebyshev grid of order $q$ on $[-1/2,1/2]$. $L_t^A(x)$ and $L_t^B(k)$ are defined accordingly.

Butterfly structure

To realize the above idea, the butterfly algorithm [Michielsen 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 $e^{2\pi i \Phi(x,k^t_0)}$ is available.

Fast butterfly algorithm

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

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) are valid since $w(B) \leq 1/\sqrt{N}$. Then for $x \in A$, $u^B(x) \approx \sum_{t=1}^{r_x} e^{2\pi i \Phi(x,k^t_0)} \delta_t^{AB}$, where $\delta_t^{AB} := \frac{1}{\lambda_t^{B}(k^t_0)} \frac{k^t_0 - k^t_i}{k^t_0 - k^t_i}$. Since $A \subset A_p$, this is of course true for any $x \in A$. On the other hand, $e^{2\pi i \Phi(x,k^t_0)}$ also has a low-rank approximation of equivalent sources at the current level,

$$u^B(x) \approx \sum_{t=1}^{r_x} e^{2\pi i \Phi(x,k^t_0)} \delta_t^{AB},$$

An easy calculation suggests

$$\delta_t^{AB} = \frac{4}{r_x} \sum_{c=1}^{r_x} \sum_{r=1}^{r_x} \beta_t^{AB}(k^t_0), \delta_t^{AB} = \frac{4}{r_x} \sum_{c=1}^{r_x} \sum_{r=1}^{r_x} \beta_t^{AB}(k^t_0)e^{2\pi i \Phi(x^t_0,k^t_0)} \delta_t^{AB}.$$
Fast Radon Transform

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

$$
\delta_t^{AB} = \sum_{l=1}^{r_\varepsilon} e^{-2\pi i \Phi(x^l, k^l)} \delta_t^A
,$$

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

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

$$
\delta_t^{AB} = \frac{4}{r_\varepsilon} \left( \sum_{c=1}^{r_\varepsilon} \alpha_t^{AB}(x^c_A) \delta_t^{A^c} \right).
$$

Substituting $\alpha_t^{AB}$ in (1) gives

$$
\delta_t^{AB} = \sum_{c=1}^{r_\varepsilon} e^{-2\pi i \Phi(x^c, k_0(B))} \sum_{r'=1}^{r_\varepsilon} \left( L_t^{A^c}(x^c) e^{-2\pi i \Phi(x^r, k_0(B))} \delta_t^{A^c} \right).
$$

5. **Termination.** Finally we reach $l = 0$, and $B$ is the entire domain $K$. For each $x \in A$,

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

We thus set (after plugging in $\alpha_t^{AB}$ in (1))

$$u(x) = e^{-2\pi i \Phi(x, k_0(B))} \left( \sum_{c=1}^{r_\varepsilon} \left( L_t^{A^c}(x) e^{-2\pi i \Phi(x^c, k_0(B))} \delta_t^{A^c} \right) \right), \quad x \in A.
$$

**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 $r_\varepsilon$). Since there are log $N$ levels, the total cost is $O(N^3 \log N)$. It is not difficult to see that step 3 takes $O(N^3)$, and steps 1 and 5 take $O(N_\omega N_h)$ and $O(N_\omega N_p)$ operations. Considering the initial Fourier transform of preparing data in the $(\omega, h)$ domain, we conclude that the overall complexity of the algorithm is roughly $O(N_\omega N_h \log N_\varepsilon + C(r_\varepsilon) N^2 \log N + N_\omega N_h + N_\omega N_p)$. Normally $N_\omega = N_h$, but by symmetry and compactness of the Fourier transform $\tilde{d}(\omega, h)$ (typical for most seismic data), we can significantly shorten the domain for $\omega$, hence further reduce the computational load.

In comparison, the conventional velocity scan requires at least $O(N_\omega N_p N_h)$ 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 $e^{2\pi i \omega \sqrt{\tau^2 + r^2}}$, i.e., the range of $\tau$, $r$, $h$, and $\omega$. In practice, $N$ can often be chosen smaller than $N_\tau$, $N_p$, $N_h$, and $N_\omega$.

There is no general rule to select $N$ and the number of Chebyshev points $q_k$, $q_s$, $q_x$, $q_y$ (recall that $r_e = q_s q_x$ or $r_e = q_y q_x$). 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 $N_x \times N_h = 1000^2$. Figure 3 shows the result by the fast butterfly algorithm obtained on $N_x \times N_p = 1000^2$. For this problem, our method provides a reasonable result with $N = 32$, $q_h = q_s = 7$, $q_x = q_y = 5$ 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 5).

![2D synthetic data](image)

**2D field data**

We now consider a real 2D seismic gather in Figure 6. The sampling sizes are $N_x = N_y = 1500$, $N_h = 240$, and $N_p = 2000$. Parameters are chosen as $N = 128$, $q_h = q_s = 7$, $q_x = q_y = 5$. 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).

![2D field data](image)

**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
Fast Radon Transform

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

Following problem:

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

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 $N_t \times N_h = 1000 \times 128^2$. The output is obtained for $N_t \times N_p = 1000 \times 512$. 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 5: 2D real CMP gather, $N_t = 1500, N_h = 240$.

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

Figure 7: 3D synthetic CMP gather, $N_t = 1000, N_h_1 = N_h_2 = 128$.

Figure 8: Output of the fast butterfly algorithm, $N_t = 1000, N_p = 512, N = 64$, $q_{k_1} = q_{x_1} = 5, q_{k_2} = q_{x_2} = 3$; 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.