# A parallel windowed fast discrete curvelet transform applied to seismic processing

*Darren Thomson*, Gilles Hennenfent, Henryk Modzelewski and Felix J. Herrmann*
*Seismic Laboratory for Imaging and Modeling, Earth & Ocean Sciences Department, University of British Columbia*

**SUMMARY**

We propose using overlapping, tapered windows to process seismic data in parallel. This method consists of numerically tight linear operators and adjoints that are suitable for use in iterative algorithms. This method is also highly scalable and makes parallel processing of large seismic data sets feasible. We use this scheme to define the Parallel Windowed Fast Discrete Curvelet Transform (PWFDCT), which we apply to a seismic data interpolation algorithm. The successful performance of our parallel processing scheme and algorithm on a two-dimensional synthetic data is shown.

**INTRODUCTION**

Realistic seismic data sets, especially three-dimensional ones, can be extremely large. It is not uncommon for a data set to be orders of magnitude larger than what one could hope to fit in a computer's memory. This presents obvious challenges when attempting to process seismic data and ultimately use it to create images of the subsurface. One approach is to simply shrink the data set by downsampling or by considering only a small subsection of the full data set. In many situations, however, this is not satisfactory, and so a means for processing (and later imaging) the full data set is needed. Ideally, such a scheme would provide a straightforward way to make use of existing useful methods for processing small data sets.

The obvious solution to problems that require more memory than any single computer can offer is to use a large number of computers in parallel. Parallel systems of various types are very common in high performance computing, and are applied to problems of all sorts, including many in seismic data processing. For instance, a parallel implementation of the Fast Discrete Curvelet Transform (FDCT), which can be applied to a variety of problems in seismic processing, exists (L. Ying and Candès, 2005). However, there are limitations to the scalability of this implementation, to the extent that processing typical large seismic data sets is still intractable. As is the case with many classes of parallel algorithms, scalability is limited by growths in the amount of communication required between processors working in parallel, though this is dependent on the properties of the specific system being used (see e.g. Gupta and Kumar, 1993). Other classes of parallel algorithms involve splitting data sets into small pieces ("windows") and consider each window separately. While this is perfectly legitimate in many applications, problems can arise at window borders for a variety of reasons, including, commonly, artifacts related to the periodicity of the transform or to the discontinuity across the border.

Here, we consider the application of overlapping windows processed in parallel to a seismic data interpolation algorithm that makes use of the FDCT. These overlapping windows include a taper that is applied to overlapping regions such that the overall energy of the system is preserved throughout the operation. We restrict ourselves to two-dimensional data here, but this method is easily generalized to an arbitrary number of dimensions. This parallelization scheme addresses shortcomings of other schemes that could be applied to this problem, and can be generalized to create parallel windowed versions of a wide variety of operators.

This paper begins with a brief review of parallel algorithms in scientific computing. We then discuss our parallelization in more detail, including how it has been applied to seismic data interpolation, as well as some specific new issues that our scheme introduces to this (and other) algorithms. Finally, we show and discuss the results of applying this algorithm to a synthetic seismic data set with missing traces.

**BACKGROUND**

Parallel computation is necessary to solve realistic problems in many fields. The way in which computations are split in order to run in parallel is highly dependent on the particular problem in question, but two very basic approaches are commonly used in a wide variety of applications. Any algorithm that involves a large number of independent operations on independent data are easily run in parallel by simply giving each computing node a share of the work, without the need for any communication between nodes for the duration of the process. A large Monte Carlo experiment, for instance, is easily run in parallel in this way. Another approach to building parallel algorithms is to give each node a unique window of the data set that it is capable of handling. In some cases, these windows can have small overlaps. This is particularly useful in solving differential equations, where, for example, finite difference solvers require knowledge of a function at a given point of interest as well as the value of that same function at all of the points surrounding it (see e.g. J. Fan and Rector, 1997). However, under some circumstances, running an algorithm unchanged in parallel becomes very difficult. The FFT, for instance, requires a massive amount of communication when the data are spread across multiple nodes in parallel. As the number of nodes grows, the amount of information that needs to be communicated becomes unfeasible. By extension, this also impacts the FDCT (L. Ying and Candès, 2005).

FFT-based operations, like the FDCT, as well as other operations that are not easily run in parallel, often have desirable properties that can be exploited to find solutions to various problems. Thus, it is advantageous to have a system under which these operations can be performed in parallel on large amounts of data. One approach is to split the data into subsections, or windows, and operate on these separately. This approach is "embarrassingly parallel" - in other words, no communication between neighbouring processes is necessary during processing. However, this approach often fails due to the creation of artifacts at the edges of the windows (J. Fan and Rector, 1997) as a result of, for example, the Gibbs phenomenon, which arises when an approximation of the data (rather than the full representation in the transform domain) is taken. Approximations in transform domains are common, so a parallelization approach that eliminates the impact of edge artifacts is necessary. Even when approximations are not taken on separated windows, it is possible that differences between windows will be introduced through the process. Also, to be scalable such that it could feasibly handle large data sets, any parallel processing scheme must minimize the amount of necessary communication between nodes. Furthermore, any parallelized operation that is going to be performed repeatedly (i.e. as part of some iterative solver) needs to have a defined adjoint operation, and should preserve energy. Together, these properties create a numerically tight frame that will not introduce any growing error in an iterative process.

## THEORY

We propose a method to parallelize an arbitrary linear operator that avoids problems related to edge artifacts and preserves overall energy. It requires relatively little communication between parallel nodes, making it highly scalable. Our particular interest here is focused on a scalable parallel generalization of the FDCT, but we stress the fact that any linear operator could take the place of the FDCT in this framework.

### Parallel Windowed FDCT

The basic structures of this framework are overlapping and tapered windows. This scheme has previously been used in various parallel processing applications (see e.g. J. Fan and Rector, 1997). The details of these structures are quite flexible. In general, the sizes of the windows and the overlapping regions do not have to be uniform throughout the system. Here, though, we will only consider equally sized windows with uniform overlap throughout the system. One can express the overlap between windows by the value $\varepsilon$, which represents the depth to which one window receives adjacent data from its neighbours (Mallat, 1998), as illustrated in Figure 1. The overlapping regions of the windows are tapered such that the energy of the system remains constant when points in the data set are duplicated due to the overlaps. The tapering also ensures that edges of the data goes smoothly to zero at the edges, eliminating the potential of creating edge artifacts.

The taper is applied across the outer region of the overlapping windows, affecting points that are within a distance of $2\varepsilon$ from an edge. The tapering function $\mathbf{b}$ must satisfy the relationship

$$\mathbf{b}'^2 + \mathbf{b}''^2 = 1, \tag{1}$$

where $\mathbf{b}'$ and $\mathbf{b}''$ signify the tapers in adjacent windows that cover the same region. For consistency, we consider the application of an identical tapering function $\mathbf{b}$ to all windows. It follows from Eq. 1 that

$$b_n^2 + b_{2\varepsilon-n+1}^2 = 1, \tag{2}$$

where $n$ is an integer on the interval $[1, 2\varepsilon]$, and subject to the boundary conditions $b_1 = 0$ and $b_{2\varepsilon} = 1$. This condition ensures that the energy of the system is conserved when summing values from adjacent windows that represent the data at the same location. There are many examples of functions that satisfy this relationship (Mallat, 1998), the simplest being:

$$b_n = \sin\left(\frac{(n-1)\pi}{2(2\varepsilon-1)}\right). \tag{3}$$

To taper the end of the data set along each dimension, the function $\mathbf{b}$ is translated to the points $n = \{N, N-1, \cdots, N-2\varepsilon+1\}$, where $N$ is the total size of the overlapping windows. The tapering function is then

$$b_n = \sin\left(\frac{(N-n)\pi}{2(2\varepsilon-1)}\right). \tag{4}$$

Once the data is split into overlapping windows and tapered, operations, like the FDCT in our particular case, can be performed on each window independently. The shape of the taper function and the overlap of neighbouring windows are both shown in Figure 1. The data in this case is split into sixteen overlapping windows, four horizontally and four vertically. The dashed lines in the image represent the edges of the overlapping windows, with the region between nearby parallel lines being shared between the two windows. The taper function is shown, where one can see the value going to zero at the window edges. Since the image and all windows are square in this case, the taper functions along the vertical axis are the same as the horizontal ones that are shown. Also, note that tapering is not done at the edges of the image, since this would remove energy from the system. This issue can be successfully dealt with in a number of ways, but we omit this discussion.

We now consider the entire process in the context of linear operators acting on data vectors. In the parallel context, it is useful to distinguish between global vectors and operators, which comprise the entire parallel system, and local vectors and operators that exist only on individual nodes. Our notation will reflect this distinction. For instance, the distributed operator and vector $\mathbf{A}$ and $\mathbf{x}$ are easily distinguished from the local operator and vector $\mathbf{A}_i$ and $\mathbf{x}_i$ that exist on a node indexed by $i$. Note that there will be cases when we want to apply a local operator $\mathbf{A}_i$ to the data on every node in a global vector $\mathbf{x}$. This involves a block diagonal global operator where each block consists of the particular $\mathbf{A}_i$ corresponding to a given $\mathbf{x}_i$, which is simply a subsection of $\mathbf{x}$. We denote the block diagonal global operator for a local operator as $[\mathbf{A}_i]$.

An arbitrary global data vector is expanded into overlapping sections using the global windowing operator $\mathbf{W}$. The tapering operator $\mathbf{T}$ is then applied across the system. It is easy to see that the tapering operator is diagonal, and will have repeating patterns in blocks representing each node. Alternatively, one can simply look at the tapering operator as a local $\mathbf{T}_i$ on each node. Once $\mathbf{W}$ and $\mathbf{T}$ have been applied, any arbitrary linear operator $\mathbf{A}_i$ can be applied on each node.

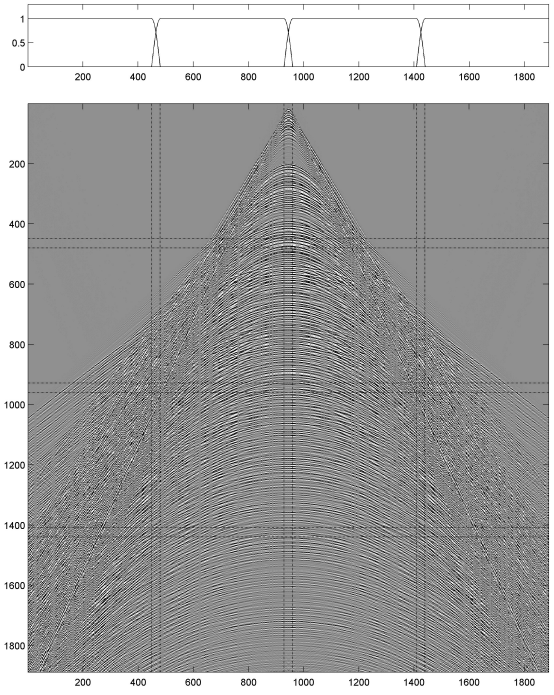



Figure 1: 2D synthetic seismic data with 30% missing vertical traces. Dotted lines represent borders of overlapping windows. Taper function is shown above for clarity.

Perhaps the most useful part of considering windowing and tapering as linear operators in matrix form is that looking at these matrices makes understanding the adjoints of these operations simple. Since $\mathbf{T}$ is diagonal, it is its own adjoint. For $\mathbf{W}$, the adjoint operation involves summing together overlapping regions. Since the forward operation was a "scatter," the adjoint becomes a "gather," where data that correspond to the same point are summed (Claerbout, 1992). In a parallel computing realization, this means that a given node sends its outer band to the nodes to which they belong, then gathers data related to its inner band from those same neigbours, and sums them together. Importantly, the combination of these processes satisfies the relation

$$\mathbf{W}^H\mathbf{T}^H\mathbf{T}\mathbf{W} = \mathbf{I}, \tag{5}$$

This ensures perfect reconstruction of the data when applying the operators followed by their adjoints. It implies that energy is preserved through the entire process. It follows that in any iterative algorithm instabilities or inaccuracies will not be introduced by this windowing and tapering process. The use of operator adjoints and the preservation of energy distinguishes our method from previous uses of overlapping and tapered windows.

At this point, it is useful to define a new operator that we will call the Parallel Windowed FDCT (PWFDCT), which is most simply described as the block diagonal global operator that is created by applying the FDCT $\mathbf{C}_i$ independently on each node after applying the global windowing and tapering operators $\mathsf{T}$ and $\mathsf{W}$. In other words,

$$\mathsf{C} := [\mathbf{C}_i]\mathsf{T}\mathsf{W}. \tag{6}$$

Since the local FDCT $\mathbf{C}$ is numerically tight, it follows that

$$\mathsf{C}^H\mathsf{C} = \mathsf{I}. \tag{7}$$

Other properties of $\mathbf{C}$ are similarly shared by $\mathsf{C}$. It is possible, then, to make use of the PWFDCT in existing algorithms that include the FDCT. It should be noted, though, that curvelets at very large scales (or, equivalenty, low frequencies) are not represented in the same way they would be if a single FDCT were performed on the global data. However, since the benefits of curvelets are mostly found at the finer scales (higher frequencies), the lack of large scale curvelet representation is not a problem. This makes a wide variety of algorithms capable of handling data sets much larger than otherwise possible. An example of a seismic data interpolation algorithm where the PWFDCT takes the place of the FDCT follows. The PWFDCT will be applied to other seismic processing problems, including primary-multiple separation and seismic deconvolution, in the future.

**Interpolation**

Hennenfent and Herrmann (2006) exploits the multiscale, multidirectional and continuity properties of seismic wavefronts which lead to sparsity of seismic data in the curvelet domain to solve the interpolation problem (see also Herrmann and Hennenfent, 2006, for more details). Reformulated using global operators and vectors, the interpolated data $\tilde{\mathbf{f}}$ is obtained by $\tilde{\mathbf{f}} = \mathsf{C}^H\tilde{\mathbf{x}}$ where

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{s.t.} \quad \|\mathbf{y} - \mathsf{R}\mathsf{C}^H\mathbf{x}\|_2 \leq \varepsilon. \tag{8}$$

In this expression, $\mathbf{y}$ represents the acquired data with missing traces in otherwise regularly sampled data, $\mathsf{R}$ a restriction (or so-called picking) operator that extracts the acquired traces from the interpolated data, $\mathbf{x}$ the PWFDCT representation of the interpolated data, and $\varepsilon$ the size of the noise present in the acquired data. Eq. (8) is solved using a large-scale problem solver for $\ell_1$-regularization minimization based on cooling method optimization and an iterative thresholding algorithm (Daubechies et al., 2004)

**Thresholding**

Algorithms that involve nonlinear thresholding in a transform domain present a problem when using tapered overlapping windows. The Curvelet-based interpolation method described above is but one example of an entire class of algorithms that use approximation or estimation in a transform domain (in particular a domain where the data in question is sparse), all of which involve thresholding. Fundamental to the approximation and estimation process is the selection of the most significant coefficients in the transform domain. The problem arises from the very fact that the overlapping regions are tapered. If we consider the FDCT, it is clear that the amplitude of curvelets representing data in the tapered regions will have their amplitudes reduced by the taper. Thus, when applying a uniform threshold value over the global system, it is likely that coefficients that would otherwise be considered "significant" will fall below the threshold solely because of the impact of the tapering. This introduces errors that have the potential to grow through the iterative processes that commonly make use of thresholding. Thus, it is important to correct threshold values to account for the impact of the tapering, such that coefficients that should be kept are not accidentally discounted.

A number of methods for correcting threshold values are possible. The problem essentially involves finding an operator in the transform domain that is the equivalent of the tapering operator. In the case of the FDCT, we are interested in the diagonal operator $\mathbf{D}$ such that

$$\mathbf{C}^H\mathbf{D}\mathbf{C} = \mathbf{T}. \tag{9}$$

This operator $\mathbf{D}$ can then be applied to the threshold vector to obtain a corrected threshold vector that can be used to more properly distinguish between significant and insignificant data in the transform (curvelet) domain. Since the FDCT is localized in both space and spatial frequency, the approximated diagonal operator $\mathbf{D}$ is expected to be an accurate approximation.

The simplest way to obtain $\mathbf{D}$ is through a large Monte Carlo sampling, where random noise realizations are tapered and then transformed. The root mean square (RMS) of all of the transformed results then gives an arbitrarily accurate approximation of $\mathbf{D}$. Optionally, the same noise realizations can also transformed without the taper applied. The RMS of the transformed tapered noise realizations can then be divided by the RMS of the transformed untapered noise realizations to remove any remaining noise artifacts from the Monte Carlo sampling.

It is also possible, in our case, to approximate $\mathbf{D}$ by evaluating the taper function at the centroid of each curvelet, and using that value as a weighting to apply to the relevant coefficient. However, this approach explicitly ignores the spacial extent of curvelets, since the value of the taper at the centroid of a given curvelet will in general not represent the overall effect of the tapering on that curvelet.

Other methods for finding $\mathbf{D}$ are possible, but further discussion is omitted here.

**Scalability**

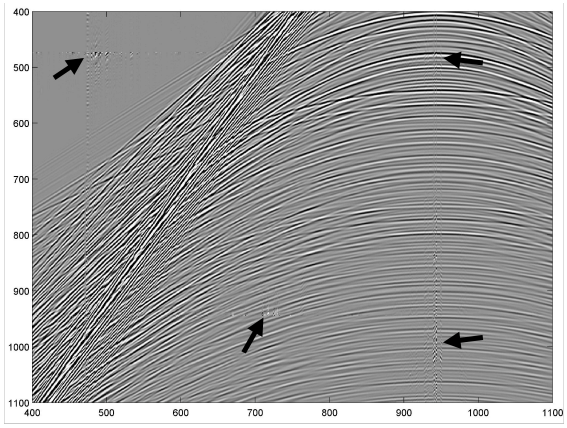
The parallelization scheme described herein is expected to be highly scalable. The computational cost will be, as in any parallel operation, have two aspects. The first is the cost of the operations on each node. In the language of this paper, this is the cost of the arbitrary local operator $\mathbf{A}_i$, or, for the Parallel Windowed FDCT, the local FDCT $\mathbf{C}_i$.

In general, though, the limiting factor in most parallel processing applications is the cost of communication between parallel nodes, and here our method has advantages. The communication is contained in the windowing operator $\mathsf{W}$. In the forward operation, each node only needs to communicate with the nodes containing adjoining data windows. It is straightforward to verify that each node needs to communicate $(M+N)\,2\varepsilon - 4\varepsilon^2$ points, where $M$ and $N$ are the dimensions of the overlapping windows. It is important to note that the amount of communication does not depend on the number of windows used or the size of the full data set. This distinguishes our approach from, for instance, the parallel FFT and other operations based on it, and implies that the method we describe will scale to very large sizes without growth in the amount of necessary communication between neighbouring nodes, which is especially important when communication between nodes is costly in relation to computation (Gupta and Kumar, 1993).

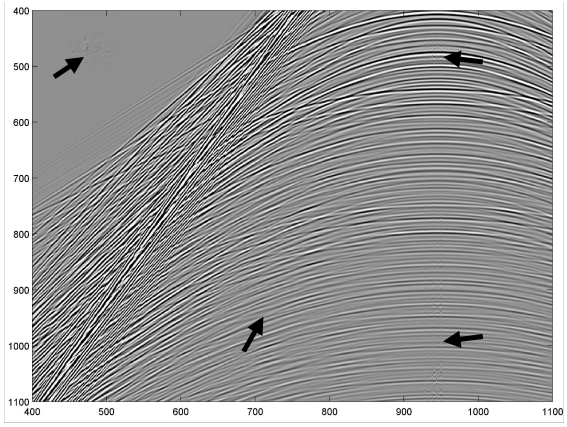
## RESULTS

The interpolation algorithm described above was tested, making use of the PWFDCT, on the synthetic seismic data set with missing vertical traces shown in Figure 1. The particular global windowing and tapering operators $\mathsf{W}$ and $\mathsf{T}$ divided the data set into sixteen windows, with the overlap and taper parameterized by $\varepsilon = 16$.

To demonstrate the importance of using overlapping and tapering, we did a similar interpolation using windows that were neither tapered nor overlapping. Through most of these windows, the algorithm works just as well as it does in overlapping windows. However, edge artifacts are evident near window borders, as would be expected. In Figure 1, we show a subsection of the full output of interpolation runs using overlapping and non-overlapping windows. The same subsection is used for both, and contains one full window and parts of its nearest neighbours. In the example using non-overlapping windows (Figure 2(a)), the edges of the windows are obvious from the artifacts that appear. When overlapping windows are used (Figure 2(b)), the artifacts are no longer evident, and it is not clear at all where the window borders are. In essence, the interpolation performs just as well in the proximity of the window edges as it does in the middle of the window, which is clearly untrue for non-overlapping windows. The importance of overlapping and tapering is thus, as expected, clear from this example. We




(a)




(b)

Figure 2: Interpolation output for (a) non-overlapping windows and for (b) overlapping, tapered windows with threshold correction. Arrows indicate locations of artifacts in (a) and show the improvement in (b).

also compared different methods for correcting the threshold value in the curvelet domain to account for the taper. When the taper was not taken into account in thresholding, errors were evident in the overlapping region. When correcting threshold values by using a Monte Carlo sampling or by evaluating the taper function at curvelet centroids, artifacts related to erroneous thresholding are eliminated. We omit the figures demonstrating this due to space limitations.

## CONCLUSIONS

Since seismic data sets are typically very large, it is important to have the capability to process data sets much larger than a single computer could potentially hold in memory. In order to run many algorithms in parallel, it is not sufficient to process data in separate pieces. At the same time, these same algorithms are often not scalable in their normal form due to exponential growth in the amount of communication between nodes that they require. For these reasons, we have developed a scheme that involves overlapping, tapered data windows that can be processed in parallel that is highly scalable since the communication costs do not grow with the number of processing nodes. We have used this method to define the PWFDCT, but we stress that this method is general and that the FDCT can be replaced by an arbitrary operator acting on each overlapping window independently as desired.

We applied the PWFDCT to a seismic data interpolation algorithm that is shown, in another presentation to this conference, to be successful using the FDCT. We have demonstrated that good results can be achieved with the PWFDCT in this algorithm, and shown the importance of the overlapping and tapering. Furthermore, we have demonstrated that it is possible to correct for the effect of tapering on threshold values in the transform domain.

Much future work is expected to arise from the ideas described herein. In particular, we will apply the PWFDCT to other seismic data processing algorithms, enabling them to work on much larger data sets than is currently feasible. The parallelization method described here will also be applied to other operators besides the FDCT. Finally, we hope that this method will open up new possibilities in parallel signal processing in a variety of fields.

## ACKNOWLEDGMENTS

The authors would like to thank the authors of the Fast Discrete Curvelet Transform (FDCT) for making this code available at www.curvelet.org. This work was in part financially supported by NSERC Discovery Grant 22R81254 of Felix J. Herrmann and was carried out as part of the SINBAD project with support, secured through ITF (the Industry Technology Facilitator), from the following organizations: BG Group, BP, Chevron, ExxonMobil and Shell.

## REFERENCES

Claerbout, J. F., 1992, Earth soundings analysis: Processing versus inversion: Blackwell Scientific publishing.

Daubechies, I., M. Defrise, and C. de Mol, 2004, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint: Comm. Pure Appl. Math., 1413–1457.

Gupta, A. and V. Kumar, 1993, The scalability of FFT on parallel computers: IEEE Transaction on Parallel and Distributed Systems.

Hennenfent, G. and F. J. Herrmann, 2006, Application of stable seismic signal recovery to seismic interpolation. (submitted for presentation at 76th SEG Conference & Exhibition).

Herrmann, F. J. and G. Hennenfent, 2006, Non-parametric seismic data recovery with curvelet frames. (to be submitted).

J. Fan, K.T. Nihei, L. M. N. C. and J. Rector, 1997, Overlap domain decomposition method for wave propagation: Presented at the 67th SEG Conference & Exhibition.

L. Ying, L. D. and E. Candès, 2005, 3D discrete curvelet transform. submitted for publication.

Mallat, S., 1998, A wavelet tour of signal processing: Academic Press.