Five-Dimensional Interpolation: exploring different Fourier operators

Daniel Trad *

ABSTRACT

Five-Dimensional interpolation has become a very popular method to pre-condition data for migration. Many different implementations have been developed in the last decade, most of them sharing a similar dataflow and principles, which is applying sparseness to a transform and mapping back to a new seismic geometry. In this report, I explore three different ways to implement the mapping between data and model in the context of Fourier transforms. These three methods are multidimensional Fast Fourier transform, Discrete Fourier Transforms, and Non-Equidistant Fast Fourier transforms. I incorporate the three operators inside the same inversion algorithm, to be able to perform a fair comparison between them. By keeping all the same, we can focus on the actual performance differences. In addition, I discuss some the implementation details, similarities and differences.

INTRODUCTION

Five-Dimensional interpolation has become a very popular method to pre-condition data for migration. Most implementations have a similar dataflow and principles, which is applying sparseness to a transform and mapping back to a new seismic geometry. Applying the sparseness to the transform has an effect of filtering out sampling artifacts, because these are, by the definition of the transform, non-sparse. Looking from this high level of generality, all these different implementations are very similar. This similarity extends also to other dataflow details like overlapping five dimensional windows. Usually some version of either a conjugate gradient inversion or an anti-leakage inversion are applied to enforce the sparseness constraint. So what do all these implementations differ on? The most important difference among them is on the transformation to connect data and models, in other words, how they represent the data and which basis functions are utilized. Some implementations use Fourier transforms, some use Hankel transforms, others curvelets, frames, Radon, Prediction Filters (Fxy) and Green functions based on modeling/migration operators.

Many implementations used in the industry are based on Fourier transforms. Since much work in science has been done on the development of transformations, and many of them have properties that make them more appropriate for the representation of seismic events, we must ask why is Fourier still the most attractive tool for this job, at least from the practical point of view. Fourier transforms are very efficient, and in the presence of good sampling they can be inverted very easily. Although some of the other transformations mentioned above are better representations of the geometrical shape of seismic events, sometimes also tend to have difficulty to represent small details of the data and therefore may produce more artificial-looking interpolated data. One interesting example is the case of Green functions which is used in Least Squares Migration. Although prob-

*CREWES

ably the best of all the approximations of the physics of seismic, the uncertainties related to the weighting of the transformation lead to complex scenarios where some events suffer overshooting in amplitude, and some others undershooting, which creates a complex convergence pattern. Other transforms, for example multidimensional Radon transforms, are computationally more expensive to apply than multidimensional Fourier transforms. Transforms based on multi-resolution, like curvelets and frames, contain highly redundant information that increases the dimensionality of the model and therefore makes the inverse more expensive. Hankel transforms have also become quite popular for its capability to attenuate noise, although are less capable than Fourier transforms for handling very large windows. Fxy transforms, although the most efficient to handle aliasing, have difficulties to work in very sparse 4D spatial grids.

Another factor that strongly influences the efficiency and flexibility of all these transforms is whether they use exact input spatial locations or some approximation achieved by multi-dimensional binning, either direct or with some intermediate interpolation. When using exact locations, these transformations become computationally expensive to invert. When using binning these transformations are efficient and can handle large window sizes and gaps, but lose precision and flexibility to adapt to narrow azimuths and long offsets. When binning 5D data from standard geometries, the resulting grids turn to be very sparse. What is interesting about Fourier transforms, and maybe the reason why they are still used the most in industry, is that they seem to adapt well for either approach. The subject of this paper is to explore how Fourier interpolation performs in one or the other case. I compare a five-dimensional interpolation with three different operators applying the three approaches mentioned above, binning, exact locations, and approximated with intermediate approximations. Special care is taken of using the same inversion and dataflow for the three cases, leaving differences strictly confined to the differences in operators. A similar comparison but using also other operators different from Fourier will be addressed in a future paper.

Sparse least squares inversion and operators

The two key concepts in this paper are sparse least squares (LS) inversion and operators. The general philosophy for sparse least squares inversion is explained in Claerbout (1992); Sacchi and Ulrych (1996) and Trad et al. (2003). A detailed description of the sparse algorithm I use in this work is in Liu and Sacchi (2004) and Trad (2009).

Given a transformation of some model $\mathbf{m}$ to some data $\mathbf{d}$ through some mathematical operator $\mathbf{L}$,

$$\mathbf{d} = \mathbf{L}\mathbf{m}. \tag{1}$$

we can approximately solve the inverse problem of recovering the model that produced the data by inverting the operator in a least squares sense. The meaning of the different components of equation (1) depends on the details of the operator and are often associated with well-known transformations. For the case of interpolation $\mathbf{L}$ contains some form of a transform that maps data at their exact or binned locations to a domain where continuous reflections and diffractions appear condensed to as few as possible elements of the model. Also, to be used in interpolation, we need sampling artifacts, like discontinuities and irregularities of the data, to map to many elements on the model, that is in a non-sparse manner. However, the transform must be able to represent details of the data that, although chang-

ing quickly, may represent signal that we want to preserve. If possible we want to have unwanted coherent events, for example ground-roll, to map to different parts of the model than reflections or diffractions, so we can filter them out from the model; or if they don't, then they should be properly interpolated as well, like multiples, so they can be predicted and subtracted by other means. Usually it is not possible to achieve all these goals at the same time, so we often give up some of them and try to remove or attenuate these events before interpolation.

In this paper $\mathbf{L}$ is a synthesis of the data in terms of plane waves (Fourier transform). The other operators mentioned in the introduction are not plane waves but some other choice of basis functions. In all cases, the interpolation $\mathbf{L}$ contains a sampling operator that represents the signature of the seismic acquisition. In general, we are trying to eliminate this signature by deconvolving $\mathbf{L}$ by least squares inversion. If there is no acquisition signature, the Fourier transform is trivial to invert and amounts simply to apply inverse Fourier transform, since the adjoint of regularly sampled Fourier transform is also the inverse.

To invert operators, we define a cost or objective function, that is a mathematical expression that measures the undesired characteristics of the model. The most common is to find a model that honors the data in a least error sense, measure on some norm, and has a minimum of information not required by the data. This statement of goals is commonly presented as

$$\begin{aligned} &\text{minimize } \|\mathbf{W}_m\mathbf{m}\|_p^p \\ &\text{subject to } \|\mathbf{W}_d(\mathbf{d}-\mathbf{L}\mathbf{m})\|_q^q = \phi_d \end{aligned} \tag{2}$$

where ϕ_d is some estimate of the noise level in the data plus a residual due to the failure of the proposed model to explain the data. p and q indicate that different norms can be applied to measure the norm of vectors. $\mathbf{W}_d$ could be a matrix or vector of data weights, often a diagonal matrix containing the inverse of the standard deviation of the data, but more generally it could be any kind of filter that leaves out bad data. For the case of interpolation, for example, $\mathbf{W}_d$ should nullify any information from unrecorded traces, otherwise the model is forced to predict zeroes on those traces (honoring the empty traces). $\mathbf{W}_m$ is a matrix of model weights that we can design to enhance our preference regarding the model, for example resolution or smoothness.

We can now set the misfit and undesired characteristics of the solution in a cost function and, by minimizing it, obtain the model that better approximates the desired solution. Thus, the model can be found by solving the system of equations

$$(\lambda\mathbf{W}_m^T\mathbf{W}_m + \mathbf{L}^T\mathbf{W}_d^T\mathbf{W}_d\mathbf{L})\mathbf{m} = \mathbf{L}^T\mathbf{W}_d^T\mathbf{W}_d\mathbf{d}\,. \tag{3}$$

where λ is the second trade-off hyper-parameter that will allow a different weight to be assigned to the misfit and model constraints. The system of equations (3) is usually solved iteratively by fixing the model weights to some previous estimation (like the spatial spectra from the previous temporal frequency) and applying a linear minimization by conjugate gradient algorithm (CG).

In practice equation (3) is not solved directly because it contains a parameter λ that requires estimation and has the potential for a zero division when the model weight goes

to zero (zero model). An easier way to implement this formulation is to incorporate these weights and filters into the general operator $\mathbf{L}$, which is just a change of variables in the fundamental equations. By applying a right preconditioning, the modeling equation (1) becomes

$$\mathbf{d} = \mathbf{L}\mathbf{W}_m^{-1}\mathbf{W}_m\mathbf{m} \tag{4}$$

and the optimization problem (2) is now

$$\begin{aligned} &\text{minimize } \|\tilde{\mathbf{m}}\|_p^p \\ &\text{subject to } \|\mathbf{W}_d(\mathbf{d} - \mathbf{L}\mathbf{W}_m^{-1}\tilde{\mathbf{m}})\|_q^q = \phi_d \end{aligned} \tag{5}$$

where $\tilde{\mathbf{m}} = \mathbf{W}_m\mathbf{m}$. The minimization of the cost (5) produces the following system of equations

$$(\lambda\mathbf{I} + \mathbf{W}_m^{-T}\mathbf{L}^T\mathbf{W}_d^T\mathbf{W}_d\mathbf{L}\mathbf{W}_m^{-1})\tilde{\mathbf{m}} = \mathbf{W}_m^{-T}\mathbf{L}^T\mathbf{W}_d^T\mathbf{W}_d\mathbf{d}\,. \tag{6}$$

Hence, the effect of the $\mathbf{W}_m$, also known as right preconditioning, is to set the model weights as part of the modeling rather than a penalizing factor in the cost function. The system (6) can then be solved by setting $\lambda = 0$ (no regularization) and letting the number of internal iterations in the CG algorithm play the role of regularizer. Notice equation 6 is very simple if we consider a new operator $\tilde{\mathbf{L}} = \mathbf{W}_d\mathbf{L}\mathbf{W}_m^{-1}$ and a new model $\tilde{\mathbf{m}}$. This new operator contains a data filter (selection filter) and a model filter that enforces sparseness.

OPERATOR IMPLEMENTATION

The are several possible operator choices for multidimensional interpolation. In this report, I will discuss only Fourier transformations, because they are the most commonly applied in the industry. The three most common choices are:

- Fast Fourier transform (FFT) with multidimensional binning and grid adaptation.
- Discrete Fourier transform (DFT) using exact spatial locations
- Non-Equidistant Fast Fourier transform (NFFT) using interpolation plus FFT.

To make sure differences are solely due to the operators and no other details of the implementation, I have written one 5D interpolation module including the three operators. By using an Object-Oriented Programming approach I encapsulate all aspects of the algorithm in common classes. Each operator is implemented as a separate class, and a simple parameter switch allows one to change from one to another implementation. In the following I explain each operator.

Fast Fourier transform operator

As described in Liu and Sacchi (2004) and Trad (2009) a multidimensional FFT can be applied on each frequency slice but carefully binning is required because FFT algorithms assume data are regular. In multi-dimensions, binning is not trivial for several reasons as explained in Trad (2014). A complete 5D grid implies a data set acquired with line interval equal to group interval, which is never done in real seismic. Therefore, when binning real

data on a 4D spatial grid, the data can only fill a minor percentage, typically between 1 to 5 percent, leaving for the algorithm to infill 99 to 95 % of the remainder of the data.

Binning along inline crossline directions is relatively straightforward, being a sensible choice the bin size, which is usually calculated to match the Fresnel zone size after migration. On the other hand, binning along offset and azimuth, or inline crossline offsets can be quite complicated, depending on the fold of the data, the shot patch pattern, the offset range, the presence of gaps, and the distance to the border of the survey. Doing the right binning is critical for the FFT approach. Coarse binning introduces time jitering that affects sparseness in the Fourier space, and creates a problem with traces that fall into the same bins. Fine binning makes the model size larger and therefore harder to solve. For example, reducing binning by half in one direction forces interpolation to infill twice as many grid cells, and reducing the bin size by half on each direction makes the problem 16 times larger.

Although it is very difficult to automatically consider all these details, an industrial application of this algorithm has to adjust the binning size to deal with these issues. One way to account for the geometry is to adjust the offset and azimuth binning according to the fold and the offset range. For example, in my implementation the user pre-defines the percentage of alive traces required, usually 3% for wide azimuth data or 5% for narrow azimuth data. With this information, the algorithm calculates the optimal size of the offset/azimuth binning intervals to achieve this target. More sophisticated schemes are possible, but this simple scheme helps to adjust the grid with the data fold, as the interpolation works on different areas of the survey. Another aspect of binning is how to adjust the intervals such that there are no extrapolated traces, but instead new traces are created only inside the region of support of the acquisition. For example, the azimuth range can be made automatic, depending of the actual coverage. The minimum/maximum offsets along inline/crossline directions can be made to adjust the patch size in the input data.

Discrete Fourier transform

Perhaps the simplest way to calculate the Fourier transform of irregular data is to use the mathematical definition, which for 1D is:

$$U_j = \sum_{xmin}^{xmax} \exp\left(-ik_j x_i\right) \times u_i \tag{7}$$

In this equation u_i is the value of the sample at position x_i and U_k is the value of the DFT at wavenumber k_j. Because the k axis is usually made regular, the k_j value is replaced by an index j times a Δk interval. For 5D interpolation, k_i, u_i become 4D vectors $\mathbf{k_i}, \mathbf{u_i}$. The summation in equation 7 becomes four nested summations.

$$\mathbf{U_j} = \sum_{\mathbf{x1min}}^{\mathbf{x1max}} \sum_{\mathbf{x2min}}^{\mathbf{x2max}} \sum_{\mathbf{x3min}}^{\mathbf{x3max}} \sum_{\mathbf{x4min}}^{\mathbf{x4max}} \exp\left(-\mathbf{ik_j x_i}\right) \times \mathbf{u_i} \tag{8}$$

Because of the four nested loops, this formulation is very slow in multi-dimensions

and requires large amounts of memory. In practice, although eliminates the need for a complicated binning scheme, the DFT method may not necessarily be the best choice for wide azimuth data. To speed up this algorithm we can reduce the operator to calculate only the wavenumbers inside the FK volume with physical significance. At each temporal frequency ω the model is calculated only between a minimum and maximum wavenumber given by the maximum slope the requested by the user. A simple way to define these boundaries is to specify the temporal frequency at which we want the wavenumbers to be full. Some other computational tricks are required to make this formulation practical. It helps if memory is available, to pre-calculate the operator or use sin-cos tables. In addition, careful alignment of memory allocations, loop vectorization, and multi-threading, help to make the algorithm feasible. Possible the best way to implement this method is using GPUs or Xeon Phi processors, both approaches that we are currently investigating.

Parameterization for this approach is very similar to the binning approach. Although in principle there is no sampling interval to specify, in practice we need to specify them just the same as in the binning approach, so the algorithm can calculate the Nyquist frequency on each spatial dimension. However, the algorithm has more flexibility to adjust to the spatial ranges, being much easier to adjust for example for streamer acquisitions where the azimuth range changes continuously from near to far offsets. To understand the significance of this last statement, consider for example streamer data, where every inline is sampled only at one crossline offset. The DFT can handle this situation, but for the FFT approach, this is clearly insufficient, since the 4D spatial grid requires each inline bin to have all the crossline offset values.

Non-Equidistant Fast Fourier transform

As explained above, the DFT approach, although flexible and precise, is computationally very expensive when working in multidimensions. A more efficient approach of calculating the Fourier spectrum of irregular sampled data is to use some intermediate, cheap, and localized interpolation to move the traces to the bin centers in the 4D spatial grid, and then apply a multi-dimensional FFT.

To put all the operators in the same context, we can think of the binning + FFT method described first as a zero-order interpolation (nearest neighbor) plus FFT. A more sophisticated approach would be, for example, to use a higher order polynomial interpolation instead of zero order. In this work, I have used the method called Non-equidistant Fast Fourier transform (Duijndam and Schonewille, 1999). In particular, I have used the libraries from the Technical University of Chemnitz (Keiner et al., 2009).

In this approach, the irregular data are convolved with a smoothing window to relocate the samples onto a regular grid. Notice this is an irregular multi-dimensional convolution, so implementation is a bit different from traditional convolution where both signals are regular with identical sampling interval. A multidimensional FFT is applied to map the resulting regular convolved (bandpass filtered) signal to the Fourier domain. The effect of the window can be removed by deconvolution, which can be done in the Fourier domain since all samples are now in regular locations. Figure 1, taken from Gulati and Fergusson (2009), shows the flow chart calculation for the NFFT spectrum. Basically, the NFFT

perform a similar calculation as a Riemann NDFT summation, except that the data are first convolved with a window $g(x)$ to transform it into a regularly sampled signal. The FFT can then be applied for the spectrum with efficiency $N\log(N)$. Then the window $g(x)$ can be deconvolved in the Fourier domain $1/G(m)$. This produces only an approximated spectrum, but by fine tuning the window parameters and length of the FFTs it is possible to reduce the error to negligible values.

One problem with this algorithm is that it uses significant memory space to store the multidimensional window parameters pre-calculated at front for efficiency. This makes this algorithm less suitable to use multithreading to simultaneously process several groups, like it can be done for the FFT. Although a multithreaded version of this library exists, in practice it does not seem to improve much efficiency in my testing. Another issue is that the library has some constraints for the window parameterization: it requires polynomials with an even number of parameters, and a minimum oversampling factor for these windows. This requires careful calculation of padding factors, and research is still ongoing to obtain a general an optimal parameterization. In addition, spatial dimensions are always specified with ranges between 0-1, which makes zero padding a bit more complicated than the other cases.

Notes on parallelization

All these operations are computationally expensive, so normally interpolations are run using computer clusters. Parallelization is usually better when is coarse-grained because it leads to less communication through the network. Because of that, the most efficient parallelization for interpolation is to process different windows (zones) in different nodes, which is possible with all the three operators. However, there is a difference among the three approaches on how to use more efficiently multithreading on each node. For FFT, where memory requirement is not too high, it is more efficient to use different threads to interpolate different windows. This is because multithreaded FFT is not as efficient (factors of only 2 or 3 are common when using several threads). For DFT, memory requirements are much higher because of the pre-computing of the Fourier operator, so running different windows in different threads is usually not feasible. However, for the DFT algorithm multi-threading at the outer summation level is very efficient, so using one window per node is the best approach, giving accelerations close to the number of threads. For NFFT, the situation is more similar to FFT, where efficiency improvements are not very high. Unfortunately, memory requirements are high because of the pre-computation of the multi-dimensional windows, so it is often not possible to process more than one group per node. This makes efficiency improvements for NFFTs not as high as expected. There is, however, another reason why NFFT performs more efficiently than FFTs: the NFFT grid does not need to be reduced as much since binning errors are decreased by the algorithm. This factor gives the NFFT the potential to be more efficient, as well as more precise. The actual improvements depend on the geometries and implementations.

Output geometries

All these operators are recalculated for every new input group because they depend on the input and output coordinates. A slightly different situation occurs depending on whether input and output coordinates are the same (infilling) or not (regularization). To make the examples clear, since they illustrate all the different cases, I proceed to explain first what these cases are.

There are several ways to classify interpolation. At the top of the list we can talk either about **infilling** or **regularization**. For infilling the goal is to create new seismic traces without changing or altering the existing survey. On the other hand, regularization attempts to replace an existing survey with a different one containing the same geophysical information. Both approaches have advantages and disadvantages. The most important advantage of infilling is that it is reliable since original traces are kept, and easier to evaluate because new traces are inserted in between existing traces and visual inspection is easier. When infilling a survey, it is natural to stay close to the original illumination pattern of the acquisition. New traces are rarely far from the original data, except for the case of gaps, which are easy to detect. It is possible to smoothly control the amount of interpolation, for example by applying a discarding threshold on the acceptable average distance between new and original traces. The drawback is that we do not completely solve the acquisition pattern since irregularities on the original survey may produce artifacts on the migration algorithm.

Regularization, on the other hand, requires the practitioner to make sure the new geometry illumination do not depart significantly from the original. This is less of a problem if the same type of geometry is used, for example, orthogonal to orthogonal, or streamer to streamer, which is more natural when surface consistent, acquisition type of geometries are used. These surveys are designed starting from shot and receivers with a given illumination patter. On the contrary, surveys designed starting from midpoints with a regular grid of offset/azimuth or inline/crossline offsets (flooding type), may easily deviate from original illumination patterns, so they require careful control. See Trad (2009, 2014) for more detailed discussions.

Since the industry is moving towards Reverse time migration algorithms (RTM), it is important to emphasize that flooding type of geometries cannot be used for RTM. These and similar algorithms require to have surface consistent geometries because they operate shot by shot or using plane-waves calculated from shots. For RTM the optimal design is a hybrid that keeps the same shots in the same locations without increasing their number, but instead regularize the receivers to make line interval close to group interval and therefore make finite difference algorithms more precise.

Whichever the approach adopted to create the output geometry, from the algorithmic point of view, infilling and regularization share the same algorithm to estimate the Fourier spectra (model), but they differ on the final step of creating new traces. In regularization, the operator used to create the new traces is different from the operator used to estimate the spectrum, whether infilling uses the same operator. This has a bigger impact for the DFT and NFFT operators, because of the time it takes building these operators and their memory

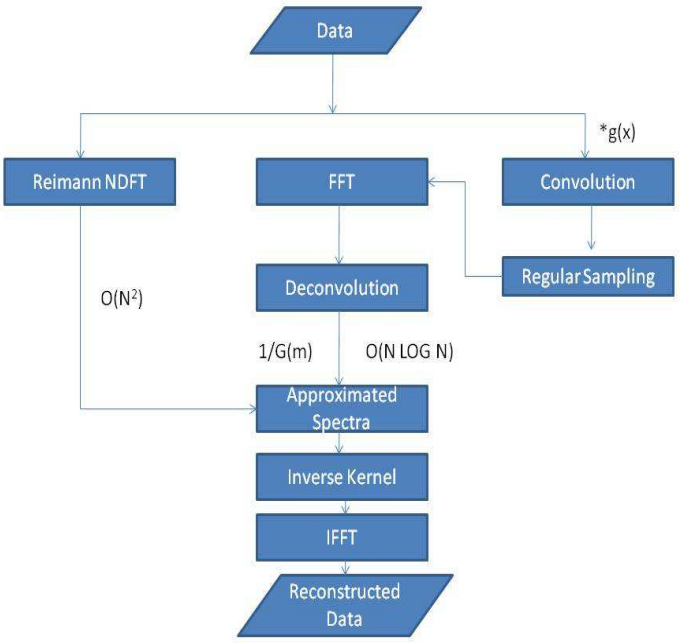

FIG. 1. Diagram showing NFFT calculation (Gulati and Fergusson, 2009)

requirements. For the FFT approach, only the binning part of the operator changes.

EXAMPLES

As a first example, Figure 2 shows a test for regularization when the input traces are all populated and the goal is only to move the traces to regular intervals. This test is intended to detect differences between binning, exact or approximated locations. In purpose, I did not apply normal moveout correction (NMO), to illustrate the behaviour for curved events. We see that the three operators succeed in regularizing the data and binning errors are not visible. Figure 3 shows a similar example but this time the input data contains only 25% of alive traces. In this more demanding conditions there start to appear some minor differences between the three operators, but overall the three results are reasonable. As expected FFT was the fastest, DFT the slowest, and NFFT in between. However, for this small example(2D data, 3D interpolation) the differences are negligible.

In both regularization examples I used shot/receiver coordinates as the dimensions to regularize to. This is feasible in simple cases, for example 2D geometries. In real 3D geometries, usually acquisition coordinates (i.e. shot x/y, receiver x/y) are not adequate for interpolation because acquisition lines are set too far apart. The most common approach is to use midpoints x/y, and either offset xy or offset azimuths, as in the following examples.

As a very different test, we compare the three operators in a difficult situation simulating Horizontal Transverse Isotropy (HTI). Figure 4a shows the locations of shots and receivers that contribute to an inline/crossline window on a problematic area of a real geometry. Figure 4b shows the locations of shots and receivers that are created when using flooding regularization. Because this type of geometries is created from uniform coverage

on midpoint/offset/azimuth, shots and receivers are not distributed on regular patterns and there are as many of them as traces. Figure 4c shows the coverage for input and output on the offset x/y domains. Because the flooding option was designed in polar coordinates, all output cdps have the same number of offsets for each azimuth. This is particularly well suited to study patterns along offset/azimuth. The input on the other hand, is distributed across all the domain, but localized between -180 and -22 degrees. I reduce the operator azimuth range to contain only the azimuth range in input to avoid extrapolation. This truncation of the azimuth coverage makes the test much harder for the FFT approach than for the other operators, and we will see its effect on the results. Figure 4d shows the coverage on the midpoint X/Y domain. Again, we see the uniform coverage from the output compared to the input. However, the midpoint coverage is quite optimal, and the difficulty of the test comes only from the azimuth truncation and amplitude variation.

Figure 5a shows a super CDP (3 cdps) on the original survey, for offsets between 0-1500 meters. Figure 5b shows the corresponding central CDP (only one) from the flooding geometry regularized using the FFT4 operator. Because of the polar nature of the geometry, it is possible to see clearly the azimuth pattern. However, the pattern has truncation between cycles, because they do not cover the 360 degrees but only half the circle. One of the azimuths cannot be predicted because there are no original traces to sample it (as seen in Figure 4c). The algorithm is struggling to follow the variations because of the poor sampling. Figure 5c shows the result using the DFT operator. The results have improved, probably because the DFT manages to accommodate the irregular nature of the azimuth in the input data a bit better (no binning). Figure 5d is the equivalent using the NFFT algorithm, and is quite comparable to the DFT result.

In terms of running time, the FFT approach took 4 minutes (1 thread), the DFT approach 14 minutes (10 threads) and the NFFT took 12 minutes (1 thread). As mentioned before the DFT algorithm is more expensive but it does a better use of multithreading so the time difference is not significant on this example. However, in practice, the FFT algorithm would be able to run 10 groups simultaneously, so when comparing running time for a survey the difference would be around 20 times. The NFFT, would be a bit better than the DFT but not as much since it requires too much memory to run simultaneously many groups in the same node.

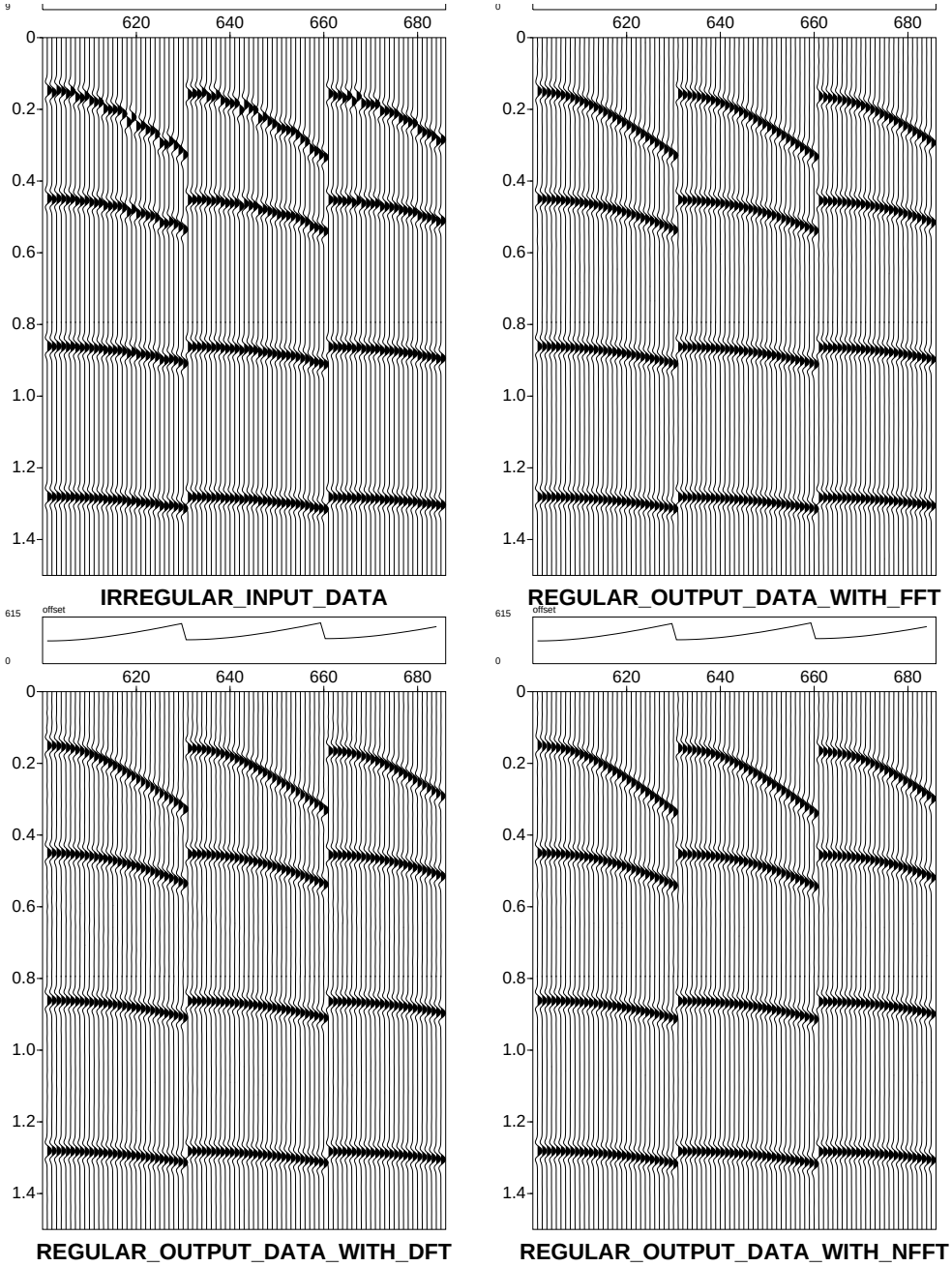

FIG. 2. Comparison of operators for regularization (offset values at the top of each window). There are no major differences, showing that binning does not introduce visible errors for typical running parameters, even for curved events.

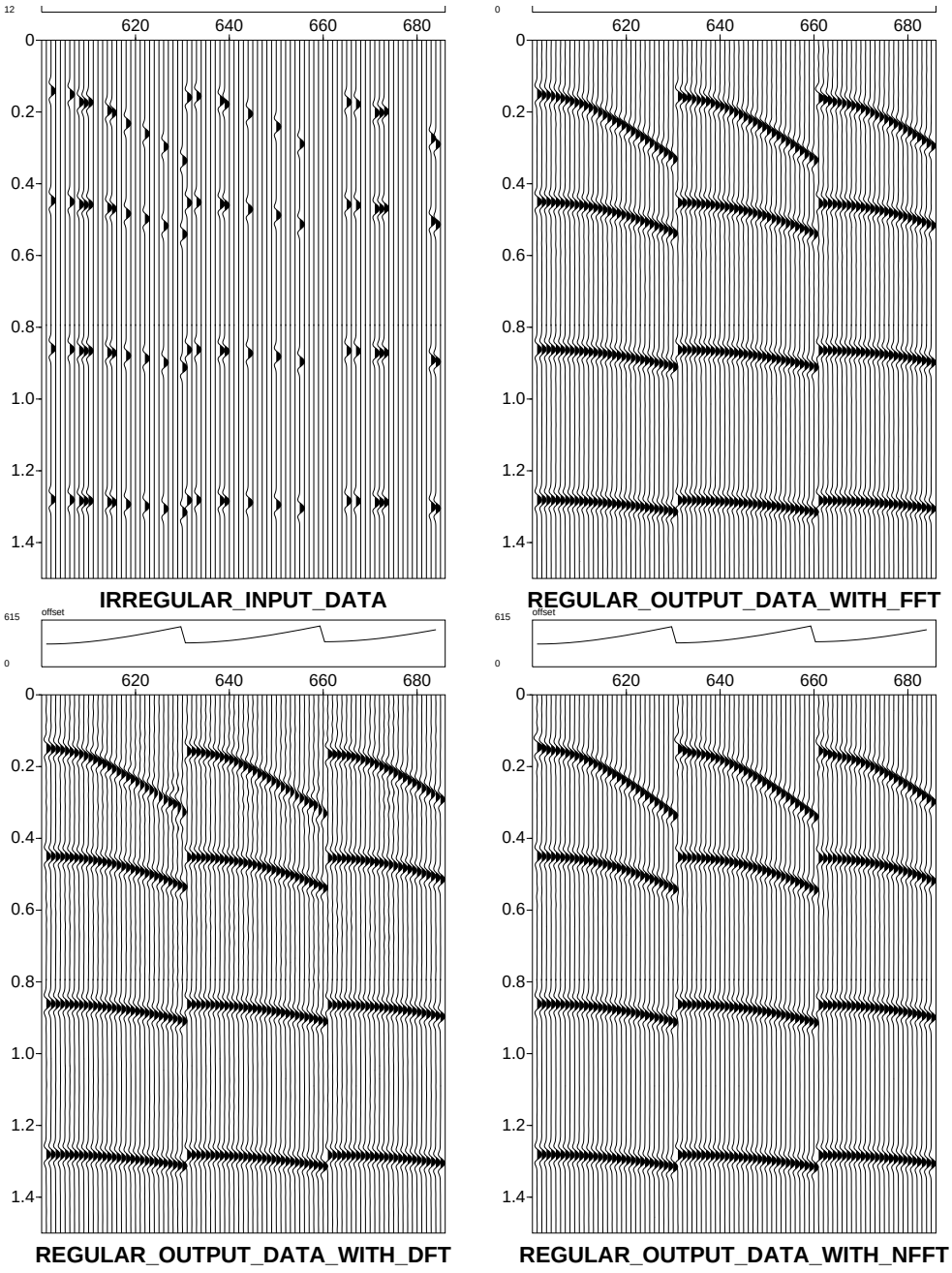

FIG. 3. Similar comparison but this time input data has been randomly decimated to 25% (1 alive every 4 traces)

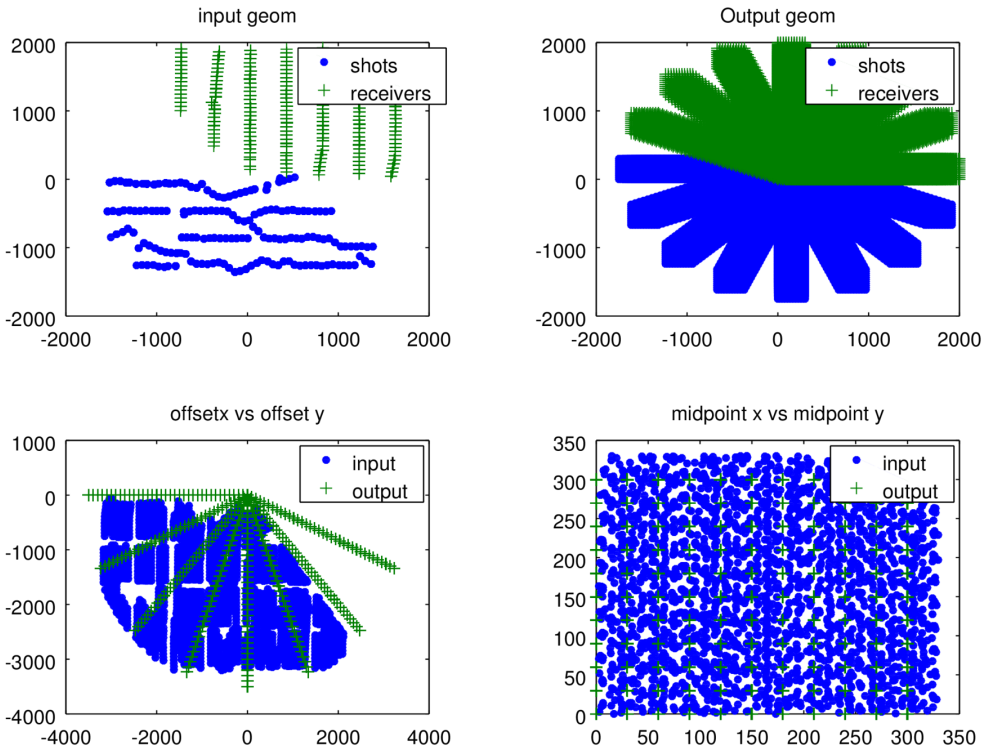

FIG. 4. Input and output geometries for the case of flooding regularization. a) contributing shots and receivers from a window on a challenging area of a real geometry, b) shots and receivers generated when flooding is applied. c) OffsetX/Y for input and output. d) midpoints X/Y for input and output

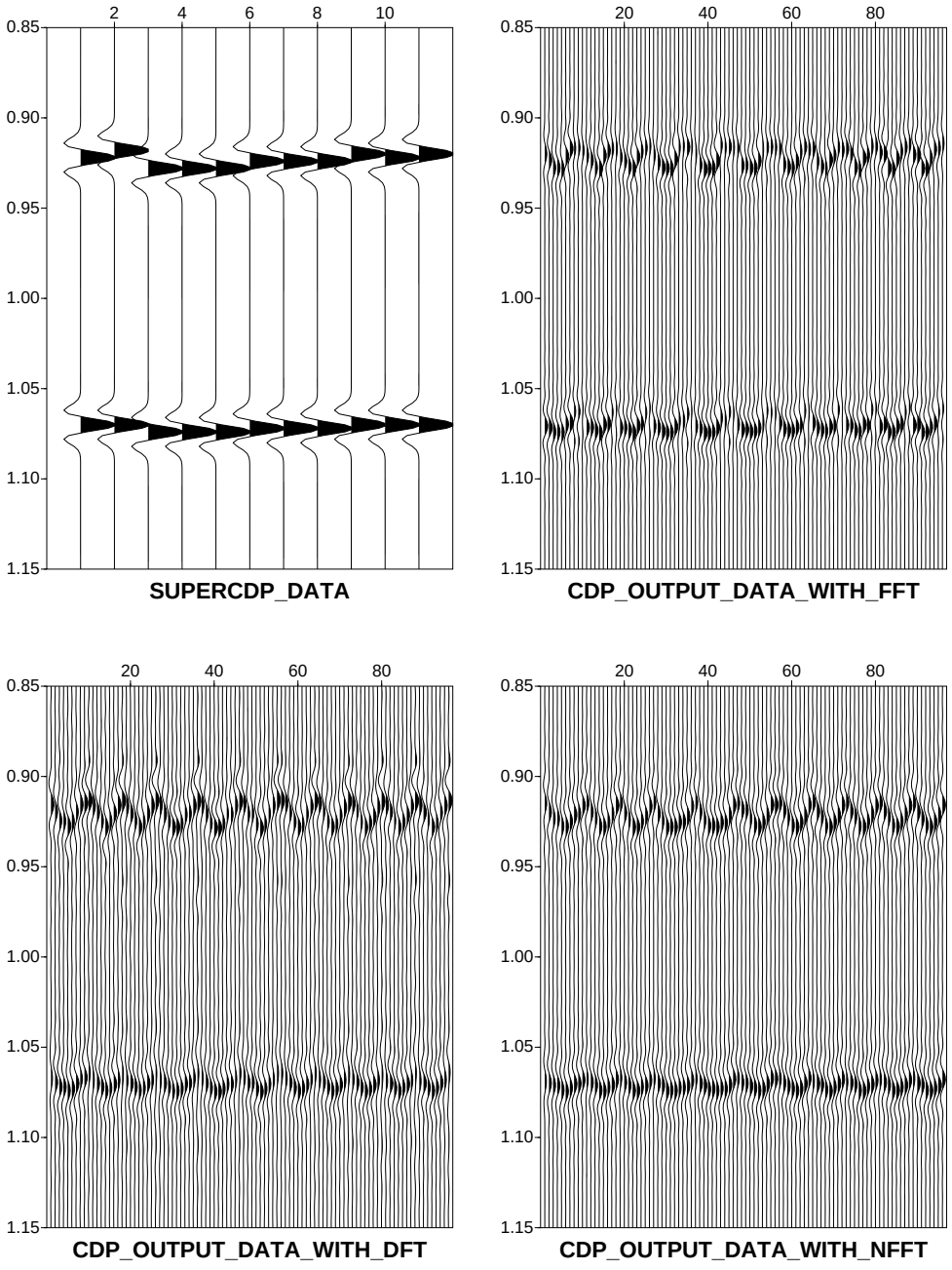

FIG. 5. Near offset comparison for a sparse geometry with azimuth truncation. a) Super-cdp in input (3 cdps). b),c),d) regularized single cdp using FFT4D, DFT and NFFT operators

The final test is a comparison of the three operators for a real data set from Brooks, Alberta (CO_2 sequestration project). The geometry is orthogonal, with 100 meters line spacing and 10 meter group spacing (5x5 bin size) as seen in Figure 6. The data are not structured, which unfortunately make the test a bit limited, but it will serve to illustrate some differences between these methods in the context of very sparse surveys after removing some receiver lines.

For real 3D data, we usually have to use large windows in terms of inline/crossline to be able to deal with large minimum offsets in between lines. This makes computational differences more significant than in the previous synthetic tests. I eliminated from the input every second receiver line, making now the spacing 200m (or 40 bins). To allow each window to cover a full box I set the window size to 40 inlines $\times$ 40 crosslines. For the other dimensions, I use 20 offsets and 8 azimuths, with intervals of 50 meters and 45 degrees. Actually, these intervals are too large, but I let the program to handle this issue. Figure 7a shows one input group located at the center of the survey. We see the two large gaps from two missing receiver lines (data sorted by receiver location). Figure 7b,c,d shows the interpolations done with the FFT, DFT and NFT respectively. The three have some deficiencies, but overall the three results look reasonable. Let us now inspect more closely by selecting one of the removed receiver lines and compare with the original data not included in the interpolation.

Figure 8a shows a window into the original data without decimation, with all traces belonging to a removed receiver line (none of these traces where in the input). Figure 8b shows the corresponding created traces using the FFT algorithm. We can identify the same reflectors in both, although the FFT result is a bit cleaner as expected. Some fine details may be missing in the interpolated traces because of the large offset/azimuth binning interval, although the FFT algorithm has reduced this grid automatically by a factor a 2 in both offset and azimuth directions. Figure 8c contains the created traces for the DFT algorithm. The main difference with the FFT result is stronger filtering, because of the slower convergence of the algorithm with DFT (all tests used 30 iterations). See Trad (2009) for a discussion about this. Also, because of the high memory requirement of the DFT operator for these large windows I was forced to reduce by half the DFT Nyquist frequency for inlines and crosslines, which may explain some of the additional filtering (but not all since the data is quite flat). Figure 8d shows the result for the NFT algorithm. This result is a bit unexpected, because there is some low frequency energy that does not exist in the original traces. That may be explained by insufficient numerical regularization at low frequencies. During the FFT algorithm, most frequencies under 9Hz didn't reach 30 iterations, indicating insufficient signal in the data to do so, but for the DFT and NFFT these frequencies were calculated. These results suggest perhaps they should not have.

Figure 9 shows the statistics for the central group coming from the FFT algorithm. These values illustrate how the automatic gridding works. First line tells us that for the target coverage of the 4D spatial grid (3%) we only need 7680 alive traces, but the window has 42347. Because there are enough traces the program will reduce the offset and azimuth intervals by 2 giving new values of 25meters and 22.5 degrees. After doing that, the code is using 99% of the traces. The remainder 1% (315 traces) fall on the same 4D bins as other traces, so the program does not use them (but it will pass them to the output unchanged).

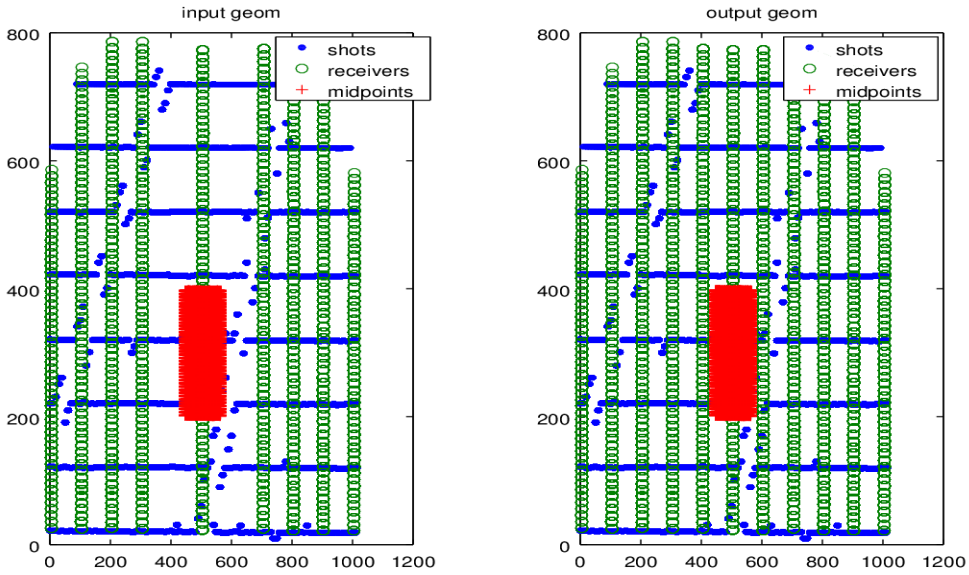

FIG. 6. Input and output geometries for Brooks data set, and one interpolation window

Of these traces in repeated bins, however, 62 are both new and therefore only one will be created. This leaves the output with missing 62 new traces. These can be changed by making them exactly equal to the new 62 traces created. During the processing of this group, 11682 were created for receiver lines 400 and 600. The values ndim and ldim show the length along each dimension before and after zero padding. We see we have now 40 offsets and 16 azimuths instead of the original settings of 20 and 8 respectively. The output traces are quite reasonable and would help during migration.

Figure 10 show some information coming from the NFFT libraries, indicating that the algorithm used polynomials of degree 3 to interpolate, but the FFT inside each dimension used 52, 52, 26, and 10 points respectively. A very small oversampling was used, close to 1.1 to reduce computation time and memory usage. The time to calculate the plan was of 11 seconds which is not significant. Each frequency took approximately the same time to calculate, close to 27 seconds. This is different from the DFT approach where frequencies take increasingly more time with frequency because of the cone shape of the multidimensional Fourier spectra.

The computational time for the FFT approach was only 8 minutes, the DFT approach took 3hours, the NFFT approach took 50 minutes. In these tests, I limited the frequencies to a maximum of 60Hz, but these differences would become larger as going to higher frequencies since the FFT approach has constant effort with frequency, but the DFT becomes slower. For the DFT to be efficient in a production environment, windows must be smaller, like in the previous tests. Alternatively, it is possible to implement DFTs in GPUs or Xenon Phi processors. For this test the best results have come from the FFT approach, but that is influenced by the large window size, and the simple structure of the data tests. For more complex environments, as suggested by the previous synthetic tests differences may be in favour of the DFT or NFFT approaches. Further tests for complex environments will be followed in a future paper.

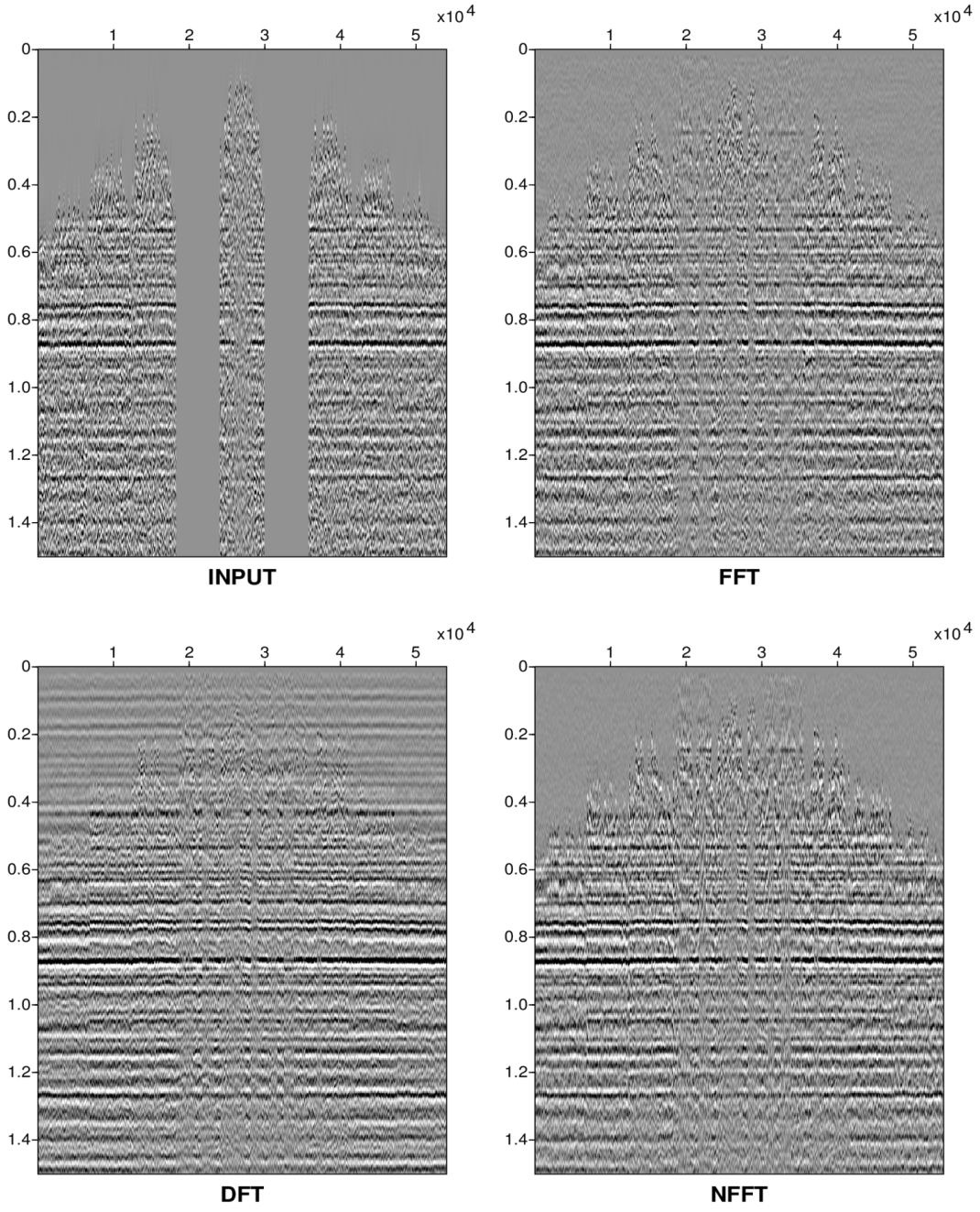

FIG. 7. Comparison for one zone of 40 inlinex $\times$ 40 crosslines, whith offsets between 0-1000 meters. Traces sorted by receiver location. a) Input with missing receiver lines, b) FFT, c) DFT, d) NFFT

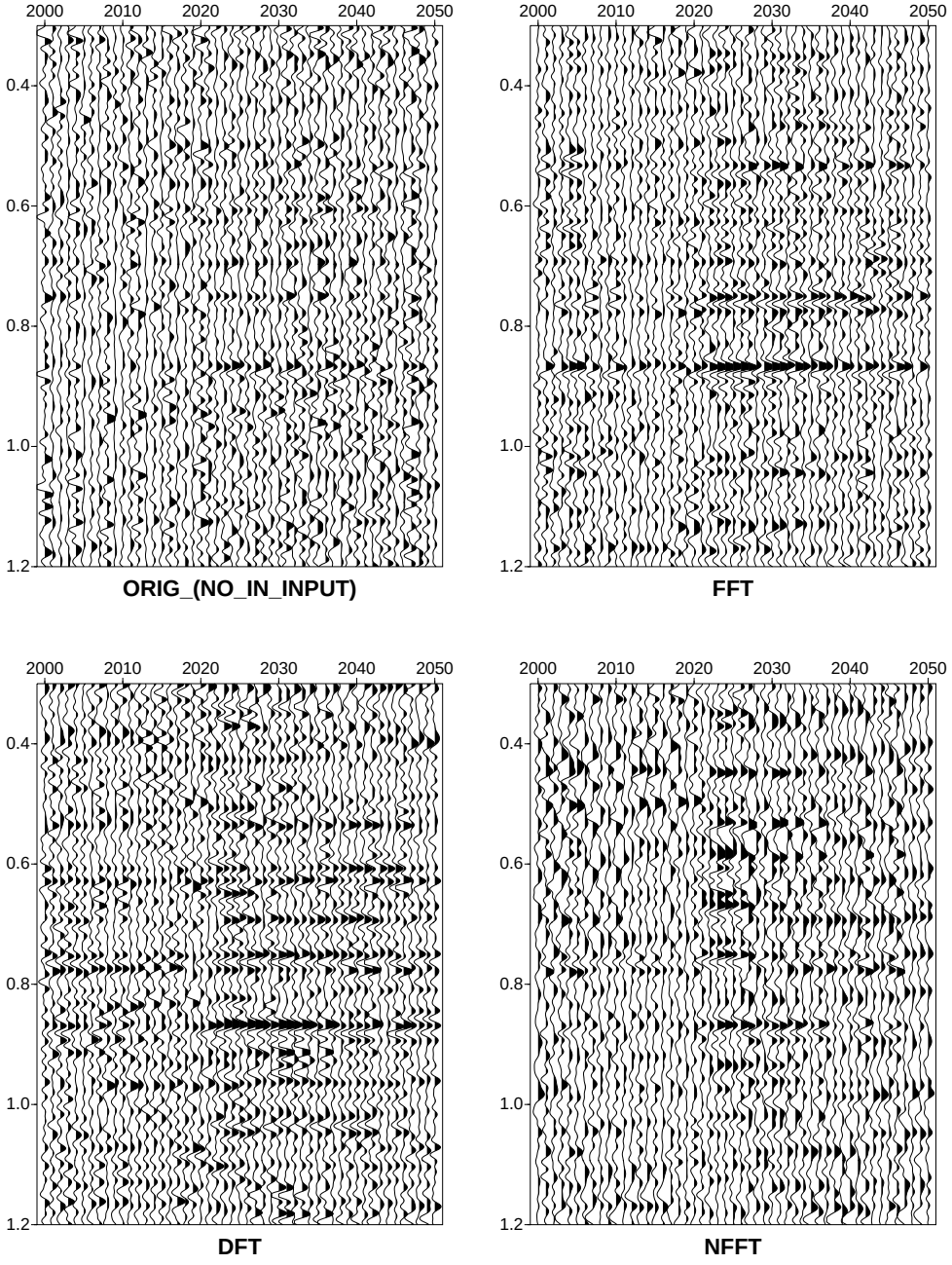

FIG. 8. Comparison between an original receiver line, absent from the calculation, and the created receiver line using the three operators.

```
 target in percentage 3 , alive traces 42347 , required 7680
 decimation along 2==>2
 decimation along 3==>2
 INFO: number of traces to use = 53715
 INFO: number of repeated traces = 315
 INFO: percentage of used traces = 99
 INFO: percentage of orig/grid traces = 20
 REPEATED: 2 NEW = 62
 REPEATED: 2 ORI = 240
 REPEATED: ORIG-NEW = 13
ndim[0] = 40
ldim[0] = 48
ndim[1] = 40
ldim[1] = 48
ndim[2] = 40
ldim[2] = 48
ndim[3] = 16
ldim[3] = 20
```

FIG. 9. Log information illustrating how the automatic gridding works for FFT

```
nx = 54030
N=703040
do not use threads for nfftw
 NFFT init operator done in 19 ms
plan.m=3
plan.M_total=54030
plan.N[0]=52
plan.N[1]=52
plan.N[2]=26
plan.N[3]=10
plan.n[0]=57
plan.n[1]=57
plan.n[2]=28
plan.n[3]=11
plan.sigma[0]=1.09615
plan.sigma[1]=1.09615
plan.sigma[2]=1.07692
plan.sigma[3]=1.1
 NFFT scaling done in 2 ms
PRECOMPUTE PLAN
 NFFT precompute done in 11471 ms
freq = 2.92969 status = 0
qdatascale=0.242597  completed 30 iterations freq = 2.92969  done in 27042 ms
datascale=0.219276  completed 30 iterations freq = 3.41797  done in 26601 ms
datascale=0.207111  completed 30 iterations freq = 3.90625  done in 26497 ms
datascale=0.196183  completed 30 iterations freq = 4.39453  done in 27061 ms
datascale=0.214673  completed 30 iterations freq = 4.88281  done in 26720 ms
datascale=0.203725  completed 30 iterations freq = 5.37109  done in 27775 ms
datascale=0.211676  completed 30 iterations freq = 5.85937  done in 26698 ms
datascale=0.251943  completed 30 iterations freq = 6.34766  done in 27399 ms
datascale=0.298891  completed 30 iterations freq = 6.83594  done in 27143 ms
datascale=0.294947  completed 30 iterations freq = 7.32422  done in 25631 ms
datascale=0.337757  completed 30 iterations freq = 7.8125  done in 26364 ms
datascale=0.375628  completed 30 iterations freq = 8.30078  done in 25994 ms
datascale=0.415022  completed 30 iterations freq = 8.78906  done in 26208 ms
```

FIG. 10. Log information illustrating how the NFFT algorithm works.

CONCLUSIONS

In this paper, we have discussed how three different Fourier operators behave during 5D interpolation. By implementing one program with the three options this work attempts to make a fair comparison by keeping all the same except the mapping between data and model. The first operator, standard multidimensional FFT with binning, seems to work quite well as long as binning is carefully implemented to adjust to differences in fold across different areas of a survey and offset ranges. DFT operator is more precise and flexible but computationally very expensive. This operator is however a good benchmark tool to understand how approximations to exact locations affect the other operators. NFFT operator is a compromise in terms of speed of flexibility. Although very promising in the synthetic examples, it seems to have some complications for very large windows and it needs some fine tuning. When very large windows are required, like in the case of sparse orthogonal geometries, the efficiency and performance of the FFT operator is still very superior to the other two. This conclusion however may not hold for very structured data, as the synthetics with HTI anisotropy showed. This paper represents just a first step in understanding how different operators compare, and further investigation with other data sets and non-Fourier operators will be necessary to complete the conclusions in this work.

ACKNOWLEDGMENTS

I thank CMC Research Institutes, Inc (CMC). for access to the seismic data and CREWES sponsors for contributing to this seismic research. I also gratefully acknowledge support from NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 461179-13.

REFERENCES

Claerbout, J., 1992, Earth sounding analysis, Processing versus inversion: Blackwell Scientific Publications, Inc.

Duijndam, A. J. W., and Schonewille, M. A., 1999, Nonuniform fast fourier transform: Geophysics, **64**, No. 2, 539–551.

Gulati, A., and Fergusson, R., 2009, NFFT: Algorithm for irregular sampling: CREWES Research Report, **21**.

Keiner, J., Kunis, S., and Potts, D., 2009, Using nfft 3—a software library for various nonequispaced fast fourier transforms: ACM Trans. Math. Softw., **36**, No. 4, 19:1–19:30.
URL `http://doi.acm.org/10.1145/1555386.1555388`

Liu, B., and Sacchi, M. D., 2004, Minimum weighted norm interpolation of seismic records: GEOPHYSICS, **69**, No. 6, 1560–1568, `http://dx.doi.org/10.1190/1.1836829`.

Sacchi, M. D., and Ulrych, T. J., 1996, Estimation of the discrete fourier transform, a linear inversion approach: GEOPHYSICS, **61**, No. 4, 1128–1136.

Trad, D., 2009, Five-dimensional interpolation: Recovering from acquisition constraints: GEOPHYSICS, **74**, No. 6, V123–V132.

Trad, D., 2014, Five-dimensional interpolation: New directions and challenges: CSEG Recorder, March, 40–46.

Trad, D., Ulrych, T., and Sacchi, M., 2003, Latest views of the sparse radon transform: GEOPHYSICS, **68**, No. 1, 386–399.