2D/3D Kirchhoff-based Wavefield Continuation

Anthony Barone[1] and Anthony Vassiliou[2]*
[1]The University of Texas at Austin, Institute for Geophysics, [2]GeoEnergy Inc., Houston, Texas

Summary

In this paper, we overview both 2D and 3D Kirchhoff-based Wavefield continuation, and outline its usefulness for re-datuming / topography correction and spatial sampling regularization. We will quickly describe the background theory, broadly overview both 2D and 3D implementations, and show some example results of pre- and post-continuation data.

Introduction

Many important data processing techniques that are used in typical seismic processing workflows rely on one or both the of following assumptions: 1) that data are spatially located on a regular grid, and 2) that data are all located at the same elevation. Seismic surveys are often designed such that these assumptions are close to correct, but (in particular for land data) they are never exactly met. Kirchhoff-based Wavefield continuation offers an effective and efficient method for re-datuming seismic data to accurately account for the effects of topography. This method inherently produces an output that is regularly gridded at user-specified intervals, which both regularizes the spatial gridding of the data and "fills in" minor holes in the data coverage. Combined, these effects ensure the aforementioned data processing assumptions are met rather than just approximated, which in turn will improve the accuracy of all subsequent data processing steps in a given seismic processing workflow.

In this work, we overview the theory and implementation of both 2D and 3D Kirchhoff-based wavefield continuation. We implement these continuations in the frequency domain, where the generalized workflow is:

> *Data FFT → LOOP over frequencies: {LOOP over output grid: {window data around output location → apply Kirchhoff weights → sum}} → Data IFFT.*

We implement this such that, at a given frequency, the full *{window → weight → sum}* operation is implemented using a single [sparse] (matrix)*(matrix)*(matrix) operation. The 2D and 3D algorithms are functionally identical, except that 1) the Kirchhoff weights are computed with a different formula, and 2) the 3D case includes some additional indexing steps. Due in large part to the availability of extremely efficient and open-source FFT and BLAS algorithms, the entire wavefield continuation process is extremely efficient. For example: a when run on a single consumer-class CPU (our tests used Sandy-Bridge and Ivy-Bridge i7's), a 1-way continuation of the ~6 GB open-source Stratton 3D dataset ran in a matter of minutes and had relatively modest memory requirements, making it feasible for use with large-scale problems without needing supercomputer-level hardware.

Background

The theory behind Kirchhoff-based Wavefield continuation dates back to the late 1970's and early 1980's, and was largely due to the work of John Berryhill (e.g., Berryhill 1979, 1984, 1986) and others (e.g., Berkhout, 1982). There have been subsequent refinements/modifications to the method (e.g., Bevc, 1995 and 1997), though these are dealing more with implementation than with the underlying theory. We refer you to these papers for a more thorough derivation of the method, but the "important" equation, often referred to simple as "the Kirchhoff Integral", is shown below in Equation 1. The analogous discrete version is shown in Equation 2.

$$P(\boldsymbol{r}_{out},\omega) = \oiint P(\boldsymbol{r}_{in},\omega)\left(\frac{\partial r}{\partial n}\frac{1-i\omega\tau}{2\pi r^2}\right)e^{i\omega\tau}dS\,, \quad (1)$$

$$P(\boldsymbol{r}_{out},\omega) = \sum_x\sum_y P(\boldsymbol{r}_{in},\omega)\left(\frac{\partial r}{\partial n}\frac{1-i\omega\tau}{2\pi r^2}\right)e^{i\omega\tau}dx\,dy\,, \quad (2)$$

where $P(\boldsymbol{r},\omega)$ is the (pressure) wavefield, $\boldsymbol{r}$ is spatial location of the input or output, ω is angular frequency, τ is traveltime between $\boldsymbol{r}_{in}$ and $\boldsymbol{r}_{out}$, r is the Euclidean distance between $\boldsymbol{r}_{in}$ and $\boldsymbol{r}_{out}$, and n is a unit vector that is orthogonal to the input datum surface (S). The $\frac{\partial r}{\partial n}$ term is related to $\cos\theta$ between r and n, and is related to geometric spreading. Equations 1 – 2 correspond to a Green's function of $G = \frac{e^{ikr}}{r} - \frac{e^{ikr'}}{r'}$, where $k = \frac{\omega}{r}$. Since we use a frequency domain implementation, Equation 2 can be directly evaluated for the 3D case. For the 2D case, the $\oiint(\ldots)dS$ integral can be transformed to $\iint(\ldots)dxdy$ and the wavefield can be assumed invariant in either x or y, producing the following solution:

$$P(\boldsymbol{r}_{out},\omega) = \int P(\boldsymbol{r}_{in},\omega)\int\left(\frac{\partial r}{\partial n}\frac{1-i\omega\tau}{2\pi r^2}\right)e^{i\omega\tau}dxdy = \int P(\boldsymbol{r}_{in},\omega)\left(\frac{\partial r}{\partial n}\frac{\sqrt{-i\omega\tau}}{r\sqrt{2\pi}}\right)\left(1-\frac{1}{i\omega\tau}\right)e^{i\omega\tau}dx\,. \quad (3)$$

Equation 3 can be discretized in the same way Equation 2 was. Equations 2 and 3 form the basis of the 3D and 2D Kirchhoff wavefield continuation methods, respectively. One can implement this using a simple nested loop, though it is possible to achieve a speed improvement of several orders of magnitude by using a matrix-based approach.

2D/3D Kirchhoff-Based Wavefield Continuation

Method

The exact specifics of the algorithm are proprietary information, but in this section we will broadly overview the steps used. All work was done in MATLAB, which is inherently designed to handle matrix operations such as these quickly and easily. It is worth quickly noting that all matrix operations in MATLAB are implemented using an external BLAS/LAPACK library (in this case Intel's MKL is used), meaning that these operations in effect only rely on MATLAB to act as a wrapper. MATLAB is frequently thought of as a "slow" language, and admittedly often is, but any matrix-based operation in MATLAB runs effectively as fast as natively calling the same BLAS/LAPACK library from a "fast" language like C/C++ or Fortran would. The same applies to FFT's, which MATLAB implements using the FFTW library.

Our Kirchhoff-based wavefield continuation algorithm follows the following general steps. Note that we describe this in terms of "source" and "receiver" indices, though this same process can be applied to other data sorts as well.

1. ***Data Loading and FFT***. This is fairly straightforward. Depending on available memory and the data size, one may want to reorder the data such that primary sort is frequency and resave it to disk. Zero-padding the data is optional but recommended – the up/downward continuation implements a circular shift, and without zero-padding any data shifted above/below the top/bottom of the data's time window will be circular shifted to the bottom/top of the data.

2. ***Gridding Analysis***. The data needs to be resorted such that each source point and each receiver point each have a unique row and column in the data. This produces a data array of size [# frequencies, # shots, # receivers]. This array will have sparsity equal to the average of $\frac{\#\ active\ receivers}{\#\ total\ receivers}$ from all shots.

   (*Optional*) if one wants to add reciprocity to the data, it should be set up and perhaps implemented in this step.

3. ***Computing Kirchhoff Weight Precursors***. Prior to beginning the main summation loop, it is beneficial to pre-compute all the frequency-independent quantities required to compute the Kirchhoff weights. This includes, for example, distances and traveltimes. Two weighting arrays need to be constructed – one that weights data grouped by source index and one that weights data grouped by receiver index. In addition to the Kirchhoff summation weights, we incorporate a Tukey/cosine transition window at each data edge directly into these weighting arrays. When generating weights one may choose to use the full weight or drop the "near-field" term for a "far-field" approximation.

4. ***Main Loop Over Frequencies***. At each frequency, one needs to first construct the actual Kirchhoff weights by incorporating the frequency dependent effects into the "weight precursors" computed in step 3. Because weights depend on either source or receiver index but not both, the weights can be applied using 2 split operations (this is mathematically equivalent to splitting a 2D convolution into two 1D convolutions). If the entire dataset has not been converted to use the gridding scheme found in Step 2, then the currently frequency needs to be extracted and converted to that scheme.

The beauty behind this approach is in how simply and efficiently the actual summation is implemented. Because we have weights that can be split by source and receiver indices and we have data that has been reordered such that each row and column corresponds to a particular source or receiver, we can implement the summation throughout the entire dataset (at a given frequency) using a single (matrix)*(matrix)*(matrix) operation of the form:

$$D_{NEW} = W_R\, D_{OLD}\, W_S\,, \tag{4}$$

where W is an array of Kirchhoff summation weights for source-grouped or receiver-grouped data and D represents the data. These arrays can be either dense or sparse, though (in particular in the 3D case) there is often a substantial benefit from making them sparse. Note that multiple continuations can be combined into a single operation of the form $(W_R^1 \ldots W_R^N D_{OLD}\, W_S^N \ldots W_S^1)$. Also note that only the positive frequencies need to be run in the main loop – the desired output signal is real, so positive and negative frequencies must be complex conjugates of each other.

5. ***Data IFFT***. After finishing and main loop, a standard IFFT was performed to bring the data back to the time domain, and the data were resorted so that the primary sort was trace number, not time/frequency. Depending on your data processing workflow the IFFT and data resorting may or not be required.

Results and Discussion

We test our codes using a synthetic 2D dataset and a real 3D dataset. The 2D dataset is an extremely dense and regularly sampled dataset with offsets up to 20+ km, and serves as a near-ideal test to ensure the theory is working correctly. The 3D dataset is the open-source Stratton 3D data available from SEG (in particular, we used the SEGY files named *swath_[1-4]_geometry.sgy*). This serves as a more realistic test case with less dense and irregularly gridded data. We will show example images from both the 2D and 3D continuations and then will quickly overview the efficiency algorithm (with a focus on the 3D continuation).

2D/3D Kirchhoff-Based Wavefield Continuation

2D Wavefield Continuation with Synthetic Data

Figure 1 shows a snapshot of a single shot gather being upward continued and then downward continued back to the original datum. This is intended as a proof-of-concept, and so data is densely sampled, largely noise-free, and on a flat regular grid, representing a best-case scenario. The 5 sub-images in Figure 1 show the original data (a), the upward continued data with far-field weights (b) and full weights (c), and the upward + downward continued data with far-field weights (d) and full weights (e). This clearly shows that under ideal conditions the algorithm produces a nearly 100% error free continuation of the Wavefield. As expected, using only the far-field weights introduces some error at very near offsets that is not present when using the full Kirchhoff summation weights.

3D Wavefield Continuation with the Stratton 3D Dataset

Figure 2 shows the source/receiver layout of the Stratton 3D dataset, and is colored based on (interpolated) surface elevation. Since we used reciprocity, the shots and receivers are interchangeable from each other. Figure 3 shows the sparsity pattern of the re-gridded input data. We incorporated reciprocity, resulting in just under 2,500 sources and 2,500 receivers on an irregular sparse grid with ~977,000 traces (meaning ~16% of the grid points contain data). Note that the order of the shot/receiver indices is unimportant, it only matters that they have a unique row/column associated with them. This allows one to optionally reorder the data to attempt to improve the speed of computations with it. Figure 3 shows the data reordered using an Approximate Minimum Degree algorithm. Lastly, Figure 4 shows an example receiver line from the dataset upward continued 1500±20m (max surface elevation relief is ~40m). We used a replacement velocity of 3,000 m/s, thus the 1st arrivals should be at a 2-way time of ~1000 ms.

Efficiency

We will focus on the Stratton 3D data for describing efficiency, as this data is closer to real-world usage than our synthetic 2D example. This dataset contains ~6 GB of data, with 3000 time samples ($\Delta t = 2\ ms$). We used data from nearly all sources and receivers (a few locations along the survey edges were removed), for a total of about 500,000 traces (~977,000 after adding reciprocity). To conserve memory we down-sampled and windowed the data in time, though this only effects the number of iterations in the main loop – the time per iteration and the initial setup are unaffected by this (with the exception of the FFT/IFFT steps, though these are extremely fast regardless). All computations used double precision floats (mostly since MATLAB has poor support for sparse single matrices). The execution time break-down was as follows:

1. Auxiliary setup: ~180 Seconds (3 minutes)
   - Data/header loading + ibm2ieee: ~45 seconds
   - FFT + IFFT + Data resorting: ~45 seconds
   - Re-gridding analysis and setup: ~60 seconds
   - Tukey window application + Other: ~30 seconds

2. Main Loop: 3.8 ± 0.2 Seconds Per Frequency
   - ~3.5% to extract and re-grid current frequency
   - ~1.5% to generate frequency-specific weights
   - ~95% to implement $(W_R\ D\ W_S)$ matrix operation
   - Again note that only positive frequencies need to be run. 1 frequency in → 2 time samples out.

These results were implemented using a 4C/8T Ivy-Bridge i7 running at ~3.1 GHz. It is worth noting that this CPU is missing some features (notably AVX2 and FMA) and uses slower DDR3 memory when compared to more modern CPU's. These missing features can be utilized by most BLAS/LAPACK implementations for matrix operations, suggesting that the time per frequency should be significantly less (likely less than half) by simply switching to a newer CPU architecture.

Conclusion

In this work we overview the theory behind 2D/3D Kirchoff-based wavefield continuation and outlined a very efficient matrix-based method the implement both 2D and 3D continuations. We demonstrated this method works and runs quickly enough to process large datasets with relatively standard consumer-grade hardware. Wavefield continuation has the promise of becoming an excellent tool for pre-processing data before "standard" seismic processing workflows begin in order to accurately transform the data to a flat and regularly gridded datum, which is required by many standard processing methods. An efficient and accurate continuation tool has the potential to significantly improve numerous data processing workflows with a minimal amount of extra computational expense, making it a potentially extremely valuable tool.

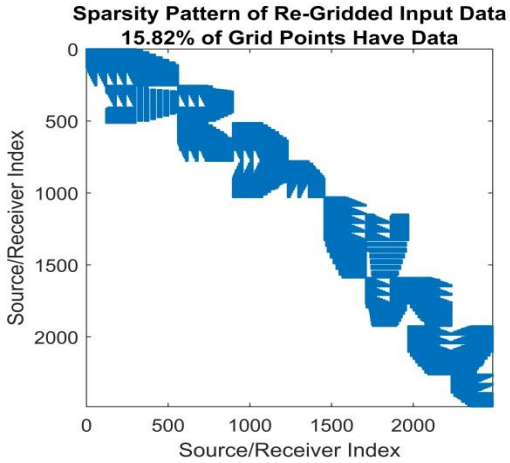

Figure 3: The sparsity pattern of the data after being re-gridded to have each column and each row represent a unique shot / receiver.

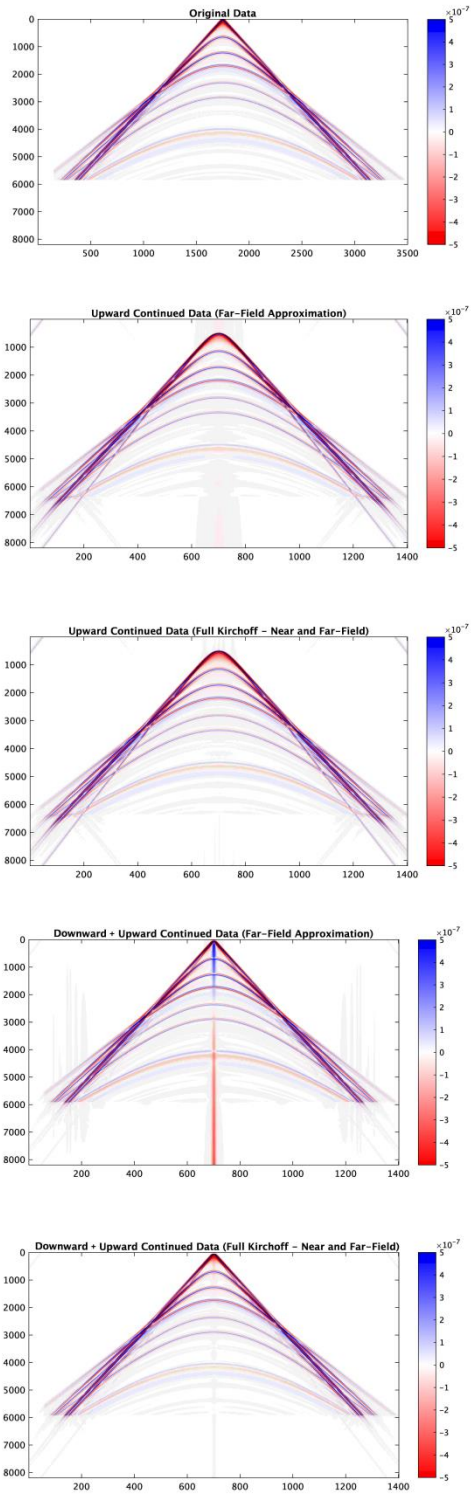

Figure 1: a set of images showing the 2D wavewfeild continuation. (Top to Bottom): (a) the original data, (b) the upward continued data with only far-field weights, (c) the upward continued data witrh near- and far-field weights, (d) the upward+downward continued data with only far-field weights, (e) the upward+downwardd continued data with near- and far-field weights,

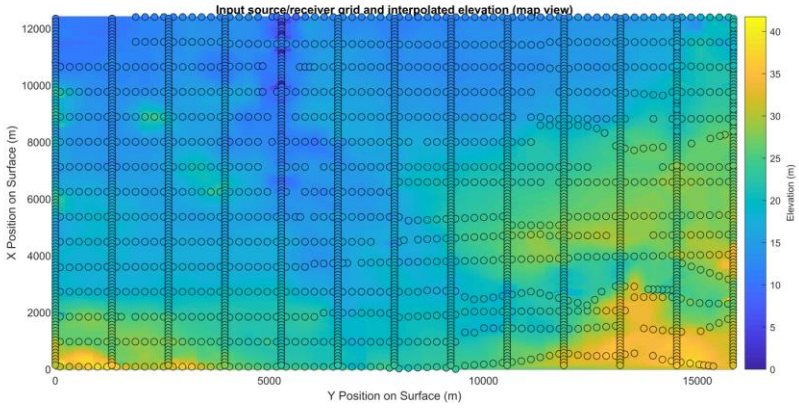

Figure 2: The grid layout of sources and receivers, and the (interpolated) surface elevation. Color represents the surface elevation, with a maximum elevation relief of just over 40m.

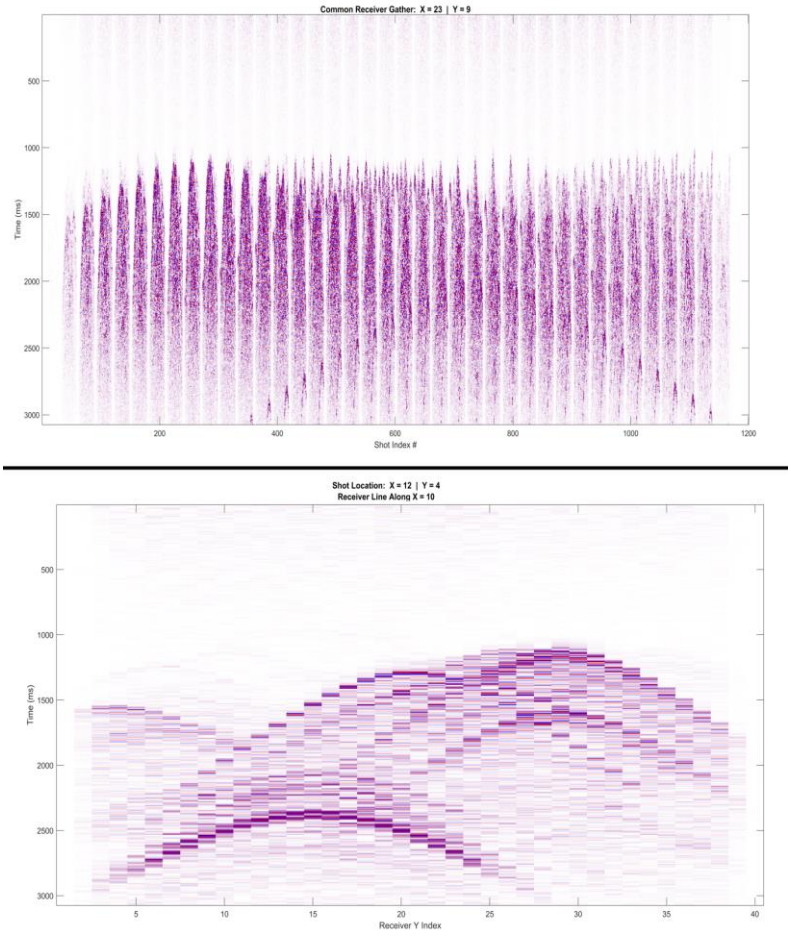

Figure 4: Two example figures fhowing the output of the 3D wavefield continuation. The upper figure is a common receiver gather showing all shot lines for 1 selected receiver. Note that due to reciprocity this is identical to the shot gather with the same position. The lower figure shows a common shot gather, but isolates one particular receiver line with a constant X value. The X and Y labels are in terms of indicies, where [X, Y] is on a [30, 40] grid.