Copyright
by
Sarah Yvonne Greer
2018

The Thesis committee for Sarah Yvonne Greer
Certifies that this is the approved version of the following thesis:

A data matching algorithm and its applications in seismic data analysis

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Sergey Fomel, Supervisor

Dr. Kyle Spikes

Dr. Clark Wilson

A data matching algorithm and its applications in seismic data analysis

by

Sarah Yvonne Greer

THESIS

Presented to the Faculty of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

BACHELOR OF SCIENCE WITH HONORS

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2018

Acknowledgments

I have received an incredible amount of support during my time as an undergraduate student. First and foremost, I'd like to thank my advisor, Dr. Sergey Fomel, for his helpful guidance over the past couple of years. I genuinely believe that working with him has beneficially shaped the trajectory of my life. Through his guidance and the opportunities he has given me, he has helped me to realize how much I truly enjoy learning, showed me the importance of reproducible computational research, introduced me new ways of thinking about and approaching problems, and inspired me to continue my studies in graduate school. For all that, I will be forever grateful.

Many professors from the geological sciences and mathematics departments with whom I have interacted have also greatly helped me during my undergraduate career. I am especially thankful to Dr. Todd Arbogast, Dr. Jaime Barnes, Dr. Mark Cloos, Dr. Luc Lavier, Dr. Francesco Maggi, Dr. Richard Tsai, and my committee members, Dr. Kyle Spikes and Dr. Clark Wilson, for their guidance during my time at UT Austin.

I am also grateful to many students and research scientists I've had the pleasure of interacting with: Sean Bader, Luke Decker, Preston Fussee-Durham, Zhicheng Geng, Eric Goldfarb, Ben Gremillion, Tom Hess, Ken Ikeda, Harpreet Kaur, Dr. Dmitrii Merzlikin, Nam Pham, Mason Phillips, Kelly Regimbal, Karl Schleicher, Yunzhi Shi, Dr. Junzhe Sun, Dr. Yanadet Sripanich, Dr. Xinming Wu, Dr. Zhiguang Xue, and Dr. Yunan Yang.

Last but not least, I would like to express my sincere gratitude to my family

for all their love and support over the years. Esme, Joel, Sam, Atticus, and Kali—you all mean the world to me and I wouldn't be the person I am today without you in my life.

SARAH YVONNE GREER

The University of Texas at Austin
April 2018

A data matching algorithm and its applications in seismic data analysis

Sarah Yvonne Greer, B.S.

The University of Texas at Austin, 2018

Supervisor: Dr. Sergey Fomel

Data matching applications are present in many computational geophysics problems. In this thesis, I introduce a new algorithm to aid in data matching that balances local frequency content between seismic data sets. Then, I provide a few examples where applying this algorithm to seismic data helps improve results. These applications include multi-component image registration, matching and merging high-resolution and legacy seismic images, and improving migration resolution by approximating the least-squares Hessian. The applications are applied to several 2D and 3D, real and synthetic data sets.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data matching is a conceptually simple problem that is present in many computational geophysics applications. Essentially, given two data sets, data matching can be thought of as finding the transformation from one data set to the other. This is useful in applications such as multi-component seismic image registration, where two separate seismic images, one P-wave image and one S- wave image, are acquired over the same area. These must be properly matched, as the two images are not temporally aligned and have different frequency and amplitude content, in order for them to be directly compared and interpreted.

Many other geophysics applications can be addressed from a data matching standpoint, and fall under a few main categories. Applications are present for data of different physics, such as tying synthetic seismograms from well log data to surface seismic data (Herrera et al., 2014; Bader et al., 2018) and multicomponent seismic image registration (Fomel and Backus, 2003a; Fomel et al., 2005; Hardage et al., 2011), which is discussed in chapter 3. Other problems consist of matching differently acquired data, such time-lapse image registration (Fomel and Jin, 2009) and merging legacy and high-resolution seismic data (Greer and Fomel, 2018), which is a primary application in this thesis and is addressed in chapter 4. Data matching problems are also present for matching data and ideal models, such as in deconvolution or approximating the inverse Hessian to improve migration resolution (Hu and Schuster,

1998; Guitton, 2017; Greer et al., 2018), which is discussed further in chapter 5. An approach to these many data matching problems is to find the operation to match the data using a combination of three data matching operations—filtering, scaling, and shifting. In this thesis, I will discuss these three data matching operators, outline several methods and applications of seismic data matching, and introduce a new method for matching frequency content between data sets.

The primary inspiration for much of the work in this thesis comes from the example of matching and merging high-resolution and legacy seismic images, as discussed in Greer and Fomel (2017b) and Greer and Fomel (2018). In this example, two seismic data sets, each acquired over the same area but with different technologies, are first matched in frequency, amplitude, and time, before being merged together to produce a third image. This new image includes the best signal characteristics from the two initial images while minimizing their weaknesses. Much of the theory behind Chapter 3 was developed in application to this example, but was later extended to other examples. As a result, I will be using data from this application throughout much of this thesis.

This thesis contains two primary data sets that I will be referring to. The first pair, henceforth called the *Apache* data sets, are two 2D lines, acquired over the same area but with different methods, from the Gulf of Mexico. These two images will always be plotted in the *seismic* color map.

The second pair, the *P-cable* data sets, are two 3D volumes, acquired over the same area but with different methods, from a different area from the *Apache* data sets in the Gulf of Mexico (Petersen et al., 2010; Meckel et al., 2017). These two images will always be plotted in grayscale. In parts of this thesis, I will show a 2D line from

this 3D data set for simplicity of plotting.

Two other data sets, both 2D, were used in this paper. In chapter 3, I use two multi-component images to demonstrate an adaptation for the proposed frequency balancing algorithm in an application of multi-component image registration (Fomel, 2007a). In chapter 5, I use the Sigsbee synthetic data set (Paffenholz et al., 2002) in application of improving migration resolution using non-stationary matching.

This thesis is organized as follows. In the second chapter, I overview the basics behind the three data matching operations—scaling, shifting, and filtering. In chapter three, I introduce two methods for balancing local frequency content in seismic data sets. In the fourth chapter, I show the first example of data matching—matching and merging high-resolution and legacy seismic images. In the chapter five, I show an example of data matching to improve migration resolution. Finally, in chapter six, I provide concluding remarks.

Chapter 2

Background

Seismic data contain fundamentally non-stationary variations in attributes such as frequency and amplitude content. For example, Figure 2.1 shows a sample seismic image and its local frequency content (Greer and Fomel, 2017a). Data matching problems involve finding some transformation, or set of transformations, that can be applied to one data set to best match some attribute with another data set. In data matching applications, in particular, it is crucial to acknowledge the non-stationary variations present in seismic data. Therefore, any transformations we may apply in a data matching problem must be variant in all data dimensions to account for these non-stationary variations.

Local seismic attributes are useful in the analysis of these non-stationary variations that are naturally present in seismic data Fomel (2007a). The calculation of these attributes uses iterative inversion with shaping regularization Fomel (2007b) to measure signal characteristics in local regions of data, rather than specifying windows or looking at instantaneous attributes. For the procedures described in this thesis, local attributes are better-suited attributes than instantaneous attributes. Take, for example, local frequency (Figure 2.1), as opposed to instantaneous frequency. Local frequency allows the comparison of frequency content in a local region of samples, as opposed to instantaneous frequency, which allows for a point by point comparison of frequency values between images. Because the corresponding reflections between

images may not be precisely aligned in time, using the local frequency attribute to balance frequency content between images would allow for more accurate frequency balancing than instantaneous frequency. Additionally, local frequency is a more geologically accurate attribute than instantaneous frequency because it honors time-frequency uncertainty and does not contain physically unrealistic negative or extremely high frequency values (Fomel, 2007a).

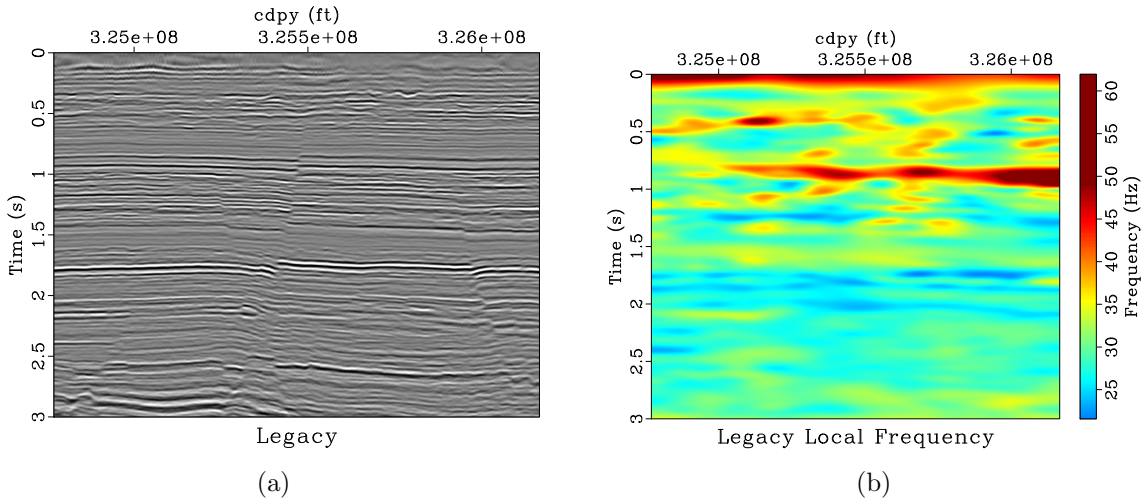


Figure 2.1: A seismic image (a) and its local frequency content (b). Because its local frequency varies both spatially and temporally, we can say its frequency content is *non-stationary*. chapter-background/../chapter-locfreq/merge legacy,low-freq

In this thesis, I propose solving data matching problems by balancing smoothly varying non-stationary attributes, such as local seismic attributes, between two data sets. This is done using three primary data matching operations—filtering, scaling, and shifting—to effectively balance these attributes across seismic data sets. These operations are applied to different data sets to correctly match them for analysis or further processing.

To demonstrate the three matching operations in action, I apply them con-

secutively to an example 2D trace shown in Figure 2.2(a). The data used in this example comes from a line from the *P-cable* data sets. In this example, we want to find a transformation, or a set of transformations, to apply to the red trace to match with the black trace. I use the three data matching operators by first filtering (Figure 2.2(b)), then scaling (Figure 2.2(c)), and finally shifting (Figure 2.2(d)). The three operations are described in more detail in the following sections.

FILTERING

Perhaps one of the most visibly obvious differences between data sets comes from differences in frequency content. For example, Figure 2.2(a) shows two traces that need to be matched; the red trace has visibly higher frequency content than the black trace which makes it difficult to observe that they may both contain data from the same model. This makes the correlation of these two signals, both visually and computationally, difficult as they have information in different frequency bands. In this situation, we want to remove the higher frequency variations from the red trace to match the lower frequency content with the black trace, since the high frequency content is not physically present in the black trace.

There are several ways to do this. An naive first approach would be to apply a bandpass filter to the red trace such that the passband covers only frequencies that are present in the black trace. However, this stationary operation does not take into account the non-stationary frequency variations that may be naturally present in the data. In order to properly match the data in frequency content, we propose balancing local frequency content between data sets.

We instead do this by applying a non-stationary triangle smoothing operator

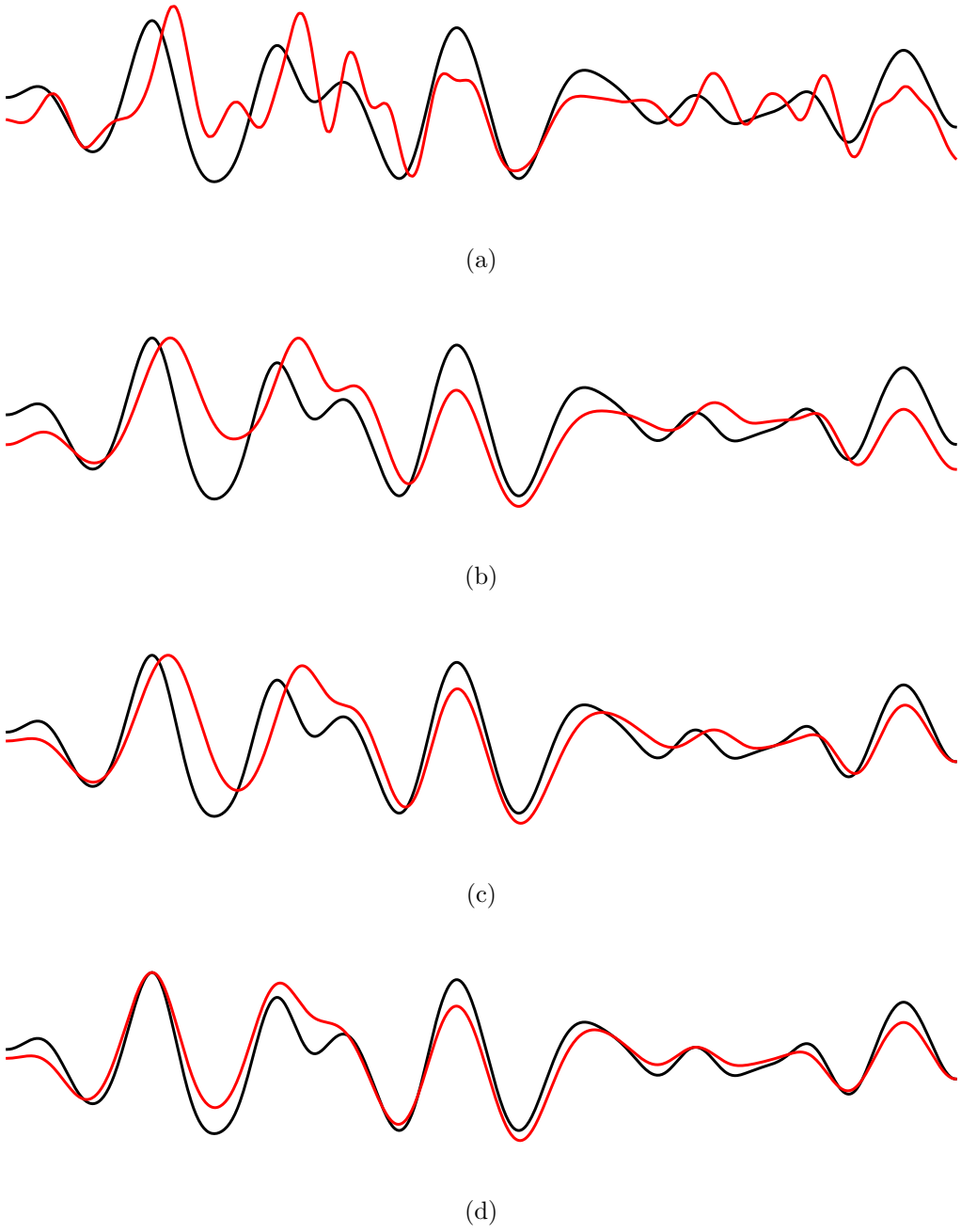


Figure 2.2: (a) Two traces that need to be matched—we will match the red trace to the black trace. (b) Red trace after filtering. (c) Red trace from (b) after scaling. (d) The final result: red trace from (c) after shifting, which now better matches with the black trace. The amount and order in which these operations is applied affects the final result (Figure 2.3). chapter-background/dmExample one0,one1,one2,one3

to the red trace to match local frequency content with the black trace. We define the *radius* of this operator as the number of samples in a specified dimension that are averaged over in a triangle weight. We allow the radius to vary in all dimensions to account for the potential non-stationary frequency variations present in the data. This is a linear operation—if we represent the filtering operation by a matrix applied to the trace represented by a vector, the matrix would be diagonally banded, where the size of the band at a particular index is related to the size of the smoothing radius at that point.

Chapter 3 is dedicated to the discussion of how to find this frequency balancing operator. Figure 2.2(b) shows the two traces after balancing the local frequency content of the red trace to match the black trace.

SCALING

The second primary data matching operation is scaling. Amplitudes of the data we are trying to match may not be initially correct—scaling attempts to balance the amplitudes between these data. Scaling is also a linear operation, and can be thought of as a diagonal operation; if the scaling operator is represented by a matrix multiplication to the data vector, this matrix only contains terms along the diagonal.

In this thesis, the diagonal scaling operator is found by first calculating the amplitude envelope of the data that need to be matched. Then, the scaling weight is calculated by smoothly dividing the amplitude envelopes of the two traces, and represent the diagonal elements of the scaling operator.

SHIFTING

If two events are slightly misaligned in time, finding the transformation that correctly aligns these events is referred to as *shifting*. Dynamic time warping is a common algorithm used for the alignment of events of two time series, both for geophysical and other applications, as diverse as speech recognition and finance (Herrera and van der Baan, 2012; Hale, 2013; Müller, 2007; Tsinaslanidis et al., 2014).

In this thesis, I calculate the shifting operator by finding the time shifts that maximize local similarity between the two data sets. Local similarity extends the concept of local seismic attributes to global correlation coefficients, and is effective for measuring the match of seismic events in a local region of samples (Fomel, 2007a). Unlike filtering and scaling, shifting is a non-linear transformation.

In certain situations, these data must be correctly aligned in space instead of time; for example, in a depth-migrated image. In these cases, *time* shift is a misnomer. However, since the data we are typically dealing with are time series, the shifting operation that we find and apply is considered a function of time.

The result after finding and applying the time shift to the red trace to match it to the black trace is in Figure 2.2(d). This shifting operation is discussed in more detail in chapters 3 and 4.

REPRESENTATION OF OPERATORS

Data matching problems are applicable for data sets that have the same physical model, yet have different characteristics that make their comparison difficult. For example, two seismic traces that were acquired over the same area but with different

acquisition methods would benefit by applying data matching operations before their direct comparison. These three operations can be applied to one data set to match it with the other.

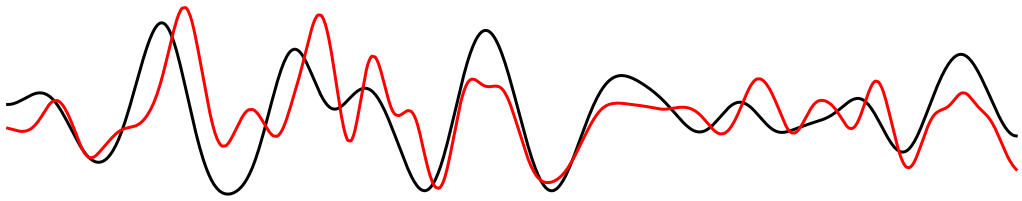
When applying one operation, we assume the other two operators have already been applied, when this may not be the case. This is why we balance *local* seismic attributes instead of instantaneous ones—we want the match to be accurate in a region of data points rather than a point-by-point match. These operators must be smooth enough such that any misalignment of one attribute does not affect the result of balancing the others. For example, if a scaling operator is applied before the traces have been correctly aligned in time by shifting, too precise of an amplitude balancing operation could inadvertently balance amplitudes to incorrect events.

The order and amount of these operators affects the final result. After correcting for frequency variations, amplitude variations, and time shifts, additional corrections can be applied to further refine the match if necessary. In some cases, multiple rounds of applying these operators and in different orders may be beneficial for the best match. These three operators are noncommutative, so the order in which they are applied matters. Applying filtering before scaling produces a different result from applying scaling before filtering. An illustration of this property is shown in Figure 2.3.

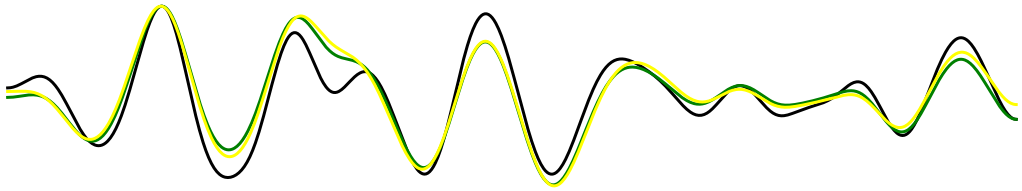
Operator	**Representation**
Shifting	$\mathbf{d}_2(x) = \mathbf{d}_1(x + \mathbf{s}(x))$
Scaling	$\mathbf{d}_2(x) = \mathbf{w}(x)\mathbf{d}_1(x)$
Filtering	$\mathbf{D}_2(k) = \mathbf{W}(k)\mathbf{D}_1(k)$

Table 2.1: Mathematical representation of the three data matching operators.

A summary of the three data matching operations is shown in Table 2.1. Both



(a)



(b)

Figure 2.3: (a) A lower-resolution seismic trace (black) and a higher-resolution seismic trace (red) acquired over the same area. (b) Data matching operations are noncommutative—the order in which they are applied matters. When matching the red trace to the black trace, the green trace had first smoothing, then amplitude balancing, and finally shifting; the yellow trace had first amplitude balancing, then smoothing, and finally shifting. The operation order matters and affects the final result. chapter-background/dmExample bef,aft

scaling and filtering are linear operations, while shifting is generally a non-linear operation. Additionally, as scaling is to weighting in the time domain, filtering is to weighting in the Fourier domain. The next chapter discusses a few methods and examples of filtering, or frequency balancing, between two data sets.

Chapter 3

Balancing local frequency content in seismic data using non-stationary smoothing

Seismic data can experience non-stationary frequency variations caused by attenuation. This problem is encountered when matching multiple data sets, such as in multicomponent image registration, because signals with differing frequency content are hard to correlate. In this chapter, we introduce two methods for balancing frequency content between data sets while taking into account non-stationary frequency variations. Both methods involve finding and applying a non-stationary smoothing operator to minimize the local frequency difference between data sets. Numerical examples demonstrate that the proposed method improves multicomponent image registration and matching images of differing resolution.

INTRODUCTION

Matching seismic data has many applications in geophysical processing methods, such as multi-component image registration, time-lapse image registration, matching well-ties to seismic data, and merging seismic data sets (Ursenbach et al., 2013; Fomel and Backus, 2003b; Lumley et al., 2003; Herrera and van der Baan, 2012). Typically, the workflow for matching data involves finding the optimal time shift, or amount

Parts of this chapter were first published in Greer and Fomel (2017a), Greer and Fomel (2017b), and Greer and Fomel (2018). This work was done under the supervision of Dr. Sergey Fomel.

of stretching and compressing, of one trace relative to the other that produces the greatest similarity between the two traces, as seen in dynamic time warping and local similarity scanning (Hale, 2013; Fomel and Jin, 2009; Herrera et al., 2014). However, when the two signals that need to be aligned have different spectral content, their comparison can be difficult. Most seismic data experience non-stationary, or spatially and temporally variant, frequency content caused by attenuation. This problem was discussed in application to multicomponent image registration by Fomel and Backus (2003b), who applied frequency balancing methods to improve registration results. Liu and Fomel (2012) proposed using local time-frequency decomposition (LTFD) to balance frequencies between multicomponent data during registration. However, LTFD is a relatively computationally expensive method.

In this chapter, we propose methods for removing non-stationary frequency differences that limit the effectiveness of matching data. We suggest applying the either of the proposed methods to processing flows that involve matching data before attempting to find the time shift to align their signal content. To balance frequency content, we use a non-stationary smoothing operator with an adjustable smoothing radius to apply to the higher frequency data set. Our first approach finds the smoothing operator directly, but is based on the primary assumption that the data can be modeled by a summation of Ricker wavelets. Our second approach of finding the smoothing radius is based on understanding what the smoothing operator physically does, and takes the form of an optimization problem which is solved using an iterative method. We introduce these methods and apply them to examples of merging high-resolution and conventional seismic images and multicomponent image registration.

METHOD

Two signals of differing frequencies are more difficult to correlate than signals of similar frequencies. For example, Figure 3.5 shows two seismic images representing the same subsurface, except they have distinctly different spectral content, as shown in Figure 3.7(a). In order to be directly comparable, these two images should have similar frequency content. Here, we look at local frequency (Fomel, 2007a), which can be thought of as a smoothed estimate of instantaneous frequency (White, 1991). Local frequency is a more geologically accurate attribute than instantaneous frequency because it honors time-frequency uncertainty and does not contain physically unrealistic negative or extremely high frequency values (Fomel, 2007a).

In order to balance local frequency content, we propose smoothing the higher frequency data using a non-stationary triangle smoothing operator with an adjustable radius. Here, the radius at each point is the number of samples in time that that specific data point gets averaged over in a triangle weight.

In order to find the temporally and spatially variable smoothing radius that we need to apply to the higher frequency image to balance local frequency content with the lower frequency image, we take two approaches. The first approach is finding the smoothing radius based on assumptions of what the data we are looking at can be represented by, and the second approach is based on understanding physically what non-stationary smoothing does to a data set. Both approaches work well in different situations—the first approach is less computationally expensive than the second approach, but is only applicable when the two data sets fit the assumptions that are used to calculate what the smoothing radius is. The second approach finds the smoothing radius in an iterative manner, but manages to work well for any data

set given to it.

Theoretical smoothing radius

To demonstrate the effectiveness of finding the smoothing radius using this method, we use the *Apache* seismic data sets. The two initial images are shown in Figure 3.1, and their frequency content are shown in Figure 3.2(a).

To balance the non-stationary frequency variations between data sets, we use a simple triangle smoothing operator with an adjustable radius. Here, the radius at each point is the amount of time, in seconds, that that specific data point gets averaged over in a triangle weight in the temporal direction. We specifically look at local frequency (Fomel, 2007a), which is a time-dependent frequency attribute, and can be thought of as a smoothed estimate of instantaneous frequency (White, 1991).

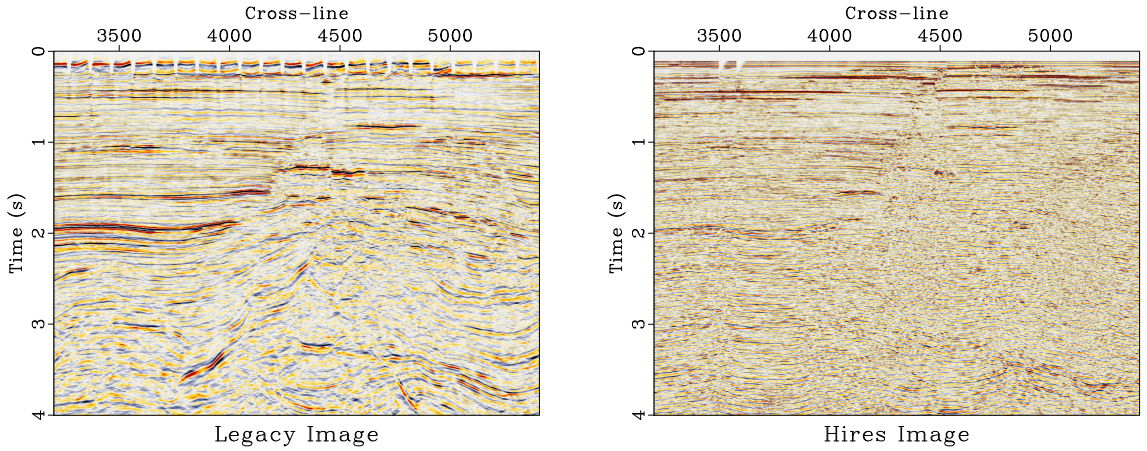
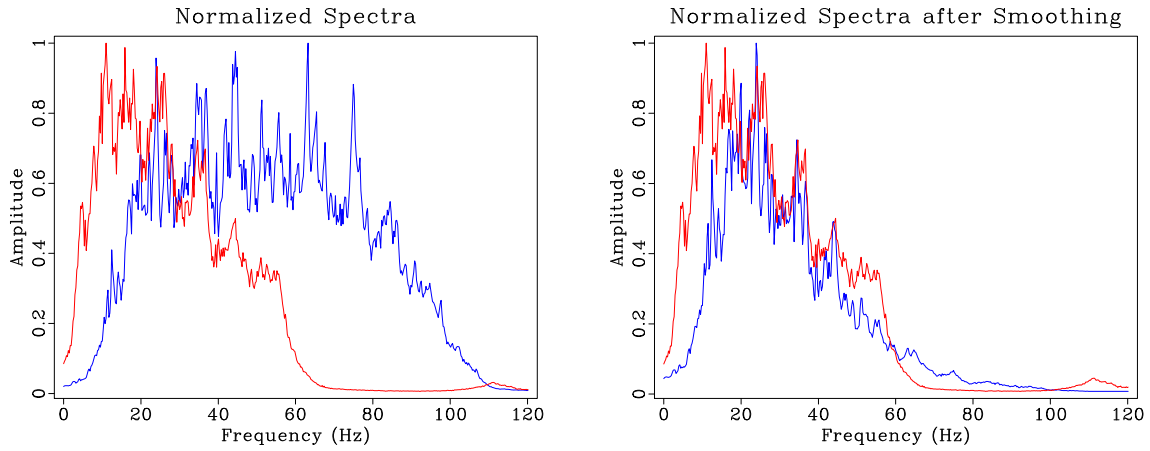


Figure 3.1: The initial legacy (a) and high-resolution (b) images. chapter-locfreq/../chapter-merge/apache legacy,hires

This method is based off of the primary assumption that the signal can be represented by Ricker wavelets convolved with the Earth's reflectivity series. Since



(a) (b)

Figure 3.2: Spectra for the entire image display of the legacy (red) and high-resolution (blue) images before (a) and after (b) spectral balancing using the theoretical method. chapter-locfreq/../chapter-merge/apache nspectra,hires-smooth-spec

the data we are working with are seismic, this can be a good assumption (Gholamy and Kreinovich, 2014).

The justification for triangle smoothing is that it is a simple approximation to Gaussian smoothing. The frequency response of the triangle smoothing filter (Claerbout, 1992) is

$$T(f) = \text{sinc}^2\left(\frac{2\pi f \Delta t}{2}\right) \approx 1 - \frac{(2\pi f)^2(\Delta t)^2}{12} \,. \tag{3.1}$$

This frequency response resembles that of a Gaussian:

$$G(f) = e^{-\alpha f^2} \approx 1 - \alpha f^2 \,. \tag{3.2}$$

If the signals' spectra can be represented by Ricker wavelets,

$$S_n(f) = A_n \left(\frac{f}{f_n}\right)^2 e^{-\left(\frac{f}{f_n}\right)^2} \tag{3.3}$$

where, in image n, S_n is the frequency spectrum, f_n is the peak frequency, and A_n is

the amplitude, Gaussian smoothing can transform the signal to a different dominant frequency.

Because we are smoothing the high-resolution image to match it with the legacy image, we can relate the high-resolution frequencies, S_h, to the legacy frequencies, S_l, such that

$$S_l(f) = Ae^{-\alpha f^2} S_h(f) \tag{3.4}$$

where $A = A_l/A_h$,

$$\alpha = \frac{1}{f_l^2} - \frac{1}{f_h^2} , \tag{3.5}$$

and the subscripts l and h correspond to the legacy and high-resolution images, respectively.

Combining equations (3.1), (3.2), and (3.5) leads to the specification of the triangle smoothing radius as

$$\Delta t \approx \frac{1}{2\pi}\sqrt{12\left(\frac{1}{f_l^2} - \frac{1}{f_h^2}\right)} . \tag{3.6}$$

Here, Δt is the radius of smoothing, measured in seconds, applied to the high-resolution image to match the frequency content with the legacy image at each sample. The calculated smoothing radius for this data set is shown in Figure 3.3.

We measure local frequencies in both images and apply smoothing specified by equation (3.6) to the high-resolution image. Since this is only an approximation of what the smoothing radius should be under ideal conditions, we adjust the constant 12 in the equation to achieve a better match. In this example, this effectively reduces the difference between the spectral content of the images, as shown in Figure 3.2(b). Figure 3.4 shows the difference in local frequencies before and after smoothing. After smoothing, the frequency difference is minimized.

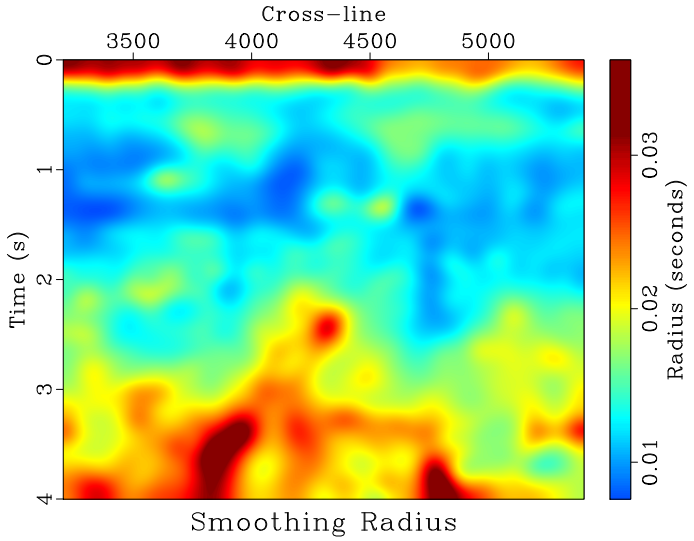


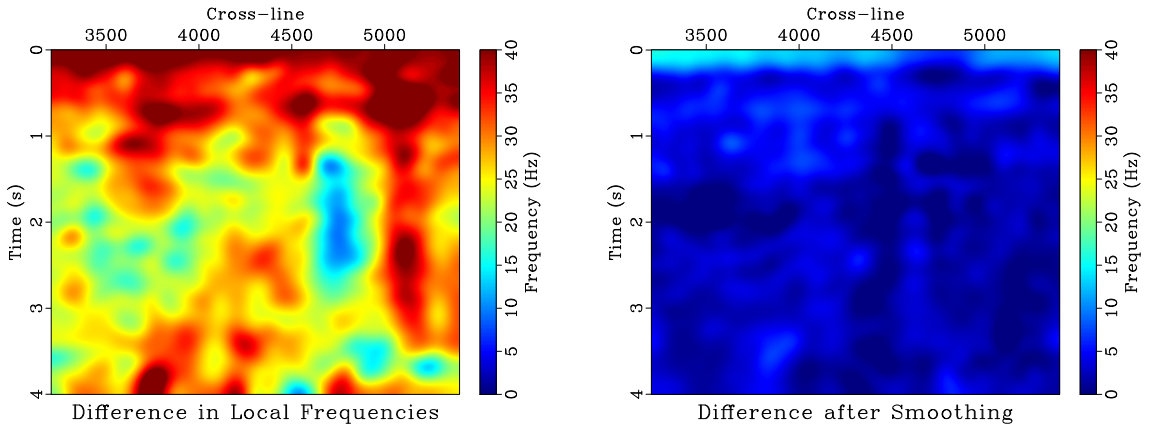
Figure 3.3: Calculated spatially and temporally variable smoothing radius. This represents the number of seconds in the temporal direction that the high-resolution image gets averaged over in a triangle weight to balance local frequency content with the legacy image at each sample. chapter-locfreq/../chapter-merge/apache rect

This method works well for simple data sets with overlapping frequency content. However, more complicated data sets, as seen in the next example, may require additional steps for successful frequency balancing.

Iterative method for calculating the smoothing radius

The theoretical smoothing radius works in some situations, like in the first example of the next chapter, but it doesn't work in situations where there is not much overlap in frequency content between data sets, like in the second example of the next chapter. In this case, we use an iterative method to find what the smoothing radius should be that is based on the physical understanding of what smoothing is doing.

The goal of this method is to find the temporally and spatially variable smooth-



(a) (b)

Figure 3.4: Difference in local frequencies between the legacy and high-resolution images before (a) and after (b) balancing their frequency content by non-stationary smoothing. chapter-locfreq/../chapter-merge/apache freqdif,freqdif-filt

ing radius, $\mathbf{R}$, that minimizes the difference in local frequencies between the two data sets. This can be shown in the objective function

$$\min_{\mathbf{R}\in[1,N]} \left\| \mathbf{F}[\mathbf{S_R}\mathbf{d}_h] - \mathbf{F}[\mathbf{d}_l] \right\| , \tag{3.7}$$

where $\mathbf{S_R}$ is the non-stationary smoothing operator of smoothing radius $\mathbf{R}$, $\mathbf{d}_h$ is the higher frequency data, $\mathbf{d}_l$ is the lower frequency data, $\mathbf{F}$ is the local frequency operator, and N is the maximum size of the smoothing radius. Although smoothing is a linear operation, the smoothed data, $\mathbf{S_R}\mathbf{d}_h$, depends non-linearly on R. However, the objective from equation (3.7) is nearly convex, and we choose to use an intuitive iterative approach to find an approximate smoothing radius.

The main premise behind the method comes from the fact that, in general, the greater the smoothing radius, the more high frequencies are attenuated by smoothing.

1. The smoothing radius is too *small* at a specified point if, after smoothing, the

higher frequency data has *higher* local frequency than the lower frequency data. Thus, the smoothing radius must be *increased* at that point.

2. The smoothing radius is too *large* at a specified point if, after smoothing, the higher frequency data has *lower* local frequency than the lower frequency data. Thus, the smoothing radius must be *decreased* at that point.

We apply these assumptions using a line-search method:

$$\mathbf{R}^{(i+1)} = \mathbf{R}^{(i)} + c\mathbf{r}^{(i)}, \tag{3.8}$$

where $\mathbf{R}^{(i)}$ is the smoothing radius at the ith iteration, c is a scalar constant that can be thought of as the step length, and $\mathbf{r}^{(i)}$ is the residual at the ith iteration, which can be thought of as the search direction, and is defined as

$$\mathbf{r} = \mathbf{F}[\mathbf{S}_{\mathbf{R}}\mathbf{d}_h] - \mathbf{F}[\mathbf{d}_l]\ . \tag{3.9}$$

It can be noted that when equation (3.9) is positive, the higher frequency data still has a higher local frequency value at that specific point than the lower frequency data, thus the higher frequency data is under-smoothed and the smoothing radius should be increased at that point. This follows the form of the first assumption. The second assumption is used when equation (3.9) is negative. When equation (3.9) is zero, the correct radius has been found and no further corrections are made. Thus, it is justifiable to set the search direction from equation (3.8) equal to the residual.

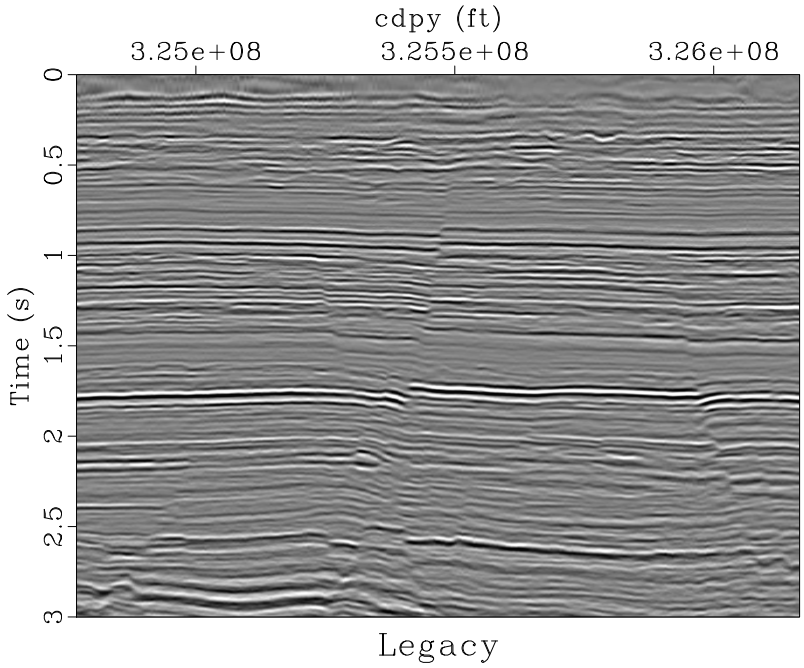
Using the assumptions, we can choose an intial guess for the smoothing radius and continually adjust the smoothing radius until we achieve the desired result of balancing the local frequency content between the two data sets. In practice, this method produces an acceptable solution in approximately 5 iterations and exhibits

sublinear convergence. After smoothing the higher frequency data with the estimated radius, we use the lower frequency and smoothed higher frequency data to estimate time shifts and align the two data sets.

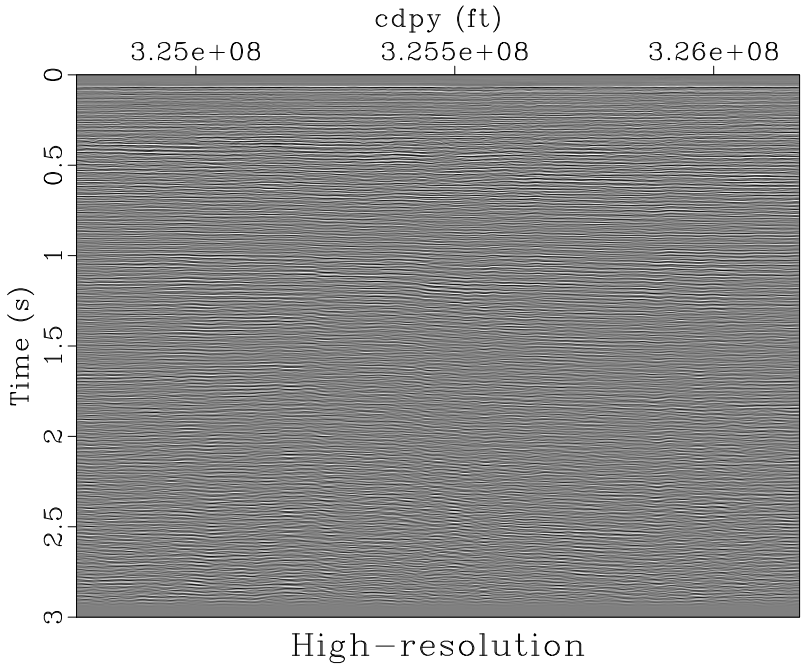
This method is applicable to workflows that require matching data with different frequency content. Here, we demonstrate that using this algorithm to match high-resolution and legacy seismic images produces better results.

High-resolution seismic data, such as those acquired with the P-cable acquisition system, can produce very detailed images of the near subsurface (Meckel and Mulcahy, 2016). When compared to conventional seismic images, high-resolution images have a higher dominant frequency and a larger frequency bandwidth. However, they usually lack low frequency content and depth coverage that is present in conventional seismic images. As a result, successful interpretation of high-resolution images can be aided by matching with legacy data coverage over the same area.

Example legacy and high-resolution images of the same subsurface are shown in Figure 3.5. The first step in matching the two images is to ensure that they both have a similar frequency bandwidth so they are directly comparable. The average frequency spectra for the two images are shown in Figure 3.7(a). From this, it is evident that there is almost no overlap in frequency bandwidth between the two images. To address this problem, we apply a low-cut filter to the legacy image to remove some of the lower frequencies that are simply not present in the high-resolution image. Next, we implement the method described in the previous section to balance local frequency content between the two images. The difference in local frequencies (residual, by equation 3.9) between the high-resolution and legacy images before balancing frequency content and after 5 iterations of the algorithm in equation (3.8)



(a)



(b)

Figure 3.5: Initial legacy (a) and high-resolution (b) images. These images show the same subsurface geology, but look remarkably different as the high-resolution image has distinctly higher frequency content than the legacy image. chapter-locfreq/merge legacy,hires-agc

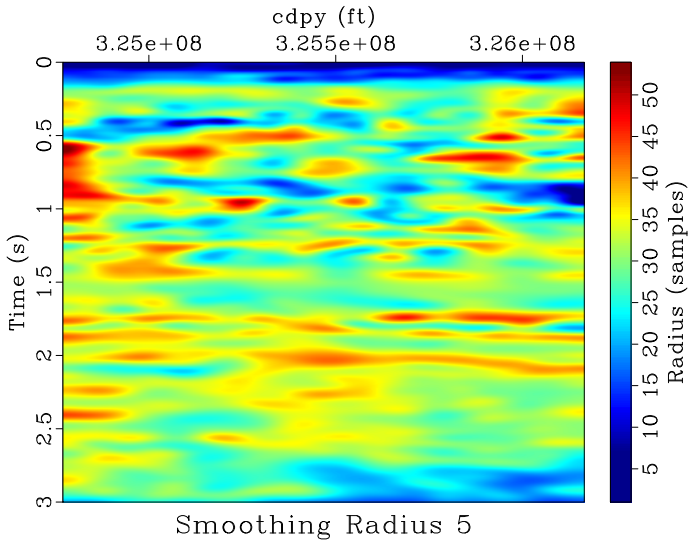
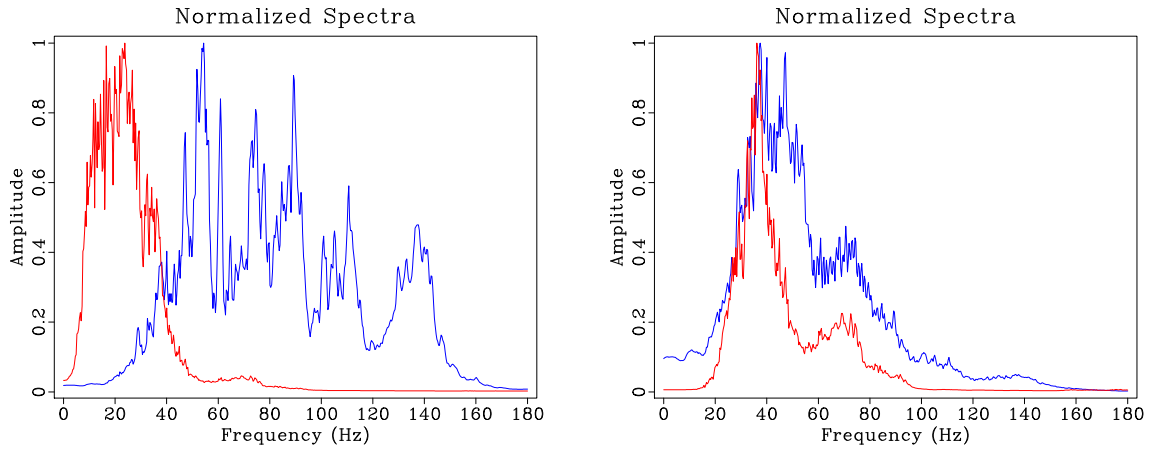


Figure 3.6: The smoothing radius, which is a function of time and space, this method produces after 5 iterations. This represents the number of samples in time that the high-resolution image needs to be smoothed over in a triangle weight to balance local frequency content with the legacy image. chapter-locfreq/merge rect5

is shown in Figure 3.8. After balancing local frequencies, the images show a similar spectral bandwidth (Figure 3.7(b)), which helps increase the correlation between the two images and makes matching reflections better defined.

After the frequency content is matched, the optimal time shift is found to align signal content between the legacy and high-resolution images. We then apply this time shift to the original high-resolution image—the frequency content is only degraded for the purpose of finding the time shift.

An application of aligning the high-resolution and legacy images is discussed in the next chapter.



(a) (b)

Figure 3.7: Spectral content of the legacy (red) and high-resolution (blue) images before (a) and after (b) spectral balancing using the theoretical method. chapter-locfreq/merge nspectra-orig,high-smooth-spec5

Iterative method for calculating the smoothing radius: A modification

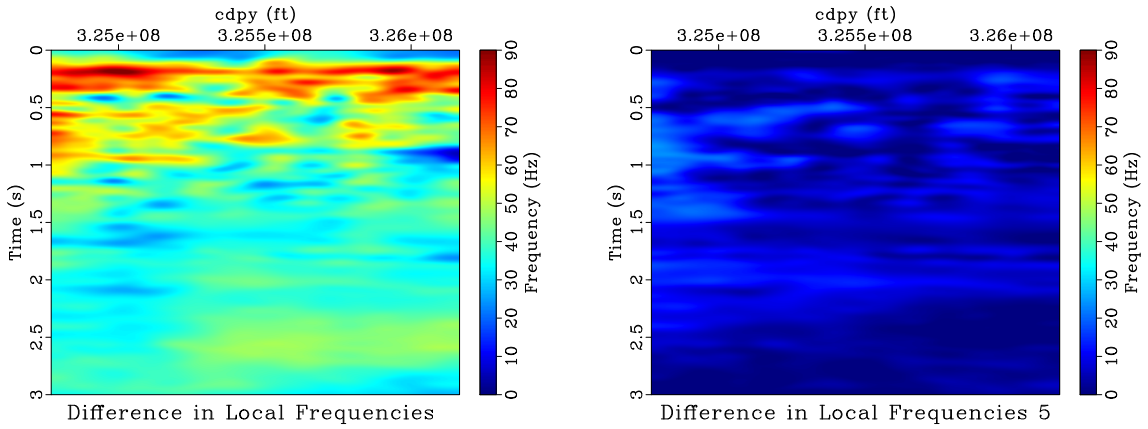
The algorithm previously presented works well in cases when one data set has clearly higher frequency content than the other. However, in some cases, the two data sets may have similar local frequency content, but they may still need to be matched. We illustrate a modified version of the previous algorithm by demonstrating it on an example of multi-component seismic image registration.

Multicomponent seismic image registration is an important step before the interpretation of P and S images of the subsurface. It involves warping the space of S images to align reflections with the analogous reflections of P images (Fomel and Backus, 2003b; Fomel et al., 2005).

Figure 3.9 shows PP and SS images from a 9-component seismic survey (Fomel, 2007a). To properly register the images, we follow the method proposed in Fomel

et al. (2005). It consists of three primary steps: (1) initial registration of PP and SS images using initial interpretation and well-log analysis; (2) balancing frequency and amplitude content; and (3) final registration using residual local similarity scanning. We incorporate our method of balancing frequency content into the second step in this process.

Before initial registration, the PP image has much higher frequency content than the SS image. After the SS image is temporally compressed to PP time for initial registration, the two images have more similar frequency content. However, additional frequency balancing is still needed before residual registration. This poses a problem as neither image has distinctly higher frequencies than the other, so both images need to be smoothed in different areas to balance frequency content. In order to do this, we modify the proposed method to include two separate smoothing operators—one for each image.



(a) (b)

Figure 3.8: Difference in local frequencies (residual) between the legacy and high-resolution images before (a) and after (b) the 5th iteration of frequency balancing. chapter-locfreq/merge freqdif,freqdif-filt5

We modify the objective in equation (3.7) as

$$\min_{\mathbf{R}\in[-N,-1]\cup[1,N]} \left\| \mathbf{F}[\mathbf{S}_{\mathbf{R}_p}\mathbf{d}_p] - \mathbf{F}[\mathbf{S}_{\mathbf{R}_s}\mathbf{d}_s] \right\| , \tag{3.10}$$

where $\mathbf{d}_p$ and $\mathbf{d}_s$ are the PP and SS images, respectively, and $\mathbf{S}_{\mathbf{R}_p}$ and $\mathbf{S}_{\mathbf{R}_s}$ are the non-stationary smoothing operators for the PP and SS images, respectively. We also modify the residual from equation (3.9) as

$$\mathbf{r} = \mathbf{F}[\mathbf{S}_{\mathbf{R}_p}\mathbf{d}_p] - \mathbf{F}[\mathbf{S}_{\mathbf{R}_s}\mathbf{d}_s] \, . \tag{3.11}$$

The ideal radius is still found using the same line-search from equation (3.8), except we allow the smoothing radius to be negative. Physically, a negative smoothing radius would signify that the image should be sharpened at that particular point instead of smoothed. In this case, instead of trying to sharpen the PP image at that particular point, we choose to smooth the SS image by the negative part of the smoothing radius. Thus, we define the ith components of $\mathbf{R}_p$ and $\mathbf{R}_s$ as

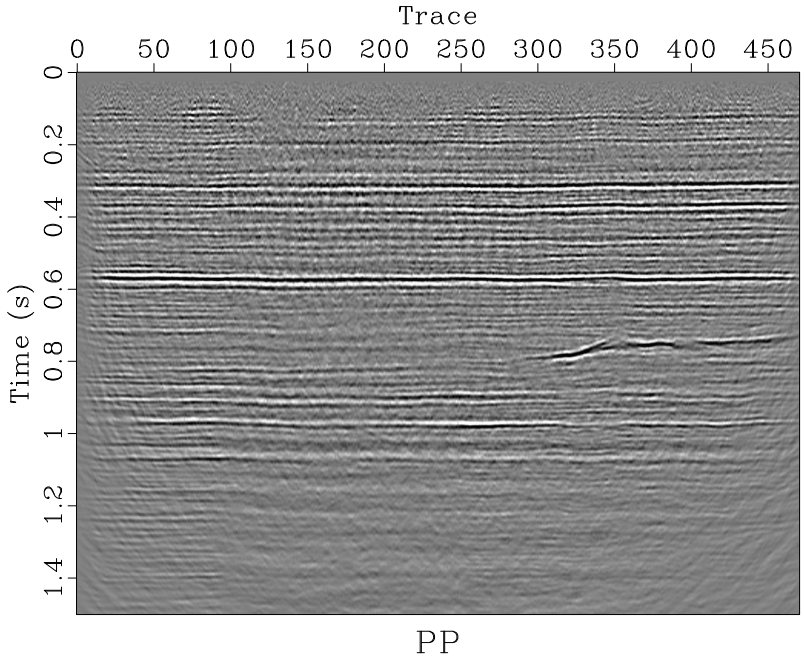
$$R_{p,i} = \begin{cases} R_i & \text{if } R_i \geq 1 \\ 1 & \text{otherwise} \end{cases} \tag{3.12}$$

and

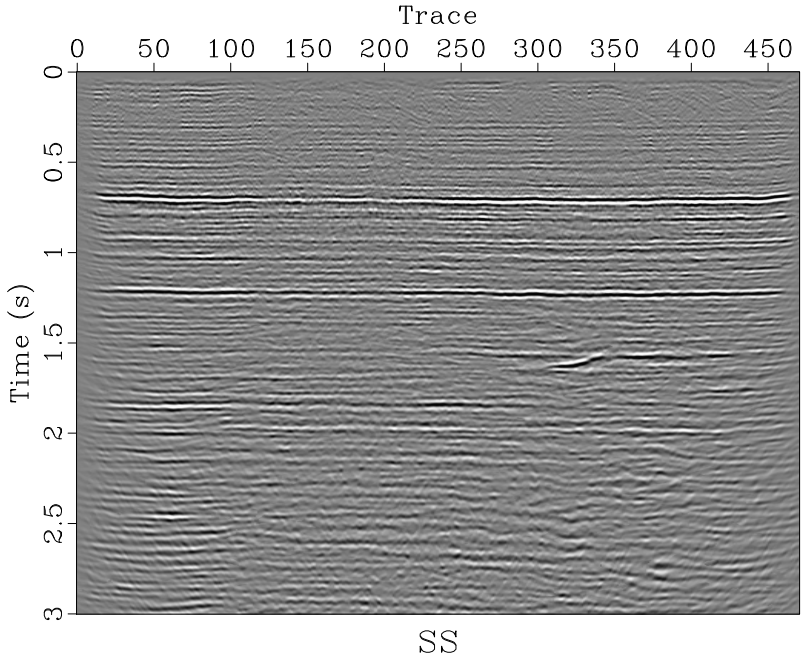
$$R_{s,i} = \begin{cases} |R_i| & \text{if } R_i \leq -1 \\ 1 & \text{otherwise} \end{cases} \tag{3.13}$$

where R_i is the ith component of $\mathbf{R}$ and a radius of 1 represents no smoothing. This allows each image to be smoothed in different areas to balance the frequency content between the two images.

The results of using this spectral balancing method are shown in Figure 3.10. Comparable results were achieved in Liu and Fomel (2012), who used local time-frequency decomposition (LTFD) to balance the spectral content between the two



(a)



(b)

Figure 3.9: Initial PP (a) and SS (b) images. chapter-locfreq/vecta pp,ss

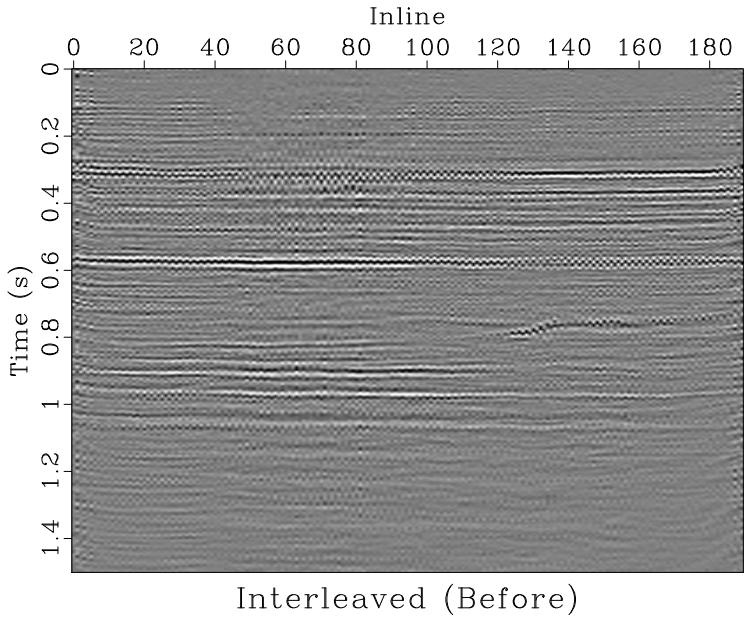
images. However, the method we propose in this chapter is more straightforward and significantly less computationally expensive.

So far, this algorithm and its modification have been discussed when the smoothing radius is only calculated along one dimension. However, Chapter 5 provides an application where the smoothing radius is calculated in multiple directions.

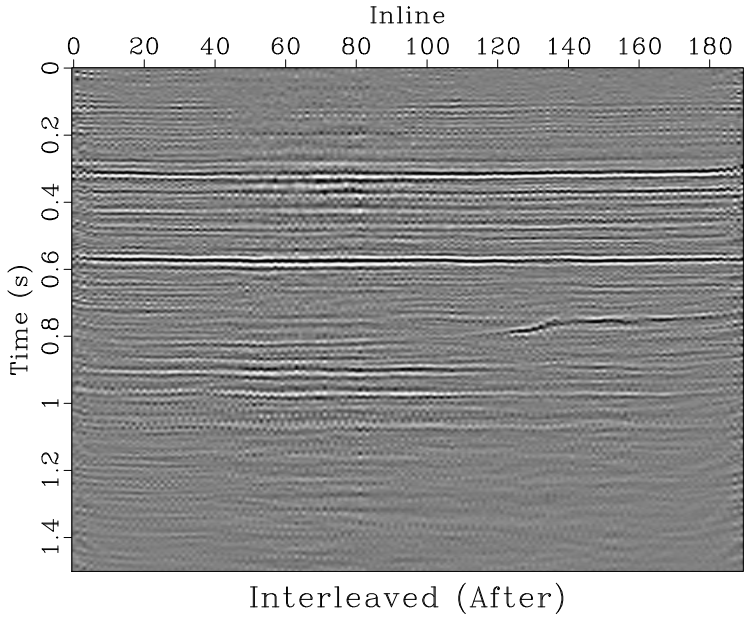
CONVERGENCE ANALYSIS

Since this algorithm was developed from observations and intuition based on what smoothing is physically doing, no error analysis or convergence criteria has been developed in this thesis. However, it is observed that this algorithm generally exhibits sublinear convergence. Additionally, the initial guess for what the smoothing radius should be negligibly impacts the final result. Figure 3.11(a) shows the convergence of the algorithm from equation (3.8) when applied to the *P-cable* data set from the second example in this chapter (Figure 3.5). It is evident that no matter the initial guess, the resulting convergence is relatively unaffected, and with the correct choice of step length c, converges in few iterations. After convergence, continuing iterations do not affect the final result.

Figure 3.11(b) shows the convergence when choosing different values of c, the step length from Equation (3.8). From this, it is evident that the step length is important for fast and stable convergence. If too large a step length is chosen, the method does not converge to the ideal solution. If the step length is too small, the algorithm takes many more iterations to converge. With the "correct" step length, however, the algorithm usually converges in under 10 iterations in practice.

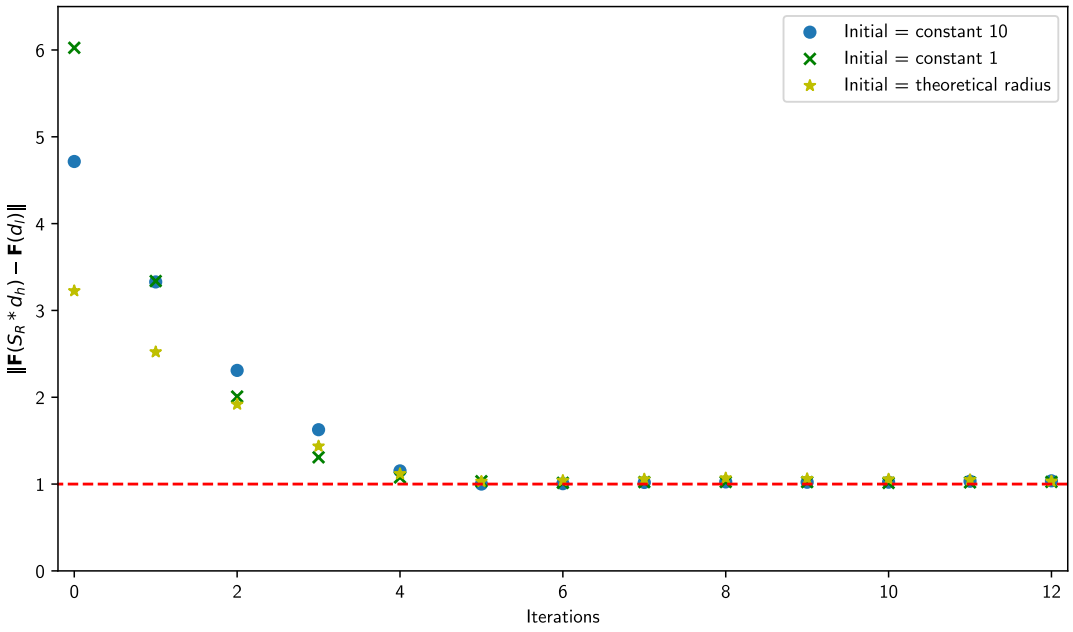


(a)

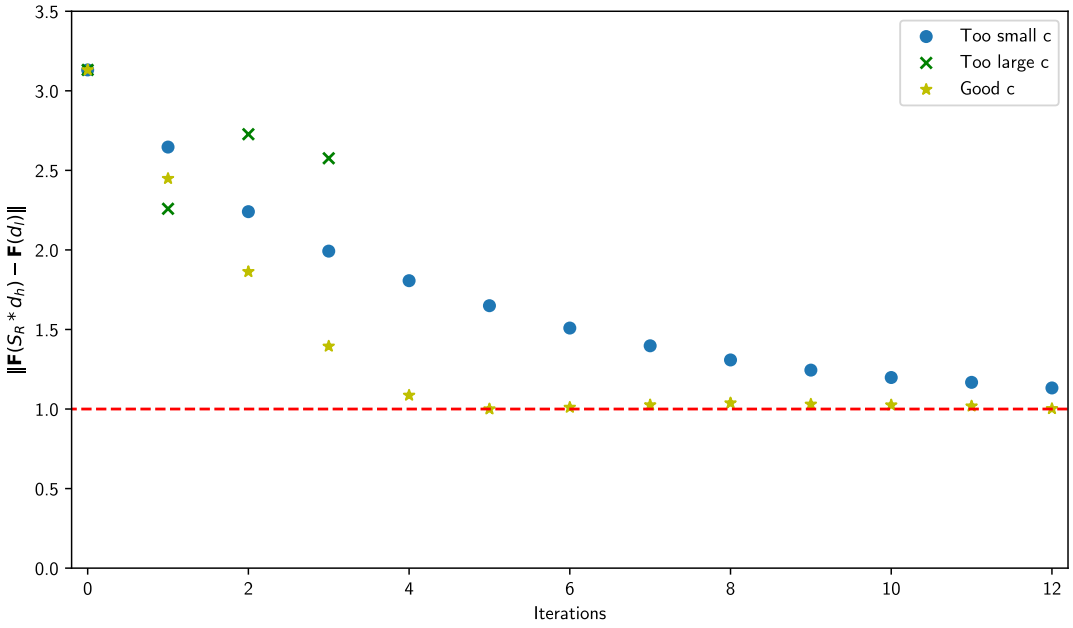


(b)

Figure 3.10: Interleaved PP and warped SS traces before (a) and after (b) frequency balancing and residual registration. After registration, the signal content between the two initial images is more aligned; for example, the reflections around 0.3 and 0.6 s. This indicates a successful registration. chapter-locfreq/vecta before,after



(a)



(b)

Figure 3.11: Convergence of the algorithm from Equation (3.8) for (a) different initial guesses, and (b) different choices of c, the step length, using the theoretical radius as the initial guess. chapter-locfreq/convergence all,scalar

CONCLUSIONS

It is difficult to compare signals with differing frequency content because signals with differing frequencies are hard to correlate. In this chapter, we proposed two methods of balancing frequency content between data. The first one takes a theoretical approach of what we expect the data to be, and the second one takes the form of an optimization problem solved by a simple iterative algorithm. This algorithm is a relatively inexpensive and effective method compared to previously proposed methods of balancing frequencies between data sets. We applied these methods to examples of matching seismic images of differing resolution and for multicomponent image registration.

ACKNOWLEDGEMENTS

We thank the sponsors of the Texas Consortium for Computational Seismology (TCCS) for their financial support.

Chapter 4

Matching and merging high-resolution and legacy seismic images

When multiple seismic surveys are acquired over the same area using different technologies that produce data with different frequency content, it may be beneficial to combine these data to produce a broader bandwidth volume. We have developed a workflow for matching and blending seismic images obtained from shallow high-resolution seismic surveys and conventional surveys conducted over the same area. The workflow consists of three distinct steps: (1) balancing the amplitudes and frequency content of the two images by non-stationary smoothing of the high-resolution image, (2) estimating and removing variable time shifts between the two images, and (3) blending the two images together by least-squares inversion. Our workflow is applied successfully to images from the Gulf of Mexico.

INTRODUCTION

Modern high-resolution seismic acquisition systems, such as P-cable (Petersen et al., 2010; Meckel and Mulcahy, 2016), can produce detailed images of the subsurface at shallow depths. These images often need to be compared with those previously produced using legacy images from conventional seismic acquisition. In comparison with

Parts of this chapter were first published in Greer and Fomel (2017b). The peer-reviewed journal version appears as Greer and Fomel (2018). This work was done under the supervision of Dr. Sergey Fomel.

high-resolution images, conventional images have generally lower frequency content and correspondingly lower resolution, but better signal content at greater depth. To reconcile the differences between the two types of images, they need to be properly matched.

Analogous problems occur when interpreting images from multicomponent seismic acquisition. In particular, single-component PP and converted PS images often exhibit significantly different frequency content and need to be balanced for accurate registration (Hardage et al., 2011; Fomel and Backus, 2003a; Fomel et al., 2005).

When multiple data sets are acquired over the same area using different technologies, it is likely that these data sets contain signal content from different frequency bands, which correspondingly contain unique information about the subsurface. In order to utilize all available information, it may be beneficial to produce a consolidated broad-bandwidth volume that combines these data sets. Many previous methods of seismic data merging have been developed for the purpose of increasing the spatial coverage of the merged volume by combining data from partially overlapping areas. This usually involves steps such as rebinning for data alignment, along with spectral and phase matching. As a result, both the initial and resultant merged images poses similar signal characteristics, most notably spectral content (Mohan et al., 2007; Zhou et al., 2014; Al-Inaizi et al., 2004). However, previously Carter and Pambayuning (2009) are successful in applying a frequency domain merge to two separate seismic volumes over the same area to broaden the effective bandwidth.

In this chapter, we consider the problem of matching seismic images obtained in the same area with different resolution. Using techniques borrowed from multi-

component image processing (Fomel et al., 2005), we propose a multistep approach. First, we balance the two images in amplitude and frequency content. As a result, the resolution of the high-resolution image is temporarily degraded to match the resolution of the legacy data. Next, we measure shifts between images using local similarity scanning (Fomel, 2007a; Fomel and Jin, 2009). Finally, when the images are aligned and matched, we create a blended image using least-squares inversion in the time domain. The blended image has a wider frequency bandwidth than the two initial images, along with coherent signal content with depth from the legacy image and detailed shallow coverage from the high-resolution image.

We test the proposed approach using data from the Gulf of Mexico. A 2D image is used to illustrate the method, followed by an example applied to 3D data.

METHOD

The initial legacy and high-resolution example images are shown in Figure 4.1. They both underwent standard marine processing flows, and are assumed to be optimally processed. The images show similar structures, particularly at shallow depths, but with strikingly different resolution. The main difference comes from the broader frequency bandwidth of the high-resolution image in comparison with that of the legacy image. Therefore, our first step in comparing the two images is balancing their spectral content.

Balancing spectral content

Analyzing the spectra of the legacy and high-resolution images, as seen in Figure 3.2(a), it is clear that the high-resolution image has a wider range of frequencies

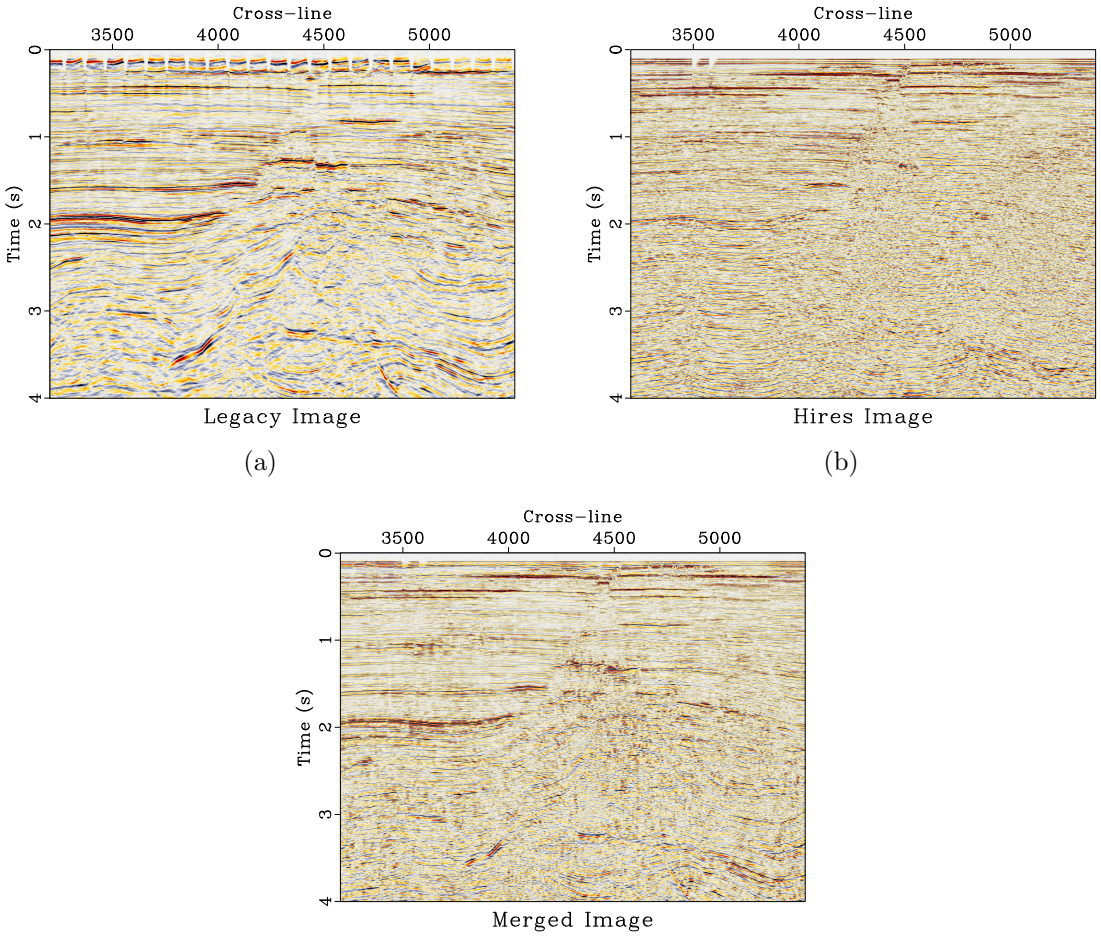


Figure 4.1: The initial legacy (a) and high-resolution (b) images. The merged image (c) is the final product of the proposed workflow: the combination of both the legacy and high-resolution images. chapter-merge/apache legacy,hires,merge2-reverse

with a higher dominant frequency than the legacy image. In order to match these images, our first step is to balance their spectral content. We can achieve this by attenuating the high frequencies of the high-resolution image to match the lower frequency content of the legacy image. One approach to frequency balancing is to apply a stationary bandpass filter to the high-resolution image. However, this does not take into account local frequency variations in each image caused by seismic wave attenuation. A more effective method, which accounts for temporally and spatially variable frequency content that is present in most seismic data, is to apply a non-stationary filter using frequency information from both images. To accomplish this, we use a simple triangle smoothing operator with an adjustable radius. Here, the radius at each point is the amount of time, in seconds, that that specific data point gets averaged over in a triangle weight in the temporal direction. In this example, we find the radius of smoothing using the theoretical approach described in the previous chapter, so

$$\Delta t \approx \frac{1}{2\pi}\sqrt{12\left(\frac{1}{f_l^2}-\frac{1}{f_h^2}\right)}\,. \tag{4.1}$$

Here, f_l and f_h are the local frequencies of the legacy and high-resolution images, respectively, and Δt is the radius of smoothing, measured in seconds, applied to the high-resolution image to match the frequency content with the legacy image at each sample. The spectral content of the high resolution image after balancing local frequencies clearly closer resembles that of the legacy image, as shown in Figure 3.2(b).

This method works well for simple data sets with overlapping frequency content. However, more complicated data sets, as seen in the next example, may require additional steps for successful frequency balancing.

In addition to frequency balancing, we initially attempted accounting for wavelet phase differences between data sets at this step. However, we saw this had a negligible impact on the final result, so we decided not to include it in this method.

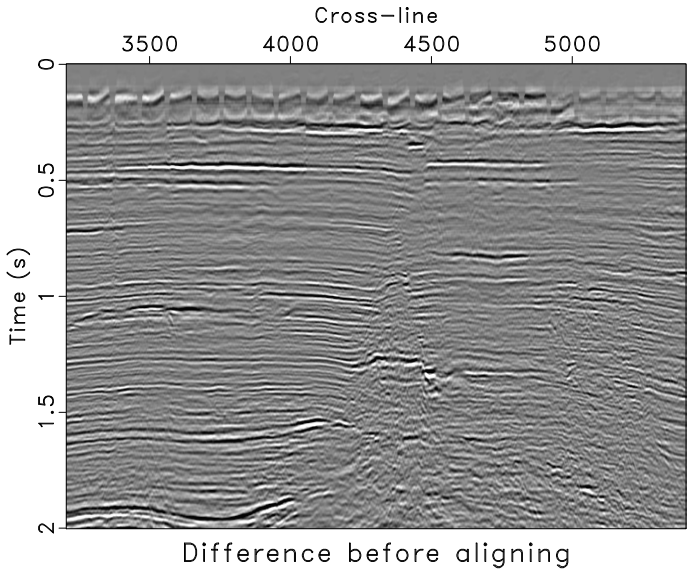
Measuring time shifts

After balancing the spectral content, we attempt to account for possible time shifts of the high-resolution image relative to the legacy image, which can be caused by changes in acquisition and processing parameters. We measure this shift using local similarity scanning (Fomel, 2007a; Fomel and Jin, 2009). In this method, we detect the relative time shift by first calculating the local similarity at different time shifts of the high-resolution image relative to the legacy image (Fomel and Jin, 2009). From this, the trend of the highest similarity is picked and represents the relative time shift between the two images.

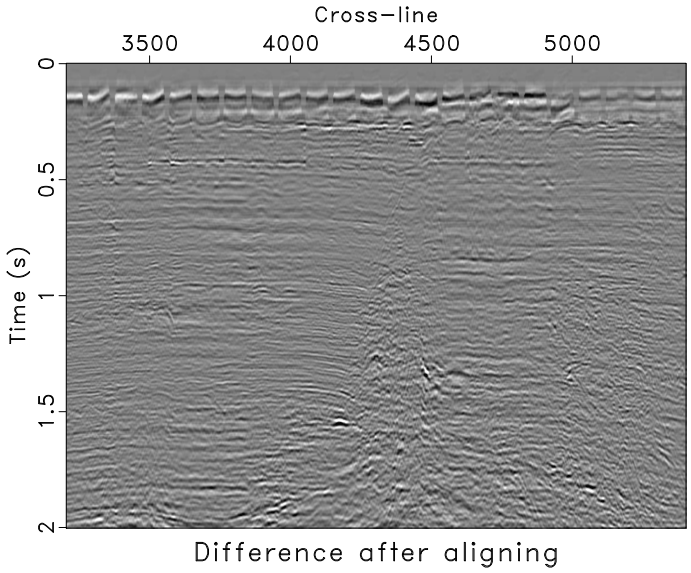
Next, we apply the estimated time shift to the original high-resolution image—the frequency content is only degraded for the purpose of finding the time shift that best aligns the signal content between the two images. The differences between the two images before and after the time shift correction are shown in Figure 4.2, and demonstrate a noticeably better match resulting from the time shift. Similar results were achieved using amplitude-adjusted plane-wave destruction, which involves balancing amplitudes and temporally aligning the data sets simultaneously (Phillips and Fomel, 2016).

Creating the blended image

Since the high-resolution and legacy images contain information about the same subsurface, we can attempt to create an optimal image of this area by blending



(a)



(b)

Figure 4.2: Difference between the legacy and smoothed high-resolution images before (a) and after (b) aligning the data sets. Before accounting for time shifts, much of the signal content did not align in time, so coherent reflections were subtracted out. After accounting for time shifts, the reflections are more aligned, so much of the subtracted information is noise. chapter-merge/apache diff0,diff1

the two images together to combine the strengths of each while minimizing their weaknesses. We can achieve this by imposing two conditions. First, the blended image should match the high-resolution image, particularly in the shallow part. Second, after smoothing with the non-stationary smoothing operator, the blended image should match the legacy image. We combine the two conditions together in the least-squares system

$$\begin{bmatrix} \mathbf{W_h} \\ \mathbf{W_l S} \end{bmatrix} \mathbf{b} \approx \begin{bmatrix} \mathbf{W_h\, h} \\ \mathbf{l} \end{bmatrix} , \tag{4.2}$$

where $\mathbf{h}$ denotes the high-resolution image, $\mathbf{l}$ is the legacy image, $\mathbf{b}$ is the desired blended image, $\mathbf{W_h}$ and $\mathbf{W_l}$ are the diagonal weighting matrices for the high-resolution and legacy images, respectively, and $\mathbf{S}$ is the non-stationary smoothing specified by equation (4.1). The formal solution of the least-squares problem (4.2) is

$$\mathbf{b} \;=\; \left(\mathbf{W_h}^2 + \mathbf{S}^T\mathbf{W_l}^2\mathbf{S}\right)^{-1} \left(\mathbf{W_h}^2\,\mathbf{h} + \mathbf{S}^T\,\mathbf{W_l}\mathbf{l}\right) . \tag{4.3}$$

Alternatively, equation (4.3) can be rearranged as

$$\begin{aligned} \mathbf{b} \;&=\; \left(\mathbf{W_h}^2 + \mathbf{S}^T\mathbf{W_l}^2\mathbf{S}\right)^{-1} \left(\mathbf{W_h}^2\,\mathbf{h} + \mathbf{S}^T\mathbf{W_l}^2\mathbf{S}\mathbf{h} - \mathbf{S}^T\mathbf{W_l}^2\mathbf{S}\mathbf{h} + \mathbf{S}^T\,\mathbf{W_l}\mathbf{l}\right) \\ &=\; \left(\mathbf{W_h}^2 + \mathbf{S}^T\mathbf{W_l}^2\mathbf{S}\right)^{-1} \left(\left(\mathbf{W_h}^2 + \mathbf{S}^T\mathbf{W_l}^2\mathbf{S}\right)\mathbf{h} + \mathbf{S}^T\mathbf{W_l}\left(\mathbf{l} - \mathbf{W_l}\mathbf{S}\mathbf{h}\right)\right) \\ &=\; \mathbf{h} + \left(\mathbf{W_h}^2 + \mathbf{S}^T\mathbf{W_l}^2\mathbf{S}\right)^{-1} \mathbf{S}^T\,\mathbf{W_l}\left(\mathbf{l} - \mathbf{W_l}\mathbf{S}\mathbf{h}\right) , \end{aligned} \tag{4.4}$$

which makes it evident that the merged image is simply the high-resolution image with some changes.

The weights, $\mathbf{W_h}$ and $\mathbf{W_l}$, are applied to the images to bring out the desired qualities from each image. For example, since we prefer the high-resolution image in the shallow section, we weight it stronger there and gradually taper the weight to preference the legacy image with depth. In addition, we estimate the legacy weight,

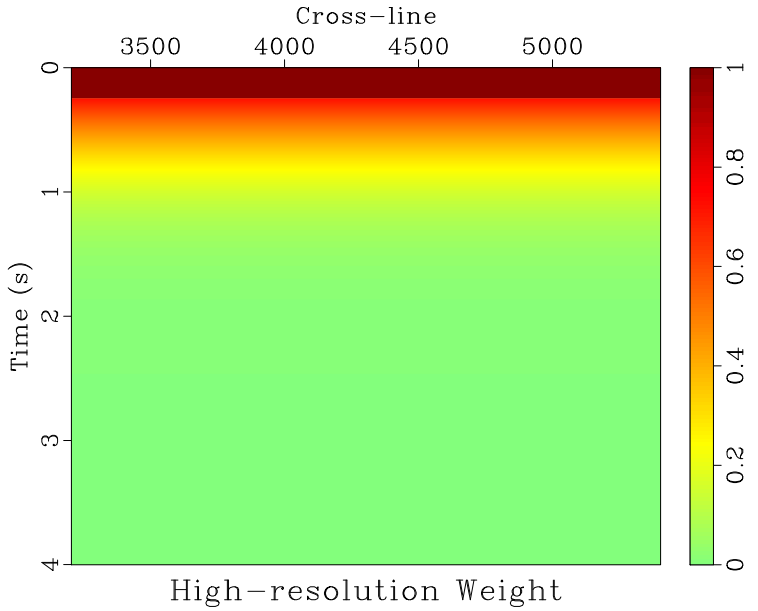
$\mathbf{W_l}$, to balance the legacy image's amplitudes with respect to the high-resolution image. The specific values we used for the weights are shown in Figure 4.3.

We implement the inversion in equation (4.4) iteratively using the method of conjugate gradients (Hestenes and Stiefel, 1952). The resultant blended image, shown in Figure 4.1(c), retains the higher frequencies from the high-resolution image while incorporating the lower frequencies from the legacy image (Figure 4.4). The broader frequency bandwidth corresponds to an increase in resolution and leads to a more detailed and interpretable image. As a result, the blended image resembles the high-resolution image but has a marked decrease in noise and extended coverage with depth.

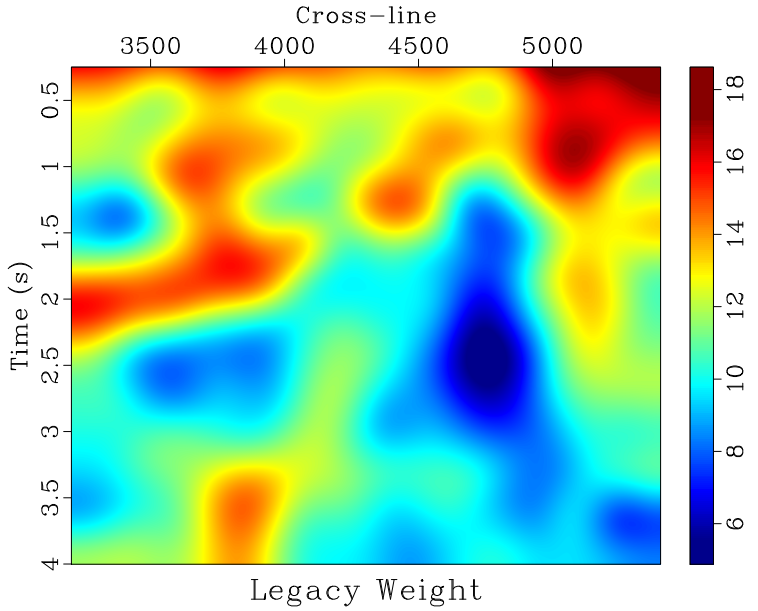
Although the method presented in this chapter is applied to post-stack images, the general method is likely flexible enough to be extended to pre-stack data. Applications of matching legacy and high-resolution seismic data are also seen in time-lapse image registration, where the accurate interpretation of 4D time-lapse data heavily depends on dataset alignment and uniform processing (Ross and Altan, 1997). The method from this chapter could also be used in 4D time-lapse processing; particularly the steps involving frequency balancing and accounting for time shifts.

P-CABLE EXAMPLE

Our second example refers to data from the inner shelf of the Gulf of Mexico, just off of San Luis Pass, Texas (Meckel et al., 2017). The high-resolution P-cable data set was acquired from a shallow marine environment in the Gulf of Mexico. The area of interest for this survey was the near subsurface, and a high frequency source was used which allows for exceptional resolution in the shallow section, at the expense of



(a)



(b)

Figure 4.3: The weights for the high-resolution (a) and legacy (b) images for the least-squares merge. The high-resolution weight is strongly weighted in the shallow part and blends to favor the legacy image with depth. The legacy weight is selected to boost the legacy image's relative amplitudes to match that of the high-resolution image. chapter-merge/apache hweight,lweight-reverse

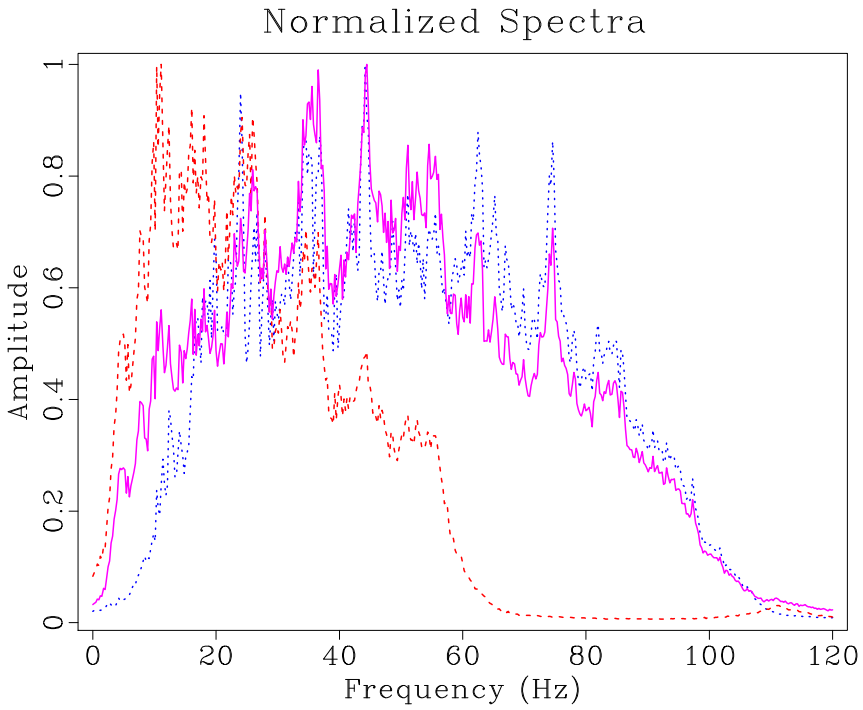
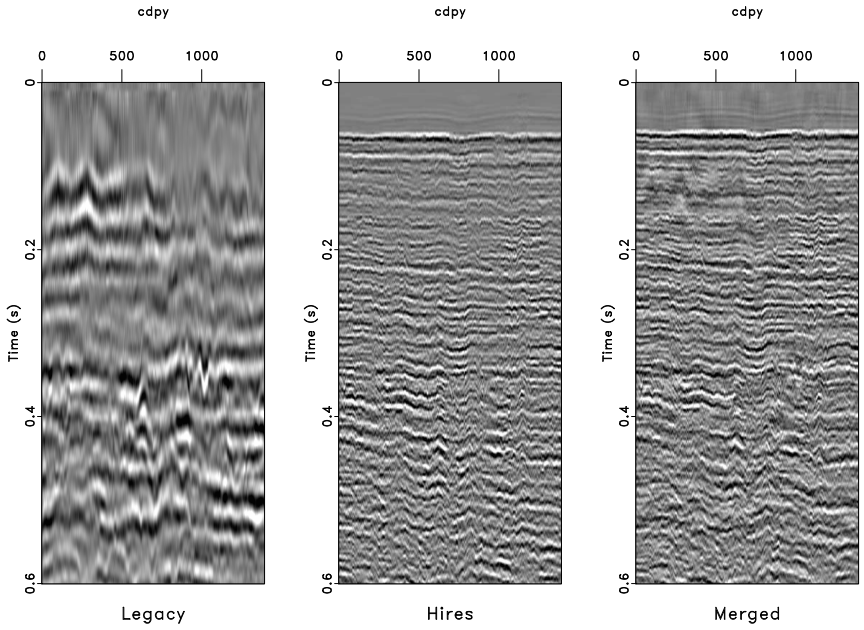


Figure 4.4: The spectra of the entire image display of the legacy (red dashed), high-resolution (blue dotted), and merged (magenta solid) images for the first data set. chapter-merge/apache nspectra22-reverse

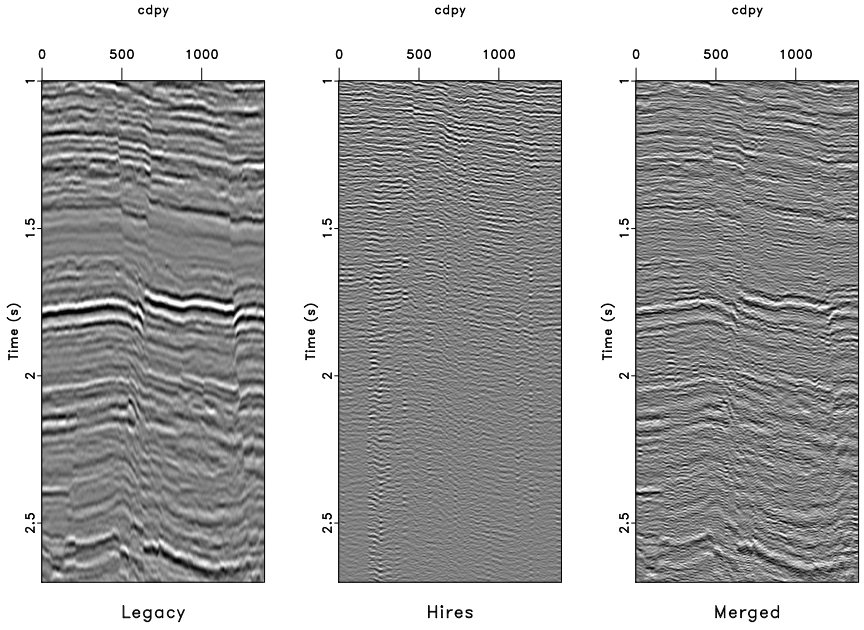
less coherent signal information at depth caused by attenuation (Meckel and Mulcahy, 2016). In addition, very little low-frequency information is present in the P-cable data because of the high-frequency source used, which makes balancing spectral content particularly difficult. The other image comes from legacy data coverage over the same area, which has better signal continuity at depth than the high-resolution P-cable data. This is apparent by looking at the first few hundred milliseconds of data for both data sets (Figure 4.5).

Due to the nature of acquisition, the high resolution image has very dense spatial coverage, providing detailed time slices of the near subsurface (Meckel and Mulcahy, 2016). The legacy image has lower spatial resolution. As a result, when matching the high-resolution and legacy images spatially, the high spatial resolution of the high-resolution image must be degraded to match that of the legacy image. We rebinned the legacy and high-resolution images to align them spatially for comparison (Figure 4.7). We chose to spatially down-sample the high-resolution image to match the legacy image as opposed to interpolating the legacy image to the high-resolution image's spatial grid to prevent potentially introducing inaccurate data in the merged image.

There is a definite separation in frequency content when comparing the legacy and high-resolution images (Figure 4.6). Because there is not much overlap in frequency bandwidth, balancing their spectral content is challenging. In addition, a primary assumption made in deriving the theoretical smoothing radius (equation 4.1) is that the signal is modeled by a summation of Ricker wavelets, which may not be a correct assumption. As a result, additional steps must be taken beyond applying the smoothing specified by equation (4.1) to ensure matching frequency content.



(a)



(b)

Figure 4.5: The first 600 ms of data from a sample line from the legacy, high-resolution, and merged image (a). The same images with depth for the legacy, high-resolution, and merged images (b). The merged image resembles the high-resolution image in the shallow parts and incorporates the more coherent lower frequency information from the legacy image with depth. chapter-merge/pcable window1,window2

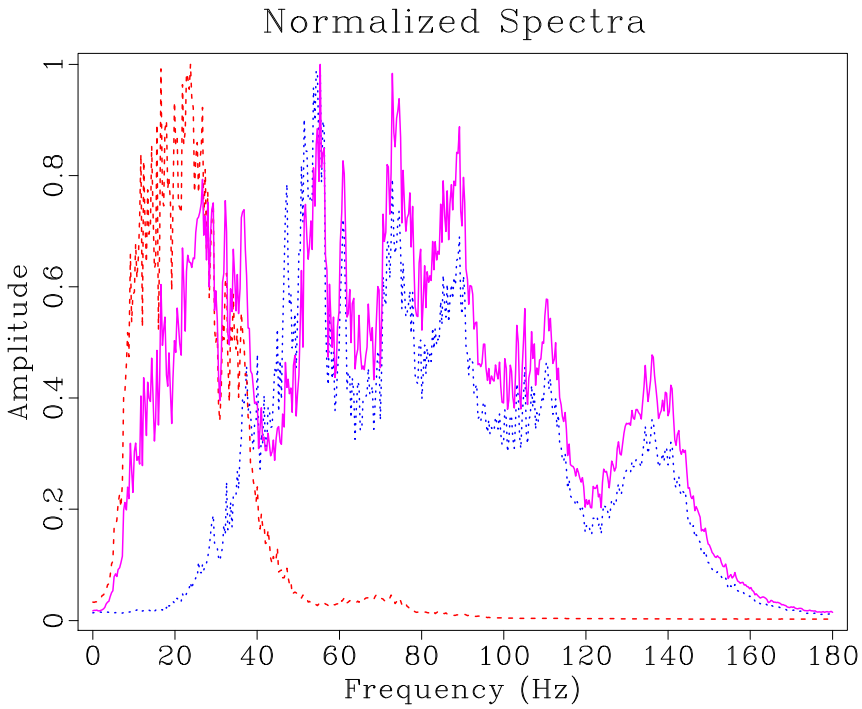


Figure 4.6: The spectral content of the entire image display of the legacy (red dashed), high-resolution (blue dotted), and resultant merged (magenta solid) images for the second data set. chapter-merge/pcable nspectra2

We first apply a low-cut filter to the legacy data to remove the low frequency information that is simply not present in the high-resolution image. Next, we adjust the non-stationary smoothing radius using the simple iterative algorithm (Greer and Fomel, 2017a) that was described in the previous chapter.

After this, we use the low-cut filtered legacy and smoothed high-resolution images to find estimated time shifts we need to apply to the high-resolution image to align the reflections with the legacy image. Then, we apply this estimated time shift to the original high-resolution image and blend it with the original legacy image as specified by equations (4.2) and (4.4).

The resultant merged image is shown in Figure 4.7(c). The frequency content of the merged image is shown in Figure 4.6. Here, the merged image spans the frequency bandwidth of the two initial images, thus producing a high resolution volume including optimal signal characteristics from the two initial images.

CONCLUSIONS

We propose an approach to matching seismic images of different resolutions. Our first step is non-stationary smoothing of the high-resolution image to match the spectral content and amplitudes of the legacy image. Next, we estimate the relative time shifts using local similarity scanning. After matching the two images, we create a blended image by least-squares inversion, which effectively combines the best features of the two images: the broader frequency bandwidth of the high-resolution image with the reflection continuity and deeper coverage of the legacy image. The final result is an interpretable blended image that has higher temporal resolution than either of the two initial images. Two example applications using high-resolution and legacy seismic

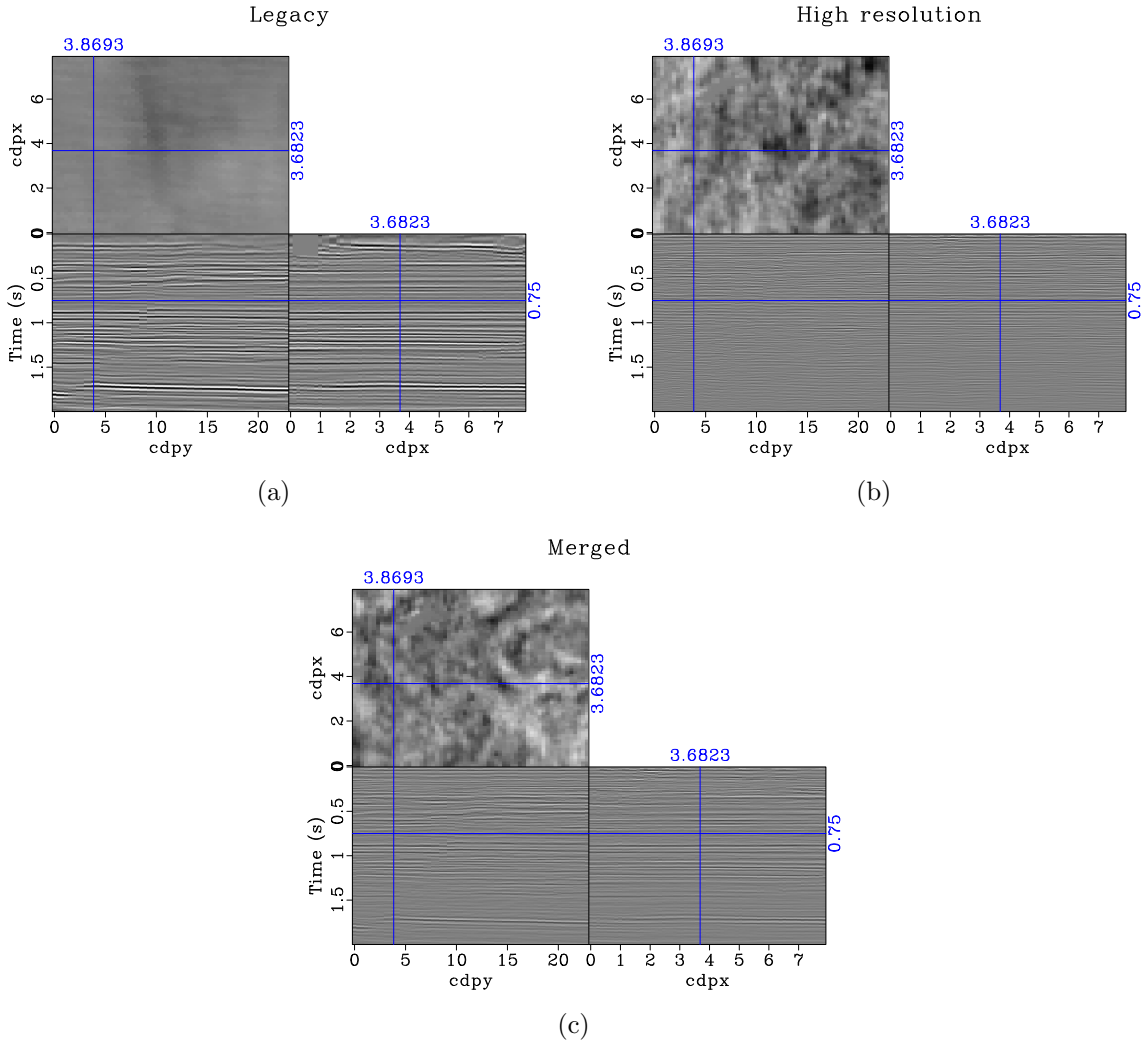


Figure 4.7: The legacy (a), high-resolution (b), and resultant merged (c) images of the second data set. chapter-merge/pcable2 legacy4,hires4,merge3

images from the Gulf of Mexico demonstrate the effectiveness of our method.

ACKNOWLEDGMENTS

We thank Michael DeAngelo, August Lau, Tip Meckel, and Chuan Yin for useful discussions and Apache and Fieldwood for providing the data used in the first example. We also thank the sponsors of the Texas Consortium for Computational Seismology (TCCS) for their financial support.

Chapter 5

Improving migration resolution by approximating the least-squares Hessian using non-stationary amplitude and frequency matching

We propose using two non-stationary operators to represent the amplitude and frequency variations in the least-squares Hessian to account for the principal differences between conventional and least-squares migrated images. The calculation and application of these operators are computationally inexpensive when compared to one iteration of least-squares migration, and it increases the resolution and amplitude fidelity of the migrated image. Successful results are achieved on an application of reverse-time migration to the Sigsbee synthetic data set.

INTRODUCTION

Least-squares migration can produce an accurate migrated seismic image. However, the process may be computationally expensive as it is typically performed in an iterative manner, where each iteration involves forward modeling and migration. Although conventional seismic migration is less computationally expensive than least-squares migration, it generally contains migration artifacts affecting amplitude fidelity and resolution (Dong et al., 2012; Dai and Schuster, 2013). This may be attributed to the

Parts of this chapter were first published in Greer et al. (2018). This work was done under the supervision of Dr. Sergey Fomel, and Dr. Zhiguang Xue assisted in the migration of the Sigsbee synthetic data set.

fact that, while least-squares migration inverts for subsurface reflectivity by finding the least-squares model solution, standard migration amounts to applying a single adjoint operation (Claerbout, 1992).

Various methods have attempted to correct these differences by finding and applying an approximation to the inverse Hessian operator, $\mathbf{H}^{-1} = \left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)^{-1}$, to a conventional migrated image. Here, $\mathbf{L}$ is a standard forward-modeling operator, and $\mathbf{L}^{\mathsf{T}}$ is its adjoint—the migration operator. These methods usually take the form of two approaches—for preconditioning before least-squares migration and as a single operation to improve accuracy of a conventionally migrated image. Previously, this has been done by migration deconvolution (Hu et al., 2001; Yu et al., 2006), approximating the diagonal of $\mathbf{H}^{-1}$ to account for amplitude effects (Rickett, 2003; Sacchi et al., 2007), and by finding and applying a bank of non-stationary matching filters (Guitton, 2004, 2017) or deblurring filters (Aoki and Schuster, 2009) to a conventionally migrated image. This traditionally is done using a sliding window approach, where windowed regions and partial overlap regions are specified, and different matching filters are specified in each region (Schuster, 2017).

In this paper, we take a different approach. We note that the two primary differences between least squares migrated images and conventional migrated images are amplitude and frequency variations (Hou and Symes, 2015, 2016). Here, we treat this as a data matching problem between two conventionally migrated images, and find separate operations to account for both amplitude and frequency variations.

This approach enables us to rely on local seismic attributes to measure and apply amplitude and frequency balancing operations instead of using a sliding window approach (Fomel, 2007a). This allows for the smooth estimation and application

of these matching operations instead of applying them in discrete windows. Since the operations for balancing amplitude and frequency content are calculated and applied separately from each another, the effect of each operation can be adjusted independently, which is another advantage of the proposed method. To test the proposed approach, we apply this method to an example of reverse-time migration on the Sigsbee synthetic data set (Paffenholz et al., 2002).

THEORY

The goal of least-squares migration is to find the image, $\hat{\mathbf{r}}$, that minimizes

$$p(\hat{\mathbf{r}}) = \frac{1}{2} \left\| \mathbf{d} - \mathbf{L}\hat{\mathbf{r}} \right\|_2^2 \ , \tag{5.1}$$

where $\mathbf{L}$ is the forward modeling operator, representing seismic wave propagation through the subsurface, and $\mathbf{d}$ is the acquired seismic data. This can be solved by the least-squares formulation to find $\hat{\mathbf{r}}$:

$$\hat{\mathbf{r}} = \left(\mathbf{L}^\mathsf{T}\mathbf{L}\right)^{-1} \mathbf{L}^\mathsf{T}\mathbf{d} \ , \tag{5.2}$$

where the migration operator, $\mathbf{L}^\mathsf{T}$, is adjoint to the forward modeling operator and is typically sparse, and $\left(\mathbf{L}^\mathsf{T}\mathbf{L}\right)^{-1}$ is the inverse Hessian operator. Equation (5.2) is usually solved iteratively, typically requiring multiple iterations of forward modeling and remigrating the seismic image (Kuehl and Sacchi, 2003; Xue et al., 2016). Conventional migration is less computationally expensive:

$$\mathbf{m}_0 = \mathbf{L}^\mathsf{T}\mathbf{d}. \tag{5.3}$$

However, conventionally migrated images generally exhibit less correct amplitude and frequency content than least-squares migrated images (Dutta et al., 2014). By

combining equations (5.2) and (5.3), it is evident that

$$\hat{\mathbf{r}} = \left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)^{-1}\mathbf{m}_0 \ , \tag{5.4}$$

so $\mathbf{m}_0$ is a distorted version of $\hat{\mathbf{r}}$, and $\hat{\mathbf{r}}$ can be recovered from $\mathbf{m}_0$ by finding a good approximation of $\left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)^{-1}$.

METHOD

In order to approximate $\left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)^{-1}$, we follow the modeling and remigration process of Guitton (2004). We begin by forward modeling the migrated image, $\mathbf{m}_0$:

$$\mathbf{d}_1 = \mathbf{L}\mathbf{m}_0 \ . \tag{5.5}$$

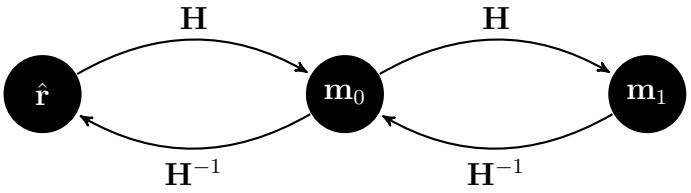
We then remigrate $\mathbf{d}_1$:

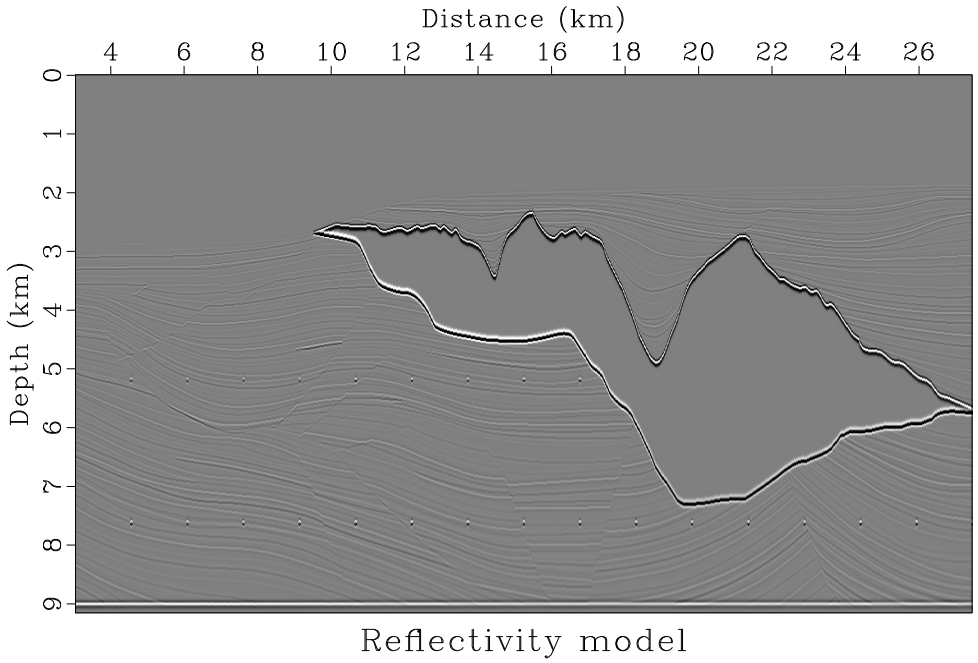
$$\mathbf{m}_1 = \mathbf{L}^{\mathsf{T}}\mathbf{d}_1 = \left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)\mathbf{m}_0 \ , \tag{5.6}$$

and then find the operator, $\left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)^{-1}$, that satisfies

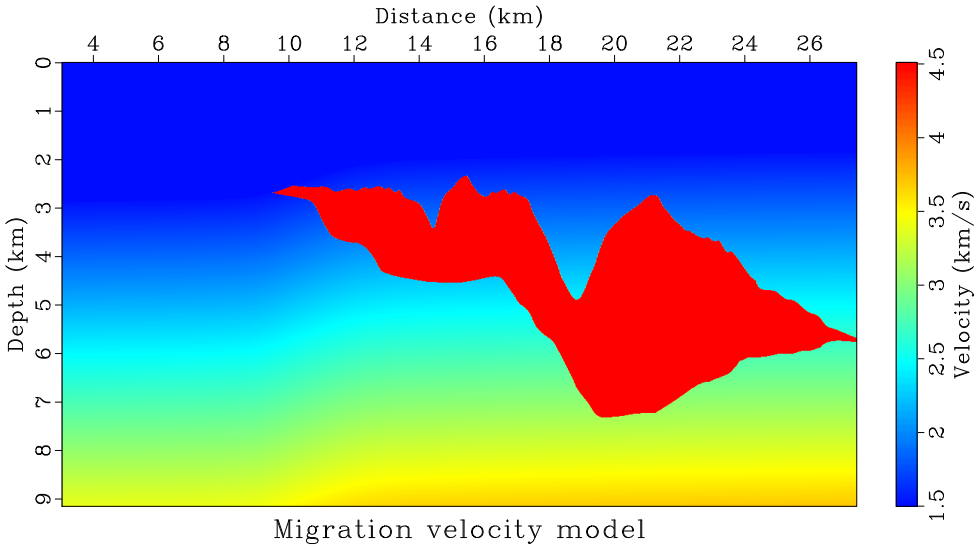
$$\mathbf{m}_0 = \left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)^{-1}\mathbf{m}_1 \ . \tag{5.7}$$

Therefore, the inverse Hessian operator, $\mathbf{H}^{-1} = \left(\mathbf{L}^{\mathsf{T}}\mathbf{L}\right)^{-1}$, that must be applied to $\mathbf{m}_0$ to get $\hat{\mathbf{r}}$ can be calculated by first finding the transformation, $\mathbf{H}$, that maps $\mathbf{m}_0$ to $\mathbf{m}_1$, and then inverting it. This can be interpreted as a data matching problem between $\mathbf{m}_1$ and $\mathbf{m}_0$.



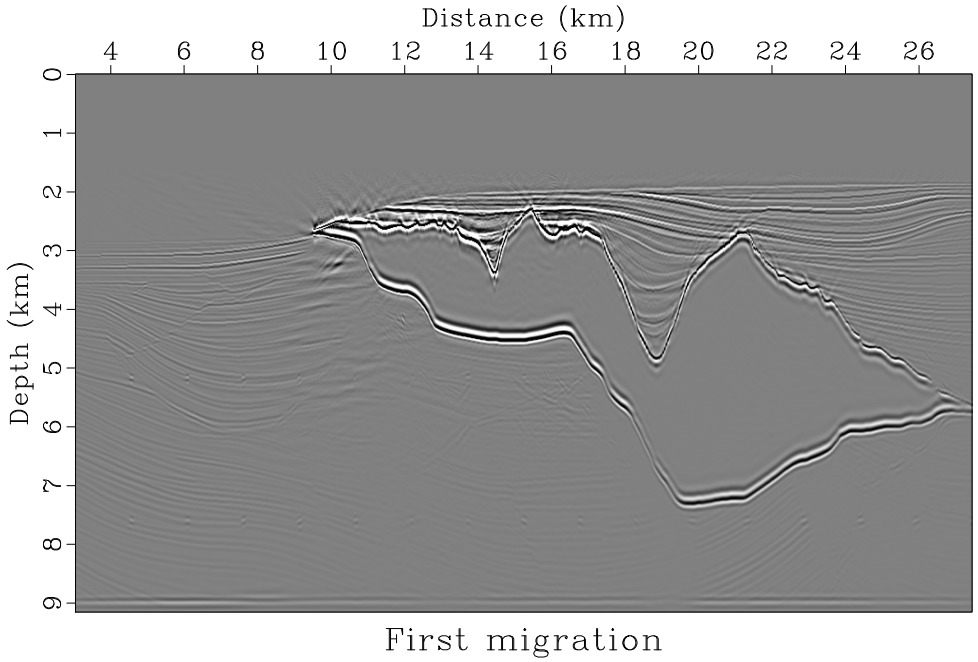


(a)

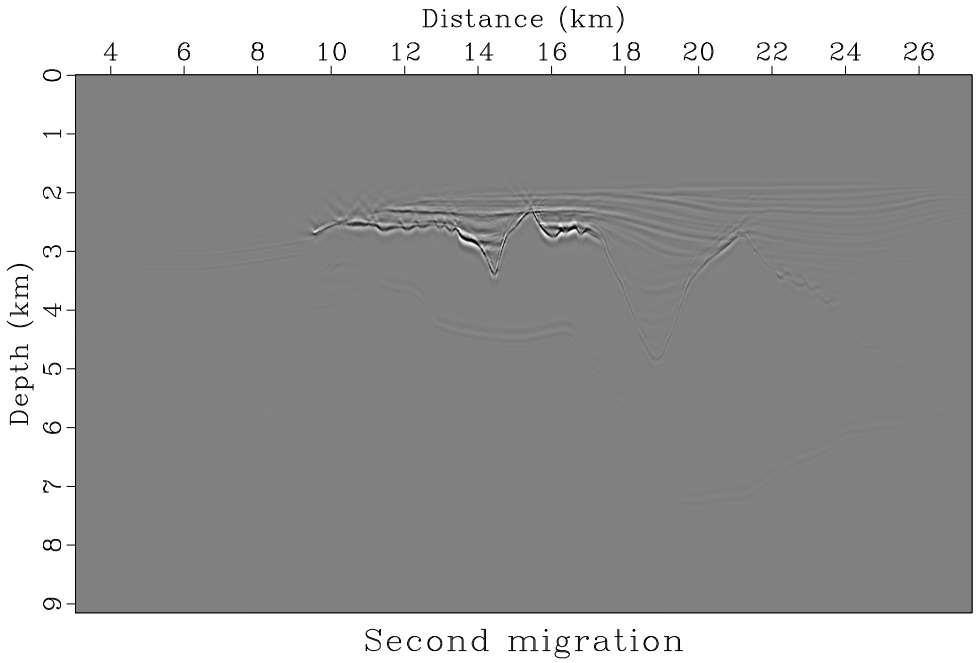


(b)

Figure 5.1: The Sigsbee model reflectivity (a), and migration velocity model (b). chapter-mighes/sigsbee mod,vel-migration



(a)



(b)

Figure 5.2: The first migrated image, $\mathbf{m}_0$ (a), and the second migrated image, $\mathbf{m}_1$ (b). chapter-mighes/sigsbee image0,image1

Since the primary differences between conventionally migrated images and least-squares migrated images amount to amplitude and frequency variations, we use two separate non-stationary operators to represent the forward Hessian—one to account for amplitude variations, and the other to account for frequency variations. Therefore, our approximation of $\mathbf{H}$ can be calculated from some application of a non-stationary amplitude balancing operator, $\mathbf{A}$, and a non-stationary frequency balancing operator, $\mathbf{S}$. We first calculate the forward Hessian since the forward frequency balancing operation, $\mathbf{S}$, is well defined and simple to calculate. We then invert our approximation to the Hessian and apply it to $\mathbf{m}_0$ to correct the amplitude and frequency content of the migrated image.

Amplitude operator

First, we choose to find an amplitude balancing operation that, when applied to $\mathbf{m}_0$, balances the amplitudes of $\mathbf{m}_0$ with respect to $\mathbf{m}_1$. This operation can be equated to a trace-by-trace multiplication of a diagonal matrix to each trace, where the matrix changes for every trace. We estimate it by first calculating the amplitude envelope of the traces in $\mathbf{m}_0$ and $\mathbf{m}_1$, and then smoothly dividing them. The corresponding diagonal weighting operator, $\mathbf{A}$, can be applied such that $\mathbf{m}_1 \approx \mathbf{A}\mathbf{m}_0$ to balance the amplitudes of each trace. Since this is a linear operation that only has diagonal elements, it is trivial to find the inverse operator, $\mathbf{A}^{-1}$.

Frequency operator

In addition to amplitude corrections, we also attempt to account for the decrease in resolution of a conventional migrated image compared to its corresponding least-squares migrated image. This loss in resolution can be equated to the fact that

$\mathbf{L}^\mathsf{T}\mathbf{L}$ acts as a blurring operator, where the conventional migrated image is a blurred version of the ideal least-squares migrated image (Hu et al., 2001). We choose to approximate this blurring using non-stationary triangle smoothing.

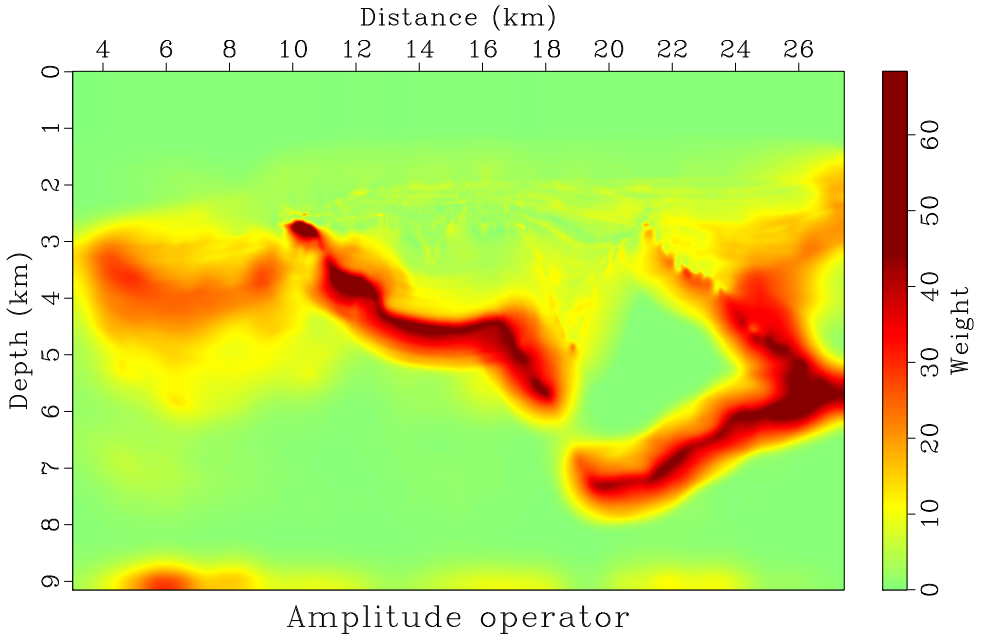
We begin by first calculating the local frequency of the two initial images (Fomel, 2007a). Local frequency is a temporally and spatially varying frequency attribute that smoothly varies across the image without windows. Our goal is to find a transformation that we can apply to $\mathbf{m}_0$ that matches the local frequency content with $\mathbf{m}_1$. To do this, we propose using non-stationary triangle smoothing. This approach involves finding and applying a non-stationary smoothing operator, which is the amount of samples, in both dimensions, that $\mathbf{m}_0$ will be averaged over in a triangle weight, to balance the local frequency content with $\mathbf{m}_1$.

We find the smoothing radius iteratively using the method of Greer and Fomel (2017a) from chapter 3, with a modification that allows the smoothing radius to be calculated in both spatial directions. Essentially, this is found by choosing an initial guess of a smoothing radius, $\mathbf{R}^{(0)}$, and updating it iteratively such that

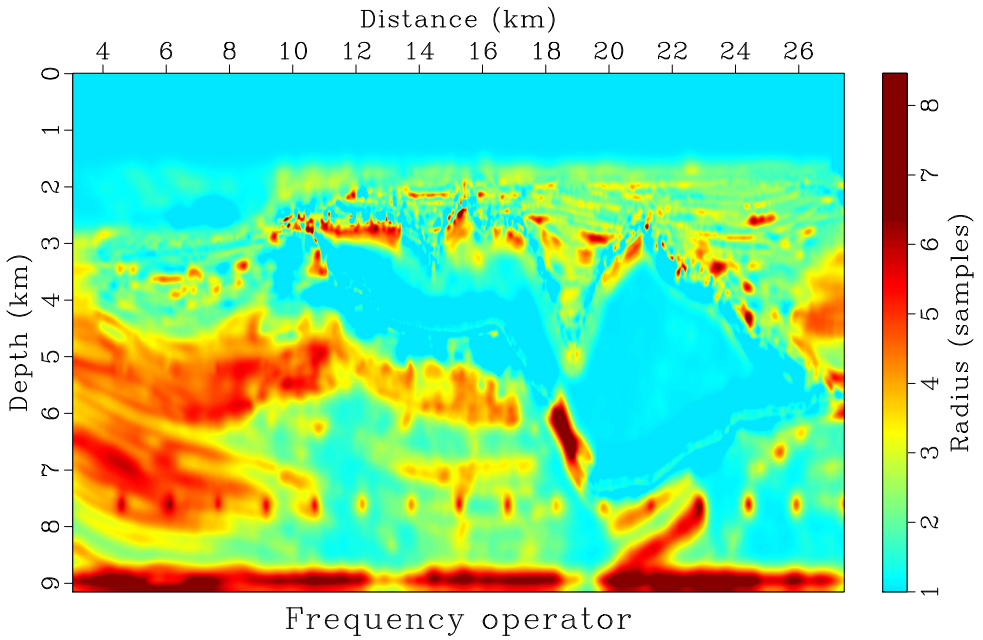
$$\mathbf{R}^{(i+1)} = \mathbf{R}^{(i)} + \alpha \left[\mathbf{F}[\mathbf{S}_{\mathbf{R}^{(i)}}\mathbf{m}_0] - \mathbf{F}[\mathbf{m}_1]\right] \ , \tag{5.8}$$

where $\mathbf{F}$ is the local frequency operator, $\mathbf{S}_{\mathbf{R}^{(i)}}$ is the smoothing operator of radius $\mathbf{R}$ at the ith iteration, and α is a scalar constant that represents the step length. After a small number of iterations, the smoothing operator is found that, once applied to $\mathbf{m}_0$, balances local frequency content with $\mathbf{m}_1$.

For this particular application using depth migration, this operator should technically be specified to balance *wavenumber* instead of *frequency*. However, it is kept as frequency to keep consistent terminology with the algorithm developed in chapter 3.



(a)



(b)

Figure 5.3: The forward amplitude balancing weight, **A** (a) and the smoothing radius (b), which represents the number of samples in both dimensions that $\mathbf{m}_0$ must be smoothed over in a triangle weight to balance the local frequency content with $\mathbf{m}_1$. This represents the forward smoothing operation, **S**. chapter-mighes/sigsbee a0,rect10b

Calculating the inverse Hessian

Now that we have found the forward operators that separately balance amplitude and frequency content from $\mathbf{m}_0$ to $\mathbf{m_1}$, where $\mathbf{m}_1 \approx \mathbf{H}\mathbf{m}_0$, we want to find what combination of $\mathbf{A}$ and $\mathbf{S}$ best approximates $\mathbf{H}$. Since $\mathbf{H} = \mathbf{L}^\mathsf{T}\mathbf{L}$ is symmetric, we want our approximation of $\mathbf{H}$ to be as close to symmetric as possible. Therefore, we define $\mathbf{H}$ as

$$\mathbf{H} \approx \mathbf{A}^{1/2}\mathbf{S}\mathbf{A}^{1/2} \,, \tag{5.9}$$

where $\mathbf{A}$ is the operator that balances the amplitudes of $\mathbf{m}_0$ with respect to $\mathbf{m}_1$, and $\mathbf{S}$ is the operator that balances the local frequency content of $\mathbf{m}_0$ with respect to $\mathbf{m}_1$, both defined previously. Applying the operations in this order allows the approximation of $\mathbf{H}$ to be symmetric, since both $\mathbf{A}$ and $\mathbf{S}$ are symmetric operations. Splitting up our approximation to the Hessian into two separate operators allows us to control how much of each operation and the order of each operation goes into correcting the image, and see how it affects the resulting image. Now that we have found the forward Hessian, $\mathbf{H}$, such that $\mathbf{H}\mathbf{m}_0 \approx \mathbf{m}_1$ using data matching operators, we want to find the inverse of this operator, $\mathbf{H}^{-1}$, such that $\hat{\mathbf{r}} \approx \mathbf{H}^{-1}\mathbf{m}_0$ provides us with the least-squares image. This is found as

$$\mathbf{H}^{-1} \approx \left(\mathbf{A}^{1/2}\mathbf{S}\mathbf{A}^{1/2}\right)^{-1} = \mathbf{A}^{-1/2}\mathbf{S}^{-1}\mathbf{A}^{-1/2} \,. \tag{5.10}$$

Since the amplitude operators only contain diagonal terms, they are simple to invert. However, $\mathbf{S}^{-1}$ is non-trivial to calculate since inverse smoothing can create physically unrealistic high-frequency data if inverted incorrectly without regularization.

Figure 5.4 shows transfer functions for a stationary forward and inverse triangle smoothing operator of a radius of 10 samples. In the forward case, triangle

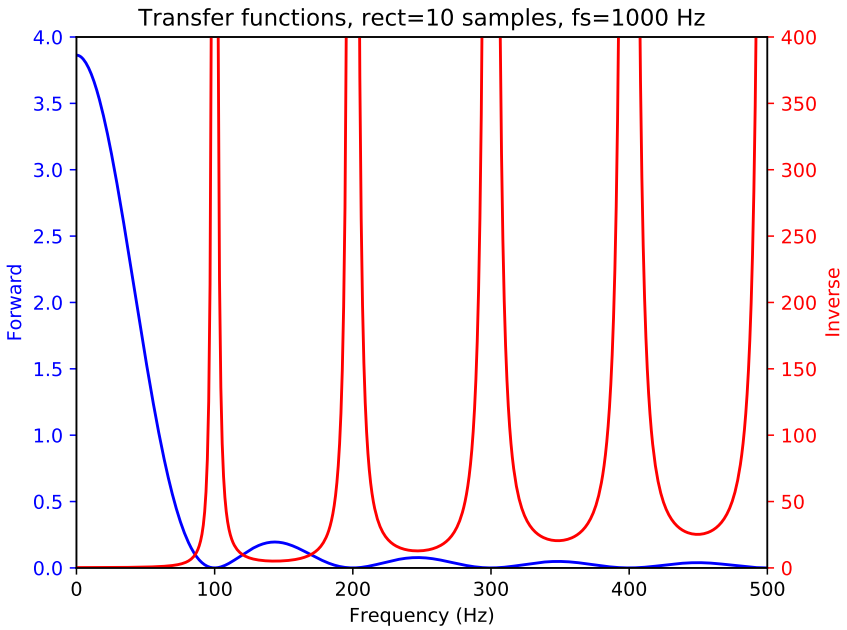


Figure 5.4: Transfer functions for a stationary forward triangle smoothing operator (blue) and its inverse (red). chapter-mighes/triop tf

smoothing acts as a low-pass filter. However, its inverse can introduce high frequency information, which is physically unrealistic for the data we are working with.

Therefore, $\mathbf{S}^{-1}$ must be calculated with care to ensure the inverted data is physically plausible. We iteratively invert $\mathbf{S}$ using shaping regularization (Fomel, 2007b), where the shaping operator is a bandpass filter picked to ensure the passband contains only physically possible frequencies for the given data set. The cost of applying our approximation to $\mathbf{H}^{-1}$ in equation (5.10) is $\mathcal{O}(N)$, where N is the image size. The constant is small, typically around 10 for the number of iterations, and the calculation and application of this approximation is computationally insignificant compared to one iteration of least-squares migration.

EXAMPLE

We demonstrate the effectiveness of this method on the 2D Sigsbee model. The Sigsbee2A 2D synthetic data set was created to mimic the geology of the Sigsbee escarpment in the Gulf of Mexico (Paffenholz et al., 2002). A fixed-spread acquisition survey is generated, which consists of 301 shots spaced every 122 m. The source wavelet for generating the synthetic data is a Ricker wavelet centered at 10 Hz. The record length of the synthetic data is 10 s with a time step of 4 ms. We use reverse-time migration (RTM) as our migration operator.

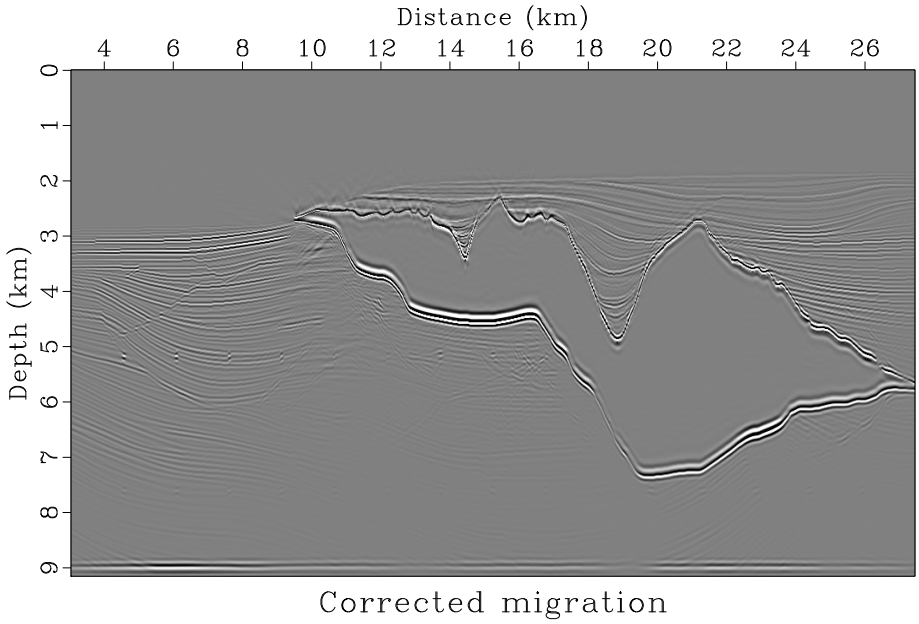


Figure 5.5: The corrected migrated image, found by applying equation (5.10) to $\mathbf{m}_0$. chapter-mighes/sigsbee migdec-shap

We begin with the sub-surface reflectivity model (Figure 5.1(a)) and migration velocity model (Figure 5.1(b)). Next, we forward model the seismic data and migrate it to get our first conventionally migrated image, $\mathbf{m}_0$ (Figure 5.2(a)). Then, we forward model $\mathbf{m}_0$ and remigrate that data to get $\mathbf{m}_1$ (Figure 5.2(b)). This provides

us with the two images, $\mathbf{m}_0$ and $\mathbf{m}_1$, that we can use to find the operation $\mathbf{H}$ that maps $\mathbf{m}_0$ to $\mathbf{m}_1$.

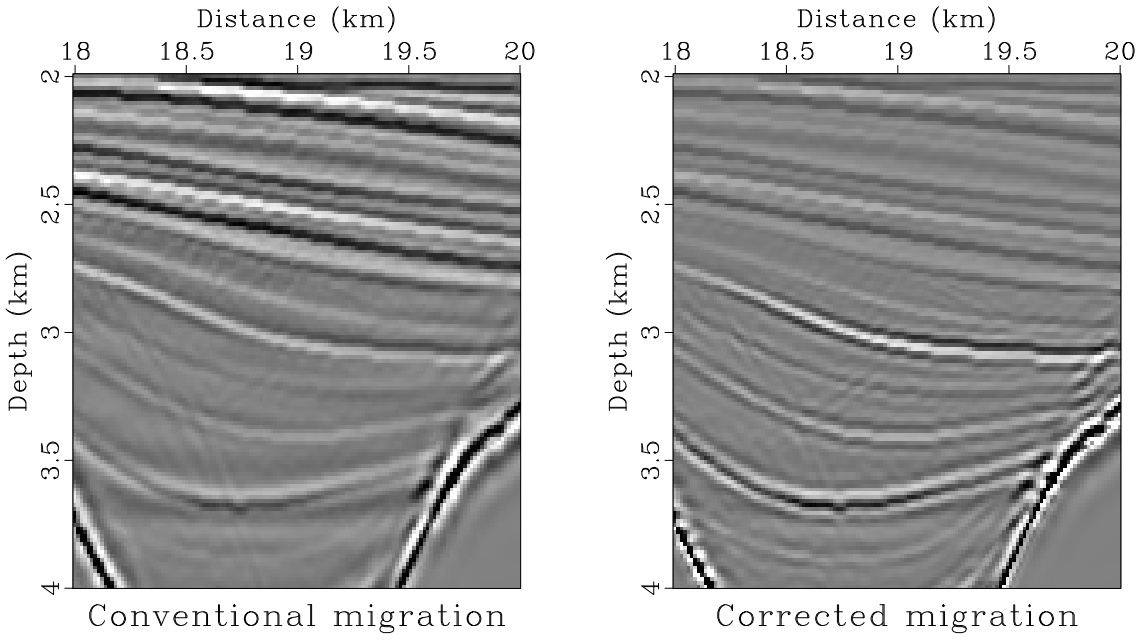
Next, we calculate and apply the data matching operations as described in the previous section. The forward amplitude balancing weight, $\mathbf{A}$, is shown in Figure 5.3(a). The calculated radius for the forward smoothing operation is shown in Figure 5.3(b). After applying these two operators to $\mathbf{m}_0$ as described by equation (5.10), we produce the corrected migrated image, as shown in Figure 5.5. This corrected image better represents the subsurface reflectivity than the conventionally migrated image (Figure 5.2(a)), as it exhibits more correct amplitude content and higher resolution comparable with the reflectivity model.

Figure 5.6 shows a windowed section of the reflectivity model, the conventionally migrated image, and the corrected migrated image. The corrected migrated image exhibits clearly higher resolution and has more correct and consistent amplitude content than the conventionally migrated image.

In addition to directly applying this operator to the conventionally migrated image to improve resolution, this operator can be used as a preconditioner in iterative least-squares migration. In this case, the corrected migrated image could be used as an initial model for least-squares migration for faster convergence.

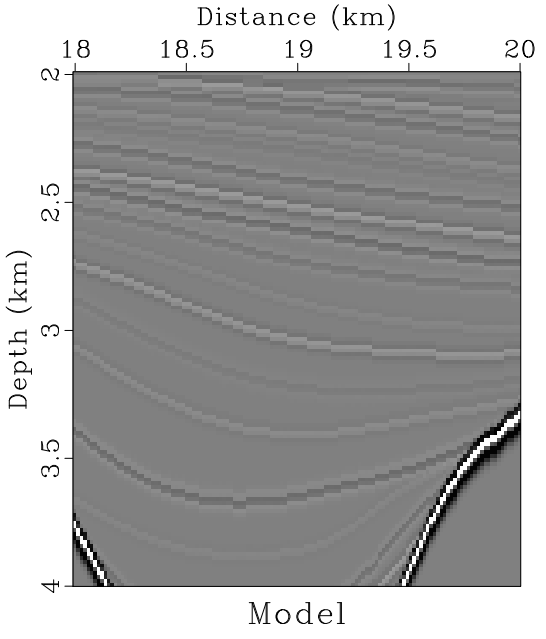
CONCLUSIONS

Least-squares migration can produce an accurate migrated image, but it is more computationally expensive than conventional migration. In this paper, we apply an approximate inverse Hessian operator to a conventional migrated image to approximate the least-squares migrated image. Since the primary differences between least-squares



(a)

(b)



(c)

Figure 5.6: The first migrated image (a), the corrected migrated image (b), and the Sigsbee model reflectivity (c). chapter-mighes/sigsbee image0-w3,migdec-w3,mod-w3

migration and conventional migration amount to amplitude and frequency variations, we approximate the forward Hessian by calculating frequency and amplitude matching operators. The amplitude matching operator is found by calculating the amplitude envelopes of migrated and remigrated images and smoothly dividing them, and the frequency matching operator is found using an iterative algorithm and non-stationary smoothing. The Hessian is approximated by a combination of these two operators to ensure symmetry. This method involves a windowless approach and is cheap to calculate and apply. Additionally, by defining the Hessian with two separate operators, we can examine, and control, the "ingredients" of the Hessian operator, and see how changing them impacts the final image.

After the forward Hessian is calculated, we invert it iteratively using shaping regularization, and apply it to the conventionally migrated image to get an approximation of the least-squares migrated image. Successful results are achieved on the 2D Sigsbee synthetic model with reverse-time migration as the migration operator.

ACKNOWLEDGEMENTS

We thank the sponsors of the Texas Consortium for Computational Seismology (TCCS) for their financial support. The examples in this paper can be reproduced using the Madagascar open-source software package (Fomel et al., 2013).

Chapter 6

Conclusions

Summary

In this thesis, I discussed several methods and applications of data matching in seismic data analysis.

Chapter 2 focuses on introducing the three data matching operators that are used in this thesis—shifting, scaling, and filtering.

Chapter 3 introduces different methods of frequency balancing using non-stationary smoothing. The first method to find the non-stationary smoothing *radius*, or number of samples each data point is averaged over with a triangle weight, took a theoretical approach based off of the assumption that the data we observe can be modeled by a summation of Ricker wavelets. This method worked well in certain situations, but was not robust enough to work for any given data set. In the second method, I introduced an iterative algorithm to find the smoothing radius, and this method converges quickly and works well in several presented situations. Finally, a modification to this algorithm was shown and allows smoothing for more complex data sets.

This chapter also discusses two applications of these algorithms—the frequency balancing algorithm was demonstrated on an application of matching high-resolution and legacy seismic images, and the modified algorithm was demonstrated on an application of multicomponent seismic image registration.

Chapter 4 goes into more detail of the application of matching and merging high-resolution and legacy seismic images. This example takes two seismic volumes, acquired over the same area but using different technologies, and first matches them before merging them together to produce an optimized third image. First, the method is demonstrated on a 2D line from the Gulf of Mexico. Then, the method is applied to a 3D seismic volume from a different part of the Gulf of Mexico.

Chapter 5 discusses another application of improving migration resolution by approximating the least-squares Hessian using non-stationary data matching operations. An approximation to the least-squares Hessian can calculated by solving a data matching problem between two conventionally migrated images, and can be represented by the combination of amplitude and frequency balancing operations. An example is applied to a 2D synthetic Sigsbee data set.

Future work

In the future, the work presented in chapter 5 should be extended to involve real data and 3D examples. It also could benefit by comparing the results of the proposed approach taken in the chapter to other previous approaches presented to approximate the least-squares Hessian (Hu and Schuster, 1998; Dong et al., 2012; Casasanta et al., 2017; Dai and Schuster, 2013; Sacchi et al., 2007; Aoki and Schuster, 2009; Yu et al., 2006; Hu et al., 2001), to see how it compares in different situations.

Another extension of this data matching procedure may be to incorporate the phase of the signal to be matched. Negligible improvements were made when trying to incorporate phase corrections into the high-resolution and legacy data matching problem of chapter 4, but other data matching problems may benefit from these

corrections.

Several applications of data matching were discussed in this thesis. However, there are still many applications that may remain unaddressed from a data matching standpoint. Problems such as seismic and well-log tying, deconvolution, automatic gain control (AGC), and surface-related multiple elimination (SRME) can also be recast as data matching problems, and looking at these problems in a new light may bring advancements to computational geophysics.

Appendix

CODE

The examples in this thesis were implemented with the Madagascar open-source software environment for reproducible computational experiments (Fomel et al., 2013). The package is available at `http://www.ahay.org/`.

The reproducible document for the results in this thesis, including code, is available at `http://www.sygreer.com/research/honorsThesis`. However, some of the data used in this thesis are proprietary, so those results may not directly be reproducible.

The scripts and programs used to produce the examples in this thesis are below.

Table 1: Figures and the scripts to generate them

Figures	**Directory**	**Listings**
2.1	chapter-locfreq/merge/	2, 3, 4
2.2, 2.3	chapter-background/dmExample/	1
3.1, 3.2, 3.3, 3.4	chapter-merge/apache/	10
3.5, 3.6, 3.7, 3.8	chapter-locfreq/merge/	2, 3, 4
3.9, 3.10	chapter-locfreq/vecta/	5, 6, 7, 8
3.11	chapter-locfreq/convergence/	9
4.1, 4.2, 4.3, 4.4	chapter-merge/apache/	10
4.5, 4.6	chapter-merge/pcable/	11
4.7	chapter-merge/pcable2/	12, 13, 14
5.1, 5.2, 5.3, 5.5, 5.6	chapter-merge/mighes/	15, 16, 17
5.4,	chapter-merge/triop/	18, 19 20

Chapter 2

Listing 1: chapter-background/dmExample/dmExample.py

```
#!/usr/bin/env python

from scipy.fftpack import fft
from numpy import linspace, concatenate, asarray, sin, cos, pi
from numpy.linalg import norm
import matplotlib.pyplot as plt

# fonts and such
bg_color = 'black'
fg_color = 'white'
fig = plt.figure(facecolor=bg_color, edgecolor=fg_color)
axes = plt.axes(facecolor=bg_color, frameon=True)

def foldConv(signal,filt):
    """ 1D convolution with folding """

    # length of arrays and center
    fl = len(filt)
    sl = len(signal)
    center = int(len(filt)/2)

    # initialize convolved signal
    conv = [0]*sl

    # filter must be smaller than signal
    if fl > sl:
        raise ValueError, "Filter longer than signal"

    # convolve
    for i in range(0,sl):
        # center
        conv[i] += filt[center]*signal[i]

        # left
        for j in range(0,center):
            idx = abs(i-center+j)
            conv[i] += filt[j]*signal[idx]

        # right
        for j in range(center + 1, fl):
            idx = i+j-center
            if idx >= sl:
                idx = (sl-1) - (idx - (sl-1))
            conv[i] += filt[j]*signal[idx]

    return conv

def triOper(rect):
    """ create triangle smoothing operator """
    if rect < 1:
        raise ValueError, "Rect minimum size 1"

    # create triangle operator
    t=linspace(1,rect-1,rect-1)
    trian = concatenate((t,[rect],t[::-1]),axis=0)

    # normalize and return
    return trian/norm(trian)
```

```
59
60 def sigPlot(name):
61     fig.set_size_inches(10, 2)
62     frame1 = plt.gca()
63     frame1.axes.get_xaxis().set_visible(False)
64
65     plt.ylim(ymin=-1.2, ymax=1.2)
66     axes.spines['left'].set_visible(False)
67     axes.spines['bottom'].set_visible(False)
68     plt.savefig('Fig/%s'%name, format='pdf', bbox_inches='tight',
69                 transparent=True, dpi=200)
70
71     plt.cla()
72
73 def freqPlot(signal, name, col):
74     scale=20                    # freq scale
75     smallTri = triOper(4)       # smoothing
76     num = 5000                  # freq density
77
78     Y = fft(signal, num)
79     Y = Y[range(len(signal)/scale)]
80     Y = foldConv(Y,smallTri)
81     plt.plot(abs(Y/max(Y)), color=col)
82     plt.xlabel("Frequency (Hz)")
83     fig.set_size_inches(3, 2)
84     frame1 = plt.gca()
85     frame1.axes.get_xaxis().set_visible(True)
86     axes.spines['left'].set_visible(True)
87     axes.spines['bottom'].set_visible(True)
88     plt.xticks([])
89     plt.savefig(name, format='pdf', bbox_inches='tight', transparent=True, dpi=200)
90     plt.cla()
91
92 def main():
93
94     # read data and convert
95     f = open('trace.txt', 'r')
96     signal = f.readlines()
97     signal = [y.strip() for y in signal]
98     signal = map(float, signal)
99     signal = asarray(signal)
100
101    signal = signal[3900:5000]
102
103    # create triangle operators
104    tri = triOper(50)
105
106    leg = foldConv(signal,tri)
107
108    # time
109    t = linspace(0,len(signal),len(signal))
110
111    # amplitude
112    amp = 0.1*sin(2*pi*t) - 0.2*cos(2*pi*2*t)
113    t2 = t + 3*(1*sin(2*pi*t) - 4*cos(2*pi*2*t)+4)
114
115    # plot everything
116    ########################################
117    # display stuffs
118    frame1 = plt.gca()
119    frame1.axes.get_yaxis().set_visible(False)
```

```
    # plot color stuff -- white on black
    axes.xaxis.set_tick_params(color=fg_color, labelcolor=fg_color)
    axes.yaxis.set_tick_params(color=fg_color, labelcolor=fg_color)
    axes.xaxis.label.set_color(fg_color)
    axes.yaxis.label.set_color(fg_color)
    axes.spines['right'].set_visible(False)
    axes.spines['top'].set_visible(False)
    axes.spines['left'].set_visible(False)
    axes.spines['bottom'].set_visible(False)
    fig.set_size_inches(15, 4)
    frame1.axes.get_xaxis().set_visible(False)
    for spine in axes.spines.values():
        spine.set_color(fg_color)
    ########################################
    signal = signal / max(signal)
    signal = signal + amp
    leg = leg / max(leg)

    # time domain
    plt.plot(t, leg, color="black")
    plt.plot(t2, signal, color="red")
    sigPlot("bef.pdf")


    plt.plot(t, leg, color="black")
    plt.plot(t2, signal, color="red")
    sigPlot("one0.pdf")

    # BALANCE FREQ THEN AMP
    # balance freq
    sig2 = foldConv(signal,tri)
    sig2 = sig2 / max(sig2)

    plt.plot(t, leg, color="black")
    plt.plot(t2, sig2, color="red")
    sigPlot("one1.pdf")

    # balance amp
    ampBal = foldConv(leg,tri)
    ampBal = foldConv(ampBal,tri)
    ampBal = ampBal / (max(ampBal)*1.5)
    sig3 = sig2 + ampBal
    sig3 = sig3 / max(sig3)

    plt.plot(t, leg, color="black")
    plt.plot(t2, sig3, color="red")
    sigPlot("one2.pdf")

    # time shifts

    plt.plot(t, leg, color="black")
    plt.plot(t, sig3, color="red")
    sigPlot("one3.pdf")


    # BALANCE AMP THEN FREQ

    # balance amp
    ampBalb = foldConv(leg,tri)
    ampBalb = foldConv(ampBalb,tri)
```

```
    ampBalb = ampBalb / (max(ampBalb)*1.5)
    sigb2 = signal + ampBal
    sigb2 = sigb2 / max(sigb2)

    plt.plot(t, leg, color="black")
    plt.plot(t2, sigb2, color="yellow")
    sigPlot("two1.pdf")

    # balance freq
    sigb3 = foldConv(sigb2,tri)
    sigb3 = sigb3 / max(sigb3)

    plt.plot(t, leg, color="black")
    plt.plot(t2, sigb3, color="yellow")
    sigPlot("two2.pdf")

    # time shifts
    plt.plot(t, leg, color="black")
    plt.plot(t, sigb3, color="yellow")
    sigPlot("two3.pdf")

    #after
    plt.plot(t, leg, color="black")
    plt.plot(t, sig3, color="green")
    plt.plot(t, sigb3, color="yellow")
    sigPlot("aft.pdf")

if __name__ == "__main__":
    main()
```

Chapter 3

Listing 2: chapter-locfreq/merge/SConstruct

```
from rsf.proj import *
from radius import radius

# Initial figures
Result('legacy','grey title="Legacy"')
Result('hires-agc','hires','agc rect1=2000 rect2=5 | grey title="High-resolution"')

Flow('legacy-spec','legacy','spectra all=y')
Result('nspectra-orig','high-spec legacy-spec',
    '''cat axis=2 ${SOURCES[1]} |
       scale axis=1 | window max1=180 |
       graph title="Normalized Spectra" label2="Amplitude" unit2=""''')

# Balance local frequency
flol=40
corrections = 5
Flow('legacyfilt','legacy','bandpass flo=%d '%(flol))
radius('hires','legacyfilt', corrections, [0.13,0.2,0.3,0.5,0.5],
       bias=0, clip=90, rect1=80, rect2=16, maxval=90 )

End()
```

Listing 3: chapter-locfreq/merge/radius.py

```
from rsf.proj import *

def radius(high, low,                # initial high-resolution and legacy images
           niter,                    # number of corrections
           c,                        # 'step length' for radius corrections. Can
                                     # be type int or float for constant c
                                     # or type array for changing c.
           bias=-15, clip=30,        # bias and clip for display
           rect1=40, rect2=80,       # radius for local frequency calculation
           maxrad=1000,              # maximum allowed radius
           theor=True,               # use theoretical smoothing radius
           scale=9,                  # scale for theoretical smoothing radius
           initial=10,               # initial value for contant smoothing radius
           minval=0,
           maxval=25,
           titlehigh="Hires",
           titlelow="Legacy"):

    if type(c) is float or type(c) is int:
        c = [c]*niter

    def seisplot(name):
        return 'grey title="%s"'%name

    locfreq = '''iphase order=10 rect1=%d rect2=%d hertz=y complex=y |
                 put label="Frequency" unit=Hz'''%(rect1,rect2)

    def locfreqplot(name):
        return 'grey mean=y color=j scalebar=y title="%s" '%name

    freqdif = 'add scale=-1,1 ${SOURCES[1]} | put label=Frequency'

    def freqdifplot(num):
        return '''grey allpos=y color=j scalebar=y mean=y
                  title="Difference in Local Frequencies %s"
                  clip=%d bias=%d minval=%d
                  maxval=%d''' %(num,clip,bias,minval,maxval)

    specplot = '''cat axis=2 ${SOURCES[1]} |
                  scale axis=1 | window max1=180 |
                  graph title="Normalized Spectra" label2="Amplitude" unit2=""'''

    def rectplot(name):
        return '''grey color=j mean=y title="%s" scalebar=y barlabel=Radius
                  barunit=samples'''%name

    smooth = 'nsmooth1 rect=${SOURCES[1]}'

    Result(high, seisplot(titlehigh))
    Result(low, seisplot(titlelow))

    Flow('high-freq',high,locfreq)
    Result('high-freq',locfreqplot('%s Local Frequency'%titlehigh))

    Flow('low-freq',low,locfreq)
    Result('low-freq',locfreqplot('%s Local Frequency'%titlelow))

    locfreq2 = '''iphase order=10 rect1=1 rect2=1 hertz=y complex=y |
                  put label="Frequency" unit=Hz'''

    Flow('low-freq2',low,locfreq2)
```

```
    Result('low-freq2',locfreqplot('%s Instantaneous Frequency'%titlelow))

    Flow('freqdif','low-freq high-freq',freqdif)
    Result('freqdif',freqdifplot(''))

    # initial smoothing radius
    if (theor):
        from math import pi
        Flow('rect0','low-freq high-freq',''' math f1=${SOURCES[1]}
        output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" '''%(scale,2*pi*0.001))
    else:
        Flow('rect0','low-freq','math output=%f'%initial)

    Result('rect0',rectplot("Smoothing Radius 0"))

    Flow('high-smooth0','%s rect0' % high,smooth)
    Result('high-smooth0', seisplot("%s Smooth 0"%titlehigh))

    Flow('high-spec',high,'spectra all=y')
    Flow('low-spec',low,'spectra all=y')
    Flow('high-smooth-spec0','high-smooth0','spectra all=y')
    Result('nspectra','high-spec low-spec',specplot)
    Result('high-smooth-spec0','high-smooth-spec0 low-spec',specplot)

    Flow('high-smooth-freq0','high-smooth0',locfreq)
    Result('high-smooth-freq0',
           locfreqplot("%s Local Frequency Smoothed %d" %(titlehigh,0)))

    Flow('freqdif-filt0','low-freq high-smooth-freq0',freqdif)
    Result('freqdif-filt0',freqdifplot('0'))

    prog=Program('radius.c')
    for i in range(1, niter+1):
        j = i-1
        Flow('rect%d'%i,'rect%d freqdif-filt%d %s'%(j,j,prog[0]),
             './${SOURCES[2]} freq=${SOURCES[1]} c=%f'%c[j])
        Result('rect%d'%i,rectplot("Smoothing Radius %d")%i)

        Flow('high-smooth%d'%i,'%s rect%d'%(high,i),smooth)
        Result('high-smooth%d'%i, seisplot('%s Smooth %d'%(titlehigh,i)))

        Flow('high-smooth-spec%d'%i,'high-smooth%d'%i,'spectra all=y')
        Result('high-smooth-spec%d'%i,'high-smooth-spec%d low-spec'%i,specplot)

        Flow('high-smooth-freq%d'%i,'high-smooth%d'%i,locfreq)
        Result('high-smooth-freq%d'%i,
               locfreqplot('%s Local Frequency Smoothed %d'%(titlehigh,i)))

        Flow('freqdif-filt%d'%i,'low-freq high-smooth-freq%d'%i,freqdif)
        Result('freqdif-filt%d'%i,freqdifplot(str(i)))
```

Listing 4: chapter-locfreq/merge/radius.c

```
/* smoothing radius (min = 1) */
#include <rsf.h>
#include <math.h>

int main (int argc, char* argv[])
{
    int n1, n1f, n2, n2f, i, n12, n12f;
    float *rect, *fr, maxrad, c, *rad;
```

```
    sf_file in, out, freq;

    sf_init (argc,argv);

    in = sf_input("in");
    freq = sf_input("freq");
    out = sf_output("out");

    if (!sf_histint(in,"n1",&n1)) sf_error("No n1= in input.");
    if (!sf_histint(freq,"n1",&n1f)) sf_error("No n1= in frequency difference.");

    n2 = sf_leftsize(in,1);
    n2f = sf_leftsize(freq,1);

    n12 = n1*n2;
    n12f = n1f*n2f;

    if (n1 != n1f) sf_error("Need matching n1");
    if (n2 != n2f) sf_error("Need matching n2");

    if (!sf_getfloat("c",&c)) c=1.;
    if (!sf_getfloat("maxrad",&maxrad)) maxrad=1000.;

    rect = sf_floatalloc(n12);
    sf_floatread(rect,n12,in);

    fr = sf_floatalloc(n12f);
    sf_floatread(fr,n12,freq);

    rad = sf_floatalloc(n12);

    for (i=0; i < n12; i++) {

        /* update radius */
      rad[i] = rect[i]+c*fr[i];

        /* set constraint conditions: [1, maxrad] */
        if (rad[i] > maxrad)
                rad[i] = maxrad;
        if (rad[i] < 1.0)
                rad[i] = 1.0;
    }

    sf_floatwrite(rad,n12,out);
    exit(0);
}
```

Listing 5: chapter-locfreq/vecta/SConstruct

```
from rsf.proj import *
from rsf.recipes.beg import server as private
import newwarplocfreq

##############################################################
#  Modified from $RSFSRC/book/tccs/ltft/vecta/SConstruct  #
##############################################################

trace=300

Flow('line.asc',None,
     'echo %d 0 %d 4 n1=4 data_format=ascii_float in=$TARGET' %
```

```
     (trace,trace))
Plot('line','line.asc',
     '''
     dd form=native | dd type=complex |
     graph min2=0 max2=4 min1=-0.5 max1=471.5 pad=n wantaxis=n wanttitle=n
     ''')

for mode in ['pp','ss']:
    data = 'bend_l1_%cmig_enhanc.sgy' % mode[1]
    Fetch(data,'vecta',private)
    Flow(mode,data,
         '''
         segyread tape=$SOURCE read=data |
         window n2=471 | scale axis=2 | put label2=Trace
         ''',stdin=0)
    Result(mode,mode,'Overlay')
    Result('v'+mode,[mode,'line'],'Overlay')

nails = Split('''
0.32 0.72
0.57 1.22
0.97 1.97
''')

Flow('nails0.asc',None,
     'echo %s n1=2 n2=%d in=$TARGET data_format=ascii_float' %
     (string.join(nails,' '),len(nails)/2))
Flow('nails','nails0.asc','dd form=native')
Flow('nreal','nails','window n1=1')
Flow('nimag','nails','window f1=1')
Plot('nails','nreal nimag',
     '''
     cmplx ${SOURCES[:2]} |
     graph min1=0 max1=2 min2=0 max2=4 symbol='o' wanttitle=n
     label1="PP time (s)" label2="SS time (s)" plotcol=5
     symbolsz=15 labelfat=3 font=2 titlefat=3
     ''',stdin=0)

Flow('fit','nails pp1','linefit pattern=${SOURCES[1]}')
Plot('fit',
     '''
     graph min1=0 max1=2 min2=0 max2=4 title="Line Fit"
     labelfat=4 font=2 titlefat=4
     ''')
Result('vnails','fit nails','Overlay')

Flow('fit0','fit','math output=input-x1 | spray o=0 d=1 n=471')

newwarplocfreq.nwarp2('vec','pp','ss','fit0',
                      nx=471,
                      inter=5,
                      tmax=1.5,
                      ss=1,
                      trace=trace,
                      gmax=2.3,
                      gmin=1.5,
                      dt=0.002,
                      g0=0.9,
                      ng=41,
                      rect1=50,
                      rect2=50,
```

```
                        fmax=70,
                        frect=20,
                        fmin=20,
                        frame1=285,
                        iter=2,
                        clip=0.39)

Result('pi','ppi','Overlay')
Result('si','vec-si-0','Overlay')

box = 'box x0=%g y0=%g label="%s" xt=%g yt=%g lab_fat=3'
x = 201
y = 242
w = 2
w1= 9

Plot('label01',None,box % (8.2,4.5,"A",0.5,-0.5))
Plot('label02',None,box % (8.2,4.5,"B",0.5,-0.5))
Flow('frame.asc',None,
     'echo %s n1=10 data_format=ascii_int in=$TARGET' % \
     string.join(map(str,(x,y,x+w,y,x+w,y+w1,x,y+w1,x,y))))
Plot('frame','frame.asc',
     '''
     dd type=complex form=native |
     graph min1=199 max1=204 min2=233 max2=254 pad=n plotfat=10 plotcol=3
     wantaxis=n wanttitle=n
     ''')

Result('before','vec-in0-0','Overlay')
Result('after','vec-in1-1','Overlay')
End()
```

Listing 6: chapter-locfreq/vecta/newwarplocfreq.py

```
from rsf.proj import *
import math, string, sys
import rsf.recipes.version as version
from radius2 import radius2

##############################################################
#  Modified from $RSFSRC/book/tccs/ltft/vecta/newwarp.py  #
##############################################################

warp0 = '''
warp1 other=${SOURCES[1]} warpin=${SOURCES[2]}
verb=1 nliter=0
'''

getmask = 'add scale=1,-1 ${SOURCES[1]} | mask min=0 | dd type=float'

def psrect(rect):
    return '''
    math min=${SOURCES[2]} max=${SOURCES[1]}
    output="sqrt(1+%d*(1/min^2-1/max^2)*input)" | dd type=int
    ''' % rect

def pprect(rect):
    return '''
    math min=${SOURCES[2]} max=${SOURCES[1]}
    output="sqrt(1+%d*(1/max^2-1/min^2)*(1-input))" | dd type=int
    ''' % rect
```

```
28
balance = '''
30 nsmooth1 rect=${SOURCES[1]} |
abalance rect1=100 order=100 other=${SOURCES[2]}
32 '''

34 def simil(rect1=1,rect2=1):
    return '''
36     similarity other=${SOURCES[1]} rect1=%d rect2=%d
    ''' % (rect1,rect2)
38
def warping(niter,rect1=1,rect2=1,rect3=1):
40     return '''
    warp1 other=${SOURCES[1]} warpin=${SOURCES[2]}
42     warpout=www.rsf
    verb=1 nliter=%d noamp=1 rect1=%d rect2=%d rect3=%d > ${TARGETS[1]} &&
44     warpadd < ${SOURCES[3]} add=www.rsf > $TARGET &&
    rm www.rsf
46     ''' % (niter,rect1,rect2,rect3)

48 def pick(min2,max2,rect1=1,rect2=1,rect3=1,an=0.5):
    return '''
50     window min2=%g max2=%g |
    pick rect1=%d rect2=%d rect3=%d an=%g |
52     window''' % (min2,max2,rect1,rect2,rect3,an)

54 def warpscan(ng,g0,gmax,rect1=1,rect2=1,rect3=1,rect4=1):
    dg = (gmax-g0)/(ng-1)
56     return '''
    warpscan other=${SOURCES[1]} niter=100
58     ng=%d dg=%g g0=%g rect1=%d rect2=%d rect3=%d rect4=%d |
    math output='(1+input)^4' |
60     window''' % (ng,dg,g0,rect1,rect2,rect3,rect4)

62 def warp2gamma(ss):
    return '''
64     math output="input+x1" |
    smoothder %s
66     ''' % ('| math output="2*input-1" ','')[ss]

68 def warp2egamma(ss):
    return '''
70     math output="(input+x1)/x1" %s
    ''' % ('| math output="2*input-1" ','')[ss]
72
def nwarp2(name,       # name prefix
74            pp,ps,      # PP and PS images
              warp,       # initial warp
76            nx,         # number of traces
              tmax,       # maximum time for display
78            tmin=0,     # minimum time for display
              j2=1,       # trace sumsampling
80            trace=None, # seleted trace
              o2=0,       # trace start
82            gmin=1,     # minimum gamma
              gmax=4,     # maximum gamma
84            niter=20,   # warping iterations
              dt=0.004,   # time sampling
86            fmin=0,     # minimum frequency
              fmax=40,    # maximum frequency
88            frect=12,   # frequency smoothing
```

```
            frame1=10,  # time frame
            ng=101,     # number of gammas
            g0=0.96,    # first gamma
            pmin=0,     # minimum gamma for picking
            pmax=2,     # maximum gamma for picking
            an=0.5,     # anisotropy for picking
            rect1=50,   # vertical smoothing
            rect2=50,   # lateral smoothing
            iter=2,     # number of iterations
            ss=0,       # PP-PS (0) or PP-SS (1)
            inter=1,    # interleaving
            clip=6      # display clip
            ):


    if version.old_version():
        return # think how to do it better

    interg = '''
    pad n2=%d | put n2=%d n3=%d | stack
    ''' % ((nx/inter+1)*inter,inter,nx/inter+1)
    inter = 2*inter
    interw = '''
    pad n2=%d | put n2=%d n3=%d | stack
    ''' % ((nx/inter+1)*inter,inter,nx/inter+1)

    if trace:
        for case in (pp,ps,warp):
            Flow(case+'1',case,'window n2=1 min2=%d' % trace)
        warp1(name+'t',pp+'1',ps+'1',warp+'1',tmax,tmin,
              gmin,gmax,niter,
              dt,fmin,fmax,frect,ng,g0,pmin,pmax,an,rect1,iter,ss)
    else:
        trace=10

    def plot(title):
        return '''
        window min1=%g max1=%g |
        grey title="%s" label1=Time unit1=s clip=%g
        label2=Trace unit2=
        ''' % (tmin,tmax,title,clip)

    vplot = plot('Vp/Vs') + '''
    clip=%g scalebar=y color=j bias=%g minval=%g maxval=%g
    ''' % (0.5*(gmax-gmin),0.5*(gmin+gmax),gmin,gmax)

    balance = '''
    nsmooth1 rect=${SOURCES[1]} |
    abalance rect1=%d rect2=%d order=100 other=${SOURCES[2]}
    ''' % (rect1,rect2)

    ifreq = 'iphase rect1=%d rect2=%d order=100' % (2*rect1,2*rect2)

    def freqplot(title):
        return '''
        scale scale dscale=%g |
        %s clip=%g bias=%g color=j scalebar=y barlabel="Frequency (Hz)"
        ''' % (0.5/(math.pi*dt),plot(title),(fmax-fmin)*0.25,(fmax+fmin)*0.5)

    def specplot(title):
        return '''
```

```
        cat axis=2 ${SOURCES[1]} |
        graph title="%s" max1=%g label1="Frequency (Hz)"
        dash=0,1 plotfat=7 label2=
        ''' % (title,4*fmax)

    def giplot(title):
        return '''
        interleave axis=2 ${SOURCES[1]} |
        window min1=%g max1=%g | scale axis=1 |
        grey title="Interleaved (%s)" label1=Time unit1=s
        label2="Inline" unit2=
        ''' % (tmin,tmax,title)

    def wiplot(title):
        return '''
        interleave axis=2 ${SOURCES[1]} |
        window min1=%g max1=%g | scale axis=1 |
        wiggle poly=y transp=y yreverse=y
        title="Interleaved (%s)"
        label1=Time unit1=s label2="In-line"
        ''' % (tmin,tmax,title)

    Plot(pp,plot('PP'))
    Flow(pp+'i',pp,ifreq)
    Plot(pp+'i',freqplot('PP Local Frequency'))

    Result(pp+'line',pp,'Overlay')

    PS = ('PS','SS')[ss]

    Plot(ps,
         '''
         window min1=%g max1=%g |
         grey title="%s" label1=Time unit1=s
         ''' % (tmin,tmax*2.,PS))

    Flow(pp+'s0',pp,'spectra all=y')

    scanplot = '''
    window min1=%g max1=%g |
    byte gainpanel=all allpos=y |
    grey3 frame1=%d frame3=%d frame2=%d color=j flat=n
    label1=Time unit1=s label3="In-line" label2="Relative Gamma"
    wanttitle=n
    ''' % (tmin,tmax,frame1,(trace-o2)/j2,ng/2)

    simplot = '''
    window min1=%g max1=%g |
    grey title="%s" allpos=y
    color=j clip=1
    label1="Time (s)"
    ''' % (tmin,tmax,'%s')

    warpit = warping(niter,200,200)

    for i in range(iter):
        wrp = warp

        ##################
        # INITIAL WARPING
        ##################
```

```
212         def n(s):
                return '%s-%s-%d' % (name,s,i)
214
            psw = n('psw')
216         Flow(psw,[ps,pp,wrp],warp0)
            Plot(psw,plot('Warped ' + PS))
218
            dif = n('dif')
220         Plot(dif,[psw,pp],
                 'add scale=1,-1 ${SOURCES[1]} | ' + plot('Difference'))
222
            gamma = n('gamma')
224         Flow(gamma,wrp,warp2gamma(ss))
            Plot(gamma,vplot)
226
            Result(psw+'all',[pp,psw,dif,gamma],'TwoRows')
228
            psw1 = n('psw1')
230         pp1 = n('pp1')
            Flow(pp1,pp,'window n2=1 f2=286')
232         Flow(psw1,psw,'window n2=1 f2=286')
            ####
234         Result(psw,plot('PSW'))

236         ####

238         ppps = n('ppps')
            Result(ppps,[psw1,pp1],
240                '''
                   add scale=1,-1 ${SOURCES[1]} |
242                cat ${SOURCES[0]} ${SOURCES[1]} axis=2 |
                   dots gaineach=n Xscreenwd=9.225 Xscreenht=5.2 Xyyscale=0.8
244                labels="Difference:SS warped:PP" label1=Time unit1=s
                   title="LTF transform balancing"
246                ''')

248         ####################
            # SPECTRAL BALANCING
250         ####################

252         si = n('si')
            Flow(si,psw,ifreq)
254         Plot(si,freqplot(PS + ' Local Frequency'))
            nc = 1
256         radius2(pp,psw,
                    niter=5,
258                 c=[0.7,0.45,0.35,0.35,0.5],
                    bias=-20, clip=30,
260                 rect1=rect1, rect2=rect2,
                    theor=False, initial=1,
262                 minval=-20, maxval=10,
                    titlehigh='PP', titlelow='PSW',
264                 it=i )

266         psws = n('psws')
            pps  = n('pps')
268         Flow(psws,'low-smooth%d%i'%(nc,i),'cp')
            Flow(pps,'high-smooth%d%i'%(nc,i),'cp')
270
            # balance amplitudes
```

```
272         pswsb = n('pswsb')
            Flow(pswsb,[psws,pps],
274              'abalance other=${SOURCES[1]} rect1=%i rect2=%i order=100'%(50,50))

276         Result(pps,plot("PP smoothed"))
            Result(psws,plot("PS smoothed "))
278         Result(pswsb,plot("PS smoothed balanced"))


280
            ##############
282         ## GAMMA SCAN
            ##############
284
            g1 = 2-g0
286         warpscan2 = warpscan(ng,g0,g1,rect1,1,int(0.5+rect2/j2))

288         sc = n('sc')
            sr = n('sr')
290         pr = n('pr')

292         Flow(pr, pps,'cp')
            Flow(sr, pswsb,'cp')
294
            Flow(sr+'2',sr,'window j2=%d' % j2)
296         Flow(pr+'2',pr,'window j2=%d' % j2)

298         Flow(sc,[sr+'2',pr+'2'],warpscan2)
            Result(sc,scanplot)
300
            pk = n('pk')
302
            if i==0:
304             Flow(pk+'0',sc,pick(max(pmin,g0),min(pmax,g1),
                                    rect1,4*rect2/j2,an=an))
306         else:
                Flow(pk+'0',sc,pick(g0,g1,rect1,4*rect2/j2,an=an))
308
            Flow(pk,pk+'0',
310              '''
                 transp memsize=500 |
312              spline n1=%d d1=1 o1=%g |
                 transp memsize=500  |
314              math output="(input-1)*x1"
                 ''' % (nx,o2))
316
            ###################
318         if i == 0:
                in0 = n('in0')
320             Flow(pr+'in0',pr,interg)
                Flow(sr+'in0',sr,interg)
322             Plot(in0,    [pr+'in0',sr+'in0'],giplot('Before'))

324             Flow(pr+'in0w',pr,interw)
                Flow(sr+'in0w',sr,interw)
326             Plot(in0+'w',[pr+'in0w',sr+'in0w'],wiplot('Before'))

328             Result(in0,in0,'Overlay')
                Result(in0+'w',in0+'w','Overlay')
330         ##################

332         #########
```

```
        # WARPING
        #########

        warp = n('wrp')
        Flow([warp,psw+'2'],[sr,pr,pk,wrp],warpit,stdout=-1)
        #Flow([warp,psw+'2'],[psw,pr,pk,wrp],warpit,stdout=-1)
        Plot(psw+'2',plot('Warped ' + PS))

        dif = n('dif2')
        Plot(dif,[psw+'2',pr],
             'add scale=1,-1 ${SOURCES[1]} | ' + plot('Difference'))

        gamma = n('gamma2')
        Flow(gamma,warp,warp2gamma(ss))
        Plot(gamma,vplot)

        if i == iter-1:
            in1 = n('in1')
            Flow(pr+'in1',pr,interg)
            Flow(psw+'2in1',psw+'2',interg)
            Plot(in1,[pr+'in1',psw+'2in1'],giplot('After'))
            Flow(pr+'in1w',pr,interw)
            Flow(psw+'2in1w',psw+'2',interw)
            Plot(in1+'w',[pr+'in1w',psw+'2in1w'],wiplot('After'))
            Result(in1,in1,'Overlay')
            Result(in1+'w',in1+'w','Overlay')

            sim1 = n('sim1')
            Flow(sim1,[pr,psw+'2'],simil(rect1,rect2))
            Result(sim1,simplot % 'After')

            Flow(psw+'1',[ps,pp,warp],warp0)
            Result(psw+'1',plot('Warped ' + PS))

            rt = n('rt')
            Flow(psw+'i',psw+'1',ifreq)
            Flow(rt,psw+'i',
                 '''
                 math output="sqrt(1+12*(1/input^2-1/%g^2))" |
                 dd type=float
                 ''' % (fmax*2*math.pi*dt))

            dl = n('dl')
            Flow(dl,[psw+'1',rt],
                 '''
                 dd type=float | deblur rect=${SOURCES[1]}
                 verb=y niter=100 eps=0.04 nliter=1
                 ''')
            Result(dl,
                   '''
                   window min1=%g max1=%g |
                   grey title="Deblurred %s" label1="Time (s)"
                   ''' % (tmin,tmax,PS))

            Flow('e'+gamma,warp,warp2egamma(ss))
            Result(gamma,'e'+gamma,vplot)

        g0 = (g0+1)*0.5


def warp1(name,      # name prefix
```

```
394           pp,ps,      # PP and PS images
              warp,       # initial warp
396           tmax,       # maximum time for display
              tmin=0,     # minimum time for display
398           gmin=1,     # minimum gamma
              gmax=4,     # maximum gamma
400           niter=20,   # warping iterations
              dt=0.004,   # time sampling
402           fmin=0,     # minimum frequency
              fmax=40,    # maximum frequency
404           frect=12,   # frequency smoothing
              ng=101,     # number of gammas
406           g0=0.96,    # first gamma
              pmin=0,     # minimum gamma for picking
408           pmax=2,     # maximum gamma for picking
              an=0.5,     # anisotropy for picking
410           rect1=50,   # vertical smoothing
              iter=2,     # number of iterations
412           ss=0
              ):
414
    if version.old_version():
416       return # think how to do it better

418   graph = '''
      graph wanttitle=n min2=%g max2=%g min1=%g max1=%g
420   wherexlabel=t wheretitle=b crowd=0.8 label2="Vp/Vs"
      ''' % (gmin,gmax,tmin,tmax)
422
      dplot ='''
424   add scale=1,-1 ${SOURCES[1]} |
      cat ${SOURCES[0]} ${SOURCES[1]} axis=2 |
426   window min1=%g max1=%g |
      dots gaineach=0
428   labels="Difference:PS warped:PP" label1=Time unit1=s
      ''' % (tmin,tmax)
430
      def iphase(title):
432       return '''
          cat axis=2 ${SOURCES[1]} |
434       scale dscale=%g |
          graph title="Local Frequency (%s)" label1="Time (s)"
436       min2=%g max2=%g min1=%g max1=%g
          dash=0,1 label2="Frequency (Hz)"
438       ''' % (0.5/(math.pi*dt),title,fmin,fmax,tmin,tmax)

440   warpit = warping(niter,200)

442   for i in range(iter):
          ##################
444       # INITIAL WARPING
          ##################
446       wrp = warp

448       def showpick(case):
              return '''
450           graph transp=y min2=%g max2=%g min1=%g max1=%g
              yreverse=y plotcol=%d plotfat=%d
452           wantaxis=n wanttitle=n pad=n
              ''' % (g0,g1,tmin,tmax,(7,0)[case],(5,1)[case])
454
```

```
        def n(s):
            return '%s-%s-%d' % (name,s,i)

        gamma = n('gamma')
        Flow(gamma,wrp,warp2gamma(ss));
        Plot(gamma,graph)

        psw = n('psw')
        Flow(psw,[ps,pp,wrp],warp0)
        Plot(psw,[psw,pp],dplot)

        Result(psw,[gamma,psw],'OverUnderAniso')
```

Listing 7: chapter-locfreq/vecta/radius2.py

```
from rsf.proj import *

def radius2(high, low,                # initial high-resolution and legacy images
            niter,                    # number of corrections
            c,                        # step length for radius corrections. Can
                                      # be type int or float for constant c
                                      # or type array for changing c.
            bias=-15, clip=30,        # bias and clip for display
            rect1=40, rect2=80,       # radius for local frequency calculation
            maxrad=1000,              # maximum allowed radius
            theor=True,               # use theoretical smoothing radius
            scale=9,                  # scale for theoretical smoothing radius
            initial=10,               # initial value for contant smoothing radius
            minval=0, maxval=25,      # minval and maxval for freqdif-filt display
            titlehigh="Hires",        # high title
            titlelow="Legacy",        # low title
            it=0):                    # correction number (from newarp.py)

    if type(c) is float or type(c) is int:
        c = [c]*niter

    def seisplot(name):
        return 'grey title="%s"'%name

    locfreq = '''iphase order=10 rect1=%d rect2=%d hertz=y complex=y |
                 put label="Frequency" unit=Hz'''%(rect1,rect2)

    def locfreqplot(name):
        return 'grey mean=y color=j scalebar=y title="%s" '%name

    freqdif = 'add scale=-1,1 ${SOURCES[1]} | put label=Frequency'

    def freqdifplot(num):
        return '''grey allpos=y color=j scalebar=y mean=y
                  title="Difference in Local Frequencies %d"
                  clip=%d bias=%d minval=%d maxval=%d''' %(num,22,-15,-15,8)

    specplot = '''cat axis=2 ${SOURCES[1]} |
                  scale axis=1 | window max1=180 |
                  graph title="Normalized Spectra" label2="Amplitude" unit2=""'''

    def rectplot(name):
        return '''grey color=j mean=y title="%s" scalebar=y barlabel=Radius
                  barunit=samples'''%name

    smooth = 'nsmooth1 rect=${SOURCES[1]}'
```

```
    Flow('high-freq%i'%it,high,locfreq)
    Result('high-freq%i'%it,locfreqplot('%s Local Frequency'%titlehigh))

    Flow('low-freq%i'%it,low,locfreq)
    Result('low-freq%i'%it,locfreqplot('%s Local Frequency'%titlelow))

    Flow('freqdif%i'%it,'low-freq%i high-freq%i'%(it,it),freqdif)

    Result('freqdif%i'%it,freqdifplot(-1))

    # initial smoothing radius
    if (theor):
        from math import pi
        Flow('rect0%i'%it,'low-freq%i high-freq%i'%(it,it),
        '''math f1=${SOURCES[1]}
        output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" '''%(scale,2*pi*0.001))
    else:
        Flow('rect0%i'%it,'low-freq%i'%it,'math output=%f'%initial)

    Result('rect0%i'%it,rectplot("Smoothing Radius 0"))

    Flow('high-smooth0%i'%it,'%s rect0%i'% (high,it),smooth)
    Result('high-smooth0%i'%it, seisplot("%s Smooth 0"%titlehigh))

    Flow('high-spec%i'%it,high,'spectra all=y')
    Flow('low-spec%i'%it,low,'spectra all=y')
    Flow('high-smooth-spec0%i'%it,'high-smooth0%i'%it,'spectra all=y')
    Result('nspectra%i'%it,'high-spec%i low-spec%i'%(it,it),specplot)
    Result('high-smooth-spec0%i'%it,
           'high-smooth-spec0%i low-spec%i'%(it,it),specplot)

    Flow('high-smooth-freq0%i'%it,'high-smooth0%i'%it,locfreq)
    Result('high-smooth-freq0%i'%it,
            locfreqplot("%s Local Frequency Smoothed %d" %(titlehigh,0)))

    Flow('freqdif-filt0%i'%it,'low-freq%i high-smooth-freq0%i'%(it,it),freqdif)
    Result('freqdif-filt0%i'%it,freqdifplot(0))

    prog=Program('radius2.c')
    for i in range(1, niter+1):
        j = i-1
        Flow('rect%d%i rect-low%d%i rect-high%d%i'%(i,it,i,it,i,it),
             'rect%d%i freqdif-filt%d%i %s'%(j,it,j,it,prog[0]),
             '''./${SOURCES[2]} freq=${SOURCES[1]} low=${TARGETS[1]}
                high=${TARGETS[2]} c=%f'''%c[j])
        Result('rect%d%i'%(i,it),rectplot("Smoothing Radius %d")%i)
        Result('rect-low%d%i'%(i,it),rectplot("Smoothing Radius %d low")%i)
        Result('rect-high%d%i'%(i,it),rectplot("Smoothing Radius %d high")%i)

        Flow('high-smooth%d%i'%(i,it),'%s rect-high%d%i'%(high,i,it),smooth)
        Result('high-smooth%d%i'%(i,it), seisplot('%s Smooth %d'%(titlehigh,i)))

        Flow('low-smooth%d%i'%(i,it),'%s rect-low%d%i'%(low,i,it),smooth)
        Result('low-smooth%d%i'%(i,it), seisplot('%s Smooth %d'%(titlelow,i)))

        Flow('high-smooth-spec%d%i'%(i,it),'high-smooth%d%i'%(i,it),
             'spectra all=y')
        Flow('low-smooth-spec%d%i'%(i,it),'low-smooth%d%i'%(i,it),
             'spectra all=y')
        Result('high-smooth-spec%d%i'%(i,it),
```

```
                'high-smooth-spec%d%i low-smooth-spec%d%i'%(i,it,i,it),specplot)

        Flow('high-smooth-freq%d%i'%(i,it),'high-smooth%d%i'%(i,it),locfreq)
        Result('high-smooth-freq%d%i'%(i,it),
                locfreqplot('%s Local Frequency Smoothed %d'%(titlehigh,i)))

        Flow('low-smooth-freq%d%i'%(i,it),'low-smooth%d%i'%(i,it),locfreq)
        Result('low-smooth-freq%d%i'%(i,it),
                locfreqplot('%s Local Frequency Smoothed %d'%(titlelow,i)))

        Flow('freqdif-filt%d%i'%(i,it),
             'low-smooth-freq%d%i high-smooth-freq%d%i'%(i,it,i,it),freqdif)
        Result('freqdif-filt%d%i'%(i,it),freqdifplot(i))
```

Listing 8: chapter-locfreq/vecta/radius2.c

```
/* smoothing radii */

#include <rsf.h>

int main (int argc, char* argv[])
{
    int n1, n1f, n2, n2f, i, n12, n12f;
    float *rect, *fr, maxrad, c, *rad, *rad_low, *rad_high;
    sf_file in, out, freq, low, high;

    sf_init (argc,argv);

    in = sf_input("in");
    freq = sf_input("freq");
    out = sf_output("out");
    low = sf_output("low");
    high = sf_output("high");

    if (!sf_histint(in,"n1",&n1)) sf_error("No n1= in input.");
    if (!sf_histint(freq,"n1",&n1f)) sf_error("No n1= in frequency difference.");

    n2 = sf_leftsize(in,1);
    n2f = sf_leftsize(freq,1);

    n12 = n1*n2;
    n12f = n1f*n2f;

    if (n1 != n1f) sf_error("Need matching n1");
    if (n2 != n2f) sf_error("Need matching n2");

    if (!sf_getfloat("c",&c)) c=1.;
    if (!sf_getfloat("maxrad",&maxrad)) maxrad=1000.;

    rect = sf_floatalloc(n12);
    sf_floatread(rect,n12,in);

    fr = sf_floatalloc(n12f);
    sf_floatread(fr,n12,freq);

    rad = sf_floatalloc(n12);
    rad_low = sf_floatalloc(n12);
    rad_high = sf_floatalloc(n12);

    /* constraint conditions: [-maxrad, -1] U [1, maxrad] */
    for (i=0; i < n12; i++) {
```

```
47          /* update radius */
        rad[i] = rect[i]+c*fr[i];
49
          /* set maximum allowed radius */
51          if (rad[i] > maxrad) rad[i] = maxrad;

53          /* low radius */
            if (rad[i] < 0){
55              rad_high[i] = 1;
                rad_low[i] = -1 * rad[i];
57              if (rad_low[i] < 1) rad_low[i] = 1;

59          /* high radius */
            }else{
61              rad_low[i] = 1;
                rad_high[i] = rad[i];
63              if (rad_high[i] < 1) rad_high[i] = 1;
            }
65      }

67      sf_floatwrite(rad,n12,out);
        sf_floatwrite(rad_low,n12,low);
69      sf_floatwrite(rad_high,n12,high);
        exit(0);
71  }
```

Listing 9: chapter-locfreq/convergence/plot.py

```
#!/usr/bin/python
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.font_manager as font_manager

fontpath = '/home/sarah/.fonts/cmunss.ttf'

prop = font_manager.FontProperties(fname=fontpath)
mpl.rcParams['font.family'] = prop.get_name()

# first plot
niter = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
niter2 = [0, 1, 2, 3]
norm1= [32475928, 25392492, 19320528, 14462173, 11263933, 10371235, 10477612,
        10634981, 10760000, 10680567, 10628854, 10559621, 10391489]
norm2= [32475928, 23425132, 28284798, 26716290]
norm3= [32475928, 27446940, 23236344, 20663992, 18737512, 17107040, 15649435,
        14493308, 13569843, 12907267, 12425467,12109981, 11745056]
norm1[:] = [x / 10371235.0 for x in norm1]
norm2[:] = [x / 10371235.0 for x in norm2]
norm3[:] = [x / 10371235.0 for x in norm3]

plt.figure(figsize=(9.60,5.40), dpi=1000)
l1 = plt.scatter(niter,norm3)
l2 = plt.scatter(niter2,norm2, marker='x', color='g')
l3 = plt.scatter(niter,norm1, marker='*', color='y')

plt.legend((l1, l2, l3),('Too small c', 'Too large c','Good c'))
plt.plot([-1, 14], [1, 1],'r--')
plt.xlim(-0.2, 12.2)
plt.ylim(0, 3.5)
plt.xlabel("Iterations")
```

```
plt.ylabel(r'$\Vert\mathbf{F}(S_R * d_h) -  \mathbf{F}(d_l) \Vert$')

plt.savefig('Fig/scalar.pdf', format='pdf', bbox_inches='tight')


# second plot
niter = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
norm1= [47484892, 33515692, 23253388, 16376629, 11599820, 10069987, 10116965,
        10258426, 10321123, 10282613, 10289243, 10398856, 10446601]
norm2= [60672360, 33617148, 20220940, 13173313, 10842506, 10398470, 10210745,
        10296503, 10313730, 10298821, 10204044, 10263224, 10321556]
norm3= [32475928, 25392492, 19320528, 14462173, 11263933, 10371235, 10477612,
        10634981, 10760000, 10680567, 10628854, 10559621, 10391489]
norm1[:] = [x / 10069987.0 for x in norm1]
norm2[:] = [x / 10069987.0 for x in norm2]
norm3[:] = [x / 10069987.0 for x in norm3]

plt.figure(figsize=(9.60,5.40), dpi=1000)
l1 = plt.scatter(niter,norm1)
l2 = plt.scatter(niter,norm2, marker='x', color='g')
l3 = plt.scatter(niter,norm3, marker='*', color='y')

plt.legend((l1, l2, l3),('Initial = constant 10','Initial = constant 1',
    'Initial = theoretical radius'))
plt.plot([-1, 14], [1, 1],'r--')
plt.xlim(-0.2, 12.2)
plt.ylim(0, 6.5)
plt.xlabel("Iterations")
plt.ylabel(r'$\Vert\mathbf{F}(S_R * d_h) -  \mathbf{F}(d_l) \Vert$')

plt.savefig('Fig/all.pdf', format='pdf', bbox_inches='tight')
```

Chapter 4

Listing 10: chapter-merge/apache/SConstruct

```
from rsf.proj import *
from rsf.recipes.beg import server

data = {'legacy':'i455_legacy3d_pstm_4ms.sgy',
        'hires':'i455_sc2dpoststkmig_1ms.sgy'}

for key in data.keys():
    Fetch(data[key],'apache',server)

    # Convert from SEGY format to Madgascar RSF format
    Flow([key,'t'+key,key+'.asc',key+'.bin'],data[key],
             '''
             segyread tfile=${TARGETS[1]} hfile=${TARGETS[2]}
             bfile=${TARGETS[3]} | put label2=Cross-line o2=3205 d2=1
             ''')

    # Display
    Result(key,
           '''
           grey color=seismic title="%s Image"
           ''' % key.capitalize())

```

```
    # Look at spectra
    spectrum = key + '-spec'
    Flow(spectrum,key,'spectra all=y')
    Result(spectrum,'graph title="%s Spectrum" ' % key.capitalize())

Result('hires2',
       '''
       grey color=seismic title="Hires Image"
       ''')

# Sample both at 2 ms
Flow('hires2','hires','bandpass fhi=250 | window j1=2')
Flow('legacy2','legacy','spline n1=2001 d1=0.002 o1=0')
Flow('legacy4','legacy','spline n1=4001 d1=0.001 o1=0')

for key in data.keys():
    spectrum = key + '-spec2'
    Flow(spectrum,key+'2','spectra all=y')

Result('spectra','hires-spec2 legacy-spec2',
       'cat axis=2 ${SOURCES[1]} | graph title=Spectra')

Result('nspectra','hires-spec2 legacy-spec2',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=120 |
       graph title="Normalized Spectra" label2="Amplitude"
       ''')


# BANDPASS FILTERING
lof = 18  # other value: 21
hif = 38  # other value: 35

Flow('hires3', 'hires2', 'bandpass fhi='+str(hif)+' flo='+str(lof))
Flow('legacy3', 'legacy2', 'bandpass fhi='+str(hif)+' flo='+str(lof))

for key in data.keys():
    filtspec = key + '-spec3'
    Flow(filtspec, key+'3', 'spectra all=y')

Result('filtnspectra','hires-spec3 legacy-spec3',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 |
       graph title="Filtered Normalized Spectra"
       ''')

Result('hires3', 'grey color=seismic title="Inline hires filtered, 455 Dataset"')
Result('legacy3', 'grey color=seismic title="Inline legacy filtered, 455 Dataset"')


# SMOOTHING APPLIED TO HIRES
# Measure local frequency
for key in data.keys():
    freq = key+'-freq'
    Flow(freq,key,'''iphase order=10 rect1=%d rect2=80 hertz=y complex=y |
                     put label=Frequency unit=Hz''' % (40,160)[key=='hires'])
    Result(freq,'''grey mean=y color=j scalebar=y title="%s Local Frequency"
                   minval=10 maxval=75''' % key.capitalize())

# Difference in local frequencies
Flow('freqdif','legacy-freq hires-freq',
```

```
              'remap1 n1=4001 d1=0.001 o1=0 | add scale=-1,1 ${SOURCES[1]}')

Result('freqdif','''grey allpos=y color=j scalebar=y minval=0 maxval=40 clip=40
                    mean=y title="Difference in Local Frequencies" ''')

# Try stationary smoothing
for rect in (5,10,15,20):
    smooth = 'smooth%d' % rect
    Flow(smooth,'hires','''smooth rect1=%d | bandpass fhi=250 | window j1=2 |
                           spectra all=y''' % rect)
    Result(smooth,[smooth,'legacy-spec2'],'''cat axis=2 ${SOURCES[1]} |
           scale axis=1 | window max1=125 | graph title="rect=%d"''' % rect)

# Nonstationary smoothing applied to hires to match with legacy
from math import pi

scale=6.75 # theoretically should be 12

Flow('rect','legacy-freq hires-freq',
     '''remap1 n1=4001 d1=0.001 o1=0 | math f1=${SOURCES[1]}
     output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g"''' % (scale,2*pi*0.001))

Result('rect','''grey color=j mean=y title="Smoothing Radius"
                  scalebar=y barlabel=Radius barunit=samples''')

Flow('hires-smooth','hires rect','nsmooth1 rect=${SOURCES[1]} | window j1=4')
Flow('hires-smooth-spec','hires-smooth','spectra all=y')
Result('hires-smooth-spec','hires-smooth-spec legacy-spec','''cat axis=2
        ${SOURCES[1]} | scale axis=1 | window max1=120 | graph
        title="Normalized Spectra after Smoothing" label2="Amplitude"''')
Result('hires-smooth', 'grey color=seismic title="Hires Smooth"')

# Difference in local frequencies with nonstationary smoothing applied to hires
Flow('hires-smooth-freq','hires-smooth','''iphase order=10 rect1=40 rect2=80
      hertz=y complex=y | put label=Frequency unit=Hz''')
Result('hires-smooth-freq',
      'grey mean=y color=j scalebar=y title="Hires Local Frequency Smoothed"')

Flow('freqdif-filt','legacy-freq hires-smooth-freq','add scale=-1,1 ${SOURCES[1]}')

Result('freqdif-filt','''grey allpos=y color=j scalebar=y minval=0 maxval=40
                         clip=40 mean=y title="Difference after Smoothing"''')


# BALANCE AMPLITUDES
Flow('hires-balanced','hires-smooth legacy',
     'abalance other=${SOURCES[1]} rect1=80 rect2=160')

Flow('hires-balanced-reverse','hires-smooth legacy',
     'abalance other=${SOURCES[1]} rect1=80 rect2=160 reverse=n')


# SELECT FIRST TRACE
# First trace = cross-line 3700
for case in ('legacy','hires-balanced','hires-balanced-reverse'):
    trace = case + '-trace'
    Flow(trace,case,'window n2=1 min2=3700')

# take the difference
Flow('trace-diff','legacy-trace hires-balanced-trace','add ${SOURCES[1]} scale=1,-1')

```

```
Flow('trace-diff-reverse','legacy-trace hires-balanced-reverse-trace',
     'add ${SOURCES[1]} scale=1,-1')

Result('traces','legacy-trace hires-balanced-trace trace-diff',''cat axis=2
     ${SOURCES[1:3]} | dots gaineach=n labels=Legacy:Hires:Difference yreverse=y'')

Result('traces-reverse',''legacy-trace hires-balanced-reverse-trace
        trace-diff-reverse'', ''cat axis=2 ${SOURCES[1:3]} |
        dots gaineach=n labels=Legacy:Hires:Difference yreverse=y'')

Result('traces-reverse-diff','trace-diff trace-diff-reverse',''cat axis=2
        ${SOURCES[1]} | dots gaineach=n labels=reverse=y:reverse=n yreverse=y'')


# BALANCE PHASE OF FIRST TRACE
rotates = []
for angle in range(-180,181,5):
    rotate = 'legacy-rotate%d' % angle
    Flow(rotate,'legacy-trace','envelope hilb=y phase=%d' % angle)
    rotates.append(rotate)

Flow('rotates',rotates,
     'cat axis=2 ${SOURCES[1:%d]} | put o2=-180 d2=5' % len(rotates))

Flow('phasematch','hires-balanced-trace rotates',
     '''
     spray axis=2 n=73 o=-180 d=5 label=Angle unit=degree |
     similarity other=${SOURCES[1]} rect1=160
     ''')

Plot('phasematch','grey color=j allpos=y title="Phase Similarity" ')

Flow('phasepick','phasematch','pick rect1=20 vel0=0')

Plot('phasepick',''graph plotcol=7 yreverse=y transp=y min2=-180 max2=180 pad=n
      wanttitle=n wantaxis=n'')

Result('phasematch','phasematch phasepick','Overlay')

Flow('phaseslice','rotates phasepick','slice pick=${SOURCES[1]}')

Flow('phase-diff','phaseslice hires-balanced-trace','add ${SOURCES[1]} scale=1,-1')

Result('phase-traces','phaseslice hires-balanced-trace phase-diff',
     'cat axis=2 ${SOURCES[1:3]} | dots labels=Legacy:Hires:Difference yreverse=y')


# TIME SHIFT OF FIRST TRACE
# Scanning different time shifts
Flow('warpscan','hires-balanced-trace legacy-trace',
     '''
     warpscan shift=y ng=101 g0=-0.05 dg=0.001
     other=${SOURCES[1]} rect1=20
     ''')

Flow('warppick','warpscan',
     'scale axis=2 | pick rect1=10 vel0=0.02 an=0.1')

Plot('warpscan',
     '''
     grey allpos=y color=j title="Shift Scan"
```

```
     label2=Shift unit2=s
     ''')

# This is what was removed
Plot('warppick',
     '''
     graph plotfat=3 plotcol=7 wanttitle=n wantaxis=n
     min2=-0.05 max2=0.05 pad=n yreverse=y transp=y
     ''')

Result('warpscan','warpscan warppick','Overlay')

# Apply picked shift
Flow('warp','hires-balanced-trace legacy-trace warppick',
     'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')

Flow('warp-diff','legacy-trace warp','add ${SOURCES[1]} scale=1,-1')

Result('wtraces','legacy-trace warp warp-diff','''cat axis=2 ${SOURCES[1:3]} |
        dots gaineach=n labels=Legacy:Warped:Difference yreverse=y''')


# TIME SHIFT OF PHASE ADJUSTED FIRST TRACE
Flow('warpscanr','hires-balanced-trace phaseslice',
     '''
     warpscan shift=y ng=101 g0=-0.05 dg=0.001
     other=${SOURCES[1]} rect1=20
     ''')

Flow('warppickr','warpscanr',
     'scale axis=2 | pick rect1=10 vel0=0.02 an=0.1')

Plot('warpscanr',
       '''
       grey allpos=y color=j title="Shift Scan"
       label2=Shift unit2=s
       ''')

# This is what was removed
Plot('warppickr',
     '''
     graph plotfat=3 plotcol=7 wanttitle=n wantaxis=n
     min2=-0.05 max2=0.05 pad=n yreverse=y transp=y
     ''')

Result('warpscanr','warpscanr warppickr','Overlay')

# Apply picked shift
Flow('warpr','hires-balanced-trace phaseslice warppickr',
     'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')

Flow('warp-diffr','phaseslice warpr','add ${SOURCES[1]} scale=1,-1')

Result('phase-wtraces','phaseslice warpr warp-diffr','''cat axis=2 ${SOURCES[1:3]} |
        dots gaineach=n labels=Legacy:Warped:Difference yreverse=y''')

# RUN WARPSCAN ON THE WHOLE IMAGE
Flow('warpscan3','hires-balanced legacy',
     '''
     warpscan shift=y ng=51 g0=0 dg=0.001
     other=${SOURCES[1]} rect1=20 rect3=40
```

```
     ''')
268
Result('warpscan3','''byte gainpanel=all allpos=y bar=bar.rsf | transp plane=23 |
270        grey3 color=j frame1=500 frame2=1000 frame3=25 title="Local Similarity Scan"
        label3="Time Shift" unit3=s scalebar=y barlabel=Similarity''')
272
Flow('warppick3','warpscan3',
274     'scale axis=2 | pick rect1=10 rect2=20 vel0=0.02 an=0.1')

276 Result('warppick3','''grey color=j allpos=y title="Estimated Time Shift"
                      scalebar=y barlabel=Time barunit=s''')
278
Flow('diff0','hires-balanced legacy','add scale=1,-1 ${SOURCES[1]}')
280
# Apply picked shift
282 Flow('warp3','hires-balanced legacy warppick3',
     'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')
284
Flow('diff1','warp3 legacy','add scale=1,-1 ${SOURCES[1]}')
286
Result('diff0','window max1=2 | grey title="Difference before warping" clip=36.5')
288 Result('diff1','window max1=2 | grey title="Difference after warping" clip=36.5')


290
# SHIFT WITHOUT BALANCING AMPLITUDES
292 Flow('warp-hires','warppick3','remap1 n1=4001 d1=0.001 o1=0')

294 Flow('hires-warp','hires warp-hires',
     'warp1 other=${SOURCES[0]} warpin=${SOURCES[1]} nliter=0')
296
Flow('hires-warp-freq','hires-warp','''iphase order=10 rect1=160 rect2=80 hertz=y
298                                    complex=y | put label=Frequency unit=Hz''')

300 Result('hires-warp-freq','''grey mean=y color=j scalebar=y
        title="Warped Hires Local Frequency"''')
302
Flow('rect2','legacy-freq hires-warp-freq','''remap1 n1=4001 d1=0.001 o1=0 |
304          math f1=${SOURCES[1]}
             output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g"''' % (scale,2*pi*0.001))
306
Result('rect2','''grey color=j mean=y title="Smoothing Radius" scalebar=y
308                barlabel=Radius barunit=samples''')

310 Flow('hires-warp-smooth','hires-warp rect2','nsmooth1 rect=${SOURCES[1]}')


312


314 # CREATE THE MERGED IMAGE
Flow('hires-warp-balance lweight','hires-warp-smooth legacy4',
316       'abalance weight=${TARGETS[1]} other=${SOURCES[1]} rect1=320 rect2=160')

318 Flow('hires-warp-balance-reverse lweight-reverse','hires-warp-smooth legacy4',
   'abalance weight=${TARGETS[1]} other=${SOURCES[1]} rect1=320 rect2=160 reverse=n')
320
Result('lweight', '''window min1=0.25 |
322                    grey color=j scalebar=y title="Legacy Weight" mean=y''')

324 Result('lweight-reverse','''window min1=0.25 |
                           grey color=j scalebar=y title="Legacy Weight" mean=y''')
326
```

```
Flow('lweight2','lweight','cut max1=0.25 | clip2 lower=0 | scale dscale=0.5')

Flow('hires-warp-diff','hires-warp-balance legacy4',
     'add ${SOURCES[1]} scale=-1,1 | pad beg3=1')

Flow('hires-warp-diff-reverse','hires-warp-balance-reverse legacy4',
     'add ${SOURCES[1]} scale=-1,1 | pad beg3=1')

# function for weight
f3 = "5*erfc(x1)"

# fix top to be hires
Flow('weight2','lweight',
     'math output=1 | cut min1=0.25 | smooth rect1=50 repeat=4 | add $SOURCE')

Flow('weight3','hires-warp weight2',
     'math output=1 | cut min1=0.25 | add ${SOURCES[1]}')

Flow('weight5','lweight','math output=\"'+f3+'\"| cut max1=0.25')
Flow('hweight','lweight weight5',
     'math output=5 | cut min1=0.25 | add ${SOURCES[1]} | scale dscale=1')
Result('hweight', 'grey color=j scalebar=y title="Hires Weight"')
Result('lweight2', 'grey color=j scalebar=y title="Legacy Weight" mean=y')
Flow('lweight3', 'hweight', 'math output=1')

# right-hand side
Flow('hires-legacy','hires-warp legacy4 hweight',
     'mul ${SOURCES[2]} | cat axis=3 ${SOURCES[1]}')

Flow('merge','hires-legacy legacy4 rect2 hweight lweight3',
     '''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}''')

Result('merge','grey color=seismic title="Merged Image" ')

Flow('merge3','hires-warp-diff hires-warp rect2 hweight lweight',
     '''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]} | add ${SOURCES[1]}''')

Result('merge3','grey color=seismic title="Merged Image" ')

Flow('merge1','hires-warp-diff hires-warp rect2 hweight lweight2',
     '''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]} ''')

Flow('hweight-reverse','hweight','cp')

Flow('merge1-reverse','''hires-warp-diff-reverse hires-warp rect2 hweight-reverse
     lweight-reverse''','''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]} ''')

Flow('merge2','hires-warp merge1','add ${SOURCES[1]}')

Flow('merge2-reverse','hires-warp merge1-reverse','add ${SOURCES[1]}')

Result('merge1',
       'grey color=seismic title="Difference between Merged and Hires" clip=3.7011')

Result('merge2','grey color=seismic title="Merged Image" ')

# Difference between high-resolution and merged image
```

```
Result('merge1-reverse',''bandpass fhi=250 | window j1=3 |
       grey color=seismic title="Difference between Merged and Hires" clip=3.7011''')

# Final merged image
Result('merge2-reverse',
       'bandpass fhi=250 | window j1=3 | grey color=seismic title="Merged Image"')


# Display the spectra
Flow('hires-smooth4','hires-smooth','spline n1=4001 d1=0.001 o1=0')
Flow('merge-spec4','merge','spectra all=y')
Flow('legacy4-spec4','legacy4','spectra all=y')
Flow('hires-spec4','hires-warp','spectra all=y')
Result('nspectra2','legacy4-spec4 hires-spec4 merge-spec4',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
       graph title="Normalized Spectra" label2="Amplitude"
       ''')

Flow('merge2-spec4','merge2','spectra all=y')

Flow('merge2-spec4-reverse','merge2-reverse','spectra all=y')

Result('nspectra22b','hires-spec4 merge2-spec4',
       '''
       cat axis=2 ${SOURCES[1]} | window max1=120 |
       graph title="Spectra" dash=1,0
       ''')

Result('nspectra22','legacy4-spec4 hires-spec4 merge2-spec4',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
       graph title="Normalized Spectra" dash=2,1,0
       ''')

Result('nspectra22b-reverse','hires-spec4 merge2-spec4-reverse',
       '''
       cat axis=2 ${SOURCES[1]} | window max1=120 |
       graph title="Spectra" dash=1,0
       ''')

# Final Spectra
Result('nspectra22-reverse','hires-spec4 legacy4-spec4 merge2-spec4-reverse',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
       graph title="Normalized Spectra " dash=2,1,0 label2="Amplitude"
       ''')
End()
```

Listing 11: chapter-merge/pcable/SConstruct

```
from rsf.proj import *
from rsf.recipes.beg import server

data = {'legacy':'txla3d_merge_east_galveston_match.sgy',
        'hires':'east_galveston_FNL_MIG.sgy'}

for key in data.keys():
    Fetch(data[key],'gom',server)

    # Convert from SEGY format to Madgascar RSF format
```

```
    Flow([key,'t'+key,key+'.asc',key+'.bin'],data[key],
         '''
         segyread tfile=${TARGETS[1]} hfile=${TARGETS[2]} bfile=${TARGETS[3]}
         ''')
    Result(key+'3',key,'intbin xk=iline yk=xline | put label2=Inline label3=Crossline
     | byte gainpanel=all | grey3 frame1=500 frame2=200 frame3=200 title=%s' % key.
    capitalize())

Flow('legacy2','legacy','intbin xk=iline yk=xline | put label2=Inline label3=
    Crossline | byte gainpanel=all ' )
Result('legacy','legacy2','grey3 frame1=500 frame2=200 frame3=200 title="Legacy"' )

Flow('hires2','hires','intbin xk=iline yk=xline | put label2=Inline label3=Crossline
    | byte gainpanel=all ')
#Result('hires','hires2','grey3 frame1=500 frame2=200 frame3=200 title="Hires"')

# Legacy
Flow('lbin3','tlegacy','intbin3 head=$SOURCE')
Flow('lcdpx','lbin3','headermath output=cdpx | window n2=1')
Flow('lcdpy','lbin3','headermath output=cdpy | window n2=1')

Result('lcdpx',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y title=CDPX
       minval=362681.65 maxval=378226.78 bias=370454.22 clip=7772.57
       barlabel=Distance label1=Cross-line label2=Inline
       title="Legacy CDPX"
       ''')

Result('lcdpy',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y title=CDPY
       minval=3250286.50 maxval=3260368.75 bias=3255327.63 clip=5071.13
       barlabel=Distance label1=Cross-line label2=Inline
       title="Legacy CDPY"
       ''')

Flow('lcdpxf','lcdpx','dd type=float | scale dscale=0.01')
Flow('lcdpyf','lcdpy','dd type=float | scale dscale=0.01')
Flow('lcdp','lcdpxf lcdpyf','cmplx ${SOURCES[1]}')
Result('lcdp','graph title="CDP" label1=x label2=y min1=362681.65 max1=378226.78 min2
    =3250286.50 max2=3260368.75 plotcol=1')

# Hires
Flow('hbin3','thires','intbin3 head=$SOURCE')
Flow('hcdpx','hbin3','headermath output=cdpx | window n2=1')
Flow('hcdpy','hbin3','headermath output=cdpy | window n2=1')

Result('hcdpx',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y title=CDPX
       minval=362681.65 maxval=378226.78 bias=370454.22 clip=7772.57
       barlabel=Distance label1=Cross-line label2=Inline
       title="Hires CDPY"
       ''')

Result('hcdpy',
       '''
```

```
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y title=CDPY
       minval=3250286.50 maxval=3260368.75 bias=3255327.63 clip=5071.13
       barlabel=Distance label1=Cross-line label2=Inline
       title="Hires CDPX"
       ''')

Flow('hcdpxf','hcdpx','dd type=float | scale dscale=0.01')
Flow('hcdpyf','hcdpy','dd type=float | scale dscale=0.01')
Flow('hcdp','hcdpxf hcdpyf','cmplx ${SOURCES[1]}')
Result('hcdp','graph title="CDP" label1=x label2=y min1=362681.65 max1=378226.78 min2
    =3250286.50 max2=3260368.75 plotcol=2')

# Overlay hires and legacy
Result('cdp','Fig/lcdp Fig/hcdp','Overlay')

# Zoom
Result('hcdpzoom','hcdp','graph title="CDP zoom" label1=x label2=y min1=372681.65
    max1=374226.78 min2=3255286.50 max2=3255768.75 symbol=*')
Result('lcdpzoom','lcdp','graph title="CDP zoom" label1=x label2=y min1=372681.65
    max1=374226.78 min2=3255286.50 max2=3255768.75 symbol=* plotcol=567 plotfat=3')
Result('cdpzoom','Fig/hcdpzoom Fig/lcdpzoom','Overlay')


# Rotate hires and legacy
import math

a=111.9
#a=-2.9 #for lined up legacy instead of hires

cs=math.cos(math.radians(a))
sn=math.sin(math.radians(a))
ox=36976140
oy=325545425

# Rotate them both a degrees around (ox, oy)
Flow('tlegacyr','tlegacy','dd type=float | headermath key=unass1 output="(cdpx-%f)*(%
    f)+(cdpy-%f)*(%f)+%f" | headermath key=unass2 output="-(cdpx-%f)*(%f)+(cdpy-%f)
    *(%f)+%f"'%(ox, cs, oy, sn, ox, ox, sn, oy, cs, oy))
Flow('thiresr','thires','dd type=float | headermath key=unass1 output="(cdpx-%f)*(%f)
    +(cdpy-%f)*(%f)+%f" | headermath key=unass2 output="-(cdpx-%f)*(%f)+(cdpy-%f)*(%f
    )+%f"'%(ox, cs, oy, sn, ox, ox, sn, oy, cs, oy))

# View new lcdpx and lcdpy
Flow('tlegacyrint','tlegacyr','dd type=int')
Flow('lbinr','tlegacyrint','intbin3 head=$SOURCE')
Flow('lcdpxr','lbinr','headermath output=unass1 | window n2=1')
Flow('lcdpyr','lbinr','headermath output=unass2 | window n2=1')

Result('lcdpxr',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y
       barlabel=Distance label1=Cross-line label2=Inline
       title="Legacy CDPX"
       ''')

Result('lcdpyr',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y
```

```
       barlabel=Distance label1=Cross-line label2=Inline
       title="Legacy CDPY"
       ''')

# Display cdpx and cdpy on map
Flow('lcdpxfr','lcdpxr','dd type=float | scale dscale=0.01')
Flow('lcdpyfr','lcdpyr','dd type=float | scale dscale=0.01')
Flow('lcdpr','lcdpxfr lcdpyfr','cmplx ${SOURCES[1]}')


# View new lcdpx and lcdpy
Flow('thiresrint','thiresr','dd type=int')
Flow('hbin3r','thiresrint','intbin3 head=$SOURCE')
Flow('hcdpxr','hbin3r','headermath output=unass1 | window n2=1')
Flow('hcdpyr','hbin3r','headermath output=unass2 | window n2=1')

Result('hcdpxr',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y
       barlabel=Distance label1=Cross-line label2=Inline
       title="Hires CDPX"
       ''')

Result('hcdpyr',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y
       barlabel=Distance label1=Cross-line label2=Inline
       title="Hires CDPY"
       ''')

# Display hcdpxr and hcdpyr on map
Flow('hcdpxfr','hcdpxr','dd type=float | scale dscale=0.01')
Flow('hcdpyfr','hcdpyr','dd type=float | scale dscale=0.01')
Flow('hcdpr','hcdpxfr hcdpyfr','cmplx ${SOURCES[1]}')

#Display
min12=361987.24
max12=376783.28
min22=3246055.68
max22=3263662.08

Result('hcdpr2','hcdp','graph title="Hires" label1=x label2=y min1=%d max1=%d min2=%d
     max2=%d' %(min12, max12, min22, max22))
Result('hcdpr','graph title="Hires" label1=x label2=y min1=%d max1=%d min2=%d max2=%d
    ' %(min12, max12, min22, max22))
Result('lcdpr','graph title="Legacy" label1=x label2=y min1=%d max1=%d min2=%d max2=%
    d' %(min12, max12, min22, max22))

Result('hcdprzoom2','hcdpr','graph title="Hires" grid1=y grid2=y g1num=50 g2num=50
    label1=x label2=y min1=368000.00 max1=370000.00 min2=3253000.00 max2=3254000.00
    symbol=*')
Result('hcdprzoom','hcdpr','graph title="Hires" grid1=y grid2=y g1num=10 g2num=10
    label1=x label2=y min1=369000.00 max1=369100.00 min2=3253000.00 max2=3253100.00
    symbol=*')
Result('lcdprzoom','lcdpr','graph title="Legacy" grid1=y grid2=y g1num=50 g2num=50
    label1=x label2=y min1=368000.00 max1=370000.00 min2=3253000.00 max2=3254000.00
    symbol=*')

Result('cdpr','Fig/lcdpr Fig/hcdpr','Overlay')
```

```
Result('cdprzoom','Fig/hcdprzoom Fig/lcdprzoom','Overlay')


# Display masked legacy
# Mask out unneeded legacy values
Flow('lmaskx','tlegacyr','headermath output=unass1 | mask min=36790556 max=37090852 |
     dd type=float | put d1=1 d2=1 o1=0 o2=0')
Flow('lmasky','tlegacyr','headermath output=unass2 | mask min=324757760 max=326225952
     | dd type=float | put d1=1 d2=1 o1=0 o2=0')
Flow('lmask','lmaskx lmasky','math y=${SOURCES[1]} output="input*y" | dd type=int')
Flow('tlegacymask','tlegacyr lmask','headercut mask=${SOURCES[1]}')

Flow('tlegacymaskint','tlegacymask','dd type=int')
Flow('lbin3mask','tlegacymaskint','intbin3 head=$SOURCE')
Flow('lcdpxmask','lbin3mask','headermath output=unass1 | window n2=1')
Flow('lcdpymask','lbin3mask','headermath output=unass2 | window n2=1')

# New windowed lcdpx and lcdpy
Result('lcdpxmask',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y
       minval=367905.66 maxval=370908.52 bias=369407.09 clip=1501.43
       barlabel=Distance label1=Cross-line label2=Inline
       title="Legacy CDPX"
       ''')

Result('lcdpymask',
       '''
       dd type=float | scale dscale=0.01 |
       grey color=j mean=y scalebar=y
       minval=3247577.60 maxval=3262259.52 bias=3254918.56 clip=7340.96
       barlabel=Distance label1=Cross-line label2=Inline
       title="Legacy CDPY"
       ''')

Flow('lcdpxfmask','lcdpxmask','dd type=float | scale dscale=0.01')
Flow('lcdpyfmask','lcdpymask','dd type=float | scale dscale=0.01')
Flow('lcdpmask','lcdpxfmask lcdpyfmask','cmplx ${SOURCES[1]}')

# Maxed legacy
Result('lcdpmask','graph title="Legacy" label1=x label2=y min1=%d max1=%d min2=%d
    max2=%d' %(min12, max12, min22, max22))

# min and max values for zoomed version
mmin1=367800.00
mmax1=mmin1+300
mmin2=3253000
mmax2=mmin2+300

Result('lcdpmaskzoom','lcdpmask','graph title="Legacy" grid1=y grid2=y g1num=50 g2num
    =50 label1=x label2=y min1=%f max1=%f min2=%f max2=%f symbol=x' %(mmin1, mmax1,
    mmin2, mmax2))
Result('hcdpmaskzoom','hcdpr','graph title="Hire" grid1=y grid2=y g1num=50 g2num=50
    label1=x label2=y min1=%f max1=%f min2=%f max2=%f symbol=*' %(mmin1, mmax1, mmin2
    , mmax2))

Result('cdprmask','Fig/lcdpmask Fig/hcdpr','Overlay')
Result('cdpwindowzoom','Fig/hcdpmaskzoom Fig/lcdpmaskzoom','Overlay')
```

```
228
# Windowed legacy data
230 Flow('legacy5mask','legacy lmask','headercut mask=${SOURCES[1]}')
Flow('legacy5','legacy5mask','intbin xk=iline yk=xline | put label2=Inline label3=
    Crossline')
232 Result('legacy5','byte gainpanel=all | grey3 frame1=500 frame2=200 frame3=200 title=
    Legacy')

234
# Rebin legacy and hires based on cdpx' and cdpy'
236 Flow('bin-legacy','legacy tlegacyr','transp | bin head=${SOURCES[1]} xkey=89 ykey=90
    xmin=36790556 xmax=37090852 ymin=324757760 ymax=326225952 nx=67 ny=327')
Result('bin-legacy','transp | put label2=cdpx label1=cdpy unit1=ft unit2=ft | byte
    gainpanel=all | window max3=3 | grey3 frame1=35 frame2=26 frame3=125 title=Legacy
    ')
238 Flow('bin-hires','hires thiresr','transp | bin head=${SOURCES[1]} xkey=89 ykey=90
    xmin=36790556 xmax=37090852 ymin=324757760 ymax=326225952 nx=67 ny=327')
Result('bin-hires','transp | put label2=cdpx label1=cdpy unit1=ft unit2=ft | byte
    gainpanel=all | window max3=3 | grey3 frame1=35 frame2=26 frame3=1000 title=Hires
    ')
240
# transpose back
242 Flow('legacytr','bin-legacy','transp plane=31 | put label3=cdpx label2=cdpy unit3=ft
    unit2=ft')
Result('legacytr','byte gainpanel=all | window max1=3 | grey3 frame3=20 frame2=15
    frame1=125 title=Legacy')
244
# because it takes too long to rebuild
246 #Ignore('hirestr.rsf','bin-hires.rsf')
Flow('hirestr','bin-hires','transp plane=31 | put label3=cdpx label2=cdpy unit3=ft
    unit2=ft')
248 Result('hirestr','byte gainpanel=all | grey3 frame3=20 frame2=15 frame1=1000 title=
    Hires')

250 #Ignore('hiresx.rsf','hirestr.rsf')

252 # select one x and window out min and max for hires
frame=18
254 Flow('legacyx','legacytr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame)
Result('legacyx','grey color=seismic title=Legacy')
256 Flow('hiresx','hirestr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame)
Result('hiresx','grey color=seismic title=Hires')
258



260


262 # Match time axis - sample at 0.5 ms (?) (1000 Hz max) and window to 3 seconds (*x
    windows)
Flow('hires4','hiresx','spline d1=0.0005 n1=6001 o1=0')
264 Flow('legacy4','legacyx','spline d1=0.0005 n1=6001 o1=0')
Result('hires4','byte gainpanel=all | grey title="High Resolution"')
266 Result('legacy4','byte gainpanel=all | grey title=Legacy')

268
######## TEST ANOTHER LINE #######
270 frame2=22
Flow('legacyx2','legacytr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame2)
272 Result('legacyx2','grey color=seismic title=Legacy')
Flow('hiresx2','hirestr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame2)
274 Result('hiresx2','grey color=seismic title=Hires')
```

```
Flow('hires42','hiresx2','spline d1=0.0005 n1=6001 o1=0')
Flow('legacy42','legacyx2','spline d1=0.0005 n1=6001 o1=0')
Result('hires42','byte gainpanel=all | grey title="High Resolution"')
Result('legacy42','byte gainpanel=all | grey title=Legacy')


######## TEST ANOTHER LINE #######
frame3=25
Flow('legacyx3','legacytr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame3)
Result('legacyx3','grey color=seismic title=Legacy')
Flow('hiresx3','hirestr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame3)
Result('hiresx3','grey color=seismic title=Hires')
Flow('hires43','hiresx3','spline d1=0.0005 n1=6001 o1=0')
Flow('legacy43','legacyx3','spline d1=0.0005 n1=6001 o1=0')
Result('hires43','byte gainpanel=all | grey title="High Resolution"')
Result('legacy43','byte gainpanel=all | grey title=Legacy')



# Look at spectra
Flow('legacy-spec','legacy4','spectra all=y | put label2=Amplitude unit2= ')
Result('legacy-spec','graph title="Legacy Spectrum"')
Flow('hires-spec','hires4','spectra all=y | put label2=Amplitude unit2= ')
Result('hires-spec','graph title="Hires Spectrum"')

Result('spectra','hires-spec legacy-spec','cat axis=2 ${SOURCES[1]} | graph title=
    Spectra label2=Amplitude')
Result('nspectra','hires-spec legacy-spec','cat axis=2 ${SOURCES[1]} | scale axis=1 |
     window max1=180 | graph title="Normalized Spectra"')

# Measure Local Frequency
Flow('legacy-freq','legacy4','iphase order=10 rect1=80 rect2=16 hertz=y complex=y |
    put label=Frequency unit=Hz')
Flow('hires-freq','hires4','iphase order=10 rect1=80 rect2=16 hertz=y complex=y | put
     label=Frequency unit=Hz')
Result('legacy-freq','grey mean=y color=j scalebar=y title="Legacy Local Frequency"')
Result('hires-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency"')

Flow('freqdif','legacy-freq hires-freq','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif','grey allpos=y color=j scalebar=y mean=y title="Difference in Local
    Frequenies"')


########


# Gain (?)
#Flow('hirespow','hires4','pow pow1=1')
#Result('hirespow','grey title="Hires"')
#Flow('hirespow-freq','hirespow','iphase order=10 rect1=80 rect2=16 hertz=y complex=y
     | put label=Frequency unit=Hz')
#Result('hirespow-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency
    "')
#Flow('hirespow-spec','hirespow','spectra all=y')
#Result('hirespow-spec','hirespow-spec hires-spec legacy-spec',
#       '''
#       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
#       graph title="Filtered Normalized Spectra" label2="Amplitude" unit2=""
#       ''')
#

```

```
# Bandpass filter test
flo = 25
fhi = 75
flol = 40

#sferf
#Flow('hiresfilt','hirespow','bandpass fhi=%d flo=%d'%(fhi,flo))
Flow('hiresfilt','hires4','bandpass fhi=%d flo=%d'%(fhi,flo))
Flow('legacyfilt','legacy4','bandpass flo=%d '%(flol))

Flow('hiresfilt-spec','hiresfilt','spectra all=y')
Flow('legacyfilt-spec','legacyfilt','spectra all=y')

Flow('hiresagc','hires4','shapeagc rect1=1000 rect2=5')
Result('hiresagc','grey title="Hires AGC"')
Flow('hiresagc-spec','hiresagc','spectra all=y')

# blue = legacy yellow = hires
Result('filtnspectra','hiresagc-spec hiresfilt-spec legacyfilt-spec legacy-spec hires
    -smooth-spec hires-spec',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | scale axis=1 | window max1=180 |
       graph title="Filtered Normalized Spectra" label2="Amplitude" unit2=""
       ''')
Result('hiresfilt', 'grey title="hiresfilt"')
Result('legacyfilt', 'grey title="legacyfilt"')

# Measure Local Frequency
Flow('legacyfilt-freq','legacyfilt','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz')
Flow('hiresfilt-freq','hiresfilt','iphase order=10 rect1=80 rect2=16 hertz=y complex=
    y | put label=Frequency unit=Hz')
Flow('hiresagc-freq','hiresagc','iphase order=10 rect1=80 rect2=16 hertz=y complex=y
    | put label=Frequency unit=Hz')
Result('legacyfilt-freq','grey mean=y color=j scalebar=y title="Legacy Local
    Frequency"')
Result('hiresfilt-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency"
    ')
Result('hiresagc-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency"'
    )

Flow('freqdif2','legacyfilt-freq hiresfilt-freq','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif2','grey allpos=y color=j scalebar=y mean=y title="Difference in Local
     Frequenies"')

# Nonstationary smoothing applied to hires to match with legacy
scale=12
Flow('rect','legacyfilt-freq hiresagc-freq','math f1=${SOURCES[1]} output="sqrt(%g
    *(1/(input*input)-1/(f1*f1)))/%g"' %(scale,2*math.pi*0.001))
#Flow('rect','legacyfilt-freq hires-freq','math f1=${SOURCES[1]} output="sqrt(%g*(1/(
    input*input)-1/(f1*f1)))/%g"' %(scale,2*math.pi*0.001))
Result('rect','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')

#Flow('hires-smooth','hires4 rect','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth','hiresagc rect','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec','hires-smooth','spectra all=y')
Result('hires-smooth-spec','hires-smooth-spec legacyfilt-spec',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
       graph title="Normalized Spectra after Smoothing 1" label2="Amplitude"
```

```
       ''')
Result('hires-smooth', 'grey title="Hires Smooth"')

# Difference in local frequencies with nonstationary smoothing applied to hires
Flow('hires-smooth-freq','hires-smooth','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz ')
Result('hires-smooth-freq','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
Flow('freqdif-filt','legacyfilt-freq hires-smooth-freq','add scale=-1,1 ${SOURCES[1]}
    ')
Result('freqdif-filt','grey allpos=y color=j scalebar=y mean=y title="Difference in
    Local Frequencies after Filtering" ')


# SECOND TIME
# there is a relationship between the frequency difference and the ideal radius size
# account for it here
Flow('rect2','rect freqdif-filt','math s=${SOURCES[1]} output="input+0.25*s"')
Result('rect2','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')

#Flow('hires-smooth2','hires4 rect2','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth2','hiresagc rect2','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec2','hires-smooth2','spectra all=y')
Result('hires-smooth-spec2','hires-smooth-spec2 legacyfilt-spec',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
       graph title="Normalized Spectra after Smoothing 2" label2="Amplitude"
       ''')
Result('hires-smooth2', 'grey title="Hires Smooth"')

# Difference in local frequencies with nonstationary smoothing applied to hires
Flow('hires-smooth-freq2','hires-smooth2','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz ')
Result('hires-smooth-freq2','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
Flow('freqdif-filt2','legacyfilt-freq hires-smooth-freq2','add scale=-1,1 ${SOURCES
    [1]}')
Result('freqdif-filt2','grey allpos=y color=j scalebar=y mean=y clip=30 bias=-15
    title="Difference in Local Frequencies after Filtering 2" ')


# THIRD TIME
Flow('rect3','rect2 freqdif-filt2','math s=${SOURCES[1]} output="input+0.15*s"')
Result('rect3','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')

#Flow('hires-smooth3','hires4 rect3','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth3','hiresagc rect3','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec3','hires-smooth3','spectra all=y')
Result('hires-smooth-spec3','hires-smooth-spec3 legacyfilt-spec',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
       graph title="Normalized Spectra after Smoothing 3" label2="Amplitude"
       ''')
Result('hires-smooth3', 'grey title="Hires Smooth"')

# Difference in local frequencies with nonstationary smoothing applied to hires
Flow('hires-smooth-freq3','hires-smooth3','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz ')
Result('hires-smooth-freq3','grey mean=y color=j scalebar=y title="Hires Local
```

```
    Frequency Smoothed" ')
Flow('freqdif-filt3','legacyfilt-freq hires-smooth-freq3','add scale=-1,1 ${SOURCES
    [1]}')
Result('freqdif-filt3','grey allpos=y color=j scalebar=y mean=y clip=30 bias=-15
    title="Difference in Local Frequencies after Filtering 3" ')


# FOURTH TIME
Flow('rect4','rect3 freqdif-filt3','math s=${SOURCES[1]} output="input+0.35*s"')
Result('rect4','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')

#Flow('hires-smooth4','hires4 rect4','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth4','hiresagc rect4','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec4','hires-smooth4','spectra all=y')
Result('hires-smooth-spec4','hires-smooth-spec4 legacyfilt-spec',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
       graph title="Normalized Spectra after Smoothing 4" label2="Amplitude"
       ''')
Result('hires-smooth4', 'grey title="Hires Smooth"')

# Difference in local frequencies with nonstationary smoothing applied to hires
Flow('hires-smooth-freq4','hires-smooth4','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz ')
Result('hires-smooth-freq4','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
Flow('freqdif-filt4','legacyfilt-freq hires-smooth-freq4','add scale=-1,1 ${SOURCES
    [1]}')
Result('freqdif-filt4','grey allpos=y color=j scalebar=y mean=y clip=30 bias=-15
    title="Difference in Local Frequencies after Filtering 4" ')


# FIFTH TIME
# mask necessary to "over-smooth" the high frequency differences
Flow('mask5','freqdif-filt4','mask max=14 | dd type=float')
Flow('mask5b','mask5','math output="abs(input-1)"') # inverse mask
Flow('rect5b','mask5b','math output=20*input | smooth rect1=50')
Flow('rect5c','rect4 freqdif-filt4','math s=${SOURCES[1]} output="input+0.25*s"')
Flow('rect5',' rect5b rect5c','math m=${SOURCES[1]} output=input+m')
Result('rect5','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
#Flow('rect5','rect4 freqdif-filt3','math s=${SOURCES[1]} output="input+0.25*s"')
#Result('rect5','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')

#Flow('hires-smooth5','hires4 rect5','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth5','hiresagc rect5','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec5','hires-smooth5','spectra all=y')
Result('hires-smooth-spec5','hires-smooth-spec5 legacyfilt-spec',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
       graph title="Normalized Spectra after Smoothing 5" label2="Amplitude"
       ''')
Result('hires-smooth5', 'grey title="Hires Smooth"')

# Difference in local frequencies with nonstationary smoothing applied to hires
Flow('hires-smooth-freq5','hires-smooth5','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz ')
Result('hires-smooth-freq5','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
```

```
Flow('freqdif-filt5','legacyfilt-freq hires-smooth-freq5','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif-filt5','grey allpos=y color=j scalebar=y mean=y clip=30 bias=-15 title="Difference in Local Frequencies after Filtering 5" ')

Result('freqdif-filt6','freqdif-filt5','grey allpos=y color=j bias=0 clip=100 scalebar=y mean=y title="Difference in Local Frequencies after Filtering" ')

############################

Flow('hires-smooth2f','hires-smooth5','bandpass fhi=60 nphi=1 flo=20')

Flow('hires-smooth-spec2f','hires-smooth2f','spectra all=y')
Result('hires-smooth-spec2f','hires-smooth-spec2f legacyfilt-spec',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
       graph title="Normalized Spectra after Smoothing" label2="Amplitude"
       ''')
Flow('hires-smooth-freq2f','hires-smooth2f','iphase order=10 rect1=80 rect2=16 hertz=y complex=y | put label=Frequency unit=Hz ')
Result('hires-smooth-freq2f','grey mean=y color=j scalebar=y title="Hires Local Frequency Smoothed" ')
Flow('freqdif-filt2f','legacyfilt-freq hires-smooth-freq2f','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif-filt2f','grey allpos=y color=j scalebar=y mean=y clip=30 bias=-15 title="Difference in Local Frequencies after Filtering 2f" ')

################################################################################


# Warpscan
# Balance Amplitues

#Flow('hires-balanced','hires-smooth5 legacyfilt','abalance other=${SOURCES[1]} rect1=100 rect2=260') # Experiment with rect2
Flow('hires-balanced','hires-smooth5 legacyfilt','abalance other=${SOURCES[1]} rect1=100 rect2=10 ')
Result('hires-balanced','grey title="Hires balanced"')
Flow('hires-despike','hires-balanced','despike2 wide1=2')
#Flow('hires-despike','hires-balanced2','despike2 wide1=2')
Result('hires-despike','grey title="Hires balanced"')


Flow('hires-balanced2','hires-smooth5 legacyfilt','abalance other=${SOURCES[1]} rect1=100 rect2=260')
Result('hires-balanced2','grey title="Hires balanced"')

Flow('hires-balanced-reverse','hires-smooth5 legacyfilt','abalance other=${SOURCES[1]} rect1=100 rect2=260 reverse=n')
Result('hires-balanced-reverse','grey title="Hires balanced"')

# Select first trace
tr=3.245e8
#tr=3.250e8
for case in ('legacy4','hires-balanced','hires-balanced-reverse','legacyfilt','hires-despike'):
    trace = case + '-trace'
    Flow(trace,case,'window n2=1 min2=%d'%tr)

# take the difference
Flow('trace-diff','legacyfilt-trace hires-despike-trace','add ${SOURCES[1]} scale
```

```
    =1,-1')

Flow('trace-diff-reverse','legacyfilt-trace hires-balanced-reverse-trace','add ${
    SOURCES[1]} scale=1,-1')

Result('traces','legacyfilt-trace hires-despike-trace trace-diff',
               'cat axis=2 ${SOURCES[1:3]} | dots gaineach=n labels=Legacy:Hires:
    Difference yreverse=y')

Result('traces-reverse','legacyfilt-trace hires-balanced-reverse-trace trace-diff-
    reverse',
               'cat axis=2 ${SOURCES[1:3]} | dots gaineach=n labels=Legacy:Hires:
    Difference yreverse=y')

# justification of reverse
Result('traces-reverse-diff','trace-diff trace-diff-reverse',
               'cat axis=2 ${SOURCES[1]} | dots gaineach=n labels=reverse=y:reverse=n
     yreverse=y')


# TIME SHIFT OF FIRST TRACE
# Scanning different time shifts
Flow('warpscan','hires-despike-trace legacyfilt-trace',
               '''
               warpscan shift=y ng=56 g0=-0.05 dg=0.001
               other=${SOURCES[1]} rect1=300
               ''')

Flow('warppick','warpscan','scale axis=2 | pick rect1=50 vel0=-0.005 an=0.1')

Plot('warpscan',
             '''
             grey allpos=y color=j title="Shift Scan"
             label2=Shift unit2=s
             ''')

# This is what was removed
Plot('warppick',
             '''
             graph plotfat=3 plotcol=7 wanttitle=n wantaxis=n
             min2=-0.05 max2=0.005 pad=n yreverse=y transp=y
             ''')

Result('warpscan','warpscan warppick','Overlay')

# Apply picked shift
Flow('warp','hires-despike-trace legacyfilt-trace warppick','warp1 other=${SOURCES
    [1]} warpin=${SOURCES[2]} nliter=0')

Flow('warp-diff','legacyfilt-trace warp','add ${SOURCES[1]} scale=1,-1')

Result('wtraces','legacyfilt-trace warp warp-diff','cat axis=2 ${SOURCES[1:3]} | dots
     gaineach=n labels=Legacy:Warped:Difference yreverse=y')

# warpscan on image
Flow('warpscan3','hires-despike legacyfilt',
     '''
     warpscan shift=y ng=56 g0=-0.05 dg=0.001
     other=${SOURCES[1]} rect1=100 rect2=5
     ''')
```

```
Result('warpscan3','''byte gainpanel=all allpos=y bar=bar.rsf | transp plane=23 |
       grey3 color=j frame1=500 frame2=1000 frame3=25 title="Local Similarity Scan"
       label3="Time Shift" unit3=s scalebar=y barlabel=Similarity''')

Flow('warppick3','warpscan3',
     'scale axis=2 | pick rect1=10 rect2=20 rect3=10 vel0=-0.0 an=0.1')

Result('warppick3','grey color=j allpos=y title="Estimated Time Shift" scalebar=y
    barlabel=Time barunit=s clip=0.015 bias=-0.01')

Flow('diff0','hires-despike legacyfilt','add scale=1,-1 ${SOURCES[1]}')

# Apply picked shift
Flow('warp3','hires-despike legacyfilt warppick3',
     'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')

Flow('diff1','warp3 legacyfilt','add scale=1,-1 ${SOURCES[1]}')

Result('diff0','grey clip=1594.26 title="Difference before warping"')
Result('diff1','grey clip=1764.31 title="Difference after warping"')


# SHIFT WITHOUT BALANCING AMPLITUDES
Flow('hires-warp','hiresagc warppick3',
     'warp1 other=${SOURCES[0]} warpin=${SOURCES[1]} nliter=0')

Result('hires-warp', 'grey title="Hires Warped Image"')

Flow('hires-warp-freq','hires-warp','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz')

Result('hires-warp-freq','grey mean=y color=j scalebar=y title="Warped Hires Local
    Frequency" ')

# create smoothing operator
# use legacyfilt or legacy4?
Flow('rect6','legacyfilt-freq hires-warp-freq',
     'math f1=${SOURCES[1]} output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" ' % (
    scale,2*math.pi*0.001))
Result('rect6','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
Flow('hires-warp-smooth','hires-warp rect6','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-warp-smooth-freq','hires-warp-smooth','iphase order=10 rect1=80 rect2=16
    hertz=y complex=y | put label=Frequency unit=Hz ')
Result('hires-warp-smooth-freq','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
Flow('freqdif-smooth-filt','legacyfilt-freq hires-warp-smooth-freq','add scale=-1,1 $
    {SOURCES[1]}')
Result('freqdif-smooth-filt','grey allpos=y color=j scalebar=y mean=y clip=30 bias
    =-15 title="Difference in Local Frequencies after Filtering 1" ')

# Second Time
Flow('rect7','rect6 freqdif-smooth-filt','math s=${SOURCES[1]} output="input+0.25*s"'
    )
Result('rect7','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
Flow('hires-warp-smooth2','hires-warp rect7','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-warp-smooth-freq2','hires-warp-smooth2','iphase order=10 rect1=80 rect2
    =16 hertz=y complex=y | put label=Frequency unit=Hz ')
Result('hires-warp-smooth-freq2','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
```

```
Flow('freqdif-smooth-filt2','legacyfilt-freq hires-warp-smooth-freq2','add scale=-1,1
     ${SOURCES[1]}')
Result('freqdif-smooth-filt2','grey allpos=y color=j scalebar=y mean=y clip=30 bias
    =-15 title="Difference in Local Frequencies after Filtering 1" ')

# Third time
Flow('rect8','rect7 freqdif-smooth-filt2','math s=${SOURCES[1]} output="input+0.25*s"
    ')
Result('rect8','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
Flow('hires-warp-smooth3','hires-warp rect8','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-warp-smooth-freq3','hires-warp-smooth3','iphase order=10 rect1=80 rect2
    =16 hertz=y complex=y | put label=Frequency unit=Hz ')
Result('hires-warp-smooth-freq3','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
Flow('freqdif-smooth-filt3','legacyfilt-freq hires-warp-smooth-freq3','add scale=-1,1
     ${SOURCES[1]}')
Result('freqdif-smooth-filt3','grey allpos=y color=j scalebar=y mean=y clip=30 bias
    =-15 title="Difference in Local Frequencies after Filtering 1" ')

# Fourth time
Flow('rect9','rect8 freqdif-smooth-filt3','math s=${SOURCES[1]} output="input+0.25*s"
    ')
Result('rect9','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
Flow('hires-warp-smooth4','hires-warp rect9','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-warp-smooth-freq4','hires-warp-smooth4','iphase order=10 rect1=80 rect2
    =16 hertz=y complex=y | put label=Frequency unit=Hz ')
Result('hires-warp-smooth-freq4','grey mean=y color=j scalebar=y title="Hires Local
    Frequency Smoothed" ')
Flow('freqdif-smooth-filt4','legacyfilt-freq hires-warp-smooth-freq4','add scale=-1,1
     ${SOURCES[1]}')
Result('freqdif-smooth-filt4','grey allpos=y color=j scalebar=y mean=y clip=30 bias
    =-15 title="Difference in Local Frequencies after Filtering 1" ')

# Final time
Flow('mask10','freqdif-smooth-filt4','mask max=14 | dd type=float')
Flow('mask10b','mask10','math output="abs(input-1)"') # inverse mask
Flow('rect10b','mask10b','math output=20*input | smooth rect1=50')
Flow('rect10c','rect9 freqdif-filt4','math s=${SOURCES[1]} output="input+0.25*s"')
Flow('rect10',' rect10b rect10c','math m=${SOURCES[1]} output=input+m')
# rect10 = smoothing operator
Result('rect10','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')

Flow('hires-warp-smooth10','hires-warp rect10','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-warp-smooth10','hires-warp rect10','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec10','hires-warp-smooth10','spectra all=y')
Result('hires-warp-smooth10', 'grey title="Hires Smooth"')

# CREATE THE BLENDED IMAGE
# reverse=n
Flow('hires-warp-balance-reverse lweight-reverse2','hires-warp-smooth10 legacy4','
    abalance weight=${TARGETS[1]} other=${SOURCES[1]} rect1=320 rect2=160 reverse=n')

#Flow('lweight-reverse','lweight-reverse2','math output=1/input')
Flow('lweight-reverse3','lweight-reverse2','math output=5000 | cut min1=0.25 ')
Flow('lweight-reverse4','lweight-reverse2','cut max1=0.25 ')
Result('lweight-reverse4', 'grey color=j scalebar=y title="Legacy Weight" mean=y')
Flow('lweight-reverse','lweight-reverse3 lweight-reverse4','math fourth=${SOURCES[1]}
     output=input+fourth')
```

```
672
Result('lweight-reverse', 'grey color=j scalebar=y title="Legacy Weight" mean=y')
674
Flow('hires-warp-diff-reverse','hires-warp-balance-reverse legacy4',
676      'add ${SOURCES[1]} scale=-1,1 | pad beg3=1')


678
f = "erfc(x1)"
680
# weight for merge2-reverse
682 #Flow('weight5','lweight-reverse','math output="%s"| cut max1=0.25 | scale dscale=10'
      % f)
    #Flow('hweight','lweight-reverse weight5','math output=100 | cut min1=0.25 | add ${
    SOURCES[1]} | scale dscale=100')
684 #Result('hweight', 'grey color=j scalebar=y title="Hires Weight" mean=y')
    Flow('weight5','lweight-reverse','math output="%s"| cut max1=0.65 | scale dscale=10'
    % f)
686 Flow('hweight','lweight-reverse weight5','math output=5000000 | cut min1=0.65 |
    smooth rect1=300 | add ${SOURCES[1]} | scale dscale=0.001')
    Result('hweight', 'grey color=j scalebar=y title="Hires Weight" mean=y')
688


690 Flow('merge1-reverse','hires-warp-diff-reverse hires-warp rect10 hweight lweight-
    reverse',
      '''
692      conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
      lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}
694      ''')
    Flow('merge','hires-warp merge1-reverse','add ${SOURCES[1]}')
696
# Merge with different weight
698
Flow('mergeagc','merge','shapeagc rect1=1000 rect2=5')
700 Result('mergeagc','grey title="Merged"')

702 Result('mergeagcwindow','mergeagc','window min1=0 max1=0.6 | grey title="Merged"')
    Result('hires4window','hires4','window min1=0 max1=0.6 | grey title="Hires"')
704 Result('legacy4window','legacy4','window min1=0 max1=0.6 | grey title="Legacy"')

706 Result('mergeagcwindow2','mergeagc','window min1=1 max1=2.7 | grey title="Merged"')
    Result('hires4window2','hires4','window min1=1 max1=2.7 | grey title="Hires"')
708 Result('hires4window2agc','hires4','shapeagc rect1=1000 rect2=5 | window min1=1 max1
    =2.7 | grey title="Hires"')
    Result('legacy4window2','legacy4','window min1=1 max1=2.7 | grey title="Legacy"')
710
#Result('windowed','Fig/legacy4window Fig/hires4window Fig/mergeagcwindow Fig/
    legacy4window2 Fig/hires4window2agc Fig/mergeagcwindow2', 'TwoRows')
712 Result('window1','Fig/legacy4window Fig/hires4window Fig/mergeagcwindow', '
    SideBySideAniso')
    Result('window2','Fig/legacy4window2 Fig/hires4window2agc Fig/mergeagcwindow2', '
    SideBySideAniso')
714
# Difference between high-resolution and merged image
716 Result('merge1-reverse','shapeagc rect1=1001 rect2=5 | bandpass fhi=250 | window j1=3
      |grey title="Difference between Merged and Hires" clip=3.7011')
# Final merged image
718 # merge2-reverse
    Result('merge','grey title="Merged" ')
720


722 # Display the spectra
```

```
Flow('legacy4-spec4','legacy4','spectra all=y')
Flow('hires-spec4','hires-warp','spectra all=y')
Flow('merge-spec4','merge','spectra all=y')

# merge Final Spectra
# nspectra22-reverse
Result('nspectra2','legacy4-spec4 hires-spec4 merge-spec4',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=180 |
       scale axis=1 | graph title="Normalized Spectra " dash=2,1,0
       label2="Amplitude" unit2=""
       ''')
       #scale axis=1 | graph title="Spectra "
       #label2="Amplitude" unit2=""

Result('three', "Fig/legacy4 Fig/hires4 Fig/merge ", 'SideBySideAniso')     # for
    newsletter

End()
```

Listing 12: chapter-merge/pcable2/SConstruct

```
from rsf.proj import *
from rsf.recipes.beg import server
from radius import radius

data = {'legacy':'TxLa_merge_PSTM_enhanced_subset1.sgy',
        'hires':'pcable_tpdcn_mig_subset1.sgy'}

for key in data.keys():
    Fetch(data[key],'gom',server)

    # Convert from SEGY format to Madgascar RSF format
    Flow([key,'t'+key,key+'.asc',key+'.bin'],data[key],
         '''
         segyread tfile=${TARGETS[1]} hfile=${TARGETS[2]} bfile=${TARGETS[3]}
         ''')
    Result(key+'3',key,'intbin xk=iline yk=xline | put label2=Inline label3=Crossline
     | byte gainpanel=all | grey3 frame1=500 frame2=200 frame3=200 title=%s' % key.
    capitalize())

Flow('legacy2','legacy','intbin xk=iline yk=xline | put label2=Inline label3=
    Crossline | byte gainpanel=all ' )
Result('legacy','legacy2','grey3 frame1=150 frame2=20 frame3=20 title="Legacy"' )

Flow('hires2','hires','intbin xk=iline yk=xline | put label2=Inline label3=Crossline
    | byte gainpanel=all ')
Result('hires','hires2','grey3 frame1=500 frame2=200 frame3=200 title="Hires"' )


min1=36935190/1000
max1=37188381/1000
min2=325465425/1000
max2=325648750/1000

# Legacy
Flow('lbin3','tlegacy','intbin3 head=$SOURCE')
Flow('lcdpx','lbin3','headermath output=cdpx/1000-%d | window n2=1 | dd type=float'%
    min1)
Flow('lcdpy','lbin3','headermath output=cdpy/1000-%d | window n2=1 | dd type=float'%
    min2)
```

```
Flow('lcdp','lcdpx lcdpy','cmplx ${SOURCES[1]}')
Result('lcdp','graph title="CDP" min1=%d max1=%d min2=%d max2=%d plotcol=6 plotfat=2
    dash=2' %(-5, 255, -1, 185))
#Result('lcdp','graph title="CDP" label1=x label2=y plotcol=6 dash=2')

# Hires
Flow('hbin3','thires','intbin3 head=$SOURCE')
Flow('hcdpx','hbin3','headermath output=cdpx/1000-%d | window n2=1 | dd type=float'%
    min1)
Flow('hcdpy','hbin3','headermath output=cdpy/1000-%d | window n2=1 | dd type=float'%
    min2)
Flow('hcdp','hcdpx hcdpy','cmplx ${SOURCES[1]}')
Result('hcdp','graph title="CDP" label1=x label2=y unit1=kft unit2=kft min1=%d max1=%
    d min2=%d max2=%d plotcol=5 plotfat=2 dash=2' %(-5, 255, -1, 185))
#Result('hcdp','graph title="CDP" label1=x label2=y plotcol=5 dash=2')

Result('cdp','Fig/lcdp Fig/hcdp','Overlay')



### ORIGINAL COORDS

min12=36268165
max12=37822678
min22=325028650
max22=326036875

# Legacy
Flow('lcdpx2','lbin3','headermath output=cdpx | window n2=1 | dd type=float')
Flow('lcdpy2','lbin3','headermath output=cdpy | window n2=1 | dd type=float')
Flow('lcdp2','lcdpx2 lcdpy2','cmplx ${SOURCES[1]}')
Result('lcdp2','graph title="CDP" label1=x label2=y unit1=kft unit2=kft min1=%d max1
    =%d min2=%d max2=%d plotcol=6 plotfat=2 dash=2' %(min12, max12, min22, max22))

# Hires
Flow('hcdpx2','hbin3','headermath output=cdpx | window n2=1 | dd type=float')
Flow('hcdpy2','hbin3','headermath output=cdpy | window n2=1 | dd type=float')
Flow('hcdp2','hcdpx2 hcdpy2','cmplx ${SOURCES[1]}')
Result('hcdp2','graph title="CDP" label1=x label2=y unit1=kft unit2=kft min1=%d max1
    =%d min2=%d max2=%d plotcol=5 plotfat=2 dash=2' %(min12, max12, min22, max22))

Result('cdp2','Fig/lcdp2 Fig/hcdp2','Overlay')


#####


from math import cos, sin, radians, pi

a=111.9
a2=-2.95 #for lined up legacy to hires
cs=cos(radians(a))
sn=sin(radians(a))
cs2=cos(radians(a2))
sn2=sin(radians(a2))
ox=36976140
oy=325545425

# Rotate them both a degrees around (ox, oy)
Flow('tlegacyr','tlegacy','dd type=float | headermath key=unass1 output="(cdpx-%f)*(%
    f)+(cdpy-%f)*(%f)+%f" | headermath key=unass2 output="-(cdpx-%f)*(%f)+(cdpy-%f)
```

```
    *(%f)+%f" | dd type=int'%(ox, cs, oy, sn, ox, ox, sn, oy, cs, oy))
Flow('thiresr','thires','dd type=float | headermath key=unass1 output="(cdpx-%f)*(%f)
    +(cdpy-%f)*(%f)+%f" | headermath key=unass2 output="-(cdpx-%f)*(%f)+(cdpy-%f)*(%f
    )+%f" | dd type=int'%(ox, cs, oy, sn, ox, ox, sn, oy, cs, oy))

# Rotate them as legacy
Flow('tlegacyr2','tlegacy','dd type=float | headermath key=unass1 output="(cdpx-%f)
    *(%f)+(cdpy-%f)*(%f)+%f" | headermath key=unass2 output="-(cdpx-%f)*(%f)+(cdpy-%f
    )*(%f)+%f" | dd type=int'%(ox, cs2, oy, sn2, ox, ox, sn2, oy, cs2, oy))
Flow('thiresr2','thires','dd type=float | headermath key=unass1 output="(cdpx-%f)*(%f
    )+(cdpy-%f)*(%f)+%f" | headermath key=unass2 output="-(cdpx-%f)*(%f)+(cdpy-%f)*(%
    f)+%f" | dd type=int'%(ox, cs2, oy, sn2, ox, ox, sn2, oy, cs2, oy))

# Legacy rotated: View new lcdpx and lcdpy
Flow('lbinr2','tlegacyr2','dd type=int | intbin3 head=$SOURCE')
Flow('lcdpxr2','lbinr2','headermath output=unass1 | window n2=1 | dd type=float')
Flow('lcdpyr2','lbinr2','headermath output=unass2 | window n2=1 | dd type=float')
Flow('lcdpr2','lcdpxr2 lcdpyr2','cmplx ${SOURCES[1]}')

# Legacy rotated: View new hcdpx and hcdpy
Flow('hbinr2','thiresr2','intbin3 head=$SOURCE')
Flow('hcdpxr2','hbinr2','headermath output=unass1 | window n2=1 | dd type=float')
Flow('hcdpyr2','hbinr2','headermath output=unass2 | window n2=1 | dd type=float')
Flow('hcdpr2','hcdpxr2 hcdpyr2','cmplx ${SOURCES[1]}')

# View new lcdpx and lcdpy
Flow('lbinr','tlegacyr','dd type=int | intbin3 head=$SOURCE')
Flow('lcdpxr','lbinr','headermath output=unass1 | window n2=1 | dd type=float')
Flow('lcdpyr','lbinr','headermath output=unass2 | window n2=1 | dd type=float')
Flow('lcdpr','lcdpxr lcdpyr','cmplx ${SOURCES[1]}')

# View new hcdpx and hcdpy
Flow('hbinr','thiresr','intbin3 head=$SOURCE')
Flow('hcdpxr','hbinr','headermath output=unass1 | window n2=1 | dd type=float')
Flow('hcdpyr','hbinr','headermath output=unass2 | window n2=1 | dd type=float')
Flow('hcdpr','hcdpxr hcdpyr','cmplx ${SOURCES[1]}')

# Display values
min12=36826024
max12=37083984
min22=325314688
max22=325608608

# for legacy
min13=36938656
max13=37183520
min23=325475808
max23=325646976

Result('hcdpr','graph title="CDP rotated" label1=x label2=y min1=%d max1=%d min2=%d
    max2=%d' %(min12, max12, min22, max22))
Result('lcdpr','graph title="CDP rotated" label1=x label2=y min1=%d max1=%d min2=%d
    max2=%d' %(min12, max12, min22, max22))
Result('cdpr','Fig/lcdpr Fig/hcdpr','Overlay')
# legacy rotated results
Result('hcdpr2','graph title="CDP rotated" label1=x label2=y min1=%d max1=%d min2=%d
    max2=%d' %(min13, max13, min23, max23))
Result('lcdpr2','graph title="CDP rotated" label1=x label2=y min1=%d max1=%d min2=%d
    max2=%d' %(min13, max13, min23, max23))
Result('cdpr2','Fig/lcdpr2 Fig/hcdpr2','Overlay')
```

```
# Mask values
min13=36915196
max13=36993752
min23=325348256
max23=325576544

# Display masked legacy
Flow('lmaskx','tlegacyr','headermath output=unass1 | mask min=%d max=%d | dd type=
    float | put d1=1 d2=1 o1=0 o2=0' %(min13, max13))
Flow('lmasky','tlegacyr','headermath output=unass2 | mask min=%d max=%d | dd type=
    float | put d1=1 d2=1 o1=0 o2=0' %(min23, max23))
Flow('lmask','lmaskx lmasky','math y=${SOURCES[1]} output="input*y" | dd type=int')
Flow('tlegacymask','tlegacyr lmask','headercut mask=${SOURCES[1]} | dd type=int')

Flow('lbin3mask','tlegacymask','intbin3 head=$SOURCE')
Flow('lcdpxmask','lbin3mask','headermath output=unass1 | window n2=1 | dd type=float'
    )
Flow('lcdpymask','lbin3mask','headermath output=unass2 | window n2=1 | dd type=float'
    )
Flow('lcdpmask','lcdpxmask lcdpymask','cmplx ${SOURCES[1]}')
Result('lcdpmask','graph title="Legacy" label1=x label2=y min1=%d max1=%d min2=%d
    max2=%d dash=2' %(min12, max12, min22, max22))
Result('cdpmask','Fig/lcdpmask Fig/hcdpr','Overlay')

# Windowed legacy data
Flow('legacymask','legacy lmask','headercut mask=${SOURCES[1]} | intbin xk=iline yk=
    xline | put label2=Inline label3=Crossline')
Result('legacymask','byte gainpanel=all | grey3 frame1=150 frame2=20 frame3=20 title=
    Legacy')

#nx=35
#ny=30
nx=65
ny=60
#nx=57
#ny=52

Flow('tlegacyrf','tlegacyr','dd type=float')
Flow('bin-legacy legacyfold','legacy tlegacyrf','transp | bin head=${SOURCES[1]}
    interp=2 fold=${TARGETS[1]} xkey=89 ykey=90 xmin=%d xmax=%d ymin=%d ymax=%d nx=%d
     ny=%d' %(min13, max13, min23, max23, nx, ny))
Result('legacyfold','grey color=j title="Legacy Fold" scalebar=y')
Result('bin-legacy','transp plane=13| put label2=cdpy label1=Time unit1=s unit2=ft
    unit3=ft label3=cdpx | byte gainpanel=all | grey3 frame1=150 frame2=26 frame3=125
     title=Legacy')

Flow('thiresrf','thiresr','dd type=float')
Flow('bin-hires hiresfold','hires thiresrf','transp | bin head=${SOURCES[1]} interp=2
     fold=${TARGETS[1]} xkey=89 ykey=90 xmin=%d xmax=%d ymin=%d ymax=%d nx=%d ny=%d'
    %(min13, max13, min23, max23, nx, ny))
Result('hiresfold','grey color=j title="Hires Fold" scalebar=y')
Result('bin-hires','transp plane=13| put label2=cdpy label1=time unit1=s unit2=ft
    unit3=ft label3=cdpx | byte gainpanel=all | grey3 frame1=500 frame2=26 frame3=125
     title=Hires')


Flow('legacytr','bin-legacy','transp plane=31 | put label3=cdpx label2=cdpy unit3=ft
    unit2=ft')
Result('legacytr','byte gainpanel=all |  window max1=2 |grey3 frame3=20 frame2=15
    frame1=125 title=Legacy')
Flow('hirestr','bin-hires','transp plane=31 | put label3=cdpx label2=cdpy unit3=ft
```

```
    unit2=ft')
Result('hirestr','byte gainpanel=all | grey3 frame3=20 frame2=15 frame1=1000 title=
    Hires')

# window out min and max for hires
Flow('legacy4','legacytr','spline d1=0.0005 n1=3999 o1=0 ')

Result('legacy4','put o1=0 o2=0 o3=0 d2=.386929 unit2= d3=.122744 unit3= | byte
    gainpanel=all | grey3 title=Legacy frame1=1500 frame2=10 frame3=30')
Result('hires4','put o1=0 o2=0 o3=0 d2=.386929 unit2= d3=.122744 unit3= | byte
    gainpanel=all | grey3 title="High resolution" frame1=1500 frame2=10 frame3=30')
Flow('hires4','hirestr','spline d1=0.0005 n1=3999 o1=0 ')

# Look at spectra
Flow('legacy-spec','legacy4','spectra all=y | put label2=Amplitude unit2= ')
Result('legacy-spec','graph title="Legacy Spectrum"')
Flow('hires-spec','hires4','spectra all=y | put label2=Amplitude unit2= ')
Result('hires-spec','graph title="Hires Spectrum"')

Result('spectra','hires-spec legacy-spec','cat axis=2 ${SOURCES[1]} | graph title=
    Spectra label2=Amplitude')
Result('nspectra','hires-spec legacy-spec','cat axis=2 ${SOURCES[1]} | scale axis=1 |
     window max1=300 | graph title="Normalized Spectra"')

# Measure Local Frequency
Flow('legacy-freq','legacy4','iphase order=10 rect1=40 rect2=6 rect3=3 hertz=y
    complex=y | put label=Frequency unit=Hz')
Flow('hires-freq','hires4','iphase order=10 rect1=40 rect2=6 rect3=3 hertz=y complex=
    y | put label=Frequency unit=Hz')
Result('legacy-freq','grey mean=y color=j scalebar=y title="Legacy Local Frequency"')
Result('hires-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency"')

Flow('freqdif','legacy-freq hires-freq','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif','byte gainpanel=all bar=bar.rsf allpos=y | grey3 frame1=400 frame2
    =40 frame3=30 allpos=y color=j scalebar=y mean=y title="Difference in Local
    Frequenies"')

flol = 25
fhil = 125
Flow('legacyfilt','legacy4','bandpass flo=%d '%(flol))
Flow('hiresfilt','hires4','bandpass fhi=%d '%(fhil))
Flow('legacyfilt-spec','legacyfilt','spectra all=y')

Flow('hiresagc','hires4','shapeagc rect1=1000 rect2=5 rect3=3')
Result('hiresagc','grey title="Hires AGC"')
Flow('hiresagc-spec','hiresagc','spectra all=y')

# blue = legacy yellow = hires
Result('filtnspectra','hiresagc-spec legacyfilt-spec legacy-spec hires-spec',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | scale axis=1 | window max1=180 |
       graph title="Filtered Normalized Spectra" label2="Amplitude" unit2=""
       ''')
Result('legacyfilt', 'byte gainpanel=all | grey3 frame1=400 frame2=40 frame3=30 title
    ="legacyfilt"')

# Measure Local Frequency
Flow('legacyfilt-freq','legacyfilt','iphase order=10 rect1=40 rect2=6 rect3=3 hertz=y
     complex=y | put label=Frequency unit=Hz')
Flow('hiresagc-freq','hiresagc','iphase order=10 rect1=40 rect2=6 rect3=3 hertz=y
    complex=y | put label=Frequency unit=Hz')
```

```
Result('legacyfilt-freq','grey mean=y color=j scalebar=y title="Legacy Local
    Frequency"')
Result('hiresagc-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency"'
    )


Flow('freqdif2','legacyfilt-freq hires-freq','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif2','byte gainpanel=all bar=bar.rsf allpos=y | grey3 frame1=400 frame2
    =40 frame3=30 allpos=y color=j scalebar=y mean=y title="Difference in Local
    Frequenies"')

# Nonstationary smoothing applied to hires to match with legacy
scale=12

corrections=2
radius('hiresfilt','legacyfilt', corrections, [0.13,0.2,0.3,0.5,0.5], bias=0, clip
    =65, rect1=80, rect2=16, rect3=16, minval=0, maxval=65 )

Flow('rect','legacyfilt-freq hiresagc-freq','math f1=${SOURCES[1]} output="sqrt(%g
    *(1/(input*input)-1/(f1*f1)))/%g"' %(scale,2*pi*0.001))
Result('rect','byte gainpanel=all bar=bar.rsf allpos=y | grey3 color=j frame1=400
    frame2=40 frame3=30 mean=y title="Smoothing Radius" scalebar=y barlabel=Radius
    barunit=samples')

Flow('hires-smooth','hiresagc rect','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec','hires-smooth','spectra all=y')
Result('hires-smooth-spec','hires-smooth-spec legacy-spec',
       '''
       cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
       graph title="Normalized Spectra after Smoothing 1" label2="Amplitude"
       ''')
Result('hires-smooth', 'grey title="Hires Smooth"')

# Difference in local frequencies with nonstationary smoothing applied to hires
Flow('hires-smooth-freq','hires-smooth','iphase order=10 rect1=40 rect2=6 rect3=3
    hertz=y complex=y | put label=Frequency unit=Hz ')
Result('hires-smooth-freq','byte gainpanel=all bar=bar.rsf allpos=y | grey3 mean=y
    color=j scalebar=y title="Hires Local Frequency Smoothed" frame1=500 frame2=10
    frame3=10 ')
Flow('freqdif-filt','legacyfilt-freq hires-smooth-freq','add scale=-1,1 ${SOURCES[1]}
    ')
Result('freqdif-filt','byte gainpanel=all bar=bar.rsf allpos=y | grey3 frame1=400
    frame2=40 frame3=30 allpos=y color=j scalebar=y mean=y title="Difference in Local
     Frequenies"')

# Warpscan
# Balance Amplitues

hs = 'high-smooth%i'%corrections

Flow('hires-balanced','%s hires-smooth legacyfilt'%hs,'abalance other=${SOURCES[1]}
    rect1=100 rect2=30 rect3=10')
Result('hires-balanced','grey title="Hires balanced"')
Flow('hires-despike','hires-balanced','despike2 wide1=2')
Result('hires-despike','byte gainpanel=all | grey3 frame1=400 frame2=40 frame3=30
    title="Hires balanced"')


Flow('warpscan3','hires-despike legacyfilt',
     '''
     warpscan shift=y ng=56 g0=-0.05 dg=0.001
```

```
     other=${SOURCES[1]} rect1=100 rect2=10 rect3=10 rect4=10
     ''')

Result('warpscan3','''byte gainpanel=all allpos=y bar=bar.rsf | transp plane=23 |
       grey3 color=j frame1=500 frame2=40 frame3=30 title="Local Similarity Scan"
       label3="Time Shift" unit3=s scalebar=y barlabel=Similarity''')


Flow('warppick3','warpscan3',
     'scale axis=2 | pick rect1=20 rect2=40 rect3=20 vel0=-0.0 an=0.08')

Result('warppick3','grey color=j allpos=y title="Estimated Time Shift" scalebar=y
    barlabel=Time barunit=s clip=0.015 bias=-0.01')

Flow('diff0','hires-despike legacyfilt','add scale=1,-1 ${SOURCES[1]}')

# Apply picked shift
Flow('warp3','hires-despike legacyfilt warppick3',
     'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')

Flow('diff1','warp3 legacyfilt','add scale=1,-1 ${SOURCES[1]}')

Result('diff0','grey clip=6606.94 title="Difference before warping"')
Result('diff1','grey clip=7072.5 title="Difference after warping"')

Flow('diff2','diff0 diff1','add scale=1,-1 ${SOURCES[1]}')
Result('diff2','grey title="Difference before and after warping"')

# SHIFT WITHOUT BALANCING AMPLITUDES
Flow('hires-warp','hires4 warppick3',
     'warp1 other=${SOURCES[0]} warpin=${SOURCES[1]} nliter=0')

Result('hires-warp', 'grey title="Hires Warped Image"')

Flow('hires-warp-freq','hires-warp','iphase order=10 rect1=80 rect2=16 rect3=8 hertz=
    y complex=y | put label=Frequency unit=Hz')

Result('hires-warp-freq','byte gainpanel=all allpos=y bar=bar.rsf | grey3 mean=y
    color=j scalebar=y title="Warped Hires Local Frequency" frame1=500 frame2=10
    frame3=30')

### second
Flow('hires-warpfilt','hires-warp','bandpass fhi=%d '%(fhil))
radius('hires-warpfilt','legacyfilt', corrections, [0.13,0.2,0.3,0.5,0.5], bias=0,
    clip=65, rect1=80, rect2=16, rect3=16, minval=0, maxval=65, ind='warp')

rt = 'rect%i'%corrections
rect_warp = "rect%i%s"%(corrections,'warp')
# smoothed warped hires
Flow('hires-warp-smooth','hires-warp %s'%rect_warp,'nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec2','hires-warp-smooth','spectra all=y')
Result('hires-warp-smooth', 'grey title="Hires Smooth"')

# CREATE THE BLENDED IMAGE
Flow('hires-warp-balance-reverse lweight-reverse2','hires-warp-smooth legacy4','
    abalance weight=${TARGETS[1]} other=${SOURCES[1]} rect1=320 rect2=160 rect3=30
    reverse=n')

Flow('lweight-reverse3','lweight-reverse2','math output=500 | cut min1=0.25 ')
Result('lweight-reverse3', 'grey color=j scalebar=y title="Legacy Weight" mean=y')
Flow('lweight-reverse4','lweight-reverse2','cut max1=0.25 ')
```

```
Result('lweight-reverse4', 'grey color=j scalebar=y title="Legacy Weight" mean=y')
Flow('lweight-reverse','lweight-reverse3 lweight-reverse4','math fourth=${SOURCES[1]}
     output=input+fourth')

Result('lweight-reverse', 'grey color=j scalebar=y title="Legacy Weight" mean=y')

Flow('hires-warp-diff-reverse','hires-warp-balance-reverse legacy4',
     'add ${SOURCES[1]} scale=-1,1 | pad beg4=1')


f = "erfc(x1)"

# weight for merge2-reverse
Flow('weight5','lweight-reverse','math output="%s"| cut max1=0.25 | scale dscale=10'
    % f)
Flow('hweight','lweight-reverse weight5','math output=100 | cut min1=0.25 | add ${
    SOURCES[1]} | scale dscale=100')
Result('hweight', 'byte gainpanel=all bar=bar.rsf allpos=y | grey3 color=j scalebar=y
     title="Hires Weight" mean=y frame1=600 frame2=10 frame3=30')

Flow('merge1-reverse','hires-warp-diff-reverse hires-warp %s hweight lweight-reverse'
     % rect_warp,
     '''
     conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}
     ''')

# Difference between high-resolution and merged image
Result('merge1-reverse','bandpass fhi=250 | window j1=3 |grey title="Difference
    between Merged and Hires" clip=3.7011')
# Final merged image
Flow('merge','hires-warp merge1-reverse','add ${SOURCES[1]}')
Result('merge','grey title="Merged" ')

# 0.125s - 0.45s. reappears at 0.625s - 1.15s
#Result('merge3','merge','byte gainpanel=all | grey3 title="Merged" frame1=1500
    frame2=10 frame3=30 ')
#Result('merge3','merge','byte gainpanel=all | grey3 title="Merged" frame1=1500
    frame2=40 frame3=40 ')
#Result('merge3','merge','put o1=0 o2=0 o3=0 d2=.386929 unit2=kft d3=.122744 unit3=
    kft | byte gainpanel=all | grey3 title="Merged" frame1=1500 frame2=10 frame3=30')
Result('merge3','merge','put o1=0 o2=0 o3=0 d2=.386929 unit2= d3=.122744 unit3= |
    byte gainpanel=all | grey3 title="Merged" frame1=1500 frame2=10 frame3=30')

Result('merge-cut','merge','window min1=0.5 max1=1.0 | grey title="Merged"')
Result('hires-cut','hires4','window min1=0.5 max1=1.0 | grey title="Hires"')
Result('legacy-cut','legacy4','window min1=0.5 max1=1.0 | grey title="Hires"')
Result('merge4','merge','transp plane=13 |  grey title="Merged" frame1=1500 frame2=10
     frame3=30 clip=3.7011')

Result('hires-despike4','hires-despike','transp plane=13 |  grey title="Hires" frame1
    =1500 frame2=10 frame3=30 ')
Result('legacy-filt4','legacyfilt','transp plane=13 |  grey title="Legacy" frame1
    =1500 frame2=10 frame3=30 ')

# Display the spectra
Flow('legacy4-spec4','legacy4','spectra all=y')
Flow('hires-spec4','hires-warp','spectra all=y')
Flow('merge-spec4','merge','spectra all=y')

# merge Final Spectra
```

```
# nspectra22-reverse
Result('nspectra2','legacy4-spec4 hires-spec4 merge-spec4',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=600 |
       scale axis=1 | graph title="Spectra "
       label2="Amplitude" unit2=""
       ''')

Result('nspectra3','legacy4-spec4 hires-spec4 merge-spec4',
       '''
       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=300 |
       scale axis=1 | graph title="Spectra "
       label2="Amplitude" unit2="" dash=2,1,0
       ''')

End()
```

Listing 13: chapter-merge/pcable2/radius.py

```
from rsf.proj import *

def radius(high, low,               # initial high-resolution and legacy images
           niter,                   # number of corrections
           c,                       # 'step length' for radius corrections. Can be type int or float for constant c
                                    # or type array for changing c.
           bias=-15, clip=30,       # bias and clip for display
           rect1=40, rect2=80, rect3=1,     # radius for local frequency calculation
           maxrad=1000,             # maximum allowed radius
           theor=True,              # use theoretical smoothing radius
           scale=9,                 # scale for theoretical smoothing radius
           initial=10,              # initial value for contant smoothing radius
           minval=0,
           maxval=25,
           titlehigh="Hires",
           titlelow="Legacy",
           ind = ''):

    if type(c) is float or type(c) is int:
        c = [c]*niter

    def seisplot(name):
        return 'grey title="%s"'%name

    locfreq = '''iphase order=10 rect1=%d rect2=%d rect3=%d hertz=y complex=y |
                 put label="Frequency" unit=Hz'''%(rect1,rect2,rect3)

    def locfreqplot(name):
        return 'grey mean=y color=j scalebar=y title="%s" '%name

    freqdif = 'add scale=-1,1 ${SOURCES[1]} | put label=Frequency'
    def freqdifplot3(num):
        return '''byte gainpanel=all bar=bar.rsf allpos=y |
                  grey3 frame1=400 frame2=40 frame3=30 allpos=y color=j scalebar=y mean=y
                  title="Difference in Local Frequenies %s"''' %(num)

    def freqdifplot(num):
        return '''grey allpos=y color=j scalebar=y mean=y title="Difference in Local Frequencies %s"
```

```
                clip=%d bias=%d minval=%d maxval=%d''' %(num,clip,bias,minval,
    maxval)

    specplot = '''cat axis=2 ${SOURCES[1]} |
                scale axis=1 | window max1=180 |
                graph title="Normalized Spectra" label2="Amplitude" unit2=""'''

    def rectplot(name):
        return 'grey color=j mean=y title="%s" scalebar=y barlabel=Radius barunit=
    samples'%name

    smooth = 'nsmooth1 rect=${SOURCES[1]}'

    #Result(high, seisplot(titlehigh))
    #Result(low, seisplot(titlelow))
    def n(name):
            return name+str(ind)
    high_freq = n('high-freq')
    low_freq = n('low-freq')

    Flow(high_freq,high,locfreq)
    Result(high_freq,locfreqplot('%s Local Frequency'%titlehigh))

    Flow(low_freq,low,locfreq)
    Result(low_freq,locfreqplot('%s Local Frequency'%titlelow))

    freqdif_init = n('freqdif-init')

    Flow(freqdif_init,[low_freq, high_freq],freqdif)
    Result(freqdif_init,freqdifplot3(''))

    # initial smoothing radius
    rect = n('rect0')
    if (theor):
        from math import pi
        Flow(rect,[low_freq, high_freq],'''math f1=${SOURCES[1]}
              output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" '''%(scale,2*pi*0.001)
    )
    else:
        Flow(rect,low_freq,'math output=%f'%initial)

    Result(rect,rectplot("Smoothing Radius 0"))

    high_smooth = n('high-smooth0')
    Flow(high_smooth,[high,rect],smooth)
    Result(high_smooth, seisplot("%s Smooth 0"%titlehigh))

    high_spec = n('high-spec')
    low_spec = n('low-spec')
    Flow(high_spec,high,'spectra all=y')
    Flow(low_spec,low,'spectra all=y')

    high_ss = n('high-smooth-spec0')
    nspec = n('nspectra-diff')
    Flow(high_ss,high_smooth,'spectra all=y')
    Result(nspec,[high_spec, low_spec],specplot)
    Result(high_ss,[high_ss, low_spec],specplot)

    high_sf = n('high-smooth-freq0')
    Flow(high_sf,high_smooth,locfreq)
    Result(high_sf,locfreqplot("%s Local Frequency Smoothed %d" %(titlehigh,0)))
```

```
98
    freqdif_filt = n('freqdif-filt0')
100 Flow(freqdif_filt,[low_freq, high_sf],freqdif)
    #Result('freqdif-filt0',freqdifplot('0'))
102 Result(freqdif_filt,freqdifplot3('0'))

104 prog=Program('radius.c')
    rad = str(prog[0])
106 for i in range(1, niter+1):

108     j = i-1
        rect = n('rect%i'%i)
110     rectj = n('rect%i'%j)
        fdfj = n('freqdif-filt%i'%j)
112
        Flow(rect,[rectj,fdfj,rad],'./${SOURCES[2]} freq=${SOURCES[1]} c=%f'%c[j])
114     Result(rect,rectplot("Smoothing Radius %d")%i)

116     hs = n('high-smooth%i'%i)
        hss = n('high-smooth-spec%i'%i)
118     hsf = n('high-smooth-freq%i'%i)
        fdf = n('freqdif-filt%i'%i)
120
        Flow(hs,[high,rect],smooth)
122     Result(hs, seisplot('%s Smooth %d'%(titlehigh,i)))

124     Flow(hss,hs,'spectra all=y')
        Result(hss,[hss,low_spec],specplot)
126
        Flow(hsf,hs,locfreq)
128     Result(hsf,locfreqplot('%s Local Frequency Smoothed %d'%(titlehigh,i)))

130     Flow(fdf,[low_freq, hsf],freqdif)
        Result(fdf,freqdifplot(str(i)))
```

Listing 14: chapter-merge/pcable2/SConstruct

```
/* smoothing radius (min = 1) */
2
#include <rsf.h>
4 #include <math.h>

6 int main (int argc, char* argv[])
{
8     int n1, n1f, n2, n2f, i, n12, n12f;
      float *rect, *fr, maxrad, c, *rad;
10    sf_file in, out, freq;

12    sf_init (argc,argv);

14    in = sf_input("in");
      freq = sf_input("freq");
16    out = sf_output("out");

18    if (!sf_histint(in,"n1",&n1)) sf_error("No n1= in input.");
      if (!sf_histint(freq,"n1",&n1f)) sf_error("No n1= in frequency difference.");
20
      n2 = sf_leftsize(in,1);
22    n2f = sf_leftsize(freq,1);
```

```
    n12 = n1*n2;
    n12f = n1f*n2f;

    if (n1 != n1f) sf_error("Need matching n1");
    if (n2 != n2f) sf_error("Need matching n2");

    if (!sf_getfloat("c",&c)) c=1.;
    if (!sf_getfloat("maxrad",&maxrad)) maxrad=1000.;

    rect = sf_floatalloc(n12);
    sf_floatread(rect,n12,in);

    fr = sf_floatalloc(n12f);
    sf_floatread(fr,n12,freq);

    rad = sf_floatalloc(n12);

    for (i=0; i < n12; i++) {

        /* update radius */
      rad[i] = rect[i]+c*fr[i];

        /* set constraint conditions: [1, maxrad] */
        if (rad[i] > maxrad)
                rad[i] = maxrad;
        if (rad[i] < 1.0)
                rad[i] = 1.0;
    }

    sf_floatwrite(rad,n12,out);
    exit(0);
}
```

Chapter 4

Listing 15: chapter-mighes/sigsbee/SConstruct

```
from rsf.proj import *
from radius import radius

maxWin=80
pclip=99
rad=5

refl = "ref-true.rsf"
image0 = "image0.rsf"
image1 = "image1.rsf"
Result('mod','ref-true','grey title="Reflectivity model" screenratio=0.6')
Flow('env0','image0','envelope')
Flow('env1','image1','envelope')
Result('env0','grey title=Model scalebar=y color=j')
Result('env1','grey title=Model scalebar=y color=j')
Result('image1','grey screenratio=0.6 bias=0 clip=0.0004 title="Second migration"')
Result('image0','grey screenratio=0.6 title="First migration"')

Result('vel-migration','grey title=Model scalebar=y    color=j screenratio=0.5 bias=3
     clip=2 barlabel=Velocity barunit="km/s" title="Migration velocity model"')
```

```
Result('vel-stratigraphy','grey title=Model scalebar=y color=j screenratio=0.5
    wanttitle=n')

Flow('a0','env0 env1','divn den=${SOURCES[1]} niter=500 rect1=3 rect2=5')

Result('a0','grey title="Amplitude operator" color=j scalebar=y barlabel="Weight"
    barunit= screenratio=0.6')
Result('rect10b','rect10','grey color=j mean=y title="Frequency operator" scalebar=y
    barlabel=Radius barunit=samples screenratio=0.6')


# matching (sfdivn)
Flow('a0image1','image1 a0','math w=${SOURCES[1]} output="input*w"')
Result('a0image1','grey title=a0image1')

Flow('a0image0','image0 a0','math w=${SOURCES[1]} output="input/w"')
Result('a0image0','grey title=a0image1')

# frequency balancing
niter=10
radius('image0','a0image1', niter, 0.20, bias=-3, clip=15, rect1=rad, rect2=rad,
       maxval=20, minval=-7, theor=False, initial=1,
       titlehigh="First Migration", titlelow="Second Migration")

Flow('env0b','high-smooth%i'%niter,'envelope')
Flow('env1b','image1','envelope')

Flow('a1','env0b env1b','divn den=${SOURCES[1]} niter=500 rect1=3 rect2=5')


Flow('mod0','image0 a0','math w=${SOURCES[1]} output="input*(abs(w))^(0.1)"')
Result('mod%i'%(0),'grey title="mod0"')

def forward(mod,dat):
    Flow(dat,'%s rect%d'%(mod,niter),'nsmooth rect1=${SOURCES[1]}')

def backward(dat,mod):
    Flow(mod,'%s rect%d'%(dat,niter),'nsmooth rect1=${SOURCES[1]}')

# Shaping regularization
# mod_{n+1} = S[B[d] + m_{n} - BF[m_{n}]]
n=10

prog=Program('freqfilt.c')
Flow('add-spec','image0-spec image1-spec','math kir2=${SOURCES[1]} output="2*input-
    kir2"')
Flow('filt','add-spec','scale axis=1 | mask min=0.05 | dd type=float | smooth rect1
    =10')
Result('filt','graph')
Result('add-spec','graph')
for it in range(n):
    mod = 'mod%d' % it
    dat = 'dat%d' % it
    forward(mod,dat)
    bak = 'bak%d' % it
    backward(dat,bak)
    new = 'mod%d' % (it+1)
    Flow(new,[mod,'mod0',bak],'''
        add scale=1,1,-1 ${SOURCES[1:3]} |
        bandpass fhi=32.8084''')

```

```
Flow('modb0','image0','cp')
for it in range(n):
    mod = 'modb%d' % it
    dat = 'datb%d' % it
    forward(mod,dat)
    bak = 'bakb%d' % it
    backward(dat,bak)
    new = 'modb%d' % (it+1)
    Flow(new,[mod,'modb0',bak],'''
        add scale=1,1,-1 ${SOURCES[1:3]} |
        bandpass fhi=32.8084''')
Result('modb%i'%(n-1),'grey title="Corrected migrated image"')


Flow('migdec-shap','mod%i a0'%(n-1),'math w=${SOURCES[1]} output="input*(abs(w))
    ^(0.1)"')

Result('migdec-shap','grey title="Corrected migration" screenratio=0.6')
Result('mod%i'%(n-1),'grey title="Corrected migrated image"')

##############################################################################
Result('mod-w1','ref-true',      'window min1=2 max1=3 min2=10 max2=16 | grey title=
    model')
Result('migdec-w1','migdec-shap','window min1=2 max1=3 min2=10 max2=16 | grey title=
    migdec')
Result('image0-w1','image0',     'window min1=2 max1=3 min2=10 max2=16 | grey title=
    image0')

Result('mod-w2','ref-true',      'window min1=2 max1=2.5 min2=22 max2=26 | grey
    screenratio=0.5 screenht=7 title=Model')
Result('migdec-w2','migdec-shap','window min1=2 max1=2.5 min2=22 max2=26 | grey
    screenratio=0.5 screenht=7 title="Corrected migration"')
Result('image0-w2','image0',     'window min1=2 max1=2.5 min2=22 max2=26 | grey
    screenratio=0.5 screenht=7 title="Conventional migration"')

Result('mod-w3','ref-true',      'window min1=2 max1=4 min2=18 max2=20 | grey
    screenratio=1.15 screenht=10 title=Model')
Result('migdec-w3','migdec-shap','window min1=2 max1=4 min2=18 max2=20 | grey
    screenratio=1.15 screenht=10 title="Corrected migration"')
Result('image0-w3','image0',     'window min1=2 max1=4 min2=18 max2=20 | grey
    screenratio=1.15 screenht=10 title="Conventional migration"')

Result('mod-w4','ref-true',      'window min1=2 max1=3 min2=18 max2=20 | grey title=
    model')
Result('migdec-w4','migdec-shap','window min1=2 max1=3 min2=18 max2=20 | grey title=
    migdec')
Result('image0-w4','image0',     'window min1=2 max1=3 min2=18 max2=20 | grey title=
    image0')

Result('mod-w5','ref-true',      'window min1=4 max1=8 min2=4 max2=5 | grey title=
    model')
Result('migdec-w5','migdec-shap','window min1=4 max1=8 min2=4 max2=5 | grey title=
    migdec')
Result('image0-w5','image0',     'window min1=4 max1=8 min2=4 max2=5 | grey title=
    image0')

Result('mod-w6','ref-true',      'window min1=5 max1=6 min2=14 max2=15 | grey title=
    model')
Result('migdec-w6','migdec-shap','window min1=5 max1=6 min2=14 max2=15 | grey title=
    migdec')
```

```
Result('image0-w6','image0',     'window min1=5 max1=6 min2=14 max2=15 | grey title=
    image0')

End()
```

Listing 16: chapter-mighes/sigsbee/radius.py

```
from rsf.proj import *

def radius(high, low,               # initial high-resolution and legacy images
           niter,                   # number of corrections
           c,                       # 'step length' for radius corrections. Can be
    type int or float for constant c
                                    # or type array for changing c.
           bias=-15, clip=30,       # bias and clip for display
           rect1=40, rect2=80,      # radius for local frequency calculation
           maxrad=1000,             # maximum allowed radius
           theor=True,              # use theoretical smoothing radius
           scale=9,                 # scale for theoretical smoothing radius
           initial=10,              # initial value for contant smoothing radius
           minval=0,
           maxval=25,
           titlehigh="Hires",
           titlelow="Legacy"):

    if type(c) is float or type(c) is int:
        c = [c]*niter

    def seisplot(name):
        return 'grey title="%s"'%name

    locfreq = '''iphase order=10 rect1=%d rect2=%d hertz=y complex=y |
                 put label="Frequency" unit=Hz'''%(rect1,rect2)

    def locfreqplot(name):
        return 'grey mean=y color=J scalebar=y title="%s" '%name

    freqdif = 'add scale=-1,1 ${SOURCES[1]} | put label=Frequency'

    def freqdifplot(num):
        return '''grey allpos=y color=j scalebar=y mean=y title="Difference in Local
    Frequencies %s"
                  clip=%d bias=%d minval=%d maxval=%d''' %(num,clip,bias,minval,
    maxval)

    specplot = '''cat axis=2 ${SOURCES[1]} |
                  scale axis=1 |
                  graph title="Normalized Spectra" label2="Amplitude" unit2=""'''

    def rectplot(name):
        return 'grey color=j mean=y title="%s" scalebar=y barlabel=Radius barunit=
    samples'%name

    smooth = 'nsmooth1 rect=${SOURCES[1]}'
    #smooth = 'nsmooth rect1=${SOURCES[1]} rect2=${SOURCES[1]}'

    #Result(high, seisplot(titlehigh))
    #Result(low, seisplot(titlelow))

    Flow('high-freq',high,locfreq)
    Result('high-freq',locfreqplot('%s Local Frequency'%titlehigh))
```

```
51
    Flow('low-freq',low,locfreq)
53  Result('low-freq',locfreqplot('%s Local Frequency'%titlelow))

55  Flow('freqdif','low-freq high-freq',freqdif)
    Result('freqdif',freqdifplot(''))
57
    # initial smoothing radius
59  if (theor):
        from math import pi
61      Flow('rect0','low-freq high-freq','''math f1=${SOURCES[1]}
              output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" '''%(scale,2*pi*0.001)
    )
63  else:
        Flow('rect0','low-freq','math output=%f'%initial)
65
    Result('rect0',rectplot("Smoothing Radius 0"))
67
    Flow('high-smooth0','%s rect0' % high,smooth)
69  Result('high-smooth0', seisplot("%s Smooth 0"%titlehigh))

71  Flow('high-spec',high,'spectra all=y')
    Flow('low-spec',low,'spectra all=y')
73  Flow('high-smooth-spec0','high-smooth0','spectra all=y')
    Result('nspectra','high-spec low-spec',specplot)
75  Result('high-smooth-spec0','high-smooth-spec0 low-spec',specplot)

77  Flow('high-smooth-freq0','high-smooth0',locfreq)
    Result('high-smooth-freq0',locfreqplot("%s Local Frequency Smoothed %d" %(
    titlehigh,0)))
79
    Flow('freqdif-filt0','low-freq high-smooth-freq0',freqdif)
81  Result('freqdif-filt0',freqdifplot('0'))

83  prog=Program('radius.c')
    for i in range(1, niter+1):
85      j = i-1
        Flow('rect%d'%i,'rect%d freqdif-filt%d %s'%(j,j,prog[0]),'./${SOURCES[2]}
    freq=${SOURCES[1]} c=%f maxrad=%f'%(c[j],maxrad))
87      Result('rect%d'%i,rectplot("Smoothing Radius %d")%i)

89      Flow('high-smooth%d'%i,'%s rect%d'%(high,i),smooth)
        Result('high-smooth%d'%i, seisplot('%s Smooth %d'%(titlehigh,i)))
91
        Flow('high-smooth-spec%d'%i,'high-smooth%d'%i,'spectra all=y')
93      Result('high-smooth-spec%d'%i,'high-smooth-spec%d low-spec'%i,specplot)

95      Flow('high-smooth-freq%d'%i,'high-smooth%d'%i,locfreq)
        Result('high-smooth-freq%d'%i,locfreqplot('%s Local Frequency Smoothed %d'%(
    titlehigh,i)))
97
        Flow('freqdif-filt%d'%i,'low-freq high-smooth-freq%d'%i,freqdif)
99      Result('freqdif-filt%d'%i,freqdifplot(str(i)))
```

Listing 17: chapter-mighes/sigsbee/radius.c

```
/* smoothing radius (min = 1) */
2
#include <rsf.h>
4 #include <math.h>
```

```
int main (int argc, char* argv[])
{
    int n1, n1f, n2, n2f, i, n12, n12f;
    float *rect, *fr, maxrad, c, *rad;
    sf_file in, out, freq;

    sf_init (argc,argv);

    in = sf_input("in");
    freq = sf_input("freq");
    out = sf_output("out");

    if (!sf_histint(in,"n1",&n1)) sf_error("No n1= in input.");
    if (!sf_histint(freq,"n1",&n1f)) sf_error("No n1= in frequency difference.");

    n2 = sf_leftsize(in,1);
    n2f = sf_leftsize(freq,1);

    n12 = n1*n2;
    n12f = n1f*n2f;

    if (n1 != n1f) sf_error("Need matching n1");
    if (n2 != n2f) sf_error("Need matching n2");

    if (!sf_getfloat("c",&c)) c=1.;
    if (!sf_getfloat("maxrad",&maxrad)) maxrad=1000.;

    rect = sf_floatalloc(n12);
    sf_floatread(rect,n12,in);

    fr = sf_floatalloc(n12f);
    sf_floatread(fr,n12,freq);

    rad = sf_floatalloc(n12);

    for (i=0; i < n12; i++) {

        /* update radius */
      rad[i] = rect[i]+c*fr[i];

        /* set constraint conditions: [1, maxrad] */
        if (rad[i] > maxrad)
                rad[i] = maxrad;
        if (rad[i] < 1.0)
                rad[i] = 1.0;
    }

    sf_floatwrite(rad,n12,out);
    exit(0);
}
```

Listing 18: chapter-mighes/triop/triOp.py

```
#!/usr/bin/env python2

from solve import *

from numpy import empty, linspace, concatenate, pi, pad
from numpy.linalg import pinv, norm
from scipy.signal import butter, filtfilt, freqz
from scipy import fft, arange
```

```
import matplotlib.pyplot as plt

def main():
    print('Triange smoothing')
    print('functions include...')
    print('    foldConv(signal, filt)')
    print('    foldConvOp(triOper, sl)')
    print('    triOper(rect)')
    print('    lowPassFilter(data, highcut, fs, order, plot)')
    print('    plotSpec(y,Fs,col="red")')

def foldConv(signal,filt):
    """ 1D convolution with folding """

    # length of arrays and center
    fl = len(filt)
    sl = len(signal)
    center = int(len(filt)/2)

    # initialize convolved signal
    conv = [0]*sl

    # filter must be smaller than signal
    if fl > sl:
        raise ValueError, "Filter longer than signal"

    # convolve
    for i in range(0,sl):
        # center
        conv[i] += filt[center]*signal[i]

        # left
        for j in range(0,center):
            idx = abs(i-center+j)
            conv[i] += filt[j]*signal[idx]

        # right
        for j in range(center + 1, fl):
            idx = i+j-center
            if idx >= sl:
                idx = (sl-1) - (idx - (sl-1))
            conv[i] += filt[j]*signal[idx]

    return conv

def foldConvOp(triOper, sl, plot=False):
    """ 1D convolution with folding """

    # length of arrays and center
    fl = len(triOper)
    center = int(fl/2)

    # initialize convolved signal
    conv = [0]*sl
    mat = empty([sl,sl])

    # filter must be smaller than signal
    if fl > sl:
        raise ValueError, "Filter longer than signal"

    # convolve
```

```
    for i in range(0,sl):
        # center
        mat[i,i] += triOper[center]

        # left
        for j in range(0,center):
            idx = abs(i-center+j)
            mat[i,idx] += triOper[j]

        # right
        for j in range(center + 1, fl):
            idx = i+j-center
            if idx >= sl:
                idx = (sl-1) - (idx - (sl-1))
            mat[i,idx] += triOper[j]

    if plot: # plot the matrix
        plt.imshow(mat)
        plt.title("Forward operator")
        plt.show()

    return mat

def triOper(rect):
    """ create triangle smoothing operator """
    if rect < 1:
        raise ValueError, "Rect minimum size 1"

    # create triangle operator
    t = linspace(1,rect-1,rect-1)
    trian = concatenate((t,[rect],t[::-1]),axis=0)
    # normalize and return
    return trian/norm(trian)

def lowPassFilter(data, highcut, fs, order=5, plot=True):
    nyq = 0.5 * fs
    high = highcut / nyq

    b, a = butter(order, high, btype='lowpass')
    y = filtfilt(b, a, data) # zero phase
    w, h = freqz(b, a)

    if plot:
        plt.plot(w*fs/(2*pi), abs(h), 'b')
        plt.ylabel('Amplitude')
        plt.xlabel('Frequency (Hz)')
        plt.title('Filter transfer function')
        plt.xlim([-5,120])
    return y

def plotSpec(y,Fs,col="red"):
    n = 5000
    y = pad(y, n, 'constant', constant_values=0) # increase resolution
    n = len(y)
    k = arange(n)
    T = n/Fs
    frq = k/T
    frq = frq[range(n/2)]

    Y = fft(y)/n
    Y = Y[range(n/2)]
```

```
131
    plt.plot(frq,abs(Y)/max(abs(Y)),col)
133     plt.xlabel('Frequency (Hz)')
    plt.ylabel('Amplitude')
135     plt.title('Frequency Spectra')

137
if __name__ == "__main__":
139     main()
```

Listing 19: chapter-mighes/triop/trfu.py

```
1 #! /usr/bin/env python2

3 from triOp import *
import numpy as np
5 import matplotlib.pyplot as plt

7 # create an impulse
spike = np.zeros(100)
9 spike[50] = 1

11 # smooth spike to get impulse respnse of filter
rect = 10
13 tri = triOper(10)
smoo = foldConv(spike,tri)
15
# create filter
17 a = 1            # filter a term
fs = 1/0.001    # sampling frequency
19
# forward
21 w, h = freqz(smoo, a)

23 # inverse
w2, h2 = freqz(a, smoo)
25


27 # plot everything
fig, ax1 = plt.subplots()
29 ax1.set_title('Transfer functions, rect=%i samples, fs=%i Hz'%(rect,fs))
#ax1.set_title('Transfer functions')
31 ax1.plot(w*fs/(2*pi),  abs(h ), 'b')
ax1.set_ylim([0,4])
33 ax1.set_xlim([0,500])
ax1.set_xlabel('Frequency (Hz)')
35 ax1.set_ylabel('Forward', color='b')
ax1.tick_params('y', colors='b')
37 ax2 = ax1.twinx()
ax2.plot(w2*fs/(2*pi), abs(h2), 'r')
39 ax2.set_ylim([0,400])
ax2.set_ylabel('Inverse', color='r')
41 ax2.tick_params('y', colors='r')
fig.tight_layout()
43 #plt.show()
plt.savefig('Fig/tf.pdf', format='pdf', bbox_inches='tight')
```

Listing 20: chapter-mighes/triop/solve.py

```
1 #!/usr/bin/env python2
```

```
from triOp import *
from numpy import zeros, matmul, identity
from numpy.linalg import norm, matrix_rank
def main():
    print('Generic systems solver')
    print('functions include...')
    print('    conjGrad(A, b, tol, maxItr)')
    print('    irls(A, b, tol, maxItr, nliter)')
    print('    lsConjGrad(A, b, tol, maxItr)')
    print('    regLsConjGrad(A, b, alpha, tol, maxItr)')

def conjGrad(A, b, tol=1e-8, maxItr=1500, x=0):
    '''
    conjugate gradients solver
    does NOT check if A is SPD
    '''
    n = len(b)

    if norm(x) == 0:
        x = zeros(n)

    r = b - A.dot(x)
    p = r

    res0 = norm(r)
    for i in range(1,maxItr):
        Ap = A.dot(p)
        pAp = p.dot(Ap)
        alpha = r.dot(p)/pAp

        x += alpha*p
        r -= alpha*Ap

        if norm(r)/res0 <= tol:
            break

        beta = -Ap.dot(r)/pAp
        p = r + beta * p
    return x

def irls(A, b, nrm=1, tol=1e-8, maxItr=1500, nliter=10):
    # find L2 solution
    At = A.transpose()
    A = matmul(A,At)
    b = At.dot(b)
    psinv = conjGrad(A,b,tol,maxItr)

    # find nrm solution
    for i in range (1,nliter):
        print("IRLS nliter # %i"%i)
        r = b - A.dot(psinv)
        R = zeros((len(b),len(b)))
        for j in range(1,len(r)):
            R[j,j] = abs(r[i]**(nrm-2));

        A = matmul(matmul(At,R),A)
        b = matmul(At,R).dot(b)
        psinv = conjGrad(A,b,tol,maxItr)

    return psinv
```

```
63
def lsConjGrad(A, b, tol=1e-8, maxItr=1500):
65     At = A.transpose()
       Als = matmul(At,A)
67     bls = At.dot(b)
       print ("Matrix rank: %i" %matrix_rank(Als))
69     print ("Full rank: %i" %len(bls))
       return conjGrad(Als, bls, tol, maxItr)
71
def regLsConjGrad(A, b, alpha=0.5, tol=1e-8, maxItr=1500):
73     At = A.transpose()
       Als = matmul(At,A) + alpha*identity(len(b))
75     bls = At.dot(b)
       return conjGrad(Als, bls, tol, maxItr)
77
def shaping(A, b, alpha=0.5, tol=1e-8, maxItr=1500, nliter=5, sz=30, tp="bp"):
79     At = A.transpose()
       Als = matmul(At,A) + alpha*identity(len(b))
81     bls = At.dot(b)

83     x = zeros(len(bls))

85     if tp=="tri":
           tri=triOper(sz)
87         for i in range(0, nliter):
               x = foldConv(x, tri)
89             x = conjGrad(Als, bls, tol, maxItr, x)
               print("Iter: %i"%i)
91
       elif tp=="bp":
93         for i in range(0, nliter):
               x = lowPassFilter(x, sz, 1/0.001, plot=True)
95             x = conjGrad(Als, bls, tol, maxItr, x)
               print("Iter: %i"%i)
97
           #plt.show()
99
       return x
101
if __name__ == "__main__":
103    main()
```

Bibliography

Al-Inaizi, S., R. Bridle, and N. Nakhla, 2004, Comparison of pre and post-stack super-merge of ten 3D blocks: Presented at the SEG Technical Program Expanded Abstracts 2004, Society of Exploration Geophysicists.

Aoki, N., and G. T. Schuster, 2009, Fast least-squares migration with a deblurring filter: GEOPHYSICS, **74**, WCA83–WCA93.

Bader, S., X. Wu, and S. Fomel, 2018, Missing log data interpolation and semiautomatic seismic well ties using data matching techniques: Interpretation. (Submitted).

Carter, D., and S. Pambayuning, 2009, Extended bandwidth by a frequency domain merge of two 3D seismic volumes: The Leading Edge, **28**, 386–386.

Casasanta, L., F. Perrone, G. Roberts, A. Ratcliffe, K. Purcell, A. Jafargandomi, and G. Poole, 2017, Applications of single-iteration Kirchhoff least-squares migration: Presented at the SEG Technical Program Expanded Abstracts 2017, Society of Exploration Geophysicists.

Claerbout, J. F., 1992, Earth Soundings Analysis: Processing Versus Inversion: Blackwell Scientific Publications.

Dai, W., and G. T. Schuster, 2013, Plane-wave least-squares reverse-time migration: GEOPHYSICS, **78**, S165–S177.

Dong, S., J. Cai, M. Guo, S. Suh, Z. Zhang, B. Wang, and Z. Li, 2012, Least-squares reverse time migration: towards true amplitude imaging and improving the resolution: Presented at the SEG Technical Program Expanded Abstracts 2012, Society of Exploration Geophysicists.

Dutta, G., Y. Huang, W. Dai, X. Wang, and G. T. Schuster, 2014, Making the most out of the least (squares migration): 84th Annual International Meeting, SEG, Expanded Abstracts, 4405–4410.

Fomel, S., 2007a, Local seismic attributes: Geophysics, **72**, A29–A33.

———, 2007b, Shaping regularization in geophysical-estimation problems: GEOPHYSICS, **72**, R29–R36.

Fomel, S., M. Backus, K. Fouad, B. Hardage, and G. Winters, 2005, A multistep approach to multicomponent seismic image registration with application to a West Texas carbonate reservoir study: 75th Ann. Internat. Mtg, Soc. of Expl. Geophys., 1018–1021.

Fomel, S., and M. M. Backus, 2003a, Multicomponent seismic data registration by least squares: Presented at the SEG Technical Program Expanded Abstracts, Society of Exploration Geophysicists.

———, 2003b, Multicomponent seismic data registration by least squares: 73rd Annual International Meeting, Soc. of Expl. Geophys., 701–784.

Fomel, S., and L. Jin, 2009, Time-lapse image registration using the local similarity attribute: Geophysics, **74**, A7–A11.

Fomel, S., P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, 2013, Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments: Journal of Open Research Software, **1**, e8.

Gholamy, A., and V. Kreinovich, 2014, Why ricker wavelets are successful in processing seismic data: Towards a theoretical explanation: Presented at the 2014 IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES), IEEE.

Greer, S., and S. Fomel, 2017a, Balancing local frequency content in seismic data

using non-stationary smoothing: SEG Technical Program Expanded Abstracts, Society of Exploration Geophysicists, 4278 – 4282.

———, 2017b, Matching and merging high-resolution and legacy seismic images: SEG Technical Program Expanded Abstracts, Society of Exploration Geophysicists, 5933 – 5937.

———, 2018, Matching and merging high-resolution and legacy seismic images: GEOPHYSICS, **83**, V115–V122.

Greer, S., Z. Xue, and S. Fomel, 2018, Improving migration resolution by approximating the least-squares hessian using non-stationary amplitude and frequency matching: Presented at the 88th Ann. Internat. Mtg, Society of Exploration Geophysicists. (Submitted).

Guitton, A., 2004, Amplitude and kinematic corrections of migrated images for nonunitary imaging operators: GEOPHYSICS, **69**, 1017–1024.

———, 2017, Fast 3D least-squares RTM by preconditioning with nonstationary matching filters: Presented at the SEG Technical Program Expanded Abstracts 2017, Society of Exploration Geophysicists.

Hale, D., 2013, Dynamic warping of seismic images: GEOPHYSICS, **78**, S105–S115.

Hardage, B. A., M. V. DeAngelo, P. E. Murray, and D. Sava, 2011, Multicomponent seismic technology: Society of Exploration Geophysicists.

Herrera, R. H., S. Fomel, and M. van der Baan, 2014, Automatic approaches for seismic to well tying: Interpretation, **2**, SD9–SD17.

Herrera, R. H., and M. van der Baan, 2012, Guided seismic-to-well tying based on dynamic time warping: 82nd Annual International Meeting, Society of Exploration Geophysicists, 1–5.

Hestenes, M. R., and E. Stiefel, 1952, Methods of Conjugate Gradients for Solving

Linear Systems: Journal of Research of the National Bureau of Standards, **49**, 409–436.

Hou, J., and W. Symes, 2016, Accelerating least squares migration with weighted conjugate gradient iteration: Presented at the 78th EAGE Conference and Exhibition 2016, EAGE Publications BV.

Hou, J., and W. W. Symes, 2015, An approximate inverse to the extended born modeling operator: GEOPHYSICS, **80**, R331–R349.

Hu, J., and G. T. Schuster, 1998, Migration deconvolution: Presented at the Mathematical Methods in Geophysical Imaging V, SPIE.

Hu, J., G. T. Schuster, and P. A. Valasek, 2001, Poststack migration deconvolution: GEOPHYSICS, **66**, 939–952.

Kuehl, H., and M. D. Sacchi, 2003, Least-squares wave-equation migration for AVP/AVA inversion: Geophysics, **68**, 262–273.

Liu, Y., and S. Fomel, 2012, Seismic data analysis using local time-frequency decomposition: Geophysical Prospecting, **61**, 516–525.

Lumley, D., D. C. Adams, M. Meadows, S. Cole, and R. Wright, 2003, 4D seismic data processing issues and examples: 73rd Annual International Meeting, Society of Exploration Geophysicists, 1394–1397.

Meckel, T., N. L. Bangs, and T. Hess, 2017, 3D multichannel seismic field data from the San Luis Pass region off Galveston, Texas, acquired by the R/V Brooks-McCall in 2013 (BM1310).

Meckel, T. A., and F. J. Mulcahy, 2016, Use of novel high-resolution 3D marine seismic technology to evaluate Quaternary fluvial valley development and geologic controls on shallow gas distribution, inner shelf, Gulf of Mexico: Interpretation, **4**, SC35–SC49.

Mohan, T. R. M., C. B. Yadava, S. Kumar, K. K. Mishra, and K. Niyogi, 2007, Prestack merging of land 3D vintages — a case history from Kavery Basin, India: Presented at the SEG Technical Program Expanded Abstracts 2007, Society of Exploration Geophysicists.

Müller, M., 2007, Dynamic time warping, *in* Information Retrieval for Music and Motion: Springer Berlin Heidelberg, 69–84.

Paffenholz, J., B. McLain, J. Zaske, and P. Keliher, 2002, Subsalt multiple attenuation and imaging: Observations from the Sigsbee2B synthetic dataset: 72nd Annual International Meeting, SEG, Expanded Abstracts, 2122–2125.

Petersen, C. J., S. Buenz, S. Hustoft, J. Mienert, and D. Klaeschen, 2010, High-resolution P-Cable 3D seismic imaging of gas chimney structures in gas hydrated sediments of an Arctic sediment drift: Marine and Petroleum Geology, **27**, 19811994.

Phillips, M., and S. Fomel, 2016, Seismic time-lapse image registration using amplitude-adjusted plane-wave destruction: SEG Technical Program Expanded Abstracts, Society of Exploration Geophysicists, 5473–5478.

Rickett, J. E., 2003, Illumination-based normalization for wave-equation depth migration: GEOPHYSICS, **68**, 1371–1379.

Ross, C. P., and M. S. Altan, 1997, Time-lapse seismic monitoring: Some shortcomings in nonuniform processing: The Leading Edge, **16**, 931–937.

Sacchi, M. D., J. Wang, and H. Kuehl, 2007, Estimation of the diagonal of the migration blurring kernel through a stochastic approximation: Presented at the SEG Technical Program Expanded Abstracts 2007, Society of Exploration Geophysicists.

Schuster, G. T., 2017, Seismic inversion: Society of Exploration Geophysicists.

Tsinaslanidis, P., A. Alexandridis, A. Zapranis, and E. Livanis, 2014, Dynamic time

warping as a similarity measure: Applications in finance: Presented at the Hellenic Finance and Accounting Association.

Ursenbach, C., P. Cary, and M. Perz, 2013, Limits on resolution enhancement for PS data mapped to PP time: The Leading Edge, **32**, 64–71.

White, R. E., 1991, Properties of instantaneous seismic attributes: The Leading Edge, **10**, 26–32.

Xue, Z., Y. Chen, S. Fomel, and J. Sun, 2016, Seismic imaging of incomplete data and simultaneous-source data using least-squares reverse-time migration with shaping regularization: Geophysics, **81**, no. 1, S11–S20.

Yu, J., J. Hu, G. T. Schuster, and R. Estill, 2006, Prestack migration deconvolution: GEOPHYSICS, **71**, S53–S62.

Zhou, J., J. Sun, and X. Ma, 2014, Application of Gabor transform to amplitude spectrum matching for merging seismic surveys: Presented at the SEG Technical Program Expanded Abstracts 2014, Society of Exploration Geophysicists.

Vita

Sarah graduated with high honors from Cinco Ranch High School in Katy, Texas in 2014 and came to The University of Texas at Austin (UT Austin) to pursue her undergraduate studies. She is a candidate for a Bachelor of Science degree in Geophysics from the Jackson School of Geosciences and a Bachelor of Science degree in Mathematics from the College of Natural Sciences. After graduation, she plans on pursuing a PhD in Mathematics and Computational Science at the Massachusetts Institute of Technology.

Permanent address: 2175 Stonecrest Drive, Eugene, OR, 97401, USA

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.