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 (**????????**), 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.

CODE

The examples in this thesis were implemented with the Madagascar open-source software environment for reproducible computational experiments (**?**). 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**
??	chapter-locfreq/merge/	2, 3, 4
??, ??	chapter-background/dmExample/	1
??, ??, ??, ??	chapter-merge/apache/	10
??, ??, ??, ??	chapter-locfreq/merge/	2, 3, 4
??, ??	chapter-locfreq/vecta/	5, 6, 7, 8
??	chapter-locfreq/convergence/	9
??, ??, ??, ??	chapter-merge/apache/	10
??, ??	chapter-merge/pcable/	11
??	chapter-merge/pcable2/	12, 13, 14
??, ??, ??, ??, ??	chapter-merge/mighes/	15, 16, 17
??,	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)

def sigPlot(name):
    fig.set_size_inches(10, 2)
    frame1 = plt.gca()
    frame1.axes.get_xaxis().set_visible(False)

    plt.ylim(ymin=-1.2, ymax=1.2)
    axes.spines['left'].set_visible(False)
    axes.spines['bottom'].set_visible(False)
    plt.savefig('Fig/%s'%name, format='pdf', bbox_inches='tight',
                transparent=True, dpi=200)

    plt.cla()
```

```
73 def freqPlot(signal, name, col):
       scale=20                    # freq scale
75     smallTri = triOper(4)       # smoothing
       num = 5000                  # freq density
77
       Y = fft(signal, num)
79     Y = Y[range(len(signal)/scale)]
       Y = foldConv(Y,smallTri)
81     plt.plot(abs(Y/max(Y)), color=col)
       plt.xlabel("Frequency (Hz)")
83     fig.set_size_inches(3, 2)
       frame1 = plt.gca()
85     frame1.axes.get_xaxis().set_visible(True)
       axes.spines['left'].set_visible(True)
87     axes.spines['bottom'].set_visible(True)
       plt.xticks([])
89     plt.savefig(name, format='pdf', bbox_inches='tight', transparent=True, dpi=200)
       plt.cla()
91
   def main():
93
       # read data and convert
95     f = open('trace.txt', 'r')
       signal = f.readlines()
97     signal = [y.strip() for y in signal]
       signal = map(float, signal)
99     signal = asarray(signal)

101    signal = signal[3900:5000]

103    # create triangle operators
       tri = triOper(50)
105
       leg = foldConv(signal,tri)
107
       # time
109    t = linspace(0,len(signal),len(signal))

111    # amplitude
       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)

115    # plot everything
       ########################################
117    # display stuffs
       frame1 = plt.gca()
119    frame1.axes.get_yaxis().set_visible(False)

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

139    # 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__":
```

```
209     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,
68                      gmin=1.5,
                        dt=0.002,
70                      g0=0.9,
                        ng=41,
72                      rect1=50,
                        rect2=50,
74                      fmax=70,
                        frect=20,
76                      fmin=20,
                        frame1=285,
78                      iter=2,
                        clip=0.39)
80
Result('pi','ppi','Overlay')
82 Result('si','vec-si-0','Overlay')

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

90 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))
92 Flow('frame.asc',None,
     'echo %s n1=10 data_format=ascii_int in=$TARGET' % \
94     string.join(map(str,(x,y,x+w,y,x+w,y+w1,x,y+w1,x,y))))
Plot('frame','frame.asc',
96     '''
     dd type=complex form=native |
98     graph min1=199 max1=204 min2=233 max2=254 pad=n plotfat=10 plotcol=3
     wantaxis=n wanttitle=n
100    ''')

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

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

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

6 ##############################################################
#  Modified from $RSFSRC/book/tccs/ltft/vecta/newwarp.py  #
8 ##############################################################

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

14
getmask = 'add scale=1,-1 ${SOURCES[1]} | mask min=0 | dd type=float'
16
def psrect(rect):
18     return '''
     math min=${SOURCES[2]} max=${SOURCES[1]}
20     output="sqrt(1+%d*(1/min^2-1/max^2)*input)" | dd type=int
     ''' % rect
22
def pprect(rect):
24     return '''
     math min=${SOURCES[2]} max=${SOURCES[1]}
26     output="sqrt(1+%d*(1/max^2-1/min^2)*(1-input))" | dd type=int
```

```
    ''' % rect

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

def simil(rect1=1,rect2=1):
    return '''
    similarity other=${SOURCES[1]} rect1=%d rect2=%d
    ''' % (rect1,rect2)

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

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

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

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

def warp2egamma(ss):
    return '''
    math output="(input+x1)/x1" %s
    ''' % ('| math output="2*input-1" ','')[ss]

def nwarp2(name,       # name prefix
           pp,ps,      # PP and PS images
           warp,       # initial warp
           nx,         # number of traces
           tmax,       # maximum time for display
           tmin=0,     # minimum time for display
           j2=1,       # trace sumsampling
           trace=None, # seleted trace
           o2=0,       # trace start
           gmin=1,     # minimum gamma
           gmax=4,     # maximum gamma
           niter=20,   # warping iterations
           dt=0.004,   # time sampling
           fmin=0,     # minimum frequency
           fmax=40,    # maximum frequency
           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
        ##################

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

        psw = n('psw')
        Flow(psw,[ps,pp,wrp],warp0)
        Plot(psw,plot('Warped ' + PS))

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

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

        Result(psw+'all',[pp,psw,dif,gamma],'TwoRows')

        psw1 = n('psw1')
        pp1 = n('pp1')
```

```
Flow(pp1,pp,'window n2=1 f2=286')
Flow(psw1,psw,'window n2=1 f2=286')
####
Result(psw,plot('PSW'))

####

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

####################
# SPECTRAL BALANCING
####################

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

psws = n('psws')
pps  = n('pps')
Flow(psws,'low-smooth%d%i'%(nc,i),'cp')
Flow(pps,'high-smooth%d%i'%(nc,i),'cp')

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

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


##############
## GAMMA SCAN
##############

g1 = 2-g0
warpscan2 = warpscan(ng,g0,g1,rect1,1,int(0.5+rect2/j2))

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

Flow(pr, pps,'cp')
Flow(sr, pswsb,'cp')

Flow(sr+'2',sr,'window j2=%d' % j2)
Flow(pr+'2',pr,'window j2=%d' % j2)

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
334     #########

336     warp = n('wrp')
        Flow([warp,psw+'2'],[sr,pr,pk,wrp],warpit,stdout=-1)
338     #Flow([warp,psw+'2'],[psw,pr,pk,wrp],warpit,stdout=-1)
        Plot(psw+'2',plot('Warped ' + PS))
340
        dif = n('dif2')
342     Plot(dif,[psw+'2',pr],
             'add scale=1,-1 ${SOURCES[1]} | ' + plot('Difference'))
344
        gamma = n('gamma2')
346     Flow(gamma,warp,warp2gamma(ss))
        Plot(gamma,vplot)
348
        if i == iter-1:
350         in1 = n('in1')
            Flow(pr+'in1',pr,interg)
352         Flow(psw+'2in1',psw+'2',interg)
            Plot(in1,[pr+'in1',psw+'2in1'],giplot('After'))
354         Flow(pr+'in1w',pr,interw)
            Flow(psw+'2in1w',psw+'2',interw)
356         Plot(in1+'w',[pr+'in1w',psw+'2in1w'],wiplot('After'))
            Result(in1,in1,'Overlay')
358         Result(in1+'w',in1+'w','Overlay')

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

364         Flow(psw+'1',[ps,pp,warp],warp0)
            Result(psw+'1',plot('Warped ' + PS))
366
```

```
            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
          pp,ps,      # PP and PS images
          warp,       # initial warp
          tmax,       # maximum time for display
          tmin=0,     # minimum time for display
          gmin=1,     # minimum gamma
          gmax=4,     # maximum gamma
          niter=20,   # warping iterations
          dt=0.004,   # time sampling
          fmin=0,     # minimum frequency
          fmax=40,    # maximum frequency
          frect=12,   # frequency smoothing
          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
          iter=2,     # number of iterations
          ss=0
          ):

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

    graph = '''
    graph wanttitle=n min2=%g max2=%g min1=%g max1=%g
    wherexlabel=t wheretitle=b crowd=0.8 label2="Vp/Vs"
    ''' % (gmin,gmax,tmin,tmax)

    dplot ='''
    add scale=1,-1 ${SOURCES[1]} |
    cat ${SOURCES[0]} ${SOURCES[1]} axis=2 |
    window min1=%g max1=%g |
    dots gaineach=0
    labels="Difference:PS warped:PP" label1=Time unit1=s
    ''' % (tmin,tmax)

    def iphase(title):
        return '''
        cat axis=2 ${SOURCES[1]} |
        scale dscale=%g |
```

```
        graph title="Local Frequency (%s)" label1="Time (s)"
        min2=%g max2=%g min1=%g max1=%g
        dash=0,1 label2="Frequency (Hz)"
        ''' % (0.5/(math.pi*dt),title,fmin,fmax,tmin,tmax)

    warpit = warping(niter,200)

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

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

        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++) {

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

        /* set maximum allowed radius */
        if (rad[i] > maxrad) rad[i] = maxrad;

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

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

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

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
     ''')

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 vel0=0.02 an=0.1')

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

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

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

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

Result('diff0','window max1=2 | grey title="Difference before warping" clip=36.5')
Result('diff1','window max1=2 | grey title="Difference after warping" clip=36.5')


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

Flow('hires-warp','hires warp-hires',
     'warp1 other=${SOURCES[0]} warpin=${SOURCES[1]} nliter=0')

Flow('hires-warp-freq','hires-warp','''iphase order=10 rect1=160 rect2=80 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"''')

Flow('rect2','legacy-freq hires-warp-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))
```

```
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

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

330 Flow('hires-warp-diff','hires-warp-balance legacy4',
      'add ${SOURCES[1]} scale=-1,1 | pad beg3=1')
332
Flow('hires-warp-diff-reverse','hires-warp-balance-reverse legacy4',
334       'add ${SOURCES[1]} scale=-1,1 | pad beg3=1')

336 # function for weight
f3 = "5*erfc(x1)"
338
# fix top to be hires
340 Flow('weight2','lweight',
      'math output=1 | cut min1=0.25 | smooth rect1=50 repeat=4 | add $SOURCE')
342
Flow('weight3','hires-warp weight2',
344       'math output=1 | cut min1=0.25 | add ${SOURCES[1]}')

346 Flow('weight5','lweight','math output=\"'+f3+'\"| cut max1=0.25')
Flow('hweight','lweight weight5',
348       'math output=5 | cut min1=0.25 | add ${SOURCES[1]} | scale dscale=1')
Result('hweight', 'grey color=j scalebar=y title="Hires Weight"')
350 Result('lweight2', 'grey color=j scalebar=y title="Legacy Weight" mean=y')
Flow('lweight3', 'hweight', 'math output=1')
352
# right-hand side
354 Flow('hires-legacy','hires-warp legacy4 hweight',
      'mul ${SOURCES[2]} | cat axis=3 ${SOURCES[1]}')
356
Flow('merge','hires-legacy legacy4 rect2 hweight lweight3',
358       '''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
      lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}''')
360
Result('merge','grey color=seismic title="Merged Image" ')
362
Flow('merge3','hires-warp-diff hires-warp rect2 hweight lweight',
364       '''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
      lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]} | add ${SOURCES[1]}''')
366
Result('merge3','grey color=seismic title="Merged Image" ')
368
Flow('merge1','hires-warp-diff hires-warp rect2 hweight lweight2',
370       '''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
      lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]} ''')
372
Flow('hweight-reverse','hweight','cp')
```

```
374
Flow('merge1-reverse','''hires-warp-diff-reverse hires-warp rect2 hweight-reverse
376     lweight-reverse''','''conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]} ''')
378
Flow('merge2','hires-warp merge1','add ${SOURCES[1]}')
380
Flow('merge2-reverse','hires-warp merge1-reverse','add ${SOURCES[1]}')
382
Result('merge1',
384       'grey color=seismic title="Difference between Merged and Hires" clip=3.7011')

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

388 # Difference between high-resolution and merged image
Result('merge1-reverse','''bandpass fhi=250 | window j1=3 |
390       grey color=seismic title="Difference between Merged and Hires" clip=3.7011''')

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


396
# Display the spectra
398 Flow('hires-smooth4','hires-smooth','spline n1=4001 d1=0.001 o1=0')
Flow('merge-spec4','merge','spectra all=y')
400 Flow('legacy4-spec4','legacy4','spectra all=y')
Flow('hires-spec4','hires-warp','spectra all=y')
402 Result('nspectra2','legacy4-spec4 hires-spec4 merge-spec4',
       '''
404       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
       graph title="Normalized Spectra" label2="Amplitude"
406       ''')

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

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

412 Result('nspectra22b','hires-spec4 merge2-spec4',
       '''
414       cat axis=2 ${SOURCES[1]} | window max1=120 |
       graph title="Spectra" dash=1,0
416       ''')

418 Result('nspectra22','legacy4-spec4 hires-spec4 merge2-spec4',
       '''
420       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
       graph title="Normalized Spectra" dash=2,1,0
422       ''')

424 Result('nspectra22b-reverse','hires-spec4 merge2-spec4-reverse',
       '''
426       cat axis=2 ${SOURCES[1]} | window max1=120 |
       graph title="Spectra" dash=1,0
428       ''')

430 # Final Spectra
Result('nspectra22-reverse','hires-spec4 legacy4-spec4 merge2-spec4-reverse',
       '''
432       cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
       graph title="Normalized Spectra " dash=2,1,0 label2="Amplitude"
434       ''')
436 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')


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


# Rebin legacy and hires based on cdpx' and cdpy'
Flow('bin-legacy','legacy tlegacyr','transp | bin head=${SOURCES[1]} xkey=89 ykey=90
    xmin=36790556 xmax=37090852 ymin=324757760 ymax=326225952 nx=67 ny=327')
Result('bin-legacy','transp | put label2=cdpx label1=cdpy unit1=ft unit2=ft | byte
    gainpanel=all | window max3=3 | grey3 frame1=35 frame2=26 frame3=125 title=Legacy
    ')
Flow('bin-hires','hires thiresr','transp | bin head=${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
    ')

# transpose back
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')

# because it takes too long to rebuild
#Ignore('hirestr.rsf','bin-hires.rsf')
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')

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

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



# 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')
Flow('legacy4','legacyx','spline d1=0.0005 n1=6001 o1=0')
Result('hires4','byte gainpanel=all | grey title="High Resolution"')
Result('legacy4','byte gainpanel=all | grey title=Legacy')


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

```
410
412 # THIRD TIME
    Flow('rect3','rect2 freqdif-filt2','math s=${SOURCES[1]} output="input+0.15*s"')
414 Result('rect3','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
        Radius barunit=samples')

416 #Flow('hires-smooth3','hires4 rect3','nsmooth1 rect=${SOURCES[1]}')
    Flow('hires-smooth3','hiresagc rect3','nsmooth1 rect=${SOURCES[1]}')
418 Flow('hires-smooth-spec3','hires-smooth3','spectra all=y')
    Result('hires-smooth-spec3','hires-smooth-spec3 legacyfilt-spec',
420         '''
            cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
422         graph title="Normalized Spectra after Smoothing 3" label2="Amplitude"
            ''')
424 Result('hires-smooth3', 'grey title="Hires Smooth"')

426 # 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 ')
428 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]}')
430 Result('freqdif-filt3','grey allpos=y color=j scalebar=y mean=y clip=30 bias=-15
        title="Difference in Local Frequencies after Filtering 3" ')


432
    # FOURTH TIME
434 Flow('rect4','rect3 freqdif-filt3','math s=${SOURCES[1]} output="input+0.35*s"')
    Result('rect4','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
        Radius barunit=samples')
436
    #Flow('hires-smooth4','hires4 rect4','nsmooth1 rect=${SOURCES[1]}')
438 Flow('hires-smooth4','hiresagc rect4','nsmooth1 rect=${SOURCES[1]}')
    Flow('hires-smooth-spec4','hires-smooth4','spectra all=y')
440 Result('hires-smooth-spec4','hires-smooth-spec4 legacyfilt-spec',
            '''
442         cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
            graph title="Normalized Spectra after Smoothing 4" label2="Amplitude"
444         ''')
    Result('hires-smooth4', 'grey title="Hires Smooth"')
446
    # Difference in local frequencies with nonstationary smoothing applied to hires
448 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" ')
450 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" ')
452


454 # FIFTH TIME
    # mask necessary to "over-smooth" the high frequency differences
456 Flow('mask5','freqdif-filt4','mask max=14 | dd type=float')
    Flow('mask5b','mask5','math output="abs(input-1)"') # inverse mask
458 Flow('rect5b','mask5b','math output=20*input | smooth rect1=50')
    Flow('rect5c','rect4 freqdif-filt4','math s=${SOURCES[1]} output="input+0.25*s"')
460 Flow('rect5',' rect5b rect5c','math m=${SOURCES[1]} output=input+m')
    Result('rect5','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
        Radius barunit=samples')
462 #Flow('rect5','rect4 freqdif-filt3','math s=${SOURCES[1]} output="input+0.25*s"')
    #Result('rect5','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
        Radius barunit=samples')
464
    #Flow('hires-smooth5','hires4 rect5','nsmooth1 rect=${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')

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 beg3=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', '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)
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')


Flow('merge1-reverse','hires-warp-diff-reverse hires-warp rect10 hweight lweight-
    reverse',
     '''
     conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}
     ''')
Flow('merge','hires-warp merge1-reverse','add ${SOURCES[1]}')

# Merge with different weight

Flow('mergeagc','merge','shapeagc rect1=1000 rect2=5')
Result('mergeagc','grey title="Merged"')

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

Result('mergeagcwindow2','mergeagc','window min1=1 max1=2.7 | grey title="Merged"')
Result('hires4window2','hires4','window min1=1 max1=2.7 | grey title="Hires"')
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"')

#Result('windowed','Fig/legacy4window Fig/hires4window Fig/mergeagcwindow Fig/
    legacy4window2 Fig/hires4window2agc Fig/mergeagcwindow2', 'TwoRows')
Result('window1','Fig/legacy4window Fig/hires4window Fig/mergeagcwindow', '
    SideBySideAniso')
Result('window2','Fig/legacy4window2 Fig/hires4window2agc Fig/mergeagcwindow2', '
    SideBySideAniso')

# Difference between high-resolution and merged image
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
# merge2-reverse
Result('merge','grey title="Merged" ')


# 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)))


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

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

        j = i-1
        rect = n('rect%i'%i)
        rectj = n('rect%i'%j)
        fdfj = n('freqdif-filt%i'%j)

        Flow(rect,[rectj,fdfj,rad],'./${SOURCES[2]} freq=${SOURCES[1]} c=%f'%c[j])
        Result(rect,rectplot("Smoothing Radius %d")%i)

        hs = n('high-smooth%i'%i)
        hss = n('high-smooth-spec%i'%i)
        hsf = n('high-smooth-freq%i'%i)
```

```
        fdf = n('freqdif-filt%i'%i)

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

        Flow(hss,hs,'spectra all=y')
        Result(hss,[hss,low_spec],specplot)

        Flow(hsf,hs,locfreq)
        Result(hsf,locfreqplot('%s Local Frequency Smoothed %d'%(titlehigh,i)))

        Flow(fdf,[low_freq, hsf],freqdif)
        Result(fdf,freqdifplot(str(i)))
```

Listing 14: chapter-merge/pcable2/SConstruct

```
/* 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;
    }
```

```
52
    sf_floatwrite(rad,n12,out);
54  exit(0);
}
```

Chapter 4

Listing 15: chapter-mighes/sigsbee/SConstruct

```
1  from rsf.proj import *
   from radius import radius
3
   maxWin=80
5  pclip=99
   rad=5
7
   refl = "ref-true.rsf"
9  image0 = "image0.rsf"
   image1 = "image1.rsf"
11 Result('mod','ref-true','grey title="Reflectivity model" screenratio=0.6')
   Flow('env0','image0','envelope')
13 Flow('env1','image1','envelope')
   Result('env0','grey title=Model scalebar=y color=j')
15 Result('env1','grey title=Model scalebar=y color=j')
   Result('image1','grey screenratio=0.6 bias=0 clip=0.0004 title="Second migration"')
17 Result('image0','grey screenratio=0.6 title="First migration"')

19 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')
21
   Flow('a0','env0 env1','divn den=${SOURCES[1]} niter=500 rect1=3 rect2=5')
23
   Result('a0','grey title="Amplitude operator" color=j scalebar=y barlabel="Weight"
       barunit= screenratio=0.6')
25 Result('rect10b','rect10','grey color=j mean=y title="Frequency operator" scalebar=y
       barlabel=Radius barunit=samples screenratio=0.6')


27
   # matching (sfdivn)
29 Flow('a0image1','image1 a0','math w=${SOURCES[1]} output="input*w"')
   Result('a0image1','grey title=a0image1')
31
   Flow('a0image0','image0 a0','math w=${SOURCES[1]} output="input/w"')
33 Result('a0image0','grey title=a0image1')

35 # frequency balancing
   niter=10
37 radius('image0','a0image1', niter, 0.20, bias=-3, clip=15, rect1=rad, rect2=rad,
          maxval=20, minval=-7, theor=False, initial=1,
39        titlehigh="First Migration", titlelow="Second Migration")

41 Flow('env0b','high-smooth%i'%niter,'envelope')
   Flow('env1b','image1','envelope')
43
   Flow('a1','env0b env1b','divn den=${SOURCES[1]} niter=500 rect1=3 rect2=5')
45


47 Flow('mod0','image0 a0','math w=${SOURCES[1]} output="input*(abs(w))^(0.1)"')
   Result('mod%i'%(0),'grey title="mod0"')
49
   def forward(mod,dat):
51     Flow(dat,'%s rect%d'%(mod,niter),'nsmooth rect1=${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=""'''
```

```
39
    def rectplot(name):
41      return 'grey color=j mean=y title="%s" scalebar=y barlabel=Radius barunit=
    samples'%name

43  smooth = 'nsmooth1 rect=${SOURCES[1]}'
    #smooth = 'nsmooth rect1=${SOURCES[1]} rect2=${SOURCES[1]}'
45
    #Result(high, seisplot(titlehigh))
47  #Result(low, seisplot(titlelow))

49  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) */

#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 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)]

    plt.plot(frq,abs(Y)/max(abs(Y)),col)
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Amplitude')
    plt.title('Frequency Spectra')


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

Listing 19: chapter-mighes/triop/trfu.py

```
#! /usr/bin/env python2
```

```
from triOp import *
import numpy as np
import matplotlib.pyplot as plt

# create an impulse
spike = np.zeros(100)
spike[50] = 1

# smooth spike to get impulse respnse of filter
rect = 10
tri = triOper(10)
smoo = foldConv(spike,tri)

# create filter
a = 1           # filter a term
fs = 1/0.001    # sampling frequency

# forward
w, h = freqz(smoo, a)

# inverse
w2, h2 = freqz(a, smoo)


# plot everything
fig, ax1 = plt.subplots()
ax1.set_title('Transfer functions, rect=%i samples, fs=%i Hz'%(rect,fs))
#ax1.set_title('Transfer functions')
ax1.plot(w*fs/(2*pi),  abs(h ), 'b')
ax1.set_ylim([0,4])
ax1.set_xlim([0,500])
ax1.set_xlabel('Frequency (Hz)')
ax1.set_ylabel('Forward', color='b')
ax1.tick_params('y', colors='b')
ax2 = ax1.twinx()
ax2.plot(w2*fs/(2*pi), abs(h2), 'r')
ax2.set_ylim([0,400])
ax2.set_ylabel('Inverse', color='r')
ax2.tick_params('y', colors='r')
fig.tight_layout()
#plt.show()
plt.savefig('Fig/tf.pdf', format='pdf', bbox_inches='tight')
```

Listing 20: chapter-mighes/triop/solve.py

```
#!/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

def lsConjGrad(A, b, tol=1e-8, maxItr=1500):
    At = A.transpose()
    Als = matmul(At,A)
    bls = At.dot(b)
    print ("Matrix rank: %i" %matrix_rank(Als))
    print ("Full rank: %i" %len(bls))
    return conjGrad(Als, bls, tol, maxItr)

def regLsConjGrad(A, b, alpha=0.5, tol=1e-8, maxItr=1500):
    At = A.transpose()
    Als = matmul(At,A) + alpha*identity(len(b))
    bls = At.dot(b)
    return conjGrad(Als, bls, tol, maxItr)

def shaping(A, b, alpha=0.5, tol=1e-8, maxItr=1500, nliter=5, sz=30, tp="bp"):
    At = A.transpose()
    Als = matmul(At,A) + alpha*identity(len(b))
    bls = At.dot(b)

    x = zeros(len(bls))

    if tp=="tri":
        tri=triOper(sz)
        for i in range(0, nliter):
            x = foldConv(x, tri)
            x = conjGrad(Als, bls, tol, maxItr, x)
```

```
            print("Iter: %i"%i)

    elif tp=="bp":
        for i in range(0, nliter):
            x = lowPassFilter(x, sz, 1/0.001, plot=True)
            x = conjGrad(Als, bls, tol, maxItr, x)
            print("Iter: %i"%i)

        #plt.show()

    return x

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