

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 be 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

```

1 #!/usr/bin/env python
3 from scipy.fftpack import fft
from numpy import linspace, concatenate, asarray, sin, cos, pi

```

```

5 from numpy.linalg import norm
import matplotlib.pyplot as plt

7
8 # fonts and such
9 bg_color = 'black'
fg_color = 'white'
11 fig = plt.figure(facecolor=bg_color, edgecolor=fg_color)
axes = plt.axes(facecolor=bg_color, frameon=True)
13
14 def foldConv(signal,filt):
15     """ 1D convolution with folding """
16
17     # length of arrays and center
18     fl = len(filt)
19     sl = len(signal)
20     center = int(len(filt)/2)
21
22     # initialize convolved signal
23     conv = [0]*sl
24
25     # filter must be smaller than signal
26     if fl > sl:
27         raise ValueError, "Filter longer than signal"
28
29     # convolve
30     for i in range(0,sl):
31         # center
32         conv[i] += filt[center]*signal[i]
33
34         # left
35         for j in range(0,center):
36             idx = abs(i-center+j)
37             conv[i] += filt[j]*signal[idx]
38
39         # right
40         for j in range(center + 1, fl):
41             idx = i+j-center
42             if idx >= sl:
43                 idx = (sl-1) - (idx - (sl-1))
44             conv[i] += filt[j]*signal[idx]
45
46     return conv
47
48 def triOper(rect):
49     """ create triangle smoothing operator """
50     if rect < 1:
51         raise ValueError, "Rect minimum size 1"
52
53     # create triangle operator
54     t=linspace(1,rect-1,rect-1)
55     trian = concatenate((t,[rect],t[::-1]),axis=0)
56
57     # normalize and return
58     return trian/norm(trian)
59
60 def sigPlot(name):
61     fig.set_size_inches(10, 2)
62     frame1 = plt.gca()
63     frame1.axes.get_xaxis().set_visible(False)
64
65     plt.ylim(ymin=-1.2, ymax=1.2)
66     axes.spines['left'].set_visible(False)
67     axes.spines['bottom'].set_visible(False)
68     plt.savefig('Fig/%s'%name, format='pdf', bbox_inches='tight',
69                 transparent=True, dpi=200)
70
71     plt.cla()

```

```

73 def freqPlot(signal, name, col):
74     scale=20                                # freq scale
75     smallTri = triOper(4)                   # smoothing
76     num = 5000                               # freq density
77
78     Y = fft(signal, num)
79     Y = Y[range(len(signal)/scale)]
80     Y = foldConv(Y, smallTri)
81     plt.plot(abs(Y/max(Y)), color=col)
82     plt.xlabel("Frequency (Hz)")
83     fig.set_size_inches(3, 2)
84     frame1 = plt.gca()
85     frame1.axes.get_xaxis().set_visible(True)
86     axes.spines['left'].set_visible(True)
87     axes.spines['bottom'].set_visible(True)
88     plt.xticks([])
89     plt.savefig(name, format='pdf', bbox_inches='tight', transparent=True, dpi=200)
90     plt.cla()
91
92 def main():
93
94     # read data and convert
95     f = open('trace.txt', 'r')
96     signal = f.readlines()
97     signal = [y.strip() for y in signal]
98     signal = map(float, signal)
99     signal = asarray(signal)
100
101     signal = signal[3900:5000]
102
103     # create triangle operators
104     tri = triOper(50)
105
106     leg = foldConv(signal, tri)
107
108     # time
109     t = linspace(0, len(signal), len(signal))
110
111     # amplitude
112     amp = 0.1*sin(2*pi*t) - 0.2*cos(2*pi*2*t)
113     t2 = t + 3*(1*sin(2*pi*t) - 4*cos(2*pi*2*t)+4)
114
115     # plot everything
116     #####
117     # display stuffs
118     frame1 = plt.gca()
119     frame1.axes.get_yaxis().set_visible(False)
120
121     # plot color stuff -- white on black
122     axes.xaxis.set_tick_params(color=fg_color, labelcolor=fg_color)
123     axes.yaxis.set_tick_params(color=fg_color, labelcolor=fg_color)
124     axes.xaxis.label.set_color(fg_color)
125     axes.yaxis.label.set_color(fg_color)
126     axes.spines['right'].set_visible(False)
127     axes.spines['top'].set_visible(False)
128     axes.spines['left'].set_visible(False)
129     axes.spines['bottom'].set_visible(False)
130     fig.set_size_inches(15, 4)
131     frame1.axes.get_xaxis().set_visible(False)
132     for spine in axes.spines.values():
133         spine.set_color(fg_color)
134     #####
135     signal = signal / max(signal)
136     signal = signal + amp
137     leg = leg / max(leg)
138
139     # time domain
140     plt.plot(t, leg, color="black")

```

```

141 plt.plot(t2, signal, color="red")
142 sigPlot("bef.pdf")
143
144
145 plt.plot(t, leg, color="black")
146 plt.plot(t2, signal, color="red")
147 sigPlot("one0.pdf")
148
149 # BALANCE FREQ THEN AMP
150 # balance freq
151 sig2 = foldConv(signal,tri)
152 sig2 = sig2 / max(sig2)
153
154 plt.plot(t, leg, color="black")
155 plt.plot(t2, sig2, color="red")
156 sigPlot("one1.pdf")
157
158 # balance amp
159 ampBal = foldConv(leg,tri)
160 ampBal = foldConv(ampBal,tri)
161 ampBal = ampBal / (max(ampBal)*1.5)
162 sig3 = sig2 + ampBal
163 sig3 = sig3 / max(sig3)
164
165 plt.plot(t, leg, color="black")
166 plt.plot(t2, sig3, color="red")
167 sigPlot("one2.pdf")
168
169 # time shifts
170
171 plt.plot(t, leg, color="black")
172 plt.plot(t, sig3, color="red")
173 sigPlot("one3.pdf")
174
175
176 # BALANCE AMP THEN FREQ
177
178 # balance amp
179 ampBalb = foldConv(leg,tri)
180 ampBalb = foldConv(ampBalb,tri)
181 ampBalb = ampBalb / (max(ampBalb)*1.5)
182 sigb2 = signal + ampBal
183 sigb2 = sigb2 / max(sigb2)
184
185 plt.plot(t, leg, color="black")
186 plt.plot(t2, sigb2, color="yellow")
187 sigPlot("two1.pdf")
188
189 # balance freq
190 sigb3 = foldConv(sigb2,tri)
191 sigb3 = sigb3 / max(sigb3)
192
193 plt.plot(t, leg, color="black")
194 plt.plot(t2, sigb3, color="yellow")
195 sigPlot("two2.pdf")
196
197 # time shifts
198 plt.plot(t, leg, color="black")
199 plt.plot(t, sigb3, color="yellow")
200 sigPlot("two3.pdf")
201
202 #after
203 plt.plot(t, leg, color="black")
204 plt.plot(t, sig3, color="green")
205 plt.plot(t, sigb3, color="yellow")
206 sigPlot("aft.pdf")
207
208 if __name__ == "__main__":

```

```
209 main()
```

Chapter 3

Listing 2: chapter-locfreq/merge/SConstruct

```
1 from rsf.proj import *
2 from radius import radius
3
4 # Initial figures
5 Result('legacy','grey title="Legacy"')
6 Result('hires-agc','hires','agc rect1=2000 rect2=5 | grey title="High-resolution"')
7
8 Flow('legacy-spec','legacy','spectra all=y')
9 Result('nspectra-orig','high-spec legacy-spec',
10       '''cat axis=2 ${SOURCES[1]} |
11         scale axis=1 | window max1=180 |
12         graph title="Normalized Spectra" label2="Amplitude" unit2=""''')
13
14 # Balance local frequency
15 flol=40
16 corrections = 5
17 Flow('legacyfilt','legacy','bandpass flo=%d'%(flol))
18 radius('hires','legacyfilt', corrections, [0.13,0.2,0.3,0.5,0.5],
19       bias=0, clip=90, rect1=80, rect2=16, maxval=90 )
20
21 End()
```

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

```
1 from rsf.proj import *
2
3 def radius(high, low,
4           niter,
5           c,
6           bias=-15, clip=30,
7           rect1=40, rect2=80,
8           maxrad=1000,
9           theor=True,
10          scale=9,
11          initial=10,
12          minval=0,
13          maxval=25,
14          titlehigh="Hires",
15          titlelow="Legacy"):
16
17     # initial high-resolution and legacy images
18     # number of corrections
19     # 'step length' for radius corrections. Can
20     # be type int or float for constant c
21     # or type array for changing c.
22     # bias and clip for display
23     # radius for local frequency calculation
24     # maximum allowed radius
25     # use theoretical smoothing radius
26     # scale for theoretical smoothing radius
27     # initial value for constant smoothing radius
28
29     if type(c) is float or type(c) is int:
30         c = [c]*niter
31
32     def seisplot(name):
33         return 'grey title="%s"'%name
34
35     locfreq = '''iphase order=10 rect1=%d rect2=%d hertz=y complex=y |
36               put label="Frequency" unit=Hz'''%(rect1,rect2)
37
38     def locfreqplot(name):
39         return 'grey mean=y color=j scalebar=y title="%s"'%name
40
41     freqdif = 'add scale=-1,1 ${SOURCES[1]} | put label=Frequency'
42
43     def freqdifplot(num):
```

```

35         return '''grey allpos=y color=j scalebar=y mean=y
36             title="Difference in Local Frequencies %s"
37             clip=%d bias=%d minval=%d
38             maxval=%d''' %(num,clip,bias,minval,maxval)
39
40 specplot = '''cat axis=2 ${SOURCES[1]} |
41             scale axis=1 | window max1=180 |
42             graph title="Normalized Spectra" label2="Amplitude" unit2=""'''
43
44 def rectplot(name):
45     return '''grey color=j mean=y title="%s" scalebar=y barlabel=Radius
46         barunit=samples'''%name
47
48 smooth = 'nsmooth1 rect=${SOURCES[1]}'
49
50 Result(high, seisplot(titlehigh))
51 Result(low, seisplot(titlelow))
52
53 Flow('high-freq',high,locfreq)
54 Result('high-freq',locfreqplot('%s Local Frequency'%titlehigh))
55
56 Flow('low-freq',low,locfreq)
57 Result('low-freq',locfreqplot('%s Local Frequency'%titlelow))
58
59 locfreq2 = '''iphase order=10 rect1=1 rect2=1 hertz=y complex=y |
60     put label="Frequency" unit=Hz'''
61
62 Flow('low-freq2',low,locfreq2)
63 Result('low-freq2',locfreqplot('%s Instantaneous Frequency'%titlelow))
64
65 Flow('freqdif','low-freq high-freq',freqdif)
66 Result('freqdif',freqdifplot(''))
67
68 # initial smoothing radius
69 if (theor):
70     from math import pi
71     Flow('rect0','low-freq high-freq',''math f1=${SOURCES[1]}
72     output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g"'''%(scale,2*pi*0.001))
73 else:
74     Flow('rect0','low-freq','math output=%f'%initial)
75
76 Result('rect0',rectplot("Smoothing Radius 0"))
77
78 Flow('high-smooth0','%s rect0' % high,smooth)
79 Result('high-smooth0', seisplot("%s Smooth 0"%titlehigh))
80
81 Flow('high-spec',high,'spectra all=y')
82 Flow('low-spec',low,'spectra all=y')
83 Flow('high-smooth-spec0','high-smooth0','spectra all=y')
84 Result('nspectra','high-spec low-spec',specplot)
85 Result('high-smooth-spec0','high-smooth-spec0 low-spec',specplot)
86
87 Flow('high-smooth-freq0','high-smooth0',locfreq)
88 Result('high-smooth-freq0',
89     locfreqplot("%s Local Frequency Smoothed %d" %(titlehigh,0)))
90
91 Flow('freqdif-filt0','low-freq high-smooth-freq0',freqdif)
92 Result('freqdif-filt0',freqdifplot('0'))
93
94 prog=Program('radius.c')
95 for i in range(1, niter+1):
96     j = i-1
97     Flow('rect%d'%i,'rect%d freqdif-filt%d %s'%(j,j,prog[0]),
98         './${SOURCES[2]} freq=${SOURCES[1]} c=%f'%c[j])
99     Result('rect%d'%i,rectplot("Smoothing Radius %d"%i))
100
101     Flow('high-smooth%d'%i,'%s rect%d'%(high,i),smooth)
102     Result('high-smooth%d'%i, seisplot('%s Smooth %d'%(titlehigh,i)))

```

```

103     Flow('high-smooth-spec%d'%i,'high-smooth%d'%i,'spectra all=y')
104     Result('high-smooth-spec%d'%i,'high-smooth-spec%d low-spec'%i,specplot)
105
106     Flow('high-smooth-freq%d'%i,'high-smooth%d'%i,locfreq)
107     Result('high-smooth-freq%d'%i,
108           locfreqplot('%s Local Frequency Smoothed %d'%(titlehigh,i)))
109
110     Flow('freqdif-filt%d'%i,'low-freq high-smooth-freq%d'%i,freqdif)
111     Result('freqdif-filt%d'%i,freqdifplot(str(i)))

```

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

```

/* smoothing radius (min = 1) */
2 #include <rsf.h>
#include <math.h>
4
int main (int argc, char* argv[])
6 {
    int n1, n1f, n2, n2f, i, n12, n12f;
    float *rect, *fr, maxrad, c, *rad;
    sf_file in, out, freq;
10
    sf_init (argc,argv);
12
    in = sf_input("in");
    freq = sf_input("freq");
    out = sf_output("out");
14
    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.");
16
    n2 = sf_leftsize(in,1);
    n2f = sf_leftsize(freq,1);
18
    n12 = n1*n2;
    n12f = n1f*n2f;
20
    if (n1 != n1f) sf_error("Need matching n1");
    if (n2 != n2f) sf_error("Need matching n2");
22
    if (!sf_getfloat("c",&c)) c=1.;
    if (!sf_getfloat("maxrad",&maxrad)) maxrad=1000.;
24
    rect = sf_floatalloc(n12);
    sf_floatread(rect,n12,in);
26
    fr = sf_floatalloc(n12f);
    sf_floatread(fr,n12f,freq);
28
    rad = sf_floatalloc(n12);
30
    for (i=0; i < n12; i++) {
32
        /* update radius */
        rad[i] = rect[i]+c*fr[i];
34
        /* set constraint conditions: [1, maxrad] */
        if (rad[i] > maxrad)
            rad[i] = maxrad;
        if (rad[i] < 1.0)
            rad[i] = 1.0;
36
    }
38
    sf_floatwrite(rad,n12,out);
    exit(0);
40
}
42
52
54

```


Listing 5: chapter-locfreq/vecta/SConstruct

```

1  from rsf.proj import *
2  from rsf.recipes.beg import server as private
3  import newwarplocfreq
4
5  #####
6  # Modified from $RSFSRC/book/tccs/ltft/vecta/SConstruct #
7  #####
8
9  trace=300
10
11  Flow('line.asc',None,
12      'echo %d 0 %d 4 n1=4 data_format=ascii_float in=$TARGET' %
13      (trace,trace))
14  Plot('line','line.asc',
15      '',
16      'dd form=native | dd type=complex |
17      graph min2=0 max2=4 min1=-0.5 max1=471.5 pad=n wantaxis=n wanttitle=n
18      ''')
19
20  for mode in ['pp','ss']:
21      data = 'bend_l1_%cmig_enhanc.sgy' % mode[1]
22      Fetch(data,'vecta',private)
23      Flow(mode,data,
24          '',
25          'segypread tape=$SOURCE read=data |
26          window n2=471 | scale axis=2 | put label2=Trace
27          ''',stdin=0)
28      Result(mode,mode,'Overlay')
29      Result('v'+mode,[mode,'line'],'Overlay')
30
31  nails = Split(''
32  0.32 0.72
33  0.57 1.22
34  0.97 1.97
35  '')
36
37  Flow('nails0.asc',None,
38      'echo %s n1=2 n2=%d in=$TARGET data_format=ascii_float' %
39      (string.join(nails,' '),len(nails)/2))
40  Flow('nails','nails0.asc','dd form=native')
41  Flow('nreal','nails','window n1=1')
42  Flow('nimag','nails','window f1=1')
43  Plot('nails','nreal nimag',
44      '',
45      'cplx ${SOURCES[:2]} |
46      graph min1=0 max1=2 min2=0 max2=4 symbol='o' wanttitle=n
47      label1="PP time (s)" label2="SS time (s)" plotcol=5
48      symbolsz=15 labelfat=3 font=2 titlefat=3
49      ''',stdin=0)
50
51  Flow('fit','nails pp1','linefit pattern=${SOURCES[1]}')
52  Plot('fit',
53      '',
54      'graph min1=0 max1=2 min2=0 max2=4 title="Line Fit"
55      labelfat=4 font=2 titlefat=4
56      ''')
57  Result('vnails','fit nails','Overlay')
58
59  Flow('fit0','fit','math output=input-x1 | spray o=0 d=1 n=471')
60
61  newwarplocfreq.nwarp2('vec','pp','ss','fit0',
62      nx=471,
63      inter=5,
64      tmax=1.5,
65      ss=1,
66      trace=trace,

```

```

68         gmax=2.3,
        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
81 Result('pi','ppi','Overlay')
82 Result('si','vec-si-0','Overlay')
83
84 box = 'box x0=%g y0=%g label="%s" xt=%g yt=%g lab_fat=3'
85 x = 201
86 y = 242
87 w = 2
88 w1= 9
89
90 Plot('label01',None,box % (8.2,4.5,"A",0.5,-0.5))
91 Plot('label02',None,box % (8.2,4.5,"B",0.5,-0.5))
92 Flow('frame.asc',None,
93      '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))))
95 Plot('frame','frame.asc',
96      'dd type=complex form=native |
97      graph min1=199 max1=204 min2=233 max2=254 pad=n plotfat=10 plotcol=3
98      wantaxis=n wanttitle=n
99      ')
100
101 Result('before','vec-in0-0','Overlay')
102 Result('after','vec-in1-1','Overlay')
103 End()

```

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

```

1 from rsf.proj import *
2 import math, string, sys
3 import rsf.recipes.version as version
4 from radius2 import radius2
5
6 #####
7 # Modified from $RSFSRC/book/tccs/ltft/vecta/newwarp.py #
8 #####
9
10 warp0 = '''
11 warp1 other=${SOURCES[1]} warpin=${SOURCES[2]}
12 verb=1 nliter=0
13 '''
14
15 getmask = 'add scale=1,-1 ${SOURCES[1]} | mask min=0 | dd type=float'
16
17 def psrect(rect):
18     return '''
19     math min=${SOURCES[2]} max=${SOURCES[1]}
20     output="sqrt(1+%d*(1/min^2-1/max^2)*input)" | dd type=int
21     ''' % rect
22
23 def pprect(rect):
24     return '''
25     math min=${SOURCES[2]} max=${SOURCES[1]}
26     output="sqrt(1+%d*(1/max^2-1/min^2)*(1-input))" | dd type=int

```

```

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

34 def simil(rect1=1,rect2=1):
    return '''
36     similarity other=${SOURCES[1]} rect1=%d rect2=%d
    ''' % (rect1,rect2)
38

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

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

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

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

68 def warp2gamma(ss):
    return '''
70     math output="(input+x1)/x1" %s
    ''' % ('| math output="2*input-1" ',') [ss]
72

74 def nwarp2(name,          # name prefix
    pp,ps,                 # PP and PS images
    warp,                  # initial warp
76     nx,                  # number of traces
    tmax,                  # maximum time for display
78     tmin=0,              # minimum time for display
    j2=1,                  # trace sumsampling
80     trace=None,          # selected trace
    o2=0,                  # trace start
82     gmin=1,              # minimum gamma
    gmax=4,                # maximum gamma
84     niter=20,            # warping iterations
    dt=0.004,              # time sampling
86     fmin=0,              # minimum frequency
    fmax=40,               # maximum frequency
88     frect=12,            # frequency smoothing
    frame1=10,             # time frame
90     ng=101,              # number of gammas
    g0=0.96,               # first gamma
92     pmin=0,              # minimum gamma for picking
    pmax=2,                # maximum gamma for picking
94     an=0.5,              # anisotropy for picking

```

```

100         rect1=50,      # vertical smoothing
101         rect2=50,      # lateral smoothing
102         iter=2,        # number of iterations
103         ss=0,          # PP-PS (0) or PP-SS (1)
104         inter=1,       # interleaving
105         clip=6         # display clip
106     ):
107
108     if version.old_version():
109         return # think how to do it better
110
111     interg = '''
112     pad n2=%d | put n2=%d n3=%d | stack
113     ''' % ((nx/inter+1)*inter,inter,nx/inter+1)
114     inter = 2*inter
115     interw = '''
116     pad n2=%d | put n2=%d n3=%d | stack
117     ''' % ((nx/inter+1)*inter,inter,nx/inter+1)
118
119     if trace:
120         for case in (pp,ps,warp):
121             Flow(case+'1',case,'window n2=1 min2=%d' % trace)
122             warp1(name+'t',pp+'1',ps+'1',warp+'1',tmax,tmin,
123                  gmin,gmax,niter,
124                  dt,fmin,fmax,frect,ng,g0,pmin,pmax,an,rect1,iter,ss)
125     else:
126         trace=10
127
128     def plot(title):
129         return '''
130         window min1=%g max1=%g |
131         grey title="%s" label1=Time unit1=s clip=%g
132         label2=Trace unit2=
133         ''' % (tmin,tmax,title,clip)
134
135     vplot = plot('Vp/Vs') + '''
136     clip=%g scalebar=y color=j bias=%g minval=%g maxval=%g
137     ''' % (0.5*(gmax-gmin),0.5*(gmin+gmax),gmin,gmax)
138
139     balance = '''
140     nsmooth1 rect=${SOURCES[1]} |
141     abalance rect1=%d rect2=%d order=100 other=${SOURCES[2]}
142     ''' % (rect1,rect2)
143
144     ifreq = 'iphase rect1=%d rect2=%d order=100' % (2*rect1,2*rect2)
145
146     def freqplot(title):
147         return '''
148         scale scale dscale=%g |
149         %s clip=%g bias=%g color=j scalebar=y barlabel="Frequency (Hz)"
150         ''' % (0.5/(math.pi*dt),plot(title),(fmax-fmin)*0.25,(fmax+fmin)*0.5)
151
152     def specplot(title):
153         return '''
154         cat axis=2 ${SOURCES[1]} |
155         graph title="%s" max1=%g label1="Frequency (Hz)"
156         dash=0,1 plotfat=7 label2=
157         ''' % (title,4*fmax)
158
159     def gipplot(title):
160         return '''
161         interleave axis=2 ${SOURCES[1]} |
162         window min1=%g max1=%g | scale axis=1 |
163         grey title="Interleaved (%s)" label1=Time unit1=s
164         label2="Inline" unit2=
165         ''' % (tmin,tmax,title)

```

```

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

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

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

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

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

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

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

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

202 warpit = warping(niter,200,200)

204 for i in range(iter):
206     wrp = warp

208     #####
209     # INITIAL WARPING
210     #####

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

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

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

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

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

228     psw1 = n('psw1')
229     pp1 = n('pp1')

```

```

232 Flow(pp1,pp,'window n2=1 f2=286')
233 Flow(psw1,psw,'window n2=1 f2=286')
234 #####
235 Result(psw,plot('PSW'))
236
237 #####
238
239 ppps = n('ppps')
240 Result(ppps,[psw1,pp1],
241         '','
242         add scale=1,-1 ${SOURCES[1]} |
243         cat ${SOURCES[0]} ${SOURCES[1]} axis=2 |
244         dots gaineach=n Xscreenwd=9.225 Xscreenht=5.2 Xyyscale=0.8
245         labels="Difference:SS warped:PP" label1=Time unit1=s
246         title="LTF transform balancing"
247         ',')
248
249 #####
250 # SPECTRAL BALANCING
251 #####
252
253 si = n('si')
254 Flow(si,psw,ifreq)
255 Plot(si,freqplot(PS + ' Local Frequency'))
256 nc = 1
257 radius2(pp,psw,
258         niter=5,
259         c=[0.7,0.45,0.35,0.35,0.5],
260         bias=-20, clip=30,
261         rect1=rect1, rect2=rect2,
262         theor=False, initial=1,
263         minval=-20, maxval=10,
264         titlehigh='PP', titlelow='PSW',
265         it=i )
266
267 psws = n('psws')
268 pps = n('pps')
269 Flow(psws,'low-smooth%d%i'%(nc,i),'cp')
270 Flow(pps,'high-smooth%d%i'%(nc,i),'cp')
271
272 # balance amplitudes
273 pswsb = n('pswsb')
274 Flow(pswsb,[psws,pps],
275       'abalance other=${SOURCES[1]} rect1=%i rect2=%i order=100'%(50,50))
276
277 Result(pps,plot("PP smoothed"))
278 Result(psws,plot("PS smoothed "))
279 Result(pswsb,plot("PS smoothed balanced"))
280
281 #####
282 ## GAMMA SCAN
283 #####
284
285 g1 = 2-g0
286 warpscan2 = warpscan(ng,g0,g1,rect1,1,int(0.5+rect2/j2))
287
288 sc = n('sc')
289 sr = n('sr')
290 pr = n('pr')
291
292 Flow(pr, pps,'cp')
293 Flow(sr, pswsb,'cp')
294
295 Flow(sr+'2',sr,'window j2=%d' % j2)
296 Flow(pr+'2',pr,'window j2=%d' % j2)
297
298 Flow(sc,[sr+'2',pr+'2'],warpscan2)

```

```

300 Result(sc,scanplot)
302
303 pk = n('pk')
304
305 if i==0:
306     Flow(pk+'0',sc,pick(max(pmin,g0),min(pmax,g1),
307         rect1,4*rect2/j2,an=an))
308
309 else:
310     Flow(pk+'0',sc,pick(g0,g1,rect1,4*rect2/j2,an=an))
311
312 Flow(pk,pk+'0',
313     ',,'
314     transp memsize=500 |
315     spline n1=%d d1=1 o1=%g |
316     transp memsize=500 |
317     math output="(input-1)*x1"
318     ',,' % (nx,o2))
319
320 #####
321 if i == 0:
322     in0 = n('in0')
323     Flow(pr+'in0',pr,interg)
324     Flow(sr+'in0',sr,interg)
325     Plot(in0, [pr+'in0',sr+'in0'],giplot('Before'))
326
327     Flow(pr+'in0w',pr,interw)
328     Flow(sr+'in0w',sr,interw)
329     Plot(in0+'w', [pr+'in0w',sr+'in0w'],wiplot('Before'))
330
331     Result(in0,in0,'Overlay')
332     Result(in0+'w',in0+'w','Overlay')
333 #####
334
335 #####
336 # WARPING
337 #####
338
339 warp = n('warp')
340 Flow([warp,psw+'2'],[sr,pr,pk,wrp],warppit,stdout=-1)
341 #Flow([warp,psw+'2'],[psw,pr,pk,wrp],warppit,stdout=-1)
342 Plot(psw+'2',plot('Warped ' + PS))
343
344 dif = n('dif2')
345 Plot(dif,[psw+'2',pr],
346     'add scale=1,-1 ${SOURCES[1]} | ' + plot('Difference'))
347
348 gamma = n('gamma2')
349 Flow(gamma,warp,warp2gamma(ss))
350 Plot(gamma,vplot)
351
352 if i == iter-1:
353     in1 = n('in1')
354     Flow(pr+'in1',pr,interg)
355     Flow(psw+'2in1',psw+'2',interg)
356     Plot(in1,[pr+'in1',psw+'2in1'],giplot('After'))
357     Flow(pr+'in1w',pr,interw)
358     Flow(psw+'2in1w',psw+'2',interw)
359     Plot(in1+'w', [pr+'in1w',psw+'2in1w'],wiplot('After'))
360     Result(in1,in1,'Overlay')
361     Result(in1+'w',in1+'w','Overlay')
362
363     sim1 = n('sim1')
364     Flow(sim1,[pr,psw+'2'],simil(rect1,rect2))
365     Result(sim1,simplot % 'After')
366
367     Flow(psw+'1',[ps,pp,warp],warp0)
368     Result(psw+'1',plot('Warped ' + PS))

```

```

368         rt = n('rt')
369         Flow(psw+'i',psw+'1',ifreq)
370         Flow(rt,psw+'i',
371             '''
372             math output="sqrt(1+12*(1/input^2-1/%g^2))" |
373             dd type=float
374             ''' % (fmax*2*math.pi*dt))
375
376         dl = n('dl')
377         Flow(dl,[psw+'1',rt],
378             '''
379             dd type=float | deblur rect=${SOURCES[1]}
380             verb=y niter=100 eps=0.04 nliter=1
381             '''
382         Result(dl,
383             '''
384             window min1=%g max1=%g |
385             grey title="Deblurred %s" label1="Time (s)"
386             ''' % (tmin,tmax,PS))
387
388         Flow('e'+gamma,warp,warp2egamma(ss))
389         Result(gamma,'e'+gamma,vplot)
390
391     g0 = (g0+1)*0.5
392
393 def warp1(name,      # name prefix
394         pp,ps,      # PP and PS images
395         warp,        # initial warp
396         tmax,        # maximum time for display
397         tmin=0,      # minimum time for display
398         gmin=1,      # minimum gamma
399         gmax=4,      # maximum gamma
400         niter=20,    # warping iterations
401         dt=0.004,    # time sampling
402         fmin=0,      # minimum frequency
403         fmax=40,     # maximum frequency
404         frect=12,    # frequency smoothing
405         ng=101,      # number of gammas
406         g0=0.96,     # first gamma
407         pmin=0,      # minimum gamma for picking
408         pmax=2,      # maximum gamma for picking
409         an=0.5,      # anisotropy for picking
410         rect1=50,    # vertical smoothing
411         iter=2,      # number of iterations
412         ss=0
413         ):
414
415     if version.old_version():
416         return # think how to do it better
417
418     graph = '''
419     graph wanttitle=n min2=%g max2=%g min1=%g max1=%g
420     wherexlabel=t wheretitle=b crowd=0.8 label2="Vp/Vs"
421     ''' % (gmin,gmax,tmin,tmax)
422
423     dplot = '''
424     add scale=1,-1 ${SOURCES[1]} |
425     cat ${SOURCES[0]} ${SOURCES[1]} axis=2 |
426     window min1=%g max1=%g |
427     dots gaineach=0
428     labels="Difference:PS warped:PP" label1=Time unit1=s
429     ''' % (tmin,tmax)
430
431     def iphase(title):
432         return '''
433         cat axis=2 ${SOURCES[1]} |
434         scale dscale=%g |

```



```

436     graph title="Local Frequency (%s)" label1="Time (s)"
437     min2=%g max2=%g min1=%g max1=%g
438     dash=0,1 label2="Frequency (Hz)"
439     ''' % (0.5/(math.pi*dt),title,fmin,fmax,tmin,tmax)
440
441     warpit = warping(niter,200)
442
443     for i in range(iter):
444         #####
445         # INITIAL WARPING
446         #####
447         wrp = warp
448
449         def showpick(case):
450             return '''
451             graph transp=y min2=%g max2=%g min1=%g max1=%g
452             yreverse=y plotcol=%d plotfat=%d
453             wantaxis=n wanttitle=n pad=n
454             ''' % (g0,g1,tmin,tmax,(7,0)[case],(5,1)[case])
455
456         def n(s):
457             return '%s-%s-%d' % (name,s,i)
458
459         gamma = n('gamma')
460         Flow(gamma,wrp,warp2gamma(ss));
461         Plot(gamma,graph)
462
463         psw = n('psw')
464         Flow(psw,[ps,pp,wrp],warp0)
465         Plot(psw,[psw,pp],dplot)
466
467         Result(psw,[gamma,psw], 'OverUnderAniso')

```

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

```

1 from rsf.proj import *
2
3 def radius2(high, low,
4             niter,
5             c,
6             bias=-15, clip=30,
7             rect1=40, rect2=80,
8             maxrad=1000,
9             theor=True,
10            scale=9,
11            initial=10,
12            minval=0, maxval=25,
13            titlehigh="Hires",
14            titlelow="Legacy",
15            it=0):
16
17     # initial high-resolution and legacy images
18     # number of corrections
19     # step length for radius corrections. Can
20     # be type int or float for constant c
21     # or type array for changing c.
22     # bias and clip for display
23     # radius for local frequency calculation
24     # maximum allowed radius
25     # use theoretical smoothing radius
26     # scale for theoretical smoothing radius
27     # initial value for constant smoothing radius
28     # minval and maxval for freqdif-filt display
29     # high title
30     # low title
31     # correction number (from newarp.py)
32
33     if type(c) is float or type(c) is int:
34         c = [c]*niter
35
36     def seisplot(name):
37         return 'grey title="%s" '%name
38
39     locfreq = '''iphae order=10 rect1=%d rect2=%d hertz=y complex=y |
40                put label="Frequency" unit=Hz'''%(rect1,rect2)
41
42     def locfreqplot(name):
43         return 'grey mean=y color=j scalebar=y title="%s" '%name
44
45     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)

```

```

102     Result('low-smooth%d%i'%(i,it), seisplot('%s Smooth %d'%(titlelow,i)))
104     Flow('high-smooth-spec%d%i'%(i,it), 'high-smooth%d%i'%(i,it),
105         'spectra all=y')
106     Flow('low-smooth-spec%d%i'%(i,it), 'low-smooth%d%i'%(i,it),
107         'spectra all=y')
108     Result('high-smooth-spec%d%i'%(i,it),
109         'high-smooth-spec%d%i low-smooth-spec%d%i'%(i,it,i,it), specplot)
110     Flow('high-smooth-freq%d%i'%(i,it), 'high-smooth%d%i'%(i,it), locfreq)
111     Result('high-smooth-freq%d%i'%(i,it),
112         locfreqplot('%s Local Frequency Smoothed %d'%(titlehigh,i)))
114     Flow('low-smooth-freq%d%i'%(i,it), 'low-smooth%d%i'%(i,it), locfreq)
115     Result('low-smooth-freq%d%i'%(i,it),
116         locfreqplot('%s Local Frequency Smoothed %d'%(titlelow,i)))
118     Flow('freqdif-filt%d%i'%(i,it),
119         'low-smooth-freq%d%i high-smooth-freq%d%i'%(i,it,i,it), freqdif)
120     Result('freqdif-filt%d%i'%(i,it), freqdifplot(i))

```

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

```

1  /* smoothing radii */
3  #include <rsf.h>
5  int main (int argc, char* argv[])
6  {
7      int n1, n1f, n2, n2f, i, n12, n12f;
8      float *rect, *fr, maxrad, c, *rad, *rad_low, *rad_high;
9      sf_file in, out, freq, low, high;
11
12     sf_init (argc,argv);
13
14     in = sf_input("in");
15     freq = sf_input("freq");
16     out = sf_output("out");
17     low = sf_output("low");
18     high = sf_output("high");
19
20     if (!sf_histint(in,"n1",&n1)) sf_error("No n1= in input.");
21     if (!sf_histint(freq,"n1",&n1f)) sf_error("No n1= in frequency difference.");
22
23     n2 = sf_leftsize(in,1);
24     n2f = sf_leftsize(freq,1);
25
26     n12 = n1*n2;
27     n12f = n1f*n2f;
28
29     if (n1 != n1f) sf_error("Need matching n1");
30     if (n2 != n2f) sf_error("Need matching n2");
31
32     if (!sf_getfloat("c",&c)) c=1.;
33     if (!sf_getfloat("maxrad",&maxrad)) maxrad=1000.;
34
35     rect = sf_floatalloc(n12);
36     sf_floatread(rect,n12,in);
37
38     fr = sf_floatalloc(n12f);
39     sf_floatread(fr,n12f,freq);
40
41     rad = sf_floatalloc(n12);
42     rad_low = sf_floatalloc(n12);
43     rad_high = sf_floatalloc(n12);
44
45     /* constraint conditions: [-maxrad, -1] U [1, maxrad] */

```

```

45     for (i=0; i < n12; i++) {
46
47         /* update radius */
48         rad[i] = rect[i]+c*fr[i];
49
50         /* set maximum allowed radius */
51         if (rad[i] > maxrad) rad[i] = maxrad;
52
53         /* low radius */
54         if (rad[i] < 0){
55             rad_high[i] = 1;
56             rad_low[i] = -1 * rad[i];
57             if (rad_low[i] < 1) rad_low[i] = 1;
58
59             /* high radius */
60         }else{
61             rad_low[i] = 1;
62             rad_high[i] = rad[i];
63             if (rad_high[i] < 1) rad_high[i] = 1;
64         }
65     }
66
67     sf_floatwrite(rad,n12,out);
68     sf_floatwrite(rad_low,n12,low);
69     sf_floatwrite(rad_high,n12,high);
70     exit(0);
71 }

```

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

```

1  #!/usr/bin/python
2  import matplotlib.pyplot as plt
3  import matplotlib as mpl
4  import matplotlib.font_manager as font_manager
5
6  fontpath = '/home/sarah/.fonts/cmunss.ttf'
7
8  prop = font_manager.FontProperties(fname=fontpath)
9  mpl.rcParams['font.family'] = prop.get_name()
10
11 # first plot
12 niter = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
13 niter2 = [0, 1, 2, 3]
14 norm1= [32475928, 25392492, 19320528, 14462173, 11263933, 10371235, 10477612,
15         10634981, 10760000, 10680567, 10628854, 10559621, 10391489]
16 norm2= [32475928, 23425132, 28284798, 26716290]
17 norm3= [32475928, 27446940, 23236344, 20663992, 18737512, 17107040, 15649435,
18         14493308, 13569843, 12907267, 12425467,12109981, 11745056]
19 norm1[:] = [x / 10371235.0 for x in norm1]
20 norm2[:] = [x / 10371235.0 for x in norm2]
21 norm3[:] = [x / 10371235.0 for x in norm3]
22
23 plt.figure(figsize=(9.60,5.40), dpi=1000)
24 l1 = plt.scatter(niter,norm3)
25 l2 = plt.scatter(niter2,norm2, marker='x', color='g')
26 l3 = plt.scatter(niter,norm1, marker='*', color='y')
27
28 plt.legend((l1, l2, l3),('Too small c', 'Too large c','Good c'))
29 plt.plot([-1, 14], [1, 1], 'r--')
30 plt.xlim(-0.2, 12.2)
31 plt.ylim(0, 3.5)
32 plt.xlabel("Iterations")
33 plt.ylabel(r'$\frac{\mathbf{F}(S_R * d_h)}{\mathbf{F}(d_l)}$')
34
35 plt.savefig('Fig/scalar.pdf', format='pdf', bbox_inches='tight')
36
37

```

```

39 # second plot
niter = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
norm1= [47484892, 33515692, 23253388, 16376629, 11599820, 10069987, 10116965,
41      10258426, 10321123, 10282613, 10289243, 10398856, 10446601]
norm2= [60672360, 33617148, 20220940, 13173313, 10842506, 10398470, 10210745,
43      10296503, 10313730, 10298821, 10204044, 10263224, 10321556]
norm3= [32475928, 25392492, 19320528, 14462173, 11263933, 10371235, 10477612,
45      10634981, 10760000, 10680567, 10628854, 10559621, 10391489]
norm1[:] = [x / 10069987.0 for x in norm1]
47 norm2[:] = [x / 10069987.0 for x in norm2]
norm3[:] = [x / 10069987.0 for x in norm3]
49
plt.figure(figsize=(9.60,5.40), dpi=1000)
51 l1 = plt.scatter(niter,norm1)
l2 = plt.scatter(niter,norm2, marker='x', color='g')
53 l3 = plt.scatter(niter,norm3, marker='*', color='y')

plt.legend((l1, l2, l3),('Initial = constant 10','Initial = constant 1',
55      'Initial = theoretical radius'))
plt.plot([-1, 14], [1, 1], 'r--')
57 plt.xlim(-0.2, 12.2)
plt.ylim(0, 6.5)
59 plt.xlabel("Iterations")
plt.ylabel(r'$\backslash$Vert\mathbf{F}(S_R * d_h) - \mathbf{F}(d_l) \backslash$Vert$')
61
63 plt.savefig('Fig/all.pdf', format='pdf', bbox_inches='tight')

```

Chapter 4

Listing 10: chapter-merge/apache/SConstruct

```

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

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

6
for key in data.keys():
8     Fetch(data[key], 'apache', server)

10     # Convert from SEGy format to Madagascar RSF format
Flow([key, 't'+key, key+'.asc', key+'.bin'], data[key],
12     '''
        segyread tfile=${TARGETS[1]} hfile=${TARGETS[2]}
14         bfile=${TARGETS[3]} | put label2=Cross-line o2=3205 d2=1
        ''')

16
# Display
18 Result(key,
        '''
20         grey color=seismic title="%s Image"
        ''' % key.capitalize())

22
# Look at spectra
24 spectrum = key + '-spec'
Flow(spectrum, key, 'spectra all=y')
26 Result(spectrum, 'graph title="%s Spectrum" ' % key.capitalize())

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

32
# Sample both at 2 ms

```

```

34 Flow('hires2','hires','bandpass fhi=250 | window j1=2')
Flow('legacy2','legacy','spline n1=2001 d1=0.002 o1=0')
36 Flow('legacy4','legacy','spline n1=4001 d1=0.001 o1=0')

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

42 Result('spectra','hires-spec2 legacy-spec2',
    'cat axis=2 ${SOURCES[1]} | graph title=Spectra')
44
Result('nspectra','hires-spec2 legacy-spec2',
46     'cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=120 |
48     graph title="Normalized Spectra" label2="Amplitude"
    ')
50

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

56 Flow('hires3','hires2','bandpass fhi='+str(hif)+' flo='+str(lof))
Flow('legacy3','legacy2','bandpass fhi='+str(hif)+' flo='+str(lof))
58

for key in data.keys():
60     filtspec = key + '-spec3'
    Flow(filtspec, key+'3', 'spectra all=y')
62

Result('filtnspectra','hires-spec3 legacy-spec3',
64     'cat axis=2 ${SOURCES[1]} | scale axis=1 |
66     graph title="Filtered Normalized Spectra"
    ')
68

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

72
# SMOOTHING APPLIED TO HIRES
74 # Measure local frequency
for key in data.keys():
76     freq = key + '-freq'
    Flow(freq,key,'','iphase order=10 rect1=%d rect2=80 hertz=y complex=y |
78     put label=Frequency unit=Hz''' % (40,160)[key=='hires'])
    Result(freq,'','grey mean=y color=j scalebar=y title="%s Local Frequency"
80     minval=10 maxval=75''' % key.capitalize())

82 # Difference in local frequencies
Flow('freqdif','legacy-freq hires-freq',
84     'remap1 n1=4001 d1=0.001 o1=0 | add scale=-1,1 ${SOURCES[1]}')

86 Result('freqdif','','grey allpos=y color=j scalebar=y minval=0 maxval=40 clip=80
    mean=y title="Difference in Local Frequencies" ''')
88

# Try stationary smoothing
90 for rect in (5,10,15,20):
    smooth = 'smooth%d' % rect
92     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]} |
94     scale axis=1 | window max1=125 | graph title="rect=%d"' % rect)
96

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

100 scale=6.75 # theoretically should be 12

```

```

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

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

108 Flow('hires-smooth','hires rect','nsmooth1 rect=${SOURCES[1]} | window j1=4')
110 Flow('hires-smooth-spec','hires-smooth','spectra all=y')
112 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"''')
114 Result('hires-smooth','grey color=seismic title="Hires Smooth"')

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

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

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

126

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

132 Flow('hires-balanced-reverse','hires-smooth legacy',
      'abalance other=${SOURCES[1]} rect1=80 rect2=160 reverse=n')
134

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

142 # take the difference
Flow('trace-diff','legacy-trace hires-balanced-trace','add ${SOURCES[1]} scale=1,-1')
144

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

148 Result('traces','legacy-trace hires-balanced-trace trace-diff',''cat axis=2
      ${SOURCES[1:3]} | dots gaineach=n labels=Legacy:Hires:Difference yreverse=y''')
150

Result('traces-reverse',''legacy-trace hires-balanced-reverse-trace
152 trace-diff-reverse''', ''cat axis=2 ${SOURCES[1:3]} |
      dots gaineach=n labels=Legacy:Hires:Difference yreverse=y''')
154

Result('traces-reverse-diff','trace-diff trace-diff-reverse',''cat axis=2
156 ${SOURCES[1]} | dots gaineach=n labels=reverse=y:reverse=n yreverse=y''')

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

166 Flow('rotates',rotates,
      'cat axis=2 ${SOURCES[1:%d]} | put o2=-180 d2=5' % len(rotates))
168

Flow('phasematch','hires-balanced-trace rotates',

```

```

170     '''
171     spray axis=2 n=73 o=-180 d=5 label=Angle unit=degree |
172     similarity other=${SOURCES[1]} rect1=160
173     '''
174
175 Plot('phasematch','grey color=j allpos=y title="Phase Similarity" ')
176
177 Flow('phasepick','phasematch','pick rect1=20 vel0=0')
178
179 Plot('phasepick','''graph plotcol=7 yreverse=y transp=y min2=-180 max2=180 pad=n
180     wanttitle=n wantaxis=n''')
181
182 Result('phasematch','phasematch phasepick','Overlay')
183
184 Flow('phaseslice','rotates phasepick','slice pick=${SOURCES[1]}')
185
186 Flow('phase-diff','phaseslice hires-balanced-trace','add ${SOURCES[1]} scale=1,-1')
187
188 Result('phase-traces','phaseslice hires-balanced-trace phase-diff',
189     'cat axis=2 ${SOURCES[1:3]} | dots labels=Legacy:Hires:Difference yreverse=y')
190
191
192 # TIME SHIFT OF FIRST TRACE
193 # Scanning different time shifts
194 Flow('warpscan','hires-balanced-trace legacy-trace',
195     '''
196     warpscan shift=y ng=101 g0=-0.05 dg=0.001
197     other=${SOURCES[1]} rect1=20
198     ''')
199
200 Flow('warppick','warpscan',
201     'scale axis=2 | pick rect1=10 vel0=0.02 an=0.1')
202
203 Plot('warpscan',
204     '''
205     grey allpos=y color=j title="Shift Scan"
206     label2=Shift unit2=s
207     ''')
208
209 # This is what was removed
210 Plot('warppick',
211     '''
212     graph plotfat=3 plotcol=7 wanttitle=n wantaxis=n
213     min2=-0.05 max2=0.05 pad=n yreverse=y transp=y
214     ''')
215
216 Result('warpscan','warpscan warppick','Overlay')
217
218 # Apply picked shift
219 Flow('warp','hires-balanced-trace legacy-trace warppick',
220     'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')
221
222 Flow('warp-diff','legacy-trace warp','add ${SOURCES[1]} scale=1,-1')
223
224 Result('wtraces','legacy-trace warp warp-diff','''cat axis=2 ${SOURCES[1:3]} |
225     dots gaineach=n labels=Legacy:Warped:Difference yreverse=y''')
226
227
228 # TIME SHIFT OF PHASE ADJUSTED FIRST TRACE
229 Flow('warpscanr','hires-balanced-trace phaseslice',
230     '''
231     warpscan shift=y ng=101 g0=-0.05 dg=0.001
232     other=${SOURCES[1]} rect1=20
233     ''')
234
235 Flow('warppickr','warpscanr',
236     'scale axis=2 | pick rect1=10 vel0=0.02 an=0.1')

```



```

238 Plot('warpscanr',
      ',,'
240     grey allpos=y color=j title="Shift Scan"
      label2=Shift unit2=s
242     ',,')

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

250 Result('warpscanr','warpscanr warppickr','Overlay')
252
# Apply picked shift
254 Flow('warpr','hires-balanced-trace phaseslice warppickr',
      'warpl other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')
256
Flow('warp-diffr','phaseslice warpr','add ${SOURCES[1]} scale=1,-1')
258
Result('phase-wtraces','phaseslice warpr warp-diffr','cat axis=2 ${SOURCES[1:3]} |
260     dots gaineach=n labels=Legacy:Warped:Difference yreverse=y')

262 # RUN WARPSCAN ON THE WHOLE IMAGE
Flow('warpscan3','hires-balanced legacy',
264     ',,'
      warpscan shift=y ng=51 g0=0 dg=0.001
266     other=${SOURCES[1]} rect1=20 rect3=40
      ',,')

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

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

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

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

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

```

```

306 Result('rect2', 'grey color=j mean=y title="Smoothing Radius" scalebar=y
308         barlabel=Radius barunit=samples')
310 Flow('hires-warp-smooth', 'hires-warp rect2', 'nsmooth1 rect=${SOURCES[1]}')
312
314 # CREATE THE MERGED IMAGE
315 Flow('hires-warp-balance lweight', 'hires-warp-smooth legacy4',
316       'abalance weight=${TARGETS[1]} other=${SOURCES[1]} rect1=320 rect2=160')
318 Flow('hires-warp-balance-reverse lweight-reverse', 'hires-warp-smooth legacy4',
319       'abalance weight=${TARGETS[1]} other=${SOURCES[1]} rect1=320 rect2=160 reverse=n')
320
321 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 |
325         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',
331       'add ${SOURCES[1]} scale=-1,1 | pad beg3=1')
332
333 Flow('hires-warp-diff-reverse', 'hires-warp-balance-reverse legacy4',
334       'add ${SOURCES[1]} scale=-1,1 | pad beg3=1')
336
337 # function for weight
338 f3 = "5*erfc(x1)"
339
340 # fix top to be hires
341 Flow('weight2', 'lweight',
342       'math output=1 | cut min1=0.25 | smooth rect1=50 repeat=4 | add $SOURCE')
344
345 Flow('weight3', 'hires-warp weight2',
346       'math output=1 | cut min1=0.25 | add ${SOURCES[1]}')
348
349 Flow('weight5', 'lweight', 'math output=\'+f3+\'| cut max1=0.25')
350 Flow('hweight', 'lweight weight5',
351       'math output=5 | cut min1=0.25 | add ${SOURCES[1]} | scale dscale=1')
352 Result('hweight', 'grey color=j scalebar=y title="Hires Weight")
353 Result('lweight2', 'grey color=j scalebar=y title="Legacy Weight" mean=y')
354 Flow('lweight3', 'hweight', 'math output=1')
356
357 # right-hand side
358 Flow('hires-legacy', 'hires-warp legacy4 hweight',
359       'mul ${SOURCES[2]} | cat axis=3 ${SOURCES[1]}')
360
361 Flow('merge', 'hires-legacy legacy4 rect2 hweight lweight3',
362       'conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
363       lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}')
364
365 Result('merge', 'grey color=seismic title="Merged Image" ')
366
367 Flow('merge3', 'hires-warp-diff hires-warp rect2 hweight lweight',
368       'conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
369       lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]} | add ${SOURCES[1]}')
370
371 Result('merge3', 'grey color=seismic title="Merged Image" ')
372
373 Flow('merge1', 'hires-warp-diff hires-warp rect2 hweight lweight2',
374       'conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
375       lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}')
376
377 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
380 Flow('merge2','hires-warp merge1','add ${SOURCES[1]}')
382
384 Flow('merge2-reverse','hires-warp merge1-reverse','add ${SOURCES[1]}')
386
388 Result('merge1',
    'grey color=seismic title="Difference between Merged and Hires" clip=3.7011')
388
386 Result('merge2','grey color=seismic title="Merged Image" ')
388
388 # Difference between high-resolution and merged image
390 Result('merge1-reverse','''bandpass fhi=250 | window j1=3 |
    grey color=seismic title="Difference between Merged and Hires" clip=3.7011''')
392
392 # Final merged image
394 Result('merge2-reverse',
    'bandpass fhi=250 | window j1=3 | grey color=seismic title="Merged Image"')
396
396 # Display the spectra
398 Flow('hires-smooth4','hires-smooth','spline n1=4001 d1=0.001 o1=0')
398 Flow('merge-spec4','merge','spectra all=y')
400 Flow('legacy4-spec4','legacy4','spectra all=y')
400 Flow('hires-spec4','hires-warp','spectra all=y')
402 Result('nspectra2','legacy4-spec4 hires-spec4 merge-spec4',
    ',,')
404
404 cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
    graph title="Normalized Spectra" label2="Amplitude"
406 ',,')
408
408 Flow('merge2-spec4','merge2','spectra all=y')
410
410 Flow('merge2-spec4-reverse','merge2-reverse','spectra all=y')
412
412 Result('nspectra22b','hires-spec4 merge2-spec4',
    ',,')
414
414 cat axis=2 ${SOURCES[1]} | window max1=120 |
    graph title="Spectra" dash=1,0
416 ',,')
418
418 Result('nspectra22','legacy4-spec4 hires-spec4 merge2-spec4',
    ',,')
420
420 cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
    graph title="Normalized Spectra" dash=2,1,0
422 ',,')
424
424 Result('nspectra22b-reverse','hires-spec4 merge2-spec4-reverse',
    ',,')
426
426 cat axis=2 ${SOURCES[1]} | window max1=120 |
    graph title="Spectra" dash=1,0
428 ',,')
430
430 # Final Spectra
432 Result('nspectra22-reverse','hires-spec4 legacy4-spec4 merge2-spec4-reverse',
    ',,')
434
434 cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=120 | scale axis=1 |
    graph title="Normalized Spectra " dash=2,1,0 label2="Amplitude"
436 ',,')
436 End()

```

Listing 11: chapter-merge/pcable/SConstruct

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

```

2 from rsf.recipes.beg import server

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

6
8 for key in data.keys():
    Fetch(data[key], 'gom', server)

10 # Convert from SEGY format to Madagascar RSF format
11 Flow([key, 't'+key, key+'.asc', key+'.bin'], data[key],
12      '',
13      segyread tfile=${TARGETS[1]} hfile=${TARGETS[2]} bfile=${TARGETS[3]}
14      '')

15 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())

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

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

22 # Legacy
23 Flow('lbin3', 'tlegacy', 'intbin3 head=$SOURCE')
24 Flow('lcdpx', 'lbin3', 'headermath output=cdpX | window n2=1')
25 Flow('lcdpy', 'lbin3', 'headermath output=cdpY | window n2=1')

26 Result('lcdpx',
27      '',
28      dd type=float | scale dscale=0.01 |
29      grey color=j mean=y scalebar=y title=CDPX
30      minval=362681.65 maxval=378226.78 bias=370454.22 clip=7772.57
31      barlabel=Distance label1=Cross-line label2=Inline
32      title="Legacy CDPX"
33      '')

34 Result('lcdpy',
35      '',
36      dd type=float | scale dscale=0.01 |
37      grey color=j mean=y scalebar=y title=CDPY
38      minval=3250286.50 maxval=3260368.75 bias=3255327.63 clip=5071.13
39      barlabel=Distance label1=Cross-line label2=Inline
40      title="Legacy CDPY"
41      '')

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

46 # Hires
47 Flow('hbin3', 'thires', 'intbin3 head=$SOURCE')
48 Flow('hcdpx', 'hbin3', 'headermath output=cdpX | window n2=1')
49 Flow('hcdpy', 'hbin3', 'headermath output=cdpY | window n2=1')

50 Result('hcdpx',
51      '',
52      dd type=float | scale dscale=0.01 |
53      grey color=j mean=y scalebar=y title=CDPX
54      minval=362681.65 maxval=378226.78 bias=370454.22 clip=7772.57
55      barlabel=Distance label1=Cross-line label2=Inline
56      title="Hires CDPX"
57      '')

```

```

66 Result('hcdpy',
        'dd type=float | scale dscale=0.01 |
68         grey color=j mean=y scalebar=y title=CDPY
        minval=3250286.50 maxval=3260368.75 bias=3255327.63 clip=5071.13
70         barlabel=Distance label1=Cross-line label2=Inline
        title="Hires CDPX"
72         '))

74 Flow('hcdpxf','hcdpx','dd type=float | scale dscale=0.01')
75 Flow('hcdpyf','hcdpy','dd type=float | scale dscale=0.01')
76 Flow('hcdp','hcdpxf hcdpyf','cmlpx ${SOURCES[1]}')
77 Result('hcdp','graph title="CDP" label1=x label2=y min1=362681.65 max1=378226.78 min2
        =3250286.50 max2=3260368.75 plotcol=2')
78
79 # Overlay hires and legacy
80 Result('cdp','Fig/lcdp Fig/hcdp','Overlay')

82 # Zoom
83 Result('hcdpzoom','hcdp','graph title="CDP zoom" label1=x label2=y min1=372681.65
        max1=374226.78 min2=3255286.50 max2=3255768.75 symbol=*)
84 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')
85 Result('cdpzoom','Fig/hcdpzoom Fig/lcdpzoom','Overlay')
86
87 # Rotate hires and legacy
88 import math
89
90 a=111.9
91 #a=-2.9 #for lined up legacy instead of hires
92
93 cs=math.cos(math.radians(a))
94 sn=math.sin(math.radians(a))
95 ox=36976140
96 oy=325545425
97
98 # Rotate them both a degrees around (ox, oy)
99 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))
100 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))
102
103 # View new lcdpx and lcdpy
104 Flow('tlegacyrint','tlegacyr','dd type=int')
105 Flow('lbinr','tlegacyrint','intbin3 head=$SOURCE')
106 Flow('lcdpxr','lbinr','headermath output=unass1 | window n2=1')
107 Flow('lcdpyr','lbinr','headermath output=unass2 | window n2=1')
108
109 Result('lcdpxr',
        'dd type=float | scale dscale=0.01 |
110         grey color=j mean=y scalebar=y
111         barlabel=Distance label1=Cross-line label2=Inline
112         title="Legacy CDPX"
113         '))
114
115 Result('lcdpyr',
        'dd type=float | scale dscale=0.01 |
116         grey color=j mean=y scalebar=y
117         barlabel=Distance label1=Cross-line label2=Inline
118         title="Legacy CDPY"
119         '))
120
121 # Display cdpX and cdpY on map
122
123
124

```

```

126 Flow('lcdpxfr','lcdpxr','dd type=float | scale dscale=0.01')
127 Flow('lcdpyfr','lcdpyr','dd type=float | scale dscale=0.01')
128 Flow('lcdpr','lcdpxfr lcdpyfr','cplx ${SOURCES[1]}')

130
131 # View new lcdpx and lcdpy
132 Flow('thiresrint','thiresr','dd type=int')
133 Flow('hbin3r','thiresrint','intbin3 head=$SOURCE')
134 Flow('hcdpxr','hbin3r','headermath output=unass1 | window n2=1')
135 Flow('hcdpyr','hbin3r','headermath output=unass2 | window n2=1')
136
137 Result('hcdpxr',
138     '',
139     dd type=float | scale dscale=0.01 |
140     grey color=j mean=y scalebar=y
141     barlabel=Distance label1=Cross-line label2=Inline
142     title="Hires CDPX"
143     '')
144
145 Result('hcdpyr',
146     '',
147     dd type=float | scale dscale=0.01 |
148     grey color=j mean=y scalebar=y
149     barlabel=Distance label1=Cross-line label2=Inline
150     title="Hires CDPY"
151     '')
152
153 # Display hcdpxr and hcdpyr on map
154 Flow('hcdpxfr','hcdpxr','dd type=float | scale dscale=0.01')
155 Flow('hcdpyfr','hcdpyr','dd type=float | scale dscale=0.01')
156 Flow('hcdpr','hcdpxfr hcdpyfr','cplx ${SOURCES[1]}')

157 #Display
158 min12=361987.24
159 max12=376783.28
160 min22=3246055.68
161 max22=3263662.08

162
163 Result('hcdpr2','hcdpr','graph title="Hires" label1=x label2=y min1=%d max1=%d min2=%d
164     max2=%d' %(min12, max12, min22, max22))
165 Result('hcdpr','graph title="Hires" label1=x label2=y min1=%d max1=%d min2=%d max2=%d
166     ' %(min12, max12, min22, max22))
167 Result('lcdpr','graph title="Legacy" label1=x label2=y min1=%d max1=%d min2=%d max2=%d
168     ' %(min12, max12, min22, max22))

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

178 Result('cdpr','Fig/lcdpr Fig/hcdpr','Overlay')
179 Result('cdprzoom','Fig/hcdprzoom Fig/lcdprzoom','Overlay')

180
181 # Display masked legacy
182 # Mask out unneeded legacy values
183 Flow('lmaskx','tlegacyr','headermath output=unass1 | mask min=36790556 max=37090852 |
184     dd type=float | put d1=1 d2=1 o1=0 o2=0')
185 Flow('lmasky','tlegacyr','headermath output=unass2 | mask min=324757760 max=326225952
186     | dd type=float | put d1=1 d2=1 o1=0 o2=0')
187 Flow('lmask','lmaskx lmasky','math y=${SOURCES[1]} output="input*y" | dd type=int')

```

```

184 Flow('tlegacymask','tlegacyr lmask','headercut mask=${SOURCES[1]}')
Flow('tlegacymaskint','tlegacymask','dd type=int')
186 Flow('lbin3mask','tlegacymaskint','intbin3 head=${SOURCE}')
Flow('lcdpxmask','lbin3mask','headermath output=unass1 | window n2=1')
188 Flow('lcdpymask','lbin3mask','headermath output=unass2 | window n2=1')

190 # New windowed lcdpx and lcdpy
Result('lcdpxmask',
192     '''
        dd type=float | scale dscale=0.01 |
194         grey color=j mean=y scalebar=y
        minval=367905.66 maxval=370908.52 bias=369407.09 clip=1501.43
196         barlabel=Distance label1=Cross-line label2=Inline
        title="Legacy CDPX"
198     ''')

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

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

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

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

222 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))

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

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

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

```

```

    gainpanel=all | window max3=3 | grey3 frame1=35 frame2=26 frame3=1000 title=Hires
    ')
240
# transpose back
242 Flow('legacytr','bin-legacy','transp plane=31 | put label3=cdpX label2=cdpy unit3=ft
    unit2=ft')
    Result('legacytr','byte gainpanel=all | window max1=3 | grey3 frame3=20 frame2=15
    frame1=125 title=Legacy')
244
# because it takes too long to rebuild
246 #Ignore('hirestr.rsF','bin-hires.rsF')
    Flow('hirestr','bin-hires','transp plane=31 | put label3=cdpX label2=cdpy unit3=ft
    unit2=ft')
248 Result('hirestr','byte gainpanel=all | grey3 frame3=20 frame2=15 frame1=1000 title=
    Hires')

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

252 # select one x and window out min and max for hires
    frame=18
254 Flow('legacyx','legacytr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame)
    Result('legacyx','grey color=seismic title=Legacy')
256 Flow('hiresx','hirestr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame)
    Result('hiresx','grey color=seismic title=Hires')
258
260

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

268
##### TEST ANOTHER LINE #####
270 frame2=22
    Flow('legacyx2','legacytr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame2)
272 Result('legacyx2','grey color=seismic title=Legacy')
    Flow('hiresx2','hirestr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame2)
274 Result('hiresx2','grey color=seismic title=Hires')
    Flow('hires42','hiresx2','spline d1=0.0005 n1=6001 o1=0')
276 Flow('legacy42','legacyx2','spline d1=0.0005 n1=6001 o1=0')
    Result('hires42','byte gainpanel=all | grey title="High Resolution"')
278 Result('legacy42','byte gainpanel=all | grey title=Legacy')

280
##### TEST ANOTHER LINE #####
282 frame3=25
    Flow('legacyx3','legacytr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame3)
284 Result('legacyx3','grey color=seismic title=Legacy')
    Flow('hiresx3','hirestr','window n3=1 f3=%d max1=3 f2=3 n2=310' %frame3)
286 Result('hiresx3','grey color=seismic title=Hires')
    Flow('hires43','hiresx3','spline d1=0.0005 n1=6001 o1=0')
288 Flow('legacy43','legacyx3','spline d1=0.0005 n1=6001 o1=0')
    Result('hires43','byte gainpanel=all | grey title="High Resolution"')
290 Result('legacy43','byte gainpanel=all | grey title=Legacy')

292

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

300 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"')
302
# Measure Local Frequency
304 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')
306 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"')
308
Flow('freqdif','legacy-freq hires-freq','add scale=-1,1 ${SOURCES[1]}')
310 Result('freqdif','grey allpos=y color=j scalebar=y mean=y title="Difference in Local
    Frequenies"')

312 #####
314
316 # Gain (?)
#Flow('hirespow','hires4','pow pow1=1')
318 #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')
320 #Result('hirespow-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency
    "')
#Flow('hirespow-spec','hirespow','spectra all=y')
322 #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=""
    #
    #
324 #
326 #
328 #
# Bandpass filter test
330 flo = 25
fhi = 75
332 flol = 40

334 #sferf
#Flow('hiresfilt','hirespow','bandpass fhi=%d flo=%d'%(fhi,flo))
336 Flow('hiresfilt','hires4','bandpass fhi=%d flo=%d'%(fhi,flo))
Flow('legacyfilt','legacy4','bandpass flo=%d'%(flol))
338
Flow('hiresfilt-spec','hiresfilt','spectra all=y')
340 Flow('legacyfilt-spec','legacyfilt','spectra all=y')

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

346 # 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=""
    #
    #
350 #
352 Result('hiresfilt','grey title="hiresfilt"')
Result('legacyfilt','grey title="legacyfilt"')
354
# Measure Local Frequency
356 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')
358 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"')
360 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"
')
362
Flow('freqdif2','legacyfilt-freq hiresfilt-freq','add scale=-1,1 ${SOURCES[1]}')
364 Result('freqdif2','grey allpos=y color=j scalebar=y mean=y title="Difference in Local
Frequencies"')

# Nonstationary smoothing applied to hires to match with legacy
scale=12
368 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))
370 Result('rect','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
Radius barunit=samples')

372 #Flow('hires-smooth','hires4 rect','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth','hiresagc rect','nsmooth1 rect=${SOURCES[1]}')
374 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 |
378 graph title="Normalized Spectra after Smoothing 1" label2="Amplitude"
',,,)
380 Result('hires-smooth','grey title="Hires Smooth"')

382 # 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 ')
384 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]}
')
386 Result('freqdif-filt','grey allpos=y color=j scalebar=y mean=y title="Difference in
Local Frequencies after Filtering" ')

388
# SECOND TIME
390 # there is a relationship between the frequency difference and the ideal radius size
# account for it here
392 Flow('rect2','rect freqdif-filt','math s=${SOURCES[1]} output="input+0.25*s"')
Result('rect2','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
Radius barunit=samples')
394
#Flow('hires-smooth2','hires4 rect2','nsmooth1 rect=${SOURCES[1]}')
396 Flow('hires-smooth2','hiresagc rect2','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec2','hires-smooth2','spectra all=y')
398 Result('hires-smooth-spec2','hires-smooth-spec2 legacyfilt-spec',
',,,
cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
400 graph title="Normalized Spectra after Smoothing 2" label2="Amplitude"
',,,)
402 Result('hires-smooth2','grey title="Hires Smooth"')

404
# Difference in local frequencies with nonstationary smoothing applied to hires
406 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" ')
408 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
434 # FOURTH TIME
Flow('rect4','rect3 freqdif-filt3','math s=${SOURCES[1]} output="input+0.35*s"')
Result('rect4','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')

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]}')

```

```

466 Flow('hires-smooth5','hiresagc rect5','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec5','hires-smooth5','spectra all=y')
468 Result('hires-smooth-spec5','hires-smooth-spec5 legacyfilt-spec',
,,,
470     cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
graph title="Normalized Spectra after Smoothing 5" label2="Amplitude"
472     '')
Result('hires-smooth5', 'grey title="Hires Smooth"')
474
# Difference in local frequencies with nonstationary smoothing applied to hires
476 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" ')
478 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" ')
480
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" ')
482
#####
484
Flow('hires-smooth2f','hires-smooth5','bandpass fhi=60 nphi=1 flo=20')
486
Flow('hires-smooth-spec2f','hires-smooth2f','spectra all=y')
488 Result('hires-smooth-spec2f','hires-smooth-spec2f legacyfilt-spec',
,,,
490     cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
graph title="Normalized Spectra after Smoothing" label2="Amplitude"
492     '')
Flow('hires-smooth-freq2f','hires-smooth2f','iphase order=10 rect1=80 rect2=16 hertz=
y complex=y | put label=Frequency unit=Hz ')
494 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 ${SOURCES
[1]}')
496 Result('freqdif-filt2f','grey allpos=y color=j scalebar=y mean=y clip=30 bias=-15
title="Difference in Local Frequencies after Filtering 2f" ')
498
#####
500
# Warpscan
502 # Balance Amplitues
504 #Flow('hires-balanced','hires-smooth5 legacyfilt','abalance other=${SOURCES[1]} rect1
=100 rect2=260') # Experiment with rect2
Flow('hires-balanced','hires-smooth5 legacyfilt','abalance other=${SOURCES[1]} rect1
=100 rect2=10 ')
506 Result('hires-balanced','grey title="Hires balanced"')
Flow('hires-despike','hires-balanced','despike2 wide1=2')
508 #Flow('hires-despike','hires-balanced2','despike2 wide1=2')
Result('hires-despike','grey title="Hires balanced"')
510
512 Flow('hires-balanced2','hires-smooth5 legacyfilt','abalance other=${SOURCES[1]} rect1
=100 rect2=260')
Result('hires-balanced2','grey title="Hires balanced"')
514
Flow('hires-balanced-reverse','hires-smooth5 legacyfilt','abalance other=${SOURCES
[1]} rect1=100 rect2=260 reverse=n')
516 Result('hires-balanced-reverse','grey title="Hires balanced"')
518
# Select first trace
tr=3.245e8
520 #tr=3.250e8

```

```

for case in ('legacy4','hires-balanced','hires-balanced-reverse','legacyfilt','hires-
despike'):
522     trace = case + '-trace'
    Flow(trace,case,'window n2=1 min2=%d'%tr)
524
# take the difference
526 Flow('trace-diff','legacyfilt-trace hires-despike-trace','add ${SOURCES[1]} scale
    =1,-1')
528 Flow('trace-diff-reverse','legacyfilt-trace hires-balanced-reverse-trace','add ${
    SOURCES[1]} scale=1,-1')
530 Result('traces','legacyfilt-trace hires-despike-trace trace-diff',
    'cat axis=2 ${SOURCES[1:3]} | dots gaineach=n labels=Legacy:Hires:
    Difference yreverse=y')
532 Result('traces-reverse','legacyfilt-trace hires-balanced-reverse-trace trace-diff-
    reverse',
534     'cat axis=2 ${SOURCES[1:3]} | dots gaineach=n labels=Legacy:Hires:
    Difference yreverse=y')
536 # justification of reverse
Result('traces-reverse-diff','trace-diff trace-diff-reverse',
538     'cat axis=2 ${SOURCES[1]} | dots gaineach=n labels=reverse=y:reverse=n
    yreverse=y')
540
# TIME SHIFT OF FIRST TRACE
542 # Scanning different time shifts
Flow('warpscan','hires-despike-trace legacyfilt-trace',
544     '','
    warpscan shift=y ng=56 g0=-0.05 dg=0.001
546     other=${SOURCES[1]} rect1=300
    ''')
548 Flow('warppick','warpscan','scale axis=2 | pick rect1=50 vel0=-0.005 an=0.1')
550 Plot('warpscan',
552     '','
    grey allpos=y color=j title="Shift Scan"
554     label2=Shift unit2=s
    ''')
556 # This is what was removed
558 Plot('warppick',
560     '','
    graph plotfat=3 plotcol=7 wanttitle=n wantaxis=n
562     min2=-0.05 max2=0.005 pad=n yreverse=y transp=y
    ''')
564 Result('warpscan','warpscan warppick','Overlay')
566 # Apply picked shift
Flow('warp','hires-despike-trace legacyfilt-trace warppick','warp1 other=${SOURCES
    [1]} warpin=${SOURCES[2]} nliter=0')
568 Flow('warp-diff','legacyfilt-trace warp','add ${SOURCES[1]} scale=1,-1')
570 Result('wtraces','legacyfilt-trace warp warp-diff','cat axis=2 ${SOURCES[1:3]} | dots
    gaineach=n labels=Legacy:Warped:Difference yreverse=y')
572 # warpscan on image
574 Flow('warpscan3','hires-despike legacyfilt',
576     '','
    warpscan shift=y ng=56 g0=-0.05 dg=0.001
578     other=${SOURCES[1]} rect1=100 rect2=5
    ''')

```

```

580 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"
582    label3="Time Shift" unit3=s scalebar=y barlabel=Similarity''')

584 Flow('warppick3','warpscan3',
    'scale axis=2 | pick rect1=10 rect2=20 rect3=10 vel0=-0.0 an=0.1')
586
Result('warppick3','grey color=j allpos=y title="Estimated Time Shift" scalebar=y
    barlabel=Time barunit=s clip=0.015 bias=-0.01')
588
Flow('diff0','hires-despike legacyfilt','add scale=1,-1 ${SOURCES[1]}')
590
# Apply picked shift
592 Flow('warp3','hires-despike legacyfilt warppick3',
    'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')
594
Flow('diff1','warp3 legacyfilt','add scale=1,-1 ${SOURCES[1]}')
596
Result('diff0','grey clip=1594.26 title="Difference before warping"')
598 Result('diff1','grey clip=1764.31 title="Difference after warping"')

600
# SHIFT WITHOUT BALANCING AMPLITUDES
602 Flow('hires-warp','hiresagc warppick3',
    'warp1 other=${SOURCES[0]} warpin=${SOURCES[1]} nliter=0')
604
Result('hires-warp','grey title="Hires Warped Image"')
606
Flow('hires-warp-freq','hires-warp','iphase order=10 rect1=80 rect2=16 hertz=y
    complex=y | put label=Frequency unit=Hz')
608
Result('hires-warp-freq','grey mean=y color=j scalebar=y title="Warped Hires Local
    Frequency" ')
610
# create smoothing operator
612 # use legacyfilt or legacy4?
Flow('rect6','legacyfilt-freq hires-warp-freq',
614     'math f1=${SOURCES[1]} output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" ' % (
        scale,2*math.pi*0.001))
Result('rect6','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
616 Flow('hires-warp-smooth','hires-warp rect6','nsmooth1 rect=${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 ')
618 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]}')
620 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" ')

622 # Second Time
Flow('rect7','rect6 freqdif-smooth-filt','math s=${SOURCES[1]} output="input+0.25*s"')
624 Result('rect7','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
Flow('hires-warp-smooth2','hires-warp rect7','nsmooth1 rect=${SOURCES[1]}')
626 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" ')
628 Flow('freqdif-smooth-filt2','legacyfilt-freq hires-warp-smooth-freq2','add scale=-1,1
    ${SOURCES[1]}')
Result('freqdif-smooth-filt2','grey allpos=y color=j scalebar=y mean=y clip=30 bias
    =-15 title="Difference in Local Frequencies after Filtering 1" ')
630
# Third time
632 Flow('rect8','rect7 freqdif-smooth-filt2','math s=${SOURCES[1]} output="input+0.25*s"')

```

```

    ')
    Result('rect8','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
634 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 ')
636 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]}')
638 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" ')

640 # Fourth time
    Flow('rect9','rect8 freqdif-smooth-filt3','math s=${SOURCES[1]} output="input+0.25*s"
    ')
642 Result('rect9','grey color=j mean=y title="Smoothing Radius" scalebar=y barlabel=
    Radius barunit=samples')
    Flow('hires-warp-smooth4','hires-warp rect9','nsmooth1 rect=${SOURCES[1]}')
644 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" ')
646 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" ')

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

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

662 # CREATE THE BLENDED IMAGE
664 # 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')
666 #Flow('lweight-reverse','lweight-reverse2','math output=1/input')
668 Flow('lweight-reverse3','lweight-reverse2','math output=5000 | cut min1=0.25 ')
    Flow('lweight-reverse4','lweight-reverse2','cut max1=0.25 ')
670 Result('lweight-reverse4','grey color=j scalebar=y title="Legacy Weight" mean=y')
    Flow('lweight-reverse','lweight-reverse3 lweight-reverse4','math fourth=${SOURCES[1]}
    output=input+fourth')

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

678 f = "erfc(x1)"
680
682 # 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')

```



```

684 #Result('hweight', 'grey color=j scalebar=y title="Hires Weight" mean=y')
Flow('weight5', 'lweight-reverse', 'math output="%s" | cut max1=0.65 | scale dscale=10'
    % f)
686 Flow('hweight', 'lweight-reverse weight5', 'math output=5000000 | cut min1=0.65 |
    smooth rect1=300 | add ${SOURCES[1]} | scale dscale=0.001')
Result('hweight', 'grey color=j scalebar=y title="Hires Weight" mean=y')
688
690 Flow('merge1-reverse', 'hires-warp-diff-reverse hires-warp rect10 hweight lweight-
    reverse',
    ',')
692     conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
    lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}
694     ',')
Flow('merge', 'hires-warp merge1-reverse', 'add ${SOURCES[1]}')
696
# Merge with different weight
698
Flow('mergeagc', 'merge', 'shapeagc rect1=1000 rect2=5')
700 Result('mergeagc', 'grey title="Merged"')

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

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

722 # Display the spectra
Flow('legacy4-spec4', 'legacy4', 'spectra all=y')
724 Flow('hires-spec4', 'hires-warp', 'spectra all=y')
Flow('merge-spec4', 'merge', 'spectra all=y')
726
# merge Final Spectra
728 # nspectra22-reverse
Result('nspectra2', 'legacy4-spec4 hires-spec4 merge-spec4',
    ',')
730     cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=180 |
    scale axis=1 | graph title="Normalized Spectra " dash=2,1,0
    label2="Amplitude" unit2=""
732     ',')
#scale axis=1 | graph title="Spectra "
734     #label2="Amplitude" unit2=""

736
738 Result('three', "Fig/legacy4 Fig/hires4 Fig/merge ", 'SideBySideAniso') # for
    newsletter
740 End()

```


Listing 12: chapter-merge/pcable2/SConstruct

```

1 from rsf.proj import *
2 from rsf.recipes.beg import server
3 from radius import radius
4
5 data = {'legacy':'TxLa_merge_PSTM_enhanced_subset1.sgy',
6         'hires':'pcable_tpdn_mig_subset1.sgy'}
7
8 for key in data.keys():
9     Fetch(data[key], 'gom', server)
10
11     # Convert from SEG-Y format to Madagascar RSF format
12     Flow([key, 't'+key, key+'.asc', key+'.bin'], data[key],
13          '',
14          segyread tfile=${TARGETS[1]} hfile=${TARGETS[2]} bfile=${TARGETS[3]}
15          '')
16     Result(key+'3', key, 'intbin xk=iline yk=xline | put label2=Inline label3=Crossline
17           | byte gainpanel=all | grey3 frame1=500 frame2=200 frame3=200 title=%s' % key.
18           capitalize())
19
20 Flow('legacy2', 'legacy', 'intbin xk=iline yk=xline | put label2=Inline label3=
21     Crossline | byte gainpanel=all ')
22 Result('legacy', 'legacy2', 'grey3 frame1=150 frame2=20 frame3=20 title="Legacy"')
23
24 Flow('hires2', 'hires', 'intbin xk=iline yk=xline | put label2=Inline label3=Crossline
25     | byte gainpanel=all ')
26 Result('hires', 'hires2', 'grey3 frame1=500 frame2=200 frame3=200 title="Hires"')
27
28 min1=36935190/1000
29 max1=37188381/1000
30 min2=325465425/1000
31 max2=325648750/1000
32
33 # Legacy
34 Flow('lbin3', 'tlegacy', 'intbin3 head=$SOURCE')
35 Flow('lcdpx', 'lbin3', 'headermath output=cdp/1000-%d | window n2=1 | dd type=float'%
36     min1)
37 Flow('lcdpy', 'lbin3', 'headermath output=cdpy/1000-%d | window n2=1 | dd type=float'%
38     min2)
39 Flow('lcdp', 'lcdpx lcdpy', 'cmplx ${SOURCES[1]}')
40 Result('lcdp', 'graph title="CDP" min1=%d max1=%d min2=%d max2=%d plotcol=6 plotfat=2
41     dash=2' %(-5, 255, -1, 185))
42 #Result('lcdp', 'graph title="CDP" label1=x label2=y plotcol=6 dash=2')
43
44 # Hires
45 Flow('hbin3', 'thires', 'intbin3 head=$SOURCE')
46 Flow('hcdpx', 'hbin3', 'headermath output=cdp/1000-%d | window n2=1 | dd type=float'%
47     min1)
48 Flow('hcdpy', 'hbin3', 'headermath output=cdpy/1000-%d | window n2=1 | dd type=float'%
49     min2)
50 Flow('hcdp', 'hcdpx hcdpy', 'cmplx ${SOURCES[1]}')
51 Result('hcdp', 'graph title="CDP" label1=x label2=y unit1=kft unit2=kft min1=%d max1=%
52     d min2=%d max2=%d plotcol=5 plotfat=2 dash=2' %(-5, 255, -1, 185))
53 #Result('hcdp', 'graph title="CDP" label1=x label2=y plotcol=5 dash=2')
54
55 Result('cdp', 'Fig/lcdp Fig/hcdp', 'Overlay')
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

# Legacy
58 Flow('lcdpx2','lbin3','headermath output=cdpdx | window n2=1 | dd type=float')
Flow('lcdpy2','lbin3','headermath output=cdpy | window n2=1 | dd type=float')
60 Flow('lcdp2','lcdpx2 lcdpy2','cmplx ${SOURCES[1]}')
Result('lcdp2','graph title="CDP" label1=x label2=y unit1=kft unit2=kft mini=%d max1
      =%d min2=%d max2=%d plotcol=6 plotfat=2 dash=2' %(min12, max12, min22, max22))
62
# Hires
64 Flow('hcdpx2','hbin3','headermath output=cdpdx | window n2=1 | dd type=float')
Flow('hcdpy2','hbin3','headermath output=cdpy | window n2=1 | dd type=float')
66 Flow('hcdp2','hcdpx2 hcdpy2','cmplx ${SOURCES[1]}')
Result('hcdp2','graph title="CDP" label1=x label2=y unit1=kft unit2=kft mini=%d max1
      =%d min2=%d max2=%d plotcol=5 plotfat=2 dash=2' %(min12, max12, min22, max22))
68
Result('cdp2','Fig/lcdp2 Fig/hcdp2','Overlay')
70
72 #####
74
76 from math import cos, sin, radians, pi
78 a=111.9
a2=-2.95 #for lined up legacy to hires
cs=cos(radians(a))
80 sn=sin(radians(a))
cs2=cos(radians(a2))
82 sn2=sin(radians(a2))
ox=36976140
84 oy=325545425

86 # 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))
88 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))

90 # 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))
92 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)+%f" | dd type=int'%(ox, cs2, oy, sn2, ox, ox, sn2, oy, cs2, oy))

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

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

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

112 # View new hcdpx and hcdpy
Flow('hbinr','thiresr','intbin3 head=$SOURCE')
114 Flow('hcdpxr','hbinr','headermath output=unass1 | window n2=1 | dd type=float')

```

```

116 Flow('hcdpyr','hbinr','headermath output=unass2 | window n2=1 | dd type=float')
Flow('hcdpr','hcdpxr hcdpyr','cmplx ${SOURCES[1]}')

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

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

130 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))
132 Result('cdpr','Fig/lcdpr Fig/hcdpr','Overlay')
# legacy rotated results
134 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))
136 Result('cdpr2','Fig/lcdpr2 Fig/hcdpr2','Overlay')

138 # Mask values
min13=36915196
140 max13=36993752
min23=325348256
142 max23=325576544

144 # 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))
146 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=${SOURCES[1]} output="input*y" | dd type=int')
148 Flow('tlegacymask','tlegacyr lmask','headercut mask=${SOURCES[1]} | dd type=int')

150 Flow('lbin3mask','tlegacymask','intbin3 head=${SOURCE}')
Flow('lcdpxmask','lbin3mask','headermath output=unass1 | window n2=1 | dd type=float'
)
152 Flow('lcdpymask','lbin3mask','headermath output=unass2 | window n2=1 | dd type=float'
)
Flow('lcdpmask','lcdpxmask lcdpymask','cmplx ${SOURCES[1]}')
154 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')

156 # Windowed legacy data
158 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')

160 #nx=35
162 #ny=30
nx=65
164 ny=60
#nx=57
166 #ny=52

168 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))

```

```

170 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')
172
Flow('thiresrf','thiresr','dd type=float')
174 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')
176 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')
178
Flow('legacytr','bin-legacy','transp plane=31 | put label3=cdpx label2=cdpy unit3=ft
unit2=ft')
180 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')
182 Result('hirestr','byte gainpanel=all | grey3 frame3=20 frame2=15 frame1=1000 title=
Hires')
184 # window out min and max for hires
Flow('legacy4','legacytr','spline d1=0.0005 n1=3999 o1=0 ')
186
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')
188 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 ')
190
# Look at spectra
192 Flow('legacy-spec','legacy4','spectra all=y | put label2=Amplitude unit2= ')
Result('legacy-spec','graph title="Legacy Spectrum"')
194 Flow('hires-spec','hires4','spectra all=y | put label2=Amplitude unit2= ')
Result('hires-spec','graph title="Hires Spectrum"')
196
Result('spectra','hires-spec legacy-spec','cat axis=2 ${SOURCES[1]} | graph title=
Spectra label2=Amplitude')
198 Result('nspectra','hires-spec legacy-spec','cat axis=2 ${SOURCES[1]} | scale axis=1 |
window max1=300 | graph title="Normalized Spectra"')
200 # 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')
202 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"')
204 Result('hires-freq','grey mean=y color=j scalebar=y title="Hires Local Frequency"')
206
Flow('freqdif','legacy-freq hires-freq','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif','byte gainpanel=all bar=bar.rsrf allpos=y | grey3 frame1=400 frame2
=40 frame3=30 allpos=y color=j scalebar=y mean=y title="Difference in Local
Frequenies"')
208
flol = 25
210 fhil = 125
Flow('legacyfilt','legacy4','bandpass flo=%d'%(flol))
212 Flow('hiresfilt','hires4','bandpass fhi=%d'%(fhil))
Flow('legacyfilt-spec','legacyfilt','spectra all=y')
214
Flow('hiresagc','hires4','shapeagc rect1=1000 rect2=5 rect3=3')
216 Result('hiresagc','grey title="Hires AGC"')
Flow('hiresagc-spec','hiresagc','spectra all=y')
218
# blue = legacy yellow = hires

```

```

220 Result('filtnspectra','hiresagc-spec legacyfilt-spec legacy-spec hires-spec',
    ',,'
222     cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | scale axis=1 | window max1=180 |
    graph title="Filtered Normalized Spectra" label2="Amplitude" unit2=""
224 ',,')
Result('legacyfilt', 'byte gainpanel=all | grey3 frame1=400 frame2=40 frame3=30 title
    ="legacyfilt"')
226
# Measure Local Frequency
228 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')
230 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"')
232
234 Flow('freqdif2','legacyfilt-freq hires-freq','add scale=-1,1 ${SOURCES[1]}')
Result('freqdif2','byte gainpanel=all bar=bar.rsfs allpos=y | grey3 frame1=400 frame2
    =40 frame3=30 allpos=y color=j scalebar=y mean=y title="Difference in Local
    Frequencies"')
236
# Nonstationary smoothing applied to hires to match with legacy
238 scale=12
240 corrections=2
radius('hiresfilt','legacyfilt', corrections, [0.13,0.2,0.3,0.5,0.5], bias=0, clip
    =65, rect1=80, rect2=16, rect3=16, minval=0, maxval=65 )
242
Flow('rect','legacyfilt-freq hiresagc-freq','math f1=${SOURCES[1]} output="sqrt(%g
    *(1/(input*input)-1/(f1*f1)))/%g"'%(scale,2*pi*0.001))
244 Result('rect','byte gainpanel=all bar=bar.rsfs allpos=y | grey3 color=j frame1=400
    frame2=40 frame3=30 mean=y title="Smoothing Radius" scalebar=y barlabel=Radius
    barunit=samples')
246 Flow('hires-smooth','hiresagc rect','nsmooth1 rect=${SOURCES[1]}')
Flow('hires-smooth-spec','hires-smooth','spectra all=y')
248 Result('hires-smooth-spec','hires-smooth-spec legacy-spec',
    ',,'
250     cat axis=2 ${SOURCES[1]} | scale axis=1 | window max1=180 |
    graph title="Normalized Spectra after Smoothing 1" label2="Amplitude"
252 ',,')
Result('hires-smooth', 'grey title="Hires Smooth"')
254
# Difference in local frequencies with nonstationary smoothing applied to hires
256 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.rsfs allpos=y | grey3 mean=y
    color=j scalebar=y title="Hires Local Frequency Smoothed" frame1=500 frame2=10
    frame3=10 ')
258 Flow('freqdif-filt','legacyfilt-freq hires-smooth-freq','add scale=-1,1 ${SOURCES[1]}
    ')
Result('freqdif-filt','byte gainpanel=all bar=bar.rsfs allpos=y | grey3 frame1=400
    frame2=40 frame3=30 allpos=y color=j scalebar=y mean=y title="Difference in Local
    Frequencies"')
260
# Warpscan
262 # Balance Amplitudes
264 hs = 'high-smooth%i'%corrections
266 Flow('hires-balanced','%s hires-smooth legacyfilt'%hs,'abalance other=${SOURCES[1]}
    rect1=100 rect2=30 rect3=10')
Result('hires-balanced','grey title="Hires balanced"')
268 Flow('hires-despike','hires-balanced','despike2 wide1=2')
Result('hires-despike','byte gainpanel=all | grey3 frame1=400 frame2=40 frame3=30

```

```

270     title="Hires balanced'')
272 Flow('warpscan3','hires-despike legacyfilt',
273     ',,'
274     warpscan shift=y ng=56 g0=-0.05 dg=0.001
275     other=${SOURCES[1]} rect1=100 rect2=10 rect3=10 rect4=10
276     ',')
278 Result('warpscan3',''byte gainpanel=all allpos=y bar=bar.rsrf | transp plane=23 |
279     grey3 color=j frame1=500 frame2=40 frame3=30 title="Local Similarity Scan"
280     label3="Time Shift" unit3=s scalebar=y barlabel=Similarity'')
282
283 Flow('warppick3','warpscan3',
284     'scale axis=2 | pick rect1=20 rect2=40 rect3=20 vel0=-0.0 an=0.08')
286 Result('warppick3','grey color=j allpos=y title="Estimated Time Shift" scalebar=y
287     barlabel=Time barunit=s clip=0.015 bias=-0.01')
288 Flow('diff0','hires-despike legacyfilt','add scale=1,-1 ${SOURCES[1]}')
290 # Apply picked shift
291 Flow('warp3','hires-despike legacyfilt warppick3',
292     'warp1 other=${SOURCES[1]} warpin=${SOURCES[2]} nliter=0')
294 Flow('diff1','warp3 legacyfilt','add scale=1,-1 ${SOURCES[1]}')
296 Result('diff0','grey clip=6606.94 title="Difference before warping"')
297 Result('diff1','grey clip=7072.5 title="Difference after warping"')
298
299 Flow('diff2','diff0 diff1','add scale=1,-1 ${SOURCES[1]}')
300 Result('diff2','grey title="Difference before and after warping"')
302 # SHIFT WITHOUT BALANCING AMPLITUDES
303 Flow('hires-warp','hires4 warppick3',
304     'warp1 other=${SOURCES[0]} warpin=${SOURCES[1]} nliter=0')
306 Result('hires-warp','grey title="Hires Warped Image"')
308 Flow('hires-warp-freq','hires-warp','iphase order=10 rect1=80 rect2=16 rect3=8 hertz=
309     y complex=y | put label=Frequency unit=Hz')
310 Result('hires-warp-freq','byte gainpanel=all allpos=y bar=bar.rsrf | grey3 mean=y
311     color=j scalebar=y title="Warped Hires Local Frequency" frame1=500 frame2=10
312     frame3=30')
312 ### second
313 Flow('hires-warpfilt','hires-warp','bandpass fhi=%d'%(fhi1))
314 radius('hires-warpfilt','legacyfilt', corrections, [0.13,0.2,0.3,0.5,0.5], bias=0,
315     clip=65, rect1=80, rect2=16, rect3=16, minval=0, maxval=65, ind='warp')
316 rt = 'rect%i'%corrections
317 rect_warp = "rect%i%s"%(corrections,'warp')
318 # smoothed warped hires
319 Flow('hires-warp-smooth','hires-warp %s'%rect_warp,'nsmooth1 rect=${SOURCES[1]}')
320 Flow('hires-smooth-spec2','hires-warp-smooth','spectra all=y')
321 Result('hires-warp-smooth','grey title="Hires Smooth"')
322
323 # CREATE THE BLENDED IMAGE
324 Flow('hires-warp-balance-reverse lweight-reverse2','hires-warp-smooth legacy4',
325     'abalance weight=${TARGETS[1]} other=${SOURCES[1]} rect1=320 rect2=160 rect3=30
326     reverse=n')
327
328 Flow('lweight-reverse3','lweight-reverse2','math output=500 | cut min1=0.25 ')
329 Result('lweight-reverse3','grey color=j scalebar=y title="Legacy Weight" mean=y')
330 Flow('lweight-reverse4','lweight-reverse2','cut max1=0.25 ')
331 Result('lweight-reverse4','grey color=j scalebar=y title="Legacy Weight" mean=y')

```

```

330 Flow('lweight-reverse','lweight-reverse3 lweight-reverse4','math fourth=${SOURCES[1]}
    output=input+fourth')

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

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

336

338 f = "erfc(x1)"

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

344 Flow('merge1-reverse','hires-warp-diff-reverse hires-warp %s hweight lweight-reverse'
    % rect_warp,
    ',')
    conjgrad legacy rect=${SOURCES[2]} hweight=${SOURCES[3]}
346     lweight=${SOURCES[4]} niter=20 mod=${SOURCES[1]}
348     ',')

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

356 # 0.125s - 0.45s. reappears at 0.625s - 1.15s
358 #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 ')
360 #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')

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

368 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 ')

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

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

384

```

```

386 Result('nspectra3','legacy4-spec4 hires-spec4 merge-spec4',
      '''
      cat axis=2 ${SOURCES[1]} ${SOURCES[2]} | window max1=300 |
388 scale axis=1 | graph title="Spectra "
      label2="Amplitude" unit2="" dash=2,1,0
390 ''')
392 End()

```

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

```

from rsf.proj import *
2
def radius(high, low,          # initial high-resolution and legacy images
4     niter,                  # number of corrections
     c,                      # 'step length' for radius corrections. Can be
     type int or float for constant c
6     bias=-15, clip=30,      # or type array for changing c.
     rect1=40, rect2=80, rect3=1, # bias and clip for display
8     maxrad=1000,           # radius for local frequency calculation
     theor=True,             # maximum allowed radius
10    scale=9,               # use theoretical smoothing radius
     initial=10,             # scale for theoretical smoothing radius
12    minval=0,              # initial value for constant smoothing radius
14    maxval=25,
     titlehigh="Hires",
16    titlelow="Legacy",
     ind = '' ):
18
19 if type(c) is float or type(c) is int:
20     c = [c]*niter
21
22 def seisplot(name):
23     return 'grey title="%s"' % name
24
25 locfreq = '''iphae order=10 rect1=%d rect2=%d rect3=%d hertz=y complex=y |
26     put label="Frequency" unit=Hz''' % (rect1, rect2, rect3)
27
28 def locfreqplot(name):
29     return 'grey mean=y color=j scalebar=y title="%s" ' % name
30
31 freqdif = 'add scale=-1,1 ${SOURCES[1]} | put label=Frequency'
32 def freqdifplot3(num):
33     return '''byte gainpanel=all bar=bar.rsf allpos=y |
34     grey3 frame1=400 frame2=40 frame3=30 allpos=y color=j scalebar=y
35     mean=y
36     title="Difference in Local Frequencies %s"''' % (num)
37
38 def freqdifplot(num):
39     return '''grey allpos=y color=j scalebar=y mean=y title="Difference in Local
40     Frequencies %s"
41     clip=%d bias=%d minval=%d maxval=%d''' % (num, clip, bias, minval,
42     maxval)
43
44 specplot = '''cat axis=2 ${SOURCES[1]} |
45     scale axis=1 | window max1=180 |
46     graph title="Normalized Spectra" label2="Amplitude" unit2=""'''
47
48 def rectplot(name):
49     return 'grey color=j mean=y title="%s" scalebar=y barlabel=Radius barunit=
50     samples' % name
51
52 smooth = 'nsmooth1 rect=${SOURCES[1]}'
53
54 #Result(high, seisplot(titlehigh))
55 #Result(low, seisplot(titlelow))

```



```

52     def n(name):
53         return name+str(ind)
54     high_freq = n('high-freq')
55     low_freq = n('low-freq')
56
57     Flow(high_freq,high,locfreq)
58     Result(high_freq,locfreqplot('%s Local Frequency'%titlehigh))
59
60     Flow(low_freq,low,locfreq)
61     Result(low_freq,locfreqplot('%s Local Frequency'%titlelow))
62
63     freqdif_init = n('freqdif-init')
64
65     Flow(freqdif_init,[low_freq, high_freq],freqdif)
66     Result(freqdif_init,freqdifplot3(''))
67
68     # initial smoothing radius
69     rect = n('rect0')
70     if (theor):
71         from math import pi
72         Flow(rect,[low_freq, high_freq],''math f1=${SOURCES[1]}
73             output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" ''',(scale,2*pi*0.001)
74     )
75     else:
76         Flow(rect,low_freq,'math output=%f'%initial)
77
78     Result(rect,rectplot("Smoothing Radius 0"))
79
80     high_smooth = n('high-smooth0')
81     Flow(high_smooth,[high,rect],smooth)
82     Result(high_smooth, seisplot("%s Smooth 0"%titlehigh))
83
84     high_spec = n('high-spec')
85     low_spec = n('low-spec')
86     Flow(high_spec,high,'spectra all=y')
87     Flow(low_spec,low,'spectra all=y')
88
89     high_ss = n('high-smooth-spec0')
90     nspec = n('nspectra-diff')
91     Flow(high_ss,high_smooth,'spectra all=y')
92     Result(nspec,[high_spec, low_spec],specplot)
93     Result(high_ss,[high_ss, low_spec],specplot)
94
95     high_sf = n('high-smooth-freq0')
96     Flow(high_sf,high_smooth,locfreq)
97     Result(high_sf,locfreqplot("%s Local Frequency Smoothed %d" %(titlehigh,0)))
98
99     freqdif_filt = n('freqdif-filt0')
100    Flow(freqdif_filt,[low_freq, high_sf],freqdif)
101    #Result('freqdif-filt0',freqdifplot('0'))
102    Result(freqdif_filt,freqdifplot3('0'))
103
104    prog=Program('radius.c')
105    rad = str(prog[0])
106    for i in range(1, niter+1):
107
108        j = i-1
109        rect = n('rect%i'%i)
110        rectj = n('rect%i'%j)
111        fdfj = n('freqdif-filt%i'%j)
112
113        Flow(rect,[rectj,fdj,rad],'./${SOURCES[2]} freq=${SOURCES[1]} c=%f'%c[j])
114        Result(rect,rectplot("Smoothing Radius %d"%i))
115
116        hs = n('high-smooth%i'%i)
117        hss = n('high-smooth-spec%i'%i)
118        hsf = n('high-smooth-freq%i'%i)

```

```

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

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

124     Flow(hss,hs,'spectra_all=y')
124     Result(hss,[hss,low_spec],specplot)

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

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

```

Listing 14: chapter-merge/pcable2/SConstruct

```

/* smoothing radius (min = 1) */
2
#include <rsf.h>
4 include <math.h>

6 int main (int argc, char* argv[])
{
8     int n1, n1f, n2, n2f, i, n12, n12f;
9     float *rect, *fr, maxrad, c, *rad;
10    sf_file in, out, freq;

12    sf_init (argc,argv);

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

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

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

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

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

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

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

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

38    rad = sf_floatalloc(n12);

40    for (i=0; i < n12; i++) {
42
44        /* update radius */
44        rad[i] = rect[i]+c*fr[i];

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

```

```

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

```

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]}')

```

```

53 def backward(dat,mod):
    Flow(mod,'%s rect%d'%(dat,niter),'nsmooth rect1=${SOURCES[1]}')
55
56 # Shaping regularization
57 # mod_{n+1} = S[B[d] + m_{n} - BF[m_{n}]]
    n=10
59
60 prog=Program('freqfilt.c')
61 Flow('add-spec','image0-spec image1-spec','math kir2=${SOURCES[1]} output="2*input-
    kir2"')
    Flow('filt','add-spec','scale axis=1 | mask min=0.05 | dd type=float | smooth rect1
        =10')
62 Result('filt','graph')
63 Result('add-spec','graph')
64 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'')
65
66
67
68
69
70
71
72
73
74
75
76
77 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 ${SOURCES[1:3]} |
            bandpass fhi=32.8084'')
78
79 Result('modb%i'%(n-1),'grey title="Corrected migrated image"')
80
81
82
83
84
85
86
87
88
89
90
91 Flow('migdec-shap','mod%i a0'%(n-1),'math w=${SOURCES[1]} output="input*(abs(w))
    ^{(0.1)}"')
92
93 Result('migdec-shap','grey title="Corrected migration" screenratio=0.6')
94 Result('mod%i'%(n-1),'grey title="Corrected migrated image"')
95
96 #####
97 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')
98
99 Result('image0-w1','image0', 'window min1=2 max1=3 min2=10 max2=16 | grey title=
    image0')
100
101 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"')
102
103 Result('image0-w2','image0', 'window min1=2 max1=2.5 min2=22 max2=26 | grey
    screenratio=0.5 screenht=7 title="Conventional migration"')
104
105 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"')
106
107 Result('image0-w3','image0', 'window min1=2 max1=4 min2=18 max2=20 | grey
    screenratio=1.15 screenht=10 title="Conventional migration"')

```

```

109 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')
111 Result('image0-w4','image0',      'window min1=2 max1=3 min2=18 max2=20 | grey title=
    image0')

113 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')
115 Result('image0-w5','image0',      'window min1=4 max1=8 min2=4 max2=5 | grey title=
    image0')

117 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')
119 Result('image0-w6','image0',      'window min1=5 max1=6 min2=14 max2=15 | grey title=
    image0')

121 End()

```

Listing 16: chapter-mighes/sigsbee/radius.py

```

1 from rsf.proj import *

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

17

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

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

23

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

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

29

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

33 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,
35    maxval)

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

```

```

39 def rectplot(name):
40     return 'grey color=j mean=y title="%s" scalebar=y barlabel=Radius barunit=
41         samples'%name
42
43     smooth = 'nsmooth1 rect=${SOURCES[1]}'
44     #smooth = 'nsmooth rect1=${SOURCES[1]} rect2=${SOURCES[1]}'
45
46     #Result(high, seisplot(titlehigh))
47     #Result(low, seisplot(titlelow))
48
49     Flow('high-freq',high,locfreq)
50     Result('high-freq',locfreqplot('%s Local Frequency'%titlehigh))
51
52     Flow('low-freq',low,locfreq)
53     Result('low-freq',locfreqplot('%s Local Frequency'%titlelow))
54
55     Flow('freqdif','low-freq high-freq',freqdif)
56     Result('freqdif',freqdifplot(''))
57
58     # initial smoothing radius
59     if (theor):
60         from math import pi
61         Flow('rect0','low-freq high-freq','math f1=${SOURCES[1]}
62             output="sqrt(%g*(1/(input*input)-1/(f1*f1)))/%g" '''%(scale,2*pi*0.001)
63         )
64     else:
65         Flow('rect0','low-freq','math output=%f'%initial)
66
67     Result('rect0',rectplot("Smoothing Radius 0"))
68
69     Flow('high-smooth0','%s rect0' % high,smooth)
70     Result('high-smooth0', seisplot("%s Smooth 0"%titlehigh))
71
72     Flow('high-spec',high,'spectra all=y')
73     Flow('low-spec',low,'spectra all=y')
74     Flow('high-smooth-spec0','high-smooth0','spectra all=y')
75     Result('nspectra','high-spec low-spec',specplot)
76     Result('high-smooth-spec0','high-smooth-spec0 low-spec',specplot)
77
78     Flow('high-smooth-freq0','high-smooth0',locfreq)
79     Result('high-smooth-freq0',locfreqplot("%s Local Frequency Smoothed %d" %(
80         titlehigh,0)))
81
82     Flow('freqdif-filt0','low-freq high-smooth-freq0',freqdif)
83     Result('freqdif-filt0',freqdifplot('0'))
84
85     prog=Program('radius.c')
86     for i in range(1, niter+1):
87         j = i-1
88         Flow('rect%d'%i,'rect%d freqdif-filt%d %s'%(j,j,prog[0]),'./${SOURCES[2]}
89         freq=${SOURCES[1]} c=%f maxrad=%f'%(c[j],maxrad))
90         Result('rect%d'%i,rectplot("Smoothing Radius %d"%i))
91
92         Flow('high-smooth%d'%i,'%s rect%d'%(high,i),smooth)
93         Result('high-smooth%d'%i, seisplot('%s Smooth %d'%(titlehigh,i)))
94
95         Flow('high-smooth-spec%d'%i,'high-smooth%d'%i,'spectra all=y')
96         Result('high-smooth-spec%d'%i,'high-smooth-spec%d low-spec'%i,specplot)
97
98         Flow('high-smooth-freq%d'%i,'high-smooth%d'%i,locfreq)
99         Result('high-smooth-freq%d'%i,locfreqplot('%s Local Frequency Smoothed %d'%(
100             titlehigh,i)))
101
102         Flow('freqdif-filt%d'%i,'low-freq high-smooth-freq%d'%i,freqdif)
103         Result('freqdif-filt%d'%i,freqdifplot(str(i)))

```

Listing 17: chapter-mighes/sigsbee/radius.c

```

2  /* smoothing radius (min = 1) */
4  #include <rsf.h>
4  #include <math.h>
6  int main (int argc, char* argv[])
{
8      int n1, n1f, n2, n2f, i, n12, n12f;
      float *rect, *fr, maxrad, c, *rad;
10     sf_file in, out, freq;

12     sf_init (argc,argv);

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

18     if (!sf_histint(in,"n1",&n1)) sf_error("No n1= in input.");
      if (!sf_histint(freq,"n1",&n1f)) sf_error("No n1= in frequency difference.");
20
22     n2 = sf_leftsize(in,1);
      n2f = sf_leftsize(freq,1);

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

26
28     if (n1 != n1f) sf_error("Need matching n1");
      if (n2 != n2f) sf_error("Need matching n2");

30     if (!sf_getfloat("c",&c)) c=1.;
      if (!sf_getfloat("maxrad",&maxrad)) maxrad=1000.;
32
34     rect = sf_floatalloc(n12);
      sf_floatread(rect,n12,in);

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

38
40     rad = sf_floatalloc(n12);

42     for (i=0; i < n12; i++) {
44         /* update radius */
          rad[i] = rect[i]+c*fr[i];

46         /* set constraint conditions: [1, maxrad] */
          if (rad[i] > maxrad)
              rad[i] = maxrad;
48         if (rad[i] < 1.0)
              rad[i] = 1.0;
50     }

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

```

Listing 18: chapter-mighes/triop/triOp.py

```

1  #!/usr/bin/env python2
3  from solve import *
5  from numpy import empty, linspace, concatenate, pi, pad
      from numpy.linalg import pinv, norm
7  from scipy.signal import butter, filtfilt, freqz
      from scipy import fft, arange

```

```

9 import matplotlib.pyplot as plt

11 def main():
    print('Triange smoothing')
    print('functions include...')
13     print('    foldConv(signal, filt)')
15     print('    foldConvOp(triOper, sl)')
    print('    triOper(rect)')
17     print('    lowPassFilter(data, highcut, fs, order, plot)')
    print('    plotSpec(y,Fs,col="red")')

19
20 def foldConv(signal,filt):
21     """ 1D convolution with folding """

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

27     # initialize convolved signal
29     conv = [0]*sl

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

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

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

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

51     return conv

53
54 def foldConvOp(triOper, sl, plot=False):
55     """ 1D convolution with folding """

57     # length of arrays and center
    fl = len(triOper)
59     center = int(fl/2)

61     # initialize convolved signal
    conv = [0]*sl
63     mat = empty([sl,sl])

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

69     # convolve
    for i in range(0,sl):
71         # center
        mat[i,i] += triOper[center]

73         # left
75         for j in range(0,center):
            idx = abs(i-center+j)

```



```

77         mat[i,idx] += triOper[j]
79
80         # right
81         for j in range(center + 1, fl):
82             idx = i+j-center
83             if idx >= sl:
84                 idx = (sl-1) - (idx - (sl-1))
85                 mat[i,idx] += triOper[j]
86
87         if plot: # plot the matrix
88             plt.imshow(mat)
89             plt.title("Forward operator")
90             plt.show()
91
92     return mat
93
94 def triOper(rect):
95     """ create triangle smoothing operator """
96     if rect < 1:
97         raise ValueError, "Rect minimum size 1"
98
99     # create triangle operator
100    t = linspace(1,rect-1,rect-1)
101    trian = concatenate((t,[rect],t[::-1]),axis=0)
102    # normalize and return
103    return trian/norm(trian)
104
105 def lowPassFilter(data, highcut, fs, order=5, plot=True):
106     nyq = 0.5 * fs
107     high = highcut / nyq
108
109     b, a = butter(order, high, btype='lowpass')
110     y = filtfilt(b, a, data) # zero phase
111     w, h = freqz(b, a)
112
113     if plot:
114         plt.plot(w*fs/(2*pi), abs(h), 'b')
115         plt.ylabel('Amplitude')
116         plt.xlabel('Frequency (Hz)')
117         plt.title('Filter transfer function')
118         plt.xlim([-5,120])
119     return y
120
121 def plotSpec(y,Fs,col="red"):
122     n = 5000
123     y = pad(y, n, 'constant', constant_values=0) # increase resolution
124     n = len(y)
125     k = arange(n)
126     T = n/Fs
127     frq = k/T
128     frq = frq[range(n/2)]
129
130     Y = fft(y)/n
131     Y = Y[range(n/2)]
132
133     plt.plot(frq,abs(Y)/max(abs(Y)),col)
134     plt.xlabel('Frequency (Hz)')
135     plt.ylabel('Amplitude')
136     plt.title('Frequency Spectra')
137
138 if __name__ == "__main__":
139     main()

```

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

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

```

3 from triOp import *
  import numpy as np
5 import matplotlib.pyplot as plt

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

11 # smooth spike to get impulse response of filter
  rect = 10
13 tri = triOper(10)
  smoo = foldConv(spike, tri)
15

17 # create filter
  a = 1          # filter a term
  fs = 1/0.001   # sampling frequency
19

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

23 # inverse
  w2, h2 = freqz(a, smoo)
25

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

```

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

```

1 #!/usr/bin/env python2

3 from triOp import *
  from numpy import zeros, matmul, identity
5 from numpy.linalg import norm, matrix_rank
  def main():
7     print('Generic systems solver')
      print('functions include...')
9     print('    conjGrad(A, b, tol, maxItr)')
      print('    irls(A, b, tol, maxItr, nliter)')
11    print('    lsConjGrad(A, b, tol, maxItr)')
      print('    regLsConjGrad(A, b, alpha, tol, maxItr)')
13
  def conjGrad(A, b, tol=1e-8, maxItr=1500, x=0):
15     '''
      conjugate gradients solver
17     does NOT check if A is SPD
      '''
19     n = len(b)

21     if norm(x) == 0:

```

```

23         x = zeros(n)
25         r = b - A.dot(x)
25         p = r
27         res0 = norm(r)
27         for i in range(1,maxItr):
29             Ap = A.dot(p)
29             pAp = p.dot(Ap)
31             alpha = r.dot(p)/pAp
33             x += alpha*p
33             r -= alpha*Ap
35
35             if norm(r)/res0 <= tol:
37                 break
39             beta = -Ap.dot(r)/pAp
39             p = r + beta * p
41         return x
43 def irls(A, b, nrm=1, tol=1e-8, maxItr=1500, nliter=10):
43     # find L2 solution
45     At = A.transpose()
45     A = matmul(A,At)
47     b = At.dot(b)
47     psinv = conjGrad(A,b,tol,maxItr)
49
49     # find nrm solution
51     for i in range (1,nliter):
51         print("IRLS nliter # %i"%i)
53         r = b - A.dot(psinv)
53         R = zeros((len(b),len(b)))
55         for j in range(1,len(r)):
55             R[j,j] = abs(r[i]**(nrm-2));
57
57         A = matmul(matmul(At,R),A)
59         b = matmul(At,R).dot(b)
59         psinv = conjGrad(A,b,tol,maxItr)
61
61         return psinv
63
63 def lsConjGrad(A, b, tol=1e-8, maxItr=1500):
65     At = A.transpose()
65     Als = matmul(At,A)
67     bls = At.dot(b)
67     print ("Matrix rank: %i" %matrix_rank(Als))
69     print ("Full rank: %i" %len(bls))
69     return conjGrad(Als, bls, tol, maxItr)
71
71 def regLsConjGrad(A, b, alpha=0.5, tol=1e-8, maxItr=1500):
73     At = A.transpose()
73     Als = matmul(At,A) + alpha*identity(len(b))
75     bls = At.dot(b)
75     return conjGrad(Als, bls, tol, maxItr)
77
77 def shaping(A, b, alpha=0.5, tol=1e-8, maxItr=1500, nliter=5, sz=30, tp="bp"):
79     At = A.transpose()
79     Als = matmul(At,A) + alpha*identity(len(b))
81     bls = At.dot(b)
83
83     x = zeros(len(bls))
85
85     if tp=="tri":
85         tri=triOper(sz)
87         for i in range(0, nliter):
87             x = foldConv(x, tri)
89             x = conjGrad(Als, bls, tol, maxItr, x)

```

```
91         print("Iter: %i"%i)
92     elif tp=="bp":
93         for i in range(0, nliter):
94             x = lowPassFilter(x, sz, 1/0.001, plot=True)
95             x = conjGrad(Als, bls, tol, maxItr, x)
96             print("Iter: %i"%i)
97
98             #plt.show()
99
100     return x
101
102 if __name__ == "__main__":
103     main()
```