

Geophysical data processing examples

-- Open-source Geophysics --



Yang Liu

College of Geo-exploration Science and Technology,
Jilin University

07/11/2017

yangliu1979@jlu.edu.cn

■ Geophysical data processing

■ Assignment

■ Examples

- Seismic data

- Gravity data

- Controlled-source ElectroMagnetic data

■ Geophysical data processing

■ Assignment

■ Examples

- Seismic data

- Gravity data

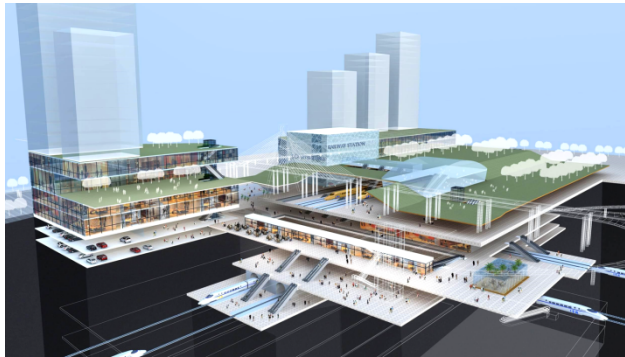
- Controlled-source ElectroMagnetic data

- **Why Geophysical data**

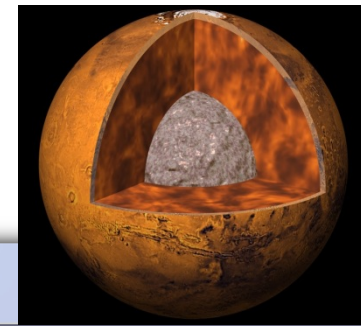
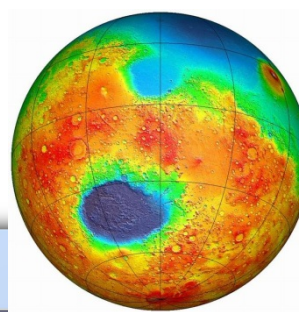
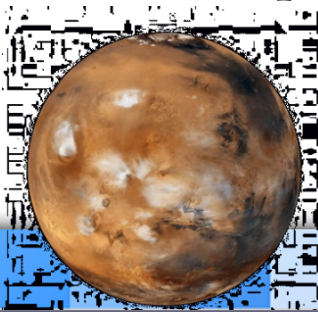
- From **Oil and Gas** to **Solid Mineral Resource**.

- Elastic parameters are not always valid.
- May use more rock properties.

- Urban underground space development.

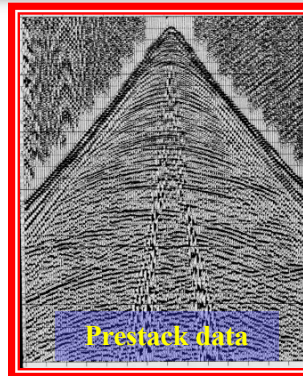


- From Earth to other planets.

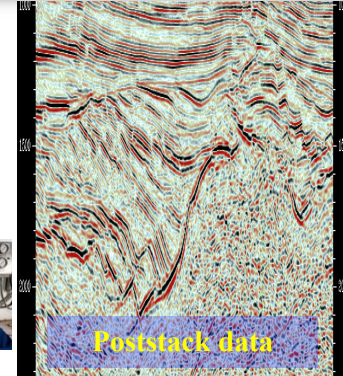
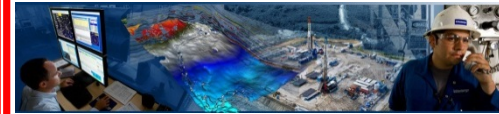


Open-source Geophysical data processing

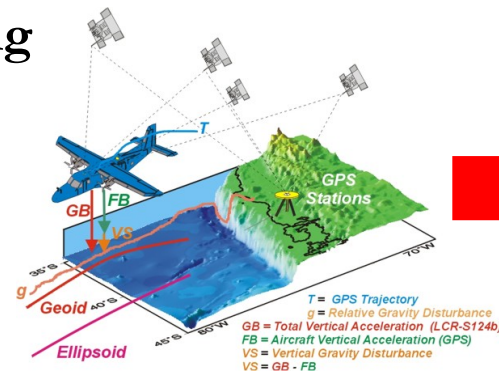
- Seismic data processing



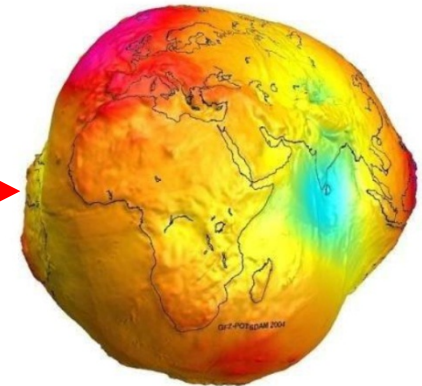
Seismic data processing



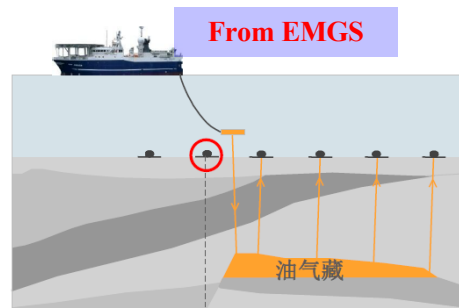
- Gravity data processing



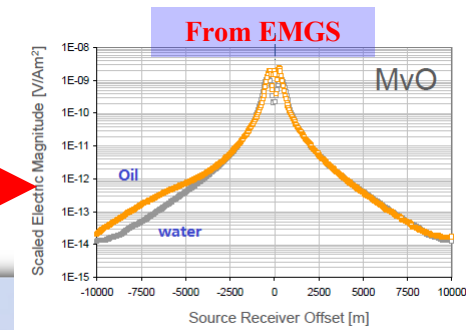
Gravity data processing



- Electromagnetic data processing



CSEM data processing



- Geophysical data processing

- **Assignment**

- Examples

 - Seismic data

 - Gravity data

 - Controlled-source ElectroMagnetic data

- The tutorial sets up a processing workflow for a field seismic land data.
- The tutorial sets up a processing framework for a field gravity data.
- The tutorial introduces a key preprocessing step for a synthetic CSEM data.

Before writing script

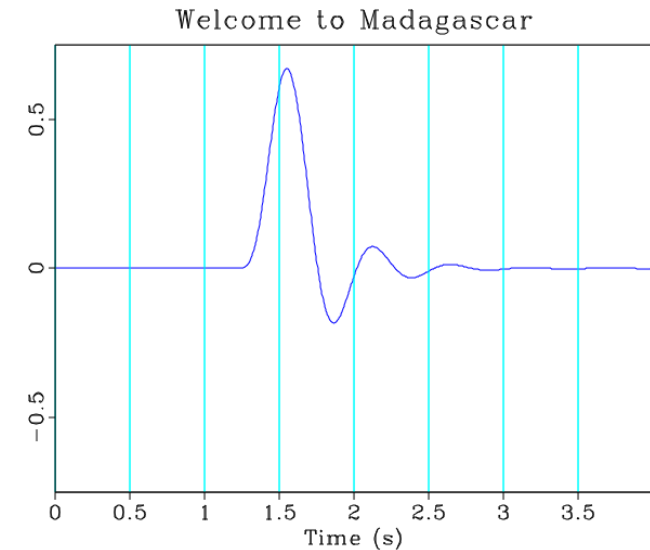
- Using Self-document

- Find which module you want to use by “sfdoc -k **keyword**”
- Read self-document by just running module name and find parameters, examples, and source code.
- Command-line Test

```
sfspike n1=1000 k1=300 | sfbandpass fhi=2 phase=1 | \  
sfwiggle clip=0.02 title="Welcome to Madagascar" | sfpopen
```

- Following examples is the best way to learn.

- Everything is reproducible.



- Geophysical data processing

- Assignment

- **Examples**

 - **Seismic data**

 - Gravity data

 - Controlled-source Electromagnetic data

- **Get data and convert to RSF format**
- **Initial data checking**
- **Initial signal analysis and quality control**
 - **Statics correction**
 - **First break mute**
 - **Subsampling with anti-aliasing filter**
 - **Ground-roll attenuation**
 - **Spiking deconvolution**
- **Convert from CSP to CMP**
- **Velocity analysis**
- **NMO and brute stacking**
- **Poststack denoising**
- **Toy migration**

The first step to do:

- **Open a new text file and name it as “SConstruct”.**
- **Begin to write your script for workflow.**

Get data and convert to RSF format

Always start here!

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

```
tgz = '2D_Land_data_2ms.tgz'
```

If you have had the data in the directory, comment these lines.

```
Fetch(tgz,  
      server='http://www.freeusp.org',  
      top='RaceCarWebsite/TechTransfer/Tutorials/Processing_2D',  
      dir='Data')
```

```
files = map(lambda x: 'Line_001.' + x, Split('TXT SPS RPS XPS sgy'))
```

```
Flow(files, tgz,  
      'gunzip -c $SOURCE | tar -xvf -', stdin=0, stdout=-1)
```

```
Flow('line tline', 'Line_001.sgy', 'segypread tfile=${TARGETS[1]}')
```


Check SEG-Y data header

\$scons tline.rs

\$sfheaderattr < tline.rs

91 headers, 71284 traces

	key	min	max	mean
trac	0	1 @ 0	71284 @ 71283	35642.5
tracr	1	1 @ 0	71284 @ 71283	35642.5
fldr	2	231 @ 0	481 @ 71000	356
tracf	3	-1 @ 0	282 @ 283	140.496
ep	4	32 @ 0	282 @ 71000	157
cdpt	6	1 @ 0	284 @ 283	142.5
trid	7	0 @ 0	1 @ 2	0.992958
nhs	9	1 @ 0	1 @ 0	1
scalel	19	-10000 @ 0	-10000 @ 0	-10000
scalco	20	-10000 @ 0	-10000 @ 0	-10000
counit	25	3 @ 0	3 @ 0	3
ns	38	1501 @ 0	1501 @ 0	1501
dt	39	2000 @ 0	2000 @ 0	2000
gain	40	3 @ 0	3 @ 0	3
igc	41	1 @ 0	1 @ 0	1
afilf	51	207 @ 0	207 @ 0	207
afils	52	298 @ 0	298 @ 0	298
hcf	56	207 @ 0	207 @ 0	207
hcs	58	298 @ 0	298 @ 0	298
year	59	1998 @ 0	1998 @ 0	1998

Initial data checking

```
Result('first','line',
```

```
'''
```

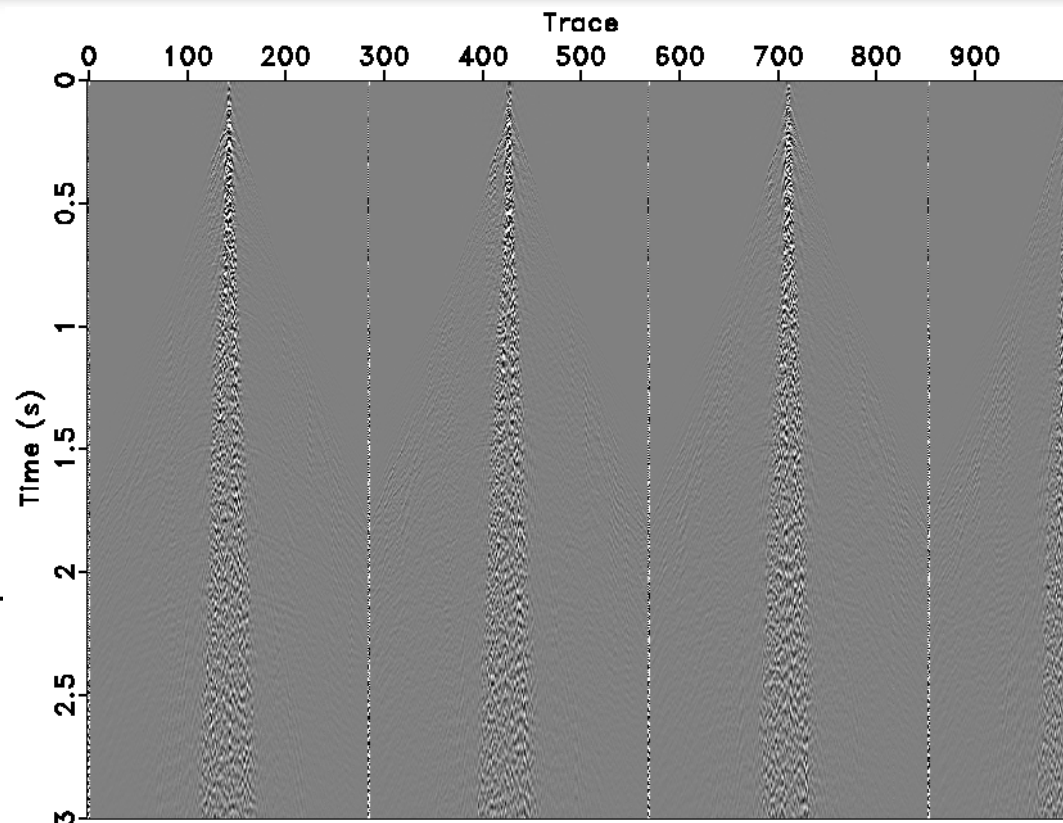
```
    window n2=1000 |
```

```
    agc rect1=250 rect2=100 |
```

```
    grey title="First 1000 traces"
```

```
    font=2 labelsz=6 labelfat=4
```

```
''')
```



First 1000 traces

```
$sfin line.rsf
```

```
line.rsf:
```

```
in="/your directory/line.rsf@"
```

```
esize=4 type=float form=native
```

```
n1=1501 d1=0.002 o1=0 label1="Time" unit1="s"
```

```
n2=71284 d2=1 o2=0 label2="Trace"
```

```
106997284 elements 427989136 bytes
```

```
lines = {'S':251,'R':782}
color = {'S':4, 'R':2}
for case in 'SR':
    # X-Y geometry
    Flow(case+'.asc','Line_001.%cPS' % case,
        '"awk 'NR > 20 {print $8, " ", $9}' "'')
    Flow(case,case+'.asc',
        '"
        echo in=$SOURCE data_format=ascii_float n1=2 n2=%d |
        dd form=native
        "' % lines[case],stdin=0)
    Plot(case,
        '"
        scale dscale=0.001 | dd type=complex |
        graph symbol=* title=%c plotcol=%d
        min1=684 max1=705 min2=3837 max2=3842
        "' % (case,color[case]))
```

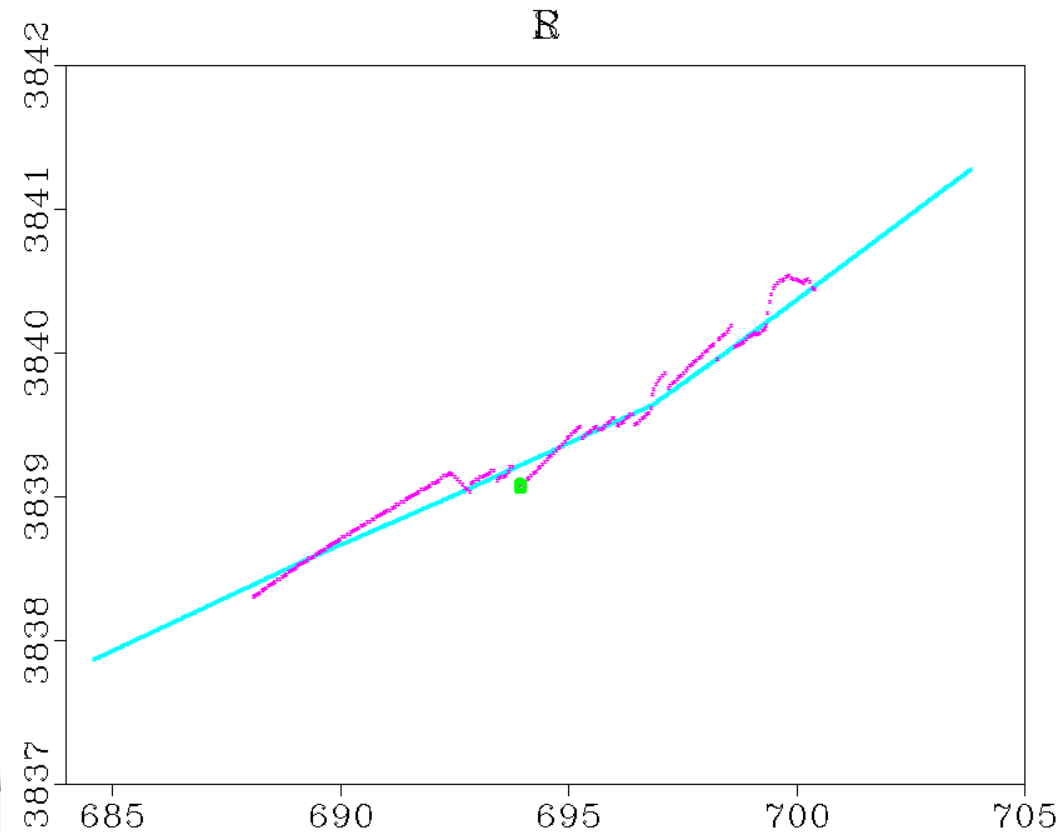
Plot shots and receivers position

```
Plot('s118', 'S',  
    ""
```

```
    window n2=1 f2=118 | scale dscale=0.001 | dd type=complex |  
    graph symbol=O wanttitle=n plotcol=3 symbolsz=4 plotfat=10  
    min1=684 max1=705 min2=3837 max2=3842
```

```
    "")
```

```
Result('SRO','R S s118','Overlay')
```



```
# Arrange receiver coordinates
```

```
shots = []
```

```
for shot in range(lines['S']):
```

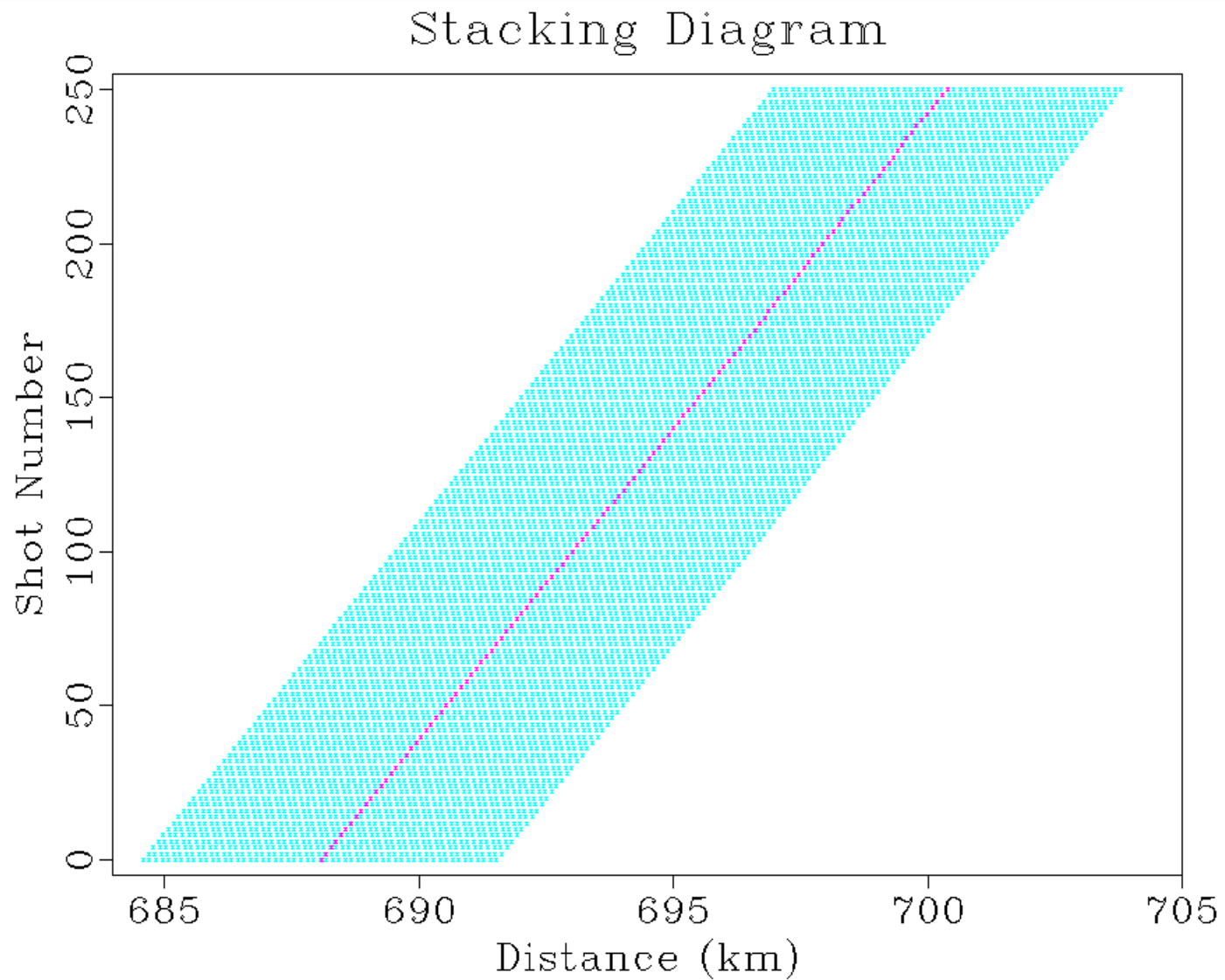
```
    line = 'line%d' % shot
```

```
    Flow(line,'R','window f2=%d n2=282' % (2*shot))
```

```
    shots.append(line)
```

```
Flow('rece',shots,'rcat axis=3 ${SOURCES[1:%d]}' % len(shots))
```

```
Plot('rec', 'rece',  
    ""  
    window n1=1 j2=5 j3=2 | scale dscale=0.001 |  
    rtoc | math output="input+I*x2" |  
    graph symbol=* plotcol=%d title="Stacking Diagram"  
    label1=Distance unit1=km label2="Shot Number"  
    min1=684 max1=705  
    "" % color['R'])  
Plot('sou', 'S',  
    ""  
    window n1=1 j2=2 | scale dscale=0.001 |  
    rtoc | math output="input+I*x1" |  
    graph symbol=* plotcol=%d wanttitle=n wantaxis=n min1=684 max1=705  
    "" % color['S'])  
Result('diagram', 'rec sou', 'Overlay')
```



Display one shot using true geometry

```
Flow('sour', 'S', 'spray axis=2 n=282 o=0 d=1")
```

```
Flow('sx', 'sour', 'window n1=1 | scale dscale=0.001')
```

```
Flow('sy', 'sour', 'window n1=1 f1=1 | scale dscale=0.001')
```

```
Flow('rx', 'rece', 'window n1=1 | scale dscale=0.001')
```

```
Flow('ry', 'rece', 'window n1=1 f1=1 | scale dscale=0.001')
```

```
Flow('offset', 'sx sy rx ry',
```

```
""
```

```
  math SX=${SOURCES[0]} SY=${SOURCES[1]}
```

```
  RX=${SOURCES[2]} RY=${SOURCES[3]}
```

```
  output="sqrt((RX-SX)^2+(RY-SY)^2)"
```

```
  "")
```


Map to regular shot gathers

```
Flow('lines', 'line',  
    ""
```

```
    intbin xk=cdpt yk=fldr | window f2=2 |
```

```
    put
```

```
    label3=Source d3=0.05 o3=688 unit3=km
```

```
    label2=Offset d2=0.025 o2=-3.5 unit2=km
```

```
    label1=Time unit1=s
```

```
    "")
```

```
Result('lines',  
    ""
```

```
    transp memsize=1000 plane=23 |
```

```
    byte gainpanel=each |
```

```
    grey3 frame1=500 frame2=100 frame3=120 flat=n
```

```
    title="Raw Data"
```

```
    "")
```

```
[yourcomputer directory]$ sfin lines.rsf
```

```
lines.rsf:
```

```
in="$RSFDATA/.../.../lines.rsf@"
```

```
esize=4 type=float form=native
```

```
n1=1501 d1=0.002 o1=0 label1="Time" unit1="s"
```

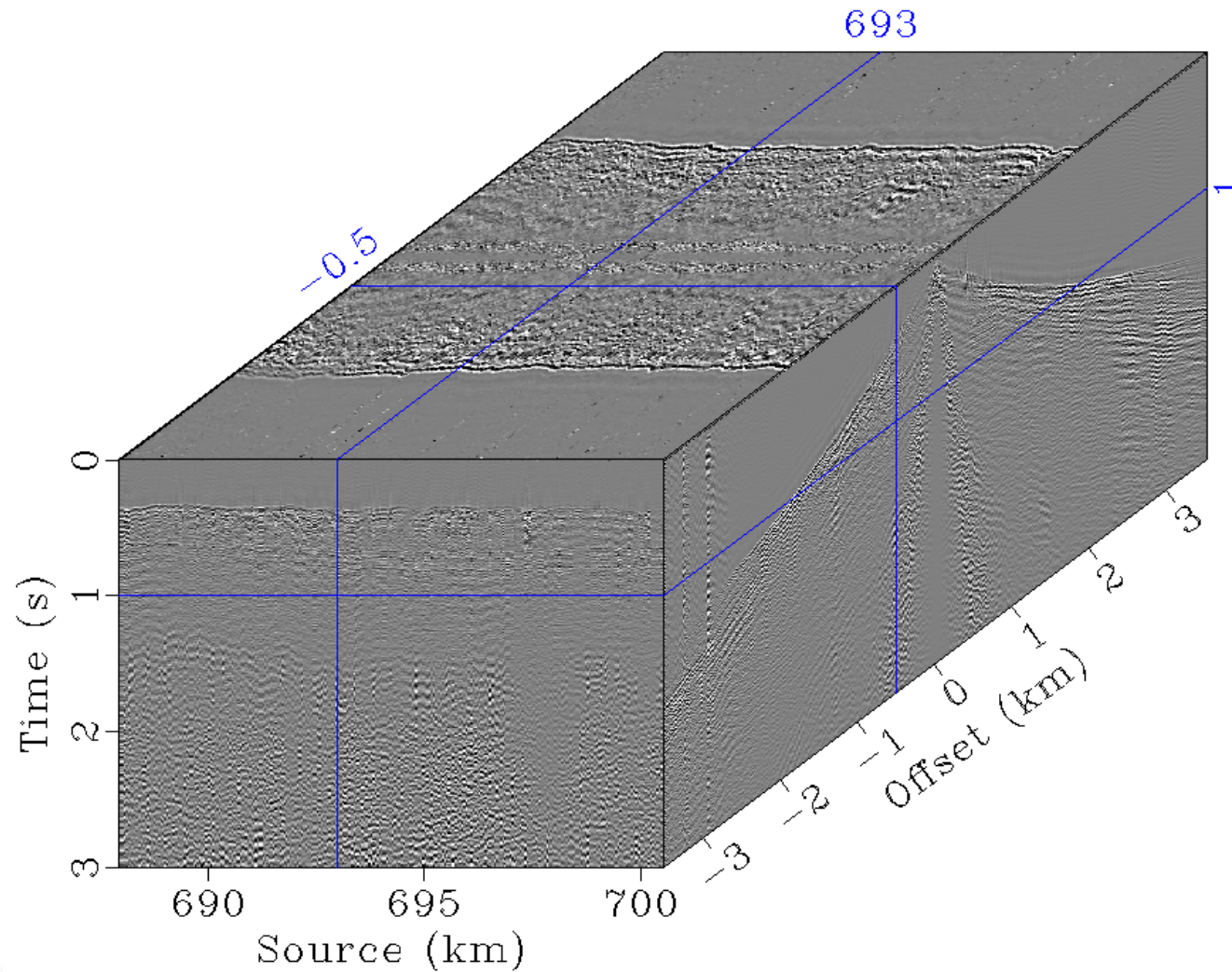
```
n2=282 d2=0.025 o2=-3.5 label2="Offset" unit2="km"
```

```
n3=251 d3=0.05 o3=688 label3="Source" unit3="km"
```

```
_106243782 elements 424975128 bytes
```

2-D shot gathers (assuming regular coordinates)

Raw Data

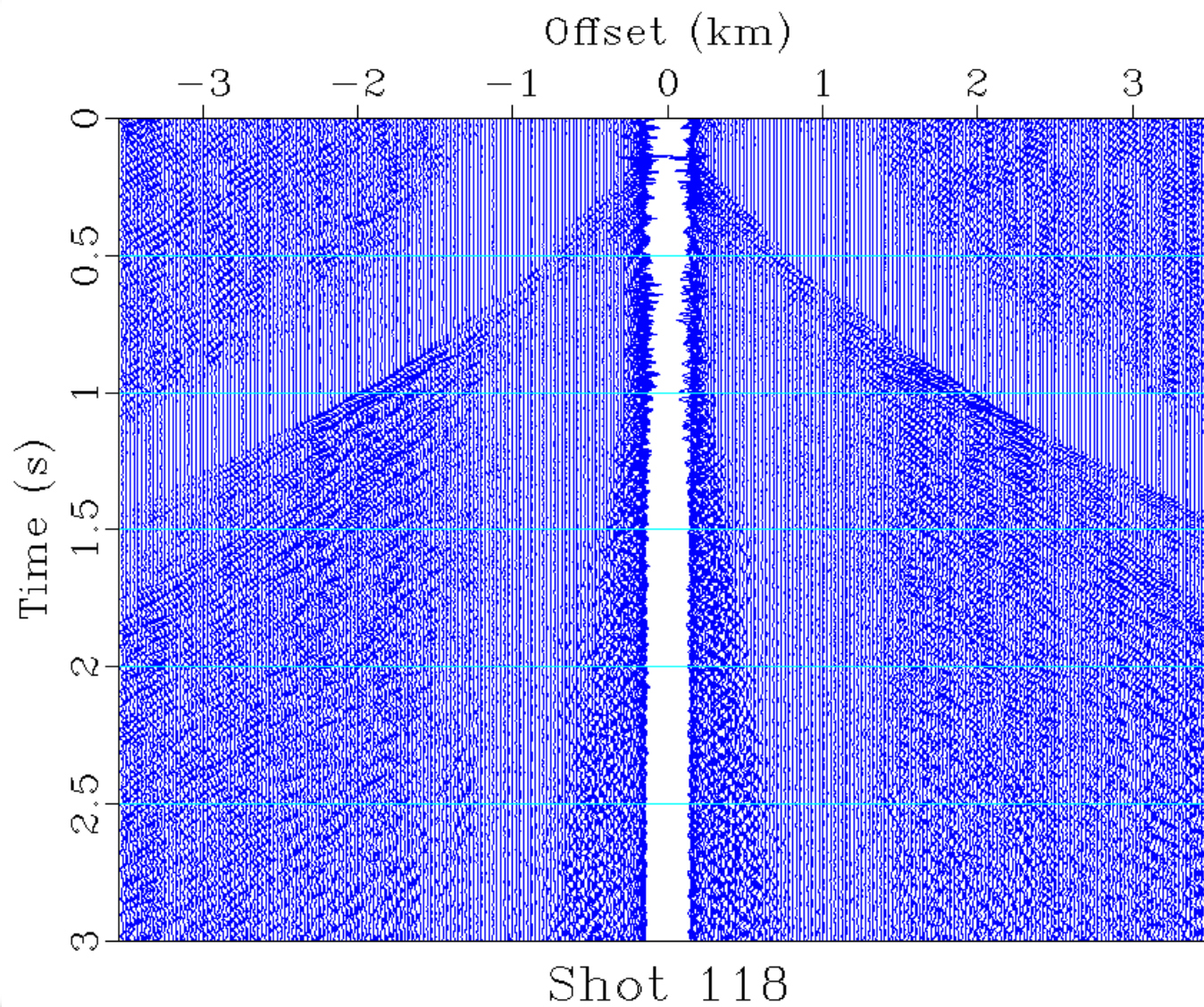


Plot No. 118 shot gather using true geometry



```
Flow('shot118', 'lines', 'window n2=1 f2=118')
Flow('offset118', 'offset', 'window n2=1 f2=118')
Flow('boff1', 'offset118', 'window n1=141 | math output="-input"')
Flow('boff2', 'offset118', 'window f1=141')
Flow('boff118', 'boff1 boff2', 'cat axis=1 ${SOURCES[1]} ')
Result('shot118', 'shot118 boff118',
    ""
    agc rect1=50 rect2=50 |
    wiggle xpos=${SOURCES[1]} transp=y yreverse=y poly=y
    wherexlabel=t wheretitle=b title="Shot 118"
    "")
```

Plot No. 118 shot gather



Convert shots to CMPs

(assuming regular geometry, improve by yourself)



```
Flow('rcmps mask', 'lines',
```

```
""
```

```
shot2cmp mask=${TARGETS[1]} half=n |
```

```
put o2=-1.75 d2=0.05 label2="Half-offset"
```

```
""
```

```
Result('rcmps',
```

```
""
```

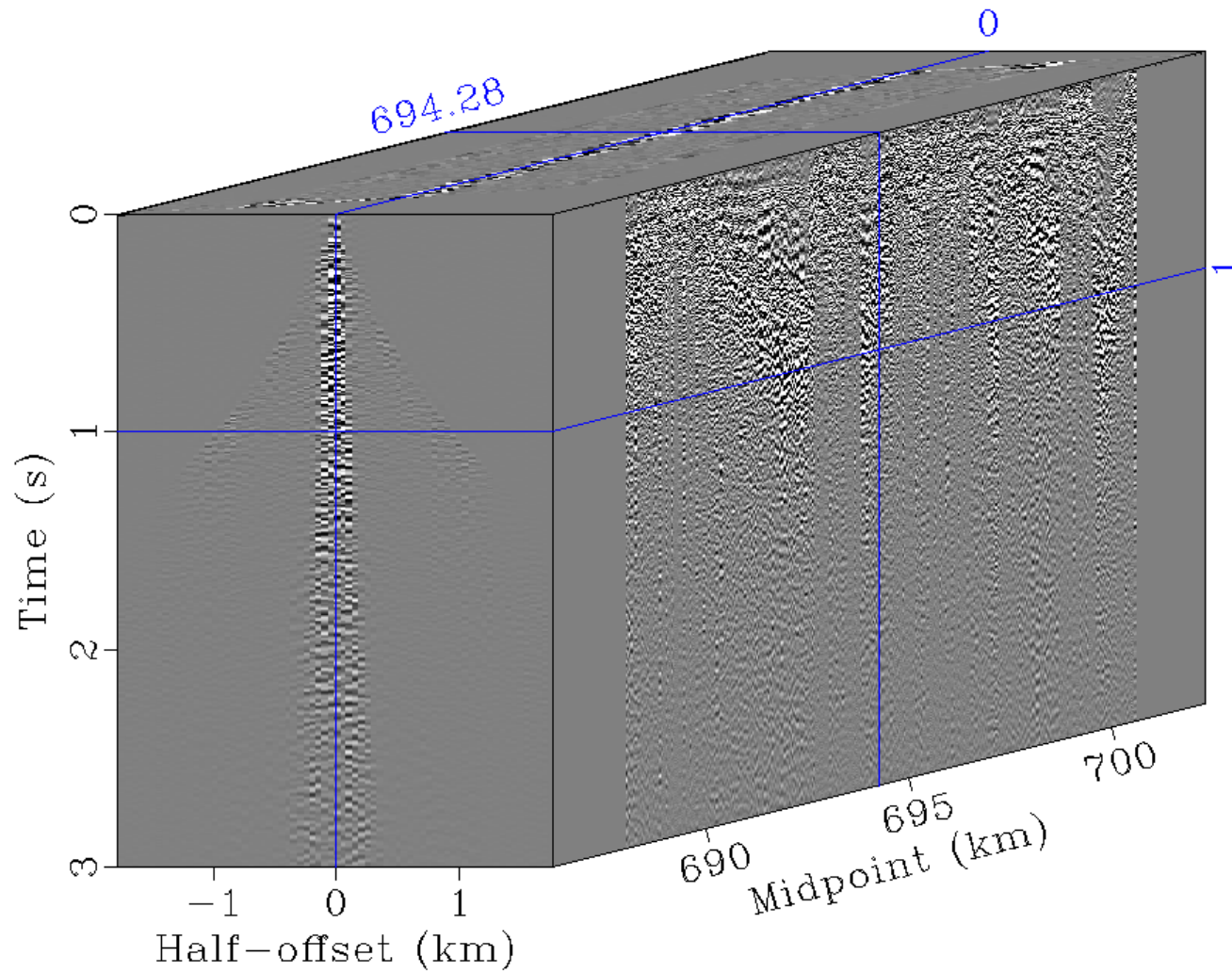
```
byte gainpanel=each | window j3=3 |
```

```
grey3 frame1=500 frame2=35 frame3=214 flat=n
```

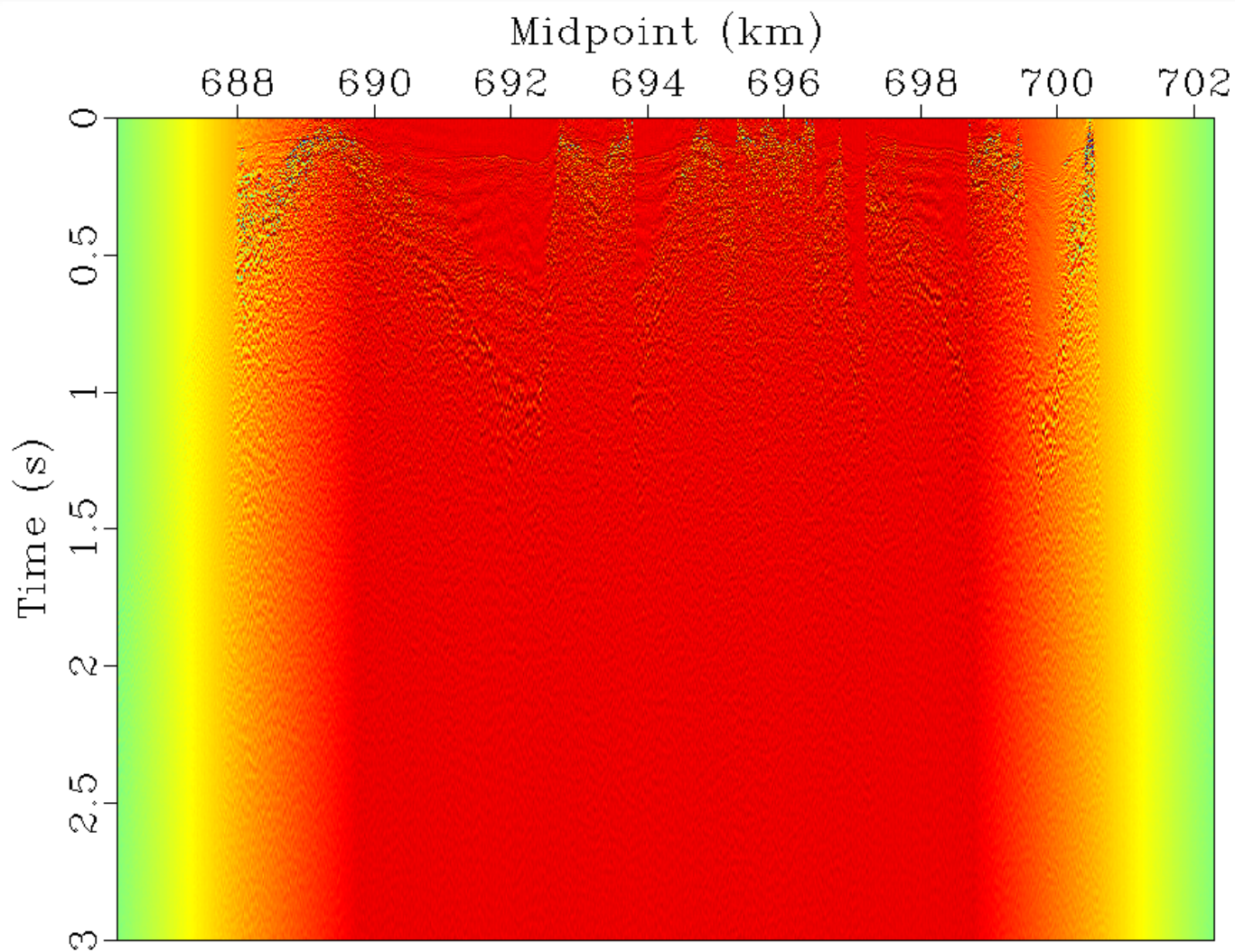
```
point1=0.8 point2=0.4 title="CMPs"
```

```
""
```


CMPs



```
Flow('fold', 'mask', 'dd type=float | stack axis=1 norm=n')  
Flow('rstack', 'rcmps', 'stack')  
Result('rstack', 'fold rstack',  
    ""  
    spray axis=1 n=1501 d=0.002 o=0 label=Time unit=s |  
    add scale=1,1000 ${SOURCES[1]} |  
    grey color=j title="Raw Stack (with Fold)"  
    "")
```



Raw Stack (with Fold)

for case in 'SR':

```
stat = case+'-statics'
```

```
Flow(stat+'.asc','Line_001.%cPS' % case,"awk 'NR > 20 {print $10}' """)
```

```
Flow(stat,stat+'.asc',
```

```
""
```

```
echo in=$SOURCE data_format=ascii_float n1=%d | dd form=native |  
scale dscale=%g | put label=Time unit=s
```

```
"" % (lines[case],1./1900),stdin=0)
```

Line_001.TXT

Reference Velocity.....: 1900.0

```
# Compute statics corrections
```

```
shots = []
```

```
for shot in range(lines['S']):
```

```
    line = 'statics-line%d' % shot
```

```
    Flow(line,'R-statics','window f1=%d n1=282' % (2*shot))
```

```
    shots.append(line)
```

```
Flow('rstat',shots,'cat axis=2 ${SOURCES[1:%d]}' % len(shots))
```

```
Flow('statics','S-statics rstat',
```

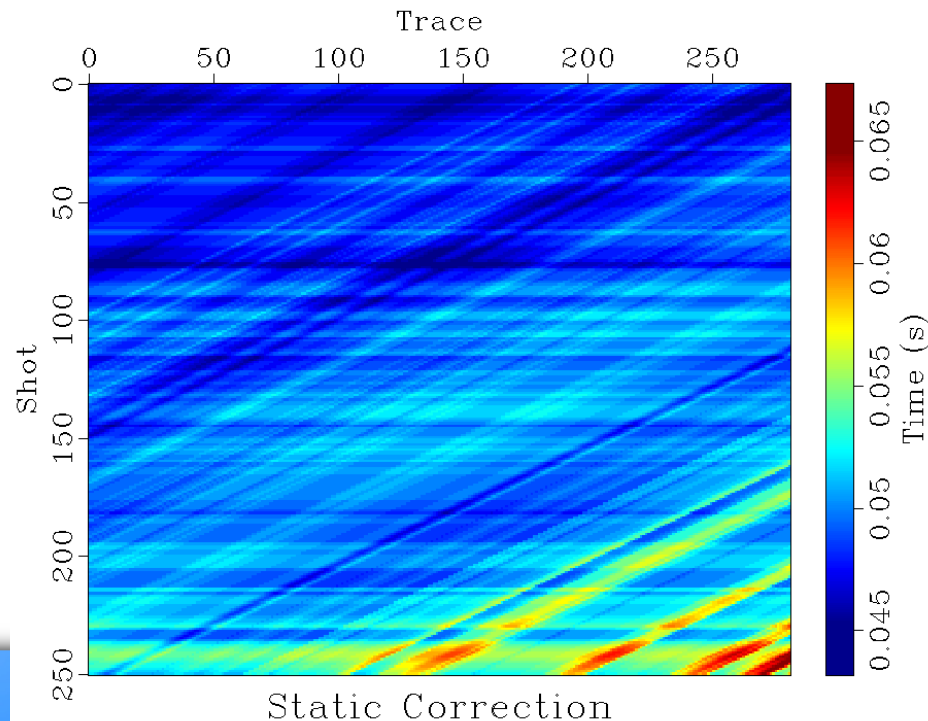
```
    ""
```

```
    spray axis=1 n=282 o=0 d=1 |
```

```
    add ${SOURCES[1]}
```

```
    "")
```

```
Result('statics',  
      ""  
      grey color=j scalebar=y title="Static Correction"  
      transp=n label1=Trace label2=Shot bias=0.055  
      "")  
# Apply statics correction  
Flow('lines2','lines statics','datstretch datum=${SOURCES[1]}')
```



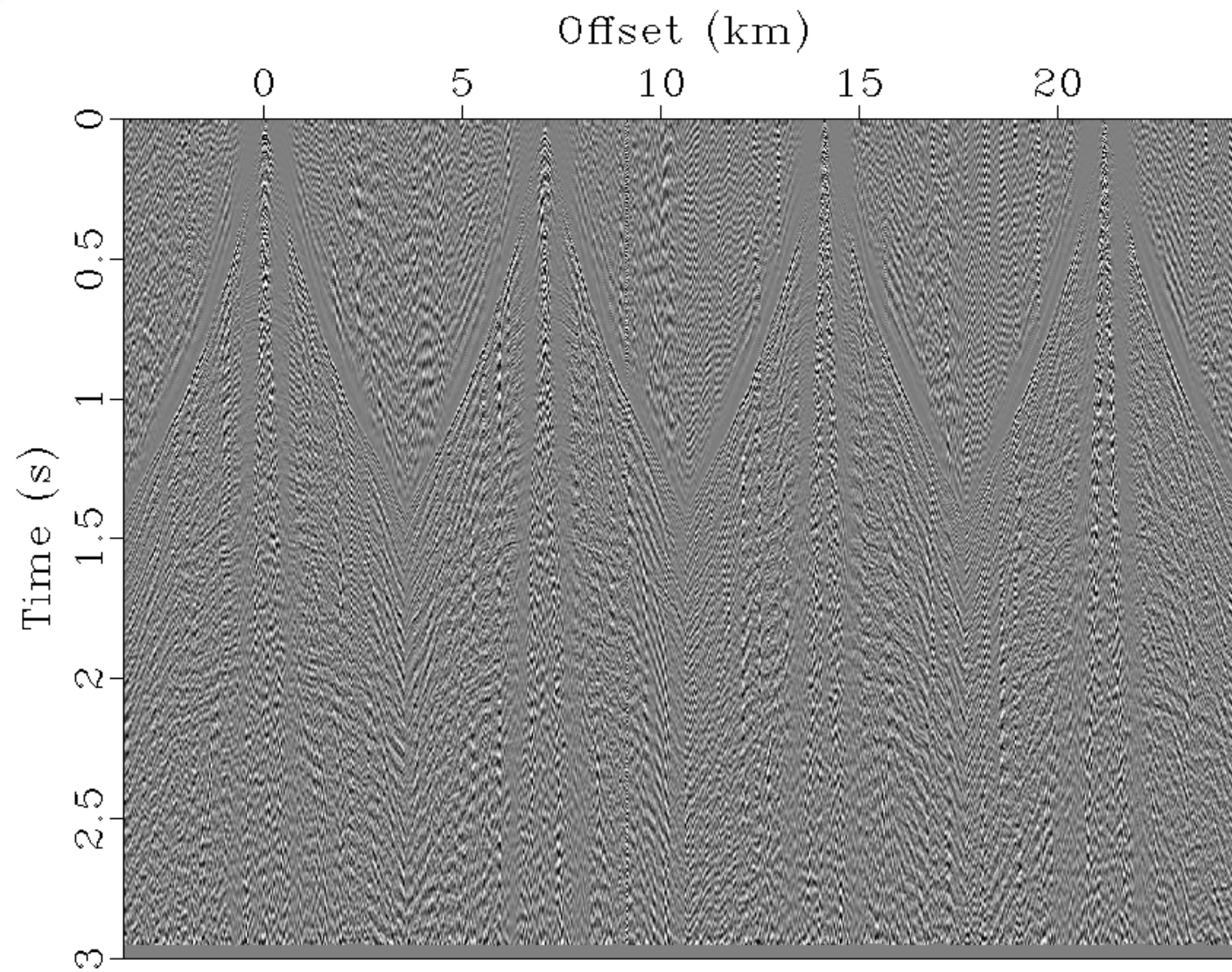
Select 4 shots every tenth sequential shot

```
Flow('inpmute', 'lines2',  
    ""
```

```
    window f2=198 j2=10 n3=4  
    ""
```

```
Result('inpmute',  
    ""
```

```
    put n2=1128 n3=1 |  
    agc rect1=50 rect2=20 | grey wanttitle=n  
    ""
```



Select muting parameter for background noise

```
Flow('outmute', 'inpmute',  
    ""
```

```
    mutter t0=0. v0=5.2
```

```
    ""
```

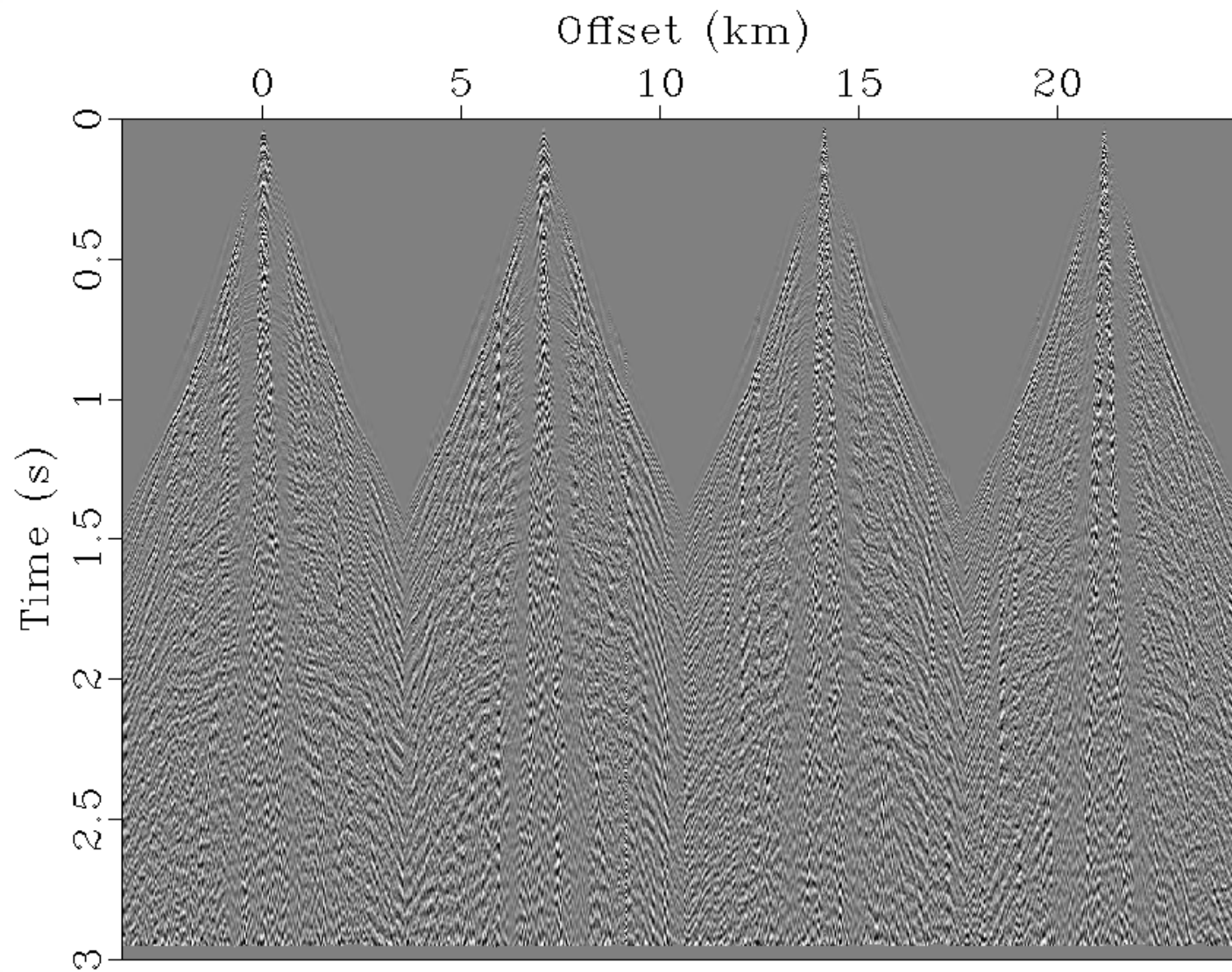
```
Result('outmute',
```

```
    ""
```

```
    put n2=1128 n3=1 |
```

```
    agc rect1=50 rect2=20 | grey wanttitle=n
```

```
    ""
```

Apply muting on all shots

First break muting for all shots

```
Flow('mutes', 'lines2',  
    ""  
    mutter t0=0. v0=5.2 |  
    "")
```


Shot 198

Flow('shot198', 'mutes', 'window n3=1 f3=198')

Plot('shot198', 'grey title="Shot 198" labelfat=4 titlefat=4')

Spectra

Flow('spec198', 'shot198', 'spectra2')

Plot('spec198',

'''

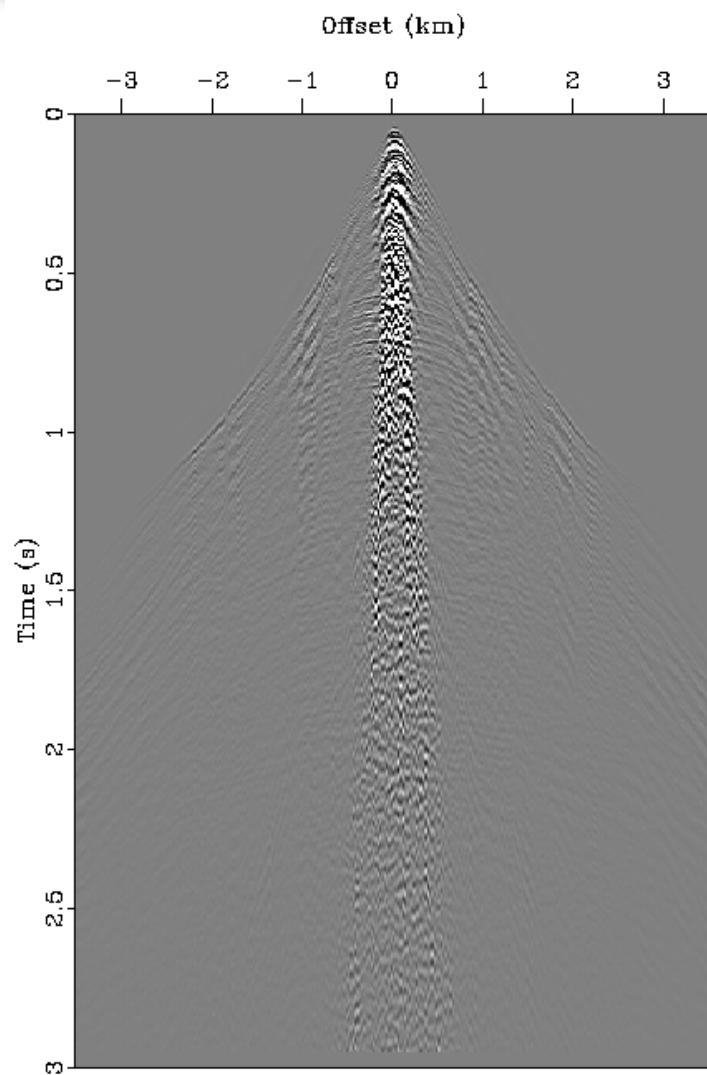
grey color=j yreverse=n title="Spectra 198" bias=0.08

label1=Frequency unit1=Hz label2=Wavenumber unit2=1/km

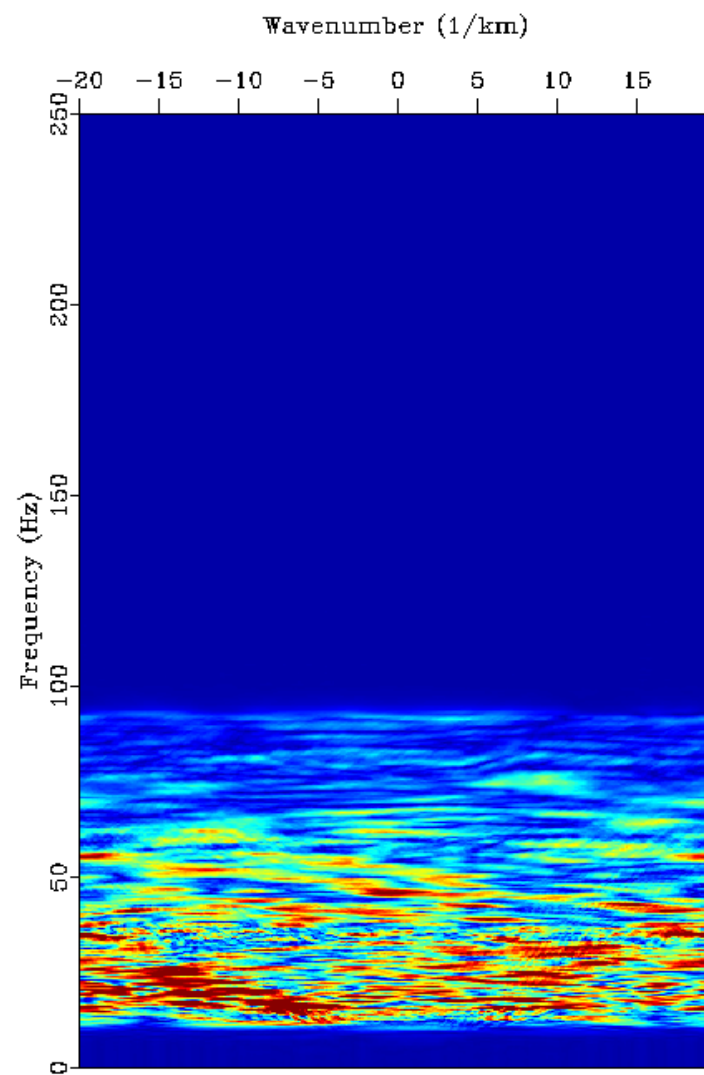
labelfat=4 titlefat=4

''')

Result('input198', 'shot198 spec198', 'SideBySideAniso')



Shot 198



Spectra 198

Test anti-aliasing filter

Flow('spike',None, 'spike n1=1501 d1=0.002 o1=0 k1=750 mag=1 nsp=1')

Flow('bandp', 'spike', 'bandpass flo=3 fhi=125 nphi=8')

Result('sbandp', 'bandp',

'''

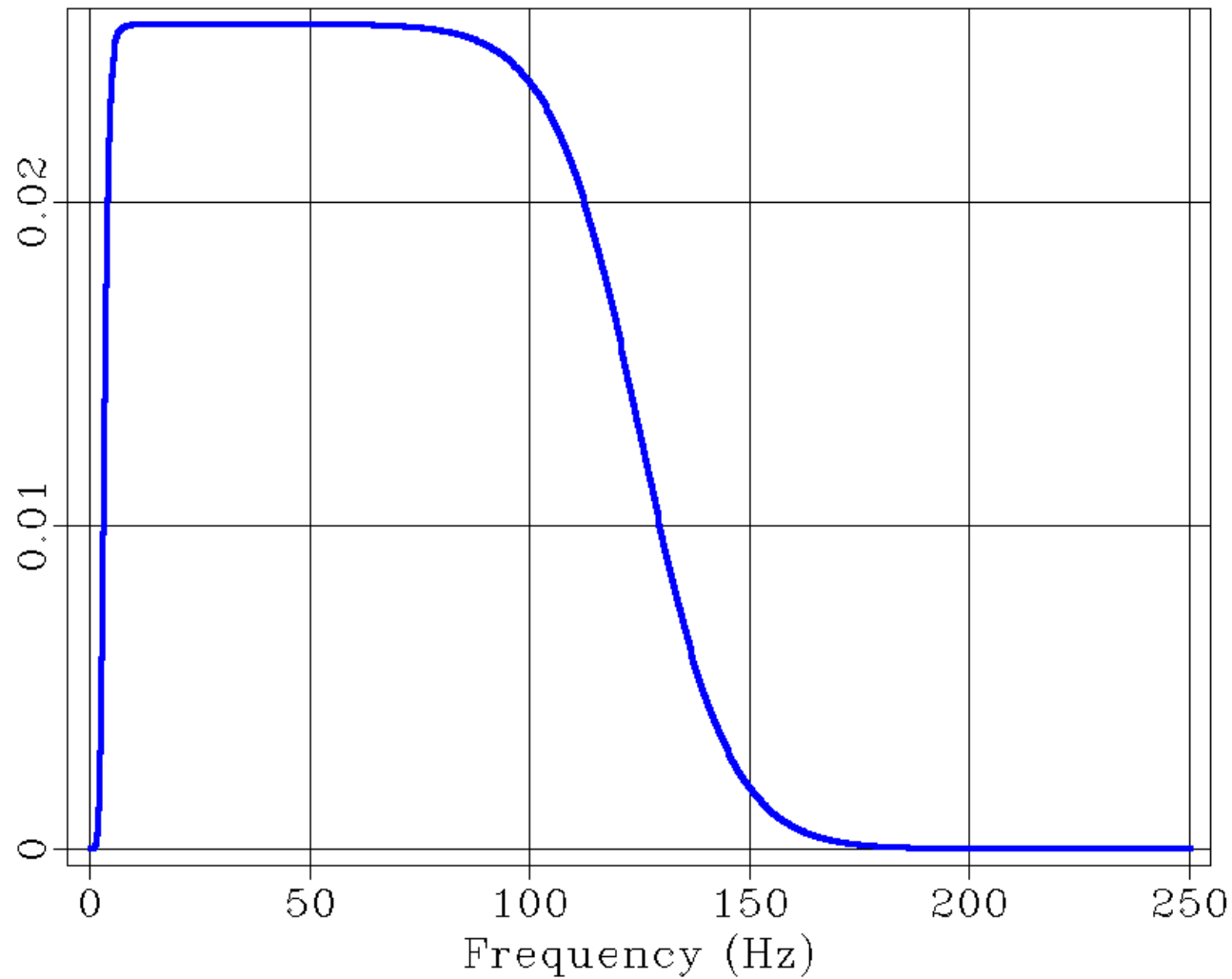
spectra |

graph title="Transfer function"

labelsz=8 plotfat=10 grid=y

''')

Transfer function

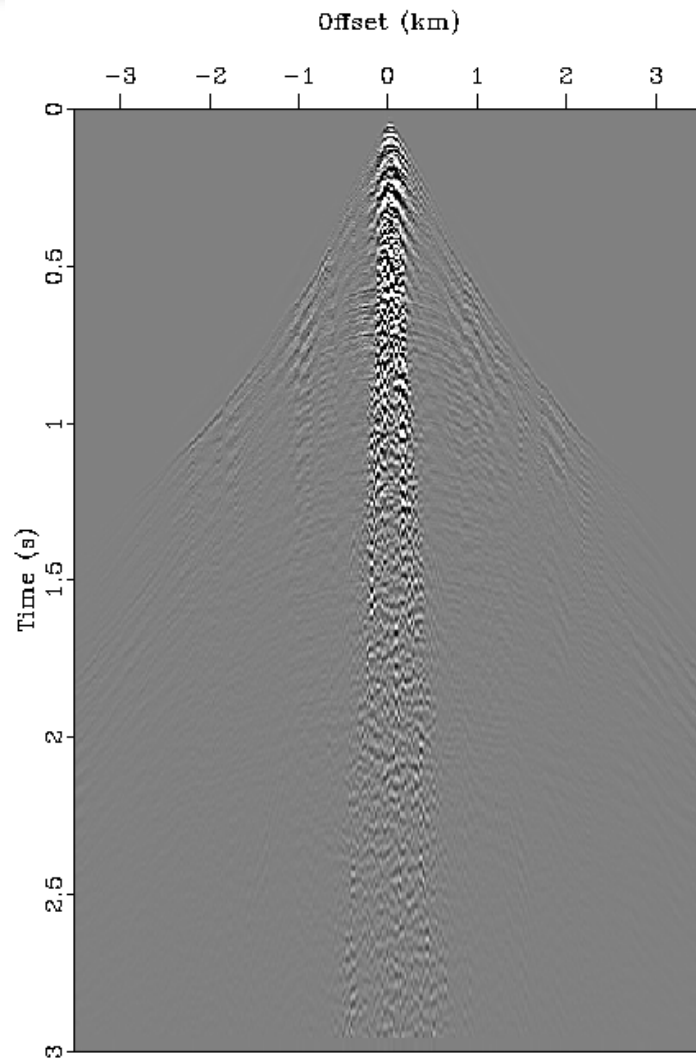


Subsampling all shots to 4ms

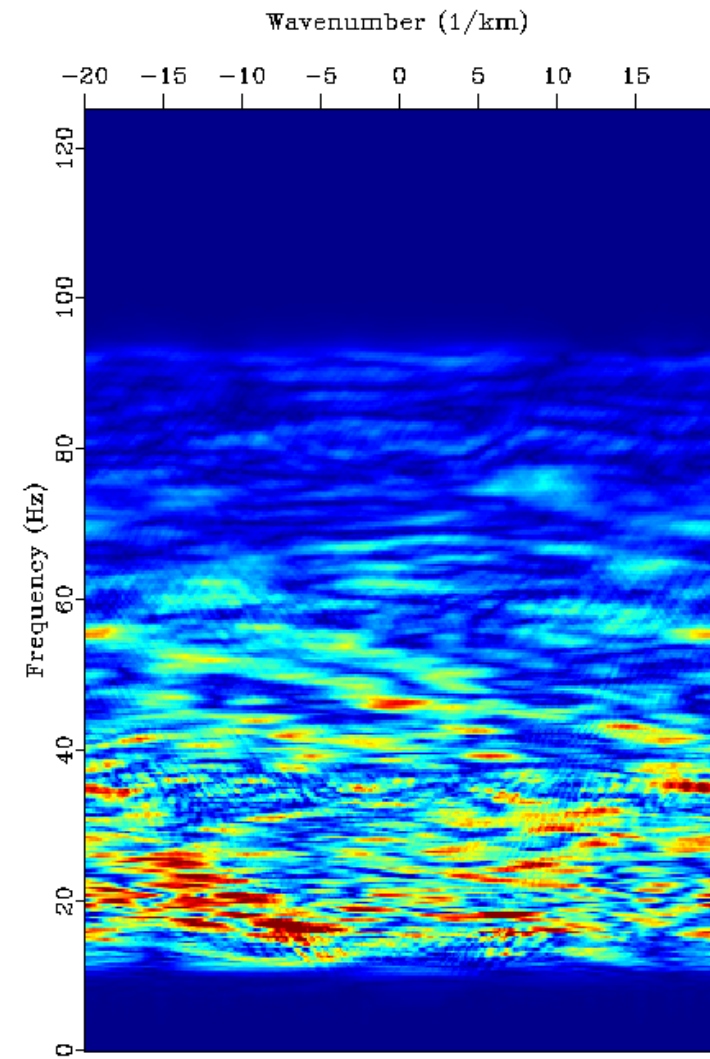
```
Flow('subsample', 'mutes',  
    'bandpass flo=3 fhi=125 nphi=8 | window j1=2')  
Flow('subshot198', 'subsample', 'window n3=1 f3=198')  
Plot('subshot198', 'grey title="Subsampled 198" labelfat=4 titlefat=4')
```

Spectra

```
Flow('subspec198', 'subshot198', 'spectra2')  
Plot('subspec198',  
    ""  
    grey color=j yreverse=n title="Spectra 198" bias=0.08  
    label1=Frequency unit1=Hz label2=Wavenumber unit2=1/km  
    labelfat=4 titlefat=4  
    "")  
Result('output198', 'subshot198 subspec198', 'SideBySideAniso')
```



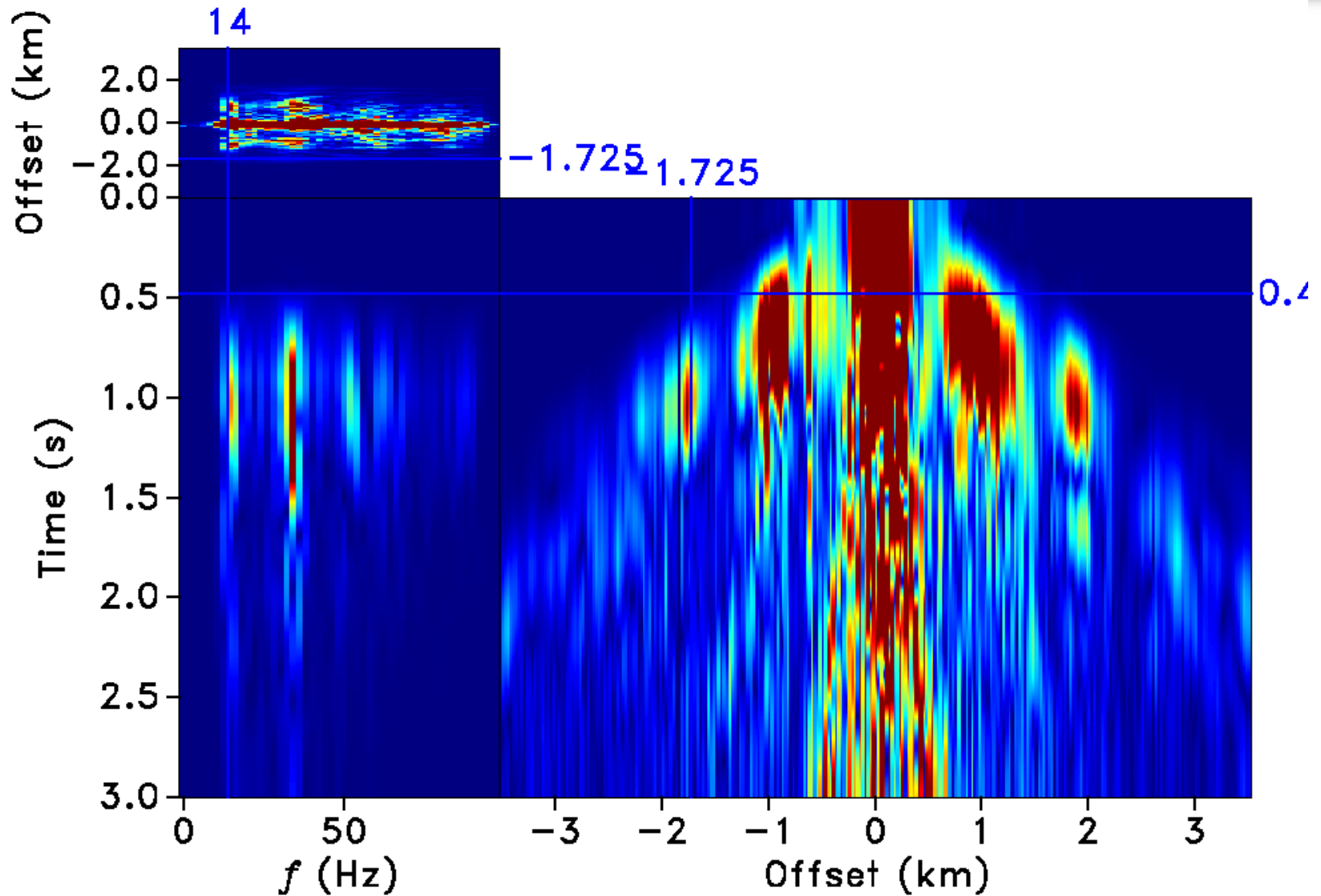
Subsampled 198



Spectra 198

```
Flow('ltft198', 'subshot198',  
    ""  
    ltft rect=20 verb=n nw=50 dw=2 niter=50  
    "")  
Result('ltft198',  
    ""  
    math output="abs(input)" | real |  
    byte allpos=y gainpanel=100 pclip=99 |  
    grey3 color=j frame1=120 frame2=7 frame3=71 label1=Time flat=y  
    unit1=s label3=Offset label2="\F5 f \F-1" unit3=km  
    screenht=10 screenratio=0.7 parallel2=n format2=%3.1f  
    point1=0.8 point2=0.3 wanttitle=n labelfat=4 font=2 titlefat=4  
    "")
```


Time-frequency (TF) spectra



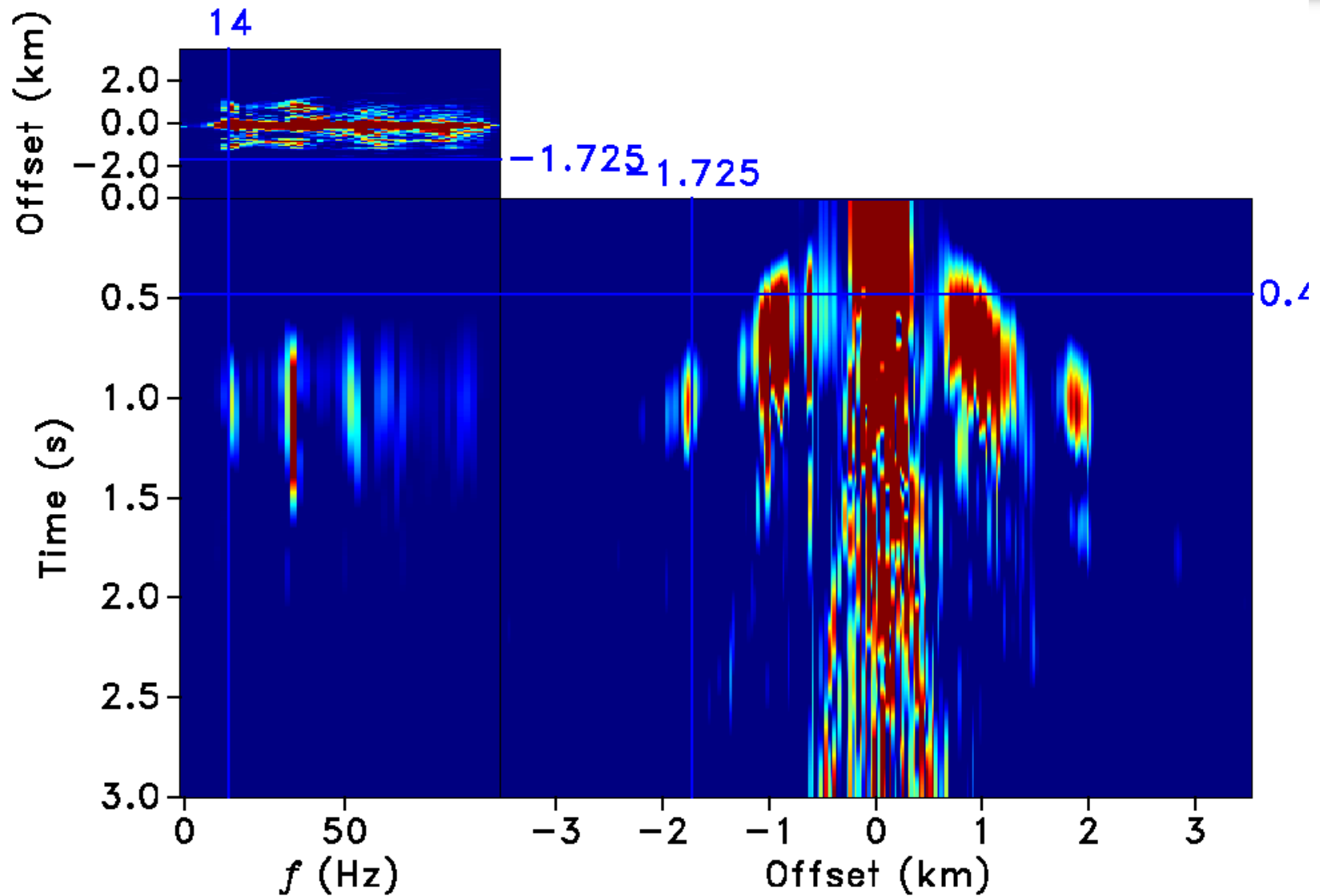
```
Flow('thr198', 'ltft198',  
    ""
```

```
    transp plane=23 memsize=1000 |  
    threshold2 pclip=25 verb=y |  
    transp plane=23 memsize=1000  
    "")
```

```
Result('thr198',  
    ""
```

```
    math output="abs(input)" | real |  
    byte allpos=y gainpanel=100 pclip=99 |  
    grey3 color=j frame1=120 frame2=7 frame3=71 label1=Time flat=y  
    unit1=s label3=Offset label2="\F5 f \F-1" unit3=km  
    screenht=10 screenratio=0.7 parallel2=n format2=%3.1f  
    point1=0.8 point2=0.3 wanttitle=n labelfat=4 font=2 titlefat=4  
    "")
```

Thresholded TF spectra



```
Flow('noise198', 'thr198', 'ltft inv=y | mutter t0=-0.5 v0=0.7')
```

```
Plot('noise198',
```

```
    'grey title="Ground-roll 198" unit2=km labelfat=4 titlefat=4')
```

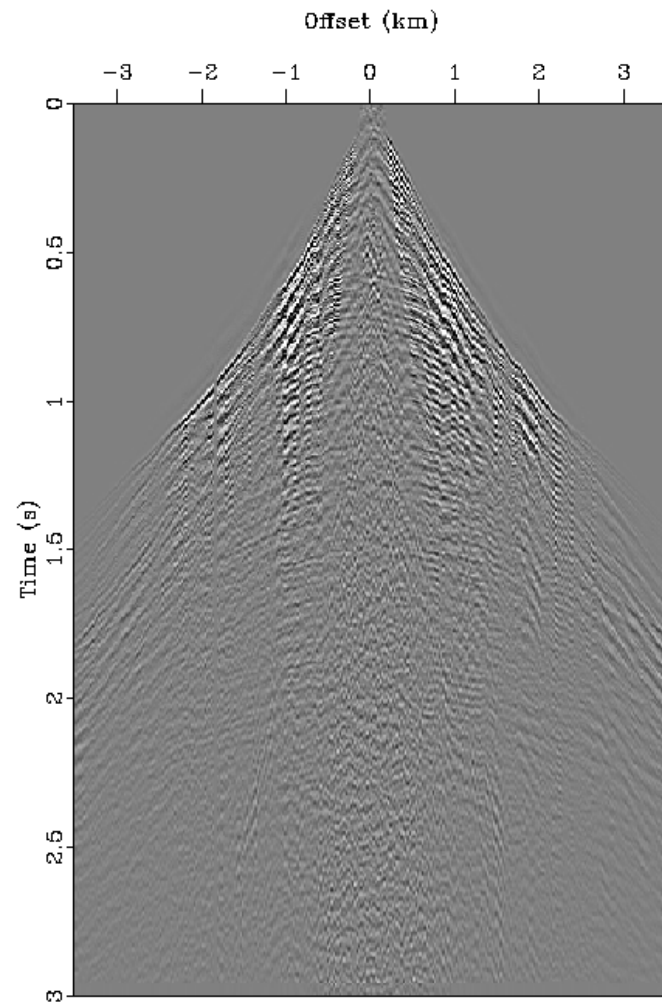
```
Flow('signal198', 'subshot198 noise198', 'add scale=1,-1 ${SOURCES[1]} ')
```

```
Plot('signal198',
```

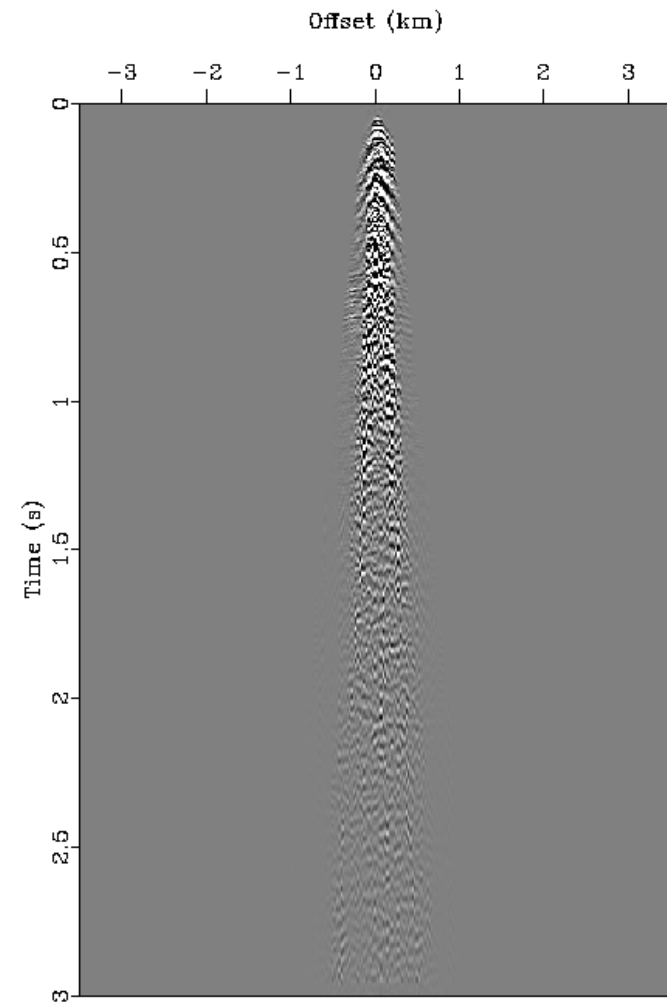
```
    'grey title="Signal 198" labelfat=4 titlefat=4')
```

```
Result('sn198', 'signal198 noise198', 'SideBySideAniso')
```

Signal and noise separation



Signal 198



Ground-roll 198

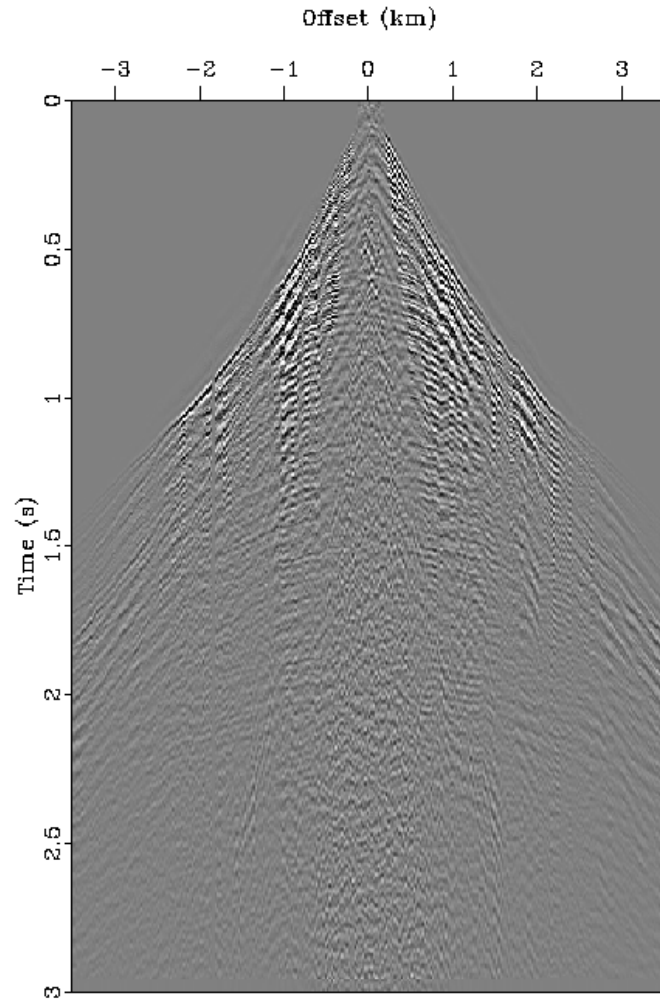
Spiking Deconvolution

```
Flow('decon198','signal198',  
      'pef minlag=.004 maxlag=.140 pnoise=.01 mincorr=0 maxcorr=3')
```

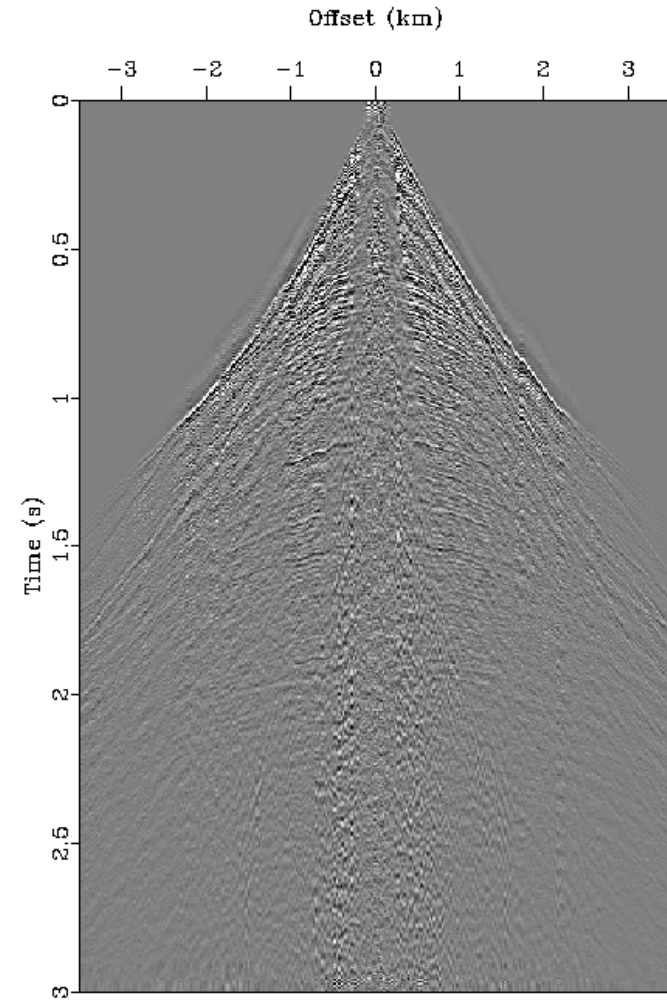
```
Plot('decon198',  
      'grey title="Wiener PEF 198" labelfat=4 titlefat=4')
```

```
Result('sd198','signal198 decon198','SideBySideAniso')
```

Wiener prediction-error filter



Signal 198



Wiener PEF 198

Process the whole dataset (cheating here!)



```
Flow('noise','subsample',
```

```
""
```

```
threshold2 pclip=20 verb=y
```

```
""
```

```
Flow('signals','subsample noise','add scale=1,-1 ${SOURCES[1]}')
```

Spiking Deconvolution on entire 2-D line

```
Flow('decons','signals',
```

```
'pef minlag=.004 maxlag=.140 pnoise=.01 mincorr=0 maxcorr=3')
```

Process the whole dataset (try by yourself)

```
## # Apply LTFT on entire 2-D line
## Flow('ltfts', 'subsample',
##      ""
##      ltft rect=20 verb=y nw=50 dw=2 niter=50
##      "",split=[3,282],reduce="cat axis=4")
## Flow('thresholds', 'ltfts',
##      ""
##      transp plane=23 memsize=1000 | threshold2 pclip=25 verb=y |
##      # # transp plane=23 memsize=1000
##      "",split=[3,251])
## Flow('noise', 'thresholds',
##      ""
##      ltft inv=y |
##      mutter t0=-0.5 v0=0.7 |
##      "")
## Flow('signals', 'subsample noise', 'add scale=1,-1 ${SOURCES[1]} ')
```

Convert shots to CMPs

(assuming regular geometry, improve by yourself)



Convert shots to CMPs

Flow('cmps', 'signals',

'''

mutter v0=3. |

shot2cmp half=n | put o2=-1.75 d2=0.05 label2="Half-offset"

''')

Result('cmps',

'''

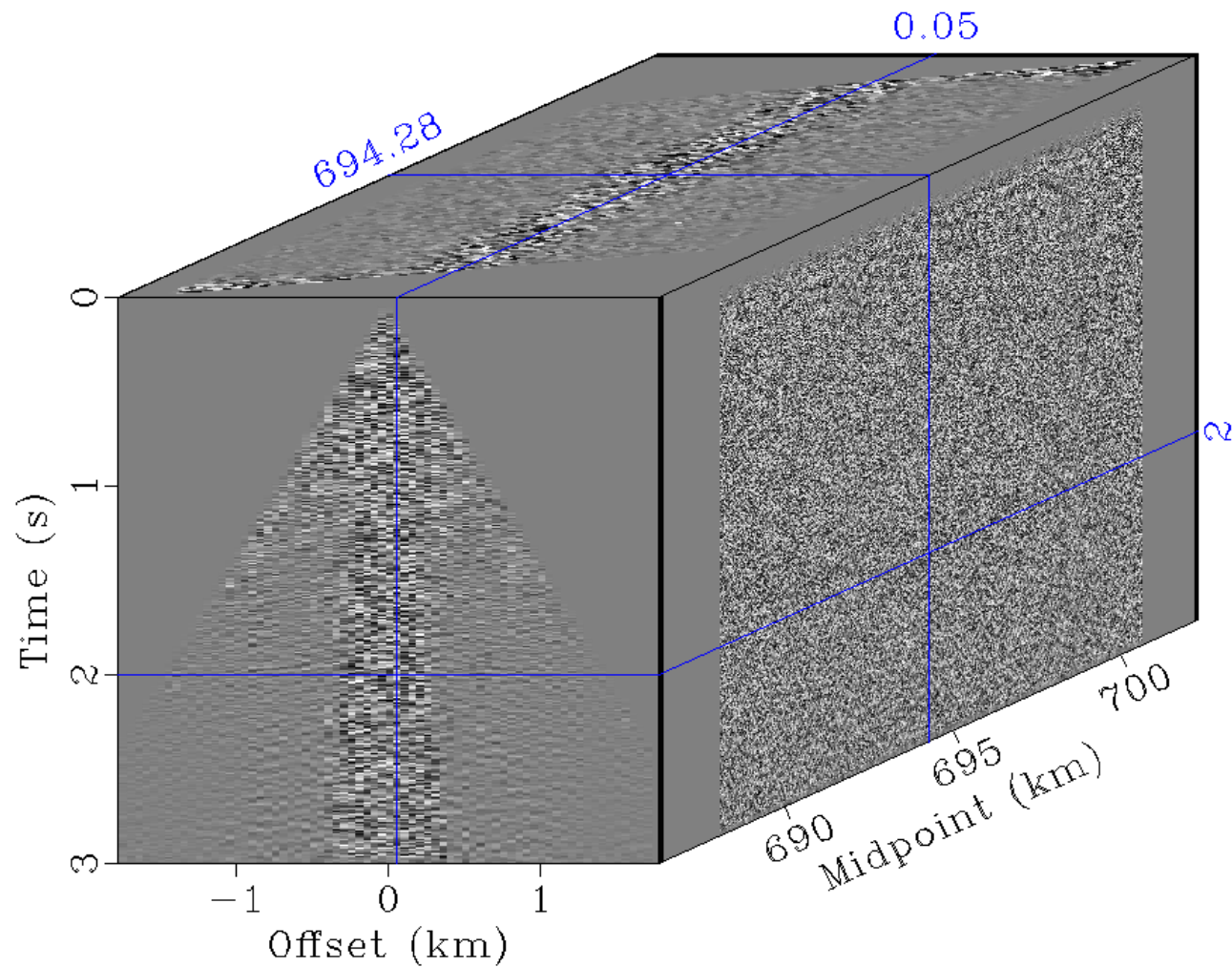
byte gainpanel=each |

grey3 frame1=500 frame2=36 frame3=642 flat=n

title="CMP gathers" point1=0.7 label2=Offset label3=Midpoint

''')

CMP gathers



Set up velocity scan parameters

$v_0 = 1.0$

$dv = 0.05$

$nv = 75$

Velocity scanning for all CMP gathers

Flow('scn', 'cmps',

'''

vscan semblance=y v0=%g nv=%d dv=%g half=y str=0 |

mutter v0=0.9 t0=-4.5 inner=y

''' % (v0,nv,dv),split=[3,1285])

Flow('vel', 'scn', 'pick rect1=15 rect2=25 gate=100 an=10 | window')

Result('vel',

'''

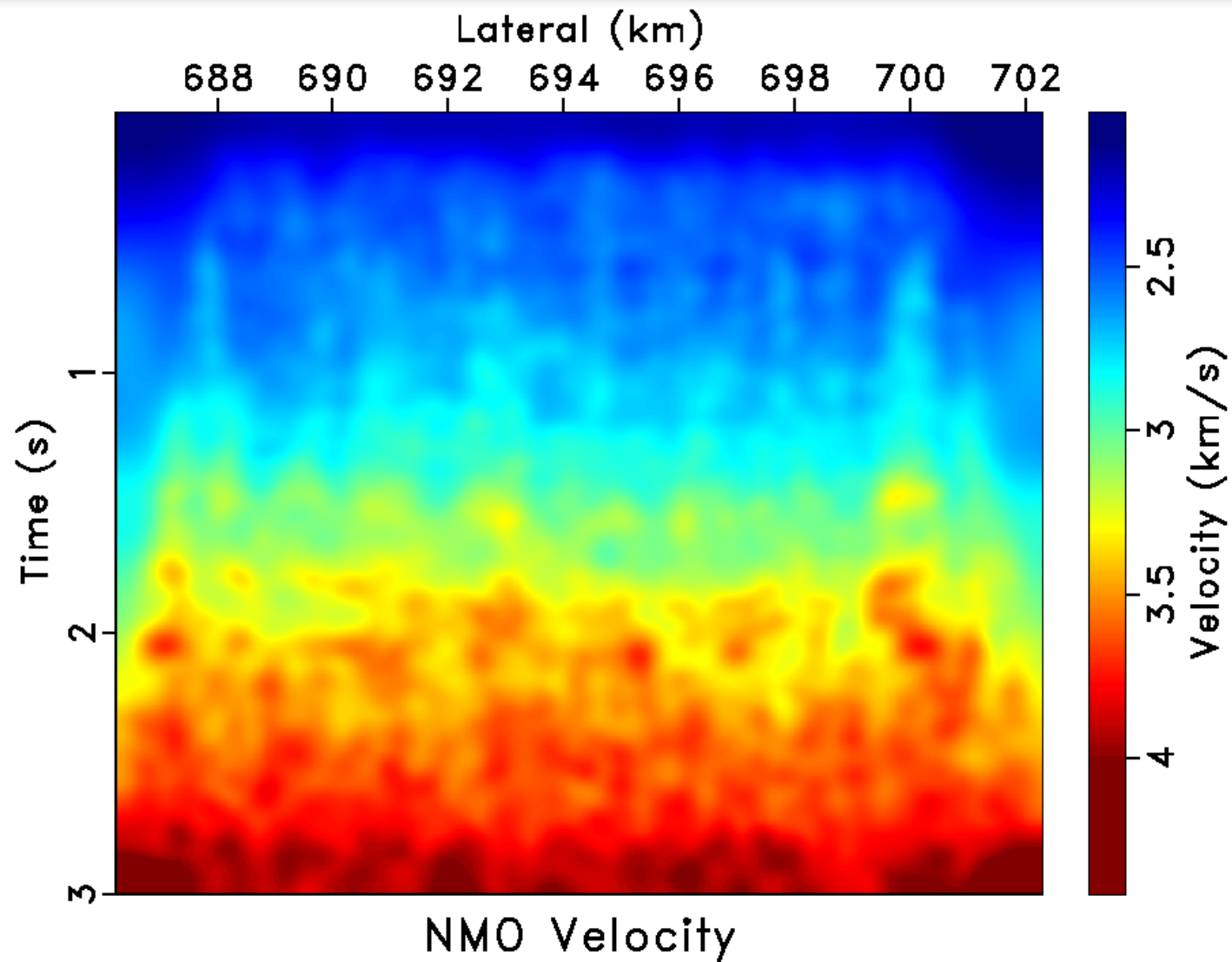
grey title="NMO Velocity" label1="Time" label2="Lateral"

color=j scalebar=y allpos=y bias=2.1 barlabel="Velocity"

barreverse=y o2num=1 d2num=1 n2tic=3 labelfat=4 font=2 titlefat=4

''')

Display NMO velocity



NMO

Flow('nmo', 'cmps vel',

'''

nmo velocity=\${SOURCES[1]} half=y

''')

Result('nmo',

'''

byte gainpanel=each |

grey3 frame1=500 frame2=36 frame3=642 flat=n

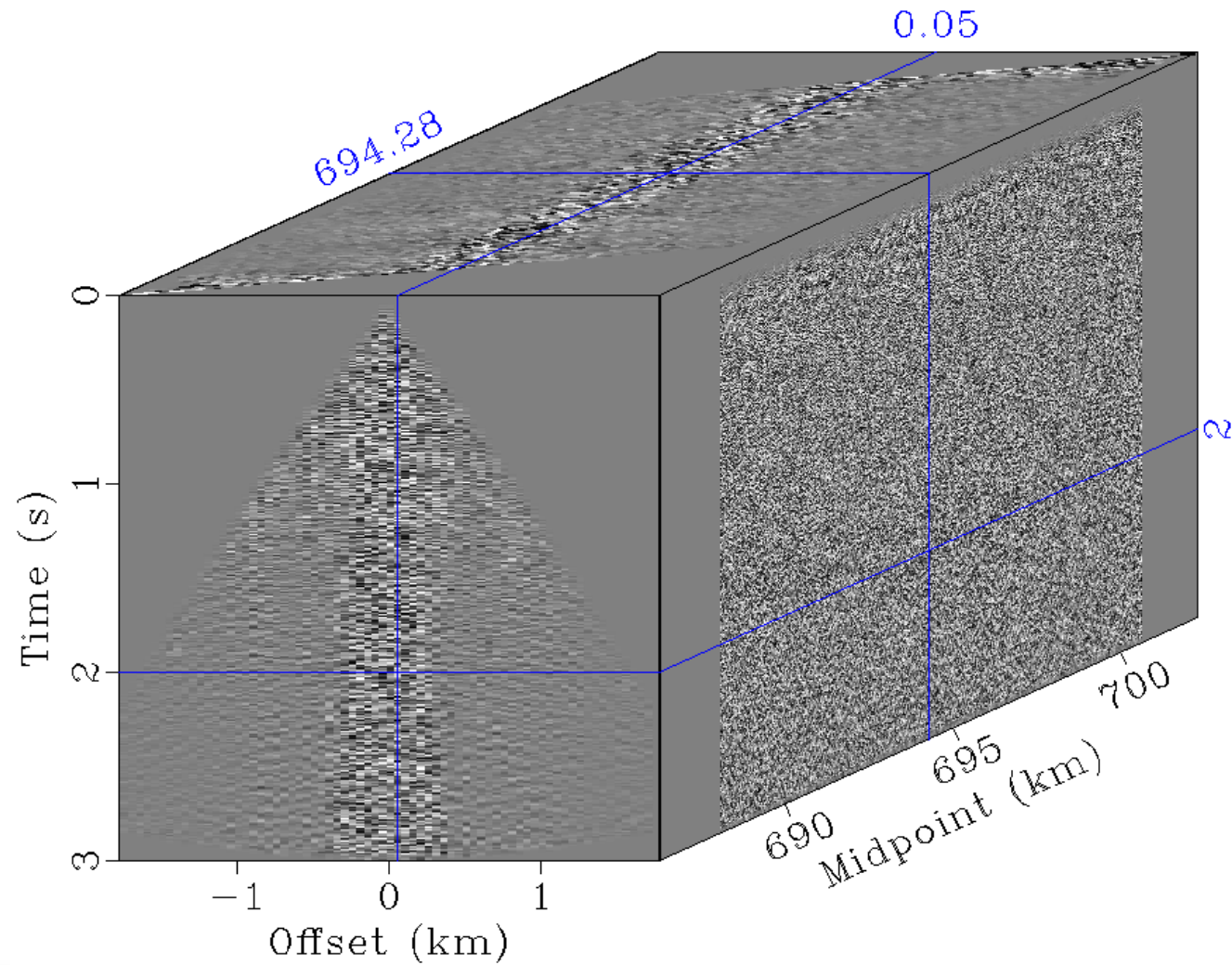
title="NMOed Data" point1=0.7

label2=Offset label3=Midpoint

''')

Display NMOed CMPs

NMOed Data



Brute stacking

Flow('bstack', 'nmo', 'stack')

Result('bstack',

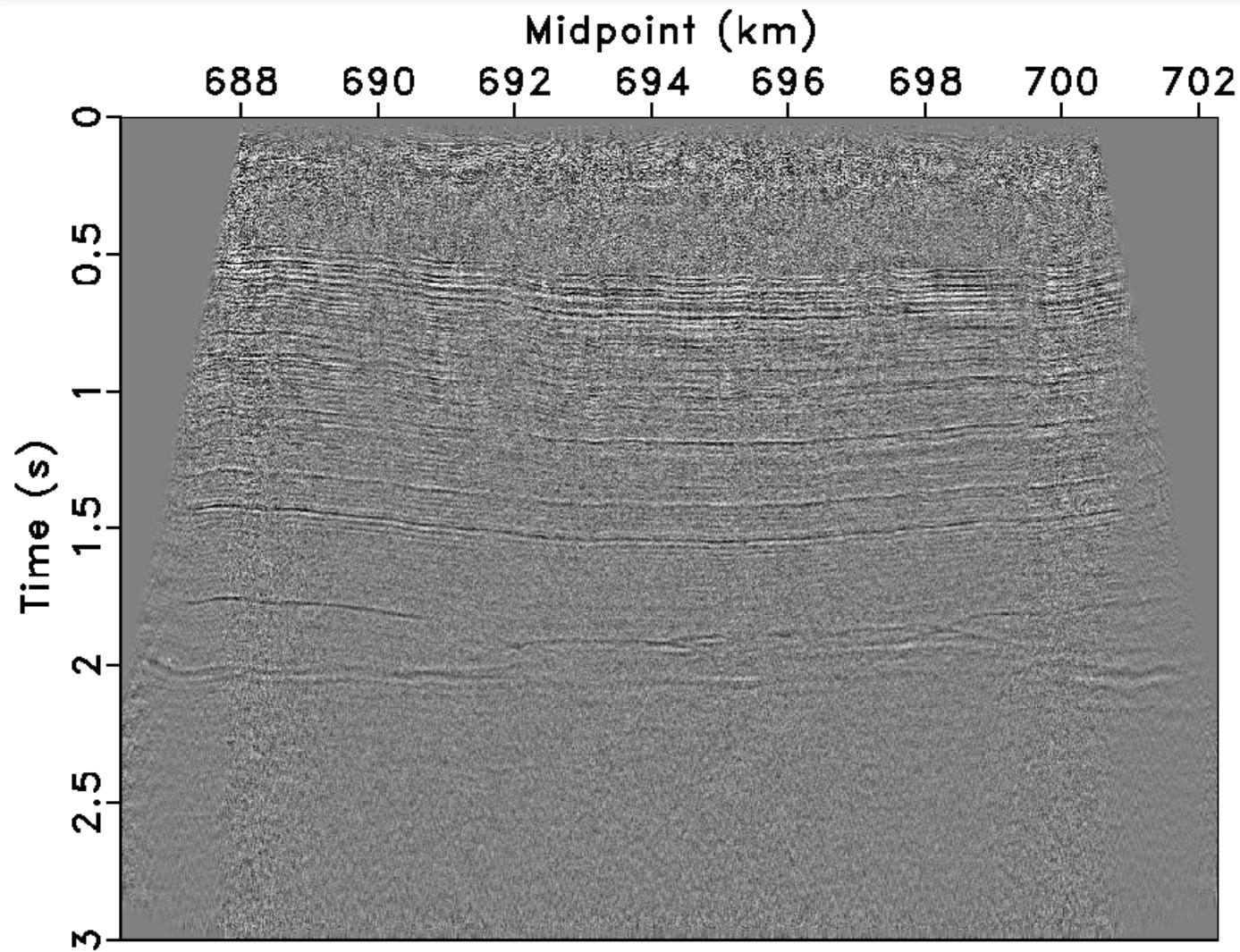
""

agc rect1=50 |

grey title="Brute stacking" labelfat=4 font=2 titlefat=4

""
)

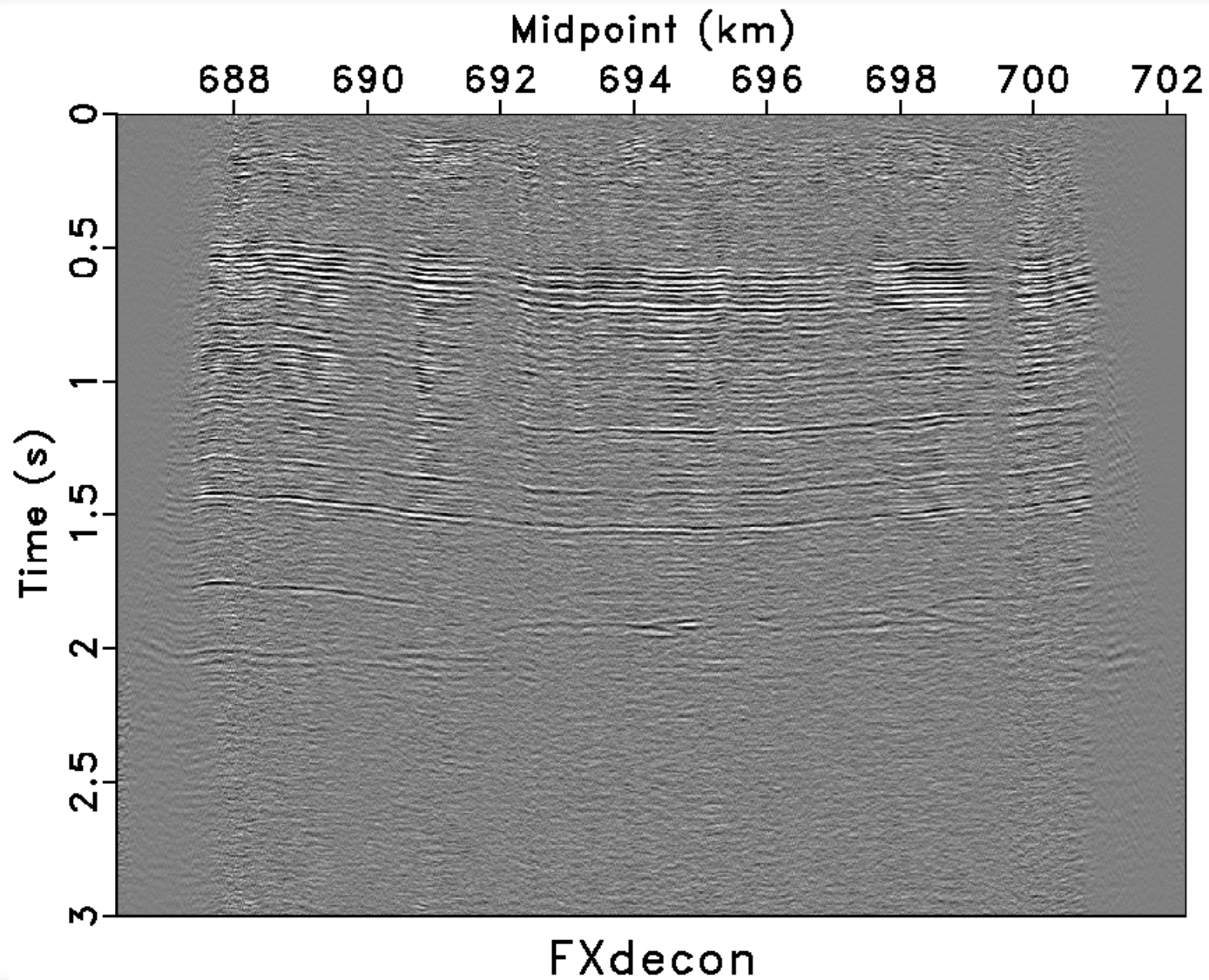
Display brute stacking result



Brute stacking

```
# Poststack denoising
Flow('fxdecon','bstack',
    ""
    fxdecon verb=y |
    bandpass flo=12 |
    fxdecon verb=y |
    bandpass flo=12 |
    fxdecon verb=y |
    bandpass flo=12
    "")
Result('fxdecon',
    ""
    grey title="FXdecon" labelfat=4 font=2 titlefat=4
    "")
```


Display denoising result



Toy prestack Kirchhoff time migration

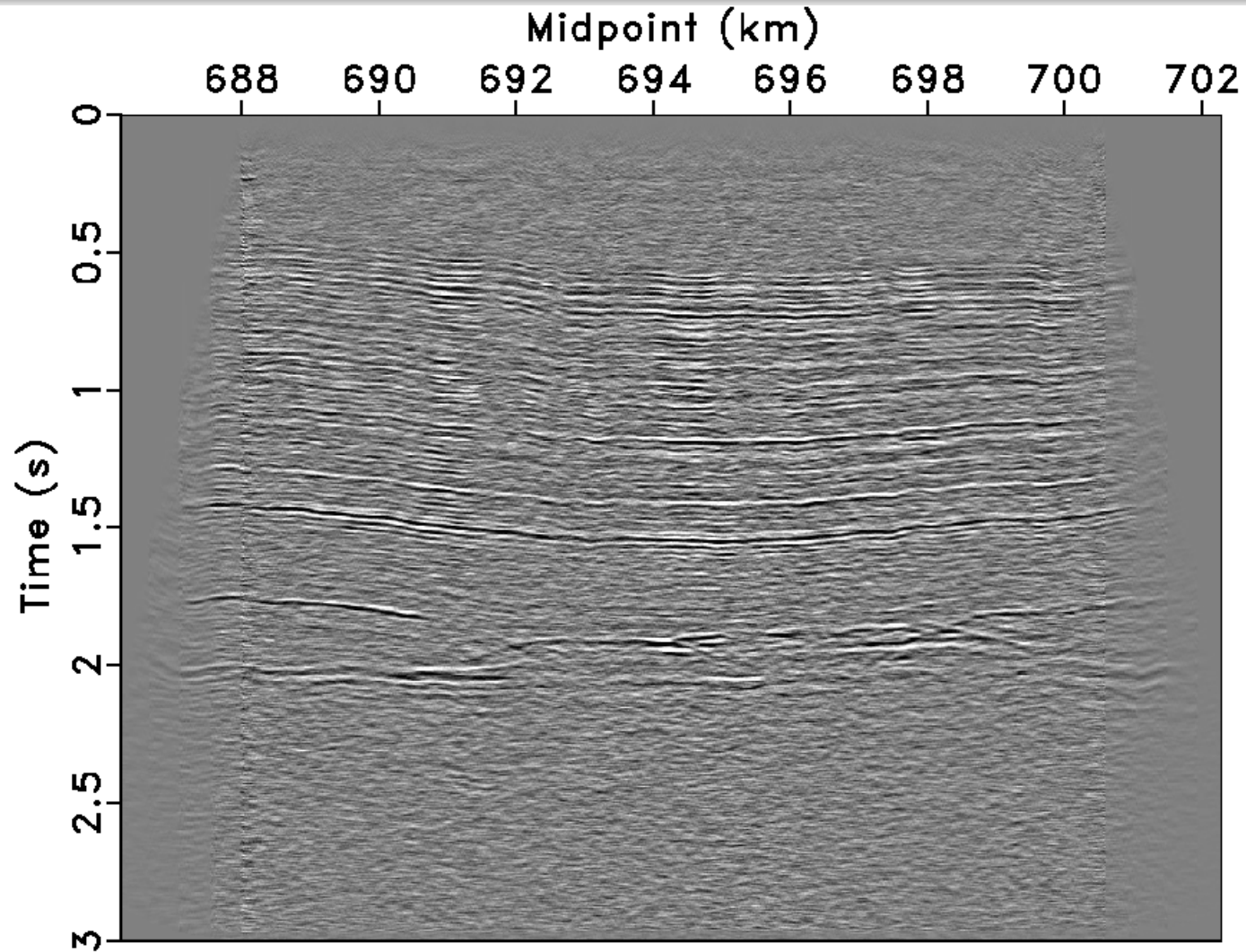


```
# Prestack Kirchhoff time migration
Flow('tcmps', 'cmps', 'transp memsize=1000 plane=23')
Flow('pstm', 'tcmps vel',
    ""
    mig2 vel=${SOURCES[1]} apt=5 antialias=1
    "",split=[3,71,[0]],reduce='add')
Result('pstm',
    ""
    grey title="Prestack kirchhoff time migration"
    labelfat=4 font=2 titlefat=4
    "")

End()
```

Always stop here!

Display migration result



Prestack kirchhoff time migration

- Geophysical data processing

- Assignment

- **Examples**

 - Seismic data

 - **Gravity data**

 - Controlled-source Electromagnetic data

Processing framework for a field gravity data



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

```
# Data convert
```

```
Flow('gravity.asc','gravity.dat',
```

```
    ""
```

```
    awk -F ',' 'BEGIN{count=0}
```

```
    {count=count+1;print count, " ",$1, " ", $2, " ", $3 }'
```

```
    ""
```

```
Flow('gravity','gravity.asc',
```

```
    ""
```

```
    echo in=$SOURCE data_format=ascii_float n1=4 n2=46472 |
```

```
    dd form=native |
```

```
    put d1=1 o1=0 d2=1 o2=0
```

```
    "",stdin=0)
```

Processing framework for a field gravity data

```
# Receiver Coordinates
```

```
Flow('x','gravity','window n1=1 f1=1')
```

```
Flow('y','gravity','window n1=1 f1=2')
```

```
Flow('location','x y','cmplx $SOURCES')
```

```
Result('location',
```

```
""
```

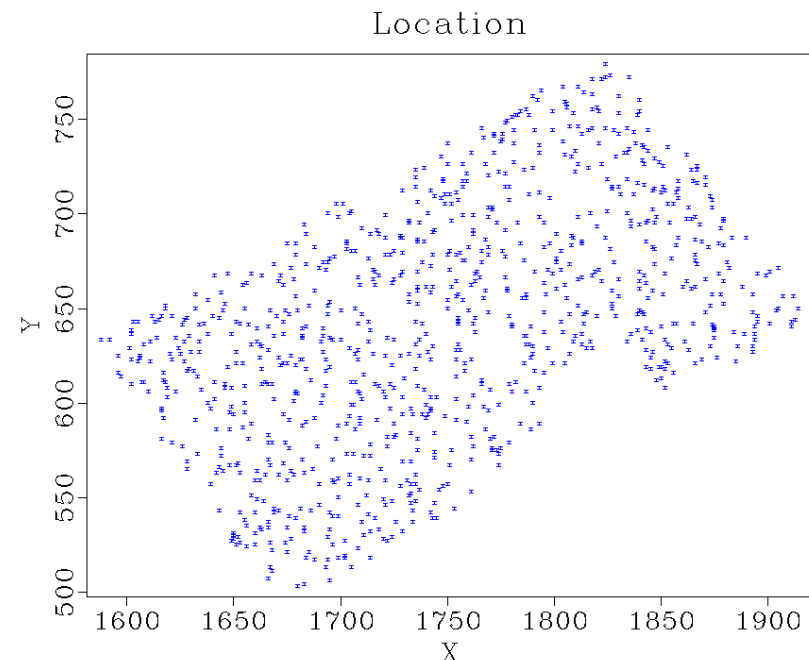
```
    window j1=50 |
```

```
    graph symbol=x title="Location"
```

```
    transp=n label1=X label2=Y
```

```
    "")
```

```
Flow('off','x y','cat ${SOURCES[1]} axis=2 | transp plane=12')
```



Processing framework for a field gravity data

```
Flow('g','gravity','window n1=1 f1=3')
```

```
# Binning to a regular grid
```

```
nx=340
```

```
x0=1585
```

```
dx=1
```

```
ny=287
```

```
y0=500
```

```
dy=1
```

```
Flow('bin fold','g off',
```

```
'''
```

```
bin head=${SOURCES[1]} fold=${TARGETS[1]} interp=2
```

```
xkey=0 ykey=1
```

```
nx=%d x0=%g dx=%g
```

```
ny=%d y0=%g dy=%g
```

```
''' % (nx,x0,dx,ny,y0,dy))
```

```
Result('fold',
```

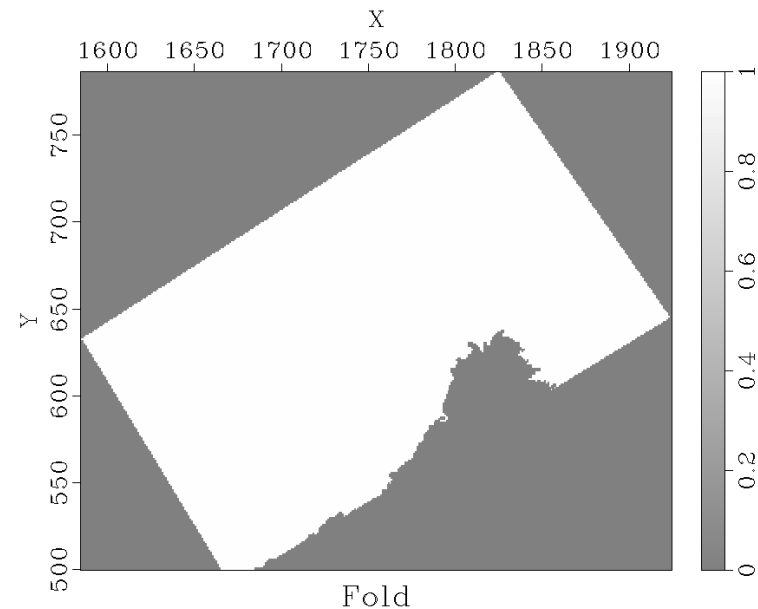
```
'''
```

```
reverse |
```

```
grey transp=n scalebar=y title=Fold
```

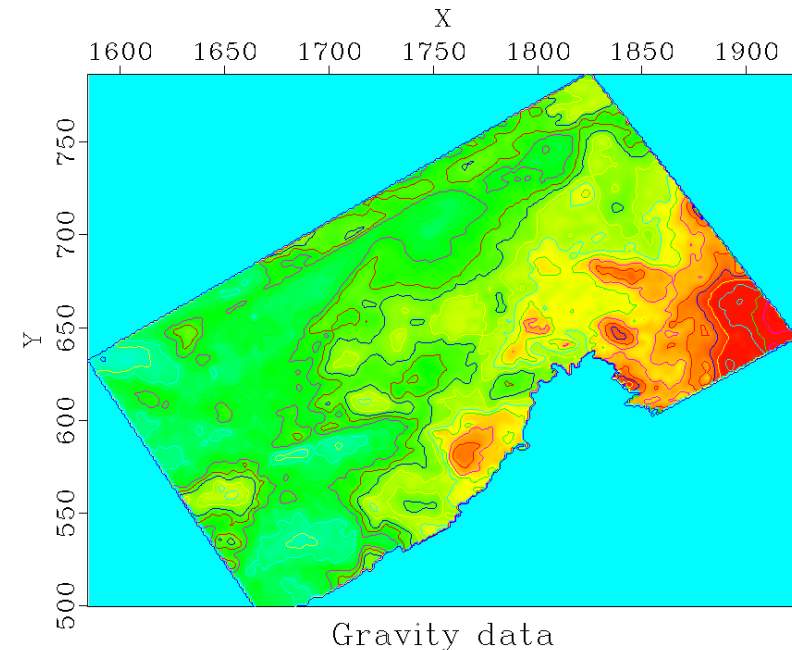
```
label1=X label2=Y
```

```
''')
```



Processing framework for a field gravity data

```
Plot('cbin','bin',  
    ""  
    reverse |  
    contour nc=20 c0=-95 dc=5  
    transp=n allpos=n wanttitle=n wantaxis=n  
    "")  
Plot('gbin','bin',  
    ""  
    reverse |  
    grey transp=n color=a title="Gravity data"  
    label1=X label2=Y  
    "")  
Result('bin','gbin cbin','Overlay')
```



Processing framework for a field gravity data

Nonlocal median filter

```
Flow('mtm','bin','mtm nfw1=10 nfw2=4 pclip=80')
```

```
Plot('cmtm','mtm',
```

```
'''
```

```
reverse |
```

```
contour nc=50 c0=-95 dc=5
```

```
transp=n allpos=n wanttitle=n wantaxis=n
```

```
''')
```

```
Plot('gmtm','mtm',
```

```
'''
```

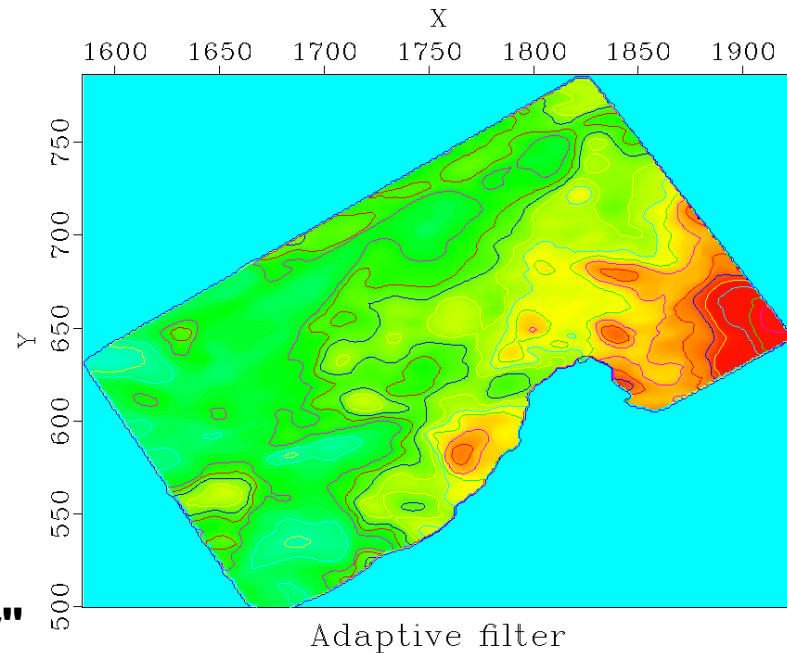
```
reverse |
```

```
grey transp=n color=a title="Adaptive filter"
```

```
label1=X label2=Y
```

```
''')
```

```
Result('mtm','gmtm cmtm','Overlay')
```



Processing framework for a field gravity data

x derivative using simple first-order derivative

Flow('sxder','bin','igrad')

Plot('csxder','sxder',

""

reverse |

contour nc=1 c0=-1 dc=1

transp=n allpos=n wanttitle=n wantaxis=n

""

Plot('gsxder','sxder',

""

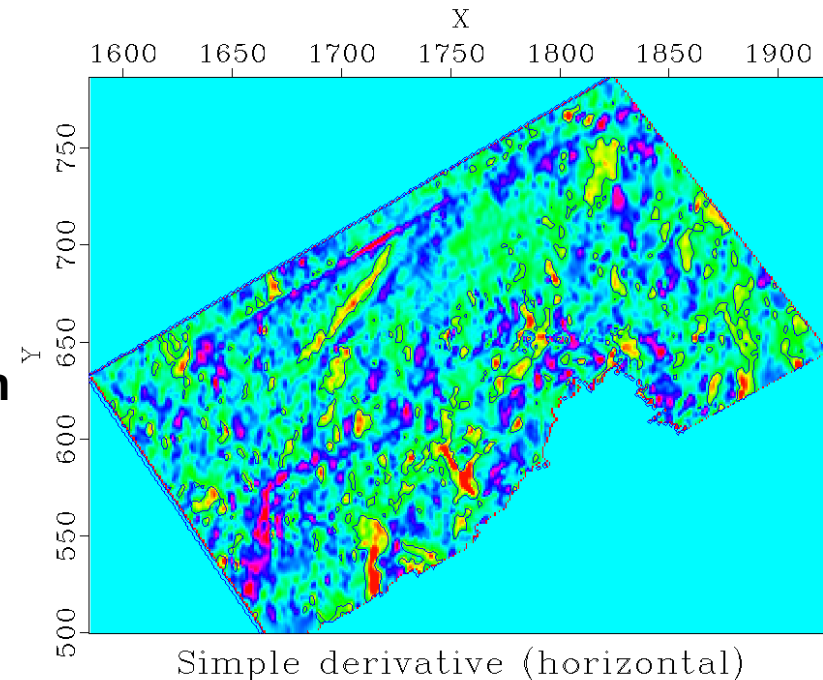
reverse |

grey transp=n title="Simple derivative (horizontal)"

color=a label1=X label2=Y

""

Result('sxder','gsxder csxder','Overlay')



Processing framework for a field gravity data

```
# y derivative using FIR Hilbert transform
```

```
Flow('hxder','bin','transp | deriv order=20 | transp')
```

```
Plot('chxder','hxder',
```

```
""
```

```
reverse |
```

```
contour nc=1 c0=-1 dc=1 plotcol=1
```

```
transp=n allpos=n wanttitle=n wantaxis=n
```

```
""
```

```
Plot('ghxder','hxder',
```

```
""
```

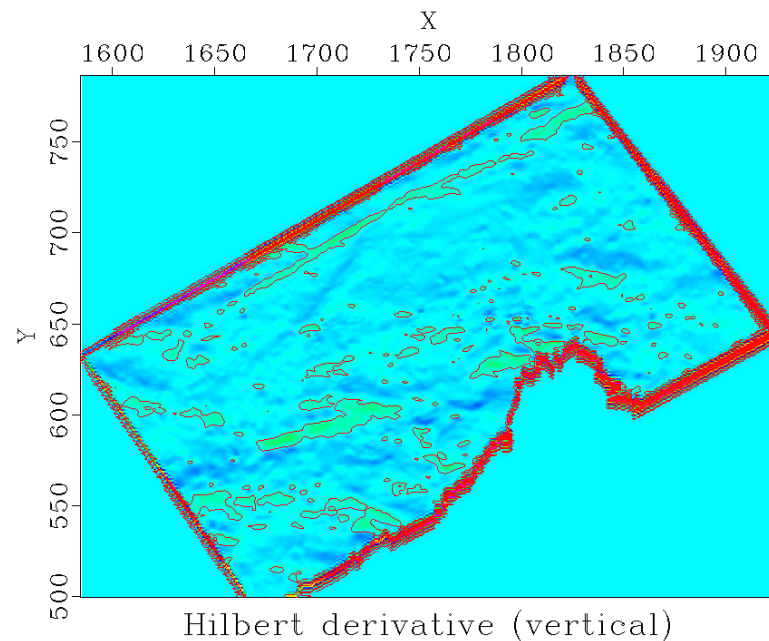
```
reverse |
```

```
grey transp=n title="Hilbert derivative (vertical)"
```

```
color=a label1=X label2=Y
```

```
""
```

```
Result('hxder','ghxder chxder','Overlay')
```



Processing framework for a field gravity data

```
# Continuation
```

```
Flow('cont','bin',
```

```
""
```

```
    gravcon z=-0.01 iter=n verb=y
```

```
    ""
```

```
Result('cont',
```

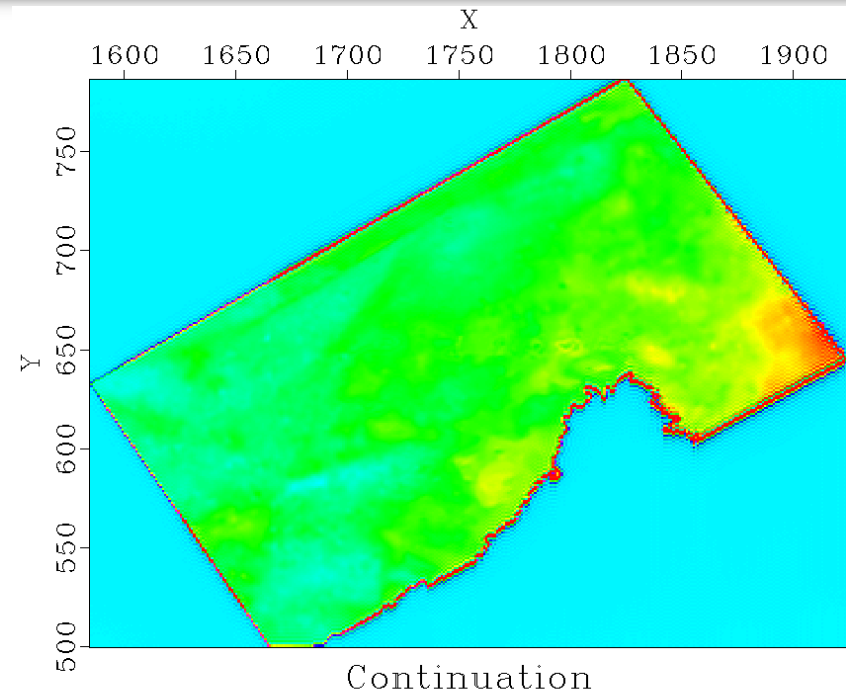
```
    ""
```

```
    reverse |
```

```
    grey transp=n color=a title="Continuation"
```

```
    scalebar=y label1=X label2=Y
```

```
    ""
```



- Geophysical data processing

- Assignment

- **Examples**

 - Seismic data

 - Gravity data

 - **Controlled-source Electromagnetic data**

Toy preprocessing workflow for CSEM data



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

```
Flow('Ts','Ts.H','dd form=native')  
Result('Ts','graph title= label1=Time unit1=s label2=Amplitude')  
Flow('spectra','Ts','fft1 | math output="abs(input*(2/45001))" | real')  
Result('spectra','graph title= min1=0 max1=1.5')
```

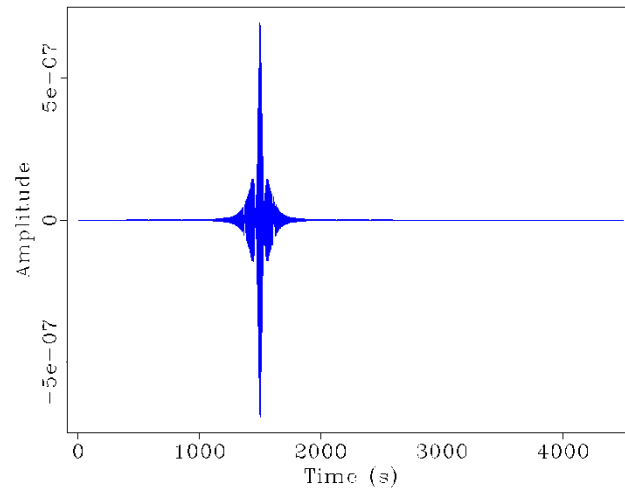
```
# random white noise #
```

```
Flow('tsa','Ts','noise type=y range=2e-10 seed=1000')  
Result('tsa','graph title= ')
```

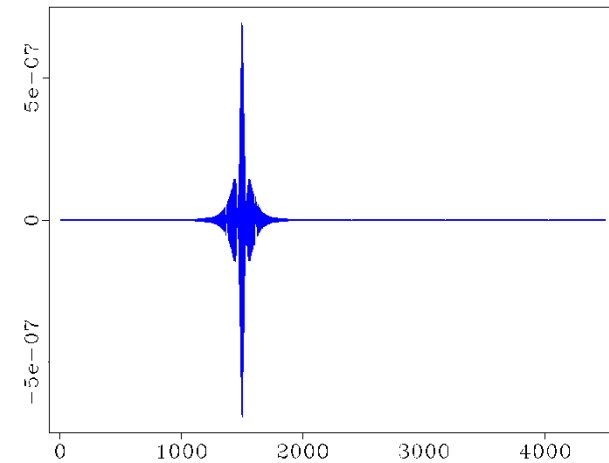
```
# Calculate MVO curve #
```

```
Flow('Exmvotsa','tsa',  
      'mvo f=0.08 m=1 nnw=125 n=1 | window n1=360 | put d1=1')  
Result('Exmvotsa',' graph title= ')  
End()
```

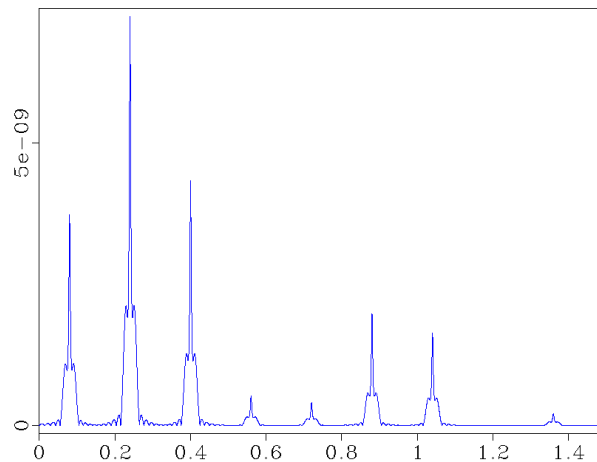
Toy preprocessing workflow for CSEM data



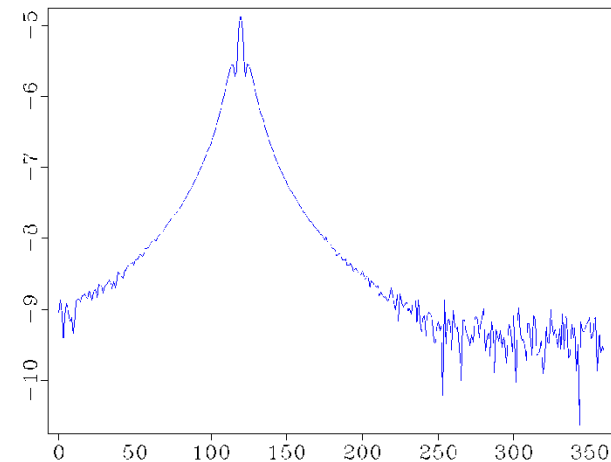
Add noise



Frequency Spectra



Time to Frequency conversion



To be continued ...

Try “scons view ” to see all figures one by one.

**Modify/add your own modules,
improve the final results ...**



<http://www.ahay.org>

THANKS FOR ATTENTION AND HAVE A FUN!