# Homework 5

*Thomas Bayes*


## ABSTRACT

This homework has three parts.

1. A theoretical question related to covariance estimation.
2. Stochastic simulation of natural patterns.
3. Missing data interpolation for ocean floor topography.


## PREREQUISITES

The homework code is available from the `Madagascar` repository by running

```
svn co https://github.com/ahay/src/trunk/book/geo384h/hw5
```

## THEORY

Suppose that we use the gradient operator for data interpolation:

$$\min |\nabla \mathbf{m}|^2 \ . \tag{1}$$

This approach roughly corresponds to minimizing the surface area and represents the behavior of a soap film or a thin rubber sheet.

The corresponding inverse model covariance operator is the negative Laplacian $\mathbf{C}_m^{-1} = \nabla^T \nabla = -\nabla^2$. The corresponding covariance operator corresponds to the Green's function $G(\mathbf{x})$ that solves

$$-\nabla^2 G = \delta(\mathbf{x} - \mathbf{x}_0) \ . \tag{2}$$

In 2-D, the Green's function has the form

$$G(\mathbf{x}) = A - \frac{\ln |\mathbf{x} - \mathbf{x}_0|}{2\pi} \tag{3}$$

with some constant $A$.

To derive equation (3), we can introduce polar coordinates around $\mathbf{x}_0$ with the radius $r = |\mathbf{x} - \mathbf{x}_0|$ and note that the Laplacian operator for a radially-symmetric function $\phi(r)$ in polar coordinates takes the form

$$\nabla^2 \phi = \frac{1}{r} \frac{d}{dr} \left( r \frac{d\phi}{dr} \right) \tag{4}$$

Away from the point $\mathbf{x}_0$, solving

$$\frac{1}{r} \frac{d}{dr} \left( r \frac{dG}{dr} \right) = 0 \tag{5}$$

leads to $G(r) = A + B \ln r$. To find the constant $B$, we can integrate $\nabla^2 G$ over a circle with some small radius $\epsilon$ around the origin and apply the Green's theorem

$$-1 = \iint \nabla^2 G dx\, dy = \oint \nabla G \cdot \vec{ds} = \int_0^{2\pi} \left. \frac{\partial G}{\partial r} \right|_{r=\epsilon} \epsilon\, d\theta = 2\pi\, B\ . \tag{6}$$

Derive the model covariance function $G(\mathbf{x})$ which corresponds to replacing equation (1) with equation

$$\min \left| \nabla^2 \mathbf{m} \right|^2 \tag{7}$$

and approximates the behavior of a thin elastic plate.

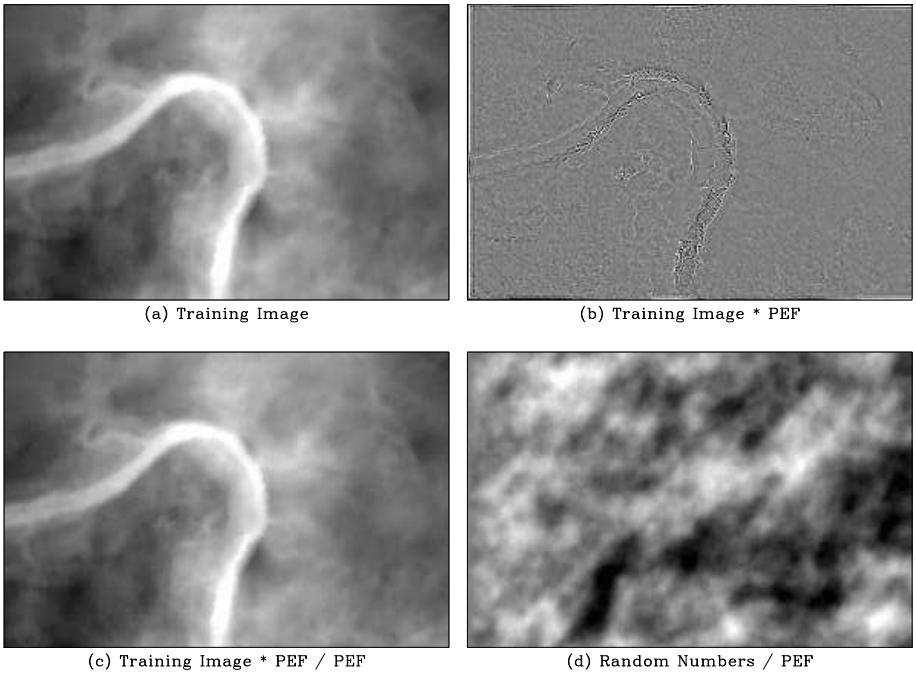
## NATURAL PATTERNS

In this section we will extract multidimensional spatial patterns from natural images using the method of Claerbout and Brown (1999). Four examples, shown in Figures 1-4, contain:

1. Seismic horizon slice.
2. A slice from a CT-scan of a rock sample.
3. A remote-sensing satellite image.
4. Your own data (to be replaced by you).

In each of the cases, we follow the same workflow:

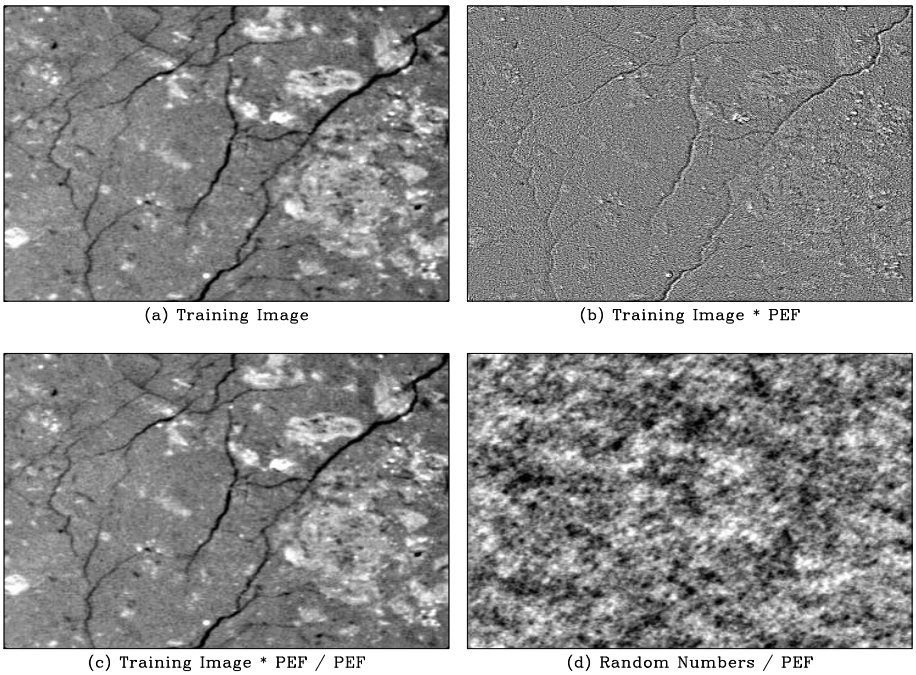
1. Remove a linear trend from the data.
2. Estimate a multi-dimensional prediction-error filter (PEF) on a helix.
3. Apply the inverse of the estimated PEF to random normally-distributed numbers to create a random spatial texture, which shares the covariance with the input.



(a) Training Image (b) Training Image * PEF

(c) Training Image * PEF / PEF (d) Random Numbers / PEF

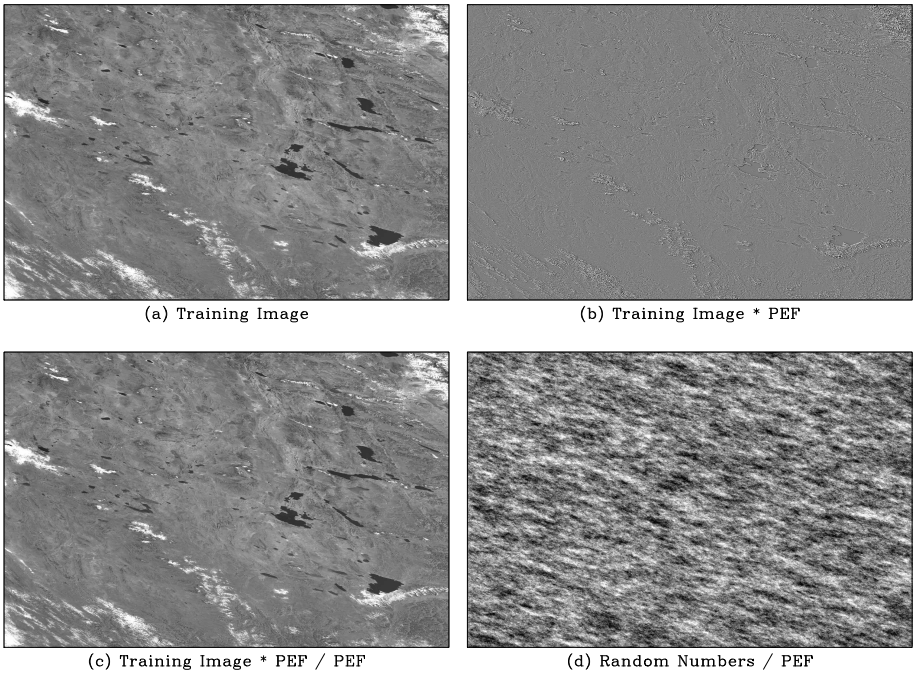
Figure 1: Pattern extraction from a seismic time horizon.



(a) Training Image (b) Training Image * PEF

(c) Training Image * PEF / PEF (d) Random Numbers / PEF

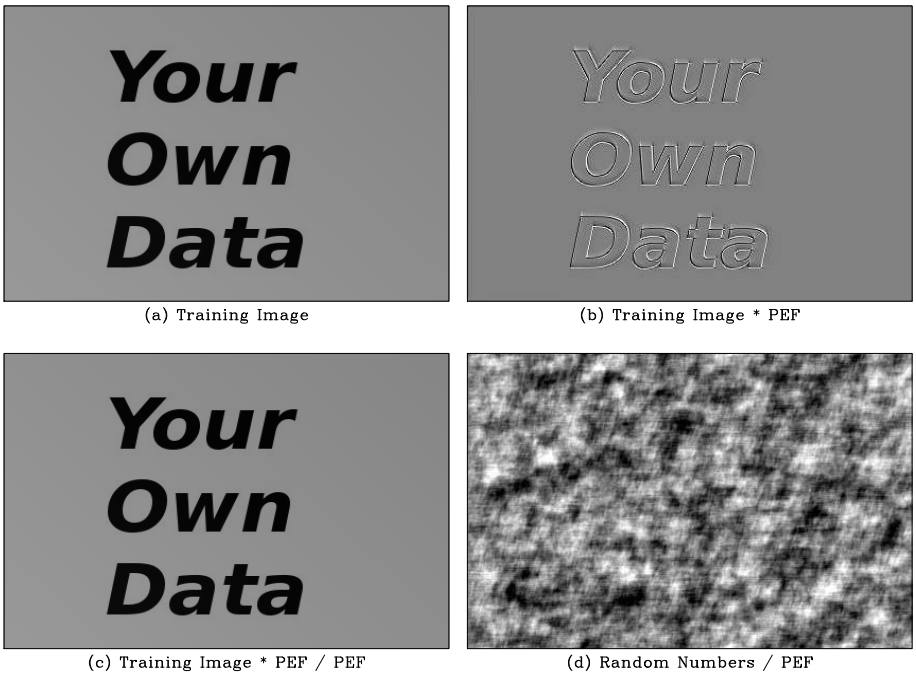
Figure 2: Pattern extraction from a CT-scan of a rock sample.



(a) Training Image (b) Training Image * PEF

(c) Training Image * PEF / PEF (d) Random Numbers / PEF

Figure 3: Pattern extraction from a satellite image.




(a) Training Image (b) Training Image * PEF

(c) Training Image * PEF / PEF (d) Random Numbers / PEF

Figure 4: Pattern extraction from your own data.

```
from rsf.proj import *

g = 'grey crowd1=0.96 crowd2=0.85 wantaxis=n title="%s" '

# Seismic horizon
Fetch('horizon.asc','hall')
Flow('horizon','horizon.asc',
     '''
     echo in=$SOURCE data_format=ascii_float n1=3 n2=57036 |
     dd form=native | window n1=1 f1=-1 |
     put
     n2=291 o2=35.031 d2=0.01 label2=y unit2=km
     n1=196 o1=33.139 d1=0.01 label1=x unit1=km
     ''')

# CT-scan slice
Fetch('slice.rsf','ctscan')
Flow('circle','slice','dd type=float')
Flow('square','circle',
     'window window n1=366 n2=366 f1=73 f2=73')

# Satellite data
Fetch('mod.dat','ying')
Flow('sat','mod.dat',
     '''
     echo in=$SOURCE n1=1200 n2=1200 n3=7
     data_format=native_int |
     put d1=1 d2=1 o1=0 o2=0 d3=1 o3=0 |
     window n3=1 f3=3 | dd type=float |
     scale axis=2
     ''', stdin=0)

# Your own data
Fetch('your.dat','ying')
Flow('your','your.dat','dd type=float')

textures = ('horizon','square','sat','your')

for txt in textures:
    # Remove trend
    #--------------
    m = txt+'-one'
    x = txt+'-x'
    y = txt+'-y'
```

```
45      Flow(m,txt,'math output=1')
46      Flow(x,txt,'math output=x1 | scale axis=2')
47      Flow(y,txt,'math output=x2 | scale axis=2')
48      flt = txt+'-flt'
49      trd = txt+'-trd'
50      Flow([flt,trd],[m,x,y,txt],
51            '''
52            cat ${SOURCES[1:3]} |
53            lpf match=${SOURCES[3]} pred=${TARGETS[1]}
54            rect1=1000 rect2=1000
55            ''')
56      t = txt+'-dtrd'
57      Flow(t,[txt,trd],'add scale=1,-1 ${SOURCES[1]}')
58
59      # Original
60      #----------
61      Plot(t,g % '(a) Training Image')
62
63      # Estimate PEF
64      #--------------
65      pef = txt+'-pef'
66      lag = txt+'-lag'
67      Flow([pef,lag],t,
68            'hpef niter=50 a=10,10 lag=${TARGETS[1]}')
69
70      # PEF residual
71      #--------------
72      wht = txt+'-wht'
73      Flow(wht,[t,pef],'helicon filt=${SOURCES[1]}')
74      Plot(wht,g % '(b) Training Image * PEF')
75
76      # Reconstruct original
77      #----------------------
78      rec = txt+'-rec'
79      Flow(rec,[wht,pef],'helicon filt=${SOURCES[1]} div=y')
80      Plot(rec,g % '(c) Training Image * PEF / PEF')
81
82      # Synthesized image
83      #-------------------
84      syn = txt+'-syn'
85      Flow(syn,[t,pef],
86                '''
87                noise rep=y seed=2014 |
88                helicon filt=${SOURCES[1]} div=y
89                ''')
```

```
90      Plot(syn,g % '(d) Random Numbers / PEF')
91
92      Result(txt,[t,wht,rec,syn],'TwoRows')
93
94  End()
```

Your task:

1. Change directory to `hw5/pattern`
2. Run

   ```
   scons view
   ```

   to reproduce the figures on your screen.
3. Modify the `SConstruct` file to replace Figure 4 with the figure containing your own data.
4. Why does the method fail in extracting some of the patterns? Did it succeed in extracting patterns from your data?

## MISSING OCEAN-FLOOR DATA INTERPOLATION

SeaBeam is an apparatus for measuring water depth both directly under a boat and somewhat off to the sides of the boat's track. In this part of the assignment, we will use a benchmark dataset from Claerbout (2014): SeaBeam data from a single day of acquisition. The original data are shown in Figure 5a.

Program `interpolate.c` implements two alternative methods: regularized inversion using convolution with a multi-dimensional filter and preconditioning, which uses the inverse operation (recursive deconvolution or polynomial division on a helix) for model reparameterization.

Our first attempt is to use the Laplacian-factor filter from the previous homework. The results from the two methods are shown in Figure 6. They are not very successful in hiding the "acquisition footprint".

Next, we will try to estimate the model covariance from the available data. The covariance can be estimated using the prediction-error filter (PEF) method (Claerbout and Brown, 1999). A random realizations of the model pattern using this methods is shown in Figure 7.

Our new attempt to interpolate missing data using the helical prediction-error filter is shown in Figure 8.

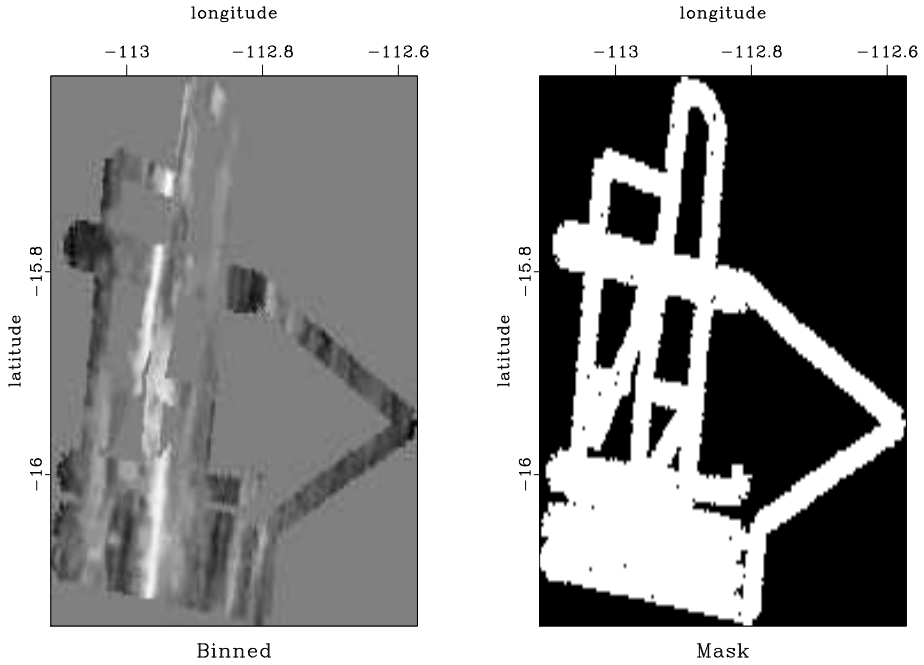



Figure 5: (a) Water depth measurements from one day of SeaBeam acquisition. (b) Mask for locations of known data.

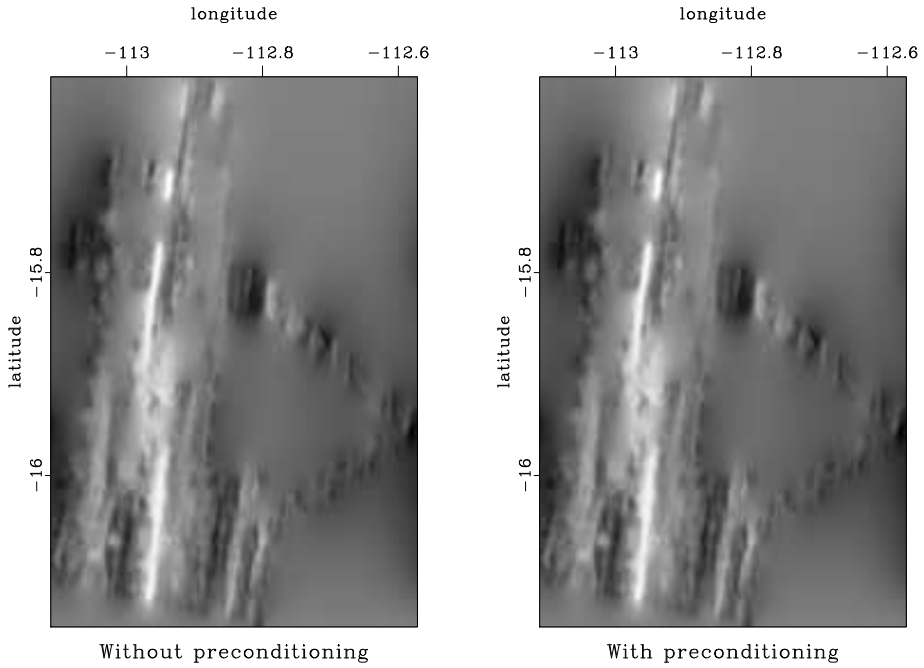



Figure 6: Left: missing data interpolation using regularization by convolution with a Laplacian factor. Right: missing data interpolation using model reparameterization by deconvolution (polynomial division) with a Laplacian factor.

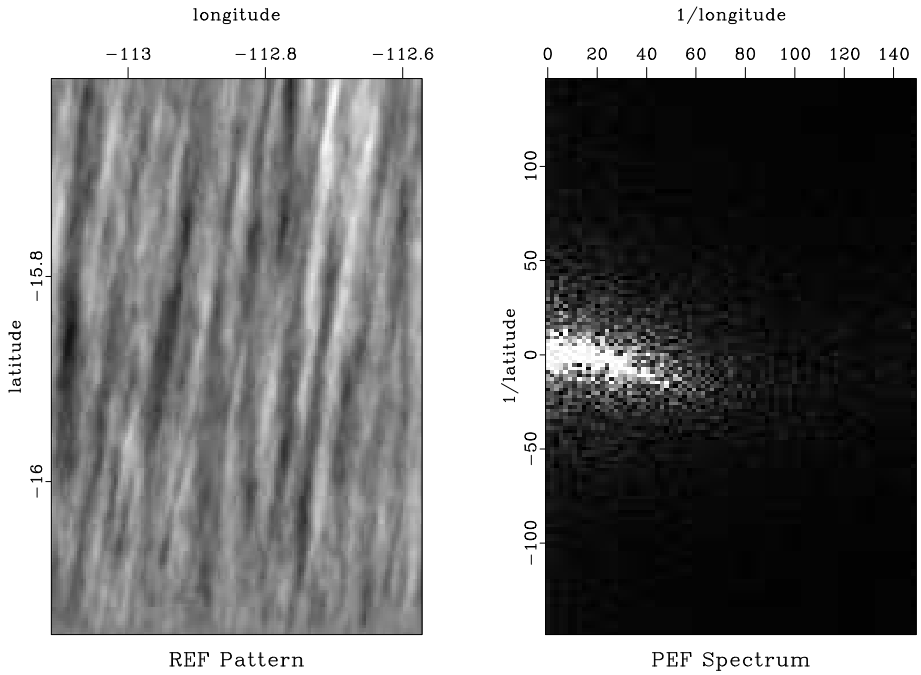



Figure 7: Left: data pattern generated using PEF method for inverse covariance estimation. Right: its Fourier spectrum.

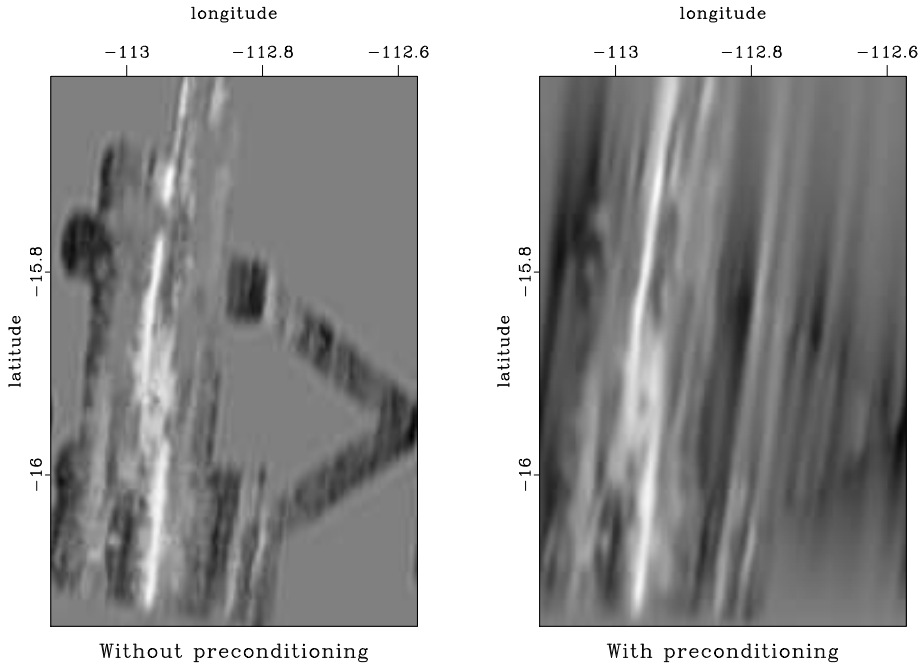



Figure 8: Left: missing data interpolation using regularization by convolution with a prediction-error filter. Right: missing data interpolation using model reparameterization by deconvolution (polynomial division) with a prediction-error filter.

seabeam/interpolate.c

```
/* Multi-dimensional missing data interpolation. */

#include <rsf.h>

int main(int argc, char* argv[])
{
    int na,ia, niter, n,i;
    float a0, *mm, *zero;
    bool prec, *known;
    char *lagfile;
    sf_filter aa;
    sf_file in, out, filt, lag, mask;

    sf_init (argc,argv);
    in = sf_input("in");
    filt = sf_input("filt");
    /* filter for inverse model covariance */
    out = sf_output("out");

    n = sf_filesize(in);

    if (!sf_getbool("prec",&prec)) prec=true;
    /* If y, use preconditioning */
    if (!sf_getint("niter",&niter)) niter=100;
    /* Number of iterations */

    if (!sf_histint(filt,"n1",&na)) sf_error("No n1=");
    aa = sf_allocatehelix (na);

    if (!sf_histfloat(filt,"a0",&a0)) a0=1.;

    /* Get filter lags */
    if (NULL == (lagfile = sf_histstring(filt,"lag"))) {
        for (ia=0; ia < na; ia++) {
            aa->lag[ia]=ia+1;
        }
    } else {
        lag = sf_input(lagfile);
        sf_intread(aa->lag,na,lag);
        sf_fileclose(lag);
    }

    /* Get filter values */
```

```
    sf_floatread(aa->flt,na,filt);
    sf_fileclose(filt);

    /* Normalize */
    for (ia=0; ia < na; ia++) {
        aa->flt[ia] /= a0;
    }

    mm = sf_floatalloc(n);
    zero = sf_floatalloc(n);
    known = sf_boolalloc(n);

    /* specify known locations */
    mask = sf_input("mask");
    sf_floatread(mm,n,mask);
    sf_fileclose(mask);

    for (i=0; i < n; i++) {
        known[i] = (bool) (mm[i] != 0.0f);
        zero[i] = 0.0f;
    }

    /* Read input data */
    sf_floatread(mm,n,in);

    if (prec) {
        sf_mask_init(known);
        sf_polydiv_init(n, aa);
        sf_solver_prec(sf_mask_lop, sf_cgstep, sf_polydiv_lop,
                       n, n, n, mm, mm, niter, 0., "end");
    } else {
        sf_helicon_init(aa);
        sf_solver (sf_helicon_lop, sf_cgstep, n, n, mm, zero,
                   niter, "known", known, "x0", mm, "end");
    }

    sf_floatwrite(mm,n,out);

    exit (0);
}
```

seabeam/SConstruct

```
from rsf.proj import *

```

```
# Download data
Fetch('apr18.h','seab')
Flow('data','apr18.h','dd form=native')

def grey(title):
    return '''
    grey pclip=100 labelsz=10 titlesz=12
    transp=n yreverse=n
    label1=longitude label2=latitude title="%s"
    ''' % title

def sgrey(title):
    return '''
    grey labelsz=10 titlesz=12 allpos=y
    transp=n yreverse=n title="%s"
    label1=1/longitude label2=1/latitude
    ''' % title

# Bin to regular grid
Flow('bin','data',
     '''
     window n1=1 f1=2 | math output='(2978-input)/387' |
     bin head=$SOURCE niter=150 nx=160 ny=160 xkey=0 ykey=1
     ''')

# Mask for known data
Flow('msk','bin',
     '''
     math output="abs(input)" |
     mask min=0.001 | dd type=float
     ''')

Plot('bin',grey('Binned'))
Plot('msk',grey('Mask') + ' allpos=y')

Result('seabeam0','bin msk','SideBySideAniso')

# Laplacian factorization

Flow('lag0.asc',None,
     '''
     echo 1 160 n1=2 n=160,160
     data_format=ascii_int in=$TARGET
     ''')
Flow('lag0','lag0.asc','dd form=native')
```

```
48
49 Flow('flt0.asc','lag0',
50       '''
51       echo -1 -1 a0=2 n1=2 lag=$SOURCE
52       data_format=ascii_float in=$TARGET
53       ''',stdin=0)
54 Flow('flt0','flt0.asc','dd form=native')
55
56 Flow('lapflt laplag','flt0',
57       'wilson eps=1e-3 lagout=${TARGETS[1]}')
58
59 # Missing data interpolation with Laplacian
60
61 niter=100
62
63 program = Program('interpolate.c')
64
65 for prec in (0,1):
66     interp='linterp%d' % prec
67     Flow(interp,'bin msk lapflt %s' % program[0],
68          '''
69          ./${SOURCES[3]} prec=%d niter=%d
70          mask=${SOURCES[1]} filt=${SOURCES[2]}
71          ''' % (prec,niter))
72     Plot(interp,grey('%s preconditioning' %
73                      ('Without','With')[prec]))
74 Result('lseabeam','linterp0 linterp1','SideBySideAniso')
75
76 # Random realization of white noise
77 seed = 112018  # CHANGE ME!!!
78
79 Flow('white','bin',
80       '''
81       noise rep=y seed=%d var=0.02
82       ''' % seed)
83
84 # Estimate PEF
85 Flow('pef lag','bin msk',
86       '''
87       hpef maskin=${SOURCES[1]} lag=${TARGETS[1]}
88       a=5,3 niter=50
89       ''')
90
91 Flow('rand','white pef',
92       'helicon filt=${SOURCES[1]} div=y')
```

```
93  Plot('rand',grey('REF Pattern'))
94
95  Flow('frand','rand','spectra2')
96  Plot('frand',sgrey('PEF Spectrum'))
97
98  Result('rand','rand frand','SideBySideAniso')
99
100 # Missing data interpolation with PEF
101
102 niter=20 # CHANGE ME!!!
103
104 for prec in (0,1):
105     interp='interp%d' % prec
106     Flow(interp,'bin msk pef %s' % program[0],
107          '''
108          ./${SOURCES[3]} prec=%d niter=%d
109          mask=${SOURCES[1]} filt=${SOURCES[2]}
110          ''' % (prec,niter))
111     Plot(interp,grey('%s preconditioning' %
112                      ('Without','With')[prec]))
113 Result('seabeam','interp0 interp1','SideBySideAniso')
114
115 End()
```

Your task:

1. Change directory to `hw5/seabeam`
2. Run

   ```
   scons view
   ```

   to reproduce the figures on your screen.
3. Modify the `SConstruct` file to find the number of iterations required for both methods shown in Figure 8 to achieve similar results.
4. A method for generating multiple realizations of missing data interpolation is:
   (a) Start with a random realization $\mathbf{m}_0$ such as the one shown in Figure 7.
   (b) Instead of estimating $\mathbf{m}$ such that $\mathbf{K}\,\mathbf{m} = \mathbf{d}$, estimate $\mathbf{x}$ such that

   $$\mathbf{K}\,\mathbf{x} = \mathbf{d} - \mathbf{K}\,\mathbf{m_0}\ .$$

   (c) The estimate for $\mathbf{m}$ is then $\widehat{\mathbf{m}} = \widehat{\mathbf{x}} + \mathbf{m_0}$.

Implement several realizations of missing data interpolation using several realizations of $\mathbf{m}_0$. You can do it by modifying either `SConstruct` or `interpolate.c`.

5. Include your results in the paper.

## COMPLETING THE ASSIGNMENT

1. Change directory to `hw5`.
2. Edit the file `paper.tex` in your favorite editor and change the first line to have your name instead of Bayes's.
3. Run

   ```
   sftour scons lock
   ```

   to update all figures.
4. Run

   ```
   sftour scons -c
   ```

   to remove intermediate files.
5. Run

   ```
   scons pdf
   ```

   to create the final document.
6. Submit your result (file `paper.pdf`) on paper or by e-mail.

## REFERENCES

Claerbout, J., 2014, Geophysical image estimation by example: Lulu.

Claerbout, J., and M. Brown, 1999, Two-dimensional textures and prediction-error filters: 61st Mtg., Eur. Assn. Geosci. Eng., Session:1009.