Homework 5

Eliakim Hastings Moore

ABSTRACT

This homework has the following parts:

1. A theoretical question related to covariance estimation.
2. Missing data interpolation using compressive properties of the 2-D Fourier transform.
3. Compression of sand dune images.

PREREQUISITES

Completing the computational part of this homework assignment requires

- `Madagascar` software environment available from `http://www.ahay.org/`
- LaTeX environment with `SEG`TeX available from `http://www.ahay.org/wiki/SEGTeX`

To do the assignment on your personal computer, you need to install the required environments. Please ask for help if you don't know where to start.

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

```
svn co http://svn.code.sf.net/p/rsf/code/trunk/book/geo391/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 ϵ 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.

PROJECTION ONTO CONVEX SETS

The goal of the next exercise is to figure out if one can use compactness of the Fourier transform to reconstruct missing data. The missing parts are created artificially by cutting holes in the original data (Figure 1).

Figures 2a and 2b show the digital Fourier transform of the original data and the data with holes. We observe again that the support of the data in the Fourier domain is compact thanks to the data smoothness. Cutting holes in the physical domain creates discontinuities that make the Fourier response spread beyond the original

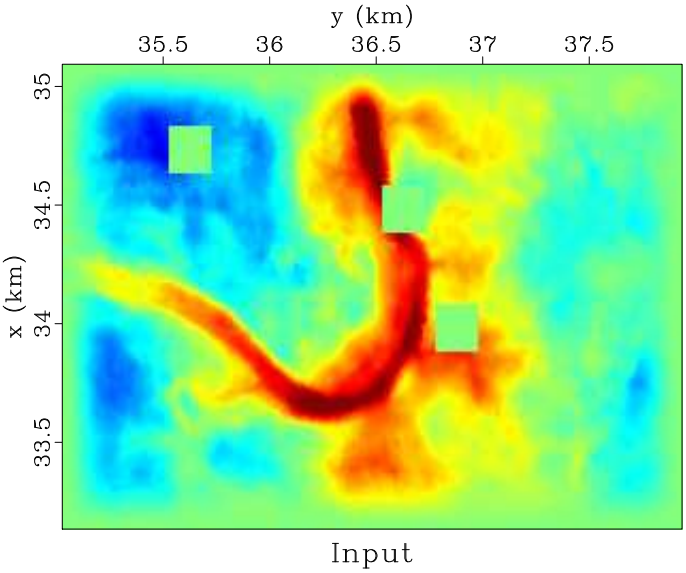
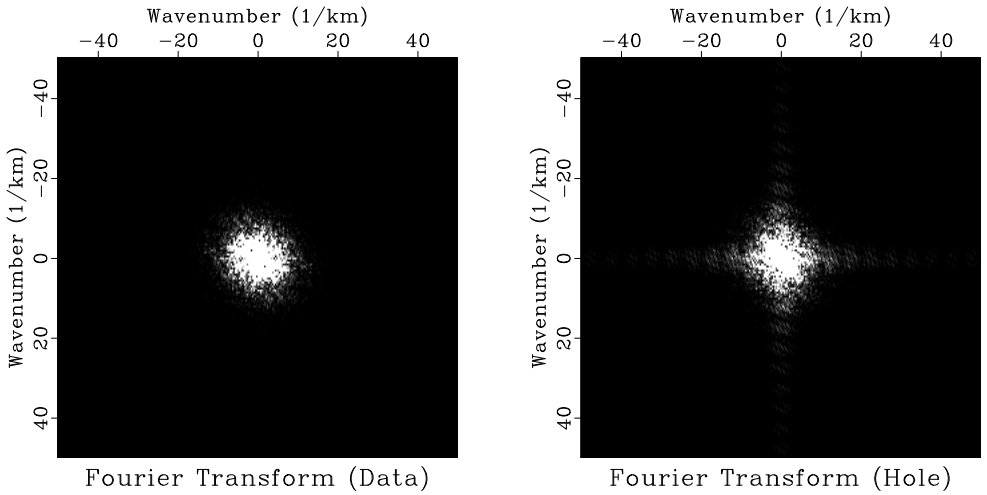


Figure 1: Seismic depth slice after removing selected parts of the data.



a b

Figure 2: Fourier transform of the original data (a) and data with holes with holes (b). The absolute value is displayed

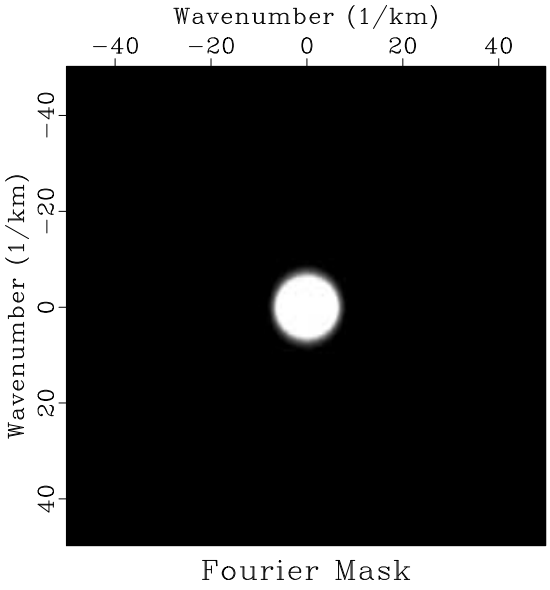


Figure 3: Fourier-domain mask for selecting a convex set.

support. Figure 3 shows a Fourier-domain mask designed to contain the support of the original data.

To accomplish the task of missing data interpolation, we will use an iterative method known as POCS (*projection onto convex sets*). By definition, a convex set $\mathcal{C}$ is a set of functions such that, for any $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ from the set, $g(\mathbf{x}) = \lambda\, f_1(\mathbf{x}) + (1-\lambda)\, f_2(\mathbf{x})$ (for $0 \le \lambda \le 1$) also belongs to the set. A projection onto a convex set means finding a function in the set that is of the shortest distance to the given function. The POCS theorem states that if one wants to find a function that belongs to the intersection of two convex sets C_1 and C_2, the task can be accomplished iteratively by alternating projections onto the two sets.

In our example, C_1 is the set of all functions that are equal to the known data outside of the holes. C_2 is the set of all functions that have a predifined compact support in the Fourier domain (and therefore are smooth in the physical domain). The algorithm consists of the following steps:

1. Apply 2-D Fourier transform.
2. Multiply by a Fourier-transform mask to enforce compact support.
3. Apply inverse 2-D Fourier transform.
4. Replace data outside of the holes with known data.
5. Repeat.

The output after 5 iterations is shown in Figure 4.

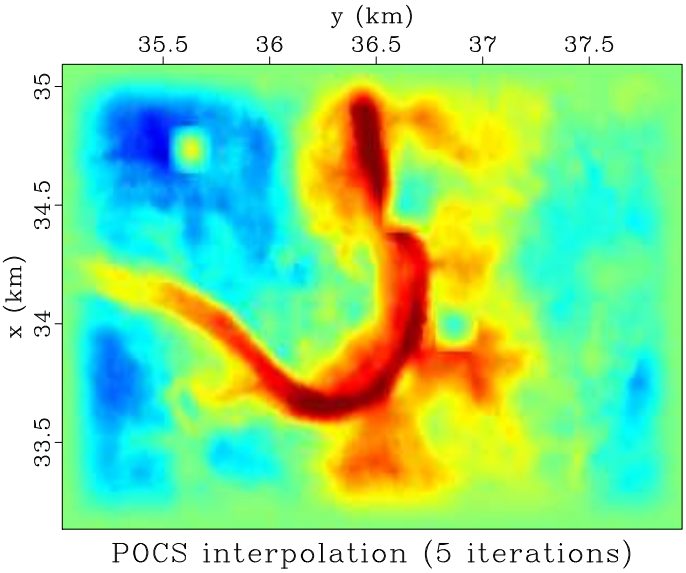


Figure 4: Missing data interpolated by iterative projection onto convex sets.

```
from rsf.proj import *

# Download data
Fetch('horizon.asc','hall')

# Convert format
Flow('data','horizon.asc',
     '''
     echo in=$SOURCE data_format=ascii_float n1=3 n2=57036 |
     dd form=native | window n1=1 f1=-1 | add add=-65 |
     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 |
     costaper nw1=25 nw2=25
     ''')

# Display
def plot(title):
    return '''
    grey color=j title="%s"
    transp=y yreverse=n clip=14
    ''' % title
Result('data',plot('Horizon'))

# Cut three square holes (!!! CHANGE ME !!!)
cut = '''
cut n1=20 n2=20 f1=125 f2=150 |
```

```
cut n1=20 n2=20 f1=150 f2=50  |
cut n1=20 n2=20 f1=75  f2=175
'''

Flow('hole','data',cut)
Flow('mask','data',
     'math output=1 | %s | math output=1-input' % cut)
Plot('hole',plot('Input'))
Result('hole','Overlay')

# Fourier transform
forw = 'rtoc | fft3 axis=1 pad=1 | fft3 axis=2 pad=1'
back = 'fft3 axis=2 inv=y | fft3 axis=1 inv=y | real'

for data in ('data','hole'):
    fft = 'fft-'+data
    Flow(fft,data,forw)
    Result(fft,
           '''
           math output="abs(input)" | real |
           grey allpos=y title="Fourier Transform (%s)"
           screenratio=1
           ''' % data.capitalize())

# Create Fourier mask
Flow('fft-mask','fft-hole',
     '''
     real | math output="x1*x1+x2*x2" | mask min=50 |
     dd type=float | math output=1-input |
     smooth rect1=5 rect2=5 repeat=3 | rtoc
     ''')
Result('fft-mask',
       '''
       real |
       grey allpos=y title="Fourier Mask" screenratio=1
       ''')

# POCS iterations
niter=5 # !!! CHANGE ME !!!

data = 'hole'
plots = ['hole']
for iter in range(niter):
    old = data
    data = 'data%d' % iter
```

```
    # 1. Forward FFT
    # 2. Multiply by Fourier mask
    # 3. Inverse FFT
    # 4. Multiply by space mask
    # 5. Add data outside of hole
    Flow(data,[old,'fft-mask','mask','hole'],
         '''
         %s | mul ${SOURCES[1]} |
         %s | mul ${SOURCES[2]} |
         add ${SOURCES[3]}
         ''' % (forw,back))
    Plot(data,plot('Iteration %d' % (iter+1)))
    plots.append(data)
# Put frames in a movie
Plot('pocs',plots,'Movie',view=1)

# Last frame
Result('pocs',data,
       plot('POCS interpolation (%d iterations)' % niter))

End()
```

Your task:

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

   ```
   scons view
   ```

   to reproduce the figures on your screen.
3. Additionally, you can run

   ```
   scons pocs.vpl
   ```

   to see a movie of different iterations.
4. By modifying appropriate parameters in the `SConstruct` file and repeating computations, find out
   (a) How many iterations are required for convergence?
   (b) How large can you make the holes and still be able to achieve a reasonably good reconstruction?
5. **EXTRA CREDIT** for finding a different convex set or a different iteration strategy for either faster or more accurate missing data reconstruction.

COMPRESSION OF SAND DUNE IMAGES

Figure 5: Image of sand dunes in a river.

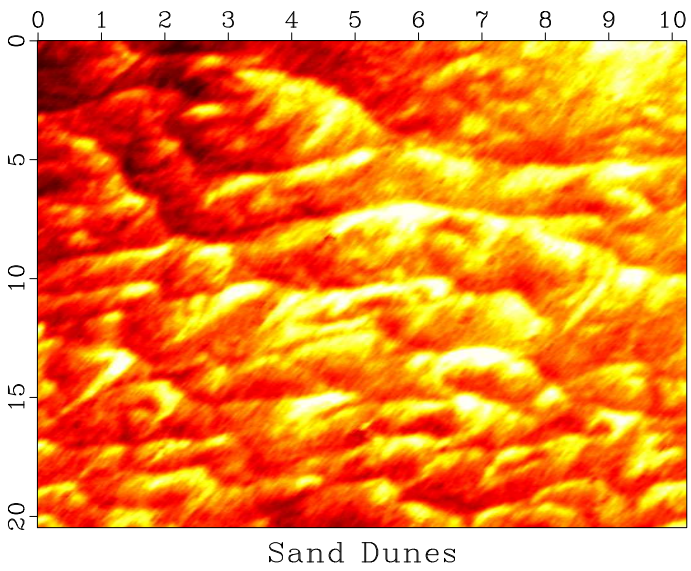
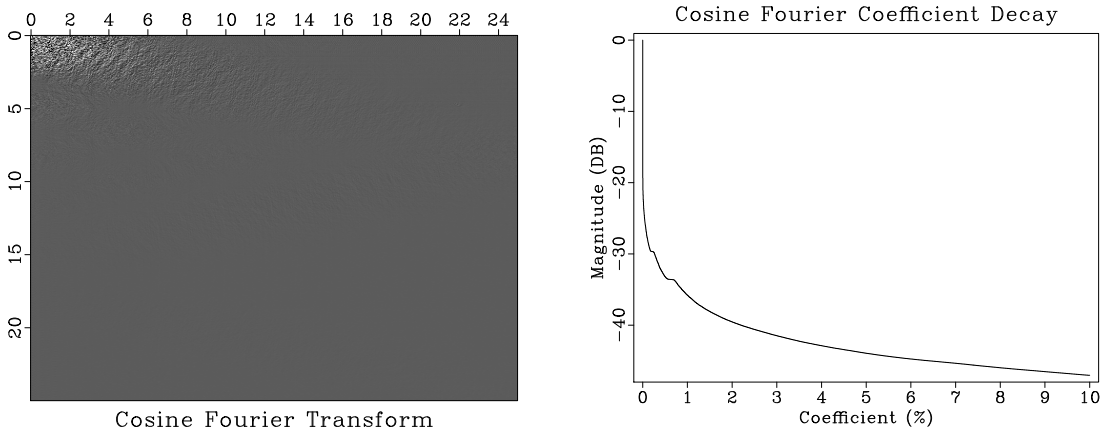


Figure 5 shows an image of sand dunes at the bottom of a river[1]. In this part of the assignment, you will try to compress the image by applying different transforms.



a b

Figure 6: (a) Sand dunes image in the cosine Fourier domain. (b) Fourier coefficients sorted by absolute value and displayed on the logarithmic (decibel) scale.

Figure 6a shows the image after applying the *cosine transform* (a version of the Fourier transform that keeps coefficients real). Notice both compactness and sparsity in the Fourier domain. To analyze the sparsity pattern, Figure 6b shows Fourier coefficients after sorting them by absolute value. The rate of coefficient decay is a measure of sparsity.

[1]courtesy of Ryan Ewing

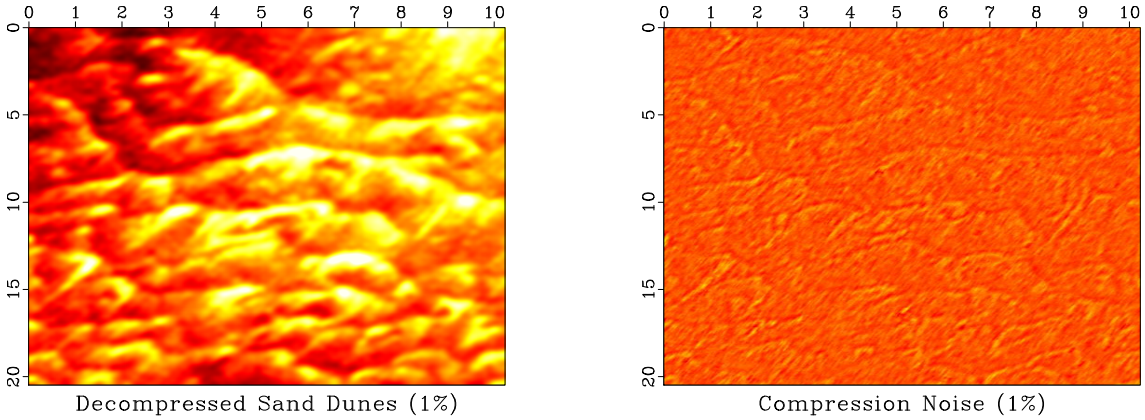


Figure 7: (a) Sand dunes image reconstructed after thresholding. (b) Compression noise.

Figure 7 shows the result of shrinkage (soft thresholding) of Fourier coefficients using 1% threshold and the difference between the reconstructed image and the true image.

Your task:

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

   ```
   scons view
   ```

   to reproduce the figures on your screen.
3. Modify the `SConstruct` file to adjust the threshold percentage to the level that makes noise negligible.
4. Modify the `SConstruct` file to use DWT (the *digital wavelet transform*) instead of the cosine transform. Compare the results. Which transform has more sparsity and provides better compression?
5. **EXTRA CREDIT** for finding and implementing a transform with an even better compression.

dunes/SConstruct

```
from rsf.proj import *

# Critical parameter
perc = 1 # percentage for thresholding

# Download data
Fetch('dunes3.HH','dunes')
Flow('dunes','dunes3.HH','dd form=native')

# Window size
n1=1024
n2=512

# Plotting macro
def plot(title):
    return '''
    grey color=H bias=-213 clip=150 title="%s"
    ''' % title

# Display data
Flow('dune','dunes','window n3=1 n1=%d n2=%d' % (n1,n2))
Result('dune',plot('Sand Dunes'))

# Transform dictionary
######################
transforms = {
    'cos': ('Cosine Fourier',
            'cosft sign1=1 sign2=1',
            'cosft sign1=-1 sign2=-1'),
    'dwt': ('Digital Wavelet',
            '''
            dwt type=b inv=y unit=y | transp |
            dwt type=b inv=y unit=y | transp
            ''',
            '''
            transp | dwt type=b inv=y unit=y adj=y |
            transp | dwt type=b inv=y unit=y adj=y
            ''')
    }

transform = transforms['cos']

# Apply forward transform
```

```
Flow('cos','dune',transform[1])
Result('cos',
       'grey title="%s Transform" ' % transform[0])

# Sort coefficients
Flow('sort','cos',
     '''
     put n1=%d n2=1 d1=%g label1=Coefficient unit1=%% |
     math output="abs(input)" | sort | scale axis=1 |
     math output="10*log(input)/log(10)"
     ''' % (n1*n2,100.0/(n1*n2-1)))
Result('sort',
       '''
       window max1=10 |
       graph title="%s Coefficient Decay"
       label2=Magnitude unit2=DB
       ''' % transform[0])

# Threshold and inverse transform
Flow('thr','cos','threshold pclip=%g' % perc)
Flow('inv','thr',transform[2])
Plot('inv',plot('Decompressed Sand Dunes (%g%%)' % perc))

# Noise = Data - Signal
Flow('diff','dune inv','add scale=1,-1 ${SOURCES[1]}')
Plot('diff',
     plot('Compression Noise (%g%%)' % perc) + ' bias=0')

Result('inv','inv diff','SideBySideIso')

End()
```

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 Moore'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.