

Homework 4

Magnus Hestenes

ABSTRACT

This homework has four parts.

1. Theoretical questions related to the conjugate-gradients algorithm.
2. Compression of sand dune images.
3. Attenuation of surface-wave noise in seismic data collected over sand dunes using match filtering.
4. Predicting data patterns using match filtering.

CONJUGATE GRADIENTS

The following algorithm finds a solution to the least-squares optimization problem $\min \|\mathbf{F} \mathbf{m} - \mathbf{d}\|^2$, where \mathbf{d} is data, \mathbf{m} is the model we want to estimate, and \mathbf{F} is a linear operator that connects them.

CONJUGATE GRADIENTS($\mathbf{F}, \mathbf{d}, N$)

```
1   $\mathbf{m} \leftarrow \mathbf{0}$ 
2   $\mathbf{r} \leftarrow -\mathbf{d}$ 
3  for  $n \leftarrow 1, 2, \dots, N$ 
4    do
5       $\mathbf{g}_m \leftarrow \mathbf{F}^T \mathbf{r}$ 
6       $\mathbf{g}_r \leftarrow \mathbf{F} \mathbf{g}_m$ 
7       $\rho \leftarrow \mathbf{g}_m^T \mathbf{g}_m$ 
8      if  $n = 1$ 
9        then  $\beta \leftarrow 0$ 
10       else  $\beta \leftarrow \rho / \hat{\rho}$ 
11        $\begin{bmatrix} \mathbf{s}_m \\ \mathbf{s}_r \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{g}_m \\ \mathbf{g}_r \end{bmatrix} + \beta \begin{bmatrix} \mathbf{s}_m \\ \mathbf{s}_r \end{bmatrix}$ 
12        $\alpha \leftarrow -\rho / (\mathbf{s}_r^T \mathbf{s}_r)$ 
13        $\begin{bmatrix} \mathbf{m} \\ \mathbf{r} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{m} \\ \mathbf{r} \end{bmatrix} + \alpha \begin{bmatrix} \mathbf{s}_m \\ \mathbf{s}_r \end{bmatrix}$ 
14        $\hat{\rho} \leftarrow \rho$ 
15  return  $\mathbf{m}$ 
```

1. How much storage does this algorithm require? If the model size is N_m and the data size is N_d , how much additional storage do we need to allocate?
2. In applications, it is often advantageous to apply *model reparameterization* or *preconditioning*. Suppose that, instead of solving for \mathbf{m} directly, you first solve for \mathbf{x} such that $\mathbf{m} = \mathbf{P} \mathbf{x}$. Show how to incorporate the linear preconditioning operator \mathbf{P} in the algorithm above.
3. Prove that the output of the algorithm after N iterations is

$$\mathbf{m}_N = \sum_{n=1}^N \frac{\mathbf{s}_n \mathbf{s}_n^T}{\mathbf{s}_n^T \mathbf{F}^T \mathbf{F} \mathbf{s}_n} \mathbf{F}^T \mathbf{d}, \quad (1)$$

where \mathbf{s}_n is the model step \mathbf{s}_m at n -th iteration.

COMPRESSION OF SAND DUNE IMAGES

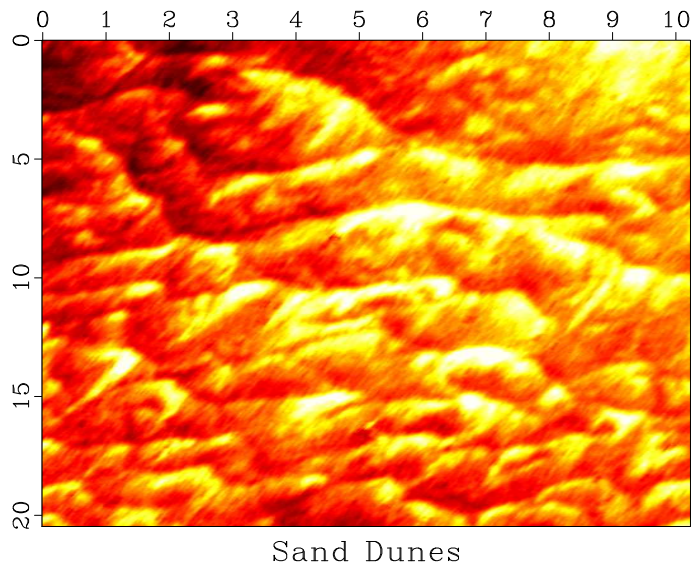


Figure 1: Image of sand dunes in a river.

Figure 1 shows an image of sand dunes at the bottom of a river¹. In this part of the assignment, you will try to compress the image by applying different transforms.

Figure 2(a) 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 2(b) shows Fourier coefficients after sorting them by absolute value. The rate of coefficient decay is a measure of sparsity.

Figure 3 shows the result of shrinkage (soft thresholding) of Fourier coefficients using 1% threshold and the difference between the reconstructed image and the true image.

¹courtesy of Ryan Ewing

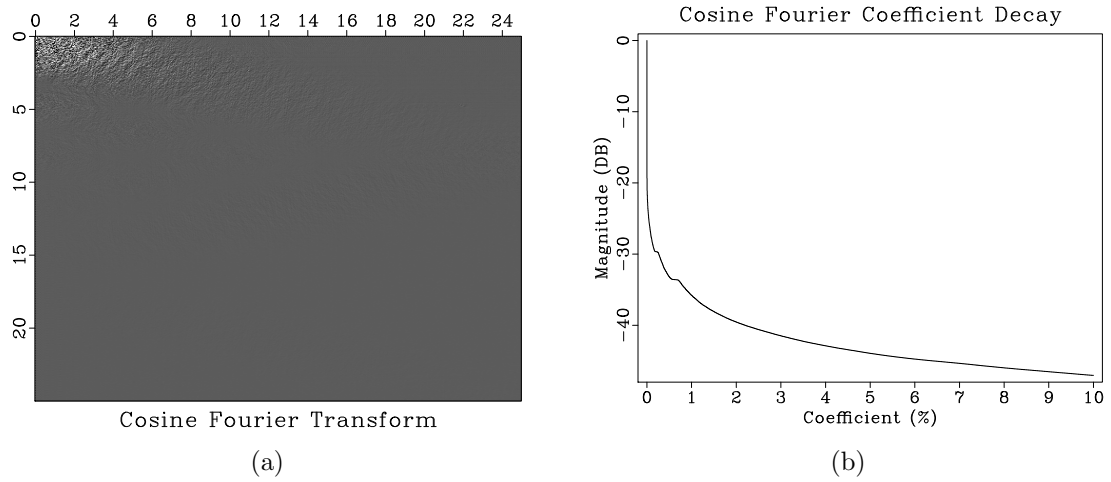


Figure 2: (a) Sand dunes image in the cosine Fourier domain. (b) Fourier coefficients sorted by absolute value and displayed on the logarithmic (decibel) scale.

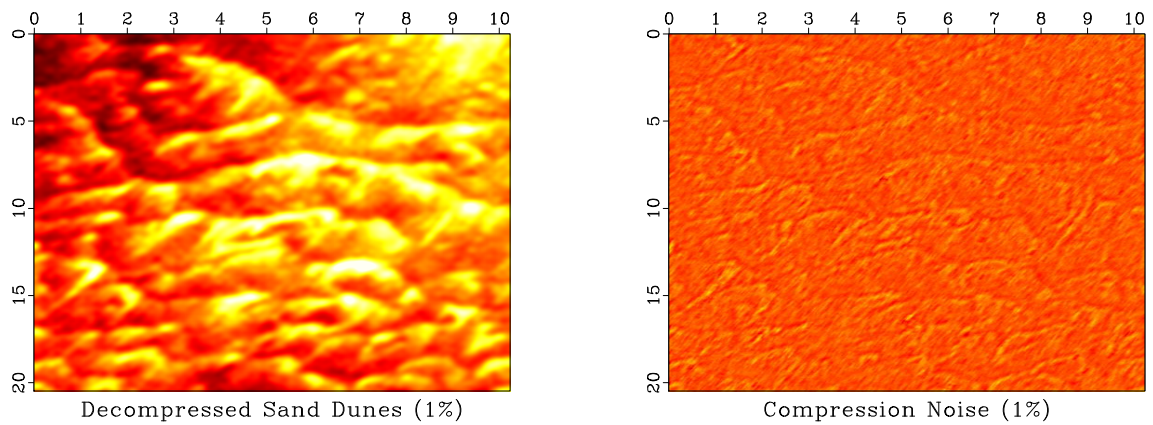


Figure 3: (a) Sand dunes image reconstructed after thresholding. (b) Compression noise.

Your task:

1. Change directory to `geo391/hw4/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.

```

1 from rsfproj import *
2
3 # Critical parameter
4 #####
5 perc = 1 # percentage for thresholding
6 #####
7
8 # Download data
9 Fetch('dunes3.HH', 'dunes')
10 Flow('dunes', 'dunes3.HH', 'dd form=native')
11
12 # Window size
13 n1=1024
14 n2=512
15
16 # Plotting macro
17 def plot(title):
18     return '''
19         grey color=H bias=-213 clip=150 title="%s"
20         ''' % title
21
22 # Display data
23 Flow('dune', 'dunes', 'window n3=1 n1=%d n2=%d' % (n1, n2))
24 Result('dune', plot('Sand Dunes'))
25
26 # Transform dictionary

```

```

27 #####
28 transform = {
29     'cos': ('Cosine Fourier',
30           'cosft sign1=1 sign2=1',
31           'cosft sign1=-1 sign2=-1'),
32     'dwt': ('Digital Wavelet',
33           '','
34           dwt type=b inv=y unit=y | transp |
35           dwt type=b inv=y unit=y | transp
36           '','
37           '','
38           transp | dwt type=b inv=y unit=y adj=y |
39           transp | dwt type=b inv=y unit=y adj=y
40           '',')
41     }
42
43 # Apply forward transform
44 Flow('cos', 'dune', transform['cos'][1])
45 Result('cos',
46       'grey title="%s Transform" ' % transform['cos'][0])
47
48 # Sort coefficients
49 Flow('sort', 'cos',
50     '','
51     put n1=%d n2=1 d1=%g label1=Coefficient unit1=%% |
52     math output="abs(input)" | sort | scale axis=1 |
53     math output="10*log(input)/log(10)"
54     '',' % (n1*n2,100.0/(n1*n2-1)))
55 Result('sort',
56     '','
57     window max1=10 |
58     graph title="%s Coefficient Decay"
59     label2=Magnitude unit2=DB
60     '',' % transform['cos'][0])
61
62 # Threshold and inverse transform
63 Flow('thr', 'cos', 'threshold pclip=%g' % perc)
64 Flow('inv', 'thr', transform['cos'][2])
65 Plot('inv', plot('Decompressed Sand Dunes (%g%%)' % perc))
66
67 # Noise = Data - Signal
68 Flow('diff', 'dune inv', 'add scale=1,-1 ${SOURCES[1]}')
69 Plot('diff',
70     plot('Compression Noise (%g%%)' % perc) + ' bias=0')
71

```

```

72 Result('inv','inv diff','SideBySideIso')
73
74 End()

```

MATCH FILTERING FOR ATTENUATION OF SURFACE SEISMIC WAVES

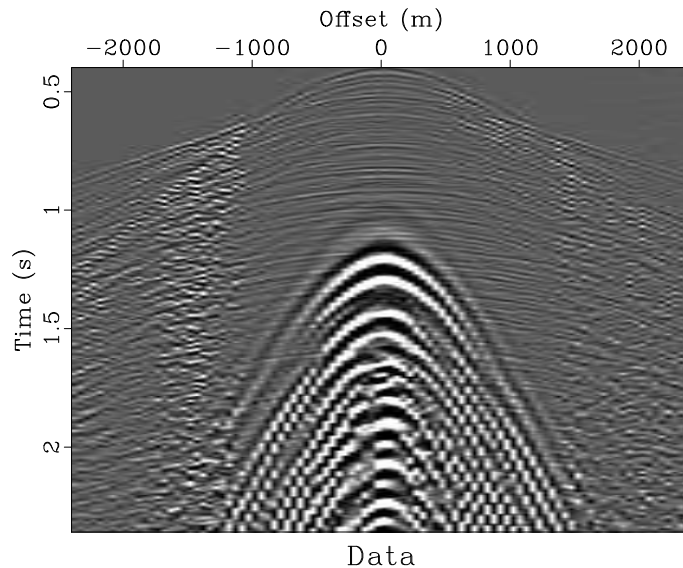


Figure 4: Seismic shot record from sand dunes in the Middle East. The data are contaminated by ground roll propagating in the sand.

Figure 4 shows a section out of a seismic shot record collected over sand dunes in the Middle East. The data are contaminated by ground roll propagating in the sand. A major data analysis task is to separate the signal (reflection waves) from the noise (surface waves).

A look at the data spectrum (Figure 5 shows that the noise is mostly concentrated at low frequencies. We can use this fact to create a noise model by low-pass filtering.

Figure 6 shows the noise model from low-pass filtering and inner muting and the result of subtracting this model from the data. Our next task is to match the model to the true noise by solving the least-squares optimization problem

$$\min \|\mathbf{N}_0 \mathbf{f} - \mathbf{d}\|^2, \quad (2)$$

where d is the data, f is a *matching filter*, and \mathbf{N}_0 represents convolution of the noise model \mathbf{n}_0 with the filter. After minimization, $\mathbf{n} = \mathbf{N}_0 \mathbf{f}$ becomes the new noise model, and $\mathbf{d} - \mathbf{n}$ becomes the estimated signal. Match filtering is implemented in program `match.c`. Some parts of this program are left out for you to fill.

```

19 #include <rsf.h>
20
21 int main(int argc, char* argv[])

```

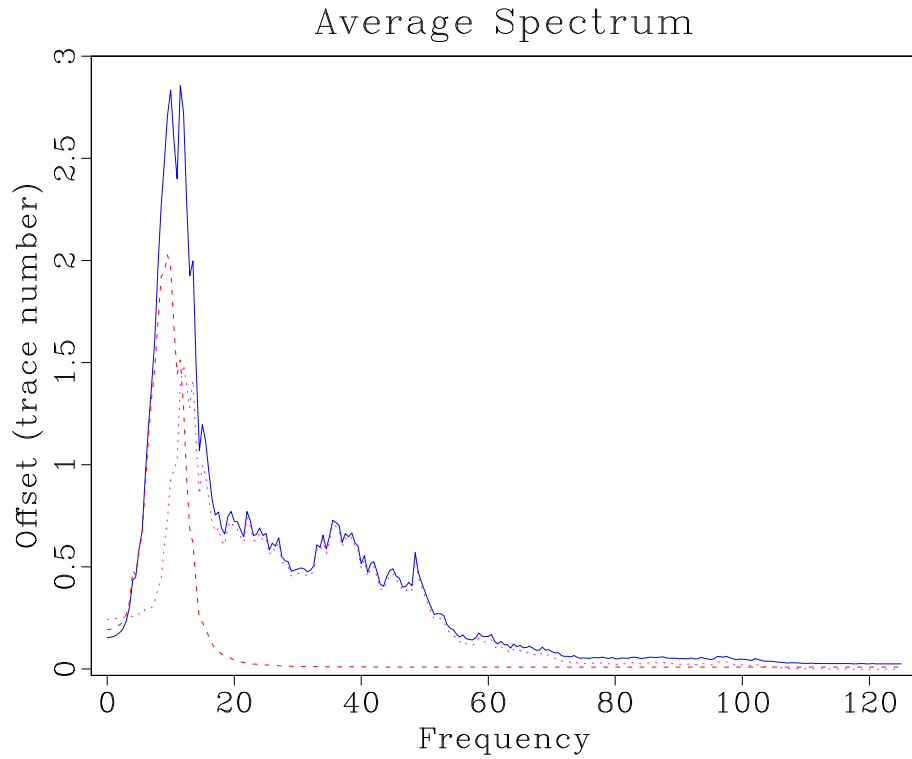


Figure 5: Data spectrum. Solid line – original data. Dashed line – initial noise model and signal model.

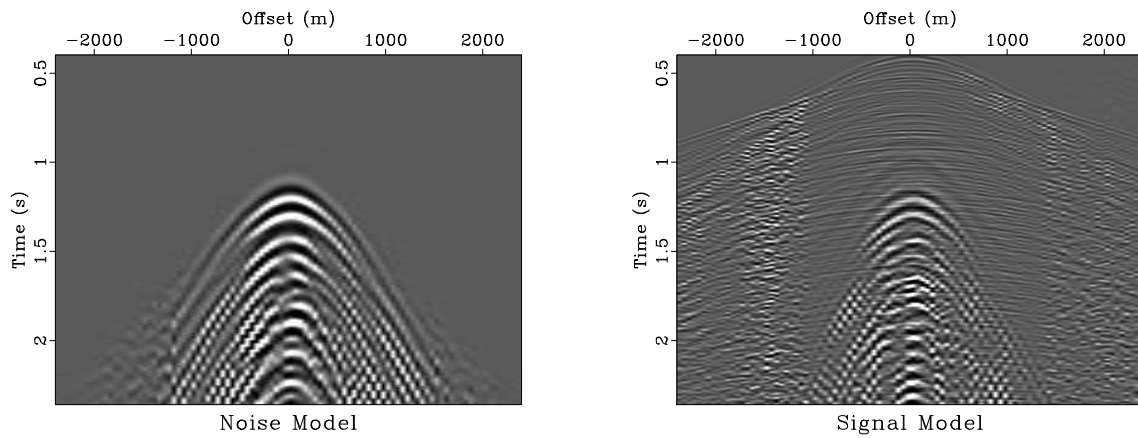


Figure 6: (a) Noise model created by low-pass filtering of the original data. (b) Result of subtraction of the noise model from the data.

```

22 {
23     bool adj;
24     int n1, n2, i1, i2, i, j, nf;
25     float *data, *noiz, *filt;
26     sf_file inp, out, oth;
27
28     sf_init(argc, argv);
29     inp = sf_input("in");
30     out = sf_output("out");
31     oth = sf_input("other");
32
33     if (!sf_getbool("adj",&adj)) adj=false;
34     /* adjoint flag */
35
36     if (adj) {
37         /* input data, output filter */
38         if (!sf_histint(inp,"n1",&n1)) sf_error("No n1=");
39         if (!sf_histint(inp,"n2",&n2)) sf_error("No n2=");
40         if (!sf_getint("nf",&nf)) sf_error("Need nf=");
41         /* filter size */
42
43         sf_putint(out,"n1",nf);
44         sf_putint(out,"n2",1);
45     } else {
46         /* input filter, output data */
47         if (!sf_histint(inp,"n1",&nf)) sf_error("No n1=");
48         if (!sf_histint(oth,"n1",&n1)) sf_error("No n1=");
49         if (!sf_histint(oth,"n2",&n2)) sf_error("No n2=");
50
51         sf_fileflush(out,oth); /* copy data dimensions */
52     }
53
54     filt = sf_floatalloc(nf);
55     data = sf_floatalloc(n1);
56     noiz = sf_floatalloc(n1);
57
58     if (adj) {
59         for (i=0; i < nf; i++) /* !!! COMPLETE LINE !!! */
60     } else {
61         sf_floatread(filt,nf,inp);
62     }
63
64     for (i2=0; i2 < n2; i2++) {
65         sf_floatread(noiz,n1,oth);
66

```

```

67     if (adj) {
68         sf_floatread /* !!! COMPLETE LINE !!! */
69     } else {
70         for (i1=0; i1 < n1; i1++) data[i1] = 0.;
71     }
72
73     for (i=0; i < nf; i++) {
74         for (i1=0; i1 < n1; i1++) {
75             j=i1-i+nf/2; /* symmetric filter */
76
77             /* zero value boundary conditions */
78             if (j < 0 || j >= n1) continue;
79
80             if (adj) {
81                 filt[i] += /* !!! COMPLETE LINE !!! */
82             } else {
83                 data[i1] += noiz[j]*filt[i];
84             }
85         }
86     }
87
88     if (!adj) sf_floatwrite(data,n1,out);
89 }
90
91 if (adj) sf_floatwrite /* !!! COMPLETE LINE !!! */
92
93 exit(0);
94 }

```

Your task:

1. Change directory to `geo391/hw4/match`
2. Run


```
scons view
```

 to reproduce the figures on your screen.
3. Modify the `match.c` file to fill in missing parts.
4. Test your modifications by running the dot product test.

```
scons dot.test
```

Repeating this several times, make sure that the numbers in the test match.

5. Modify the `SConstruct` file to display the results of match filtering and include them in your assignment.
6. **EXTRA CREDIT** for improving the results by either finding better parameters or by finding a better algorithm.

```

1 from rsfproj import *
2
3 # Critical parameters
4 #####
5 cut = 12 # cutoff frequency
6 nf = 11 # filter length
7 #####
8
9 # Download data
10 Fetch( 'dune3D.H', 'mideast' )
11
12 # Plotting macro
13 def grey( title ):
14     return '''
15         window n1=490 |
16         grey clip=2.5 title="%s"
17         label1=Time unit1=s label2=Offset unit2=m
18         ''' % title
19
20 # Select one 2-D slice
21 Flow( 'data', 'dune3D.H',
22     '''
23     dd form=native |
24     window n3=1 f3=2 n1=500 f1=100 |
25     scale dscale=100
26     ''' )
27 Result( 'data', grey( 'Data' ) )
28
29 # Create noise model by low-pass filtering
30 Flow( 'noise0', 'data',
31     '''
32     bandpass fhi=%g |
33     mutter half=n v0=1500 t0=0.8 hyper=y tp=0.12 |
34     cut n1=90
35     ''' % cut )
36 Plot( 'noise0', grey( 'Noise Model' ) )
37
38 # Signal = Data - Noise
39 Flow( 'signal0', 'data noise0', 'add scale=1,-1 ${SOURCES[1]} ' )

```

```

40 Plot('signal0',grey('Signal Model'))
41
42 Result('noise0','noise0 signal0','SideBySideIso')
43
44 # Plot spectrum
45 Plot('spec','data',
46     'spectra all=y | graph title="Average Spectrum" max2=3')
47 Plot('nspec0','noise0',
48     '','
49     spectra all=y |
50     graph wanttitle=n wantaxis=n max2=3 plotcol=5 dash=1
51     ''')
52 Plot('sspec0','signal0',
53     '','
54     spectra all=y |
55     graph wanttitle=n wantaxis=n max2=3 plotcol=4 dash=2
56     ''')
57 Result('spec0','spec nspec0 sspec0','Overlay')
58
59 # Matching filter program
60 match = Program('match.c')[0]
61
62 # Dummy filter
63 Flow('filt0',None,'spike n1=%d' % nf)
64
65 # Dot product test
66 Flow('dot.test','%s data noise0 filt0' % match,
67     '','
68     dottest ./${SOURCES[0]} nf=%d
69     dat=${SOURCES[1]} other=${SOURCES[2]}
70     mod=${SOURCES[3]}
71     '' % nf,stdin=0,stdout=-1)
72
73 # Conjugate-gradient optimization
74 Flow('filt','data %s noise0 filt0' % match,
75     '','
76     conjgrad ./${SOURCES[1]} nf=%d niter=%d
77     other=${SOURCES[2]} mod=${SOURCES[3]}
78     '' % (nf,2*nf))
79
80 # Extract new noise and signal
81 Flow('noise','filt %s noise0' % match,
82     './${SOURCES[1]} other=${SOURCES[2]}')
83 Flow('signal','data noise','add scale=1,-1 ${SOURCES[1]}')
84

```

85 | End()

MATCH FILTERING CONTINUED

Both the sand dune image and the seismic record are sections from 3-D data (time-lapse measurements in one case and 3-D seismic acquisition in the other).

1. In either of the examples, try using match filtering to predict one 3-D frame from another. You may need to modify the program to make the filter two-dimensional.
2. Include your results in the paper.

COMPLETING THE ASSIGNMENT

1. Change directory to `geo391/hw4`.
2. Edit the file `paper.tex` in your favorite editor and change the first line to have your name instead of Hestenes's.
3. Run

```
sftour scons lock  
sftour scons -c
```

and

```
scons pdf
```

4. Submit your result (file `paper.pdf`) by printing it out or by e-mail.