

Homework 2

Joseph Fourier

ABSTRACT

This homework has four parts.

1. Theoretical questions related to digital filtering.
2. Analyzing data by applying a running median filter.
3. Analyzing data by applying Fourier compression.
4. Applying one of these algorithms to your own data.

DIGITAL FILTERING

You can either write your answers on paper or edit them in the file `hw2/paper.tex`. Please show all the mathematical derivations that you perform.

1. You are given a table of numbers x_1, x_2, \dots, x_N and y_1, y_2, \dots, y_N and want to fit a parabolic model $y(x) = a + bx + cx^2$ to them.
 - (a) Using the method of least squares, find a set of linear equations for coefficients a , b , and c .
 - (b) Using the method of least squares, find a , b , and c for the case of $N = 2$.
2. The matrix in equation (1) represents a convolution operator with zero boundary conditions.

$$\mathbf{F} = \begin{bmatrix} f_1 & f_0 & 0 & 0 & 0 & 0 \\ f_2 & f_1 & f_0 & 0 & 0 & 0 \\ f_3 & f_2 & f_1 & f_0 & 0 & 0 \\ 0 & f_3 & f_2 & f_1 & f_0 & 0 \\ 0 & 0 & f_3 & f_2 & f_1 & f_0 \\ 0 & 0 & 0 & f_3 & f_2 & f_1 \end{bmatrix}. \quad (1)$$

The operator is implemented in the C code below.

```
15 void conv_lop (bool adj, bool add,  
16               int nx, int ny, float* xx, float* yy)  
17 /*< linear operator >*/
```

```

18 {
19     int f, x, y;
20
21     assert (ny == nx);
22     sf_adjnull (adj, add, nx, ny, xx, yy);
23
24     for (f=0; f < nf; f++) {
25         for (y = SFMAX(0, f-1); y < nx + f-1; y++) {
26             x = y - f + 1;
27             if( adj) {
28                 /* add code */
29             } else {
30                 yy[y] += xx[x] * ff[f];
31             }
32         }
33     }
34 }

```

- (a) Modify the matrix and the program to implement periodic boundary conditions.
- (b) Add the code for the adjoint (matrix transpose) operator.
3. Consider the parabolic filter $F(Z)$ defines as

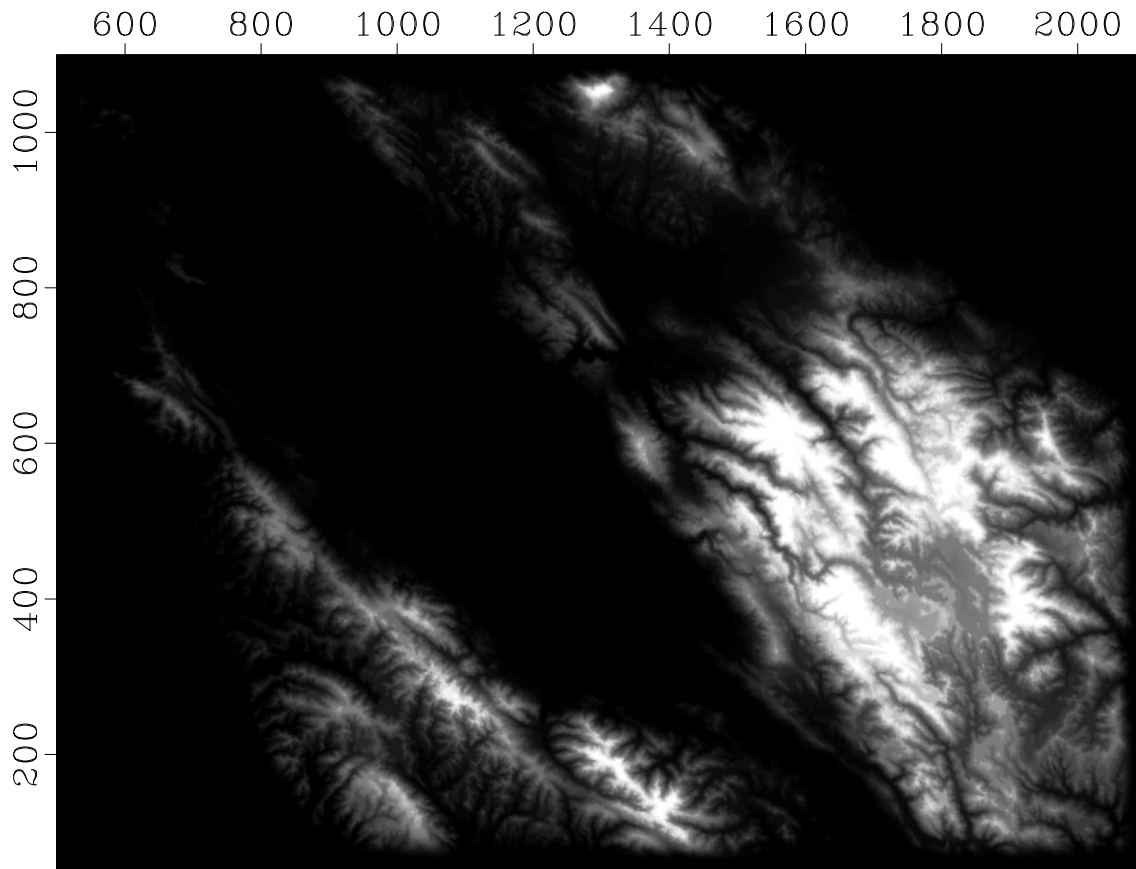
$$F(Z) = 1 + 4Z + 9Z^2 + \dots + N^2Z^{N-1} . \quad (2)$$

- (a) Show that this filter can be implemented with recursive filtering (polynomial division).
- (b) What is the advantage of recursive filtering? Does it depend on N ?
4. Using helical boundary conditions, a two-dimensional filter $F_2(Z_1, Z_2)$ can be mapped to a one-dimensional filter $H(Z)$ according to equation

$$H(Z) = F_2 \left(Z, Z^{N_1} \right) , \quad (3)$$

where N_1 is the length of the first (fast) axis. How would you map a three-dimensional filter $F_3(Z_1, Z_2, Z_3)$?

RUNNING MEDIAN AND RUNNING MEAN FILTERS



Elevation of San Francisco Bay

Figure 1: Digital elevation map of the San Francisco Bay Area.

We return the digital elevation map of the San Francisco Bay Area, shown in Figure 1.

In this exercise, we will separate the data into “signal” and “noise” by applying running mean and median filters. The result of applying a running median filter is shown in figure 2. Running median effectively smooths the data by removing local outliers.

The algorithm is implemented in program `running.c`.

```

/* Apply running mean or median filter */

#include <rsf.h>

static float median(int n, float *a)
/* find median by slow sorting, changes a */
{

```



```

        break;
    case 2: /* mean */
    default:
        /* ADD CODE */
        break;
    }
}

/* write out */
sf_floatwrite( signal[0], n1*n2, out );
}

exit(0);
}

```

1. Change directory to `geo391/hw2/running`
2. Run

```
scons view
```

to reproduce the figures on your screen.

3. Modify the `running.c` program and the `SConstruct` file to compute running mean instead of running median. Compare the results.
4. **EXTRA CREDIT** for improving the efficiency of the running median algorithm. Run

```
scons time.vpl
```

to display a figure that compares the efficiency of running median computations using the slow sorting from function `median` in program `running.c` and the fast quantile algorithm (library function `sf_quantile`). Your goal is to make the algorithm even faster. Consider parallelization, reusing previous window results, other fast sorting strategies, etc.

```

1 from rsfproj import *
2
3 # Download data
4 Fetch( 'bay.h', 'bay' )
5
6 # Window and taper
7 Flow( 'bay', 'bay.h',

```

```

8         '''
9         dd form=native |
10        window f2=500 n2=1600 f1=50 n1=1050 |
11        costaper nw1=50 nw2=50
12        '''
13
14 # Display
15 def plot(title):
16     return '''
17     grey crowd=0.85 clip=1111 allpos=y yreverse=n
18     title="%s"
19     ''' % title
20
21 Result('bay',plot('Elevation of San Francisco Bay'))
22
23 # Running median program
24 run = Program('running.c')
25
26 #####
27 # MODIFY ME
28 #####
29 Flow('med','bay %s' % run[0],
30     './${SOURCES[1]} w1=20 w2=20 what=0')
31 Result('med',plot('Signal'))
32
33 # Difference
34 Flow('res','bay med','add scale=1,-1 ${SOURCES[1]}')
35 Result('res',plot('Noise') + ' allpos=n')
36
37 # slow or fast
38 for case in range(2):
39
40     ts = []
41     ws = []
42
43     time = 'time%d' % case
44     wind = 'wind%d' % case
45
46     # loop over window size
47     for w in range(3,16,2):
48         itime = '%s-%d' % (time,w)
49         ts.append(itime)
50
51         iwind = '%s-%d' % (wind,w)
52         ws.append(iwind)

```

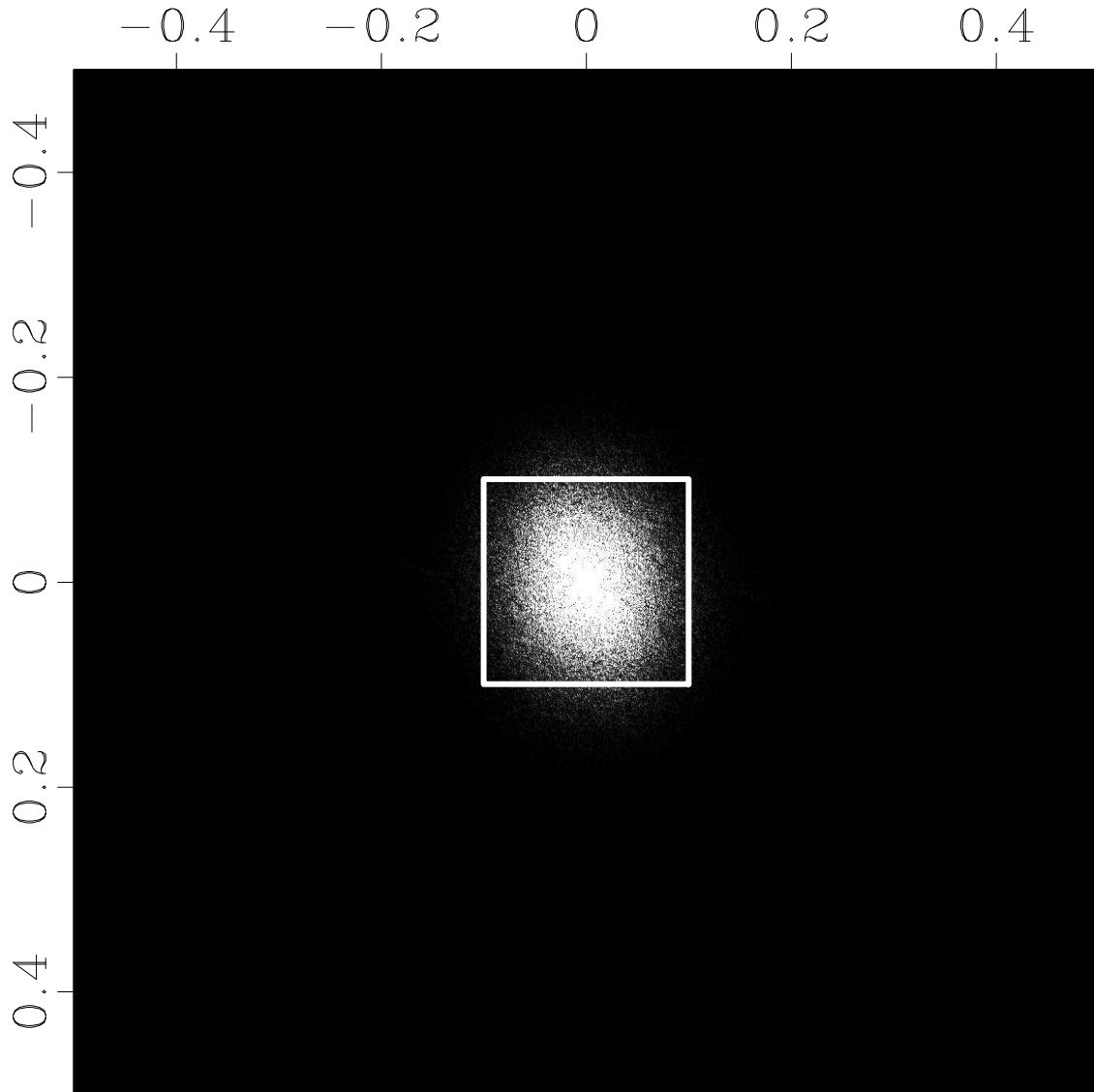
```

53
54     # measure CPU time
55     Flow(iwind, None, 'spike n1=1 mag=%d' % (w*w))
56     Flow(itime, 'bay %s' % run[0],
57           ', , ,
58           ( (/usr/bin/time -f "%S %U"
59             ./${SOURCES[1]} < ${SOURCES[0]}
60             w1=%d w2=%d what=%d > /dev/null ) 2>&1 )
61             > time.out &&
62             (tail -1 time.out;
63              echo in=time0.asc n1=2 data_format=ascii_float)
64             > time0.asc &&
65             dd form=native < time0.asc | stack axis=1 norm=n
66             > $TARGET &&
67             /bin/rm time0.asc time.out
68             ' ' ' % (w,w,case), stdin=0, stdout=-1)
69
70     Flow(time, ts, 'cat axis=1 ${SOURCES[1:%d]}' % len(ts))
71     Flow(wind, ws, 'cat axis=1 ${SOURCES[1:%d]}' % len(ws))
72
73     # complex numbers for plotting
74     Flow('c'+time, [wind, time],
75           ', , ,
76           cat axis=2 ${SOURCES[1]} |
77           transp |
78           dd type=complex
79           ' ' ')
80
81     # Display CPU time
82     Plot ('time', 'ctime0 ctime1',
83           ', , ,
84           cat axis=1 ${SOURCES[1]} | transp |
85           graph dash=0,1 wanttitle=n
86           label2="CPU Time" unit2=s
87           label1="Window Size" unit1=
88           ' ' ', view=1)
89
90     End()

```

FOURIER COMPRESSION

The goal of your next assignment is to find a compressed representation of the data in the Fourier transform domain. Figure 3 shows the Fourier transform of the digital



Fourier Transform

Figure 3: Absolute value of the Fourier transform of the digital elevation data. The frame inside shows a window selected for compression.

elevation data from Figure 1. We can see that most of the energy gets concentrated near the center (zero frequency).

There are two alternative ways to compress data in the Fourier domain:

- One approach is to select a range of frequencies that contain the most important information. An advantage of this approach is the ability to subsample the original data by transforming back from a windowed range of frequencies. The results from this method are shown in Figure 4.
- Another approach is to zero all Fourier coefficients below a certain threshold value, regardless of which frequencies they represent. The results from this method are shown in Figure 5. Figure 6 shows a selected threshold plotted against the histogram of Fourier coefficients.

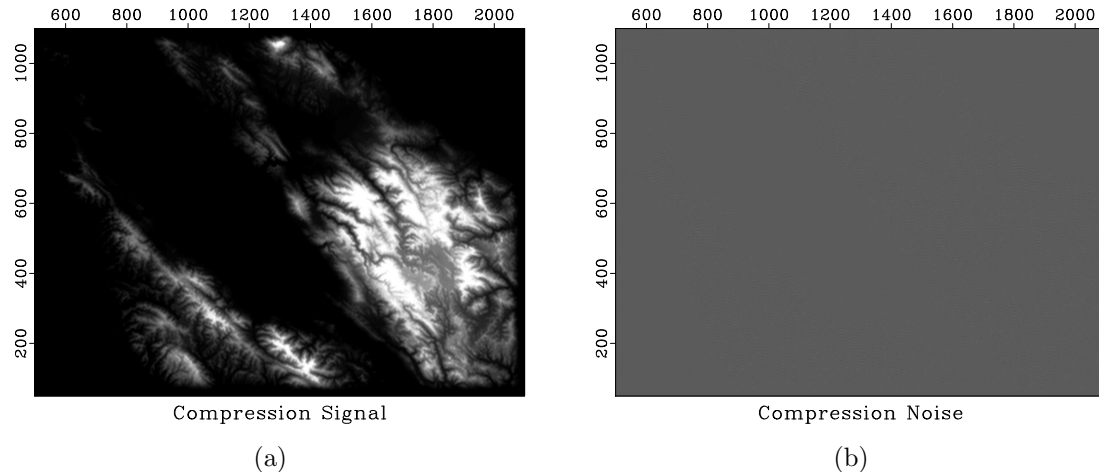


Figure 4: Data separated into signal (a) and noise (b) by applying Fourier compression with windowing.

1. Change directory to `geo391/hw2/fourier`
2. Run

```
scons view
```

to reproduce the figures on your screen.

3. Modify the `SConstruct` file to decrease the size of the window so that the noise level increases in Figure 4(b). How do you measure the noise level? Find a level that you find negligibly small.
4. Modify the `SConstruct` file to increase the threshold value so that the compression achieves the same quality as in the previous case. The noise level in Figure 5(b) should match that in Figure 4(b).

5. Compare the number of nonzero Fourier coefficients in both cases. Which method achieves a better compression?
6. **EXTRA CREDIT** for finding a way for a better compression of the data in the Fourier domain. Your data reconstruction should have the same noise level, yet the number of non-zero coefficients in the Fourier domain should be smaller.

```

1 from rsfproj import *
2
3 # Download data
4 Fetch( 'bay.h', 'bay' )
5
6 # Window and taper
7 Flow( 'bay', 'bay.h',
8       ' ',
9       dd form=native |
10      window f2=500 n2=1600 f1=50 n1=1050 |
11      costaper nw1=50 nw2=50
12      ' ')
13
14 # Display
15 Result( 'bay',
16         ' ',
17         grey crowd=0.85 clip=1111 allpos=y yreverse=n
18         title="Elevation of San Francisco Bay"
19         ' ')
20
21 # 2-D Fourier Transform
22 Flow( 'fft', 'bay',
23       ' rtoc | fft3 axis=1 pad=1 | fft3 axis=2 pad=1' )
24 Plot( 'fft',
25       ' ',
26       math output="abs(input)" | real |
27       grey title="Fourier Transform" allpos=y screenratio=1
28       ' ')
29
30 # A. Compression by Windowing
31 #####
32
33 cut = 0.1 # Fixed
34
35 # Plot a frame
36 Flow( 'frame.asc', None,
37       'echo %s n1=10 in=$TARGET data_format=ascii_float' %
38       ' '.join( map( str, [ cut ]*3+[ -cut ]*4+[ cut ]*3)))

```

```

39 Plot('frame', 'frame.asc',
40     ' ',
41     dd type=complex form=native |
42     graph plotfat=5 plotcol=0
43     min1=-0.5 min2=-0.5 max1=0.5 max2=0.5 screenratio=1
44     wantaxis=n wanttitle=n pad=n transp=y yreverse=y
45     ' ')
46
47 Result('fft', 'fft frame', 'Overlay')
48
49 # Cut a square hole
50 Flow('fcut', 'fft',
51     ' ',
52     cut min1=%g max1=%g min2=%g max2=%g
53     ' ' % (-cut, cut, -cut, cut))
54
55 # Inverse FFT
56 Flow('cut', 'fcut',
57     'fft3 axis=2 inv=y | fft3 axis=1 inv=y | real')
58 Result('cut',
59     ' ',
60     grey crowd=0.85 clip=1111 yreverse=n
61     title="Compression Noise"
62     ' ')
63
64 Flow('sig', 'bay cut', 'add scale=1,-1 ${SOURCES[1]}')
65 Result('sig',
66     ' ',
67     grey crowd=0.85 clip=1111 allpos=y yreverse=n
68     title="Compression Signal"
69     ' ')
70
71 # B. Compression by Thresholding
72 #####
73
74 thr = 10000 # Change Me
75
76 # Plot histogram
77 Plot('hist', 'fft',
78     ' ',
79     math output="abs(input)" | real |
80     histogram o1=0 d1=%g n1=101 |
81     dd type=float | scale axis=1 |
82     graph title="Scaled Histogram"
83     ' ' % (3*thr/100))

```

```

84 Flow('line.asc',None,
85       'echo 0 0 0 1 n1=4 data_format=ascii_float in=$TARGET')
86 Plot('line','line.asc',
87       ',')
88       dd type=complex form=native |
89       graph min1=-1 max1=2 plotcol=5 wantaxis=n wanttitle=n
90       ',')
91 Result('hist','hist line','Overlay')
92
93 # Thresholding
94 Flow('fthr','fft','thr thr=%g' % thr)
95
96 # Inverse FFT
97 Flow('thr','fthr',
98       'fft3 axis=2 inv=y | fft3 axis=1 inv=y | real')
99 Result('thr',
100        ',')
101        grey crowd=0.85 clip=1111 allpos=y yreverse=n
102        title="Compression Signal"
103        ',')
104
105 # Subtract from Data
106 Flow('noi','bay thr','add scale=1,-1 ${SOURCES[1]}')
107 Result('noi',
108        ',')
109        grey crowd=0.85 clip=1111 yreverse=n
110        title="Compression Noise"
111        ',')
112
113 End()

```

YOUR OWN DATA

Your final task is to apply one of the data analysis techniques of the previous sections to your own data:

1. Select a dataset suitable for running mean/median filters or for Fourier compression.
2. Apply one the algorithm of the previous sections and choose appropriate parameters.
3. Include the results in your homework.

COMPLETING THE ASSIGNMENT

1. Change directory to `geo391/hw2`.
2. Edit the file `paper.tex` in your favorite editor and change the first line to have your name instead of Fourier's.

3. Run

```
sftour scons lock
```

and

```
scons pdf
```

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