Viewing and editing strainmeter data

In this section of the course we will review
1. Obtaining raw strainmeter data
2. Working with SEED data
3. Viewing and editing strainmeter data.

1. Obtaining raw strainmeter data

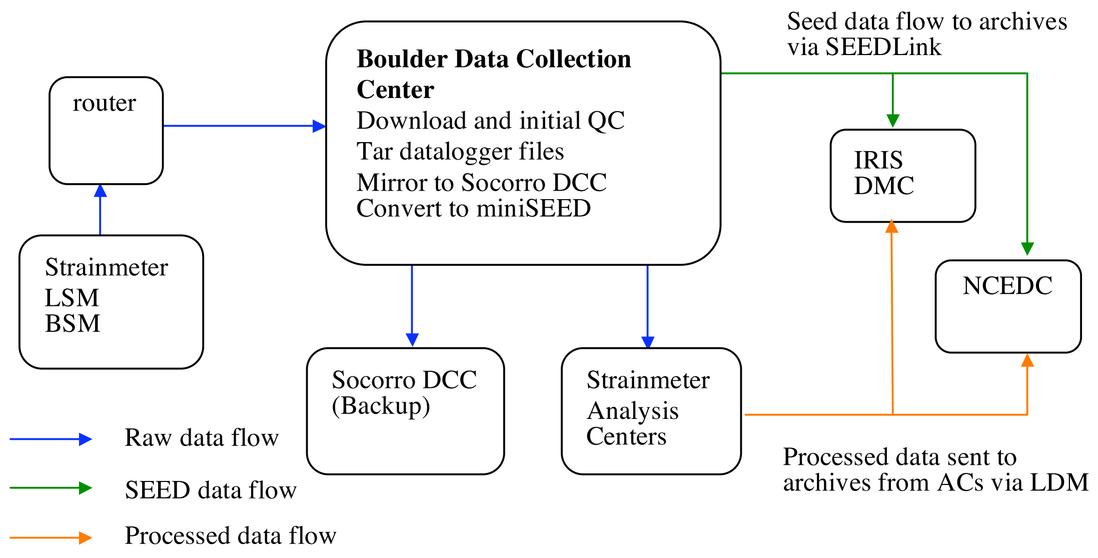


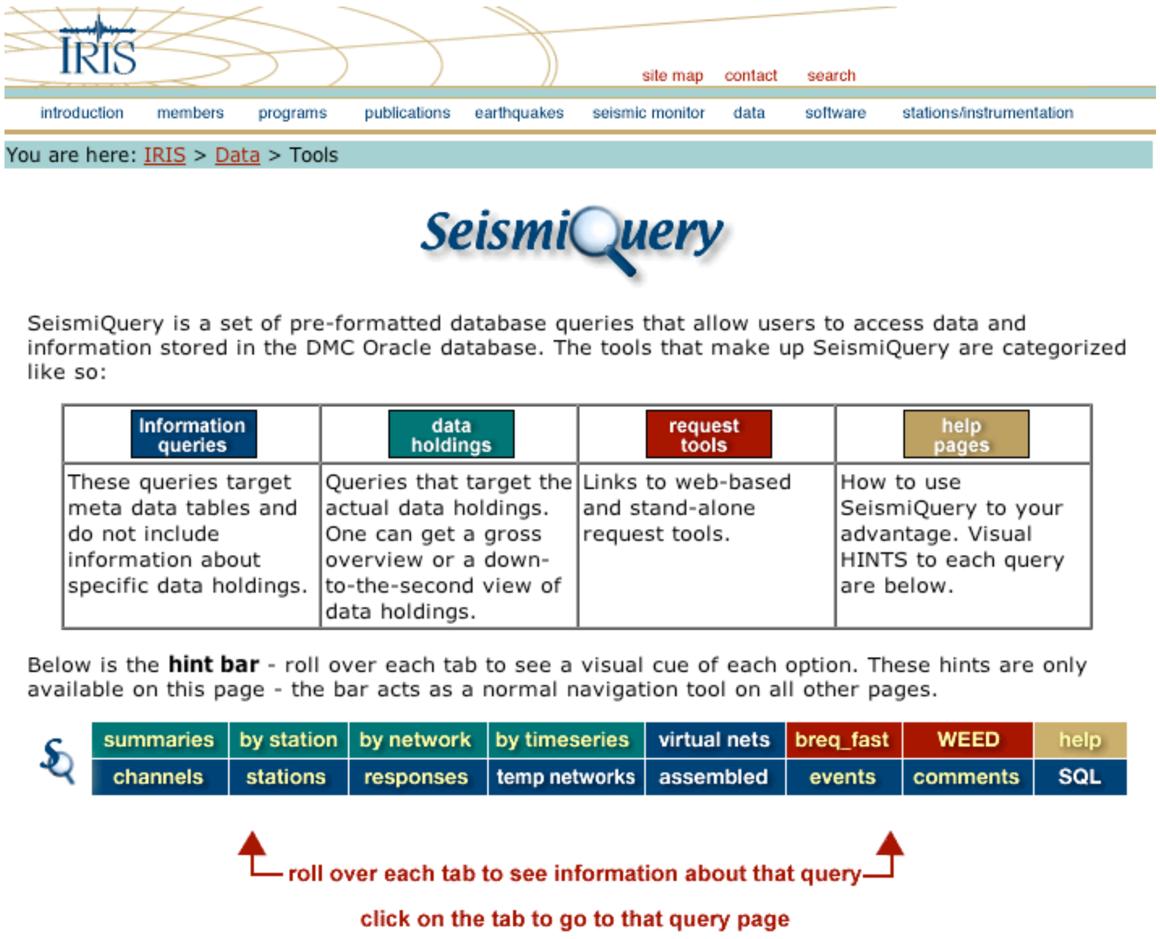
Figure 1. PBO Strainmeter Data Flow

Raw strainmeter data flows from the strainmeter to PBO's Boulder Data Collection Center (BDCC). In Boulder the data undergo basic quality control, are converted to miniSEED and then forwarded to the archive centers, the IRIS Data Management Center (IRIS DMC) and the Northern California Data Center (NCEDC). The data will be archived in the format in which it comes off the data logger, bottle format for the borehole strainmeters and ICE-9 format for the laser strainmeters, and SEED. The idea being that we are archiving SEED data and the data as it comes directly from the datalogger with no changes made to it at all.

The native data logger files will be archived in a very rudimentary way, an FTP site where you can download 24-hour chunks of data in tar files. You will not be able to query this database. The SEED database will be fully managed and considered the primary database. You will be able to query the database to see what strainmeters where operational at what times, what channels are measured and what other data are measured at the site. You will also be able to retrieve station metadata from the SEED database.

The 1 Hz and 1/300 Hz laser strainmeter data plus associated diagnostic data will be updated at 24-hour intervals. The 20 Hz and 1 Hz borehole strainmeter data will be updated hourly and the 1/600 Hz data and associated diagnostic data updated every 24 hours.

There are web-based interfaces with which to query the SEED database. SeismicQuery is a tool provided by IRIS DMC http://www.iris.edu/SeismiQuery/. NCEDC also uses SeismicQuery http://quake.geo.berkeley.edu/SeismiQuery/ .



SeismiQuery is a set of pre-formatted database queries that allow users to access data and information stored in the DMC Oracle database. The tools that make up SeismiQuery are categorized like so:

Information queries	data holdings	request tools	help pages
These queries target meta data tables and do not include information about specific data holdings.	Queries that target the actual data holdings. One can get a gross overview or a down-to-the-second view of data holdings.	Links to web-based and stand-alone request tools.	How to use SeismiQuery to your advantage. Visual HINTS to each query are below.

Below is the **hint bar** - roll over each tab to see a visual cue of each option. These hints are only available on this page - the bar acts as a normal navigation tool on all other pages.

Figure 2 SeismicQuery web page at the IRIS DMC web site.

There is also a program called VASE that allows you to view the data before you download it http://www.iris.edu/DHI/install_clients/Vase2.1_Build_Output/Web_Installers/install.htm. Figure 3 shows the strain measured at 1 Hz on gage 1 of 4 Bay Area strainmeters during the M7.2, 15 June 2005, Crescent City earthquake.

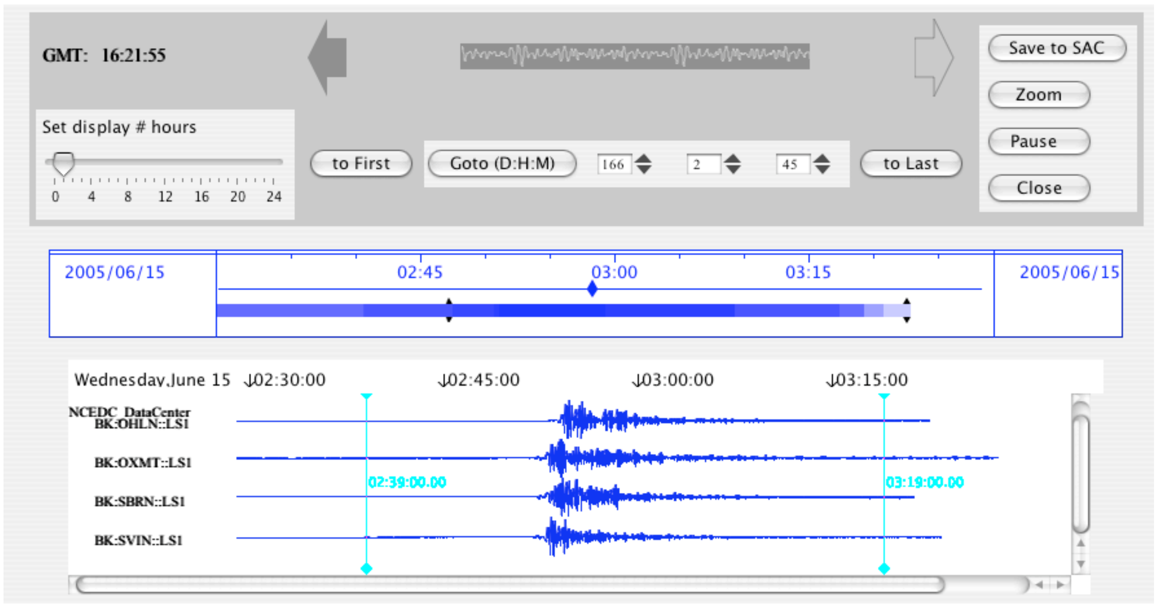


Figure 3. Strain changes on 4 Bay Area strainmeters during the June 15 2005, Crescent City earthquake displayed using VASE2.1.

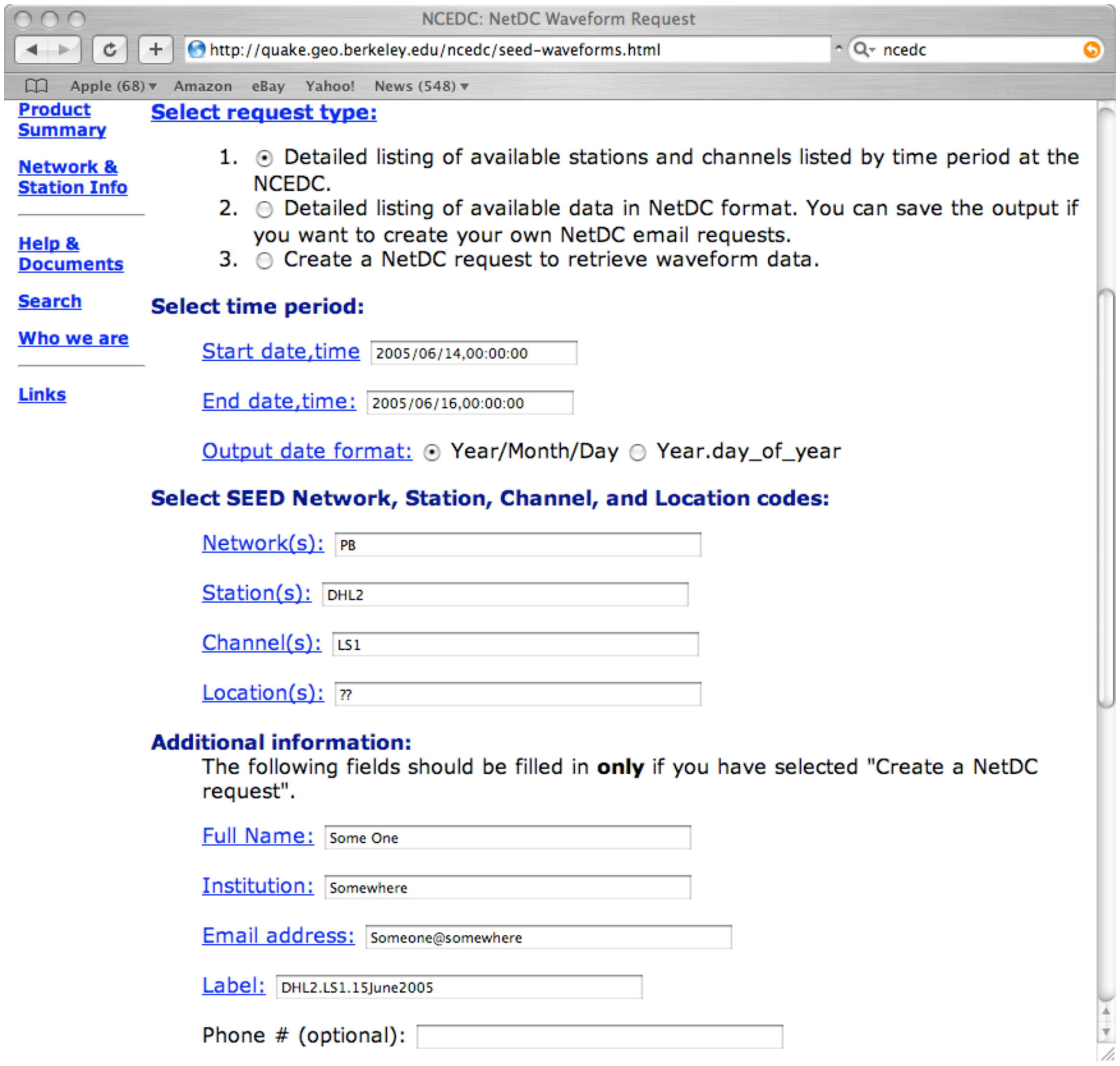
All data channels recorded by the strainmeters are uniquely identified in the archives by a set of four SEED codes:

NETWORK CODE	2 characters, PB = PBO stations
STATION CODE	4 characters, unique to each strainmeter
CHANNEL CODE	3 characters, describes the sample rate and quantity being measured
LOCATION CODE	2 characters, used to differentiate between two channels with the same channel code.

For example, the strain measurement from the Durmid Hill laser strainmeter is uniquely identified in the archives by the network code PB, station code DHL2, the channel code LS1 and the location code LM. The SEED channel codes that describe each data stream from a PBO strainmeters are given as an appendix to these notes.

We can download SEED strainmeter data by filling out web based query forms at either of the archive centers or using VASE. IRIS has an excellent online tutorial on how to retrieve data from its website (http://www.iris.washington.edu/manuals/DATutorial.htm).

We will look at the NCEDC web site http://quake.geo.berkeley.edu/ncedc/seed-waveforms.html (Figure 4). To see all the channels measured by a strainmeter you must know the network code (always PB for PBO instruments) and the 4-character station code. The borehole strainmeters will have the 4-character code BXXX or PXXX, where XXX is a 3-digit number, the station codes of currently operating laser strainmeters are DHL2 and GVS1. Entering just the network code PB and the station code should allow you to see what channels were recorded for a particular time period. To download 2 days of 1 Hz strain data from the laser strainmeter at Durmid Hill around the time of the M7.2 Crescent City Earthquake, 15 June 02:50:54 UTC, one would fill in the network code, PB, station code, DHL2 and channel code, LS1 as in figure 4.


NCEDC: NetDC Waveform Request
http://quake.geo.berkeley.edu/ncedc/seed-waveforms.html
ncedc
Apple (68) Amazon eBay Yahoo! News (548)
Product Summary
Network & Station Info
Help & Documents
Search
Who we are
Links

Select request type:

1. [✓ Detailed listing of available stations and channels listed by time period at the NCEDC.]
2. Detailed listing of available data in NetDC format. You can save the output if you want to create your own NetDC email requests.
3. Create a NetDC request to retrieve waveform data.

Select time period:

Start date,time	2005/06/14,00:00:00
End date,time:	2005/06/16,00:00:00
Output date format:	[✓ Year/Month/Day] Year.day_of_year

Select SEED Network, Station, Channel, and Location codes:

Network(s):	PB
Station(s):	DHL2
Channel(s):	LS1
Location(s):	??

Additional information:
The following fields should be filled in **only** if you have selected "Create a NetDC request".

Full Name:	Some One
Institution:	Somewhere
Email address:	Someone@somewhere
Label:	DHL2.LS1.15June2005
Phone # (optional):	

This form generates a request that is sent to NCEDC. The details we provided would produce the following request form.

```
.NETDC_REQUEST
.NAME Some One
.INST Somewhere
.EMAIL Someone@somewhere
.MERGE NO
.LABEL DHL2.LS1.15June2005
.END
.DATA * PB DHL2  "--" RS1 "2005 06 14 00 00 00.8528" "2005 06 15 23 59 59.8528"
```

You are notified by email when the data has been placed in an area you can retrieve it via anonymous FTP. Usually the email arrives within a few minutes.

2. Working with SEED data

The data retrieved from the archives using the request format above will be in SEED. A SEED file contains metadata, such as the coordinates, scale factors, filter information and, time series data. The file can be read using the program rdseed which can be downloaded from the IRIS website http://www.iris.edu/manuals/ . We will review the commands useful for this course using a SEED file containing 600-second interval data recorded by the Garin volumetric dilatometer in the San Francisco Bay Area from the 10th to the 17th June 2005. The SEED codes for the data channel are network, UL, station, SFGAR, channel, RVO and location code V0. The example seed file is called SFGAR.RV0.CrescentCity2005.

You can examine the contents of the file by typing

```
rdseed –c –f SFGAR.RV0.CrescentCity2005
```

The output shows the start and stop times and all channels in the SEED volume. You can examine the instrument responses and station information by typing

```
rdseed -s -f SFGAR.RV0.CrescentCity2005 > channels.txt
```

The file channels.txt contains all the metadata for each of the channels in the SEED file. The rdseed program allows you to write out the SEED file in 6 different formats, SAC, AH, CSS, miniSEED, SEED, SAC ASCII and SEGY.

As an example of how to convert SEED into other formats which may be more compatible with software that you have we will take our SEED file, convert it to an ASCII time stamped file and plot it with GMT4.0. The first step is to convert the SEED file into Seismic Analysis Code (SAC) ASCII format. We do this using rdseed,

```
rdseed –o 6 –d  –f SFGAR.RV0.CrescentCity2005
```

where –o 6 informs the program we want output in SAC ASCII, -d means we want a data dump and –f is followed by the input file name. The program should produce the file 2005.161.00.00.00.0000.UL.SFGAR.V0.RV2.D.SAC_ASC, which we will rename as SFGAR.V0.RV2.D.SAC_ASC.

```
mv 2005.161.00.00.00.0000.UL.SFGAR.V0.RV2.D.SAC_ASC SFGAR.V0.RV2.D.SAC_ASC
```

The Perl program rdsacascii will read a SAC ASCII file and output the data in various formats. Typing -h shows the output formats,

```
rdsacascii -h
Usage: rdsacascii -h -f infile -o k|g|t|n
    -h     help
    -f     infile
    -o     output format
           k Kaleidagrpah
           g GMT4.0
           t tsview
           d data only  (to convert to niolib rdsacascii -i infile -o d | datput 100 )
```

The script is run by typing

```
rdsacascii –f SFGAR.V0.RV2.D.SAC_ASC -o g > SFGAR.V0.RV2.D.SAC_ASC.txt
```

where -f is followed by the input filename and -o by the output format. The d option will print only the time series data and the character "a" after all the data have been printed which is required is the output is to be piped into the PIASD data conversions programs. The output is directed to SFGAR.V0.RV2.D.SAC_ASC.txt, which contains the time and value of each data point. The data will be in GMT4.0 format; the file should contain 1008 points and the first line should be 2005-06-10T00:00:00.0000 00055313. The data can then be plotted using a simple GMT script such as plot.sh, which produced figure 5.

```
plot.sh SFGAR.V0.RV2.D.SAC_ASC.txt
```

An alternative method would in handling SEED data would be to convert the data to miniSEED and use the msi program written by Chad Trabent (chad@iris.washington.edu) at IRIS who has developed a set of tools to handle miniSEED data (http://www.iris.edu/chad/).

3. Viewing and editing strainmeter data.

There are several packages available to edit raw strainmeter data, each has it own advantages. Here are a few interactive editing programs,

- Kaleidagraph. http://www.synergy.com/. Kaleidagraph is a commercial plotting program (~$140). It handles dates well; you can mask, edit and manipulate the data using user-defined or inbuilt mathematical formulae. It can be used to produce publication quality figures of the data. The only disadvantage is that it does not record edits.

- Tsview. A Matlab based program developed by the GPS community (http://www-gpsg.mit.edu/~tah/GGMatlab/ and http://www.ngs.noaa.gov/gps-toolbox/Herring1.htm). Tsview is free as long as you have Matlab. There is a binary available that you can run without the Matlab license but there may be portability issues. A nice feature is that it records edits as you make them. This program expects the data in a decimal year. If you use tsview and want to try it with strain data, **rdsacascii** will convert a SAC ASCII file into a format that will be readable by tsview. The output format option should be t for tsview.

- CREDIT. Part of the Programs for the Interactive Analysis of Strainmeter Data (PIASD) package (Agnew, 2004). Advantages, it is free and runs on UNIX/LINUX/OSX platforms. This program allows you to record edits and contains features to remove offsets in the data using linear interpolation.

We did not want anybody to have to purchase any commercial software or get a hold of Matlab for this course so we will look at CREDIT.

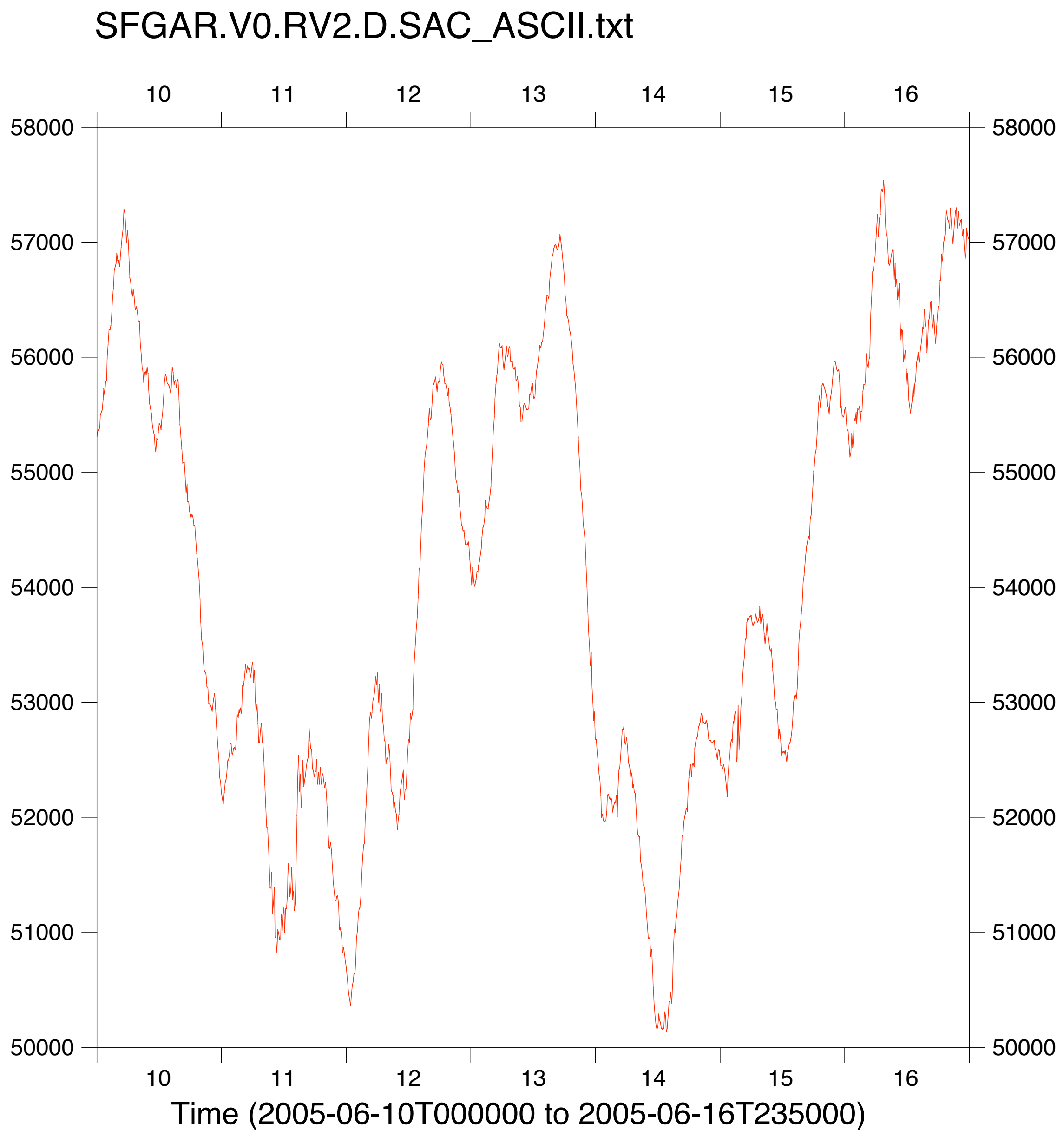


Figure 5. GMT plot of SEED file SFGAR.V0.RV2.SAC_ASCII.txt .

CREDIT is a part of the Programs for the Interactive Analysis of Strainmeter Data (PIASD) , Agnew, 2004. Credit reads and writes binary files in niolib format. Niolib filename are numbers and are dependant on the data type.

data type	file number	filename
short integers	100 to 999	I100.D to I999.D
long integers	1100 to 1999	L100.D to L999.D
real values	-100 to -999	R100.D to R999.D

Often editing strain or tilt data is an iterative process. Estimating step sizes in the data, coseismic or instrumental, is best done with the tides removed. However, steps must be removed from the data before tidal analysis. For this reason editing strain or tilt data is usually at least a two stage process. In the first stage steps and outliers are removed from the raw data with the tides still in the data. The tidal signal is estimated from the resulting edited data set. In the second stage the tidal signal is removed from the raw data, using the results from the tidal analysis, and step sizes are re-estimated without the presence of tides in the data. Several iterations may be necessary if there are many steps before a final data is set found.

The file L109.D in the example data set (/home/data/credit_example/class_example) contains examples of what you may need to edit out of strain or tiltmeter data. The file contains almost 2 months of 10-minute interval data from a volumetric-type strainmeter in the Bay Area. We will plot the data, edit some obvious outliers, flag periods of noisy data and correct offsets.

First we need to start an Xterm and set it to tectronix mode by typing xterm –t. We start credit by typing **credit** and loading the file L109.D in the credit example directory.

credit

File number	**1109**
Start term	**1**
Plot limits	**0 0**
Mode of operation (1 for editing, 0 for display)	**1**
Decimation factor (typically 1 for editing)	**1**
Number for previous edits file (0 for none):	**0**
Predictor weights (type 1 if wanted)	**0**
File number for writing out edits:	**1209**

Credit plots 1000 points at a time by default. Hitting **<return>** will scroll you through the file 1000 points at a time.

Editing glitches

There are 3 obvious glitches in the first data window, points 1 to 1000 (figure 6). To get rid of glitches we will go into cursor edit mode. Type

c <return>

this turns the cursor on; a small cross will appear. Once in cursor edit mode you do not need to type return after typing.

To zoom in on the first glitch move the cross hairs until they are over the spike. Where the cross hairs are vertically does not matter. Type

=

after hitting = 3 times you should see figure 7. If you zoom in too much typing

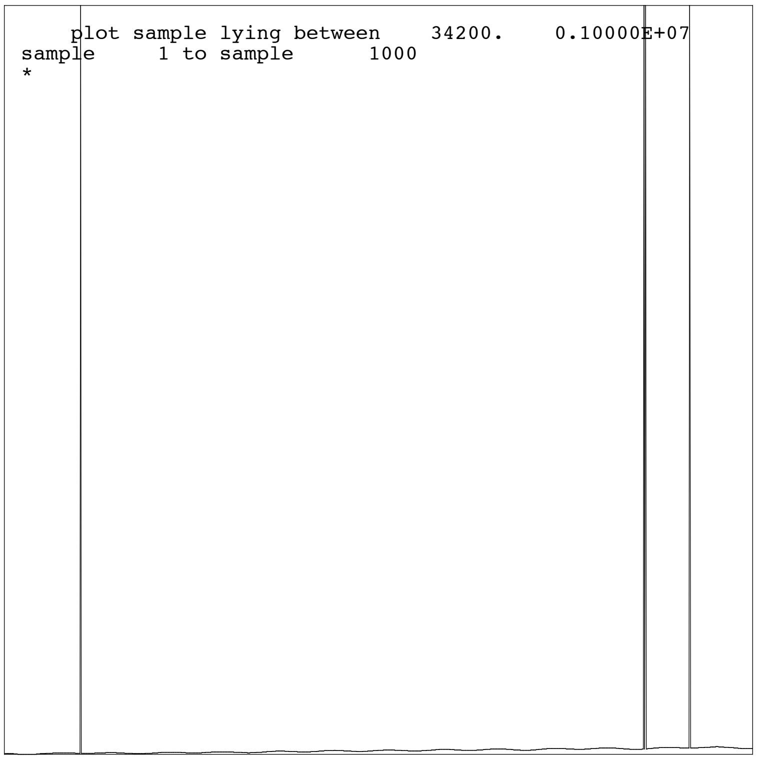
–

will zoom out the plot. To edit a **g**litch type

g .

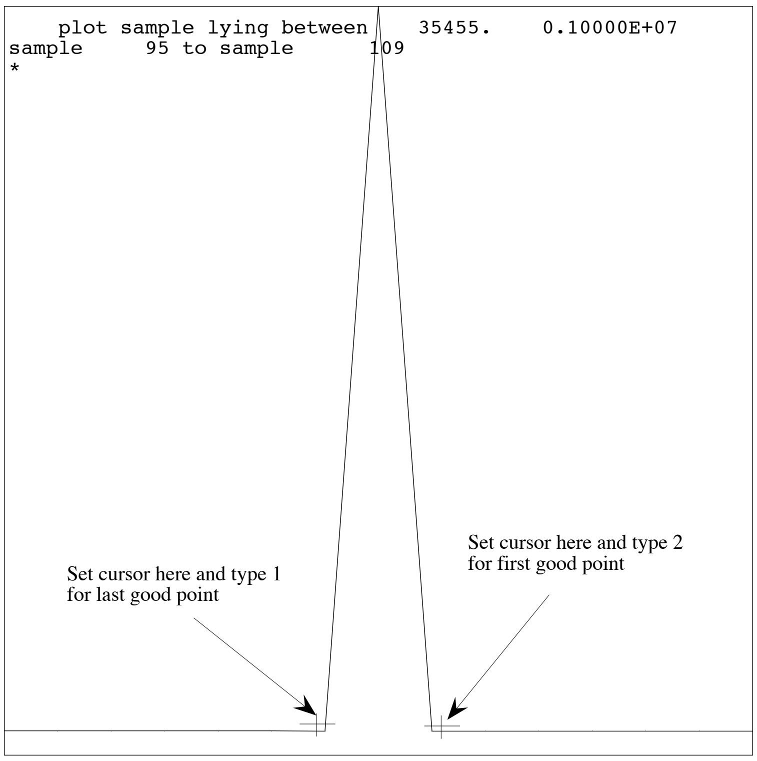
You will be told,

Set cursor on last good point and type 1
Then on first good point and type 2 .



```
    plot sample lying between    34200.    0.10000E+07
sample     1 to sample      1000
*
```

Figure 6.



```
    plot sample lying between    35455.    0.10000E+07
sample     95 to sample      109
*
```

Figure 7

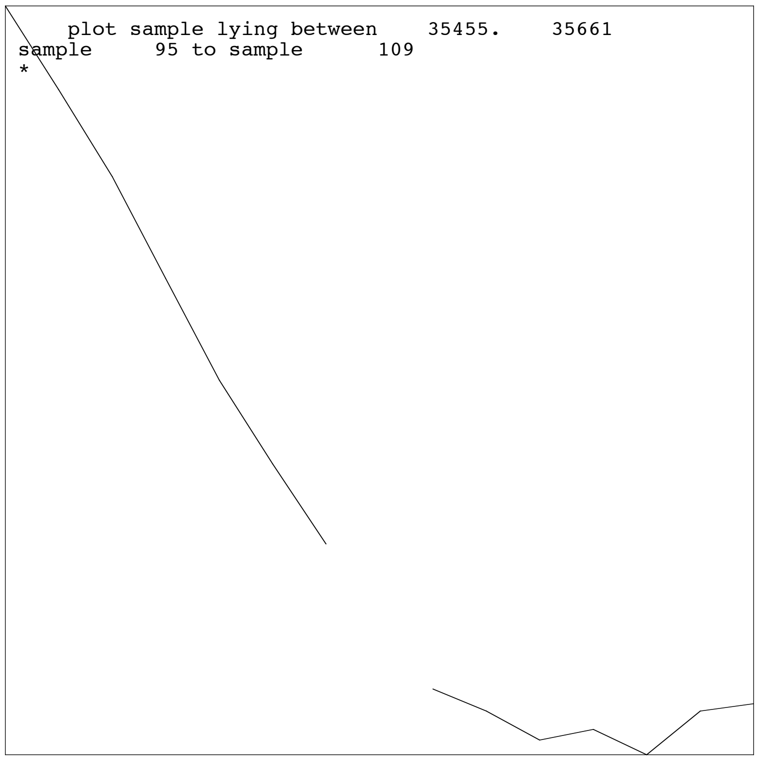
The last good point refers to the last good data point preceding the glitch. The first good point refers to the first good point immediately following the glitch. Move the arrows to the points shown in figure 7 and type

1

while positioned over the last good point and type

2

while positioned over the first good point. You do not need to type **<return>** after typing 1 or 2. The screen should automatically redraw as shown in figure 8.



```
     plot sample lying between    35455.    35661
 sample     95 to sample      109
 *
```

Figure 8.

You can exit cursor edit mode by typing

h

and then

<return>

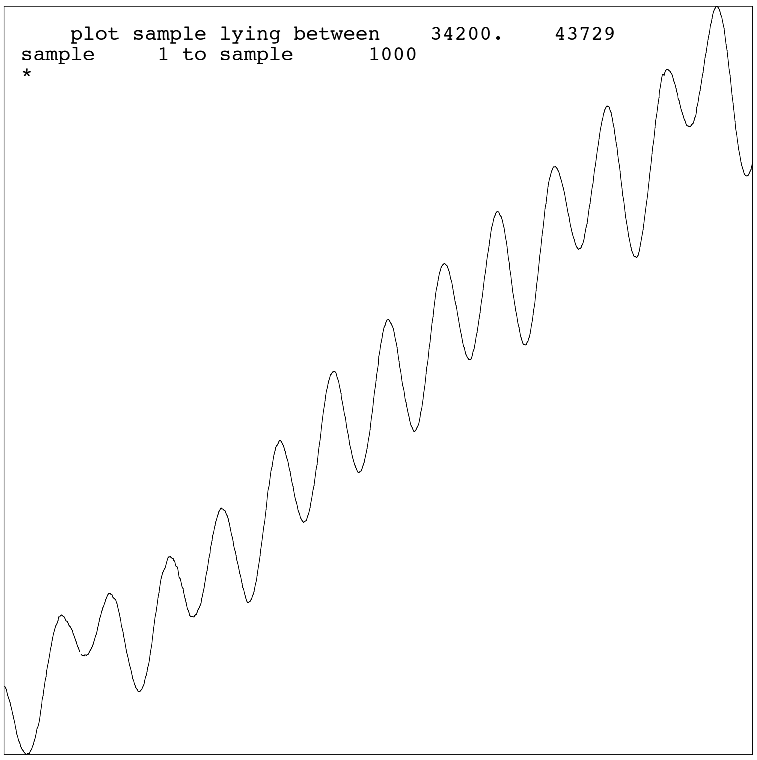
The first 1000 points will be redrawn. There are two remaining glitches between points 1 and 1000. Remove these glitches using the same method used to remove glitch one. The figure should redraw as in figure 9.

Editing sections of noisy data

Typing

<return>

will draw points 2001 to 3000. There is a period of noisy data shown with no obvious offset. You can remove sections of bad data in the same manner you would remove a glitch. Type



```
    plot sample lying between    34200.    43729
sample     1 to sample      1000
*
```

Figure 9.

c <return>

to get into cursor edit mode and then

+

to zoom in (figure 10). Type

g

Place the cursor over the last good point and type

1

and then over the first good point after the bad section and type

2 .

The plot should redraw with the bad section missing. Type

h

to exit edit mode and then

<return>

To view points 1001 to 2001 again.

Editing offsets

Typing

<return>

will draw points 2000 to 3000 where there is an offset (figure 10). To see the x and y value of positions of points go into cursor mode again

c <return>

and type

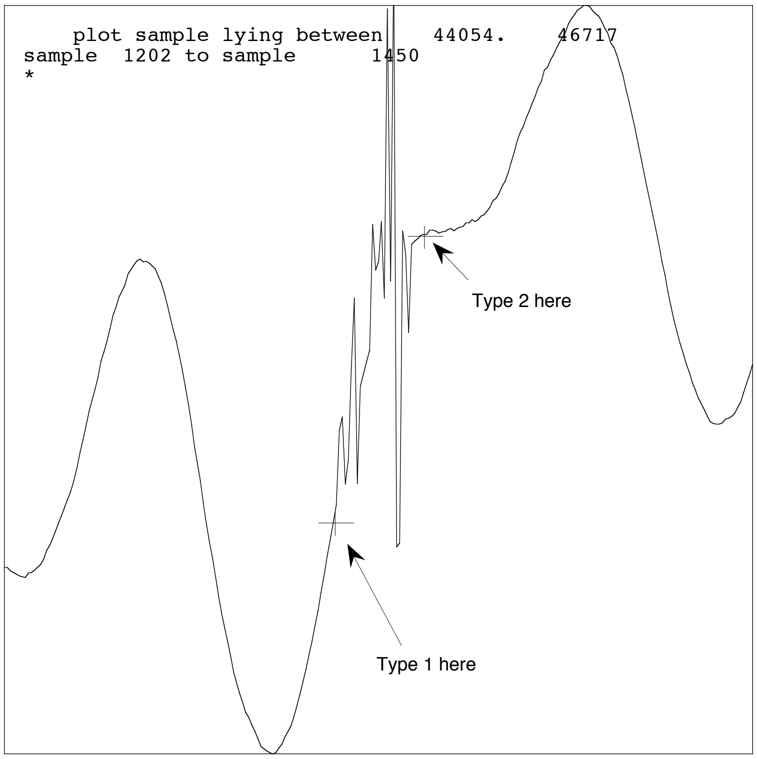


Figure 10.

p

We can zoom in on the offset by placing the cross near the offset and typing,

= .

You should see a plot similar to figure 11 (gray dotted line shows the offset removed). To edit an offset type

O .

Again place the cursor on the last good point and type

1

and the first good point and type

2 .

If we type **!** we can see the offset correction options;

3 - set point selected by vertical cursor to level of horizontal cursor
4 - enter the offset at the keyboard
5 - make the offset -1 times the last one,
6 - offset set by linear interpolation

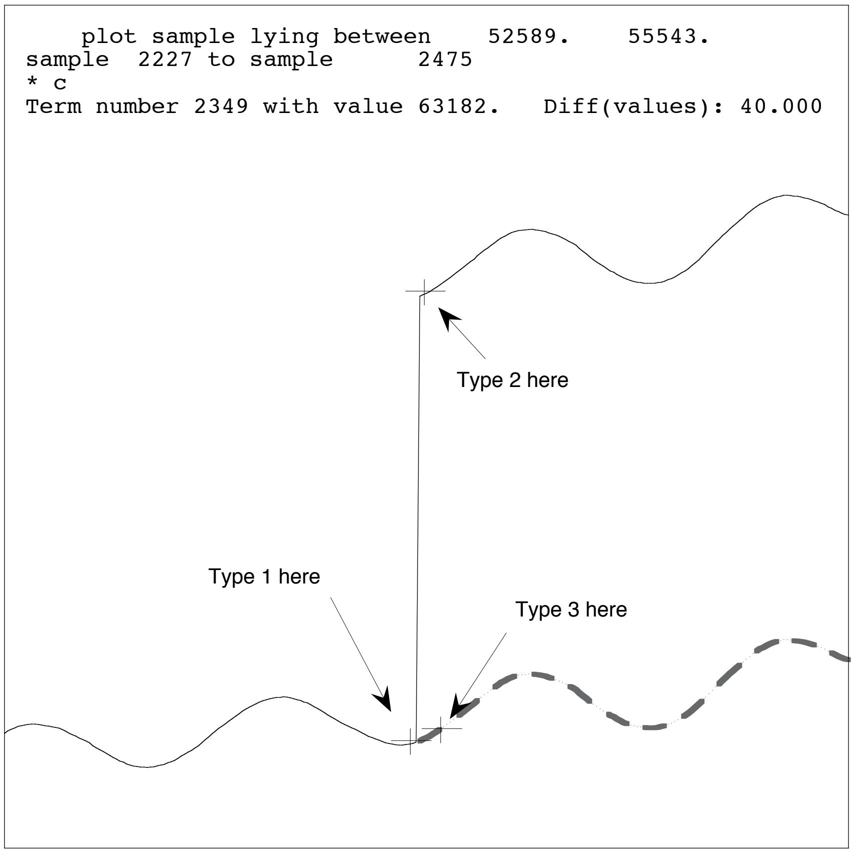
This example is that of a step between two data points. We shall remove the step using option three. Type

3

Move the cursor to the vertical position at which the first good data point would lie if the step where not there and type 3. The figure should be redrawn as the gray dotted line in figure 11. Typing

h

should bring us back to the cursor mode and typing **<return>** will replot points 2001 to 3000.



```
    plot sample lying between    52589.    55543.
sample  2227 to sample      2475
* c
Term number 2349 with value 63182.   Diff(values): 40.000
```

Figure 11.

Now we will look at how to correct an offset using option 6, by linear interpolation. If we type

<return>

we should now be looking at points 3000 to 4000 (figure 12). There is a period of noisy data before and after an offset. First, zoom in on the offset so you can identify the last and first good points by typing

c

And then

=

Then type

o

to enter offset correction mode. Select the last good point before the bad data and the first good point after it as before by typing **1** and **2**. Then type **!** to see the options and enter **6**. We now will select periods of good data before and after the offset. Credit will fit a straight line to each of these data sections and then calculate the offset that will make the lines connect. To identify the periods of good data zoom out and look at the data that preceded the offset. Type

- -

to zoom out twice (figure 13).

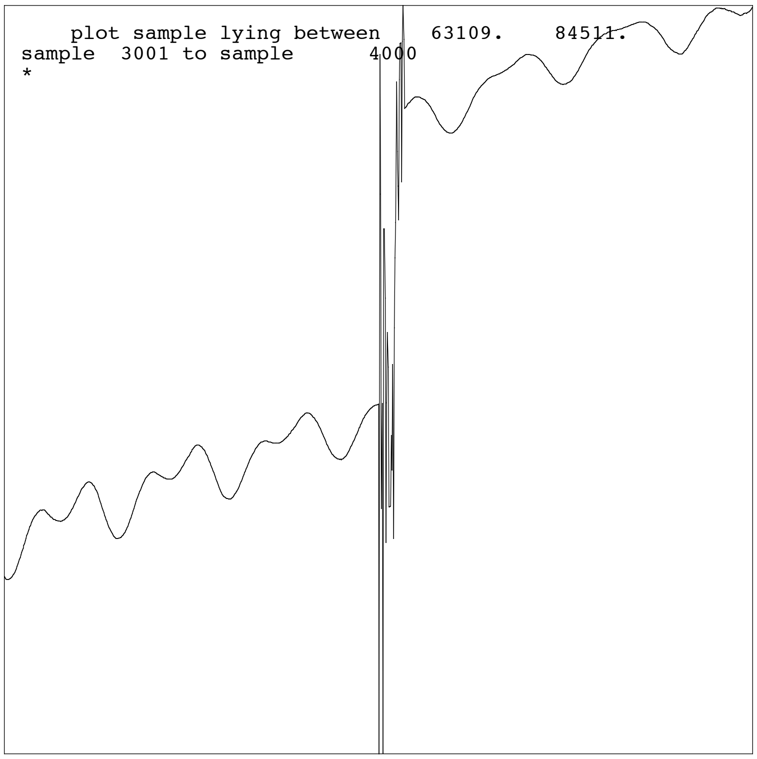


Figure 12

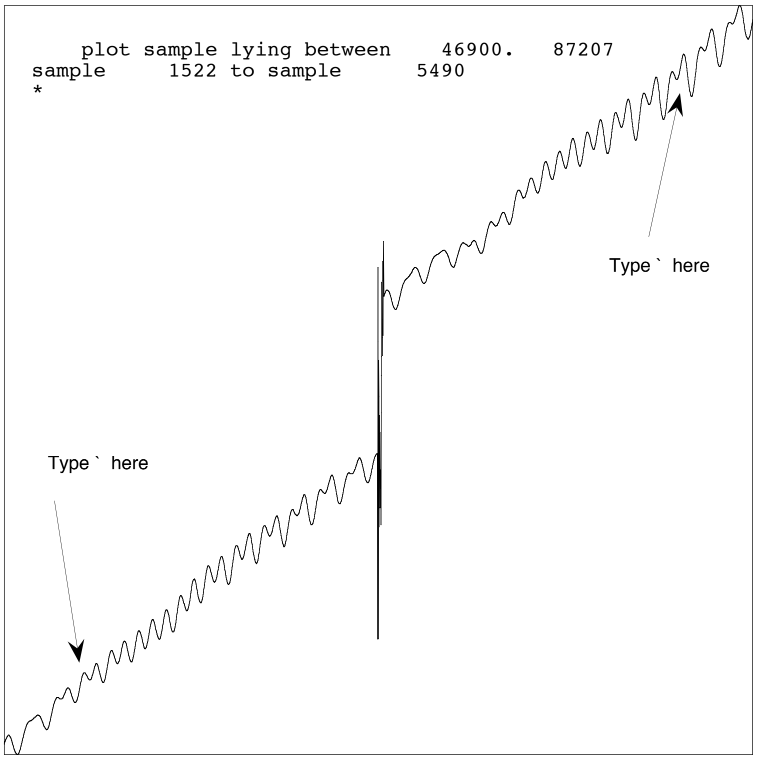


Figure 13

Select the start of the good data by placing the cursor over the point and typing

```
`
```

A straight line will be fit between this point and the last good data point before the offset. Find the point to mark on the right hand side of the offset by zooming out and then zooming in again around the point we are interested in. When we have found the point we want type

```
`
```

over the point. The plot should redraw with the offset removed and the bad data not plotted. If we get out of cursor mode by typing

h

we can see the entire data set by repeatedly zooming out by typing

c <return>

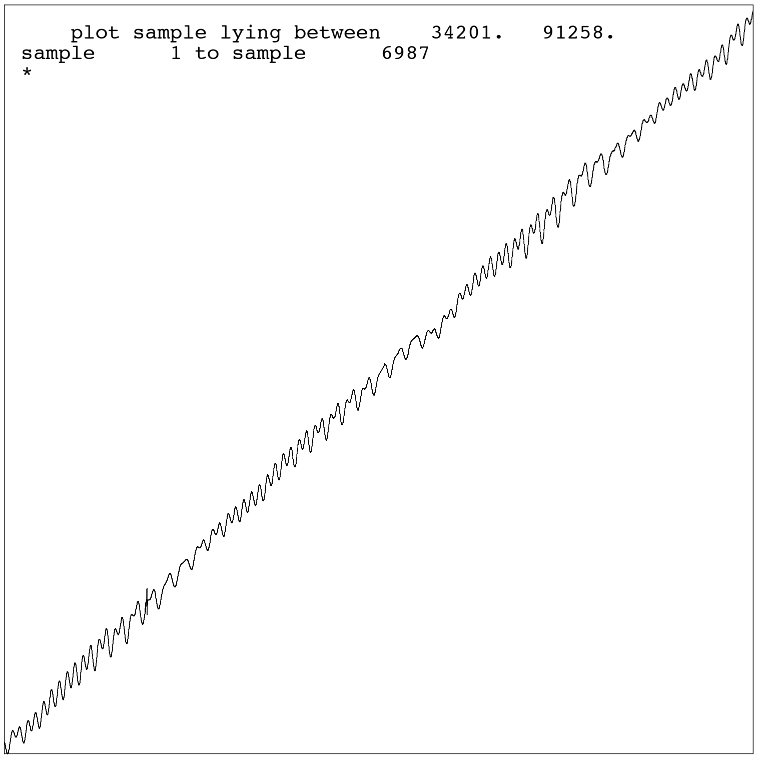
- - - - - -

We should see a plot similar to figure 14. To exit the program type

h

then

x .



```
    plot sample lying between    34201.   91258.
sample      1 to sample      6987
*
```

Figure 14

Our editing will have been saved to file L209.dat. The next time we run credit on this section of data at the line “Number for previous edits file (0 for none):” if we enter

1209

then we will be presented with the choice to either apply the edits or view the edits on the plot.

We can view an ASCII version of the edits file by typing

putdat 1209 > 1209.edit.txt

The very first term in the file is the number of edits. The edit information is then stored in triplets. The first term is the number of terms from the preceding edit to the last good term before the next edit. The second is the offset to be applied following the edit and the third is the number of bad points. If we edit bad data as in a glitch and do not correct an offset the offset value is simply zero.

Looking at the first 5 lines of the file 1209.edit.txt we should see numbers similar to the following,

```
6          100        0          2          753
0          2          57         0          2
394        0          28         1008       -9989
2          1144       -7570      46         0
0          0          0          0          0
```

There are six edits in the file. The 100th term is the last good point before the first edit. No offset correction was applied (it was an outlier). Two points were marked as bad. The next set of edit information follows. The last good point was 753 terms from the first good term following the previous edit, again no offset correction was applied and 2 points were marked as bad. If we want to produce a cleaned data set we can run the program autocl. This will ask for the edits file and the raw data file and apply the edits. The clean data are output to a separate file.

Often once the data set has been cleaned we want to decimate it to use it for tidal analysis or look at longer-term trends. We will work through the steps to do this with our sample data set. The first program to run after editing is autocl, which applies the edits (L209.D) to the raw data stream (L109.D) and creates a clean data set (L309.D). The raw data file remains unaltered. Because we will want to filter and decimate the data we are going to instruct autocl to linearly interpolate over the data we have flagged as bad in the edit file. Run autocl by typing

```
autocl
        What option (h for help)     ; h
        There are four options for editing a file:
          n - interpolate, with offset (normal)
          s - add slope to bad data and offset
          o - interpolate, no offset
          c - fill with constant, with offset
          f - fill with constant, no offset
          and to get a binary edits file (1's where there are edits
          and 0's elsewhere) use any option with an input file number
          of zero

        What option (h for help)     ; n
         Input file number and length (-1 for all) ; 1109 -1
         Output file number                    ; 1309
         Edits file number                     ; 1209
         Initial offset                 ; 0
```

The next step is to filter and decimate the data. We use the program deki to do this. We will reduce the 10-minute interval data to 1-hour interval data. Deki creates a zero phase shift low pass filter, which is adequate for 10-minute data. We can design the filter to our specifications. We will instruct the program that frequencies in the stopband should be attenuated by 60 dB, the width of the transition band between the passband and stopband is 0.2 times the Nyquist frequency and that the corner frequency is 0.89 times the Nyquist frequency. We input our cleaned example file L309.D. The output will contain real values so we will give it the file number -409 and it will be written to a file named L409.D. When the program asks us to specify a start up option we will say option 3 as the program then calculates where the first point in the filtered data set lies with respect to the original data set.

```
Deki
        Input file number; 1309
        Output file number; -409
        Begin and end terms of input series; 1 -1
        Decimation integer; 6
        Do you want internal (1) or external (0) weights?; 1
        Nyquist frequency of input series; 6
        Attenuation in stopband(s), in db (>0); 60
        Width of transition band(s); .2
        Total number of weights required is  219
        Type 1 if ok, 0 if not; 1
        Filter type (1 for lowpass, 2 highpass, 3 bandpass, 4 bandstop); 1
        Corner frequency (center of transition band); .89
        Start, end, interval freq (0 0 0 to go on); 0 0 0
        Startup option (-10 for choices); -10
        Available options are:
             0 - no padding: in doing convolution first term of input
                 will be multiplied by first weight
             1 - pad beginning of series with npad zeroes
             2 - pad beginning of series with npad numbers equal to
                 the first term of the input series
                 (for both 1 and 2, npad will be such that for a zero phase
                 shift filter (with an odd number
                 of weights) the first output point
                 will be at the same time as the first input point,
                 for -1 or -2, npad will equal the number of weights)
             3 - start convolution within series at a place that will put the
                 first output point at a nice time (for a zero phase shift filter)
            -3 - as 3, but get time from header
        Startup option (-10 for choices); 3
        time of beginning term of input series (d,h,m,s); 92 0 0 0
        sample interval of series (sec); 600
```

If everything runs the program should print out the following,

```
       first output point will be at  92 19  0  0
           1129 terms written to file   -409
```

Now we have a clean, filtered data set, but perhaps we would like to go back and remove the points we interpolated over and replace them with outlier flags so we know not to use them in subsequent analysis. If this is the case we need to run autocl again on the filtered data set. This time we should chose option f which fills bad data with an outlier flag rather than interpolate and not apply the offset correction as we already did that in the first run of autocl. Before we can do this however, we need to reformat our 1209 edit file so it is also a 1-hour interval file. We use shred to do this.

```
shred
What option (1-7)                    ; 1
Start time of output (ydhms, or nnnh for header); 2004 92 19  0 0
Sample interval of output series (sec) ; 3600
Length of output series              ; 1129
Start time of input (s to stop, [file]h to use header  e.g. 400h )
 (Use data, **NOT EDITS**, header number)  ;2004 92 0 0 0
Sample interval of input series (sec) ; 600
Length of input series               ; 1399
 2004 92 19 0 0 2004 92 0 0 0 134546216 3600 600 1129 6993
 edits file in; 1209
 edits file out; 1509
     7 terms (     2 edits) in file 1509
Start time of input (s to stop, [file]h to use header  e.g. 400h )
 (Use data, **NOT EDITS**, header number)  ;s
```

The final step is to run autocl using option f to replace gaps with the value 999999.0.

```
autocl
What option (h for help)     ; h
 There are four options for editing a file:
  n - interpolate, with offset (normal)
  s - add slope to bad data and offset
  o - interpolate, no offset
  c - fill with constant, with offset
  f - fill with constant, no offset
  and to get a binary edits file (1's where there are edits
  and 0's elsewhere) use any option with an input file number
  of zero

 What option (h for help)     ; f
 Input file number and length (-1 for all) ; -409
-1
 Output file number                  ; -609
 Edits file number                  ; 1509
 Fill constant                  ; 999999.0
```

We can print out an ASCII version of our finished file using putdat,

```
putdat -609 > clean.filtered.decimated.txt
```

We should have accumulated the following files

L109.D	Raw data, full sample rate
L209.D	Edit file, full sample rate
L309.D	Cleaned data, full sample rate, offsets removed, interpolation of bad data
R409.D	Lowpass filtered and decimated version of L309.D
L509.D	Edit file resampled to same sample interval as R409.D
R609.D	Final data set, offsets removed, bad data replaced by flag, lowpass filtered and decimated to a lower sampling rate.

In the directory /home/data/**/credit_example/Parkfield there are subdirectories containing data from 5 volumetric dilatometers for the period 1st August to 1st October 2004 . There is also a README file containing a suggested filename convention.

Suggested naming convention

	Donnalee	Frolich	Jack Canyon	Red Hills	Vineyard
raw data	101	102	103	104	105
full sample rate edits	201	202	203	204	205
corrected data (autocl: opt n)	301	302	303	304	305
decimated to 1 hour	401	402	403	404	405
decimated edit file	501	502	503	504	505
final file (autocl: opt f)	601	602	603	604	605

start date: 2004 214 00 00 00
original sample rate: 10 mins

SEED Codes for PBO Laser and Gladwin Tensor Strainmeters

Channel codes.

SEED channel codes are 3 character codes that describe the sample rate and the quantity being measured. For a full description of SEED naming conventions see the SEED manual available at the IRIS web site. The first character describes the sample rate. PBO strainmeter data are collected at 20 Hz, 1Hz , 10 second interval and > 300 second intervals. The corresponding letters are, 20 Hz = B, 1 Hz = L, 100 s = V and >300 s interval = R . The 2nd character describes the signal, e.g., S refers to linear strain. The third character is an orientation code. Liner strain measured at 1Hz would have the SEED channel code LS1. Table 1 lists the signal and orientation codes for channels measured at the laser strainmeters and GTSMs.

Table 1. Strainmeter signal and orientation codes.

Code	Description
Laser Strainmeter	
DV	vacuum pressure
S1	main interferometer
X1	laser optical anchor
X2	laser optical anchor
E1	voltage reference channel 1
K2	box temperature
KI	room temperature
KO	air temperature
KD	ground temperature
DO	atmospheric pressure
U0	pyranometer (sunlight)
R0	rain
E2	voltage reference channel 2
X3	alternate laser optical anchor
X4	alternate laser optical anchor
GTSM	
S1	GTSM gage 1
S2	GTSM gage 2
S3	GTSM gage 3
S4	GTSM gage 4
DD	down hole pore pressure
KD	temperature at pore pressure
DO	atmospheric pressure
RO	rainfall
K1	logger temperature
KD	downhole temperature
EO	solar amps
E1	battery voltage
K2	power box temp
E2	system amps
CA	gage1 calibration
CB	gage2 calibration
CC	gage3 calibration
CD	gage4 calibration

Location codes
Location codes are used to distinguish between more than one measurement of the same quantity. For example the room temperature is measured at both ends of the laser strainmeter. The location codes for the strainmeters are given in Table 2.

Table 2. Strainmeter location codes.

Location Code	Description
Laser Strainmeter	
LI	laser interferometer end
LR	laser retroreflector end
LM	laser midpoint
GTSM	
T0	GTSM strainmeter system
TS	surface measurement at GTSM
TP	measurement at the pore pressure monitor
T[1,2,3,4,5,6]	calibration location codes