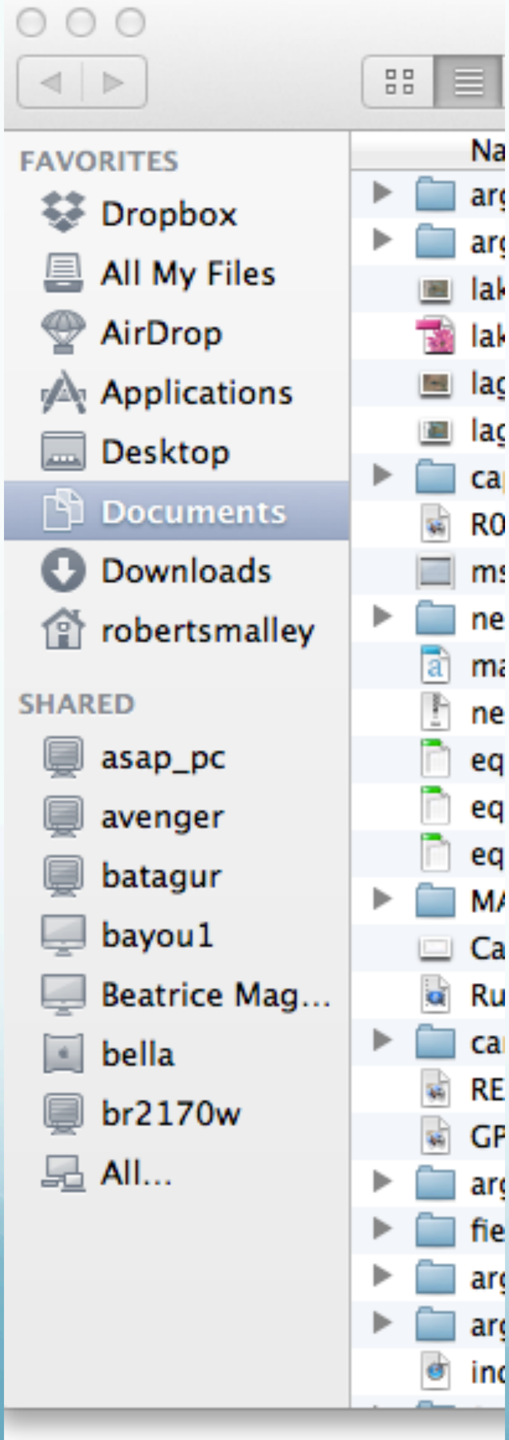
Data Analysis in Geophysics
ESCI 7205

Bob Smalley
Room 103 in 3892 (long building), x-4929

Tu/Th - 13:00-14:30
CERI MAC (or STUDENT) LAB

Lab – 7, 09/17/13



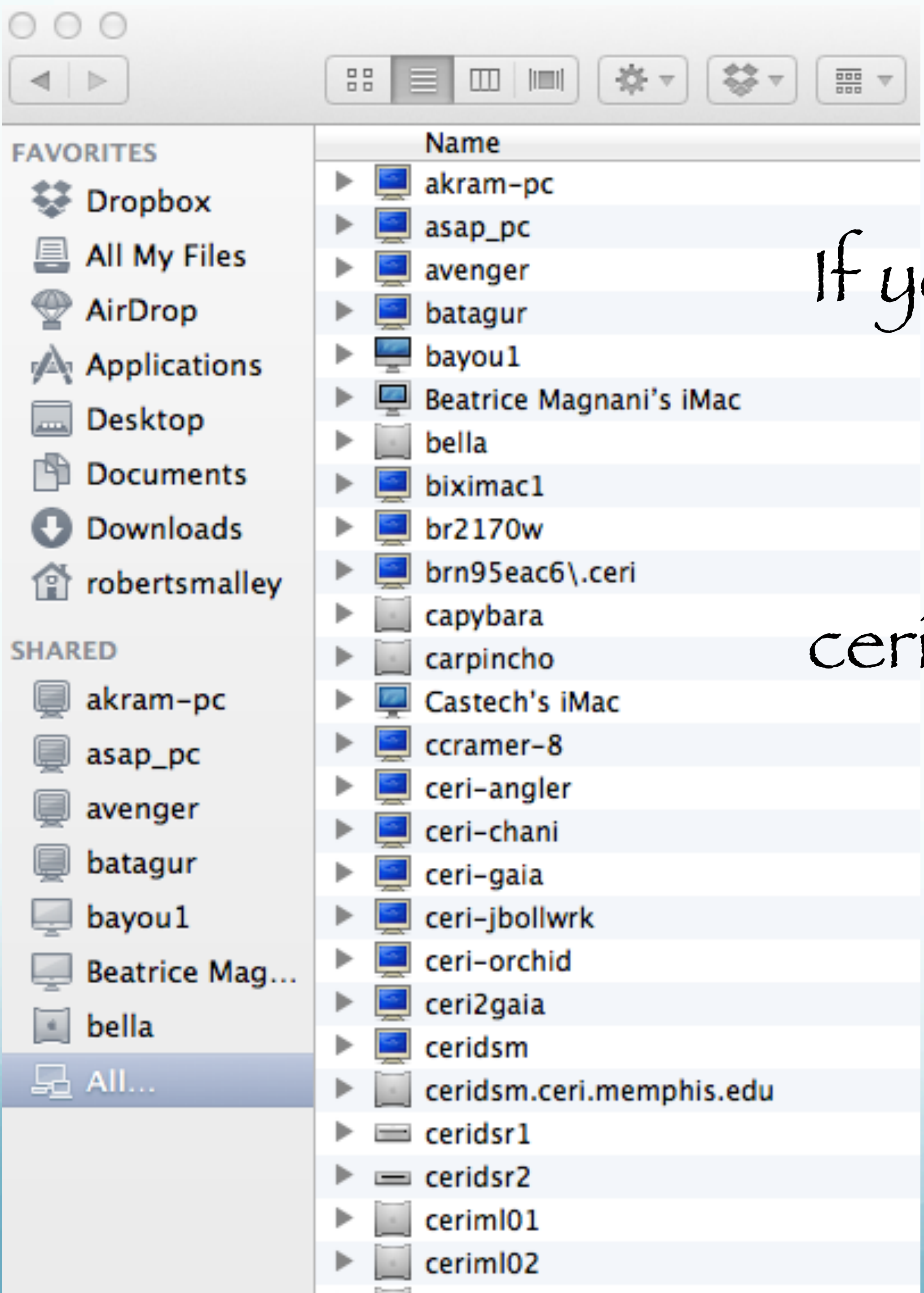
Aside

Sharing files on the machines in the mac lab.

Along the left side of folder windows you will see a list of folders, other computers, etc.

If you see one called "ceridsm" click on it.

If you don't see one called "ceridsm" (as here) click on "All"

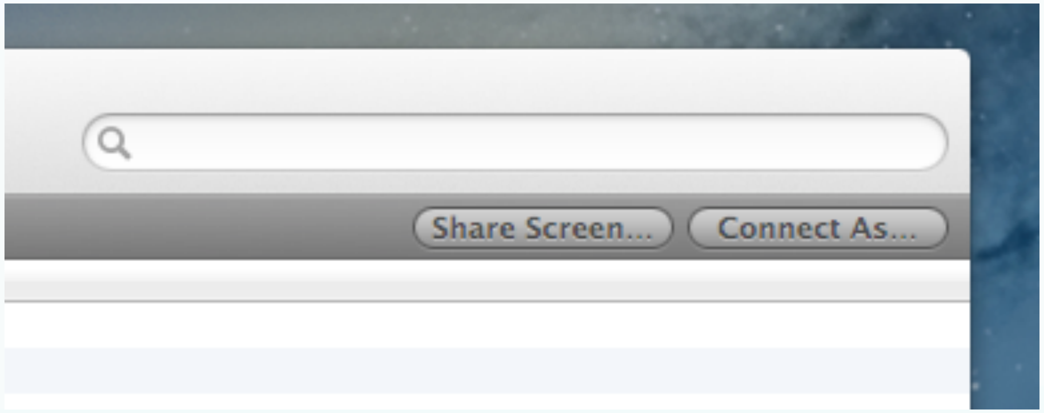


akram-pc
asap_pc
avenger
batagur
bayou1
Beatrice Magnani's iMac
bella
biximac1
br2170w
brn95eac6\.ceri
capybara
carpincho
Castech's iMac
ccramer-8
ceri-angler
ceri-chani
ceri-gaia
ceri-jbollwrk
ceri-orchid
ceri2gaia
ceridsm
ceridsm.ceri.memphis.edu
ceridsr1
ceridsr2
ceriml01
ceriml02

If you clicked on "All" you will see this.

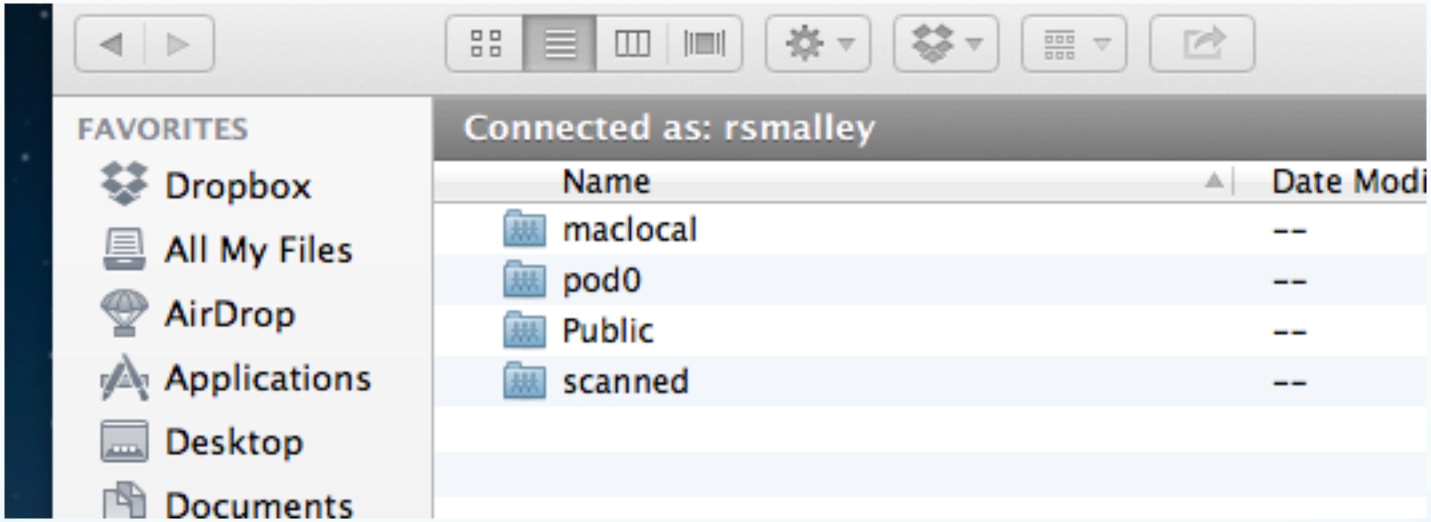
Double click on ceridsm.ceri.memphis.edu

If you get a blank window click on "Connect As"



and log in using the dialog box that comes up.

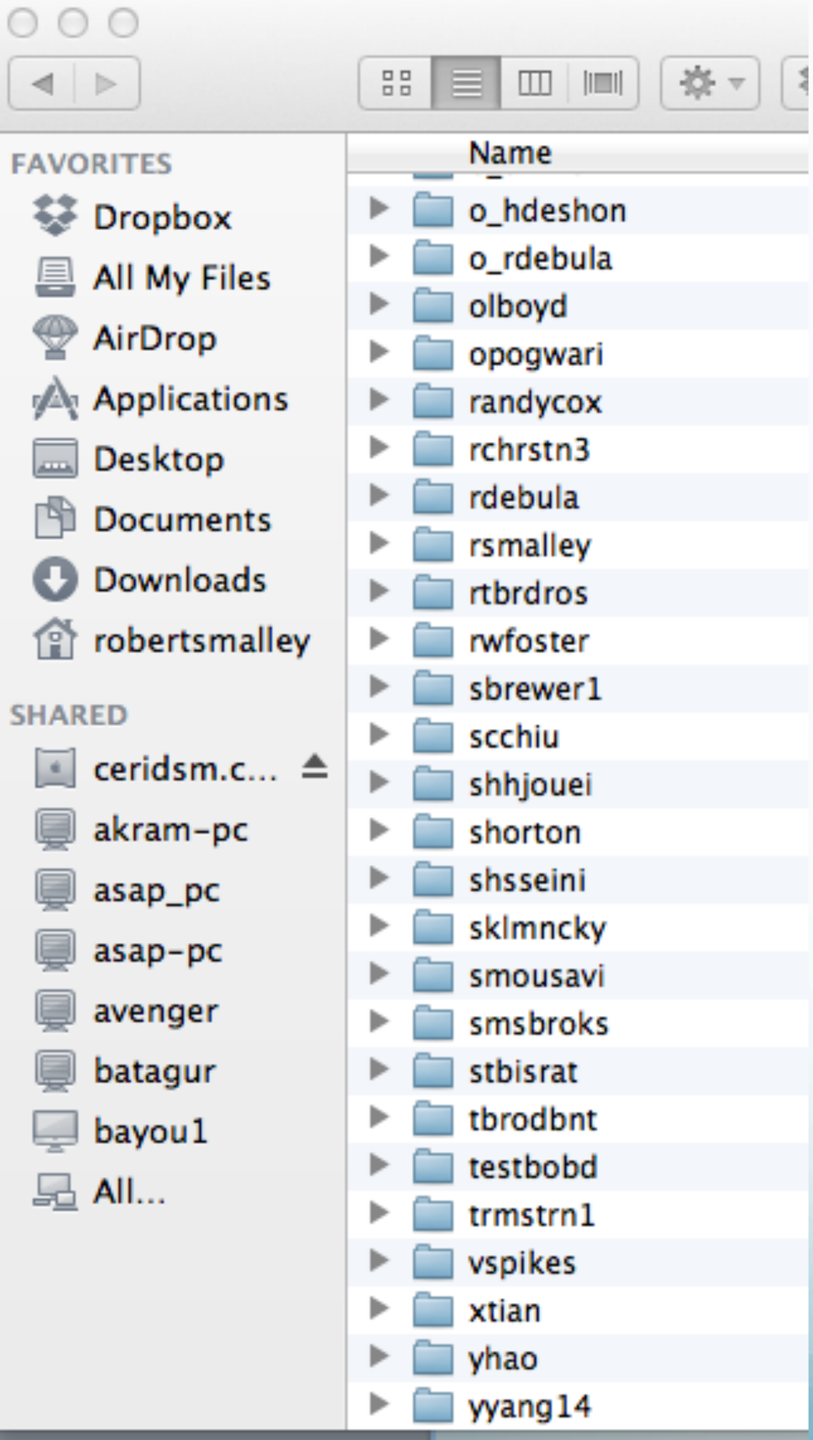
This will bring you to a window that looks like this



Connected as: rsmalley

Name	Date Modi
maclocal	--
pod0	--
Public	--
scanned	--

Double click on "pod0"



o_hdeshon
o_rdebula
olboyd
opogwari
randycox
rchrstn3
rdebula
rsmalley
rtbrdros
rwfoster
sbrewer1
scchiu
shhjouei
shorton
shsseini
sklmncky
smousavi
smsbroks
stbisrat
tbrodbnt
testbobd
trmstrn1
vspikes
xtian
yhao
yyang14

This will open a window with all the home folders for accounts on the Macs in the Mac Lab.

Now navigate around normally (you may not be able to enter folders if the owner has protected them, at which point you need to talk to them if you really need some help).

Returning to SAC

So far we have some files in memory.

If we simply read in another file(s) – the new data will clobber what we have.

If we need to read in more data (say we have processed the data we've read in and now want a spectral ratio of the processed data with the original data) we have to use the "more" option to read in additional data (to not clobber what is there).

```
read more filename
```

In general SAC does commands to <u>all</u> the files in memory.

If the command is starting from scratch (like a read) it clobbers what is already there.

Some commands require certain pairs of files (N and E for example for rotating seismograms).

We have now seen 4 SAC commands (but only used 3).

```
read filename
write still to be determined
plot
qdp on or off
```

Fortunately (in the newer versions of UNIX-SAC) the OS handles the command entry and you can still move around through the command line or "history" with the cursor keys and use regular expressions to build names (wildcards, etc.).

Let's try a few more things.

Here I have to be a little more careful when I specify the file name. I want to read in all 3 components of the seismogram.

(I'm also demonstrating a feature of SAC, if SAC does not understand a command [something you typed], it passes it to the OS for further processing. Based on the output of the "`ls`" command, I don't want SAC to read in the "`.ai`", "`.ps`", or "`.tif`" format files – although if I try to read them SAC will generate an error message that it cannot understand them and just skip/ignore them).

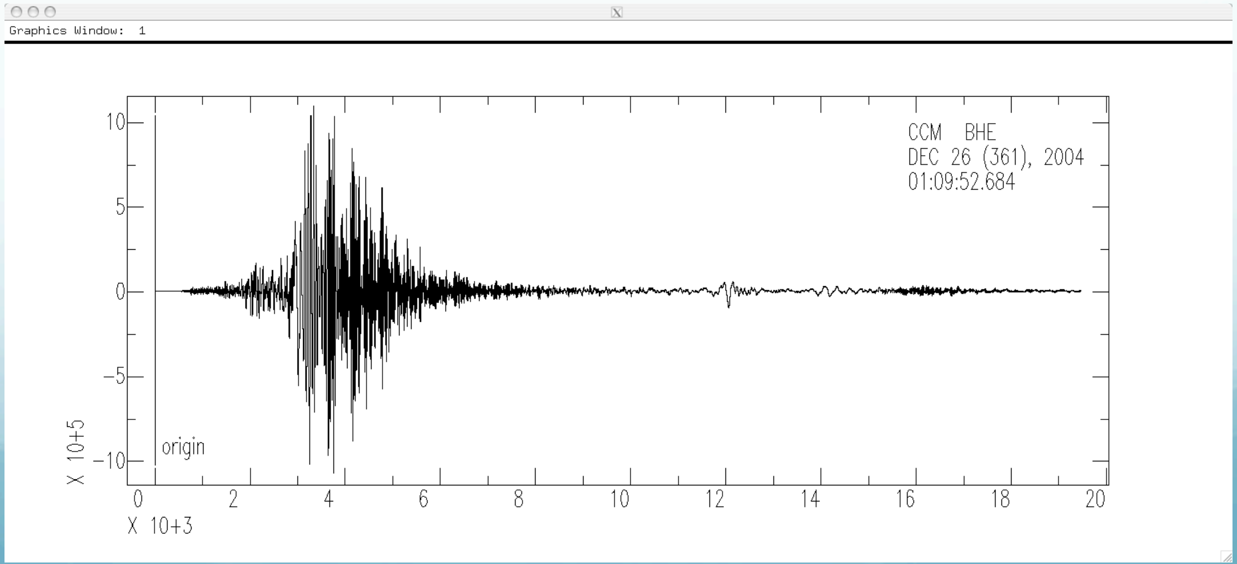
```
SAC> ls *sumatra*bh*
ccm_sumatra_.bhe     ccm_sumatra_.bhn      ccm_sumatra_.bhz
ccm_sumatra_bhz.ai   ccm_sumatra_bhz.ps    ccm_sumatra_bhz.tif
SAC> r *sumatra*bh?
ccm_sumatra_.bhe ccm_sumatra_.bhn ccm_sumatra_.bhz
```

Try the command "plot".

```
SAC> plot
Waiting
```

SAC plots the traces <u>one at a time</u>, in the order they are stored in memory (these happen to be in alphabetical order – **BHE**, **BHN** and **BHZ** – due to the wildcard expansion) (And that we don't have to keep resetting **qdp** to off, it remembers it.) . Each time you enter a <CR> it plots the next trace.

(and says Waiting if there are more traces to display, or the prompt if not).

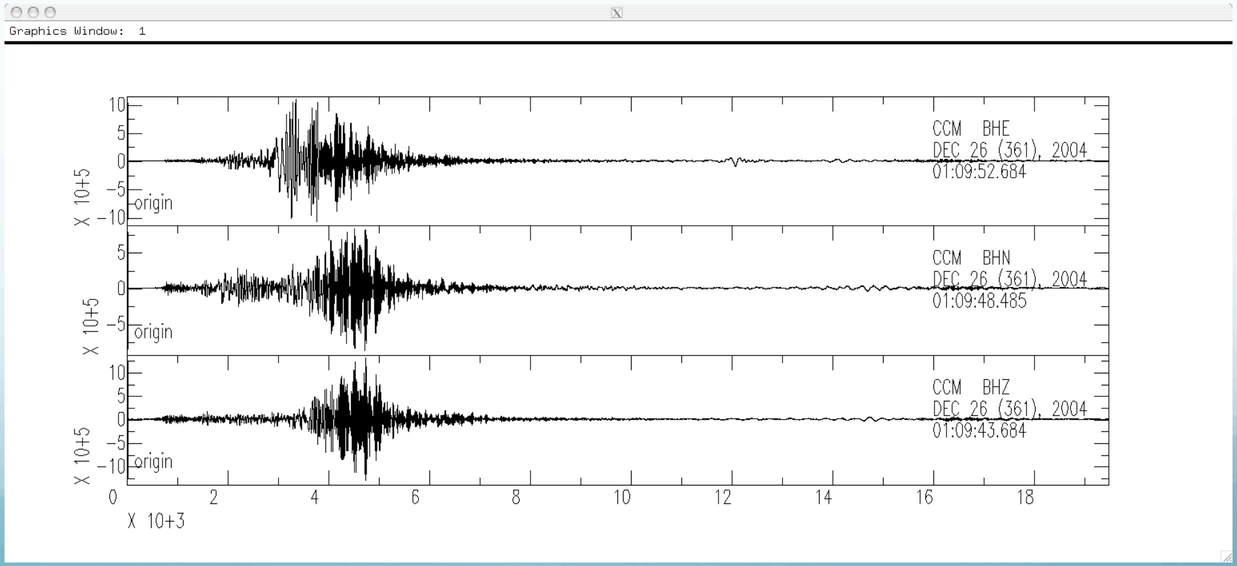


New command – `plot1` ("`p1`").

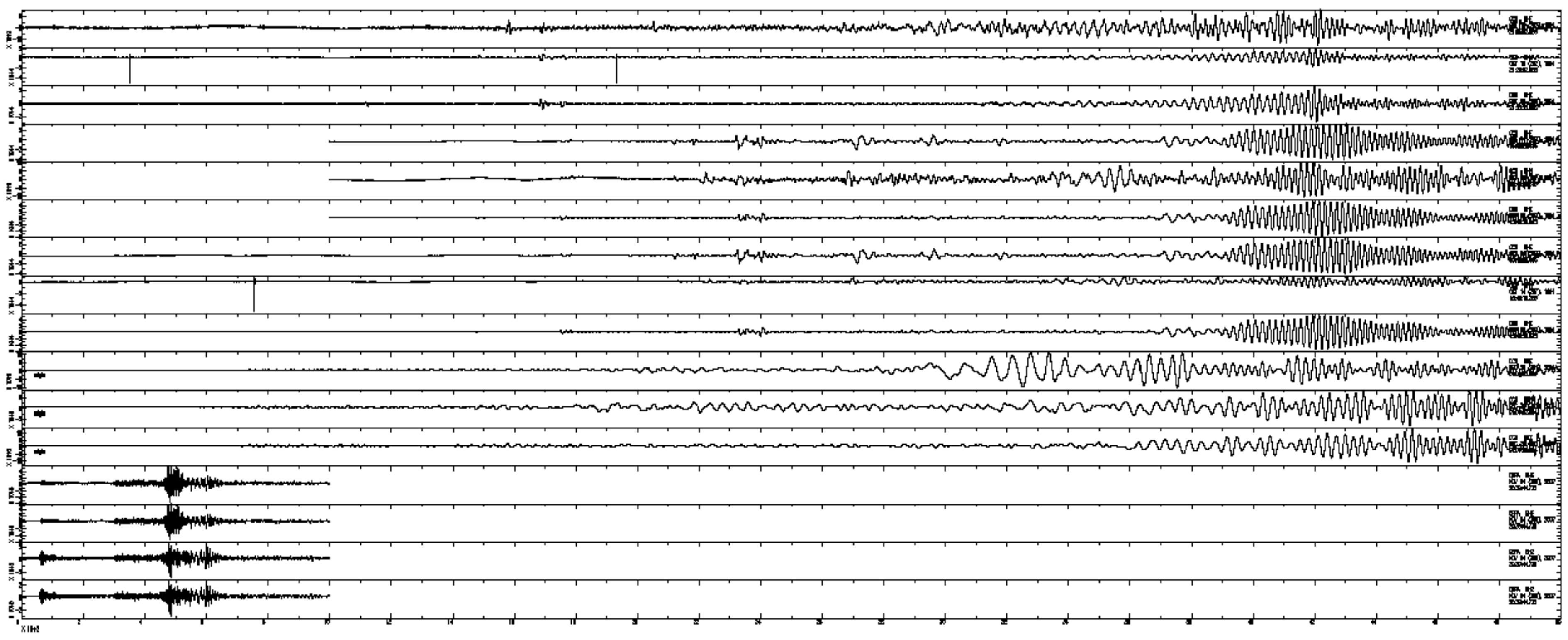
```
SAC> plot1
SAC>
```

This command plots all the data in memory on one plot.

Also notice that the prompt returns so we can enter more commands.



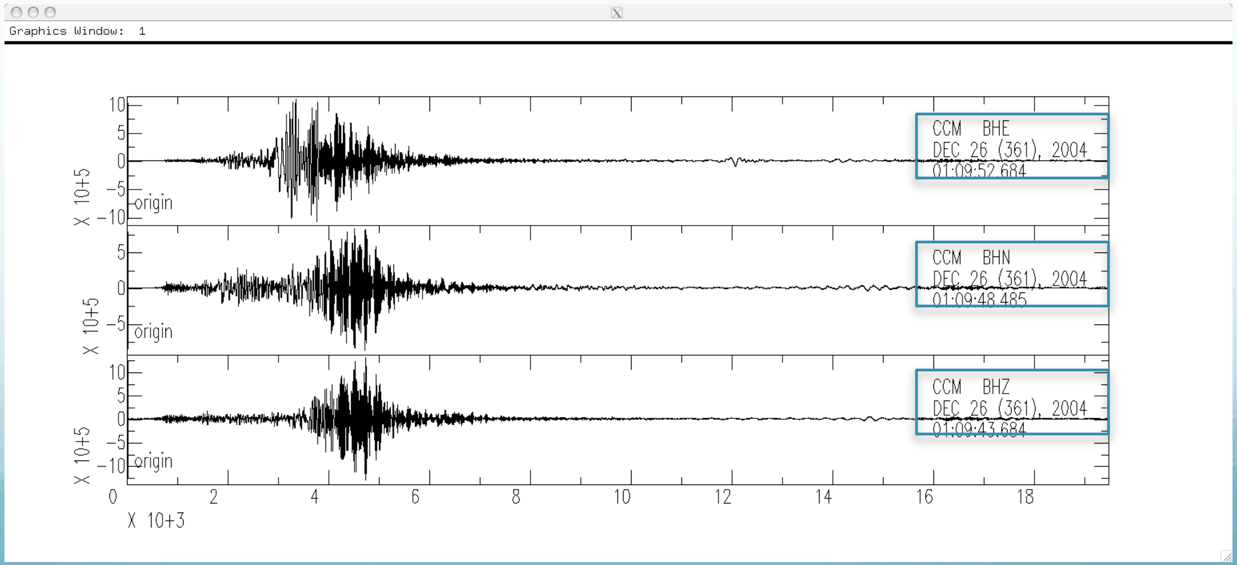
If you have too many traces it plots a mess



Say I want to process these three traces together.

UGLY little detail.

Notice that the three traces do not start at the same time (and we will see that they are not the same lengths, either).

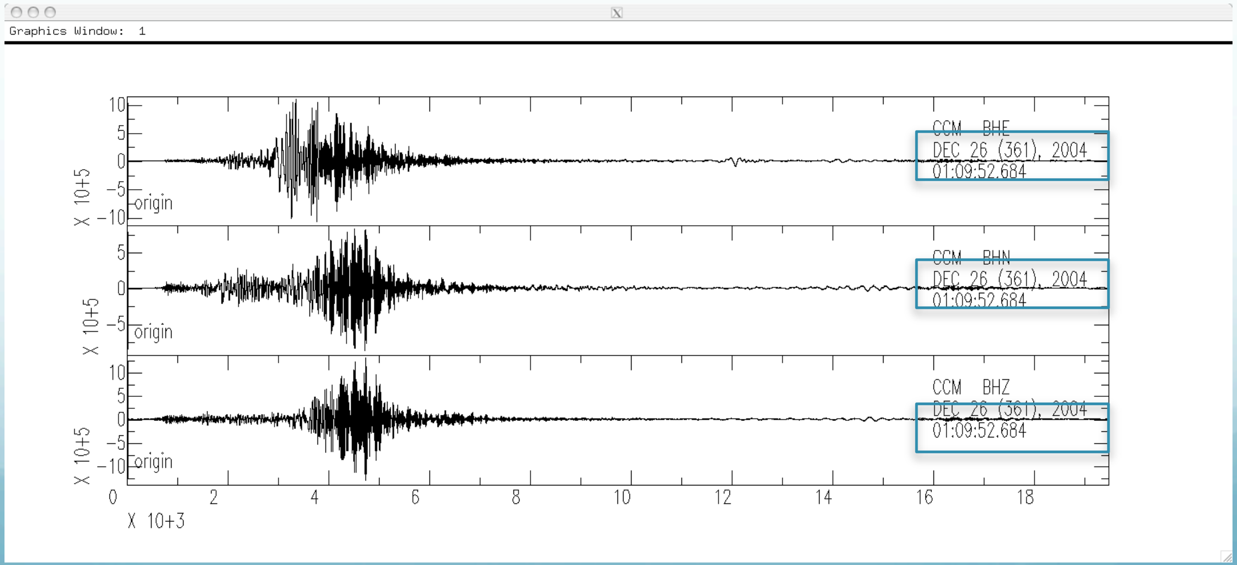


We can fix the time alignment using

synchronize ("**synch**"): which modifies (the headers of) the files in memory so that they all have the same reference time.

(Does not otherwise modify the data. If I need to combine two traces rotate them for instance, and they are not the same length, SAC will complain and not do it.)

```
SAC> synch
SAC> plot1
```



How do we find out what do we have in memory?

What metadata is available about the seismic data?

(metadata is data about data – examples are the name of the seismic station and component, the sampling rate, etc. This information is stored in the SAC header.)

`listhdr (lh)`: lists the headers of the files in memory.

```
SAC> listhdr
  FILE: ccm_sumatra_.bhe - 1
 ----------------------

       NPTS = 389396
          B = 0.000000e+00
          E = 1.946975e+04
     IFTYPE = TIME SERIES FILE
      LEVEN = TRUE
      DELTA = 5.000000e-02
       IDEP = UNKNOWN
     DEPMIN = -1.073057e+06
     DEPMAX = 1.091875e+06
     DEPMEN = 8.429739e+02
    OMARKER = 7.315 (origin )
     KZDATE = DEC 26 (361), 2004
     KZTIME = 01:09:52.684
     IZTYPE = BEGIN TIME
      KSTNM = CCM
      CMPAZ = 9.000000e+01
     CMPINC = 9.000000e+01
       STLO = -9.124470e+01
       DIST = 8.818225e+03
         AZ = 1.854116e+02
        BAZ = 2.013326e+02
     LOVROK = TRUE
      NVHDR = 6
     LPSPOL = TRUE
     LCALDA = TRUE
     KCMPNM = BHE
     KNETWK = US
```

All sorts of good stuff. Look in SAC documentation for full details.

Obvious/important –

```
File name - FILE
Number points – NPTS
Beginning time offset - B
Sampling rate – DELTA
Start date – KZDATE
Start time –KZTIME
Station – KSTN
Orientation – CMPAZ
```

May have info about stn location, event locn, ...

Says `Waiting` when page is full (may not be complete header listing).

`<CR>` to continue till run out of stuff to display.

(there is no way to "break" out of the commands (e.g.: `plot, listhdr`, ...) that do something for each file. You have to `<CR>` till it finishes, or ^C out and start over.)

Lining files up in time

Before synch

```
FILE: ccm_sumatra_.bhe – 1
        NPTS = 389396
           B = 0.000000e+00
      KZDATE = DEC 26 (361), 2004
      KZTIME = 01:09:52.684
FILE: ccm_sumatra_.bhn – 2
        NPTS = 389328
           B = 0.000000e+00
      KZDATE = DEC 26 (361), 2004
      KZTIME = 01:09:48.485
FILE: ccm_sumatra_.bhz – 3
        NPTS = 389600
           B = 0.000000e+00
      KZDATE = DEC 26 (361), 2004
      KZTIME = 01:09:43.684
```

After synch

```
           B = 0.000000e+00
      KZDATE = DEC 26 (361), 2004
      KZTIME = 01:09:52.684

           B = -4.199000e+00
      KZDATE = DEC 26 (361), 2004
      KZTIME = 01:09:52.684

           B = -9.000000e+00
      KZDATE = DEC 26 (361), 2004
      KZTIME = 01:09:52.684
```

To make them all start at the same time and be the same length

- Read them in, then synch (aligns them to the same relative time – the time of the file that starts last [all the **bs** are negative], but still different lengths.)

- Write them out (this clobbers the original file on the disk unless you rename them),

- Set up a `cut` (reads from a start time [which we just aligned above] to an end time with respect to the reference time, not the start time or number of samples)

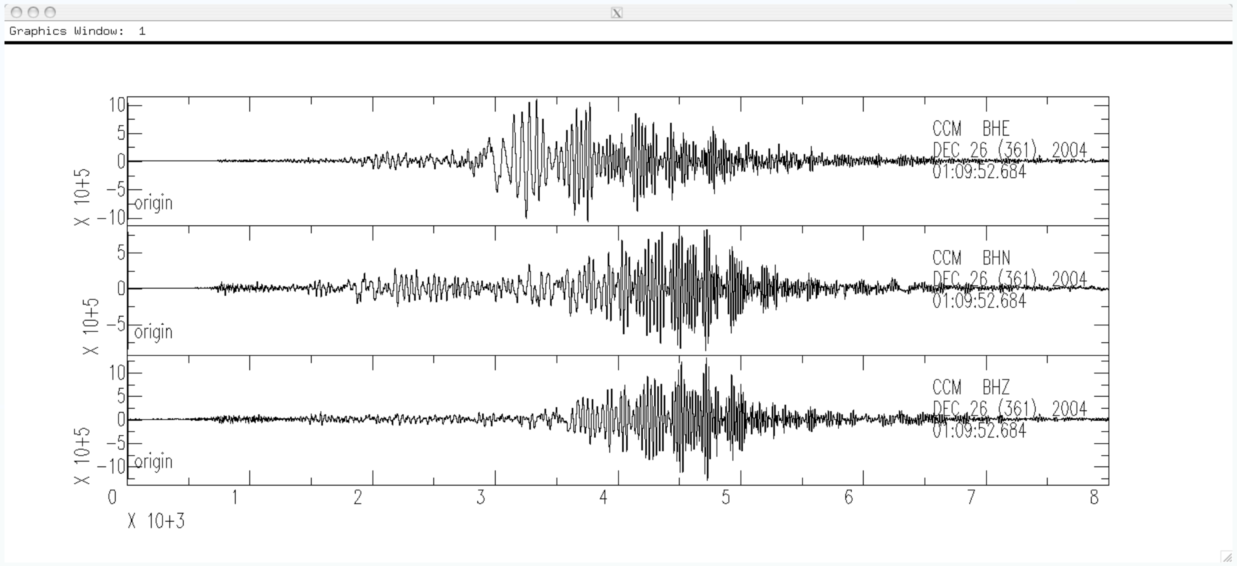
To make them all start at the same time and be the same length

- Then read again (under control of the cut, everything in memory will be the same size, [unless one was shorter, but that will not happen here]).

- Write out again (if you want to save them, clobbering what was there).

(I don't know how to do this "in-place", you need the write and re-read since the SAC commands do not modify, except by writing, files on disk, data in memory.)

```
SAC> w over
SAC> cut 0 8000
SAC> r
ccm_sumatra_.bhe ccm_sumatra_.bhn ccm_sumatra_.bhz
SAC> p1
```



And they all have the following in their header

```
  NPTS = 160001
     B = 0.000000e+00
     E = 8.000000e+03
KZDATE = DEC 26 (361), 2004
KZTIME = 01:09:52.684
```

`cut:` defines how much of a data file is to be read.

You need to re-read the data after specifying a `cut`. (specifying the cut does not effect data in memory, or the files on disk)

You can also specify the `cut` with respect to times stored in the header (`p` wave arrival time for example) `5 s` before to `30 s` after `t1` pick

```
SAC> cut t1 -5 30
SAC> r
```

Commands to see/change header values

__`listhdr (lh)`__: list the header contents.

__`readhdr (rh)`__ and __`writehdr (wh)`__: read and write headers without the data.

__`chnhdr (ch)`__: change header values.

__`copyhdr`__ : copy header values from one file to the others in memory.

Example: Say the header does not have the location of the event (if you do an "`lh`" and there is no **EVLA** – Event Latitude, or **EVLO**, - Event Longitude, reported). We can add this information to the headers of all files using **chnhdr**.

```
SAC> chnhdr EVLA = 4.079930e+01 # event latitude
SAC> chnhdr EVLO = 3.100330e+01 # event longitude
SAC> lh
. . .
EVLA = 4.079930e+01
EVLO = 3.100330e+01
. . .
SAC>
SAC> wh
```

We overwrite only the header because no changes were made to the seismic data (time series).

When you download **preprocessed** seismic data from the IRIS-DMC associated with an earthquake, it will now have the earthquake location, origin time, delta, azimuth, etc. in the header.

If you download data in some arbitrary time window (even if it has a big earthquake in it) it will not come with information about anything in particular within that time window (may be multiple events or none!).

You will have to put in the event information (if it has a event location, SAC now computes the delta and az/baz and stores it in the header for you).

Graphics Action Module

<u>REVIEW</u>

`plot (p)`: plots each signal in memory on a separate plot.

`plot1 (p1)`: plots a set of signals on a single plot with a common x axis and separate y axes.

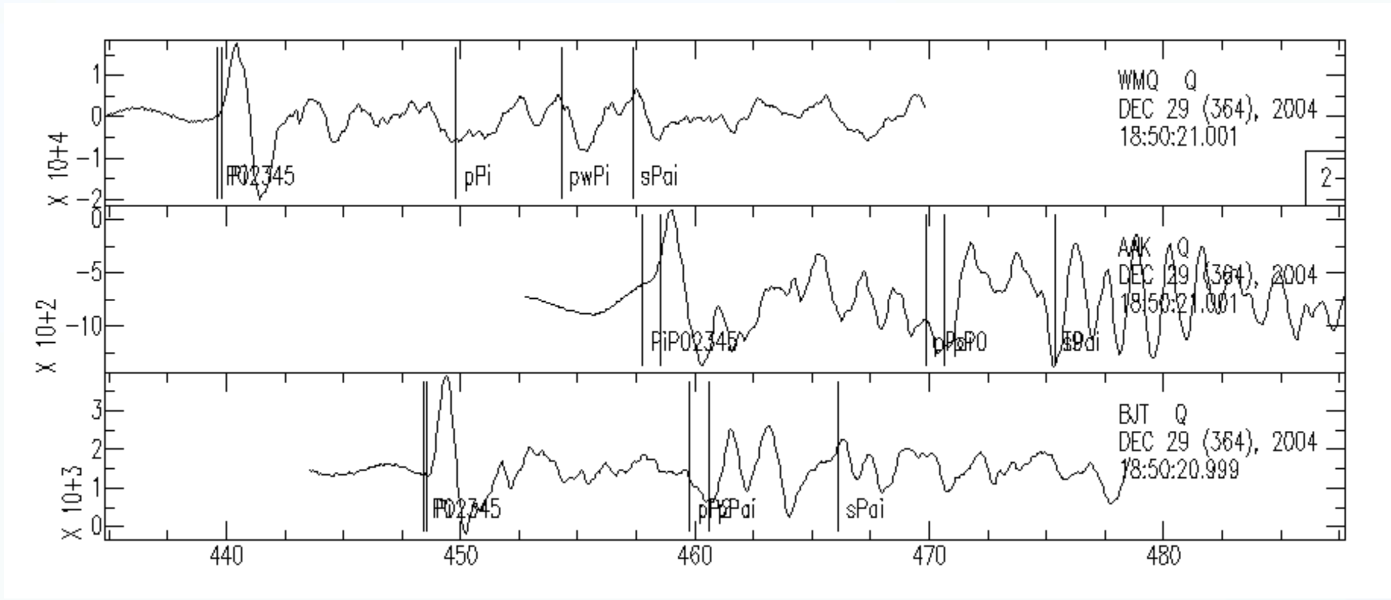
<u>NEW</u>

`plot2 (p2)`: plots set of signals on a single plot with common x and y axes (i.e. an overlay plot).

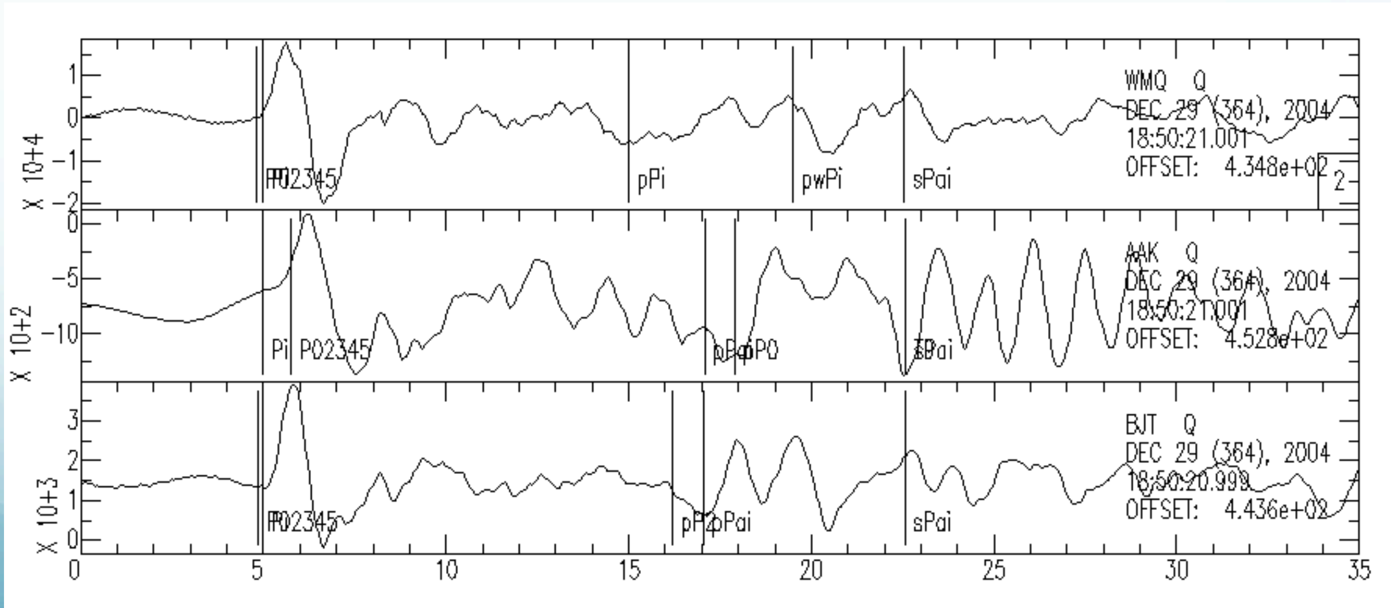
Can plot each file relative to its begin time.

(default is absolute, so the traces are shifted by actual time)

```
SAC> p1
```

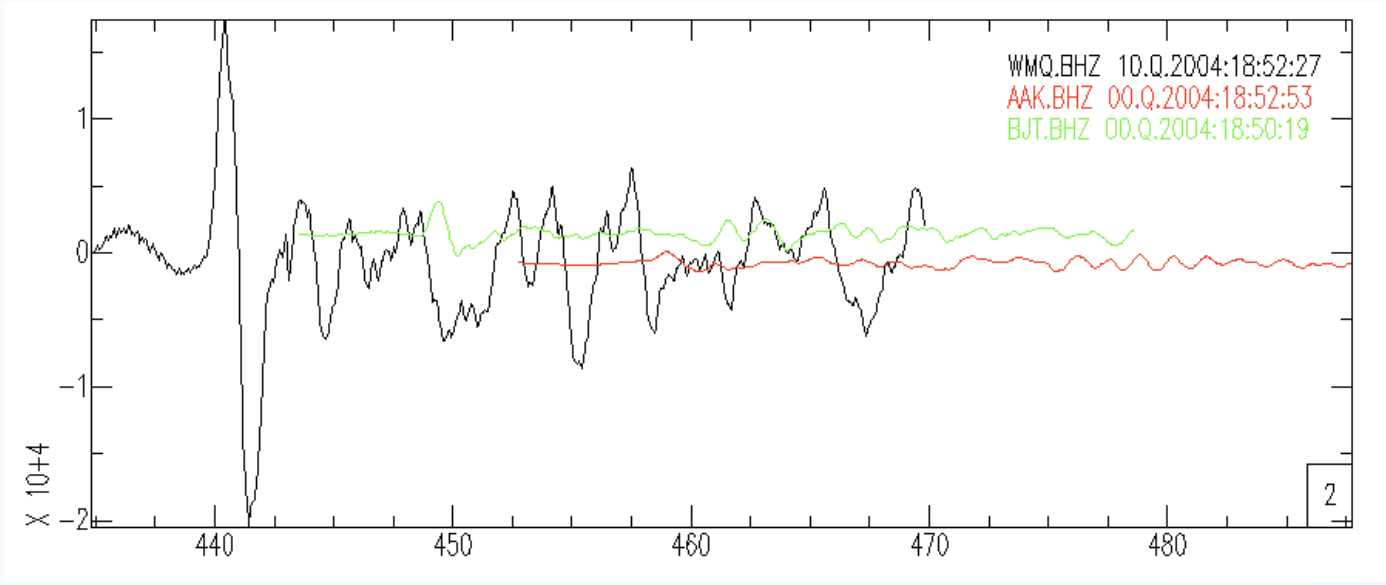


```
SAC> p1 rel
```

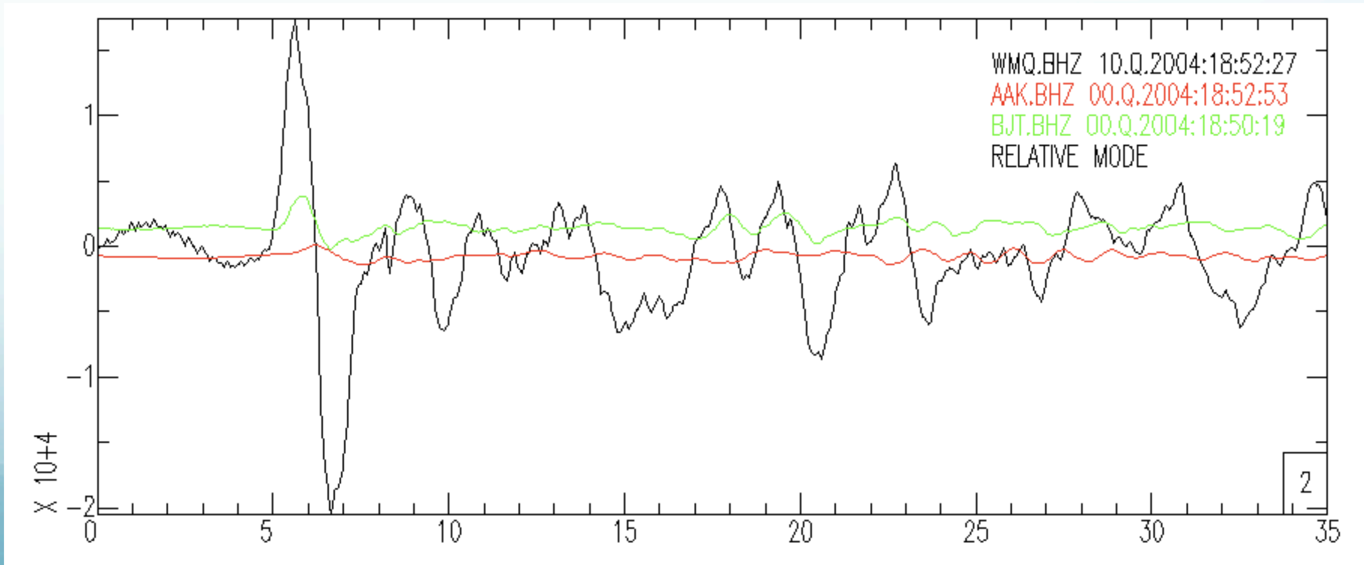


Can color traces (this is an addition since the TEK401X days – when it was green or nothing).

```
SAC> color on increment on

SAC> p2
```



```
Sac> p2 rel
```



Graphics

There are three graphics devices currently supported.

SGF (SAC Graphics File) is a file with graphics information that can get converted into postscript, etc.

X-WINDOWS is a general windowing system running on most high-resolution, bit-mapped graphics workstations
(and where our plots have been showing up)

(SUNWINDOW, is a windowing system that was available on the Sun in SunOS 4.X. Listed for completeness)

X-windows or X

X is widely used on UNIX graphics workstations and offers one of the best frameworks (UNIX opinion, X follows the UNIX philosophy so it is powerful and difficult) for developing portable window-based applications.

It should be the default graphics device when you start SAC.

Can be turned on using `begin device: (bd)`.

```
sac> bd x
```

SGF demonstrates the power (or kluginess) of SAC and UNIX.

Rather than burden the SAC program with producing graphics for a large number of possible devices
(postscript did not even exist when SAC was written)

have SAC write out a file (that is probably just the TEK401X commands) that some other programs read and translate into the appropriate format for display on any particular device.

SAC Graphics Files contain all the information needed to generate a single plot.

Each plot is stored in a separate file.

The file names are of the form "**`fnnn.sgf`**" where **`nnn`** is the plot number, beginning with `001`, resetting each time you start SAC

(so if you have some preexisting files that you have not renamed, they will get clobbered – you have been forewarned).

SGF output is turned on with the command <u>`begindevice`</u>`: (bd)`

```
sac>  bd sgf
```

Graphics output will now go to the **`sgf file`**.

You will not see it on the screen (`X display`).

There is no “save my figure” from the `X-display`

(this is UNIX and without an inordinate amount of work to bring out its power, **X** is very basic).

So if you want to make a figure for printing or sending anywhere but the `X-display`

(if it is a complicated figure you may have to first make it and look at it on the `X-display` - then).

Turn on the `sgf` device and (RE)make the with the output now going to the file.

Or if you are on a Mac or a PC you could use the screen capture function and then paste it into another file.

(there is no screen capture on the Sun, it is "pure" UNIX.)

To create a <u>postscript</u> file, you would turn on the **sgf** device, create your plot, and then run a conversion program called **sgf2ps** or **sgftops**.

```
SAC> qdp off
SAC> read ccm*_.bhz
ccm_india_.bhz ccm_solomon_.bhz ccm_sumatra_.bhz
SAC> bd sgf
SAC> p1
SAC> sgftops f001.sgf sac_example.ps
SAC> bd x
```

Unfortunately trying to display the figure using the **gs** command from within SAC falls over since **gs** also is a SAC command (plot greyscale image of data in memory). Need final "**bd x**" to send graphics back to screen.

Data Format and Header

Each signal or seismogram is stored in a separate binary or alphanumeric data file.

SAC can read data in a variety of formats:

- SAC Binary Format (most common)
- SAC ASCII Format (big)
- CSS format
- SDD format
- ASCII formats

Each data file contains a header (we have already seen a bit about the header) that describes the contents of that file.

Some header fields

Time

The SAC header contains a reference or zero time, stored as six integers (NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, NZMSEC), but normally printed in an equivalent alphanumeric format (KZDATE and KZTIME), the offset in seconds between the reference and the data start time (B) and the number of seconds to the data end time (E).

```
     B = 0.000000e+00
     E = 3.600990e+03
KZDATE = APR 06 (097), 2008
KZTIME = 02:59:59.320
```

Event and Station Info

SAC header can store station and event info

```
  KSTNM = WMQ
   STLA = 4.382110e+01
   STLO = 8.769500e+01
   STEL = 8.970000e+02
   EVLA = 3.086000e+00
   EVLO = 9.584800e+01
   EVDP = 3.040000e+01
OMARKER = 0
```

Plus metadata info about the time (**`gmt`** for example).

If the event and station information are in the header, SAC automatically calculates and stores

distance (in km)
azimuth (in degrees)
backazimuth (in degrees)
and great circle arc length (in degrees)

in the header
(SAC2000 and later, earlier versions did not do this).

```
DIST = 4.583862e+03
    AZ = 3.510350e+02
   BAZ = 1.675856e+02
 GCARC = 4.120298e+01
```

Phase Info

SAC can be used to pick and store phase information in header variables `A & T0-T9`
(although this is another place where it shows its age and is quite clumsy).

`omarker` is reserved to for the origin time.

All pick and origin times are stored in seconds from the reference time of the file.

`omarker` (origin time) is oftentimes set (incorrectly) to zero. If `amarker` and `t0marker` are not set, they will not show in a `lh`.

```
OMARKER = 0
AMARKER
T0MARKER
T1MARKER = 462.7                  (P)
T2MARKER = 834.76                 (S)
T4MARKER = 472.5                  (pP)
T6MARKER = 478                    (sP)
```

You can also store what you think the time is.

SAC data format (gory little detail)

The standard data format for SAC is binary. A binary SAC file contains a fixed-length header composed of a combination of ascii, integer and floating point data, which describe a variable length of subsequent data in floating point binary (or ascii, longer so not so popular, but it does exist) .

For the seismic data this means a single data component recorded at a single seismic station. SAC does not currently work on multiplexed data.

There is an issue with the SAC (actually all numerical, except 8 bit) binary data.

SAC data is stored according to the "big-endian" ($#*&%*) byte order (high-order byte of the number is stored in memory at the lowest address).

In many processors, however, it is stored according to the "little-endian" byte order.

Newest version of SAC is bi-endian and handles it automatically.

When given a choice between two equivalent ways to do things (store stuff in memory in this case) it will get done both ways!

How to store character data in a machine that addresses bytes (8 bits of data)

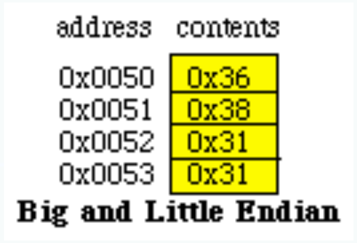
store the number 6811 as 4 ASCII characters in memory (this would be as they come from the keyboard before converting into a base two number)

It would get stored as the hexadecimal string (specified by the x)

```
            0x36383131

      value    36  3831  31

   address       1   2    3    4 …

  address    …    4     3    2    1
```

store the number 6811 as 4 ASCII characters in memory

Can also do it with memory visualized vertically (and get two ways to number/address elements).



address	contents
0x0050	0x36
0x0051	0x38
0x0052	0x31
0x0053	0x31

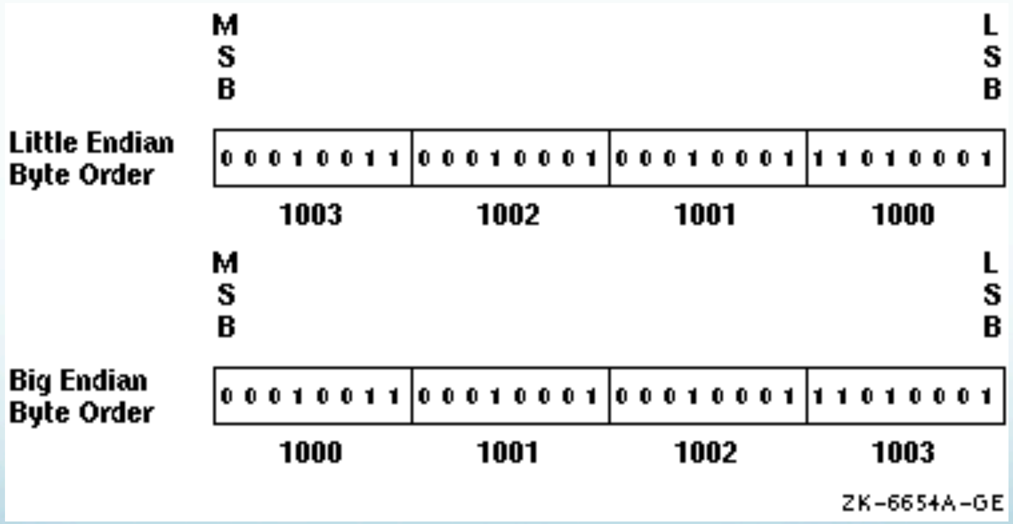
Big and Little Endian

(one can go from top or bottom – here go from top) [vertical is less obvious since in memory the bits are linearly connected and there is ambiguity in how the bits connect to words in the vertical view]

It gets more interesting when we store 16 (4 hex digits) or 32 (8 hex digits) bit numbers

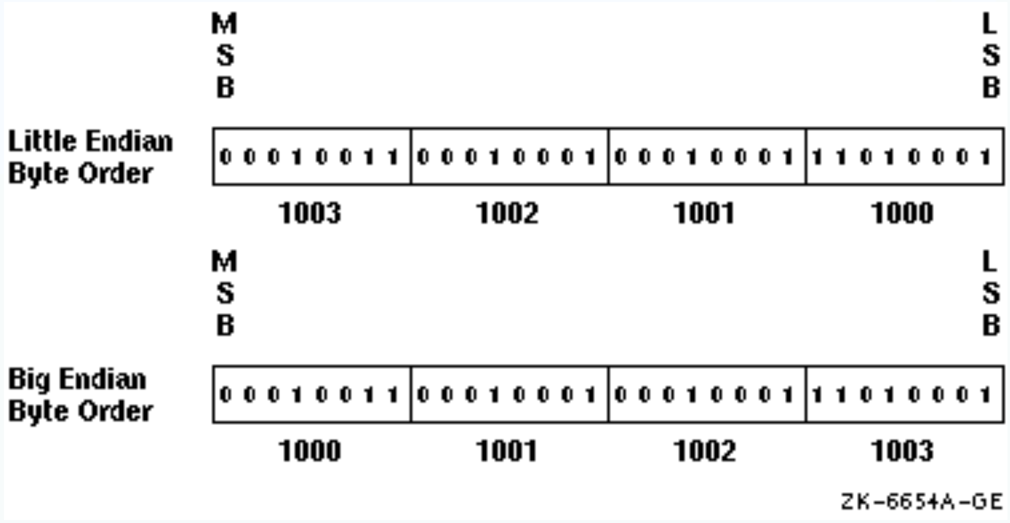
$1311_{16} = 4881_{10}$ 11d116 = 4561_{10}

$131111d1_{16} = 319885777_{10}$



$1311_{16} = 4881_{10}$ $11d116 = 4561_{10}$

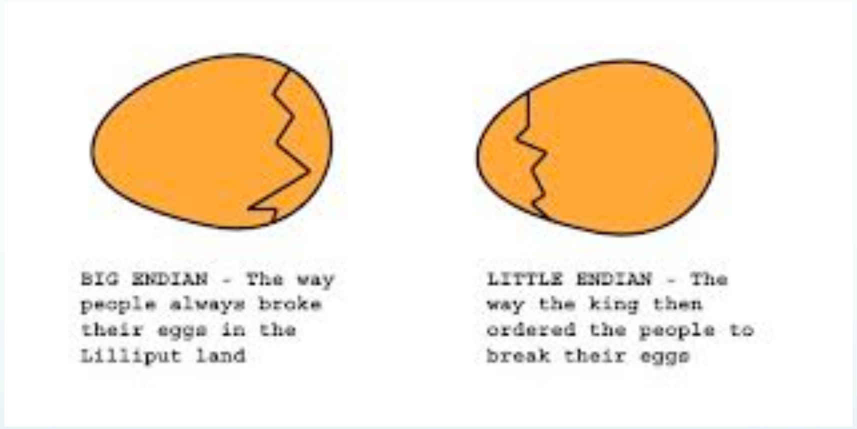
$131111d1_{16} = 319885777_{10}$



With both we specify the number by the lowest of the addresses (1000) and get the data at 1000 and 1001 for 16 bit, or 1000, 1001, 1002 and 1003 in that order for 32 bit numbers.

You have to know which one you are getting.

Names come from Jonathan Swift's "Gulliver's Travels" and the 100 year war between Lilliput and the rival kingdom of Blefuscu over which end of a soft-boiled egg to crack.


BIG ENDIAN - The way people always broke their eggs in the Lilliput land
LITTLE ENDIAN - The way the king then ordered the people to break their eggs

Trivial difference, but need to make one a standard. Like Lilliput and Blefuscu the computer people did it both ways - and it became religious.

An error will occur when data is stored in Big Endian by one computer and read in Little Endian format on another.

Moving unformatted data files between big endian and little endian computers requires that the data be converted – called "byte swapping".

Tutorial

See

http://geophysics.eas.gatech.edu/classes/SAC/

http://www.iris.edu/software/sac/manual/tutorial.html

http://moodle.glg.muohio.edu/mikeb/content/users/brudzimr/SAC/

Start SAC

Is interactive and command driven.

`funcgen`: generate various functions in memory.

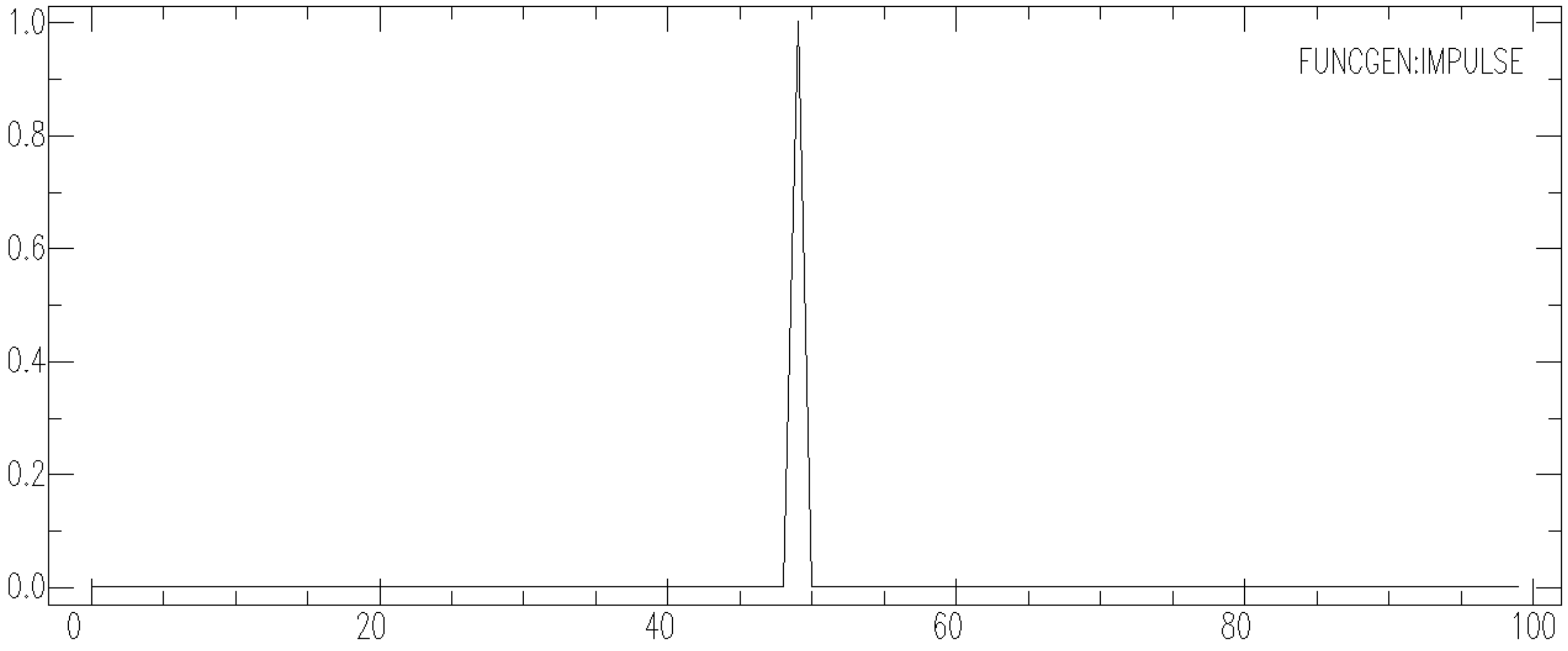
```
STEP
BOXCAR
TRIANGLE
SINE {v1 v2}
LINE {v1 v2}
IMPULSE
QUADRATIC {v1 v2 v3}
CUBIC {v1 v2 v3 v4}
~~SEISMOGRAM~~
~~DATAGEN~~
RANDOM {v1 v2}
IMPSTRIN  {n1 n2 ... nN}
```

It is VERY useful for testing the other commands on known functions.

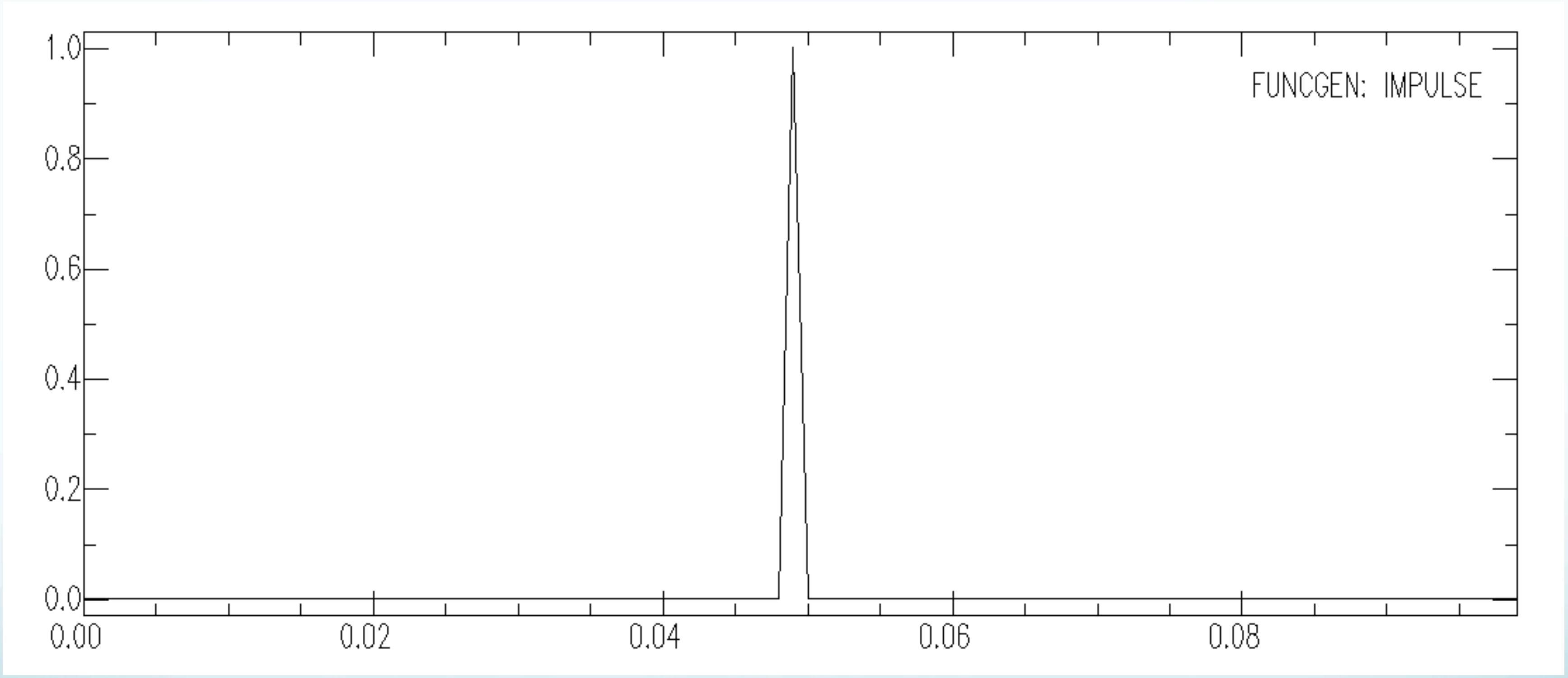
(`seismogram` is obsolete, replaced by `datagen`, but `datagen` reports it is missing the directory with the sample files. Typical!)

Start with some simple commands to generate seismic data

```
Roberts-MacBook-Pro:-bash:matlab:164 $ sac
 SEISMIC ANALYSIS CODE [06/07/2010 (Version 101.4)]
 Copyright 1995 Regents of the University of California
SAC> funcgen
SAC> p
```



```
sac> funcgen impulse delta 0.01 npts 100
sac> p
```



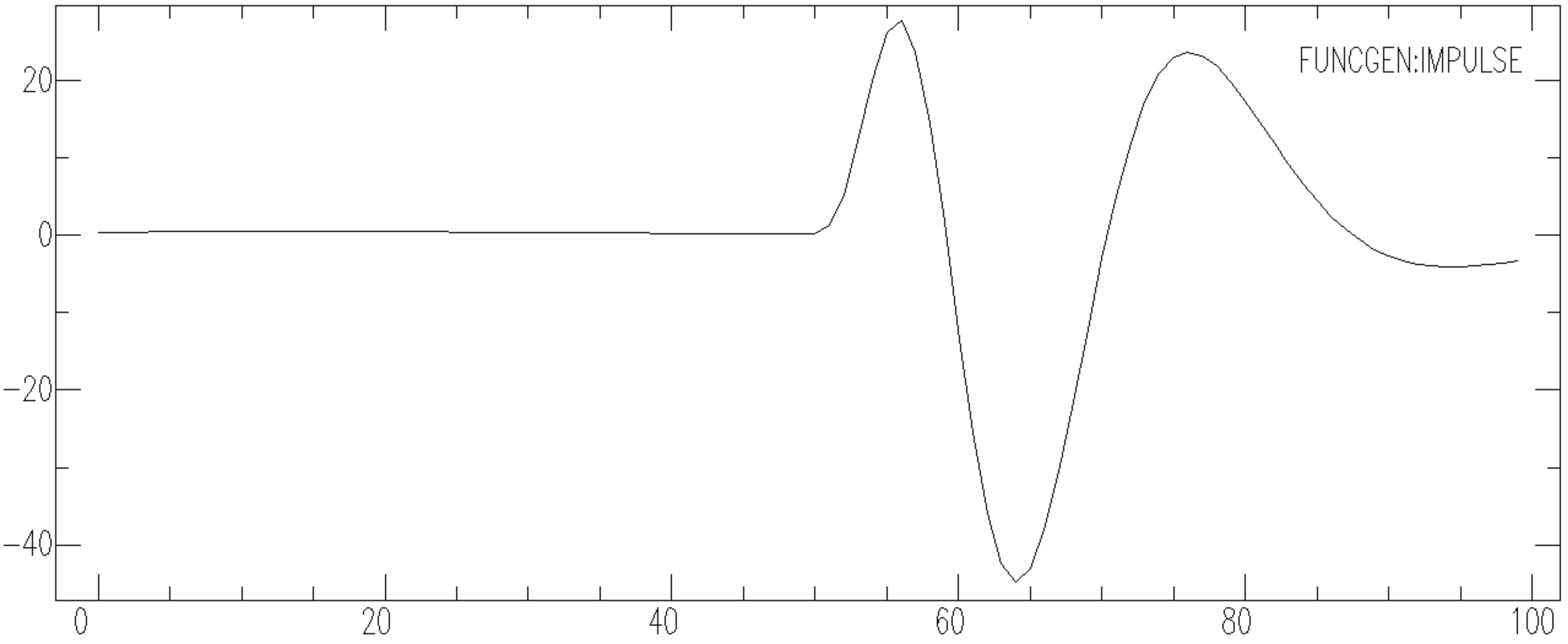
Unary Operations Module

The commands in this module perform some arithmetic operation on each data point of the signals in memory.

```
add                 abs
sub                 log,log10
mul                 exp,exp10
div                 int
sqr                 dif
sqrt
```

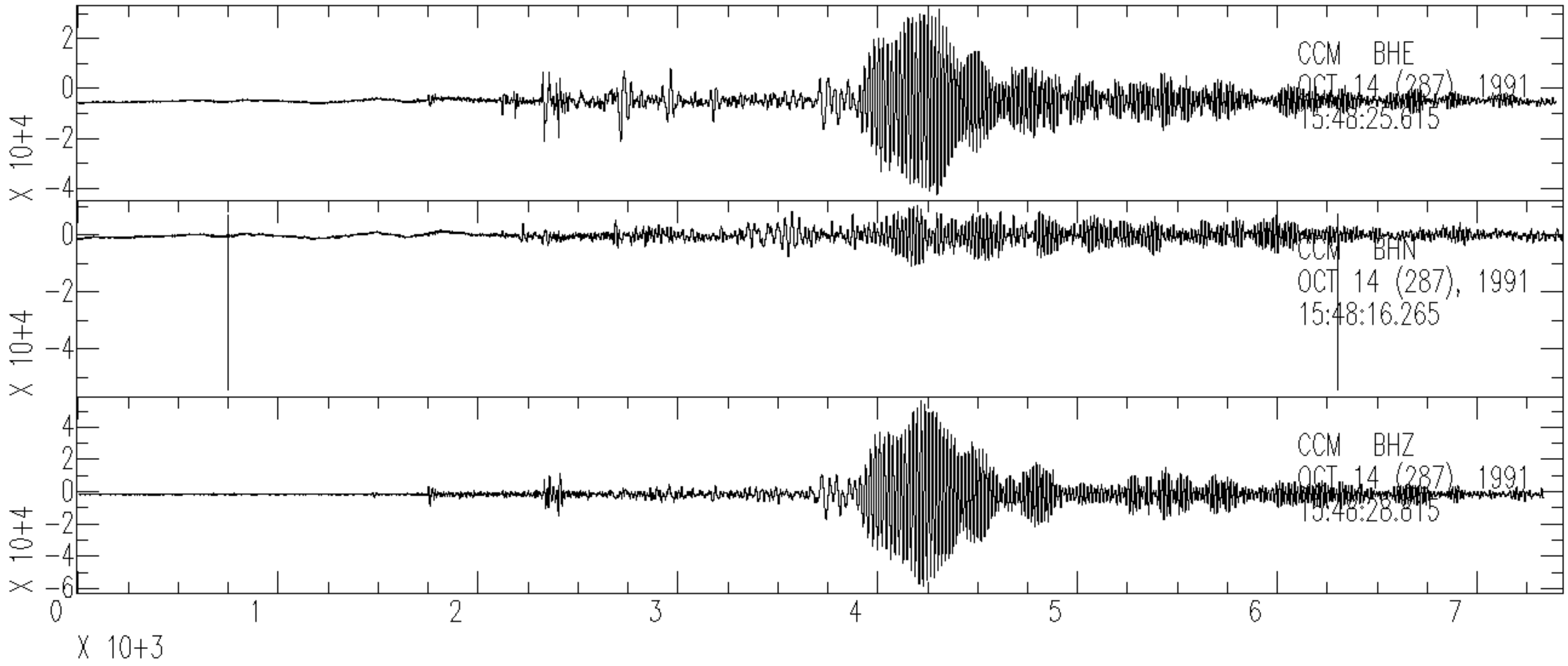
Start with some simple commands to generate seismic data

```
SAC> transfer to DWWSSN
Station (-12345   ), Channel (-12345   )
   Waveform multiplied by 1.000000 after deconvolution.

SAC> p
```



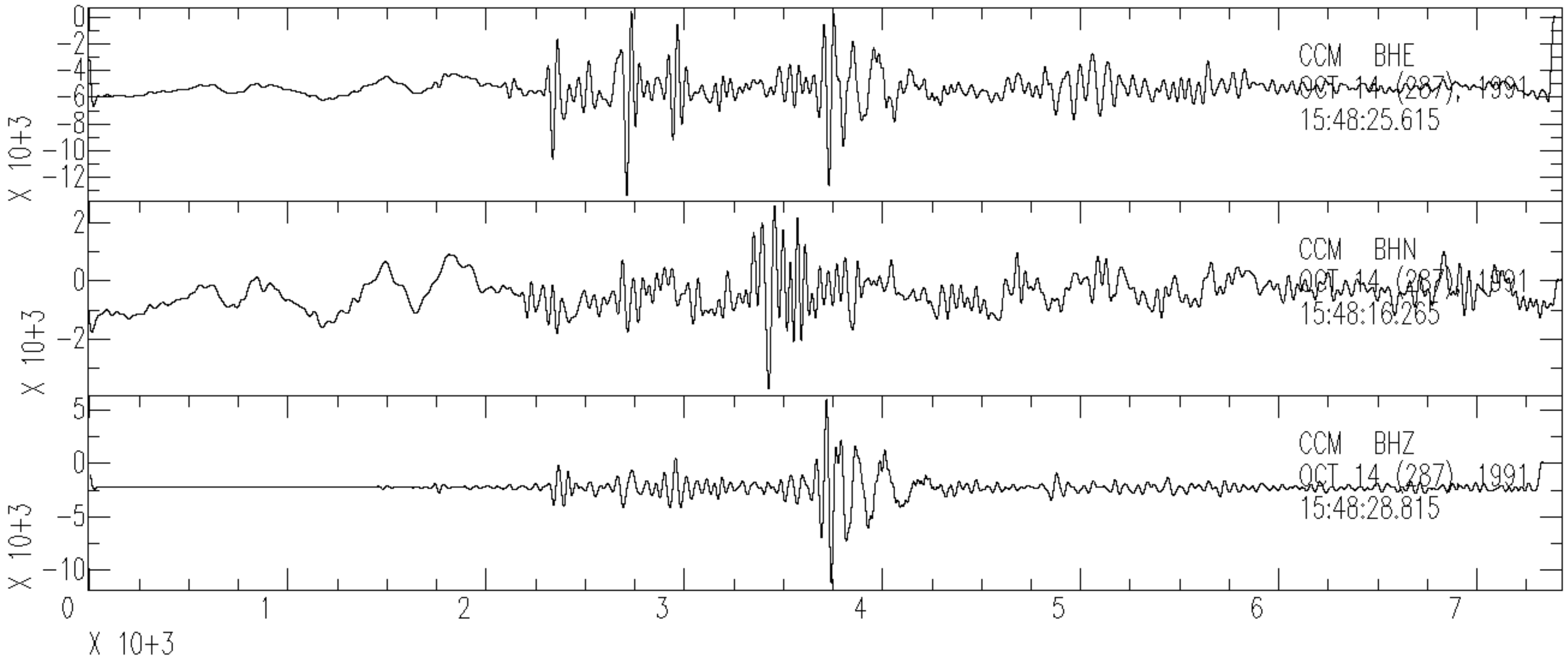
Read in some data – do some processing

```
SAC> read ./ccm_solomon*bh?
…
SAC> p1
```



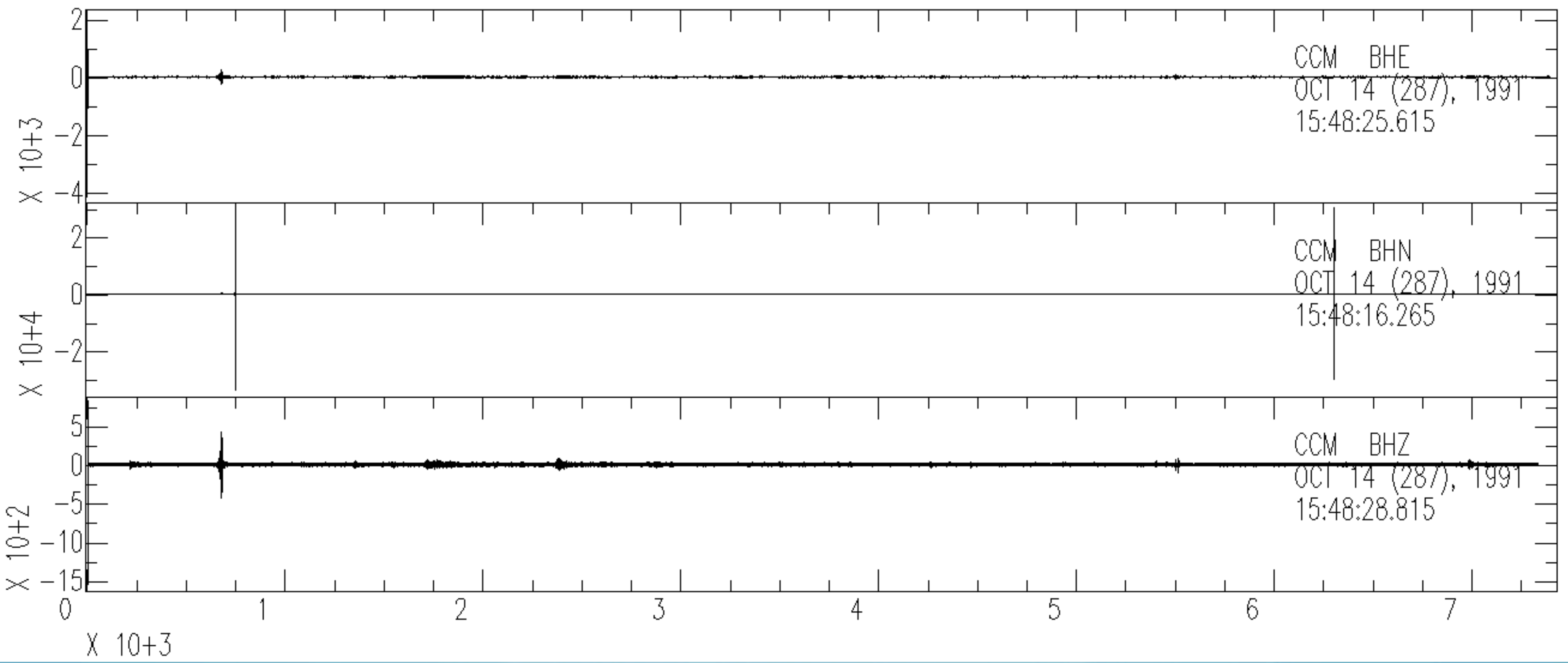
Low pass filter it

```
SAC> lp co .025 npoles 4 passes 2
SAC> p1
```



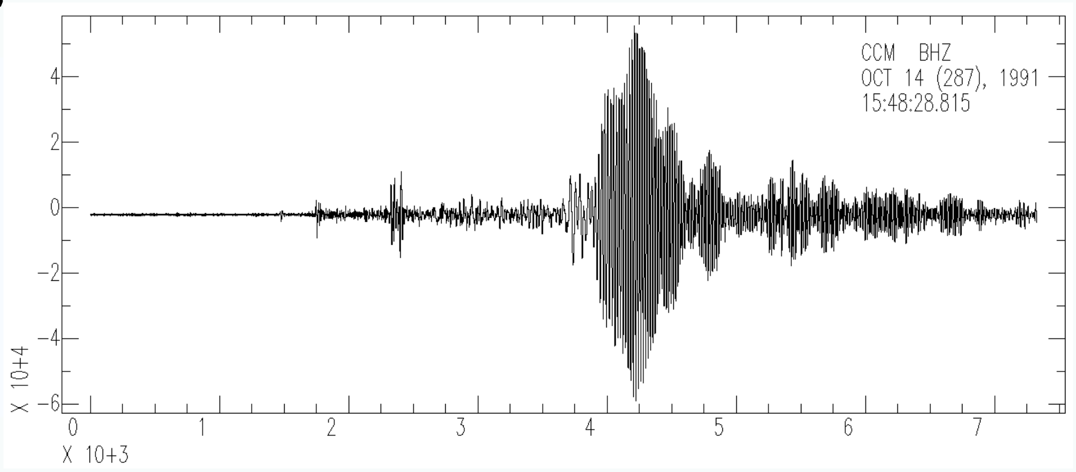
High pass filter it

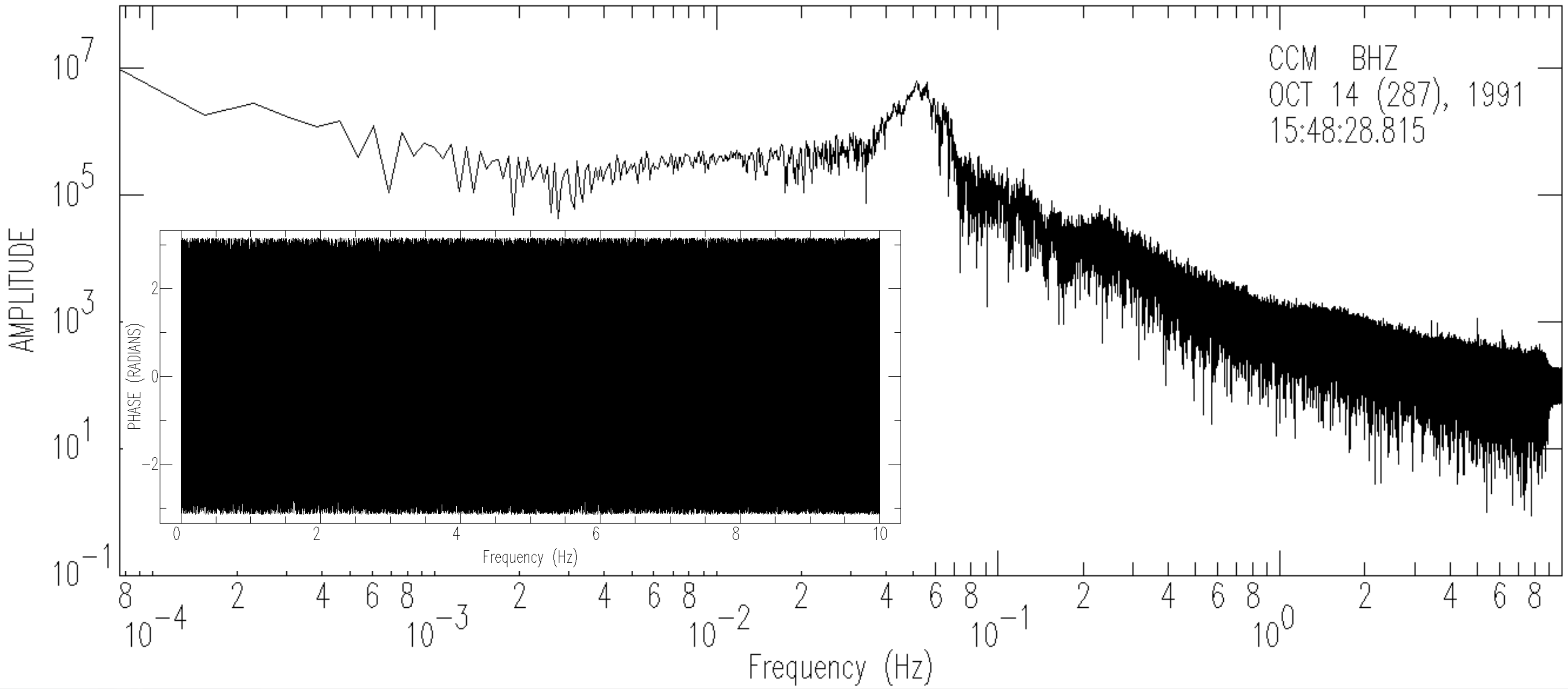
```
SAC> r
SAC> hp co 1 npoles 4
SAC> p1
```



Spectral analysis – Fourier transform

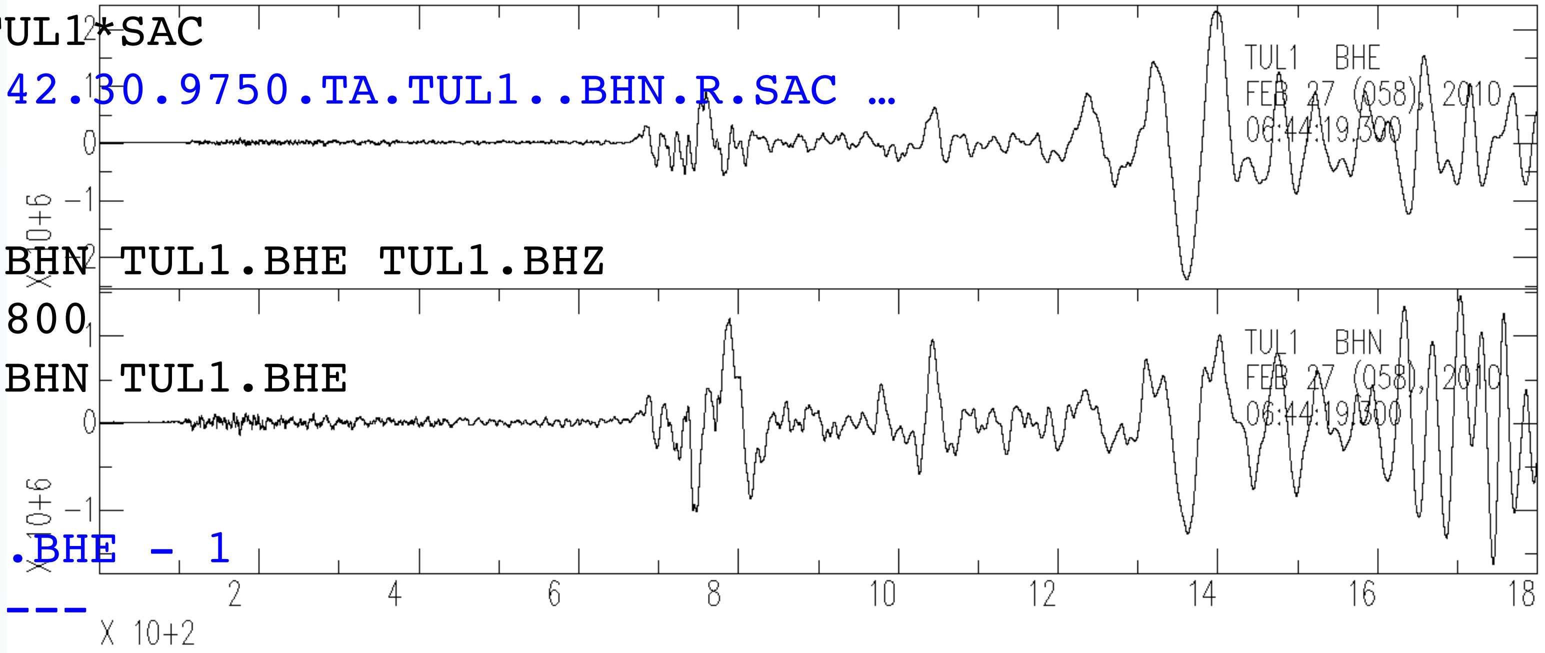
```
SAC> read ccm_solomon_*z
SAC> fft
SAC> psp
Waiting
SAC>
```

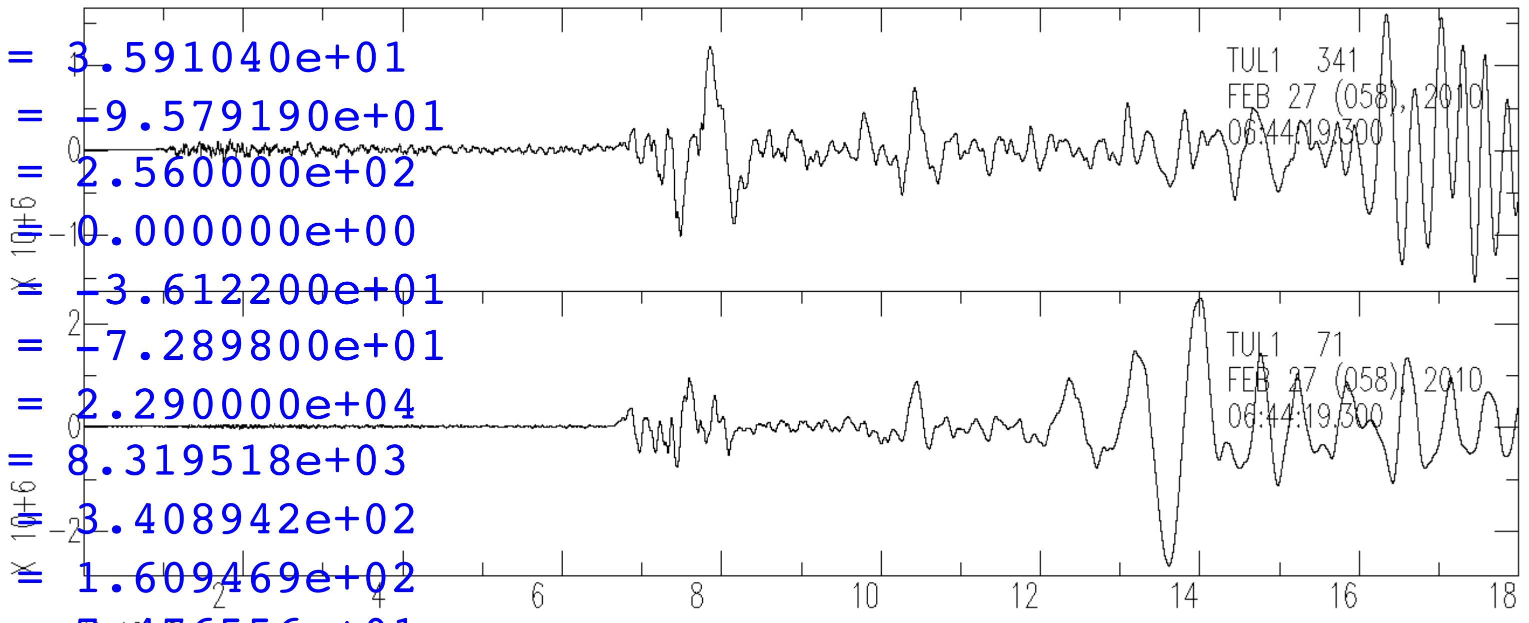


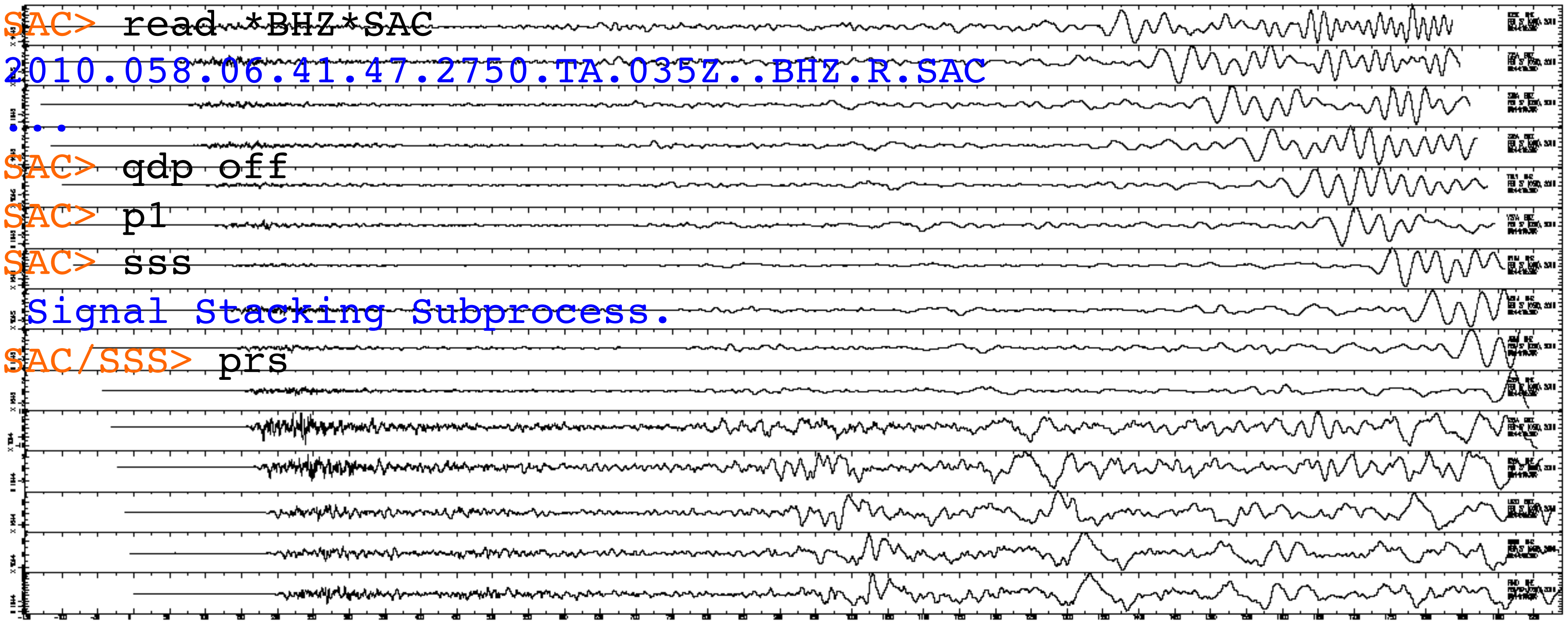


Rotate seismograms

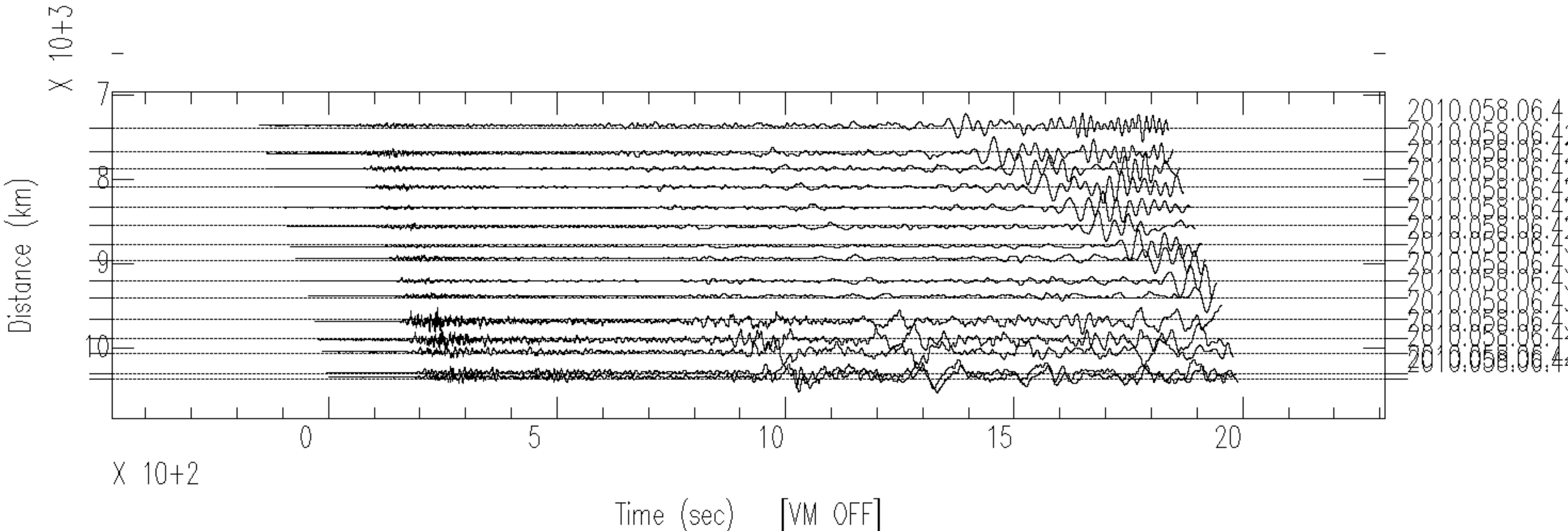
```
SAC> read *TUL1*SAC
2010.058.06.42.30.9750.TA.TUL1..BHN.R.SAC …
SAC> p1
SAC> synch
SAC> w TUL1.BHN TUL1.BHE TUL1.BHZ
SAC> cut 0 1800
SAC> r TUL1.BHN TUL1.BHE
SAC> rotate
SAC> lh
  FILE: TUL1.BHE - 1
 --------------
...
        STLA = 3.591040e+01
         STLO = -9.579190e+01
         STEL = 2.560000e+02
         STDP = 0.000000e+00
         EVLA = -3.612200e+01
         EVLO = -7.289800e+01
         EVDP = 2.290000e+04
        DIST = 8.319518e+03
           AZ = 3.408942e+02
          BAZ = 1.609469e+02
        GCARC = 7.476556e+01
```







```
SAC> read *BHZ*SAC
2010.058.06.41.47.2750.TA.035Z..BHZ.R.SAC
...
SAC> qdp off
SAC> p1
SAC> sss
Signal Stacking Subprocess.
SAC/SSS> prs
```


X 10+3
Distance (km)
X 10+2
Time (sec)
[VM OFF]