Tomographic Surface Wave Inversion

1. Preliminaries

After downloading the directory for this tutorial, start a terminal shell (with X11 running) — make sure you have a cshell running (csh or tcsh). Change directory to the tutorial directory and, if you have g77 installed (and in your path) then type "makeall". If you have gfortran installed then type "makeall95". You will need gv (ghostview) installed to make plots interactively when you run some of the scripts described below.

If you are not sure if you have the above programs available, type 'which gfortran' etc at the prompt and you'll find out whether your system knows about them.

2. Background on surface wave tomography

In this tutorial, you will produce tomographic images using real data. We use phase data for fundamental mode surface waves to construct global phase velocity maps. Such maps are typically made for periods between 250 and about 35s. Surface waves at a particular period sense structure over a wide depth range so we require maps for a whole suite of periods to find out details of seismic structure at depth (which we donÕt do in this tutorial). Even a map at a single period can give you a good idea of typical seismic properties of major geologic features, such as continental shields, old oceans, mid-oceanic ridges, large back-arc basins and, sometimes, even hotspots (thought the ability to resolve the latter is quite controversial). Traditionally, surface spherical harmonics have been chosen as the parameterization for a phase velocity map though spherical harmonics are somewhat impractical if we want to image small-scale features. Hence, more recently, as the data coverage has improved, researchers have been considering alternative parameterizations (e.g. pixels or local splines) to accomodate small-scale structure. Recently, there has also been interest in improving the theoretical basis for the inversions so that sensitivity to off great-circle path structure can be modeled. Here, we compare phase velocity maps made by using great-circle ray theory versus finite frequency kernels for a parameterization in terms of equal area pixels. The data we use are minor arc Rayleighwave phase measurements at periods of 150s (R150.raw1.sel), 100s (R100.raw1.sel), and 50s (r050.raw1.sel). These measurements were made by Goran Ekstrom's group while he was at Harvard.

3. Making a model

The first thing you have to do is to make the matrices which relate the phase measurements to the model (which is in $\delta c/c$ in percent, where c is phase velocity). The script **runphs** makes the matrix for great circle propagation and looks like

```
readphs <<!
blksw2
outputmatrix
R050.raw1.sel
3.952
!
```

'blksw2' is the file that specifies the parameterization — this one is for 2 degree pixels at the equator — changing the first number in this file changes the coarseness of the parameterization — you can try pixels of 1,2,5 or 10 degrees in size. The programs will complain if you try a block size smaller than 1 degree. The final number in the script is the average phase velocity for this period — you should use 4.280, 4.080, and 3.952 for 150, 100 , and 50 seconds respectively.

The script that makes the finite frequency kernel matrix is called **runphsff** and looks like

```
phasematff <<!
blksw2
outputmatrix
1
20 .25
3.952
R050.raw1.sel
!
```

Though the order is different, the input is the same as before except for the line "20 .25" which gives the center frequency of the calculation in mHz (20mHz=50sec) and the frequency band over which the measurement was made. Here, .25 means the band is $0.25 * 20 = 5$mHz wide so the measurements were made from 17.5mHz to 22.5mHz. You can use 0.25 for all bands but don't forget to change the first number to reflect the frequency of your data file.

Once you have computed your matrices, you might want to know how well your data sample your model. We do this by computing a 'hit count' where we add 1 each time a pixel is sampled by a matrix element. Not surprisingly, the hit count for the finite frequency matrix will be much larger than the hit count for the great circle matrix. Use script **runcount**:

```
swraycount <<!
blksw2
1
matrixfile
1
!
domaprayct blkcnt.out
```

This script produces the file 'blkcnt.out' which is plotted by **domaprayct**. The only thing you change is the matrix file name. **domaprayct** is set up to read 2 degree models so you will have to change some numbers in it to handle other resolutions. The only critical line is the one that says

```
read 90 180 1
```

which would become

```
read 180 360 1
```

if a 1 degree parameterization is used. Your ability to resolve structure and to get precise models depends mainly on data coverage so take note of what your hit count map looks like.

The next step is to make the actual model. This is done using the script **runlsqr**:

```
lsqrphs <<!
blksw2
1
matrixfile
$1
$2
500
!
dolsqmap_gv10 tomo.int3
```

$1 is the first command line argument which is the multiplier for the errors (which are typically not well-known) — you should use 1 to start with. $2 is the second command line argument and controls the amount

of smoothing — try a number between 0.1 and 10. The goal is to choose these numbers so that you end up with a reasonably smooth model that fits the data. This is where the art of inversion comes in! The final number in the script is the maximum number of iterations to be done — 500 is usually ok but you may need to make this larger if you are doing lightly smoothed high resolution inversions.

Some numbers are printed to the screen as the iterations proceed. The program uses convergence of the length of the model vector to decide when to stop since convergence of the fit to the data is usually quite rapid but the effect of smoothing will not be captured for several iterations. This script works with both great circle and finite frequency matrices. The model is in a binary file called 'tomo.int3'. This is plotted by the script **dolsqmap_gv10**. There is a color table in this script which you can adjust if you so desire.

Some of the numbers printed by the previous script reflect the fit of the model to the data but this actually includes how well you have satisfied the smoothing constraint. To see how well you actually fit the data, you should use the script **runchi**:

```
chisqphs <<!
blksw2
tomo.int3
1
matrixfile
$1
zz
.2
50
!
cfilh
```

Here, the command line argument $1 is the multiplier you used for the errors when you made the model. The program outputs the model roughness defined by $(\mathbf{D} \cdot \mathbf{x})^2$, the fit to the data as defined by χ^2/N, the unnormalized variance reduction (data not divided by errors) — multiply by 100 to get percent, and the normalized variance reduction. You want the variance reduction to be large, but more importantly, χ^2/N should be close to 1. The program also computes a histogram of residuals (plotted by **cfilh**) and gives some statistics of the residuals. Note that the residuals are divided by the measurement errors so the standard deviation of the histogram should be close to 1. The program also outputs a list of outliers (defined as being off by more than 5 standard deviations) – this is written into fort.35.

4. Resolution and error

The next step is to evaluate resolution. We do this using a checkerboard test though we actually use the shape of a spherical harmonic with a value of 1 when the spherical harmonic is positive and a value of -1 when the spherical harmonic is negative. You are asked to input the (l, m) of the harmonic you want to recover. Remember that 360 divided by l roughly gives the wavelength of the harmonic — or twice the size of an equivalent block (two blocks per wavelength). You should obviously choose l so that the implicit block size you are solving for is bigger than your model pixel size. To get something that looks like a checkerboard, you should choose m to be about half l. The script is called **runcheck**:

```
lsqrche2 <<!
blksw2
l m
1
matrixfile
$1
$2
500
!
dolsqmap_gv10 tomo.int3
```

The command line arguments are: $1 is the multiplier for the errors you used in the inversion and $2 is the smoothing you used in the actual inversion. The output checkerboard goes into 'tomo.int3' so make sure you don't overwrite your actual model.

Finally, we want to make an error map and we do this using the script **runlsqrerr** which does multiple inversions using data sets that differ by having random numbers with the statistical attributes of the data errors added to the true data vector:

```
lsqrerr <<!
blksw2
1
matrixfile
$1
$2
500
$3
!
errmod <<!
tomo.err3
$3
!
dolsqmap_gverr stdmodel
```

The command line arguments are: $1 is the multiplier for the errors you used in the inversion, $2 is the smoothing you used in the actual inversion, and $3 is the number of realizations you actually want to do (50 to 100 is fine). This program produces a binary file called 'tomo.err3' which contains all your realizations. The script next runs the program errmod which askes for the 'tomo.err3' file and the number of realizations it contains. This program produces two files: 'meanmodel' which is the mean model, and 'stdmodel' which is the error map. You can plot the mean model using **dolsqmap_gv10** and the script plots the error map using **dolsqmap_gverr**.

There are a couple of other programs which allow you to do a comparison of models as a function of wavelength. *blktosph* takes a pixel model and produces the equivalent spherical harmonic expansion as an ascii file. The inputs are obvious except for the maximum spherical harmonic degree. Remember, there is no point in computing coefficients for harmonics shorter than allowed by the pixel size. For example, a 2 degree pixel model corresponds to a minimum wavelength of about 4 degrees or a maximum harmonic degree of 360/4=90. Program *lcorr* takes two such spherical harmonic files and returns the correlation and amplitudes as a function of harmonic degree. A script to make a plot of the correlation between two models as a function of harmonic degree is provided (**runcorr**) though it is set up for 2 degree pixel models and a maximum harmonic degree of 80

```
blktosph <<!
blksw2
$1
coff1
80
!
blktosph <<!
blksw2
$2
coff2
80
!
lcorr <<!
```

coff1
coff2
1
!
cfil

The command line arguments are the two model file names. Files coff1 and coff2 contain the spherical harmonic expansion coefficients.

5. Comparison of the two theories

You may need quite different values of the smoothing and error scaling for the two inversions to get a valid comparison. The finite frequency inversion is naturally better constrained since the matrix is fuller so may need to use less smoothing to get models which are similar to the ray theory inversion. Questions you might ask: if i have two models with similar spectral content, which theory do i fit the data better with?; if i have two models with similar fit to the data, which has better resolved structure — and where is the better resolved structure?

You might like to interpret some of the features in your maps — use the fact that the longer period surface waves are sensitive to deeper structure to help you. Where do we find anomalously fast and anomalously slow regions? Do the patterns have anything to do with the continent/ocean distribution? Do we see correlation with other major tectonic features?