User's Manual
Full-Wave Ambient Noise Tomography

Compiled by
Yang Shen, Wei Zhang, Zhigang Zhang, …
Graduate School of Oceanography
University of Rhode Island

Version 1.0
Draft, 03.2011

Version Y2022.09

* This version is modified Xiaotao Yang from Shen et al. 's original version, obtained from Haiying Gao. This is not an official user manual for FWANT. This document is only for personal reference by Xiaotao Yang. The parts added by Xiaotao Yang are in blue color as this sentence.

Contents

1. Introduction

Seismic records at pairs of stations are weakly correlated due to ambient seismic waves that propagate between the stations. Empirical Green's Functions (EGFs) can be extracted from ambient noise by cross-correlating records at the stations. Numerous studies have shown that EGFs provide useful signals at a wide frequency range and can be used to image the earth structure (e.g., Shapiro et al., 2005; Nishida et al., 2009). Because EGFs do not depend on earthquake sources and the locations of these "virtual sources" are known precisely, ambient noise tomography has become a powerful tool in seismic imaging, particularly in aseismic regions.

This User's Manual provides a step-by-step explanation of the procedure and scripts used in full-wave ambient noise tomography (Shen and Zhang, 2010). The method is based on the scattering-integral approach (Zhao et al., 2005; Zhang et al., 2007; Chen et al., 2007a,b; Zhang and Shen, 2008). The users are assumed to have basic training in seismology and be familiar with shell programming. Figure 1 shows the flow chart of full-wave ambient noise tomography. Station Strain Green Tensors (SGTs) are constructed from a 3D reference model by finite-difference simulation of the response to orthogonal unit impulsive point forces acting at stations. Travel time, amplitude, or waveform anomalies are measured from observed and synthetic waveforms at stations. The station SGTs are used to calculate finite-frequency sensitivities to perturbations in Vp, Vs (or bulk and shear moduli), density, and attenuation. The measurements and structural sensitivity kernels are used to invert the earth's structure. The tomographic inversion results and additional constraints (e.g., receiver function solutions, well logs) are added to the 3D reference model. This process can be repeated to progressively improve the resolution.

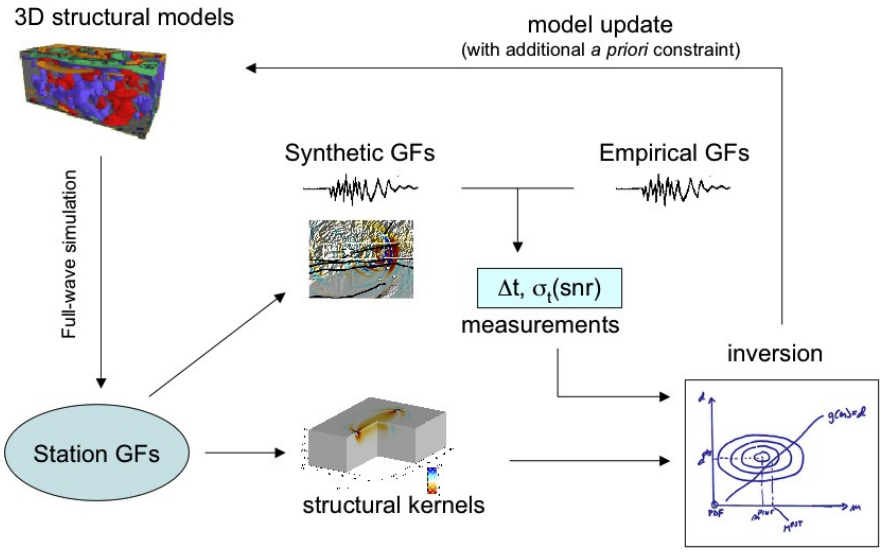


Figure 1. Flow chart of full-wave ambient noise tomography.

Depending on the dimension of the study area, the forward wave propagation simulation may be carried out in a (local) 3D Cartesian coordinate (Zhang et al., 2008; Zhang and Chen, 2011) or a spherical coordinate (Zhang et al., 2011). Examples in this User's Manual are based on wave simulation in the spherical coordinate for a regional- scale study.

We use (c/bash) shell scripts to drive nearly all data processing and computation. Modifications are needed for different users, study areas, data sets, etc. We suggest that users follow a similar directory and file structure as in the examples until they are familiar with the entire procedure.

If you use full-wave ambient noise tomography in your research, please consider citing the following reference (to be updated):

Shen, Y. and W. Zhang (2010), Full-wave ambient noise tomography of the northern Cascadia, SSA meeting (abstract), Seismological Research Letters, 81, 300.

2. Directory and file structure

We process continuous seismic data to extract EGFs on a workstation and run wave propagation simulation, kernel calculation and inversion on a Linux cluster.

On the workstation, the directory structure for data processing looks like following:

$proj_hm_dir (directory for the project, usually in user's home directory) csh (subdirectory containing shell scripts)
data_reqs (subdirectory containing data request files) matlab (matlab scripts)
STinfo (subdirectory containing station files)

In addition to $proj_hm_dir, the user needs a large internal or external disk ($disk) to store the data downloaded from the IRIS DMC and intermediate data generated in processing. When EGFs are obtained, they are copied to the "data" directory on the cluster (see below).

3. Data Processing

Data processing consists of following steps. Each involves one or more c-shell scripts listed in Appendix. Starting from (e), the scripts can be daisy-chained as in '**./csh/rerun.csh**', though power outage, matlab server/license problem, and most likely errors in the scripts may cause a breakdown of the chain, resulting in incomplete or erroneous results. We recommend that users test individual scripts with a small data sets (a few stations), before run them in a daisy chain.

A. Prepare a station list for the study area

First, find all available stations in the study area from the IRIS DMC (http://www.iris.washington.edu/SeismiQuery/station.htm). You will need the following station information in the station list: network, station, elevation, start time, end time, site like, latitude, and longitude.

Depending on the wave periods to be used, it may be unnecessary to download broadband records with a high sampling rate (e.g., BHZ at 40 sps). Some stations have multiple data streams (e.g., BHZ and LHZ). It may be sufficient to use LHZ records with a sampling rate of 1 sps. We suggest at least 7 samples per minimum period for adequate signal fidelity.

Use "gmap" at the IRIS DMC to find the stations with the right channel and time, (e.g., http://www.iris.edu/gmap/?chan=BHZ&minlat=45&maxlat=55&minlon=5&maxlon=155&timewindow=1995-2010). Copy and save the station names at the right side of the "gmap" window. Run '**./csh/sort_stn_lst.csh**' to select the wanted ones from the list of available stations.

B. Make data request files

Run '**./csh/mk_yearly_req.csh**' to general BREQ_FAST files.

For LH channels, we ask for one full-year record of a single channel at a single station in each request. For BH channels, we request 6-month or monthly record to limit the size of the resulting SEED files (large files tend to hang up during ftp download).

Note: The horizontals of borehole stations are different (LH1 and LH2).

C. Send data request files to the IRIS DMC Run '**./send_allreq.csh**'

D. Download SEED files from the IRIS DMC

After the data requests are submitted to the IRIS DMC, you will receive automated emails about the status of your request. You may also check the IRIS web site to see if the data are ready to be downloaded (http://www.iris.washington.edu/data/, view request status and shipment). Follow the instructions in the email from IRIS to download the files, if the files are small and can be downloaded within a few hours. Otherwise, go to your directory at the IRIS ftp site and make a list of the SEED files and save it in the directory $disk/seed_data.

Run '**./csh/fetch_seed.csh**' to fetch the SEED files.

This script compares the SEED files in the local directory against the list and determines which SEED files are yet to be downloaded from IRIS. You may run several ftp streams using the script. Give each ftp stream a different "tag" to uniquely identify the SEED files to be downloaded. Sometimes ftp hangs up and you have to terminate the ftp job. The ftp site has a maximum connection time, so you may run out of time before the files are all downloaded. You can restart '**./fetch_seed.csh**" to download the remaining files. Use '**./csh/find_incomplete_seed.csh**' to remove incomplete SEED files in your local directory, and then do a final sweep with '**./fetch_seed.csh**'.

E. Extract sac files and remove instrument response

Run '**./rdseed2sac_yr.csh**' to extract daily sac files from SEED and remove instrument response. The output is ground displacement. The script checks instrument reversal and gain. For password protected data, use '**./openssl.csh**' first to remove encryption.

F. Find and remove incomplete sac files

Some sac files are incomplete (missing data in the daily files). If the number of incomplete files is relatively small, then removing incomplete sac files simplifies subsequent processing.

Run './csh/find_and_rm_incomplete_sac.csh'

G. Delete stations with insufficient data Run '**./csh/count_sac_files.csh**'

H. Check station orientation

Some stations, even the GSN stations, are/were not correctly oriented. This is perhaps not an issue if instrument reversal and gain are checked in "rdseed".

To find out if the orientations are correct, run '**./csh/find_orientation.csh**' To rotate the horizontals to N and E, run '**./csh/rotate_NE2NE.csh**'
To rotate LH1 and LH2 (borehole instrument) to N and E, run
'**./csh/rotate_LH12toLHNE.csh**'

I. Re-sample

As a rule of thumb, you need ~7 points per minimum period to obtain good signal fidelity. For example, if the minimum period to be used is 15 s, it is safe to use a 0.5 sample-per-second sampling rate to reduce the cost of subsequent computation.

Run './csh/interp2halfsps.csh'

J. Frequency Time Normalization (FTN) Run '**./csh/cal_FTN_sac.csh**'

We use the signal normalization method of Ekstrom et al. (2009), instead of the commonly used one-bit normalization. This is a relatively time-consuming calculation. If necessary, you may run it on multiple nodes/cpu cores. There is no point to set a high frequency limit above what you will be able to use. The highest useable frequency depends on the

available computational resources. We set the lowest frequency and the frequency interval to half the lowest frequency to be used (For example, if we want to use up to 100 s period waves, we set the low frequency limit and frequency interval in the FTN calculation to 0.5*0.01 = 0.005 Hz).

K. Sort the FTN files by day

Run './csh/sort_FTN_by_day.csh'

L. Remove days that have less then 2 stations

Run './csh/count_sac_by_day.csh'

M. Synchronize files

SAC files extracted from SEED start at slightly different time within the sampling interval (e.g., less than +/- 1 s for LH channels). They need to be synchronized.

Run '**./csh/sync_sac.csh**'

N. Delete time segments with large (M>5.5) earthquakes Run '**./csh/del_EQs.csh**' (http://www.iris.edu/SeismiQuery/sq-events.htm)

O. Calculate cross correlation between records at pairs of stations Run '**./csh/cal_xcorr.csh**'

P. Find and remove corrupted cross correlations

Matlab crash and perhaps some other problems may cause cross-correlation to be “nan” or “0.000000e+00”. A single “nan” file spoils the stacked result for the station pair. Use '**/find_nan_files.csh**', to find and remove those corrupted files.

Q. Select station pairs

Count the number of cross correlation files in each station pairs and remove those with insufficient (e.g., < 300 days) day files '**./count_xcorr_files.csh**'

R. Stack cross correlations and calculate EGFs '**./stack_xcorr_bothsides.csh**' or '**./stack_xcorr.csh**' (folded about the origin)
Use ‘./matlab/plt_egfs_by_dist.m’ to plot EGFs as a function of distances between station pairs.

S. Split EGFs to negative and positive sections.

../../csh/split_egfs.csh

Convolve EGFs with source time function used in the calculation of SGTs,

../../csh/cnv_egf.csh

run ../../csh/find_gcp.csh to exclude station pairs outside or too close to the model boundaries;

Note: I ignore step P (find_nan_files.csh). After stacking,

(1) find_nan_EGFs.m

(2) find_nan_files.csh

(3) stack_xcorr_bothsides.csh

(4) split_sgfs.csh

(5) cnv_egf_ax.csh

(6) find_gcp.csh

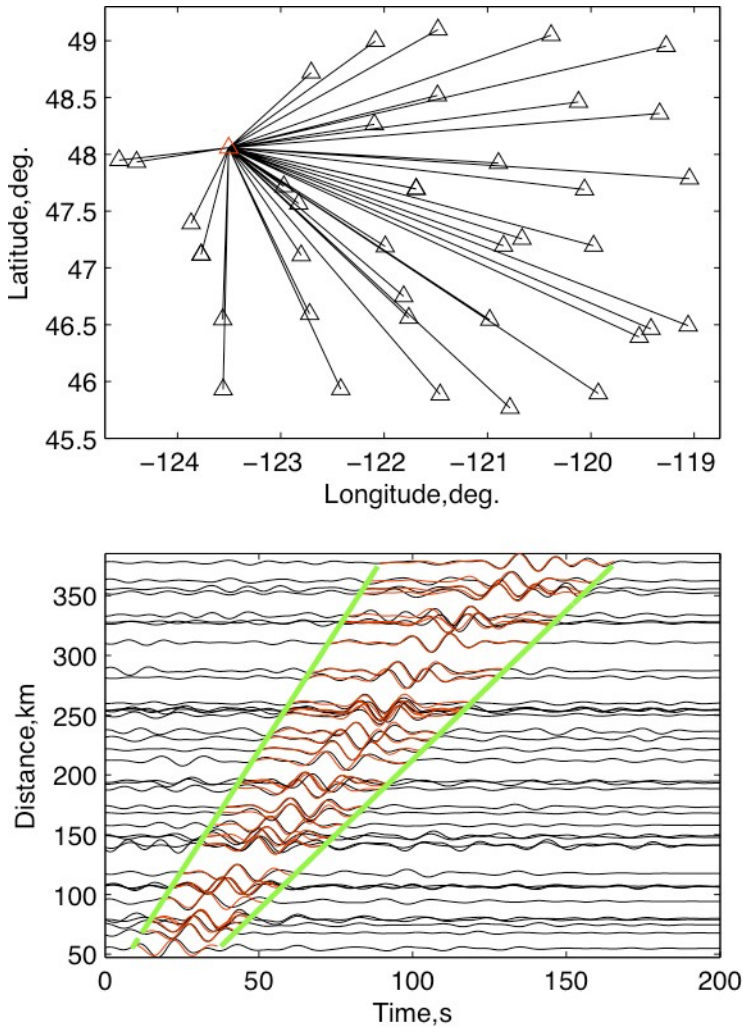


Figure 2. (Top) Lines connect the “virtual source” (red triangle) and receivers. (Bottom) Empirical Green’s Functions (black) are plotted as a function of station distances. The red lines are the synthetic Green’s functions for the 4th-iteration model of northern Cascadia (Shen and Zhang, 2010). The green lines mark the arrival times for waves at the speeds of 2 and 4 km/s. The waveforms have been filtered between 10-20 s period.

4. Wave Propagation Simulation and Inversion

On the cluster, the directory structure looks like following:

$proj_hm_dir (directory for the project, usually on the I/O node as it takes a large amount of disk space)

codes (subdirectory containing codes for wave simulation, kernel calculation, & inversion)

csh (additional shell scripts to prepare EGFs for measurements) data (subdirectory containing EGFs and derivatives)
ite_0? (?=1,2,3,.., model iteration)

matlab (matlab scripts)

mfiles (special matlab functions)

misc (miscellaneous files, including topography, coastlines, etc) model_updates (subdirectory to update the model after each inversion) STinfo

4.1 Select a model and simulation parameters

It pays to take time to select a model and optimum simulation parameters by running test simulations. See FD3Dspher User's Manual (version 1.0) for instructions on how to set up a model and run simulations. Here we summarize the main steps briefly. Define,

$sim_test = $FD3Dspher/run/$your_study_area

You may use a composite or volumetric model as your initial reference model. To define the study area, edit define_latlon.m in ./$sim_test/config.

To set up a composite model (e.g., AK135+CRUST2.0), run conf_media_composite.m

Output: SeisMedia.composite.$area.crust2.d1800.nc (copy it to $sim_test)

To generate grids in the latitude, longitude, and radial directions, run

./config/grid/creat_grid_xy.m and creat_grid_z.m

Outputs: grid[x,y,z].dat

Edit $sim_test/SeisGrid.conf using the grids in grid?.dat

Edit other configuration files (SeisFD3D.conf, SeisSource.conf, SeisStation.conf), then run ./bin/seis3d_grid

qsub pbs_media_mpi.sh

Make sure the maximum time step is OK by checking output spher.media.* Change the grids or the size of the time step in SeisFD3D.conf if necessary &

inspect the discretized media file (using ./draw_media_surf_all.m). run ./bin/seis3d_metric

run ./bin/seis3d_source run ./bin/seis3d_station qsub pbs_wave_mpi.sh

Inspect the snapshots and waveform (draw_snap_surf_spher.m and draw_seismo_single.m). If the waveforms are OK, you are ready to go to the next step.

The directory ./ite_0? should have the following subdirectories:

./sim.input (netCDF files for the coordinate and media, copied from $sim_test/input)

./sim.station (station SGTs and velocity on the free surface)

./measure (phase measurements)

./sim.kernel (sensitivity kernels)

./syn.seismograms (synthetic seismograms from each virtual source)

./inv.structure.kerVpVs (inversion)

4.2 Calculate SGTs and Seismograms (directory: sim.station)

In the subdirectory skel/fx under sim.station
Make necessary links

ln -s $sim_test/bin bin

ln -s ../../../sim.input input # Edit the configuration files

mkdir checkpoint output input.src

Create checkpoint.dat, with the following line:

0 0 0 # checkpoint, syncpoint, new nt

Note: the checkpoint file is critical for the program to run.

Make sure the source location line is empty in SeisSource.conf. Added a line above the final block to specify other unused sources as 0. Such as below:

```
#########################################################################
#                        for seis3d_source                              #
#########################################################################

distance2meter = 1.0E3
src_hyper_height = 1e10

number_of_moment_source = 0
#########################################################################
#                        single force source                            #
#########################################################################
number_of_force_source = 1
force_stf_window = 1
force_stf_type = gauss
force_stf_timefactor = 6  # gauss t0; ricker t0; bell starting
force_stf_freqfactor = 2  # gauss a;  ricker fc; bell width

# x,y,z           | start  | f0    | fx   fy   fz
# x=colatitude
```

```
# 150.0 80.0 9000.0  1.0e+16  0.0  0.0  1.0
<anchor_force>
```

In sim.station, run ./skel2station.sh to generate station directories. Before running this step, make sure the SeisFD3D.conf and the SeisSource.conf are correct and consistent with the simulation test parameters. Make sure the SOURCE_ROOT is input.src, instead of input as used by the simulation test.

Example of *skel2station.sh*:

```
list_sta="../../STinfo/craton_station_withdata.txt"
template="skel/fx"
echo $list_sta
echo $template
cat /dev/null > rest.lst0
for KSTNM in `awk '{print $1 "." $2}' ${list_sta}`; do
    KNWNM=`echo $KSTNM | awk -F. '{print $1}'`
    KEPNM=`echo $KSTNM | awk -F. '{print $2}'`
    echo $KSTNM $KNWNM $KEPNM
    if [ -e $KSTNM ]; then
      echo '$KSTNM exists'
      continue
    fi
    mkdir -p $KSTNM
    cp -R ${template} ${KSTNM}/fz

    # SeisSource.conf
    # for spherical coord.
    lon=`grep $KNWNM $list_sta | grep $KEPNM | awk '{print $3}'`
    lat=`grep $KNWNM $list_sta | grep $KEPNM | awk '{print $4}'`

    colat=`echo "90 $lat" | awk '{print $1-$2}'`

    echo $colat $lon

    echo "$colat $lon 2e10   0.0  1.0e+16   0.0 0.0 1.0" >>
${KSTNM}/fz/SeisSource.conf

    cat /dev/null > ${KSTNM}/fz/pbs_wave_mpi.sh
    cat ${template}/submit_wave_mpi.sh | while read -r strline; do
        if [ `echo $strline | grep ^"#BSUB -J" | wc -l` -eq 1 ]; then
           strline="#BSUB -J $KSTNM"
           echo "${strline}.fz" >> ${KSTNM}/fz/pbs_wave_mpi.sh
        else
           echo "$strline" >> ${KSTNM}/fz/pbs_wave_mpi.sh
        fi
    done
    echo "${KSTNM}/fz" >> rest.lst0
    cd ${KSTNM}/fz
    ./bin/seis3d_source
    cd ../..
done
```

```
/bin/cp rest.lst0 work.lst
```

Run ./subjob.csh and ./subjob_timed.csh

The scripts use a list, work.lst, to keep track which stations have been calculated.

The stations that have been calculated have "#!" before their names.

Note: to clean the node memory, /home/$user/bin, edit list_node_ipc to list nodes need to be cleaned and then run clean_ips_priv.sh

Check to make sure all stations are finished. There are two shell scripts for this purpose: search_unfinishedjobs.sh and get_unfinished_joblist.sh.

search_unfinishedjobs.sh

```
#!/bin/bash
nbad=0
sed -e 's/#!//g' work.lst > temp.lst
for work in `cat temp.lst `
do
      count=0
      if [ -f ${work}/seis3d_wave.log ]
            then
                  count=`grep -c finish ${work}/seis3d_wave.log`
      else
            count=0
      fi

      if [ ${count} -eq 0 ]
            then
                  echo ${work}
                  (( nbad = ${nbad} + 1 ))
      fi
done

echo ">>>>>>>" ${nbad} "unfinished stations"
```

get_unfinished_joblist.sh:

```
#!/bin/bash
./search_unfinishedjobs.sh | sed -e 's/\//./g' | sort | awk '{ if ( NR >
1 ) print $1 }'
```

Re-process unfinished stations. Check the reason it failed (most of time, it is because of the request time). Re-run using correct parameters, before moving on to the next step.

4.3 Extract synthetic seismograms at stations (directory: syn.seismograms) Run syn.seismograms/snap2sac_spher.m. This step can be done on the cluster through

sending matlab remote job. Then the extracted seismograms need to be copied back to the local machine for the next step, or conduct the next step on the cluster as well.

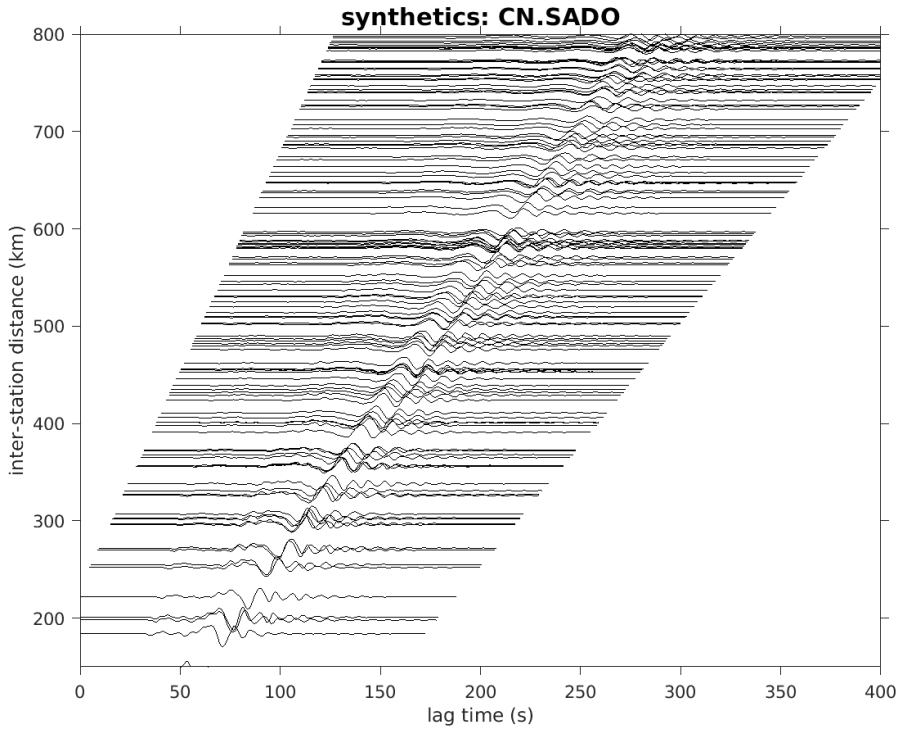
The script reads outputs from sim.station and writes seismograms at all other receivers in sac.

1. Check synthetics

This can be done before extracting all stations. You can do this even with only one station finished. This will help make sure the parameters are correct. Use ***"plot_synthetics.m"*** to plot the synthetics for all receivers.

```
addpath(genpath('/depot/xtyang/data/codes/FWANT/Codes/MatlabFiles'
))
R0=6371;
% define filter parameters. Currently not used.
T_min = 20; T_max = 50; %period min and max.
fband=[1/T_max, 1/T_min]; % frequency band
% km/s, minimum and maximum group velocities
cmin=2.5; cmax=4.5;
signal_window_extend=50;
npoints=1501;
%%%%%%%%%%%%%% END OF - Setting up global
parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
sta = 'CN.SADO';
unix(['ls -f ' sta '/*.fz.Uz.SAC > filelist3']);
synfile1=textread('filelist3' ,'%s');
nsynfile=length(synfile1);
synfbdata=zeros(npoints,nsynfile);
evla=nan(nsynfile,1);
evlo=nan(nsynfile,1);
stla=nan(nsynfile,1);
stlo=nan(nsynfile,1);
distall=nan(nsynfile,1);
for i=1:nsynfile
   filename1=char(synfile1(i));
   %clear dd1 tt
   dd3=readsac(filename1);
   dtsyn=dd3.DELTA;
   ttsyn=dd3.B:dtsyn:dd3.E;
   nptsyn=dd3.NPTS;
   Feff=1/dtsyn/2; % in increment of 2
   evla(i)=dd3.EVLA;evlo(i)=dd3.EVLO;
   stla(i)=dd3.STLA;stlo(i)=dd3.STLO;
```

```
   [distall(i),~]=distance(evla(i),evlo(i),stla(i),stlo(i));
   distall(i)=round(R0*distall(i)*pi/180.0);
   synfbdata(:,i)=dd3.DATA1;
end
%%
figure('Position',[300, -200, 800, 600]); hold on;
scale=100;
for i=1:nsynfile
   tmin=distall(i)/cmax-T_max;tmax=signal_window_extend +
distall(i)/cmin+T_max;
   itmin=round(tmin/dtsyn); itmax=round(tmax/dtsyn);
   if itmin < 1; itmin = 1; end
   if itmax > nptsyn; itmax = nptsyn; end
   plot(ttsyn(itmin:itmax),scale*synfbdata(itmin:itmax,i)+distall(
i),'k');
end
hold off;
xlim([0 400]);
ylim([150 800]);
box on;
axis on;
xlabel('lag time (s)')
ylabel('inter-station distance (km)')
set(gca,'tickdir','out')
title(['synthetics: ',sta],'FontSize',14);
saveas(gca,['synthetics_',sta,'.png'])
```



4.4 Measure phase delays (directory: measure) by cross-correlating observed and synthetic EGFs,

./measure_phase_delay.m

Assemble the measurements into a list: ./assemble_measure.csh

Output: measure_result.dat

Plot the ray paths: ./plt_GCPs.csh

Plot delay vesus distance: ./plt_data_dist_multiple.m

Apply QC procedures to prepare for the delay dataset for kernal calculation. The QC procedure will produce a “droplist” file to be used to accomplish the QC. Use shell script exclude_measurements.sh to exclude “bad” measurements.

```
#!/bin/bash
if [ $# -lt 2 ]
        then
                echo       'USAGE:       ./exclude_measurements.sh
meansurementfile excludelistfile'
                exit 1
fi
datafile=$1
listfile=$2
#nl=1
while read line
do
     #echo ${line}
     #echo ${nl}
     tline=`echo ${line} | sed -e 's/\// /g' | sed 's/_/ /g' `
     src=`echo ${tline} | awk '{print $4}'`
     rcv=`echo ${tline} | awk '{print $5}'`
     ftag=`echo ${tline} | awk '{print $12 }'`
     #echo ${src} ${rcv} ${ftag}
```

```
    c=0
    c=`grep ${src} ${listfile} | grep ${rcv} | grep -c ${ftag}`

    if [ $c == 0 ]
        then
            echo ${line} | awk '{print}'
    fi
    #(( nl = nl + 1 ))
done < ${datafile}
```

Plot EGFs vs. Synthetics: ./../matlab/plt_data_vs_syn_gao.m. Note: this is optional for earlier iterations.

4.5 Calculate sensitivity kernels (directory: sim.kernel)

Link the executables,
ln -s ../sim.input input

ln -s $FD3Dspher/code/bin bin

Generate conf files for the kernel calculation,

./kern_measure2kernelconf.sh

Note: when running ./kern_measure2kernelconf.sh, pay attention to the output: tshift= **. If there is an empty output here, it means the ../sim.station/skel/fx/SeisSource.conf is not in the right format for this script to read. Specifically, this script looks for "current"' to get the source time function parameters. In the ../sim.station/skel/fx/SeisSource.conf file, you have to add the comment "#current" in THE SAME LINE as force_stf_timefactor and force_stf_freqfactor. This is not the right way to deal with the parameter file. This needs to be taken care of in the future!

New note on 3/18/2022 by Xiaotao: I removed the "grep current" code. With this change, we need to make sure there is only one "force_stf_timefactor" line in the *SeisSource.conf* file. This step can be run on the head node but is not recommended, particularly when it needs a long time (a couple of hours). We should avoid overloading the head node as always. Here is the command to run it on the computing nodes:

```
sbatch -A xtyang -t 10:00:00 -N 2 kern_measure2kernelconf.sh
```

Creat directories for the kernel files,

./kern_conf2skel.sh

Generate synthetics in ascii in the receiver directories, .

/kern_synthetic4kernel.sh

Note: after generating the synthetics, remember to check the synthetic files to make sure the values look correct. Sometimes, there may be NaNs if the code didn't work correctly.

Generate matlab filter file,

./kern_createfilterfile.m. Now it is a function that can be called anywhere. It also prints out the filter file names for easy use when editing TomoKernel.conf.

Edit the configuration file, TomoKernel.conf. Note, pay special attention to the filter files, make sure it is in the same order as the frequency bands in measuring the phase delay. You can use the list printed by running kern_createfilterfile.m.

Edit pbs_kernel_mpi.sh or submit_kernel_mpi.sh if needed. Note that you may want to split the whole station list into a few blocks, depending on the resources available. The kernels are caculated by sequence in the station list produced by kern_conf2skel.sh. Use count_kernels.sh and clean_kernels.sh to help you. It is important (to save you time) to split the list as equal as possible. Here, equal doesn't mean the station list. Instead, you have to consider the number of receivers for each station. Use count_kernel_size.sh to help you count the nubmer of receivers in each station folder. I wrote a Python helper to help us decide the best sub-list sizes.

First run: ./count_kernel_size.sh station_conf_list > temp

Then, run the Python split_helper.py script. Here is what is in the script:

```
#this is a helper that produces the ending line number for each
block.
#this can be used when splitting the station list.

import pandas as pd
import numpy as np

nblock=5
df =
pd.read_csv('temp',names=['station','num'],delim_whitespace=True)
total = df.num.sum()
s = int(total/nblock)

cumsum=df.num.cumsum()
h = s
```

```
h0 = 0
print('block,end,size')
for i in range(nblock):
    if i < nblock-1:
        idx=np.where ((cumsum >=h)) [0][1] - 1
    else:
        idx=len(cumsum)-1

    isum=cumsum[idx]


    print([i+1,idx+1, isum-h0])

    h = isum + s
    h0=isum
```

Then run split_conf_list.sh: split_conf_list.sh

#!/bin/bash

#ending index of each block. output from split_helper.py

#[1, 267, 68318]

#[2, 356, 68846]

#[3, 426, 68961]

#[4, 523, 68239]

#[5, 629, 65210]

id1=267

id2=356

id3=426

id4=523

id5=629

awk '{ if (NR <= '$id1') print}' station_conf_list > station_conf_list1

awk '{ if (NR > '$id1' && NR <= '$id2') print}' station_conf_list > station_conf_list2

awk '{ if (NR > '$id2' && NR <= '$id3') print}' station_conf_list > station_conf_list3

awk '{ if (NR > '$id3' && NR <= '$id4') print}' station_conf_list > station_conf_list4

awk '{ if (NR > '$id4') print}' station_conf_list > station_conf_list5

Submit job, qsub pbs_kernel_mpi.sh or submit_kernel_mpi.sh

(Monitor the memory use in the nodes. If memory swap occurs, then you need to reduce the block_size to calculate the kernel in blocks).

NEW METHOD TO RUN KERNEL JOBS

- Create a submission template script: submit_kernel_mpi_template.sh
- Run conf2jobs.sh to create the list and submission script for individual stations

```
#!/bin/bash
conflist='station_conf_list'
template='submit_kernel_mpi_template.sh'
assem_template='submit_assem_kernel_template.sh'

for sta in `awk '{print $1}' $conflist`
do
     echo $sta
     stalist=${sta}.list
     stasubmit=${sta}.submit
     stasubmit_assem=${sta}.assem

     echo $sta ${sta}_conf > ${stalist}

     sed -e 's/KNJOBTEMPLATE/'${sta}'.kn/g' $template | sed -e
's/STATION_CONF_LIST_TEMPLATE/'$stalist'/g' > ${stasubmit}
     chmod a+x ${stasubmit}

     #assembly script
     sed -e 's/KAJOBTEMPLATE/'${sta}'.kn/g' ${assem_template} |
sed -e 's/STATION_CONF_LIST_TEMPLATE/'$stalist'/g' >
${stasubmit_assem}
     chmod a+x ${stasubmit_assem}
done
```

This script also produces the submission scripts for kernel assembly.

- Loop through each station and submit to standby queue by running: submit_kernel_next.sh

```
#!/bin/bash
if [ $# -lt 1 ]
     then
     echo 'USAGE: submit_kernels_next.sh unfinished_station_list'
     exit 1
fi
stalist=$1 #'unfinishedkernels_1.txt'

for sta in `awk '{print $1}' $stalist`
do
     #echo 'cleaning files'$sta
     #./clean_kernels.sh $sta
```

```
    echo 'submitting '$sta
    sbatch -A standby -t 4:00:00 ${sta}.submit
done
```

If you would like to run it on xtyang queue, you can submit the script directly. This might be needed for a few stations that need more than 4 hours to finish.

- If you have some stations already finished, run search_unfinished_kernels.sh first to only loop through the unfinished stations.

Plot sensitivity kernels,

draw_kernel_spher.m and draw_kernel_spher_multi.m

4.6 Inversion (directory: inv.structure.kerVpVs)

Generate inversion block dimensions and indexes,
inv_make_block_stride_xygridcent.m

Generate smoothness constraints,

inv_make_smooth.m

1th: first derivative (flatness); 2th: 2nd derivative (see Menke, p53)

I modified it to a function: make_smoothness_operator(blockfile).

After we have the inversion block dimension, we can assemble the kernels

./sim.kernel/run.kernel.assm.sh

(assembled results are saved in ./inv.structure.kerVpVs/G_spool)

Note: pay attention to the threshold kernel value, below which the sensitivity is ignored

Split the assembly job could save time.

Create a model directory (e.g., Freq3456.Z.model) Generate a list of the assembled kernels,

./list_G_raw.sh

Generate station and event ("virtual source") lists,

./mk_st_list.csh

Remember to change the iteration number before running this script.

Edit the following lists: inv_black_list, inv_st_list, inv_ev_list, inv_cmp_list, inv_freq_list, inv_twin_list, inv_phase_list (frequencies have to be in the same format as in ./measure/measure_result.dat)

Generate the final list used in inversion,

./inv_Gd_list.sh

I fixed bugs in inv_Gd_list.sh when looking for KEVNM and KSTNM. The old script doesn't distinguish, for example, XO.LG25 and XO.LG25A. This is caused by the use of `grep`, instead `awk` in searching for matched station names.

For future runs, use "make_inv_Gd_list.sh" instead, which has the capability of separating different frequencies. This is useful when processing large dataset to speed up this step.

Run ./Freq3456.Z.model/run.solver.1th.sh, with various damping and smoothness constraints

Use plt_slice.m to view the resulting model

Find the tradeoff between the solutions and damping and smoothing parameters, *plt_tradeoff.csh*

Select a model with the optimum variance reduction and model norm.

Variance reduction: 1- sum[(di-di_pre)^2]/sum(di^2)

Chi square: (Σ(di/σi)^2)/N

Note: di is the observed ith misfit between observation and synthetics, dt

di_pre is the predicted ith misfit dt between observation and synthetics, K(dc/c)dv LSQ: K(dc/c)dv=dt where dc/c is the velocity perturbation and dv volume.

4.7 Update model (directory: $projhmdir/model_updates)

Copy model input files,../ite_01/sim.input, to input_ite_01 and updated_input_ite_01 (media files in updated_input_ite_01 will be rewritten. If the study box is changed in the next iteration, copy input and correspoinding files from the $sim_test directory).

Copy configureation file,

cp ../ite_01/sim.station/skel/fx/SeisFD3D.conf SeisFD3D.conf_ite_01

Update the model by “TriScatteredInterp” the inversion result and adding it to the previous model, update_model_smth.m

Compare current and updated model, './plt_models.m'

Set up the directory and files for the next iteration, 'setup4next_ite.csh

Return to 4.2 and iterate until a satisfactory model is obtained.

Compare results of each iteration with matlab/hist_xcorreff.m.

References

Chen, P., T.H. Jordan, and L. Zhao, (2007a), Full three-dimensional tomography: a comparison between the scattering-integral and adjoint-wavefield methods, Geophys. J. Int., 170, 175-181.

Chen, P., L. Zhao, and T.H. Jordan (2007b), Full 3D tomography for crustal structure of the Los Angeles Region, Bull. Seismol. Soc. Am., 97 (4), 1094-1120, doi: 10.1785/0120060222.

Ekström, G., G.A. Abers, S.C. Webb (2009), Determination of surface-wave phase velocities across USArray from noise and Aki's spectral formulation, Geophys. Res. Lett., 36, L18301, doi:10.1029/2009GL039131.

Nishida, K., J.-P. Montagner, and H. Kawakatsu (2009), Global surface wave tomography using seismic hum, *Science*, **326**, 112.

Shapiro, N.M., M. Campillo, L. Stehly, M. Ritzwoller (2005), High-resolution surface-wave tomography from ambient seismic noise, Science, 307, 1615-1618.

Shen, Y., W. Zhang (2010). Full-wave ambient noise tomography of the northern Cascadia, SSA meeting (abstract), Seismological Research Letters, 81, 300.

Shen, Y., Z. Zhang, and L. Zhao (2008). Component-dependent Frechet sensitivity kernels and utility of three-component seismic records, Bull. Seism. Soc. Am., 98 (5), 2517-2525, doi: 10.1785/0120070283.

Zhang, W. and X.F. Chen (2011), 3D Elastic Wave Numerical Modeling In the Presence of Surface Topography by a Nonstaggered-Grid Finite Difference Method on Curvilinear Grid, submitted to GJI.

Zhang, W., Y. Shen, and X.F. Chen (2008), Numerical simulation of strong ground motion for the Ms 8.0 Wenchuan earthquake of 12 May 2008, Science in China Series D-Earth Sciences, **51(12)**, 1673-1682.

Zhang, W., Y. Shen, and L. Zhao (2011), 3D Seismic Wave Modeling in Spherical Coordinate by a Nonstaggered-grid Finite Difference Method on Nonuniform Grids, manuscript in preparation.

Zhang, Z. and Y. Shen (2008). Cross-dependence of finite-frequency compressional waveforms to shear seismic wave-speeds, Geophys. J. Int., 174, 941-948, doi:10.1111/j.1365-246X.2008.03840.x.

Zhang, Z., Y. Shen, and L. Zhao (2007). Finite-frequency sensitivity kernels for head waves, Geophys. J. Int. 171, 847-856.

Appendix

Data Processing Scripts

./csh/sort_stn_list.csh

```
#!/bin/csh
# select stations from a list of available stations, Y.Shen

# last modified, Y.Shen, 03-01-2011

set user = seismo_user
set area = easthemi
set wkdir = /home/$user/$area
# stnlst from IRIS DMC
set stnlst = $wkdir/STinfo/Lat45s-55n.Lon30w-155e.90-10.stns.lst
# cmplst from "gmap"
set cmplst = $wkdir/STinfo/Lat45s-55n.Lon30w-155e.90-10.VHZ.lst
# selected stations
set cmponly = $wkdir/STinfo/$area.stns.VHZ.lst

cat /dev/null  > $cmponly

foreach stn (`cat $cmplst | awk '{print $1}'`)
echo $stn
set ntwk = `echo $stn | awk -F. '{print $1}'`
set stnm = `echo $stn | awk -F. '{print $2}'`
cat $stnlst | grep $ntwk | grep $stnm >> $cmponly
end
```

./csh/mk_yearly_req.csh

```
#!/bin/csh
# generate breq_fast requests to be emailed to BREQ_FAST@dmc.iris.washington.edu,
Y.Ren

# last modified Y.Shen, 03-01-2011

set reqdate = `date +%Y.%m.%d`
set reqid = "EH".$reqdate
echo $reqid >! ../data_reqs/yearly/lastreqid

set area = easthemi
set stnlst = $area.stns.VHZ.lst

# 10 s before the year
set month1 = 12
set day1 = 31
set hour1 = 23
set min1 = 59
set sec1 = 50

# for broadband at 40 sps
#set month2 = 07
# for long period at 1 sps
set month2 = 01
set day2 = 01
set hour2 = 00
set min2 = 00
# set sec2 = 00
# 10 s after the year
set sec2 = 10

@ nc = 0
foreach station (`cat ../STinfo/$stnlst | awk '{print $1"."$2}'`)

@ nc = $nc + 1
set ntk=`echo $station | awk -F. '{print $1}'`
set stn=`echo $station | awk -F. '{print $2}'`
set byear = `head -$nc ../STinfo/$stnlst | tail -1 | awk '{print substr($3,1,4)}'`
set eyear =  `head -$nc ../STinfo/$stnlst | tail -1 |  awk '{print substr($5,1,4)}'`

echo $ntk $stn $byear $eyear
set ntkstn = $station
```

```
foreach year ( 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
2003 2004 2005 2006 2007 2008 2009 2010 )
@ year1 = $year - 1
@ year2 = $year + 1
if ( $year >= $byear & $year <= $eyear ) then

#foreach chan ( BHZ )
foreach chan ( LHN LHE )
set yearlyreqname01 = "$reqid.$ntkstn.$year.$chan.req"
#set yearlyreqname07 = "$reqid.$ntkstn.$year.$chan.$month2.req"
echo ".NAME Yang Shen" >! $yearlyreqname01
echo ".INST University of Rhode Island" >> $yearlyreqname01
echo ".MAIL South Ferry Road, Narragansett, RI 02882" >> $yearlyreqname01
echo ".EMAIL yshen@gso.uri.edu" >> $yearlyreqname01
echo ".PHONE 401 874-6848" >> $yearlyreqname01
echo ".FAX 401 874-6811" >> $yearlyreqname01
#/bin/cp $yearlyreqname01 $yearlyreqname07

echo ".LABEL $reqid.$ntkstn.$year.$chan" >> $yearlyreqname01
echo ".MEDIA ELECTRONIC" >> $yearlyreqname01
echo ".ALTERNATE MEDIA DAT" >> $yearlyreqname01
echo ".END" >> $yearlyreqname01
echo " " >> $yearlyreqname01
echo $stn $ntk $year1 $month1 $day1 $hour1 $min1 $sec1 $year2 $month2 $day2
$hour2 $min2 $sec2 1 $chan
echo $stn $ntk $year1 $month1 $day1 $hour1 $min1 $sec1 $year2 $month2 $day2
$hour2 $min2 $sec2 1 $chan >> $yearlyreqn
ame01

#echo ".LABEL $reqid.$ntkstn.$year.$chan.$month2" >> $yearlyreqname07
#echo ".MEDIA ELECTRONIC" >> $yearlyreqname07
#echo ".ALTERNATE MEDIA DAT" >> $yearlyreqname07
#echo ".END" >> $yearlyreqname07
#echo " " >> $yearlyreqname07
#echo $stn $ntk $year1 $month2 $day2 $hour2 $min2 $sec2 $year2 $month1 $day1
$hour1 $min1 $sec1 1 $chan
#echo $stn $ntk $year1 $month2 $day2 $hour2 $min2 $sec2 $year2 $month1 $day1
$hour1 $min1 $sec1 1 $chan >> $yearlyreq
name07

end # of foreach chan
endif

end # of foreach year
```

```
end # of foreach station

/bin/mv *.req ../data_reqs/yearly
date
```

./send_allreq.csh

```
#!/bin/csh
# send breq_fast requests to IRIS DMC, Y. Ren

# last modified, Y.Shen 03-01-2011

# user needs to modify SMTP username and password and sender's email address

cd ../data_reqs/yearly

set reqid = `cat lastreqid`
set files=`ls -f $reqid.*.req`

foreach file ($files)
echo $file

echo "import smtplib" >!  send.py
echo "" >>  send.py
echo "smtpserver = "\'"smtpserv.gso.uri.edu"' >>  send.py
echo "AUTHREQUIRED = 0 # if you need to use SMTP AUTH set to 1" >>  send.py
echo "smtpuser = "\'"your_user_name"\'"  # for SMTP AUTH, set SMTP username
here" >>  send.py
echo "smtppass = "\'"your_pass_word"\'"  # for SMTP AUTH, set SMTP password
here" >>  send.py
echo "" >>  send.py
echo "RECIPIENTS = ["\'"breq_fast@iris.washington.edu"\'"]" >>  send.py
echo "SENDER = "\'"your_email_address"\' >>  send.py
echo "mssg = open("\'$file\'", "\'"r"\'").read()" >>  send.py
echo "" >>  send.py
echo "session = smtplib.SMTP(smtpserver)" >>  send.py
echo "if AUTHREQUIRED:" >>  send.py
echo "    session.login(smtpuser, smtppass)" >>  send.py
echo "smtpresult = session.sendmail(SENDER, RECIPIENTS, mssg)" >>  send.py
echo "" >>  send.py
echo "if smtpresult:" >>  send.py
echo "    errstr = "\"\" >>  send.py
echo "    for recip in smtpresult.keys():" >>  send.py
echo "        errstr = "\"\"\""Could not delivery mail to: %s" >>  send.py
echo "" >>  send.py
echo "Server said: %s" >>  send.py
echo "%s" >>  send.py
echo "" >>  send.py
echo "%s"\"\"\"" % (recip, smtpresult[recip][0], smtpresult[recip][1], errstr)" >>  send.py
echo "    raise smtplib.SMTPException, errstr" >>  send.py
```

```
python send.py
sleep 5

end
```

./csh/fetch_seed.csh

```
#!/bin/csh
# fetch SEED files at the IRIS DMC, Y.Shen

# Note: Change "your_ftp_directory" at ftp.iris.washington.edu
#       Change "your_email_address"

# last modified, Y.Shen 03-01-2011

set user = seismo_user
set area = easthemi
set proj_hm_dir = /home/$user/$area
set disk = /scratch
set wkdir = $disk/$user/$area/seed_data
# ftp_seeds.lst comes from the user directory at ftp.iris.washington.edu
set seedlst = $wkdir/ftp_seeds.lst

set seedtag = `cat $proj_hm_dir/data_reqs/yearly/lastreqid`
echo $seedtag

echo "seedid (e.g., 2007.BHN.01, 2007.BHN, 2007, BHN)?"
set seedid = $<

set datetag = `date "+ %y.%m.%d.%H.%M" | awk '{print $1}'`

set script = "ftp_script_"$datetag".csh"
echo "" > $script
echo "#script to fetch seed files from iris.dmc" >> $script
echo "#generated by fetch_seed.csh" >> $script
echo "cd $wkdir" >> $script
echo "ftp 128.95.166.129 <<EOF" >> $script
echo "anonymous" >> $script
echo " " >> $script
echo "cd pub/userdata/your_ftp_directory" >> $script
echo "bin" >> $script
echo "prompt" >> $script

@ nfile = 0
foreach seedfile (`cat $seedlst | grep $seedtag | grep $seedid | awk '{print $9}'` )
set targetfile = $wkdir/$seedfile
if ( ! ( -e $targetfile ) & $nfile < 21 ) then
echo $seedfile
echo "mget $seedfile" >> $script
@ nfile = $nfile + 1
```

```
endif
end

echo "quit" >> $script
echo "EOF" >> $script

echo "ftp $seedid batch job done!" > ftp.$seedid.message
echo "mail your_email_address < ftp.$seedid.message" >> $script

chmod +x $script

# run script to fetch the seed files
./$script

# check if the selected files are indeed fetched
@ missing = 0
foreach seedfile (`cat $seedlst | grep $seedtag | grep $seedid | awk '{print $9}'` )
set targetfile = $wkdir/$seedfile
if ( ! ( -e $targetfile ) ) then
@ missing = $missing + 1
endif
end

if ( $missing == 0 ) then
echo "ftp $seedid job finished!" > job_message
else
echo "$missing seed files unfetched" > job_message
endif
mail your_email_address < job_message

/bin/rm job_message
```