Manual of ASWMS

Ge Jin[1] and James Gaherty[1]

[1]Lamont Doherty Earth Observatory, Columbia University

May 13, 2014

1 Introduction

This package can be used to measure the phase and amplitude of surface waves from the raw waveforms, and generate phase velocity maps via Eikonal and Helmholtz equations. The detailed theoritical development of this package can be found in Jin and Gaherty, 2014. In this manual, the names of files and folders are written in blue and the names of variables are written in orange.

All the adjustable parameters in this package are stored in the setup_parameters.m, and the input data should be in the folder eventmat. Here we present some details on how to set up this system on your own computer.

The program is written in Matlab, and require these toolboxes:

```
Curve Fitting Toolbox
Signal Processing Toolbox   Statistics Toolbox
Mapping Toolbox
Statistics Toolbox
```

The Mapping Toolbox is only needed if you want to plot the results using Matlab, and the Statistics Toolbox can be replaced by adding the NaN Suite in the program: `http://www.mathworks.com/matlabcentral/fileexchange/6837-nan-suite/content/nansuite`.

2 Data Preparation

2.1 Data Structure

The data format that this program reads is a event-based matlab structure, and should be saved in the folder named eventmat. The file name should follow the rule of YYYYMMDDhhmm_comp.mat, for example: 200901081921_LHZ.mat, and the structure name has to be event.

Here shows an example of this structure:

```
event =
          evla: 10.2152
          evlo: -84.2159
         otime: 6.3399e+10
            id: '200901081921'
     otimestr: '08-Jan-2009 19:21:34'
      stadata: [1x654 struct]
```

In which the field otime is the number of seconds to a certain time. The actual time is not critical, we used 1970-01-01 in the sac version, and 0000-01-01 in the IRIS version, but it has to be consist in the whole program.

The field stadata contains the waveform and megadata from each stations, here is an example:

```
event.stadata(1) =

        stla: 32.8920
        stlo: -116.4223
        stel: 1.8750
        dist: 4.1447e+03
       otime: 6.3399e+10
       delta: 1
        data: [7200x1 double]
         cmp: 'LHZ'
        stnm: 'MONP2'
```

The stadata.otime has to have the same origin time as the event.otime, and has to have the unit in second.

Besides writing your own code to transfer your data into the matlab structure, we provide two alternative ways.

2.2 Using SAC files

The script sac2eventmat.m can be used to transfer SAC files into the eventmat files.

The input SAC files have to be stored in the folder sacdata/. Each event should have its own folder with the folder's name as YYYYMMDDhhmm. The file name should include the components name defined in setup_parameters.m and end with ".sac". Here shows an example:

```
>>ls sacdata
200801011855
200801051101
200801051144
200801070312
...
```

```
...
>>ls sacdata/200801011855
200801011855.TA.G09A.LHZ.sac
200801011855.TA.G10A.LHZ.sac
200801011855.TA.G11A.LHZ.sac
200801011855.TA.G14A.LHZ.sac
...
...
```

The origin time of each sac file has to be the origin time of the earthquake, and the waveform starting time can be different. Each sac file has to have the following information in its header:

Event information: NZYEAR NZHOUR NZMIN NZSEC NZMSEC EVLA EVLO EVDP

station information: STLA STLO STEL KSTNM

data information: B DELTA KCMPNM

A file named eventlist with all the names of event folders should be generated in the sacdata folder as well. An example of this file is:

```
>>cat sacdata/eventlist
200801011855
200801051101
200801051144
200801070312
...
...
```

You can easily generate this file by using the pipe function of the shell scripts.

2.3 Using IRIS DMC Service

You can directly download the waveform from IRIS DMC by using the matlab script data_download.m. This script utilizes the DMC's matlab interface irisFetch.m, and for more details of this service please visit IRIS website:

`http://www.iris.edu/dms/nodes/dmc/software/downloads/irisFetch.m/2-0-6/`

Or Google: irisFetch.

The following parameters in the setup_parameters.m should be adjusted if you want to use this service.

```
% parameters for data downloading (if using IRIS DMC)
parameters.start_time = '2009-01-07 00:00:00';
parameters.end_time = '2009-06-08 00:00:00';
parameters.is_use_timestamp = 1;
```

```
parameters.network = '_US-ALL';
parameters.minMw = 6;
parameters.maxdepth = 50;
parameters.datalength = 7200;  % in second
```

If the parameters.end_time is empty (''), then the end time of the data fetching will be 4 days before the current date. If the is_use_timestamp is 1, then after each successful run of the data_download.m, it will save the end time of this run into a mat file named tempstamp.mat. So the next time data_download.m is run, it will start from where the last run left. This option is useful for set up a self-updated system.

The downloaded data are stored in the folder datacache first before they are further processed. It will skip the files already existing in this folder to avoid repeatly download and save running time (actually, most of the scripts in this program have this feature). Then the station responses are removed and the data are organized and transformed into the event matlab structure in the folder eventmat.

3 Parameter Adjustment

All the adjustable parameters are defined in the different sections in setup_parameters.m, and most of them are under-hood and do not need to be changed for most of the teleseismic based projects.

The key parameters that should be changed each time and the values chosen for the USArray project are:

```
parameters.proj_name = 'TA';
parameters.component = 'LHZ';    % determined by filenames
parameters.lalim=[25 50];
parameters.lolim=[-125 -65];
parameters.gridsize=0.3;    % in degrees
```

where the proj_name can be any string. The component should be included in your sac file names (if you are using sac files to input), and it will also effect the names of the output files. It is recommanded to be "LHZ" for teleseismic projects. lalim, lolim and gridsize are the defination of the tomorgraphy grid system. The final grid of the output is defined as:

```
xnode = lalim(1):gridsize:lalim(2);
ynode = lolim(1):gridsize:lolim(2);
[xi yi] = ndgrid(xnode,ynode);
```

Another group of parameters that frequently need to be adjusted are the periods and the smoothing weight in each period (smweight_array). The periods are defined as:

```
parameters.periods = [20 25 32 40 50 60 80 100];  % in seconds
```

which is the central frequencies of the narrow-band filters. The width of the narrow-band filters around 10% of the central frequency and are defined by min_width and max_width.

And the smoothing weight is defined as:

```
parameters.smweight_array = 3*[0.4 0.3 0.2 0.2 0.2 0.5 1 2];
```

It should have the same length as periods, with smallest value for the frequency bands with high SNR and short wavelength. The example shows a good ratio for the periods listed above, and you may only need to change the constant (3 in this case) for different project.

For most projects, changing these parameters should be able to provide you a good initial result. We will discuss other adjustable parameters in the later sections.

4 Run the Scripts

After you have prepared the data and adjust the parameters, you can start to run this program and get an initial result. The main sequence of the commands is listed in main_driver.m. You can go ahead and run it after you modify it by choosing one of the data preparation methods.

Here we briefly list these commands and describe the their function and input/output. We will discuss them in details in the later sections.

1. sac2eventmat.m/data_download.m
   choose one of your data preparation method, or build your own.
   Input: sac files/NaN Output: eventmat/*.mat

2. cleanup_events.m
   This script find the events in the eventmat folder that are close in time and their surface wave may interference with each other at the array location. Then the eventmat files of these events are deleted.
   Input: eventmat/*.mat Output: NaN

3. run_autowinpick.m
   This script will define the window function to isolate the energy of surface waves. It will create a new field winpara in the event structure, and also create a ASCII file in the folder winpara. If you want the window to be recalculated, you need to delete the files in winpara.
   Input: eventmat/*.mat Output: winpara/* eventmat/*.mat

4. gsdfmain.m
   This script measure the phase delay between nearby stations using cross-correlation. It is the core part of the whole program. For each event, it will generate an structure named eventcs and save it to the folder CSmeasure.
   Input: eventmat/*.mat Output: CSmeasure/*.mat

5. eikonal_eq.m
   This script will do the tomography inversion via Eikonal equation, based on the cross-correlation measurement. It will generate a structure named eventphv, which contains the apparent phase velocity maps for each event and save it to the folder eikonal.
   Input: CSmeasure/*.mat Output: eikonal/*.mat

6. stack_phv.m
   This script stack all the events existing in the folder eikonal, output a structure avgphv containing stacked apparent phase velocity map, and save it to the file eikonal_stack_LHZ.mat. This is the final result if the amplitude correction is not applied.
   Input: eikonal/*.mat Output: eikonal_stack_LHZ.mat

7. helmholtz_eq.m
   This script applys amplitude correction on the apparent phase velocity via Helmholtz equation. It generates a structure named helmholtz and save to the folder helmholtz.
   Input: eikonal/*.mat CSmeasure/*.mat Output: helmholtz/*.mat

8. stack_helm.m
   This script stacks the corrected phase velocity maps from each event and generate the final result, as a structure named avgphv and save it to the file helmholtz_stack_LHZ.mat.
   Input: helmholtz/*.mat Output: helmholtz_stack_LHZ.mat

5 Output Format

Depends on whether the amplitude correction is applied, the final result is stored either in eikonal_stack_LHZ.mat or helmholtz_stack_LHZ.mat. However, the structures avgphv in these two files have the same structure, only the one in eikonal_stack_LHZ.mat does not contain the correction related fields. Here we only present the structure in the helmholtz_stack_LHZ.mat.

Here shows an example of avgphv:

```
avgphv(4) =

          sumV: [84x201 double]
       sumV_cor: [84x201 double]
     sumweight: [84x201 double]
        GV_std: [84x201 double]
    GV_cor_std: [84x201 double]
      eventnum: [84x201 double]
            xi: [84x201 double]
            yi: [84x201 double]
         xnode: [1x84 double]
         ynode: [1x201 double]
        period: 40
        GV_cor: [84x201 double]
            GV: [84x201 double]
```

where the important fields are:

xi Latitude of the grid.

yi Longitude of the grid.

GV Phase velocity before amplitude correction.

GV_cor Phase velocity after amplitude correction.

eventnum Number of events each grid stacked.

There are map plotting scripts contained in the stack_phv.m and stack_helm.m, which can be actived by changing the variable isfigure to 1 at the beginning of these two files. You can also refer them to plot your own figures. Here shows an simple example of plotting the result:

```
load helmholtz_stack_LHZ.mat
ip = 4;  % plot the 40s result
figure(88)
clf
ax = worldmap(lalim,lolim)
surfacem(avgphv(ip).xi,avgphv(ip).yi,avgphv(ip).GV_cor);
% set the color scale
cmap = colormap('jet');
cmap = flipud(cmap);
colormap(cmap);
% set the color range
r = 0.1;  % 20 % peak to peak
meanphv = nanmean(avgphv(ip).GV_cor(:));
caxis([meanphv*(1-r) meanphv*(1+r)]);
colorbar
```

6 Technical Details

In this section we will discuss the technical details in each component of the program, and provide the users more freedom to customize the program to fit their own projects.

To be continued.....