

# Manual of ASWMS

Ge Jin<sup>1</sup> and James Gaherty<sup>1</sup>

<sup>1</sup>Lamont Doherty Earth Observatory, Columbia University

May 13, 2014

## 1 Introduction

This package is developed to measure the phase and amplitude of surface waves from the raw waveforms, and to generate phase velocity maps via Eikonal and Helmholtz equations. The detailed theoretical development of this package can be found in Jin and Gaherty, 2014.

From Section 2 to Section 5 we present you a quick work flow to set up this system on your computer, and adjust the parameters to fit your project. This setup should give a good initial result for most teleseismic studies with the array size ranging from 100 km to continual scale, and with the station spacing smaller than 100 km. You can refer to Section 6 if you want to make more customized adjustment for better result.

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 put in the folder **eventmat**.

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

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

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 input data are event-based matlab structure saved in mat files, which should be put in the folder `eventmat`. The filename for each event should follow the rule of `YYYYMMD-Dhhmm.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, which 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 project.

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. The `stadata.dist` has the unit of km, and `delta` is the sample interval in second.

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

### 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, while the waveform starting time can be different. Each sac file should 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](#)

An ASCII file named [eventlist](#) with all the names of event folders should be put 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 shell pipe function.

## 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](#). 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 triggered, it will start from where the last run left. This option is useful for setting up a self-updated system.

The downloaded data are stored in the folder [datacache](#) first before they are further processed. It skips the files already existing in this folder to avoid repeat 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 the [setup\\_parameters.m](#), and most of them are under the hood and do not need to be changed for most of the teleseismic based projects.

Here is a list of the key parameters that should be changed for each project, with their values chosen for the USArray project:

```
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 the sac filenames (if you are using sac as input), and it is also shown in the names of the output files. It is recommended to be "LHZ" for teleseismic Rayleigh-wave projects. The [lalim](#), [lolim](#) and [gridsize](#) are the definition of the tomographic grid system. The final output grid will be 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 are 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 for each period 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 smaller value for the frequency bands with higher SNR or 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 most projects, changing these parameters should be able to provide you a good initial result. We will discuss other adjustable parameters in the Section 6.

## 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 their function and input/output. We will discuss them in details in the Section 6.

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 creates a new field **winpara** in the **event** structure, and also generates 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 measures the phase delay between nearby stations using cross-correlation. It is the core part of the entire program. For each event, it generates an structure named [eventcs](#) and saves it to the folder [CSmeasure](#).

Input: eventmat/\*.mat Output: CSmeasure/\*.mat

5. [eikonal\\_eq.m](#)

This script performs the tomography inversion via Eikonal equation, based on the cross-correlation measurements. It generates 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 stacks all the events in the folder [eikonal](#), output a structure [avgphv](#) containing stacked apparent phase velocity map, and saves it to the file [eikonal\\_stack\\_LHZ.mat](#). It would be the final result if the amplitude correction is not applied.

Input: eikonal/\*.mat Output: eikonal\_stack\_LHZ.mat

7. [helmholtz\\_eq.m](#)

This script applies the amplitude correction on the apparent phase-velocity results via Helmholtz equation. It generates a structure named [helmholtz](#), which is stored in the folder [helmholtz](#).

Input: eikonal/\*.mat CSmeasure/\*.mat eikonal\_stack\_LHZ.mat Output: helmholtz/\*.mat

8. [stack\\_helm.m](#)

This script stacks the corrected phase velocity maps from each event and generates 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

Depending 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: [84 x201 double]
sumV_cor: [84 x201 double]
sumweight: [84 x201 double]
GV_std: [84 x201 double]
GV_cor_std: [84 x201 double]
eventnum: [84 x201 double]
xi: [84 x201 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 activated 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 added. ....

## 7 Reference

Jin, G., and J. B. Gaherty (2014), Surface Wave Measurement Based on Cross-correlation , Geophys. J. Int, submitted.