Manual of JDSurfG

Nanqiao Du
Version 3.0

August 17, 2021

Contents

1 Introduction

This document describes how to use `JDSurfG` package to perform 3-D joint inversion of shear wave velocity in crust and upper mantle by surface wave dispersion and gravity anomaly data. This package is consisted of three independent modules: generating gravity matrix in spherical coordinates, conducting direct surface wave tomographic inversion, and performing tomographic joint inversion of dispersion and gravity anomaly data. In joint inversion, we utilize direct surface wave tomography method [Fang et al., 2015] to compute 3-D (pseudo) sensitivity kernel of surface waves traveltime, and the adaptive Gauss-Legendre integration method [Li et al., 2011] is adopted to calculate the gravity forward matrix. Based on empirical relations between seismic velocity and density of rocks [Brocher, 2005], this package would be a useful tool to investigate 3-D shear wave structure in the crust and upper mantle. For more details, please refer to the paper [Du et al., 2021]

This package is written by `C++` and `Fortran`. `C++` is responsible for the main logic such as handling I/O, providing useful data structures and interfaces. And the `Fortran` part, derived from package DSurfTomo with several important modifications (by adding parallel mode and analytical derivatives), is used to compute surface wave traveltimes and corresponding 1D/3D sensitivity kernels.

I should note that this package was only tested by several people on similar computational conditions, so it might contain some bugs in it. I'd appreciate to receive bug reportings and modify some part of it when obtaining good suggestions. In the long term, I would add some new functions to this package, you could see the `TODO LIST` in `README`.

2 Preliminaries

This package utilizes `C++` library Eigen to handle multi-dimensional arrays, and GSL to compute Gauss-Legendre nodes' locations and weights. So you need to install these two libraries in your own computer before compiling this package. Also, because some of this package depend on template expressions, you should make sure that your `C++` compiler support `C++11` standards (`GCC` ¿= 4.8).

3 Installation

After you have downloaded the .zip file from Github, you can unzip it by using:

```
unzip JDSurfG.zip
```

Before preceding to compile this package, you should add the Include path of `Eigen` and `GSL` and the Library path of `GSL` in `JDSurfG/include/Makefile` if you don't modify your `~/.bashrc` when you install these two libraies. For `Eigen`, you could add include path as

```
EIGEN_INC = -I/path/to/your/eigen
```

For `GSL` library, a convinient way is to use `gsl-config` in the `bin` directory in the `GSL` Installation path:

```
GSL_LIB = $(shell gsl-config --libs)
GSL_INC = $(shell gsl-config --cflags)
```

Then compile this package by:

```
cd JDSurfG
make
```

Then you will find four excecutable files in the directory `bin`:

```
mkmat DSurfTomo JointSG syngrav
```

That means you have finished installation.

4 Direct Surface Wave Tomography Module

This module need 3 (4 if checkerboard or other test is required) input files:

- `DSurfTomo.in`: contains information of input model, dispersion data and inversion parameters.
- `surfdataSC.dat`: dispersion data for each station pair
- `MOD`: Initial model
- `MOD.true`: True model (if additional tests are required)

There are some points to note before we precede to the format of each file:

(1) Input model is homogeneous in latitudinal and longitudinal direction, but the grid size could vary in depth direction.

(2) The input model should be slightly larger than the target one in order to perform `B-Spline` smoothing for forward computation. To be specific, if our target model is in region $(lat_0 \sim lat_1, lon_0 \sim lon_1, 0 \sim dep_1)$, you should edit your input model at least in region $(lat_0 - dlat \sim lat_1 + dlat, lon_0 - dlon \sim lon_1 + dlon, 0 \sim dep_1 + dz)$.

(3) Stations should be far away from target model boundaries, or some warnings will be given when you running this package. This is to avoid that surface wave train travels along the models' boundaries.

Now there are the formats of every file below:

4.1 Parameter File

The parameter file is called `DSurfTomo.in` in this instruction. It is a self-explanatory file. Here is a template of this file below:

```
#################################
# DSurfTomo Program Parameters
#################################

# input model description:
33 36 25         # nlat nlon nz (no. of points in lat lon and depth direction)
35.3 99.7        # lat0 lon0 (upper left point,[lat,lon], in degree)
0.3 0.3          # dlat dlon (grid interval (degree) in lat and lon direction)

# dispersion forward/adjoint computation parameters
# Insert 'ninsert' points in the grid-based model
# to compute dispersion curves
3                # ninsert (insert several layers, 2~5)
8                # no.of threads used

# inversion parameters
20.0 1.0         # smooth damp
2.0 4.7          # minimum velocity, maximum velocity (a priori information)
10               # maximum iteration
2                # no. of threads used in Linear System Solver

# dispersion data description
16               # kmaxRc (followed by periods)
4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0 22.0 24.0 26.0 28.0 30.0 32.0 34.0
16               # kmaxRg (followed by periods)
4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0 22.0 24.0 26.0 28.0 30.0 32.0 34.0
0                # kmaxLc (followed by periods)
0                # kmaxLg (followed by periods)

# synthetic test
1                # synthetic flag (0: real data,1: synthetic)
3.5              # noise level, std value of gaussian noise
```

When you run this program, it will automatically skip all the empty lines and comments (denoted by '#'). So you could modify this file (by adding your own comments) as long as you don't change the order of these parameters. Now let's see the meaning of each parameter in the template file.

1. `Input Model Description Block`. This block contains information of input model. The first line contains three integers which denote the number of grid points in latitude,longitude,and in depth respectively. The second

line denotes coordinates of the origin point (NorthWest point), in degree. The last line denotes the grid interval in latitudinal and longitudinal direction, both in degree.

2. `Forward/Adjoint Computation Block`. This block contains parameters for forward/adjoint computation. The first line is an integer we call it "ninsert". As metioned above, we use grid nodes to parameterize our model and thus all physical parameters are defined directly at grid points. However, in order to compute surface wave dispersion by Thomas-Haskell Matrix Method [Haskell, 1953, Schwab and Knopoff, 1972], we need to convert grid-based model to layer-based one. In that case, we insert 'ninsert' points between adjacent points to do that conversion. The next integer is how many threads you want to use in calculation of dispersion curves and do ray tracing with Fast Marching Method [Rawlinson and Sambridge, 2004].

3. `Inverse Parameters Block`. This block contains all parameters used in the inverse problem. The first line indicates the smoothing and damping factor in solving the linear systems. Then the followed line is the restriction of your inverted model (the minimum and maximum velocity). The next line is an integer which is the maximum iterations you want to iterate. The last line is the number of threads used in solving linear systems, usually 2 is enough.

4. `Dispersion Data Block`. This block describes the information of your dispersion data. The first line is the number of periods your Rayleigh wave phase velocity disperion data spans. If this number is not zero, then you should write all the periods in the next line. Other lines are for Rayleigh group dispersion, Love phase, and Love group respectively.

5. `Synthetic Test Blcok`. This block contains all parameters used in synthetic test. The first one is a flag variable to denote whether you will conduct synthetic test when using this program. The next line is the noiselevel of gaussian noise, defined as below:

$$d^i_{obs} = d^i_{syn} + noiselevel * N(0,1)$$

4.2 Initial Model and True Model File

`Please Note that the format of MOD.true in current version is different from previous versions!`

The initial and true model are called `MOD` and `MOD.true` respectively in this instruction. They are used for initial input model, and for synthetic test dataset (like in checkerboard test). The format of these two files are listed below:

The first line of each files are depths (in km) of each grid points, i.e. there would be `nz` numbers in this line. The subsequent lines of `MOD` are shear wave velocities. If we store the velocity with the format V(nz,nlon,nlat) (Row-major,where $V(0,0,0)$ is the shear-wave velocity of the `top NorthWest` point at `dep(0)` km, and V(nz-1,nlon-1,nlat-1) is the velocity of `bottom SouthEast` point at dep(nz-1) km), then it should be print to MOD as following:

```
//c++
for(int i=0;i<nz;i++){
for(int j=0;j<nlon;j++){
    for(int k=0;k<nlat;k++){
        fprintf( fileptr ,"%f ",V(i,j,k));
    }
     fprintf( fileptr ,"\n");
}}
```

4.3 Dispersion Data File

This dispersion data in `example/` is surfdataSC.dat, which contains dispersions (distance/traveltime) obtained surface wave or cross-correlation functions
Here we list its first 9 lines:
# 31.962600 108.646000 1 2 0
33.732800 105.764100 3.0840
34.342500 106.020600 3.1154
33.357400 104.991700 3.0484
33.356800 106.139500 3.0820
34.128300 107.817000 3.1250
33.229300 106.800200 3.0575
# 29.905000 107.232500 1 2 0
34.020000 102.060100 2.7532

Obviously, this file contains two kinds of infomation: surface wave source or master station (lines begin with `#`) and receiver stations (without `#` in each line). For a source or master station, the format of this line is:

```
# latitude longitude period-index wavetype velotype
```

- `wavetype`: the type of surface wave, for Rayleigh it is 2, and it is 1 for Love wave.
- `velotype`: velocity type, 0 for phase velocity and 1 for group velocity.
- `period-index`: The period index number.

For example, if in our dataset, the dispersion periods for Rayleigh phase velocity is at periods 4s,5s,6s,7s, and this line contains information only at 5s, then the last 3 numbers are `2,2,0`.

Then the following lines contain all the receiver stations that you have successfully extracted dispersion curves at this period, this wave type and this velocity type. And the format of each line is:

`latitude longitude velocity`

4.4 Run This Module

After you have prepared all the required files in a folder, then you could enter into this folder,and run this command in your shell:

```
./DSurfTomo −h
```

Then you could type in all required files according to the order on the screen.

4.5 Output Files

This module will output two (temporary) folders: `kernel/` and `results/`. `kernel` is a tempoaray folder , and it contains 3-D pseudo sensitivity kenrels of surface wave travel times. `results/` contains all the intermediate models after each iteration. Here are information about the format of these files:

- `kernel/`: 3-D pseudo sensitivity kernel. The number of txt files is equal to no. of threads used in your program (see 4.1), and start from 0. The kernel matrix is Compressed Sparse Row matrix. It looks like:

  ```
  # 0 1283
  24292 2.18522
  24293 2.10063
  24322 2.29456
  24323 2.90529
  24324 2.77207
  24353 2.45755
  24354 3.14743
  24355 2.99618
  24384 2.14853
  ```

  Note there are two numbers followed '#'. The first one is the row-index, and another one is the no. of nonzeros in this row (1283 for current case). Then the following (1283) rows are column-index and matrix elements. It should be noted that this folder is only a temporary folder in the inversion process, and if will be deleted after each iteration.

- `results/`:contains models (`mod_iter*`) and surface wave travel time (`res*.dat`) of each iteration. The format of model files is:

  `longitude latitude depth Vs`,

  And the format of traveltime file is (9 columns):

  `distance observation synthetic lat1 lon1 lat2 lon2 period wavetype`.

5 Joint Inversion Module

This module needs 4-6 files in total:

- `JointSG.in`: contains information of input model, dispersion data and inversion parameters.
- `surfdataSC.dat`: surface wave dispersion data.
- `obsgrav.dat`: Gravity anomaly dataset.
- `gravmat.dat`: sensitivity kernel of gravity to density.
- `MOD`: Initial model.
- `MOD.true`: checkerboard model.
- `MOD.ref`: gravity reference model, used for compute gravity anomaly.

Here are the formats of each file.

5.1 Parameter File

The difference between `JointSG.in` and `DSurfTomo.in` is little except last three blocks:

```
# synthetic test
1                # synthetic flag (0: real data,1: synthetic)
3.5 0.05         # noiselevel for traveltime and Bourguer

# weight parameter
0.5              # parameter p (Julia(2000)) for 2 datasets, 0-1
3.5 0.05         # std parameters for two datasets

# gravity processing
# whether to remove average value of synthetic gravity data
0                # 1 remove, 0 not remove
```

In order to know the meaning of each parameter in `Weight Parameters Block`, here I list the weighted cost function of joint inverse problem [Julià et al., 2000]:

$$L = \frac{p}{N_1\sigma_1^2}\sum_{i=1}^{N_1}(d_{1,i} - d_{1,i}^o)^2 + \frac{1-p}{N_2\sigma_2^2}\sum_{i=1}^{N_2}(d_{2,i} - d_{2,i}^o)^2 \tag{1}$$

where $\mathbf{d}_{1,2}$ is the data vector for each dataset. $N_{1,2}$ is the size of each dataset and $\sigma_{1,2}$ is the corresponding errors. p is a parameter in the range [0,1] to control the contribution of each dataset manually.

The additional `gravity processing block` contains only one integer. When this integer is equal to 1, the program will automatically remove the average value of observed and synthetic gravity data. This option is applied to remove large-scale density anomalies in the deep mantle. For more details please refer to 5.4.

5.2 Gravity Data File

The format of this file is quite simple, just print all the gravity data to this file. Each line of it contains the longitude,latitude and gravity anomaly at this point.

Here I list the first 9 lines of this file to give you an example:

```
100.000000 35.000000 -224.206921
100.000000 34.950000 -224.110699
100.000000 34.900000 -241.774731
100.000000 34.850000 -240.245968
100.000000 34.800000 -234.187542
100.000000 34.750000 -246.670605
100.000000 34.700000 -238.575939
100.000000 34.650000 -241.357250
100.000000 34.600000 -260.059429
```

5.3 Gravity Matrix File

This file is generated by gravity module, please refer to chapter 6.2.

5.4 Gravity Reference Model File

Based on the definition of gravity anomaly, it reflects the density perturbance according to a normal elliptoid. Thus we need to convert the absolute anomalies to local anomalies in our research area. Before inversion, we will remove the average value of oberserved anomalies. So in the real data invesion, there are 3 steps to compute relative anomalies:

- Compute density distribution of current S-wave model and reference model (through empirical relations).
- Compute density anomaly, then utilize gravity matrix to get gravity anomalies.
- Remove average value of the synthetic anomaly data.

This model could be a user self-defined model, and you could also use the default one (the average of initial model in every depth). Please remember, when you are tackling real data, to set the integer to 1 in the gravity processing block in `JointSG.in` file.

5.5 Run This Module

After you have prepared all these required files in a folder, then you could enter into this folder, and run this command in your shell:

```
./JointSG -h
```

Then you could type in all required files according to the order on the screen.

5.6 Output Files

This module will output two temporary folders: `kernel/` and `results/`.

The format of `kernel/` is the same as the files in 4.5. In the folder `results/`, it contains models (`joint_mod_iter*`), surface wave travel time (`res_surf*.dat`) and graivty anomaly (`res_grav*.dat`) of each iteration. The format of gravity anomaly is:

```
lon lat observed-anomalies synthetic-anomalies
```

6 Gravity Module

This module requires three input files:

- `JointSG.in` or `DSurfTomo.in`: contains model information
- `MOD` : only first line of it are needed.
- `obsgrav.dat`: contains coordinates of each obeservation points.

These formats have been illustrated in previous chapters.

6.1 Run This Module

After you prepare all these required files in a folder, then you could enter into this folder,and print in your shell:

```
./mkmat -h
```

Then you could type in all required files according to the order on the screen.

6.2 Output Files

This module output only one file: `gravmat.dat`. This is the density kernel for gravity anomaly. It is also a Compressed Sparse Row Matrix, the format of it is the same as surface wave kernel in 4.5.

7 Advanced Options

7.1 Gauss-Legendre Quadrature and the Sparcity of Gravity Matrix

In `src/gravity/gravmat.cpp`, you could modify the max and min order of Gauss-Legendre Quadrature, and the maximum distance beyond which gravity matrix is zero (The attraction between two points is negligible if beyond this distance)

```
const int NMIN=3,NMAX=256;
const float maxdis = 100.0;
```

Also, because we use a preallocated memory to store gravity matrix in the program, you should make sure this memory is large enough. If your memory is small, this program will complain and stop with infomation like `"Please increase the sparse ratio!"`. In fact, the size of the memory (bytes) used is from:

$$Mem = sparse * m * n * 4$$

Where `m` is the number of observed gravity data, `n` is the total size of your model, and `sparse` is the sparse ratio of the gravity matrix. To change that memory used, you should modify this line in `src/gravity/gravmat.cpp`:

```
const float sparse = 0.2;
```

where sparse should be a floating number between 0 and 1. Note that this number is dependent on the `maxdis` in the same file. After you change (one of) these parameters, please remember to recompile this program.

7.2 Empirical Relations

Empirical relations are utilized to connect density and velocity of rocks in our program. If you have better relations in your research area, you could change this relation in `src/utils/bkg_model.cpp`

```
void MOD3d::
 empirical_relation (float vsz, float &vpz,float &rhoz)
{
    vpz = 0.9409 + 2.0947*vsz - 0.8206*pow(vsz,2)+
            0.2683*pow(vsz,3) - 0.0251*pow(vsz,4);
    rhoz = 1.6612 * vpz - 0.4721 * pow(vpz,2) +
            0.0671 * pow(vpz,3) - 0.0043 * pow(vpz,4) +
            0.000106 * pow(vpz,5);
}
```

And if you want to conduct joint inversion, please remember to change the derivative function in the same file.

7.3 Random Seed

In our program, to make sure the generated random number the same in each independent numerical experiments, we fix the random seed in `src/utils/gaussian.cpp`:

```
static default_random_engine e(11);
```

You could change 11 to other integer if required.

8 Tips and Tools

8.1 Warnings

In fact, `DSurfTomo` and `JointSG` are all dependent on two modules. First one is the forward computation of dispersion curves from 1-D model, and another one is the ray-tracing on 2-D spherical surface. There would be some possible errors in these two modules due to incompatible input model format, terrible initial model, and even the inappropriate location of your stations. To tackle possible problems, the two modules will print some information on the screen or in a output file to help you debug.

The first possible warning is `"WARNING:improper initial value in disper-no zero found"` on the screen. Then a `fort.66` file will be generated in the running directory. This error is due to a strange 1-D model that the dispersion module cannot find dispersion curves for this model. There are some possible reasons for this warning. If you find this problem for the first iteration, this problem may from that the format of your initial model is different from the required one. You could check the model in file `fort.66` and compare that with your model. Another possible reason is that you take inappropriate Tickonov regulariztion parameters (i.e. damping and smooth factors) which lead to abrupt change of previous model.

The second warning is `"Source lies outside bounds of model (lat,long)="`. This reason is caused by that the receivers are too close to the boundary of the current model that ray paths are along this boundary. If this warning happens, please enlarge your model to avoid this problem.

8.2 L-curve Analysis

Usually L-curve analysis is applied to determine the best suitable smoothing and damping factor for a inverse problem. In our numerical experiment, we find that the damping factor affects the final result smaller than smoothing factor. So we `recommend` you to keep the damping factor to a small value (such as 0.001) during your inversion, and only tune the smoothing factor.

In the folder `utils/`, we have prepared two scripts to help you tune the smoothing factor. The first one is a bash script `tune_L_curve.sh` that will generate a series of inversion results in the folder `storage/`. Please read this script carefully and then try your own examples.

The second python script `utils/lcurve.py` will help you generate a L-curve based on your results. Before running this script, please change all the required parameters in the function `main()`.

8.3 Checkpoint Restart

Sometimes you want to start from one of the results (like mod_iter5.dat) instead of the initial model. We prepared a python script `utils/out2init.py`, you could run this script like this

```
python out2init.py mod_iter5.dat MOD05
```

Then this script will convert the output model to the format of the initial model. This script will be useful in joint inversion. And also, it may help you restart from some checkpoint if some issues happend.

References

T. M. Brocher. Empirical Relations between Elastic Wavespeeds and Density in the Earth's Crust. *Bulletin of the Seismological Society of America*, 95 (6):2081–2092, 12 2005. ISSN 0037-1106. doi: 10.1785/0120050077. URL https://doi.org/10.1785/0120050077.

N. Du, Z. Li, T. Hao, X. Xia, Y. Shi, and Y. Xu. Joint tomographic inversion of crustal structure beneath the eastern Tibetan Plateau with ambient noise and gravity data. *Geophysical Journal International*, 07 2021. ISSN 0956-540X. doi: 10.1093/gji/ggab299. URL https://doi.org/10.1093/gji/ggab299. ggab299.

H. Fang, H. Yao, H. Zhang, Y.-C. Huang, and R. D. van der Hilst. Direct inversion of surface wave dispersion for three-dimensional shallow crustal structure based on ray tracing: methodology and application. *Geophysical Journal International*, 201(3):1251–1263, 03 2015. ISSN 0956-540X. doi: 10.1093/gji/ggv080. URL https://doi.org/10.1093/gji/ggv080.

N. A. Haskell. The dispersion of surface waves on multilayered media*. *Bulletin of the Seismological Society of America*, 43(1):17–34, 01 1953. ISSN 0037-1106. doi: 10.1785/BSSA0430010017. URL https://doi.org/10.1785/BSSA0430010017.

J. Julià, C. J. Ammon, R. B. Herrmann, and A. M. Correig. Joint inversion of receiver function and surface wave dispersion observations. *Geophysical Journal International*, 143(1):99–112, 2000. doi: https://doi.org/10.1046/j.1365-246x.2000.00217.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-246x.2000.00217.x.

Z. Li, T. Hao, Y. Xu, and Y. Xu. An efficient and adaptive approach for modeling gravity effects in spherical coordinates. *Journal of Applied Geophysics*, 73(3):221–231, 2011.

N. Rawlinson and M. Sambridge. Wave front evolution in strongly heterogeneous layered media using the fast marching method. *Geophysical Journal International*, 156(3):631–647, 2004.

F. Schwab and L. Knopoff. Fast surface wave and free mode computations. In *Methods in Computational Physics: Advances in Research and Applications*, volume 11, pages 87–180. Elsevier, 1972.