Manual of JDSurfG

Nanqiao Du

August 28, 2020

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. This package, containing three modules, are used to compute gravity matrix in spherical coordinates, to conduct direct surface wave tomography, and to perform tomographic joint inversion of dispersion and gravity data respectively. Obviously, this kind of joint inversion is based on one-step surface wave inversion to compute 3-D (pseudo) sensitivity kernel of surface waves, and the adaptive Gauss-Legendre integration method is utilized to handle the gravity part. Based on empirical relations between seismic velocity and density of rocks, this package would be a useful tool to investigate 3-D shear wave structure in the crust and upper mantle.

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 also, computing 1-D surface wave frechet kernel. And the `Fortran` part, derived from Hongjian Fang's DSurfTomo package with several modifications (mainly by adding parallel mode), is used to compute surface wave traveltimes and corresponding 2-D sensitivity kernels.

I should note that this package was only tested by a small part of 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 if 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 and weights. So you need to install these two libraries in your own computers before compiling this package. Also, you should make sure that your `C++` compiler support `C++-11` standards (gcc >= 4.8).

3 Installation

After you finish downloading the .zip file from Github, you can unzip it by using:

```
1  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 example:

```
path_to_eigen= -I/path-to-eigen
```

Then compile this package by:

```
cd JDSurfG
make
```

Finally you will find three excecutable files in the directory `bin/`

```
mkmat DSurfTomo JointSG
```

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 inversion one in order to perform B-spline smoothing for forward computation. To be specific, if we need to invert model in region ($lat_0 \sim lat_1, lon_0 \sim lon_1, 0 \sim dep_1$), we should edit your input model 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 invert model boundaries, or some warning will be given when you running this package. This is to avoid that surface wave train travels along the 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 file below:

```
1  33 36 25          c: nx ny nz (grid number in lat lon and depth direction)
2  35.3 99.7         c: goxd gozd (upper left point,[lat,lon])
3  0.3 0.3           c: dvxd dvzd (grid interval in lat and lon direction)
4  10.0 2            c: weight damp
5  3                 c: sublayers (computing depth kernel, 2~5)
6  2.0 4.7           c: minimum velocity, maximum velocity
7  10                c: maximum iteration
8  10                c: kmaxRc (followed by periods)
9  4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0 22.0
10 10                c: kmaxRg (followed by periods)
11 4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0 22.0
12 0                 c: kmaxLc (followed by periods)
13 0                 c: kmaxLg (followed by periods)
14 1                 c: synthetic flag (0: real data,1: synthetic)
15 0.02              c: noiselevel
```

line 1 grid points in **latitude**,**longitude** and in **depth**

line 2 lat and lon in upper left point (Note this is the upper left point of input model instead of the inversion one!)

line 3 grid interval in lat and lon direction

line 4 2-th order Tikhonov regularization parameter and damp.

line 5 change grid-based model to layer-based model, insert sublayer layers between adjacent grids. Usually 3 is enough.

line 6 minimum and maximum velocity permitted in inversion.

line 7 maximum iteration

line 8 no. of Rayleigh wave phase velocity disperion period used.

line 9 if the number in line 8 isn't zero, then list all the periods. If not, list Rayleigh wave group velocity periods, and then Love phase, Love group.

line ? conduct checkerboard test? if so, set it to 1.

line ? noiselevel, the relative amplitude of noise

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

4.2 Initial Model and True Model File

The initial and True model are called `MOD` and `MOD.true` respectively in this instruction.
These is only minor difference between these two files, where the first line of

`MOD` are depth of every grid nodes in depth direction. And this line don't exist in `MOD.true`.

The subsequent lines of `MOD` are shear wave velocities. If we store the velocity with the format V(nz,nlon,nlat), 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 from cross-correlation of long-time noise signals of 2 stations.
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
```

It is obvious that this file contains 2 kind of stations: surface wave emitting station (with `#` before each line) and receiver station (without `#` in each line). For a emitting station, the format of this line is:

`#` lat lon period-index wavetype velotype:

- wavetype: the type of surface wave, for Rayleigh it is 2,and 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 we use dispersion for Rayleigh phase velocity 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 station who you have successfully extracted signals from 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 prepare all these required files in a folder, then you could enter into this folder, and print 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 2 folders: `kernel/` and `results/`.

In `kernel/`, there are 3-D pseudo sensitivity kenrels, and `results/` contains all the models after every 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 7.1),and start from 0. the format of every txt file is : data-index,model-index(only one needs to be inverted) and the value of the kernel.
- `results/`:contains models (`mod_iter*`) and surface wave travel time (`res*.dat`) of each iteration. The format of model files is: lon lat depth S-wave, and the format of traveltime file is: interstation distance, observed traveltime and synthetic traveltime for current model.

5 Joint Inversion Module

This module need 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 mode.
- `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 two lines.

In order to know how it works, here I list the cost function of joint inversion:

$$L = \frac{p}{N_1\sigma_1^2}(d_1 - d_1^o)^2 + \frac{1-p}{N_2\sigma_2^2}(d_2 - d_2^o)^2 \tag{1}$$

- last but two line: parameter p, which is in [0,1],p=0 is equivalent to inversion by gravity only, and, respectively, p=1 means inversion by surface wave.
- last line: σ for surface wave and gravity, respectively.

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 (refer to 6-th chapter).

5.4 Gravity Reference Model File

Based on the definition of gravity anomaly, it is the perturbance according to a normal elliptoid. Thus we need to convert it to local anomalies in our research area. In fact, we remove the average value of oberserved anomalies. In each iteration, 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 results of previous step.

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.)

5.5 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:

```
./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 2 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:

Observed-anomalies Synthetic-anomalies

6 Gravity Module

This module requires 3 input files:

- `JointSG.in` or `DSurfTomo.in`: contains model information
- MOD(and MOD.true) : Initial model and true model.
- 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:

```
./mkmodel -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. The format of it is:

data-index model-index kernel

7 Advanced Options

7.1 Number of Threads used in Parallel Mode

There are one line in `include/openmp.hpp`:

```
#define nthread 4
```

Change 4 to other number according to your computation environment.

7.2 Empirical Relations

in `src/utils/empirical.f90`

```
subroutine empirical_relation (vsz,vpz,&
    rhoz)bind(c,name="empirical_relation")
    use, intrinsic :: iso_c_binding

    real(c_float), intent(in) :: vsz
    real(c_float),intent(out) :: vpz,rhoz

    vpz=0.9409 + 2.0947*vsz - 0.8206*vsz**2+ &
            0.2683*vsz**3 - 0.0251*vsz**4
    rhoz=1.6612*vpz- 0.4721*vpz**2 + &
            0.0671*vpz**3 - 0.0043*vpz**4 + &
            0.000106*vpz**5

end subroutine empirical_relation
```

You could change this relation according to your conditions. And if you want to conduct Joint inversion, please remember to change the derivative function in the same file.

7.3 Number of Gauss-Legendre Nodes and the Sparcity of Gravity Matrix

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

```
#define NMIN 5
#define NMAX 256
#define maxdis 100.0
```