tomo2D: a C++ package for 2-D joint refraction and reflection travel-time tomography

Jun Korenaga

January 7, 2019

Contents

1 Overview

This package corresponds to the joint refraction and reflection travel-time tomography presented by *Korenaga et al.* [*J. Geophys. Res.*, 105, 21591-21614, 2000]. Please read this paper before using this code. At this moment, this README provides the very minimum technical details. If you have any suggestions, find bugs, or find errors in this document, please contact the author by sending an email to *korenaga@seismo.berkeley.edu*.

gen_smesh	- generate a velocity mesh
edit_smesh	- edit a velocity mesh
stat_smesh	- do some statistical operations on a given velocity mesh
tt_forward	- calculate travel times for given sources and receivers
tt_inverse	- do travel-time inversion

NOTE: **tt_inverse** is a stand-alone application, i.e., it contains the forward calculation part, which is also separately available as **tt_forward**.

2 Installation

Make sure GNU C++ compiler (with the standard template library) is installed on your platform. Edit `Makefile` to reflect your computing environment, and then type

```
% make all
```

3 File formats

3.1 Velocity grid format (sheared grid)

1st line:	
$nx \quad nz \quad v_{\text{water}} \quad v_{\text{air}}$	- number of nodes in x and z, velocity in water and air
2nd line:	
$x_1 \quad x_2 \quad x_3 \quad ..... \quad x_{nx}$	- node's x-coordinates
3rd line:	
$b_1 \quad b_2 \quad b_3 \quad ...... \quad b_{nx}$	- corresponding bathymetry (negative for subaerial)
4th line:	
$z_1 \quad z_2 \quad z_3 \quad ...... \quad z_{nz}$	- node's z-coordinates
5th to $(4+nx)$th line:	
$v_{i1} \quad v_{i2} \quad v_{i3} \quad ...... \quad v_{i,nz}$	- velocity at node (i,j) for $i = 1...nx$ and $j = 1...nz$

The x- and z-coordinates should be in increasing order. The z-coordinate is relative to seafloor (coordinate axis is pointing downward). There is no limit to the number of nodes you can put in one line, as long as it is consistent with the first line and it is within the capacity of your computer.

An example is given below:

```
% cat sample_grid.dat
5 5 1.5 0.33
0 1 2 3 4
0 0 0 0 0
0 1 2 3 4
1.5 1.8 2.1 2.4 2.7
1.5 1.8 2.1 2.4 2.7
1.5 1.8 2.1 2.4 2.7
1.5 1.8 2.1 2.4 2.7
1.5 1.8 2.1 2.4 2.7
%
```

3.2 Traveltime data file format

The first line contains only one number, *nsrc*, which is the number of sources. The rest of the file must contain *nsrc* packets of traveltime data, each of which has the following format:

1st line of each packet:	
s x z $nrec$	- flag 's', src's coordinate (x,z), and number of receivers
2nd to $(1+nrec)$th lines:	
r x z $code$ $time$ dt	- flag 'r', rcv's coordinate (x,z), raycode (0:refraction, 1:reflection), traveltime in second, and pick error in second

For example,

```
% cat sample_tt.dat
3
s 1 0 9
r 3 0 0 0.662926 0.01
r 4 0 0 0.987085 0.01
r 6 0 0 1.61061 0.01
r 9 0 0 2.48161 0.01
r 10 0 0 2.75761 0.01
r 14 0 0 3.76555 0.01
r 15 0 0 3.98043 0.01
r 17 0 0 4.36218 0.01
r 19 0 0 4.71357 0.01
s 5 0 10
r 2 0 0 0.986902 0.01
r 3 0 0 0.662816 0.01
r 9 0 0 1.31126 0.01
r 10 0 0 1.62676 0.01
r 11 0 0 1.93552 0.01
r 12 0 0 2.23615 0.01
r 15 0 0 3.06797 0.01
r 16 0 0 3.31544 0.01
r 18 0 0 3.76822 0.01
r 19 0 0 3.98088 0.01
s 10 0 8
r 1 0 0 2.75761 0.01
r 5 0 0 1.62676 0.01
r 6 0 0 1.31376 0.01
r 8 0 0 0.664457 0.01
r 13 0 0 0.991701 0.01
r 14 0 0 1.31261 0.01
r 16 0 0 1.92478 0.01
r 19 0 0 2.75127 0.01
%
```

3.3 Correlation length formats

Horizontal and vertical correlation lengths for velocity nodes are specified in a manner very similar to the velocity grid format:

1st line:	
nx nz	- number of nodes in x and z directions
2nd line:	
x_1 x_2 x_3 ..... x_{nx}	- node's x-coordinates
3rd line:	
b_1 b_2 b_3 ...... b_{nx}	- corresponding bathymetry (negative for subaerial)
4th line:	
z_1 z_2 z_3 ...... z_{nz}	- node's z-coordinates
5th to $(4+nx)$th line:	
Lh_{i1} Lh_{i2} Lh_{i3} ...... $Lh_{i,nz}$	- horizontal correlation length at node (i,j) for $i=1...nx$ and $j=1...nz$
$(5+nx$ to $(4+2nx)$th line:	
Lv_{i1} Lv_{i2} Lv_{i3} ...... $Lv_{i,nz}$	- vertical correlation length at node (i,j) for $i=1...nx$ and $j=1...nz$

An example is given below:

```
% cat vcorr.dat
2 2
0.0 40.0
0.0 0.0
0.0 10.0
2 5
2 5
0.5 1
0.5 1
%
```

Correlation lengths for reflector nodes can be specified in a separate file as:

*i*th line:	
x_i L_i	- x coordinate and corresponding correlation length

An example is given below:

```
% cat dcorr.dat
0.0 5
40  5
%
```

Note that if a correlation length file for reflector nodes is not provided, **tt_inverse** will sample from horizontal correlation lengths for velocity nodes.

3.4 Variable damping format

Spatially variable damping is implemented by **tt_inverse -DQ***damp_file*, which is useful for squeezing tests. The file format is very similar to those for velocity grid and correlation lengths:

1st line:	
nx nz	- number of nodes in x and z directions
2nd line:	
x_1 x_2 x_3 ..... x_{nx}	- node's x-coordinates
3rd line:	
b_1 b_2 b_3 ...... b_{nx}	- corresponding bathymetry (negative for subaerial)
4th line:	
z_1 z_2 z_3 ...... z_{nz}	- node's z-coordinates
5th to $(4+nx)$th line:	
D_{i1} D_{i2} D_{i3} ...... $D_{i,nz}$	- damping weight at node (i,j) for $i=1...nx$ and $j=1...nz$

An example is given below:

```
% cat damp.dat
2 4
-115 115
0 0
0.0 5.0 5.1 14.0
1 1 100 100
1 1 100 100
%
```

Since continuous function is assumed in the file format, a thin layer (z=5.0-5.1 km in the above example) is inserted to simulate a step-wise increase in damping weight. The file format is the sheared grid format, so you can hang the squeezing depth from the seafloor, if bathymetry is specified for the above topography nodes.

4 Command Description

4.1 Manipulating velocity grid files

NAME **gen_smesh** - generate a velocity mesh

SYNOPSIS

gen_smesh [velocity options] [grid options]

DESCRIPTION

This command generates a velocity grid, which is a required input file for other programs. It is a sort of ad hoc program, which doesn't have well-structured options. See examples below to understand option semantics.

OPTIONS

-A*v0* **-B***gradient*

- specifies velocity as a function of depth, $V(z) = v0 + gradient * z$.

-C*v.in/ilayer* [**-F***jlayer/refl_file*]

- uses `v.in` of Zelt's RAYINVER to construct a velocity field. The seafloor layer must be given by *ilayer*. Use **-F** to extract *jlayer* as a reflector in *refl_file*.

-N*nx/nz* **-D***xmax/zmax*

- specifies a uniform spacing grid with $nx \times nz$ nodes, spanning from 0 to *xmax* km horizontally, and from 0 to *zmax* km depth vertically.

-X*xfile* **-Z***zfile* [**-T***tfile*]

- specifies a variable spacing grid, as defined by *xfile* and *zfile*. Optional *tfile* specifies variable bathymetry.

-E*dx* **-Z***zfile*

- creates a grid based on *v.in* given in **-C** option, with a (nearly) uniform horizontal spacing of *dx* km, a variable vertical spacing as defined by *zfile*.

EXAMPLES

Currently gen_smesh can create the following three different velocity grids:

1. Uniform spacing grid with a single velocity gradient (flat topography, no water column)
"**-A**3.0 **-B**0.5 **-N**51/51 **-D**25/10" generates with a grid for a model domain covering horizontally from 0 to 25 km and vertically from 0 to 10 km depth, with a horizontal grid spacing of 0.5 km and a vertical spacing of 0.2 km. Velocity is a function of depth as $V(z) = 3.0 + 0.5z$.
2. Variable spacing grid with a single velocity gradient
"**-A**3.0 **-B**.5 **-X***xfile* **-Z***zfile* [**-T***tfile*]" generates a grid with the same velocity variation, but the model domain and grid spacing are specified by *xfile* and *zfile*. *xfile* contains the x-coordinates of nodes like

```
0.0
0.1
0.3
0.9
1.0
```

(this is variable spacing 5 nodes, spanning from 0 km to 1 km), and *zfile* contains the z-coordinates (below seafloor) as

```
0.0
0.05
0.10
0.15
0.20
0.30
.....
```

Optional *tfile* contains bathymetry at the x-coordinates specified in *xfile*. (so the number of points in this file should be equal to that in *xfile*.)
3. Variable spacing grid based on a Zelt-formatted file
"**-C***v.in/ilayer* **-E**1.0 *-Zzfile* [**-F***jlayer/reflfile*]" transforms the input *v.in* file into our velocity grid format with a grid spacing of (approximately) 1 km. *ilayer* specifies the number of the layer corresponding to seafloor. (We need to tell the program how to interpret the input *v.in* file.) **gen_smesh** preserves the original bathymetry nodes. For example, if the bathymetry layer is defined as

```
2   0.0   0.5   2.0   4.0   4.5
0   2.2   2.5   2.8   3.1   3.0
      0     0     0     0     0
```

additional nodes are inserted between the original nodes if the spacing exceeds the given spacing (in this case, 1 km). So in this case, **gen_smesh** probably produces the x-coordinates as 0.0, 0.5, 1.25, 2.0, 3.0, 4.0, and 4.5. *zfile* contains the z-coordinates (below seafloor) of nodes as in the example 2. The option **-F** is used when you want to extract a Moho interface at the *top* of *jlayer*.

NAME **edit_smesh** - edit a velocity mesh

SYNOPSIS

edit_smesh *grid_file* **-C***cmd* [**-L***vcorr_file* **-U***upper_file*]

DESCRIPTION

This another ad-hoc program may be useful when you play with synthetic tests.

OPTIONS

-Ca - set all velocities to horizontal average

-Cp*grid* - paste *grid* on the original grid

-CP*prof* - paste 1-D profile given by *prof*

-Cs*h/v* - apply Gaussian smoothing operator with an window of *h* km (horizontal) and *v* km (vertical)

-Cr*mx/mz*
- refine mesh by *mx* for x-direction and by *mz* for z-direction

-Cc*A/h/v*
- add checkerboard pattern with amplitude *A* (%), horizontal cycle *h* km, and vertical cycle *v* km

-Cd*A/xmin/xmax/zmin/zmax*
- add a rectangular anomaly with amplitude *A* (%)

-Cg*A/x0/z0/Lh/Lv*
- add a Gaussian anomaly of $A\exp[-(x-x0)^2/Lh-(z-z0)^2/Lv]$ (%)

-Cl - remove low velocity zone

-CR*seed/A/nrand*
- randomize the velocity field

-CS*seedA/xmin/xmax/dx/zmin/zmax/dz*
- another randomization

-CG*seed/A/N/xmin/xmax/zmin/zmax*
- yet another randomization

-Cm*v/refl_file*
- set velocities below $refl\_file$ as *v*

-L*vcorr_file*
- set correlation length file used by **-Cs**

-U*upper_file*
- set upper limit depth for edit operations

EXAMPLES

If you want to embed a low velocity anomaly of 5in a subdomain of x=10-20km & z=5-8km, this can be done by

```
% edit_smesh vgrid.orig.dat -Cd-5/10/20/5/8 > vgrid.new.dat
```

If you want to add in a checkerboard pattern of 3the horizontal wavelength of 10 km and the vertical wavelength of 5 km,

```
% edit_smesh vgrid.orig.dat -Cc3/10/5 > vgrid.new.dat
```

NAME **stat_smesh** - do some statistical operations for velocity grid(s) or reflector(s)

SYNOPSIS

stat_smesh **-L***list_file* **-C***cmd* [**-R***n*]

stat_smesh **-M***mesh* **-D***cmd* [**-T***topb* **-B***botb* **-m***midb* **-P***TPcorr* **-U***vrepl* **-X***xmin/xmax* **-x***cxmin/cxmax* **-t***ctopb* **-b***cbotb*]

DESCRIPTION

This is yet another ad-hoc program, which may be used for Monte Carlo uncertainty analysis.

OPTIONS

-L*list_file*
- specifies a list of velocity grid (or reflector) files

-C*cmd* - sets an operation for a list of grids (or reflectors)

-Ca - takes ensemble average

-Cr*ave_file*
- calculates standard deviation from *ave_file*

-R*n* - assumes reflector of *n* nodes, instead of velocity grid

-M*grid* - specifies a velocity grid file

-D*cmd* - sets an operation for a single grid

-Da*avex/wlen*
- takes horizontal average at x=*avex* with a window of *wlen* (km)

-Db*xmin/xmax/dx/wlen*
- takes horizontal and vertical average with a window of *wlen* (km), from *xmin* to *xmax* with an increment of *dx*

-T*topb_file*
- sets top boundary by *topb_file*

-B*botb_file*
- sets bottom boundary by *botb_file*

-m*midb_file*
- sets middle boundary by *midb_file*

-P*Tref/Pref/dVdT/dVdP/a/b*
- applies temperature and pressure corrections to the reference condition of *Tref* (°C) and *Pref* (MPa). Temperature profile is calculated as $T = az + b$ where z is depth beneath seafloor.

-U*vrepl* - sets all velocities lower than *vrepl* to *vrepl*

-X*xmin/xmax*
- sets horizontal range for operation

-x*cxmin/cxmax* **-t***ctopb_file* **-b***cbotb_file*
- sets the region to be skipped by operation (e.g., continental crust in *Korenaga et al.* [2000])

4.2 Forward traveltime calculation

NAME **tt_forward** - forward traveltime calculation

SYNOPSIS

tt_forward **-M***grid_file* [**-G***geom_file* **-F***refl_file* **-A**] [**-N***xorder/zorder/clen/nintp/tot1/tot2* **-E***elem* **-g** **-T***ttime* **-O***obs_ttime* **-r***v0* **-D***diff* **-R***ray* **-S***src* **-I***vel* **-i***w/e/s/n/dx/dz* **-n** **-C***used_time* **-V***level*]

DESCRIPTION

This program uses a hybrid approach based on the graph method and the bending method.

OPTIONS

-M*grid_file*
- specifies a velocity grid file

-G*geom_file*
- specifies a geometry file (with the same file format as the traveltime data file - with zeros for travel time and pick error).

-F*refl_file*
- specifies a reflector file

-A - takes an extra care for reflection phase (more time-consuming)

-N*xorder/zorder/clen/nintp/tot1/tot2*
- specifies a *xorder*×*zorder* forward star in the graph method, sets the maximum segment length (*clen*), the number of interpolation points per segment (*nintp*), and tolerance levels for iterations (*tot1* for conjugate gradient and *tot2* for Brent minimization) used in the bending method

-E*elem_file*
- prints out the elements of a grid file to *elem_file*

-g - use the graph method only

-T*ttime_file*
- prints out calculated travel times to *ttime_file*

-O*obs_ttime_file*
- prints out input observed travel times to *obs_ttime_file*

-r*v0* - sets reduction velocity for travel time output

-D*diff_file*
- prints out differential travel times to *diff_file*

-R*ray_file*
- prints out ray paths to *ray_file*

-S*src_file*
- prints out source locations to *src_file*

-I*vel_file* - prints out a velocity file to *vel_file*

-i*w/e/s/n/dx/dz*
- specifies nodes and region for **-I**

-n - suppresses printing water and air velocity nodes for **-I**

-C*used_time*

-V*level* - sets verbose mode (*level* = 0 or 1)

NOTES If **-G** is not specified, only operations regarding a velocity grid will be done.

4.3 Traveltime inversion

NAME **tt_inverse** - traveltime inversion

SYNOPSIS

tt_inverse **-M***grid_file* **-G***data_file* [**-N***xorder/zorder/clen/nintp/tol1/tol2*] [**-F***refl_file* **-A** **-L***logfile* **-O***out_fn_root* [**-o***level* **-l**] **-K***dws_file*] [**-P** **-R***crit_chi* **-Q***lsqr_tol* **-s***bound* **-W***d_weight* **-V***level*] [**-CV***vcorr_file* **-CD***dcorr_file*] [iteration options] [smoothing options] [damping options]

DESCRIPTION

This command is an implementation of joint refraction and reflection traveltime tomography presented by *Korenaga et al.* [2000].

OPTIONS

-M*grid_file*
: - specifies a velocity grid file

-G*date_file*
: - specifies a traveltime data file

-N*xorder/zorder/clen/nintp/tol1/tol2*
: (see **tt_forward**)

-F*refl_file*
: (see **tt_forward**)

-A (see **tt_forward**)

-L*logfile* - sets log file, with the output format as: 1. the number of iteration, 2. the number of set, 3. the number of rejected data, 4. RMS traveltime misfit (Pg+PmP), 5. initial χ^2 (Pg+PmP), 6. the number of valid Pg data, 7. RMS traveltime misfit (Pg), 8. initial χ^2 (Pg) 9. the number of valid PmP data, 10. RMS traveltime misfit (PmP), 11. initial χ^2 (PmP), 12. CPU time used for graph solution, 13. CPU time used for bending solution, 14. smoothing weight for velocity nodes, 15. smoothing weight for depth nodes, 16. damping weight for velocity nodes, 17. damping weight for depth nodes, 18. the number of LSQR calls, 19. the total number of LSQR iteration, 20. CPU time used for LSQR, 21. predicted χ^2 based on LSQR solution (Pg+PmP), 22. average velocity perturbation, 23. average depth perturbation, 24. horizontal roughness of velocity nodes, 25. vertical roughness of velocity nodes, and 26. roughness of depth nodes.

-O*out_fn_root*
: - sets file name root for output files

-o*level* - sets output level (print out travel time residual for *level* ≥ 1; print out ray paths for *level* ≥ 2)

-l - prints out the final model only

-K*dws_file*
: - prints out DWS to *dws_file*

-P - sets pure jumping strategy

-R*crit_chi*
: - sets critical χ for robust inversion

-Q*lsqr_tol*
: - sets tolerance for LSQR algorithm

-s[*bound_file*]
: - applies 2-D filter after every iteration. The upper bound for filtering can be set by *bound_file*.

-W*d_weight*
: - sets depth kernel weighting factor

-V[*level*]
: - sets verbose level

-CV*vcorr_file*
: - sets correlation length file for velocity nodes

-CD*dcorr_file*
: - sets correlation length file for reflector

Type-1 iteration options: *many iterations with a single set of parameters*

- **-I***niter* - sets the number of maximum iterations
- **-J***target_chi2*
  - sets target χ^2
- **-SV***wsv* - applies velocity smoothing with weighting factor *wsv*
- **-SD***wsd* - applies depth smoothing with weighting factor *wsv*

Type-2 iteration options: *single iteration with many sets of parameters*

- **-SV***wsv_min/wsv_max/dw* [**-XV**]
  - tries velocity smoothing with weighting factor varying from *wsv_min* to *wsv_max* with an increment of *dw*. With **-XV**, smoothing weights will be raised to the power of 10.

-SD*wsd_min/wsd_max/dw* [**-XD**]
- tries depth smoothing with weighting factor varying from *wsd_min* to *wsd_max* with an increment of *dw*. With **-XD**, smoothing weights will be raised to the power of 10.

-TV*max_dv*
- applies velocity damping with maximum velocity perturbation of *max_dv* (%)

-TD*max_dd*
- applies depth damping with maximum velocity perturbation of *max_dd* (%)

-DV*wdv* - applies velocity damping with weighting factor *wdv*

-DD*wdd* - applies depth damping with weighting factor *wdd*

-DQ*damp_file*
- applied velocity damping with spatially variable weighting factor specified by *damp_file* (for squeezing)

---

5 Sample shell scripts

Sample shell scripts are provided with this package under the directory `sample`. Wessel and Smith's GMT tools are assumed to be installed. Edit `togrid.sh` to modify a path to GMT's **surface** command if necessary.

The following command sequence should be able to produce something like Figure 6 of *Korenaga et al.* [2000]:

```
% ./prep_model.sh
% ./make_data.sh
% ./do_inv.sh type1 1
% ./do_inv.sh type1 100
% ./make_fig.sh
```

You should then find a PS file named as `invex.ps` under this directory, which looks like: