USER MANUAL

First Arrival Delayed Time Tomography
tomoTV
By Jean Virieux

Version 0.00
Winter 2015

Contribution of Ortensia Amoroso, Stéphanie Gauthier, Jean-Luc Got, Jenny Jacques, Diana Latorre, Vadim Monteiller, Stéphane Operto, Céline Ravaut, Tiziana Vanorio, Aldo Zollo.

CAUTION A: files are different from the previous TLR3 software (conversion program is provided as it is for helping you transforming TLR3 input files into tomoTV files).

CAUTION B: this software has not yet the double difference feature developed in the original tomoTV code of Monteiller …

CAUTION C: values follow units of the International System (be aware!)

I – Introduction

General purpose

We propose a modular hypocenter-velocity tomography computer code written in FORTRAN90 (only one subroutine is written in C as the eikonal solver of Podvin & Lecomte). When one wishes locating hypocenters and inverting for velocity structures simultaneously, the different steps may require various computer resources and one may repeat a new step without performing previous ones for which information has already been obtained.

For theoretical approaches, the reading of books and articles are recommended. Books we may recommend are

- D. Zhao, 2015, Multiscale Seismic Tomography, Springer Geophysics, Springer
- K. Aki, 1993, Overview in Seismic tomography: theory and practice, edited by H.M. Iyer and K. Hirahara, pp 1.-8., Chapman & Hall, London.
- J. Bee Bednar, 1989, Geophysical Inversion, SIAM
- G. Nolet, 1987, Seismic tomography with applications in global seismology and exploration geophysics, Kluwer Academic Pub., 386 pp.
- G. Nolet, 2008, A breviary of seismic tomography, Imaging the interior of the Earth and Sun, Cambridge University Press, 339pp.

Many papers are available for seismic travel time tomography: here are those where the approach we are going to describe has been used

Le Meur, H., J. Virieux and P. Podvin, 1997, Seismic tomography of the Gulf of Corinth: a comparison of methods, Annali di Geofisica, XL, 1, pp1-25
- Ghose,S., M.W. Hamburger & J. Virieux, 1998. Three-dimensional velocity structure and earthquake locations beneath the northern Tien Shan of Kyrgyzstan, central Asia, J. Geophys. Res., 103, 2725-2748.

- Latorre, D., Virieux, J., Monfret, T., Monteiller, V., Vanorio, T., Got, J.-L. & H Lyon-Caen, 2004. A new seismic tomography of Aigion area (Gulf of Corinth, Greece) from the 1991 data set, Geophys. J. Int., 159, 1013-1031.
- Monteiller V., Got J.-L., J. Virieux & P. Okubo, 2005, An efficient algorithm for double-difference tomography and location in heterogeneous media, with an application to Kilauea volcano, J. Geophys. Res., 110, B12306, doi:10.1029/2004JB003466.
- Vanorio T., J. Virieux, P. Capuano & G. Russo, 2005, Three-dimensional seismic tomography from P wave and S wave microearthquake travel times and rock physics characterization of the Campi Flegrei Caldera, J. Geophys. Res., 110, B03201, doi:10.1029/2004JB003102.
- Gautier S., Latorre D., Virieux J., Deschamps A., Skarpelos C., Sotiriou A., Serpetsidaki A. & Tselentis A., 2006, A New Passive Tomography of the Aigion Area (Gulf Of Corinth, Greece) from the 2002 Data set, Pure appl. Geophys, 163, 431-453 .

Basically, one has to define an initial P wave velocity model (or P & S velocity models) as well as an initial set of locations for hypocenters, each of them depends on four parameters (three coordinates and an origin time). We may consider also blasts where the position is known or shots where both the position and the origin time are known.

The event information is described through a binary file in a Cartesian system (see section III). Please note that this information is going to change as we proceed through iterations.

The velocity structure is defined on a regular grid through binary files (see section III). The source information is also described through a binary file in a Cartesian system. These files might change during the inversion procedure.

The receiver information is also described through binary file in a Cartesian system. Because statics at receivers are not included in the inversion, this file is not modified during the inversion (see section III).

The data information related to travel-times and uncertainties in travel-times data is also described in a binary file. Although the information in this file could be updated through the inversion procedure, the data content is not modified. Only useful information for the inversion is stored.

We shall provide the necessary information about the data in the section II. The velocity information will be described in the section III. The two-points ray tracing will be performed in the section IV. The estimation of the sensitivity matrix will be done in the section V. The damping effect as well as the a priori information and smoothing procedure will be described in the section VI. The section VII is devoted to the inversion itself and the reconstruction of both the velocity files and the new source file. After practical information given in the section VIII, we conclude in the section IX.

Root directory architecture

Inside this directory you could install anywhere, you will find the following structure

BIN_LINUX_PL: where should be binaries of computer codes
DOC_TOMOTV: where you will find this document
SRC_OCTOBER_2022: where you will find computer source codes
MAKE.DIR: where you will find typical files you should have in the root directory as make.inc to be used by various files Makefile.

This is the minimal content of the root directory to be needed. You may find other directories depending the current status of the code.

You will find as well many directories for building your acquisition configuration and for running few examples.

Although seismic tomography is based on pretty images, no graphic tools are provided and you may rely on those you are familiar with. The graphical environment that is often used comes from CWP project through SEISMIC UNIX (SU) software as well as from GNU. We use essentially ximage for quick check-up of velocities and GNUPLOT for quick check-up of hypcenters.

Inside the directory SRC, you will find the following structure

- Makefile: the make file for compiling of the various executable programs
- Directory MODEL: where are source codes for reading the information on the velocity structure
- Directory RAYTRACING: where are source codes for doing two-points ray tracing
- Directory DERIVE: where are source codes for computing synthetic times and sensitivity matrix
- Directory PRECOND: where are source codes for weighting data and parameters as well as smoothing techniques
- Directory INVERSE: where are source codes for the inversion and the updating of parameters, velocity and hypocenter/blast parameters.
- Directory STATISTICS : where are source codes for doing statistics on stats, srcs and velocities
- Directory UTIL: where are various useful programs
- Directory THIRD_PARTY: where you will find different programs which may not be distributed (see first lines of each program)

Compiling and executing

Inside the subdirectory SRC, one should perform the command make. The argument should be all if you want to compile the different modules. Therefore, you should type “make all”. The execution of the Makefile will proceed through dedicated directories of the various modules and codes should be compiled. We have tested the code using the gfortran 4.8. of GNU distribution as well as the ifort 14 of INTEL. You may go down inside one module and type make if you want to recompile only this one.

II – Data description

Aside the velocity description, one has to deal with seismic stations information, sources information and travel-times information. As various dimensions and configurations could be met, after different trial experiences, we have decided to switch to binary files. This may be a difficulty but analysing unexpected outputs is also time-consuming.

Four files should be constructed.

File fsta

Station information: the file is a binary file fsta with equal length of 28 octets.
==========

Utilities are provided a2fsta and fsta2a for performing conversion from ascii to binary and from binary to ascii.

The first record has only one useful information: the number of stations nsta with six other dummy 4-octets variables.

The number of other records is, therefore, nsta, leading to a total of (nsta+1) records of 28 octets.
Each record following the first one describes a station with the following information

(1) id_sta: integer(kind=4) a unique id for the station (different between stations)
(2) x_sta: real(kind=4) the x position of the station in the reference frame
(3) y_sta: real(kind=4) the y position of the station in the reference frame
(4) z_sta: real(kind=4) the z position of the station in the reference frame
(5) dtp_sta: real(kind=4) the static delay time for P waves at this station
(6) dts_sta: real(kind=4) the static delay time for S waves at this station
(7) name: character(len=4) the label of the station in four letters (quite common for geographical identification … not used in computer codes). Having four letters is mandatory (no empty space).

The easiest way for constructing the fsta file is through a raw text file where each line describes one record (and, therefore, one station). By using the a2fsta program, one can convert this text file into the binary file used for the tomography. The reading of this program might be useful for tuned conversion of existing data file.

File fsrc

Source information: the file is a binary file fsrc with an equal length of 32 octets
==========

Utilities are provided a2fsrc and fsrc2a for performing conversion from ascii to binary and from binary to ascii.

The first record has the following information: the number of sources nsrc, the number of earthquakes neqks, the number of shots nshot, the number of blasts nblast, the number of out-of-the-box earthquakes neqks_out which may increase during iterations. Once a hypocenter moves out, there is no attempt to put it back without modifying fsrc (or starting with a new velocity structure). The total of earthquakes (neqks+neqks_out) stays constant. Three dummy variables of 4 octets pad the end of the record.

The number of other records is, therefore, nsrc, leading to a total of (nsrc+1) records of 32 octets.

Each record following the first one describes a source with the following information

(1) x_src: real(kind=4) the x position of the source in the reference frame
(2) y_src: real(kind=4) the y position of the source in the reference frame
(3) z_src: real(kind=4) the z position of the source in the reference frame
(4) t0_src: real(kind=4) the origin time of the source
(5) kp_src: integer(kind=4) position flag for quakes (continuous increment for easier debugging) (0 for others). If the flag is non-zero, we consider this event for position inversion.
(6) kt_src: integer(kind=4) origin time flag for quakes and blasts (continuous increment for easier debugging) (0 for others). If the flag is non-zero, we consider this event for origin time.
(7) in_src: integer(kind=4) status of the source (in=1 means in the box, out=0 means out). Once this number is set to zero, only specific editing manipulation will put back the event into the tomography optimization
(8) id_src: integer(kind=4) unique id of the source (should be unique). Although this identification number is arbitrary, keep it small will simplify the tracking during the tomography. An incremental strategy is advisable.

The easiest way for constructing the fsrc file is through a raw text file where each line describes one record. By using the a2fsrc program, one can convert this text file into the binary file used for the tomography.

BE AWARE THAT THIS FILE WILL BE MODIFIED THROUGH THE INVERSION PROCEDURE.

File fobs

Time observation: the file is a binary file fobs with an equal length of 24 octets
==========

Utilities are provided a2fobs and fobs2a for performing conversion from ascii to binary (fobs.asc ☾ fobs) and from binary to ascii (fobs ☾ fobs.asc).

The first record has the following information: the total number of picked times nt, the number of P wave picked times ntp and the number of S wave picked times nts. Three dummy variables of four octets pad the end of the record.

The number of other records is, therefore, nt, leading to a total of (nt+1) records of 24 octets. Records up to ntp+1 are related to P wave times and records from ntp+2 to ntp+2+nts related to S wave times.

Each record following the first one describes a data picked time with the following information

(1) id_obs: integer(kind=4) a unique integer for defining this data point
(2) t_obs: real(kind=4) arrival time of the picked wave (P or S)
(3) dt_obs: real(kind=4) precision in the picking
(4) id_src: integer(kind=4) id of the associated source
(5) id_sta: integer(kind=4) id of the associated station
(6) id_ray: integer(kind=4) id of the associated ray

The last information is built during the inversion and is not required as input: you could set it to zero. This is a way of tracking ray information while scanning the data in a loop from 1 to nt (total number of observations).

The easiest way for constructing the fobs file is through a raw text file where each line describes one record. By using the a2fobs program, one can convert this text file into the binary file used for the tomography.

This file will be modified through the inversion procedure for the id_ray parameter ONLY.

PLEASE consider that the ID of the data, id_dat, should stay the same during the investigation with one dataset.

Computer codes for the conversion of ascii files into binary files might be useful for tuning data in another format. They are in the directory UTIL under the directory SRC where you will find programs for conversion.

III – Definition of the velocity model (model.tomo)

File modelP

The P wave velocity structure, as well as the S wave velocity structure, is described in a binary file (direct-access file using the terminology of FORTRAN) with the following order of storage: x is the fastest, y the second and z the third. Only float values of the velocity are stored.

File modelS

The S wave velocity is described as well following the same grid description.

File model.head

Information regarding the velocity model is provided inside the file model.head: we have always one line of comments and then input values in the second line and so on.

This rectangular grid is attached to a reference frame with an absolute origin (xinv,yinv,zinv) related to the tomographic box. Numbers of points along the three directions, nxinv,nyinv,nzinv are defined as well as the grid stepping in the three direction hxinv,hyinv,hzinv. These quantities are provided through the input file “modelP.head” when scanned by the first computer code “model.tomo” which is dedicated to the construction of the fine eikonal cubic grid used by the eikonal solver (Podvin & Lecomte, 1991).

We assume that the origin of the forward cubic grid is identical to the origin of the tomographic box. Numbers of points along the three directions (nx,ny,nz) as well as the unique grid stepping h which is the same on both directions for cubic sampling. This stepping should be small enough for making accurate travel-time computations.

We consider a sampling dray along the ray which could be similar or smaller than the eikonal sampling h: you have often to tune it when difficulties are met for performing the two-points ray tracing.

Finally, one may define which nodes should be inverted. For the present time, we limit ourselves to minimal indexes (ixo,iyo,izo) and maximal indexes (nix,niy,niz) which means that we consider a smaller box inside the tomographic box for the inversion. Uptonow, we have used full grid strategy and, for potential use, the origin of the eikonal grid should be different from the inversion grid.

The basic input could be seen through a shell file for running this velocity model transformation.

========== FILE model.head

```
# P velocity only (1) or P & S velocities (2)
             1
# origin of the grid
  -1000.00000      -375.000000      -1000.00000
# dimensions nx,ny,nz
     65      4      25    strategy for 2D application
# samplings along x,y,z
   500.000000       250.000000       500.000000
# forward problem sampling (cube)
   250.000000
# ray sampling
   10.0000000
# starting node
  1 1 1
# ending node
  65 4 25
```

========== END OF THE FILE

Files modelP.fwd and modelS.fwd

Running the sampling code creating the forward model file modelP.fwd representing the cubic eikonal grid.

Two other files are created for exchanges with other programs

========== CREATED FILE timefd.par

```
2                                                  both P and S waves (2)
   1000          (warning only a guess)            max number of points along a ray
  16000.000000   11000.000000   -3000.000000      xinv,yinv,zinv
 113 113  41                                       nxfwd,nyfwd,nzfwd
   500.000000    500.000000    500.000000         h along the three directions (equal)
modelP.fwd                                         forward velocity model
modelS.fwd                                         forward velocity model
```

========== CREATED FILE inversion.par

```
  2                                                    both P and S waves (2)
  16000.000000   11000.000000   -3000.000000          xinv,yinv,zinv
  9   9  11                                           nxinv,nyinv,nzinv
   7000.000000    7000.000000    2000.000000          hxinv,hyinv,hzinv
modelP                                               binary file for P velocity structure
         1         1         1                       ixo,iyo,izo      index of small point
         9         9        11                       nix,niy,niz      index of high point
modelS                                               binary file for S velocity structure
========================
```

This program has to be run each time we modify the velocity structure or the file model.head.

IV – Two-points ray tracing (raytracing.tomo)

The ray tracing between the source and the station is performed through a finite difference eikonal solver (Podvin & Lecomte, 1991). The final output will be the ray leaving the source towards the station which is the end point of the ray. Two files are created: frays.tabl and frays.coor. The first one defines a pointer for accessing the initial point of a given ray as well as the number of points of this specific ray. Positions of all points are recorded in the frays.coor. In order to trace the necessary information for further steps, the look-up table of the ray which provides the necessary id when we want a direct extraction of the ray information is added to the file fobs.

Nothing else is done except tracing rays. These rays are computed either with a priority for stations when we have less stations than sources or the opposite way around. Whatever is the option, the final output of ray positions starts from the true source and finishes to the true station in the created files.

One should notice that travel-times are not computed at that stage. Only rays are estimated when one launches the program “raytrace.tomo”.

The file model.head is scanned for input parameters as well as the file timefd.par

In the file ‘timefd.par’ by the program ‘model.tomo’, we have set in a hard way to 15000 points as the maximum number of points along one ray. In the output file, it is useful to check the maximum number of points along rays as we shall allocate memory space in the next program. For the estimation of some memory requirement, we may have used a too optimistic algorithm as the rule of thumb. You may modify the content of timefd.par (or you may modify the code “model.tomo.f90” for increasing the specific number “15000”.

(TODO should be updated by the program …)

In the future, various modules could be added here as long as they provide those files ‘frays.*’ related to the two-points ray tracing. May we underline that these files should be preserved if you wish to repeat inversion WITHOUT repeating the ray tracing part? This is quite important if you want to investigate resolution of your solution.

V – Construction of the sensitivity matrix and time residues (derive.tomo)

The main purpose of this module “derive.tomo” is two-folds

- the estimation of partial derivatives of travel-times with respect to slowness (the inverse of the velocity) and to hypocenter positions.
- The estimation of the synthetic travel-times in a more accurate way than only through eikonal solvers. We are less sensitive to the grid stepping of the eikonal grid. We also compute residues to be stored in the file fdift

Doing so allows us to recomputed accurately synthetic travel-times at zero extra cost. Other reasons are behind this argument and more related to computer efficiency. These estimations of travel-times are significantly better than those computed in the previous module through the eikonal solver and, therefore, they are less dependent on the grid mesh used for the eikonal solver (at least two orders of magnitude in relative errors).

(TODO : make an option for only doing quakes / blasts locations)

Both files timefd.par and inversion.par ar used in this module.

Many files are generated in this step as we have built the linear system to be solved with or without constraints.

Information regarding the misfit

The misfit function as a rms value is appended to the file rms.file where one can trace the reduction of the unweighted rms.
rms.file: append file for raw rms estimation without any weight
rms.weig: append file for user weighted rms estimation

Information regarding the matrix A of the linear system Ax=b

fmat.x: matrix of sensitivity in a sparse format
fmat.id: pointer for the line number of the sensitivity matrix
fmat.ic: pointer for the column number of the sensitivity matrix
matrix.par: where to write the information on the linear system to be solved. This file does include the various names of fmat.* files, the number of non-zero elements of the sensitivity matrix, the total number of selected data essentially through a valid ray connecting the source and the receiver and the number of P time data (one can deduce S time data) and, finally, the number of parameters to be inverted.

Information regarding the rhs b of the linear system Ax=b

(dimension nd identical to nl number of rays)

fdift: time residuals only on validated data and related to the rhs vector of the inversion by lsqr. This file does include the difference between travel-time data and synthetic travel-time data in the current velocity model.

The difference of travel-times is related to (picked date + station_static-origin_date)-synthetic time. Be aware of the difference between a date (12-01-1980:04-45-23-000) and a time (234.00 sec). This file will be used for the inversion as the rhs of the system to be solved.

This file is related to the number of validated data through couples (source, station) with a successful ray tracing (nl*) and a non-zero number of partial derivatives (nd*). In other words, we have nl values where the number of rays connecting (source,stations) denoted by nl. When we do find partial derivatives, we have an additional counting number, named nd

(not explicit yet), which is related to inverted data: nd < nl < nt which is the total number of data nt.

fidnl: storage of id_dat for tracking which data we consider in the linear system Ax=b (dimension nd)
frays.xlen: estimation of the ray length for each couple (source,station) nd values
used for weighting related to the ray length (PRECOND)

Information regarding residues for each picked data
fresi: raw time residuals for all data (-999. means that there is no related rays)
fresu: user-weighted time residuals for all data (-999. means that there is no related rays).
These files have a dimension of nt (the number of data)

fnorm: fake file of nc values equal to one (nc is the number of model parameters) if one jumps over the next preconditioning step directly to the inversion. This file is normally created by preconditioning for balancing values between parameters. Therefore we need it.

VI – Weighting and smoothing between parameters (precond.tomo)

In this present form, one need a preconditioning of the linearized system to be solved as we have to mix velocity parameters (P & possibly S velocities), hypocenter coordinates and hypocenter origin date. Additionally, we may need to smooth through penalty approach.

We do expect improvements on this module in the future as combining different kinds of parameters should be done in another way. In any case, this scaling might be useful for specific applications.

We transform the system to be solved Ax=b into a new system

$$A'' y = D1.A'.D2\ y = D1\ b = b''$$
$$x = D2\ y$$

where D1 and D2 are diagonal matrices while the matrix A' has a bigger size as the matrix A when considering smoothing through penalty condition.

The matrix D2 is stored in the file fnorm (dimension nc of model space corresponding to columns) as we need it to get back the vector x.

File inversion.head

We use always a line of comments before the input line

========== inversion.head

```
# residual weighting
   20.00000000       40.00000000
# ray weighting
   100000.00000       200000.00000
# activate the smoothing
```

```
           1
# Laplacian option 1 to 3
           3
# coefficient of smoothing in x
   5.000000
# coefficient of smoothing in y
   5.000000
# coefficient of smoothing in z
   5.000000
# weights and statistics flag (0 == weight to 1 and 1 == fwei file)
1  0     (statistics not yet done: always 0)
# ratio for P wave derivative, ratio for S wave
1. 1.
# ratio for hypocenter, ratio for origin time
1. 1.
# damping parameter for lsqr   (this starting line for inversion)
 0.500000000        (here nearly none)
# maximum number of lsqr iterations
         500
# VP perturbation
   50.0000000
# VS perturbation
   35.0000000
# Horizontal shift maximum (m)
 150.00000000
# vertical shift maximum (m)
  50.00000000
# original time shift maximum (s)
  0.5
=========================================
```

Be aware that files fmat.* are modified. If one wishes looping over the preconditioning without recomputing the sensitivity matrix, one has to save initial fmat.* files.

fresv: new file related to user-defined weighted residues (same as the fresu estimated in DERIVE module … put here if one wishes to modify the weight between DERIVE and PRECOND)

fresw: new file related to all-weighted residues (including weights on residual values and lengths on rays)

fidnl: new scratch file for the association between data vector and rhs vector for the inversion

VII – Inversion using LSQR (inverse.tomo)

The next step is the inversion procedure using a conjugate gradient approach as promoted through the LSQR software.

Updating velocity models as well as earthquakes parameters and blasts parameters is achieved in this module. This is the final stage of the external iteration over models.

File inversion.head

We read lines of this file starting at line 21.

```
==================================== line 21
… continuing the file inversion.head
# damping parameter for lsqr   (this starting line for inversion)
 0.500000000        (here nearly none)
# maximum number of lsqr iterations
         500
# VP perturbation
   50.0000000
# VS perturbation
   35.0000000
# Horizontal shift maximum (m)
 150.00000000
# vertical shift maximum (m)
  50.00000000
# original time shift maximum (s)
  0.5
========================================
```

We allow for the damping parameter of the normal equation to be solved **A" y = b"** while using LSQR approach.
We need also to have a maximum in the iterations inside LSQR. Finally we could put hard constraints in the perturbation of Vp, Vs, horizontal shifts of hypocenters, vertical shift of hypocenters as well as time shifts for original times.

VIII – Practical information

Before starting your application, it is recommended to practice through synthetic examples where you can expect from these codes of travel time tomography.

The directory SYNTHETIC_SEISMICS is a skeleton directory to help you in designing your synthetic examples and all examples you will find in directories RUN* provided have been designed in this directory. (Obsolete now … you have to build up your own synthetics)

You may find that splitting tomography workflow in five elementary steps increases the complexity of the learning procedure. Other codes/softwares are often based on a single code which performs everything. We have preferred the splitting as we can design specific workflows easier to manipulate through loops monitored by the shell environment. We use the standard "sh" but one can think about python for improving the input design or the output analysis.

The design of initial parameters is important and one can think about sampling significantly the parameter space for designing few initial models. One should be aware that the output depends on the input and delayed travel times do bring an infinity of possible solutions. Therefore, the choice of the initial model will drive the optimization to the nearest minimum.

Moreover, we can perform spike tests once we believe we have obtained a reliable final result by making slight perturbation in the parameter space which allows us to avoid doing the ray tracing again. Checkboard tests are also possible, although spike tests bring more information. Bootstrapping techniques could be performed as well.

They are workflows for selecting the initial model and for qualifying the quality of the solution. They take benefit of the flexibility of the splitting as you can design the workflow you want which should not have been thought when considering a single program.

IX – Conclusion

These codes are those designed around 1995 using first F77 writing. It has been converted around 2002 using simple F90 writing.

Better understanding of these tomographic problems would certainly lead to better writing of these codes as they are combined inputs from many students.

We may hope that more efficient algorithms could be available including topography, anisotropy, attenuation … and so on.

Good luck …