EIP Sponsors User Guide no. 41, 18-12-2019

3D Monte Carlo Tomography Software Package

MCTomo User Guide

Xin Zhang[1] and Andrew Curtis[1]

[1] School of GeoSciences, University of Edinburgh,

James Hutton Road, Edinburgh, *EH9 3FE*, United Kingdom

E-mail: *x.zhang2@ed.ac.uk, andrew.curtis@ed.ac.uk*

SUMMARY

This manual describes how to install and use the 3D Monte Carlo Tomography Package (MCTomo) which implements a fully 3D Monte Carlo Tomography method using both surface and body wave data. It employs Voronoi tessellation to parameterize the subsurface, and the reversible jump Markov chain Monte Carlo method to sample the parameter space. It supports 3D travel time tomography using body wave arrival (or travel) times, surface wave frequency-dependent travel times, as well as joint inversion using both body and surface wave data.

1 PACKAGE OVERVIEW

In a variety of geoscientific applications we require 3D models of properties of the Earth's interior, and the corresponding model of uncertainties to assess their reliability. Seismic tomography is a technique which has been used widely to produce such three-dimensional models.

Since tomography is significantly nonlinear, Monte Carlo (MC) sampling methods are often used to solve tomographic problems. MC methods generate a set (or chain) of samples from the posterior probability distribution function (pdf) describing the probability of models given the observed data; thereafter these samples can be used to estimate useful information about that pdf (mean, standard deviation, etc.). The methods are quite general from a theoretical point of view so that in principle they can be applied to any tomographic problems. They have been extended to trans-dimensional inversion using the reversible jump Markov chain Monte Carlo (rj-McMC) algorithm, in which the number of parameters (hence the dimensionality of parameter space) can vary in the inversion. Consequently the parameterization itself can be simplified by adapting to the data, which improves results on otherwise high-dimensional problems.

This package implements a 3D tomography method using rj-McMC using both body and surface wave data. The 3D model is parametrized using 3D Voronoi tessellation (Sambridge et al. 1995; Kennel 2004; Jamin et al. 2018). The codes are written mainly in Fortran but combine some C++ codes. The 3D Voronoi tessellation is based on the Computational Geometry Algorithms Library (CGAL, `https://www.cgal.org`) which is a software project that provides efficient and reliable geometric algorithms in the form of a C++ library, and the Kdtree2 method (Kennel 2004). The codes use the fast marching method to calculate first arrival times for body waves (Rawlinson & Sambridge 2004; Valero-Gomez et al. 2013). For surface waves we use a two-step forward modelling method to simulate inter-receiver or source-receiver dispersion curves (Zhang et al. 2018): we do this by combining the modal approximation method (Herrmann 2013) and the 2D fast marching method. The surface wave modal approximation code (in the folder *surfmodes*) is from Computer Programs in Seismology written by Herrmann, R.B. (`http://www.eas.slu.edu/eqc/eqccps.html`). However, this code fails to calculate the correct Rayleigh/Love modes in the presence of low velocity layers when their velocities are lower

than the velocity of the top layer. We therefore impose a condition on the *prior* probability distribution over velocity models: they must have the smallest shear-velocity in the top layer (Zhang et al. 2018, 2019a). The 2D fast marching code is from Nicholas Rawlinson (`http://rses.anu.edu.au/seismology/soft/fmmcode/`).

2 INSTALLATION

2.1 Required libraries

The package is based on the CGAL library which can be built and installed by following the CGAL installation manual (`http://doc.cgal.org/latest/Manual/installation.html`). On MAC OS X, it can be installed in the following way:

sudo port install cgal

On Debian/Ubuntu use:

sudo apt-get install libcgal-dev

The MCTomo package also provides an option to include the NetCDF4 library for the support of storing generated model samples in a portable way for later use. However, this is not necessary. If you do not want to use NetCDF4 library, just leave the NETCDF variable undefined in the Makefile. On most platforms, the NetCDF4 library can be installed through a package management program, such as rpm, yum, homebrew, macports, adept and others. Alternatively, you can download it from `http://www.unidata.ucar.edu/software/netcdf/docs/`.

Once you have installed the CGAL and NetCDF4 library, open the *src/Makefile* and add the include and link path of CGAL and NetCDF4 if they are in a user-defined location.

2.2 Installation

Once you have installed the necessary libraries, in the directory *src* simply type

make

This will compile the package and put the executables in the *bin* directory.

There are two executables: *MCTomo* and *sample*. Executable *MCTomo* is the main program

used to generate samples, and *sample* is an assistant program to post-process those samples after they have been generated (to calculate means, standard deviations, etc.)

The default compiler is set to be *gnu*. If you want to use *intel* compilers, set the "COMPILE = intel" in the Makefile. The codes use Fortran 2003 standards and C++ 11 standards, so make sure your compilers support those standards.

3 INPUT AND OUTPUT

3.1 Input Files

MCTomo.inp:

This defines the control parameters for the package. It uses Fortran's namelist, so every line beginning with "!" is a comment. The input file is well commented for each control parameter.

MCTomo.inp

```
!
! This is the input file for the MCTomo package.
! It uses Fortran's namelist to input arguments. So every line beginning with ! is a
! comment. This document is well commented so that you do not need to refer
! any additional manual to use it.
!
! Grid Settings
!
&GRID_SETTINGS
 ! min and max values of x coordinate [km]:
 GRID%XMIN= -3.000000000000000     ,
 GRID%XMAX=  2.800000000000000     ,
 ! min and max values of y coordinate [km]:
 GRID%YMIN= -3.500000000000000     ,
 GRID%YMAX= 3.5000000000000000     ,
 ! min and max values of z coordinate [km]:
 GRID%ZMIN=  0.0000000000000000     ,
 GRID%ZMAX=  3.00000000000000     ,
 ! grid size in x, y, z: Number of grid points:
 GRID%NX=        59,
 GRID%NY=        71,
```

```
 GRID%NZ=        101,
 ! water layer thickness [km]: set to zero if no water layer:
 GRID%WATERDEPTH=     0,
 ! scaling factor of vertical aspect, used to balance horizontal and vertical difference in scaling
 ! For example, if the maximum horizontal dimension of the model is five times that of the vertical
 ! dimension, set GRID%SCALING=5:
 GRID%SCALING= 5,
 /
!
! Likelihood Settings
!
&LIKELIHOOD_SETTINGS
 !
 ! Data Settings
 !
 ! datatype: 0 for P or S waves only, 1 for P & S waves, 2 for surface waves only,
 ! 3 for joint P & S and surface waves
 ! (joint P or S and surface waves is not an option yet):
 LIKE_SET%DATATYPE=3,
 ! sources file, receivers file and travel times file for body waves
 ! If body waves are not used, the following files are simply ommited:
 LIKE_SET%BSOURCES_FILE='../../bsources.dat',
 LIKE_SET%BRECEIVERS_FILE='../../breceivers.dat',
 LIKE_SET%BDATA_FILE='../../btimes.dat',
 ! sources file, receivers file and travel times file for surface waves
 ! If surface waves are not used, the following files are simply ommited:
 LIKE_SET%SSOURCES_FILE='../../ssources.dat',
 LIKE_SET%SRECEIVERS_FILE='../../sreceivers.dat',
 LIKE_SET%SDATA_FILE='../../stimes_phase3.dat',
 !
 ! Control Parameters for Surface Wave modes Calculation
 !
 ! phase velocity step when searching phase velocities
 ! (the phase velocity is found by a brute force search method in an appropriate range):
 LIKE_SET%DPHASEVEL=  1.00000000E-003,
 ! 0 for love waves, 1 for rayleigh waves. Using both types of data is not an option yet:
 LIKE_SET%RAYLOV=          1,
 ! 0 for phase velocity, 1 for group velocity
 ! Note: in principle one can use both phase velocity and group velocity data. However
```

```
! phase velocity can usually be measured more precisely and also modelled more precisely, so
! we recommend using phase velocity if possible. Otherwise, (if phase velocity
! cannot be measured), use group velocities. Therefore, using both phase velocity and
! group velocity together is not implemented as an option here:
LIKE_SET%PHASEGROUP=          0,
! tolerance when searching phase velocities:
LIKE_SET%TOL=  1.0E-006,
!
! Fast Marching Method Settings
!
! using straight rays(1) or not(0); straight rays are only an option for surface waves:
LIKE_SET%isStraight=     0,
! dicing in x and y (refinement of grid);
! For example, if gridx is equal to 2, then the size of the grid cell in the x direction
! will be half of the original size
! only an option for 2D fast marching method:
LIKE_SET%GRIDX=          1,
LIKE_SET%GRIDY=          1,
! source refinement (1) or not (0);
! currently only an option for 2D fast marching method:
LIKE_SET%SGREF=          1,
! dicing (refine level) and extent (from source) of source refinement
! If above SGREF option is set to 1, the number of grid cells around the source will be increased
! by a factor of SGDIC (the size of the grid cell will be the original size divied by SGDIC).
! This process is conducted from source location to SGEXT number of cells in original grid:
LIKE_SET%SGDIC=          4,
LIKE_SET%SGEXT=          8,
! the order of fast marching, 0 for first order and 1 for mixed order
! If setting to 1, the code uses second order if possible (using first order at boundaries):
LIKE_SET%ORDER=          1,
! narrow band size (0-1) as a fraction of nx x ny, used to allocate memory
! If you cannot decide it, set to 1. Only an option for 2d fast marching:
LIKE_SET%BAND= 0.50000000000000000     ,
! If you need to estimate the source locations, set to 1 meaning that we perform fast marching from
! receivers to sources and the travel time field for each receiver is stored in memory. This provides
! fast forward calculations when changing source locations only since travel times can be drawn by
! interpolation of the stored travel time field.
! 0 means that the travel time field is not stored and every time the code re-calculates it:
LIKE_SET%DYNAMIC=          1,
```

```
 /
!
! McMC Settings
!
&MCMC_SETTINGS
 !
 ! General Settings
 !
 ! 0: randomly generate an initial model with velocity increases linearly with depth
 ! 1: sampling starts at initial 1D model defined in the file given by mcmc_set%initial_model:
 MCMC_SET%INITIALISE=         1,
 ! The minimum and maximum shear velocity Vs [km/s] for generating a model whose velocities are linearly
 ! increasing with depth if above set to 0:
 MCMC_SET%INIT_VSMIN=         0.4,
 MCMC_SET%INIT_VSMAX=         1.5,
 ! initial model file (if above initialise set to 1):
 MCMC_SET%INITIAL_MODEL='inimodel.dat',
 ! Chain id (used if mpi is not available):
 MCMC_SET%PROCESSOR=          1,
 ! Burn-in period. It is fine to change burn-in period if starting from the
 ! end point of previous runs. Make sure you reset it to be bigger than the total
 ! samples already generated. This will make the code re-sample the chain from the
 ! new burn-in value, counting from the first sample of the first run:
 MCMC_SET%BURN_IN=          500000,
 ! Thining of the chain i.e. retain every THINth model:
 MCMC_SET%THIN=             100,
 ! total samples for current run:
 MCMC_SET%NSAMPLES=         100000,
 ! running mode: 1 for the first run; >2 for continous re-running, starting where last run stopped;
 ! 0 to resume an occasionly interrupted run (e.g. interrupted because the maximum wall time has reached):
 MCMC_SET%RESUME=          2,
 ! time step for displaying intermediate information of the run:
 MCMC_SET%DISPLAY=       5000,
 ! time step for saving intermediate results in case you wish to resume the run later:
 MCMC_SET%runtime_step=       5000,
 ! noise parameterisation: 0 for fixed noise, or
 ! 1 for noise that is linearly related to time/source-receiver length
 MCMC_SET%SIGDEP=          1,
 !
```

```
! Prior Probability Distribution
!
! Max. and Min. number of cells. The minimum number should be at least 4 for 3D Delaunay triangulation:
MCMC_SET%NCELL_MAX=                300,
MCMC_SET%NCELL_MIN=                  20,
! Min. and Max. P velocity Vp [km/s]:
MCMC_SET%VPMIN=  1.60000000000000000     ,
MCMC_SET%VPMAX=  6.0000000000000000     ,
! Min. and Max. shear velocity Vs [km/s]:
MCMC_SET%VSMIN=  1.00000000000000000     ,
MCMC_SET%VSMAX=  4.0000000000000000     ,
! If noise level is not fixed. noise = n0 x traveltime + n1.
! Min. and Max. n0 values for body waves:
MCMC_SET%BN0_MIN=  0.0001000000000000000   ,
MCMC_SET%BN0_MAX=  0.0500000000000000     ,
! Min. and Max. n1 values for body waves:
MCMC_SET%BN1_MIN=  0.000000000000000     ,
MCMC_SET%BN1_MAX=  0.2000000000000000     ,
! Min. and Max. n0 values for surface waves:
MCMC_SET%SN0_MIN=  0.0001000000000000000   ,
MCMC_SET%SN0_MAX=  0.200000000000000     ,
! Min. and Max. n1 values for surface waves:
MCMC_SET%SN1_MIN=  0.000000000000000     ,
MCMC_SET%SN1_MAX=  0.2000000000000000     ,
!
! Proposal Distribution
!
! Proposal kernel for birth/death step, 1 to use a prior kernel, and 0 for a Gaussian kernel.
! Generally, a prior kernel is recommended:
MCMC_SET%KERNEL=           1,
! Proposal standard deviation of Vp when changing a velocity for shallower half (z < depth/2):
MCMC_SET%SIGMA_VP= 0.060000000000000000     ,
! Proposal standard deviation of Vp when changing a velocity for deeper half (z > depth/2):
MCMC_SET%SIGMA_VP2= 0.40000000000000000     ,
! Proposal standard deviation of Vs when changing a velocity for shallower half (z < depth/2):
MCMC_SET%SIGMA_VS= 0.030000000000000000     ,
! Proposal standard deviation of Vs when changing a velocity for deeper half (z > depth/2):
MCMC_SET%SIGMA_VS2= 0.20000000000000000     ,
! If noise level is not fixed. noise = n0 x traveltime + n1.
```

```
! Proposal standard deviation for n0 of body waves when changing noise level
MCMC_SET%SIGMA_BN0=  1.00000000E-003,
! Proposal standard deviation for n1 of body waves when changing noise level:
MCMC_SET%SIGMA_BN1=  4.00000000E-003,
! Proposal standard deviation for n0 of surface waves when changing noise level:
MCMC_SET%SIGMA_SN0=  4.00000000E-002,
! Proposal standard deviation for n1 of surface waves when changing noise level:
MCMC_SET%SIGMA_SN1=  4.00000000E-002,
! Proposal standard deviation for move step: pd for shallower half and pd2 for deeper half
! sigma is actually pd/100 multiplied by the range on each axis:
MCMC_SET%PD=  2,
MCMC_SET%PD2=  10,
!
! Source Locations.
!
! Source locations and times are generally reasonably well-determined using linearised inversion,
! so the prior is chosen to be a small box centered at the initial location for each source.
! Box dimensions xwidth, ywidth and zwidth are given in km, and source time box width twidth is in seconds:
! Set MCMC_SET%LOCATE=0 means source locations and times are fixed to initial values:
MCMC_SET%LOCATE=  1,
MCMC_SET%XWIDTH=  1,
MCMC_SET%YWIDTH=  1,
MCMC_SET%ZWIDTH=  1,
MCMC_SET%TWIDTH=  0.5,
! Proposal standard deviation for x, y, z of locations and origin time t, where proposals are additionally
! constrained to lie within the above box:
MCMC_SET%SIGMA_X=  0.15,
MCMC_SET%SIGMA_Y=  0.15,
MCMC_SET%SIGMA_Z=  0.15,
MCMC_SET%SIGMA_T=  0.02,
! The number of sources to be changed in every source change step
MCMC_SET%NLOC=  10,
!
! Parallel Tempering Related Settings
!
! Using parallel tempering (1) or not (0):
MCMC_SET%TEMPERING = 0,
! This file stores the temperatures:
MCMC_SET%TEMPERFILE = 'temperature0.dat',
```

```
! From which sample should we start the parallel tempering:
MCMC_SET%TEMPERING_START = 5000,
! Time step to perform parallel tempering. A small step will lead to a higher exchange rate but slower code
! due to synchronization between chains. So an intermediate step (~100) is recommended:
MCMC_SET%TEMPERING_STEP = 100,
! The jump type between different temperatures:
! 0, randomly choose a temperature
! 1, randomly choose a temperature between the two neighbours
! 2, choose the nearest temperature:
MCMC_SET%JUMP_TYPE= 0,
! The number of chains whose temperature = 1:
MCMC_SET%NUMBER_OF_1S= 16,
! The total number of chains (including those with temperature 1):
MCMC_SET%NUMBER_OF_TEMPERATURES= 24,
/
```

bsources.dat and ssources.dat:

These are files that define body wave and surface wave sources. The names are given by LIKE_SET%BSOURCES_FILE and LIKE_SET%SSOURCES_FILE in the MCTomo.inp file. Both files follow the same format. Each begins with one line with two numbers which describe the number of sources and the dimension of each source – i.e. x[km], y[km], z[km], origin time[s]. The dimension of one source is between two (x,y) and four (x,y,z,t). A typical source file looks like:

sources.dat

```
5   4                << No. of sources and No. of dimensions
1.0  1.0  3.00  0.0   << x[km] y[km] z[km] t[s] of the first source
1.2  1.1  3.25  0.0   << x[km] y[km] z[km] t[s] of the second source
1.4  1.2  3.50  0.0
1.6  1.3  3.75  0.0
1.8  1.4  4.00  0.0
```

breceivers.dat and sreceivers.dat:

These are files that define receivers for body waves and surface waves. The names are given by like_set%breceivers_file and like_set%sreceivers_file in the MCTomo.inp file. They have the same

format as the source files except that there is no origin time for receivers. So the dimension of one receiver is between two (x,y) and three (x,y,z).

btimes.dat and stimes.dat:

These are travel time data files for body waves and surface waves. The names are given by like_set%bdata_file and like_set%sdata_file in the MCTomo.inp file.

The first line contains the number of travel times for one source-receiver pair. In the body wave case, it should be 1 (only P-wave or S-wave data are used) or 2 (both P-wave and S-wave data are used). In the surface wave case, this number is the number of frequencies used.

The second line contains the frequencies[Hz] used in the inversion for surface waves. For body waves, this is ignored – just set it to any numbers you like (e.g. 1.0 and 2.0 for P-wave and S-wave).

Then travel times follow for each source-receiver pair. The order is first source and first receiver, first source and second receiver, ..., and so on. For each source-receiver data set, it begins with a number to show whether there are data (1) or not (0). If there are travel times, then travel times follow for each frequencies or wave types. Each line contains two columns with the travel time and its noise. If the travel time for a specific frequency or wave type is not available, set it to -1.

A typical body wave data file looks like:

bodytimes.dat

```
         2                       << No. of wave types (P and shear wave arrival times)
 0.00000000E+000  0.10000000E+001
         1                       << The first source and first receiver pair has data
 0.20555054E+001  0.42038185E-001   << P wave arrival time and noise
 0.36292020E+001  0.73450048E-001   << S wave arrival time and noise
         0                       << There is no data for the first source and second receiver
         1
 0.21147335E+001  0.43249650E-001
 -1.0             0.01              << There is no shear wave arrival time, set the time to -1.0
         1
 0.16282547E+001  0.32377696E-001
 0.29416827E+001  0.56553029E-001
         1
```

```
 0.15685067E+001  0.30971902E-001
 0.26375537E+001  0.54073814E-001
          1
 0.16676683E+001  0.33929473E-001
 0.30217724E+001  0.59239297E-001
 ...
```

And an example of a surface wave data file:

surftimes.dat

```
 11   << No. of frquencies   << Followling line contains frequencies in Hz
 2.000000  1.000000  0.500000  0.333333  0.250000  0.200000  0.166667  0.142857  0.125000  0.111111  0.100000
 0                     << There is no data for first source and first receiver
 1                     << There are data for first source and second receiver
0.992239  0.010000      << travel time and noise at the first frequency
-1.0      0.010000      << There is no data for the second frquency
0.675822  0.010000
0.620985  0.010000
0.557094  0.010000
0.520333  0.010000
0.507771  0.010000
0.496277  0.010000
0.473059  0.010000
0.478113  0.010000
0.491685  0.010000
 1
1.816842  0.010000
1.496592  0.010000
1.221621  0.010000
1.109668  0.010000
1.017638  0.010000
0.942271  0.010000
0.929245  0.010000
0.906720  0.010000
0.888402  0.010000
0.890167  0.010000
0.866338  0.010000
 1
```

```
2.353700  0.010000
1.936386  0.010000
1.599088  0.010000
1.437043  0.010000
1.304535  0.010000
1.251953  0.010000
1.193827  0.010000
1.188380  0.010000
1.181125  0.010000
1.163759  0.010000
1.157870  0.010000
...
```

---

inimodel.dat

The 1D initial model used to generate initial models for each chain. The filename is provided by mcmc_set%initial_model. In this example it is called "inimodel.dat". When this file is used (i.e. mcmc_set%initialise=1), the program randomly generates points for the Voronoi sites whose velocity is set to be the velocity at the same location in the 1D initial model. The file begins with a header line which defines the number of rows and the number of columns of this file. Each layer per line defines the depth [km], Vp [km/s] and Vs [km/s]. A typical 1D initial model file looks like:

initialmodel.dat

```
7  3                << No. of layers and No. of columns
0.000  1.9  1.28    << Depth[km]  Vp[km/s]  Vs[km/s]
0.06   2.75  1.54
0.135  3.10  1.74
0.275  3.5   1.97
1.02   4.2   2.36
1.35   5.2   2.92
2.75    6.0   3.37
```

---

temperatures0.dat

This file contains the temperature profile for parallel tempering, one temperature per line. The number of temperatures used is set by mcmc_set%number_of_temperatures (total number of temperatures) and mcmc_set%number_of_1s (the number of chains whose temperatures are equal to 1). Only those temperatures that are not equal to one should be listed in the file. The file name is given by mcmc_set%temperfile in the MCTomo.inp. A typical temperature file looks like:

temperature0.dat

```
1.0020
1.0159
1.0557
1.1429
1.3230
1.7297
3.0296
4.000
```

chains.inp

This file contains the burn-in period for the *sample* post-processing program. It begins with a line showing the number of chains you intend to use, and follows with chain id and its burn-in period in each line. The *sample* program will recalculate the mean and standard deviations for the given chains using the specific burn-in periods. A typical file looks like:

chains.inp

```
10            << No. of chains
1   1500000   << chain id and burn-in period
2   1500000
3   1500000
4   1500000
5   1500000
6   1500000
7   1500000
8   1500000
9   1500000
10   1500000
```

3.2 Output Files

The *MCTomo* program generates outputs into two folders: *Log* and *Results*. The log files are in the *Log* directory. The samples generated are stored in the *Results* directory. The details of output files in the *Results* directory are as follows:

likelihood_*.dat

These files are text files each of which stores likelihoods and misfits for each Markov chain. '*' represents the chain id. Each line contains three columns: negative logarithm likelihood value, weighted misfits, and unweighted misfits respectively.

ncells_*.dat

These are text file each of which stores number of cells for each chain.

samples_*.dat

These files are Binary files each of which stores information for each chain, which could be used later for other processing (e.g. using the *sample* program).

temperatures_*.dat

If parallel tempering is enabled, this file contains temperatures for each chain. Each line has four columns: swapping status (1 is accepted), the temperature of the current chain, the temperature of the other chain and the temperature of this sample after swapping.

Directory ini

This directory contains initial models generated for each chain in both text and binary format. The text files contains six columns per line: x[km] y[km] z[km] vp[km/s] vs[km/s] density[g/cm^3] with a top line showing the number of cells.

Directory last_run

This directory contains necessary information for the latest run so that the program can be started from this point again (e.g., by setting mcmc_set%resume=2 in *MCTomo.inp* file).

Directory resume

This directory contains necessary information for the current run so that it can be resumed from an undesired interruption (e.g., by setting mcmc_set%resume=0 in *MCTomo.inp* file).

The sample program

Once the *MCTomo* program has finished, the *sample* program can be used to calculate the mean and standard deviations if you want a different burn-in period. This program uses the outputs of *MCTomo* and the same control file along with a burn-in period file *chains.inp* (see above), simply run it in the same directory where you ran *MCTomo*. Note that if parallel tempering is enabled, it will also calculate the mean and standard deviations using all of the chains, including those chains with temperatures not equal to 1 using the importance sampling method (Geyer 1994). The output files from this program is stored in the *Results* directory as follows:

Unweighted_average.dat and Unweighted_std.dat

Text files containing the mean and standard deviations calculated from only those chains with temperatures equal to one if parallel tempering is enabled. The first line shows the number of grid points for each dimension: *nz, ny, nx*. The 3D array is then stored in column order (a $nz \times ny \times nx$ array): a file contains nz columns and $ny \times nx$ rows.

Weighted_average.dat and Weighted_std.dat

Text files which contain the mean and standard deviations calculated from all chains including chains with temperature not equal to 1 if parallel tempering is enabled. The file format is the same as above.

4 VISUALIZATION

The mean velocity model and its standard deviation are calculated based on a regular dense grid, which makes them easy to visualize (they are simply 3D arrays). Under the directory *plot*, you can find some Matlab scripts and Python scripts to visualize models. If you want to draw a 3D Voronoi diagram, the Matlab toolbox Multi-Parametric Toolbox 3 (MPT3, `http://people.ee.ethz.ch/~mpt/3/`) works well for this purpose.

5 EXAMPLES

You can find two 3D tomography examples in the *example* directory. Example 1 is a simple synthetic test described in Zhang et al. (2018) using modelled Rayleigh wave phase velocities. Details can be found in the directory *example/example1*. The second example is a real data example from a mining site as reported in Zhang et al. (2019b). This example contains inversions that use only body wave data, only surface wave data, and both types of data. Details are in the directory *example/example2*.

Note that there is also a 2D version of the above codes. The 2D tomography code is in the directory *mcmc2d*. To install it, simply type "make" in that directory. Usage is the same as in the 3D case. A 2D example is provided in the directory *example/2dExample*.

ACKNOWLEDGMENTS

The authors thank the Edinburgh Interferometry Project sponsors (Schlumberger, Equinor and Total) for supporting this research. This work used the Cirrus UK National Tier-2 HPC Service at EPCC (http://www.cirrus.ac.uk).

TERMS OF USE AND DISCLAIMER

This code is freely distributed under GNU GPLv3 for academic and industrial research. To the best of our knowledge this code is correct, error-free as of December 2019. We can not guarantee its accuracy however. If after your own detailed investigation you believe there are errors/bugs in the code or examples, please contact Xin Zhang at x.zhang2@ed.ac.uk, and/or Andrew Curtis at Andrew.Curtis@ed.ac.uk.

REFERENCES

Geyer, C. J., 1994. Estimating normalizing constants and reweighting mixtures, *Technical Report*.

Herrmann, R. B., 2013. Computer programs in seismology: An evolving tool for instruction and research, *Seismological Research Letters*, **84**(6), 1081–1088.

Jamin, C., Pion, S., & Teillaud, M., 2018. 3D triangulations, in *CGAL User and Reference Manual*, CGAL Editorial Board, 4.12 edn.

Kennel, M. B., 2004. Kdtree 2: Fortran 95 and c++ software to efficiently search for near neighbors in a multi-dimensional euclidean space, *arXiv preprint physics/0408067*.

Rawlinson, N. & Sambridge, M., 2004. Multiple reflection and transmission phases in complex layered media using a multistage fast marching method, *Geophysics*, **69**(5), 1338–1350.

Sambridge, M., Braun, J., & McQueen, H., 1995. Geophysical parametrization and interpolation of irregular data using natural neighbours, *Geophysical Journal International*, **122**(3), 837–857.

Valero-Gomez, A., Gomez, J. V., Garrido, S., & Moreno, L., 2013. The path to efficiency: fast marching method for safer, more efficient mobile robot trajectories, *IEEE Robotics & Automation Magazine*, **20**(4), 111–120.

Zhang, X., Curtis, A., Galetti, E., & de Ridder, S., 2018. 3-D Monte Carlo surface wave tomography, *Geophysical Journal International*, **215**(3), 1644–1658.

Zhang, X., Hansteen, F., Curtis, A., & de Ridder, S., 2019a. 1D, 2D and 3D Monte Carlo ambient noise tomography using a dense passive seismic array installed on the North Sea seabed, *Journal of Geophysical Research: Solid Earth*.

Zhang, X., Roy, C., Curtis, A., Nowacki, A., & Baptie, B., 2019b. 3D Tomographic Monte Carlo joint inversion of earthquake body wave travel times and ambient noise surface wave dispersion – imaging the subsurface using induced seismicity and ambient noise, *submitted*.