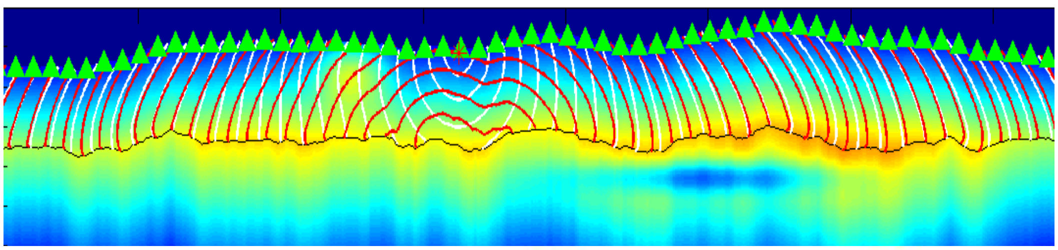
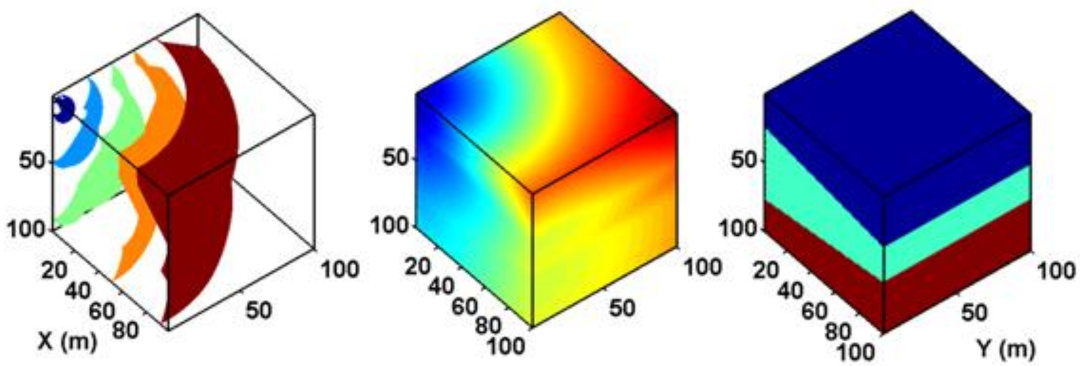
PFAST

User Manual

Version 1.1

PFAST: A High Performance Software Package for Transmission and Reflection Seismic Traveltime Tomography





By

Jun-Wei Huang

Contents

Summary

PFAST is the abbreviation of Parallel Fast sweeping method based Adjoint Seismic Tomography. This document provides the basic tutorial for users to quickly start using the program (PFAST)

(1) to obtain traveltime in 2-D/3-D arbitrarily heterogeneous isotropic models,
(2) to perform conventional first-arrival traveltime tomography in 2-D/3-D,
(3) to perform reflection-arrival tomography in 2-D (for version 1.1), and
(4) to perform joint tomography using both first and reflection arrivals in 2-D (for version 1.1).

The essential algorithm of PFAST is based on 3 ingredients:

a) Fast-sweeping method (FSM), a grid-based Eikonal equation solver,
b) Huygens' Principle to calculate reflection traveltime using FSM, and
c) The adjoint method, to obtain the gradient of the non-linear objective function without time consuming evaluation of Fréchet derivative matrix.

The details of the theory can be found in a paper published in Geophysical Journal International (Huang and Bellefleur, 2012) and one application to the arctic permafrost region with complex near surface thermokarst lakes (Huang and Bellefleur, 2011). Currently PFAST supports joint tomography in 2-D isotropic model and diving wave tomography in 3-D isotropic models. Future development includes implementation of the joint tomography algorithm in 3-D models and extension of the FSM to handle irregular meshes and anisotropic velocity models.

The source codes are written primarily in C (a little bit of C++ function overload) with MPI support. The codes have been tested successfully using OpenMPI v1.4.3 on Ubuntu 11.04. Matlab™ scripts are provided to generate the models and visualize the results. All mfiles were created on Matlab 7.12 (R2011a) and should be compatible to other versions.

The codes are grouped into one main program file and two header files, one including FSM related subprograms and one adjoint method related subprograms. Matlab™ scripts to generate models and view results are distributed with the software package.

To build this program one only needs to compile the three source files. Advanced users who may need to alter the codes are encouraged to save subprograms into individual files and create a `makefile` to build the software. An introduction of creating a `makefile` can be found at GNU Operating System website[1].

PLEASE DO NOT DISTRIBUTE. PLEASE REFER OTHER INTERESTED USERS TO THE AUTHOR.

[1] http://www.gnu.org/software/make/manual/make.html (accessed on Jan 29 2012).

Author : Jun-Wei, Dr. Huang,
Current Address: 235 Bloor Street East, Toronto M4W3Y3, Canada
Homepage: http://www.junweihuang.info /

If we want to publish results calculated with this program please give a reference to the aforementioned papers.

History of modifications (2-D):

* 21.02.2010 Version 1.00 original implementation of fast sweeping
Jun-Wei Huang
* 08.03.2010 Original Implementation of inversion
Jun-Wei Huang
* 22.03.2010 Version 1.01 Parallelization
Jun-Wei Huang
* 07.04.2010 Version 1.02 Implement Nonlinear Conjugate Gradient Method, Hestenes-Stiefel scheme
Jun-Wei Huang
* 11.04.2010 Version 1.03 Implement Strong Wolfe Condition for line searching
Jun-Wei Huang
* 12.04.2010 Version 1.04 Implement Nonlinear Conjugate Gradient Method, CGDECENT scheme according to Harger and Zhang (2005)
Jun-Wei Huang
* 13.04.2010 Version 1.05 Implement L-BFGS Quasi-Newton Method
Jun-Wei Huang
* 23.04.2010 Version 1.06 Implement handling of different number of source and active receivers
Jun-Wei Huang
* 27.04.2010 Version 1.07 Implement primary reflections from predefined reflectors, see RTpfsm2d.h
Jun-Wei Huang
* 07.05.2010 Version 1.08 Implement Joint inversion of transmission and reflection travel time
Jun-Wei Huang
* 07.11.2010 Version 1.09 Implement Spatial Varying Weighting factor for joint inversion
Jun-Wei Huang
*28.01.2012 Version 1.1 Code cleaned and re-organized for publication
Jun-Wei Huang

History of modifications (3-D):

* 21.02.2010 Version 1.0 original implementation of fast sweeping
Jun-Wei Huang
* 08.03.2010 Implemented inversion
Jun-Wei Huang
* 22.03.2010 Implemented Parallel implementation
Jun-Wei Huang
* 07.04.2010 Version 1.01 Implement Nonlinear Conjugate Gradient Method, Hestenes-Stiefel scheme
Jun-Wei Huang
* 11.04.2010 Version 1.02 Implement Strong Wolfe Condition for line searching
Jun-Wei Huang
* 12.04.2010 Version 1.03 Implement Nonlinear Conjugate Gradient Method, CGDECENT scheme according to Harger and Zhang (2005)
Jun-Wei Huang
* 13.04.2010 Version 1.04 Implement L-BFGS Quasi-Newton Method
Jun-Wei Huang
* 23.04.2010 version 1.05 Implement practical source and receiver geometry handling,
Jun-Wei Huang

Chapter 1: Compiling and Installation

Assuming the home folder is "PFAST", to compile the 2-D/3-D version of this program, you should navigate to the folder "PFAST/src" and type

```
[junwei@Junbuntu src]$ ./RT_Compile2D.sh
[junwei@Junbuntu src]$ ./T_Compile3D.sh
```

The warnings if any, such as "warning: format '%s' expects type 'char*', but argument 3 has type 'char (*)[80]'" can be safely ignored.

you will find 7 executable files under "PFAST/bin" (as shown in Figure 1):

i. RT_PFAST2D_FW: forward modeling program saving both traveltime field for each shot point and traveltime at receiver locations.
ii. RT_PFAST2D_FW_NoS: forward modeling program only saving traveltime at receiver locations.
iii. RT_PFAST2D: Joint RT (Reflection-Transmission) tomography program.
iv. RT_PFAST2D_BENCH: Joint RT (Reflection-Transmission) tomography program for benchmarking purpose, i.e., only perform the first iteration and save CPU time.
v. T_PFAST3D: Transmission (First-arrival) tomography program using FSM for traveltime calculation.
vi. T_PFAST3D_BENCH: Transmission (First-arrival) tomography program for benchmarking purpose, i.e., only perform the first iteration and save CPU time.
vii. T_PFAST3D_FW: forward modeling program only saving traveltime at receiver locations.

In the following chapters, we will illustrate the usage for each program.

```
junwei@Junbuntu:~/data/PFAST4Pub/bin$ ll
total 1280
drwxr-xr-x 2 junwei junwei   4096 2012-01-29 21:42 ./
drwxr-xr-x 7 junwei junwei   4096 2012-01-29 14:48 ../
-rwxr-xr-x 1 junwei junwei 197889 2012-01-29 15:42 RT_PFAST2D*
-rwxr-xr-x 1 junwei junwei 197895 2012-01-29 15:42 RT_PFAST2D_BENCH*
-rwxr-xr-x 1 junwei junwei 177368 2012-01-29 15:42 RT_PFAST2D_FW*
-rwxr-xr-x 1 junwei junwei 177368 2012-01-29 15:42 RT_PFAST2D_FW_NoS*
-rwxr-xr-x 1 junwei junwei 181467 2012-01-29 21:42 T_PFAST3D*
-rwxr-xr-x 1 junwei junwei 181473 2012-01-29 21:42 T_PFAST3D_BENCH*
-rwxr-xr-x 1 junwei junwei 169033 2012-01-29 21:42 T_PFAST3D_FW*
junwei@Junbuntu:~/data/PFAST4Pub/bin$
```

Figure 1. A list of total executables after compilation saved under the folder of "bin".

Chapter 2: Organizing Input Parameter Files

To launch any of the programs, all the modeling information must be organized in a specific order in a file ready to be used by PFAST. The format of the input parameter file is provided here, followed by an example. The parameters must be entered into the input file according to the following order:

1. The file name of the initial velocity,
2. The file name of the source array,
3. Dimension of the source array,
4. The file name of the receiver array,
5. Dimension of the receiver array,
6. The file name of the reflector location array,
7. Dimension of the reflector location array,
8. Size of the model,
9. Output file name of the inversed model,
10. Regularization Parameter,
11. The file name of the weighting factor for Reflection, i.e., W_R, thus W_T=1-W_R, and 0<=W_T, W_R<=1,
12. Inversion Scheme,
13. Line search Scheme.

Table 1. An example of input parameter files for PFAST (2-D).

```
#The Input file for Travel time Tomography using Fast Sweeping
Method
#software developed by Junwei Huang, starting from Mar 08, 2010
#====================
#The file name of the initial velocity
../models/RT_SynMod2D_FW.vp
#The file name of the source locations
../models/RT_SynMod2D.src
56 3
#The file name of the receiver locations for Tran and Reflection
../models/RT_SynMod2D_FW.rec
33600 4
#The file name of the reflector locations
../models/RT_SynMod2D.ref
1200 3
#Size of the model, rows, columns, sample interval along
horizontal and vertical
160.000000 600.000000 12.500000 8.000000
#===Output the inversed model===
RT_SynMod2D.finalvp
#===Regularization Parameter: nux, nuz
60 20
#===Weighting factor of Reflection (WR)
../models/RT_SynMod2D_Inv.wr
#===Inversion Scheme tag (schemetag) and Line search tag (lsrc)
#recommend: 3 2, 3 3, 1 2, 1 3, 3 1, 2 2, 2 3, 2 1, 0 0
#schemetag:
#     1-HS Nonlinear Conjugate Method
```

```
#     2-CGDESCENT Nonlinear Conjugate Method
#     3-L-BFGS quasi-newton method
#     0-Steepest Decent method
#lsr:
#     1-Secant method with exact Conditions
#     2-Secant method Line Search with Strong Wolfe Conditions
#     3-Cubic Interpolation method with Strong Wolfe Conditions
#     0-No Line Search for Steepest Decent only
3 2
```

1. **The file name of the initial velocity**
   The file name includes the path as well as the file name for the initial velocity. In the example shown in Table 1, the file name is `T_SynMod2D_FW.vp` located at `../models`, which is the relative path to the directory of running the program.

2. **The file name of the source array**
   The file name includes the path as well as the file name for the source array, which is a binary file (see 3).

3. **Dimension of the source array**
   This line lists the row and the column number of the source array. In the example shown in Table 1, the source array `RT_SynMod2D.src` is a 2D matrix with 56 rows and 3 columns. Each row defines the *x* (horizontal) coordinate, *y* (vertical) coordinate and the total number of traces recorded for that source. In a general survey, the number of recorded traces varies among shot gathers. The binary file is saved in double precision as a row major matrix, i.e., the first increasing index is the row index. For example, the matrix a(56,3) is saved as a sequence like a(1,1), a(1,2), a(1,3), a(2,1), a(2,2), a(2,3)…, a(56,1), a(56,2), a(56,3).

4. **The file name of the receiver array**
   The file name includes the path as well as the file name for the receiver array, which is a binary file (see 5).

5. **Dimension of the receiver array**
   This line lists the row and the column number of the receiver array. In the example shown in Table 1, the receiver array `RT_SynMod2D_FW.rec` is a 2D matrix with 33600 rows and 4 columns. The number of rows is equal to the total trace number of the survey and each row defines 4 columns: *x* coordinate, *y* coordinate, normal vector *x* component, and normal vector *y* component. The normal vector is perpendicular to the surface where the receivers are located and is pointing away from the model region. The binary file is saved in double precision as a row major matrix.

   To do inversion, direct and reflection arrivals must be appended as column 5, 6, and so on. With more reflectors, the number of column increases accordingly. Traces with significantly large first arrival time (e.g., 9999 second) or with 0 second reflection traveltime will be ignored by the program. This feature can be used to select a subset of the traveltime. For example, in a field survey the number traces for each shot gather can vary so does the reflection arrivals. Trace xx may have direct arrival picked but not reflection arrival time. In this case, 0 second should be assigned to this trace as a reflection traveltime.

6. The file name of the reflector location array

The file name includes the path as well as the file name for the reflector location array, which is a binary file (see 7).

7. Dimension of the reflector location array

This line lists the row and the column number of the reflector location array. In the example shown in Table 1, the reflector location array `RT_SynMod2D.ref` is a 2D matrix with 1200 rows and 3 columns. Each row contains the *y* coordinate of the reflector location at each horizontal grid point, normal vector *x* component, normal vector *y* component. The normal vector is perpendicular to the reflector interface pointing toward the source direction. The binary file is saved in double precision as a row major matrix.

The number of rows must be the product of an integer number and the column number of the model. In this example, the column number of the model is 600 (see 8), thus the number of rows for the reflector location array should be 600 for one reflector, 1200 for two reflectors, and 600**n* for *n* reflectors. Here we defined two reflectors thus the number of rows is 1200. If a segment of the reflector takes *y* coordinates larger than the maximum depth of the model, a discontinuous reflector is defined.

8. Size of the model

This line lists the row and column number of the model as well as the sample interval along the horizontal (column) and the vertical (row) direction. The sample interval is in meter. The model file is in binary format saved in double precision as a row major matrix. Specifically, vp(1,1) is the top of the model and is at the top left corner.

9. Output file name of the inversed model

This line specifies the output file name for the tomographic velocity model.

10. Regularization Parameter

The choice of regularization parameters are usually made after a few trials. A low pass filter can be expressed in the wavenumber domain as

$$f\left(k_x,k_y,k_z\right)=\frac{1}{\left(\upsilon_x k_x\right)^2+\left(\upsilon_y k_y\right)^2+\left(\upsilon_z k_z\right)^2}, \tag{1}$$

where υ_x, υ_y, υ_z, are filter parameters controlling the smoothness (or roughness) of the model in three dimensions. When $\upsilon_x = \upsilon_y = \upsilon_z = \upsilon$, filter in equation (1) is identical to the Laplacian operator. Equation 1 is identical to eq.12 in the Computer & Geosciences paper.

Increasing regularization parameters can smooth the velocity model. The trade-off in choosing the regularization parameters is between recovering as many fine structures as the data allows and maintaining the stability and reality of the inversion. According to our experience, we usually start from the primary wavelength of the seismic survey. For example, if a survey uses frequency range of 20~80 Hz. Given the primary frequency of 60 Hz and the average velocity of 3 km/s, the primary wavelength is ~50 m. Considering that transmission wave has higher vertical resolution, a smaller vertical regularization parameter (empirically less than half of the horizontal regularization parameter) may be acceptable. Therefore in this example, we chose v_x=60 and v_z=20. For surveys with sparse wave coverage of the near surface, larger regularization parameters should be used.

11. The file name of the weighting factor for Reflection, i.e., $\mathbf{W_R}$

The absolute weighting factor for reflection is W_R, and W_T=1-W_R. Empirically the transmission is more reliable in the shallow subsurface and the reflection becomes more reliable in the vicinity of the reflectors. In that case, a space varying W_R can be defined as a function of space stored as an array with the same size as the model. Although designed for joint tomography, the program `RT_PFAST2D` can also perform first-arrival tomography. In that case, the array of W_R must be 1, the reflector location array can be defined arbitrarily but must still be provided. For example, it can be the lower boundary of the model. The binary file is saved in double precision as a row major matrix, and its dimension is identical to the model file.

12. Inversion & Line search Scheme

There are 4 options for inversion and line search, respectively. The details of each scheme can be found in our paper “PFAST: A High Performance Software Package for Transmission and Reflection Seismic Traveltime Tomography” of Computer & Geosciences and references therein. For most practical problems, I recommend L-BFGS quasi-newton method and secant method line search with Strong Wolfe Conditions. For forward modeling problems, this line must still be present but the values are ignored by the program.

The order of the input file parameter must not be altered for both forward modeling (e.g., `RT_PFAST2D_FW`, `RT_PFAST2D_FW_NoS`, and `T_PFAST3D_FW`) and tomography. In the 3-D case, the reflection tomography has not been implemented and thus 6, 7, and 11 ("The file name of the reflector location array ","Dimension of the reflector location array", and "Weighting factor of Reflection") are not needed. In addition, the size of the model is defined in three dimensions. An example input file for 3-D is illustrated in Table 2.

Table 2: An example of input parameter files for PFAST (3-D).

```
#The Input file for Travel time Tomography using Fast Sweeping Method
#software developed by Junwei Huang, started from Mar 08, 2010
```

```
#====================
#The file name of the initial velocity
../models/Field_ini_dz10.vp
#The file name of the source locations
../models/Field_dz10.src
690 4
#The file name of the receiver locations
../models/Field_dz10.rec
886326 7
#Size of the model, rows, columns, sample interval
150.000000 310.000000 200.000000 20.000000 20.000000 10.000000
#===Output the inversed model===
../models/Field_dz10.finalvp
#===Regularization Parameter: nux, nuy, nuz
200 200 10
#===Inversion Scheme tag (schemetag) and Line search tag (lsrc)
#recommend: 3 2, 3 3, 1 2, 1 3, 3 1, 2 2, 2 3, 2 1, 0 0
#schemetag:
#     1-HS Nonlinear Conjugate Method
#     2-CGDESCENT Nonlinear Conjugate Method
#     3-L-BFGS quasi-newton method
#     0-Steepest Decent method
#lsrc:
#     1-Secant method with exact Conditions
#     2-Secant method with Strong Wolfe Conditions
#     3-Cubic Interpolation method with Strong Wolfe Conditions
#     0-No Line Search, i.e. Steepest Decent
3 2
```

Chapter 3: Launching PFAST and Visualizing Results

3.1 Calculation of First and/or Reflection Arrivals

Executable programs of `RT_PFAST2D_FW` and `T_PFAST3D_FW` can be used to calculate transmission & reflection traveltime in 2-D and transmission traveltime only in 3-D, respectively. In this section, we will demonstrate how to use `RT_PFAST2D_FW` to obtain traveltime field for each shot as well as first and reflected arrivals at the receiver locations. Similarly, `T_PFAST3D_FW` is used to calculate first-arrivals in a 3-D velocity model.

Step 1: Use `RT_SynMod2D.m` under folder "`PFAST/mfile`" to view and save the pre-defined 2D model and source & receiver arrays into folders: "`PFAST/models`" and the input parameter file can be found at "`PFAST/par`".

```
junwei@Junbuntu:~/data/PFAST4Pub/models$ ll
total 2600
drwxr-xr-x 2 junwei junwei    4096 2012-01-30 23:11 ./
drwxr-xr-x 7 junwei junwei    4096 2012-01-29 14:48 ../
-rwxr--r-- 1 junwei junwei 1075200 2012-01-30 23:11 RT_SynMod2D_FW.rec*
-rwxr--r-- 1 junwei junwei  768000 2012-01-30 23:11 RT_SynMod2D_FW.vp*
-rwxr--r-- 1 junwei junwei  768000 2012-01-30 23:11 RT_SynMod2D_Inv.vp*
-rwxr--r-- 1 junwei junwei   28800 2012-01-30 23:11 RT_SynMod2D.ref*
-rwxr--r-- 1 junwei junwei    1344 2012-01-30 23:11 RT_SynMod2D.src*
junwei@Junbuntu:~/data/PFAST4Pub/models$
```

Figure 2: A list of model files including reflector location, initial velocity, velocity for traveltime calculation, and source& receiver arrays.

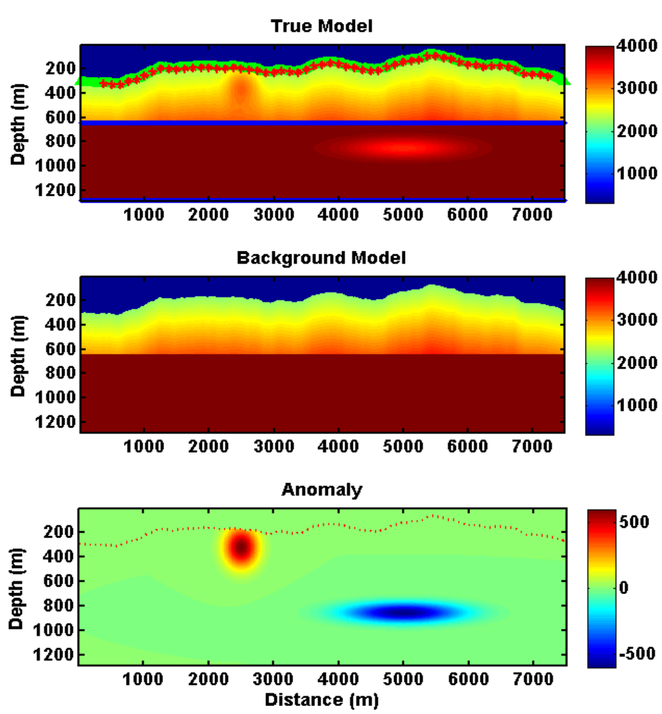


Figure 3: Matlab™ script `RT_SynMod2D.m` imports the model data and plot the true model, background model, and the anomalies. The red dots marks the irregular topography and the blue lines indicate two reflective boundaries that can be used for reflection tomography.

```
junwei@Junbuntu:~/data/PFAST4Pub/par$ mpirun -np 2 ../bin/RT_PFAST2D_FW RT_Syr
d2D_FW.inp
 ******************************
 * This is Travel Time Tomography program: Version 1.0
 * Parallel 2-D Traveltime Inversion based on Fast Sweeping Method and Adjoint
tate Technique
 *
 * written by  J.W. Huang


 * See COPYING file for copying and redistribution conditions.
 ******************************

Read input parameters from file RT_SynMod2D_FW.inp
User Parameters are as follows:

The initial velocity file... ../models/RT_SynMod2D_FW.vp
The source location file... ../models/RT_SynMod2D.src
The receiver location file... ../models/RT_SynMod2D_FW.rec
The source binary file dimension... 56 x 3
```

Figure 4. A segment of output content. The program feeds back some information on the computer screen if not directed to a file. The full content of the output can be found in the output file named "`RT_SynMod2D_FW.out`" in folder of "`par`".

The down-going (transmission) and up-going (reflection) traveltime field for each shot is saved in the folder of "`models`" (`RT_SynMod2D_FW_Src**_TTd.2d`), together with the first and reflected arrival time at the location of receivers (`RT_SynMod2D_FW_Tr2D.bin`). The Matlab™ script can reads the binary file `RT_SynMod2D_FW_Tr2D.bin` and display a few shot gathers. Figure 5 shows three shot gathers of first arrivals and two reflection arrivals. The binary file is in double precision consisting 33600 row and 3 columns saved by row major.

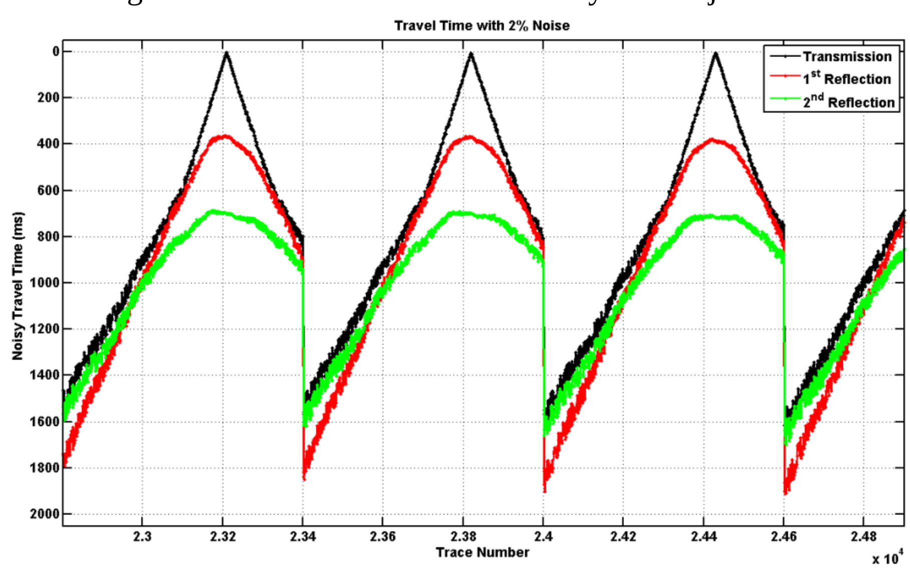


Figure 5: Three shot gathers of traveltime at the location of the surface receivers. A small amount of random noise with amplitude increasing with offset was added to the data, which will be used as observed traveltime for tomography in the next section.

The noise contaminated reflections and the first arrivals are later saved as `RT_SynMod2D_Inv_rand2p.rec` into folder "`models`". The 1st reflection is marginal in improving the tomographic velocity quality and thus the 1st reflection traveltimes are replaced by zero and ignored by the program. Users are encouraged to keep two reflections and compare the joint tomography using one reflection with that using two reflections. The input parameter file for tomography can then be generated by the same Matlab™ script. The content of the file is listed in Table 3.

Table 3: The input parameter files for 2D tomography

```
#The Input file for Travel time Tomography using Fast Sweeping Method
#software developed by Junwei Huang, starting from Mar 08, 2010
#====================
#The file name of the initial velocity
../models/RT_SynMod2D_Inv.vp
#The file name of the source locations
../models/RT_SynMod2D.src
56 3
#The file name of the receiver locations for Tran and Reflection
../models/RT_SynMod2D_Inv_rand2p.rec
33600 7
#The file name of the reflector locations
../models/RT_SynMod2D.ref
1200 3
#Size of the model, rows, columns, sample interval
160.000000 600.000000 12.500000 8.000000
#===Output the inversed model===
../models/RT_SynMod2D.finalvp
#===Regularization Parameter: nux, nuz
60 20
#===Weighting factor of Reflection (WR)
../models/RT_SynMod2D_Inv.wr
#===Inversion Scheme tag (schemetag) and Line search tag (lsrc)
#recommend: 3 2, 3 3, 1 2, 1 3, 3 1, 2 2, 2 3, 2 1, 0 0
#schemetag:
#     1-HS Nonlinear Conjugate Method
#     2-CGDESCENT Nonlinear Conjugate Method
#     3-L-BFGS quasi-newton method
#     0-Steepest Decent method
#lsr:
#     1-Secant method with exact Conditions
#     2-Secant method Line Search with Strong Wolfe Conditions
#     3-Cubic Interpolation method with Strong Wolfe Conditions
#     0-No Line Search for Steepest Decent only
3 2
```

The traveltime field is saved as a three-dimensional matrix in double precision. The first two dimensions are identical to the size of the model, i.e., number of row x number of column. The third dimension equals to the number of reflectors +1. The down-going (transmission) and up-going (reflection) traveltime field for shot 4 is loaded into

Matlab™ and shown in Figure 6. The traveltime field was plotted as contours superimposed on the true velocity model. For surveys with a great amount of sources, it is not so desirable to store the traveltime field for all shots. In this case, program `RT_PFAST2D_FW_NoS` can be used and only the traveltime at the receiver locations will be outputted.

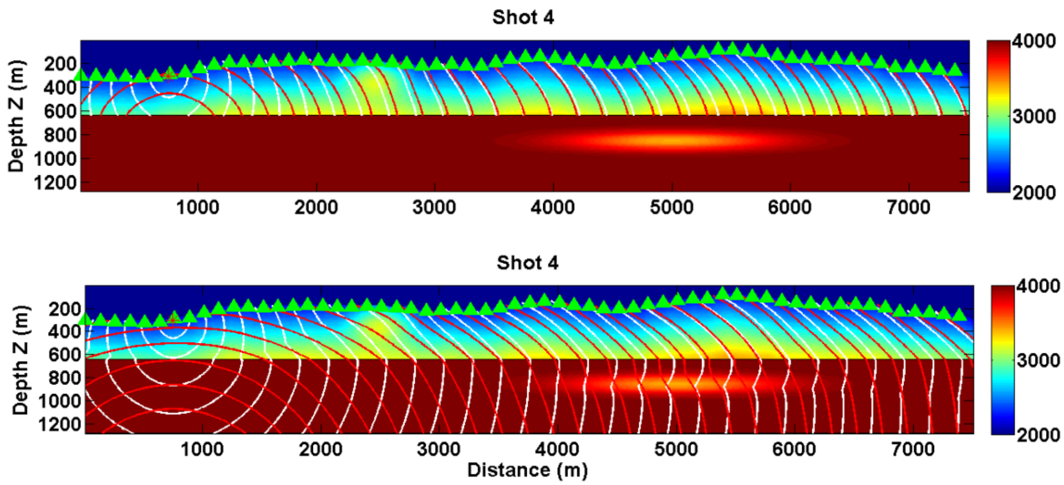


Figure 6: Transmission (white lines) and reflection (red lines) traveltime field generated by shot 4. The color bar shows the value of the propagation velocity.

3.2 2-D Joint Tomography

Joint tomography can be performed using program `RT_PFAST2D`. A typical command in this example is:

```
mpirun -np 2 ../bin/RT_PFAST2D RT_SynMod2D_Inv.inp 50 1
mpirun -np 2 ../bin/RT_PFAST2D RT_SynMod2D_Inv.inp 50 2
mpirun -np 2 ../bin/RT_PFAST2D RT_SynMod2D_Inv.inp 50 3
```

where mpirun invokes the MPI, the number of processors is defined after -np, and 50 is the memory size needed by the L-BFGS quasi-Newton method. The last tag indicates the transmission tomography only (1), reflection tomography only (2) and the joint tomography (3).

Each iteration outputs the updated velocity model named as `RT_SynMod2D_Inv_*` into the folder of "`models`" together with the velocity perturbation for reflection, transmission and joint tomography before and after regularization for quality control purpose. Figure 7 shows the content of the folder after a completed run of transmission tomography. The binary files (same size as the model file) shows the gradient (velocity perturbation) for each iteration (Figure 8) and the ASCII files (*.txt) stores the average residual or the global error for the object function of joint, transmission, and reflection tomography after each iteration.

```
RT_SynMod2D.finalvp
RT_SynMod2D_Inv_1
RT_SynMod2D_Inv_2
RT_SynMod2D_Inv_3
RT_SynMod2D_Inv_4
RT_SynMod2D_Inv_5
RT_SynMod2D_Inv_6
RT_SynMod2D_Inv_7
RT_SynMod2D_Inv_af_regu_R1
RT_SynMod2D_Inv_af_regu_R2
RT_SynMod2D_Inv_af_regu_R3
RT_SynMod2D_Inv_af_regu_R4
RT_SynMod2D_Inv_af_regu_R5
RT_SynMod2D_Inv_af_regu_R6
RT_SynMod2D_Inv_af_regu_R7
RT_SynMod2D_Inv_af_regu_RT1
RT_SynMod2D_Inv_af_regu_RT2
RT_SynMod2D_Inv_af_regu_RT3
RT_SynMod2D_Inv_af_regu_RT4
RT_SynMod2D_Inv_af_regu_RT5
RT_SynMod2D_Inv_af_regu_RT6
RT_SynMod2D_Inv_af_regu_RT7
RT_SynMod2D_Inv_af_regu_T1
RT_SynMod2D_Inv_af_regu_T2
RT_SynMod2D_Inv_af_regu_T3
RT_SynMod2D_Inv_af_regu_T4
RT_SynMod2D_Inv_af_regu_T5
RT_SynMod2D_Inv_af_regu_T6
RT_SynMod2D_Inv_af_regu_T7
RT_SynMod2D_Inv_AveRes_TR_T_R1.txt
RT_SynMod2D_Inv_bf_regu_R1
RT_SynMod2D_Inv_bf_regu_R2
RT_SynMod2D_Inv_bf_regu_R3
RT_SynMod2D_Inv_bf_regu_R4
RT_SynMod2D_Inv_bf_regu_R5
RT_SynMod2D_Inv_bf_regu_R6
RT_SynMod2D_Inv_bf_regu_R7
RT_SynMod2D_Inv_bf_regu_RT1
RT_SynMod2D_Inv_bf_regu_RT2
RT_SynMod2D_Inv_bf_regu_RT3
RT_SynMod2D_Inv_bf_regu_RT4
RT_SynMod2D_Inv_bf_regu_RT5
RT_SynMod2D_Inv_bf_regu_RT6
RT_SynMod2D_Inv_bf_regu_RT7
RT_SynMod2D_Inv_bf_regu_T1
RT_SynMod2D_Inv_bf_regu_T2
RT_SynMod2D_Inv_bf_regu_T3
RT_SynMod2D_Inv_bf_regu_T4
RT_SynMod2D_Inv_bf_regu_T5
RT_SynMod2D_Inv_bf_regu_T6
RT_SynMod2D_Inv_bf_regu_T7
RT_SynMod2D_Inv_GlobalErr_TR_T_R1.txt
```

Figure 7: The output files of the transmission tomography include the updated velocity and the velocity perturbation for reflection, transmission and joint tomography before and after regularization.

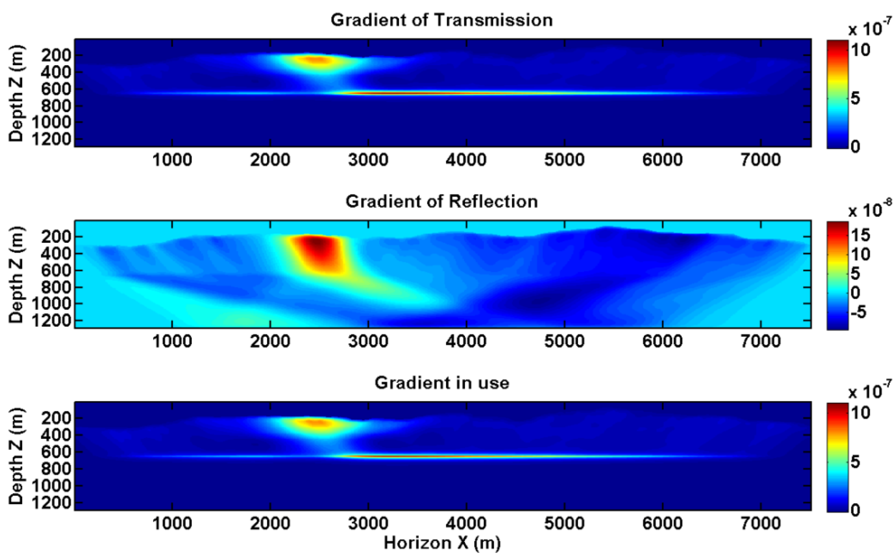


Figure 8a: The gradient (velocity perturbation) after regularization. The velocity perturbations for both transmission and reflection traveltimes are calculated and the velocity perturbation for transmission arrivals is used to update the velocity model.

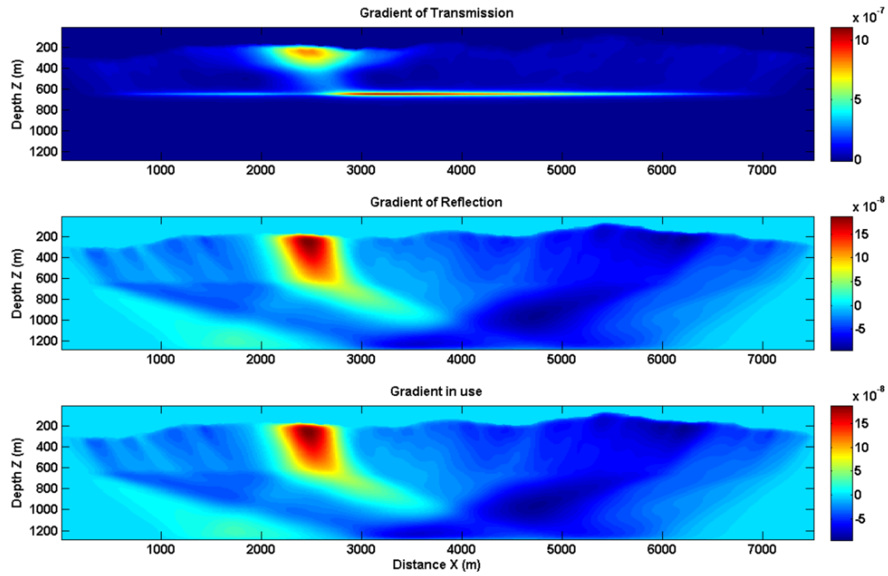


Figure 8b: The gradient (velocity perturbation) after regularization. The velocity perturbations for both transmission and reflection traveltimes are calculated and the velocity perturbation for reflection arrivals is used to update the velocity model.

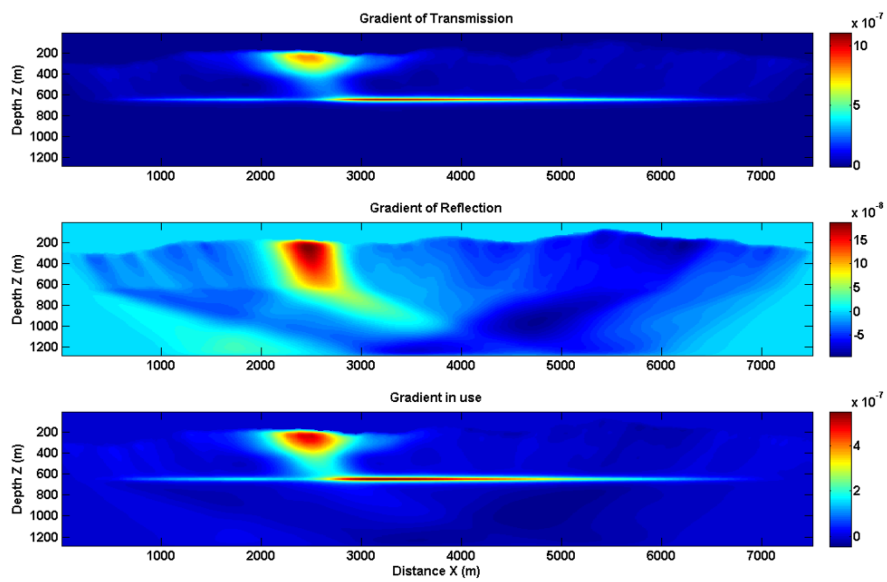


Figure 8c: The gradient (velocity perturbation) after regularization. The velocity perturbations for both transmission and reflection traveltimes are calculated and the velocity perturbation for both arrivals is used to update the velocity model.

The final recovered velocity is visualized using the Matlab™ script. The transmission, reflection, and joint tomographic models are shown in Figure 9. We observe that the joint tomography provides the best velocity models. Notice that the traveltimes are contaminated by random noise. Users are encouraged to increase the level of noise to further test the robustness of joint tomography.

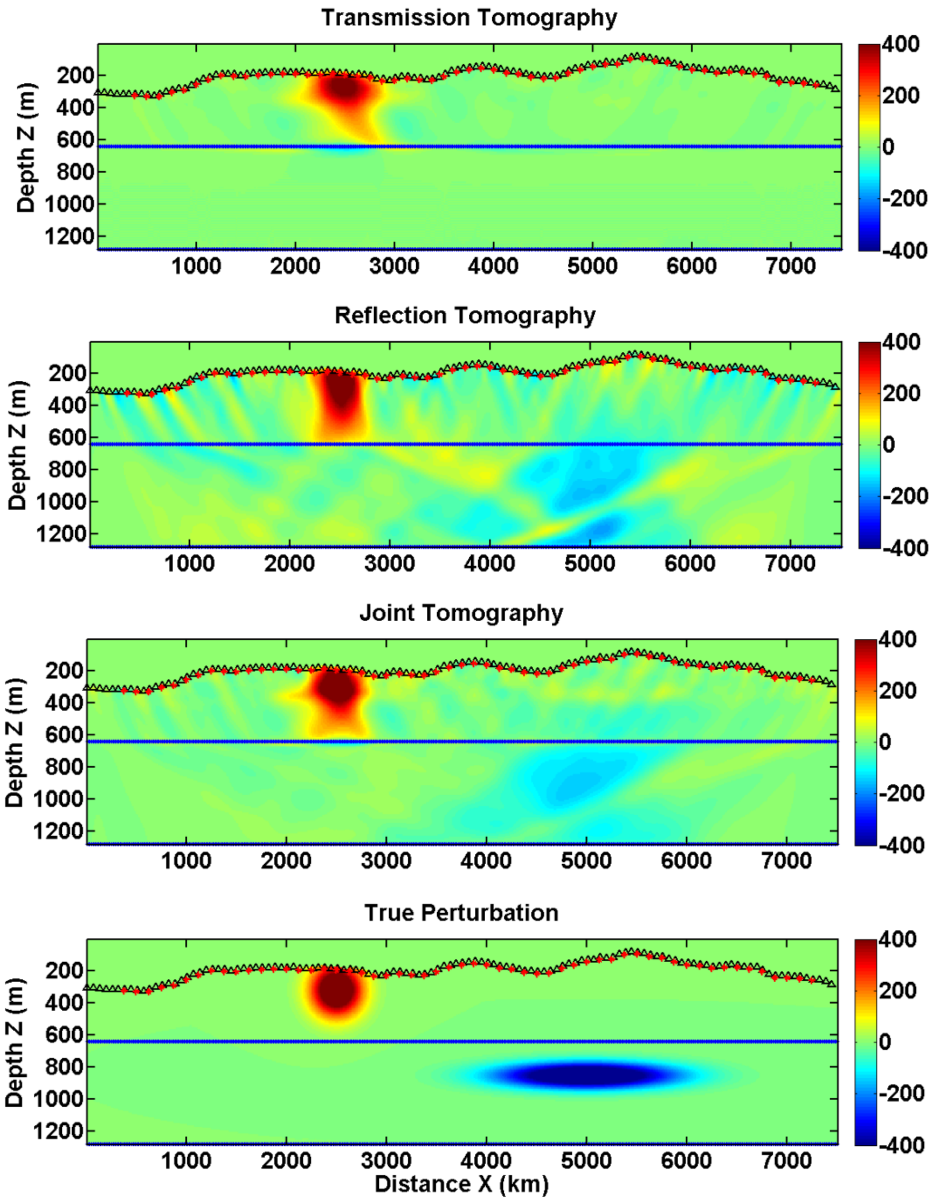


Figure 9: The recovered anomalies by conventional first-arrival tomography, reflection tomography and joint tomography.

The benchmarking program (`RT_PFAST2D_BENCH`) can be used in the same way as `RT_PFAST2D`. However, no files will be saved and only the first iteration is performed. The total CPU time will be printed on the screen or directed to a file.

3.3 3-D first-arrival traveltime and tomography

In this section, we demonstrate how to use `T_PFAST3D_FW`, `T_PFAST3D` to calculate the first arrivals and perform first-arrival traveltime tomography in 3D. Program `T_PFAST3D_BENCH` is again perform only one iteration and only print the CPU time on the screen. The purpose of running the benchmark program is to evaluate the scalability of the program on your distributed memory system. Notice the size of the model is significantly large than the 2D model in the previous section (see the parameter file in Table 2). Thus a high demand of memory of CPU time is expected, even on high performance computers.

In this section, we run `T_PFAST3D_FW`, `T_PFAST3D` on a 4 nodes (16 cores) cluster. The source, receiver, and initial model can be visualized in Matlab™ using T_Field3D.m (see Figure 10). As an example, we can calculate first arrivals at the location of receivers using the initial model. However, we are not going to use the calculated first arrivals for transmission tomography, since the first arrivals from the field data have already stored in the `Field3D.rec` file. On our 16 core cluster, the forward modeling took 23.6 hours and the standard output of the forward modeling program can be viewed in the `Field3D_FW.out` file. A segment of the calculated the first arrivals is shown in Figure 11. Notice that the first arrival times were based on the initial model (Figure 10). The CPU time for 3D tomography is proportional to the number of interactions. Usually, each iteration consists 3 times forward modeling.

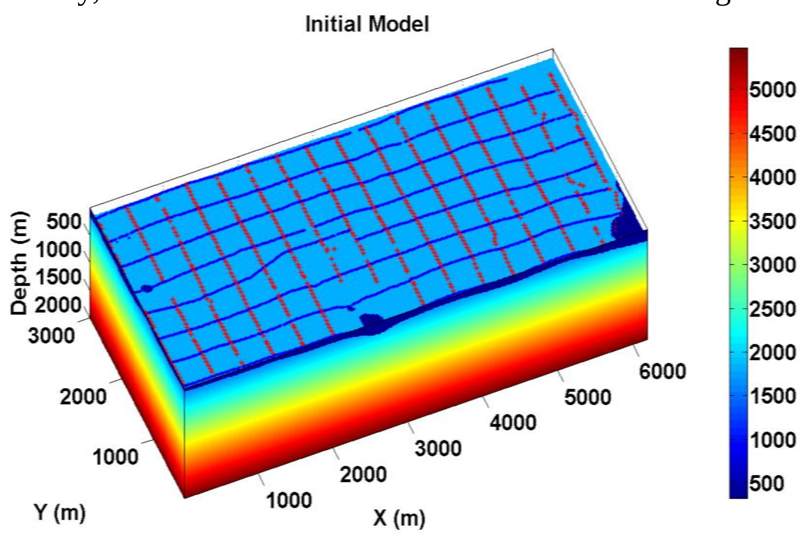


Figure 10: The initial model for transmission tomography using `T_PFAST3D`. The first arrivals from the field data have been stored in the `Field3D.rec` file. The sources (red stars) and the receivers (blue dots) were distributed on the surface following the topography and avoid major water bodies. The color bar presents the value of propagation velocity.

Both the forward modeling and the transmission tomography can be launched by typing

```
mpirun -np 16 ../bin/T_PFAST3D_FW Field3D.inp
mpirun -np 16 ../bin/T_PFAST3D Field3D.inp
```

where `T_PFAST3D_FW` launches the forward modeling and `T_PFAST3D` starts the transmission tomography. The default number of memory used in L-BFGS quasi-Newton method is 5. It can be reduced according to the profile of your machine.

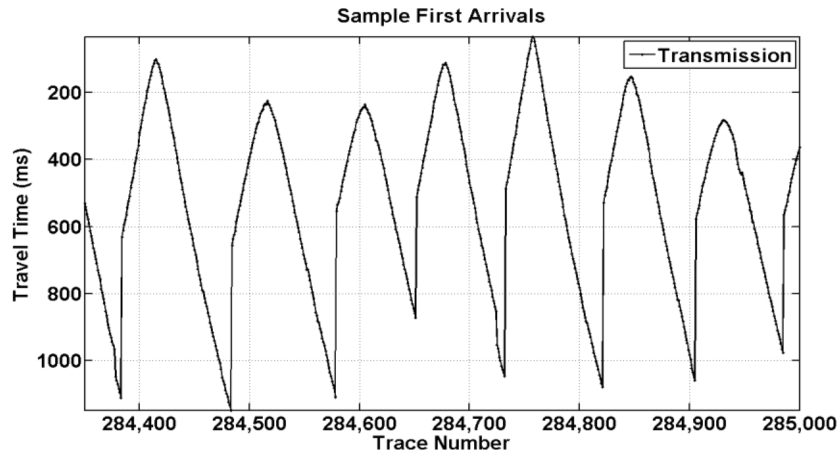


Figure 11: The calculated first arrivals based on the model in Fig. 10.

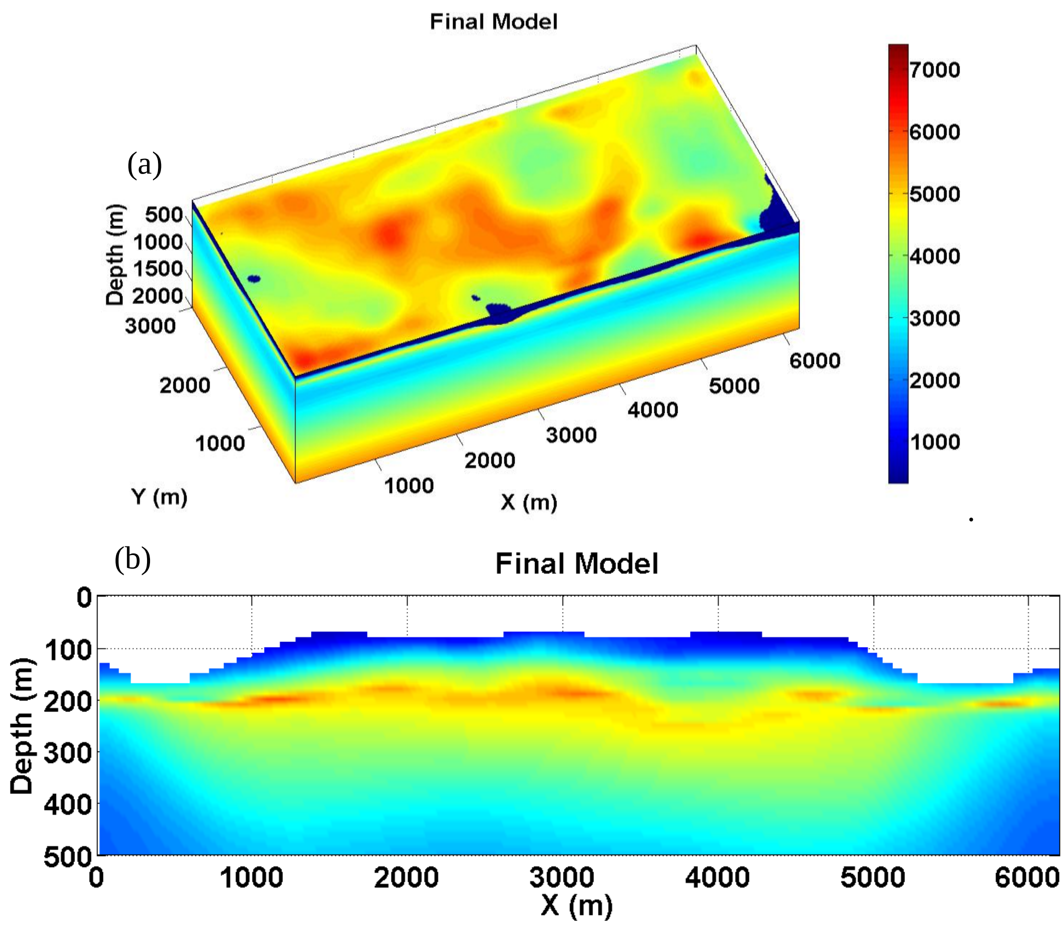


Figure 12: The tomographic model after 16 iterations. (a) The velocity structure at the near surface is consistent with the topography. (b) a slice at Y=1500 m shows the recovered refractor at depth around 200 m.

After 14 iterations, the tomographic result presents geologically consistent velocity model (shown in Figure 12). Certainly the model thickness (2000 m) is larger than what the diving waves can recover. A strong refractor is recovered by large offset data and the depth (or the elevation) of the refractor can be picked and displayed as a function of location (see Figure. 13). Users are encouraged to visualize the final model (modvp) on Matlab™ to locate the deepest recovered and reliable features. Matlab™ script `T_Field3D.m` can be used for this purpose.

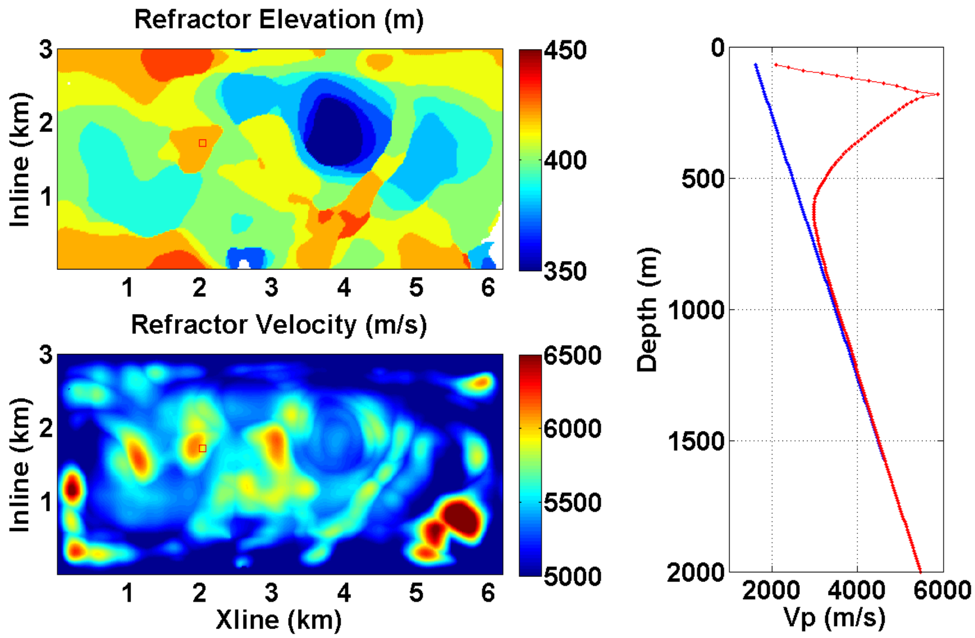


Figure 13: The location of the refractor is defined as the turning point of the velocity profile. As example on the right side, and recovered velocity (red line) reaches a local maximum around 240 m and the peak velocity is about 5800 m/s. The picked refractor and the peak velocity are thus displayed as a function of location shown in the upper left corner and lower left corner, respectively.

Similar to the 2D version, the program outputs not only the update model but also the velocity perturbation before and after regularization for quality control purpose. In addition to those binary files, PFAST also creates ASCII file recording the global error and the average residual after each iteration. The values can be plotted as a function of iteration number, and the rate of convergence can be estimated.

Appendix A: Article Abstract

In this paper, we present an open source software, PFAST (Parallel Fast sweeping method based Adjoint Seismic Tomography), which can perform transmission, reflection, and joint traveltime tomography. Unlike ray based tomography, this program utilizes a grid-based Eikonal equation solver to obtain seismic traveltime and circumvent the non-linearity of conventional ray shooting and bending approaches. The adjoint method is used to obtain the gradient of the non-linear objective function without explicit evaluation of Fréchet derivative matrix, which is usually computationally prohibitive for large scale problems. When combined with Huygens' Principle, the grid-based Eikonal equation solver calculates both transmission and reflection traveltime and the joint tomography can further mitigate the ambiguity of inverse modeling, increase the reliability of the tomographic model, and reveal deeper features not visible to the conventional first-arrival seismic tomography.

This paper describes the theoretical basis of the program, provides the details of the parallel implementation and demonstrates the scalability of the high performance program on a distributed memory system. We then evaluate the performance of transmission, reflection and joint tomography on a synthetic model and a two-dimensional (2-D) seismic survey conducted in Northwest Territories of Canada where complex thermokarst lakes and heterogeneous permafrost dominate the shallow subsurface.

Appendix B: To-do List

Currently PFAST v1.1 only supports isotropic model descritized in regular grid for joint tomography in 2D and first arrival tomography in 3D. To cover more general situations, PFAST shall be extended to, in the order of priority,
(1) handle anisotropic velocity,
(2) handle irregular meshes, and
(3) include reflection tomography in 3D.

Considering that our tomography algorithm makes use of the adjoint technique, the future plans can be implemented as long as the forward engine (Fast Sweeping Method) is coded to handle anisotropy, irregular mesh and reflection traveltime. Due to time limit, we are unable to achieve the above goals without your help. If you are interested in joining the develop team, please send an email to jw.huang@utoronto.ca. We appreciate your contribution to this piece of open source software and help others to scratch their itches.

References

Harger, W.W. and Zhang, H.C., 2005, A new Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search, SIAM, J. OPTIM., Vo. 16, No. 1, pp. 170-192"

Huang, J.W., and Bellefleur, G., 2011, Joint Transmission and Reflection Traveltime Tomography in Imaging Permafrost and Thermokarst lakes in Northwest Territories, Canada, The Leading Edge.

Huang, J.W., and Bellefleur, G., 2012, Joint Transmission and Reflection Traveltime Tomography using the Fast Sweeping Method and the Adjoint-state Technique, Geophysical Journal International.