GUIDE TO USING TIME-DISTANCE INVERSION SOFTWARE: FORWARD CALCULATION

JAVORNIK

The forward calculation uses sensitivity kernels and the velocity of convecting flows to calculate travel-time maps. The velocities are sometimes called models. This document contains example parameter files for the forward calculation along with an explanation of the input and output data files. The document ForwardNotes.pdf contains equations and additional details about the calculations.

The Time-Distance Inversion code is called *invert* at the command line. All the inversion code is written in C. The software uses several shared libraries. The shared libraries parse the input file, create output directories, read and write binary data files, and perform matrix and linear equation operations, and Fourier transforms. The libraries provide wrappers for the external software packages: C FITS File Subroutine Library (CFITSIO), Linear Algebra PACKage (CLAPACK), (CBLAS) Fastest Fourier Transform in the West (FFTW), Lex, and YACC.

Each software releases contains the shared libraries, documentation, and sample data as tar files. All software installations require core.tar and then any combination of the specific inversion tar files. Doc.tar contains code documentation. The test.tar files contain sample data along with the expected results. Forward calculations require core.tar and fwd.tar. The sample data files are in the fwd_test.tar file. The README file contains instructions to build and run the inversion code.

1. General Directions

These general directions apply to running the forward calculation.

1.1. How to use.

```
invert parameterFile -p -c -d
```

The *invert* command takes an input parameter file (explained in section **??**) and any command line options (Table 1), then finds an empty version of the output directory (creating one if needed), and writes the resulting output files. Table 2 contains the possible problem dimensions. Table 3 contains the input parameters.

1.2. **Input and Output Data Files.** Input and output data files follow the Flexible Image Transport System (FITS) format. If an input FITS file ends with (.gz), it is read as a compressed file and is expanded in memory.

Date: August 19, 2013.

TABLE 1. Command-Line Flags

Flag	Action
$-p$	indicates parsing the input file only (no inversion)
$-c$	copies all input files to the output directory
$-d$	dumps inversion intermediate values for debugging
parameterFile	contains the name and location of all input data files and all inversion parameters.

TABLE 2. Possible Problems

Dimension	Description
Forward	Solve forward calculation. Gaussian noise is added over a range of scales.
2d	Drop z dependence, use only 2-D kernels(x,y) and only one scatterer.
3d-scalar	Include z dependence, use 3-D kernels(x,y,z) and only one scatterer.
3d-vector	Many scatterers, $\alpha > 1$, which in this case is $\alpha = 3$, v_x, v_y, v_z. Three kernel files correspond to each output map file.

TABLE 3. Parameter File Contents

Parameter	Forward
dimension	2D, 3D-scalar 3D-vector
output base directory	yes
base dir for kernels	yes
base dir for amplitude	yes
kernel-amplitude pairs	K, amplitude
model (flow velocities)	$\boldsymbol{v}$
looping parameters	noise scale
overlap (3-D only)	Θ
random number seed	yes

The parameter file contains directory paths, file names and parameter values. The output base directory contains all the output files generated by the inversion program along with a copy of the parameter file. The output base directory also contains copies of all the files specified in the parameter file, if the '-c' flag is used. If the output directory does not exist, the invert program creates it. If the output base directory exists and is empty, then the invert program dumps the output files there. If the output directory is not empty, then, the invert program adds a version (_v2) to the output directory name. The

invert program continues to append version numbers to the output directory name until an empty directory is found or the directory does not exist.

Output files include the travel-time maps, (d∼). The header of each output FITS file contains the parameter values (noise scale) used to generate the data. The Examples/testdata directory contains example parameter files. The Examples/testdata/output directory contains sample output files for each parameter file.

2. Example of a Forward Calculation

2.1. **Input Parameter File.** The input parameter file for the forward calculation is very similar in style to the parameters files for inversions.

The dimension of the forward problem (3D scalar or 3D vector) determines the number of kernels expected in each line of the input parameter file. If the problem is 3D scalar, there is one kernel on each line. If the problem is 3D vector, there are three kernel files in each line. Each line of the paramter file that specifies a kernel corresponds to an output travel-time map. For example, if there are five lines in the parameter file that contain kernels, then there will be five travel-time maps in the output directory with names like 1_d˜1.fits, 1_d˜2.fits, 1_d˜3.fits, 1_d˜4.fits, 1_d˜5.fits. Noise amplitude files are the same dimensions as the output travel-time maps. Noise amplitude files indicate the amplitude and location for added noise. There is one amplitude file for each kernel. The overlap file (Θ) has the same meaning as in inversions. The model (true flow velocities) is an input data file. There is no input covariance file in the forward calculation. A range scales the noise amplitude; a zero scale means no noise. The noise scale uses linear or log steps. Travel-time maps with the same noise scale have the same leading file name. For example, all the travel-time maps with 1_d˜*.fits have the same noise scale, and the noise scale is different than all maps with file names 2_d˜*.fits. The random number generator seed is an optional input parameter.

Here is a sample parameter file:

```
 3d_vector forward
testdata/work/fwd/3dv/ReducedMF_8_10
! base directory for Kernel files
testdata/input/3dv/ReducedMF_8_10
! base directory for noise Amplitude files
testdata/input/3dv/MoatFlow
kern_x_x_1.fits kern_x_y_1.fits kern_x_z_1.fits noiseAmp1.fits
kern_y_x_2.fits kern_y_y_2.fits kern_y_z_2.fits noiseAmp1.fits
kern_oi_x_3.fits kern_oi_y_3.fits kern_oi_z_3.fits noiseAmp1.fits
! model data
testdata/input/3dv/ReducedMF_8_10/q.fits
! overlap matrix for integrating dz
testdata/input/3dv/MoatFlow/R9.fits
! optional seed for random number generator
seed 12345678
```

```
! scale factors for noise amplitude
0 1e-3 log
```

The noise at location $(\boldsymbol{r}) = (x, y)$ in map a is added using $n_a(\boldsymbol{r}) = A_a(\boldsymbol{r})random(i)s_j$ where A is the amplitude specified in the noise amplitude file, the ith random number and s_j is the jth step in the range of noise scale factors.

2.2. **Output Files.** The output files are:

```
1_d~1.fits  2_d~2.fits  3_d~3.fits  5_d~1.fits  seed.txt
1_d~2.fits  2_d~3.fits  4_d~1.fits  5_d~2.fits
1_d~3.fits  3_d~1.fits  4_d~2.fits  5_d~3.fits
2_d~1.fits  3_d~2.fits  4_d~3.fits  ReducedMF_8_10
```

There are five sets of maps distinguished by the first digit in the file names of the form

```
<scale factor>_d~<map number>.fits.
```

The code sets a non-zero, lower bound of three log steps smaller than the upper bound when the noise scale minimum value is zero, the upper bound is less than one, and the steps are logarithmic. In this case, the noise scale has values 0, 1×10^{-6}, 1×10^{-5}, 1×10^{-4}, and 1×10^{-3}. There are three maps in each set. The file 'seed.txt' contains the seed used by the random number generator. The file 'ReducedMF_8_10' is a copy of the input parameter file.

To calculate the noise covariance, first set the minimum value of the noise scale to zero. Then, subtract all the maps with a scale factor greater than zero from the map with no noise.

3. Possible Errors

The files errorHandler.h and the files *ErrorMessages.h contain most of the errors reported by the software. Third party software packages report a few additional errors. These include error messages from CFITSIO, LAPACK, and FFTW which the inversion software catches and replaces with error messages contained in errorHandler.h. Some error messages from CFITSIO library propagate without wrapping or replacement.

Generally, the errors come from unexpected input, runtime complications or a problem writing to files.

- Unexpected input:
  - matrix and vectors of incompatible sizes
  - matrix and vectors are too large
  - input files don't exist
- Run-time complications:
  - Linear equation solve fails to converge. The resulting output is NaN and an error message is reported.
  - Out of memory error; checks that all mallocs are successful, otherwise, report error and abort.
- Problems with output:
  - cannot write output files
  - directory does not exist