GUIDE TO USING TIME-DISTANCE INVERSION SOFTWARE: OLA

JAVORNIK

The Time-Distance Inversion Software calculates inversions for local helioseismology using multi-channel deconvolution (MCD) subtractive optimally localized averaging (OLA) approach in the Fourier domain. This document contains example parameter files along with an explanation of the input and output data files. The documents OLANotes.pdf and CrossTalk.pdf contain the equations and additional details about the calculations.

The inversion code consists of a Python script, called *invert*, and several shared libraries for parsing input files and calculating inversions. The *invert* script coordinates parsing of an input file, constructs directories for the results, and passes the information to an inversion function written in C. The shared libraries contain core operations such as matrix and linear equation operations, Fourier transforms, reading and writing binary data files, and parsing input parameters. The libraries provide wrappers for the external software packages: C FITS File Subroutine Library (CFITSIO), Linear Algebra PACKage (LAPACK), Fastest Fourier Transform in the West (FFTW), Lex, and YACC.

A software release bundles the libraries, scripts, documentation, and sample data into tar files. All installations require core.tar and then any combination of the specific inversion files. Doc.tar contains code documentation and the test.tar files contain sample data along with the expected results. For OLA inversions download and expand, core.tar and ola.tar. To run the sample data, download the ola_test.tar file. After expanding the tar files, follow the instructions in the README file to build and run the inversion code.

1. General Directions

These general directions apply to running all inversion problems including the forward calculation. The following sections contain specific information for each type of calculation.

1.1. **How to use.**

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

The python script, *invert*, verifies files and directories exist, copies input files, finds an empty version of the output directory, and calls the inversion code. Table 1 explains the command-line flags.

Table 2 contains the possible problem dimensions along with a brief description of the inversion method. Table 3 contains the specific parameters for each inversion.

Date: September 17, 2012.

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
OLA	Solve MCD SOLA inversion for a number of μ and σ_h.
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. Solve for only one α at a time. Three kernel files correspond to a single map file.

TABLE 3. Parameter File Contents

Parameter	OLA
dimension	2D, 3D-scalar, 3D-vector
output base directory	yes
base dir for kernels	yes
base dir for maps	yes
kernel-map pairs	K and τ
noise covariance	Λ
regularization parameters	μ, σ_h, (ν)
overlap (3-D only)	Θ
$f(z)$ for Target function (3-D only)	f
*Averaging kernel specification is not required for OLA inversions, because all averaging kernels are calculated.	

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.

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. Averaging kernels and noise covariances are calculated for every height (z-value) listed. The inversion code computes the error covariance matrix by the formal propagation of errors. For OLA inversions, the averaging kernel and noise covariance are calculated at every height.

Output files include the solution, averaging kernels, noise/error covariance. The header of each output FITS file contains the parameter values (μ, σ_h, ν) used to generate the data. The naming convention for the output files follows the format $\mu\_\sigma\_matrix.fits$ for OLA output files. The crosstalk parameter (ν) is optional. If the crosstalk parameter is used in an input parameter file, then the output file names follow the format $\mu\_\sigma\_\nu\_matrix.fits$ Matrix names for OLA are q, w, T, AvgKern, and Noise. OLA inversions contain averaging kernels for all heights in one file. All averaging kernel and noise/error covariance are compressed FITS files, with the extension (.fits.gz).

The Examples/testdata directory contains example parameter files for each inversion and dimension. The Examples/testdata/output directory contains sample output files for each parameter file.

2. Example of an OLA Inversion

OLA inversions have a target function as an input file. For RLS inversions, there is no choice of target function. OLA inversion input files and output files are similar to those of RLS inversions, with the exception of an overlap matrix and a target function.

A good starting point for the target function is to make it wide (σ_h somewhere around 30 or 40 pixels). If the target function is too narrow or steep, the noise or error will be too big.) **NOTE: The center point of target function must be the same as the center point of Kernel and Map. Currently, the center point, or zero pixel, has coordinates (81,81) in array index 1 to 160.**

2.1. **Input Parameter File.**

```
3d_scalar ola
testdata/work/ola/3d/Simple
! base directory for Kernel files
testdata/input/3d/Simple
! base directory for Map files (input data files)
testdata/input/3d/Simple
!input Kernel file (Kern1.fits) Map1.fits (input data file)
K1Z1.fits map.fits
! noise covariance matrix; must be NxNxMxM, N = 2, M = 1 = # of kernel-map pairs
testdata/input/3d/Simple/C1.fits
```

```
!input Regularization matrix file (R.fits)  Theta matrix in OLA
testdata/input/3d/Simple/R.fits
! MuMin MuMax NMu SigmaMin SigmaMax NSigma
0.0 1.0  3              100 200 1
! for target function, Fi
testdata/input/3d/Simple/f.fits
```

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

```
0_1_T.fits
1_1_Noise.fits  1_1_w.fits  1_1_AvgKern.fits  1_1_q.fits

2_1_Noise.fits  2_1_w.fits  2_1_AvgKern.fits  2_1_q.fits

3_1_Noise.fits  3_1_w.fits  3_1_AvgKern.fits  3_1_q.fits

Simple
```

The naming of the output files follows the convention μ_σ_h_matrix.fits. The file 0_1_T contains the full target function from expanding the f.fits function for the z-dimension with a 2D-Gaussian in the xy-dimension. (OLANotes.pdf contains more details about the target function.) For each permutation of the values of μ and σ_h, there is an averaging kernel, noise, weights (w.fits) and estimate of the model (q.fits) from the inversion.

3. Example of an OLA Inversion with crosstalk parameter

The parameter ν controls the amount of crosstalk.

3.1. **Input Parameter File.**

```
3d_vector_z ola
testdata/work/ola/3dv/testReducedMF_8_10_xt_noise_z
! base directory for Kernel files
testdata/input/3dv/ReducedMF_8_10
! base directory for map files
testdata/work/fwd/3dv/ReducedMF_8_10
kern_x_x_1.fits kern_x_y_1.fits kern_x_z_1.fits 4_d~1.fits
kern_y_x_2.fits kern_y_y_2.fits kern_y_z_2.fits 4_d~2.fits
kern_oi_x_3.fits kern_oi_y_3.fits kern_oi_z_3.fits 4_d~3.fits
! noise covariance matrix; must be NxNxMxM, N = 10, M = 3 = # of kernel-map pairs
testdata/input/3dv/MoatFlow/C3N10_IFFT_0p000001.fits
! Theta file for integrating dz
testdata/input/3dv/MoatFlow/R9.fits
! Mu(min,max,N) Sigma(min,max,N) Nu(min,max,N)
1e-1 1 2     1 10 1     0 10 log
! for target function, Fi
```

```
testdata/input/3dv/MoatFlow/f1.fits
```

3.2. **Output Files.**

```
0_1_T_1.fits
0_1_T_2.fits
0_1_T_3.fits

1_1_1_AvgKern_1.fits    2_1_1_AvgKern_1.fits
1_1_1_AvgKern_2.fits    2_1_1_AvgKern_2.fits
1_1_1_AvgKern_3.fits    2_1_1_AvgKern_3.fits
1_1_1_Noise.fits        2_1_1_Noise.fits
1_1_1_q.fits            2_1_1_q.fits
1_1_1_w.fits            2_1_1_w.fits

1_1_2_AvgKern_1.fits    2_1_2_AvgKern_1.fits
1_1_2_AvgKern_2.fits    2_1_2_AvgKern_2.fits
1_1_2_AvgKern_3.fits    2_1_2_AvgKern_3.fits
1_1_2_Noise.fits        2_1_2_Noise.fits
1_1_2_q.fits            2_1_2_q.fits
1_1_2_w.fits            2_1_2_w.fits

1_1_3_AvgKern_1.fits    2_1_3_AvgKern_1.fits
1_1_3_AvgKern_2.fits    2_1_3_AvgKern_2.fits
1_1_3_AvgKern_3.fits    2_1_3_AvgKern_3.fits
1_1_3_Noise.fits        2_1_3_Noise.fits
1_1_3_q.fits            2_1_3_q.fits
1_1_3_w.fits            2_1_3_w.fits
```

```
testReducedMF_8_10_xt_noise_z
```

The output file testReducedMF_8_10_xt_noise_z is simply a copy of the input parameter file. The target function for the only σ_h value has three versions, one for each flow velocity. The remaining output files are grouped by the permutations of the μ, σ_h, and ν parameter values. The naming convention for output files follows μ_σ_h_ν_matrix_$\boldsymbol{v}_{x,y,orz}$.fits.

4. Removing $k = 0$ From OLA Inversions

Sometimes when calculating the vertical flow (inverting for 3d_vector_z), the matrix equations when $\boldsymbol{k} = 0$ are difficult to solve. Sometimes, the equations cannot be written and trying to solve for $\boldsymbol{k} = 0$ causes the inversion results to be shifted. To help correct for the shift, the output matrices are calculated with the weights $w(\boldsymbol{k} = 0) = 0$. The output files follow the same naming convention with the addition of 'k0' to the file name.

5. General Target Function

There is a special option to allow any type of target function. If the input file for the target function has compatible dimensions with the kernels, then the target function is used 'as is' and the σ_h parameter is ignored. The expected format for the target function is a FITS file with the first axis containing the y-dimension, the second axis containing the x-dimension, the third axis containing the z-dimension, and the fourth axis containing the scatter dimension.

6. Possible Errors

The files errorHandler.h *ErrorMessages.h, and scripts/messages.py 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; check that all mallocs are successful, otherwise, report error and abort.
- Problems with output:
  - cannot write output files
  - directory does not exist