GUIDE TO USING TIME-DISTANCE INVERSION SOFTWARE: SOLA

JAVORNIK

One method to solve time-distance inversions for local helioseismology uses multi-channel deconvolution (MCD) subtractive optimally localized averaging (SOLA) in the Fourier domain. This document describes how to use the Time-Distance Inversion software for SOLA inversions. The documents OLANotes.pdf and CrossTalk.pdf contain the equations for the SOLA calculation.

All the inversion code is written in C. The software is composed of several shared libraries. The shared libraries parses the input file, create output directories, read and write binary data files, perform matrix and linear equation operations, and perform Fourier transforms. The libraries provide wrappers for the external software packages used by the inversion code. The external software packages are: C FITS File Subroutine Library (CFITSIO), Linear Algebra PACKage (CLAPACK), (CBLAS) Fastest Fourier Transform in the West (FFTW), Lex, and YACC.

Each software release contains the shared libraries, documentation, and sample data 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. SOLA inversions require core.tar and ola.tar. The sample data files are in the ola_test.tar file. The README file contains instructions to build and run the inversion code.

The rest of this document contains instructions to run the inversion code, an explanation of the input parameter file, sample input data files, and sample output data files.

1. General Directions

The command to run the Time-Distance Inversion code is *invert*. For example, the command:

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

The *invert* command takes an input parameter file (explained in section 1.0.1) 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 along with a brief description of the inversion method. Table 3 contains the specific parameters for each inversion.

1.0.1. *Parameter File.* 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

Date: November 26, 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
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, ϵ)
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.	

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 (e.g. _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.

1.0.2. *Input Data Files.* Input and output data files follow the Flexible Image Transport System (FITS) format. The inversion code reads compressed input FITS files ending with (.gz) as well.

1.0.3. *Output Data Files.* 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. A crosstalk parameter in the input parameter file, adds an index in the output file names and they 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. The inversion produces averaging kernel and noise/error covariance files in compressed FITS format, with the extension (.fits.gz). The inversion code computes the error covariance matrix by the formal propagation of errors.

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. RLS inversions, do not have a choice for the target function. OLA and RLS inversions have similar input and output files, with the exception of an overlap matrix and a target function, which is unique to OLA.

A good starting point for the target function is a wide function. The σ_h parameter controls the width of the target function. A narrow or steep target function increases the noise in the fit. **NOTE: The center point of the target function must be the same as the center point of the kernels and the maps. For example, the center point, or zero pixel, has coordinates (81,81) when the pixels in the horizontal direction range from 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
```

Output file names follow 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.) Each permutation of the values of μ and σ_h, generates 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_ϵ_matrix_$\boldsymbol{v}_{x,y,orz}$.fits. NOTE: if using ϵ, ν must also be used. ϵ cannot be used without ν. Epsilon controls the spread of the weights.

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 unrestrained. 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. Other Options

5.1. **General Target Function.** A special option allows 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.

5.2. **Solving for Multiple Target Depths: SOLA Inversions ONLY.** An input target file with dimensions $N_{fz_0} \times N_z$, where N_{fz_0} is greater than one, solves for multiple target depths. The output file names have an additional parameter that is always immediate before the matrix name. The parameter indicates the target depth used to produce the result. For example, an output file for an inversion using all the options follows the format: $\mu\_\sigma\_\nu\_\epsilon\_f_{z0}\_matrix.fits$

Table 4 contains other options for SOLA inversions.

Table 4. Other Options

Makefile Variable	What it does
USE_SINGLE	switches from double precision to single precision
WHICH_CC	changing the C-compiler between GNU and Intel
USE_SVD	switchs linear equation solver between LUDecomposition and SVD
CALC_COND	turns on condition number calculation and calculates the mean of the weights

6. 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