University of Michigan
Space Physics Research Laboratory

TIDI Data Processing Software	CAGE No.	0TK63
	Drawing No.	055-4276
Invert Design and Maintenance	Project	TIDI
	Contract No.	NASW-5-5049
	Page	1 of11

REVISION RECORD

Rev	Description	Date	Author
	Initial Release	31-Dec-03	A. Marshall

Contents

1. References

1) Gell, D. “Line of Sight File Format”, SPRL File 055-4191.

2) Gell, D. “Profile File Format”, SPRL File 055-3532.

3) Gell, D. “File Naming Convention”, SPRL File 055-3545.

4) Ortland, D. “Invert Algorithm Description, SPRL File 055-????.

5) Marshall, A. “Invert User’s Guide”, SPRL File 055-4280.

2. Introduction

The purpose of this document is to educate the maintenance programmer about the Invert program so that s/he can:

- Correct any errors that are found.
- Modify the behavior of the program.
- Rebuild the program as needed when support modules (system libraries, TIDI libraries and object files, etc.) are modified.

2.1 Intended Audience

This document assumes that the reader is a programmer with a good working knowledge of the Fortran programming language, reasonable facility with the Unix operating system, and some understanding of the TIDI data system. An understanding of the netCDF file format is also helpful, since most TIDI data files are of this type.

3. Program Structure

3.1 Overview

The Invert program consumes the TIDI level 1b Line of Sight (LOS) data which are in a single LOS file (see ref. 1 for the file format), and produces a level 2 Profile file (see ref. 2). LOS files will generally contain data for period of about 24 hours, beginning at 00h00 UTC and extending thru the end of the last measurement set of the day (which could be 2-3 minutes into the next day), and are organized sequentially in time, with each ‘record’ (one value or set of values for each variable) detailing the observation made by a single telescope (of which TIDI has four) for a given time. Each LOS ‘record’ consists of a measured emission spectrum, and the collection conditions (time, latitude, longitude, altitude, etc.) for the measurement.

Invert collects a ‘scan’ of data (measurements made with a single telescope during a short time interval as it is moved either up or down to view different altitudes), and processes it (also referred to as inverting it) by using a model of the TIDI instrument (‘forward model’) to determine what atmospheric conditions (wind, temperature, etc.) would create spectra like those measured.

The best fitting atmospheric conditions (the winds and temperatures, etc. which produce spectra most like the measurements) are copied into an output structure which is then written to the netCDF formatted Profile file.

3.2 Program Files

The Invert program is made up of the following modules (routines, subroutines, and functions), which are contained in files presently located in /tidi/tidi_software/invert. After the module name, the file which holds the module is listed, followed by a brief description of the function and/or flow of the module.

INVERT: (from invert.F)
Main routine – opens and reads most input files, calls routines to read input data, process it, and write results

INVERSE_MODEL: (inverse_model.F)
Recover atmospheric profiles from measurements.

OPEN_READ_LOS_FILE: (read_los_file.F)
Open existing LOS file (read-only) and get dimensions and variable ids

CLOSE_LOS_FILE: (read_los_file.F)
Close existing LOS file

READ_LOS_FILE: (read_los_file.F)
Read a 'record' (one measurement) of LOS data, including spectrum and collection conditions

READ_LOS_FILE_BIN_TABLE: (read_los_file_bin_table.F)
Read binning tables from LOS file

READ_LOS_GLOBAL_ATTRIBUTES: (read_los_global_attributes.F)
Read all global attributes from an LOS file

CREATE_OPEN_PRF_FILE: (profile_file.F)
Open a new PRF file and define dimensions, attributes, and variables as needed

CLOSE_PRF_FILE: (profile_file.F)
Close PRF output file

WRITE_PRF_FILE: (profile_file.F)
Write variables (collection conditions and inverted LOS profiles) as needed

GRID_INIT: (grid_init.F)
Determine recovery grid index system and interpolate and average LOS quantities onto recovery grid for use as the initial guess

INIT_GUESS: (init_guess.F)

Compute LOS wind and temperature to be used for initial guess for all measurements in same mode for current scan by performing grid search (by wind speed and temperature) of difference between modeled and measured spectra

FORMOD: (formod.F)
Compute forward model (brightnesses) and first order perturbation kernels for a set of conditions

O2LINSTR: (o2linstr.F)
Compute O_2 line strength corrections for a given temperature

QUADRATURE: (quadrature.F)
Compute quadrature weights for LOS integrals (based on trapezoidal rule)

MATRIX_INVERT: (matrix_invert.F)
Invert matrix using svdcmp algorithm

Also included as part of the Invert code are the following 'include' files (and their use):

inverse_common.inc holds control parameters and intermediate results used by inverse_model and the routines it calls

invert_params.inc holds and defines parameters used by inverse_model and the routines it calls

los_define_read.inc holds values used in reading the netCDF variables in an LOS file

make_prf.inc holds and defines various values used in the creation and writing of prf file variables

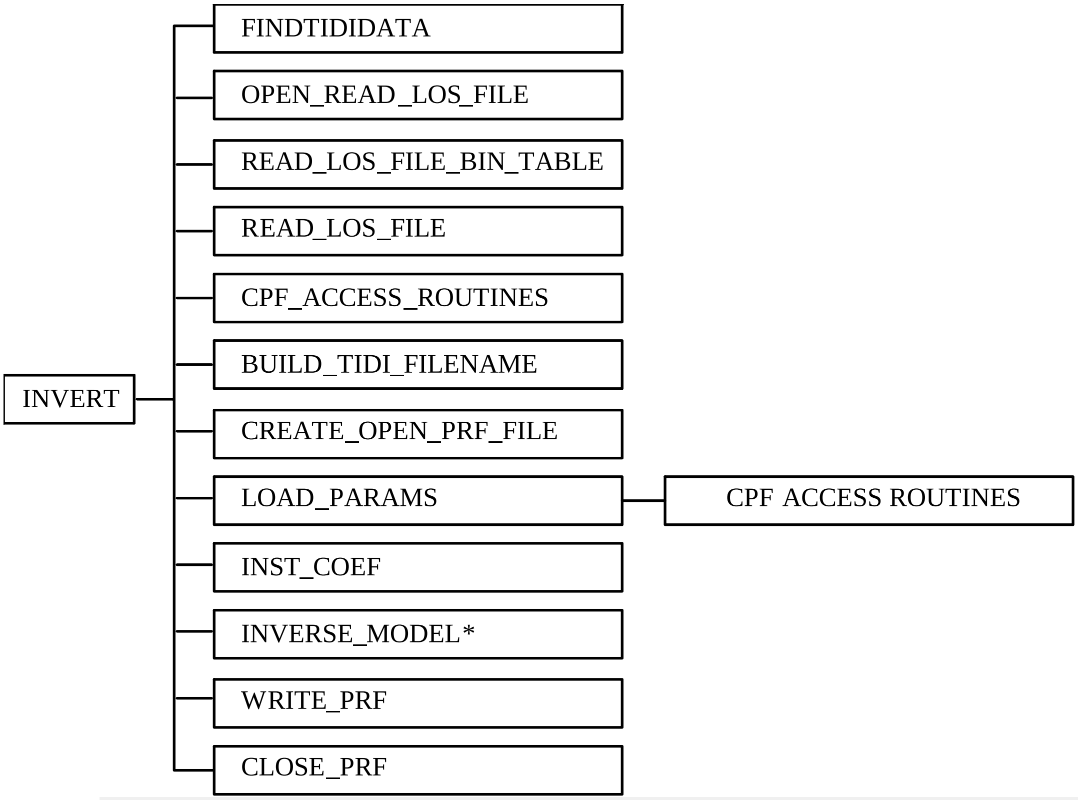
read_prf_global_attributes.inc defines a common block used to read / hold some of the prf files' global attributes

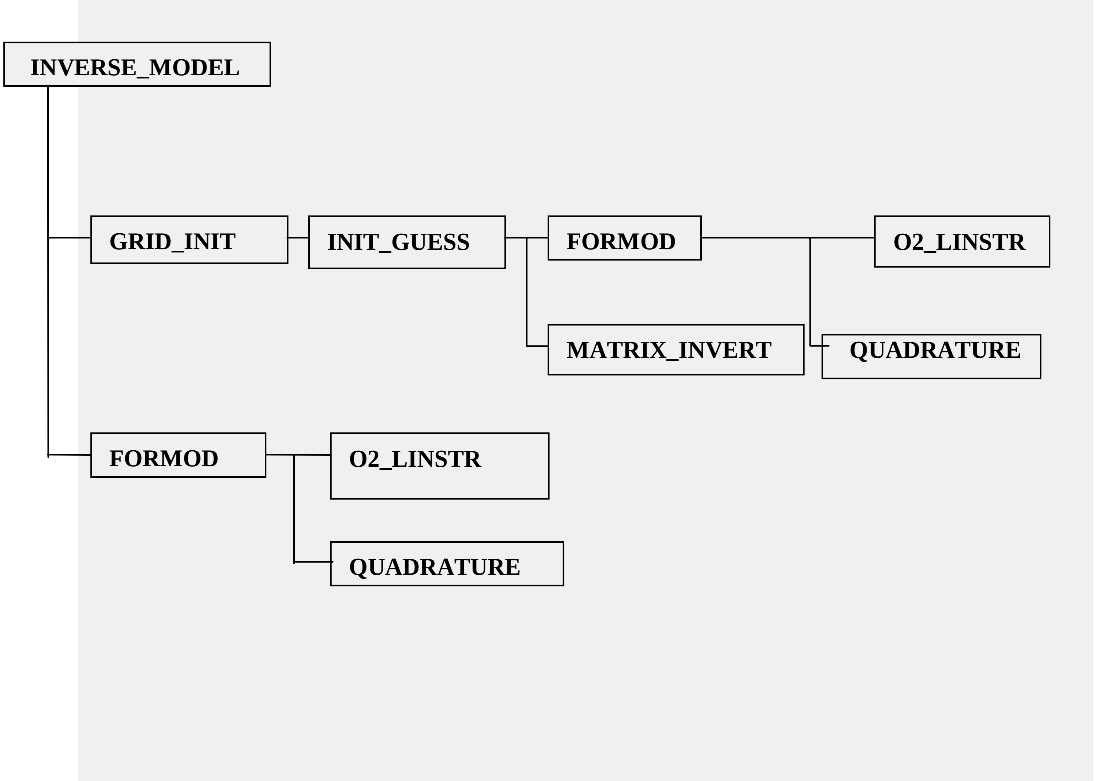
scan_struct.inc holds the 'scan' of los measurements and conditions, which is passed to inverse_model

TIDI_PROFILE.INC defines the data structure profile_struct, which is used to write results to the profile file

3.3 Calling Structure

The Invert program calling structure is shown in the following block diagram.





3.4 Input and Output Files

The following lists the input and output files for the Invert program:

TIDI_*.LOS Input file containing line-of-sight measurements corrected for instrument effects, including spectra and collection conditions

*.CPF Input ‘constant parameter file’, containing numerical constants and instrument parameters

day_mode_control.dat Input file containing the control parameters (ex. which emissions to process and what results to recover) for current run, for daytime table ids.

night3_control.dat Input file containing the control parameters (ex. which emissions to process and what results to recover) for current run, for nighttime table ids.

TIDI_*.PRF Output file containing inverted profiles of atmospheric quantities (ex. inverted line-of-sight winds, temperatures, etc.) and their collection conditions.

For the exact form of the cpf, prf, and los file names, please refer to the File Naming Convention (ref. 3), which is used for these files.

4. Theory of operation

The following pseudo code describes the functioning of the most important and complicated routines, invert and inverse_model, in some detail:

```
INVERT (main routine)

Initialization: (on any error here, abort processing)
        Get any ‘command line’ inputs (input file name, debugging flag)
        Open / read control files – get names of model files, inversion controls
        Open LOS input file by calling open_read_los_file
        Create / open profile output file by calling create_open_prf_file

Prepare Model: (on any error here, abort processing)
        Load parameters from CPF by calling read_cpf and load_params
        Open and read model files for inversion if required (not currently used)

Main Processing:
        While we’re not done, loop (here continue thru errors as best we can)

                Read an LOS measurement by calling read_los_file (on read error, signal and
                abort) (on end of file, process binned data then exit)

                If measurement is part of current scan, add to binned / average values
        Else (if scan table entry value decreases, or if we already have data for a given
        telescope and scan table index, or if there's a time gap between prior
        measurement and this one, or if scan table id changes) process prior scan:

            Fill structure to pass to inverse_model routine
            If needed, prepare model as above
            Call inverse_model to invert the measurements (report any errors, but
            continue processing)
            If inversion was successful:
                Fill output structure, including average values (like latitude)
                Write output to profile file by calling write_prf_file (on write
                error, signal and abort processing)
            End if
            Zero out binned / average values
            Add new data to binned / average values
        End else
    End loop
```

Clean up:

```
    Close input and output files by calling close_los_file and close_prf_file
    Close model files for inversion if required (not currently used)
    Determine and write any final status message, set exit code
```

INVERSE_MODEL (inversion subroutine)

Recover atmospheric profiles from measurements by:

Defining recovery grid and determining an initial guess for profiles by calling grid_init and init_guess.

Generating kernels and a forward model spectrum for current conditions by calling formod

Computing chi^2 of modeled vs. measured brightnesses

Update profile values

If fit is getting better (chi^2 is getting smaller) and we haven't iterated too many times,
loop to 'generating kernels'

Please refer to the Invert Algorithm Description (ref. 4) for further detail regarding the inversion.

5. Maintenance Activities

5.1 Extending

Currently, the Invert code only processes emissions collected using one set of filters, fw_config 3 (O_2 0-0 P9). Other emissions / filter sets can be added to the processing by changing the day and night control files to 'admit' these emissions, adding the relevant filter transmissions and reference velocities to the cpf, and altering load_params.F to read the new values. Also, please check to see that there are no explicit data 'filters' in invert.F keeping these measurements from being used. This section of the code can be found at the comment 'more filters'.

The altitude range over which Invert recovers winds (etc.) is controlled by two factors. The array alt_retrieved in the prf file is created from the variable z, which is read from the cpf (called recov_grid there). Z extends from 50 to 600 km. The variables lo_recov_alt and hi_recov_alt are used to both filter the incoming LOS data (which must be between lo_recov_alt and hi_recov_alt to be processed) and to ensure that the recovered profiles are no larger then they need to be (by limiting alt_retrieved and all recoveries which are written to the prf file to between lo_recov_alt and hi_recov_alt). These ranges can be changed within the limits of the recov_grid values in the cpf by altering lo_recov_alt and hi_recov_alt in the day and night control files from which they are read. No recoveries will take place outside the range of input data, in any case.

Although the profile file format description (ref. 1) references several densities (ex. o3density, the recovered ozone density), no algorithm for recovery of any density has been incorporated into Invert. Code to recover any of the densities should be called after a successful return from the inverse_model subroutine, at which point the winds, temperatures, and volume emission rates are available from which to determine densities.

5.2 Compiling and Building Invert

This program is compiled using the f77 Fortran compiler. It is crucial to note that the module file names end with < .F > , a capital letter, in order to invoke the preprocessor to properly process include file directives and other directives needed to accommodate more than one machine architecture.

Copies of the modules needing change should be made in the programmer's sandbox by checking them out using the mkssi program on the HPUX machines. Once changes are made, the simple command “make” is issued and any changed modules are recompiled and all required modules are linked to form the executable. If include files were added or removed from any modules, the command “make depend” should be issued first to update the list of dependencies on the include files (only needs to be done once, the platform does not matter). The makefile (tidi/tidi_software/invert/makefile) will determine which machine architecture is the target (HPUX or Sun, currently) and place the objects files and the executable in a different directory for each architecture. The makefile details the files which need to be compiled and how to compile them, as well as the dependencies between the files, and will need to be updated if any additional files are added to the program.

Once it is verified that the changes resulted in the desired effect, and no new problems have been introduced, the modules should be checked in and the copies in the project directory resynchronized. Make should then be invoked from the project directory for each machine, followed by “make release” to place the new executable where the production scripts can find it.

See the comments in the makefile for more specific information, such as how to change code involved in reading/writing netCDF files.

5.3 Running Invert

Invert can be run interactively, or in batch mode, with the following commands:

```
/tidi/tidi_software/invert/hpux/invert ‘calling parameters’      (on the HP machines)
```

or

```
/tidi/tidi_software/invert/sun/invert ‘calling parameters’       (on the Sun machine, tidi05)
```

The following parameters are most frequently used:

-i input_filename (used to pass the program an input file location and name)

-o output_filename (used to pass the program an output file location and name)

Further parameters, offering more control of the program, are described in the Invert User's Guide (ref. 5).

Appendix A, Auxiliary Programs

The IDL program /tidi/tidi_software/invert/read_prf.pro reads a PRF file into a structure.