Inverse Problem Program

Rafael E. Banchs

INTRODUCTION

This report presents a brief description of the computational implementation of the inverse version of the time harmonic field electric logging problem. First, the structure of the program and its six modules are presented. Then, a discussion of each module and its most important related subroutines follows. Finally, a brief description of the program user's interface is presented.

GENERAL STRUCTURE OF THE PROGRAM

The "Inverse_problem" program is composed by seven principal modules that perform specific functions. They are the main_program module, which is composed basically by the main program and its function is to call the subroutines in the other modules; the i/o&control module that is in charge of the input/output operations and the validation of the data and the program results; the electromagnetics module, which is in charge of solving the electromagnetic equations for the basic current element; the linear_algebra module, which models the specific logging tool by using the method of moments; the optimization module, which implements two local (gradient methods and Born approximation) and two global (simulated annealing and genetic algorithms) search methods; and the random_generator module, which provides the random numbers required by some of the modules.

The input data is provided by two input files to known: "Inversion Data" and "Optimization". The file "Inversion Data" provides the set of measurements that will be used as inversion data, as well as the information related to the tool and the earthen formation used for the generation of such data. The file "Optimization" provides the information related to the inverse modeling

algorithms to be used and other important parameters as the inversion data mask, noise characteristics, control flags, number of experiments, etc...

The output data is given in six different types of output files depending on the experimental results and on the type of output data requested by the user in the input file “Optimization”. The output file “Inversion Log” is always generated and it contains the recovered conductivity values for each of the performed experiments. On the other hand, the output files “Gradients_out”, “Born_out”, “Annealing_out” and “Genetic_out” are generated only upon request and they contain the evolution of the search process for the respective inversion algorithm. Finally, the file “Failure_report” is generated only when any error or warning occurs during the program execution.

Figure 1 presents a diagram that summarizes the basic structure of the program, the interactions among its modules and the input/output data flow.

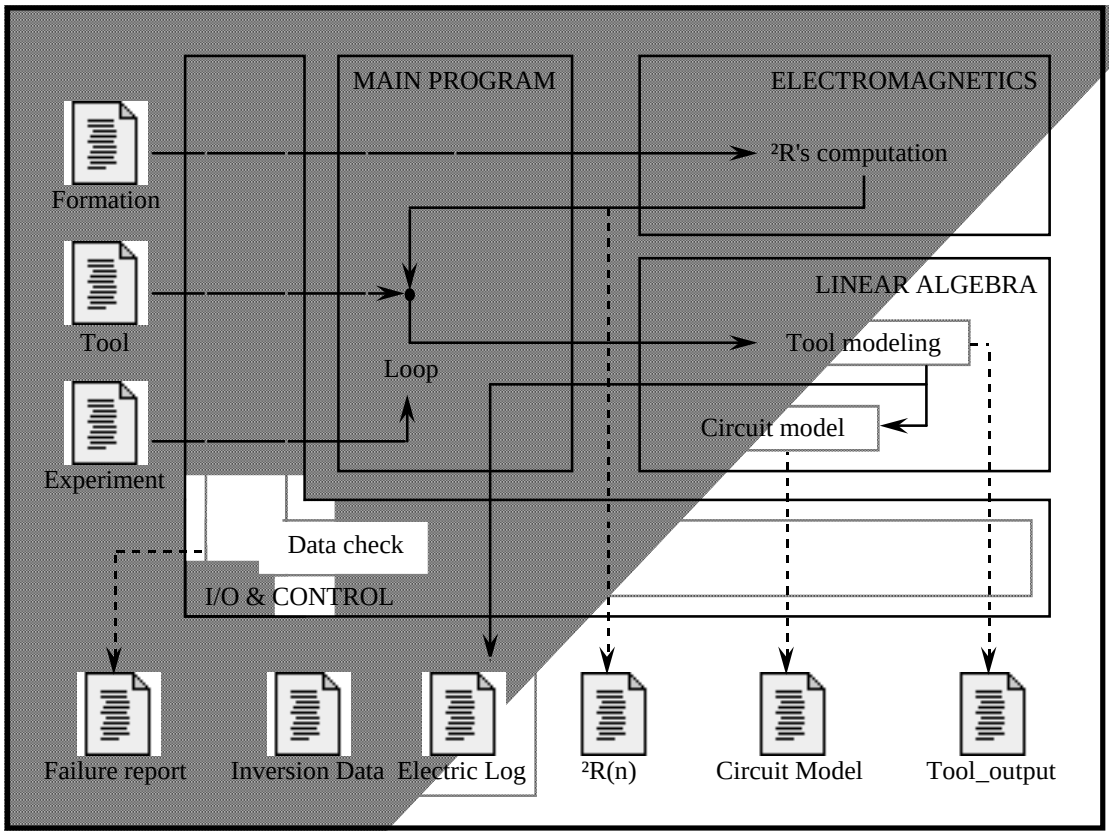

Figure 1: Basic structure of the program “Direct_problem”.

As is illustrated in Figure 1 the “Inverse_Problem” program is composed of seven modules. They are:

1. *Main_program:* This module is specific to the “Inverse_Problem” program and its structure is very similar to the equivalent module in the “Direct_Problem” program. Its basic function is to initialize and finish the execution, as well as to call the subroutines in the other modules according to the specifications in the control flags.
2. *I/O&control:* This is also specific of the “Inverse_Problem” program, but is similar to its equivalent in the “Direct_Problem” program. Three basic functions are performed by this module: loading of input data, failure control, and output data formatting and saving.

3. *Electromagnetics:* This is a module common to the “Direct_Problem” and the “Inverse_Problem” programs. As was already discussed, in section C.1., it is responsible for calculating the set of coefficients ΔR_n.
4. *Linear_algebra:* It is also common for both the “Direct_Problem” and the “Inverse_Problem” programs. Again, as was discussed in section C.1., it computes the logging device’s response.
5. *Local_search:* This module, which is specific of the “Inverse_Problem” program, implements the local search algorithms of gradient methods and the Born approximation. It recursively calls the “Electromagnetics” and the “Linear_algebra” modules.
6. *Global_search:* This module, also specific of the “Inverse_Problem” program, implements the global search methods of simulated annealing and genetic algorithms. It recursively calls the “Electromagnetics” and the “Linear_algebra” modules.
7. *Random_generator:* This module is specific of the “Inverse_Problem” program also. Its basic function is to provide the random values required by some of the other modules. Since the seed is updated by using minutes and seconds every time the inverse program is launched, different pseudo-random sequences are obtained at each execution.