4-7 January 2010, Orlando, Florida

Development of a User Interface for a Regression Analysis Software Tool

N. Ulbrich* and T. Volden**
Jacobs Technology Inc., Moffett Field, California 94035–1000

An easy–to–use user interface was implemented in a highly automated regression analysis tool. The user interface was developed from the start to run on computers that use the Windows, Macintosh, Linux, or UNIX operating system. Many user interface features were specifically designed such that a novice or inexperienced user can apply the regression analysis tool with confidence. Therefore, the user interface's design minimizes interactive input from the user. In addition, reasonable default combinations are assigned to those analysis settings that influence the outcome of the regression analysis. These default combinations will lead to a successful regression analysis result for most experimental data sets. The user interface comes in two versions. The text user interface version is used for the ongoing development of the regression analysis tool. The official release of the regression analysis tool, on the other hand, has a graphical user interface that is more efficient to use. This graphical user interface displays all input file names, output file names, and analysis settings for a specific software application mode on a single screen which makes it easier to generate reliable analysis results and to perform input parameter studies. An object–oriented approach was used for the development of the graphical user interface. This choice keeps future software maintenance costs to a reasonable limit. Examples of both the text user interface and graphical user interface are discussed in order to illustrate the user interface's overall design approach.

I. Introduction

A highly automated regression analysis software tool was developed at Ames Research Center during the past 5 years that uses optimized regression models for the analysis of multivariate experimental data sets (see Refs. [1] and [2] for more detail). The tool is called BALFIT. It was originally intended for the processing of wind tunnel strain–gage balance data sets (Ref. [3]). Since 2007, however, it is also applicable to general multivariate global regression analysis problems.

One of the goals for the development of the regression analysis software tool was to support users of the most popular desktop systems. Therefore, an implementation language had to be selected for BALFIT's development that is supported on multiple operating systems running on different processor architectures. In addition, the chosen implementation language needed a library of state–of–the–art numerical analysis routines and built–in graphical visualization capability in order to speed–up the development of BALFIT. Several implementation languages satisfy these requirements. The authors chose the Interactive Data Language (IDL) for the development of BALFIT because of their familiarity with the language and the availability of the free "IDL Virtual Machine" which greatly simplifies the distribution of compiled code to a larger group of users (see Ref. [4] for a description of IDL).

The BALFIT software package itself consists of three major modules: (i) an *analysis module*, (ii) a *Text User Interface* (*TUI*) module, and (iii) a *Graphical User Interface* (*GUI*) module. Each module's design has features that make it possible to analyze experimental data quickly and reliably. Three features were implemented in the *analysis module*, i.e., BALFIT's compute engine, in support of this goal: (i) a regression

* Aerodynamicist; Jacobs Technology Inc.
** Computer Engineer; Jacobs Technology Inc.

model optimization algorithm (see Ref. [1]), (ii) a simplified data input file format, and (iii) an automated analysis report generation feature. Both versions of BALFIT's user interface, i.e., the *TUI* and the *GUI*, allow the user to interact with the *analysis module*. They assign (i) input file names, (ii) output file names, and (iii) analysis settings for a specific analysis task. Reasonable default combinations of the analysis settings are used in both the *TUI* and the *GUI* that make it possible for a novice or inexperienced user to generate regression models of the given experimental data that meet strict statistical quality requirements.

Differences between the *TUI* and the *GUI* exist because they were developed for different purposes. The *TUI* primarily supports the ongoing development of the *analysis module*. The *GUI*, on the other hand, is intended for the efficient use of BALFIT in a production environment. Table 1 below lists advantages and disadvantages of the two versions of BALFIT's user interface.

Table 1: Comparison of *TUI* and *GUI* (*advantage* $\Longrightarrow [+]$, *disadvantage* $\Longrightarrow [-]$)

ATTRIBUTE	*TUI*	*GUI*
DEVELOPMENT EFFORT	$[+]$	$[-]$
MAINTENANCE	$[+]$	$[-]$
EASE–OF–USE	$[-]$	$[+]$
RUN–TIME ENVIRONMENT COSTS	$[-]$	$[+]$

BALFIT may be run on an analysis computer using either the *TUI* or the *GUI* version of the executable. The *TUI* version can only be run using the "IDL Development Environment," i.e., a licensed version of the IDL software. It is written in sequential logic and can easily be modified. The *GUI* version, on the other hand, can be run using a free runtime utility called "IDL Virtual Machine." Therefore, the *GUI* version does not require a licensed version of the IDL software package. Overall, the *GUI* is easier to use than the *TUI*. It was developed (i) to provide a modern interface to the user and (ii) to minimize software distribution costs.

A first version of the *GUI* was completed in 2006 (see Ref.[5]). Significant improvements of the *analysis module* were made since that time. Now, the *analysis module* supports classical regression analysis as well as wind tunnel strain–gage balance data analysis applications. Therefore, it became necessary to completely revise the original version of the *GUI*. It was also decided to rewrite the *GUI* using an object–oriented approach. This improvement will make it easier to maintain the software in the future.

In the next section of the paper the architecture of the BALFIT application is reviewed. Then, key features of the *TUI* and the *GUI* are discussed.

II. Model–View–Controller Architecture

The BALFIT application uses the so–called "Model–View–Controller" (MVC) architecture. The MVC architecture is a design pattern that simplifies application development and maintenance (see Ref. [6] and [7] for more detail). The goal of the MVC design pattern is to separate the application object (*Model*) from the way it is represented to the user (*View*) and from the way in which the user controls it (*Controller*). It achieves this by splitting the application into the three logical components shown in Fig. 1a. The three components are described as follows:

Model – A model represents an application's data and contains the logic for accessing and manipulating that data. In BALFIT, the model component is the *analysis module*. It contains all logic related to performing regression analysis tasks and provides an interface for the other components.

View – The view controls the "look and feel" of the application and provides facilities (i) to collect data from the user and (ii) to display messages from the model (*analysis module*).

Controller – The controller is responsible for intercepting and translating user input into actions to be performed by the model (*analysis module*). Together, the *View* and the *Controller* define the "user interface" that the user sees when using the BALFIT application (see Fig. 1b).

The MVC architecture supports multiple views of the same model and provides efficient modularity which allows development of components to proceed in parallel. These features have allowed both a *TUI* and *GUI* to be developed independently. In addition, this architecture simplifies growth which has allowed

BALFIT to be extended to support both classical regression analysis and strain–gage balance data analysis tasks. Each analysis task in BALFIT is implemented in a separate *View* and *Controller* object which is tailored specifically for the task. This arrangement makes it possible to add analysis tasks to the *GUI* without concern for affecting other analysis tasks.

Figure 2 shows the actual implementation of the MVC architecture in BALFIT. BALFIT's user interface module and the *analysis module* are depicted. The user interface module, i.e., the *TUI* or the *GUI*, is used to select (i) the regression analysis type, (ii) the analysis task (by assigning the software application mode), (iii) the analysis settings, and (iv) the input and output file names. These selections are sent to the *analysis module*. The user interface module also displays all diagnostic or error messages that it receives from the *analysis module*. The analysis module is BALFIT's data processing and compute engine. It processes the input files, performs the data analysis as specified by the selected analysis settings, and creates output files that contain the results of the analysis. One of the output files, i.e., the analysis report, is prepared in PostScript format. It may be converted to the more convenient Portable Document Format (PDF) using a PostScript–To–PDF converter like *GhostScript* or *Adobe Acrobat Distiller*.

BALFIT supports two regression analysis types: (i) classical regression analysis and (ii) strain–gage balance data analysis. The first analysis type allows the user to solve regression analysis problems of general multivariate experimental data sets. The second analysis type performs the regression analysis of wind tunnel strain–gage balance calibration data by applying an iterative technique that is used in the aerospace testing community. Three software application modes are supported if classical regression analysis is chosen (see Fig. 3a). Seven software application modes are supported if strain–gage balance data analysis is chosen (see Fig.3b). Each software application mode uses different analysis settings that influence the final result of the data analysis task. A combination of default assignments of these analysis settings was selected such that a good analysis result for most experimental data sets can be obtained. – In the next two sections of the paper key elements of both the *TUI* and the *GUI* are discussed in greater detail.

III. Text User Interface (*TUI*)

The basic layout of BALFIT's *TUI* goes back to 2004 when the development of BALFIT first started. The *TUI* is the prototype of BALFIT's user interface. It is primarily used (i) to develop features of the analysis module and (ii) to test different combinations of analysis settings that influence the outcome of the data analysis. The *TUI* uses simple sequential logic. Therefore, it can easily be modified if changes in the *analysis module* cause changes in the user interface. All user interaction with the *TUI* is done in a "question–and–answer" like fashion. First, a selection (an input or output file name or an analysis setting) is made. Then, the user is asked to confirm the selection. These steps have to be repeated for every input that is needed for a selected software application mode.

Many combinations of analysis settings for each software application mode were tested using the *TUI* in order to find specific combinations of default settings for the user interface that will lead to a good analysis result for each analysis task. This "expert" knowledge is hidden in the user interface's combination of default analysis settings. It makes it easier for an inexperienced user to produce good analysis results without getting lost in the large number of possible setting combinations. The default choice for a specific analysis setting is the first option that is presented to the user in the user interface.

A single software application mode has been chosen to illustrate the layout of both the *TUI* and the *GUI* screen. It is the "Data Reduction Matrix Calculation" mode that is needed for the regression analysis of strain–gage balance data (see also Fig. 3b). Figure 4 shows the typical layout of the *TUI* screen for this example. In the *TUI* every input file name, analysis setting, and output file name must be selected in sequential order. Hidden connections between different analysis settings and the contents of certain input files may exist. The *TUI*'s logic takes these connections into account and will only request input for those analysis settings that have not yet implicitly been assigned. This feature prevents the user from accidently selecting incompatible analysis settings that may lead to confusing analysis results.

IV. Graphical User Interface (*GUI*)

The official release of BALFIT has a *GUI* that is more efficient to use and has lower runtime environment

costs than the *TUI*. As an example, Fig. 5a shows the overall layout of the *GUI* for the software application mode that was used for the discussion of the *TUI* (see also Fig. 4). Several design patterns were used in the *GUI* in order to improve its usability. A total of six so–called "interaction design patterns" were applied during the development of the *GUI* (see Refs. [8], and [9] for a detailed description of these patterns). They can be described as follows:

Sovereign Posture – The *GUI* is setup such that BALFIT dominates the screen. It occupies the user's full attention and allows all controls for a task to be seen at once.

Stack of Working Surfaces – Analysis tasks for each regression analysis type are arranged as a set of tabbed pages. This approach makes every task easily visible and gives each task space for its controls.

Status Display – A panel is shared by all tasks that provides space for the application to display status messages.

Form – All inputs, settings, and outputs for each task are arranged on a single screen, allowing the user to see every setting needed to perform the task at a glance.

Small Set of Values – Every user–controlled setting has choices presented like the old–fashioned automobile radio buttons. This setup allows the user to simultaneously see all choices and select only one.

Go Back to a Safe Place – This feature of BALFIT allows the user to return to a set of known analysis settings. The user does this operation using the "Reset Options" button.

In contrast to the *TUI*, which uses a "question–and–answer" based workflow, the *GUI*'s workflow is designed to be much simpler. It is modeled on the familiar task of "filling out" and "submitting" a form. The user merely selects an analysis task by clicking on the corresponding tab. This action brings the task's form to the foreground. The form is arranged with required inputs at the top, various analysis settings in the middle, and outputs towards the bottom. The user simply supplies inputs and values for all settings and, after completion, submits the form by clicking on the "GO" button. This action signals the *analysis module* (i) to perform the requested task and (ii) to display its report in another window using a helper application. During the task execution, feedback is provided from the *analysis module* to the *GUI*'s status display at the bottom of the application window.

Most challenges associated with the deployment of a cross–platform application like BALFIT were addressed by simply selecting IDL for its development. Some operating system related problems, however, still remained to be solved. These problems were addressed by separating details of the runtime environment differences from the compiled application. Then, operating system dependent parameters can simply be read from a *Preferences File* and a *GUI Properties File*. This basic concept is depicted in Fig. 5b. The usefulness of this approach can easily be illustrated using a few examples:

Preferences File – BALFIT produces reports in the PostScript language. These reports must be viewed using a "helper" application. The helper application may either view the PostScript file directly (e.g., using *Ghostview* or *GSview*), or translate it to PDF. The resulting PDF file may be viewed with the *Acrobat Reader* application. Once a report is generated, BALFIT executes the helper application which displays the report. Each desktop system can potentially use different applications to perform these functions. BALFIT allows the helper application to be specified in the *Preferences File* that is read at application start. This makes it possible to specify this detail independent of the compiled application, and allows a user to choose the helper application(s) based on their preference. The *Preferences File* is also a convenient place to store the path to the user's working directory, so that the application can "remember" where the user was last working and resume in this directory at startup.

GUI Properties File – The *GUI*'s presentation is dependent on the user's screen resolution and the fonts available. These details vary from installation to installation. BALFIT uses a *GUI Properties File* to hold font name and size information that allows these properties to be specified separate from the compiled application.

In some cases, there are interactions between different analysis settings or between a single analysis setting and the contents of an input file. BALFIT automatically takes care of these dependencies in order to keep things simple for the user. In a *TUI*, these interactions are easy to handle since the user interface logic is *sequential* and the order of setting specification is predictable. A *GUI*, on the other hand, is *event–driven*, making the order of specification impossible to predict. BALFIT's *GUI* design models the dependency between settings explicitly using a "behaviorial design pattern" called *mediator* (cf. Ref. [7]). This design pattern makes it possible to clearly define relationships between settings and to enforce logic that controls

how the settings behave independent of the order of specification. In addition, this design pattern simplifies maintenance, since the interaction between settings is defined in one place and not spread throughout the application's logic.

Figures 6a and 6b provide an example of how BALFIT handles the dependency between two settings. The settings are called "Write Regression Coefficient Matrix to file" and "Output File, Regression Coefficient Matrix." In Fig. 6a, the setting "Write Regression Coefficient Matrix to file" is set to "no", meaning that a regression coefficient matrix will not be written to a file. Because of this, the widget allowing an output file name to be specified is disabled. In Fig. 6b, the user has changed the setting "Write Regression Coefficient Matrix to file" to "yes", meaning that a regression coefficient matrix file should be written. BALFIT responds by executing logic that makes the "Output File, Regression Coefficient Matrix" setting active, allowing this file name to be specified.

V. Documentation of User Interface Selections

In regression analysis, like in any scientific analysis, it is of critical importance to record all steps and selections that were made in order to obtain the final analysis result. Only this approach makes it possible for a researcher to know how a specific analysis result was obtained if it has to be revisited at a future point in time. Therefore, all final user interface selections (input file names, output file names, and analysis settings) assigned in both the *TUI* and the *GUI* are automatically saved and reported on the analysis settings page of BALFIT's analysis report. The information shown on this page directly links the final analysis results to selections that were made in the user interface at the time the regression analysis was performed. Figure 7 shows the contents of the analysis settings page from the analysis report file for the two user interface examples that are depicted in Fig. 4 and Fig. 5a.

VI. Summary and Conclusions

A user interface was implemented in a highly automated regression analysis tool called BALFIT. The user interface was developed so that the analysis tool's executable can be run on multiple operating systems. The user interface comes in two versions. The *TUI* version is easy to modify and maintain. Therefore, it is primarily used for the ongoing development of BALFIT's analysis module. The official release of BALFIT, on the other hand, has a *GUI*. The *GUI* is easier to use than the *TUI*. In addition, the *GUI* has lower runtime environment costs enabling an efficient distribution of the software to a larger group of users.

BALFIT was developed from the start as a tool that makes it possible for an inexperienced user to obtain regression analysis results for multivariate experimental data sets with confidence. Key features of BALFIT's modules make it possible to achieve this goal. The analysis module uses an optimization process to identify a regression model that meets strict statistical quality requirements. The user interfaces recommend default analysis setting combinations that yield good analysis results. Finally, the *GUI*s ease–of–use and the fact that all final user interface selections are saved and displayed in the analysis report file make it easier to generate repeatable, reliable, and well documented analysis results.

VII. Acknowledgements

The authors would like to thank Jon Bader of NASA Ames Research Center for his critical and constructive review of the final manuscript. The work reported in this paper was partially supported by NASA's Aeronautic Test Program and the Wind Tunnel Division at Ames Research Center under contract NNA09DB39C.

VIII. References

[1]Ulbrich, N., "Regression Model Optimization for the Analysis of Experimental Data," AIAA 2009–1344, paper presented at the 47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida, January 2009.

[2]Ulbrich, N. and Volden, T., "Regression Analysis of Experimental Data Using an Improved Math Model Search Algorithm," AIAA 2008–0833, paper presented at the 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 2008.

[3]Ulbrich, N. and Volden, T., "Strain–Gage Balance Calibration Analysis Using Automatically Selected Math Models," AIAA 2005–4084, paper presented at the 41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Tucson, Arizona, July 2005.

[4]Research Systems, Inc., "IDL User's Guide," Interactive Data Language, Version 6.1, Boulder, Colorado, July 2004.

[5]Ulbrich, N. and Volden, T., "Development of a New Software Tool for Balance Calibration Analysis," AIAA 2006-3434, paper presented at the 25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference, San Francisco, California, June 2006.

[6]Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M., *Pattern–Oriented Software Architecture – A System of Patterns*, 1st ed., Vol. 1, Wiley Series in Software Design Patterns, John Wiley & Sons Ltd., Chichester, New York, 1996, pp.125–143.

[7]Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns: Elements of Reusable Object–Oriented Software*, 1st ed., Addison Wesley Professional Computing Series, Addison Wesley Longman, Inc., Reading, Massachusetts, 13th printing, 1997.

[8]Tidwell, J., *Designing Interfaces: Patterns for Effective Interaction Design*, 1st ed., O'Reilly Media, Inc., Sebastopol, California, 2005.

[9]Cooper, A., *About Face: The Essentials of User Interface Design*, 1st ed., IDG Books Worldwide, Inc., An International Data Group Company, Foster City, California, 1995.

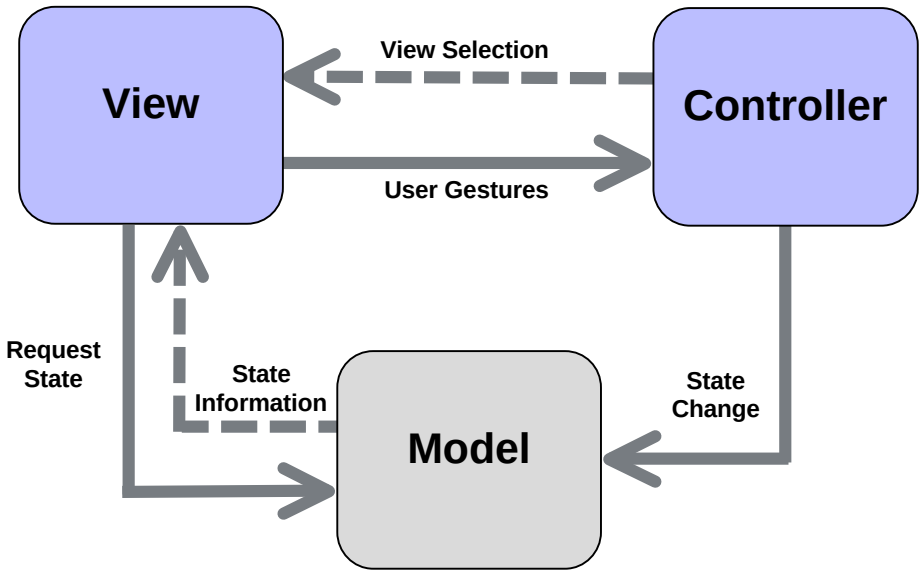


Fig. 1a Description of the "Model–View–Contoller" (MVC) architecture (see also Ref. [7], pp.4–6).

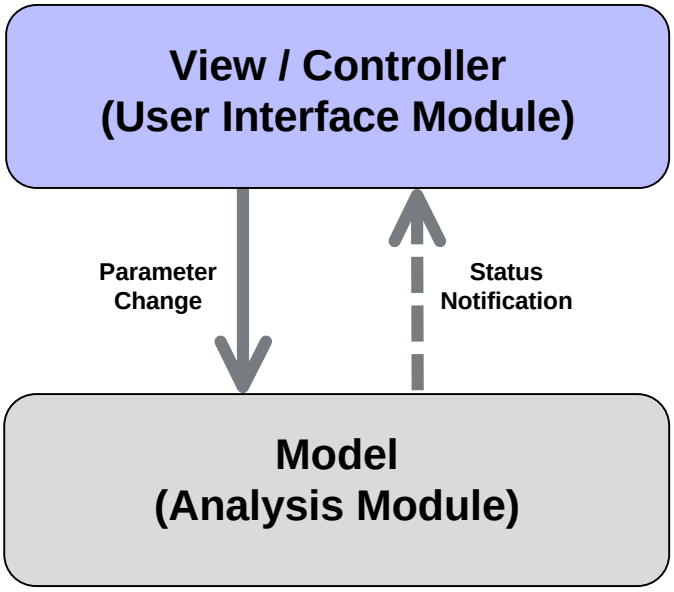


Fig. 1b The connection between the MVC architecture and replacable user interfaces.

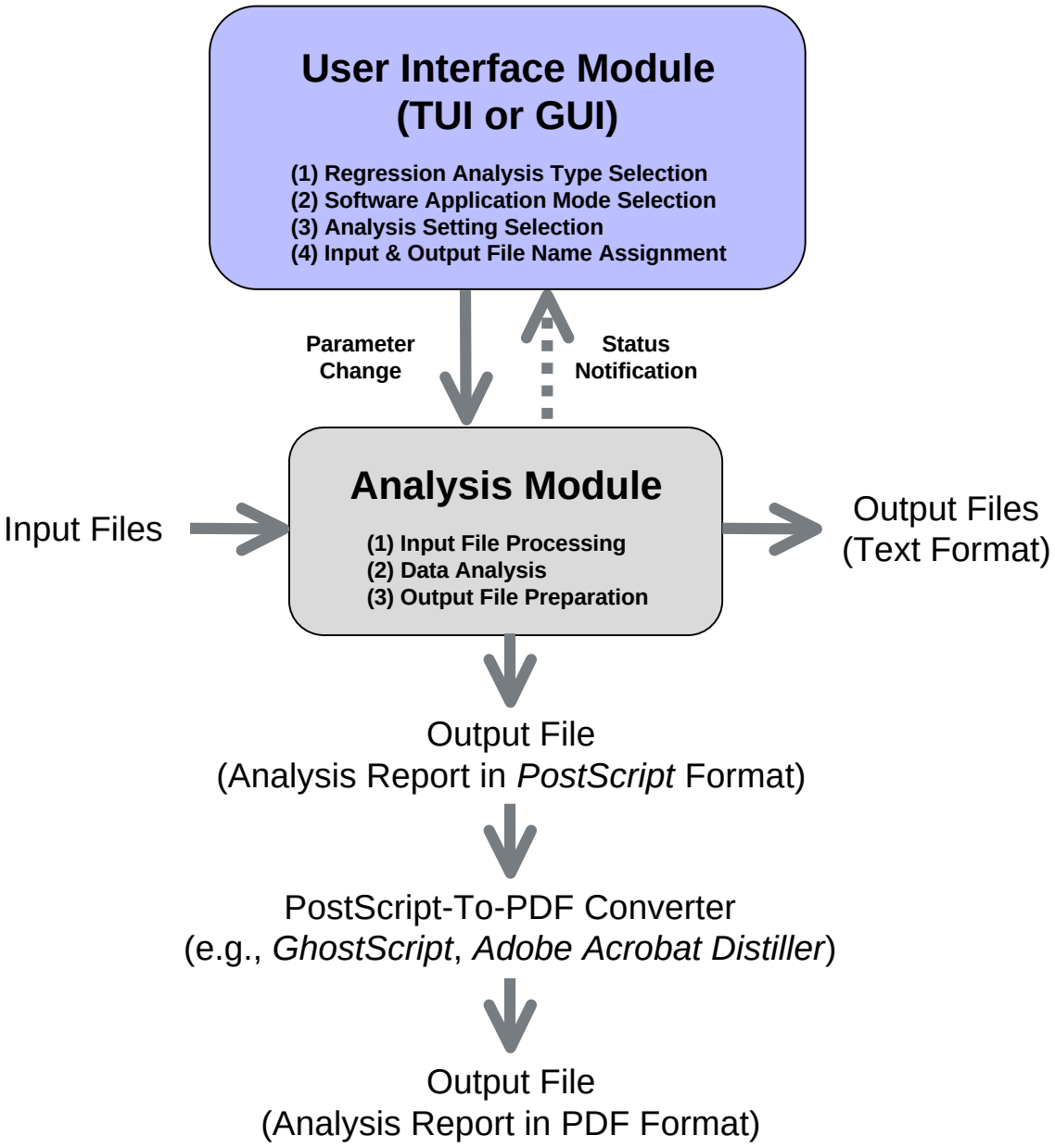


Fig. 2 Implementation of the MVC architecture in BALFIT.

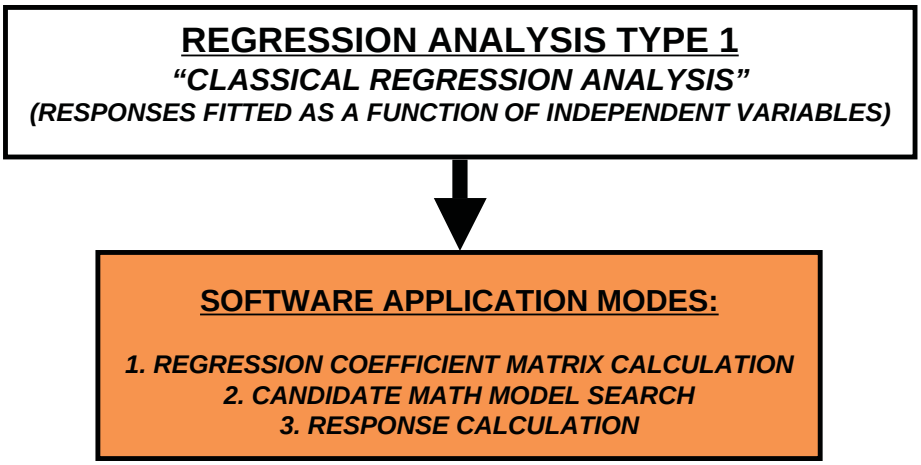


Fig. 3a Software application mode choices for classical regression analysis applications.

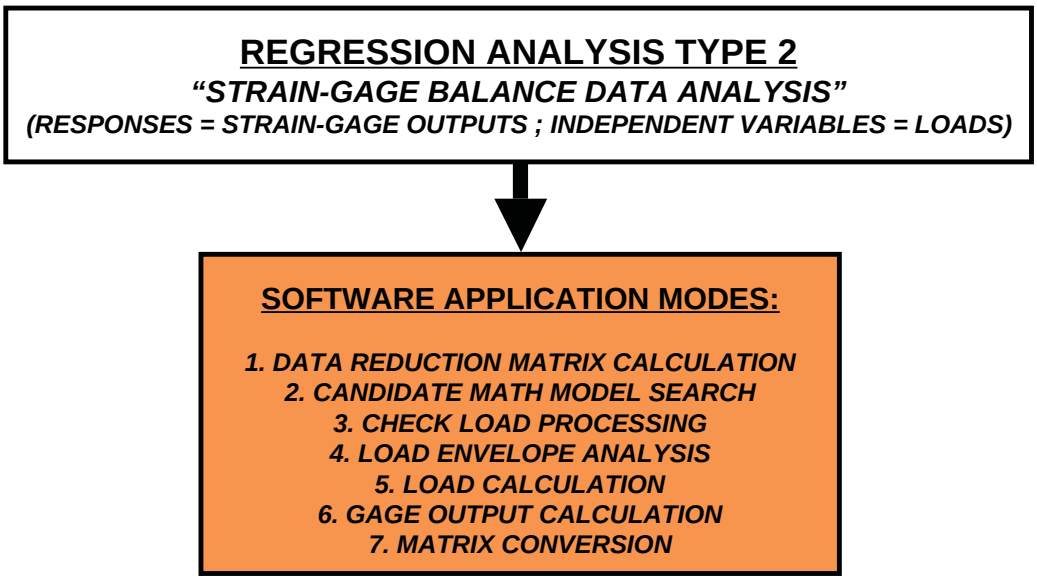


Fig. 3b Software application mode choices for wind tunnel balance data analysis applications.

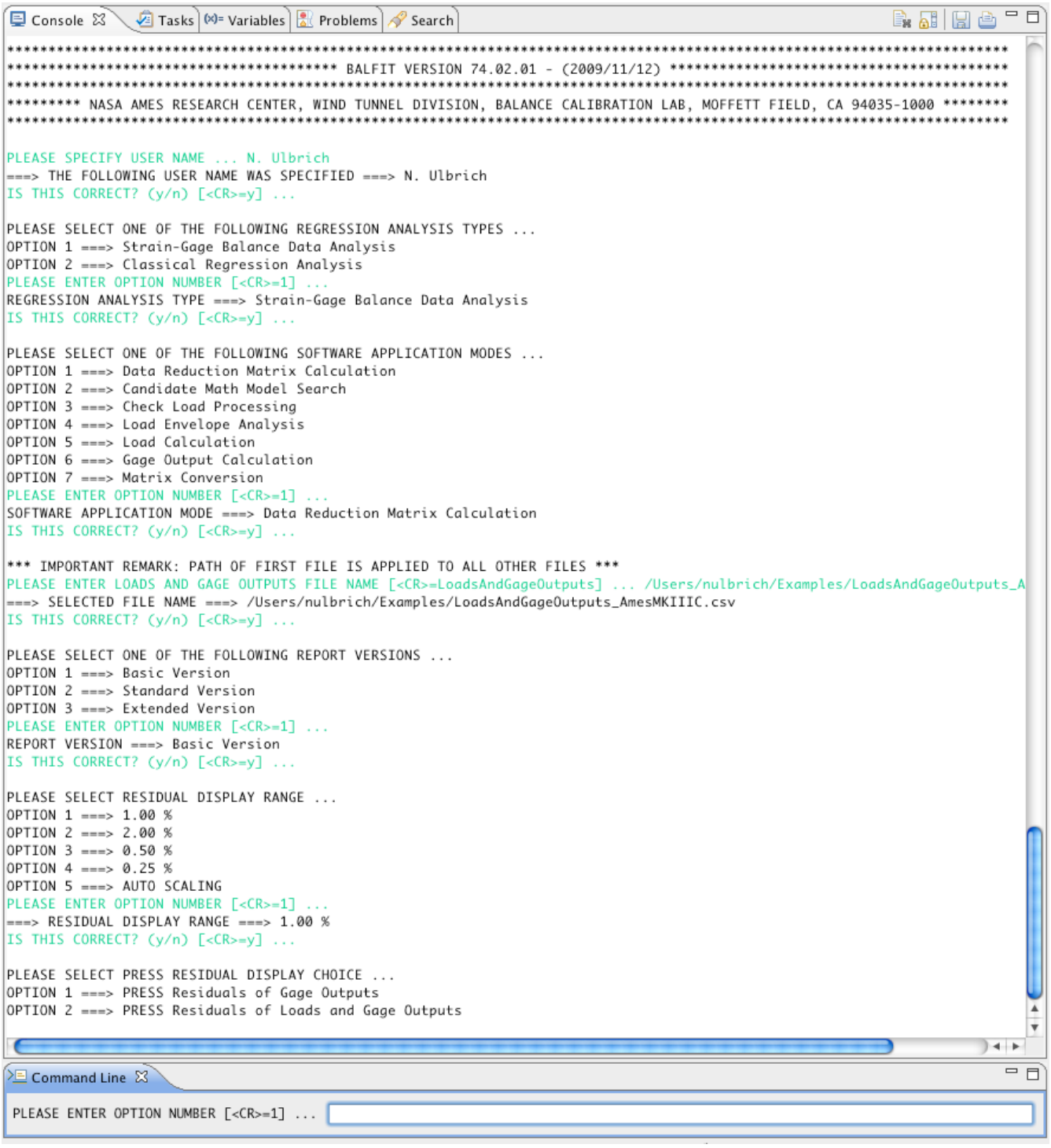


Fig. 4 Example of text user interface screen for "Data Reduction Matrix Calculation" mode.

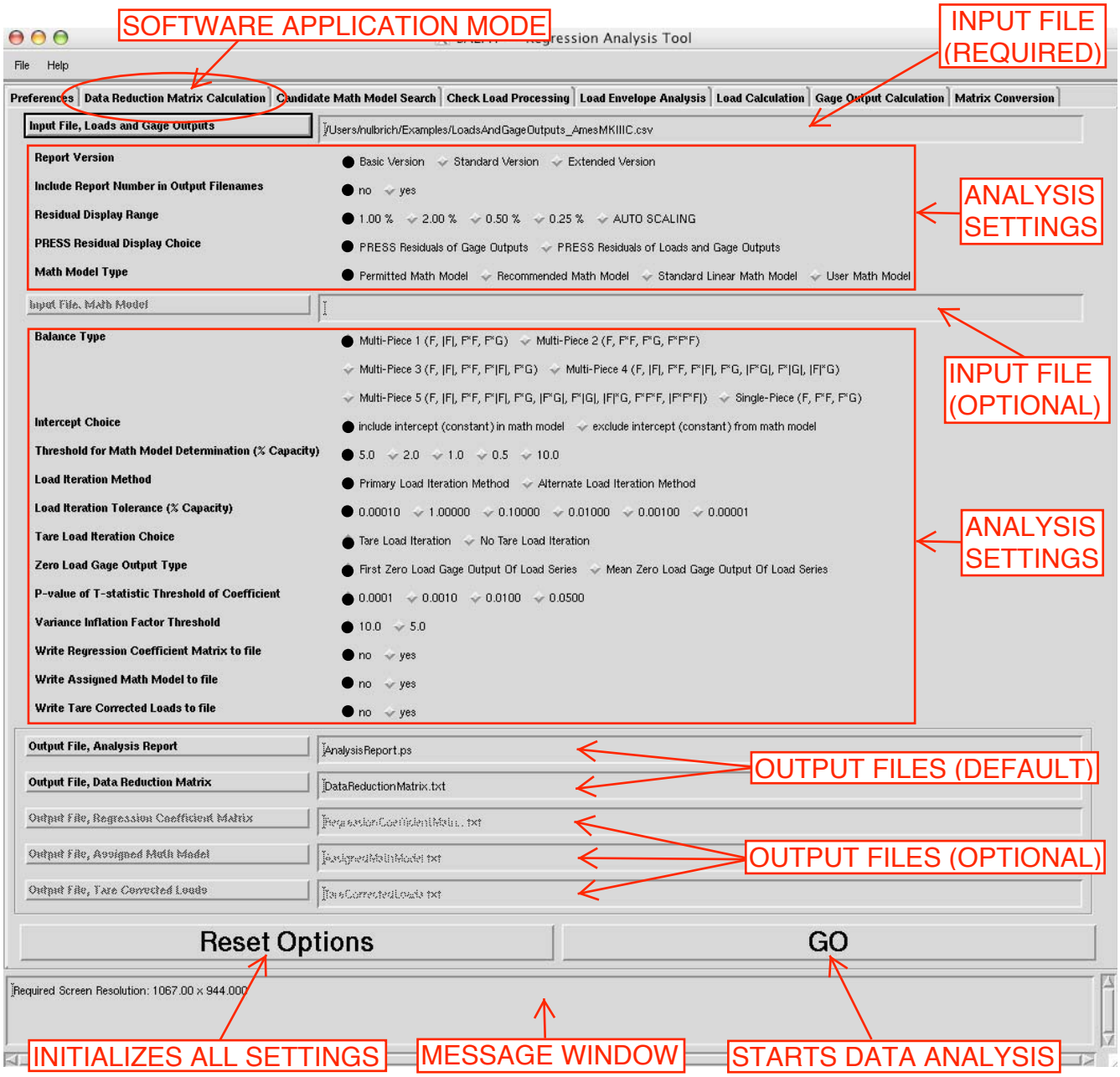


Fig. 5a Example of graphical user interface screen for "Data Reduction Matrix Calculation" mode.

Fig. 5b Management of installation specific graphical user interface characteristics.

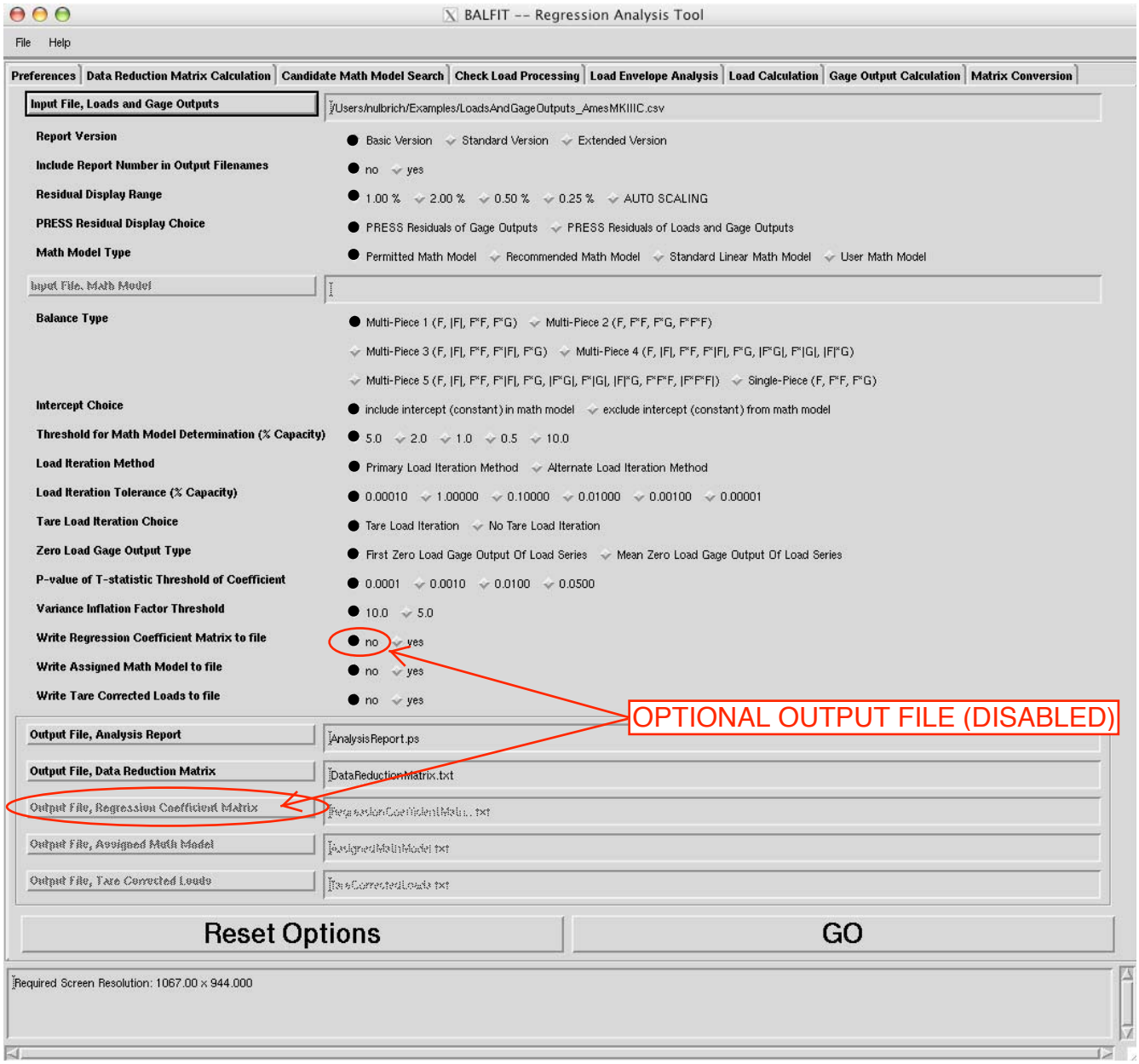


Fig. 6a Example of an interaction between settings of the graphical user interface screen (situation 1).

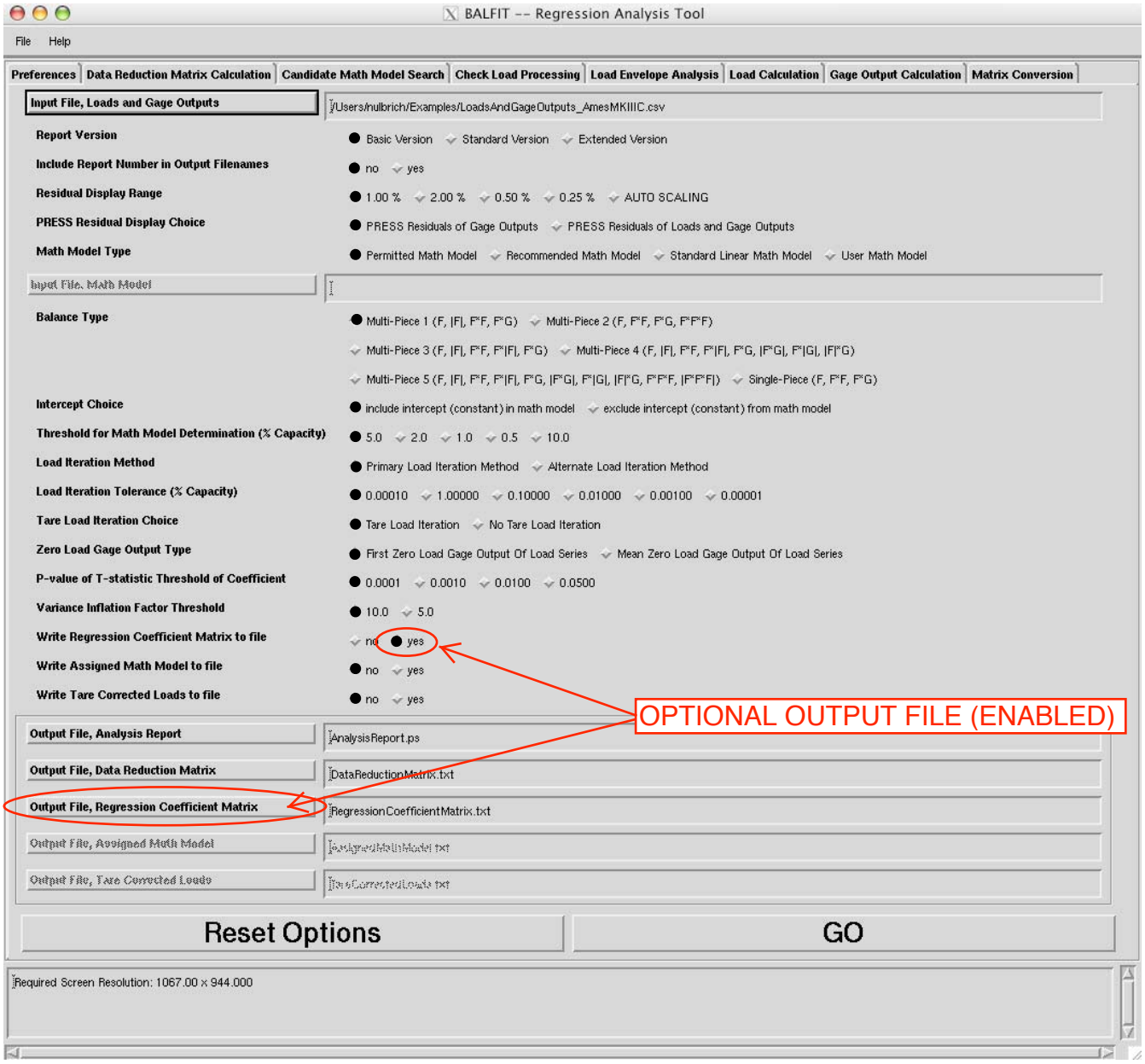
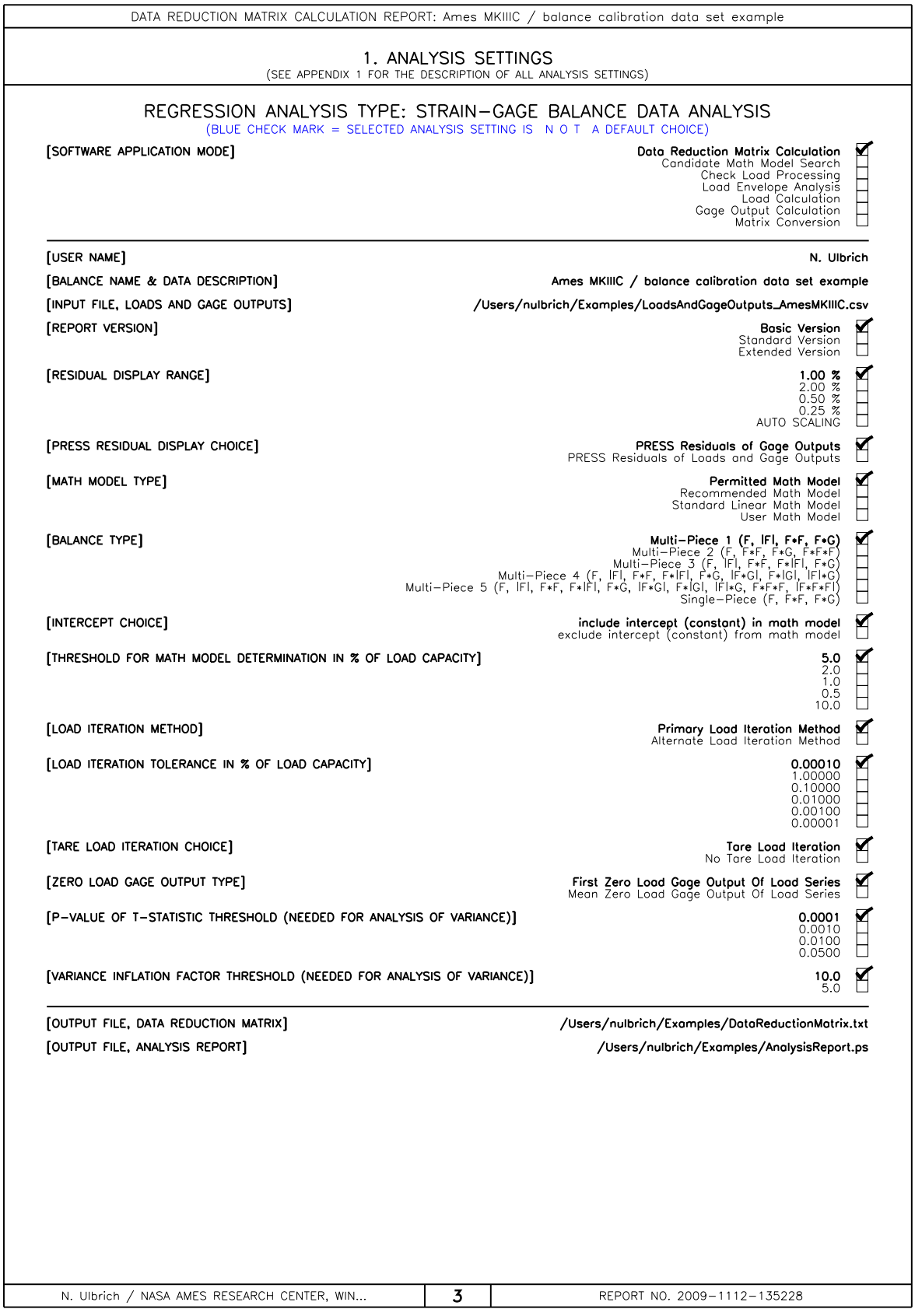


Fig. 6b Example of an interaction between settings of the graphical user interface screen (situation 2).



DATA REDUCTION MATRIX CALCULATION REPORT: Ames MKIIIC / balance calibration data set example

1. ANALYSIS SETTINGS

(SEE APPENDIX 1 FOR THE DESCRIPTION OF ALL ANALYSIS SETTINGS)

REGRESSION ANALYSIS TYPE: STRAIN-GAGE BALANCE DATA ANALYSIS

(BLUE CHECK MARK = SELECTED ANALYSIS SETTING IS N O T A DEFAULT CHOICE)

[SOFTWARE APPLICATION MODE]	**Data Reduction Matrix Calculation**	☑
	Candidate Math Model Search	☐
	Check Load Processing	☐
	Load Envelope Analysis	☐
	Load Calculation	☐
	Gage Output Calculation	☐
	Matrix Conversion	☐
[USER NAME]	**N. Ulbrich**	
[BALANCE NAME & DATA DESCRIPTION]	**Ames MKIIIC / balance calibration data set example**	
[INPUT FILE, LOADS AND GAGE OUTPUTS]	**/Users/nulbrich/Examples/LoadsAndGageOutputs_AmesMKIIIC.csv**	
[REPORT VERSION]	**Basic Version**	☑
	Standard Version	☐
	Extended Version	☐
[RESIDUAL DISPLAY RANGE]	**1.00 %**	☑
	2.00 %	☐
	0.50 %	☐
	0.25 %	☐
	AUTO SCALING	☐
[PRESS RESIDUAL DISPLAY CHOICE]	**PRESS Residuals of Gage Outputs**	☑
	PRESS Residuals of Loads and Gage Outputs	☐
[MATH MODEL TYPE]	**Permitted Math Model**	☑
	Recommended Math Model	☐
	Standard Linear Math Model	☐
	User Math Model	☐
[BALANCE TYPE]	**Multi-Piece 1 (F, |F|, F*F, F*G)**	☑
	Multi-Piece 2 (F, F*F, F*G, F*F*F)	☐
	Multi-Piece 3 (F, |F|, F*F, F*|F|, F*G)	☐
	Multi-Piece 4 (F, |F|, F*F, F*|F|, F*G, |F*G|, F*|G|, |F|*G)	☐
	Multi-Piece 5 (F, |F|, F*F, F*|F|, F*G, |F*G|, F*|G|, |F|*G, F*F*F, |F*F*F|)	☐
	Single-Piece (F, F*F, F*G)	☐
[INTERCEPT CHOICE]	**include intercept (constant) in math model**	☑
	exclude intercept (constant) from math model	☐
[THRESHOLD FOR MATH MODEL DETERMINATION IN % OF LOAD CAPACITY]	**5.0**	☑
	2.0	☐
	1.0	☐
	0.5	☐
	10.0	☐
[LOAD ITERATION METHOD]	**Primary Load Iteration Method**	☑
	Alternate Load Iteration Method	☐
[LOAD ITERATION TOLERANCE IN % OF LOAD CAPACITY]	**0.00010**	☑
	1.00000	☐
	0.10000	☐
	0.01000	☐
	0.00100	☐
	0.00001	☐
[TARE LOAD ITERATION CHOICE]	**Tare Load Iteration**	☑
	No Tare Load Iteration	☐
[ZERO LOAD GAGE OUTPUT TYPE]	**First Zero Load Gage Output Of Load Series**	☑
	Mean Zero Load Gage Output Of Load Series	☐
[P-VALUE OF T-STATISTIC THRESHOLD (NEEDED FOR ANALYSIS OF VARIANCE)]	**0.0001**	☑
	0.0010	☐
	0.0100	☐
	0.0500	☐
[VARIANCE INFLATION FACTOR THRESHOLD (NEEDED FOR ANALYSIS OF VARIANCE)]	**10.0**	☑
	5.0	☐
[OUTPUT FILE, DATA REDUCTION MATRIX]	**/Users/nulbrich/Examples/DataReductionMatrix.txt**	
[OUTPUT FILE, ANALYSIS REPORT]	**/Users/nulbrich/Examples/AnalysisReport.ps**	

Fig. 7 Example of analysis settings page from report file for "Data Reduction Matrix Calculation" mode.