Implementation of A Java-Based Graphical User Interface for GSTARS 2.1

Francisco J. M. Simões[1], David Patterson[2], and Chih Ted Yang[3]

Abstract

In this paper we summarize the approach and methodology behind the development of a graphical user interface (GUI) for GSTARS 2.1. GSTARS 2.1 is a numerical model for the simulation of flow and sedimentation in large alluvial rivers, and is the most recent version of an archetype command line program that communicates with the user by means of ASCII files with a rigid format. The interface developed, using the Java programming language, wraps the program in a new graphical and interactive user environment without the need to change any of the original numerical code, allowing the re-use of legacy code.

Keywords: numerical model, GUI, graphical user interface, Java, alluvial river simulation.

1. Introduction

The weight of legacy codes is still an important consideration in the development of new applications in hydraulics. The thousands of lines of code usually needed even in relatively small applications are many times still intermingled by code developed many years earlier. Moreover, the need to retain some compatibility with old data files puts an additional burden in the development of new versions of codes that were first written when the alphanumeric 25-line by 80-column terminal was the state-of-the-art. However, current demands on applications require friendly environments based on graphical user interfaces, usually integrated in networked solutions that are platform-independent. In this paper we present our approach to make both paradigms coexist.

GSTARS is a program implementing a hydraulics and sediment transport model for the simulation of large alluvial rivers. It was first released in 1986 (Molinas and Yang, 1986) and was written in a mix of FORTRAN 77 and the older FORTRAN IV programming languages. The code was developed for mainframe computers and used a rigid input/output (I/O) structure based on ASCII files with sequential, fixed-width fields. All data post-processing was accomplished using text editors, spreadsheets, and other third-party programs, a solution that was far from efficient but that represents well the practice at the time.

[1] US Bureau of Reclamation, Sedimentation and River Hydraulics Group (D-8540), P.O. Box 25007, Denver, CO 80225, USA.
[2] Colorado State University, Integrated Decision Support Group, 601 Howes Street, University Services Building, Room 410, Ft. Collins, CO 80523, USA.
[3] US Bureau of Reclamation, Sedimentation and River Hydraulics Group (D-8540), P.O. Box 25007, Denver, CO 80225, USA

More recently, an updated and improved version of the code, GSTARS 2.0 (Yang et al., 1998) was released. This newer release incorporated improvements in the mathematical models and numerical engine, but for the most part left unattended the I/O system, which remained based on sequential, fixed-width record, ASCII files. (A number of basic graphing programs for data post-processing are included in the GSTARS 2.0 distribution package.)

The present effort for the newest release of the code, GSTARS 2.1, is concentrated on the I/O system. Although some changes were made to the numerical engine — i.e., to the FORTRAN code that implements the mathematical model — such as adding side discharges by tributaries and bug fixes, these represent only a minor portion of the overall GSTARS 2.1 code development.

2. The Numerical Engine

GSTARS 2.1 (Yang and Simões, 2000) is a numerical model for the simulation of the in-channel sedimentation processes in large (wide) alluvial rivers. The backwater model is based on an improved version of the one-dimensional algorithm of Molinas and Yang (1986), and can compute mixed flow regimes (subcritical, supercritical, and mixed flows — see Figure 1).

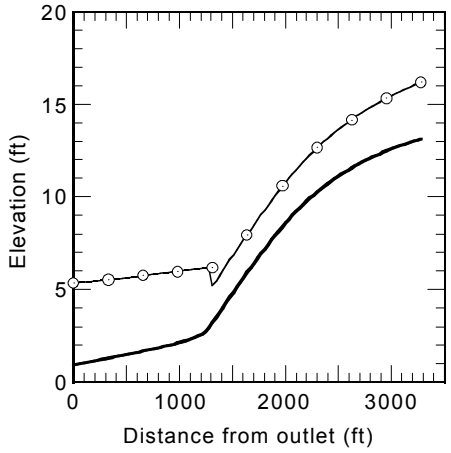

Figure 1. GSTARS 2.1 simulation and analytic solution in a channel with trapezoidal cross section (MacDonald e al., 1997). The flow is subcritical in the first and last thirds of the channel, and supercritical in the middle, with a visible hydraulic jump. The thick line is the channel's bed, the thin line is the analytic free-surface, and the circles denote the computed solutions at the computational cross sections.

GSTARS 2.1 is a quasi-steady model, in which the water discharge hydrograph is approximated by bursts of constant discharge. Sediment routing is decoupled from the backwater computations and can be accomplished with a different time step size. Exener's equation is used to rout the sediments in a semi-two-dimensional manner, based on the stream tube concept. Each cross section is divided in regions of equal conveyance, and sediment is routed through each region (stream tube) independently — see Figure 2. Although each stream tube has the same discharge, in general they will have different cross-sectional areas and, therefore, different average velocity. Carrying capacities, bed sorting, and fractional transport are computed independently in each stream tube, and stream tube boundary locations are recalculated after each time step.

The model incorporates different methods of transport, ranging from clay to silt, sand, and gravel. Non-equilibrium sediment transport and bank stability analysis are some of the optional features included. The sediment continuity equation can be applied to both the bed and the banks of the river channel. Channel width changes are computed based on the theory of total stream power minimization. A more detailed description of the model can be found in Yang and Simões (2000).

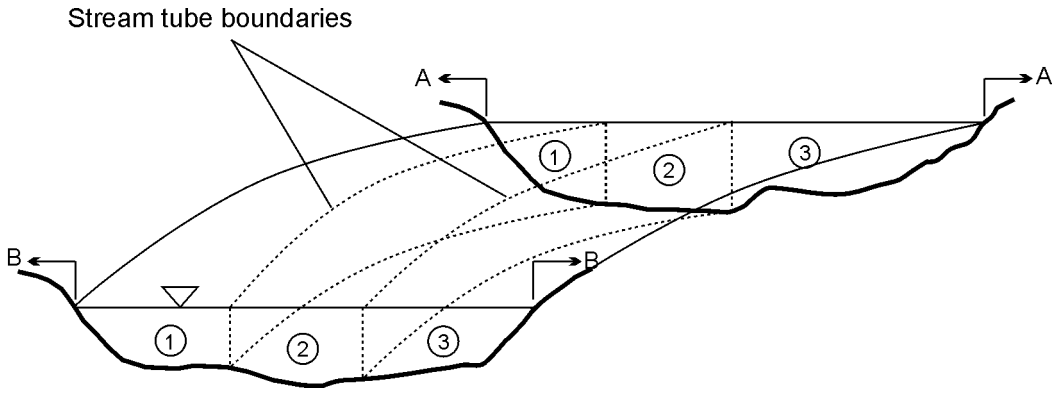

Figure 2. Representation of the use of stream tubes in GSTARS 2.1. The discharge is the same in all stream tubes, i.e., $q_1 = q_2 = q_3$. However, because of different roughness and cross sectional areas, A, the velocity $V = q/A$ will vary along each stream tube and within each cross section. For example, for cross section B, $A_1 \neq A_2 \neq A_3$, therefore $V_1 \neq V_2 \neq V_3$, where the subscripts refer to stream tube number. Similarly, within the same stream tube $A_A \neq A_B$, therefore $V_A \neq V_B$, where now the subscripts refer to the cross section.

3. Graphical User Interface

3.1 Approach

The GSTARS 2.1 numerical engine has a standard command line interface, communicating with the user using static files: the input files are prepared, the program is executed using the information contained in those files and, upon completion and normal termination, a number of output files are produced. A new run of the program involves the preparation of a new input file and the creation of more output files by the program. The approach taken in the development of a new I/O system for the program maintains that basic process, but encapsulates it within a new interface layer. This layer provides the graphical and interactive environment to prepare the input file required for model execution, and processes the output information for graphical display of results.

During execution of the numerical code, some informational messages are printed in the standard output device, such as the status of the run (percentage of computations completed) and any runtime error messages that may occur. Those messages are also passed directly to the graphical user interface (GUI) and displayed in real time. This is accomplished by a basic message-passing mechanism that captures all the intermediary ASCII output produced by the numerical engine (a one-way communication path).

The approach chosen is schematically represented in Figure 3. It is very common in the development of similar types of application. From our point of view, there are two main

advantages to it. First, it allows the existing code to remain untouched. Indeed, in the GSTARS 2.1 release, the user still has the capability of generating the input files using an ASCII text editor end executing the program using the command line, bypassing entirely the GUI. The user retains full access to the input and output data files, which may be useful for importing selective data to/from other applications, such as spreadsheets or CAD programs.

The other important advantage resides in the possibility of reusing the code for other models. Although developed specifically to accommodate the needs of GSTARS 2.1, the main GUI code can be easily modified and expanded for other numerical models. For example, to use the same GUI for the HEC-6 model (USACE, 1993), most of the work would consist in modifying the modules that read and write the ASCII files used to communicate with the numerical model. Of course that, since HEC-6 has different capabilities from GSTARS 2.1, some additional work would be required to include them in the GUI's input screens. Nevertheless, this approach makes it easy to use the same GUI code in future model developments, as well as for existing legacy codes based on standard command line interfaces.

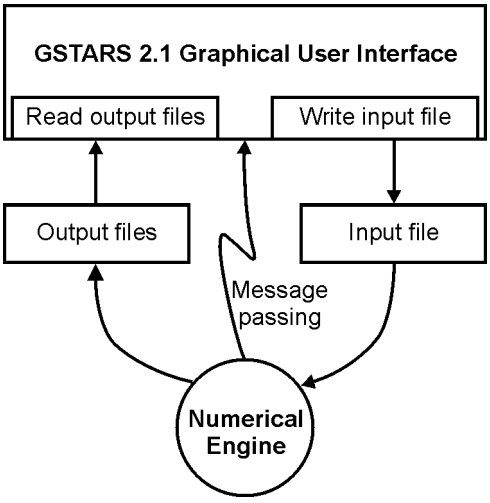

Figure 3. Basic architecture followed in the development of the graphical user interface for GSTARS 2.1.

3.2 Methodology

As described in the previous sections, the GSTARS 2.1 graphical interface is a separate component to the model that functions as pre- and post-processor. Sun's Java (see www.javasoft.com) was the programming language chosen. The primary advantage of using Java is that it has been designed to run on multiple operating systems, including Microsoft Windows (95/98/NT) and Linux. Currently, the interface runs only as an application and not as an applet[†], but after further development there is the possibility of running the interface as an applet in a web browser, which could simplify the distribution of the interface software. Java also has a set of functions for connecting to local or networked databases, which could be another important consideration in future development.

[†] An applet is a program designed to be executed from within another application. Unlike an application, applets cannot be executed directly from the operating system, but a well-designed applet can be invoked from many different applications.

The interface was broken into two modules: the graphical user interface dialogs and the controller. The graphical user interface dialogs, which form the physical part of the GUI, were designed using IBM's Visual Age for Java integrated development environment (see www.ibm.com/software/ad/vajava). Using a graphical design board, buttons and other user interface elements are laid out, allowing for rapid prototyping and development. When the user is happy with the appearance of a component, the compiler can output it as a Java class, complete with hooks to each of the component's text fields and buttons (see Figure 4).

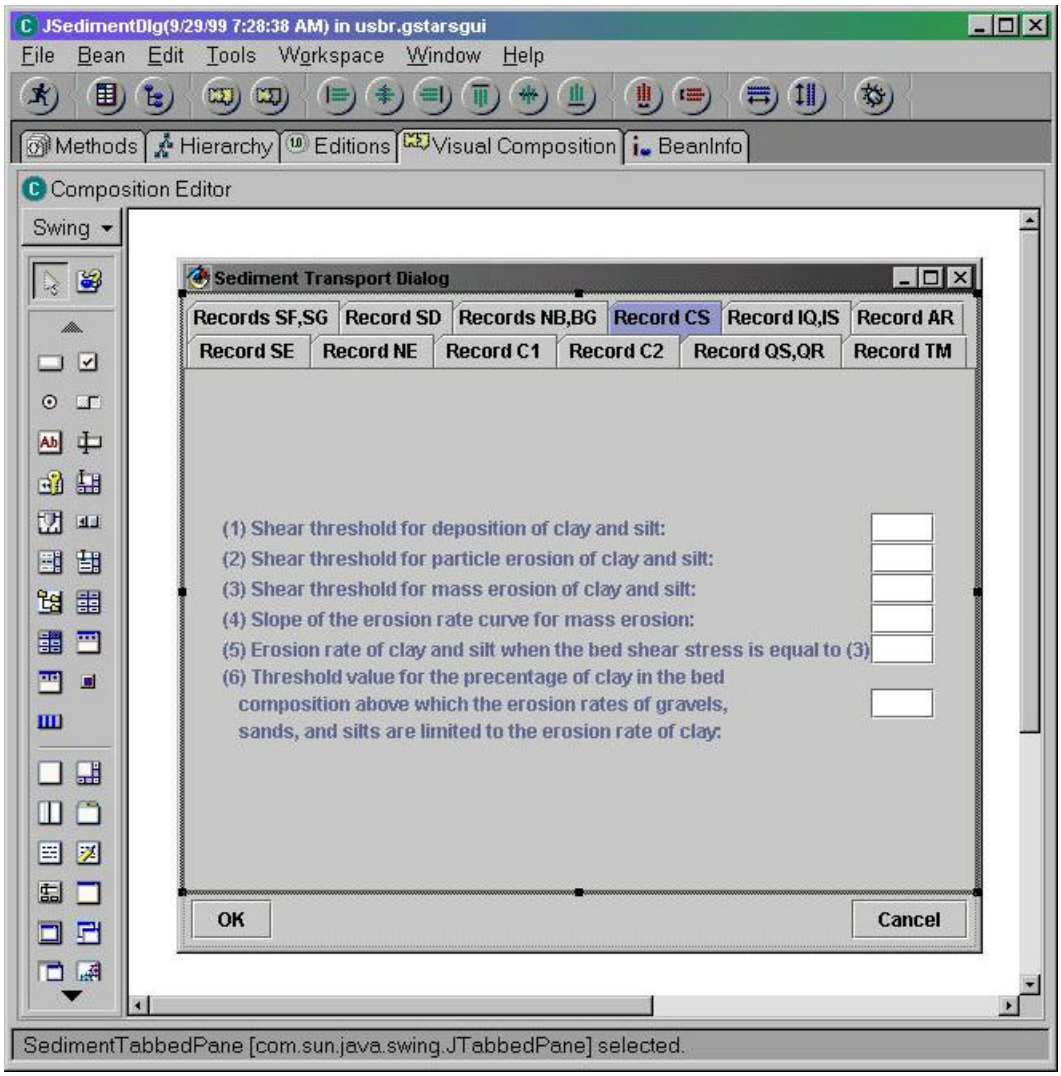

Figure 4. Layout and design of the sediment transport dialog in VisualAge for Java. Using this editor, human-readable Java code that creates and positions the labels, tabs, and text fields of the dialog can be generated.

The controller, which is the code that gives functionality to the GUI dialogs, was written using JPython. JPython is a scripting language written in Java (see www.jpython.com). Where Java is a statically-typed language similar to C++, JPython is an object oriented, dynamically-typed language, and contains native high-level data structures such as lists and dictionaries. This allows for faster development and reduces the number of lines of code, making maintenance easier. Of course, such a system is not without its drawbacks, which include the additional

overhead involved with translating JPython code to the equivalent Java. The distribution of the interface is also more complex because of the JPython libraries that need to be installed. The controller functions by subclassing (extending) the dialog components that were written in Java. These JPython classes provide the implementation of each of the buttons, menus, and other interface elements.

A third-party Java class, PtPlot 3.1, was used to perform the graphing functions of the interface (see Ptolemy.eecs.Berkeley.edu/java/ptplot3.1/Ptolemy/plot/doc). While powerful in its own right, a major consideration was that the source code was available. This allows the programmer to easily make modifications without having to worry about copyright issues or waiting for the needed feature to come along in a future development.

4. Concluding Remarks

At the writing of this paper (May 2000), the GUI was in advanced stage of development and it was running successfully on PC workstations with Windows 98 (see Figure 5) with most of its functionality. The final release will be available to the public with distributions for Windows 95/98 and NT, with a Linux distribution also planned for later.

The GUI successfully read all of the existing data files prepared for GSTARS 2.0 (versions of the model older than 2.0 are no longer supported by the US Bureau of Reclamation). Both the model input and output files can be imported to the GUI, which means that existing data can be analyzed with this tool without the need to rerun the model. This is particularly important for archived project data in the cases for which the model input data files can no longer be found.

Testing of the GUI has been carried out on a Pentium II CPU at 333 MHz (8 Mbytes graphics card) and on a Pentium III at 550 MHz (16 Mbytes graphics card), both with 250 Mbytes of RAM. Although the GUI performed almost equally well on both machines, some performance issues remain to be solved at this time. The most important is the system slow down observed when a background image is used as a template in the main graphics window (the GUI allows the user to work over an image, such as an aerial photograph or a digitized map). However, since this is a cosmetic feature that is not essential for the proper function of the GUI, it will not interfere with the development of the remaining aspects of the interface.

The GUI, GSTARS 2.1 numerical model, manual, and other accompanying files will be placed for free download on the Web. Interested readers should point their browsers to www.usbr.gov/srhg/ and follow the links therein.

References

MacDonald, I., Baines, M.J., Nichols, N.K., and Samuels, P.G., “Analytic Benchmark Solutions for Open-Channel Flows,” *J. Hydr. Engng. ASCE* **123**(11), pp. 1041-1045, 1997.

Molinas, A., and Yang, C.T., “Generalized Water Surface Profile Computation,”, *J. Hydr. Engng. ASCE* **11**(3), pp. 381-397, 1985.

Molinas, A., and Yang, C.T., “Computer Program User’s Manual for GSTARS (Generalized Stream Tube model for Alluvial River Simulation),” US Dept. of the Interior, Bureau of Reclamation, Technical Service Center, Denver, CO, 1986.

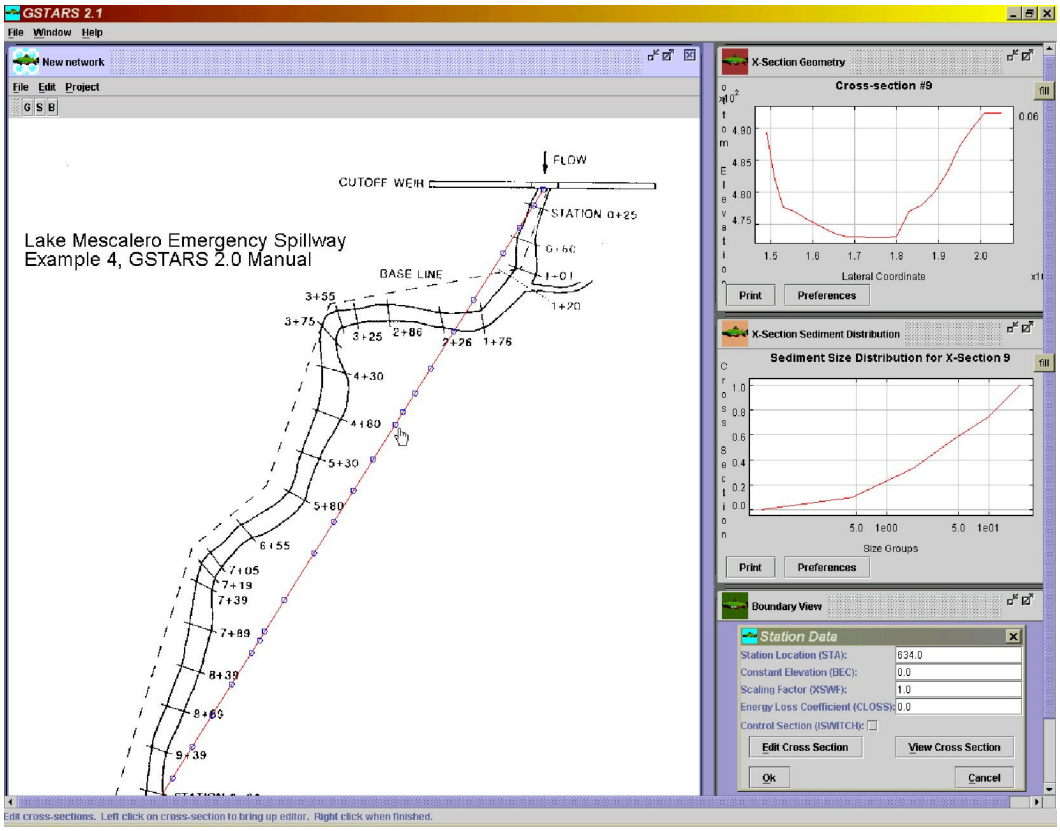

Figure 5. Screen capture of the GSTARS 2.1 users graphical interface showing a river segment being prepared for model run. The panels to the right show the selected cross section geometry, bed particle size gradation, and other cross-sectional parameters used by the model. The numerical grid is shown on the straight line, which is being displayed over a rasterized schematic map of the region being studied.

USACE, "HEC-6 Scour and Deposition in Rivers and Reservoirs," User's Manual, U.S. Army Corps of Engineers, Hydrologic Engineering Center, 1993.

Yang, C.T., and Simões, F.J.M., "User's Manual for GSTARS 2.1 (Generalized Stream Tube model for Alluvial River Simulation version 2.1)," US Dept. of the Interior, Bureau of Reclamation, Technical Service Center, Denver, CO, 2000. (To appear.)

Yang, C.T., Treviño, M.A., and Simões, F.J.M., "User's Manual for GSTARS 2.0 (Generalized Stream Tube model for Alluvial River Simulation version 2.0)," US Dept. of the Interior, Bureau of Reclamation, Technical Service Center, Denver, CO, 1998.