USGS
science for a changing world

Great Lakes Restoration Initiative

# Approaches in Highly Parameterized Inversion: PESTCommander, a Graphical User Interface for File and Run Management Across Networks

Techniques and Methods Book 7, Chap. C8

U.S. Department of the Interior
U.S. Geological Survey

# Approaches in Highly Parameterized Inversion: PESTCommander, a Graphical User Interface for File and Run Management Across Networks

By Marinko Karanovic, Christopher T. Muffels, Matthew J. Tonkin, and Randall J. Hunt

Great Lakes Restoration Initiative

Techniques and Methods, Book 7, Chap. C8

**U.S. Department of the Interior**
**U.S. Geological Survey**

**U.S. Department of the Interior**
KEN SALAZAR, Secretary

**U.S. Geological Survey**
Marcia K. McNutt, Director

U.S. Geological Survey, Reston, Virginia: 2012



Suggested citation:
Karanovic, M., Muffels, C.T., Tonkin, M.J., and Hunt, R.J., 2012, Approaches in highly parameterized inversion—PESTCommander, a graphical user interface for file and run management across networks: U.S. Geological Survey Techniques and Methods, book 7, chap. C8, 9 p.

# Contents



# Figures

# Approaches in Highly Parameterized Inversion: PESTCommander, a Graphical User Interface for File and Run Management Across Networks

By Marinko Karanovic[1], Christopher T. Muffels[1], Matthew J. Tonkin[1], and Randall J. Hunt[2]

## Abstract

Models of environmental systems have become increasingly complex, incorporating increasingly large numbers of parameters in an effort to represent physical processes on a scale approaching that at which they occur in nature. Consequently, the inverse problem of parameter estimation (specifically, model calibration) and subsequent uncertainty analysis have become increasingly computation-intensive endeavors. Fortunately, advances in computing have made computational power equivalent to that of dozens to hundreds of desktop computers accessible through a variety of alternate means: modelers have various possibilities, ranging from traditional Local Area Networks (LANs) to cloud computing. Commonly used parameter estimation software is well suited to take advantage of the availability of such increased computing power.

Unfortunately, logistical issues become increasingly important as an increasing number and variety of computers are brought to bear on the inverse problem. To facilitate efficient access to disparate computer resources, the PESTCommander program documented herein has been developed to provide a Graphical User Interface (GUI) that facilitates the management of model files ("file management") and remote launching and termination of "slave" computers across a distributed network of computers ("run management"). In version 1.0 described here, PESTCommander can access and ascertain resources across traditional Windows LANs: however, the architecture of PESTCommander has been developed with the intent that future releases will be able to access computing resources (1) via trusted domains established in Wide Area Networks (WANs) in multiple remote locations and (2) via heterogeneous networks of Windows- and Unix-based operating systems. The design of PESTCommander also makes it suitable for extension to other computational resources, such as those that are available via cloud computing.

Version 1.0 of PESTCommander was developed primarily to work with the parameter estimation software PEST; the discussion presented in this report focuses on the use of the PESTCommander together with Parallel PEST. However, PESTCommander can be used with a wide variety of programs and models that require management, distribution, and cleanup of files before or after model execution. In addition to its use with the Parallel PEST program suite, discussion is also included in this report regarding the use of PESTCommander with the Global Run Manager GENIE, which was developed simultaneously with PESTCommander.

## Introduction

Although programs such as the parameter estimation code PEST can require many model runs, each individual model run is generally entirely independent of all other runs; therefore, the communication overhead needed to execute the model runs in parallel using a large number of computers is theoretically very small. This type of parallel computing problem is referred to as "embarrassingly parallel" (Foster, 1995). By using a master-slave protocol such as is implemented in the Parallel PEST (PPEST) suite of programs (Doherty, 2010a; 2010b), such embarrassingly parallel problems are conceptually trivial to manage: the user distributes the necessary model files to the slave computers, and then a communication protocol is used to execute the necessary model runs on the slave computers and to gather results at the master computer. PPEST, BeoPEST (Schreüder, 2009), and PEST++ (Welter and others, 2012) employ this type of strategy in undertaking the embarrassingly parallel problems inherent in parameter estimation.

A practical difficulty that arises, however, is one of logistics. Before the embarrassingly parallel problem can be executed, one must organize and distribute all required files to each slave computer and have the ability to invoke and cancel a run. Furthermore, because the interim results on a slave are no longer needed once they have been harvested by the master, cleaning up the slave computers after completion of model

[1] S.S. Papadopulos & Assoc., Inc.

[2] U.S. Geological Survey.

runs also is desirable. As a practical matter, overfilling non-volatile memory (for example, a computer's hard drive) can be problematic because computational efficiency will degrade as a result of carrying the burden of expendable files from model calibration efforts such as the Great Lakes Restoration Initiative, whereby Great Lake watershed-scale models are calibrated, climate and land-use scenarios are simulated, and uncertainty analyses are performed.

PESTCommander was developed to facilitate the distribution and cleanup of model calibration files employed when using PPEST, BeoPEST, and/or PEST++ via a user-friendly Graphical User Interface (GUI). In version 1.0, the PESTCommander GUI provides the modeler (1) an intuitive interface to access the available networked computer resources; (2) automated capabilities to create folders and to copy, reconcile, and clean up files located on identified slave computers; and (3) a means of launching and terminating slave computers.

## Purpose and Scope

The purpose of PESTCommander is twofold. First, PESTCommander enables users to rapidly set up a parallel parameter estimation simulation using a combination of networked (and potentially, cloud-based) slave computers. The GUI provides the user an interface to the available local networked computer resources and automated capabilities to create folders and to copy, reconcile, and clean up files located on identified slave computers. This capability is fully implemented within version 1.0 of the PESTCommander program. Secondly, PESTCommander allows users to efficiently deploy and terminate slave computers to facilitate their management. Because PESTCommander is designed to distribute files for parallel model runs over a local network by using the model independence approach of PPEST, all parameter-estimation-related terminology and concepts are based on conventions presented and/or cited by Doherty (2010a) and Doherty and Hunt (2010), which, for brevity, are therefore not repeated here.

The PESTCommander interface was designed such that potential users with limited interest in the underlying methods and codes may focus on the directions for use with the examples presented for PPEST. However, a second, higher level purpose of this report is to provide the programming background for PESTCommander. This in-depth discussion is intended to convey the more advanced concepts of program design to the reader to facilitate the integration of code developed by others.

This report is structured as follows: First, we describe the capabilities of the underlying modules that form the basis of PESTCommander. Next, we describe the structure of PESTCommander, introducing the concepts of the FileTree, FileTable, Network Manager, and Remote Control modules. This discussion is followed by an outline of the specific steps taken when using PESTCommander to facilitate a parallelized model analysis. Finally, we describe the current level of compatibility of PESTCommander with the global parallel run manager GENIE (Muffels and others, 2012) and with execution of parallel simulations in a cloud computing environment (Fienen and others, 2011; Carter and others, 2011); we expect that the degree of compatibility with both GENIE and cloud computing will be enhanced in later releases of PESTCommander.

## Design Concepts

PESTCommander is developed in the object-oriented language Python. As with many other object-oriented languages, the objects provided in Python are an efficient and extensible means of organizing and designing an intuitive GUI. Python includes an extensive suite of modules, classes, exceptions, and other structures that can be readily used as building blocks for sophisticated programs, and new built-in modules are easily developed and integrated by using other programming languages such as C++.

Specifically, PESTCommander was developed by using the Python[3] PyQt framework. PyQt is a set of open source Python bindings for Nokia's Qt[4] application framework (*http://qt.nokia.com/products/*), which runs on all platforms that are supported by Qt, including Microsoft Windows[5], Mac OS[6] X, and Linux[7]. Qt is implemented in C++ and is fully object oriented—it includes more than 600 classes, all with widely applicable defaults and useful out-of-the-box functionality and all of which are able to be customized and subclassed to meet programmer's development-specific requirements. PyQt was selected as the framework for the development of PESTCommander because it brings together the Qt C++ cross-platform application framework with the cross-platform interpreted language Python, combining the benefits of both languages and platforms.

PESTCommander was developed by incorporating two main PyQt modules: first, QtCore for file management; and second, QtGui for GUI development. Specifically—

1. The QtCore module contains the core non-GUI classes, including the event loop and the Qt signal and slot mechanism necessary to execute PESTCommander functionality. QtCore also includes platform-independent abstractions for Unicode, threads, mapped files, shared memory, regular expressions, and user and application settings.

[3] "Python" is a registered trademark of the Python Software Foundation

[4] Qt® is a registered trademark of Nokia Corporation and/or its subsidiaries.

[5] Windows® and Windows Azure™ are registered trademarks of the Microsoft group of companies.

[6] Mac OS® is a registered trademark of Apple Inc.

[7] Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

2. The QtGui module contains the majority of the GUI classes. These include a number of table, tree, and list classes based on the model-view-controller design pattern. Also provided is a sophisticated 2D canvas widget capable of storing thousands of items, including ordinary GUI widgets.

## Using PESTCommander

The PESTCommander user interface is developed on a modular form/panel structure. This structure allows users to rearrange, resize, detach, hide, and show each module/panel. Also, this modular structure opens the possibility of enhancing the user interface in future versions by creating new modules which can be easily incorporated into existing interface. This version 1.0 release of the PESTCommander interface contains four main modules and is illustrated by using Windows XP naming conventions in figure 1. The four modules are the following:

1. *FileTree.*—Used for easy and fast local browsing of folders and drives. Selecting a folder in this module will expand the folder content in the FileTable module. Once selected, folder sorting capabilities are enabled by selecting appropriate column headers in the FileTable module.

2. *FileTable.*—Used for folder and file selection. Simultaneous selection of multiple instances of files (or folders) can be accomplished by holding down the control key then selecting the desired files (or folders). A range of files (or folders) can be selected by holding down the shift key, then selecting the first file in the range, and—with the shift key still depressed—scrolling down and selecting the last file in the range.

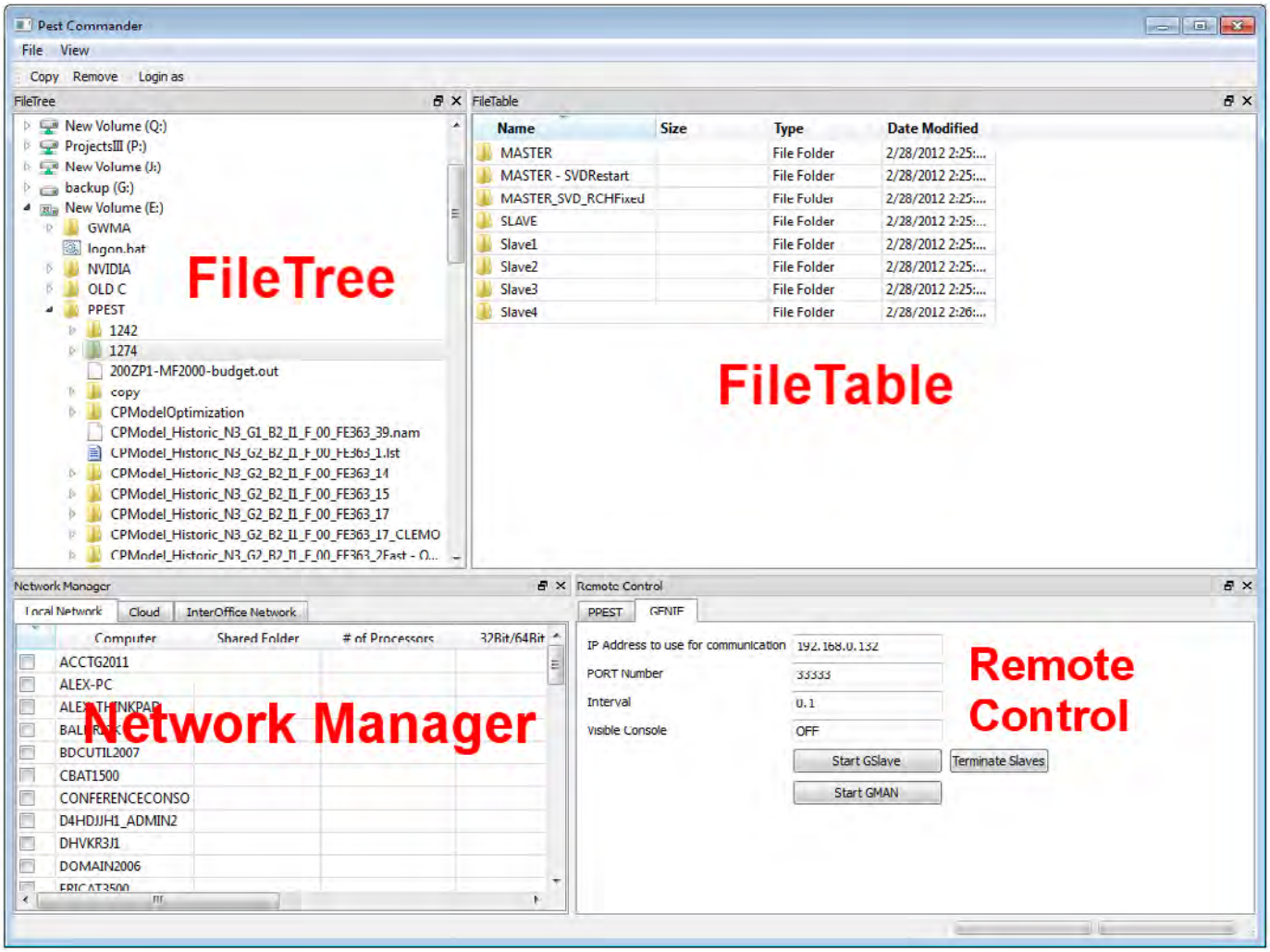


**Figure 1.** PESTCommander Graphical User Interface. The IP Address in the Remote Control section is an example non-routable address and used for illustration only.

3. *Network Manager*.—Used for slave computer management and selection. With this module, users can select/deselect network computers, browse for shared folders, and assign the number of slaves per computer to be used for parallel processing, based upon the resources reported as available for each computer.

4. *Remote Control*—Used for remote launching of slave computers. When the "PPEST" tab is selected, the user can remotely start and terminate PPEST slaves without using external programs such as PSEXEC or PSKILL that have been commonly used for this purpose. When the "GENIE" tab is selected, the user can remotely start and terminate the GSLAVE program on selected remote computers and start the GMAN application on the host computer The reader is referred to Muffels and others (2012) for further details on GENIE.

PESTCommander enables users to (1) rapidly set up a combination of networked slave computers, (2) distribute necessary model files to slave computers, (3) delete unnecessary files after model runs are completed, and (4) start or terminate slaves on remote computers. To successfully access all resources identified by PESTCommander, the user must have administrative permissions associated with their user ID on all slave computers. This can be accomplished one of two ways: first, the user account can be globally assigned these permissions for all occasions; alternatively, the user can temporarily enter the User Name and Password of the user that possesses administrative privileges by selecting the "Login as" button.

## Slave Computers Selection Procedures

When PESTCommander is started for the first time, the GUI will open in a default viewing mode: the FileTree module will show a list of local and network drives, the FileTable will not show any selected files, and the Network Manager module will show only a list of available network computers on the current Local Area Network (LAN) domain. A mouse right-click on the desired network computer name will select a slave computer, which will then open the folder browser (fig. 2), displaying the available (that is, shared) folders on the slave computer. The user can then select the desired shared folder to use on the slave computer.

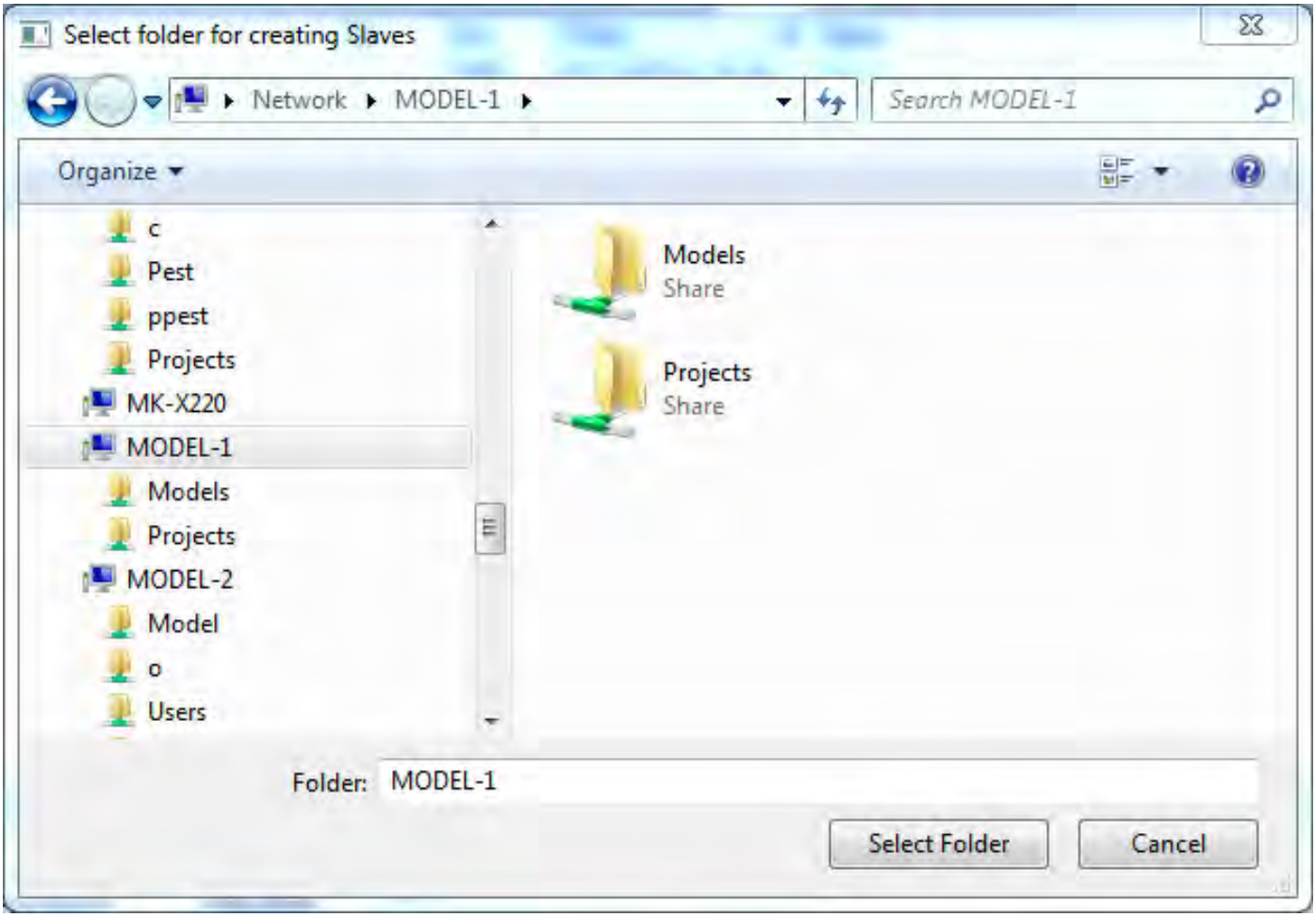


**Figure 2.** Browsing for shared folders in a Local Area Network.

When a slave computer is selected for the first time, PESTCommander will read and store information about the resources that are available on that computer: this information includes the total number of processors (that is, cores), operating system type (32-/64-bit), and maximum CPU clock speed. In addition, PESTCommander will automatically assign the number of individual slave instances (determined as the total number of cores minus 1) that can potentially be used for parameter estimation on that slave computer. This automatic configuration can be overridden manually by entering a new number of individual slave instances to be used on that computer. The user can select or deselect the slave computer by means of a check box in the first column of the local network table in Network Manager (fig. 1).

PESTCommander can save a specific slave setup (configuration) between PESTCommander sessions, which avoids having to select slaves every time the GUI is opened if a certain configuration is the most common. A session is retained when a PEST Run Management File, or RMF (**.rmf*) (Doherty, 2010a), is created and saved by selecting "File" followed by "Save." A saved RMF can be opened by selecting "File" followed by "Open." PESTCommander will read the computer names from the RMF, compare them to the list of computers available within the LAN, and highlight which resources are available in the box next to the computer name. PESTCommander will also calculate how many slaves were specified for each computer, tally the current total available, and populate a "# of Slaves" column in Network Manager.

## File Management Procedures

The following actions are available in the PESTCommander GUI to help manage and distribute files needed for parameter estimation:

1. Create directories on the slave computers for running PPEST (or GENIE).
2. Copy files from the master to the slaves.
3. Clean up (delete) files from the slave computers.
4. Remove model directories from the slave computers.

Before these actions can be invoked, the user must first select the drive (folder) where the model files are located with

Local Network

Local Network | Cloud | InterOffice Network

| | Computer | Shared Folder | # of Processors | 32Bit/64Bit | Max Clock Speed | # of Slaves |
|---|---|---|---|---|---|---|
| ☐ | MW | | | | | |
| ☐ | OPT755D9N0HWH1 | | | | | |
| ☐ | QUADCORE_1 | | | | | |
| ☐ | SLAVE1 | | | | | |
| ☐ | SLAVE3 | | | | | |
| ☑ | SLAVE5 | //Slave5/ppest | 16 | 64 | 2261 | 15 |
| ☐ | SLAVE6 | | | | | |
| ☑ | SLAVE7 | //Slave7/ppest | 8 | 64 | 3201 | 7 |
| ☐ | TERMINAL2004 | | | | | |
| ☐ | STORE2004 | | | | | |
| ☐ | T3400 | | | | | |
| ☐ | TERMINAL2000 | | | | | |
| ☐ | TERMINAL2006 | | | | | |
| ☐ | TPI7 | | | | | |
| ☐ | VOSTRO-400 | | | | | |
| ☐ | WORKGROUP2003 | | | | | |

**Figure 3.** Slave information list showing two Windows XP slaves selected.

the FileTree module, then select files and/or folders with the FileTable module. File/folder selection is by use of a mouse left-click. Multiple selections can be achieved by holding the control key (nonsequential selections) or the shift key (sequential selections).

When the "Copy" button is selected, PESTCommander will check whether an RMF is currently saved and open; if not, the user will be prompted to save one. The RMF is important to save because the RMF root filename (that is, the filename without the file extension) will be used to create a project subfolder under the local computer shared folder, which in turn will be used to loop through each slave for each selected computer and generate slave subfolder(s) following the naming convention "Slave[1 to *n*]" where *n* is the *n*th slave on slave computer. For example, if the shared folder is on a Windows XP operating system with the name "\\Slave7_PC\PPest\New Folder\", an RMF filename *ModelRun1.rmf* will generate the following new folders if the user has selected two slaves to run on that computer:

\\Slave7_PC\PPest\New Folder\ ModelRun1\Slave1\
and
\\Slave7_PC \PPest\New Folder\ ModelRun1\Slave2\

All selected files/folders will be copied into these two folders; before copying begins, however, the user will be warned that this action may overwrite already existing files and folders.

When the "Remove" button is selected, PESTCommander will loop through all selected computers and select each shared folder and project folder name that matches the active RMF filename. When a matching folder is found, PESTCommander will loop and search for folder(s) with name "Slave[1 to *n*]" where *n* is the *n*th slave on slave computer. From this list, PESTCommander will remove that folder as well as all files/folders inside. If no additional files/folder exists in the project folder, the project folder will be removed too. **Because this is a network delete command, any removed files will not be saved in the local computer recycle bin and will not be retrievable once deleted.**

## Remote Launching Procedures

The Remote Control module allows users to start and terminate slave computers across the LAN. In the current release of PESTCommander (version 1.0), this capability is fully implemented for PPEST use and partially implemented for use with GENIE.

*PPEST Interface.*—To initiate slave computers for calibration runs using PPEST, the user should select the "PPEST" tab and then enter the name of the batch file used by PEST to run the model in the "Model Command Line" text box. If the "Start Slaves" button is pressed, the GUI will loop through all selected network computers and initiate the designated number of slaves. Similarly, pressing the "Terminate Slaves" button will terminate the initiated slaves. Note that pressing the "Terminate Slaves" button will not terminate existing model runs—it will terminate only the PSLAVE program used by PPEST to run models on remote computers (fig. 4). It is theoretically possible to terminate the model runs if their process ID is determined at the moment they are executed; however, this functionality is not currently implemented in the first release of PESTCommander.

*GENIE Interface.*—Currently (version 1.0), PESTCommander is partially integrated with GENIE. PESTCommander provides a list of available slave computers and their shared folders, allows remote launching and termination of GSLAVE programs on selected slaves, and allows the launch of the GMAN application on the host computer. Before starting slaves, the user needs to enter the necessary GENIE communication parameters: these are the IP address used for communication between GLSLAVE and GMAN, port number, and interval, all of which are described by Muffels and others (2012). By default, the PESTCommander GUI will use the IP address of the computer where the GUI is started, although the user can use any active IP address in the LAN (fig. 5).

# Future Directions

## Operability with Wide Area Networks

The functionality of the PESTCommander version (1.0) described in this report is fully compatible with, documented for, and tested with local Windows networks and remote computers that can be accessed via TCP/IP communications. The current PESTCommander design also provides a suitable framework for distributing files for executing model simulations across remote Wide Area Networks (WANs) via trusted domains. The following additional capabilities, however, are needed to provide seamless file distribution and run execution across remote WAN environments:

- Procedures for robust and secure firewall connections to allow access to remote (nontrusted domain) networks.
- Procedures to distribute files using TCP/IP protocols.
- Procedures to securely start and stop slaves on remote computers that can operate (when approved) despite and without compromising firewalls—for example, via secure port forwarding.

## Integration with GENIE and Cloud Computing

Version 1.0 of PESTCommander provides a flexible framework for distributing, reconciling, and cleaning up files from slave computers on locally accessed network resources; however, PESTCommander does not seamlessly operate with the run manager GENIE. This section describes additional capabilities that are needed to seamlessly integrate PESTCommander with the general run TCP/IP run manager GENIE (Muffels and others, 2012) and/or with cloud computing resources.

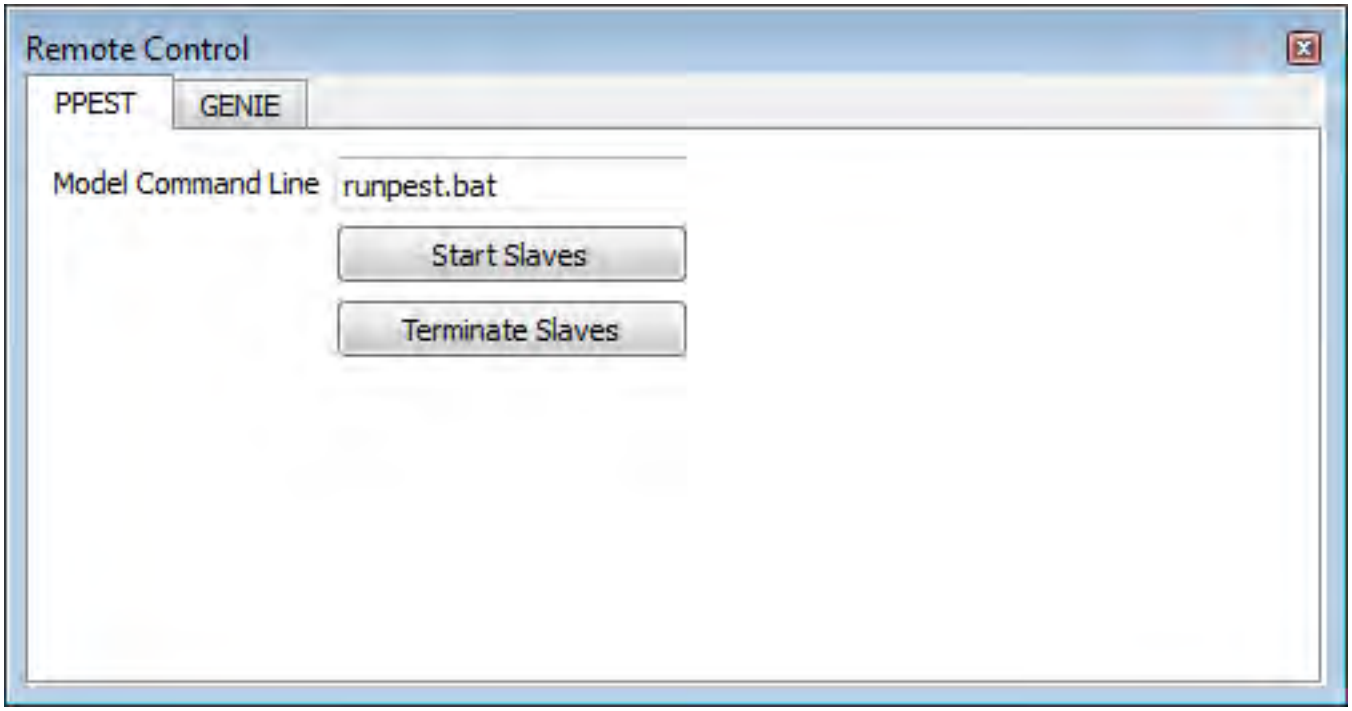


**Figure 4.** PPEST Remote Control module.

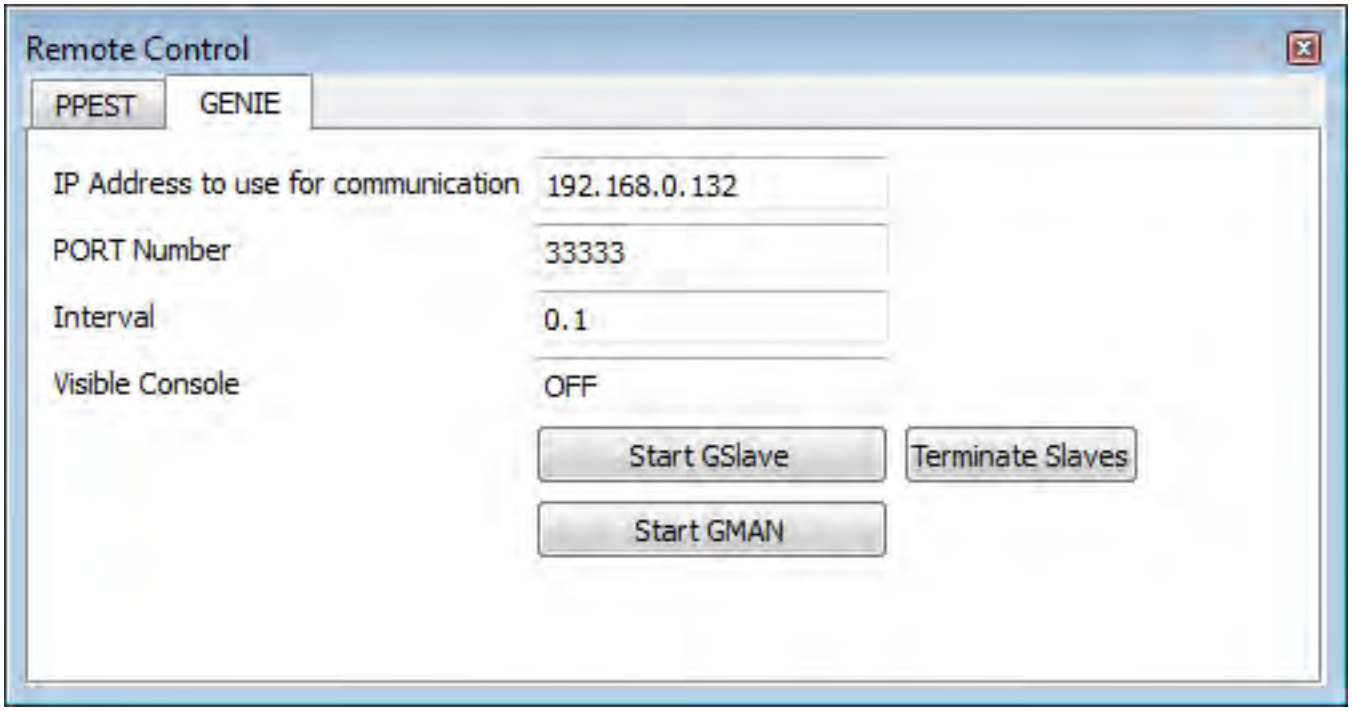


**Figure 5.** GENIE Remote Control module. The IP Address shown is an example non-routable address.

### GENIE Interface—Remote Execution Procedures

GENIE is a general parallel run manager that communicates between computers via TCP/IP; thus, conceivably any computing resource connected to the Internet is a possible computing node within a distributed computing framework. With GENIE, the user can start the master run manager (GMAN), remotely initiate slaves (GSLAVE), and monitor parallel run progress. Currently, PESTCommander is not fully integrated with GENIE because it does not monitor parallel run progress, but it provides a list of slaves and shared folders, initiates and terminates GLSLAVE on remote computers, and starts GMAN on the host computer. The following additional capabilities, however, would be required to enable PESTCommander to seamlessly monitor and restart model runs by using GENIE:

- Procedures to monitor parallel run progress.
- Procedures to terminate and restart (master) parallel processes.

### Cloud Computing Capabilities

Hunt and others (2010) present an overview of a cloud computing paradigm for the embarrassingly parallel parameter estimation problem and state that one current limitation to this approach is the lack of utilities that allow automatic and seamless access to computing resources hosted on the cloud. Recently, Fienen and others (2011) documented cloudPEST, a Python module with functions to facilitate deployment, launching, and termination of BeoPEST nodes on the Amazon Elastic Compute Cloud (Amazon EC2; *http://aws.amazon.com/ec2*). Given that cloudPEST is a set of specific command-line based procedures, it is not considered accessible to many users, especially those most comfortable with GUIs. Because PESTCommander is a GUI-based environment, future efforts will evaluate the efficacy of developing PESTCommander cloud modules. One potential complexity is that, in addition to Amazon EC2 cloud computing, other potential cloud computing vendors currently are available (2012), including GoGrid (*http://www.gogrid.com*), Rackspace[8] (*www.rackspace.com/cloud*), and Windows Azure (*http://www.microsoft.com/windowsazure*).

Currently, PESTCommander is developed in Python, which is well suited for cloud module development; however, the following capabilities would be required to enable PESTCommander to duplicate the capability currently available for LANs to cloud computing:

- Procedures to initiate and configure cloud instances.
- Procedures for file distribution on cloud machines.
- Procedures to manage cloud sessions.
- Procedures to provide real-time monitoring cloud runs to ensure all cloud sessions are terminated as soon as appropriate and that the metered cost of accessing the cloud is minimized.

## Limitations of Version 1.0 of PESTCommander

Principal limitations of version 1.0 of PESTCommander are the following:

- Only Windows local networking (LAN) protocols are utilized by PESTCommander; additional work is needed to enable heterogeneous Windows- and UNIX-operating-system based networks.
- No TCP/IP file distribution capability is currently available.
- Users must have administrator rights on all computing resources.

Because an objective of PESTCommander was to make it accessible to many applications, the overarching development philosophy was thus to provide an extensible design that is suitable for the addition of new capabilities and features. As a result, the addition of appropriate capabilities could readily address the aforementioned limitations.

Although this program has been used by the U.S. Geological Survey (USGS), no warranty, expressed or implied, is made by the USGS or the U.S. Government as to the accuracy and functioning of the program and related program material nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith.

[8] Rackspace® is a registered service mark of Rackspace US, Inc. in the United States and/or other countries.

# References


Carter, J.T.V., Fienen, M.N., and Dahlstrom, D.J., 2011, Automated launching of remote processors for highly parameterized inverse modeling on a heterogeneous system of Cloud-based and local computers, *in* MODFLOW and More 2011, Integrated Hydrologic Modeling, International Ground Water Modeling Center, Colorado School of Mines, Golden, Colo., June 6–8, 2011: p. 637–641.

Doherty, J., 2010a, PEST, Model independent parameter estimation—User manual (5th ed., with slight additions): Brisbane, Australia, Watermark Numerical Computing, 336 p.

Doherty, J., 2010b, Addendum to the PEST manual: Brisbane, Australia, Watermark Numerical Computing, 247 p.

Doherty, J.E., and Hunt, R.J., 2010, Approaches to highly parameterized inversion—A guide to using PEST for groundwater-model calibration: U.S. Geological Survey Scientific Investigations Report 2010–5169, 59 p. (Also available at *http://pubs.usgs.gov/sir/2010/5169/*.)

Fienen, M.N., Kunicki, T.C., and Kester, D.E., 2011, Cloud-PEST—A Python module for cloud-computing deployment of PEST, a program for parameter estimation: U.S. Geological Survey Open-File Report 2011–1062, 22 p. (Also available at *http://pubs.usgs.gov/of/2011/1062*.)

Foster, I., 1995, Designing and building parallel programs: Boston, Mass., Addison-Wesley Pearson Education, ISBN 9780201575941, 430 p.

Hunt, R.J., Luchette, J., Schreuder, W.A., Rumbaugh, J.O., Doherty, J., Tonkin, M.J., and Rumbaugh, D.B., 2010, Using a cloud to replenish parched groundwater modeling efforts: Ground Water, v. 48, no. 3, p. 360–365, *doi:10.1111/j.1745-6584.2010.00699.x*.

Muffels, C.T., Schreüder, W.A., Doherty, J.E., Karanovic, M., Tonkin, M.J., Hunt, R.J., and Welter, D.E., 2012, Approaches in highly parameterized inversion—GENIE, A general model-independent TCP/IP run manager: U.S. Geological Survey Techniques and Methods book 7, chap. C6, 26 p. (Also available at *http://pubs.usgs.gov/tm/tm7c6/*.)

Schreüder, W.A., 2009, Running BeoPEST, *in* Tonkin, M.J., ed.. Proceedings, PEST Conference 2009, Potomac, Md., November 1–3, 2009: Bethesda, Md., S.S. Papadopulos and Associates, p. 228–240.

Welter, D.E., Doherty, J.E., Hunt, R.J., Muffels, C.T., Tonkin, M.J., and Schreüder, W.A., 2012, Approaches in highly parameterized inversion—PEST++, a Parameter ESTimation code optimized for large models: U.S. Geological Survey Techniques and Methods book 7, chap. C5, 47 p. (Also available at *http://pubs.usgs.gov/tm/tm7c5/*.)