

# Interfacing PC-based MATLAB Directly to HPC Resources

John Nehrbass, Siddharth Samsi, Juan Carlos Chaves, Jose Unpingco, Brian Guilfoos, Stanley Ahalt,  
Ashok Krishnamurthy, Alan Chalker, and Judy Gardiner  
*Ohio Supercomputer Center, Columbus, OH*  
{nehrbass, samsi, jchaves, unpingco, guilfoos, ahalt, ashok, alanc, judithg}@osc.edu

## Abstract

*Many DoD HPC users, particularly in the SIP area, run codes developed with MATLAB and related applications (MatlabMPI, StarP, pMatlab, etc.). There is a desire to run codes from a desktop instance of MATLAB and connect to and interact with codes running on HPC resources. The PET SIP team has developed and demonstrated technology that makes this possible.*

*The SSH toolbox for MATLAB enables users to connect to and use HPC resources using SSH without leaving the MATLAB environment. The toolbox uses a freely available implementation of SSH, a modified version of which is also used by the DoD HPCMP. The SSH toolbox consists of a Windows DLL written in C, which is used by MATLAB to communicate with the SSH client. The toolbox provides simple MATLAB commands for users to connect to remote resources, run code, retrieve results and end the SSH session. The complexity of the DLL interface and most of the security needs are hidden from the user, making this a very easy to use and powerful toolbox. Since the main component of the toolbox is written in C and packaged as a DLL, the toolbox can also be extended to work with other programming languages such as Java, Python and Octave. MATLAB-style documentation for the toolbox also makes it easy to obtain help on various aspects of the toolbox and a GUI-based installer makes distribution easier.*

*This technology provides a revolutionary way of providing support to the DoD. Software developers are now able to provide all the hooks to a complicated HPC environment, thus removing the burden of end users.*

## 1. Introduction

### 1.1. MATLAB Overview

MATLAB is a high level programming language used widely in the scientific and engineering communities

for research and development. MATLAB has been identified by the DARPA High Productivity Computer Systems Program (<http://www.highproductivity.org>) as one of the leading high productivity languages and has many powerful capabilities that allow users to manipulate and visualize data. Specialized toolboxes provide additional functionalities that range over 80 areas of computational science. Additionally, MATLAB provides the capability for users to interface with external libraries. In this paper, we will illustrate how this capability may be exploited and interfaced with code written in C.

The ability of MATLAB to use functions from external libraries makes it possible to extend its functionality by writing specialized libraries to perform tasks which may not be done easily using MATLAB alone. In the implementation of the SSH Toolbox, a Windows-based DLL written in C was used to provide MATLAB with the ability to interact with a freely available SSH client.

### 1.2. SSH Overview

Secure Shell (SSH) is the most widely used protocol/tool for connecting to remote High Performance Computing (HPC) resources. There are a number of free as well as proprietary implementations of SSH available. In the development of the SSH Toolbox described in this paper, the PuTTY SSH client was used. PuTTY is a free SSH client that works on Windows as well as Unix/Linux. PuTTY works with ordinary username/password based authentication mechanisms but can be configured to work with other authentication mechanisms such as Kerberos.

## 2. Methodology

### 2.1. Overview and Implementation of SSH Toolbox

As shown in Figure 1, the SSH Toolbox enables a user to interact with and control a SSH session from the MATLAB command window. This process involves starting the PuTTY SSH client, setting up a new session and finally connecting to the remote host. Once a connection has been established, the user can now send commands to the SSH window from within MATLAB. Any output from the command executed on the remote host can be retrieved by the user using commands provided with the Toolbox.

The SSH Toolbox has two distinct paths of communication which make up the major parts of the toolbox:

1. MATLAB–SSH communication: This communication path enables the sending of commands from MATLAB to the SSH client window. This is facilitated by the use of a custom DLL which was written specifically to receive data from MATLAB and pass it on to the SSH client using the appropriate Windows API calls.
2. SSH–MATLAB communication: This communication path is used to retrieve any messages/output written to the standard output/error streams on the remote host. These messages can be system generated or generated from the output of the commands that the user sends to the remote host via the SSH client. This communication is buffered through the use of log files that are created by the PuTTY SSH client. In our implementation, the output information is not actively sent to the MATLAB command window by the SSH client; rather, the data is read by MATLAB when the user specifically requests it.

### 2.2. MATLAB – SSH Communication Interface

The MATLAB – SSH communication interface is built using a Windows DLL which is responsible for controlling the SSH sessions based on the instructions sent by MATLAB, as shown in Figure 2.

#### 2.2.1. Windows Programming Overview and Implementation

The SSH Toolbox was developed primarily for the Windows operating system. This section discusses in brief, the Windows programming methodology used for

writing the DLL interface between MATLAB and the SSH client.

Windows GUI programs are different from console based programs in the important aspect that a Windows GUI program does not make explicit calls to obtain input from the user. Instead, input from the user is provided to the application by the operating system via a message queue. Windows GUI applications are event driven and they process input based on the messages obtained from the message queue. The Windows platform supports a number of messages used to specify the actions taken by an application. For example, the WM\_CHAR message is used to send a single character to the target application. By sending a series of WM\_CHAR messages, the custom DLL written for the SSH Toolbox may send user names, remote sever name/IP address, and other relevant data to the SSH configuration utility.

The Windows API provides the *SendMessage* or *PostMessage* function to send a message to a target application's window(s). The *SendMessage* function is a blocking function and does not return until the target window has processed the message. On the other hand the *PostMessage* function returns immediately and does not wait for the target window to process the message that was sent. Either of these two functions may be used for communication between applications, but their use may be dictated by the requirements of an application. For example, in the SSH Toolbox, it is necessary to set a hostname for the remote server before attempting to connect. To ensure that the order is maintained, the *SendMessage* function is used to first set the hostname. This ensures that the DLL sending the message cannot ask the SSH client to connect until the hostname has been received by the SSH client.

Windows messages/events are generated by the operating system itself, but they can also be generated by other Windows applications. The Windows programming API provides functions to generate these messages, which can be used to simulate keyboard and/or mouse events. This provides a powerful mechanism for applications to interact with other Windows programs. By sending the appropriate Windows message, an application could emulate an actual user. However, in order to control an external application in this manner, the following points are important:

1. The controlling application must have a handle to the external application window.
2. The controlling application must be have information regarding which messages are handled by the application to be controlled.
3. IDs' for the controls (menus, text boxes, etc.) in the child application must be known to the controlling application. This enables the controlling application to set the text in edit boxes, push buttons on the GUI, check radio

boxes, etc., as needed. These IDs may be obtained using various commercial and freely available tools.

4. It must be ensured that the user does not close the application being controlled programmatically or perform tasks that change the target application GUI while it is running.

Using the above methodology, the SSH Toolbox implements communication from MATLAB to the SSH client. The dll *sshinterface.dll* is responsible for

1. Launching the PuTTY executable.
2. Setting up a new SSH session which includes setting username, remote host and log file.
3. Starting the session as configured in step 2.

These functions are performed using the appropriate functions in the Windows messaging API.

### 2.3. SSH – MATLAB Communication Interface

The second important part of the SSH Toolbox is the ability to retrieve the output of the commands that are sent to the remote host. In this case, the output of interest comprises of any messages written to the standard output stream and error stream on the remote host. The retrieval of user files and/or data is addressed in a later section.

The SSH Toolbox relies on the PuTTY SSH client and its ability to enable session logging in order to retrieve the output from the commands executed on the remote host. The PuTTY client allows users to control session logging through its configuration interface. The log file to be used can also be specified by the user before starting the new session.

When a user opens a new SSH session from MATLAB, the SSH Toolbox configures session logging for that particular session. A unique file name is generated for the log and the various options are selected as shown in Figure 3. The name and full path of this log file is made available to MATLAB through the *sshinterface.dll*.

Once a new session has been established using the *sshconnect* function, the following additional tasks are also performed in the *sshconnect* function:

1. Obtain the name and full path to the actual log file being created by the PuTTY client.
2. Create a temporary file which is used exclusively by MATLAB.
3. Copy the contents of the PuTTY log file to the temporary file.

When the user calls the *sshstdout* command to retrieve the output from a Unix command, the *sshstdout* function opens both the PuTTY log file and a second temporary file which is a copy of the PuTTY log file before the Unix command was issued. The contents of the two files are compared and any new information from the actual log file is displayed in MATLAB. The

*sshstdout* function then copies the actual log file to the temporary log file so that the contents of the two are the same. This process is repeated each time the user retrieves the output from a command run on the remote host.

### 2.4. Hooking into 3rd Party Technologies

The SSH Toolbox currently works with the PuTTY open source SSH client. However, due to the manner in which the toolbox is implemented it can be used seamlessly with the modified PuTTY client distributed by the HPCMPO which uses Kerberos for authentication. The open source version of the PuTTY SSH client is capable of simple username/password based authentication as well as kerberos based authentication. This flexibility allows the SSH Toolbox to meet the needs of a wider user community.

### 2.5. Install Wizard

The SSH toolbox comes with a MATLAB based installer. The installer performs the following tasks:

- Determine full path to the SSH client executable.
- Determine the version of the SSH client being used.
- Add the SSH Toolbox to the MATLAB Toolbox menu.
- Configure the SSH Toolbox to use the appropriate SSH client.
- Configure the SSH Toolbox to use the SSH Agent if needed.

The current version of the toolbox installer has been implemented as a MATLAB based GUI. The installer GUI also has help comments to assist users through the installation. The installer is a simple interface that takes the user through a series of prompts that include determining the path to the SSH client executable and ends with the configuration of the MATLAB toolbox.

The SSH toolbox installer creates a configuration file for use with the SSH toolbox functions. This configuration file contains the path to the SSH client to be used by the toolbox. The use of a configuration file makes it easier for the toolbox to startup the SSH client without having to search for the program each time. The configuration file also contains the locations of the DLL and header file which are actually used for the inter-process communication. Although the file is a simple MATLAB script, users are not expected to modify the script as this might cause the toolbox to not function as expected.

## 2.6. Basic Commands (open connection, send, receive, rcp, disconnect)

The SSH Toolbox provides basic functionality in order to communicate with the remote host using SSH. This includes functions to connect to a remote host, send commands to the remote host, retrieve any messages/results printed to the standard output/error and disconnect from the remote host.

1. **sshconnect** – The *sshconnect* command is used to initiate a new remote session and connect to the desired remote host. The command is invoked as follows :
2. `sshconnect('oscw.osc.edu', 'user001');`
3. **sshdisconnect** – This command is used to end the current SSH session, disconnect from the remote host and close the SSH client. Currently this toolbox does not support multiple simultaneous SSH sessions and hence, the disconnect command does not require any additional information.
4. **sshsendcommand** – The *sshsendcommand* is used to run the desired command(s) on the remote host.
5. **sshstdout** – This function is used to retrieve any information/message(s) written to stdout and stderr on the remote host as a result of running a command/program on the remote host. The user must call this function explicitly in order to view the output from a command.
6. **hidesshwindow** – This command is used to hide the current SSH client window. By making the SSH window invisible, the toolbox can ensure that the user does not accidentally close the SSH client or interfere with it in any manner.
7. **showsshwindow** - This command is used to make the SSH client window visible again.

Using the above commands, a simple SSH session can be invoked in the following manner:

```
>> sshconnect('oscw.osc.edu', 'user01'); %  
Connect to remote host  
>> sshsendcommand('ls -l'); % Send command to  
remote host  
>> sshstdout; % Retrieve the output from the  
command. the output is displayed here  
>> sshdisconnect; % Disconnect from the remote  
host
```

## 2.7. File/Data Transfer

File and data transfers are achieved using the secure copy utility that is a part of the PuTTY distribution. The *pscp* utility which ships with the open source version of PuTTY is used to securely copy data/files to and from the remote host. An important part of the ability to transfer files is to have the PuTTY agent running. If the agent is

not running, the user will be prompted for the password each time a transfer is initiated.

## 2.8. SSH Agent and Kerberos

The SSH Toolbox relies on the ability to connect to remote hosts without prompting for the user password. In the Kerberos version of the PuTTY SSH client (distributed by the HPCMPO), the process is relatively straightforward because a user must obtain a valid Kerberos ticket before attempting to connect to a remote host. In this case, with the availability of a valid ticket, the user can connect to multiple remote hosts without the need to enter a password.

On the other hand, not all HPC centers support Kerberos and rely on simple username/password based authentication via SSH. In this case, a user has to enter their password each time when connecting to multiple hosts or using the *pscp* utility to transfer data/file(s). The solution to this problem is to use the SSH Agent when using the open source version of the PuTTY client. Based on the version of the PuTTY SSH client being used, the SSH Toolbox checks to see whether the SSH Agent is running. In case the user is running the open source version of PuTTY, the SSH Agent is required before starting a new session. If the Toolbox was installed properly, the SSH Agent is started before starting a new SSH session.

## 3. Results

The SSH Toolbox allows users to seamlessly interact with remote HPC resources from MATLAB. Functions are provided to allow users to connect, run remote commands, retrieve results and disconnect from the HPC resource. A non trivial face recognition example written in MATLAB was also used to illustrate the capabilities of this tool. The tool is also flexible enough to meet the needs of a wide range of users since it can work with Kerberos as well as open source SSH technology.

## Acknowledgements

This publication was made possible through support provided by DoD HPCMP PET activities through Mississippi State University under contract. The opinions expressed herein are those of the author(s) and do not necessarily reflect the views of the DoD or Mississippi State University.

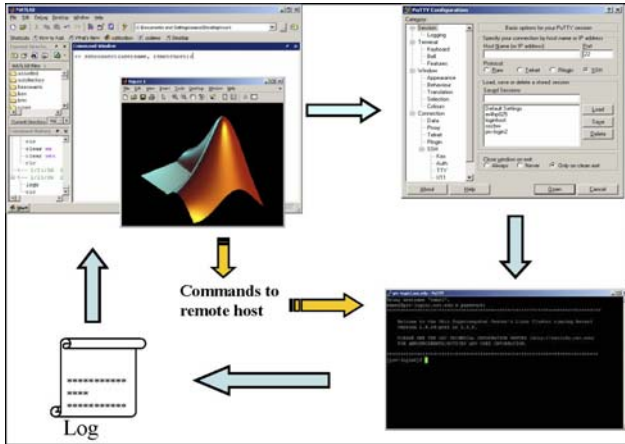


Figure 1. Overview of SSH Toolbox operation

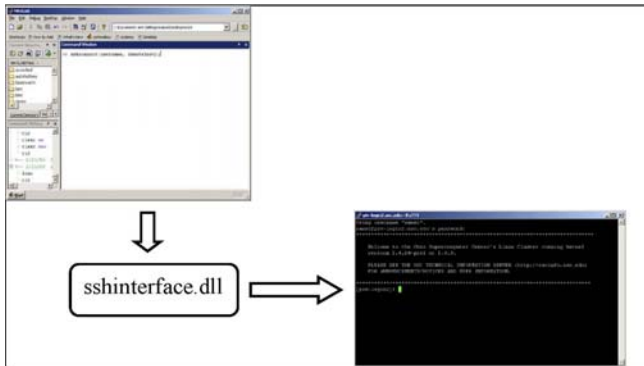


Figure 2. MATLAB – SSH communications

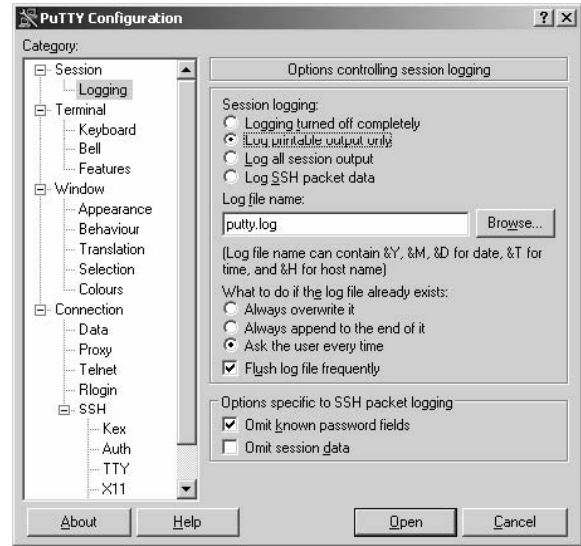


Figure 3. PuTTY configuration options