1 Overview

In this section, we present an overview of Scilab. The first subsection introduces the open source project associated with the creation of this document. Then we present the software, licence and scientific aspects of Scilab. In the third subsection, we describe the methods to download and install Scilab on Windows, GNU/Linux and Mac operating systems. In the remaining subsections, we describe various sources of information needed when we have to get some help from Scilab or from other users. We describe the built-in help pages and analyse the mailing lists and wiki which are available online. Finally, we take a moment to look at the demonstrations which are provided with Scilab.

1.1 Introduction

This document is an open-source project. The LaTeX sources are available on the Scilab Forge:

`http://forge.scilab.org/index.php/p/docintrotoscilab/`

The LaTeX sources are provided under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License:

`http://creativecommons.org/licenses/by-sa/3.0`

The Scilab scripts are provided on the Forge, inside the project, under the `scripts` sub-directory. The scripts are available under the CeCiLL licence:

`http://www.cecill.info/licences/Licence_CeCILL_V2-en.txt`

1.2 Overview of Scilab

Scilab is a programming language associated with a rich collection of numerical algorithms covering many aspects of scientific computing problems.

From the software point of view, Scilab is an *interpreted* language. This generally allows to get faster development processes, because the user directly accesses a high-level language, with a rich set of features provided by the library. The Scilab language is meant to be extended so that user-defined data types can be defined with possibly overloaded operations. Scilab users can develop their own modules so that they can solve their particular problems. The Scilab language allows to dynamically compile and link other languages such as Fortran and C: this way, external libraries can be used as if they were a part of Scilab built-in features. Scilab also interfaces LabVIEW, a platform and development environment for a visual programming language from National Instruments.

From the license point of view, Scilab is a free software in the sense that the user does not pay for it and Scilab is an open source software, provided under the Cecill license [2]. The software is distributed with source code, so that the user has an access to Scilab's most internal aspects. Most of the time, the user downloads and installs a binary version of Scilab, since the Scilab consortium provides Windows, Linux and Mac OS executable versions. Online help is provided in many local languages.

From the scientific point of view, Scilab comes with many features. At the very beginning of Scilab, features were focused on linear algebra. But, rapidly, the number of features extended to cover many areas of scientific computing. The following is a short list of its capabilities:

- Linear algebra, sparse matrices,
- Polynomials and rational functions,
- Interpolation, approximation,
- Linear, quadratic and non linear optimization,
- Ordinary Differential Equation solver and Differential Algebraic Equations solver,
- Classic and robust control, Linear Matrix Inequality optimization,
- Differentiable and non-differentiable optimization,
- Signal processing,
- Statistics.

Scilab provides many graphics features, including a set of plotting functions, which allow to create 2D and 3D plots as well as user interfaces. The Xcos environment provides a hybrid dynamic systems modeler and simulator.

1.3 How to get and install Scilab

Whatever your platform is (i.e. Windows, Linux or Mac), Scilab binaries can be downloaded directly from the Scilab homepage

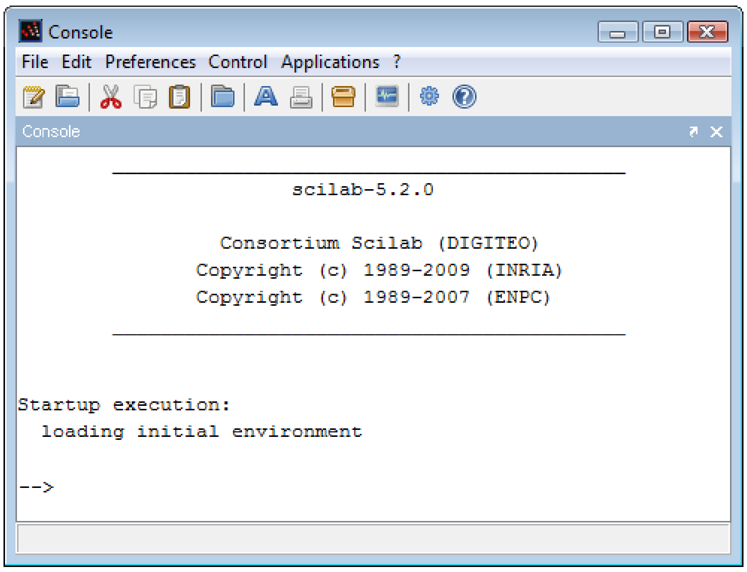
`http://www.scilab.org`

or from the Download area

`http://www.scilab.org/download`

Scilab binaries are provided for both 32 and 64-bit platforms so that they match the target installation machine.

Scilab can also be downloaded in source form, so that you can compile Scilab by yourself and produce your own binary. Compiling Scilab and generating a binary is especially interesting when we want to understand or debug an existing feature, or when we want to add a new feature. To compile Scilab, some prerequisites binary files are necessary, which are also provided in the Download center. Moreover, a Fortran and a C compiler are required. Compiling Scilab is a process which will not be detailed further in this document, because this chapter is mainly devoted to the external behavior of Scilab.



```
        ___________________________________________
                         scilab-5.2.0

                  Consortium Scilab (DIGITEO)
                Copyright (c) 1989-2009 (INRIA)
                Copyright (c) 1989-2007 (ENPC)
        ___________________________________________


Startup execution:
  loading initial environment

-->
```

Figure 1: Scilab console under Windows.

1.3.1 Installing Scilab under Windows

Scilab is distributed as a Windows binary and an installer is provided so that the installation is really easy. The Scilab console is presented in figure 1. Several comments may be made about this installation process.

On Windows, if your machine is based on an Intel processor, the Intel Math Kernel Library (MKL) [7] enables Scilab to perform faster numerical computations.

1.3.2 Installing Scilab under Linux

Under Linux, the binary versions are available from Scilab website as `.tar.gz` files. There is no need for an installation program with Scilab under Linux: simply unzip the file in one target directory. Once done, the binary file is located in *<path>/scilab-5.x.x/bin/scilab*. When this script is executed, the console immediately appears and looks exactly the same as on Windows.

Notice that Scilab is also distributed with the packaging system available with Linux distributions based on Debian (for example, Ubuntu). This installation method is extremely simple and efficient. Nevertheless, it has one little drawback: the version of Scilab packaged for your Linux distribution may not be up-to-date. This is because there is some delay (from several weeks to several months) between the availability of an up-to-date version of Scilab under Linux and its release in Linux distributions.

For now, Scilab comes on Linux with a binary linear algebra library which guarantees portability. Under Linux, Scilab does not come with a binary version of ATLAS [1], so that linear algebra is a little slower for that platform, compared to Windows.

1.3.3 Installing Scilab under Mac OS

Under Mac OS, the binary versions are available from Scilab website as a `.dmg` file. This binary works for Mac OS versions starting from version 10.5. It uses the Mac OS installer, which provides a classical installation process. Scilab is not available on Power PC systems.

Scilab version 5.2 for Mac OS comes with a Tcl / Tk library which is disabled for technical reasons. As a consequence, there are some small limitations on the use of Scilab on this platform. For example, the Scilab / Tcl interface (TclSci), the graphic editor and the variable editor are not working. These features will be rewritten in Java in future versions of Scilab and these limitations will disappear.

Still, using Scilab on a Mac OS system is easy, and uses the shorcuts which are familiar to the users of this platform. For example, the console and the editor use the Cmd key (Apple key) which is found on Mac keyboards. Moreover, there is no right-click on this platform. Instead, Scilab is sensitive to the Control-Click keyboard event.

For now, Scilab comes on Mac OS with a linear algebra library which is optimized and guarantees portability. Under Mac OS, Scilab does not come with a binary version of ATLAS [1], so that linear algebra is a little slower for that platform.

1.4 How to get help

The most simple way to get the online help integrated to Scilab is to use the function `help`. Figure 2 presents the Scilab help window. To use this function, simply type "`help`" in the console and press the <Enter> key, as in the following session.

```
help
```

Suppose that you want some help about the `optim` function. You may try to browse the integrated help, find the optimization section and then click on the `optim` item to display its help.

Another possibility is to use the function `help`, followed by the name of the function, for which help is required, as in the following session.

```
help optim
```

Scilab automatically opens the associated entry in the help.

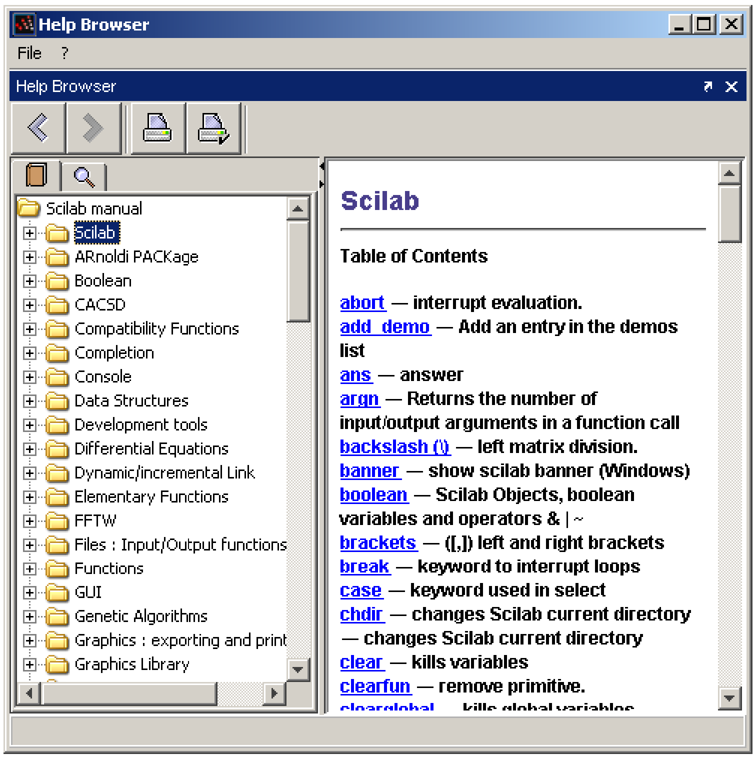
We can also use the help provided on the Scilab web site

http://www.scilab.org/product/man

This page always contains the help for the up-to-date version of Scilab. By using the "search" feature of my web browser, I can most of the time quickly find the help page I need. With that method, I can see the help pages for several Scilab commands at the same time (for example the commands `derivative` and `optim`, so that I can provide the cost function suitable for optimization with `optim` by computing derivatives with `derivative`).

A list of commercial books, free books, online tutorials and articles is presented on the Scilab homepage:

http://www.scilab.org/publications



Scilab

Table of Contents

abort — interrupt evaluation.
add_demo — Add an entry in the demos list
ans — answer
argn — Returns the number of input/output arguments in a function call
backslash (\) — left matrix division.
banner — show scilab banner (Windows)
boolean — Scilab Objects, boolean variables and operators & | ~
brackets — ([,]) left and right brackets
break — keyword to interrupt loops
case — keyword used in select
chdir — changes Scilab current directory
— changes Scilab current directory
clear — kills variables
clearfun — remove primitive.
clearglobal — kills global variables

Figure 2: Scilab help window.

1.5 Mailing lists, wiki and bug reports

The mailing list *users@lists.scilab.org* is designed for all Scilab usage questions. To subscribe to this mailing list, send an e-mail to *users-subscribe@lists.scilab.org*. The mailing list *dev@lists.scilab.org* focuses on the development of Scilab, be it the development of Scilab core or of complicated modules which interacts deeply with Scilab core. To subscribe to this mailing list, send an e-mail to *dev-subscribe@lists.scilab.org*.

These mailing lists are archived at:

http://dir.gmane.org/gmane.comp.mathematics.scilab.user

and:

http://dir.gmane.org/gmane.comp.mathematics.scilab.devel

Therefore, before asking a question, users should consider looking in the archive if the same question or subject has already been answered.

A question posted on the mailing list may be related to a very specific technical point, so that it requires an answer which is not general enough to be public. The address *scilab.support@scilab.org* is designed for this purpose. Developers of the Scilab team provide accurate answers via this communication channel.

The Scilab wiki is a public tool for reading and publishing general information about Scilab:

http://wiki.scilab.org

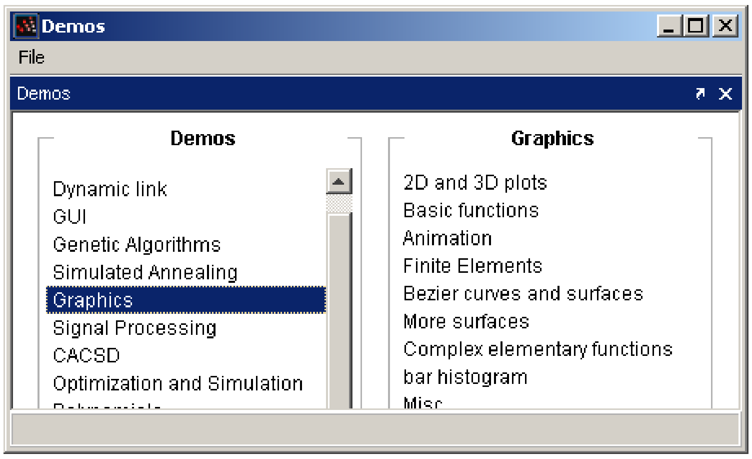


Figure 3: Scilab demos window.

It is used both by Scilab users and developers to publish information about Scilab. From a developer's point of view, it contains step-by-step instructions to compile Scilab from the sources, dependencies of various versions of Scilab, instructions to use Scilab source code repository, etc...

The Scilab Bugzilla *http://bugzilla.scilab.org* allows to submit a report each time we find a new bug. It may happen that this bug has already been discovered by someone else. This is why it is advised to search the bug database for existing related problems before reporting a new bug. If the bug is not reported yet, it is a very good thing to report it, along with a test script. This test script should remain as simple as possible, which allows to reproduce the problem and identify the source of the issue.

An efficient way of getting up-to-date information is to use RSS feeds. The RSS feed associated with the Scilab website is

`http://www.scilab.org/en/rss_en.xml`

This channel regularly delivers press releases and general announces.

1.6 Getting help from Scilab demonstrations and macros

The Scilab consortium maintains a collection of demonstration scripts, which are available from the console, in the menu *? > Scilab Demonstrations*. Figure 3 presents the demonstration window. Some demonstrations are graphic, while some others are interactive, which means that the user must type on the <Enter> key to go on to the next step of the demo.

The associated demonstrations scripts are located in the Scilab directory, inside each module. For example, the demonstration associated with the optimization module is located in the file

```
<path>\scilab-5.2.0\modules\optimization\demos\datafit\datafit.dem.sce
```

Of course, the exact path of the file depends on your particular installation and your operating system.

Analyzing the content of these demonstration files is often an efficient solution for solving common problems and to understand particular features.

Another method to find some help is to analyze the source code of Scilab itself (Scilab is indeed open-source!). For example, the `derivative` function is located in

```
<path>\scilab-5.2.0\modules\optimization\macros\derivative.sci
```

Most of the time, Scilab macros are very well written, taking care of all possible combinations of input and output arguments and many possible values of the input arguments. Often, difficult numerical problems are solved in these scripts so that they provide a deep source of inspiration for developing your own scripts.

1.7 Exercises

Exercise 1.1 (***Installing Scilab***) Install the current version of Scilab on your system: at the time where this document is written, this is Scilab v5.2. It is instructive to install an older version of Scilab, in order to compare current behavior against the older one. Install Scilab 4.1.2 and see the differences.

Exercise 1.2 (***Inline help:*** `derivative`) The `derivative` function allows to compute the numerical derivative of a function. The purpose of this exercise is to find the corresponding help page, by various means. In the inline help, find the entry corresponding to the `derivative` function. Find the corresponding entry in the online help. Use the console to find the help.

Exercise 1.3 (***Asking a question on the forum***) You probably already have one or more questions. Post your question on the users' mailing list *users@lists.scilab.org*.

2 Getting started

In this section, we make our first steps with Scilab and present some simple tasks we can perform with the interpreter.

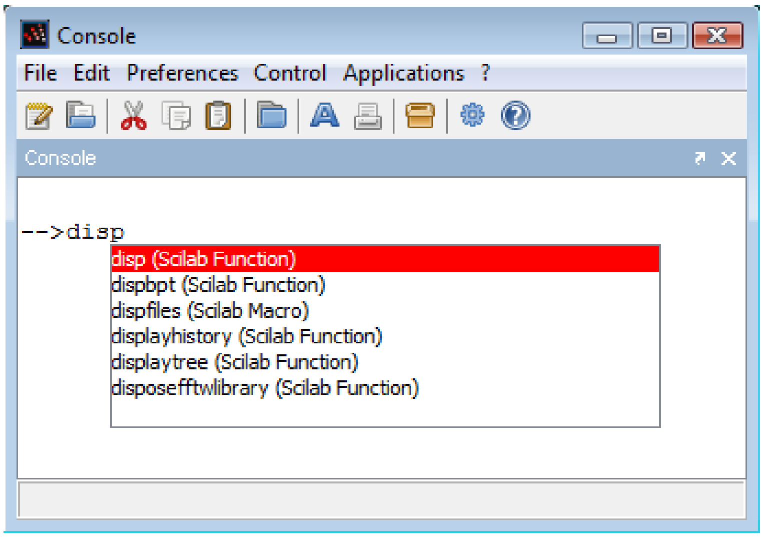
There are several ways of using Scilab and the following paragraphs present three methods:

- using the console in the interactive mode,
- using the `exec` function against a file,
- using *batch* processing.

2.1 The console

The first way is to use Scilab interactively, by typing commands in the console, analyzing the results and continuing this process until the final result is computed. This document is designed so that the Scilab examples which are printed here can be copied into the console. The goal is that the reader can experiment by himself Scilab behavior. This is indeed a good way of understanding the behavior of the program and, most of the time, it allows a quick and smooth way of performing the desired computation.

In the following example, the function `disp` is used in the interactive mode to print out the string "Hello World!".



```
-->disp
```

disp (Scilab Function)
dispbpt (Scilab Function)
dispfiles (Scilab Macro)
displayhistory (Scilab Function)
displaytree (Scilab Function)
disposefftwlibrary (Scilab Function)

Figure 4: The completion in the console.

```
-->s="Hello World!"
 s  =
 Hello World!
-->disp(s)
 Hello World!
```

In the previous session, we did not type the characters "-->" which is the *prompt*, and which is managed by Scilab. We only type the statement `s="Hello World!"` with our keyboard and then hit the `<Enter>` key. Scilab answer is `s =` and `Hello World!`. Then we type `disp(s)` and Scilab answer is `Hello World!`.

When we edit a command, we can use the keyboard, as with a regular editor. We can use the left ← and right → arrows in order to move the cursor on the line and use the <Backspace> and <Suppr> keys in order to fix errors in the text.

In order to get access to previously executed commands, use the up arrow ↑ key. This allows to browse the previous commands by using the up ↑ and down ↓ arrow keys.

The <Tab> key provides a very convenient completion feature. In the following session, we type the statement `disp` in the console.

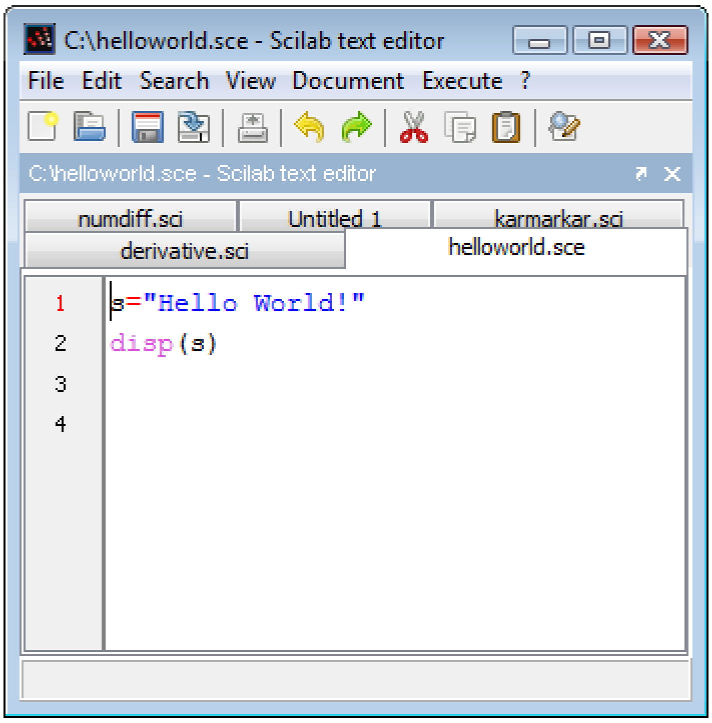
```
-->disp
```

Then we can type on the <Tab> key, which makes a list appear in the console, as presented in figure 4. Scilab displays a listbox, where items correspond to all functions which begin with the letters "disp". We can then use the up and down arrow keys to select the function we want.

The auto-completion works with functions, variables, files and graphic handles and makes the development of scripts easier and faster.

2.2 The editor

Scilab version 5.2 provides a new editor which allows to edit scripts easily. Figure 5 presents the editor during the editing of the previous "Hello World!" example.


C:\helloworld.sce - Scilab text editor
File Edit Search View Document Execute ?
C:\helloworld.sce - Scilab text editor
numdiff.sci
Untitled 1
karmarkar.sci
derivative.sci
helloworld.sce

```
s="Hello World!"
disp(s)
```

Figure 5: The editor.

The editor can be accessed from the menu of the console, under the *Applications* > *Editor* menu, or from the console, as presented in the following session.

```
-->editor()
```

This editor allows to manage several files at the same time, as presented in figure 5, where we edit five files at the same time.

There are many features which are worth to mention in this editor. The most commonly used features are under the *Execute* menu.

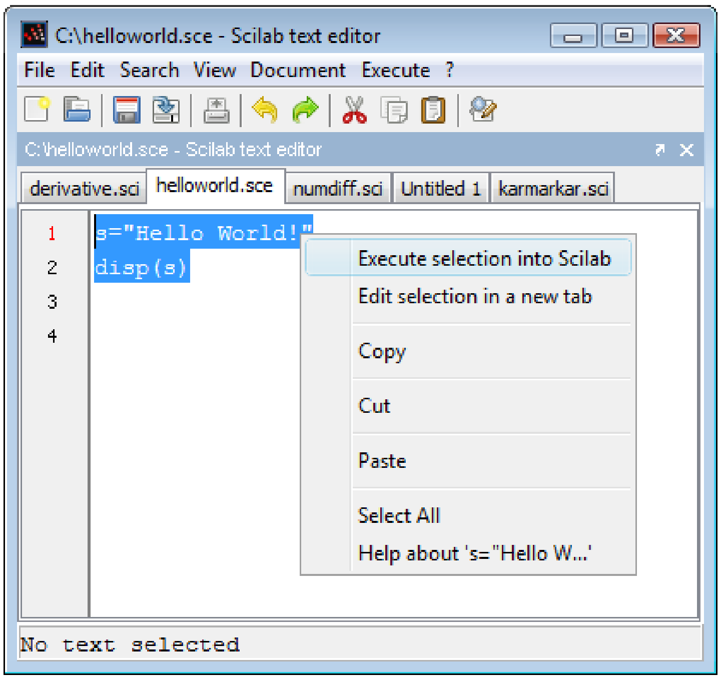
- *Load into Scilab* allows to execute the statements in the current file, as if we did a copy and paste. This implies that the statements which do not end with the semicolon ";" character will produce an output in the console.

- *Evaluate Selection* allows to execute the statements which are currently selected.

- *Execute File Into Scilab* allows to execute the file, as if we used the `exec` function. The results which are produced in the console are only those which are associated with printing functions, such as `disp` for example.

We can also select a few lines in the script, right click (or Cmd+Click under Mac), and get the context menu which is presented in figure 6.

The *Edit* menu provides a very interesting feature, commonly known as a "pretty printer" in most languages. This is the *Edit* > *Correct Indentation* feature, which


C:\helloworld.sce - Scilab text editor
File Edit Search View Document Execute ?
C:\helloworld.sce - Scilab text editor
derivative.sci
helloworld.sce
numdiff.sci
Untitled 1
karmarkar.sci
Execute selection into Scilab
Edit selection in a new tab
Copy
Cut
Paste
Select All
Help about 's="Hello W...'

```
s="Hello World!"
disp(s)
```

```
No text selected
```

Figure 6: Context menu in the editor.

automatically indents the current selection. This feature is extremelly convenient, as it allows to format algorithms, so that the `if`, `for` and other structured blocks are easy to analyze.

The editor provides a fast access to the inline help. Indeed, assume that we have selected the `disp` statement, as presented in figure 7. When we right-click in the editor, we get the context menu, where the *Help about "disp"* entry allows to open the help page associated with the `disp` function.

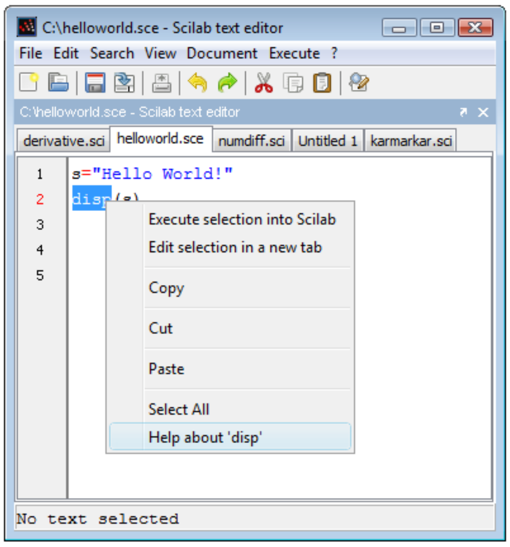
2.3 Docking

The graphics in Scilab version 5 has been updated so that many components are now based on Java. This has a number of advantages, including the possibility to manage docking windows.

The docking system uses Flexdock [10], an open-source project providing a Swing docking framework. Assume that we have both the console and the editor opened in our environment, as presented in figure 8. It might be annoying to manage two windows, because one may hide the other, so that we constantly have to move them around in order to actually see what happens.

The Flexdock system allows to drag and drop the editor into the console, so that we finally have only one window, with several sub-windows. All Scilab windows are dockable, including the console, the editor, the help and the plotting windows. In figure 9, we present a situation where we have docked four windows into the console window.

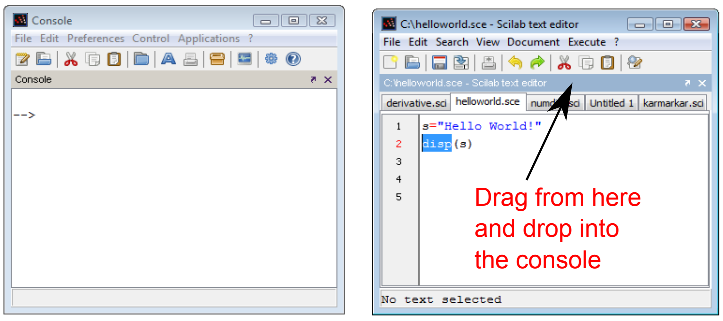
In order to dock one window into another window, we must drag and drop the



```
s="Hello World!"
disp(s)
```

C:\helloworld.sce - Scilab text editor
File Edit Search View Document Execute ?
C:\helloworld.sce - Scilab text editor
derivative.sci
helloworld.sce
numdiff.sci
Untitled 1
karmarkar.sci
Execute selection into Scilab
Edit selection in a new tab
Copy
Cut
Paste
Select All
Help about 'disp'
No text selected

Figure 7: Context help in the editor.

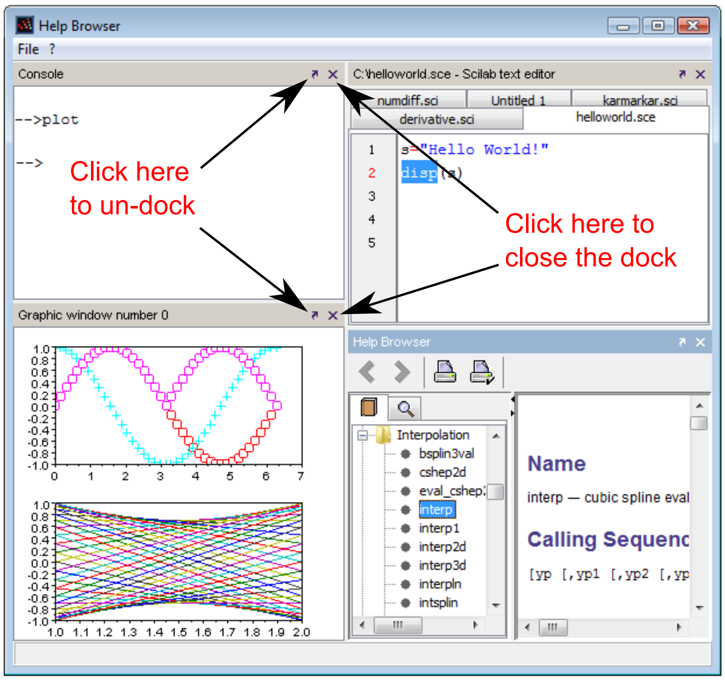


```
-->
```

```
s="Hello World!"
disp(s)
```

Console
File Edit Preferences Control Applications ?
Console
C:\helloworld.sce - Scilab text editor
File Edit Search View Document Execute ?
C:\helloworld.sce - Scilab text editor
derivative.sci
helloworld.sce
numdiff.sci
Untitled 1
karmarkar.sci
Drag from here and drop into the console
No text selected

Figure 8: The title bar in the source window. In order to dock the editor into the console, drag and drop the title bar of the editor into the console.



```
-->plot

-->
```

```
s="Hello World!"
disp(s)
```

```
[yp [,yp1 [,yp2 [,yp
```

Figure 9: Actions in the title bar of the docking window. The round arrow in the title bar of the window allows to undock the window. The cross allows to close the window.

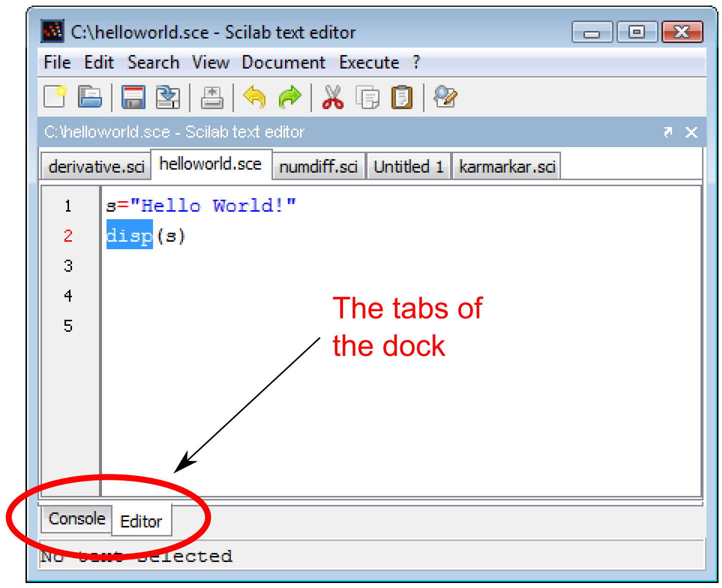
source window into the target window. To do this, we left-click on the title bar of the docking window, as indicated in figure 8. Before releasing the click, let us move the mouse over the target window and notice that a window, surrounded by dotted lines is displayed. This "phantom" window indicates the location of the future docked window. We can choose this location, which can be on the top, the bottom, the left or the right of the target window. Once we have chosen the target location, we release the click, which finally moves the source window into the target window, as in figure 9.

We can also release the source window *over* the target window, which creates tabs, as in figure 10.

2.4 Using `exec`

When several commands are to be executed, it may be more convenient to write these statements into a file with Scilab editor. To execute the commands located in such a file, the `exec` function can be used, followed by the name of the script. This file generally has the extension `.sce` or `.sci`, depending on its content:

- files having the `.sci` extension contain Scilab functions and executing them loads the functions into Scilab environment (but does not execute them),
- files having the `.sce` extension contain both Scilab functions and executable statements.


C:\helloworld.sce - Scilab text editor
File Edit Search View Document Execute ?
C:\helloworld.sce - Scilab text editor
derivative.sci
helloworld.sce
numdiff.sci
Untitled 1
karmarkar.sci
The tabs of the dock
Console
Editor
No text selected

```
s="Hello World!"
disp(s)
```

Figure 10: Docking tabs.

Executing a `.sce` file has generally an effect such as computing several variables and displaying the results in the console, creating 2D plots, reading or writing into a file, etc...

Assume that the content of the file `myscript.sce` is the following.

```
disp("Hello World !")
```

In the Scilab console, we can use the `exec` function to execute the content of this script.

```
-->exec("myscript.sce")
-->disp("Hello World !")
 Hello World !
```

In practical situations, such as debugging a complicated algorithm, the interactive mode is used most of the time with a sequence of calls to the `exec` and `disp` functions.

2.5 Batch processing

Another way of using Scilab is from the command line. Several command line options are available and are presented in figure 11. Whatever the operating system is, binaries are located in the directory *scilab-5.2.0/bin*. Command line options must be appended to the binary for the specific platform, as described below. The *-nw* option allows to disable the display of the console. The *-nwni* option allows to launch the non-graphics mode: in this mode, the console is not displayed and plotting functions are disabled (using them will generate an error).

-*e* instruction	execute the Scilab instruction given in instruction
-*f* file	execute the Scilab script given in the file
-*l* lang	setup the user language 'fr' for french and 'en' for english (default is 'en')
-*mem* N	set the initial stacksize
-*ns*	if this option is present, the startup file scilab.start is not executed
-*nb*	if this option is present, then Scilab welcome banner is not displayed
-*nouserstartup*	don't execute user startup files `SCIHOME/.scilab` or `SCIHOME/scilab.ini`
-*nw*	start Scilab as command line with advanced features (e.g., graphics)
-*nwni*	start Scilab as command line without advanced features
-*version*	print product version and exit

Figure 11: Scilab command line options.

- Under Windows, two binary executable are provided. The first executable is `WScilex.exe`, the usual, graphics, interactive console. This executable corresponds to the icon which is available on the desktop after the installation of Scilab. The second executable is `Scilex.exe`, the non-graphics console. With the `Scilex.exe` executable, the Java-based console is not loaded and the Windows terminal is directly used. The `Scilex.exe` program is sensitive to the -*nw* and -*nwni* options.

- Under Linux, the `scilab` script provides options which allow to configure its behavior. By default, the graphics mode is launched. The `scilab` script is sensitive to the -*nw* and -*nwni* options. There are two extra executables on Linux: `scilab-cli` and `scilab-adv-cli`. The `scilab-adv-cli` executable is equivalent to the -*nw* option, while the `scilab-cli` is equivalent to the -*nwni* option[8].

- Under Mac OS, the behavior is similar to the Linux platform.

In the following Windows session, we launch the `Scilex.exe` program with the -*nwni* option. Then we run the `plot` function in order to check that this function is not available in the non-graphics mode.

```
D:\Programs\scilab-5.2.0\bin>Scilex.exe -nwni
        ___________________________________________
                       scilab-5.2.0
                 Consortium Scilab (DIGITEO)
               Copyright (c) 1989-2009 (INRIA)
               Copyright (c) 1989-2007 (ENPC)
        ___________________________________________
Startup execution:
  loading initial environment
-->plot()
       !--error 4
Undefined variable: plot
```

The most useful command line option is the *-f* option, which allows to execute the commands from a given file, a method generally called *batch* processing. Assume that the content of the file `myscript2.sce` is the following, where the `quit` function is used to exit from Scilab.

```
disp("Hello World !")
quit()
```

The default behavior of Scilab is to wait for new user input: this is why the `quit` command is used, so that the session terminates. To execute the demonstration under Windows, we created the directory *"C:\scripts"* and wrote the statements in the file `C:\scripts\myscript2.sce`. The following session, executed from the MS Windows terminal, shows how to use the *-f* option to execute the previous script. Notice that we used the absolute path of the `Scilex.exe` executable.

```
C:\scripts>D:\Programs\scilab-5.2.0\bin\Scilex.exe -f myscript2.sce
        ___________________________________________
                       scilab-5.2.0
                 Consortium Scilab (DIGITEO)
               Copyright (c) 1989-2009 (INRIA)
               Copyright (c) 1989-2007 (ENPC)
        ___________________________________________
Startup execution:
  loading initial environment
 Hello World !
C:\scripts>
```

Any line which begins with the two slash characters "//" is considered by Scilab as a comment and is ignored. To check that Scilab stays by default in interactive mode, we comment out the `quit` statement with the "//" syntax, as in the following script.

```
disp("Hello World !")
//quit()
```

If we type the "`scilex -f myscript2.sce`" command in the terminal, Scilab will now wait for user input, as expected. To exit, we interactively type the `quit()` statement in the terminal.

2.6 Exercises

Exercise 2.1 (***The console***) Type the following statement in the console.

```
atoms
```

Now type on the <Tab> key. What happens? Now type the "I" letter, and type again on <Tab>. What happens?

Exercise 2.2 (***Using exec***) When we develop a Scilab, script we often use the `exec` function in combination with the `ls` function, which displays the list of files and directories in the current directory. We can also use the `pwd`, which displays the current directory. The `SCI` variable contains the name of the directory of the current Scilab installation. We use it very often to execute the scripts which are provided in Scilab. Type the following statements in the console and see what happens.

```
pwd
SCI
ls(SCI+"/modules")
```