Contract FP6-IST 508009

ACE

Antenna Centre of Excellence

Instrument: Network of Excellence

Thematic Priority: IST - Information Society Technologies
Mobile and wireless systems beyond 3G

Deliverable A3.1D5
First test of VALab prototype

Due date of deliverable: 31/12/2005
Actual submission date: 31/12/2005

Start date of project: 1/1/2004 Duration: 24 months

Organisation name of lead contractor for this deliverable: UNIFI

Revision 1

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Number: FP6-IST-508009-A3.1D5	
Workpackage: 3.1-3	
Estimated Person Months: 5	
Dissemination level (PU,PP,RE,CO): PU	
Nature (R, P, D, O): Report	
Version: 1	
Total Number of Pages: 51	
File name: ACE-A3.1D5-1.pdf	
Editors: Angelo Freni	
Participants: UNIFI	

Abstract

This report contains the description of the first VALab prototype.
This includes the description of the available services.

Keyword List

INTERNET, SOFTWARE

Document Evolution		
Rev. 1.0 Draft A	20/06/2005	Draft Edition
Rev. 1.0 Draft B	15/07/2005	Draft Edition
Rev. 1.0 Draft C	30/09/2005	Draft Edition, update the training section after the ESoA courses
Rev. 1.0	31/12/2005	First Edition

Table of contents

1. Introduction to the VALab services

1.1 Overview of the activity and its objective

For a high quality education in antenna design the availability of numerical tools is extremely helpful, since it provides an increased understanding of the physics behind an antenna problem and also reduces the need for expensive and bulky antenna measurement equipment.

Therefore, ideally a comprehensive antenna education, which includes different tools that today are being developed with little coordination among different institutions and research centers, should be of great value. Since the software is proprietary to the universities or companies that have developed them, sharing it directly is a problem.

The key feature of the VALab working package is to develop something more than a regular website where the student or/and the researcher can download only static pages, despite the fact that they can move from one to the other using hyperlinks. Moreover, the idea is to provide them the possibility of integrating some executable modules, already available among the ACE network, into their own codes. In the virtual laboratory all the material that the partners have decided to share with the other participants should be available for the researchers, according to the restrictions that the owners have indicated.

Moreover, the virtual laboratory would also include educational material and examples of different antennas design.

According to the requirements are arisen as far as the project was progressing, the VALab website has became also a support to the European School of Antennas.

The VALab website is organized according to the following basic scheme.

We expect that the users will consult the VALab to study a particular antenna problem in depth. In the beginning they can access the available educational material. This allows them to improve their knowledge in that field and also their know-how as far as the problem solution is concerned.

At this point, the users might be interested in putting into practice what they have learned; therefore, they can try to design a simple antenna or improve their knowledge on a specific antennas topic. This is the aim of the step-by-step design tool section. In particular, in this section they can find a few examples of antennas design and several tests on the basis of the courses of the European School of Antennas.

The key feature of this section is to let the users interact with the system in order to find a solution to the proposed problem. Moreover, in order to help the users better understand the physics behind the problem; there will be a few tests, in which they can observe the influence of the design parameters on the final results. This implies that some simple electromagnetic models have to be developed.

The users themselves have to develop numerical algorithms in order to solve more complex problems. It often happens that not all the required functions are available to or can be developed by the users. Some of them can be found on the web, but, in view of the copyright problems that arise in these cases, not all the necessary modules are downloadable.

Our idea is to let the partners share the execution of their codes on the VALab website, so that the users can “virtually” include these functions in their algorithms, which physically work in the owner entity machines. In order to test the results of the codes they have developed, the users will find some comparative solutions in the canonical problem section. These will be available not only as graphical or text-based data, but also as numerical algorithms or simulators already tested with specific benchmarks.

1.2 Brief description of the VALab services

The Virtual Antenna Laboratory (VALab) provides the following services

- Educational materials, where the users find programs, example, and anything else they might need to improve their knowledge in a specific antenna field.
- A set of step-by-step design tools which allows users to practice the design of different antennas;
- A set of canonical solutions that can be used to compare results from self-developed algorithms;
- A set of executable modules (server-side) that the users can use as a base for easily building and testing their own numerical codes (the modules remain proprietary to the single entities; they can be only used but not downloaded by the user).
- European School of Antennas supports section, where the links to the courses, to the material, the tests and the exams can be found.

1.3 Brief description of the VALab architecture

As far as the hardware features are concerned, the VALab server will be equipped with an Intel Pentium double processor with at least 1 GB RAM.

It will be set up as a Linux server based on the White Box Linux that offers all the services needed by ISPs and hosts (web server (SSL-capable), mail server (with SMTP-AUTH and TLS), DNS server, FTP server, MySQL server, POP3/IMAP, Firewall, etc.).

An Intel® Xeon™ dual processor with 8 Gb RAM running under Gentoo Linux will also be available for the remote execution of the codes.

The detailed features of the system will be defined once we know the willingness of the involved partners to share their codes. However, we have set a few guidelines concerning the software requirements for both the server and client side.

At first, the partners, in order to share their codes, have to provide a Linux server. This allows us to test all the procedures using a favorable server.

This way, we can distinguish and therefore solve the problems concerning the operative system of the server to those of the VALab performance.

Later on, we will develop a set of functions suitable for the different operating systems on which the servers run. Even though these requirements are mainly related to the remote execution section, they can be extended also to the sharing of design tools and canonical solutions.

Moreover, we have to develop a search engine that will perform an efficient database search. This will be based on a keyword search, as is the most common form of text search on the web.

We describe each submitted contribution by a list of keywords, extracted from the brief description item of each form, and then we add these to the search engine index. When a query is entered, it is checked against the search engine's index of each section of the website, based on the choice selected.

The search form will be on the main page and also in each section of the VALab website. In particular, the main page search form offers the possibility of performing an accurate search in the entire website or of choosing one of the sections.

The results will provide a direct link to the relevant web page.

In each section a search tree will also be provided. This will allow the user to go through the subject matter.

2. First tests on VALab prototype

Considering the available material, the objectives of the VALab website and the requirements of the ACE partners we focused the first tests on the VALab prototype on the pages related to the remote execution.

In particular, we performed several tests to develop a code which allows each partner the remote execution of shared routines. This code is particularly descripted in the section "remote execution service".

The aim of the prototype was to develop, for each VALab service, a sort of scheme of work so that we can test the usability and improve its performance.

Moreover, we tried to involve so many partners as possible, to verify the compatibility between the developed scheme and the available material.

In particular, in the educational material section we include some applet Java developed by the Institut d'Electronique et de Télécommunications de Rennes (IETR) besides a software developed by Chalmers University to test the efficiency of the website with different kind of materials.

As regards the canonical solution section we developed a scheme to easily add new pages. In the prototype page is included a link to the theory background, and different ways to show the reference solution. The solution is given both in graphically and text form.

This choice permits to users to see directly the solution on his screen but also copy the data from the clipboard and to use them in their own codes.

During the developing of the prototype some difficulties arisen both practical and theoretically. As descript in the introduction, the VALab website contains four different sections: Educational material, canonical solution, steep by step design tool and remote execution.

As expected the requirements of the users chanced so far the project is progressing.

In particular, the educational material, the canonical solution and the remote execution maintain the expected contents, while the step by step section contents have been modified by a section which support the European School of Antennas.

Taking into account the impressions and the remarks on the European School of Antennas courses, we develop a tool to build an online "training page" that the students can use to verify their knowledge.

The students enter to the web page of the course they attended, and they find the text both of the tests and the exams. As the latter, they answer to the questions and their answers are collect in a database so that they received immediately the final mark.

Before approaching to the final exam the can test their knowledge with several tests which have the same structure of the exam.

As far as the test is concerned the students can answer to the questions verify if it is correct or not and when necessary the will find an explanation on the theory behind the problem.

From the teachers point of view they have to prepare the tests and the exams. The tool we develop is easy to use. The lecturers have to prepare a word or a latex file adding only a few keywords so that the web page can be automatically created. Once the texts are available the courses database is automatically updated and the material is made available to the students.

Furthermore, for each course of the European School has been reserved a page where the students, that have attended to the course, the teachers and the Network partners can download all course material.

In particular, it has been developed a semi-automatic process that allows the course coordinator to easily build and upgrade the page.

3. Description of VALab services

3.1 Educational material

This section is aimed at all users interested in using computer-based/web-based educational materials. Most ACE partners have developed more or less extensive amounts of educational material such as notes, slides and videos. However, most of this data is not made available to other partners and, even more often, partners in different entities are not aware of what other colleagues have produced.

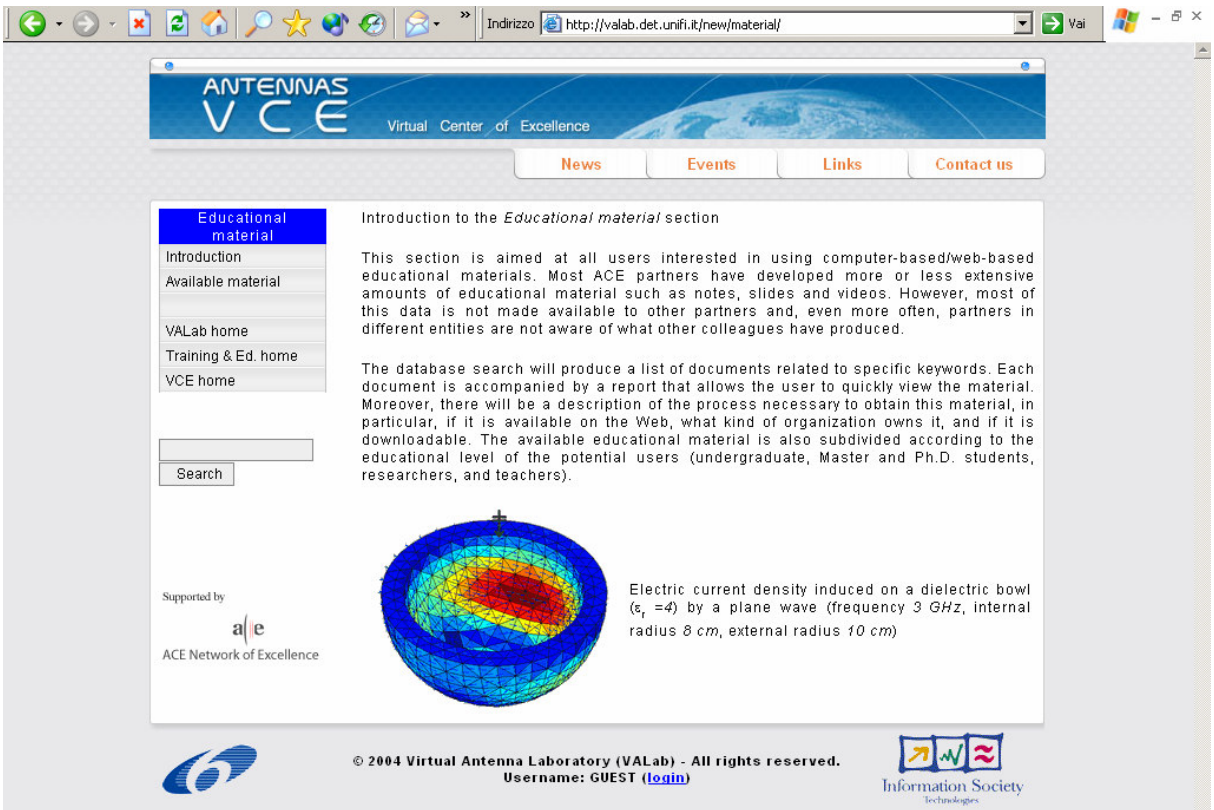


Figure 1 - Educational material - Introduction page

The introduction page (see Figure 1) contains a brief description of the aim of this section and on the left there is a menu which introduces the user to the available material page or where the user can perform a keyword search.

An example of the output of the database search is the following. If we search in the database for the string "array" we obtain output shown in Figure 2.

Figure 2 - Educational material - Database search example

Several tests have been computed on this section in order to allow to the partners to share different kind of material. The download of the notes, programs but also the possibilities to share applet java. The latter are useful for the students that can obtain an immediate result of what they have studied. On different topics is possible to perform such example so that the student, simply changing one or more parameters, can observe how the results change.

A simple example, depicted in Figure 3 is based on the design of arrays which are quite simple from a theoretical point of view but they are a very good test for this kind of execution.

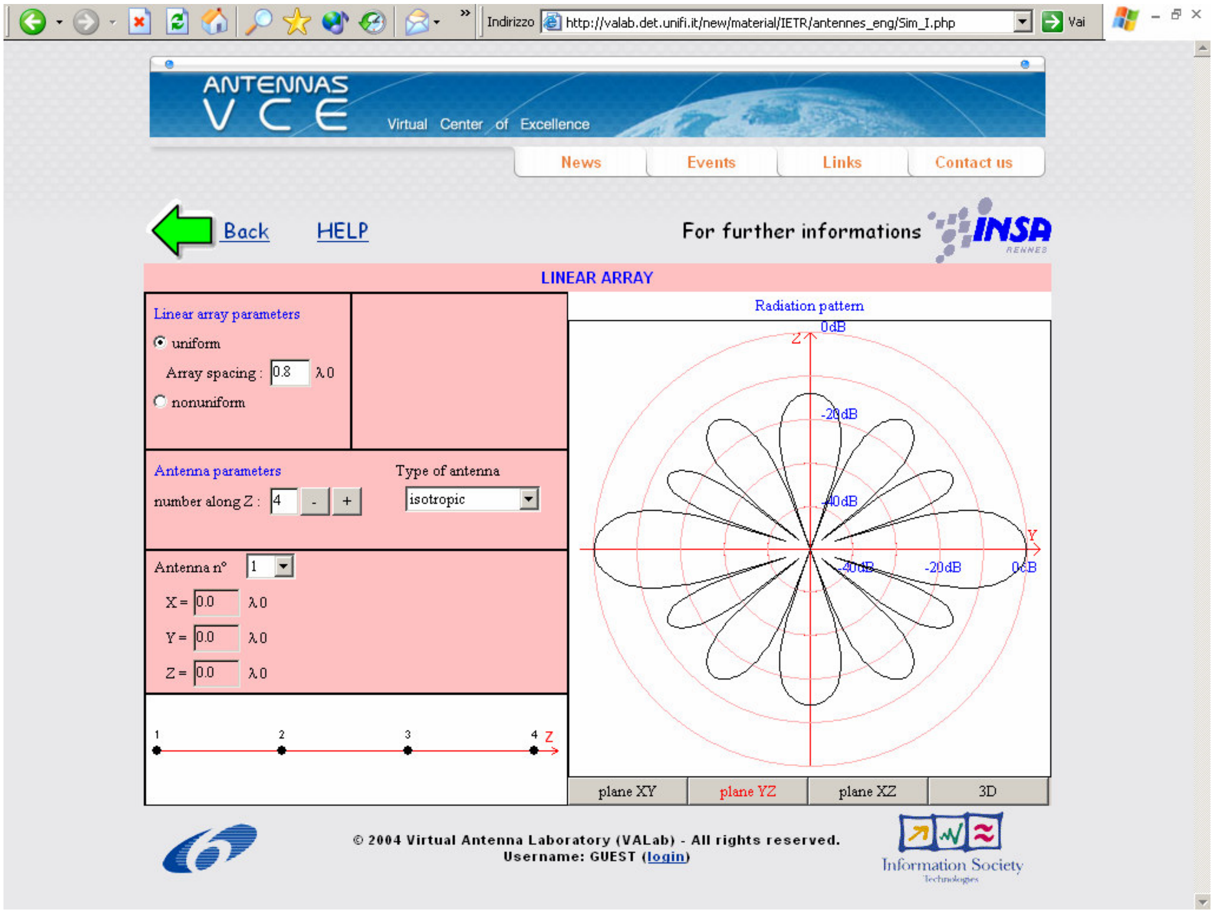


Figure 3 - Educational material- An example of the available material

3.2 Canonical solution

The aim of this section is to offer a set of solutions that the users could compare to the ones they have generated, in order to test the accuracy of their own code.

Figure 4 -Canonical solution - Introduction page

Moreover, for each canonical problem, a link to the theory is also available; it provides a theoretical background on the formulation used to achieve the specific result.

As an example of the available material we developed a typical antennas problem that is the design of a pyramidal horn.

On the top of the page the user finds a link to a *theory page* as depicted in Figure 6. It can be either a PDF file or an html page. For the example of Figure 5, clicking on the theory link a new page containing a PDF document will be open.

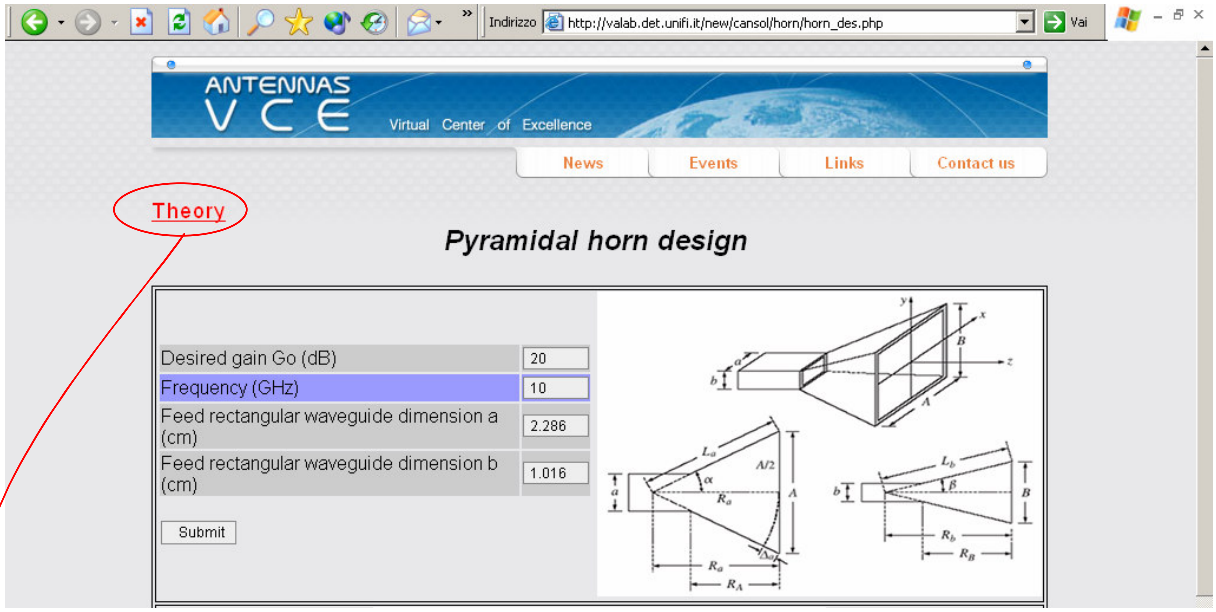
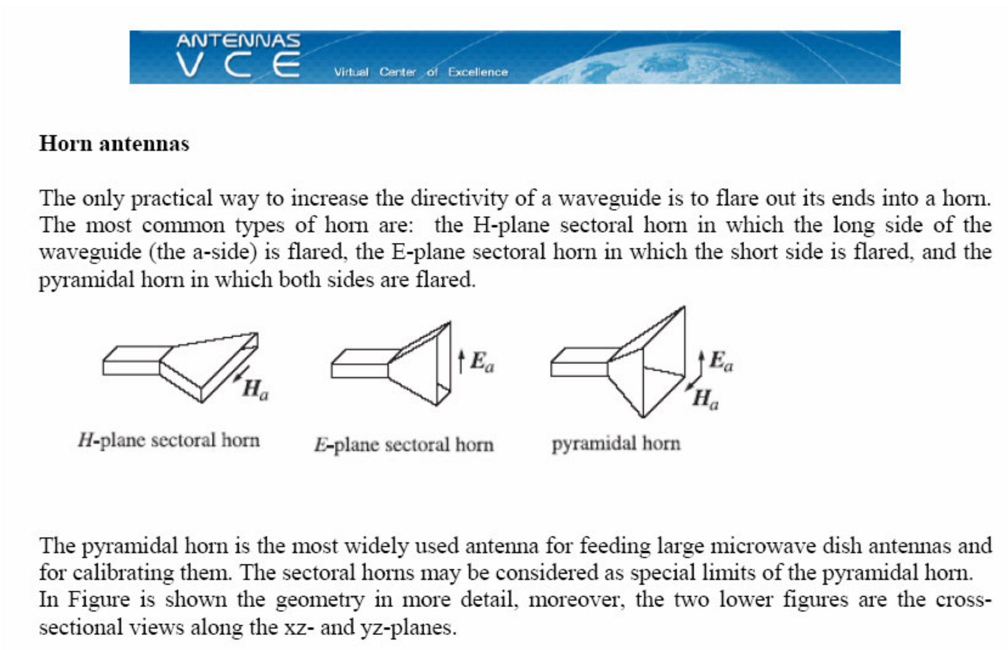


Figure 5 - Canonical solution example - Pyramidal horn design



ANTENNAS VCE Virtual Center of Excellence

Horn antennas

The only practical way to increase the directivity of a waveguide is to flare out its ends into a horn. The most common types of horn are: the H-plane sectoral horn in which the long side of the waveguide (the a-side) is flared, the E-plane sectoral horn in which the short side is flared, and the pyramidal horn in which both sides are flared.

H-plane sectoral horn *E*-plane sectoral horn pyramidal horn

The pyramidal horn is the most widely used antenna for feeding large microwave dish antennas and for calibrating them. The sectoral horns may be considered as special limits of the pyramidal horn. In Figure is shown the geometry in more detail, moreover, the two lower figures are the cross-sectional views along the xz- and yz-planes.

Figure 6 - Canonical solution - Theory page example

After studying the theory used to develop the canonical solution, and in this example also to understand the geometry used, the user fills the blanks with his value.

For the specific problem of Figure 7, the required fields are:

- the desired gain ;
- the working frequency;
- the feeding rectangular waveguide dimensions a, b.

All the quantities are referred to the figure on the right side or to the theory file.

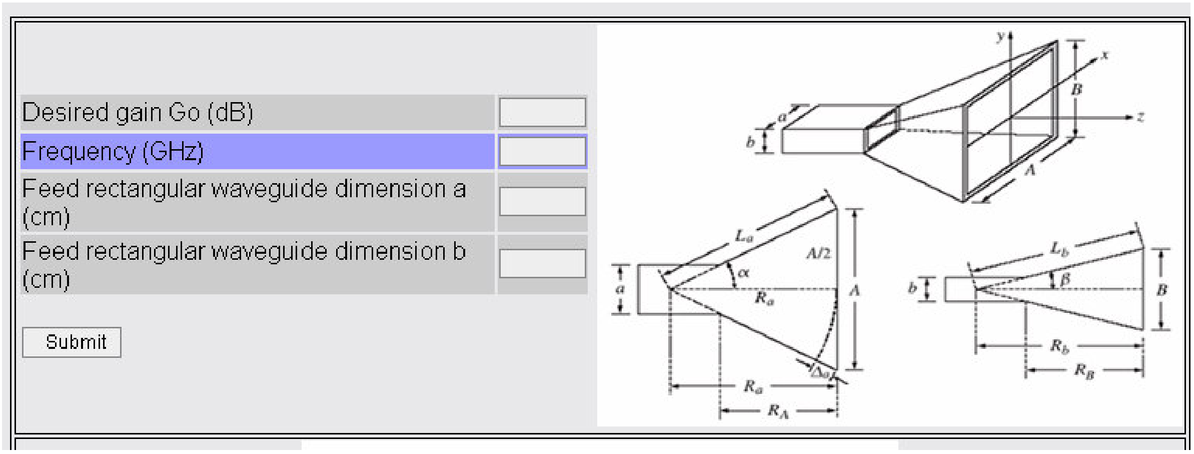


Figure 7 - Horn antennas design - Required fields

Once the values are completed the form is submitted and just below the geometry of the problem will appear the amplitude of the E-plane and H-plane pattern, as shown in Figure 8. Moreover, will be given the design parameters value, once again with reference to the geometry in figure, and also the numerical values of the amplitude of the field. The user can easily copy this values in the clipboard and then paste them in a file or in a data sheet in order to better compare them with the own results.

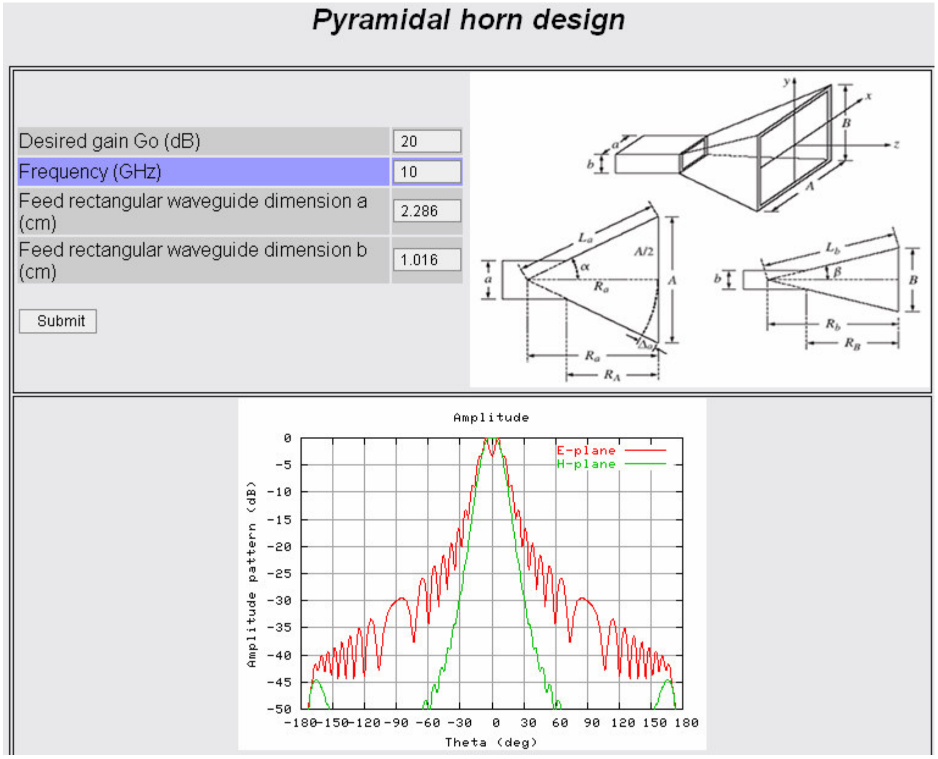


Figure 8 - Horn antennas design –Results

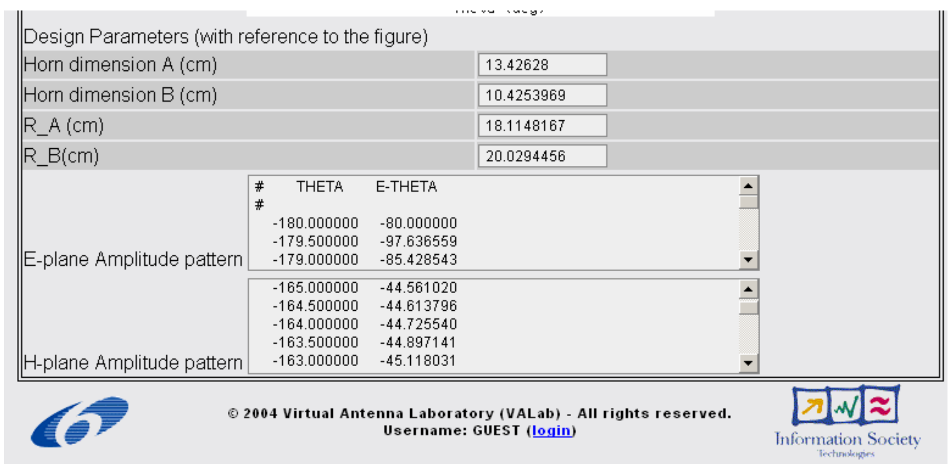


Figure 9 - Horn antennas design -Results

3.3 Remote execution service

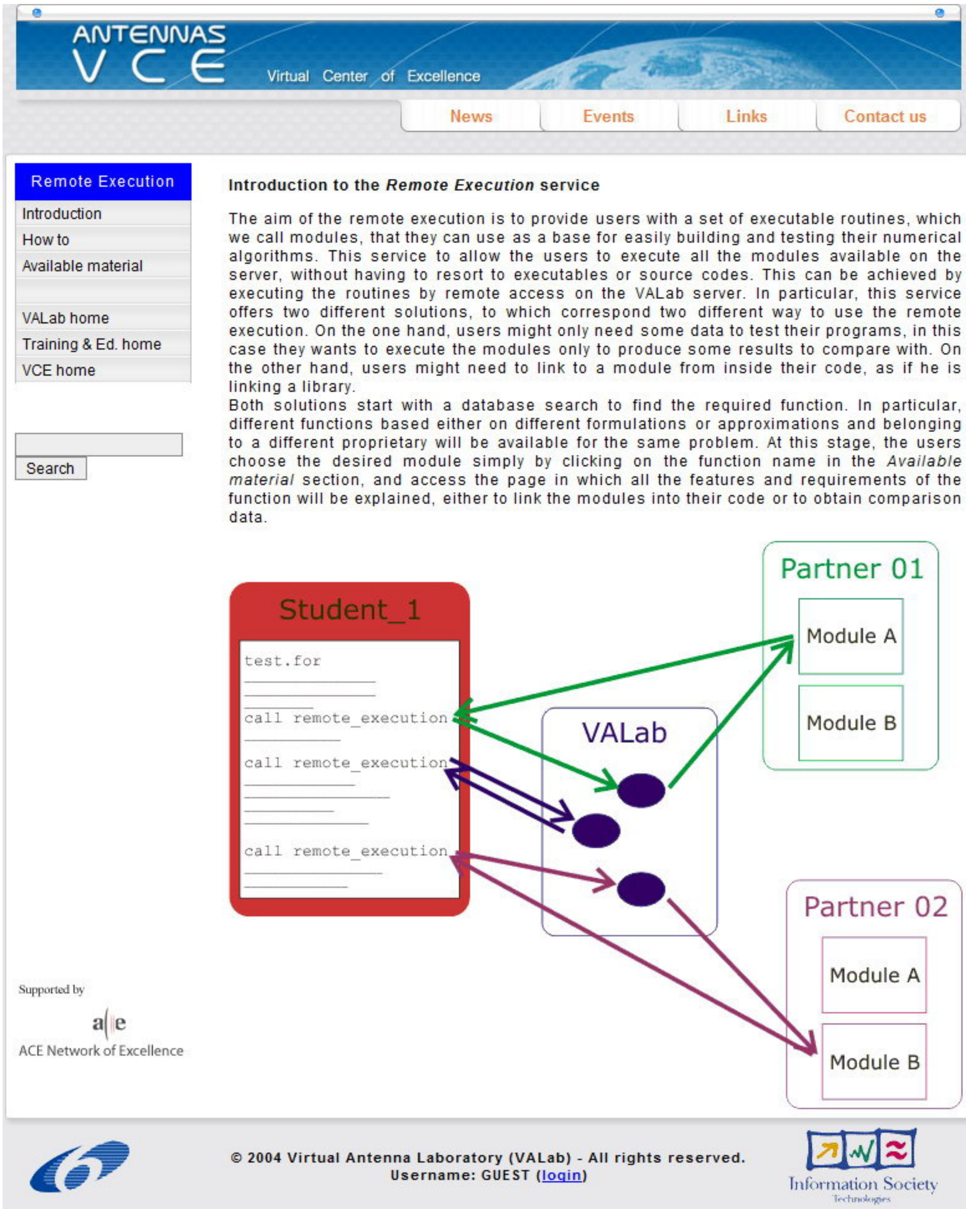


Figure 10 - Web homepage of Remote Execution service.

The aim of the remote execution is to provide users with a set of executable routines, which we will call *modules*, which can be used as a base for easily building and testing their numerical algorithms. It's a distinctiveness of this service to allow the users to execute all the modules available on the server, without having to resort to executables or source codes. This can be achieved by executing the routines by remote access on the VALab server.

In particular, this service offers two different solutions, to which correspond two different way to use the remote execution. On the one hand, users might only need some data to test their programs; in this case they want to execute the modules only to produce some results to compare with. On the other hand, users might need to link to a module from inside their code, as if he is linking a library.

Figure 11 - The *How to* web site section of the Remote Execution service.

In order to provide this service a transfer protocol has been developed by our group. A client program, named ***cctp***, necessary to interface to the server through this protocol, has been developed as well. The users must use the program ***cctp*** to have the use of executable modules disposable on the server. The ***cctp*** is available for download in both windows and linux version.

In the *How To* section (Figure 11) the users can found the ***cctp User Manual*** containing step by step instruction that allow the user to fully benefit of the Remote Execution service. A detailed description on how this protocol works and how to use the remote execution is given in the following paragraph.

After the users downloaded the cctp program and read the User Manual, the *Available Material* section (Figure 12) provides the list of all available modules with the relevant parameter to configure the cctp properly to use the module itself.

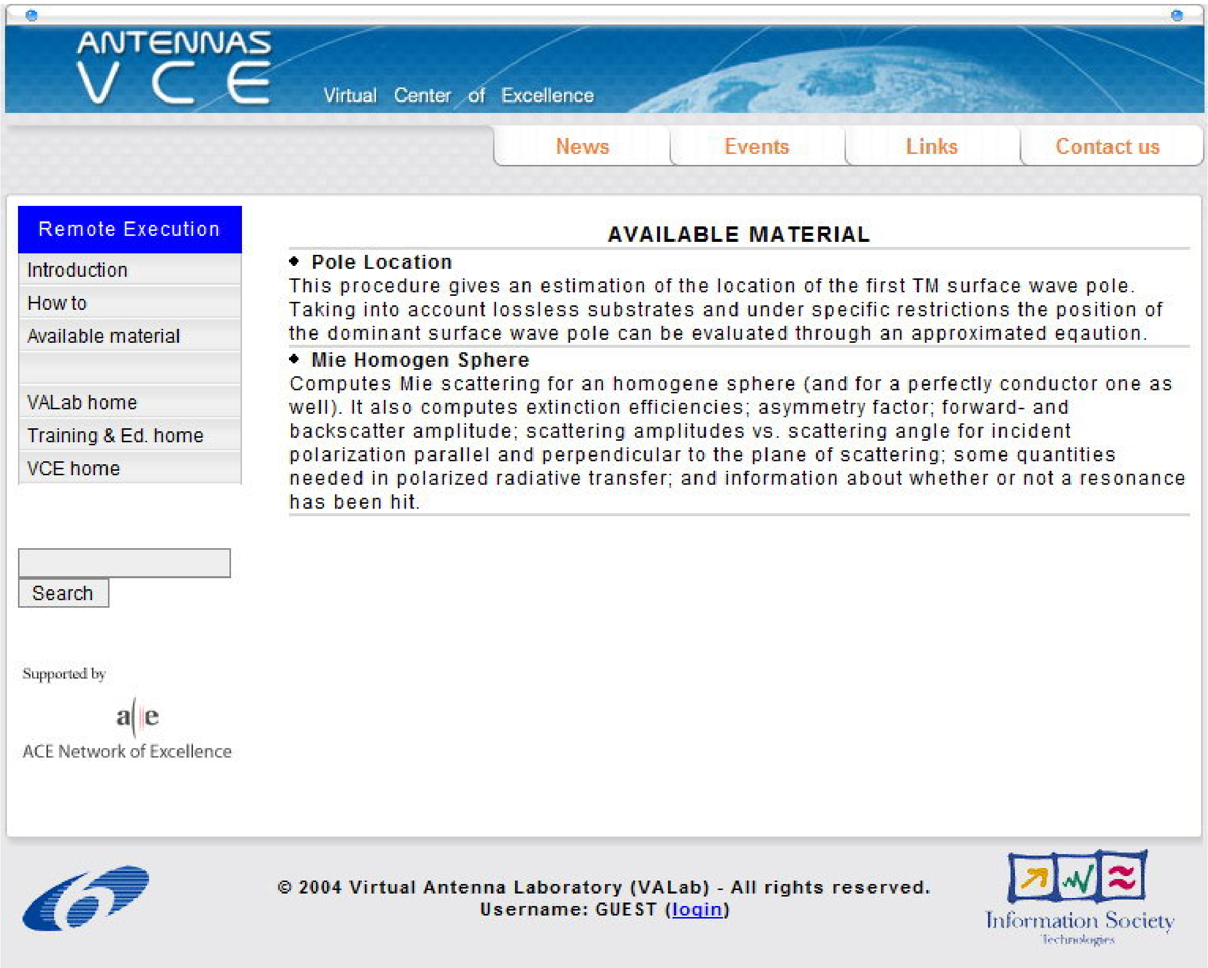


Figure 12 - The *Available Material* web site section of the Remote Execution service.

In the section *Available Material* the users can chose the desired module simply by clicking on the function name and access to the dedicated page. Here the users will find a detailed description of the executable module with all the features and requirements. A detailed explanation on how to use the module, either to link the modules into their code or to obtain comparison data, it is also furnished.

3.3.1. cctp User Manual

Introduction

cctp (that stands for Communication Channel Transfer Protocol) is the client program that allows to use the VALab remote execution. The aim of the remote execution is to enable a user to use particular routines, which we will call *modules*, without having to resort to executables or source codes. This can be done by executing the routine by remote access on the VALab server. *cctp* is the executable that allows the user to send to the VALab server the parameters and the input file(s), containing the data that a particular module requires, and to receive the output file(s), containing the results. *cctp* can also be used by the user to check and manage the status of the requested modules executions.

cctp can be used directly by command line, to obtain some data useful to compare with one's own program results, or can be recalled through a SYSTEM within any program created by the user. In this way, the module will seem like any other subroutine of the program created by the user; the only difference is that the data processing does not take place on the local system, but on a remote access system (another difference is that the user must "prepare" the input file requested by the module, and recover the data produced from the output file).

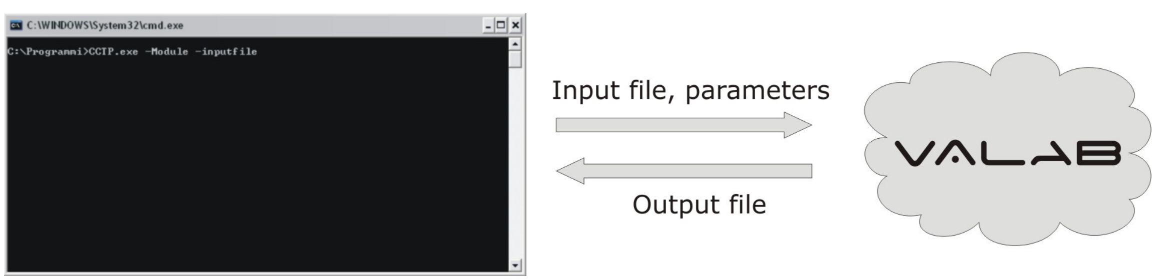


Figure 13 - Using cctp from command prompt.

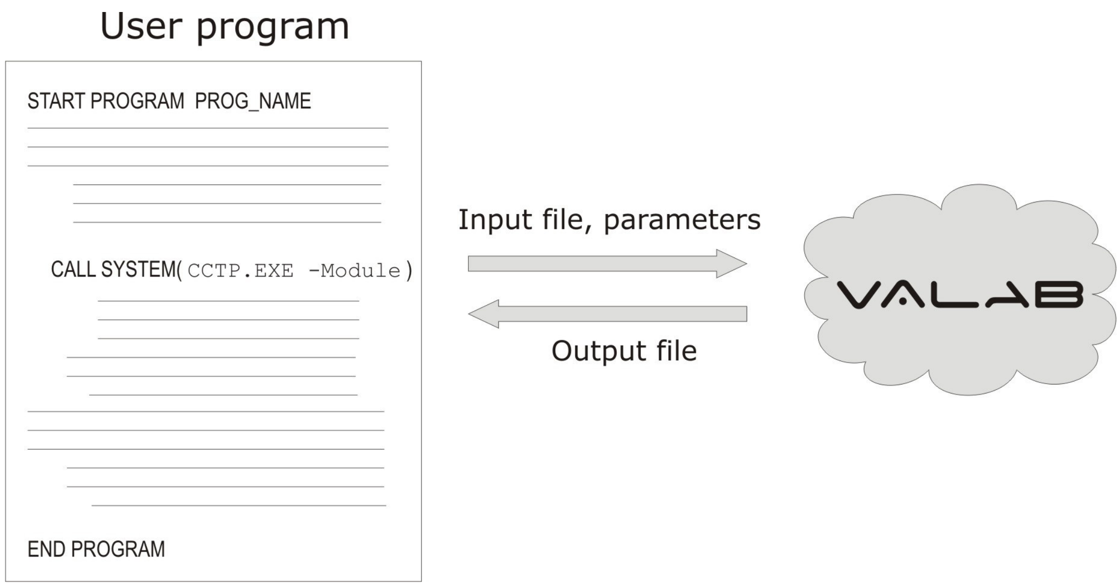


Figure 14 - Using cctp from within user program.

Options and configuration file

cctp is a program that supports several command-line options. To see all available options use `cctp -?` or `cctp -h` at command line.

The same options can also be set in the configuration file *cctp.ini*. This is a required file and must be present in the program's directory. If a same option is specified in both command line and *cctp.ini* file only the command line option will take effect. In other words the command line options always have the precedence on the options specified in the *ini* file.

Each option in the configuration file has to be specified by inserting a line with the following syntax:

option_keyword = option_value

All *option_keywords* and command-line option are case insensitive.

Just to give an example, here is reported the content of a typical *cctp.ini* configuration file.

```
#
#     cctp.ini configuration file
#

Username      = your_user
```

```
Password      = your_pass

op_mode       = sr

module_name   = mie
user_id       = 12345
process_id    =
input_file    = file1.in, file2.in
output_file   = file3.out
parameters    =
wait_time     = 20
```

From the previous example can be noted several features of the *cctp.ini* file. Let's proceed to analyze all of them.

Initially we can see how to put comments in the *ini* file. Lines beginning with the hash character "#", as well as blank lines, are ignored, so that you can put comments and format the option lines as desired.

The first two options specify your *Username* and your *Password* (these are not properly "options"; these are more like program settings). These two settings are compulsory and are the only ones that you cannot specify on command line. Remember that the registration on VALab server is required in order to use the remote execution. After the registration process a *Username* a *Password* and a *User_ID* will be assigned to you.
You can put any number of spaces before and after the equal sign "=".
If an option is not specified, you can remove the whole option line or put nothing after the equal sign "=".

If an option_keyword supports more than option_value (e.g. `input_file`), the values are separated by commas. You can put any number of spaces between values and commas.

How to use cctp

There are three main modes of operation: *Send and Receive*, *Send*, *Receive*. Plus there are other three operation modes, to be used to manage and check the requested module executions: *Status, Clear, Reset*.

The desired operation mode can be chosen by setting the right command line option. It's compulsory to specify at least one operation mode when you run *cctp*, i.e. there is no default operation mode.

Send and Receive

command-line option: `-sr`

configuration file option: `op_mode = sr`

This operation mode can be set with the option `-sr` or by placing the line `op_mode = sr` in the *cctp.ini* file.

In this mode *cctp* send the input file(s), the parameters and all the necessary information to VALab, waits for the output file(s) and return the prompt only when the output file(s) is successfully downloaded or when some errors occur.

In this operation mode the following options are compulsory:

- User_ID

  command-line option: `-u your_user_id`

  configuration file option: `user_id = your_user_id`

  This option sets the *User_ID* that you received after the registration process on VALab server. `your_user_id` is a number between 1 and 99999999.

- Module Name

  command-line option: `-m chosen_module_name`

  configuration file option: `module_name = chosen_module_name`

  This option sets the Module Name that you want executing by remote execution. `chosen_module_name` is the name provided in the web site module instructions.

The followings are optional:

- Process_ID

  command-line option: `-p process_id`

  configuration file option: `process_id = chosen_process_id`

  This option sets the Process_ID. `process_id` must be a number between 1 and 99999999. The Process_ID, together with the User_ID, is a number that entirely identifies on server the running process you requested. To set the Process_ID can be useful to subsequently retrieve module execution information.

- Input file(s)

  command-line option: `-i filename_1 filename_2 …`

  configuration file option: `input_file = filename_1, filename_2, …`

  This option specifies one or more input files to send to the module. These contain all the input information necessary to the proper execution of module itself. You (or your program) must create the appropriate input file for the chosen module. All necessary instructions can be found in the website. If more than one input file name have to be specified, separate the names with spaces in command-line and with commas in the *ini* configuration file.

- Output file(s)

  command-line option: `-o filename_1 filename_2 …`

  configuration file option: `output_file = filename_1, filename_2, …`

This option specifies one or more output files to retrieve from VALab server when the module execution will successful terminate. These files contain all the output information. All necessary explanation on the data contained in these file can be found in the website. If more than one output file name need to be specified, separate the names with spaces in command-line and with commas in the *ini* configuration file.

- Parameters
  command-line option: `-x (string of parameters)`
  configuration file option: `parameters = string of parameters`

  This option specifies some command line parameters that are passed directly to the module executable. These parameters are not interpreted by *cctp* and are treated as a character string, so you must write these parameters exactly as you want that they will appear after the module executables. The parameters string must be enclosed between brackets in the command line option (please note the space between the x and the open bracket). In the *ini* configuration file, simply write the parameters string after the equal sign "=".

- Wait time
  command-line option: `-w n`
  configuration file option: `wait_time = n`

  This option set the time between the forwarding of the input information to VALab and the first attempt to retrieve the output file. `n` is the number of seconds. The default value is 5 seconds.

Send

command-line option: `-send`
configuration file option: `op_mode = send`

This operation mode can be set with the option `-send` or by placing the line `op_mode = send` in the *cctp.ini* file.
In this mode *cctp* only send the input file(s), the parameters and all the necessary information to VALab and return the prompt only when VALab server confirms the module execution or when some errors occur. It is necessary to use *cctp* in *receive* mode to download the output file(s).

In this operation mode the following options are compulsory:

- User_ID
  command-line option: `-u your_user_id`
  configuration file option: `user_id = your_user_id`

This option sets the *User_ID* that you received after the registration process on VALab server. `your_user_id` is a number between 1 and 99999999.

- Process_ID

  command-line option: `-p process_id`

  configuration file option: `process_id = chosen_process_id`

  This option sets the Process_ID. `process_id` must be a number between 1 and 99999999. The Process_ID, together with the User_ID, is a number that entirely identifies on server the running process you requested. In *send* operation mode, it is compulsory specifing the Process_ID number, because it must be used later to retrieve the output file(s) using *cctp* in *receive* mode.

- Module Name

  command-line option: `-m chosen_module_name`

  configuration file option: `module_name = chosen_module_name`

  This option sets the Module Name that you have to execute by remote execution. `chosen_module_name` is the name provided in the web site module instructions.

The following options are optional:

- Input file(s)

  command-line option: `-i filename_1 filename_2 …`

  configuration file option: `input_file = filename_1, filename_2, …`

  This option specifies one or more input files to send to the module. These contain all the input information necessary to the proper execution of module itself. You (or your program) must create the appropriate input file for the chosen module. All necessary instruction can be found in the website. If more than one input file name need to be specified, separate the names with spaces in command-line and with commas in the *ini* configuration file.

- Output file(s)

  command-line option: `-o filename_1 filename_2 …`

  configuration file option: `output_file = filename_1, filename_2, …`

  This option specifies one or more output files to retrieve from VALab server when the module execution will successful terminate. These files contain all the output information. All necessary explanation on the data contained in these file can be found in the website. If more than one output file name need to be specified, separate the names with spaces in command-line and with commas in the *ini* configuration file.

- Parameters

  command-line option: `-x (string of parameters)` configuration file option: `parameters = string of parameters`

This option specifies some command line parameters that are passed directly to the module executable. These parameters are not interpreted by *cctp* and are treated as a character string, so you must write these parameters exactly as you want that they will appear after the module executables. The parameters string must be enclosed between brackets in the command line option (please note the space between the x and the open bracket). In the *ini* configuration file, simply write the parameters string after the equal sign "=".

Receive

command-line option: `-receive`

configuration file option: `op_mode = receive`

This operation mode can be set with the option `-receive` or by placing the line `op_mode = receive` in the *cctp.ini* file.
In this mode *cctp* only download the output file(s). It returns the prompt only when the file(s) is successfully downloaded or when some errors occur.

In this operation mode the following options are compulsory:

- User_ID
  command-line option: `-u your_user_id`
  configuration file option: `user_id = your_user_id`

  This option sets the *User_ID* that you received after the registration process on VALab server. `your_user_id` is a number between 1 and 99999999.

- Process_ID
  command-line option: `-p process_id`
  configuration file option: `process_id = chosen_process_id`

  This option sets the Process_ID. `process_id` must be a number between 1 and 99999999. The Process_ID, together with the User_ID, is a number that entirely identifies on server the running process you requested. Specify the Process_ID number you used before running the remote execution process with *cctp* in *send* mode.

The following is optional:

- Module Name
  command-line option: `-m chosen_module_name`
  configuration file option: `module_name = chosen_module_name`

This option sets the Module Name that you have executed by remote execution. `chosen_module_name` is the name provided in the web site module instructions. This option is not necessary in *receive* operation mode, because your process is already entirely identified by User_ID and Process_ID, but, if specified, it will provide a further control on your executed processes.

Status

command-line option: `-status`

configuration file option: `op_mode = status`

This operation mode can be set with the option `-status` or by placing the line `op_mode = status` in the *cctp.ini* file.

In this mode *cctp* simply shows a screen containing the status of the users' modules that are currently running. It returns the prompt immediately or when some errors occur.

In this operation mode the following options are compulsory:

- User_ID

  command-line option: `-u your_user_id`

  configuration file option: `user_id = your_user_id`

  This option set the *User_ID* that you received after the registration process on VALab server. `your_user_id` is a number between 1 and 99999999.

Clear

command-line option: `-clear`

configuration file option: `op_mode = clear`

This operation mode can be set with the option `-clear` or by placing the line `op_mode = clear` in the *cctp.ini* file.

After that you used the status operation mode to visualize the processes that are running on server as well the processes that are crashed, you can use this operation mode to remove from the server a particular module. It can be useful if the remote execution will take to much time, or if a previously requested elaboration is no more needed.

It returns the prompt only when the process are successfully removed or when some errors occur.

In this operation mode the following option is compulsory:

- User_ID

  command-line option: `-u your_user_id`

configuration file option: `user_id = your_user_id`

This option sets the *User_ID* that you received after the registration process on VALab server. `your_user_id` is a number between 1 and 99999999.

- Process_ID

  command-line option: `-p process_id`

  configuration file option: `process_id = chosen_process_id`

  This option sets the Process_ID. `process_id` must be a number between 1 and 99999999. The Process_ID, together with the User_ID, is a number that entirely identifies on server the process you wish to remove. Specify the Process_ID number of the process you wish to remove. As you have to specify the Process_ID you can remove only a process at time. To remove more processes look at the Reset operation mode.

Reset

command-line option: `-reset`

configuration file option: `op_mode = reset`

This operation mode can be set with the option `-reset` or by placing the line `op_mode = reset` in the *cctp.ini* file.

In this mode *cctp* reset the user status to the default initial values, all user information as well as all pending jobs and/or running modules on the server are reset. Note that all user data will be lost. Since all processes will be cancelled regardless their state, this mode can resolve most of critical situations (e.g. the server is not responding correctly to your requests), but should be used only in case of emergency, in any other case it is better to use the *Clear* mode.

In this operation mode the following option is compulsory:

- User_ID

  command-line option: `-u your_user_id`

  configuration file option: `user_id = your_user_id`

  This option set the *User_ID* that you received after the registration process on VALab server. `your_user_id` is a number between 1 and 99999999.

3.3.2. Remote Execution service on server side. Sharing sensitive modules from partner.

Mainly the modules reside physically on VALab server; however other universities or institutes may want to put at Remote Execution disposal their executables without put on VALab server a copy of their codes. For this purpose our group developed a system to execute the module directly on partner's server in a completely transparent way for the final user.

As mentioned before, we provide a program (***cctp***) which allows the user to send to the VALab server the input file(s), containing the data that a particular module requires, and to receive the output file(s), containing the results. If the requested modules does not reside on VALab server, the latter provides automatically to forward all the data to partner's server. Note that, as the remote access will always pass through VALab, the user will never contact partner's server directly and will never know the location of a module executed on such server. The method used to exchange the files and information is electronic mail.

If a partner wants to share one or more of their executables, we provide all necessary files, instructions, full help and assistance to configure properly their server. In the following section the basic instruction that we usually provide to our partners are reported.

3.3.3.How to install the VALab remote execution robot on partner's linux server.

1. Prerequisites:

   1.1. Perl must be installed. At least version 5.8.4 it's necessary.

   1.2. It's necessary to have a mail server installed on your server. VALab server uses SendMail and we suggest to use the same mail server (www.sendmail.org). Actually, the only strictly necessary feature that the mail server should have, is the support to the `$HOME/.forward` standard (also known as *dot-forward* or *dotforward*). See Appendix 1 for more information about SendMail.

   1.3. VALab server uses also *smrsh*. *smrsh* is a restricted shell utility that provides the ability to specify, through a configuration, an explicit list of executable programs. When used in conjunction with SendMail, *smrsh* effectively limits SendMail's scope of program execution to only those programs specified in *smrsh*'s configuration. The purpose for restricting the list of programs that can be executed in this manner is to keep mail messages (either through an alias or the .forward file in a user's home directory) from being sent to arbitrary programs which are not necessarily known to be sufficiently paranoid in checking their input, and can therefore be easily subverted. See Appendices 2 and 3 for further information on *smrsh* and security issues. We strongly recommend to use *smrsh* and to place the provided *robot.pl* code in the folder */etc/smrsh/*. This precaution will eliminate any security problem related to the remote execution.

2. Extract in a temporary directory of your choice the provided archive *Valab-Remote-Execution-1.0.tar.gz*:

```
tar -xzvf Valab-Remote-Execution-1.0.tar.gz
```

   A *Valab-Remote-Execution* directory will be created. Move into the directory:

```
cd ./Valab-Remote-Execution
```

   If you list the content of the directory you can find the following files and directories:

   - readme – this file.
   - Mail-SendEasy-1.2.tar.gz – compressed file archive containing the Mail::SendEasy perl module.
   - robot – directory containing the main perl executable.
   - dot-forward – directory containing a .forward example file.
   - prog_data – directory containing the data configuration program.

3. Install the provided perl module Mail::SendEasy.

```
(starting from the Valab-Remote-Execution directory)
```

```
tar -xzvf Mail-SendEasy-1.2.tar.gz
cd ./Mail-SendEasy-1.2
perl Makefile.PL
make
make test
make install
```

4. Copy the perl program *robot.pl* in the right directory.

```
(starting from the Valab-Remote-Execution directory)
cd ./robot
cp robot.pl /etc/smrsh/
cd /etc/smrsh/
chown root:root robot.pl
chmod +x robot.pl
```

   If you don't use the smrsh restricted shell, you can place the *robot.pl* file in any other folder you choose instead of */etc/smrsh/*.

5. Create a user called "robot".

6. Create (if it doesn't still exist) the *.forward* file in the robot home directory (*/home/robot/.forward*). Edit the *.forward* file and write the following line (with double quotes).

```
"|exec /etc/smrsh/robot.pl"
(leave a blank line)
```

   Note the blank line to be left at the end. Obviously, if you copied the *robot.pl* file in a different directory, specify that instead of */etc/smrsh/*. A copy of this file can be found in the *dot-forward* directory.

   Note: Actually, if you have the *smrsh* installed, you can just write `"|exec robot.pl"` as the entire path (if any), before the executable file name, will be automatically ignored and substituted with */etc/smrsh/*.

7. Create a directory where the remote execution temporary files will be placed. For example:

```
mkdir /home/robot/tmp/
```

8. Create a directory where the remote execution working directories will be placed. For example:

```
mkdir /home/robot/exec/
```

9. Create a directory where the remote execution log files will be placed. For example:

```
mkdir /home/robot/logs/
```

10. Now, you have to create the *prog.data* file, containing all the necessary information about the modules that you want to make available on your server.

    10.1. Copy in a directory (e.g. */home/robot/prog_data/* can be a good choice) the files *prog_data.pl* and *prog_data.ini.*

    ```
    (starting from the Valab-Remote-Execution directory)
    cd ./prog_data
    cp prog_data* /home/robot/prog_data/
    cd /home/robot/prog_data/
    chmod +x prog_data.pl
    ```

    10.2. Edit the file *progs_data.ini*. For each program you want make available, you have to write some information (hereafter called section). The following is an example of a section:

    ```
    module_name        = mie_robot
    program_name       = mie_homogen_sphere
    path               = /home/robot/progs/mie_homogen_sphere
    input_files        = input1.in  ,  input2.in
    output_files       = output.out
    default_parameters =
    error_file         = mie_homogen_sphere.err
    help_file          =
    description        =
    ```

    - *module_name* is the name that the user will specify to select the execution of the module.
    - *program_name* is the name of the called program. It is your internal name, the user will never see this name.
    - *path* is the path where is located your executable.
    - *input_file* is the list of the input files required by the program. The names are separated by commas. If the program doesn't have fixed input file names, but, for example, they will be specified by the user in the command line, simply place a * in place of names.
    - *output_file* is the list of the output files required by the program. The names are separated by commas. If the program doesn't have fixed output file names, but, for example, they will be specified by the user in the command line, simply place a * in place of names.
    - *default_parameters* is an optional string of default command line parameters passed to the executables. The parameters will be passed exactly as they appear after the =.
    - *error_file* is the name of the optional file that contains the errors code explanation of your program. If the file is specified and an error occurs, robot will show the

error code number and will search in the file for the associated text string (that usually give an extended explanation of the error). If no file is specified, in case of error, robot will show only the error code number.

- *help_file* is the name of an optional file that contains some help information that will be replied to the user in case of errors in the input parameters (command line and/or input files).
- *description* is an optional brief description of the program.

Note that all the nine fields of a section are required (not necessary in the given order). Note also that the only fields that can be empty are *default_parameters*, *error_file*, *help_file* and *description*. All the other are required. Remember that if input or output files are not required you must place a * in place of file names.

10.3. Execute the *prog_data.pl* program.

```
cd /home/robot/prog_data/
./prog_data.pl
```

10.4. Make sure that the file *prog.data* has been properly created.

```
cd /home/robot/prog_data/
ls prog.data
```

10.5. Make sure that there are no errors, that is the error file is empty.

```
cd /home/robot/prog_data/
cat ./error_file.log
```

At last, send a copy of *prog_data.ini* to valab@valab.det.unifi.it, we need some information on your programs in order to properly configure VALab server.

11. Finally, edit the file *robot.pl* that you have previously copied in the */etc/smrsh/* folder. Modify the following section at the beginning of the program, inserting the information on your chosen paths.

```
############################################################
#
#                configuration
#
############################################################
#
#          configure these parameters
#

my $log_path     = "/home/robot/logs";
my $exec_path    = "/home/robot/exec";
```

```
my $temp_path     = "/home/robot/tmp";
my $data_path     = "/home/robot/prog_data";

my $SMTP_Server    = "your_smtp_server";
my $SMTP_timeout   = 120;

#
#            end of configuration
#
############################################################
```

If you used the directory proposed in the previous examples, you need only to place the name of your SMPT server in the place of *"your_smtp_server"* (note that the double quotes are necessary).

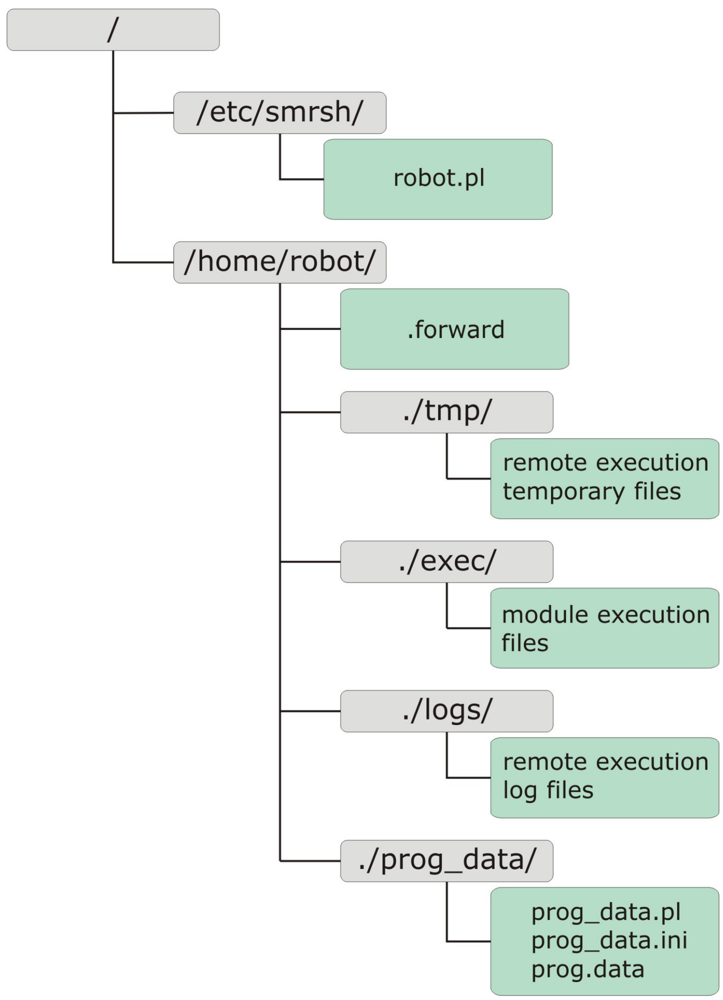


Figure 15 - Location of required files and directories.

3.3.4. Practical example

In this section we give a practical example of cctp usage. We used the client program to obtain some results from the "Mie Homogen Sphere" module available on VALab server.

The *cctp* configuration file (Figure 16) has been prepared accordingly to the instructions contained in the *cctp* User Manual and in the Mie Homogen Sphere description page. Note the differences between the parameters specified in the input_file and output_file field and that ones specified in the parameters field. The former are necessary to the *cctp* program to set which files should be sent to VALab server and which file should be downloaded when the remote execution is finished. The latter are the parameters passed directly to the module executable, in this case these configure the remote program to expect an input file named "input_file.in", and to create an output file named "output_file.out".

```
# ------------------------------------------------
# --------- cctp.exe Configuration File ---------
# ------------------------------------------------
#
# This configuration file contains all the parameters
# that you can set for the program.
#
# In the following there is a list of such parameters
# with the default value specified and the command line
# option to set the parameter from the console prompt.
#
# op_mode       =                             # -sr -send -receive -status -clear -reset
# module_name   =                             # -m
# user_id       =                             # -u
# process_id    =                             # -p
# UserName      =                             #
# Password      =                             #
# input_file    =                             # -i
# output_file   =                             # -o
# parameters    =                             # -x
# wait_time     = 5                           # -w
#
#
# ----------------- parameters ------------------
#

op_mode       = sr
module_name   = mie
user_id       = 12345678
process_id    = 001
Username      = example
Password      = exapwd
input_file    = input_mie.in
output_file   = output_mie.out
parameters    = -i input_mie.in -o output_mie.out
wait_time     =
```

Figure 16 - The cctp.ini configuration file used for the current example.

Then the input file (Figure 17) for the module "mie" has been properly compiled following the instructions and the example in the Mie Homogen Sphere description page.

```
# the fields are formatted as:
#  label [space] value
#

radius  1000.0                      (radius of the sphere [m] [required])

# At least one of them must be specified
 frequency  1.E6                    (frequency used for the analysis [Hz] [Optional])
# lambda                            (lambda used for the analysis [m] [Optional])


# Or "Complex_refractive_index" or both "Complex_electric_permettivity" and "Complex_magnetic_permeability"
Complex_refractive_index  1.42, 0.04    ( real-part, imaginary-part [-] [optional])
# Complex_electric_permettivity  0, 0        ( real-part, imaginary-part [-] [optional - required if Complex_refr
# Complex_magnetic_permeability   0, 0       ( real-part, imaginary-part [-] [optional - required if Complex_refr

 Perfect_conductor    no                 ( yes or no [required])
 Angle_step   0.5                        ( a value between  0.1 <= x <=180 [degrees] [required])
# output_file   "file_di_uscita.out"       ( name of output file [optional - required if not specified in command
```

Figure 17 - The input file for the module *mie*.

Then the *cctp* program is simply invoked from a command line window (Figure 18).

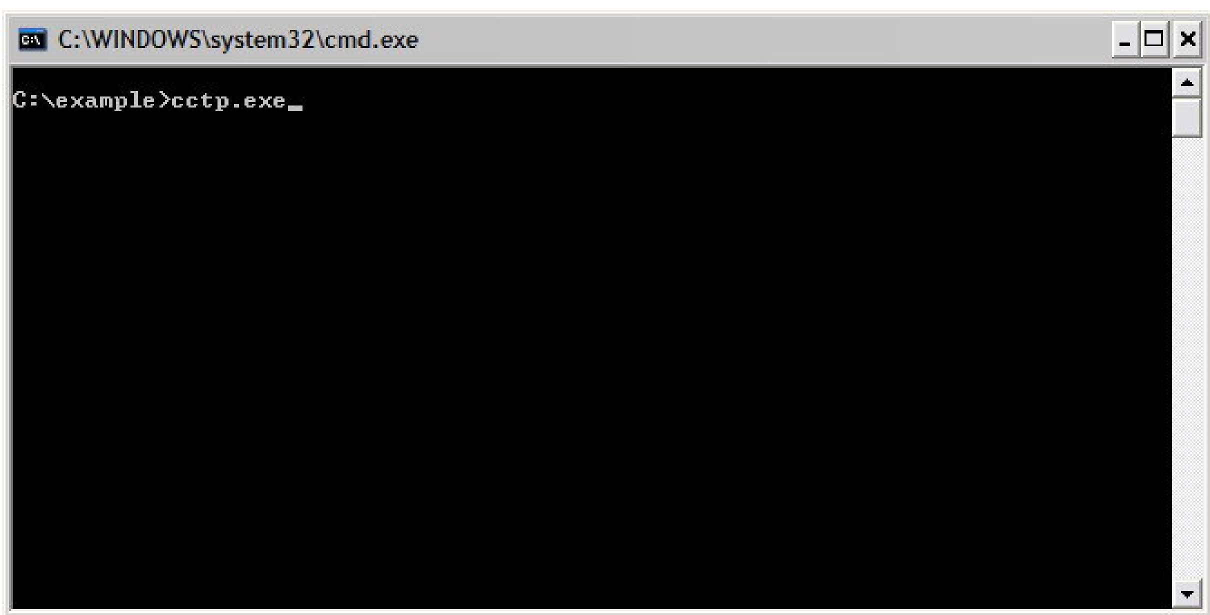


Figure 18 - Launching the *cctp* program from the command line.

The *cctp* program immediately connects to VALab server (Figure 19), request the selected module execution and provide to send all the necessary input files.

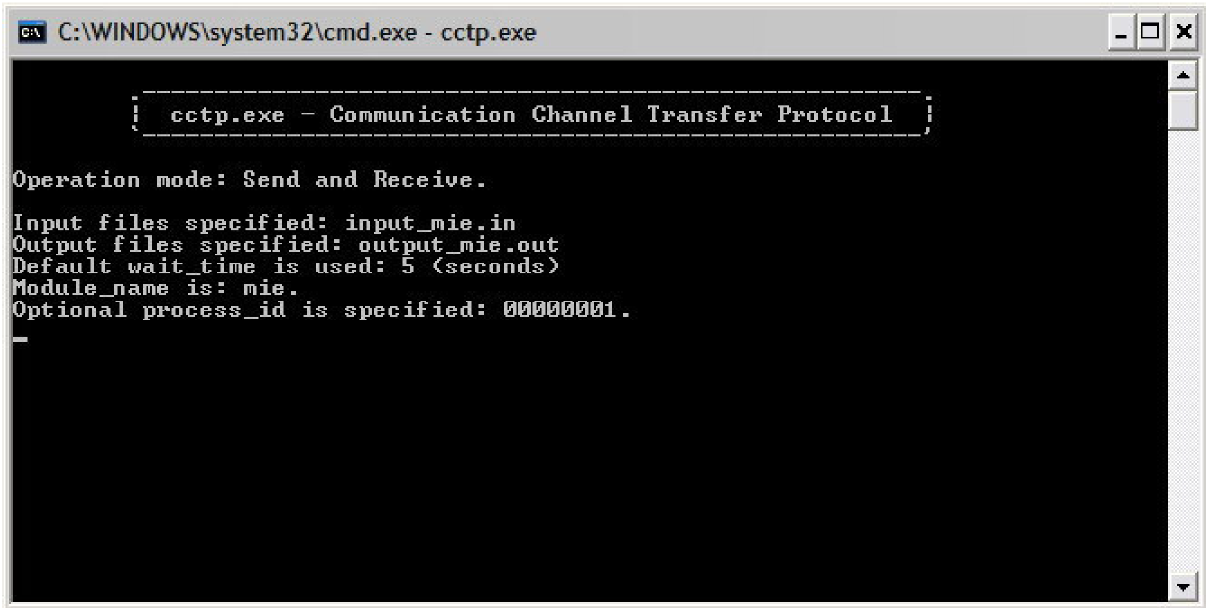


Figure 19 - *cctp* program during the connection to VALab server.

Then when the remote execution on the VALab server is terminated, the program *cctp* provides automatically to retrieve the output files following the user settings instructions. When the download is finished it returns to the prompt (Figure 20).

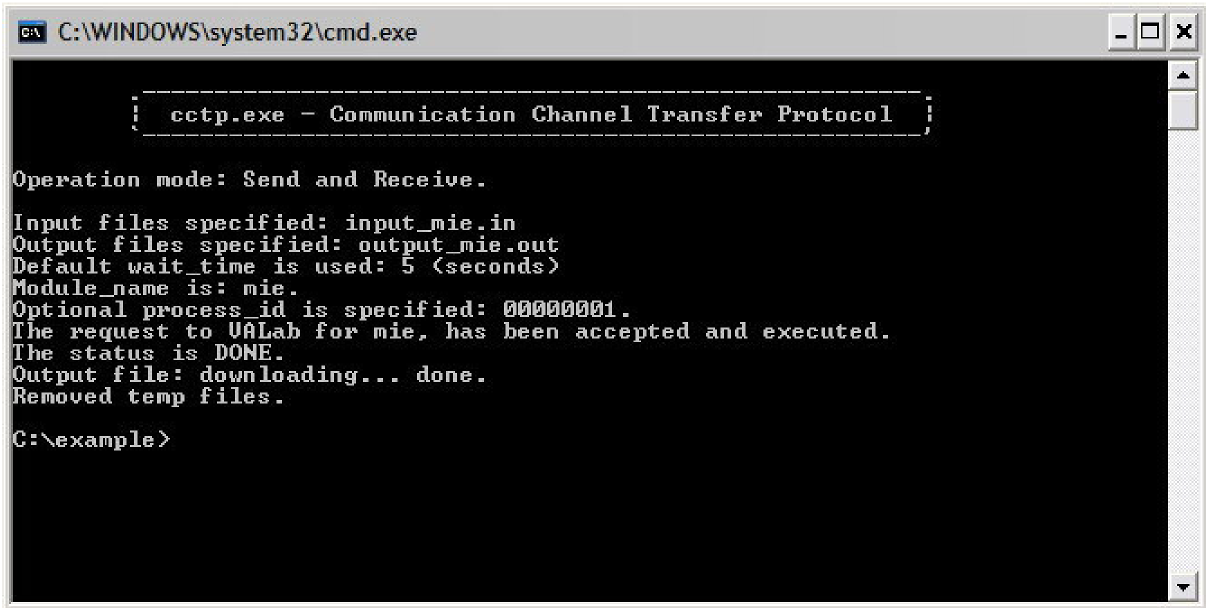


Figure 20 - *cctp* program after it successfully downloaded the output files.

In the following figure (Figure 21) a trim of the output file generated by "mie" module can be observed. It contains the data that, for example, can be useful for the user to compare with his results.

```
********************************************************************************
Dielectric sphere, Refractive index:  real    1.420  imag   4.000E-02,  Mie size parameter =    20.958
                              Calculation for 362angles equally spaced between 0 and 180 degrees.
```

Angle	Cosine	S-sub-1		S-sub-2		Intensity	Deg of Polzn
0.00	1.000000	2.54536E+02	2.17950E+01	2.54536E+02	2.17950E+01	6.52634E+04	0.0000
0.50	0.999962	2.53363E+02	2.15923E+01	2.53361E+02	2.16786E+01	6.46604E+04	0.0000
1.00	0.999849	2.49863E+02	2.09898E+01	2.49858E+02	2.13320E+01	6.28781E+04	0.0001
1.50	0.999659	2.44105E+02	2.00050E+01	2.44094E+02	2.07630E+01	6.00002E+04	0.0002
1.99	0.999394	2.36193E+02	1.86657E+01	2.36176E+02	1.99840E+01	5.61572E+04	0.0004
2.49	0.999053	2.26279E+02	1.70109E+01	2.26258E+02	1.90128E+01	5.15228E+04	0.0006
2.99	0.998637	2.14543E+02	1.50873E+01	2.14522E+02	1.78707E+01	4.62978E+04	0.0009
3.49	0.998145	2.01199E+02	1.29494E+01	2.01185E+02	1.65827E+01	4.06996E+04	0.0012
3.99	0.997578	1.86493E+02	1.06585E+01	1.86494E+02	1.51776E+01	3.49518E+04	0.0017
4.49	0.996934	1.70682E+02	8.27844E+00	1.70710E+02	1.36856E+01	2.92650E+04	0.0022
4.99	0.996216	1.54048E+02	5.87652E+00	1.54116E+02	1.21396E+01	2.38323E+04	0.0028
5.48	0.995422	1.36873E+02	3.51905E+00	1.36998E+02	1.05723E+01	1.88135E+04	0.0035
5.98	0.994552	1.19449E+02	1.27126E+00	1.19646E+02	9.01743E+00	1.43331E+04	0.0044
6.48	0.993607	1.02056E+02	-8.05990E-01	1.02343E+02	7.50716E+00	1.04733E+04	0.0055
6.98	0.992587	8.49692E+01	-2.65717E+00	8.53625E+01	6.07254E+00	7.27523E+03	0.0067
7.48	0.991492	6.84466E+01	-4.23385E+00	6.89616E+01	4.74231E+00	4.74052E+03	0.0079
7.98	0.990322	5.27216E+01	-5.49656E+00	5.33716E+01	3.54195E+00	2.83543E+03	0.0090
8.48	0.989076	3.80058E+01	-6.41524E+00	3.88010E+01	2.49358E+00	1.49866E+03	0.0087
8.98	0.987756	2.44788E+01	-6.97045E+00	2.54255E+01	1.61505E+00	6.48431E+02	0.0010
9.47	0.986361	1.22871E+01	-7.15386E+00	1.33871E+01	9.19600E-01	1.91105E+02	-0.0578
9.97	0.984892	1.54253E+00	-6.96822E+00	2.79224E+00	4.15604E-01	2.94524E+01	-0.7294

Figure 21 - A trim of the output file generated by *mie* module.

All the parameters specified in the *cctp.ini* configuration file can be also specified directly in the command line (Figure 22). Note that there are two parameters that cannot be specified by this way (i.e. Username and Password). For this reason the *cctp.ini* file is required in any case. Such command line call can be effectively used in a SYSTEM call from within a program.

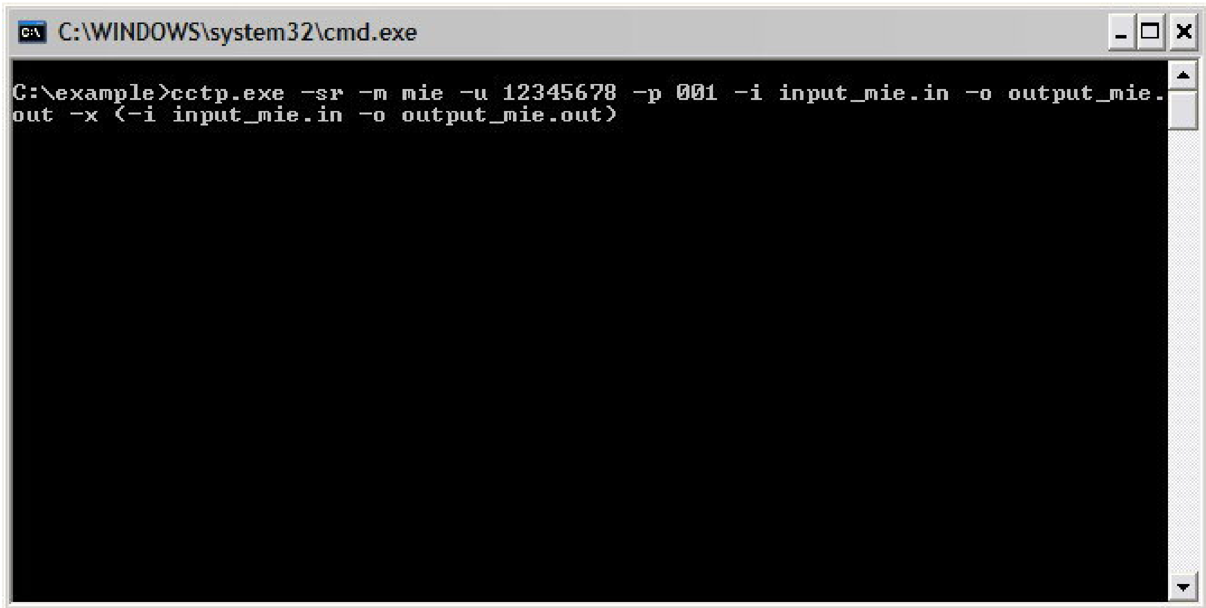


Figure 22 - Specification of the input parameters directly in the *cctp* command line.

If the user desires to control the status of his pending processes, it is enough to launch the *cctp* program with the `-status` option. The program will return a screen (Figure 23) with all the details for each process currently present on the VALab server.

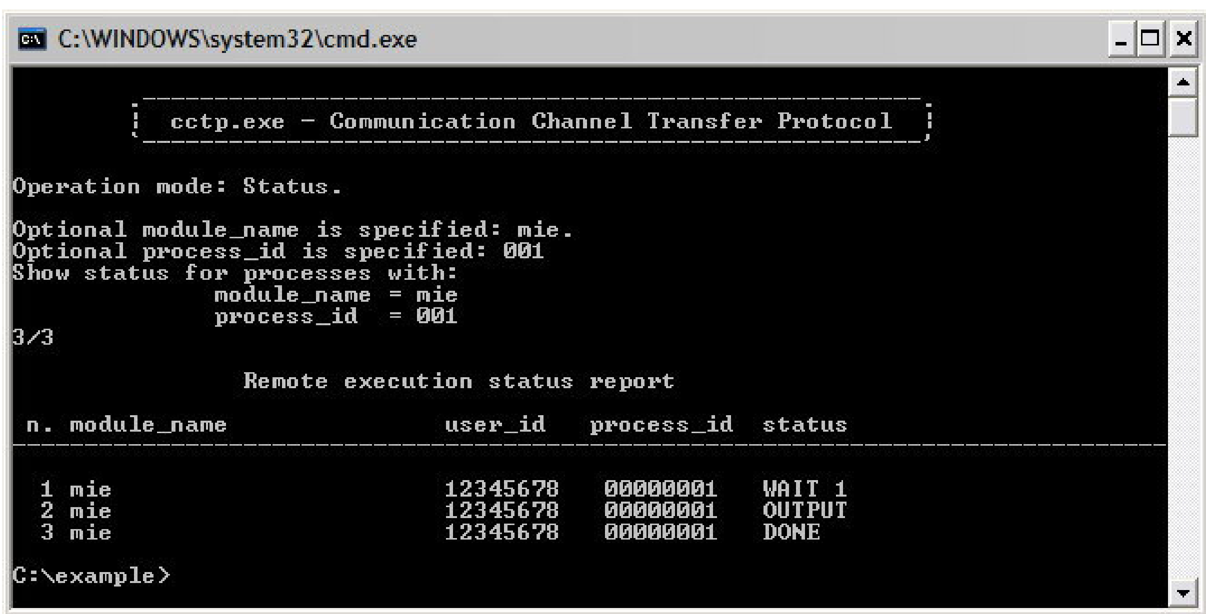


Figure 23 - Remote Execution status screen.

4. European School

As support to the European School of Antennas, for each course of the school is also available on VALab a page where the students can download the course material (transparences, notes, software, etc.). Each page is protected and only the teachers and the registered students can access the course page. Figure 24 shows, as an example, the page relevant to the "Computational EM for Antenna Analysis" course held in Turin on 19-23 September 2005.

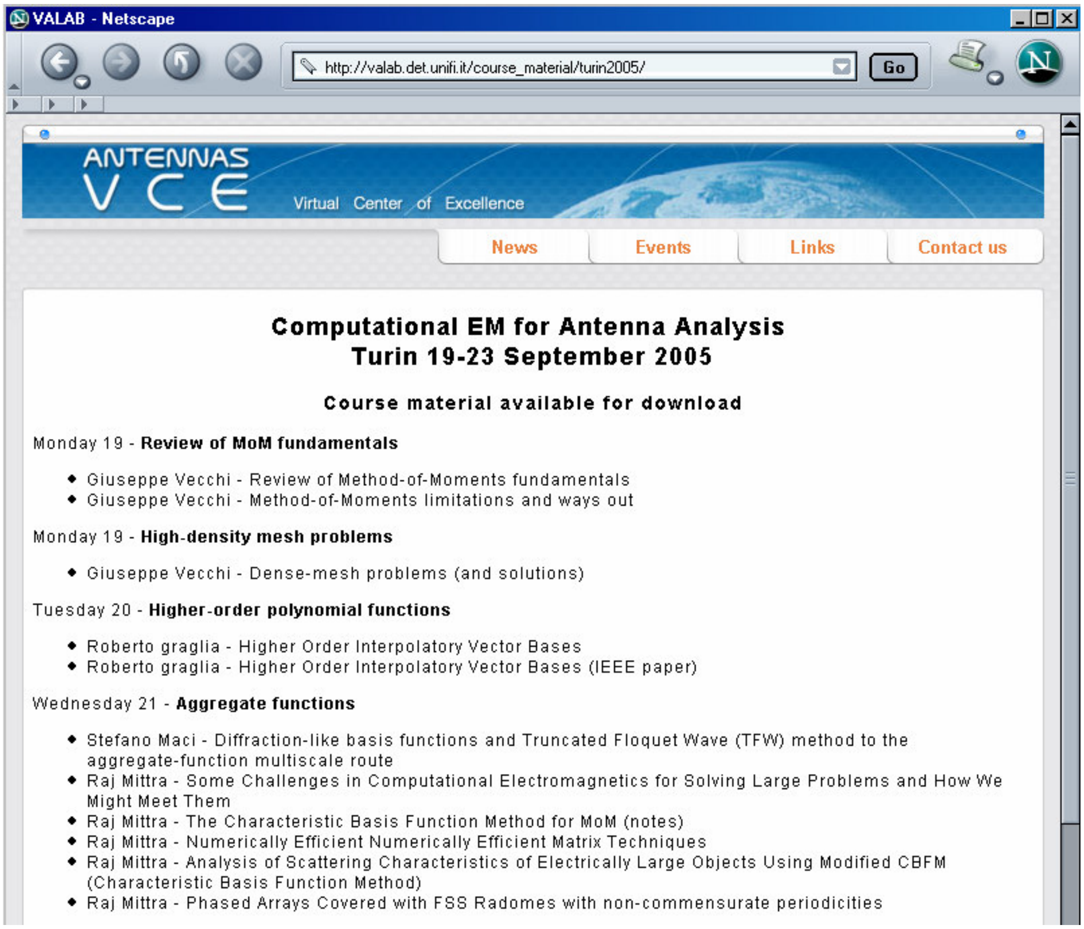


Figure 24 – Course material of the Computational EM for Antenna Analysis, Turin 19-23 September 2005

Each page is subdivided in sections (in the example of Figure 24 they are: Monday 19 – Review of MoM fundamentals, Monday 19 – High-density mesh problems, ...) and each section is further subdivided in paragraphs, describing the material that will be download by clicking on. Each paragraph starts with the symbol '♦'. When the selected material is a PDF file a new page is open and the file is loaded into (see Figure 25).

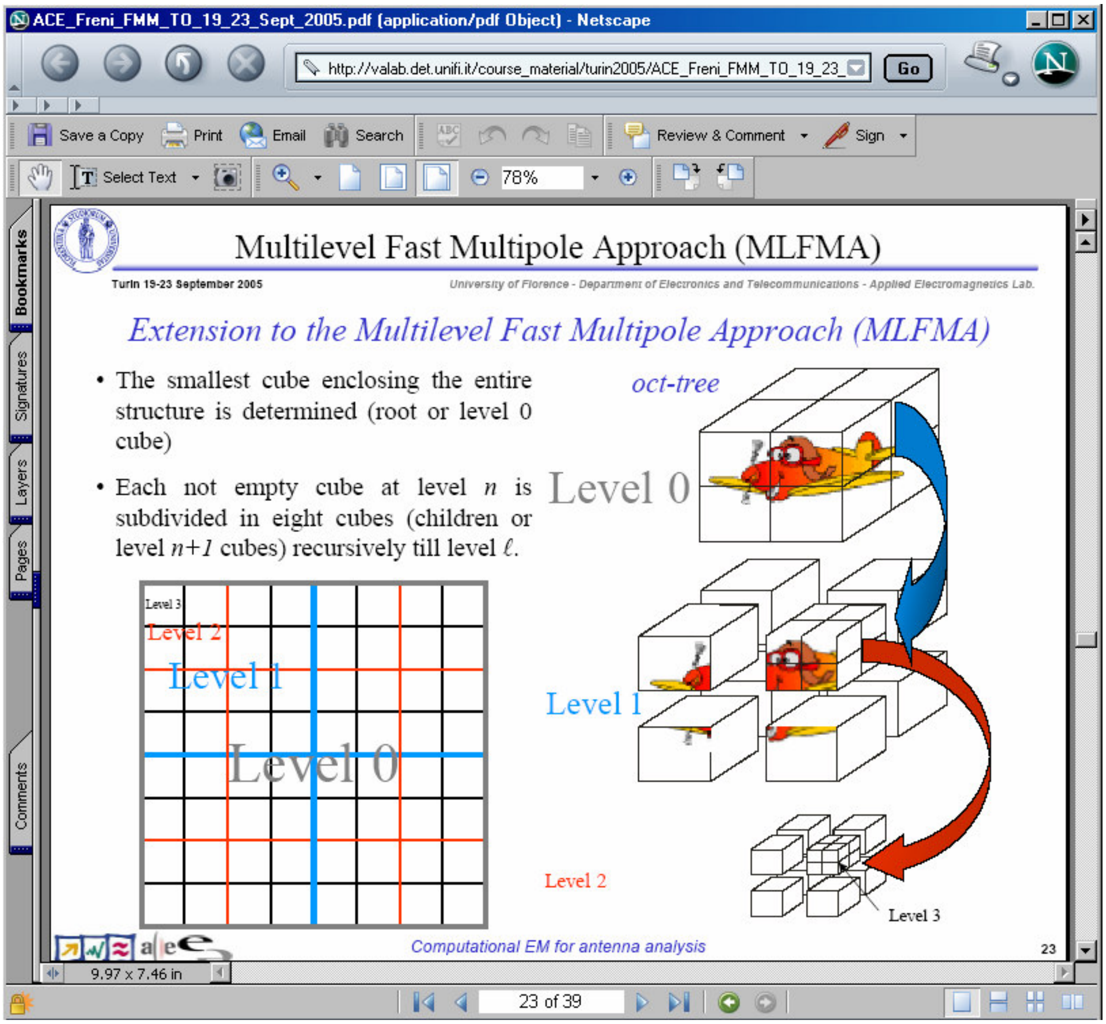


Figure 25 – Example of PDF file.

In order to do not involve the course coordinator in web stuff, a simple process has been implemented to automatically upgrade the course material page when a new file is copied in the relevant directory on VALab. In particular, the course coordinator it is not required to know how to create an html page but simply write two text files describing the structure of the page.

First the course coordinator has to edit and copy in the course material directory a text file called *master.index* containing the scheme of the course page (a template of this file will be available into the course directory after the course is scheduled in the WP3.1 activity). This *master.index* file is divided in two sections: the course title and the course contents.

- *course title* – the course coordinator has to specify the course title in between the tags <title> and </title> . A
 tag has to be specified in case of more than one title line is required.
- *course contents* – in between the tags <contents> and </contents> the course coordinator has to specify the sections constituent the page. They consist in an arbitrary number of fields separated

by the ';' symbol. Each field can span more that one text line and is consisting in two sub-fields separated by the '$' symbol. The first is the section description, the second is the logical name associated to this section. This logical name is mandatory and is needed to refer the course material to the correct section.

Furthermore, some comments can be included by starting any new line by the '#' symbol. Hereunder, we report an example of the *master.index* file

```
# --- master.index
# Course title
<title>
Computational EM for Antenna Analysis <br>
Turin 19-23 September 2005
</title>
# Course contents
<contents>
# Section description                          $ logical name
Monday 19 - Review of MoM fundamentals         $ MoM;
Monday 19 - High-density mesh problems         $ mesh;
Tuesday 20 - Higher-order polynomial functions $ poly;
Wednesday 21 - Aggregate functions             $ aggregate;
Thursday 22 - Fast methods                     $ fast;
Friday 23 - MRMoM for printed antennas         $ MRMoM;
Friday 23 - MRMoM for 3D antennas              $ MRMoM3D;
</contents>
# --- end of master.index
```

Then, the course coordinator can start to fill in the page with the course material. It is worth noting that, it is not required to upload all the course material at the same time. We can upgrade each sections separately and several time. In particular, each time that one or more new files are upload in the course directory it is required to upload also one or more *.txt* files containing the description of these files. The *.txt* files will be automatically collected by the system, read and interpreted. Each *.txt* file consists in an arbitrary number of fields separated by the ';' symbol. Each field can span more that one text line and is consisting in the following four fields separated by the '$' symbol:

1. logical name of the section where the description has to appear;
2. order number of the paragraph in the section (the descriptions will be sorted according this order number);
3. description of the material
4. name of the file that will be download when one clicks on the description.

Comments can be included by starting any new line by the '#' symbol.

Suppose that the we would like to upgrade the section "aggregate functions" with the following files:

- ACE_Maci_TFW.pdf;
- mittra_cbmom_intro.pdf;
- mittra_cbmom_part1.pdf.

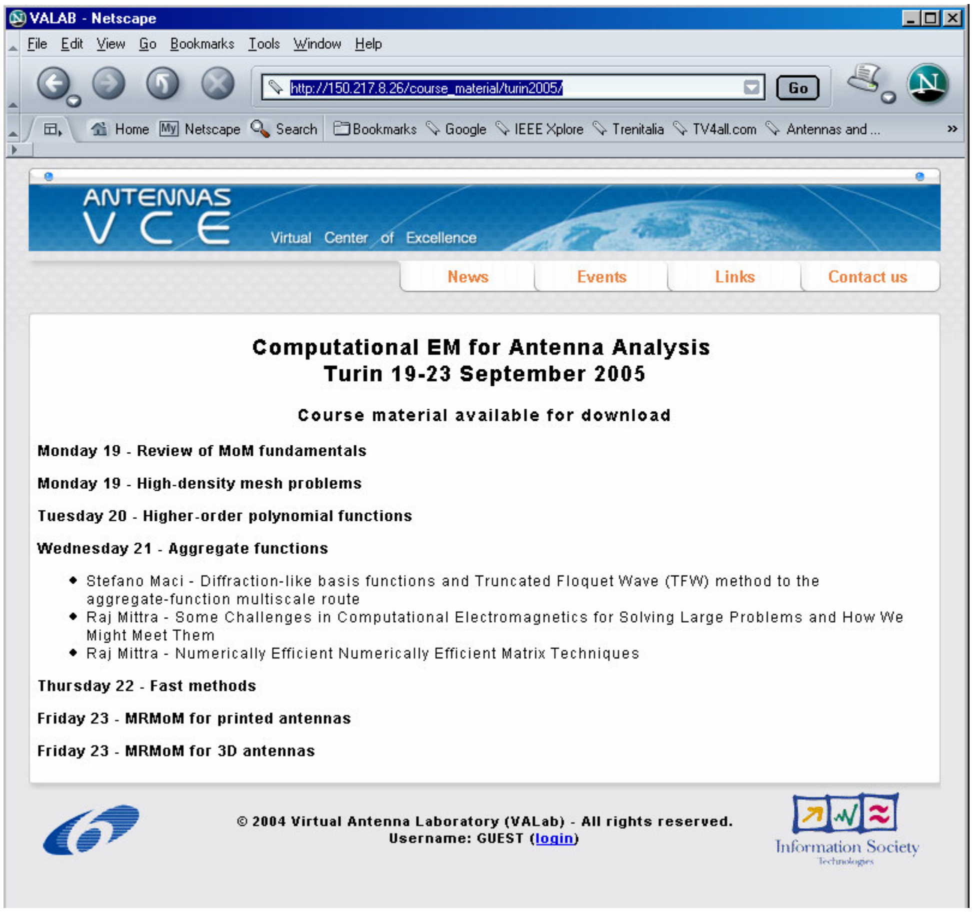


Figure 26 - Page course just after the Wednesday.txt file has been uploaded.

First, we can upload the .pdf files in the course directory, and then we need to create a related *.txt* file. The name of the .txt file is not important and can be chosen at will. In this example we choose 'Wednesday.txt.' Its contents will be:

```
# --- Wednesday.txt
# Aggregate functions
# logical name $ order $ description $ file name;
# Prof. S.Maci
  aggregate    $  100  $ Stefano Maci - Diffraction-like basis
functions and Truncated Floquet Wave (TFW) method to the aggregate-
function multiscale route
$ ACE_Maci_TFW.pdf;
# Prof. Raj Mittra
  aggregate    $  200  $ Raj Mittra - Some Challenges in Computational
Electromagnetics for Solving Large Problems and How We Might Meet Them
$ mittra_cbmom_intro.pdf;
  aggregate    $  300  $ Raj Mittra - Numerically Efficient Numerically
Efficient Matrix Techniques
$ mittra_cbmom_part1.pdf;
# --- end of Wednesday.txt
```

Just after we have uploaded the Wednesday.txt file in the course directory the course page results as in Figure 26. Suppose now to wish add the file ace_notes_cbfm.pdf containing some notes. In particular, we would like to put these notes just in between of "Raj Mittra - Some Challenges in Computational Electromagnetics for Solving Large Problems and How We Might Meet Them" (order=200) and "Raj Mittra - Numerically Efficient Numerically Efficient Matrix Techniques" (order=300). To do this, we can opportunely modify the previous Wednesday.txt or create a new *.txt* file (for example Wednesday_notes.txt) as following

```
# --- Wednesday_notes.txt
# Raj Mittra notes on Characteristic Basis Function Method for MoM
# logical name $ order $ description $ file name;
  aggregate    $  250  $ Raj Mittra – The Characteristic Basis Function
Method for MoM (notes)
$ ace_notes_cbfm.pdf;
# --- end of Wednesday_notes.txt
```

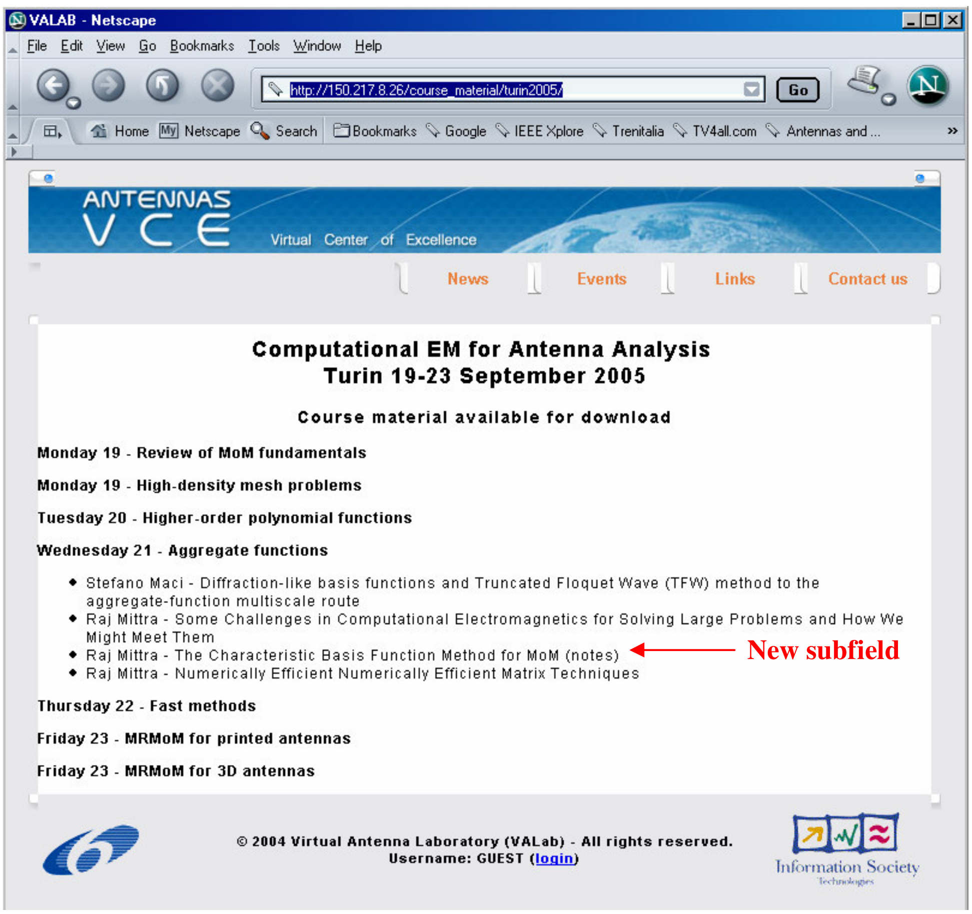


Figure 27 - Page course just after the Wednesday.txt file has been uploaded.

Just after we have uploaded the Wednesday_notes.txt file in the course directory the course page results as in Figure 27. It is worth noting that we have chosen an order number equal to 250 but any order number between 201 and 299 does the same work.

In case is present a paragraph but the corresponding file is missing in the course directory, the notice "(not available)" will be added just at the end of the material description (see note (1) in

Figure 28). It is worth noting that if more than paragraph of a section has the same order number a warning message will appear at the beginning of the web page, but no messages appear if a section does not have any subfield (see note (2) in Figure 28).

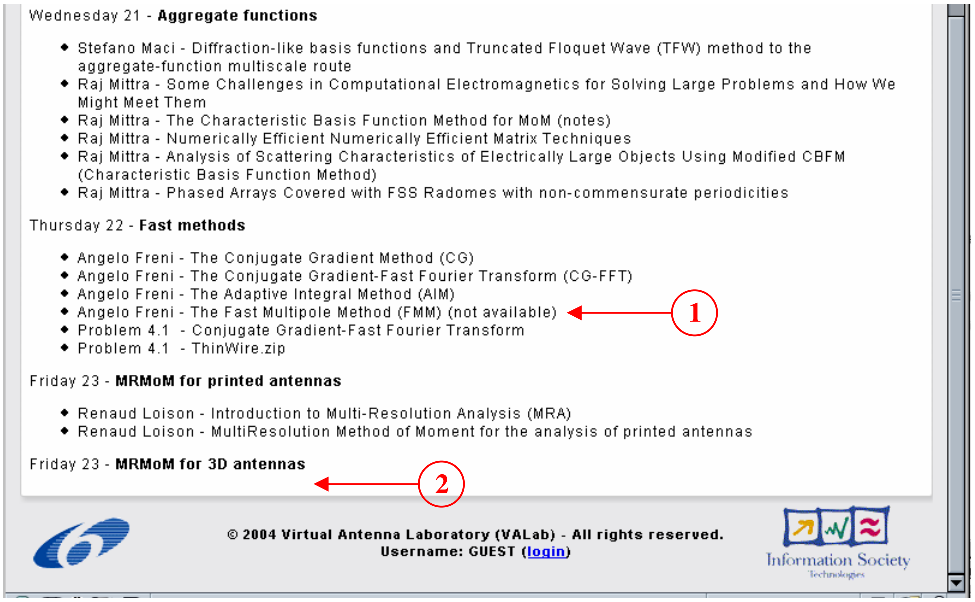


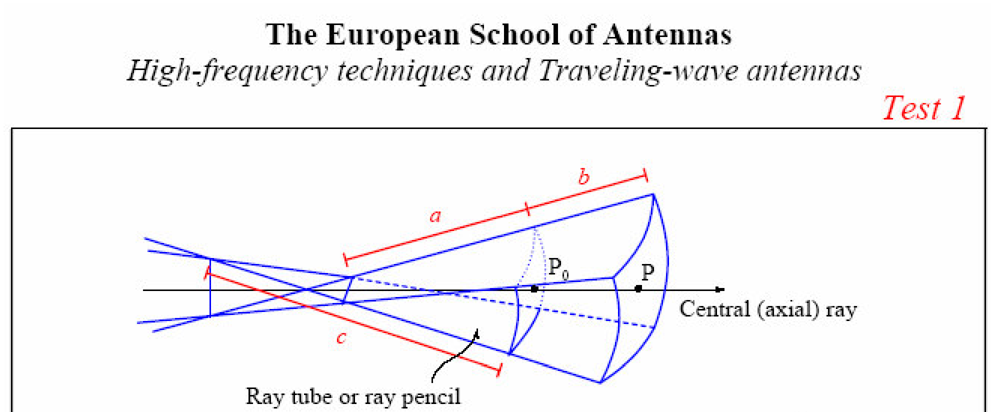
Figure 28 – Example of page course

5. Training

In this section we describe how a teacher can easily create a training page that the students can use to check their background before to join a course, and/or verify their knowledge before to undertake the course final exam.

A training page is essentially a web page containing a list of multiple-answers questions. The student marks the answers that he thinks correct and submits the page. As a result, he obtains back the same web page where for each choice is specified if the answer is correct or not. In case, a simple message can appear around the answer in order to suggest to the student how to solve the question.

Figure 29 and Figure 31 show an example of training page as it appears before its submission. Figure 32 shows an example of training page as it appears after its submission. Figure 33 shows a detail relevant to the case when a message is associated to a particular answer.



1. The below expression represents the general astigmatic ray optical field $\vec{E}(P)$ at P in terms of its value at some earlier point P_0 (reference point) on the axial ray

$$\vec{E}(P) \sim \vec{E}(P_0)\sqrt{\frac{\rho_1\rho_2}{(\rho_1+s)(\rho_2+s)}}\,e^{-jks}$$

On the above figure, mark the ray distance s,
○ a, ◉ b, ○ c,
and the caustic (or focal) distances ρ_1 and ρ_2,
○ a and b, ◉ a and c, ○ b and c,
It does not matter which you show as ρ_1 or ρ_2.

2. The caustic distances ρ_1 and ρ_2 are the principal radii of curvature of the reference wavefront curvature at P_0 in the figure shown above?
◉ true ○ false

Figure 30 – Example of training page.

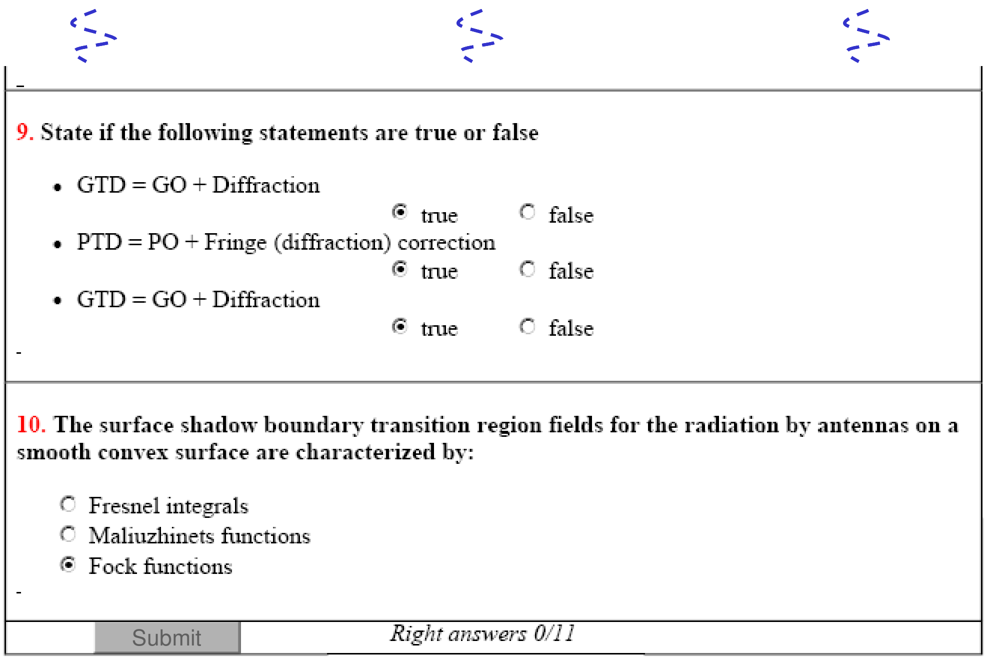


Figure 31 – Example of training page

Usually, to create a training page the teacher has to know HTML, Java script and/or PHP languages. Furthermore, the process to build a web page is long and annoying. To overcome these problems, it has been developed a program, called QUEST, that will help to construct an appropriate PHP page by starting from a simple Word or LaTeX document.

First to describe in detail the capabilities of the program QUEST we would like to introduce the basic idea.

Several programs that can translate a scientific Word or LaTeX document in an HTML document are available (eg LaTeX2HTML for LaTeX and MathType for Word). However, to produce a training page we need to automatically add same radio buttons and a process that can check the correctness of each answer.

In an HTML file the symbol '$' is non processed. This means that if the scientific document contains, for example, the special word *$command$* this string will not be corrupted when we translate the scientific document in an HTML document. So, we can:

1. write with the favourite word processor a document where are planned multiple-answer questions;
2. add to the original document some simple keywords in between two dollars symbols;
3. translate the document into an HTML document;
4. convert the HTML document in the final PHP page substituting the keywords in between the two dollars with appropriate PHP code.

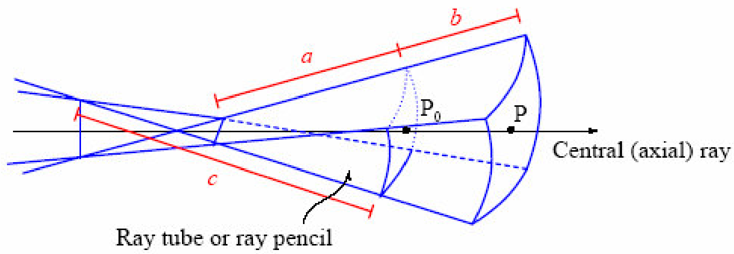
The step 4 is provided by the program QUEST.

Figure 34 and Figure 35 show an example of document containing several keywords that will be interpreted by the QUEST program. Here, all the keywords are in capital letters in order to differentiate them from the regular text, but the QUEST program is case insensitive.

The European School of Antennas

High-frequency techniques and Traveling-wave antennas

Test 1



1. The below expression represents the general astigmatic ray optical field $\vec{E}(P)$ at P in terms of its value at some earlier point P_0 (reference point) on the axial ray

$$\vec{E}(P) \sim \vec{E}(P_0)\sqrt{\frac{\rho_1\rho_2}{(\rho_1+s)(\rho_2+s)}}\,e^{-jks}$$

On the above figure, mark the ray distance s,

○ *a*, ◉ *b*, ○ *c*, ✔ **OK**

and the caustic (or focal) distances ρ_1 and ρ_2,

○ *a* and *b*, ○ *a* and *c*, ◉ *b* and *c*, ✘ **This is not the correct answer**

It does not matter which you show as ρ_1 or ρ_2.

2. The caustic distances ρ_1 and ρ_2 are the principal radii of curvature of the reference wavefront curvature at P_0 in the figure shown above?

◉ true ○ false ✔ **OK**

Figure 32 – Example of training page after its submission

8. May a leaky complex pole in a given grounded dielectric slab come arbitrarily close to the saddle point at $\theta = \pi/2$?

- ○ Yes, at the cutoff frequency of the corresponding mode
- ◉ No, never
- ○ It depends on the dielectric permittivity of the slab
- ○ Yes for TE modes, no for TM modes

✔ **OK** ***(This question was perhaps formulated in a misleading way (all the students gave a wrong answer!). The instructors meant to consider the movement of a leaky (i.e., complex) pole by varying frequency: while a real improper pole approaches the saddle point at $\theta = \pi/2$ as the operating frequency approaches the cutoff frequency, a complex pole always remains at a finite distance from the saddle point)***

Figure 33 – Training page: detail on the message associated to an answer.

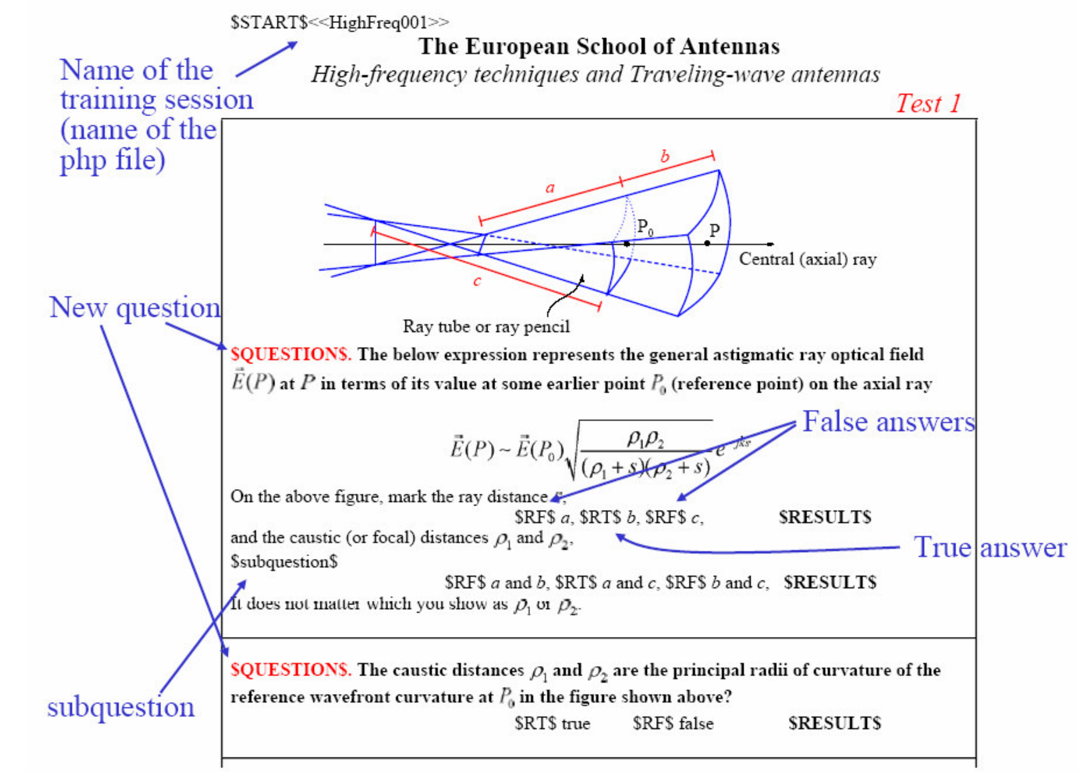


Figure 34 - Example of document containing a few keyword that are interpreted by the QUEST program.

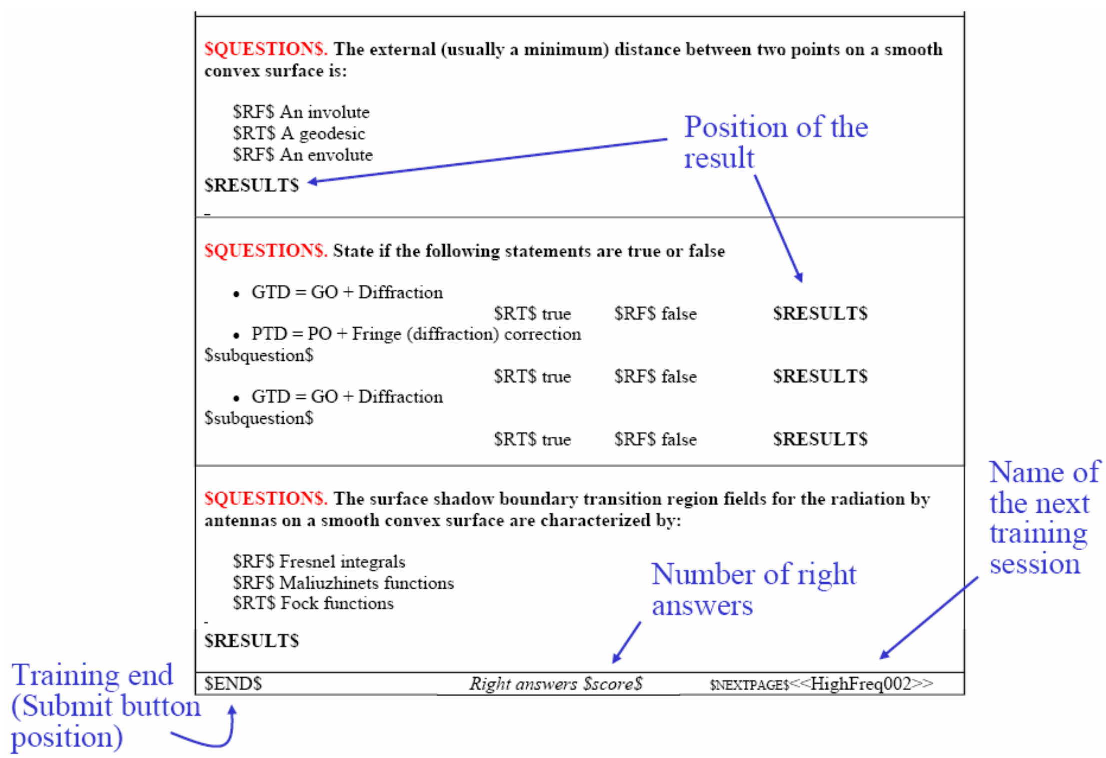


Figure 35 - Example of document containing a few keyword that are interpreted by the QUEST program.

QUEST keywords

As shown in Figure 34, at the beginning of the document is mandatory to insert the keyword $START$ followed by the name of the PHP file (name of the training session) that the QUEST program will create at the end of the process. Then, excluding the keywords $SCORE$ and $NEXTPAGE$, all the other keywords has to be specified in between the $START$ and the END keyword. In particular, the END keyword will be replaced with the '**Submit**' button.

We have planned that the document is divided in several section having a progressive number. Each section can contain one or more multiple-answer questions. Furthermore, for each multiple-answer question only one is the correct answer.

To specify a new section we have to insert the keywords $QUESTION$. This keyword will be replaced by a progressive number *N* and all the following radio button will be referred to the question number N.1. For example, the first time we have the question number 1.1, the second time the question 2.1, and so on. If inside a section we specify the keyword $SUBQUESTION$ all the following radio button will be referred to a question number equal to N.2. Each time the keyword $SUBQUESTION$ is specified into a subsection, the digit on the right is increased by an unit.

To show in the final document a radio button we have to write RF for a false answer and RT for a true one. The number of RF keywords is arbitrary but only a RT keyword must be present.

The keyword $RESULT$ specifies where, after one has submitted the page, has to appear one of the following messages

✔ **OK**

✘ **This is not the correct answer**

We can also associate with a wrong or correct answer a longer message. To do this we can use one of or both the following keywords:

$MESSAGEFALSE$<<message_body>>

$MESSAGETRUE$<<message_body>>

where message_body has to be replaced by the text of the message.

Moreover, two additional keywords can be introduced (see Figure 35 at bottom). The first is $SCORE$ and will be replaced by the number of correct answers. The second is $NEXTPAGE$<<link>>. This last keyword will be replaced by the link specified in between << and >> only if all the answers are correct.