UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Elaborato finale in **Fondamenti di Sistemi Dinamici**

Development of an installation package for the EFIT++ code on Linux distributions

Anno Accademico 2013/2014

Relatore:

Ch.mo Prof.

Alfredo Pironti

Candidato:

Pietro Liguori

matr. N46000088

Index

Introduction

The possibility to use controlled nuclear fusion as a source of alternative energy is one of the main objectives pursued by the scientific community for the benefits offered in terms of availability of raw materials and security.

In a fusion reaction, two light atomic nuclei fuse by clashing together, giving place to a heavier nucleus.

As atomic nuclei are positively charged, they tend to repel each other and therefore, for the fusion to take place, they must have sufficient kinetic energy to overcome the repulsion itself. Such energy can be provided to the nuclei by taking them to very high temperatures ($4x10^7$ K). At such temperatures, all gas atoms are dissociated into their constituents: the nuclei, and the electrons.

The electric charge density is such that the behaviour of the set of particles is totally governed by electromagnetic phenomena. This set of ionized matter subject to electromagnetic actions is defined as *plasma*.

High temperature plasma is unconfinable by material walls: therefore, the confinement device has to be of inertial or of magnetic type. The creation of a magnetic field configuration able to contain the plasma in a stable equilibrium has always represented a major problem in the production of energy by fusion.

The *Tokamak* (Russian acronym for Toroidal Chamber and Magnetic Coil) is a machine produced in the Soviet Union at the end of the sixties for the magnetic confinement of

plasma. It is characterized by a toroidal geometry in which the equilibrium is achieved by inducing an electrical current into the plasma and making it interact with a magnetic field produced by external coils. Furthermore, the current induced into the plasma also provides to its heating by means of the Joule effect.

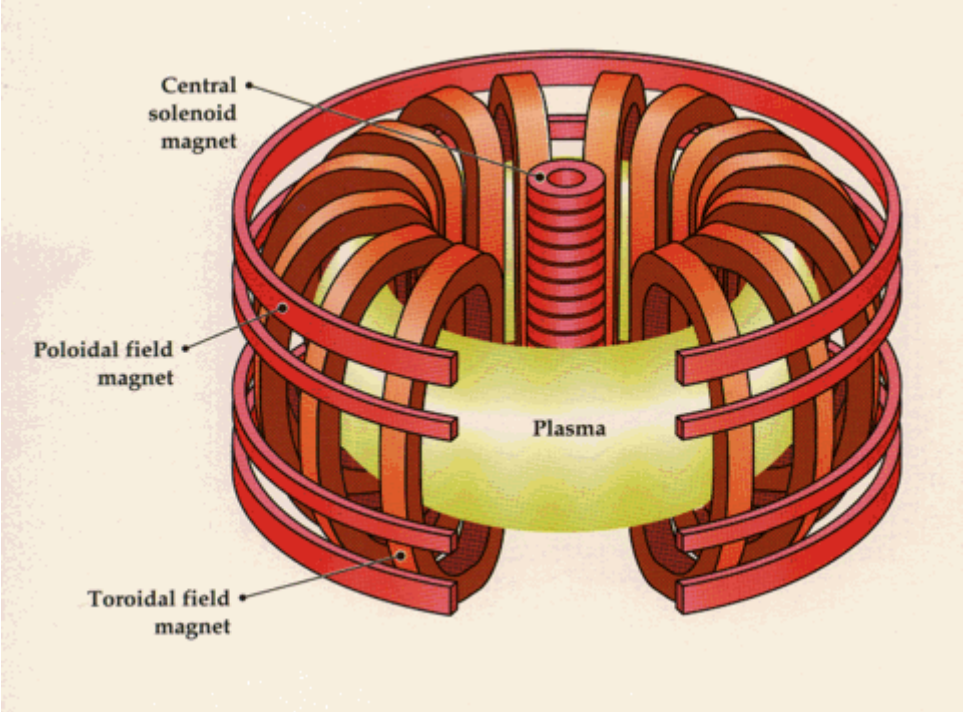

Figure 1. *Tokamak structure.*

Unfortunately, the equilibrium, so achieved, proves to be unstable and the ring of plasma in the Tokamak disintegrates in a time lapse, still today, considered to be too short and far from the necessary standards to make the fusion process energetically favourable.

Hence, the need of a controller able to act upon the plasma's evolution over time so as to ensure it remains confined for as long as possible.

In view of this, the importance of a plasma boundary identification algorithm is evident in order to control the position and the shape of the plasma itself.

EFIT (Equilibrium Fitting) is a computer code written by Lang Lao in FORTRAN 77 for the Tokomak and developed to translate measurements from plasma diagnostics into useful information like plasma geometry, stored energy, and electrical current profiles.

The *Grad-Shafranov equilibrium equation*, which describes the force equilibrium inside a plasma, is solved by using the available measurements as constraints on the toroidal electric current density.

The *EFIT++ code* is a substantial rewrite of the EFIT code originally written by Lang Lao. The original version, implemented in FORTRAN 77, has been installed on many Tokamaks. However, there has been a gradual diversification of the original source code as each installation was created by simply copying the EFIT source code.

In EFIT++, all machine dependencies have been removed and now reside either in local data files or in databases. Thus, the EFIT++ code has no pre-set values and all configurable parameters need to be explicitly set by the user. At present, output is written to an *HDF5*[1] file.

The implementation of EFIT++ is in a mix of C++ and Fortran 90. The user interface is written in C++ whereas the algorithmic elements (i.e. the construction of the tabulated response functions and the calculation of the individual equilibria) are written in Fortran 90. The reason for this mixed-language approach is a consequence of how the EFIT++ project evolved rather than due to any inherent performance issues of FORTRAN compared to C++.

The EFIT++ code sets out to compute the axisymmetric magnetic field distribution satisfying radial force balance in Tokamaks. The EFIT++ field solution provides the best fit to user-supplied synthetic and measured data relating to the magnetic fields, taking account of the uncertainties in the data. EFIT++ can calculate the magnitude of toroidal currents induced in conducting structures which may not have been explicitly measured. In addition, EFIT++ can model material with non-linear finite permeability characteristics by using an axisymmetric treatment.

In view of all this, it is clear that EFIT++ turns out to be an excellent solution to the plasma boundary identification problem and to the reconstruction of the equilibrium in the Tokamak. However, the procedure of using the EFIT++ code is not easy. In fact, the long compilation time of the code together with the many external libraries needed, and the not

[1] HDF5 is a file format designed to store and organize large amounts of numerical data.

simple compilation procedure that requires a shell script and therefore addressed only to expert users in programming, make not optimal the compilation procedure and the subsequent use of EFIT++ in terms of time and difficulty. These reasons have led to the idea of creating an installation package that would solve the above-mentioned problematic.

This study aims, precisely, to create an installation package of the EFIT++ code for Linux distributions and to highlight the advantages of its use rather than that of the standard procedure.

In the first chapter, the standard procedure for the use of EFIT++, based on the compilation of the source code, will be illustrate.

In the second chapter, the procedure for creating the installation package of the EFIT++ code for Linux distributions, the relevant installation directives and the program execution instructions will be illustrated. The Linux distribution on which the research has been carried out is *Ubuntu* and, therefore, the study will go on to illustrate how to create a *debian package* of the EFIT++ code.

This work also aims to be an actual installation manual.

Chapter 1: Installing EFIT++

As mentioned, EFIT++ is a rewrite of EFIT code written by Lao Lang in FORTRAN 77. The implementation of EFIT ++ is in a mix of C++ and Fortran 90. It is an excellent solution to the plasma boundary identification problem and to the reconstruction of the equilibrium in the Tokamak.

This chapter shows the download procedure, the compiling and the running of the EFIT++ code.

1.1 Download

The first thing to do is to create a directory and to rename it *"efit++"*. Then you have to download the main development version[2] of the EFIT++ code, i.e.:

```
mkdir efit++
cd efit++
svn checkout https://mastweb.fusion.org.uk/svnroot/efit++/development/trunk
```

After which, the EFIT++ source code and the make file will be located in the *"src"* subdirectory.

[2] The main development version used is of October 2011

1.2 External libraries

EFIT++ needs the following libraries to be compiled and installed:

Library	**Description**
Qt	Contains many facilities including a DOM parser and a regular expression evaluator.
Blitz	A C++ class library for scientific computing.
GSL	Gnu scientific library numerical library for C and C++.
IDAM	Universal API for data access.
BOOST (above version 1.35)	C++ source libraries.
pspline	Collection of Spline and Hermite interpolation tools.
lapack	Linear algebra package.
netCDF	Common data format.
netCDF C++ (local version)	Local version of the netCDF C++ library.
HDF5	Hierarchical data format.
BLAS	Routines that provide standard building blocks for performing basic vector and matrix operations.
efitxml	IDAM EFIT magnetics XML library.
xml2	Convert between XML, HTML, CSV and a line-oriented format.
ICU	Development files for International Components for Unicode.

You can easily install some of the required libraries by using *Synaptic*, the Ubuntu package management system. The following table shows which packages you have to install with Synaptic:

Library	**Package Name**
Qt	libqt4-dev
Blitz	libblitz0-dev
GSL	libgsl0-dev
BOOST	libboost-all-dev
lapack	liblapack-dev
netCDF	libnetcdf-dev
HDF5	libhdf5-serial-dev
BLAS	libblas-dev
xml2	libxml2
ICU	libicu-dev

As for the following libraries: *IDAM, pspline, netCDF C++ (local version) and efitxml*, they have to be downloaded, compiled and then installed in locations where you have normal user privileges by using the terminal and by following the proper instructions. The compilation procedure and the subsequent installation of the above-mentioned libraries are relatively complicated and require a considerable amount of time.

1.3 EFIT++ environment settings

The locations of the executable files, the include files, the object files and the libraries required by EFIT++ are specified in a single *shell script*[3]. You have to name this script *efitEnvironmentSettings.sh* and to place it in a location known to the system (i.e. a location specified in the *PATH environment variable*).

[3] A *shell script* is a computer program designed to be run by the Unix terminal. It is a text file that uses the *.sh extension* and its first line has to start with *"#! /bin/sh"*.

The location of the executable files is specified by the variable *"EFIT_LOCAL_BIN_PATH"*; the location of the object files is specified by the variable *"EFIT_LOCAL_OBJ_PATH"* and the location of the data for parallel architectures by *"EFIT_LOCAL_OBJECT_PARALLEL_PATH"*.

An example of instructions in *efitEnvironmentSettings.sh* that shows how to specify these variables is the following:

```
export EFIT_LOCAL_BIN_PATH=$HOME/efit++/bin
export EFIT_LOCAL_OBJ_PATH=$HOME/efit++/obj
export EFIT_LOCAL_OBJECT_PARALLEL_PATH=$HOME/efit++/obj_parallel
```

By so doing, the *"bin"*, *"obj"* and *"obj_parallel"* subdirectories, located in the *"$HOME/efit++"* directory, respectively contain the executable files, the object files and the files for parallel architectures.

It is also necessary to specify in the *efitEnvironmentSettings.sh* script the locations of all the libraries and of the include files required by EFIT++. Include files must be specified in the *"C_INCLUDE_PATH"* and *"CPLUS_INCLUDE_PATH"* variables, while the locations of the libraries are specified by the *"LIBRARY_PATH"* and *"LD_LIBRARY_PATH"* variables. An example of instructions contained in the script that shows how to specify these variables is the following:

```
BOOSTINC=/usr/include/boost
NETCDFINC=$HOME/netcdf/include
BOOSTLIB=/usr/lib/
NETCDFLIB=/$HOME/netcdf/lib
export C_INCLUDE_PATH=$BOOSTINC':'$NETCDFINC
export CPLUS_INCLUDE_PATH==$BOOSTINC':'$NETCDFINC
export LIBRARY_PATH==$BOOSTLIB':'$NETCDLIB
export LD_LIBRARY_PATH==$BOOSTLIB':'$NETCDLIB
```

In addition, *efitEnvironmentSettings.sh* must also contain an instruction declaring the *Fortran Flags* that specify the location of the *.mod file* of the *pspline and netcdf* libraries, e.g.:

```
export FFLAGS= "-I. -I./home/username/pspline/LINUX/mod -I/home/username/netcdf/include –g"
```

1.4 Compiling

At this point, you can compile the EFIT++ source files by first using the *source*[4] command and then the *make* command, i.e.:

```
cd efit++/src
source efitEnvironmentSettings.sh
make
```

In order to compile them you must have installed the *gfortran* and *g++ compilers*.

If the build is successful, the object files and the *efit++.exe* executable file will be respectively generated in the locations specified by *"EFIT_LOCAL_OBJ_PATH"* and by *"EFIT_LOCAL_BIN_PATH"*.

1.5 Running

To run the program, you need the following data files:

- *efitOptions.xml* (contains data describing EFIT++ specific parameters);
- *pfCircuits.xml* (describes active currents driven by pf supplies, and passive currents induced by the active currents);
- *tokamakData.xml* (contains details of a particular Tokamak, for example its geometry and measurement data).

[4] The *source* command in shell is used to execute commands from a file in the current shell. This is useful to load function or variables stored in another file.

These files have to be positioned in the directory specified by *"EFIT_LOCAL_BIN_PATH"* (the directory where is located the *efit++.exe* file) and are used both to provide the program input and to customize the output.

To run the program, you must use the following commands:

```
cd efit++/bin
source efitEnvironmentSettings.sh
./efit++.exe
```

If execution of the program is successfully completed, the *efitOut.hdf5* output file will be created in the same location of the *efit++.exe* file.

Chapter 2: Development of the Installation Package

This chapter shows how to create an installation package of the EFIT++ code for Linux distributions.

The Linux distribution used is *Ubuntu 12.04 (64-bit)*; hence, the procedure of creating a *debian package (.deb extension)*[5] will be shown.

Debian packages are standard *UNIX* archives that include two tar archives optionally compressed: one archive holds the control information and another contains the program data.

To be able to create the package of the EFIT++ code, the source code has to be compiled and the object files and the executable files have to be generated (see Chapter 1).

The installation package has been created by using *Debreate*, an aid utility for the building of debian packages. The goal is to make packaging for Debian based Linux distributions more appealing with an easy to use interface for creating distributable archives of applications, artworks, media, and more.

[5] deb is the extension of the Debian software package format and the most often used name for such binary packages.

2.1 Debreate

First, you have to download Debreate from the following URL:

```
http://debreate.sourceforge.net/
```

Then, you have to install this utility with the terminal by using the *dpkg*[6] *command*, e.g.:

```
sudo dpkg -i ./debreate_version.deb
```

Debreate needs the *python-wxgtk2.8 library* in order to work. You can install it via the terminal or with the package management system.

After that, you can execute the utility with the command:

```
debreate
```

and start the creation and the customization of the package.

Debreate is divided into several sections that will be specifically considered.

The *"Control"* section of Debreate is where the *control file* is to be created. The control file is a text file that is the core of the debian package. This file contains all the package information needed by the package manager (package name and version, package maintainer's name and email address, computer architecture for which the package is intended, package description, etc.). This section of Debreate has a field for the necessary entries of the control file.

The *"Dependencies and Conflicts"* section provides an interface to specify which packages are required for the correct installation.

The *"Files"* section is for specifying which files will make up the contents of the package.

In the *"Script"* section, you can create scripts. Maintainer scripts are executables that can

[6] *dpkg* is the software at the base of the package management system in the free operating system Debian and its numerous derivatives. It is used to install, remove, and provide information about .deb packages.

be called during the installation and/or removal processes.

In the *"Changelog"* section, you can create the *changelog file*. The changelog is a documentation of changes across different versions. Changelogs are gzip compressed text files containing the information on the package changes.

The *"Copyright"* section is simply a plain text editor where the entire copyright license or a copyright declaration with the path to a common license available on the system can be declared.

In the *"Menu Launcher"* section, you can create a single launcher to be installed to /usr/share/applications. These menu entries are normally based on text files that use the *".desktop extension"* and are often referred to as *desktop files*. Desktop files contain information about what the launcher should look like and how it should act.

The *"Build"* section is where the actual building of the package begins. Once the *"build button"* is pressed, Debreate will gather all of the information inputted into each section and start the process.

2.2 Creating the installation package of EFIT++

First, you have to create the directory *"efiit++"* that will turn out to be the content of the installation package. Then you have to start Debreate in order to create the package itself.

2.2.1 Creating the package structure

You need to create a directory and name it *"efit++"*.

This directory should contain:

- The *"src"* subdirectory containing all the source files of EFIT++ and the *efitEnvironmentSettings.sh* script file;
- The *"lib"* subdirectory containing all the libraries and header files used to execute and to compile the program;
- The *"obj"* subdirectory containing all object files that are generated during the compilation of the EFIT++ source code. This subdirectory must be specified by the

"EFIT_LOCAL_OBJ_PATH" variable in *efitEnvironmentSettings.sh* script that is located in the *"src"* subdirectory;

- The *"bin"* subdirectory containing the executable file *efit++.exe*. This subdirectory must be specified by *"EFIT_LOCAL_BIN_PATH"* variable in the *efitEnvironmentSettings.sh* script located in the *"src"* subdirectory;
- The *"obj_parallel"* subdirectory containing the data for parallel architecture. This subdirectory must be specified by the *"EFIT_LOCAL_OBJECT_PARALLEL_PATH"* variable in the *efitEnvironmentSettings.sh* script located in the *"src"* subdirectory;
- The *"test"* subdirectory containing a copy of *efit++.exe* located in the *"bin"* subdirectory, the three data files *(efitOptions.xml, pfCircuits.xml, tokamakData.xml)* and the copy of the *efitEnvironmentSettings.sh* script located in the *"src"* subdirectory. This subdirectory will contain the output file and will be used for testing.

Figure 2. *Structure of the "efit++" directory.*

Furthermore, the *efitEnvironmentSettings.sh* script must be properly edited so that all users can use it. This script will be used to indicate the location of the libraries that are needed for the execution of the program.

An example of instructions of this script after the appropriate modifications is the following:

```
BOOSTINC=../lib/boost
NETCDFINC=../lib/netcdf/include
BOOSTLIB=../lib/boost
NETCDFLIB=../lib/netcdf/lib
export C_INCLUDE_PATH=$BOOSTINC':'$NETCDFINC
export CPLUS_INCLUDE_PATH==$BOOSTINC':'$NETCDFINC
export LIBRARY_PATH==$BOOSTLIB':'$NETCDLIB
export LD_LIBRARY_PATH==$BOOSTLIB':'$NETCDLIB
```

```
export EFIT_LOCAL_BIN_PATH=../bin
export EFIT_LOCAL_OBJ_PATH=../obj
export EFIT_LOCAL_OBJECT_PARALLEL_PATH=../obj_parallel
```

As mentioned above, all the libraries necessary to execute EFIT++ have to be placed inside the *"lib"* subdirectory. The list of all the libraries can be found in section 1.2

Figure 3. *Example of the "lib" subdirectory.*

Having the *IDAM*, *pspline*, *netCDF C ++ (local version)* and *efitxml* libraries been regularly compiled and installed by the user, they can be simply copied into the *"lib"* subdirectory.

While, instead, the libraries installed with *Synaptic* and therefore installed in the system directories, can be copied to the *"lib"* subdirectory only with administrator permissions. Because of the large number of files to copy, you should use the *sudo nautilus*[7] command.

The system directories that contain these libraries are:

- usr/include/
- usr/lib/
- usr/lib/x86_64-linux-gnu/

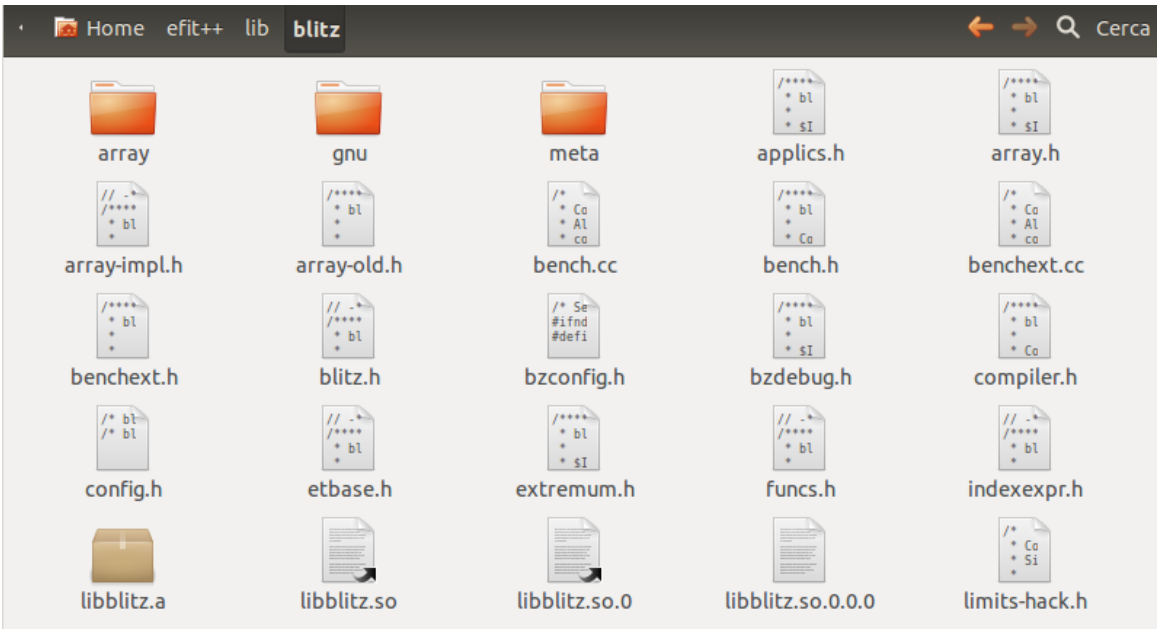

Figure 4. *This figure shows some files of the Blitz library. Header files, static (.a extension) and dynamically linked shared object (.so extension) libraries of each external library required have to be copied to the "lib" subdirectory.*

Considering that files copied from system directories can be read, written and performed only with administrator permissions, you must change the permissions to all the files located in the *"efit++"* directory and subdirectories. To do this, use the *chmod*[8] command.

[7] *GNOME Files*, formerly called *Nautilus*, is the official file manager for the GNOME desktop.
[8] *chmod* is the command and system call which may change the access permissions to file system objects (files and directories).

If, for example, the *"efit++"* directory has been created in the home directory, then insert:

```
sudo chmod –R 777 $HOME/efit++
```

2.2.3 Starting Debreate

Once created the *"efit++"* directory containing everything that is needed for the package, you have to start Debreate and particularly dwell on the *"Control"*, *"Files"* and *"Build"* sections.

In the *"Control"* section, you must specify the information required for the control file, such as the name of the program, the version, your name, your e-mail address and the computer architecture[9] for which the package is intended. You can also enter information about the program in the *"Short Desciption"* and *"Long Description"* fields.

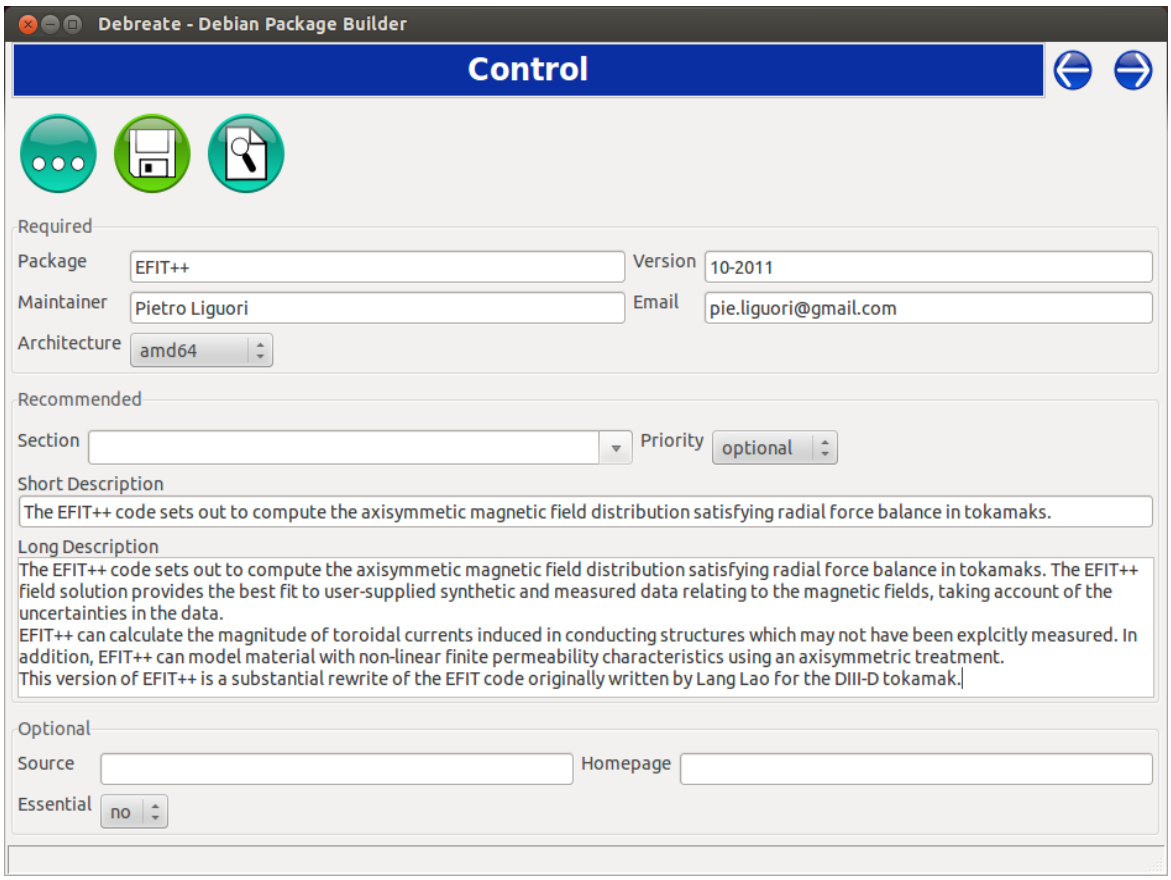

Figure 5. *The "Control" section of Debreate.*

[9] In the *"Architecture"* field, you have to select the same computer architecture on which the EFIT++ code has been compiled.

In the *"Files"* section, you must first select the *"custom"* option in *"Target"* and then digit the destination directory *"/efit++"* which will also become the package installation directory. Then you must select the previously created directory *"efit++"* (on the left) which is our package structure and finally you must press the *"Add"* button as shown in the figure below:

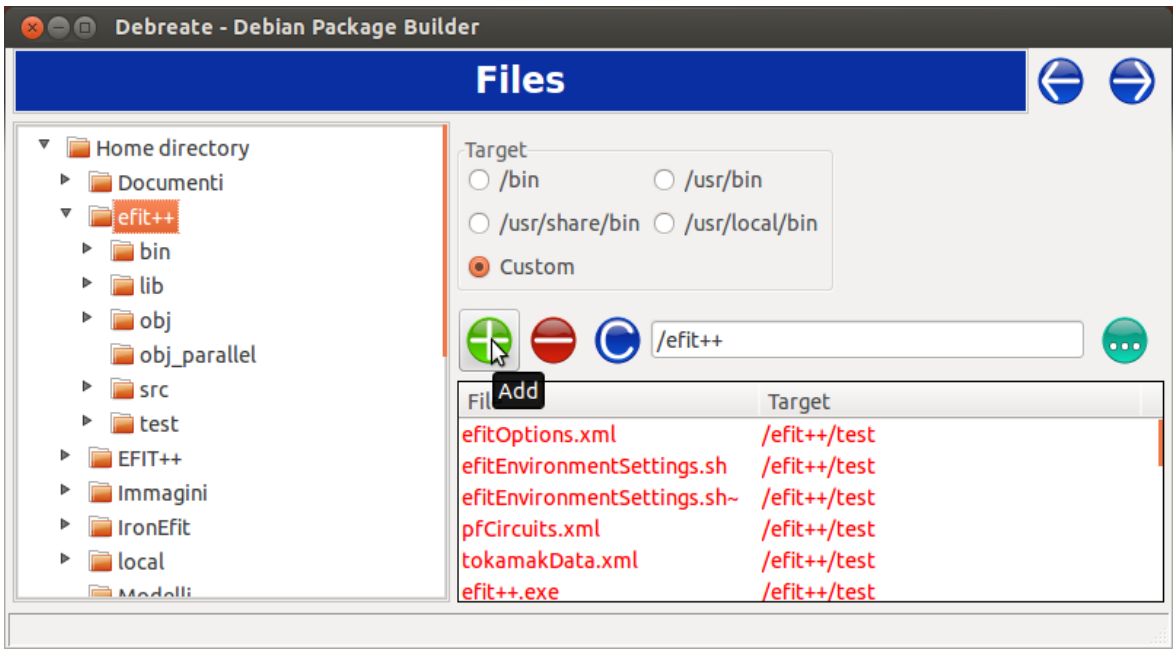

Figure 6. *The "Files" section of Debreate*

In the *"Build"* section, click on the *"build"* button: then choose the name of the package and the directory where you wish to save it.

Figure 7. *The "Build" section of Debreate.*

2.3 Installing and running the package

Now it is possible to simply install the package by using the terminal with the following command:

```
dpkg-deb   -x   /Package_Path/Package_Name.deb     /Installation_Path
```

in which:

a) the *Package_Path* indicates the debian package path;

b) the *Package_Name* indicates the name of the debian package;

c) the *Installation_Path* indicates the path where you want to install the program. It is necessary to choose a location in which the user has the read, write, and execute permissions.

For example, if the package (e.g. *EFIT ++_10-2010_amd64.deb*) is located on your *"Desktop"* and you want to install it in your *"Home directory"*, then you must use the following command:

```
dpkg-deb   -x   $HOME/Desktop/EFIT++_10-2010_amd64.deb      $HOME
```

The result of the above-mentioned operation is that the *"efit++"* directory will be created in the chosen installation path. This directory will contain all the files that were included in the package. The overall time for the package installation will be merely a few seconds. Regarding the execution of EFIT++, by using the terminal just move into the directory containing the *efit++.exe* executable file and run it after having first executed the *efitEnvironmentSettings.sh* script, i.e.:

```
cd efit++/test
source ./efitEnvironmentSettings.sh
./efit++.exe
```

Execution of the script is necessary because it indicates the locations of the libraries that have been included in the package and which are essential for the execution of the EFIT++ code.

As previously explained, the script has been conveniently modified so as to require no further editing by the user.

The program will begin to perform calculations based on the values contained in the input data file in the *"test"* subdirectory. The output will be saved in the *efitout.hdf5* file.

2.3.1 Removing the package

You can easily remove EFIT++ by deleting the *"efit++"* directory.

The cancellation does not require any special permissions.

2.3.2 Recompiling the source code

In addition to the libraries, even the header files, needed to compile the source code, have been inserted inside the installation package. Although this choice significantly increases the size of the debian package, it gives the user the option to recompile the source code contained in the *"src"* subdirectory. This option might be useful if you should decide to change the source code.

In case you want to recompile the source code, it is necessary to conveniently modify the *Fortran Flags* in line 33 of the *efitEnvironmentSettings.sh* script positioned in the *"src"* subdirectory. After this change, you can recompile the source code with these commands:

```
cd efit++/src
make clean
make
```

At the end of the recompilation process the *efit++.exe* file will be generate in the *"bin"* subdirectory. You have to copy this file to the *"test"* subdirectory before starting the program execution.

When recompiling the source code, since all the libraries have been compiled with a

specific version of *g++* and *gfortran*[10], it is absolutely necessary to install the same compiler versions used and to uninstall any other installed version.

[10] Compiler versions used are: g++ 4.6 and gfortran 4.6

Conclusions

The EFIT++ code turns out to be extremely useful for those working in the field of nuclear fusion.

However, the long-time needed both to compile the source code and to compile and install the external libraries, the not simple library compiling procedure, the use and the modification of a script make difficult the process of installing the EFIT++ code and, in terms of time, very inconvenient.

The creation of the debian package of EFIT++ code has brought significant improvements. In fact, with the newly created installation package we have the following significant advantages:

- Not having to compile the source code because it is already pre-compiled;
- Not having to either download, compile or install the libraries necessary to run EFIT++ because they are already included in the package;
- Not having to create or to edit the script *efitEnvironmentSettings.sh* because it has been conveniently modified for use by the user;
- Possibility to be able to recompile the source code after any change;
- Very short installation time that turns out to be even more significant when compared with the time needed both for the compilation of the source code and of the external libraries;
- Possibility of having all the program files located in the same directory instead as

having them dispersed throughout the system as was the case with the standard procedure.

Therefore, it is clear that the use of the installation package brings significant improvements both in terms of difficulty and in terms of speed of use.

Concerning the Debreate utility, it has been extremely useful in the development of the installation package. In fact, by having a very intuitive graphical interface, it has made the development of the debian package of the EFIT++ code sufficiently easy.

Bibliography

[1] Alfredo Pironti, Identificazione della frontiera di plasma in un Tokamak con l'espansione di Legendre-Fourier, Academic Year1990-91

[2] Lynton Appel, EFIT++ user manual, 2nd July 2010

[3] Alfredo Pironti and Michael Walker; Fusion, Tokamaks and Plasma Control; IEEE Control Systems Magazine; October 2005, Volume 25 Number 5

[4] Lynton Appel, EFIT++ Equilibrium Reconstruction, 2010

[5] Debreate Usage, Versione 0.3

[6] Theory and Computational Sciences website, https://fusion.gat.com/theory/Efitdef, October 2014

[7] Official Ubuntu Documentation, https://help.ubuntu.com/12.04/index.html, November 2014

[8] Debreate - Debian Package Builder, http://debreate.sourceforge.net, October 2014