2005 International Nuclear Atlantic Conference - INAC 2005
Santos, SP, Brazil, August 28 to September 2, 2005
Associação Brasileira de Energia Nuclear - ABEN
ISBN: 85-99141-01-5

FREE/OPEN SOURCE SOFTWARE: A STUDY OF SOME APPLICATIONS FOR SCIENTIFIC DATA ANALYSIS OF NUCLEAR EXPERIMENTS

Mário Olímpio de Menezes

Instituto de Pesquisas Energéticas e Nucleares (IPEN / CNEN - SP)
Av. Professor Lineu Prestes 2242, Cid. Universitária
05508-000 - São Paulo, SP
mario@ipen.br | mo.menezes@gmail.com

ABSTRACT

Free/Open Source Software (FOSS) has been used in science long before the formal social movement known as "Free Software/Open Source Software" came in to existence.
After the Personal Computer (PC) boom in the 80s, commercial closed source software became widely available to scientists for data analysis in this platform.
In this paper, we study some high quality FOSS, available also for free, that can be used for complex data analysis tasks. We show the results and data analysis process, aiming to expose the high quality and highly productive ways of both results and processes, while highlighting the different approach used in some of the FOSS.
We show that scientists have today in FOSS a viable, high quality alternative to commercial closed source software which, besides being ready to use, also offer the possibility of great customization or extension to fit very particular needs of many fields of scientific data analysis.
Among the FOSS, we study in this paper GNU Octave and SCILAB - free alternatives to MATLAB®; Gnuplot - free alternative to ORIGIN®-like softwares.
We also show that scientists have invaluable resources in modern FOSS programming languages such as Python, and Perl, that can be used both to do data analysis and manipulation, allowing very complex tasks to be done automatically after some few lines of easy programming.

1. INTRODUCTION

Scientific data analysis in our days is often done using computer programs that manipulate data, produce graphics and results in easy ways.

Using such programs, scientists had gained a lot since most repetitive tasks can be automated and time can be saved leaving the computer doing the hard job while the researcher can think on other problem or simply go for a good cup of coffee.

In the early days of computer development, sharing programming code was a common practice among those who wrote software. This happened mainly in academia, and the result was that everybody could benefit from each other work.

The vast majority of companies today who develop software use restrictive licenses for their software and the user have to pay for the right to use the software, while the source code is kept locked away from the user.

Free Software Foundation [1], started by Richard Stallman to support and create free software establish that to be *free*, a software must allow the user:

i. run the program, for any purpose;
ii. modify the program to suit their needs. (To make this freedom effective in practice, they must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.) They must have the freedom to redistribute copies, either gratis or for a fee;
iii. redistribute copies, either gratis or for a fee;
iv. distribute modified versions of the program, so that the community can benefit from your improvements;

The rest of this paper talks about free softwares used to do scientific data analysis. We'll give a brief description of each and then show some short examples of its use, trying to highlight the easiness to automate repetitive tasks due to the non-graphical user interface of these softwares.

2. PROGRAMS AND DATA ANALYSIS EXAMPLES

There are several free/open source software that can be used to do data analysis of nuclear experiments. Many of them are normally included in all major GNU/Linux distribution. In the following sections we will use GNU Octave, Gnuplot, Scilab and Python to do some data analysis and/or manipulation commonly done or required by researchers.

We'll use data from Neutron Radiography (NR); using GNU Octave and Gnuplot we analyze NR resolutions and using Scilab we analyze neutron cross sections.

For the Resolution studies, we obtain the pixel profile of a thin absorber image at the interface region of the direct beam and the absorber. To these data we fitted a “Edge Spread Function” - ESF, given by:

$$ESF = A + B \cdot \arctan\left(C \cdot (x - D)\right) \tag{1}$$

2.1. GNU Octave

GNU Octave is a high-level matrix-based language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab[1]. It may also be used as a batch-oriented language [2].

Octave includes a collection of tools for solving common numerical linear algebra problems, finding the roots of nonlinear equations, integrating ordinary functions, manipulating polynomials, and integrating ordinary differential and differential- algebraic equations. It is easily extensible and customizable via user-defined functions written in Octave's own language, or using dynamically-loaded modules written in C++, C, Fortran, or other

[1] Matlab is a registered trademark of the MathWorks, Inc

languages. Octave is free software distributed under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation (FSF).

Using Octave we proceed the analysis of Resolution data from Neutron Radiography technique.

A typical Octave session for this job is show in the Figure 1:

```
my_data = load "file_with_data_to_be_fitted"
function y = resol(x,p) %f(x) = a + b * atan(c * (x - d))
y = p(1) + p(2) * atan (p(3) * ( x - p(4)))
endfunction
F = "resol"
[lin,col] = size(my_data)
x = my_data(1:lin,1)
y = my_data(1:lin,2)
pin = [30; 13; 8; 0.5]
[f1, p1, kvg1, iter1, corp1, covp1, covr1, stdresid1, Z1, r21] = leasqr
(x, y, pin, F);
title(" Put here the Plot title")
xlabel("XLabel")
ylabel("YLabel")
gset key bottom
gset key right
gset output "Output_filename.eps"
gset term postscript eps enhanced linewidth 2 "Times bold" 32
plot (x,y,"+;Average 20 readings;",x,f1,"-;Edge Spread Function;")
save -text 'Logfilename.ext' p1 covp1 corp1
```

Figure 1 – A typical Octave session used to analise Neutron Radiography Resolution

Although Octave doesn't offer a nice GUI, its language is quite intuitive and a batch job can be easily done using a scripting language to automate a repetitive task on a large data set. Just save the above lines into a file and pass it as argument to octave program.

The fitted parameters are shown in Table 1.

Table 1 – Fitted parameters for Edge Spread Function data shown in Figure 3a

Parameter	*Value*	*Error*
A	90.23	0.17
B	11.12	0.17
C	4.48	0.41
D	2.68	0.01

2.2 GNUPLOT

Gnuplot is a freely distributed plotting tool with ports available for UNIX, IBM OS/2, MS Windows, DOS, Macintosh, VMS, Atari and many other platforms [3]. It can be operated in interactive mode, issuing commands at the Gnuplot prompt or in batch mode, reading commands from a file. The fact that Gnuplot doesn't offer a nice GUI it's not a drawback at all; it must be seen just as a different approach to plotting. For example, you can save Gnuplot commands in a file and load them later to re-execute in a interactive session, freeing you from trial and error approach when trying to reproduce some old graph you have done.

Using Gnuplot we proceed the analysis of Resolution data from Neutron Radiography technique.

A typical Gnuplot session for this job is shown in Figure 2:

```
set terminal postscript eps enhanced linewidth 2 "Times bold" 32
set output "Output_filename.eps"
set fit logfile 'Logfilename.ext'
f(x) = a + b * atan(c * (x - d))
a = 30
b = 15
c = 800
d = 0.15
fit f(x) 'file_with_data_to_be_fitted' via a,b,c,d
set xlabel "Scanning Coordinates (mm)"
set ylabel "Pixel Gray Level"
set title "Gd Foils - 75 frames - GNUPLOT"
set size 1.6,1.6
plot [] [60:120] 'file_with_data_to_be_fitted' title "Average 20
readings", f(x) title "Edge Spread Function"
```

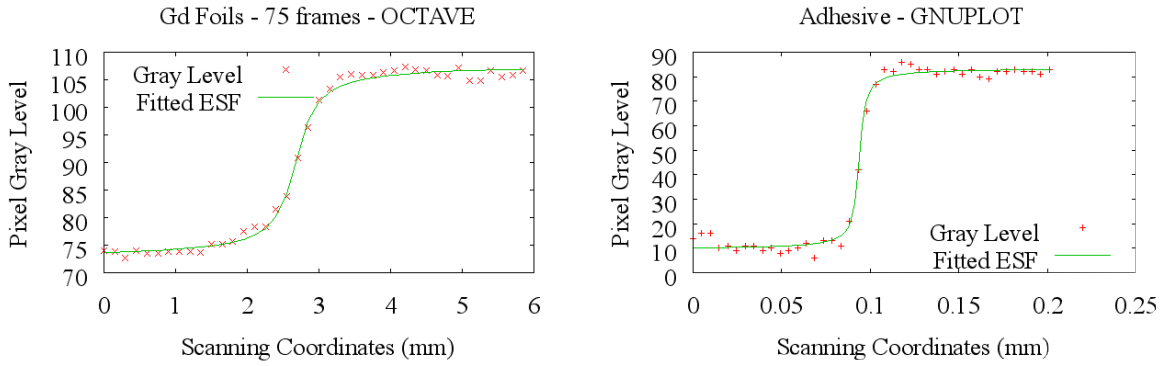
Figure 2 – A typical Gnuplot session used to analise determine NR resolution

The fitted parameters are shown in Table 2.

Table 2 – Fitted parameters for Edge Spread Function data shown in Figure 3b

Parameter	*Value*	*Error*
A	46.4703	0.4263
B	23.5311	0.3564
C	394.862	69.91
D	0.0938721	0.0003073

Figure 3 shows the fitted Edge Spread Function as well as the experimental points of resolution studies of the Neutron Radiography facility installed at the IPEN Nuclear Research Reactor IEA-R1m.



(a) **(b)**

Figure 3. Gray levels distribution at the interface region of neutron radiography images: (a) – GNU Octave; (b) - Gnuplot

2.3 SCILAB

Scilab was developed at INRIA for system control and signal processing applications [4]. Although not a free software as defined by Free Software Foundation, it's freely distributed in source code format.

One of Scilab's key features is the way it handles matrices, doing basic matrix manipulations such as concatenation, extraction or transposition, addition or multiplication. It also do more complex things such as manipulating polynomials, polynomials and transfers matrices; it also provides a variety of powerful primitives for the analysis of non-linear systems, integration of explicit and implicit dynamic systems. The scicos toolbox allows the graphic definition and simulation of complex interconnected hybrid systems.

Using Scilab, we'll do some analysis on neutron cross sections data. We have a set of files containing neutron cross sections in thermal energy range for some elements. We would like to get an idea of the effective cross sections values of some elements using a thermal neutron beam filtered by 20cm of bismuth.

The total microscopic effective cross section of an element for a filtered neutron beam can be given by [5]:

$$\sigma_T = \frac{\int M(v)\cdot T_{Bi}\cdot v\cdot \sigma_T(v)\,dv}{\int M(v)\cdot T_{Bi}(v)\cdot v\,dv} \qquad (2)$$

where:
$M(v)$ is the Maxwellian thermal neutron spectrum reaching the filter;
T_{Bi} is the Bismuth transmission for neutrons with energy v
$\sigma_T(v)$ is the total microscopic cross section for neutrons with energy v

A typical Scilab section that would do this job is shown on Figure 4:

```
No=6.023e23; barn=1.0e-24; RO=9.803; Eo=0.0258
bismuto=mopen('bismuto.sc','r')
[titul]=mfscanf(bismuto,'%s\n')
bism_dados(1:3)=mfscanf(bismuto,'%f %f %f\n')
scbism=bism_dados(1,1);masbism=bism_dados(1,2);densbism=bism_dados(1,3)
bismuto_dados=mfscanf(-1,bismuto,'%f %f\n');
cobre = mopen('cobre.sc','r')
[elemnome] = mfscanf(cobre,'%s\n')
elem_dados(1:3) = mfscanf(cobre,'%f %f %f\n')
abselem=elem_dados(1,1);maselem=elem_dados(1,2);denselem=elem_dados(1,3)
elemento_dados=mfscanf(-1,cobre,'%f %f\n');
[tr]=exp((-densbism*No*barn*20.0/masbism).*bismuto_dados(:,2));
[y]=elemento_dados(:,1).*exp(-elemento_dados(:,1)/Eo).*tr ;
integ1=intsplin(elemento_dados(:,1),y);
[y2] = y.*elemento_dados(:,2) ;
integ2=intsplin(elemento_dados(:,1),y2);
// Effective Microscopic Cross Section
micro = integ2/integ1
[y2abs]= abselem.*y.*sqrt(Eo./elemento_dados(:,1)) ;
integ2_abs=intsplin(elemento_dados(:,1),y2abs) ;
//  Absorption Effective Microscopic Cross Section
micabsor = integ2_abs/integ1
// Scattering Effective Microscopic Cross Section
micespal = micro - micabsor
// Effective Macroscopic Cross Section
macro = denselem*No*micro/maselem*barn
// Absorption Effective Macroscopic Cross Section
denselem*No*micabsor/maselem*barn)
// Scattering Effective Macroscopic Cross Section
macro -(denselem*No*micabsor/maselem*barn)
```

Figure 4 – A typical Scilab script for analysing neutron cross section.

Filenames are hardcoded but the user could be prompted to enter them, transforming this **Cu** specific script into one that could be used to do the same calculation for several elements.

Using this script we obtain the results shown on Table 3. We emphasize that these are approximate values since we were interested on estimates.

Table 3 – Cross Sections Values for Cu for a 20cm Bi filtered thermal neutron beam

Cross Section	*Value*
Total Microscopic	11.77357 barns
Absorption Microscopic	6.13965 barn
Scattering Microscopic	5.63391 barns
Total Macroscopic	0.99277 cm^{-1}
Absorption Macroscopic	0.51771 cm^{-1}
Scattering Macroscopic	0.47506 cm^{-1}

2.4 PYTHON

Python is an *interpreted, interactive, object-oriented* programming language [6]. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC). New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface.

The Python implementation is portable: it runs on many brands of UNIX, on Windows, OS/2, Mac, Amiga, and many other platforms.

Using python, we wrote a script to analyze a set of Neutron Radiography resolution data. The script used is shown in Figure 5.

```
#!/usr/bin/env python2.3

import os
import glob
import sre
os.chdir('/home/mario/docs/papers/Inac2005-ipen/dados-Marcos-Resolucao')
f = os.popen('find . -name "*.HST"')
arq = f.read()
arquivos = arq.split('\n')
f.close()
for arquivo in arquivos:
    nomebase,extensao = os.path.splitext(os.path.basename(arquivo))
    if len(nomebase) > 0:
      f = open("batch-fit.gnu", "w")
      f.write("set term png notransparent font \"Times\" 20 enhanced\n")
      f.write("set output '" + nomebase + ".png'\n" )
      f.write("set fit logfile '" + nomebase + ".log'\n")
      f.write("f(x) = a + b * atan(c * (x - d))\n")
      f.write("a = 30; b = 55; c = 8; d = 0.05 \n")
      f.write("fit f(x) '" + arquivo + "' using 2:3 via a,b,c,d \n")
      f.write("set xlabel \"Scanning Coordinates (mm)\" \n")
      f.write("set ylabel \"Pixel Gray Level\" \n")
      f.write("set title  \"" + nomebase + "\" \n")
      f.write("set tmargin 3 \n")
      f.write("set bmargin 5 \n")
      f.write("set key bottom \n")
      f.write("plot '" + arquivo + "' using 2:3 title \"Gray Level\",f(x)
title \"Fitted ESF\"\n")
      f.close()
      comando = 'gnuplot batch-fit.gnu'
      os.system(comando)
print "All data analyzed; graphs and logs produced\n"
```

Figure 5 – Python script used to batch analyze Neutron Radiography Resolution data.

After running this script, several log files were produced with fit results; several PNG files were also produced containing the data and fitted function plots. The results and plots are similar to those shown in Table 2 and in Figure 3b; they are omitted here to save space.

3. CONCLUSIONS

In this paper we had shown that FOSS of high quality are available to scientific data analysis. We also showed that these softwares, being readily available for a number of platforms, offer a number of alternatives to those considering migrating from proprietary systems to free software.

It's important to note that the non-GUI approach of some FOSS, while being intimidating at the beginning, can be regarded as a strong feature, easily allowing the use of batch oriented jobs, as shown in section 2.5 with Python language.

Due to the space limitation many other available FOSS could not be used. They can be found on Internet for several platforms; for GNU/Linux, many are included on all major distributions.

ACKNOWLEDGMENTS

I would like to thanks Marcos L. G. Andrade and Guilherme Soares Zahn, both from IPEN, for sending some of their data file to be used as example in this paper.

REFERENCES

1. **Free Software Foundation**. Available from: http://www.fsf.org [May 2005]
2. J. W. Eaton and James B. Rawlings. Ten years of Octave - recent developments and plans for the future. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, Friedrich Leisch Kurt Hornik and Achim Zeileis, editors, Vienna, Austria, March 20-22 2003.
3. **Gnuplot**. Available from: http://www.gnuplot.info [May 2005].
4. **Scilab**. *Although Scilab License isn't GPL compatible, it's available for free, with source code, and redistributed in al l major Linux distribution. It's available for Windows OS family as wel l as for Mac OS.* Available from: http://www.scilab.org [May 2005].
5. M. O. Menezes, *Real-Time Neutron Radiography*. PhD thesis, Universidade de São Paulo - USP/IPEN, 2000.
6. **Python Programming Language**. Available from: http://www.python.org [May 2005].