Why The Future Of Science Must Be In Free Software

Alessio Damato

26th June 2005

Abstract

This essay is about free software.

Nowadays the importance of software is increasing day after day, and more and more people are obliged to use software in their everyday life. This is why talking about software often means talking about people's habits with all the relative consequences: economic and pseudo-philosophical arguments are often mixed up.

This essay is an attempt to give an objective overview about how free software can be useful for anybody, showing in particular how important it can be for the development of science; for this purpose several links plus few practical examples are provided, to allow anybody to make up their own mind.

1 Introduction

I had the luck of studying electronic engineering in both Italy and the UK. As soon as I had to face proper engineering problems, I had to start using the calculating power of computers intensively. I was often requested by lectures to use a particular programme to solve some problems, and I did notice that all the programmes I was supposed to use were not free. This meant I had to use the computers of the university, that were always overcrowded and badly configured. Another possibility was to get an illegal copy of the programme and crack it, in order to use it in a computer at home or on the laptop, and I have seen a LOT of people doing it, but as it is illegal I would not call it a "possibility". Nobody did even consider the possibility to buy it: they are always VERY expensive and students are typically the people who do not have a lot of money.

Then I noticed that all my lecturers had a personal licence for the expensive software they had to work with: this is right, as they actually need it, but the very expensive price of a licence multiplied by all the lecturers is really a lot of money. Thinking about this, I realised that there is something wrong . . .

NB: since now, by the word *free* (about software) I will mean both that it is not necessary pay for it and it is open source. I know that they are not synonymous, as software can be free but not open source and vice versa, but this convention will be used to simplify the exposition.

2 Free vs Proprietary

Comparing free and non-free software under any point of view is a very big task and it is quite often the reason of flames in newsgroups or of heated arguments in forums. Here only the main differences will be considered.

First of all, free software is free, that means it is not necessary to pay for it. Thus it can be diffused anywhere, it can be installed on several workstations: piracy does not exist.

Non-free software has the big issue of compatibility of proprietary formats. It often happens that a file saved with a newer version of a software is not readable by an older one. This obliges to an (usually expensive) update even if there is no actual need for it. In free software this is a quite rare situation: they save in standards (e.g. XML), improving backwards compatibility. Anyway, when an update is compulsory, it is not such a big problem since all the necessary can be freely downloaded from Internet.

Free software is open source: anybody can look in the source to understand how it works and, just in case, modify to improve it.

Non-free software is implemented by companies trying to fit users' needs, while in free software developers and users are often the same people.

3 Freedom in Consumers' Software

Free software is slowly, but inexorably, taking over the proprietary one about the tools that are used everyday by 90 % of the users.

The good points are that the free software is safe, stable, easy to use, and obviously free. It has been argued that proprietary software has more features: this is usually true, but if nobody felt the need to implement a feature it often means that is a quite rare requirement.

The most evident case is in Internet browsing: Internet Explorer by Microsoft has been the leader for a very long time, but now a new browser was born and it is spreading around because it is safe, easy to use and with an attractive lay-out: Firefox [2].

Another example is in office suite. Microsoft Office became a standard *de facto*, but now OpenOffice.org [3] has almost the same features and it is free. Its main good point is the format: Microsoft Office saves in a closed binary format containing a lot of redundancy, private information and sometimes viruses; OpenOffice.org saves in a plain zip-compressed XML format that, optionally, can be read with any text editor. OpenOffice.org has a built-in tool to export directly in PDF, too.

After web browsers and office suites, the most common kind of application is a video player. Free software proposes a valid alternative to commercial software in this field, too: VideoLAN [5] offers all the basic features any player would need, it supports all the standard formats and it contains no spy-wares.

Finally, another very common programme is an image processor; the Gimp (GNU Image Manipulation Program) [1] was born just like an image viewer, but now it has evolved into a mature 2.x version. It is able to implement all the basic manipulation tools and it has several advanced filters and scripts.

This overview was about multi-platform software, but it must be remembered that there is a huge quantity of free software developed in Unix-like environment for the most different purposes.

4 Why Free Software Is Necessary In Science

All the good points making free software a possible alternative to the proprietary one become a need when talking about science.

Is is known that what influences the development of science most is communication: when researchers have the possibility to share and compare ideas it is harder to make mistakes. This is why Internet is helping science so much.

Another thing that is helping the development of science is the computer, with its ability to handle huge amounts of data very quickly.

The algorithms to make scientific work with the computer are mainly implemented by proprietary software. The software hoses working on it conduct research to find and improve the algorithms they need.

But obviously, when such programmes have to be sold, the algorithms they use can not be published to protect their copyright.

Here is the lack of communication: this software is used like a "black box", it is not possible to know how the are made, neglecting any possibility to fully understand, fix or modify them.

Another big issue of proprietary scientific software is the price: as they are very specific and complicated, they are *very* expensive, and so only academies and few organisations can buy them.

With free software this problems do not exist anymore. Everybody can read the (usually well-documented) source understanding and improving it. It is not necessary to pay to get free software, so anybody can get it at anytime. This is a very important point: anybody would be able to access the development of science with just a computer and an Internet connection, in order to study it, first, then to improve it, later. This way the scientific knowledge would not be isolated in universities and big academies, but it could be anywhere.

The secrets, due to copyright, will not help the development of science.

5 Scientific Free Software Nowadays

Even if most of the people studying scientific subjects do not know there is some free scientific software available, there are some interesting projects around. Some of them are multi-platform, so it is possible to run them on Windows, Linux and MacOs, others have been developed only for use in Linux and have not been ported to any other platform, yet.

5.1 Multi-platform Software

The multi-platform free scientific software is very important for several reasons.

First of all, because of their ability to be multi-platform, they can run anywhere, thus they can be easily tested and proposed to be a new standard. Moreover, at the moment, these projects are the most advanced ones.

The high level language of Matlab became *de facto* a standard in several fields, and so two free clones have been made: GNU Octave [7] and Scilab [6].

The language in GNU Octave is exactly the same as Matlab, but it comes with no GUI and it does not have all the toolboxes Matlab has.

Scilab implements a language that is very similar to Matlab's but actually different. Scilab has a clone of Matlab's programme for simulation, Simulink: Scilab's clone is called Scicos. Scilab has several toolboxes, too, downloadable from its homepage, in the "Contributions" section.

Neither GNU Octave or Scilab are as powerful as Matlab, but they are a very good beginning for further improvements.

Gnuplot [14] is an advanced plotting programme with some other features about data processing.

For symbolic computation the main programmes are Axiom [9] and Maxima [8]. Axiom is a very powerful programme: it was born like a closed-source product from IBM, but then it was published and now it is free software. By default it comes with no GUI but it can be configured to work with TeXmacs [22], thus having an easier interface. The other programme, Maxima, is not as powerful as Axiom but it comes with a very simple GUI.

They are not as powerful as the commercial ones (Mathematica, Mathcad), but they are enough for students attending their first course of Mathematics.

Featflow is a CFD (Computational Fluid Dynamics) software; it is designed for solving incompressible flow problems in 2D and 3D, providing the necessary pre- and post-processing tools.

Between all the current projects, an outstanding one is SciPy [18], a Python package.

As known, Python [17] is a powerful, high-level and easy-to-use programming language. It is very fast, elegant, and it comes with a huge standard library for any kind of purpose: from web programming to manipulation of several (free) file formats.

SciPy is a collection of several science-aided programmes. As stated in the FAQ [18], several programmes have been taken from `www.netlib.org` written in C and Fortran, SciPy is actually a thin layer between the interface and these programmes. That is why, even if it is quite a young project (compared to the other ones), it is actually quite mature as it is based on tested and stable sources. Its core is written in C, only the interface is in Python, so it is as fast and powerful as a compiled language, while it is as easy and portable as only a scripting language can be.

It is meant to provide both an interactive programming shell (Mathlab style) and a powerful non-interactive programming: being a proper programming language its possibilities are virtually unlimited.

5.2 Software for Linux

The software that is developed specifically for Linux has the limitation that it is necessary to have a working Linux box to test them, and this is not such a common situation. Morover they can not be proposed as new standards, since only Linux users will be able to use them.

On the other hand, they can use all the free software that is developed for in Linux, such as the GTK [24] and QT [25] libraries to build GUIs. In any case, for this kind of projects, the GUI is a second aspect: the most important thing is to develop a powerful engine to solve the problems, then, using the libraries, it will be quite easy to build a GUI to interface it.

At the moment, the most outstanding project is the [10] GPL Electronic Design Automation: it has the aim to build several programmes suitable for any project related to electronics. Unfortunately there is no GUI, yet, so it is quite hard to test.

Another interesting project that can be tested right now is TkGate [11], a programme to simulate digital circuits.

Other projects providing a high level programming language aided for numerical analysis (Matlab-like) are Algae [19], Yorick [16] and Yacas [20]. Unfortunately, their syntax is quite different from Matlab and they do not offer much more than the other Matlab clones (GNU Octave and Scilab), so at the moment it is not really worth considering them. Yacas has the very good point that its language can be easily customised and it supports some symbolic computation, too.

6 A More Practical Approach

What has been done in the previous sections is an overview of free software, first for generic purpose, then in scientific applications.

Science is all about results, and so in this section a few cases are considered with a pure practical approach.

6.1 A Test About Numerical Calculations

In section 5.1 it was stated that there are several free programmes that were made with the same aim of Matlab: numerical computation with the use of matrices.

Here a test is shown about the performance of Matlab against its clones. The following software has been used for this test:

- Matlab 6.5
- GNU Octave 2.1.64
- Scilab 3.0
- SciPy 0.3.2 on Python 2.3.3

```
t0 = clock;
inv(rand(N));
etime(clock,t0)
```

Figure 1: Code for Matlab and GNU Octave

```
t0 = getdate();
inv(rand(N,N));
etime(getdate(),t0)
```

Figure 2: Code for Scilab

```
from scipy import linalg
from random import random
from time import clock

N = 1000
t0 = clock()
linalg.inv([ [ random() for i in range(N) ] for j in range(N) ])
print clock() - t0
```

Figure 3: Code for SciPy

Matlab is the proprietary standard software nowadays for any numeric application, GNU Octave and Scilab are its clones, SciPy with Python is a more generic purpose software that can be used for numeric calculus, too.

The test has been the following: the programmes have been used to generate a random N x N matrix and to calculate its inverse. The time any of these programmes needed to perform these two operations against the value of N has been stored and then plotted using Matlab. The test was done on a laptop with a Intel Pentium M processor running Linux Fedora Core 2 [23]; the frequency of the processor was set to 600 MHz by using cpuspeed [21], to slow down the calculations.

The code for both Matlab and GNU Octave is in figure 1, the one for Scilab is in figure 2, finally the one for ScyPy/Python in figure 3.

The results that have been obtained varying the value of N between 100 and 1000 have been plotted in 4. In figure 5 there is the plot of the same value, but a logarithmic scale has been used for the seconds, to "zoom" the behaviour for smaller values of N.

Since the matrix was random, the values for a fixed N were sightly different for any attempt; so for any N more than one attempt was made and the mean value of the results was considered.

The analysis has not been very accurate (only a few tests have been considered): the reason is that such an accuracy was not necessary as the differences are quite evident. Matlab is approximately two times faster than GNU Octave and SciPy, and 4 times faster than Scilab.

The test that was shown before gives an idea of how the programmes

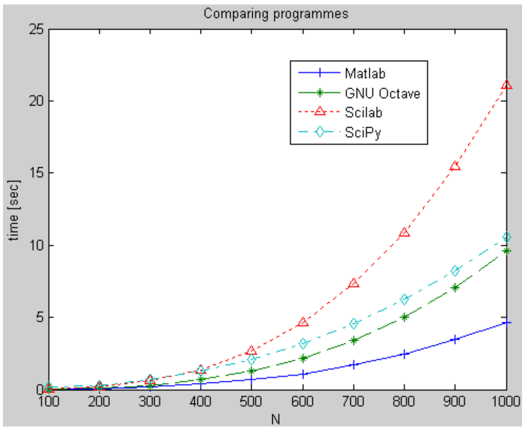


Figure 4: Computational time

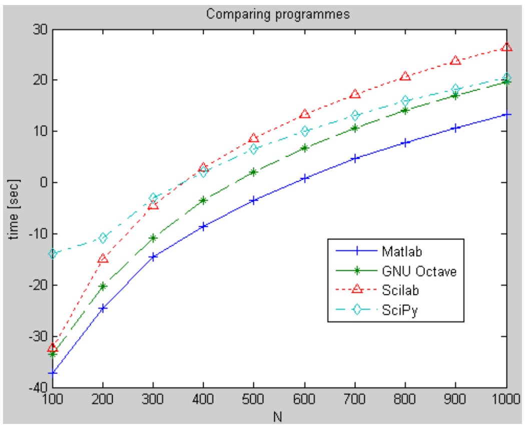


Figure 5: Computational time [dBs]

perform but, on the other hand, it does not considers several aspects.

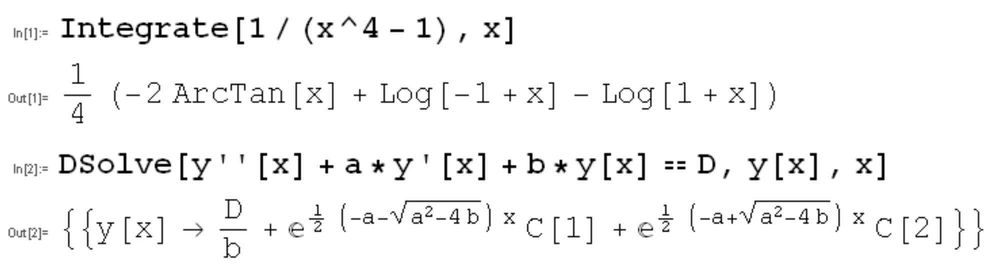
Matlab was born like a matrix manipulator. Then several extras were added (Simulink, Toolboxes, etc.) giving it the possibility to provide specific and advanced functions. Because of the structure of its core, Matlab can easily handle with efficiency anything that can be represented with a matrix of numbers, such as generic digital signals, images, etc.

As soon as any other kind of operation has to be implemented, Matlab suddenly becomes limited and difficult to use; this happens with string manipulation, external input / output with files and devices, etc.

With Matlab's clones, GNU Octave and Scilab, the situation is very similar: the approach is the same, they just have a slower core and less add-ons. Scilab has a clone of Simulink, Scicos, and more toolboxes than GNU Octave, that on the other hand has a faster core and a syntax more similar to Matlab.

SciPy has a completely different approach: it is an extension of a general-purpose high-level programming language, Python. The velocity Python (thus SciPy) manipulates matrices is almost as fast as Matlab, but Python has very good easy-to-use libraries for string manipulation, external input / output, importing any kind of file format, etc.

At the moment SciPy does not provide all the high specific scripts Matlab offers, but it is still a young project and it has several good points ensuring its future development: it is free, open-source and it is based on Python, a fast, powerful, easy-to-use and elegant language.



```
In[1]:= Integrate[1 / (x^4 - 1), x]
```

Out[1]= $\frac{1}{4}\,(-2\,\mathrm{ArcTan}[x] + \mathrm{Log}[-1 + x] - \mathrm{Log}[1 + x])$

```
In[2]:= DSolve[y''[x] + a * y'[x] + b * y[x] == D, y[x], x]
```

Out[2]= $\left\{\left\{y[x] \to \frac{D}{b} + e^{\frac{1}{2}\left(-a-\sqrt{a^2-4\,b}\right)x}\,C[1] + e^{\frac{1}{2}\left(-a+\sqrt{a^2-4\,b}\right)x}\,C[2]\right\}\right\}$

Figure 6: Mathematica output

```
(1) -> integrate(1/(x**4 -1),x)
(1) ->
         - log(x + 1) + log(x - 1) - 2atan(x)
   (1)  ------------------------------------
                           4
                                                 Type: Union(Expression Integer,...)

(2) -> y:=operator 'y
(2) ->
   (2)  y
                                                          Type: BasicOperator
(3) -> solve(D(y x,x,2) + a*D(y x,x) + b*y x = D,y,x)
(3) ->
                                   +---------+             +----------+
                                   |        2              |        2
                                 x\|- 4b + a   - a x   - x\|- 4b + a   - a x
                                 -------------------   ---------------------
                     D                    2                      2
   (3)  [particular= -,basis= [%e                   ,%e                     ]]
                     b
Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)
```

Figure 7: Axiom code

6.2 An Example About Symbolic Calculations

Another example is shown about symbolic calculation, using Mathematica 4.0 as proprietary software and Axiom 3.4 as the open source one.

The main topics of most of the basic courses about Mathematics in any university are indefinite integrals and linear differential equations. It could be helpful for any student to check its results of any exercise by using a software for symbolic computations. As an example, an integral and a linear differential equation will be solved by using both Mathematica and Axiom:

$$\int \frac{\mathrm{d}x}{x^4 - 1} \qquad\qquad y''(x) + ay'(x) + by(x) = D$$

in the equation, $y(x)$ is the unknown function and a, b and D are just constants.

Both Mathematica and Axiom (via TeXmacs) offer a graphical approach to introduce the expression but, to make it simpler the pure text input is used for both the programmes.

In figure 6 there is the screenshot of the output of Mathematica after having introduced the two problems to be solved, with their solution. It was necessary to use an image because the output is not in pure ascii.

On the other hand, in figure 7 the session of Axiom to solve the problems has been reported: since input and output are both in pure ascii, the session has been copied and pasted without using any image. As it can be seen results of the two programmes are the same, thus correct.

After this example it must be remembered that Mathematica is more powerful than Axiom in handling more complicated differential equations and integrals whose solution cannot be expressed with elementary functions. But, on the other hand, Axiom's tools are enough to help any student learning Mathematics in its first courses at the university.

6.3 A Personal Experience

While getting my degree in the UK, my final year project was about digital image processing for skin cancer recognition. I had to write a programme to extract some features from skin images, in order to understand whether a mole on the skin is cancer or not. I asked my supervisor whether I could write it using Scilab, but I was obliged to use Matlab; I did not know SciPy yet, but I do not think it would have been such a difference.

Two of the main tasks of my project were boundary recognition and tracing: I obviously started checking in Internet for any previous experience about it.

I found a (testing) programme that, for a given RGB picture of a mole, had to calculate the boundaries of the mole. It looked perfect: it was exactly what I was looking for! After a few tests, even if it was not working so good, I was interested in taking a look at the algorithm, just to take inspiration from that. Unfortunately they did not publish the source; I think the aim of the programme was mainly to show that the team who made it was working properly... Anyway, I was just wasting my time as I could not learn anything from that.

About boundary tracing, I needed an algorithm that, for a given shape in a black-white image, was returning a parametric representation of the boundaries. Matlab's Image Processing Toolbox did not provide anything about it, and I was really surprised (and pleased) to see that, instead, Scilab's Toolbox [12] has it! the script is called `follow()` and it does exactly what I was looking for. I made a conversion to Matlab (it took 5 minutes) and I included it in my project, referencing it properly.

By this example, taken from my personal experience, I hope that it was clear that closed-source software in science is not only useless, but damaging as well, while open source can speed up the development of science.

7 What To Do

Until now the discussion has been quite theoretical, a sort of a list of things that would be better to do in contrast to the things really happening. Then an example is proposed, showing how Matlab works better then its free clones. The reason is obvious: while Matlab is developed by researchers working full time on it, its clones are developed by people doing other jobs, working on them as a hobby; and it is impressive how with such less time and resources, they could make something working so good.

The question is *what to do now*: free software could provide good performance and speed up the development of science, but everybody knows it is always all about interests and money.

The only way to increase the importance of free software is to work where Linux was born: in the education and, in particular, in the University.

Nowadays computing is a necessary part in any kind of education.

Normally non-free software is taught arguing that, in the future, this is what has to be used in any application for any job. At the moment, most of the times this is actually true, but there are several other points for using free software. Teaching the basics of any programme, it is necessary to start from the basic tools, ignoring the most advanced features. Because of this, using the free equivalent of any programme would not cause any limitation.

As an example, there is no point in using Matlab to learn how to handle matrices in Matlab language, since it can be done with GNU Octave; there is no point in using Mathematica to solve simple integrals when Axiom can do it in the same way. This brings all the good points of free software (greater stability, portability, no need to buy any license, etc.) at no expense.

Any University is a centre of active culture like nowhere else, there are people with great knowledge about the subjects they study (researchers) and the most important resource of the world: students.

Students have to learn, it is their job, and looking into the source of a software is the best way to learn how it actually works. Studying the algorithm some free scientific programmes use, students will be able to improve them or add new features. This could be done for short-term assignments or as long-term final year projects.

As a short term goal, students would start to learn using free software, that means less piracy (as they would not need to crack software anymore) and less waste of money, as the University will need to buy less licenses.

As long term goal, people that studied at University will get a *modus operandi* depending on free software, so they will use it whenever they can. People will bring this knowledge of freedom to the companies they will work for, to the Universities from whom they will get their Ph.D., etc. spreading the free software. Nevertheless, free software will be improved, slowly becoming as powerful as the proprietary one.

8 Conclusion

This essay was mainly born as a vent against the system that was surrounding me. Then I started to collect some information about the issue I was concerned with, and I slowly understood that it was not only a matter of "philosophy", but it will be soon a matter of *need*. Knowledge is something that cannot be stopped and must not be stopped, in any way, otherwise the effects could be very hard.

I am not philosophically against non-free software, but I believe there must be the possibility to choose, and a collection of basic tools has to be available to give anybody the possibility to help the development of knowledge in modern science.

9 About This Document

The latest official version of this document is available for free download at www.alug.org.uk/articles/science.

Any comment or suggestion about this document is welcome at alessio.damato@email.it.

References

[1] The Gimp http://www.gimp.org/

[2] Mozilla, Firefox and Thunderbird http://www.mozilla.org/

[3] Openoffice.org http://www.openoffice.org/

[4] LaTeX http://www.latex-project.org/

[5] VideoLAN http://www.videolan.org

[6] Scilab http://scilabsoft.inria.fr/

[7] GNU Octave http://www.octave.org/

[8] Maxima http://maxima.sourceforge.net/

[9] Axiom http://axiom.axiom-developer.org/

[10] GPL Electronic Design Automation http://www.geda.seul.org/

[11] TkGate http://www.tkgate.org/

[12] SIP (Scilab Image Processing) Toolbox http://siptoolbox.sourceforge.net

[13] Featflow http://www.featflow.de

[14] Gnuplot http://www.gnuplot.info/

[15] SAL, Scientific Applications on Linux http://gd.tuwien.ac.at/opsys/linux/SAL/

[16] Yorick http://yorick-mb.sourceforge.net/

[17] Python http://www.python.org

[18] SciPy http://www.scipy.org

[19] Algae http://algae.sourceforge.net

[20] Yacas http://yacas.sourceforge.net

[21] CPUSpeed http://carlthompson.net/Software/CPUSpeed

[22] http://www.texmacs.org

[23] Fedora Core http://fedora.redhat.com

[24] GTK libraries http://www.gtk.org

[25] QT libraries http://www.trolltech.com

[26] http://scilinux.sourceforge.net/

[27] The Open Source Pages http://www.opensource.org/

[28] Cameron Laird, *Open source in the lab*, http://www-106.ibm.com/developerworks/linux/library/l-oslab/

[29] Linux Torvalds and David Diamond, *Just for Fun, The story of an accidental revolutionary*

[30] Christopher M. Kelty, *Free Software / Free Science* http://www.firstmonday.dk/issues/issue6_12/kelty/

[31] Eric Raymond, *The Cathedral and The Bazaar*, http://www.catb.org/~esr/writings/cathedral-bazaar/