Animated simulations in Computational Physics: A simple algorithm using freeware software

Braga, F. L.[1,2], **Fernandes, F. W.**[1]

[1]*Departamento de Física, Universidade Estadual de Campinas, Rua Sérgio Buarque de Holanda, No. 777, Cidade Universitária Zeferino Vaz, Barão Geraldo, Campinas, SP CEP 13083-859, Brasil.*

[2]*Departamento de Física, Instituto Tecnológico de Aeronáutica, São José dos Campos, Praça Marechal Eduardo Gomes, No. 50,Vila das Acácias, São José dos Campos, SP, CEP 12.228-900, Brasil.*

E-mail: leonciob@ifi.unicamp.br

(Received 24 November 2010, accepted 23 January 2011)

Abstract

We pretend to generate animated simulations on Computational Physics, developed using freeware software of the GNU distributions and some useful scripts developed by the authors. This work is a brief ``*How to*" where we are a showing an easy and simple way for building animated simulations of simple problems commonly used on physics graduation courses.

Keywords: Freeware software, Computational Physics Simulations, Animations**.**

Resumo

Pretendemos gerar simulações animadas de Física Computacional, desenvolvidas utilizando software das distribuições GNU e alguns scripts úteis desenvolvido pelos autores. Este trabalho é um breve manual onde estamos mostrando uma maneira fácil e simples para a construção de simulações animadas de problemas simples que são comumente usados em cursos de graduação em física.

Palavras clave: Software Gratuito, Simulações de Física Computacional, Animações**.**

PACS: 01.50.ht, 01.40.gb, 01.40.-d **ISSN 1870-9095**

I. INTRODUCTION

Created by Linus Torwalds, the operational system (OS) commonly called Linux OS has presented some advantages in the use of OS platforms. Its development is one of the most prominent examples of free and open source software collaboration. Actually, there are several Linux distributions based on these precursors, like Slackware, Debian and RedHat Linux. Initially, the Linux OS was based on a small open Unix-like user interface, and to improve its capability, GNU project was added in this OS operational platform. GNU was launched in 1984 by Richard Matthew Stallman [1], under GPL license, a GNU package has been furnishing several possibilities for the user, as running, copying and distributing many parts of one.

A large group of useful software application that can be used in several areas of knowledge, like education and research [2]. In this sense, many activities using free software can be used for teachers as a helpful tool. The most known Linux OS, as Ubuntu [3] and Fedora [4], have a group of packages with no commercial licenses in developing of knowledge. As an example, there are graphical applications, as a useful framework, to visualize and edit pictures and graphics. By the way, programs like Gimp [5] have been extensively used in manipulating radiology images and other ones with high quality processing image. GnuPlot is another example of graphical processing software, Ref. [6]. It is much used and known between professionals in the scientific community. It presents many frameworks to visualize mathematical functions and data interactively. Thus, it is very useful for researchers, graduate and post-graduate students. However, it can aid high school teachers to demonstrate complicated functions commonly known in physical and mathematical topics, where they are very teached for undergraduate students. It has many frameworks to plot 2D and 3D graphs. Further that, animated movies reveals a great idea to understanding the complicated physical phenomena in a high pedagogical concept [7, 8, 9]. In this sense, the two Linux OS cited before have several packages in editing, visualizing and developing movies using several known formats.

Due to the GPL license of some packages in some Linux OS, this operational system provides many possible improvements and capabilities that a closed package does not. Concepts in certain programming language like C++, FORTRAN and others are much useful for computing users.

Shell script, for example is a tool language commonly known in many OS Linux distributions, like Slackware and others. That can be used by administrators and single users to get advances of the OS used, making the execution of certain programs by the use of simple line commands set [10], as making connection of programs in Fortran, C++ and graphical applications.

Understanding that is very important to apply the functionalities presented in this work. Where we pretend to show a simple and useful way to build animated movies of physical phenomena using this freeware softwares.

II. THE SCRIPTS

The treatment developed in this work is a very useful platform for any Linux OS. But in general the authors use Debian Kernels distributions, as the following OS: Debian, Ubuntu, Kubuntu, and Kurumin. In any one of their versions.

A script is nothing more than a text file in which we have introduced lines of command that must be performed by the computer. As we are working on Linux OS, we have these scripts generally contains commands to be executed in a shell terminal.

To build a script is very simple. After entering the commands to be executed, we save this file with a name and then change the permission of this file. We turn it into an executable file. To make this, we use the *chmod* command. This command is used to change the permissions of certain files. It must be execute in the terminal. The first line of a terminal, shows the used machine's name, the user's name and the prompt to type the line commands. In the following example the name of the machine used was *ERA*, and the user was *user*. Then, the command to be executed should be:

$$user@ERA:/\$ \ chmod\ a + x\ script,$$

in this case, transform the script file, which is a text file, into an executable file. To run it in a terminal we use the command

$$user@ERA:/\$ \ ./script,$$

so, all the commands that are within the script file will be executed sequentially.

The software for the simulation of physical systems of interest, in our case must be made in FORTRAN language. In the following examples the programs were edited in Fortran-90. The compiler used to generate the executable files of these software was the compiler GFotran, provide by the GNU distribution. The compilation of such software should also be performed via terminal. For example, the software that will generate a simulation of a Brownian Motion [11] in 3 dimensions was named *BM3D.f90*, and must be compiled as follows:

$$user@ERA:/\$ \ gfortran\ BM3D.f90 - o\ BM,$$

in this case, the executable is *BM*.

In general, the software used to generate simulations of physical systems of interest has basic output text files that must be plotted using any graphic software. In our work, we use the software GnuPlot. This software performs graphic plots in 2 and 3 dimensions via line commands executed in a terminal. For example, the program *BM3D.f90* generates an output file containing the coordinates (x,y,z) of the particle performing a Brownian Motion. To plot this file we must perform a sequence of commands. First we initialize GnuPlot,

$$user@ERA:/\$ \ gnuplot,$$

then, we set the output figure format

$$gnuplot > set\ terminal\ png,$$

and set the shape of the graphic, making it symmetric,

$$gnuplot > set\ size\ square,$$

after that we adjust the range of x axis to be between [-50,50]

$$gnuplot > set\ xrange\ [-50:50].$$

Similarly, we can do this for the other axes of the graphic. Now we must adjust the name of the figure,

$$gnuplot > set\ output\ '\ 1.png'.$$

In this case the file name will be *1.png*. Finally, we execute the command to plot our graph in 3 dimensions:

$$gnuplot > splot\ 'XYZ.png'\ w\ points\ 2.$$

It is noteworthy here that, the *splot* command does 3-D graphics and mapping surfaces, also called *countor plots* where the third axis of the graphics are associated with colors. The name of the text file here is *xyz.dat*, the command *w points 2* is indicating the type of plot we want, in this case intend to make a plot where the points are not connected, so we put *w points*, and symbols associated with each of these points is predetermined by the format 2. Indeed, GnuPlot has a list containing the numbers for each symbol. When we want the points connected, use the command *w lines*. If we wanted to make a 2-D plot just change the *splot* command by *plot*.

As results, using this sequence of commands, it will generate a graphic file named *1.png*. Now, if we have two *png* figures as the one built before, we can use the command *convert* to merge the two images in an animated *gif* file that can be ran in a web browser or an appropriate software for viewing animated *gifs*, similar to the program Gifsicle. The command *convert*, is part of Imagemagick package that can be obtained by default in the software manager, Synaptic, common in Debian Kernels distributions. Assuming that we have only two images *png* in the same folder, to convert them into a *gif* file just require that we run in terminal the following command

$$user@ERA:/\$\ convert\ \ *.png\ ANI.gif,$$

i.e., we convert all *png* images on an animated *gif* named *ANI.gif*. If there are 100 *png* pictures all they will be packaged in a *gif* with 100 frames. Thus, the basic idea of making an animation of physical systems is the use of scripts to generate these figures that would be snapshots of the physical system during its time evolution. And finally convert them into an animated *gif*.

In general, it is of certain interest convert this animated *gif*s into another video format, such as *avi* or *flv*. For this purpose, we use the tool *ffmpeg*. The basic command that performs this conversion is,

$$user@ERA:/\$\ ffmpeg - i\ ANI.gif - s\ ANI.avi,$$

thus, convert the file *gif* for the format *avi*. Other conversions can still be made. However, these two formats are recognized worldwide on any software to view videos. Even on Linux OS and windows platforms.

III. APPLICATIONS

We present here two applications of making animations. The first case concerns the distribution of heat on a metal plate that is in contact with two heat sources, and the second case is to display a random walk in 3-D.

For the first case we have the physical problem is basically the resolution of the diffusion of heat [11];

$$\nabla^2\varphi = \frac{1}{k}\frac{\partial\varphi}{\partial t}, \qquad (1)$$

that, in two dimensions can be written as:

$$\frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} = \frac{1}{k}\frac{\partial\varphi}{\partial t}, \qquad (2)$$

k is related to the diffusion constant of heat in the plate. So we can use the Gauss-Seidel [13] to determine the flow of φ as the boundary conditions. In this example we have a metal plate in contact with two heat sources, covering two of its edges. To accomplish the above resolution of the partial differential equation (PDE), we made a software in Fortran-90 which we call Diffusion.f90. This software was created to save a text file position (x, y) of the mesh used to perform the numerical calculation. And the heat value φ associated with each point for each iteration. To carry out the rescue of such data in text files without the need to name all the time the output files, we use a trick. By default, the FORTRAN language save the output files as *fort.i*, being the index of the output file mentioned in the command *write (i,*)*, that *i* must necessarily be an integer. Because this command was in the loop iterations of the program, we use the same counter for the iteration steps and to generate the output files.

Now, after we had the output files containing the information about the points and the heat, we would need to generate a script to plot all the figures generated. For this, we set up an appropriate script for GnuPlot similar to that shown below:

```
set dgrid3d 100,100
set size square
set contour base
unset surface
set pm3d
set view 0,0
set cntrparam levels 20
set terminal png
set output '1.png'
splot 'fort1.png' w lines
set output '2.png'
splot 'fort2png'  w lines
⋮
set output '1000.png'
splot 'fort1000.png'w lines
```

This script contains several commands that are useful for making a *contour plot*. The command *dgrid3d* generates a matrix which contains information on the sites used generally in simulations of PDE's and Z axis is related to a given color, which is directly linked, in our case to the heat φ. *100,100* adjust a set of 100×100 sites used to perform the plot. The command *set size square* adjusts the output of the figure to be symmetrical and centered. The command *set hidden3d* hide sites for which we see only the colors and the boundary surface. The command *unset surface* disables the surface area in three dimensions. The command *set pm3d* loads the libraries of GnuPlot for making contour-plots. *Set view 0,0* sets the vision for seeing the chart on the plane *(x, y)*. And the command *set cntrparam levels 20* adjusts how many level curves, in this case 20 curve lines. Note that, the script contains commands to plot 1000 *png* figures. After the made-up script, it should be named, call it *SGNUPLOT* and run it using a terminal. The command to do this is

$$user@ERA:/\$\ gnuplot\ SGNUPLOT.$$

After this command is executed, it will generate 1000 *png* figures that must be converted into an animated *gif.* Finally, we convert the *gif* into an *avi* file using the command suggested in section II. And using the concept of *script* we can merge all of the process mentioned above in a single script as

```
gfotran Diffusion.f90 − o DIF
./DIF
gnuplot SGNUPLOT
convert *.png ANI.gif
ffmpeg − i ANI.gif − s ANI.avi
```

Naming this file as a *SCRIPT,* convert its permissions into an executable file and run it. The results for the simulations can be seen at the snapshot in FIGURE1.

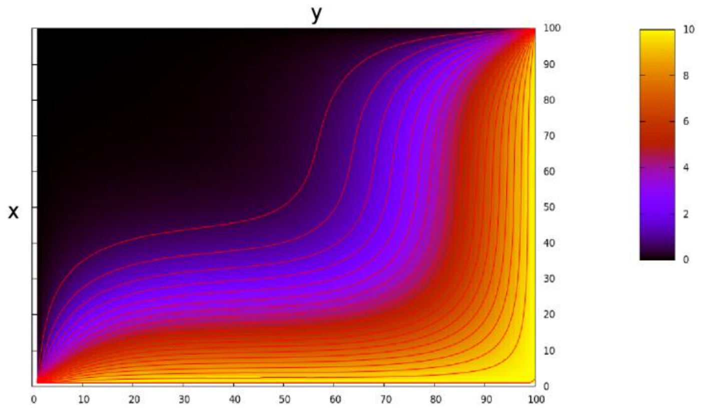

FIGURE 1. Figure showing a frame of the simulation for diffusion of heat, the heat sources are two of the edges of the plate are presented and eat green cyan.

The same procedure can be used for making an animation of a random walk with constant path in 3-D. The algorithm used to make a software that simulates a 3-D random walk [11] with constant path is quite simple. Just random draft two angles, θ and ϕ respectively in the intervals of $[0,\pi]$ and $[0,2\pi]$. To execute a step in that direction the following equations can be used

$$x_{i+1} = x_i + a\sin(\theta)\sin(\phi)\,,$$
$$y_{i+1} = y_i + a\sin(\theta)\cos(\phi),$$
$$z_{i+1} = z_i + a\cos(\theta)\,.$$

Here *a* is the step size.

In our case we have prepared a software named *RW3D.f90*, which saves at every 100 iterations the coordinates of the particle and consider 100,000 iterations. The plotting of the figures was made using a *SGNUPLOT2* slightly different from before. This script is shown below:

```
set dgrid3d 100,100
set size square
unset surface
set view 0,0
set terminal png
set output '1.png'
splot 'fort1.png' w points
⋮
set output '10000.png'
splot 'fort1000.png'w points
```

The conversion of the *png* files as well as visualization of the images was made in the same way as the previous case. In order to facilitate the handling of this program we made a script that also run all commands used. The FIGURE 2 presents a snapshot for the simulation of a random walk.

IV. CONCLUSIONS

The Animated simulations of physical systems always help understanding the mechanisms involved. This paper presents an efficient way using free software to produce these animations. The procedures shown above are useful tools for teachers and researches.

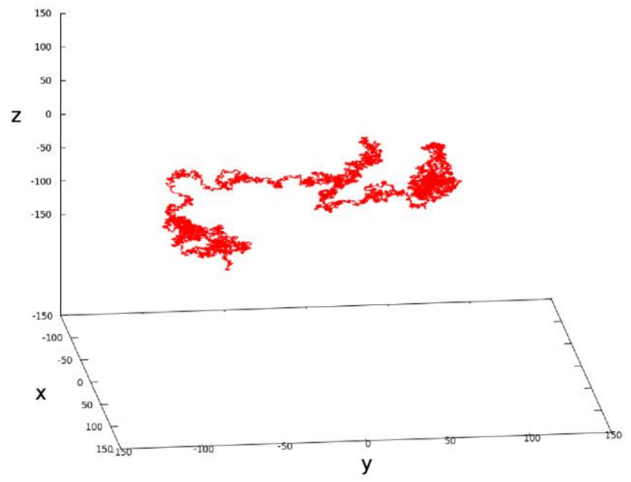

FIGURE 2. Figure showing a snapshot for a particle performing a random walk.

REFERENCES

[1] *The Gnu Operating System homepage*, http://www.gnu.org/ Accessed 8 March 2010.
[2] Wolf, M. J., Miller, K. W. and Grodzinsky, F. S., *The ethics of designing artificial agents*, Ethics and Information Technology **10**, 115–121 (2008).
[3] *Ubuntu Operating System homepage*, http://www.ubuntu-br.org, Accessed 8 March 2010.
[4] *Fedora Operating System homepage*, http://fedoraproject.org/, Accessed 8 March 2010.
[5] Solomon, R. W, *Using freeware software on reontology*. American Journal of Roentgenology **192**, 330–334, (2009).
[6] Gnuplot Application homepage, http://www.gnuplot.org/ Accessed 8 March 2010.
[7] Landau, R. H., *Web, education, and computational physics; good, bad, and ugly*, Computational Physics Communication **121**, 550–556, (1999).
[8] Sheth, C. V., *Computational physics programs in research and teaching - an African experience*, Computer Physics Communication **147**, 590–594, (2002).
[9] Fox, G. C., *From computational science to internetics: Integration of science with computer science*, Mathematics and Computers in Simulation **54**, 295–306, (2000).
[10] http://focalinux.cipsga.org.br, Accessed 8 March 2010.
[11] Vicsek, T., *Fractal Growth Phenomena*, (World Scientific Press, Singapore, 1999).
[12] Sears, F. W., *Thermodynamics, Kinetic Theory, and Statistical Thermodynamics*, (Addison Wesley Longman Press., Massachusetts, 1975).
[13] Teukolsky, S., Vetterling, W. and Flannery, B., *Numerical Recipes: The Art of Scientific Computing*, (Cambridge University Press., New York, 1996).