Gnuplot animations as a Physics teaching tool

Ananda Dasgupta
Department of Physics, Indian Institute of Science Education and Research – Kolkata.

E-mail: adg@iiserkol.ac.in

(Received 25 July 2011; accepted 17 December 2011)

Abstract

Computer based animations can go a long way towards enhancing student understanding of complex topics. In this article we will explore the use of simple scripts (most no longer than a few lines) using the open source plotting program **gnuplot** to develop animations. We feel that mastering this simple device will add considerably to the arsenal of an educator in her efforts towards greater student comprehension.

Keywords: ICT, animations, Waves and Vibrations.

Resumen

La computadora basada en animaciones puede recorrer un largo camino hacia la mejor comprensión de los estudiantes en temas complejos. En este artículo vamos a explorar el uso de scripts simples (la mayoría no son más que de unas pocas líneas), utilizando el código abierto trazado **gnuplot** programa para desarrollar animaciones. Creemos que el dominio de este sencillo dispositivo aumentará considerablemente el arsenal de un educador en sus esfuerzos hacia la mayor comprensión de los estudiantes.

Palabras clave: Las TIC, las animaciones, ondas y vibraciones.

PACS: 01.50.ht, 43.20.Bi, 02.30.Nw **ISSN 1870-9095**

I. INTRODUCTION

It is well known that an animation can greatly enhance student comprehension of complicated physics concepts. Today the educator has a vast repository of animation software, both commercial and non-commercial at her disposal. While some of these software is of excellent quality, one is often faced by the problem that using them limits oneself to the features their creators had in mind. So, while a java applet available on the web may be of very high quality, it may not exactly be what the educator needs. In such a situation, the best solution for the educator would be to write her own software. This has the obvious problem that not everyone can afford to spare the time and effort necessary to learn a full-fledged programming language. Even for those who already know the language, writing a new program each time a new concept has to be demonstrated to the class may be very demanding. In this paper, we would like to call the educator's attention to a simple, free solution that is openly available. This involves using the open source plotting program ***gnuplot*** [1] which has excellent capability of producing quick animations with only a few lines of code.

While one may argue that this solution still demands that one learns gnuplot scripting, it certainly requires much less effort to learn few commands for a limited function (in this case, plotting) program than to learn the intricacies of a general purpose programming language. In addition, one can create excellent animations with only a few lines of code (most of our scripts are about ten lines long or less!), which saves a lot of time, allowing the educator to devote more attention to other aspects of her teaching. On top of these, the other major advantages of gnuplot are

- It is open source
- It is free of cost
- It has versions which run on most operating systems.

In Section II we will provide a very brief overview of scripts in gnuplot. Here we also discuss the **reread** command - which is critical for writing animation scripts. In the next section we will we will demonstrate how to use these concepts to write simple scripts that will perform simple physics animations. Our examples will be from the field of waves and vibrations – which is a topic whose teaching benefits greatly from proper visualization. It should be very easy for the reader to use the same principles to write animation scripts suitable for whatever course she is teaching.

II. GNUPLOT SCRIPTS AND ANIMATIONS

Gnuplot is a rather large and intricate open source plotting program. The current gnuplot manual [2] runs into 224 pages! It is neither possible, nor necessary to outline all its features in this article, especially since there are lots of

wonderful resources, apart from the manual, available freely [3, 4]. Indeed the online help [5] available with the gnuplot program is detailed enough for most purposes. In the following we assume that the reader is familiar with the basics of gnuplot and is able to use it for simple plotting tasks.

In most plotting tasks, you would have to enter several gnuplot commands sequentially in order to arrive at the desired final graph. The alternative to writing the commands out, one after the other, at the gnuplot prompt, is to write them in sequence in a text file. An example of such a file called, say, '**quickFourier.plt**' (the .plt extension is not mandatory, but is the standard convention) may be

```
# file 'quickFourier.plt'
# Plots the first few terms of the Fourier expansion
# of a square wave
plot sin(x)
pause 2
plot sin(x)+sin(3*x)/3
pause 2
plot sin(x)+sin(3*x)/3+sin(5*x)/5
pause -1
```

Note that # is the gnuplot comment symbol – anything following # is ignored by the gnuplot interpreter. One way of running this script is simply typing **gnuplot quickFourier.plt** at the linux command prompt. The **pause 2** command makes gnuplot pause for 2 seconds before carrying out the next instruction. The final **pause-1** command makes gnuplot pause until the user presses any key. Without it, the final graph will disappear as soon as the script ends (which is immediately!). Of course, the user may feel the default 100 sample points too small. Then she could start gnuplot in the interactive mode and use the commands

```
gnuplot> set samples 1000
gnuplot> load 'quickFourier.plt'
```

One command that is essential for writing gnuplot animations is **reread**. When gnuplot encounters the reread command, it essentially loads the script again. So if you have a script '**easyWave.plt**' that says

```
plot sin(x-ct)
ct = ct+sp
reread
```

(note that **ct** above is a single variable, not) and at the gnuplot prompt type

```
gnuplot> ct = 0
gnuplot> sp = 0.01
gnuplot> load 'easyWave.plt'
```

will cause a sine wave to move slowly across the screen. The catch is that the unconditional reread statement puts the script into an infinite loop – and the only way out is to kill the program (by hitting **Ctrl-C**). This is easily remedied by replacing the last line of the script by

```
if (ct<10.0) reread
```

which will make the script end once the variable **ct** reaches the value 10.0. You cannot, however, run the **easyWave.plt** script by directly issuing the command **gnuplot easyWave.plt**, since the script will raise errors about the undefined variables **ct** and **sp**. A way out is to write another script **initEasyWave.plt** containing

```
ct = 0
sp = 0.01
load 'easyWave.plt'
```

and run this script by typing **gnuplot initEasyWave.plt** at the command prompt.

With this introduction we are all set to write more complex animations.

III. A COUPLE OF EXAMPLES

It is possible to write a surprisingly large collection of complex animations, using a little more than the basic methods outlined above. We will illustrate just two of them here.

A. Waves in a semi-infinite string

It is well known that small amplitude waves in a stretched string satisfies the 1-dimensional wave Eq. [6]

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{v^2}\frac{\partial^2 u}{\partial t^2}, \tag{1}$$

where $v = \sqrt{T/m}$ is the speed of the wave (T and m are the tension in the string and its mass per unit length, respectively). Eq. (1) has the d'Alembert Solution [7]

$$u(x,t) = f(x - vt) + g(x + vt). \tag{2}$$

The functions f and g are determined by the initial as well as the boundary conditions. In particular, the initial conditions

$$u(x,0) = u_0(x), \frac{\partial u}{\partial t}(x,0) = v_0(x), \tag{3}$$

is satisfied by[8]

$$f(x) = \frac{1}{2}\left(u_0(x) - \frac{1}{v}\int_{x_0}^{x} v_0(x')\,dx'\right), \tag{4a}$$

$$g(x) = \frac{1}{2}\left(u_0(x) + \frac{1}{v}\int_{x_0}^{x} v_0(x')\,dx'\right). \qquad (4b)$$

Let us consider waves propagating in a semi-infinite string stretching from a fixed end at $x = 0$ to $+\infty$. Here the initial conditions (3) are valid only for ≥ 0 , and thus, (4a) and (4b) will help us to specify the functions and only for positive x . This is not a problem for the left-going wave $g(x + vt)$, but we do need the value of $f(x)$ for negative x as well. This can be obtained from the boundary condition $u(0, t) = 0$, which is valid for all positive t. This means that for $x \geq 0$, we have[9] $f(-x) = g(x)$.

Animating the motion of this wave can be done by running **gnuplot initWaveString.plt**, where the initialization scripts **initWaveString.plt** contains the code

```
set sample 3000
h = 1.0; a = 1.0; D = 10; ct = 0.0; sp = 0.08
set yzeroaxis
set title "Wave in a semi-infinite stretched string, fixed end" font "Helvetica,16"
y0(x) = x<0?1/0:x<=D?0:x<D+a?h*(x-D)/a:x<D+2*a?h*(D+2*a-x)/a:0
V(x) = -y0(x)
f(x) = x<0?-g(-x):(y0(x)+V(x))/2
g(x) = x<0?1/0:(y0(x)-V(x))/2
y(x) = x<0?1/0:f(x-ct)+g(x+ct)
load 'waveString.plt'
```

Note that here we have chosen the initial conditions so that we begin with a wave traveling towards the left and at, the wave is shaped like a triangle (Note that in the code above, V(x) stands for). The actual animation is carried out by the code **waveString.plt**, which is very small in comparison:

```
plot [][-2:2] f(x-ct)-.95 t "f" ls 4, g(x+ct)-1.05 t "g" ls 1,y(x) t "y" ls 3
ct = ct +sp
if (ct <20.0) reread
```

If you run this code, you should be able to see the pulse travel towards the fixed end and finally get reflected back with an accompanying phase reversal.

B. Partial Sums of Fourier Series

Our second example is not really an animation. It uses the recursive function definition allowed by gnuplot to demonstrate how the partial sums for the familiar Fourier series expansions for the square wave and the triangular wave[9] approach the limit as the number of terms increase. This also consists of two scripts, **initFourier.plt**, which contains

```
#Fourier series for the square wave and the triangular wave
sq(x, n) = n == 0?
sin(x):sin((2*n+1)*x)/(2*n+1)+sq(x, n-1)
tr(x, n) = n == 0?
cos(x):cos((2*n+1)*x)/(2*n+1)**2+tr(x, n-1)
set sample 1000
n = 0
load 'Fourier.plt'
```

and **Fourier.plt**, containing

```
t = sprint ('Fourier Series summed to %d terms', n+1)
set title t font 'Helvetica,16'
p [-10:10] [-1.5:1.5] 4/pi*sq(x, n) t 'square', 8/pi**2*tr(x, n) t 'triangular'
print n+1, ' terms'
n = n+1
pause -1 "Hit enter to continue"
if (n<50) reread
pause -1 "Hit enter to quit"
```

The interested reader will find many more examples of animation codes at the author's course website [10]. There she will also find instructions on how to convert the animations into standalone movies. The latter are useful if one is going to use a system in the classroom where gnuplot is not installed. Of course, the advantage that one has in the scripts that one can explore the effects of changes in parameters and/or initial conditions by changing one or two lines of code is largely lost when previously generated movies are used.

IV. CONCLUSIONS

The codes above should illustrate that it is reasonably simple to write scripts that animate important physical concepts using gnuplot. While it is possible to construct much more sophisticated plots and animations using gnuplot (for examples you can see [11]), it is important to note that the aim here is to provide the instructors with a tool which can be used to create workable animations, which are easy to modify in a few minutes. The author feels that mastering this tool will aid an instructor considerably in her aim to help her students comprehend complicated material.

REFERENCES

[1] http://www.gnuplot.info/ Visited on June18, 2012.
[2] http://www.gnuplot.info/docs_4.4/gnuplot.pdf Visited on June18, 2012.
[3] http://www.duke.edu/~hpgavin/gnuplot.html Visited on June18, 2012.
[4]http://physicspmb.ukzn.ac.za/index.php/Gnuplot_tutorial Visited on June18, 2012.
[5] To access the online help for plotting a graph, typing help plot at the gnuplot prompt is usually adequate.

[6] Wallace, P. R., Mathematical analysis of Physical Problems, (Dover Publications, New York, 1972).
[7] Arfken, G. B. and Weber, H. J., Mathematical methods for Physicists, 6th Ed. (Academic Press, New York, 2005).
[8]http://www.iiserkol.ac.in/~ph221/animations/index.html Visited on June18, 2012.
[9] http://gnuplot-tricks.blogspot.com/ Visited on June18, 2012.