Gnuplot Basics

José Mauricio Sevilla[1]
Andrés Camilo Sevilla[1]

[1]Universidad Nacional de Colombia

5 of March of 2016

Introduction

Until now, we've been trying to make the computer do some calculations, but, how do we make plots of those data?

we're going to use an external program, not included in `C++`, eventually we're going to learn basic usage of `python` which includes a module to make plots, so in the same code we can generate the plots we want.

One of the most common, easy and powerful tools is called `Gnuplot`, in this presentation, we're giving some help about the usage, characteristics, and options so we can start to use it, and at some point, we'll show how to use this to plot a `C++` output using the bash.

Installing

First we need to install gnuplot, the current version of this software is the 5.0, so we're going to use that one[1], on linux the installing is very easy, we just need to open a terminal (ctrl+alt+t), and type

```
sudo apt-get install gnuplot5
```
[2]

And after typing the password we'll have gnuplot installed.

On Windows we can find the installer at http://www.gnuplot.info/.

[1] The most noticing difference is the colors used by default.

[2] We must have updated the repositories, if we don't we must use the gnuplot package and the 4.6 version will be installed.

Terminals

`Gnuplot` can produce a plot in the terminal, or opening other windows, or even saving in external files the plot, that's done using the *terminals* it has, we're going to start understand them with some examples, first we're going to open a terminal and change the directory, let's remember how that's done.

First of all, let's open a terminal (`ctrl`+`alt`+`t`), after that, let's change the directory by using the `cd` command, if we want to create a new one, we can do it by using the `mkdir` command, for example:

```
cd Documents
mkdir plots
cd plots
```

Now let's open gnuplot, so, in the terminal, just type gnuplot, and we'll have something like

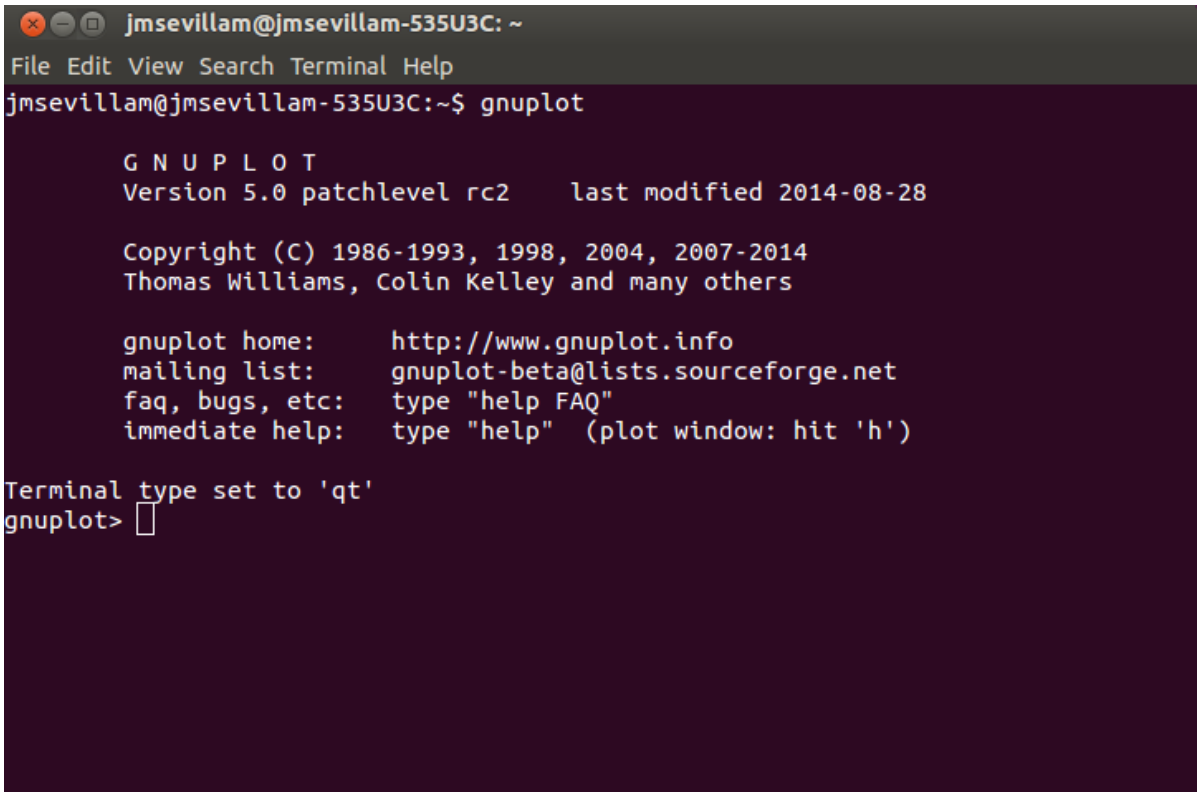

Figure 1: Gnuplot opening view.

Like we can see, the terminal used by default[3] is `'qt'`, that one open an external window to show the plot, let's try. for example, if we want to plot a function like

$$f(x) = x^2 \tag{1}$$

we have to write in the terminal

```
plot x**2
```

[3] On this `gnuplot` version

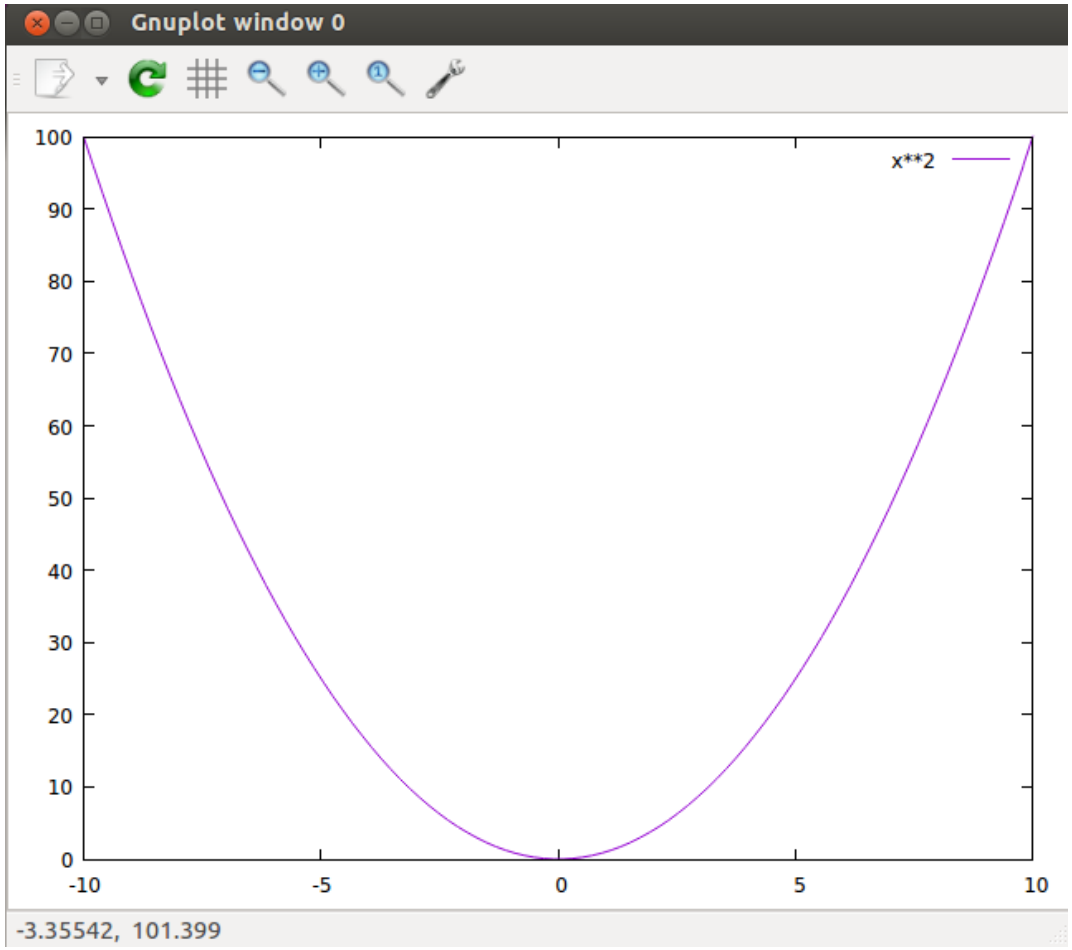

Figure 2: First plot

this interface has some options like save in pdf or png, but we'll discuss them later.

The most common terminals are shown next

- qt
- png
- jpg
- pdf
- postscript

For example, in the `pdf` no window is opened when the plot is done, but is saved in a pdf file instead.

How to use other terminals[5]

the first thing we have to do is change the terminal, and we do it by typing, for example to pdf

```
set term pdf
```

after that, if we try to plot just like we did before, the result won't be the expected, so we have to tell `gnuplot` where we want to plot, in which file.

```
set output 'plot.pdf'
```

The important part here is that the name of the file must finishes with the `.pdf` extension, due to we are using this terminal. then, we can plot anything we want, and apparently nothing happened, but a file named `plot.pdf` was created and has the information of the plot, but we must close it, and we do it changing again the terminal like[4]

```
set term qt
```

[4] in `Windows` is needed a `replot` after change the terminal.

[5] This will be used later

Binary Operators

Like we saw, we did not use the '^' character to make the square, so in the next table we'll find some of the binary operators used by `Gnuplot`

Symbol	Example	Explanation
+	a+b	Addition
-	a-b	subtraction
*	a*b	Multiplication
==	a==b	Equality
!=	a!=b	Inequality
<	a<b	Less than
<=	a<=b	Less than or equal to

Cuadro 1: Binary Operators

There are more of them, but we'll introduce them if it's needed.

Functions

Now, we may wonder, what else can we plot using gnuplot? These are some of the available functions in gnuplot

Function	Explanation	Function	Explanation
abs(x)	Absolute value[6]	log(x)	Natural Logarithm
sin(x)	Sine	sinh(x)	H. Sine
cos(x)	Cosine	cosh(x)	H. Cosine
tan(x)	Tangent	tanh(x)	H. Tangent
asin(x)	Arcsine	asinh(x)	Inv. H. Sine
acos(x)	Arccosine	acosh(x)	Inv. H. cosine
atan(x)	Arctangent	atanh(x)	Inv. H. Tangent
sqrt(x)	Square Root	besj0(x)	J_0 Bessel Function
exp(x)	Exponential	besj1(x)	J_1 Bessel Function
airy(x)	Airy Function	besy0(x)	Y_0 Bessel Function

Cuadro 2: Some functions in Gnuplot

External Files

Until this point, we've seen how to plot functions and save them, now, we're going to learn how to plot data saved in external files.

Let's start with the file, it must be written in columns separated by spaces, and the decimal point is made with a dot (.), for example in the file `data.dat` we have

1	1.0
2	1.3
3	1.6
4	1.7
5	1.9
6	2.0
7	2.2

Cuadro 3: `data.dat`

to plot the file `data.dat` the file must be in the same folder than the terminal, and just have to write

`plot 'data.dat'`, and we'll get as a result:

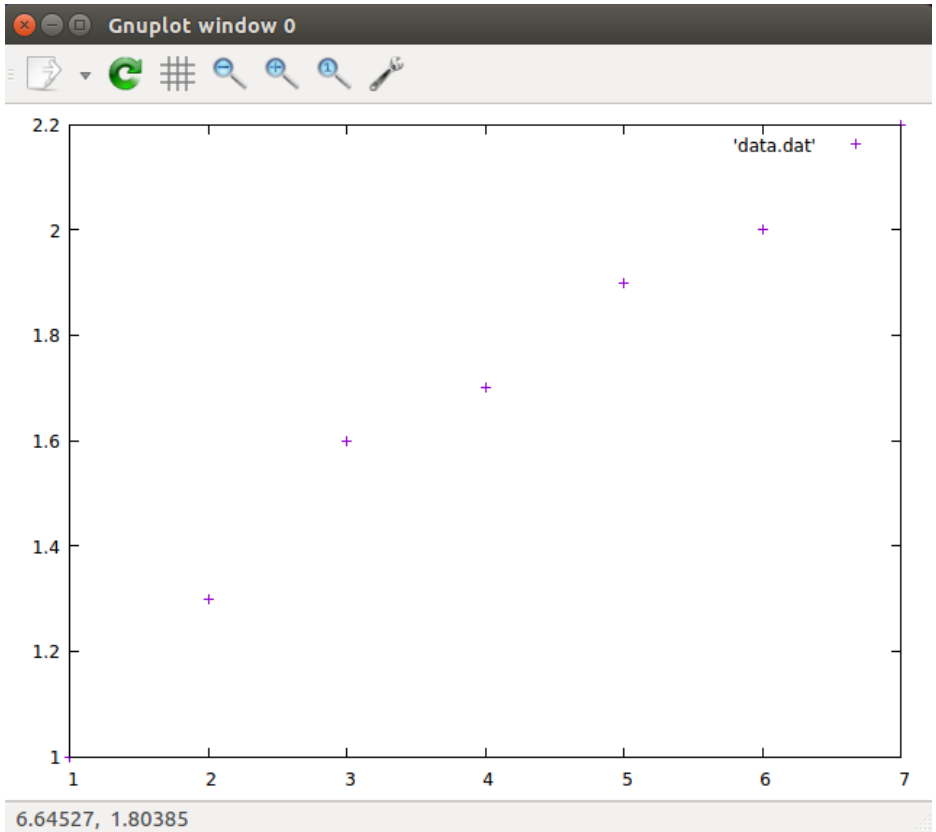

Figure 3: plot of `data.dat`

It's of our interest saving all the plots we make, so to do it, we're going to change to the `pdf` terminal, although there are a lot of them, we choose `pdf` due to using it, we do not have any resolution problem and it's compatible with LaTeX.

First, we must change to a pdf terminal, and create (or replace if already exists) a file where we want to save:

```
#We change to pdf terminal
set terminal pdf
#Then we select plot1.pdf to save the plot
set output 'plot1.pdf'
```

Now, gnuplot will plot in plot.pdf and not in the window as it used to.

if we try to open the file, it may not work, but it's due to gnuplot is still using it, so we must close it.

It's is an easy but important part, we always must close it before we can use it as much in `Windows` as in `Linux`.

We close the file by changing the terminal, so, for example:

```
set terminal qt[a]
```

[a] In `Windows`, is necessary use another plot command to close the file, for example `replot`.

Let's try it

As a result we get a new file in the working directory named `plot.pdf`, and it goes as follows:

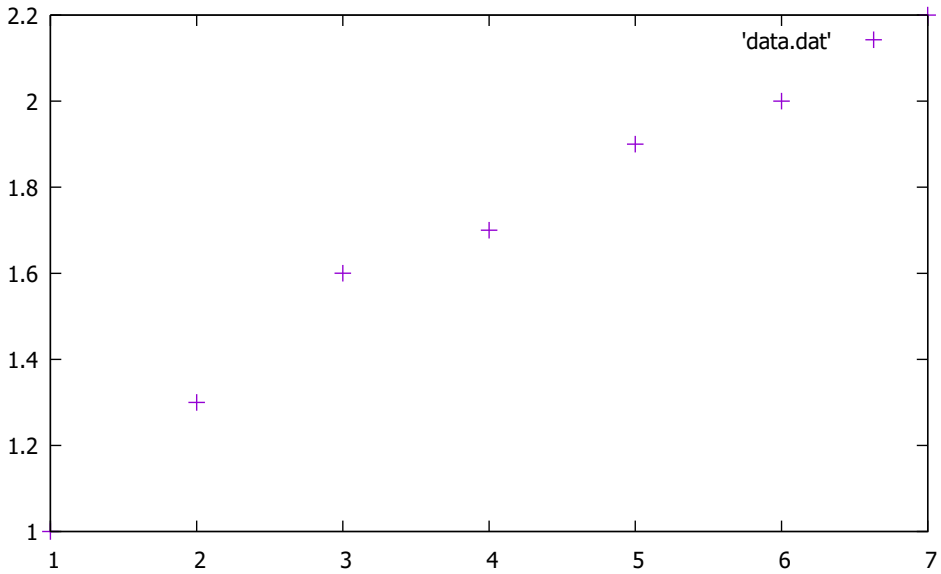

Figure 4: Plot saved in `plot.pdf`

After here, we're going to show the plots saved in pdf's files, otherwise will be specified.

Now, we're going to see how to change some aspects of the plot, like color, title, labels, scales and even size.

Fist, in the legend we see `'data.dat'`, that's the file name where the data is saved, but, what if we want to specify what represents our data?.

Let's suppose that our data represents a measurement of a experiment A, and the y axe represents length and the x time.

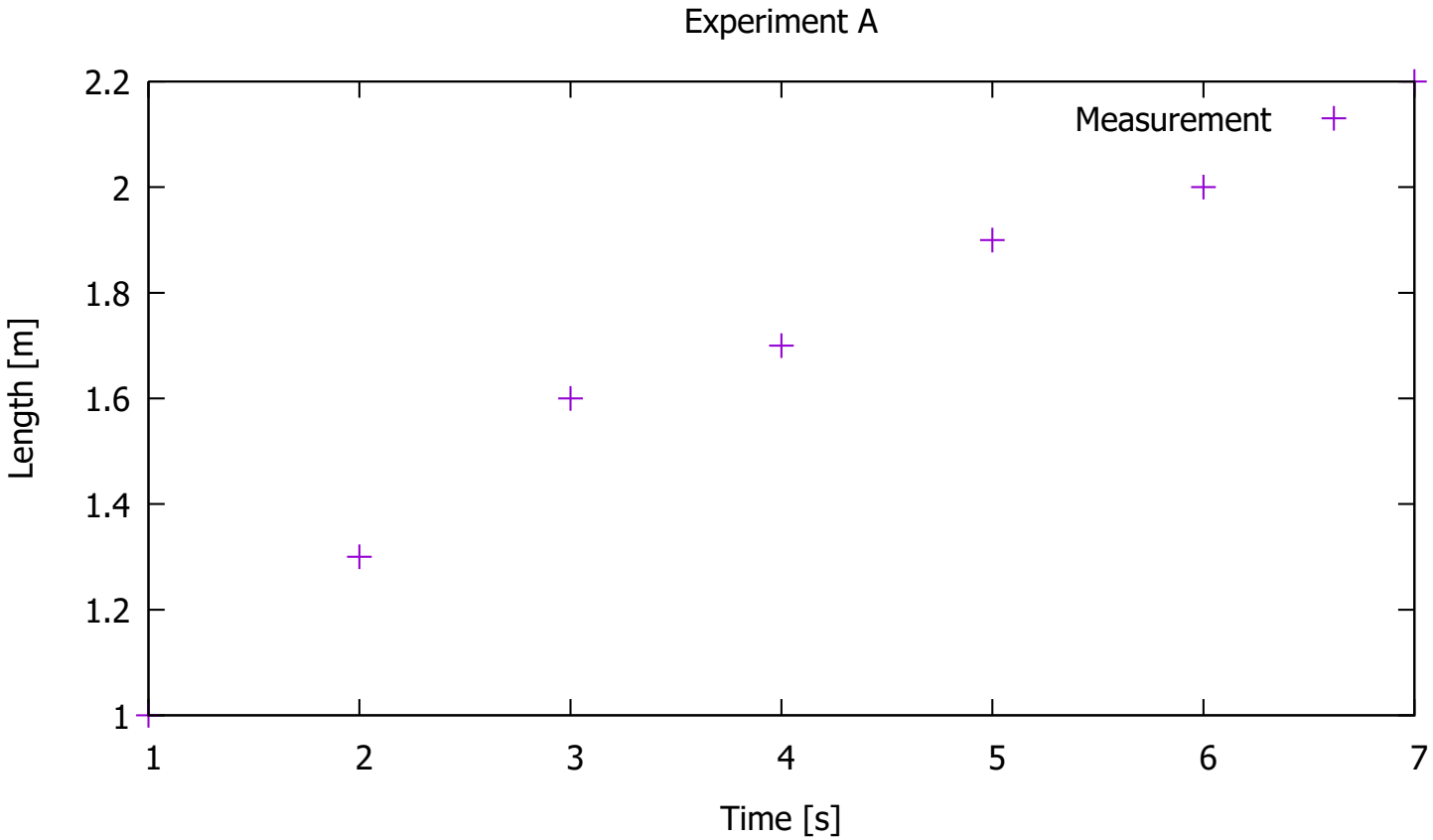

Figure 5: `plot.pdf` with label names and titles

This will make easier to understand our plot.

How do we do it?, We have to tell to gnuplot which kind of options we want to *Turn on*, for example, to make the title, we have to write

```
set title 'Experiment A'
```

If we are using a terminal like qt, wxt, etc... that doesn't need an external file to make plots, won't put the title, we have to plot again, or use replot

The labels are made as follows

```
set xlabel 'Time [s]'
```

```
set ylabel 'Length [m]'
```

Fit

We usually need to approximate our experimental data using theoretical functions, for example, we see that our data is like a linear function, but, we do not know the parameters, for example, let's use a linear function

$$f(x) = ax + b \tag{2}$$

to approximate our data, how do we find a and b?

We need to define our function on gnuplot, that's done by typing:

```
f(x)=a*x+b
```

> We need to use *, if we don't, ax will be defined as the variable.

And the fit is done like:

```
fit f(x) 'data.dat' using 1:2 via a,b
```

fit is the command to find the correct values of the regression, and it has some parameters, the fist one, is which function are we going to use f(x), then, which data file, 'data.dat' using 1:2[7], and the last one, is which variables of the functions it must find, in this case are a and b.

[7] If our file has more than two columns, the command using allow us to specify which columns we want to use

To plot the data and the regression in the same file, we use a comma (`,`)

```
plot 'data.dat' t 'Measurement', f(x)
```

getting as as result:

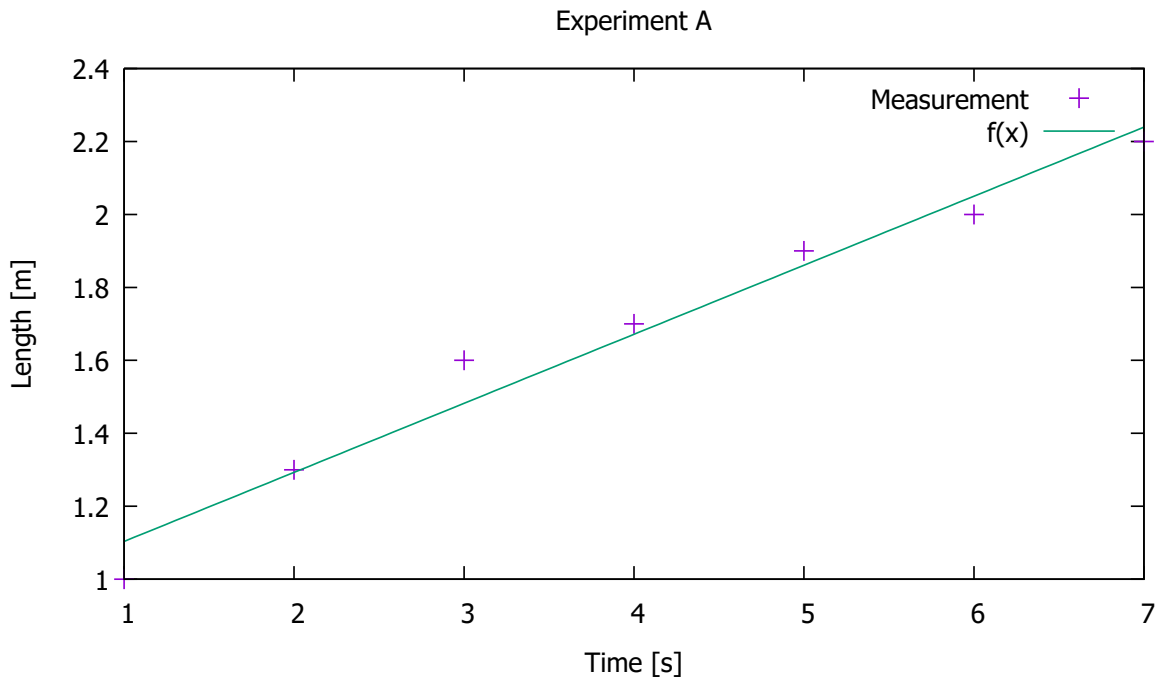

Figure 6: `plot.pdf` with regression

*

before talking more about fit, we can see that the legend is interfering with the data, but we can move it by typing: `set key left`

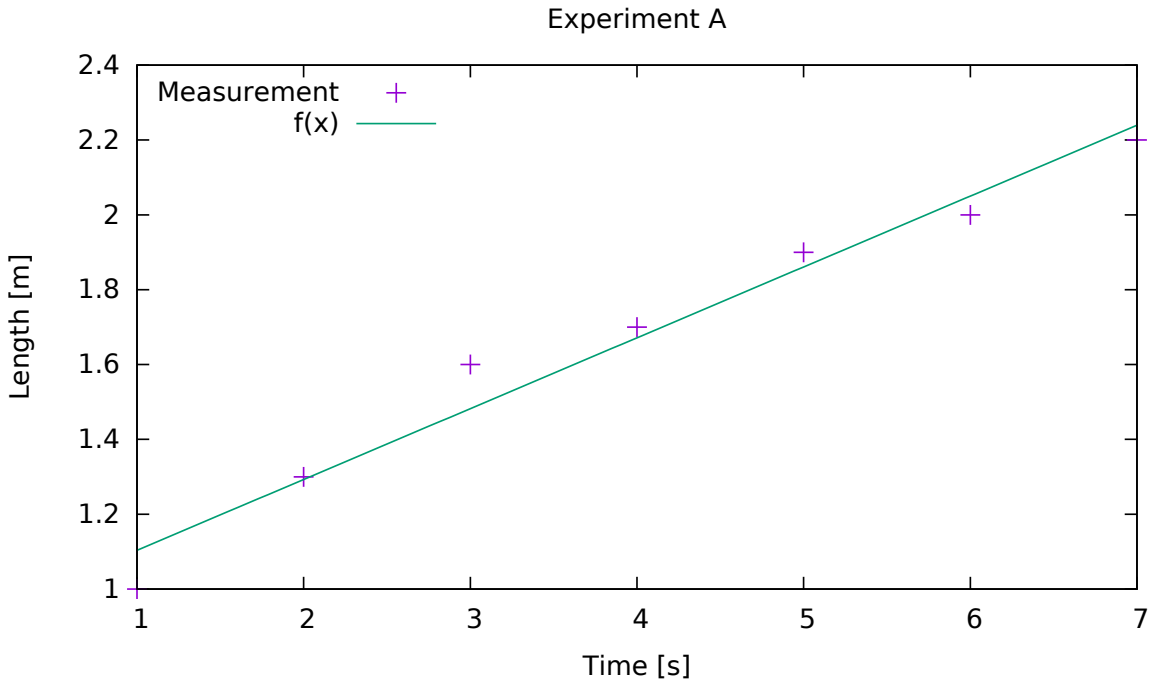

Figure 7: `plot.pdf` with regression and legend modified

Now, we're going to see one of the most important things about gnuplot.

We just ploted the fit of, in principle, experimental data but, it isn't enough for us just have a plot of it, we need the values of the parameters and their respective errors, so we'll know if our theoretical model is good enough for describing these data, or if our data is enough to compare with the theoretical model.

gnuplot helps us with that work, when we made the fit of the data in data.dat, in the terminal appear some calculations and the final results.

In the case we are studying, like we have only two parameters a y b, we have:

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a                  = 0.189286      +/- 0.0149       (7.87%)
b                  = 0.914286      +/- 0.06662      (7.287%)
```

Figure 8: `fit` result

where we get not only the values of the fit parameters but their errors too (And the percentage error).

But, what if we close our terminal, if we just have to shut down our computer, and didn't copy the values?, did we lost them?

When we use the command `fit`, a file named `fit.log` is created with all the information about the fits realized before, like date, hour, file, function, parameters, values, errors, and go on, try opening it.

Error-Bars

Let's consider another data, where we have more than two columns, for example:

1	10	2
2	30	5
3	50	10
4	80	12
5	110	15
6	145	17
7	185	18
8	225	20
9	270	22
10	315	25
11	365	30

12	415	35
13	468	40
14	523	42
15	580	45
16	640	48
17	700	50
18	760	55
19	825	60
20	894	63
21	960	70

Cuadro 4: `data2.dat`

When we plot this file like we did before, we get:

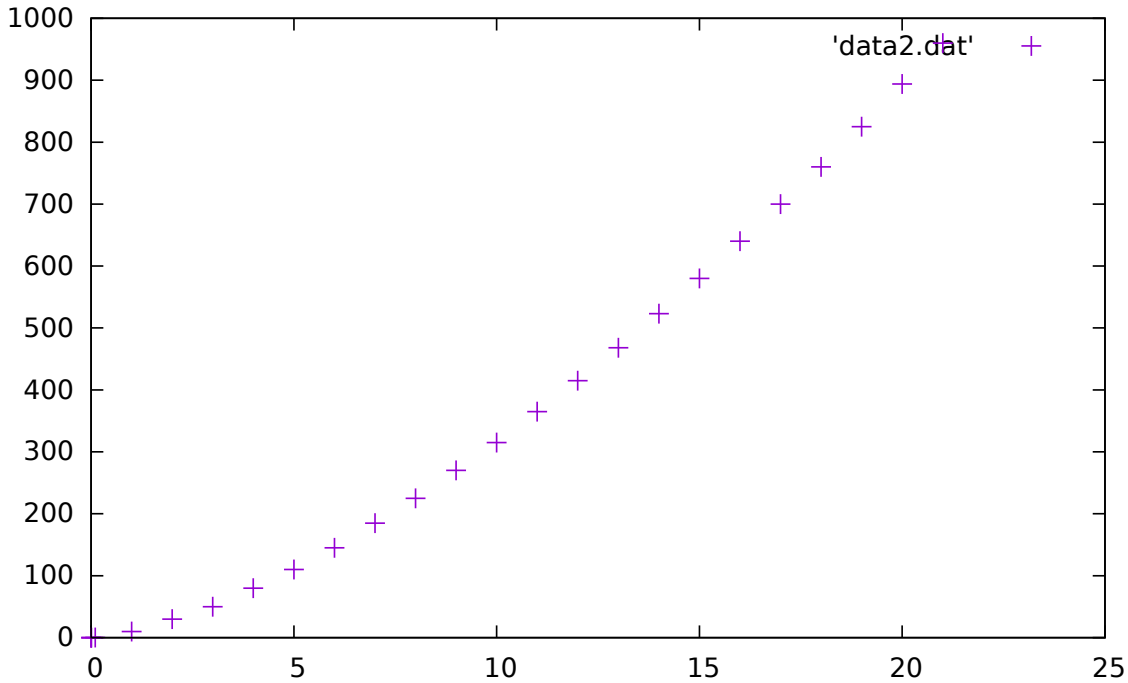

Figure 9: `plot2.pdf` result from `data2.dat`

But, why don't we see the points?, look at the scale, this data increase like a power law, *i.e.*, like a function with the form:

$$f(x) = ax^b \tag{3}$$

Having this kind of functions, motivate the change of scale, in this case, we are going to use a logscale, to do it, we just type

```
set logscale xy
```

and then, just `replot`, we get

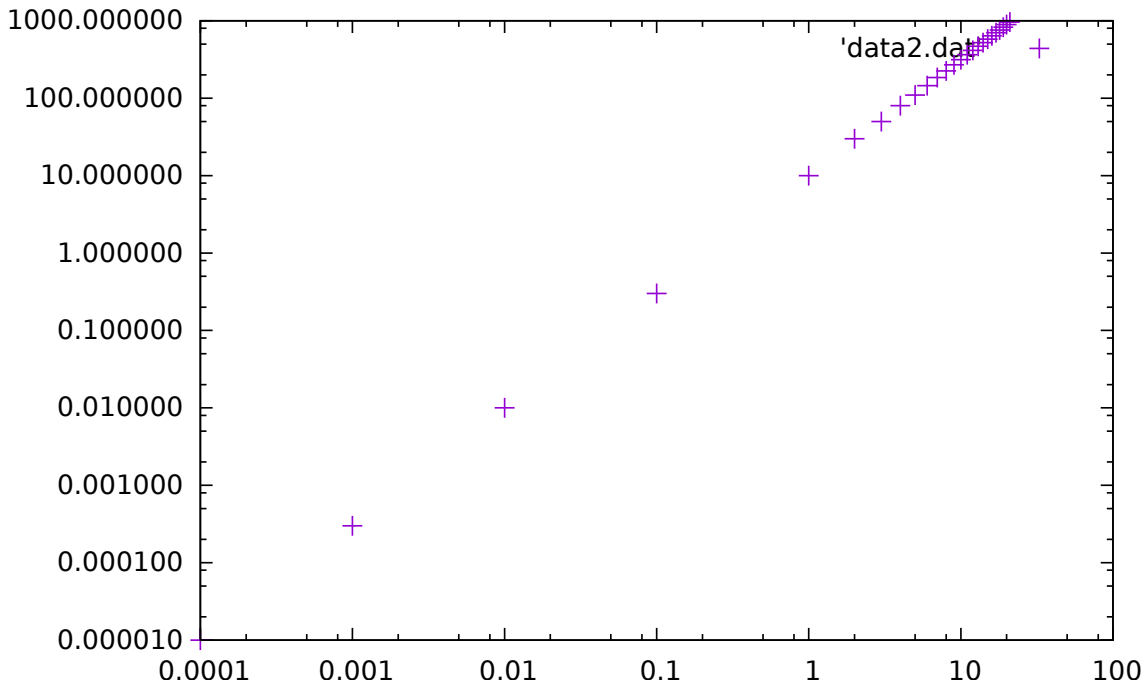

Figure 10: `plot2.pdf` using logscale.

now, we can think about the third column of the file `data2.dat` as the errors of the values in the y coordinate[8], and we want to put them in the plot, to do it, we write the plot line as follows:

```
plot ‘data2.dat’using 1:2:3 yerrorbars
```

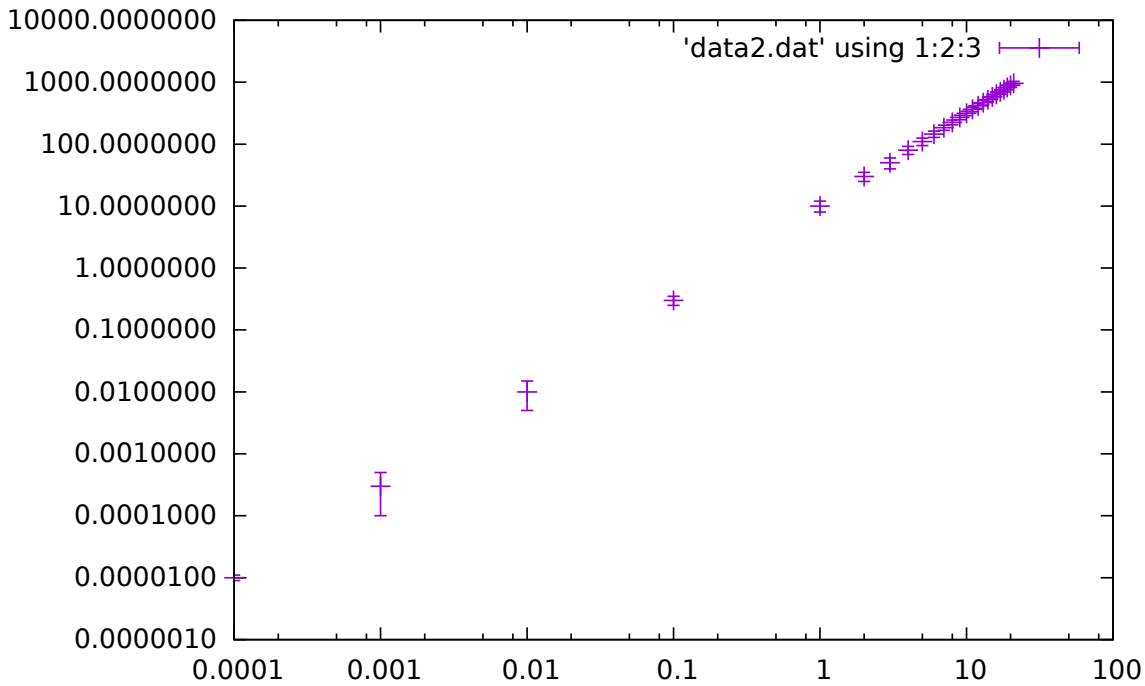

Figure 11: `plot2.pdf` using errorbars.

[8] The second column

If we take the error bars in x equal than in y, we write

```
plot 'data2.dat'using 1:2:3:3 xyerrorbars
```

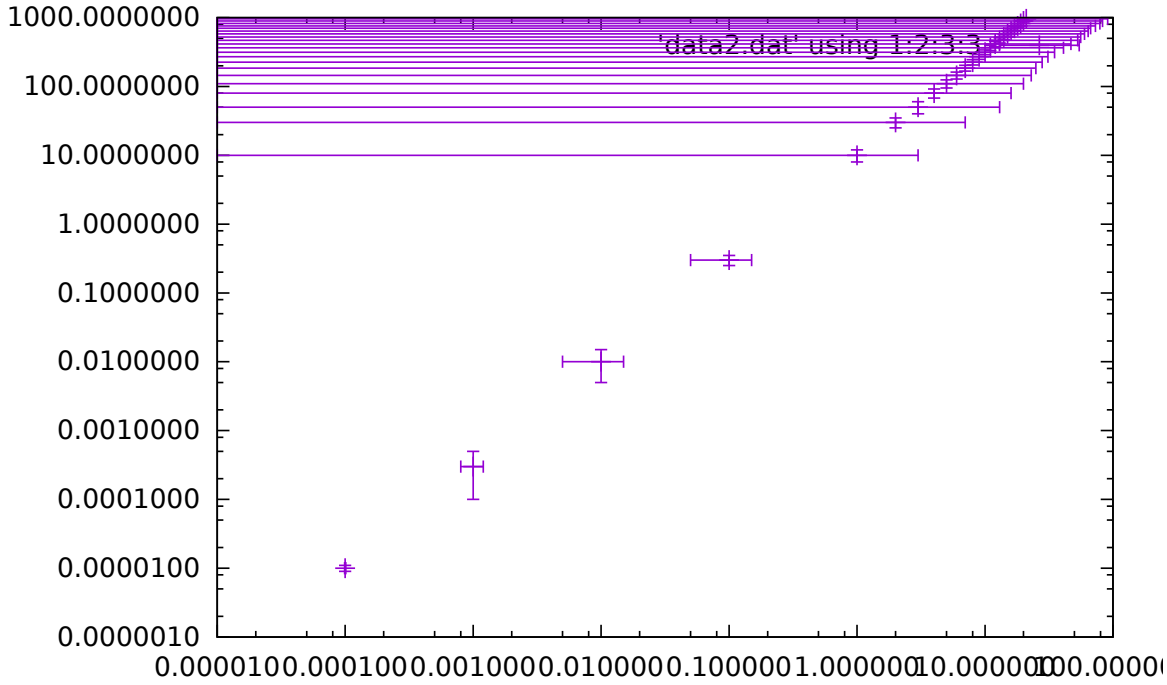

Figure 12: `plot2.pdf` using errorbars in x and y.

But, Why are they that much different? Even though they both are in logscale, they are in different order of magnitude, so, let's back to the previous case.

Let's do the fit with this kind of data

```
fit f(x) 'data2.dat'using 1:2 via a,b
```

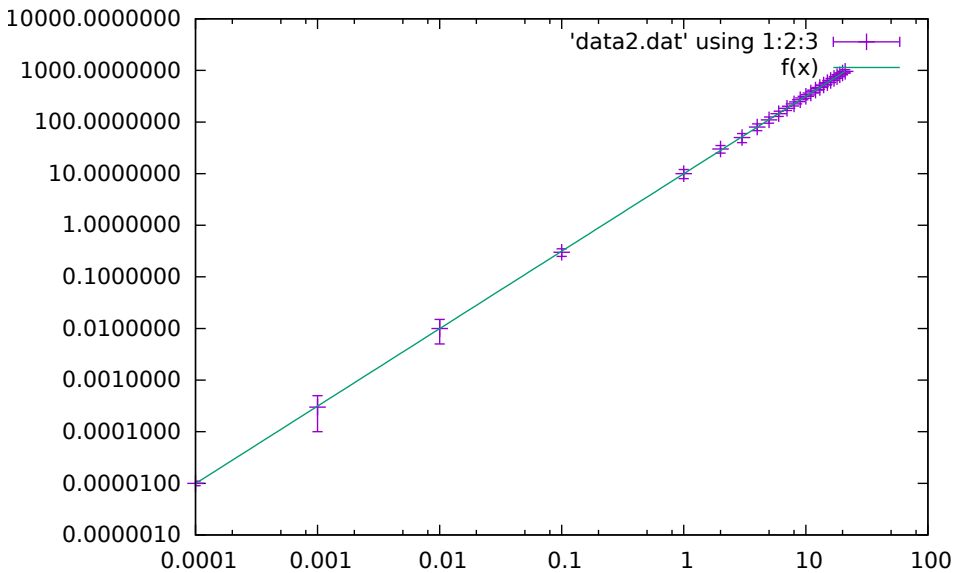

Figure 13: `plot2.pdf` and a power law as a fit.

but, we're not interested in have that scales in the y axis, it is possible to change, what if we choose powers of 10 to write these values?

That's can be done by typing:

```
set format y '% 2.0t x 10^{%L}'
```

getting as a result

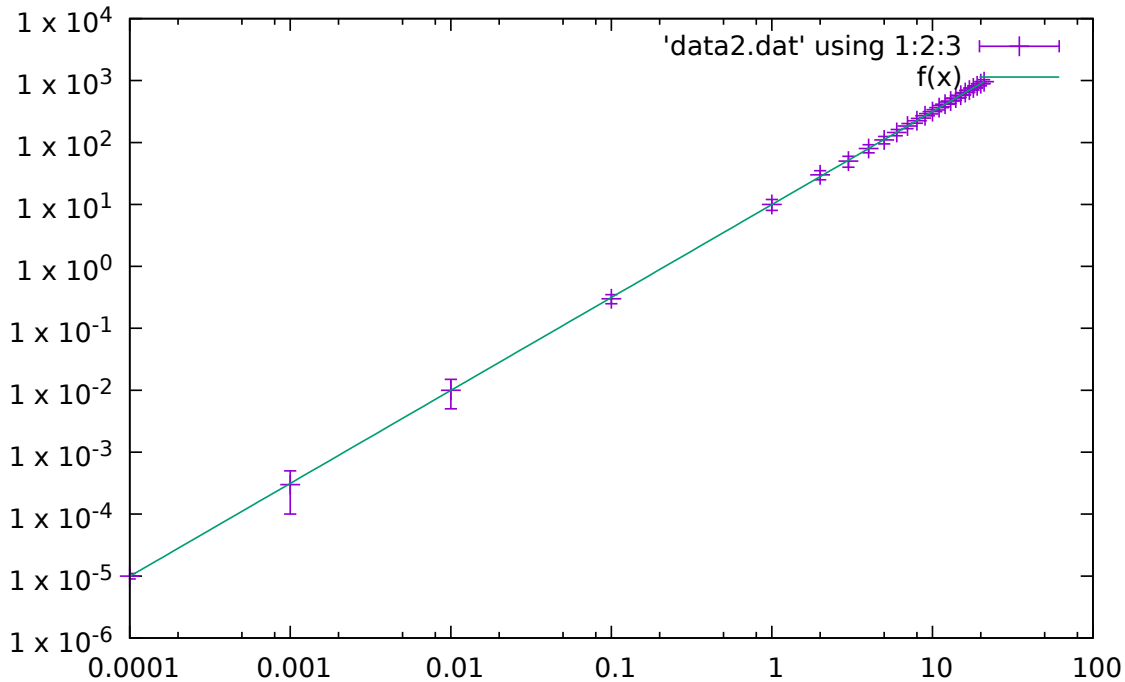

Figure 14: plot2.pdf.

finally, we can use all we've shown before, and get a plot like:

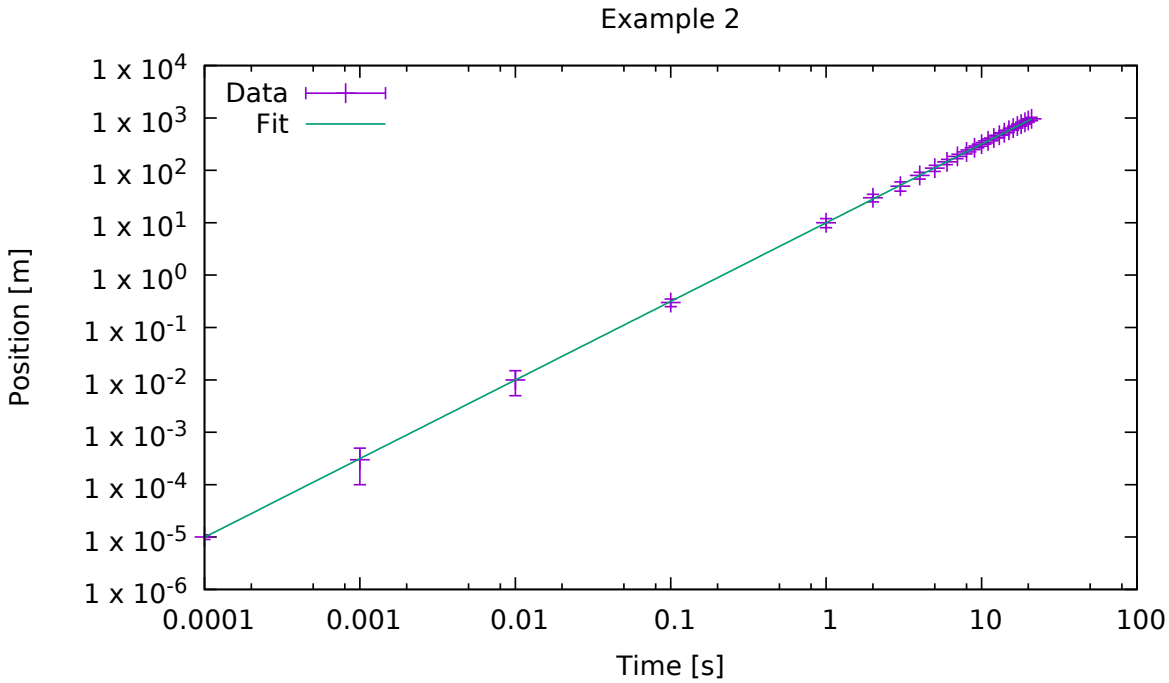

Figure 15: `plot2.pdf`.

and the fit parameters

$$a = (9{,}959 \pm 0{,}050) m/s, \quad b = (1{,}500 \pm 0{,}001) \tag{4}$$

References

GNUPLOT. *http://www.gnuplot.info/docs_5.0/gnuplot.pdf*, 2015