CGN User Manual

Felip Alàez Nadal

February 9, 2007

Contents

1 Introduction

Cgn is a library to control gnuplot from Lisp. Initially thought as simply a pipe to comunicate with gnuplot, It has grown to provide sintactic sugar for the most common operations on Gnuplot: plotting graphics, printing, saving/loading...

Cgn has become big enough to need a manual for itself. It comes here. I hope It to be useful. You can also see the examples at cgn's web at http://common-lisp.net/project/nixies

1.1 Asking and contributing to cgn

There's a mail list for cgn at common-lisp.net. You can also mail me at superratoli@gmail.com.

Contributions to cgn are welcome. Just send me them and I'll add them to cgn.

1.2 No warranty

2 Plotting functions

Cgn supports 2D and 3D plotting, with automatic reconigzation of parametrized functions. This means that you don't have to care about what code to use, just use (plot-function what-function) and cgn will generate all the code needed to do what you want. By example:

(plot-function "cos(x)")

(plot-function "cos(t),sin(t)")

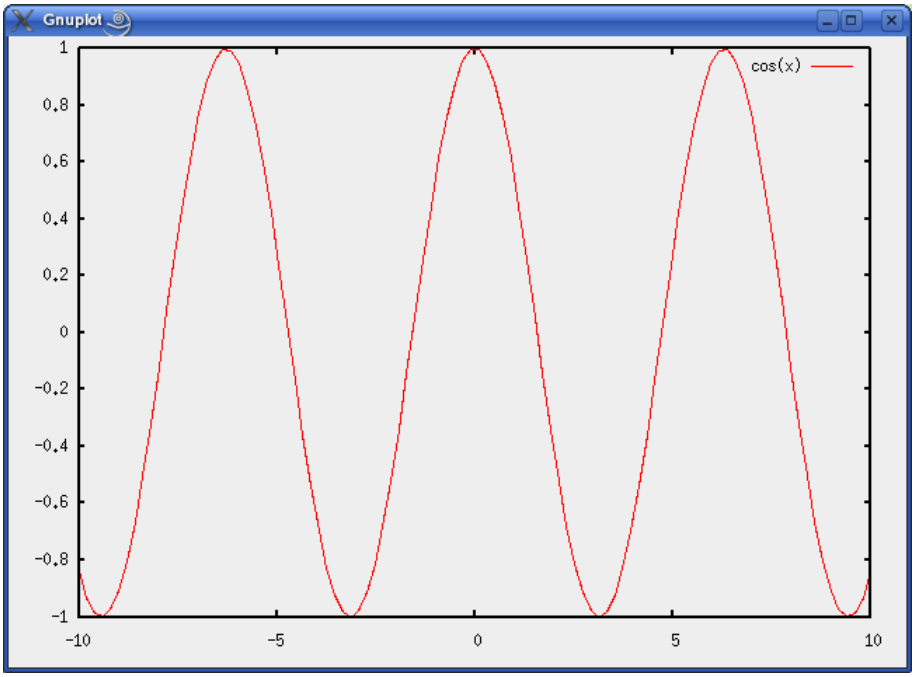

Figure 1: cos(x)

(plot-function "x*y*cos(x*y)")
(plot-function "sin(u), cos(u) , v ")

3 Plotting scatter graphics

Cgn can create scatter graphics to do that, first of all you must create a scatter-graphic description for your graphic, with:

create-scatter-graphic-description title

and then use It to create data series with

create-scatter-data-serie (sc scatter-graphic) x y x_error y_error title

where x, y, x_error and y_error are lists. Title is the title for the data serie.

At last, plot your description with plot-scatter scatter-graphic .

Example: create an scatter graphic with points from experiment1: (1,2) , (2,3), (3,4) without errorbars and points from experiment2: (2.5,1) , (3.1,2), (4.5, 7) ,(8,9) with a 10% of error for x and y.

```
(setq sc (create-scatter-graphic-description "Our experiments"))
```

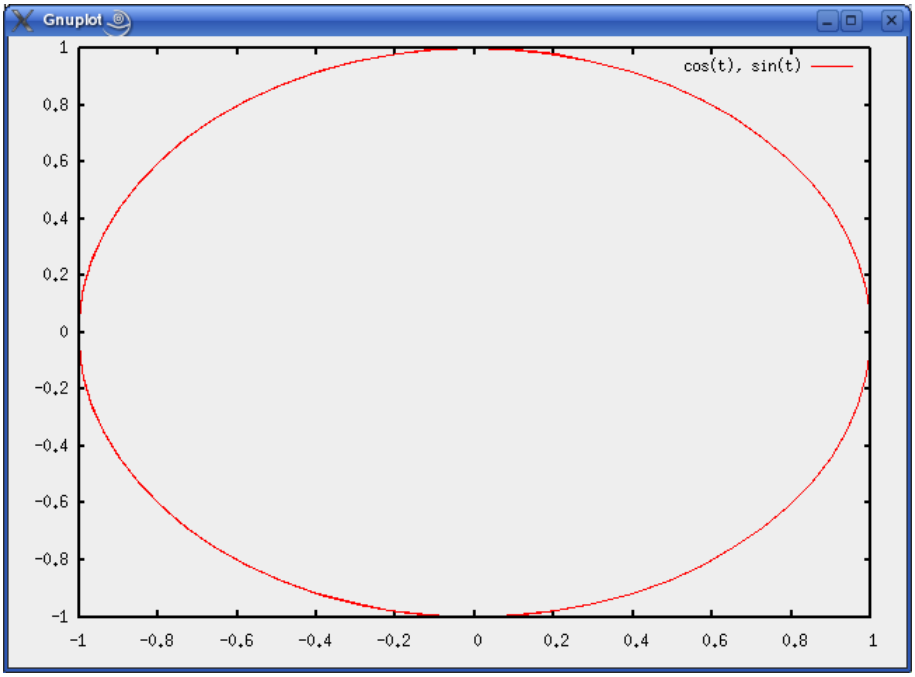

Figure 2: A circle

```
(create-scatter-data-serie sc '(1 2 3) '(2 3 4) '(0 0 0)
'(0 0 0 ) "Experiment1")

(create-scatter-data-serie sc '(2.5 3.1  4.5 8) '(1 2 7 9)
 '(0.25 0.31 0.45 0.8) '(0.1 0.2 0.7 0.9) "Experiment2")

(plot-scatter-graphic sc)
```

4 Configuring gnuplot

Cgn can do some personalization of gnuplot for you. At the moment:

```
set-grid    on-off
   set-range    eix min max
   set-title    text
```

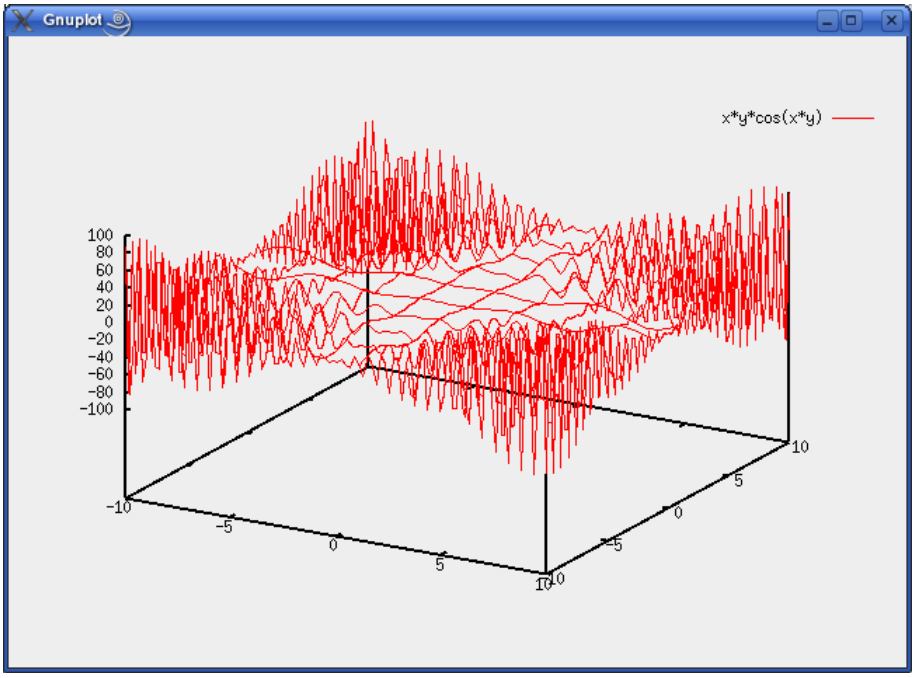

Figure 3: A f(x,y) function

5 Printing

Cgn can print directly from gnuplot. To do that, It needs to know what operative system is running[1] . Just use:

```
print-graphic   &optional (os *os*)
```

os must be set to 'w32[2] on windows. For non-windows machines, you can use whatever symbol except 'w32: linux, unix, my-beautiful-machine, etc...

6 Saving/loading

Cgn can save a gnuplot session to a file, and It can load It later. You have to understand that cgn does not save nothing. It's gnuplot who saves the

[1]This is only until release 007. Release 008 can determinate which os is running without the help of the user.

[2]There's an easy trick to conditionally set *os* to the correct symbol. Just (if (member :win32 *features*) (setq *os* 'w32) (setq *os* 'another)). This code was written for cgn 008, but It's not included in It because I don't know If works everywhere. I've tried It with sbcl and clisp and works. Please, if you've tried It with another lisp for windows, send me a mail to superratoli@gmail.com and this code will be included in cgn 009.

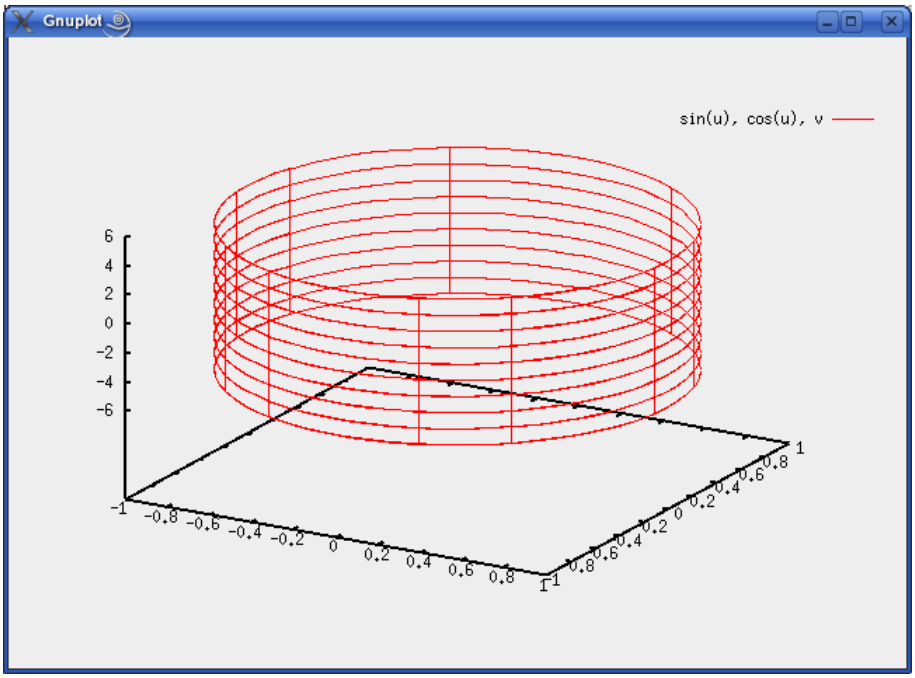

Figure 4: A beautiful cilinder

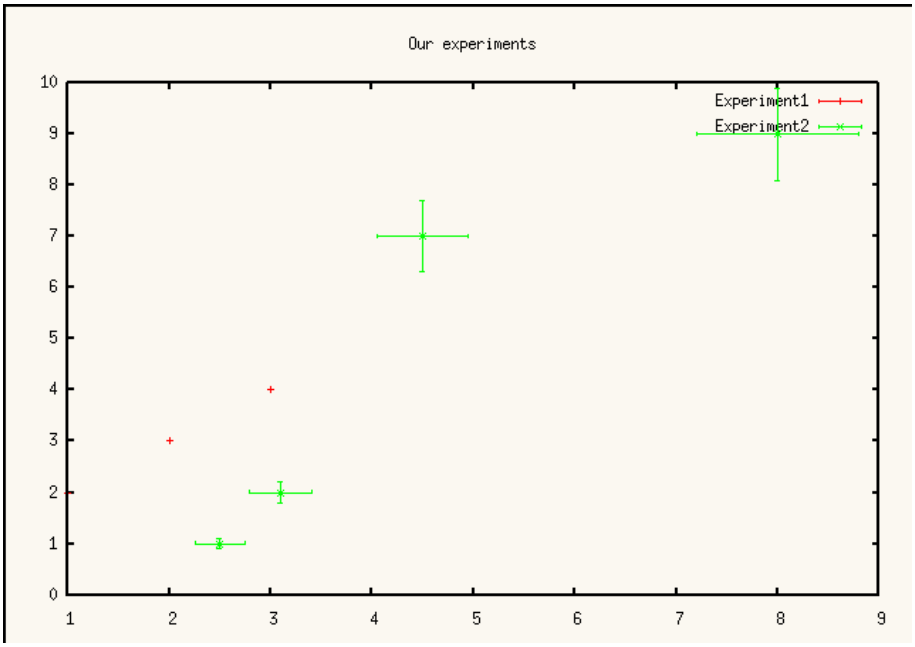

Figure 5: A scatter graphic

session. Then, when you load the saved session, you can restore gnuplot's status (plotted functions, title, etc...), but not cgn's status.

To save use:

```
save-cgn    file
```

And to load:

```
load-cgn    file
```

7 Creating animations

Since cgn 008, cgn can create animations. To do that, simply use *(animate-function what-function what-parameter initial-value increments number-of-frames rapidity)*.

8 Debugging cgn

Since cgn 008, debugging is supported. You shouldn't enable debug for a normal use. Enable It only if you are developing some kind of extension to cgn. To enable debug, just use:

```
(enable-debug)
```

And to disable It:

```
(disable-debug)
```

With debug enabled, cgn will print to *standard-output* everything that It sends to gnuplot.

9 Inside Cgn

Basically, cgn is constructed over two different parts:

- A pipe that can comunicate with gnuplot and a function to send messages to It and
- Lots of Lisp functions wrapping the most common operations you can do with gnuplot.

Cgn uses Ltk for the pipe issue. Ltk is a graphical toolkit for Lisp that uses a pipe to communicate with Tk[3]. It runs everywhere a Lisp implementation exists and is well supported and mantained. It provides a function that creates a pipe and returns a stream to the executed program. But with 2 goodies: It's mantained by other people than me (less work for me) and wraps code for lots of different Lisp implementations. Using It, cgn works everywhere.

9.1 Extending Cgn

It's easy to extend cgn. Everything you have to do is create a new function that sends to gnuplot the code that fills your needs, using format-gnuplot. format-gnuplot is just like format, but provides some features like debuggind. It also makes sure that gnuplot executes the command you send.

The first thing you have to do is creating a package for putting your code in It:

```
(defpackage :extending-cgn-example
(:use :cl :cl-user :cgn)
(:export #:set-property))

(in-package :extending-cgn-example)
```

To extend cgn It's not necessary create classes. You just have to write functions that send the code that fills your needs. At this example, we face a very common problem: there are lots of flags that control the behaviour of gnuplot. Cgn doesn't have functions to set all of them. Well, now we're gonna write a function to set all the flags that don't need paramethers.

First of all, we study what the generated code will seem:

set timestamp

set arrow

...

It becomes clear that this function will do the work:

```
(defun set-property ( property )
(format-gnuplot "set ~A" property ))
```

Now we can test this function and see what It happens:

[3]Tk is the Tcl ToolKit. It's an interpreter that, when called without arguments, waits for code on standard input. Ltk just creates the Tk code needed to create windows, buttons, etc...

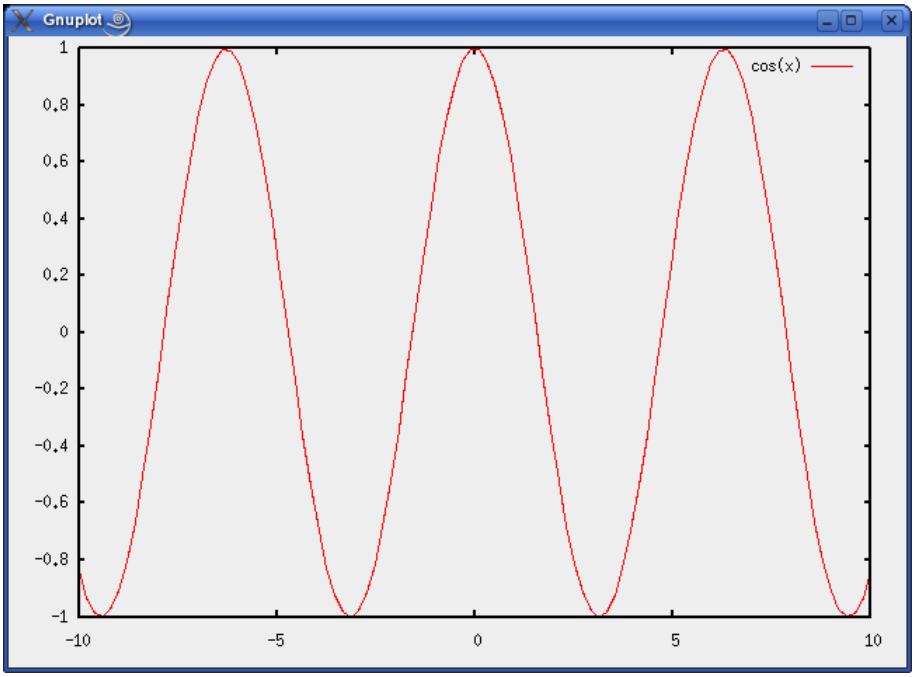

Figure 6: Before calling set-property

```
* (start-gnuplot)
* (enable-debug)
T
* (plot-function "cos(x)")
plot cos(x)
* (set-property "timestamp")
set timestamp
* (plot-function "sin(x)")
plot sin(x)
```

10 Seeing release info

Just type

```
(show-release-info)
```

And you'll see something like this:
    cgn 008 released 9-II-07

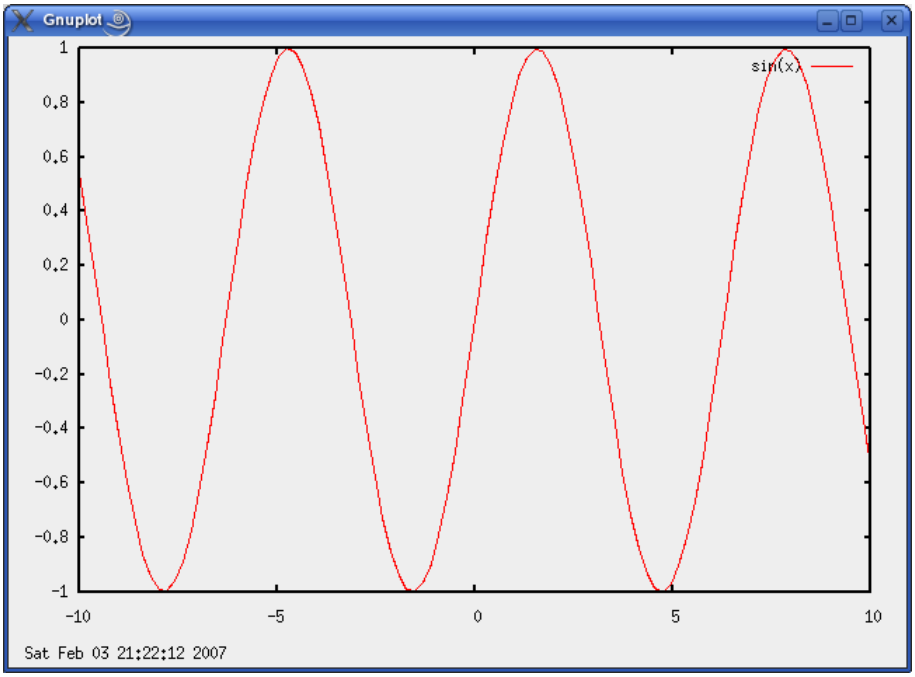

Figure 7: After calling set-property. Watch the timestamp.

11 Changelog

9-II-07: cgn 008

Added plot-scatter

Added create-scatter-data-serie

Added create-scatter-graphic-description

Added new structures : data-serie and scatter-graphic, to save information about scatter graphics.

Added show-release-info.

Added two new flags: *release* and *release-date*.

format-gnuplot now does debug if debugging is enabled.

Added enable-debug and disable-debug.

Added *debug*, a flag to control if cgn must do debug.

All the library has been updated to use the new format-gnuplot facilities.

Format-gnuplot now understands all the options that format can understand.

Replot-function now can replot every type of functions.

Created a new manual for Cgn.

Animate-function now can create animations for whatever type of function, only for one paramether.

Added animate-function, to create animations. At the moment, only for f(x), and one paramether.

plot-function now plots using the parametric form.

20-VI-06: cgn 007

Added print-graphic, to print directly from cgn.

Modified plot-function. Now can be used to plot g(x,y), not only f(x).

Added save-cgn, to save.

Added load-cgn, to load.

Added with-gnuplot, a macro which should be the prefered way to use cgn.

Added the *os* special variable, to use with with-gnuplot to let cgn know which os is running. This is used to print, at the moment.

Updated cldoc documentation.

8-VI-06: cgn 006

Corrected bug. Gnuplot keeped running later of using close-gnuplot.

Comments now translated to English. It seems that the original Catalonian documentation produced slow compilations at non Catalonian machines.

Added cldoc documentation.

12 Bibliography