Engineering Log Log Graphs Using my Gonzo Utilities

Don Lancaster
Synergetics, Box 809, Thatcher, AZ 85552

http://www.tinaja.com
don@tinaja.com
(928) 428-4073

I have long had available a free set of my **Gonzo Utilities** that can be found in our **PostScript Library**. Which I continue to aggressively personally use for **all** of my writing projects, my **PostScript as Language** programming research, and for countless other tasks.

These highly device independent and **proudly non-WYSIWYG PostScript** utilities can greatly simplify **Using PostScript as a General Purpose Computing Language**, can offer **superbly fancy typesetting**, do **fancy modeling calculations**, very much **enhance gridding and graphing**, and even provide you with **unmatched quality electronic schematics**.

They also let you **read or write most any disk file in most any language**, thus leaping tall buildings in a single bound. Expanded, they may also be used to programmatically **write complex programs in other languages**. Such as those **JavaScript** Fourier Analysis routines found on our **Magic Sinewave** page. Or the incredibly legible small fonts in our **Bitmap Typewriter**.

You can start learning the **Gonzo Utilities** with our **PostScript Beginner Projects**. Or else go to most any **.PSL** file in our **GuruGram library** or elsewhere for highly detailed use examples.

To use the **Gonzo Utilities** yourself, you **download them** into a known place in your computer or place them on a floppy or other media. Then, **after making obvious site changes**, you include **one of the following** in the beginning of your standard ASCII textfile that will hold the **PostScript** instructions that you are going to send to **Acrobat Distiller**...

```
(C:\\Documents and Settings\\don\\Desktop\\
  gonzo\\gonzo.ps) run    % use internal gonzo

(A:\\gonzo.ps) run        % use external gonzo
```

Be sure to note that…

> **You MUST use a DOUBLE reverse slash inside a PostScript string every time you really want a single reverse slash!**

The **Gonzo Utilities** are normally in a group of activatable dictionaries. At present, their **activation sequence** of **gonzo begin ps.util.1 begin nuisance begin** is built into the last lines of the **Gonzo** download. Thus, Gonzo will automatically activate most of itself on its **run** command. The separate **electronics begin** dictionary is only sometimes needed, so you will have to activate it when desired.

In normal use, you will create an **ordinary ASCII textfile** using your favorite editor or word processor that has the **(…Gonzo…) run** command near its beginning. Typically, you will have a **showpage** at the end of your document. Note that you must **NOT** end with a **quit** or a **[D]** Control-D command!

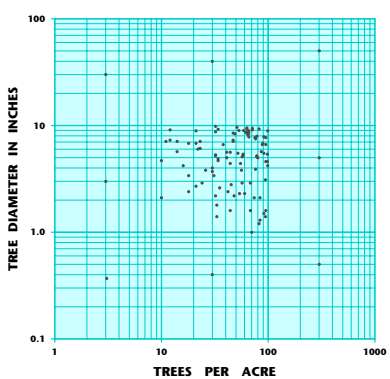
You'll then send this file to **Acrobat Distiller**, which is included in all full featured versions of **Adobe Acrobat**. While Acrobat 7 is recommended at this writing, earlier versions can also be used. It is also sometimes possible to use **GhostScript** as an **Acrobat** replacement. My personal recommendation is to always use a genuine **Adobe** distiller instead.

We recently saw a **book cover layout** example of **Gonzo Utilities** use back in our **GuruGram #61**. Many dozens of similar examples have been newly added to our new **Best Files Sampler** area on our **Guru's Lair** website home page.

Gonzo Log Log Engineering Graphs

Because of its calculation convenience, the **Gonzo Utilities** are particularly adept at presenting unusual engineering charts and graphs. Especially when some odd computation is involved in getting at the underlying graphical format.

Such as this log log scatterplot…



I have put together a new set of utility routines as **LOGLOG01.PSL**. While specific to log log plots, these should also let you conveniently do most any complex engineering chart or graph. A companion **LOGLOG01.PDF** demo gives you a full size version of our above figure. Or you can **click expand** on the figure itself.

LOGLOG01.PSL is fairly well documented internally, so what I thought I'd do here is just go over a few of the more subtle points of log log engineering graphing. What follows is best understood if you do have copies of both **LOGLOG01.PSL** and **LOGLOG01.PDF** printed out or in open windows.

The basic elements of **LOGLOG01.PSL** are a method of **creating a working grid**, a way to **draw a log log graph**, and a means of **dealing with our input data**.

Here is the **Gonzo** command to create a grid…

```
100 100 10 setgrid
```

The first two numbers are the **xoffset** and **yoffset**, while the third gives you the **magnification factor**. This creates an invisible grid that extends **infinitely** in both directions. Working at **10X** normal size (ten point squares) tends to give you numbers that are very tractable and fun to work with. Thus greatly easing your layout tasks.

Very often, you might like to make a portion of your grid visible. Either for some temporary layout or as an actual design element. While this is simply done with a **30 30 showgrid** or whatever, we'll instead need **nonlinear** graphing for our upcoming log log plot.

One caution on gridding…

Fonts MUST be activated AFTER the grid is in place!

As must the /cstretch and /sstretch font variables.

Once on our grid, we can assign 10x10 grid squares to each log decade. Here's how the background and major divisions go down for a log log plot of three decade cycles square…

```
backgroundcolor setrgbcolor 0 0 mt 30 pu 30 pr 30 pd
closepath fill

graphlinecolor setrgbcolor  line2 1 setlinecap 1 setlinejoin
[ {-0.3 0 mt 30.3 r}10 4 ] yrpt
[ {0 -0.3 mt 30.3 u}10 4 ] xrpt
```

The lines extend slightly lower and to the left to provide the **tick marks** for the upcoming decade numeric values. Format for the repeat commands is an array of **[{action} spacing #repeats]**.

Getting That Log Spacing

It is important to remember that **our lower left log location will always be 1,1**. Data scaling or offsetting may be required to force this need.

Drawing the actual log lines is rather trivial with PostScript's **log** command…

```
/loglines [ 1 2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80 90
    100 200 300 400 500 600 700 800 900 1000 ] store

line1                              % thin line

0 1 loglines length 1 sub          % draw verticals
  {/ptr exch store
   loglines ptr get
   log 10 mul 0 mt 30 u
  } for

0 1 loglines length 1 sub          % draw horizontals
  {/ptr exch store
   loglines ptr get
   0 exch log 10 mul mt 30 r
   } for
```

Each desired log value is multiplied by ten because there are ten grid units in each log decade. The size and values in your **/loglines** array will change should you want to increase the number of log decades being displayed.

Our **Gonzo Utilities** include some extremely powerful font manipulation tools. Here is how the vertical and horizontal grid callouts are put down…

```
/font1 /StoneSans-Bold 0.85 gonzofont font1 black

/xnums  [(10) (100) (1000) (10K)] store
/ynums  [(0.1) (1.0) (10) (100)] store

0 1 xnums length 1 sub {/ptr2 exch store ptr2 10 mul
-1.5 xnums ptr2 get cc } for

0 1 ynums length 1 sub {/ptr2 exch store -0.9 ptr2
10 mul 0.3 sub ynums ptr2 get cr
```

Note the respective use of **cr** and **cc** right and center justify.

Here is how the axis titles are handled…

```
/font2 /StoneSans-Bold 1.2 gonzofont font2 black

/xaxisname (TREES  PER  ACRE) store
/yaxisname (TREE  DIAMETER  IN  INCHES) store

15 -3.5 xaxisname cc

gsave -3.5 15 translate 90 rotate
0 0 yaxisname cc grestore
```

The y axis code appears a little more obtuse because of the 90 degree rotation we need for the vertical text. This should be buried down deep enough to not trip the **Acrobat** rotate feature.

The **Stone** font is particularly attractive for engineering work. Its availability is recommended for our demos here. If not available, **Acrobat** will substitute a not-quite-as-good multimaster alternative.

Needless to say, careful choice of font style and size can make a tremendous difference in the viewability of your final presentation.

Managing Your Input Data

There are several ways to handle your input data for the scatterplot dots. This can depend on how many graphs you are making and how fancy you want to get.

The simplest is to manually enter your data as a PostScript array…

```
/inputdata [ x0 y0 x1 y1 x2 y2 ... xn yn ] store
```

You can also **cut and paste** data from any textfile source. Or **run** any separate but **locally resident** textfile, with **/inputdata [** and **]** either pre- or post-appended.

Or, to get really fancy, you can use the **Gonzo Utilities** to read virtually any file in any format. Reaching down into the spreadsheet or data base of your choice.

Custom Consulting services are available for such advanced tasks.

Note that your input data is its **actual value**, and should **not** be prescaled or log converted. Such scaling and conversion should take place **only** during the actual plotting process. Either integers or reals are permitted.

The **Gonzo Utilities** have a **mt dot** command that automatically draws a proper size dot at your chosen location.

Dot plotting may need scaling to meet our 1,1 lower left log convention…

```
/xdatascale  0.1 store
/ydatascale  10 store

datadotcolor setrgbcolor

0 2 inputdata length 2 sub
  {/ptr1 exch store              % save pair location

  inputdata ptr1 get             % get x value
  xdatascale mul
  log 10 mul

  inputdata ptr1 1 add get       % get y value
  ydatascale mul
  log 10 mul

  mt dot                         % and plot dot
      } for
```

The data actually shown here may appear a tad bizarre, as it was faked. First with nine decade check points, and then with a random collection of 100 points in the middle decade.

Extensions to what we have shown here can be used for **semilog** plots in which only **one** axis is log. Or plotting continuous curves rather than scatterplot data. Or fitting noisy data. Or just about any other engineering plot task.

Click to Magnify

One final detail: You might like to present your graph in compact form in your initial report and let your viewers expand upon it for full detail. Just as we have done above.

You can do this manually with the Acrobat linking tool. Or else insert this code at the point where your **already positioned and scaled** graph is to go in your larger document…

```
cururlname (http://www.tinaja.com/glib/loglog01.pdf)
        store

mark /Rect  [ 0 0 30 30 ] /Border [ 0 0 0] /Color [ .7 0 0 ]
/Action <</Subtype /URI /URI cururlname>>
/Subtype /Link /ANN pdfmark
```

Full details are in the **PDFMark Reference Manual**.

For More Help

Bunches of more info on Gonzo and PostScript in general can be found over on our **PostScript** library page. Along with lots of tutorials and utilities.

Additional consulting services are available per our **Infopack** services on a contract, seminar, or hourly basis. Further **GuruGrams** await your ongoing support as a **Synergetics Partner**.