Applications for graphical visualization of data on Linux® are varied, from simple 2-D plots to 3-D surfaces, scientific graphics programming, and graphical simulation. Luckily, there are many open source possibilities, including gnuplot, GNU Octave, Scilab, MayaVi, Maxima, OpenDX, and others. Each has its advantages and disadvantages and targets different applications. Explore a variety of open source graphical visualization tools to better decide which is best for your application. [This article has been updated to include coverage of OpenDX - Ed.]

A short list of visualization tools

In this article, I provide a survey of a number of popular Linux data visualization tools and include some insight into their other capabilities. For example, does the tool provide a language for numerical computation? Is the tool interactive or does it operate solely in batch mode? Can you use the tool for image or digital signal processing? Does the tool provide language bindings to support integration into user applications (such as Python, Tcl, Java programming languages, and so on)? I also demonstrate the tools' graphical capabilities. Finally, I identify the strengths of each tool to help you decide which is best for your computational task or data visualization.

The open source tools that I explore in this article are (with their associated licenses):

- Gnuplot (Gnuplot Copyright, non GPL)
- GNU Octave (GPL)
- Scilab (Scilab)
- MayaVi (BSD)
- Maxima (GPL)
- OpenDX (IBM Public License)

Gnuplot

Gnuplot is a great visualization tool that has been around since 1986. It’s hard to read a thesis or dissertation without running into a gnuplot graph. Although gnuplot is command-line driven, it has grown from its humble beginnings to support a number of non-interactive applications, including its use as a plotting engine for GNU Octave.
Gnuplot is portable, operating on UNIX®, Microsoft® Windows®, Mac OS® X and many other platforms. It supports a range of output formats from postscript to the more recent PNG.

Gnuplot can operate in batch mode, providing a script of commands to generate a plot, and also in interactive mode, which allows you to try out its features to see the effect they have on your plot.

A standard math library is also available with gnuplot that corresponds to the UNIX math library. Arguments for functions support integer, real, and complex. You can configure the math library for radians or degrees (the default is radians).

For plotting, gnuplot can generate 2-D plots with the `plot` command and 3-D plots (as 2-D projections) with the `splot` command. With the `plot` command, gnuplot can operate in rectangular or polar coordinates. The `splot` command is Cartesian by default but can also support spherical and cylindrical coordinates. You can also apply contours to plots (as shown in Figure 1, below). A new style for plots, `pm3d`, supports drawing palette-mapped 3-D and 4D data as maps and surfaces.

Here's a short gnuplot example that illustrates 3-D function plotting with contours and hidden line removal. Listing 1 shows the gnuplot commands that are used, and Figure 1 shows the graphical result.

**Listing 1. Simple gnuplot function plot**

```plaintext
set samples 25
set isosamples 26
set title "Test 3D gnuplot"
set contour base
set hidden3d offset 1
splot [-12:12.01] [-12:12.01] sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
```

Listing 1 illustrates the simplicity of gnuplot’s command set. The sampling rate and density of the plot are determined by `samples` and `isosamples`, and a title is provided for the graph with the `title` parameter. The base contour is enabled along with hidden line removal, and the sinc plot is created with the `splot` command using the internally available math library functions. The result is Figure 1.
In addition to creating function plots, gnuplot is also great for plotting data contained in files. Consider the x/y data pairs shown in Listing 2 (an abbreviated version of the file). The data pairs shown in the file represent the x and y coordinates in a two-dimensional space.

**Listing 2. Sample data file for gnuplot (data.dat)**

```plaintext
56 48
59 29
85 20
93 16
...
56 48
```

If you want to plot this data in a two-dimensional space, as well as connect each data point with a line, you can use the gnuplot script shown in Listing 3.

**Listing 3. Gnuplot script to plot the data from Listing 2**

```plaintext
set title "Sample data plot"
plot 'data.dat' using 1:2 t 'data points', "data.dat" using 1:2 t "lines" with lines
```

The result is shown in Figure 2. Note that gnuplot automatically scales the axes, but you're given control over this if you need to position the plot.
Figure 2. A simple plot from gnuplot using a data file

![Simple data plot](image)

Gnuplot is a great visualization tool that is well known and available as a standard part of many GNU/Linux distributions. However, if you want basic data visualization and numerical computation, then GNU Octave might be what you're looking for.

**GNU Octave**

GNU Octave is a high-level language, designed primarily for numerical computation, and is a compelling alternative to the commercial Matlab application from The MathWorks. Rather than the simple command set offered by gnuplot, Octave offers a rich language for mathematical programming. You can even write your applications in C or C++ and then interface to Octave.

Octave was originally written around 1992 as companion software for a textbook in chemical reactor design. The authors wanted to help students with reactor design problems, not debugging Fortran programs. The result was a useful language and interactive environment for solving numerical problems.

Octave can operate in a scripted mode, interactively, or through C and C++ language bindings. Octave itself has a rich language that looks similar to C and has a very large math library, including specialized functions for signal and image processing, audio processing, and control theory.

Because Octave uses gnuplot as its backend, anything you can plot with gnuplot you can plot with Octave. Octave does have a richer language for computation, which has its obvious advantages, but you'll still be limited by gnuplot.

In the following example, from the Octave-Forge Web site (SimpleExamples), I plot the Lorentz Strange Attractor. Listing 4 shows the interactive dialog for Octave on the Windows platform with Cygwin. This example demonstrates the use of lsode, an ordinary differential equation solver.
Listing 4. Visualizing the Lorentz Strange Attractor with Octave

GNU Octave, version 2.1.50
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type `warranty'.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug-octave@bevo.che.wisc.edu>.

>> function y = lorenz( x, t )
   y = [10 * (x(2) - x(1));
        x(1) * (28 - x(3));
        x(1) * x(2) - 8/3 * x(3)];
endfunction
>> x = lsode("lorenz", [3;15;1], (0:0.01:25)');
>> gset parametric
>> gsplot x

The plot shown in Figure 3 is the output from the Octave code shown in Listing 4.

Figure 3. A Lorentz plot with Octave

GNU Octave (in concert with gnuplot) can emit multiple plots on a single page with the `multiplot' feature. Using this feature, you define how many plots to create and then define the particular plot using the `subwindow' command. After the subwindow is defined, you generate your plot normally and then step to the next subwindow (as shown in Listing 5).

Listing 5. Generating multiplots in Octave

>> multiplot(2,2)
>> subwindow(1,1)
>> t=0:0.1:6.0
>> plot(t, cos(t))
>> subwindow(1,2)
>> plot(t, sin(t))
>> subwindow(2,1)
>> plot(t, tan(t))
>> subwindow(2,2)
>> plot(t, tanh(t))
The resulting multiplot page is shown in Figure 4. This is a great feature for collecting related plots together to compare or contrast.

**Figure 4. A multiplot with GNU Octave**

You can think about Octave as a high-level language with gnuplot as the backend for visualization. It provides a rich math library and is a great free replacement for Matlab. It's also extensible, with packages developed by users for speech processing, optimization, symbolic computation, and others. Octave is in some GNU/Linux distributions, such as Debian, and can also be used on Windows with Cygwin and Mac OS X. See the Related topics section for more information on Octave.

**Scilab**

Scilab is similar to GNU Octave in that it enables numerical computation and visualization. Scilab is an interpreter and a high-level language for engineering and scientific applications that is in use around the world.

Scilab originated in 1994 and was developed by researchers at Institut national de recherche en informatique et en automatique (INRIA) and École Nationale des Ponts et Chaussées (ENPC) in France. Since 2003, Scilab has been maintained by the Scilab Consortium.

Scilab includes a large library of math functions and is extensible for programs written in high-level languages, such as C and Fortran. It also includes the ability to overload data types and operations. It includes an integrated high-level language, but has some differences from C.

A number of toolboxes are available for Scilab that provide 2-D and 3-D graphics and animation, optimization, statistics, graphs and networks, signal processing, a hybrid dynamic systems modeler and simulator, and many other community contributions.

You can use Scilab on most UNIX systems, as well as the more recent Windows operating systems. Like GNU Octave, Scilab is well documented. Because it is a European project, you can also find documentation and articles in a number of languages other than English.
When Scilab is started, a window displays allowing you to interact with the interpreter (see Figure 5).

**Figure 5. Interacting with Scilab**

In this example, I create a vector (t) with values ranging from 0 to 2PI (with a step size of 0.2). I then generate a 3-D plot (using z=f(x,y), or the surface at the point xi,yi). Figure 6 shows the resulting plot.

**Figure 6. The resulting Scilab plot from the commands in Figure 5**

Scilab includes a large number of libraries and functions that can generate plots with a minimum of complexity. Take the example of generating a simple three-dimensional histogram plot:

```plaintext
-->hist3d(5*(rand(5,5));
```

First, the `rand(5,5)` builds a matrix of size 5,5 containing random values (which I scale to a maximum of 5). This matrix is passed to the function `hist3d`. The result is the histogram plot shown in Figure 7.
Figure 7. Generating a random three-dimensional histogram plot

Scilab and Octave are similar. Both have a large base of community participation. Scilab is written in Fortran 77, whereas Octave is written in C++. Octave uses gnuplot for its visualization; Scilab provides its own. If you're familiar with Matlab, then Octave is a good choice because it strives for compatibility. Scilab includes many math functions and is very good for signal processing. If you're still not sure which one to use, try them both. They're both great tools, and you may find yourself using each of them for different tasks.

MayaVi

MayaVi, which means magician in Sanskrit, is a data visualization tool that binds Python with the powerful Visualization Toolkit (VTK) for graphical display. MayaVi also provides a graphical user interface (GUI) developed with the Tkinter module. Tkinter is a Tk interface, most commonly coupled with Tcl.

MayaVi was originally developed as a visualization tool for Computational Fluid Dynamics (CFD). After its usefulness in other domains was realized, it was redesigned as a general scientific data visualizer.

The power behind MayaVi is the VTK. The VTK is an open source system for data visualization and image processing that is widely used in the scientific community. VTK packs an amazing set of capabilities with scripting interfaces for Tcl/Tk, Java programming language, and Python in addition to C++ libraries. VTK is portable to a number of operating systems, including UNIX, Windows, and MAC OS X.

The MayaVi shell around VTK can be imported as a Python module from other Python programs and scripted through the Python interpreter. The tkinter GUI provided by MayaVi allows the configuration and application of filters as well as manipulating the lighting effects on the visualization.

Figure 8 is an example visualization using MayaVi on the Windows platform.
Figure 8. 3-D visualization with MayaVi (CT heart scan data)

MayaVi is an interesting example of extending the VTK in the Python scripting language.

Maxima

Maxima is a full symbolic and numerical computation program in the vein of Octave and Scilab. The initial development of Maxima began in the late 1960s at Massachusetts Institute of Technology (MIT), and it continues to be maintained today. The original version (a computer algebra system) was called DOE Macsyma and led the way for later development of more commonly known applications such as Mathematica.

Maxima provides a nice set of capabilities that you'd expect (such as differential and integral calculus, solving linear systems and nonlinear sets of equations) along with symbolic computing abilities. You can write programs in Maxima using traditional loops and conditionals. You'll also find a hint of Lisp in Maxima (from functions such as quoting, map and apply). Maxima is written in Lisp, and you can execute Lisp code within a Maxima session.

Maxima has a nice online help system that is hypertext based. For example, if you want to know how a particular Maxima function works, you can simply type `example(desolve)` and it provides a number of example usages.

Maxima also has some interesting features such as rules and patterns. These rules and patterns are used by the simplifier to simplify expressions. Rules can also be used for commutative and noncommutative algebra.

Maxima is much like Octave and Scilab in that an interpreter is available to interact with the user, and the results are provided directly in the same window or popped up in another. In Figure 9, I request plot of a simple 3-D graph.
Figure 9. Interacting with Maxima

```
[root@plato build-with-gcc]# maxima
Maxima 5.10.0 http://maxima.sourceforge.net
Using Lisp CLISP 2.41 (2006-10-13)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
This is a development version of Maxima. The function bug_report()
provides bug reporting information.
(%i1) plot3d((atan(-x^2+y^3/4)],[x,-4,4],[y,-4,4],[grid,60,50],
[gnuplot_preamble,"set pm3d at s:unset surfaceset contours
contoursolver
levels 20;unset key"];
(%o1)
(%i2)  
```

The resulting plot is shown in Figure 10.

Figure 10. The resulting Maxima plot from the commands in Figure 9

Open Data Explorer (OpenDX)

An overview of visualization tools wouldn't be complete without a short introduction to Open Data Explorer (OpenDX). OpenDX is an open source version IBM's powerful visualization data explorer. This tool was first released in 1991 as the Visualization Data Explorer, and is now available as open source for data visualization as well as building flexible applications for data visualization.

OpenDX has a number of unique features, but its architecture is worth mentioning. OpenDX uses a client/server model, where the client and server applications can reside on separate hosts. This allows the server to run on a system designed for high-powered number crunching (such as a shared memory multi-processor) with clients running separately on lesser hosts designed more for graphical rendering. OpenDX even allows a problem to be divided amongst a number of servers to be crunched simultaneously (even heterogeneous servers).

OpenDX supports a visual data-flow programming model that allows the visualization program to be defined graphically (see Figure 11). Each of the tabs define a "page" (similar to a function). The data is processed by the transformations shown, for example, the middle "Collect" module collects input objects into a group, and then passes them on (in this case, to the "image" module which displays the image and the "AutoCamera" module which specifies how to view the image).
Figure 11. Visual Programming with OpenDX

OpenDX even includes a module builder that can help you build custom modules.

Figure 12 shows a sample image that was produced from OpenDX (this taken from the Physical Oceanography tutorial for OpenDX from Dalhousie University). The data represents land topology data and also water depths (bathemetry).

Figure 12. Data Visualization with OpenDX

OpenDX is by far the most flexible and powerful data visualizer that I've explored here, but it's also the most complicated. Luckily, numerous tutorials (and books) have been written to bring you up to speed, and are provided in the Related topics section.

Going further

I've just introduced a few of the open source GNU/Linux visualization tools in this article. Other useful tools include Gri, PGLOT, SciGraphica, plotutils, NCAR Graphics, and ImLib3D. All are open source, allowing you to see how they work and modify them if you wish. Also, if you're looking for a great graphical simulation environment, check out Open Dynamics Engine (ODE) coupled with OpenGL.
Your needs determine which tool is best for you. If you want a powerful visualization system with a huge variety of visualization algorithms, then MayaVi is the one for you. For numerical computation with visualization, GNU Octave and Scilab fit the bill. If you need symbolic computation capabilities, Maxima is a useful alternative. Last, but not least, if basic plotting is what you need, gnuplot works nicely.
Related topics

- Browse a large number of Octave scripts, functions, and extensions at the GNU Octave Repository. You'll also find instruction for adding your own recipes to the octave-forge.
- The gnuplot home page is the place for gnuplot software downloads and documentation. You can also find a demo gallery to help you figure out what's possible with gnuplot and how to tailor these recipes for your application.
- GPlot is a Perl wrapper for Gnuplot. It's written by Terry Gliedt and may help you if you find Gnuplot to be complicated or unfriendly. GPlot loses some of the flexibility of Gnuplot, but extends many of the most common options in a much simpler way.
- GNU Octave is a high-level language for numerical computation that uses gnuplot as its graphical engine. It's a great alternative to the commercial Matlab software. Its Web site contains downloads and access to a wide range of documentation.
- You can download the MayaVi Data Visualizer at SourceForge.net. You can also find documentation here, as well as a list of features that MayaVi provides for VTK.
- Maxima is another alternative to Maple and Mathematica, in addition to the open source alternatives, Octave and Scilab. It has a distinguished lineage and supports not only numerical capabilities, but also symbolic computation with inline Lisp programming.
- The Open Data Explorer is an open source version of IBM's powerful data visualization and application development package that's a must for hardcore scientific visualizations.
- Data Explorer tutorials, maintained by Dalhousie Physical Oceanography, nicely demonstrate the power provided by DX.
- The NCAR Graphics home page provides a stable UNIX package for drawing contours, maps, surfaces, weather maps, x-y plots, and many others.
- Gri is a high-level language for scientific graphics programming. You can use it to construct x-y graphs, contour plots, and image graphs with fine control over graphing attributes.
- SciGraphica is great for data analysis and technical graphics.
- The ImLib3D library is an open source package for 3-D volumetric image processing that strives for simplicity.
- ODE is an open physics engine that's perfect for physical systems modeling. Combine this with Open/GL and you have a perfect environment for graphical simulation.
- The ROOT system is a newer object-oriented data analysis framework. ROOT is a fully featured framework with over 310 classes of architecture and analysis behaviors.
- Start developing with product trials, free downloads, and IBM Bluemix services.