Figures, Graphs, and Plots in Octave

10 Replies

This twelfth article of the mathematical journey through open source, shows the mathematical visualization in octave.

<< Eleventh Article

Mathematics is incomplete without visualization, without drawing the results, and without plotting the graphs. *octave* uses the powerful *gnuplot* as the backend of its plotting functionality. And in the frontend provides various plotting functions. Let's look at the most beautiful ones.

Basic 2D Plotting

The most basic plotting is using the *plot()* function, which takes the Cartesian x & y values. Additionally, you may pass, as how to plot, i.e. as points or lines, their style, their colour, label, etc. Supported point styles are: +, *, o, x, ^, and lines are represented by -. Supported colours are: k (black), r (red), g (green), b (blue), y (yellow), m (magenta), c (cyan), w (white). With this background, here is how you plot a sine curve, and Figure 12 shows the plot.

```
$ octave -qf
octave:1> x = 0:0.1:2*pi;
octave:2> y = sin(x);
octave:3> plot(x, y, "^b");
octave:4> xlabel("x ->");
octave:5> ylabel("y = sin(x) ->");
octave:6> title("Basic plot");
octave:7>
```

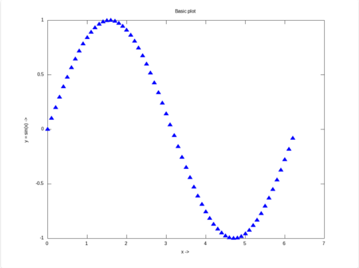

Figure 12: Basic plotting in octave

xlabel(), and *ylabel()* adds the corresponding labels, and *title()* adds the title. Multiple plots can be done on the same axis as follows, and Figure 13 shows the plots. Note the usage of *legend()* to mark the multiple plots.

```
$ octave -qf
octave:1> x = 0:0.1:2*pi;
octave:2> plot(x, sin(x), "*", x, 1 + sin(x), "-", x, cos(x), "o");
octave:3> xlabel("x ->");
octave:4> ylabel("y ->");
octave:5> legend("sine", "1 + sine", "cosine");
octave:6> title("Multiple plots");
octave:7>
```

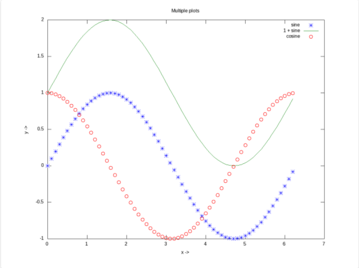

Figure 13: Multiple plots on the same axis

Advanced 2D Figures

Now, if we want to have the multiple graphs on the same sheet but with different axes as shown in Figure 14, here is how to do that:

```
octave:1> t = 0:0.1:6*pi;
octave:2> subplot(2, 1, 1);
octave:3> plot(t, 325 * sin(t));
octave:4> xlabel("t (sec)");
octave:5> ylabel("V_{ac} (V)");
octave:6> title("AC voltage curve");
octave:7> grid("on");
octave:8> subplot(2, 1, 2);
octave:9> plot(t, 5 * cos(t));
octave:10> xlabel("t (sec)");
octave:11> ylabel("I_{ac} (A)");
octave:12> title("AC current curve");
octave:13> grid("on");
octave:14> print("-dpng", "multiple_plots_on_a_sheet.png");
octave:15>
```

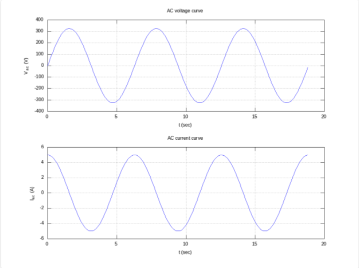

Figure 14: Multiple plots on a sheet

Note the usage of *subplot()*, taking the matrix dimensions (row, column) and the plot number to create the matrix of plots. In the example above, it created a 2×1 matrix of plots. As add-ons, we have used the *grid("on")* to show up the dotted grid lines, and *print()* to save the generated figure as a .png file.

It is not always easy to plot everything in Cartesian co-ordinates, or rather many things are easier to plot in polar co-ordinates, e.g. a spiral, circle, heart, etc. The following code & Figure 15 shows a few such examples. Shown along with is a technique of modifying the figure properties, after drawing the figure using the *set()* function. Here it modifies the line thickness.

```
octave:1> th = 0:0.1:2*pi;
octave:2> r1 = 1.1 .^ th;
octave:3> r2 = 7 * cos(th);
octave:4> r3 = 5 * (1 - cos(th));
octave:5> r = [r1; r2; r3];
octave:6> ph = polar(th, r, "-");
octave:7> set(ph, "LineWidth", 4);
octave:8> legend("spiral", "circle", "heart");
octave:9>
```

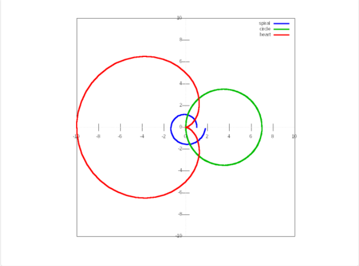

Figure 15: Polar plots

There are many other possible ways of drawing various interesting 2-D figures for all kind of mathematical & scientific requirements. So, before closing on 2-D plotting, let's look into just one more often needed drawing – plotting with log axis, and more over with two y-axes on a single plot. The function for that is *plotyy()*. Note the *plotyy()* calling the corresponding function pointers *@plot*, *@semilogy* passed to it, in the following code segment. Figure 16 shows the output.

```
octave:1> x = 0:0.1:2*pi;
octave:2> y1 = sin(x);
octave:3> y2 = exp(exp(x));
octave:4> ax = plotyy(x, y1, x, y2, @plot, @semilogy);
octave:5> xlabel("x ->");
octave:6> ylabel(ax(1), "sine ->");
octave:7> ylabel(ax(2), "e^{e^x} ->");
octave:8> title("Mixed plots");
octave:9>
```

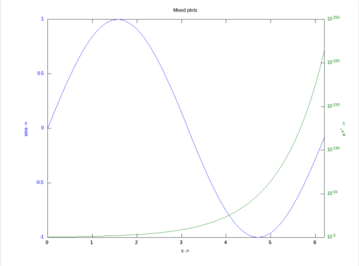

Figure 16: Mixed plots with semi log axis

3D Visualization

And finally, let's do some 3-D plotting. *plot3()* is the simplest *octave* function to do a simple 3-D drawing, taking the set of (x, y, z) points. Here's the sample code to draw the dwindling

Search

CATEGORIES

- Electronics
- Embedded Systems
- Linux
- Linux Device Drivers
- Linux Kernel Internals
- Logic
- Mathematics
- Philosophy

RECENT POSTS

- Toolchain Setup
- Toolchain over a Cup of Coffee
- Embedded Systems for your Boy Friend
- Poisoned Dish
- Weighing Stones

TAGS

555 circuit 555 timer IC circuit anger basic electronics basic electronics tutorial building electronic circuit Character Drivers deceit Device Drivers DIY dravya ego Electronic Circuits Electronics File System Drivers File System Modules File Systems fundamentals of 555 fundamentals of 555 timer IC greed karm LED LFY Linux Linux Driver Linux Kernel Development Linux Kernel Internals lower order Maths Maxima mokṡ Octave OSFY parmāṅu Polynomials procedural recursive logic Puzzles Recursion recursive logic recursive relation Resistor Semester Project simple electronic circuits terminating condition touch

RECENT COMMENTS

Waiting / Blocking in a Linux Driver Part 3 – Embitude on Waiting / Blocking in Linux Driver Part – 2
Waiting / Blocking in Linux Driver – Embitude on Waiting / Blocking in Linux Driver Part – 2
Anil Kumar Pugalia on Playing with ALSA loopback devices
Michael on Playing with ALSA loopback devices
Girish Kumar on Toolchain Setup

ARCHIVES

- April 2021
- March 2021
- February 2021
- January 2021
- December 2020
- November 2020
- October 2020
- September 2020
- August 2020
- July 2020
- June 2020
- May 2020
- April 2020
- March 2020
- February 2020
- January 2020
- December 2019
- November 2019
- October 2019
- September 2019
- August 2019
- July 2019
- June 2019
- May 2019
- April 2019
- March 2019
- February 2019
- January 2019
- December 2018
- November 2018
- October 2018
- September 2018
- August 2018
- July 2018
- June 2018
- May 2018
- April 2018
- March 2018
- February 2018
- January 2018
- December 2017
- November 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016
- September 2016
- August 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015
- November 2015
- October 2015
- September 2015
- August 2015
- July 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- November 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- April 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- May 2013
- April 2013
- March 2013
- February 2013

sinusoidal curve shown in Figure 17:

```
octave:1> x = -10:0.1:10;
octave:2> y = 10:-0.1:-10;
octave:3> z = x .* sin(x - y);
octave:4> plot3(x, y, z, "-", "LineWidth", 4);
octave:5> xlabel("x ->");
octave:6> ylabel("y ->");
octave:7> zlabel("z ->");
octave:8> title("Dwindling sinusoidal");
octave:9> grid("on");
octave:10>
```

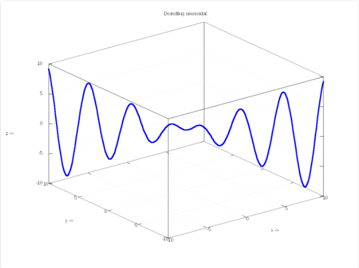

Figure 17: 3-D plot of a dwindling sinusoidal

In case, we want to plot the values of a 2-D matrix against its indices (x, y), it could be done using *mesh()*, one of the many other 3-D plotting functions. Figure 18 shows the same, created using the following code:

```
octave:1> x = 0:0.1:2*pi;
octave:2> y = 0:0.1:2*pi;
octave:3> z = sin(x)' * sin(y);
octave:4> mesh(x, y, z);
octave:5> xlabel("x ->");
octave:6> ylabel("y ->");
octave:7> zlabel("z ->");
octave:8> title("3-D waves");
octave:9>
```

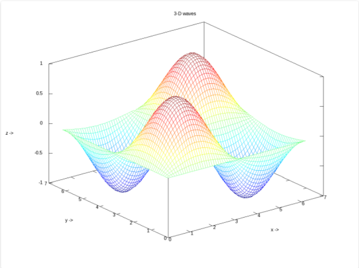

Figure 18: 3-D waves

What's next?

Hope you enjoyed the colours of drawing. Next, we would look into *octave* from a statisticians perspective.

Thirteenth Article >>

Anil Kumar Pugalia (123 Posts)

The author is a hobbyist in open source hardware and software, with a passion for mathematics, and philosopher in thoughts. A gold medallist from the Indian Institute of Science, Linux, mathematics and knowledge sharing are few of his passions. He experiments with Linux and embedded systems to share his learnings through his weekend workshops. Learn more about him and his experiments at https://sysplay.in.

Send article as PDF | Enter email address | Send

This entry was posted in Mathematics and tagged Figures, Graphs, LFY, Linux, Maths, Octave, OSFY, Plots on January 1, 2014 by Anil Kumar Pugalia.

About Anil Kumar Pugalia

The author is a hobbyist in open source hardware and software, with a passion for mathematics, and philosopher in thoughts. A gold medallist from the Indian Institute of Science, Linux, mathematics and knowledge sharing are few of his passions. He experiments with Linux and embedded systems to share his learnings through his weekend workshops. Learn more about him and his experiments at https://sysplay.in.

View all posts by Anil Kumar Pugalia →

← USB Drivers in Linux

USB Drivers in Linux (Continued) →

10 thoughts on "Figures, Graphs, and Plots in Octave"

Pingback: Polynomial Curve Fitting & Interpolation | Playing with Systems

Pingback: Octavian Statistics | Playing with Systems

David Nelson
February 16, 2018 at 12:16 am

Thank you, Anil,

I would like to eliminate the gridlines on the plot and see–and save–just color information.

From advice elsewhere, I tried separately:
surface(x,y,z,'linestyle','none'), shading('interp'), colormap(gray);
surface(x,y,z,'edgecolor','none'), shading('interp'), colormap(gray);
(because I really only want intensities) and both do result in plots which visually do not show visible lines between the values of the mesh. (I understand that the 'interp' shading interpolates missing values).

However, when I "Save-as" the result as a 'png', the image always contains residual evidence of the lines.

Also, if I may, is there a way to lock the aspect ratio of resulting plots? Despite the fact that my x,y values are symmetric, the plots are always over a rectangular grid, instead of square.

Reply ↓

Anil Kumar Pugalia Post author
February 16, 2018 at 12:31 pm

David, I am getting it perfectly fine. Just try "graphics_toolkit("gnuplot");", the first thing when you start octave, and then do your drawings, prints, etc. Let us know, if it helps.

Regarding the aspect ratio, you may try pbaspect() for plot box and daspect() for data. Without parameter, they return the current corresponding aspect ratios. With a 3-D vector, they change the corresponding aspect ratios.

Reply ↓

lakshmikruthiga
January 8, 2019 at 12:20 pm

sir, I have following code, The values of displ in stage1 and stage2 are plotted in displ vs time graph. Thus a single curve is obtained.

```
displ=zeros(100,2)
tim=zeros(100,1)
for i=2:100
v=5;
dt=1*10^-4;
function [displ_f]=stage1(displ_i)
dt=1*10^-4;
v=5;
displ_f=displ_i+v*dt;
endfunction
function [displ_f]=stage2(displ_i)
dt=1*10^-4;
v=5;
displ_f=displ_i+v*dt;
endfunction
if (displ(i-1,1)=0)
disp("IN stage 1");
[displ(i,1)]=stage1(displ(i-1,1));
displ(i,2)=v;
tim(i,1)=tim(i-1)+dt;
elseif (displ(i-1,1)>=0.020 && displ(i-1,1)==0)
disp("IN stage 2");
[displ(i,1)]=stage2(displ(i-1,1));
displ(i,2)=displ(i-1,2);
tim(i,1)=tim(i-1)+dt;
end
end
disp(" displ vs tim")
plot(tim(:,1),displ(:,1));
```

How will I get stage1 curve in orange color and stage2 curve in blue color in the same plot to know which stage has which displ values? kindly help me.

Reply ↓

Anil Kumar Pugalia Post author
January 9, 2019 at 4:18 pm

You may have to draw them as two separate curves.

BTW, I am not getting any curve from your above program.

Reply ↓

John McGeorge
November 8, 2020 at 12:48 am

Hello.I am new to Octave.Would like to plot the rear suspension in 3D.not sure how to do that.I am a 61 year old. Please need help.Thanks

Reply ↓

Anil Kumar Pugalia Post author
November 10, 2020 at 1:40 pm

Do you have the mathematical equation of the same? If yes, plug it in, as per the above examples.

Reply ↓

Dan
December 31, 2020 at 4:05 pm

Hi Anil,
Firstly thank you for this post,
I have a doubt, regarding the syntax, as i am facing an error (Octave 6.1.0 on Ubuntu)
Here is what i am trying to do:
%Plot the positive and negative examples on a
% 2D plot, using the option ‘k+’ for the positive
%examples and ‘ko’ for the negative examples.
% here is how the X looks :- 34, 56 and y looks as only: 0/1 both X and y have the same no of rows
pos = find(y==1);
neg = find(y==0);
plot(X(pos,1),X(pos,2),’k+’,’LineWidth’,2,’MarkerSize’,7);
plot(X(neg,1),X(neg,2),’ko’,’MarkerFaceColor’,’y’,’MarkerSize’,7); %If i take out the ‘MarkerFaceColor’ part %then its doing what its supposed to, i referred to the docs, the way i understood it represents the %syntax as plot (x, [y,] FMT, …) I am confused about the [y,] part and to what arguments the FMT part %applies to, is there a problem with my code or the ‘MarkerFaceColor’ property has a problem as when i add that part in it plots the ‘neg’ points twice once in yellow and then again exchanges the axes and plots them again without the color, also it erases my ‘pos’ points
title(‘Figure 1:Scatter plot of training data’);
xlabel(‘Exam 1 Score’);
ylabel(‘Exam 2 Score’);
legend(‘Admitted’,’Not Admitted’);

Reply ↓

Anil Kumar Pugalia Post author
January 6, 2021 at 9:34 am

All other things seem to be okay, except that the two plot calls should be merged into one to have both the +ve & -ve plots on the same graph. Something like: plot(X(pos, 1), X(pos, 2), ‘k+’, ‘LineWidth’, 2, ‘MarkerSize’, 7, X(neg,1), X(neg,2), ‘ko’, ‘MarkerFaceColor’, ‘y’, ‘MarkerSize’, 7);

Reply ↓

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

Proudly powered by WordPress