MATLAB Plots

John A. Gubner, *Member*, *IEEE*

I. INTRODUCTION

To get a MATLAB plot like Fig. 1, which would look good in a LaTeXed document for the IEEE Transactions, requires a little work. First, the column width in the Transactions is 21 picas (pc). So to insert a graph and make it 21 pc wide, use the commands[1]

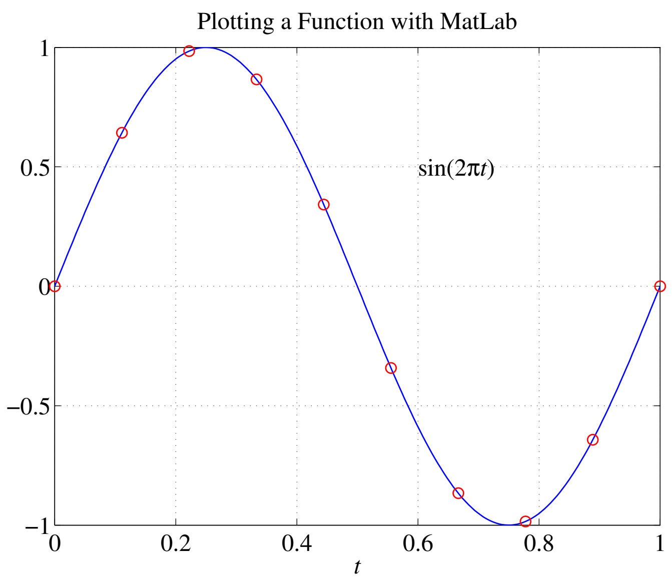

Fig. 1. A typical graph.

```
\begin{figure}
\centering\includegraphics[width=21pc]{plotfile}
\caption{Your caption.}
\label{yourlabel}
\end{figure}
```

You do not need to use the extension in the file name. If the file is `plotfile.eps` and you are running `latex`, the correct file will be used. If you use `eps2pdf` to convert `plotfile.eps` to `plotfile.pdf` and you run `pdflatex`, the correct file will be used.

The problem with the foregoing is that MATLAB generates plots that are wider than 21 pc, and so the plots, including the text, will be reduced. This makes the numbers on the graph hard to read. Therefore, it is necessary to use extra-large fonts for the numbers on the axes and in any labels or other text on the graph. It is also necessary to use thicker lines and bigger markers.

You can change the default settings of MATLAB with the following command.

```
set(0,'DefaultTextFontName','Times',...
      'DefaultTextFontSize',18,...
      'DefaultAxesFontName','Times',...
      'DefaultAxesFontSize',18,...
      'DefaultLineLineWidth',1,...
      'DefaultLineMarkerSize',7.75)
```

If you put this command in your `startup.m` file, the command will run automatically every time MATLAB starts.

To produce the graph in Fig. 1, we used the following code, after running the above command.

```
t = linspace(0,1,200);   % Create data to plot
y = sin(2*pi*t);
tau = linspace(0,1,10);
x = sin(2*pi*tau);

plot(t,y,tau,x,'ro')

grid on;   % optional

% Optionally add some text, a label, and a title
text(0.6,0.5,'sin(2\pi\itt\rm)')
xlabel('\itt')
title('Plotting a Function with MatLab')

print('-depsc2','-r600','plotfile.eps') % Print to file
```

Sometimes you do not need such a large graph to present your results clearly. To produce the small graph in Fig. 2, we made two changes to what we did before. First, before the `plot` command, we inserted the command `subplot(2,2,1)`. Second, in the `\includegraphics` command, we used `width=11pc` instead of 21 pc as we did before.

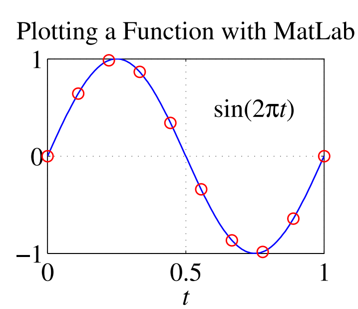

Fig. 2. A small graph.

J. A. Gubner is with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706-1691 USA (e-mail: gubner@engr.wisc.edu).

[1] Be sure to put `\usepackage{graphicx}` in your LaTeX preamble.

If we use `subplot(2,1,1)` and `width=21pc`, we get

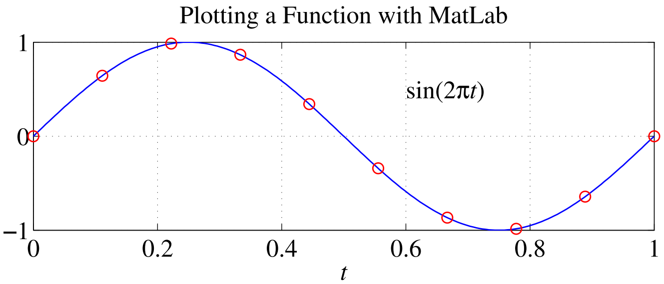

Fig. 3. A short but wide graph.

Fig. 4 shows a more complicated example. The code for this graph is shown below.

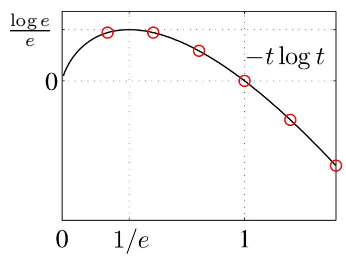

Fig. 4. A small graph of $-t \log t$.

```
% Script to plot -t log t function
f = @(t)-t.*log(t);
t = linspace(0,1.5,200);
y = f(t);
tau = linspace(0,1.5,7);
x = f(tau);

subplot(2,2,1)
plot(t,y,'k',tau,x,'ro'); grid on

xlabels = strvcat('0', ' ', '1');
ylabels = strvcat('0', ' ' );
set(gca,'XTick',[0 1/exp(1) 1],'XTickLabel',xlabels,...
     'YTick',[0 1/exp(1)],'YTickLabel',ylabels)
text('Interpreter','latex','String','$- \kern.8em t \log t$',...
   'Position',[1 1/exp(1)-.2 ])
text('Interpreter','latex','String','$\frac{\log e}{e}$',...
   'Position',[-.3 1/exp(1)])
text('Interpreter','latex','String','$1/e$',...
   'Position',[1/exp(1)-.09 -1.15])

print('-depsc2','-r600','plotfile.eps')
```

II. Histograms

The short script below simulates the sum of two uniform random variables and plots a *normalized* histogram of the result. The reason for using a normalized histogram is so that when we plot the true density over it, the heights will match as shown in Fig. 5.

More specifically, let $Z := X+Y$, where X and Y are independent with $X \sim \text{uniform}[0,1]$ and $Y \sim \text{uniform}[0,2]$. The density of Z is the convolution of the densities of X and Y. Hence, the density of Z has the shape of a trapezoid.

```
%% Histograms
n = 50000;        % Number of simulations
X = rand(1,n);
Y = rand(1,n)*2;
Z = X+Y;
nbins = 40; % Number of bins for histogram
hstgrm = makedenshist(Z,nbins);
plothist(hstgrm)
% Now plot true density of Z
z = linspace(0,3,200);
f = @(z).5*(z.*(0<=z & z<1)+(1<=z & z<2)+(3-z).* ...
    (2<=z & z<=3));
hold on
plot(z,f(z),'k'); grid on
hold off
print('-depsc2','-r600','plotfile.eps')
```

The two functions `makedenshist` and `plothist` are shown on the next page.

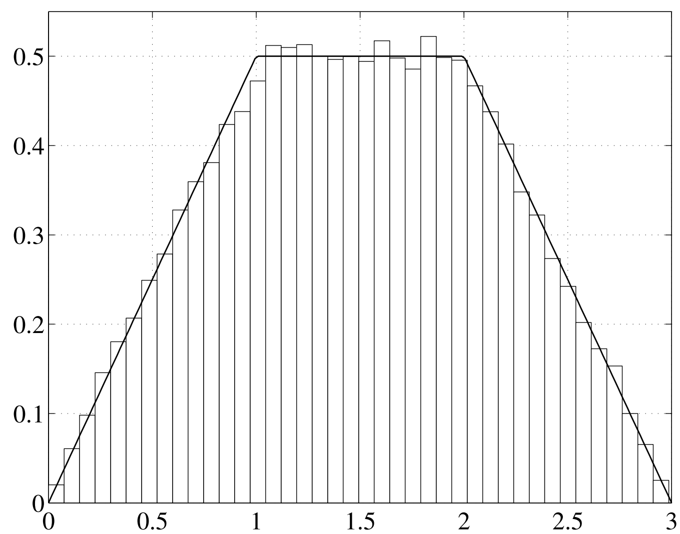

Fig. 5. A histogram normalized by the number of samples `n` and the bin width `bw` so that result has the scale of a probability density.

```
function [hstgrm,varargout] = makedenshist(Z,nbins)
% Make a density histogram with nbins bins out of the data in Z.
% We return the 2-by-nbins array hstgrm, where
% hstgrm(1,:) = the list of bin centers, and
% hstgrm(2,:) = normalized histogram heights.
%
% The command
%
%    hstgrm = makedenshist(Z,nbins)
%
% always prints the minimum and maximum data samples,
% denoted by minZ and maxZ.  Alternatively, the command
%
%   [hstgrm,minZ,maxZ] = makedenshist(Z,nbins)
%
% returns these values to you without printing them.

hstgrm = zeros(2,nbins);         % Pre-allocate space

minZ = min(Z);                   % Determine range of data
maxZ = max(Z);
if nargout==3
   varargout{1} = minZ;
   varargout{2} = maxZ;
else
   fprintf('makedenshist: Data range = [ %g , %g ].\n',minZ,maxZ)
end

e = linspace(minZ,maxZ,nbins+1); % Set edges of bins

a = e(1:nbins);                  % Compute centers of bins
b = e(2:nbins+1);                % and store result in
hstgrm(1,:) = (a+b)/2;           % hstgrm(1,:)

H = histc(Z,e);                  % Get bin heights

H(nbins) = H(nbins)+H(nbins+1);  % Put any hits on right-most
                                 % edge into last bin

% Compute and store the normalized bin heights
bw = (maxZ-minZ)/nbins;
hstgrm(2,:) = H(1:nbins)/(bw*length(Z));



function plothist(hstgrm);
% Plot a histogram generated by makedenshist.
% Actually, as long as
% hstgrm(1,:) = the list of bin centers, and
% hstgrm(2,:) = normalized histogram heights,
% plothist will work for you.

bar(hstgrm(1,:),hstgrm(2,:),'hist')
h = findobj(gca,'Type','patch');
set(h,'FaceColor','w','EdgeColor','k')
```