How to produce figures with Matlab that have the right dimensions when printed

Most of you who use Matlab may have already struggled with the problem that if your figures look good on screen, text nicely aligned, plots properly spaced, the print output can nevertheless be a total mess, because Matlab figures are not WYSIWYG (what you see is what you get...). In this tutorial, I will explain how to solve this problem. In the end, you will not have WYSIWYG, but at least YKWYGAW (you know what you get and why).

The figure window and axes

So, first we have to dive a bit into the logic of the Matlab `figure` window. The figure window has coordinates that range from 0 to 1 in the x- and y-direction. The lower left corner is (0,0), and the upper right corner is (1,1). If you want to place a plot inside the figure window, it needs coordinates. If you just use the `plot()` command, Matlab creates an axis with some default coordinates. The command

```
axes('position',[x y w h])
```

creates an axis that has its lower left corner at `(x,y)`, a width `w` and a height `h`, all relative to the figure window. I'd recommend to always use the command `axes()` to create multiple axes inside a figure window instead of `subplot()`. The advantage of `axes` is that you can put several axes on top of each other, for example to create insets, whereas a new subplot always erases a colliding axes/plot/subplot.

Setting proper dimensions for your figure

If your figure and the plots in it shall have a certain size when printed, let's say 8 cm width and 12 cm height for a typical one-column figure, we first need to specify the size and position of the figure on the paper:

```
set(gcf,'PaperUnits','centimeters')
```

This sets the units of the current figure (`gcf` = get current figure) on paper to centimeters.

```
xSize = 8; ySize = 12;
```

These are my size variables, width of 8 and a height of 12, will be used a lot later.

```
xLeft = (21-xSize)/2; yTop = (30-ySize)/2;
```

Additional coordinates to center the figure on A4-paper

```
set(gcf,'PaperPosition',[xLeft yTop xSize ySize])
```

This command sets the position and size of the figure on the paper to the desired values.

```
set(gcf,'Position',[X Y xSize*50 ySize*50])
```

This additional command resizes and repositions the figure window on the computer screen, so that it has the same aspect ratio as the final figure. Thus, you can better estimate what your figure looks like on paper or exported. It is also nice to use this option when you create multiple figures in one script, because you can place the anywhere on the screen with `X` and `Y`.

Placing axes at the desired positions

Now we want to place an axis in our figure at some specific coordinate with a specific size, for example 1.5 cm from the left border, 2 cm from the bottom, width 3 and height 2 cm. Remember that the relative coordinates in our figure window run from 0 to 1, but that we have set the dimensions on paper to `xSize` and `ySize` cm. The relative coordinate for 1.5 cm from the left border is thus

```
x = 1.5/xSize
```

and likewise for the y-coordinate, the width and the height:

```
y = 2/ySize; width = 3/xSize; height = 2/ySize;
```

Now we can generate the axis:

```
axes('position',[x y width height])
```

And for demonstration purposes, another one at `x2 = x + width + 1/xSize`, thus 1 cm to the right of the first axes:

```
axes('position',[x2 y width height])
```

Just print the figure to see the result, or use the print preview. The print preview is accurate. Basically, the procedure described above is all you need to create very complicated composed figures, and you can place your axes with the precision of fractions of millimeters if you like. Try this with the graphical interface...

Adding text and labels (a, b, ...) to the figure

To add text and labels to the figure, I use a slightly awkward approach. I don't claim that it's the best, but it works flawlessly. I simply overlay the entire figure with an invisible extra-axis, and place text within this axis:

```
axes('position',[0 0 1 1],'visible','off')
```

To place the text, you can simply refer to the coordinates you used for placing your axes:

```
text(x,y+height,'a')
text(x2,y+height,'b')
```

writes 'a' at the top of the y-axis of axis 1, and 'b' on top of the y-axis of axis 2. If you want to print the text 1 cm further to the left, just use

```
text(x-1/xSize,y+height,'a')
```

You may have to fiddle around a bit with the text alignment properties for the desired placing of the text relative to the coordinate. Finally, don't forget to scale your text-axis

```
axis([0 1 0 1])
```

so that it has the same coordinate system as the figure window, and everything really ends up in the right place.

Handles, get and set

Matlab handles are the basic concept for addressing and manipulating objects in the figure window. If you want to do specific manipulations on an axis, plot, etc., assign it a handle when creating it:

```
Axis1 = axes(); graph1 = plot(...)
```

This is especially handy if you want to change some parameters that you cannot directly specify for example in the plot command, or when Matlab overactively changes some properties. An example is the `colorbar` command, which adds a colorbar to your current axis, but also resizes it. You can restore the original size of the axis easily:

```
set(Axis1,'position,[x y w h])
```

If you do not know which properties belong to a certain object, simply use `get(HandleName)` in the command line, and you get a whole list of all properties and their current values. You can change each property using `set`, just use

```
set(HandleName,'PopertyName',Value)
```

So, that's about it, I must admit that Matlab's way of doing things is sometimes a bit odd, and using Matlab for creating complicated figures may be kind of an acquired taste, but once you've gotten used to it, it's surprisingly easy and fast.
Hope this is useful for some of you, and if you know how to do some things better or more in a more elegant way, please tell me.

Greetings, Roland