Graphics inclusions in manuscripts

C. D. Biemesderfer
NOAO, Tucson, AZ

There has been considerable development of text processing and typesetting tools over the past decade, and more and more scientists are using these complex programs to produce the material that they publish. Over the past several years, technology for creating finished-quality charts and graphics has started to come of age: versatile plotting and drawing programs are now widely available, high-resolution bit-mapped video displays are becoming commonplace, and laser printer hardware and software has matured.

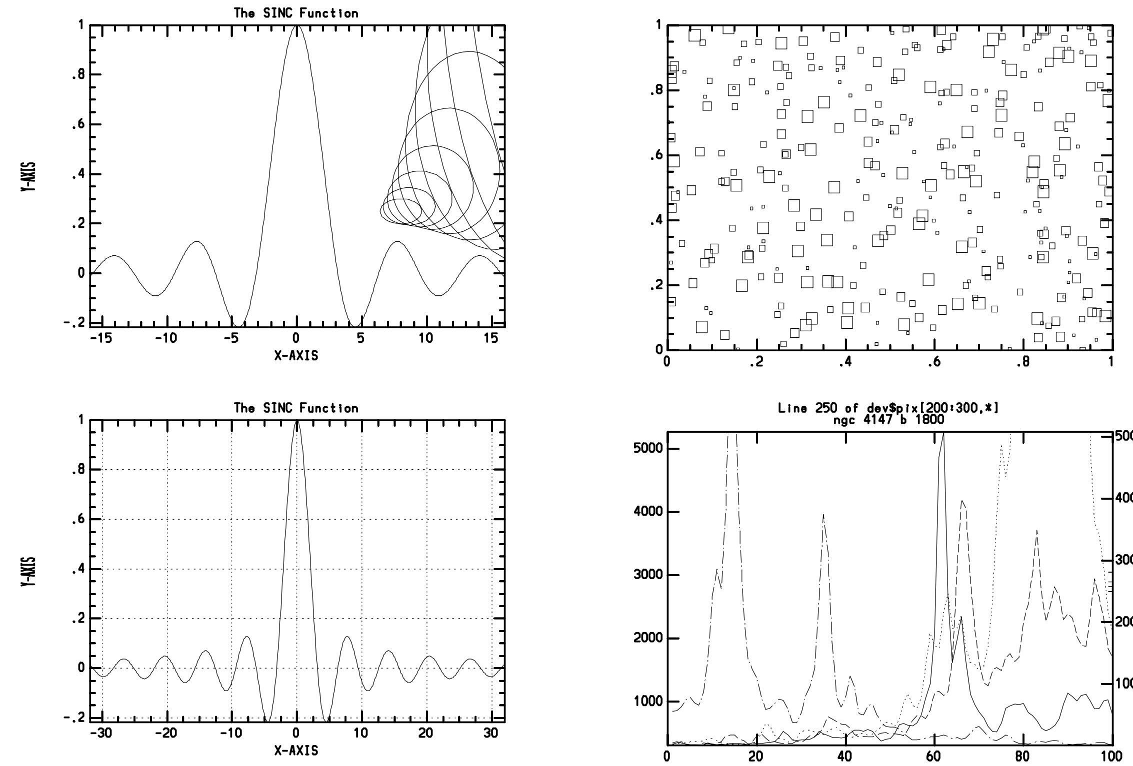

Fig. 1.— IRAF plot

It is natural for astronomers who are publishing scientific results to desire a mechanism for merging their graphical and textual data in a way that ensures a certain integrity in the page layout. Fortunately, the adoption of PostScript[1] as a de facto *page description language*

[1]PostScript is a registered trademark of Adobe Systems Incorporated.

standard by a substantial fraction of the relevant sectors of the industry makes it simpler for programmers to set up the machinery.

1. Why PostScript is good

PostScript has a number of merits that make it a good choice as the language to use for graphics exchange. Most important is that it is a *device-independent* means of describing printing on a page, and has been widely implemented across a variety of printers and display devices. It is fairly well-documented and tested, so different implementations work reliably and behave predictably.

Furthermore, PostScript is composed of text using the ASCII character set. This makes is very easy to transport files on networks, and they are susceptible to manipulation by standard text processing tools, e.g., editors, pagers, etc.

1.1. Encapsulated PostScript

It is desirable for graphics inclusions to conform to certain codes of behavior, so that the graphics can be manipulated readily and reliably. PostScript can be generated at any of several levels of so-called *structuring conventions*, which are a more or less inevitable consequence of PostScript's heritage as a programming language.

When a PostScript program is to be interpreted as a simple page description, it is convenient if it obeys some rules of form. For the purposes of graphics inclusion, the most important property is that the PostScript be *encapsulated*. What this means essentially is that the including application can determine the size and location on the page of the graphic *without* having to interpret any PostScript code.

The boundaries of such a "capsule" graphic are defined by a `BoundingBox` comment that specifies the x and y coordinates of two opposing corners of the graphic. There are other "do's and don'ts" in the quest to produce encapsulated PostScript; however, it is generally sufficient for the including application to specify that imported PostScript graphics conform to the encapsulation standard.

2. Overview of the DVIPS commands

This is intended as a cursory look at the commands that can be used to work with graphics inclusions. It is by no means necessary for authors to know *any* of this; markup language writers can easily shield the end-user from all of the details of the PostScript interface.

A graphic that is delivered to an application in a capsule defined by a bounding box can be subjected to a limited number of operations: translation, truncation, scaling, and rotation. Translation, truncation (called *clipping* in PostScript context), and scaling can be performed on each coordinate independently, hence we can identify seven primitive functions:

`hoffset`	horizontal offset	`voffset`	vertical offset
`hsize`	horizontal clip size	`vsize`	vertical clip size
`hscale`	horizontal scale factor	`vscale`	vertical scale factor
`angle`	rotation angle		

For most graphics inclusions, scaling is the most important function. It is rare to clip or rotate imported graphics, and moving the coordinate system origin from the current point is often troublesome in a text formatter. Furthermore, there is usually no reason to alter the aspect ratio of the graphic, so including encapsulated PostScript is often as simple as specifying a scale factor or a dimension and reading the file.

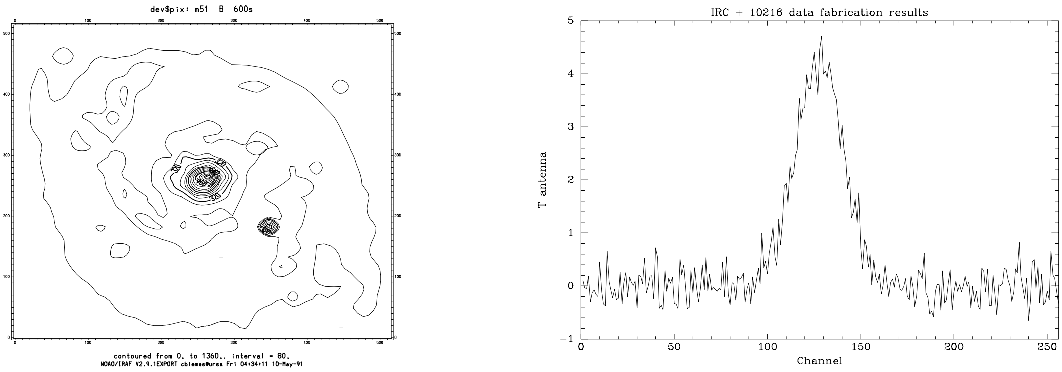

Fig. 2.— Dual IRAF plots

3. Implications for the AASTeX package

We are concerned with the import of two-dimensional graphics and grey-scale images into scientific manuscripts and other technical documentation. At this point in time, it should be adequate to require that all graphics for import be in the form of encapsulated PostScript, and to declare that figures will be imported in their entirety and with the same aspect ratio as in the original. It is then trivial to build some simple macros based on the `epsf` substyle that is supplied with the DVIPS program by its author Tom Rokicki.

The `epsf` macros we need to worry about are `\epsfxsize` and `\epsfbox`, which perform the two functions we determined in the preceding section to be critical. For purposes of having prototype macros, I wrote two simple macros:

`\plotone{FILE}` reads the PostScript in FILE and adjusts the scale such that the x-coordinate width of the graphic matches the `\textwidth` of the manuscript. Figure 1 on page 1 is an example.

`\plottwo{FILE1}{FILE2}` reads the PostScript from two files and scales each to fit across half the `\textwidth` (actually, slightly less than half so the graphics don't collide). Figure 2 on page 3 shows a pair of dueling inclusions.

The arguments to the `\plotone` and `\plottwo` commands should not include additional scaling and rotation information. These commands are suggested as part of the AASTEX markup conventions, but it is entirely reasonable for authors to use the markup syntax of DVIPS for handling encapsulated PostScript graphics. If direct access to the graphic is desired, it is necessary to use the interface syntax defined by Rokicki.

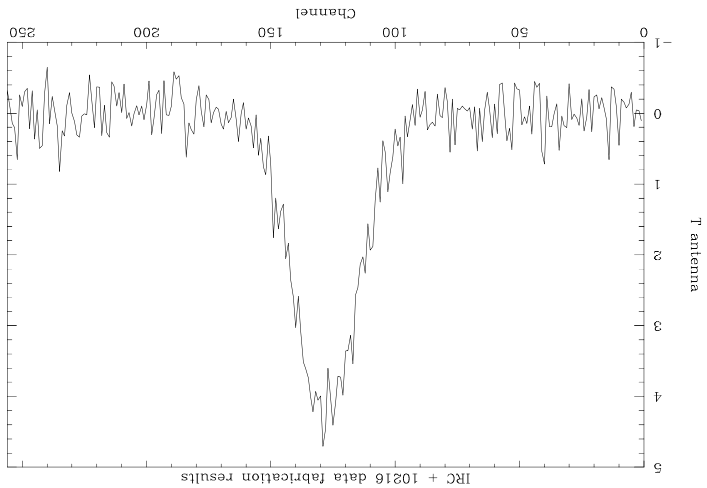

Fig. 3.— Inverted IRAF plot

One has to be careful, though, when using this approach, since the automatic link between the formatter (LATEX) and the PostScript is abandoned. One can get great effects, or totally funky ones . . .

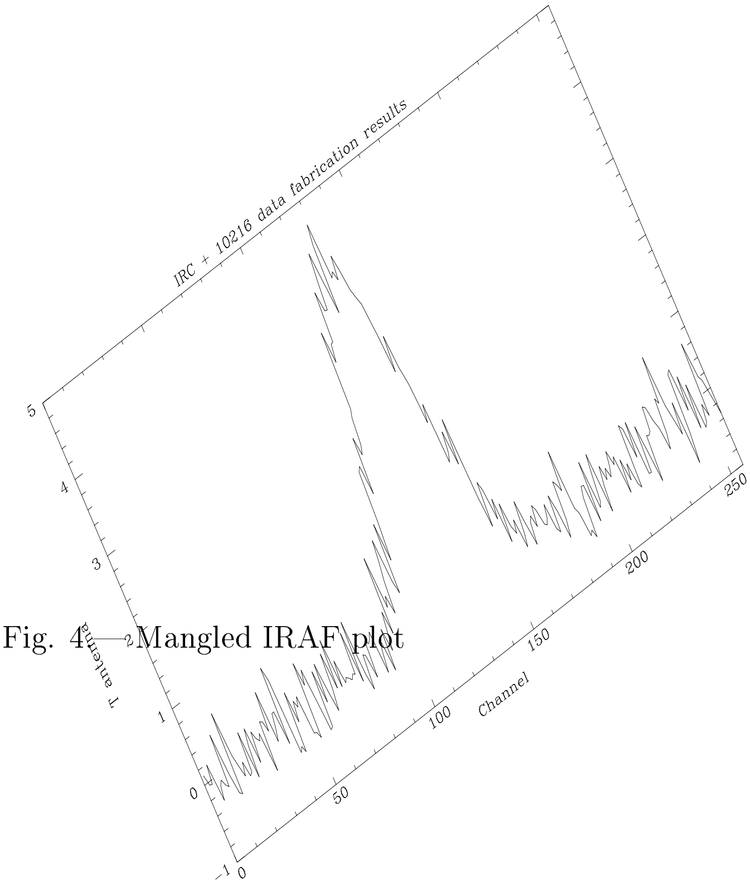

Fig. 4.— Mangled IRAF plot