Convenient Labelling of Graphics, the `WARMreader` Way

Wendy McKay
Control and Dynamical Systems
California Institute of Technology
Pasadena, CA 91125
wgm@cds.caltech.edu
URL: http://www.cds.caltech.edu/~wgm/

Ross Moore
Mathematics Department
Macquarie University
Sydney, Australia 2109
ross@mpce.mq.edu.au
URL: http://www.maths.mq.edu.au/~ross/

Abstract

This article describes a system for placing labels on included graphics in a way that does not require the user to be concerned with explicit lengths or coordinates. The full system was developed specifically for use on Macintosh computers but, due to its modularity, can be used with other systems as well.

The `WARMreader` (read 'Wendy And Ross', selecting either for the 'M') package defines macros to read information from a file, indicating the location of specially marked points where labels may be desired. It also provides a link to the XY-pic macros, which allow arbitrary labels to be attached at these points.

Two applications, *Zephyr* and *Mathematica*, are used to demonstrate techniques for creating files readable for `WARMreader`, including ways to overcome specific difficulties. Other methods can be used and `WARMreader` programmed to read the resulting data files.

Various pieces of software and techniques exist for using TEX to put labels onto included graphics. All have significant drawbacks or shortcomings. One method that is widely used, and often recommended as best for Encapsulated PostScript (EPS) files, is to first Typeset the label using *Textures* on a Macintosh, Copy the resulting typeset window, then Paste the clipboard contents into the image file, having been opened within Adobe's *Illustrator* application. Among the drawbacks of this technique are:

- dependence upon a particular computing platform: Macintosh, or PowerMac
- use of expensive commercial software: Adobe's *Illustrator*, and Blue Sky's *Textures*
- applicability to just a particular image format: Encapsulated PostScript
- the original image file must be altered (after copying, please!) to obtain the required results

Depending upon the working environment, these may not be problems at all; for example, a prepress house would be expected to have the appropriate hardware and software. Similarly an academic may have made the investment to be able to follow this strategy.

However, there is a problem which may cause great difficulties when a manuscript is submitted for publication. Suppose a labelled image needs to be resized or the labels need to be changed for some reason; e.g. the text style chosen does not blend well with the fonts and styles used elsewhere in the publication. Now the EPS file needs to be edited or regenerated in the same way as was done originally. This may no longer be possible — the software used to create it may not be available or the expertise to use it may have been lost.

The `WARMreader` solution is to use TEX itself, or LATEX, for placing the labels. It uses the XY-pic diagram macros, extending the methods presented at TUG'97[3], and available on the Web. The idea is to create a coordinate system tailored for the size of the imported image, anchoring labels at appropriate places using these coordinates. This effectively creates an overlay which allows the labels to seem to be part of the image, when in fact they have been typeset by TEX. `WARMreader` takes this further, by automating the process so that a user does not have to be concerned with coordinates when specifying the labels. Since the styles and content of the labels are specified within the TEX or LATEX source, there is no need to alter any EPS files. Furthermore, this can be done for graphics of any format that can be included within a TEX document, by whatever means. The only requirement is the ability to create a `.bb` file,[1] containing information in an appropriate form.

For LATEX, the PSfrag package, as described in *The LATEX Graphics Companion* [1, pp. 460–462], provides similar functionality for EPS files, by treating parts of the file as tags to be later replaced by blocks of TEX-typeset material. This technique has several limitations, apart from being available only for LATEX, and not usable with graphics formats other than PostScript. For best results, the PostScript file "should ideally be designed with PSfrag in mind", and for systematic use, it "requires a good understanding of both the PostScript language and the application generating the figures"[1, p. 462]. This is because the replacement portions effectively become part of the PostScript graphic at the point where the tags occur, so are subject to, and must dovetail with, the PostScript graphics state at those places. As this includes color, size, rotation and cropping-region, great care is required to avoid later parts of the graphic obscuring earlier labels or labels being cut off at edges of the graphic. It is not possible to know exactly how the whole thing will appear until the `.dvi` file has been processed with a PostScript-aware viewer or printer, thus making it tedious to fine-tune the placement of labels.

With `WARMreader`, the labels can be regarded as occurring within a separate layer, controlled completely from within TEX or LATEX. Any graphic from any source, in any format that can be handled by the TEX installation, can be used as a "backdrop", provided that a suitable `.bb` file has been prepared. Each of the following three steps can be done quite independently; e.g. by different people using different software or techniques:

1. construction of the graphic
2. make a `.bb` file, perhaps with text for labels
3. preparation of code for processing labels within the TEX document

[1] Such files are used with LATEX's `\DeclareGraphicsRule` [1, pp. 40–41] for holding just the bounding-box information, since this is all that is needed for TEX to leave sufficient space for an image.

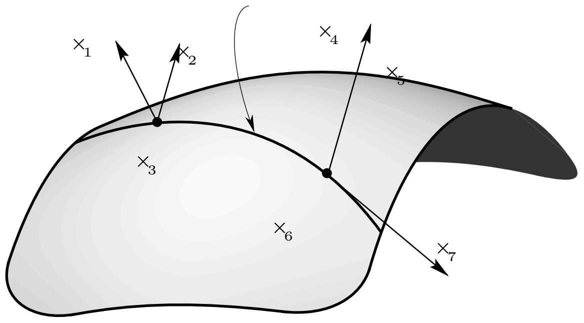


Figure 1: Imported image with "marked points" indicated explicitly.

Only the last *requires* knowledge of TEX or LATEX, though this is desirable if labels are to be completely specified in the `.bb` file. Indeed it will become apparent below that the greatest control over the final appearance, hence the best results, are obtained when these three tasks are kept completely separate.

Detailed Example with Marked Points

The best way to explain how the `WARMreader` macros work is with an example. Figure 1 shows an EPS image prepared for a mathematics text [2]. Numbered $\times$s are not part of the image but indicate "marked points", serving as anchors for placement of labels.

Information for the marked points in Fig. 1 is contained in a file named `Fig541.bb`, with the graphic itself being named `Fig541.eps`. The `.bb` file gives the natural size (in points) of the imported graphic as well as coordinates for marked points.

In addition to being numbered in sequence, a text string may be given for each marked point. This can be used to help identify why the point has been marked. It may even provide the TEX code intended to be used to specify the label, though it is not at all necessary to use it for this purpose. For example, the code used to produce Fig. 1 was as follows:

```
\begin{xy}
 \xyShowAllMarkedPoints{}{Fig5.4.1}{eps}
\end{xy}
```

Techniques to create a file such as `Fig541.bb`, see Fig. 2, are discussed towards the end of this article.

A side-effect of `\xyShowAllMarkedPoints` is to write the coordinates and text strings for each of the marked points into the TEX `.log` file. The purpose of this is to facilitate preparation of the required labels over several consecutive processing runs.

Adding Labels. There are several commands provided for placing labels anchored at marked points. The simplest, but not always the most effective, of these is useful when the required labels are provided

```
%%Creator: PICT Displayer, by David Rand, Version 1.0, March 1999
%%Title: (Fig5.4.1.eps)
%%Date: 3/13/998h49 PM
%%IMPORTANT: The following BoundingBox indicates only the size of the box, not its position!
%%BoundingBox: 0 0 224 134
%%Coordinates: LL
%%StartMarkedPoints
%%MarkedPoint: ( 38,118)  1 %F_{\lambda}^*t(m_{\lambda})
%%MarkedPoint: ( 77,115)  2 %t(m)
%%MarkedPoint: ( 62, 72)  3 %m
%%MarkedPoint: (130,123)  4 %integral curve of $X$
%%MarkedPoint: (155,107)  5 %t(m_{\lambda})
%%MarkedPoint: (113, 46)  6 %m_{\lambda}=F_{\lambda}(m)
%%MarkedPoint: (174, 38)  7 %X(m_{\lambda})
%%EndMarkedPoints
```

Figure 2: Contents of the file `Fig541.bb`, containing the “marked point” information for Fig. 1.

```
file: ./Fig5.4.1.bb
Bounding Box is (0,0)->(224,134)
Marked '1' point at ( 38,118) for F_{\lambda }^*t(m_{\lambda }).
Marked '2' point at ( 77,115) for t(m).
Marked '3' point at ( 62, 72) for m.
Marked '4' point at (130,123) for integral curve of $X$.
Marked '5' point at (155,107) for t(m_{\lambda }).
Marked '6' point at (113, 46) for m_{\lambda }=F_{\lambda }(m).
Marked '7' point at (174, 38) for X(m_{\lambda }).
Found 7 data points.
```

Figure 3: Marked point information, as it appears in the `.log` file.

as the text string accompanying each marked point. For the moment, ignore the ⟨*mods*⟩ parameter; it will be explained later.

```
\xyMarkedTxt ⟨mods⟩{⟨num⟩}
\xyMarkedText ⟨mods⟩{⟨num⟩}
\xyMarkedMath ⟨mods⟩{⟨num⟩}
\xyMarkedTxtPoints ⟨mods⟩{⟨list⟩}
\xyMarkedTextPoints ⟨mods⟩{⟨list⟩}
\xyMarkedMathPoints ⟨mods⟩{⟨list⟩}
```

The first two commands are just alternative names which give identical results. These, and the third command, set the supplied text-string as a label at marked point number ⟨*num*⟩, assuming it to contain TEX code valid in text or math mode, as the name suggests. Several marked points are handled simultaneously by the remaining commands, where the ⟨*list*⟩ consists of numbers and number ranges. Note that the fourth and fifth commands are simply alternative names which give identical results. With `\xyMarkedMathPoints`, the strings in the `.bb` file are presumed to be valid math-mode source, *without* the need for surrounding `$...$` delimiters.

In Fig. 3 it can be seen that point number 4 requires text mode whereas all others are meant for math mode. One way to do this is with the following code, which yields the results in Fig. 4:

```
\WARMprocessEPS{Fig5.4.1}{eps}{bb}
\renewcommand{\labeltextmodifiers}{++!D}
\renewcommand{\labelmathmodifiers}{+!D}
\renewcommand{\labelmathstyle}{\scriptstyle}
\renewcommand{\labeltextstyle}{\footnotesize}
\begin{xy}
 \xyMarkedImport{}
 \xyMarkedMathPoints{1-3,5-}
 \xyMarkedTextPoints{4}
\end{xy}
```

Note the following points:

- The `\WARMprocessEPS` command uses its arguments to specify the graphic image and the file to read for the marked-point information;
- The expansion of `\labeltextmodifiers` yields XY-pic ⟨*modifier*⟩s that affect the way a label is positioned with respect to its marked point, when using `\xyMarkedTextPoints` and other text mode commands. For math-mode labels there is `\labelmathmodifiers`.
- `\xyMarkedImport` extends the XY-pic command `\xyimport`. Its argument can be the name of the graphics file to be placed into the TEX or

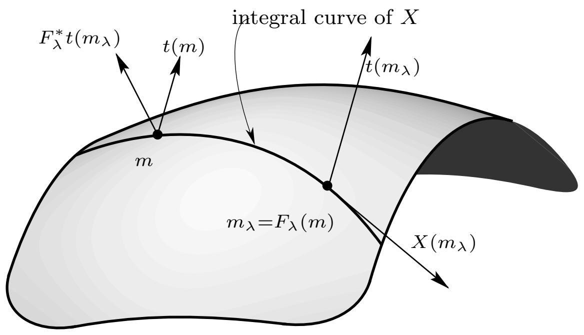


Figure 4: Imported image, with attached labels.

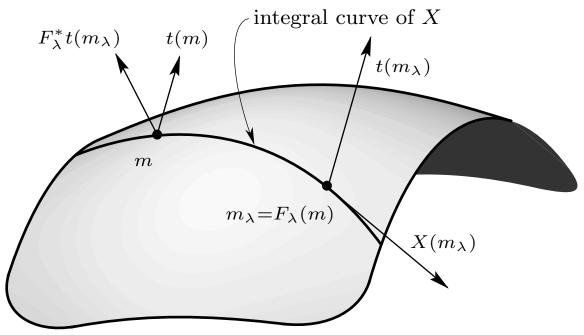


Figure 5: Imported image, with fine adjustments to the positions of labels.

LaTeX document. However, it is not required when `\WARMprocessEPS` has been used already.

- A ⟨*list*⟩ can be a comma-separated list of numbers or numeric ranges, a–b.

For extra convenience in specifying lists, the following commands are also available to put labels on all but a specified ⟨*list*⟩ of marked points:

```
\xyMarkedTxtExcept ⟨mods⟩{⟨list⟩}
\xyMarkedTextExcept ⟨mods⟩{⟨list⟩}
\xyMarkedMathExcept ⟨mods⟩{⟨list⟩}
```

The empty ⟨*list*⟩ always means to use *all* marked-points, regardless of the '`Except`'. Also, open-ended ranges such as `-3` and `5-` refer to all numbers to or from the appropriate extremity.

Commands for Styled Labels. As well as commands listed above, font size and style for text and math labels can be specified, using commands:

```
\xyMarkedStyledTxt ⟨mods⟩{⟨style⟩}{⟨num⟩}
\xyMarkedStyledText ⟨mods⟩{⟨style⟩}{⟨num⟩}
\xyMarkedStyledMath ⟨mods⟩{⟨style⟩}{⟨num⟩}
\xyMarkedStyledTxtPoints ⟨mods⟩{⟨style⟩}{⟨list⟩}
\xyMarkedStyledTextPoints⟨mods⟩{⟨style⟩}{⟨list⟩}
\xyMarkedStyledMathPoints⟨mods⟩{⟨style⟩}{⟨list⟩}
\xyMarkedStyledTxtExcept ⟨mods⟩{⟨style⟩}{⟨list⟩}
\xyMarkedStyledTextExcept⟨mods⟩{⟨style⟩}{⟨list⟩}
\xyMarkedStyledMathExcept⟨mods⟩{⟨style⟩}{⟨list⟩}
```

Allowable values for ⟨*style*⟩ in text mode are macro names that can sensibly be used with XY-pic's `\txt` command:

```
*⟨modifiers⟩\txt⟨style⟩{...balanced text...}
```

while for math mode ⟨*style*⟩ must work within in-line mathematics as follows:

```
$⟨style⟩{...balanced math...}$ .
```

Fine Adjustment of Labels. The labelled image in Fig. 4 looks quite good but there are blemishes: e.g. the text label "integral curve..." overlaps with the curved arrow, the math label "m_λ = ... " is too far from the large dot which it is meant to be labelling, and the "$t(m)$" and "$t(m_\lambda)$" are perhaps too close to the arrows they are meant to label.

The position of the text label, at marked-point number 4, could be adjusted by choosing a different set of XY-pic modifiers for the expansion of the macro `\labeltextmodifiers`. This works when there is just a single label to fine-tune but is no good when more than one needs special adjustment.

To allow many specialised adjustments, all the commands introduced so far allow XY-pic modifiers to be specified. These come immediately after the command-name, but *before* the opening brace:

```
\xyMarkedMathExcept ⟨mods⟩{⟨list⟩}
\xyMarkedStyledPoints ⟨mods⟩{⟨style⟩}{⟨list⟩}
\xyMarkedStyledTxtPoints ⟨mods⟩{⟨style⟩}{⟨list⟩}
```

The ⟨*mods*⟩ are just XY-pic ⟨*modifiers*⟩, here given a shortened name to fit the column width.

Figure 5 shows how this could be done. The source code is as follows. Note how three of the math labels are positioned with explicit XY-pic modifiers while the others use `\labelmathmodifiers`. The single text label is also positioned explicitly, to good effect, so there is no need for `\labeltextmodifiers`.

```
\WARMprocessEPS{Fig5.4.1}{eps}{bb}
\renewcommand{\labelmathmodifiers}{+!D}
\renewcommand{\labelmathstyle}{\scriptstyle}
\renewcommand{\labeltextstyle}{\footnotesize}
\begin{xy}
 \xyMarkedImport{}
 \xyMarkedMathPoints ++!D!L(.1){2}
 \xyMarkedMathPoints +!D!L(.3){5}
 \xyMarkedMathPoints ++!D{6}
 \xyMarkedMathExcept{2,4-6}
 \xyMarkedTxtPoints ++!D!L(.2){4}
\end{xy}
```

Recall the effect of the XY-pic modifiers, e.g. `+!D!L(.3)`. First, TeX sets an `\hbox` containing the typeset label. Usually this box is centered so that if there were no ⟨*modifier*⟩s the center of the label would be anchored at the marked point. The modifier `+` adds a small margin, increasing the size of the box both vertically and horizontally. Next the `!D` shifts the reference point (Down) within the box to the bottom edge; with no further modifiers, the label now appears entirely *above* the position of the marked point, with the bottom edge occuring the width of the margin away from it. Finally the `!L(.3)` nudges the reference point towards the Lefthand edge, by an amount .3 of the distance to it so that now more of the label appears on the righthand side of the marked point.

Note that nudging using `!D`, `!L`, `!R` and `!U` (Up), has the effect of shifting the label in the *opposite* direction to the specified nudge. Numerical ⟨*factor*⟩s, such as `(.3)`, are optional; if omitted, the reference point is moved all the way to the specified edge[2].

Strategies for marking points. Figure 5 shows how labels can be accurately positioned, using the locations of the marked points of Fig. 1. The marked points are away from “busy” parts of the graphic. They indicate where labels can be placed near to that part of the image being labelled yet not interfere unduly with other parts of the image.

While this is an intuitive strategy for selecting places to be marked, it can mean that adjustments, by “nudging”, are required to position the labels to best effect. Some trial-and-error is usually required before finalising the positions of all labels by choosing the best ⟨*factor*⟩s.

Marking the busy places. In many cases it is a better strategy to put marked anchor points much closer to the places to which the labels *refer*, rather than to where the labels themselves are desired. In Fig. 6 we see the same image as previously but with a different set of marked points for the same labels. For this set it is sufficient to use just a new file (`Fig541.bb2`) for the labels while retaining the same file (`Fig541.eps`) for the image itself. Indeed, that image is used 6 times in this paper, yet only one copy of the file is required.

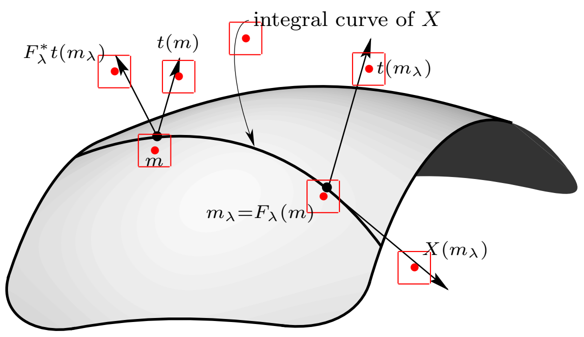


Figure 6: Attaching labels by corners and edges to places near to what they refer.

```
\WARMprocessEPS{Fig5.4.1}{eps}{bb2}
\renewcommand{\labelmathstyle}{\scriptstyle}
\renewcommand{\labeltextstyle}{\footnotesize}
\begin{xy}
 \xyMarkedImport{}
\xyShowMarkPoints{*++[red][F-:red]@{*}}{-}
 \xyMarkedMath +!DR{1}
 \xyMarkedMath +++!D{2}
 \xyMarkedMath +!U{3}
 \xyMarkedTxt +!DL{4}
 \xyMarkedMath +!L{5}
 \xyMarkedMath +!UR{6}
 \xyMarkedMath +!DL{7}
\end{xy}
```

When the marked points are chosen this way, the labels can usually be well positioned by specifying just margins and an edge or corner to be where the reference point of the label should occur. There is little need for delicate nudging and ⟨*factor*⟩s.

On the other hand, extreme accuracy is not at all necessary when choosing positions for the marked points. In this article, the `.bb` files were generated using low-resolution preview images. These need *not* be accurate scaled-down versions of the higher resolution images rendered by PostScript. Inaccuracies can be compensated for using XY-pic adjustments.

Adjusting sizes and styles. Another significant advantage of this strategy becomes apparent when the image or labels need to be resized or restyled, perhaps for use in a different context. This will almost certainly change the relative size of the labels and the image. Smaller-sized labels remain anchored to places near to what they refer. On the other hand, relatively larger labels can have been anchored so as to expand over portions of the image that are otherwise empty. In either case there may be no need to make any adjustments to the coding of labels.

```
\WARMprocessEPS{Fig5.4.1}{eps}{bb2}
\renewcommand{\labelmathstyle}{\displaystyle}
\renewcommand{\labeltextstyle}
  {\large\bfseries\sffamily}
\begin{xy}
 \xyMarkedImport{}
 ...
 ...
\end{xy}
```

[2] Refer to the XY-pic Reference Manual [6], for details of the XY-pic language for structured diagrams.

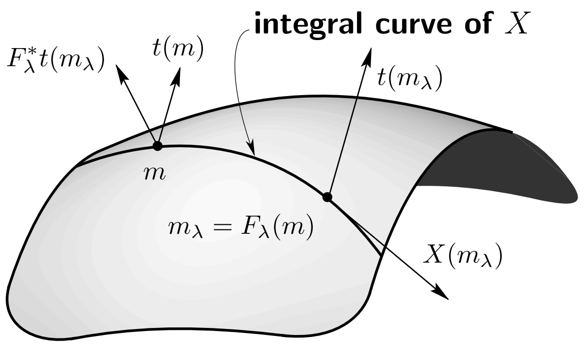


Figure 7: Labels remain well positioned with relative changes of scale.

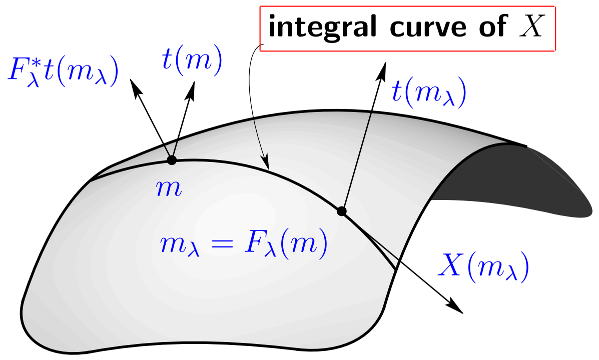


Figure 8: All labels dropped as XY-pic styled text boxes.

To LaTeX or not to LaTeX. Although the above examples have used LaTeX, the `WARMreader` macros work equally well with plain TeX, and most other formats, as does XY-pic. The only requirement is to be able to import the graphic and customise the expansion of a single macro, `\xyWARMinclude`, to suit. This macro takes as argument the name of the image file. As a practical default, it expects to be able to use the `\includegraphics` command from LaTeX's graphics package:

```
\def\xyWARMinclude#1{\includegraphics{#1}}
```

This definition can be overridden by replacing the `\includegraphics` with `\psfig` or `\epsfig` or `\epsfbox` or other command for placing an imported graphic within the TeX or LaTeX document.

There must be only one argument for the filename. The result should be an `\hbox` of exactly the size required for the image to occupy. (This is so that `\xyWARMinclude{`⟨*filename*⟩`}` can be used as the argument to an `\xyimport` command.)

Note that some macros for including graphics are *not* suitable. For example, the `\centerpicture` macro from *Textures*' `picmacs.sty` file cannot be used since it inserts stretchable 'glue' to span the whole page width; on the other hand, `\picture` from the same file can be used.

Rotations and Scaling. The requirements stated in the previous subsection allow scaling, rotating and resizing of imported graphics. For example, a rescaling can be achieved using LaTeX as follows:

```
\newcommand{\scaledfig}[2]
  {\scalebox{#1}{\includegraphics{#2}}}
\renewcommand{\xyWARMinclude}[1]
  {\scaledfig{.7}{#1}}
```

It is only the image which is resized or rescaled; the size and style of labels is controlled independently, as discussed above. When different images require different scale factors, then the definition of `\scaledfig` belongs in the document preamble and a re-definition of `\xyWARMinclude` should precede a figure, if needed. (See Fig. 13 for an example.) Optional arguments to `\includegraphics` or other command can be incorporated in a similar way.

Same locations, different labels. It is not necessary to use the text strings from the `.bb` file for the labels. Instead, the label can be specified within the TeX or LaTeX source. This is most convenient, since it means that:

- changes can be made to the labels without the need to make any adjustments to the `.eps` or `.bb` files
- the same image can be used many times with different labels
- labels may cross-reference other parts of the document; in a web document the labels could become hyperlinks, as in an "image-map"

Figure 8 uses this technique as one way to get larger sized mathematics in labels. The actual code used is shown in Fig. 9.

Using `\xyMarkedPos` allows the most flexibility amongst all the commands available for placing a label. Essentially all that it does is to move the XY-pic "current point" to the location of the marked point. Now any valid XY-pic code can be used to place anything at all at that point.

Commands to allow direct use of XY-pic code at the marked points are as follows:

```
\xyMarkedPos{⟨num⟩}⟨pos⟩*⟨object⟩
\xyShowMark{⟨pos⟩*⟨object⟩}{⟨num⟩}
\xyShowMarkPoints{⟨pos⟩*⟨object⟩}{⟨list⟩}
\xyShowMarksExcept{⟨pos⟩*⟨object⟩}{⟨list⟩}
```

In the latter three cases, if the {⟨*pos*⟩*⟨*object*⟩} is empty, then a default `\markobject` is used for each

```
\WARMprocessEPS{\exnamei}{eps}{bb2}
\renewcommand{\labeltextstyle}{\large\bfseries\sffamily}
\begin{xy}
 \xyMarkedImport{}
 \xyMarkedPos{1}*+!DR[blue]\txt\labeltextstyle{$F_{\lambda}^*t(m_{\lambda})$}
 \xyMarkedPos{2}*+++!D[blue]\txt\labeltextstyle{$t(m)$}
 \xyMarkedPos{3}*+!U[blue]\txt\labeltextstyle{$m$}
 \xyMarkedPos{4}+/u1ex/*+!DL[F-:red]\txt\labeltextstyle{integral curve of $X$}
 \xyMarkedPos{5}*+!L[blue]\txt\labeltextstyle{$t(m_{\lambda})$}
 \xyMarkedPos{6}*+!UR[blue]\txt\labeltextstyle{$m_{\lambda}=F_{\lambda}(m)$}
 \xyMarkedPos{7}*+!DL[blue]\txt\labeltextstyle{$X(m_{\lambda})$}
\end{xy}
```

Figure 9: Coding for Fig. 8 uses various XY-pic effects.

point in the ⟨*list*⟩. This is the same for the command `\xyShowAllMarkedPoints`, as was used in Fig. 1 and Fig. 6. All these commands finish with the XY-pic `\POS`-parser command so that further XY-pic drawing can be done, if desired. For a single marked point located using `\xyShowMark`, its number is also placed, using a macro `\markobjectlabel`. This expands as follows; it can be redefined if desired.

```
\def\markobjectlabel#1{\POS*\dir{x},
  *+<3pt>!U{\scriptscriptstyle#1}}
```

Symbolic names. Although all the examples so far have referred to the marked points by number, they can instead be assigned a symbolic name. Any text string suffices instead of the number within the `.bb` file. This string can be used instead of the ⟨*num*⟩ in those macros that require such an argument. Macros wanting a ⟨*list*⟩ still work since there is an internal counter as well as the symbolic name.

Format of the `.bb` files. The examples shown here have used `.bb` files in which the information is presented as in Fig. 2. This form is based on the structure of comments in PostScript files. Note how it includes a `%%BoundingBox` comment in the standard PostScript form as well as the actual marked point information.

Indeed, it is the presence of this comment that warrants the use of the extension `.bb`. In LaTeX, the `\includegraphics` command can make use of the bounding-box information contained in a file with this extension. For an EPS graphic this information could be read from the `.eps` file itself; however, since these files can be very large and can contain binary portions which TeX does not handle easily, it is often more convenient to have it extracted into a separate `.bb` file. For non-EPS graphics, all TeX requires is the bounding-box information to know how large an empty box to leave while typesetting. Having this in a separate `.bb` file is the only viable option due to the binary nature of most graphics formats. With `WARMreader`, this use of a `.bb` file has been extended to include extra marked point information.

Furthermore, with a `.eps` or other PostScript image, the contents of a `.bb` file can be pasted into the `.eps` file for easier distribution. When there is initially no `.bb` file, the `WARMreader` macros search the `.eps` file instead and a `.bb` file is created, containing the `%%BoundingBox` comment. The marked point information is included also, provided that a `%%StartMarkedPoints` comment has been encountered within the first 20 lines.

It is now apparent that the labelling strategy discussed here can be used with any graphics format provided that

- the TeX installation has a way to specify that the image file is required within the `.dvi` or other output format being produced
- a file is available, containing the size and all the marked point information, using numbered or symbolic names and (optionally) text strings

Making `.bb` files with *Zephyr*

With a Macintosh system, the easiest way to create `.bb` files for EPS graphics, and other formats, is to use David Rand's *Zephyr* [5] text and list editing program. After launching the application, a graphics file is opened by selecting the special PICT Displayer extension from the pull-down menu, as shown in Fig. 10. Choosing Display LL Coordinates prepares *Zephyr* for recording coordinate values for marked points, where the origin is at the lower-left of the image. This opens a file-dialog window, allowing the required file to be found and opened. Indeed, any file that contains a graphics preview image, in the Macintosh PICT format, can be selected from the file-dialog. It is this preview image which will be shown and used for marking points.

The Display UL Coordinates alternative can be chosen instead, to have the origin at the upper-left

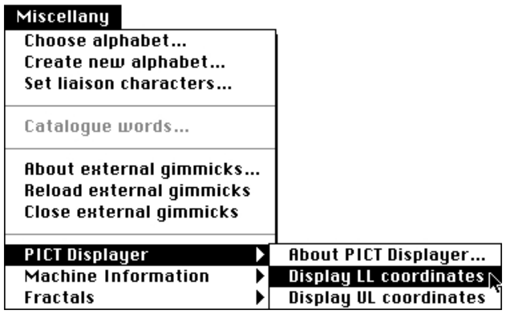
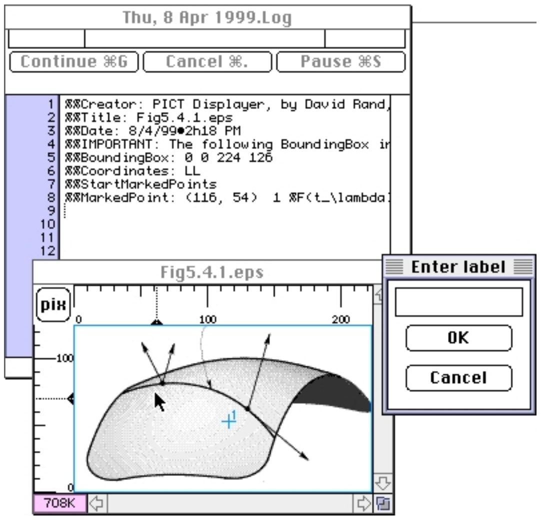


Figure 10: Opening a graphics file with the PICT Displayer in *Zephyr*.



```
%%Creator: PICT Displayer, by David Rand,
%%Title: Fig5.4.1.eps
%%Date: 8/4/99•2h18 PM
%%IMPORTANT: The following BoundingBox in
%%BoundingBox: 0 0 224 126
%%Coordinates: LL
%%StartMarkedPoints
%%MarkedPoint: (116, 54)  1 %F(t_\lambda)
```

Figure 11: Marking points for the .bb file with *Zephyr*'s PICT Displayer.

corner and with the second coordinate increasing downwards. If this is done, images in the TEX document using such coordinates should be preceded by the `\MacintoshOrigin` command.

To mark a point within the image, simply click with the mouse at the desired point. A small window will pop up, as in Fig. 11, allowing a label to be typed and the selection confirmed.

After the first point has been chosen a Log-window appears, containing bounding-box and other information, as well as data for the first marked point. A line of data is added for each subsequent point. Within the image the point is marked by a numbered cross. Guide-rules help position the cursor accurately: gradations may be inches, centimeters, or pixels. The Log-window can be seen in Fig. 11. Since it contains just plain text, the Log can be edited at any time.

When finished with an image, click its close-box (in the upper-left corner); this also adds the closing comment to the Log. Finally close the Log-window and Save As..., choosing whatever name is desired — usually ending `.bb`, though this is not compulsory.

`\MacintoshOrigin`	allow for coordinates with origin at upper-left
`\EndLineAdjust`	adjust for awkward line-end characters

End-of-line problems. Text files created on one computing platform do not always transfer to other platforms in a way that allows them to work correctly. This can happen with `.bb` files. Declaring `\EndLineAdjust` before processing the `.bb` file may alleviate a TEX error that otherwise can occur.

Annotations on *Mathematica* graphics.

The next examples have been used for teaching elementary mathematics. They were constructed using the *Mathematica* [7] software package and saved in EPS format. In fact there is more than one way to do this with *Mathematica*, which can produce `.eps` files having quite different structure and properties. The first example is in direct analogy with techniques discussed already. Extra considerations apply when the `.eps` file contains an `%%AspectRatio` comment, as in later examples.

The *Mathematica* Front-End software allows for "point-&-click" on images to obtain coordinates,[3] in the coordinate system used to calculate the image contents. This technique was used to create data files for the remaining examples; an extension `.mbb` indicates their origin.

```
\WARMprocessMMA{Q1}{eps}{mbb}
\renewcommand{\xyWARMinclude}[1]
  {\scaledfig{.7}{#1}}%
\begin{xy}
\xyMarkedImport{}
\xyMarkedPos{para}*+{}
  ,\ar@{<-}+(.7,25)*+!D\txt{base of parabola}
\xyMarkedPos{cub1}*++{}!D(.6),\ar@{<-}-(3.5,15)
 *+!U(.8)\txt{turning points\\of a cubic}="cub"
\xyMarkedPos{cub2}*++{},\ar@{<-}"cub",
\xyMarkedPos{neg1}*++{},\ar@{<-}+(.5,-25),
 *+!L(.6)\txt{local minima}="min"
\xyMarkedPos{neg2}*++{},\ar@{<-}"min",
\end{xy}
```

Click at the four edges to get the bounding-box information. Some manual editing is needed to put this into the form shown in Fig. 12. The TEX source uses the macro `\WARMprocessEPS` to read size and marked-point data. Having just a symbolic label for

[3] First click once on an image to select it, then hold down the modifier-key while clicking at the desired places within the image. When done, choose the Copy menu-item. Subsequently Paste the contents into an editable cell.

```
LDRU:{-3.89059, -43.2333, 4.21704, 43.0523}
StartData
,{1.5145, 5.44064, para}
,{2.03422, -9.67778, cub1}
,{-1.01481, 17.6091, cub2}
,{0.96013, 2.49071, neg1}
,{-1.04945, 2.49071, neg2}
EndData
```

Figure 12: Listing of `Q1.mbb`, containing the marked-point data for Fig. 13.

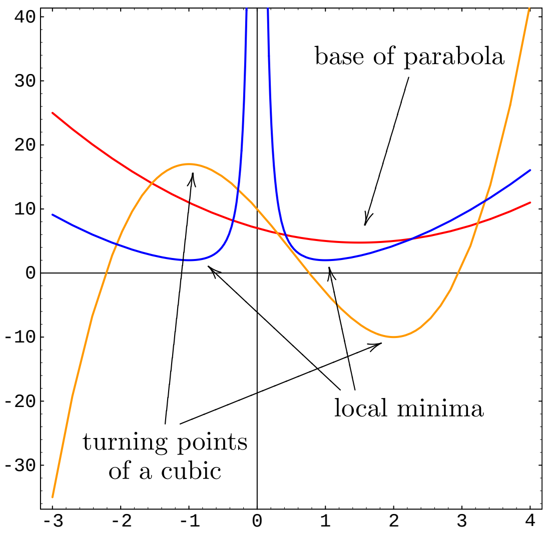


Figure 13: Labelled graphic, using *Mathematica* and `WARMreader`.

each point is quite sufficient for Fig. 13, in which the marked points are not where labels occur but are near the endpoints of arrows. Positions for the labels are determined relative to these arrow-ends, using XY-pic commands. Notice how some labels are positioned relative to one marked point, then used to draw an arrow to another.

Adjusting for Aspect Ratio. Some graphics export options in *Mathematica* result in graphics for which the bounding-box is not the same size or shape as the preview image. For instance, some have a rectangular preview but `%%BoundingBox` for a square enclosing the image.

The "aspect ratio" (i.e. height/width) of the rectangle must be known to handle such graphics correctly with `WARMreader`. This can be obtained from the `.eps` file, where it is given as a PostScript-like comment; it must be supplied as the first line in the `.mbb` file. The `\WARMprocessMMA` macro is replaced with a variant called `\WARMprocessMMAR`.

```
%%AspectRatio: .61803
%LDRU:{-2.42465, -3.80861, 6.6868, 3.25092}
```

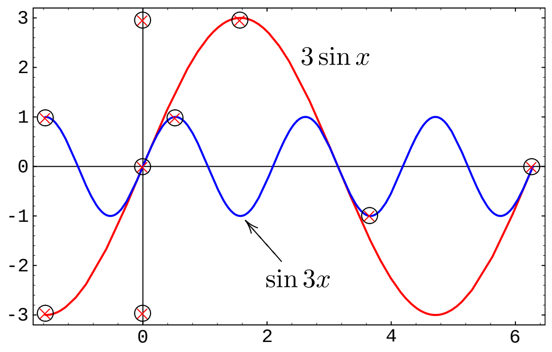


Figure 14: *Mathematica* graphic having aspect ratio $\neq 1$. Cross-hairs super-imposed at fractional multiples of π indicate accuracy of the alignment.

```
LDRU:{-2.18, -3.57, 6.50, 3.24}
StartData
,{2.29262, 2.27719, 3sinX}
,{1.54949, -0.927998, sin3X}
EndData
```

Such images sit badly in a TeX document without removing the extra space below, when the aspect ratio is greater than 1, or at left and right, when the aspect ratio is less than 1. This explains the `\vskip` commands in the following listing for Fig. 14.

```
\WARMprocessMMAR{QA1}{eps}{mbb}%
\renewcommand{\xyWARMinclude}[1]
  {\scaledfig{.7}{#1}}%
\newcommand{\Xhair}{%
 \drop[thinner][red]+[o][F-]@{x}}%
\vskip-3.5\bigskipamount
\begin{xy}
 \xyMarkedImport{}
,(0,0)\Xhair,(0,3)\Xhair,(0,-3)\Xhair
,(6.2831,0)\Xhair,(-1.5708,1)\Xhair
,(-1.5708,-3)\Xhair,(1.5708,3)\Xhair
,(.5236,1)\Xhair,(3.6652,-1)\Xhair
 \xyMarkedPos{3sinX},*++!L{3\sin x}
 \xyMarkedPos{sin3X}*+{}
  ,\ar@{<-}+(.7,-1)*+!U!L(.4){\sin 3x}
\end{xy}%
\vskip-3.5\bigskipamount
```

In most cases this is enough for good placement of labels over the imported image; fine-tuning can be done using XY-pic modifiers, as described earlier. If greater accuracy is required in establishing the coordinate system over the image, some tweaking of the bounding-box may be done inside the `.mbb` file, as in the third line of the above listing for Fig. 14. The second line, which shows the coordinates obtained from edges of the preview image, has been suppressed to allow the following line to give modified values. Note how the cross-hairs have been accurately positioned.

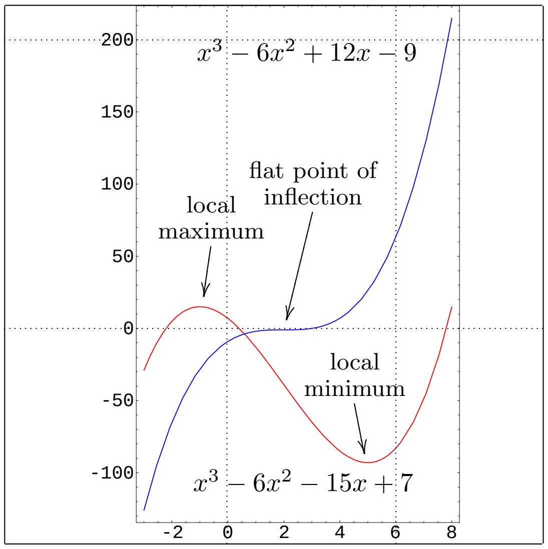


Figure 15: *Mathematica* graphic with non-trivial aspect ratio and relatively wide axis labels. The frame shows the oversized bounding-box, while dotted grid-lines indicate the accuracy of the alignment.

A further complication occurs when the graphic contains wide axis labels or tick marks. Now not all edges of the preview image need correspond to edges of the bounding box, when printed on the page. *Mathematica* rescaled the preview to include the axis labels but, on the printed page, the main part of the image is larger, with the axis labels extending into the extra space due to the aspect ratio.

To get best positioning, some visual estimation is required. An extra offset parameter is supplied with the `%%AspectRatio` comment, to measure the extent that labels would fall outside the bounding-box, if it had been rectangular, not square.

```
%%AspectRatio: 1.6 :1.294
%LDRU:{-5.24417, -171.787, 8.21328, 248.046}
LDRU:{-3.95, -148, 8.21328, 226}
StartData
,{-1.13851, 16.7698, amax}
,{4.94396, -92.2394, amin}
,{2.05478, 0.565704, bflat}
EndData
```

In the above listing of the `.mbb` file for Fig. 15, the third line gives the extents of a rectangle, with aspect ratio 1.6, that just encloses the height of the graphic. The left-hand edge of this rectangle falls roughly $1.294 = 5.24417 - 3.95$ horizontal units from the edge of the axis labels on the left.

Other formats for `.bb` data.

The `WARMreader` macros can be used to read data for marked-points from files having other formats. For a given format one needs to specify 'data-start' and 'data-end' strings, as well as patterns to be used with macros to extract the necessary components of the bounding-box and the lines of marked-point data. Stripped-down versions of these patterns are also needed, to help determine when a line does *not* match what is required. Also required is a token list, to hold the expansion part of a TEX macro to interpret the data which matches the supplied pattern. This macro must store the data appropriately for later use. Finally, there must be a TEX macro that controls the order in which the various steps are performed; i.e. reading the data file with the appropriate pattern to interpret each data line. For more specific information on what is required, consult the file `WARMreader.sty`.[4]

References

[1] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LATEX Graphics Companion.* Addison-Wesley, 1997. 1108, 1108, 1108

[2] Jerrold E Marsden, R. Abraham, and T. Ratiu. *Manifolds, Tensor Analysis, and Applications.* Springer-Verlag, 2000. (in preparation; 2nd edition 1988). 1108

[3] Ross Moore. High quality labels on included graphics, using XY-pic. *TUGboat*, 18(3):159–165, 1997. On-line version at `http://www-texdev.mpce.mq.edu.au/XyPIC/XYarticle/`. Erratum appears as [4]. 1107, 1116

[4] Ross Moore. Erratum: High quality labels on included graphics, using XY-pic, (see [3]). *TUGboat*, 19(1):61, 1998. 1116

[5] David Rand. Zephyr, a list and text editor for the Macintosh. Shareware software available on-line at `http://www.crm.umontreal.ca/~rand/Zephyr_Eng.html`, March 1999. Also in French, named Zéphyr. 1113

[6] Kristoffer H. Rose and Ross R. Moore. *XY-pic Reference Manual, version 3.7.* DIKU, University of Copenhagen, Universitetsparken 1, DK–2100 København Ø, February 1999. 1111

[7] Stephen Wolfram. Mathematica, *A System for Doing Mathematics by Computer.* Addison-Wesley, 2nd edition, 1994. Software from Wolfram Research, Inc., `http://www.wri.com/`. 1114

[4] Available from `http://www-texdev.mpce.mq.edu.au/TUG/WARM/WARMreader.sty`.