ASYMPTOTE and LaTeX: An Integration Guide*

Dario Teixeira
(darioteixeira@yahoo.com)

Version 0.95
21st March 2007

Abstract

ASYMPTOTE [1] is an exciting new language, developed by Andy Hammerlindl, John Bowman, and Tom Prince. It aims at providing a standard for the typesetting of mathematical illustrations and graphs. This document explains the various options surrounding the integration of ASYMPTOTE programmes into LaTeX documents. We begin by showing how the `asymptote.sty` package simplifies things enourmously, by allowing one to embed ASYMPTOTE code from within a LaTeX document. Nevertheless, there are a couple of caveats surrounding its compatibility with PDFLATEX and the `subfig.sty` package; we suggest workarounds for them, however. As a next step, we address the circumstances that might lead one to separate the ASYMPTOTE programme from the LaTeX document; these are mostly related with unnecessary recompilation of large ASYMPTOTE programmes. Finally, we discuss at great length the *inline* option of ASYMPTOTE: why it is necessary, and how it affects ASYMPTOTE programmes.

Contents

1. Introduction

In very much the same way that LaTeX is the *de facto* standard for typesetting scientific text, one of the goals of ASYMPTOTE was to provide a standard for mathematical illustrations [1]. It should therefore come with no surprise that ASYMPTOTE has been designed from the ground up not only to use LaTeX for its text typesetting needs, but also to integrate seemlessly with it. This document focuses on the latter aspect: it describes the various options available for the integration of ASYMPTOTE and LaTeX.

In broad terms there are two different ways of using ASYMPTOTE together with LaTeX: the most practical is to simply embed ASYMPTOTE code within your LaTeX document—the `asy` environment allows you to do just that. The other is more conventional, and involves putting your ASYMPTOTE code in a separate file, invoking the `asy` command to generate an `eps` or a `pdf`, and then inserting it into your document via the `\includegraphics` command from the `graphicx` package. Each method has its advantages and disadvantages; you should be familiar with both and choose whatever is most appropriate for each situation. Sections 2 and 3 discuss each approach at greater length.

In parallel with the decision whether to embed or to "outsource" the ASYMPTOTE code, you may also choose whether the LaTeX text in your illustration should be rendered by a child LaTeX process controlled by ASYMPTOTE (the default behaviour), or if ASYMPTOTE should surrender all control and let the main LaTeX process inherit and render the text (the so-called *inline* mode). Section 4 dives deeper into this subject, explaining the circumstances which might force you to use the non-default behaviour.

2. Embedding ASYMPTOTE code into your LaTeX document

Bundled with ASYMPTOTE comes a homonymous LaTeX package that simplifies enormously the use of ASYMPTOTE + LaTeX. It allows you to put the code of your figure inside an `asy` environment and let ASYMPTOTE take care of the rest. To make use of this functionality all you have to do is include `\usepackage{asymptote}` in the preamble of your document, as the example shown in Figure 1 illustrates. (Note that the closing statement `\end{asy}` of the `asy` environment must appear on a line of its own, without any leading spaces or trailing commands/comments. LaTeX will complain if it is not so).

Suppose your document were titled `document.tex`. The generation of the `dvi` file would be a three set process: first you would have to run `latex document`, which would produce a `document.asy` file with the asymptote code contained in all the `asy` environments you have defined in your document; then you would run `asy document` to compile the ASYMPTOTE code and generate the `eps` files with your figures; at last you would need to invoke `latex document` again to merge everything together and produce the resulting `dvi` file. (You may need additional iterations to get all references correct, particularly if you are using BIBTEX or packages for generating indices and glossaries). Figure 2 lists the sequence of commands in a makefile-like format.

2.1. ASYMPTOTE and PDFLATEX

```
\documentclass[10pt]{article}
\usepackage{graphicx}
\usepackage{asymptote}
\begin{document}
\begin{figure}
\centering
\begin{asy}
size (3cm);
draw (unitcircle);
\end{asy}
\caption{Embedded Asymptote figures are easy!}
\label{fig:embedded}
\end{figure}
\end{document}
```

Figure 1: Using the `asy` environment to embed ASYMPTOTE code into a LaTeX document. This is the most straightforward solution.

```
document.dvi: document.tex
        latex document
        asy document
        latex document
```

Figure 2: The makefile used to generate the `dvi` version of the document listed in Figure 1. Even if you are not familiar with makefiles, it should still be easy to figure out what it does: the first line simply states that the `document.dvi` file we want to generate depends on another file, `document.tex`. This means that if `document.tex` has changed (and therefore has a more recent time-stamp than `document.dvi`), the sequence of commands that follows is invoked by `make` to produce `document.dvi`. It is this sequence of commands that is particularly of interest.

2.1.1. ASYMPTOTE versions 1.14 and after

If you prefer to use PDFLATEX rather than plain LaTeX, you will be glad to know that since version 1.14, ASYMPTOTE directly supports PDFLATEX. Simply include the `pdftex` option in the `\usepackage{graphicx}` declaration, and invoke `pdflatex` rather than `latex`. Figures 3 and 4 show a sample of LaTeX code and the corresponding makefile.

2.1.2. ASYMPTOTE versions before 1.14

If for whatever reason you are stuck with a version of ASYMPTOTE prior to 1.14, you can still use it together with PDFLATEX, though the procedure is slightly more complex. Though the `asy` command itself allows the generation of `pdf` files[1] instead of the default `eps`, prior to version 1.14, the `asymptote` package only supports the embedding of `eps` files. If you use the `pdflatex` command rather than the plain `latex`, you must be well-aware that this causes problems, since PDFLATEX cannot handle `eps` files directly. Fortunately, there

[1] Or a miriad of other formats, for that matter. ASYMPTOTE supports all image formats that IMAGEMAGICK can produce. (Check the ASYMPTOTE manual for option `-f`).

```
\documentclass[10pt]{article}
\usepackage[pdftex]{graphicx}
\usepackage{asymptote}
\begin{document}
\begin{figure}
\centering
\begin{asy}
size (3cm);
draw (unitcircle);
\end{asy}
\caption{Embedded Asymptote figures are easy!}
\label{fig:embedded}
\end{figure}
\end{document}
```

Figure 3: For ASYMPTOTE versions $\geq$ 1.14, you only need to add the `pdftex` option to the `\usepackage{graphicx}` declaration if you intend to use PDFLATEX.

```
document.pdf: document.tex
        pdflatex document
        asy document
        pdflatex document
```

Figure 4: For ASYMPTOTE versions $\geq$ 1.14, you can simply call PDFLATEX directly.

is a simple way out: just include the package `epstopdf` [3] into your LaTeX document; it automatically takes care of invoking the `epstopdf` command[2] to convert on-the-fly the `eps` figures into the `pdf` format. In addition, as the documentation of `epstopdf` plainly explains, the `\write18` feature of PDFTEX must be enabled, which is achieved via the `-shell-escape` command-line option. The revised version of the programme is listed in Figure 5, with the corresponding makefile shown in Figure 6

2.2. Compatibility with the `subfig` package

The package `subfig` [4] (which supercedes `subfigure` [5]) allows you to have multiple subfigures within one figure, each with its own caption, label, etc. The code sample in Figure 7 illustrates the normal mode of using this package.

There is however one small glitch: the `subfloat` command expects a box as a parameter, and environments such as `asy` are meant to be evaluated in "outer" mode, not as boxes. As a result, you will get an error message if you were to pass an `asy` environment as a parameter to `subfloat`. Getting around this issue is a relatively simple matter however: you will need to create a new `subfloatenv` environment[3], which you should place into the preamble of your document, as listed in Figure 8.

[2] Which of course should be installed in your system. Most TeX distributions do include it by default.

[3] Thanks to Steven Douglas Cochran (the author of `subfig.sty`) for diagnosing the problem and providing this solution.

```
\documentclass[10pt]{article}
\usepackage[pdftex]{graphicx}
\usepackage{epstopdf}
\usepackage{asymptote}
\begin{document}
\begin{figure}
\centering
\begin{asy}
size (3cm);
draw (unitcircle);
\end{asy}
\caption{Embedded Asymptote figures are easy!}
\label{fig:embedded}
\end{figure}
\end{document}
```

Figure 5: For versions of ASYMPTOTE prior to 1.14, the `asy` only deals with `eps` files. Therefore you must load the `epstopdf` package if you intend to use PDFLATEX for direct `pdf` generation.

```
document.pdf: document.tex
        pdflatex -shell-escape document
        asy document
        pdflatex -shell-escape document
```

Figure 6: The makefile for PDFLATEX. Note that the `-shell-escape` option must be passed to PDFLATEX for the `epdtopdf` to function properly.

```
\documentclass[10pt]{article}
\usepackage{graphicx}
\usepackage{subfig}
\begin{document}
\begin{figure}
\subfloat[One]{\includegraphics{one}}\hfill%
\subfloat[Two]{\includegraphics{two}}
\caption{Figure with two subfigures.}
\end{figure}
\end{document}
```

Figure 7: Normal usage of the `subfig` package. Note that the `\subfloat` command expects a box as a parameter.

The usage of `subfloatenv` is depicted in the code sample listed in Figure 9. Moreover, Figure 10 shows the actual result of the code. Check the documentation of `subfig` for an overview of all the options available with the package [4].

```
\makeatletter
\newsavebox{\sfe@box}
\newenvironment{subfloatenv}[1]{%
\def\sfe@caption{#1}%
\setbox\sfe@box\hbox\bgroup\color@setgroup}%
{\color@endgroup\egroup\subfloat[\sfe@caption]%
{\usebox{\sfe@box}}}
\makeatother
```

Figure 8: Definition of a `subfloatenv` environment, which allows us to use embedded ASYMPTOTE code together with the `subfig` package.

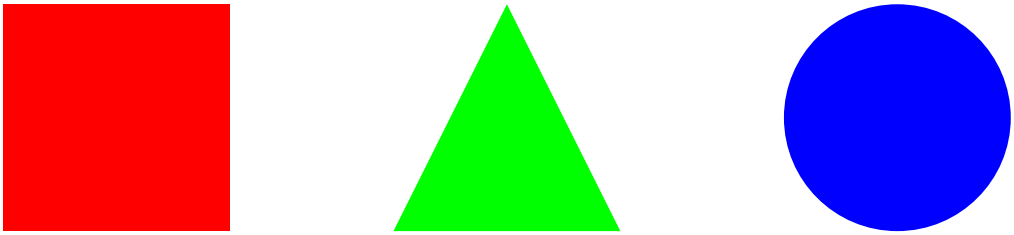
```
\begin{figure}
\centering
\begin{subfloatenv}{Square.}
\begin{asy}
size (3cm, 0);
fill (unitsquare, red);
\end{asy}
\label{fig:shapes:square}
\end{subfloatenv}\hfill%
\begin{subfloatenv}{Triangle.}
\begin{asy}
size (3cm, 0);
fill (2N--E--W--cycle, green);
\end{asy}
\label{fig:shapes:triangle}
\end{subfloatenv}\hfill%
\begin{subfloatenv}{Circle.}
\begin{asy}
size (3cm, 0);
fill (unitcircle, blue);
\end{asy}
\label{fig:shapes:circle}
\end{subfloatenv}
\caption{Three geometric shapes.}
\label{fig:shapes}
\end{figure}
```

Figure 9: Usage example of the `subfloatenv` environment.

3. Keeping your ASYMPTOTE code outside the LaTeX document

Obviously, nothing forces you to embed ASYMPTOTE code into the LaTeX document. For reasons of personal preference or organisation you may wish to put each ASYMPTOTE figure in a separate file, compile them with the `asy` command, and then include the resulting image into your LaTeX document via `\includegraphics`, just as you would do with any other external figure.

There is however one situation where the above procedure is actually preferable to



(a) Square. **(b)** Triangle. **(c)** Circle.

Figure 10: Three geometric shapes.

the easier embedding-based solution: if the ASYMPTOTE code is complex enough that it takes requires a fair amount of time and resources to compile. As of version 1.18, the `asymptote` package always regenerates the `document.asy` file after the first passage of `latex document`; this means that even if you are using a Makefile-based solution to automate the regeneration of your document, it is impossible for `make` to know whether it really is necessary to invoke `asy document`. Moreover, even if you were using a smarter Makefile system which looked at file hashes instead of timestamps, you would potentially still run into needless regeneration, since changes to *any* of the `asy` environments in your document would cause the produced `document.asy` to be different, thus forcing the regeneration of even those figures that have not changed.

4. The *inline* option

In order to understand the *inline* option, you must first comprehend the default manner by which ASYMPTOTE handles LaTeX labels. With that in mind, take a look at Figure 11, which contains a LaTeX document with embedded ASYMPTOTE code. Copy-and-paste it to your favourite text editor, save it as `document.tex` and use the steps explicited in Figure 2 to produce the various intermediate files plus the final `document.dvi`. The end result should look like Figure 12.

We are especially interested in the `document_1.eps` file which was produced after running the `asy` command. Open the file in any POSTSCRIPT viewer, and notice how it contains both the vectorial data and the accompanying text labels. By default, when the `asy` command is run it invokes a child LaTeX process to produce the labels; these are then merged with the ASYMPTOTE vectorial data to produce a self-contained `eps` file. During the second run of `latex document`, the `eps` file is included into the document with `\includegraphics`. Note that the main LaTeX process does not know (and does not care) that a child LaTeX process was used to render the labels; there is no communication whatsoever between the two.

While the default behaviour is fine for most purposes, there are a few situations where you may wish that the main LaTeX process itself also renders the labels in the ASYMPTOTE figures. Foremost, the figure may have references to labels in the main document (such as chapter or section numbers); imagine for example a diagram with the outline of your thesis, showing the dependencies between chapters. In a related note, if you use `\newcommand` to define some frequently used formulas, you may also want to use the macros inside an `asy` environment. At last, if you use custom fonts (both for text and math) in the main document, for the sake of consistency all the ASYMPTOTE figures should also use those same fonts.

```
\documentclass[10pt]{article}
\usepackage{graphicx}
\usepackage{asymptote}
\begin{document}
\begin{figure}
\centering
\begin{asy}
size (3cm, 0);
draw (N--S, Arrows);
draw (W--E, Arrows);
label ("$N$", N, N);
label ("$S$", S, S);
label ("$E$", E, E);
label ("$W$", W, W);
\end{asy}
\caption{Cardinal directions}
\label{fig:notinline}
\end{figure}
\end{document}
```

Figure 11: Without the *inline* option, ASYMPTOTE places merges the vectorial data and the text labels into the generated `eps` file. The code inside the `asy` environment has no easy access to macros defined in the main document.

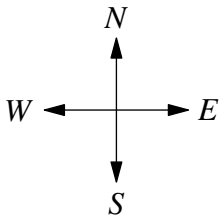


Figure 12: The result of the document in Figure 11: the four cardinal directions. The generated `eps` file contains both the vectorial data and the text labels.

While there are a few hacks that may allow you achieve the above results (the command `texpreamble` in ASYMPTOTE being quite handy for that), there is a cleaner solution: you can simply tell ASYMPTOTE to delegate the rendering of the figure labels to the main LaTeX process. This is what the *inline* option does.

If you have ever used the *"Combined EPS/LATEX"* or *"Combined PDF/LATEX"* export options in XFIG [6], or the *epslatex* terminal in GNUPLOT [7], you will feel right at home with the *inline* option. In basic terms, when the `asymptote.sty` package is loaded with this option, the `eps` file produced by ASYMPTOTE will only contain the vectorial data. The labels are placed in a separate `tex`, which essentially contains the code to include the `eps` file and to overlay the labels on top of it.

Figure 13 illustrates the use of the *inline* option. Note in particular how the ASYMPTOTE figure can now easily reference new commands defined in the main document. Moreover, if you were to change the global font settings, the ASYMPTOTE figures would also be rendered in the new settings.

```
\documentclass[10pt]{article}
\usepackage{graphicx}
\usepackage[inline]{asymptote}
\begin{document}
\newcommand{\north}{$N$}
\newcommand{\south}{$S$}
\newcommand{\east}{$E$}
\newcommand{\west}{$W$}
\begin{figure}
\centering
\begin{asy}
size (3cm, 0);
draw (N--S, Arrows);
draw (W--E, Arrows);
label ("\north", N, N);
label ("\south", S, S);
label ("\east", E, E);
label ("\west", W, W);
\end{asy}
\caption{Cardinal directions}
\label{fig:inline}
\end{figure}
\end{document}
```

Figure 13: With the *inline* option, the ASYMPTOTE code can reference commands defined in the main document.

4.1. The label size conundrum

As you may know, one of the features of ASYMPTOTE is the use of the simplex method to solve size constraints between fixed-size objects (such as labels) and variable-sized ones. However, by delegating the rendering of text labels to the main LaTeX process, using the *inline* option means that ASYMPTOTE will be unable to know the exact size of the text labels. In fact, when using *inline*, the labels are assumed to be of size 0, which can lead to the incorrect placement or scaling of objects. The document in Figure 14 (the result of the code is shown in Figure 15) illustrates the problem; since ASYMPTOTE cannot know the true size of the label, the produced bounding box is obviously wrong.

Fortunately, ASYMPTOTE provides a workaround for this issue. Bear in mind that there is no easy way to automatically solve the problem: you will always have to give ASYMPTOTE some help in determining the true size of the labels it cannot access. The implemented solution relies on a second optional parameter to *Label*, which tells ASYMPTOTE the estimated size of the first (obligatory) label. Note that this second parameter is never actually rendered, being used only as an indication. The document in Figure 16 illustrates how this

```
\documentclass[10pt]{article}
\usepackage{graphicx}
\usepackage[inline]{asymptote}
\begin{document}
\newcommand{\welcome}{Welcome!}
\begin{figure}
\centering
\begin{asy}
size (3cm, 0);
label (Label ("\welcome"), (0,0));
shipout(bbox(0.5cm));
\end{asy}
\caption{Wrong bounding box.}
\end{figure}
\end{document}
```

Figure 14: This document will produce a wrong bounding box, since ASYMPTOTE has no way of knowing the actual size of the externally defined label.

Figure 15: Wrong bounding box.

option is used, and the resulting image is shown in Figure 17. Note how the bounding box is now correct.

```
\documentclass[10pt]{article}
\usepackage{graphicx}
\usepackage[inline]{asymptote}
\begin{document}
\newcommand{\welcome}{Welcome!}
\begin{figure}
\centering
\begin{asy}
size (3cm, 0);
label (Label ("\welcome", "Welcome!"), (0,0));
shipout(bbox(0.5cm));
\end{asy}
\caption{Correct bounding box.}
\end{figure}
\end{document}
```

Figure 16: The optional second string parameter of *Label* enables us to specify the estimated size of the first parameter, thus allowing ASYMPTOTE to calculate the correct bounding box.

Figure 17: Correct bounding box.

4.2. External ASYMPTOTE code and *inline*

The example in Figure 13 assumes that the user will want to embed the ASYMPTOTE code inside the LaTeX document. Nevertheless, the *inline* option can also be used in combination with "outsourced" ASYMPTOTE code. The procedure is not quite as straightforward, though, as we will proceed to explain.

The rationale for outsourcing is of course the same described in Section 3. The basic principle is also identical, though instead of incorporating the external `eps` figure with `\includegraphics`, we must now use `\input` to include the `tex` portion of the figure (ASYMPTOTE appends an underscore character to the name of the input file; if the figure was called `cardinal.asy`, then the `tex` portion will be named `cardinal_.tex`). In addition, when invoking the `asy` command we must pass it the `-inlinetex` option, which instructs it to generate separate vectorial and label information, and to place them in an `eps` and `tex` files, respectively.

At last, ASYMPTOTE versions prior to 1.14 rely on the PSTRICKS [8] package, so if you are using one of these older versions you must also include `\usepackage{pstricks}` in the preamble of your main document. For newer versions (1.14 and later) you must include either the `color` or `xcolor` packages. Figures 18 and 19 illustrate the procedure for older and newer versions of ASYMPTOTE, respectively.

```
\documentclass[10pt]{article}
\usepackage{pstricks}
\usepackage{graphicx}
\begin{document}
\begin{figure}
\centering
\input{cardinal_}
\caption{Cardinal directions}
\label{fig:sepinline}
\end{figure}
\end{document}
```

Figure 18: The `-inlinetex` option of the `asy` enables the generation of separate `eps` and `tex` files. The latter can be included into the main document via `\input`. This code sample assumes you are using an older version of ASYMPTOTE (before 1.14), and therefore it also includes the `pstricks` package.

```
\documentclass[10pt]{article}
\usepackage{xcolor}
\usepackage{graphicx}
\begin{document}
\begin{figure}
\centering
\input{cardinal_}
\caption{Cardinal directions}
\label{fig:sepinline}
\end{figure}
\end{document}
```

Figure 19: The `-inlinetex` option of the `asy` enables the generation of separate `eps` and `tex` files. The latter can be included into the main document via `\input`. This code sample assumes you are using a newer (1.14 and above) version of ASYMPTOTE, and therefore it includes the `xcolor` package (package `color` also works) instead of `pstricks`.

References

[1] The ASYMPTOTE homepage: http://asymptote.sourceforge.net/

[2] The IMAGEMAGICK homepage: http://www.imagemagick.org/

[3] Information about `epstopdf` on CTAN:
http://www.ctan.org/tex-archive/macros/latex/contrib/oberdiek/

[4] Information about `subfig` on CTAN:
http://www.ctan.org/tex-archive/macros/latex/contrib/subfig/

[5] Information about `subfigure` on CTAN:
http://www.ctan.org/tex-archive/obsolete/macros/latex/contrib/subfigure/

[6] The XFIG homepage: http://www.xfig.org/

[7] The GNUPLOT homepage: http://www.gnuplot.info/

[8] The PSTRICKS homepage: http://tug.org/PSTricks/main.cgi