The Testflow User's Guide

Michael Shell[*][†]

January 10, 2009, Version 1.1

Contents

---

[*] See http://www.michaelshell.org/ for contact information.

[†] Manuscript created on August 13, 2002, revised January 10, 2009. The opinions expressed here are entirely that of the author. No warranty is expressed or implied. User assumes all risk.

***** Legal Notice *****

Testflow.tex is based on Rainer Schoepf's testpage.tex, the standard LaTeX test page that is included with most LaTeX systems.

1 Introduction

THIS is the documentation and user guide for testflow.tex which is designed to test the "print work flow" of LaTeX systems. Originally constructed for users of the IEEEtran.cls file, the information contained here has broad applicability to anyone wishing to obtain correct PS/PDF output on LaTeX systems. IEEEtran users often have more difficulty in this regard than the average LaTeX user as IEEE documents use Times Roman for the main text and Computer Modern for mathematics — a combination notorious for causing trouble with some older dvips configurations.

IEEEtran users should take note that the IEEE may delay (or refuse to accept) submitted PostScript or PDF documents that fail to properly use and embed the Type 1 fonts as tested by testflow.

Please note that specific recommendations are subject of frequent change, so that their timeliness cannot be guaranteed. Nevertheless, this guide might be helpful even if not always quite up to date.

1.1 Changelog:

V1.0	2002/04/15 Initial release
V1.0a	2002/08/13 Some clarifications and additions to the documentation
V1.1	2007/01/10 Update to cover teTeX 2.0/3.0 and MiKTeX 2.4 systems, new Palladio hinting and duplex tests.

2 Overview

Getting correct PostScript (PS) or Portable Document Format (PDF) output ("post processing") from LaTeX systems can be surprisingly difficult. A number of factors have conspired to create this situation.

First of all, TeX, the underlying engine of LaTeX, is designed to do typesetting only (arrange boxes of objects) and not to handle the description of those objects (such as the mathematics that governs the shape of each glyph/letter) that are to be typeset (pdfTeX not withstanding). It is a stunning revelation to many LaTeX users to realize that TeX, being able to create the most beautiful documents, is totally blind! This design choice gives TeX much immunity to changes in the rapidly evolving computer font/printing world.

However, there is a price to be paid for this benefit. Because, in TEX, the act of typesetting a document (running TEX) and the act of describing what the typeset objects (glyphs, images, etc.) look like (making PS/PDF output), are two different things (pdfTEX notwithstanding), it is possible to do one perfectly (run LATEX) and yet fail miserably in the other task (make printable output).

It is not enough to ensure that the .tex file is correct. A user must also ensure that the applications that generate the printable files from LATEX output are correctly configured for the current state of printing technology.

Secondly, a rather large number of "interaction" bugs have surfaced as changing PS/PDF writers, readers and fonts interact with each other in factorially increasing ways. Compounding this problem even further is the fact that software vendors are sometimes slow to correct flaws in their products that affect only free software like LATEX.

Finally, most books on LATEX do not provide adequate information about making printable files from LATEX.

It should be noted that this author's LATEX system is from the teTEX Unix (Linux) based distribution. A lot of the information contained here is from that perspective. However, most of this information is broadly applicable to other platforms such as MS Windows (MiKTEX, etc.). Option names provided by the graphical interfaces to the PS/PDF utilities almost always correspond closely to those provided by the basic Unix command prompt interface.

Most lacking is information related to the commercial applications such as YandY, VTEX, PCTEX, and Adobe Acrobat as the cost of obtaining various versions of each is prohibitive. Furthermore, purchasers of these systems expect, and are entitled to, some degree of support from the software manufacturer. So, technical support information like this document tends to be more vital to the users of the free systems.

Additional sources of information related to PS/PDF generation under LATEX include:

- Markus Kuhn's "Effective Scientific Electronic Publishing":
  http://www.cl.cam.ac.uk/~mgk25/publ-tips.html
- Ki-Joo Kim's "How to Create PDF from LATEX":
  http://www.geocities.ws/kijoo2000/
- Robin Fairbairns' "TEX FAQ":
  http://www.tex.ac.uk/FAQ-acrobat.html

3 Terminology

Some definitions are in order for those users who are not familiar with the terminology used here.

<texmf>: This refers to the root directory of a TEX installation. For example, for many Unix users this often is `/usr/share/texmf`, `/usr/local/teTeX/texmf` or `/usr/local/teTeX/share/texmf`. For MiKTEX, it often is: `C:\texmf`.

PostScript (PS): A Turing complete programming language, developed by Adobe Inc., which is designed for the mathematical description of the shapes of lines, curves, and other elements of letters, figures and images. PostScript interpreters, which can run PostScript code, are often found in high-end printers. Within such printers, the act of running the PostScript program produces a bitmap of pixels (this process is called rasterization) that represent the pixelized image that is to be formed (with ink or toner) on the printed page.

Portable Document Format (PDF): Also developed by Adobe Inc., PDF is a kind of subset of PostScript. PDF is not a programming language as it lacks PostScript's conditional branching ability. PDF does have very good support for indexes and hyperlinks. It is designed to be a document archiving and sharing format.

[**Side note:** PostScript and PDF are important, even if the target printer does not understand PostScript, because these languages are the primary means by which portable, precisely typeset, print documents are described today. When using non-PostScript printers, the application that is used to print the PS/PDF file does, possibly with help from another application or driver, the conversion from PS/PDF to the native printer tongue. With some recent modern "brain-dead" printers, the host CPU even does the entire rasterization for the printer.]

Font substitution: The process of using a different font, usually for purposes of viewing or printing, than that which a document originally used. Font substitution can occur when the font is not available, or when its use is otherwise not possible or discouraged.

Font embedding: The process of placing the mathematical description of the glyphs that make up a font into a file that uses these glyphs. This allows the file to be viewed/printed on a system that would otherwise not have access to the needed fonts. The main disadvantage is an increase in the size of the file.

Font subsetting: The process of embedding only those glyphs that are actually used in a font, rather than all of the glyphs in that font. Files that use subsetting can be smaller than those in which the full font is embedded. Subsetted fonts are given new, unique names within the document in which they are subsetted. Many commercial font licenses require font subsetting for any document containing the commercial font in order to reduce the possibility of font piracy.

Maximum subset percentage: For PDF making applications, the percentage of glyphs in a font that must be used beyond which the entire font is embedded instead of just a subset. For example, if the maximum subset percentage is 75%, and 77% of the glyphs in a font are used in a document file, the entire font may be embedded rather than just a subset. Thus, to always ensure subsetting, the maximum subset percentage should be set to 100% (or to 99% if that is the maximum value allowed).

Adobe Acrobat: This is Adobe's application family for creating, viewing and printing PDF files. Adobe Acrobat Distiller, available only at cost, is used to make PDF from PostScript. Those using the full version of Adobe Acrobat should use Adobe Distiller, not Adobe Pdfwriter, to make high quality PDF.

Adobe Acrobat Reader: Adobe Acrobat Reader is a free utility from Adobe (see: https://get.adobe.com/reader/) which can be used to view and print PDF files. Because Acrobat Reader is ubiquitous, it is important that all PS/PDF LaTeX output that is intended to be shared with others should work with it. Furthermore, because Acrobat Reader can be used to show useful font information about the PDF files it renders, it is useful as a diagnostic tool.

Ghostscript: Ghostscript is a freely available (the most recently released versions are subject to a few months of commercial use restrictions) PostScript interpreter and translator (see http://www.ghostscript.com/). Ghostscript is the underlying engine behind many applications, especially those in the Unix world, that can view PostScript/PDF. Ghostscript can also convert PostScript to other formats, such as PDF or to various printer languages. It can be thought of as a free alternative to Adobe's Acrobat Distiller family.

Type 1 font: In the context of this document, a font in which the glyphs are described in vector form in the PostScript language. This type of font is desirable because the glyphs will be rendered by the printer as precisely as is possible for the given printer resolution. Because type 1 fonts are rasterized only when actually viewed or printed, storing them within a document requires a constant amount of space, regardless of the resolution they ultimately will be rendered at (within the printer or on-screen viewer).

Type 3 font: In the context of this document, a font in which the glyphs are described in bitmap form. This type of font is undesirable because the glyphs will be rendered a bitmaps that often have inferior resolution and quality to that attainable by the printer. The result can be "jaggedly" looking text. The space required to store type 3 fonts within a document increases with their resolution. So, very high resolutions may not be practical for type 3 fonts.

Base PostScript fonts (also known as the "base 13" or "base 14" fonts): The fonts that are guaranteed to be resident in every PostScript (and PDF) interpreter/viewer/printer. These include Times Roman, Helvetica, Courier, with their bold, italic and italic+bold variants, and Symbol. (In LaTeX, using the PostScript fonts via times.sty: Times Roman = `\rmfamily`; Helvetica = `\sffamily`; and Courier = `\ttfamily`.) In addition to these 13, PDF readers also must also contain the ZapfDingbats font for a total of 14. Generally speaking, the base 14 fonts are not required to be embedded within a PDF file. Newer Postscript interpreters support even more fonts ("base 35"). Special thanks to Henry Churchyard for posting a clear description of this state of affairs in the newsgroup comp.fonts on April 8, 2002 in the thread "Is it base 13 or is it base 14?".

Nimbus fonts: It may come as a surprise to many people, but the base PostScript fonts are copyrighted and cannot be used for the purposes of document creation without buying a license to use them from Adobe. This license fee is also buried within the cost of every "Genuine Adobe PostScript" printer (or cartridge or SIMM or Distiller program) that is sold. Fortunately, in the early 1990's the URW font company generously donated their professional quality type 1 clones of the base 14 fonts to the X Consortium. These "Nimbus" fonts can now be used without having to pay royalty fees and are the base PostScript fonts that are distributed with the free versions of LaTeX and Ghostscript. The differences between the "real" Adobe base PostScript fonts and the Nimbus fonts are, to the vast majority of LaTeX users, insignificant.

Computer Modern (CM) fonts: The famous Dr. Donald E. Knuth, TeX's creator, also created a set of fonts to go with it. These Computer Modern fonts are very extensive in that they contain a comprehensive set of mathematical symbols. The Computer Modern fonts that Knuth released were created in the METAFONT language, which was also developed by Knuth. Unfortunately, METAFONT never became widely available in printers (like PostScript did). For these reasons, the rasterization of the Computer Modern fonts (conversion from METAFONT to type 3 bitmap) is often the default behavior of older (pre-teTeX 2.0) LaTeX systems when generating PostScript using dvips. When LaTeX systems use such type 3 fonts, the resulting PostScript (or PDF) file will not always yield the best quality possible on every printer because the rasterization process can be optimal only for a given printer resolution.

BlueSky fonts: In 1997, Blue Sky Research Inc., in conjunction with the American Mathematical Society (AMS) and YandY Inc., released its professional quality type 1 (PostScript) versions of the Computer Modern fonts into the public domain. The BlueSky fonts have since been widely distributed with the free LaTeX systems. However, some older TeX systems are not configured to use the BlueSky fonts by default. There is also another freely available set of type 1 Computer Modern fonts that are known as the BaKoMa fonts. The BaKoMa fonts are generally considered to be inferior in quality to the BlueSky fonts, so the latter is usually preferred.

CM-Super fonts: In 2001, Vladimir Volovich released the (free) CM-Super font package. The CM-Super fonts are an extremely complete type 1 font set that is a superset of the BlueSky fonts. The CM-Super package contains fonts and glyphs that were not previously available in Computer Modern (such as bold smallcaps). The CM-Super fonts can be obtained from CTAN at http://www.ctan.org/pkg/cm-super. The CM-Super fonts are required to get type 1 versions of the Computer Modern fonts when using the T1 encoding to gain access to the extended set of available glyphs:

```
\usepackage{type1ec}
\usepackage[T1]{fontenc}
```

4 Using the Testflow Files

The testflow suite consists of five main files in addition to this documentation.

`testflow.tex` is to be compiled on the user's system after which a `testflow.ps` and/or `testflow.pdf` output is to be produced. When being compiled, testflow will prompt the user for the desired paper size. **Note that this requires the use of a command shell to permit user data entry.** US letter users should answer with "letterpaper" and A4 paper users should respond with "a4paper". The user will then be asked if testflow is to also to generate a second page to test the duplex (two sided) alignment of duplex printer. Most users will want to answer "no" here. For users who do not know how to invoke LaTeX via a command shell (for shame!), the `testflow.tex` file contains commented-out commands at the beginning of the code that can be uncommented/activated as desired. Doing so will prevent testflow.tex from having to prompt the user for these settings. For example,

```
% *** paper size (choose one) ***
%\def\papertype{letterpaper}
\def\papertype{a4paper}
%
% *** duplex page test (choose one) ***
%\def\makeduplexpage{yes}
\def\makeduplexpage{no}
```

will result in a testflow.tex file that will use A4 paper without a duplex test page, without prompting the user for these options.

`testflow_ctl_LTR.ps`, `testflow_ctl_LTR.pdf`, `testflow_ctl_A4.ps`, and `testflow_ctl_A4.pdf` are "control" PostScript and PDF files, in US letter and A4 paper sizes, from a (hopefully) known good source. Because PDF is the primary means by which print documents are shared, the user will probably be interested only in the PDF versions. However, the PostScript versions are provided as a way to determine if the problem is occuring when creating PostScript or when creating PDF. For instance, if a (dvips) user is having a problem making a good PDF from `testflow.tex`, but can make a good testflow.pdf from `testflow_ctl_LTR.ps`, then chances are the problem is with their dvips or fonts (that got embedded in the PostScript file) and not the application that is making PDF from PostScript. The PostScript versions use level 1 PostScript and so should be compatible with all PostScript interpreters. The control PDF files use PDF level 1.4 and should work properly with Acrobat Reader versions 4.0.5 or later.

The idea is to have the user try to compare their output to the control files in order to determine if there is a problem and, if so, where it is occurring.

The testflow test suite cannot catch every possible type of error, but it is designed to detect the problems LaTeX users are most likely to encounter.

Because many PS/PDF generation bugs show up only when printing. it is not enough to verify that the files "look ok" on screen. The on-paper appearance of the control files should be carefully compared to that created and printed on the user's system from testflow.tex. Be sure that the application that is used to print the PDF file (e.g., Acrobat Reader) does not scale the page (i.e., ensure that any "fit to page" or "shrink/expand" printing options are not selected).

testflow.tex in no way depends on IEEEtran.cls or any other LaTeX class or package other than the basic article.cls. Problems with testflow will be caused by configuration problems with the user's LaTeX setup — not problems with class files.

Be forewarned that `testflow.tex` is designed to trigger many known bugs in PS/PDF generator applications.

5 Interpreting the Results

There are several things that testflow is designed to check. Carefully go through each of the items below to ensure that they are correct for the `testflow.pdf` that was created on the LaTeX system to be tested. The testflow PS/PDF control files serve as "correct" reference examples.

It is assumed that Acrobat Reader is what is used to view/print the PDF files.

In short, the following applications should be of versions at least as recent as:

Acrobat Reader: 4.0.5 (2000/01)

dvips: 5.86f

Ghostscript: 8.15 (7.05 and later may be acceptable and can pass testflow if correctly configured, but 8.15 and later is strongly recommended.)

URW (Ghostscript) Fonts: Unix systems should use 1.0.7pre41 (5/2005) or later versions. For MS Windows systems, the fonts included in the 8.xx series of Ghostscript are acceptable. See Section 5.4 for details.

pdfTeX: 1.20a

Computer Modern Lasy Fonts: 2001/06/06

5.1 Margins and Indicated Paper Size

Testflow provides both metric (mm) and imperial (0.1in) rulers to indicate the distance from the edges of the main text area to the edges of the paper. Testflow should have 1in (25.4mm) margins on all four edges. Note that most printers cannot print within less than 0.3in (8mm) to the edges of the paper, so the rulers will usually not be visible all the way to the edges of a printed page.

Not acceptable: the four margins are not equal on screen or on paper. Printers do not have perfect mechanical accuracy, so offsets of 0.25in (6mm) or so may be acceptable in print. However, gross errors here should be investigated and corrected.

Not acceptable: when viewing the pdf file, Adobe Acrobat reader indicates a paper size that is not correct. It should be 8.5in×11in for US letter, and about 8.26in×11.69in (210mm×297mm) for A4 pages.

By far the most common cause of incorrect margins and/or incorrect indicated paper sizes is that pdfTEX and/or dvips and/or ghostscript (ps2pdf) have been configured for a default paper size that is different from that used in the LATEX document.

PdfTEX/PdfLATEX users should see Section 6.4.2 for information on configuring pdfTEX.

dvips can be told to use a certain paper type via the command line:

```
dvips -t lettersize -o myfile.ps myfile
```

or

```
dvips -t A4size -o myfile.ps myfile
```

However, if the default paper size is not what is normally used, it is better to modify dvips configuration files. This is especially important to US letter paper users as dvips is typically set up to default to A4 paper. See Section 6.2.2 for more information on configuring dvips.

Ghostscript (ps2pdf) also has a command line option to set the paper size:

```
ps2pdf -sPAPERSIZE=a4 myfile.ps myfile.pdf
```

Ghostscript users should see Section 6.3.2 for information on configuring and using Ghostscript. Advice for Adobe Distiller is similar to that for Ghostscript.

[**Side note:** HP Laserjet II/III users should take note that a design flaw in the Canon SX engine can cause these venerable older printers to shift text (downward) on pages that quickly follow a previous page. The problem, which affects SX based printers with "high-mileage," is caused by the magnetization of the paper control solenoid levers coupled with the degradation of the solenoid stop pad material.]

5.2 Line Widths

All of the tick marks on all of the rulers have identical line thickness. Some tick marks should not appear to be thicker than others. This problem may be visible in the Acrobat Reader display, especially under normal magnifications such as 125%, as well as in its print (most obviously in the small tick marks on the metric rulers).

Acceptable, but undesirable: some of the tick marks are thicker than others.

This problem is the result of the way the PostScript algorithm handles the rasterization of lines (and other constructs). The PostScript rasterization policy is that, if *any* part of a pixel is touched by a line, then that pixel is turned on. So, depending upon where a line falls, its thickness can vary by the width of a printer's pixel. Thanks to Michael Fryd for his excellent post on this topic in the newsgroup comp.lang.postscript on August 16, 1990 in the thread "Thin line woes".

This is why dvips features a "snap grid" resolution parameter (e.g., "`-D`") that "snaps" image objects to the pixel grid of the user's printer. The problem with this approach is that one has to know the printer's resolution in advance, and, therefore, file portability is slightly compromised.

For instance, to make good PostScript for a 300dpi printer, one could do:

```
dvips -D300 -o myfile.ps myfile
```

However, for PDF converters, and printers with other than 300dpi resolution, the resulting PostScript file will be less than optimal.

Note that dvips also uses the `-D` parameter to set the resolution at which type 3 fonts are rendered. The snap grid (positioning) mechanism and the type 3 font rasterization mechanism cannot be controlled independently in current versions of dvips. Therefore, if a document is using any type 3 fonts, `-D` will have to be kept to a reasonable level (say <= 1200) to prevent the file size from becoming unacceptably large (or to prevent errors from occurring because the rasterizing applications could not handle such a high resolution for type 3 fonts). This is another reason to try to avoid the use of type 3 fonts.

This guide assumes that the user wishes to ultimately make PDF (using only type 1 fonts) and use that as a portable format to distribute documents (after all that's why they call it Portable Document Format). So, to make a PostScript file that is more suitable for PDF conversion, dvips should be told to use a very fine grid:

```
dvips -D7200 -o myfile.ps myfile
```

The testflow PostScript control files take this latter approach, and so they may have less than perfectly consistent line thicknesses when printed to printers with lower resolutions (especially below 600dpi). However, the testflow control PDF files should print very well on any decent printer.

The line thickness anomalies are also aggravated by the fact that dvips treats lines as filled rectangles instead of drawn lines. Fortunately, Timothy Van Zandt developed an alternative dvips line drawing algorithm and made it publicly available.

See Section 6.2.2 for information on how to configure dvips to obtain Postscript for making PDF with consistent line thicknesses

5.3 Fonts

Because Adobe Acrobat Reader can be used to display font information for PDF files, it is useful as a diagnostic tool. View the `testflow.pdf` under Acrobat Reader. Enlarge the window size and/or zoom out so that the entire page is visible at once. This step is necessary as Acrobat Reader may need to see the entire page before full font information is available. Now, select `File->Document Info->Fonts->List All Fonts`, or for more recent versions of Acrobat Reader, `File->Document Properties->Fonts`

Alternatively, Derek Noonburg's free xpdf viewer now comes with tools for obtaining information about PDF files, including fonts: http://www.foolabs.com/xpdf/ On systems with recent xpdf installations, one can easily do:

```
pdffonts testflow.pdf
```

to obtain comprehensive font information. Although xpdf is currently only available for Unix systems, its associated tools (including pdffonts) is also available for MS Windows platforms (in the binary releases section of the xpdf download page). In my opinion, pdffonts is a better tool than Acrobat Reader for obtaining font information. Thanks to Mukesh Agrawal for telling me about pdffonts.

Every font should be listed as a Type 1 or Type 1C. Any type 3 fonts that are listed will likely be rendered as a bitmap. The result will be "jaggedly" looking text. Type 1 fonts are typically in vector form, which means that they will always be properly rendered at their sharpest for every printer.

Every font, with the possible exception of those in the Times and Courier (which may be named as "Nimbus") families, should be listed as "Embedded" preferably "Embedded Subset".

Not acceptable: some fonts are listed as type 3.

This problem is most often caused by the fact that dvips has not been configured to use the Type 1 versions of the Computer Modern fonts (also known as the BlueSky fonts). Most LaTeX systems already have these fonts installed, so it is usually just a matter of telling dvips to use them. See Section 6.2.2 for details.

Not acceptable: fonts, other than those in the base PostScript families, (Times and Courier for testflow) are not listed as being embedded. Many, if not most, publishers require that all fonts be embedded and subsetted.

Fonts, with the possible exception of the base PostScript fonts, must be embedded or else these fonts may not be viewable in the PDF file on other platforms. See Section 6 for application configuration information. The advice for Adobe Distiller users is similar to that of Ghostscript. Some additional discussion on this topic can also be found in Section 5.9.

Science Magazine has a nice web page that illustrates the difference between Type 1 and Type 3 fonts: http://www.sciencemag.org/feature/contribinfo/prep/TeX_help/tex2pdf.dtl

Note: This is not to suggest that **every** document the user will ever generate must always contain only type 1 fonts. Some fonts are not yet available in type 1. One example is the EC fonts (which are a superset of the Computer Modern) that are currently installed on most LaTeX systems. On these systems, using T1 encoding:

```
\usepackage[T1]{fontenc}
```

will result in the use of type 3 fonts. One workaround is to also call the ae.sty package (http://www.ctan.org/pkg/ae):

```
\usepackage{ae}
```

A better approach is to install Vladimir Volovich's "CM super fonts" package, which provides type 1 versions of the EC fonts and which can be obtained at http://www.ctan.org/pkg/cm-super. Another possible way type 3 fonts can appear in a document is from EPS figures that were created by an application that uses type 3 fonts.

However, the user's LaTeX system should be able to create a `testflow.pdf` that uses only Type 1 fonts. IEEEtran.cls users should take note that the IEEE may delay acceptance of submitted work that uses type 3 fonts, especially for the main text and mathematics (for which they use the same fonts that the testflow page uses). **IEEEXplore usually requires that all fonts be Type 1, embedded and subsetted.**

5.4 Palladio Font Hinting Test

Not acceptable: Bold Palladio text looks ragged as if the baseline is not straight when viewed on-screen under light magnification (e.g., 125%). The problem does not appear under high magnifications or when printed.

The problem occurs because of a mistake in the hinting of the URW (Ghostscript) fonts which were released for Unix systems, including the most recent (as of 1/2007) Ghostscript font package `ghostscript-fonts-std-8.11.tar.gz`. It is not known to affect Ghostscript for MS Windows. The solution is to upgrade to the latest URW fonts from http://sourceforge.net/projects/gs-fonts/. Installing the fonts is easy, simply unpack the package into the exising Ghostscript fonts directory (`/usr/share/ghostscript/fonts` or on newer systems `/usr/share/fonts/default/Type1`).

5.5 The Ligature Test

Two sentences are given. Both should read "The office was affected by the five flawed mufflers." The first sentence uses ligatures for the ffi, ff, fi, fl, and ffl letter sequences and so will be slightly shorter than the one below it which does not use ligatures.

Not acceptable: British pound (sterling) symbols or other strange characters appear in the sentence that uses ligatures.

This is caused by a dvips setup that enables the "`G1`" option. This dvips feature was designed to work around a bug in the obsolete Acrobat Reader 3.0 series. Either force dvips on the command line to use "`G0`":

```
dvips -G0 -o myfile.ps myfile
```

or, better yet, alter its configuration to turn off the `G1` option. However, the best solution of all is to upgrade dvips as versions 5.86h and later have been revised so as to not cause this problem even when using the `-G1` option with non-Computer Modern fonts. See Section 6.2.2 for more details on configuring dvips.

5.6 The Large Delimiter and Operator Test

The matrix should be enclosed in brackets. The equation to the right of it should be enclosed in parentheses. The equation should have a summation symbol.

Not acceptable: some parts or all of the brackets, parentheses or summation symbols are missing. This is caused by bugs in Acrobat reader 4.0.0 and prior and/or versions of Ghostscript prior to 6.50. The solution is to upgrade both these applications.

5.7 The Minus Sign Test

The equation shown should yield a true statement.

Not acceptable: some of the minus signs are missing, especially in the superscripts. Acrobat readers prior to Version 4.0.5 have this problem. So does Ghostscript versions prior to 6.50. The solution is to upgrade both these applications.

Note: If the minus signs look like underscores, but display properly when "zoomed in" - this is not a bug, but an Acrobat Reader feature that "Greeks" text that is smaller than a certain amount. To turn this bogus thing off, select `File->Preferences->General`, or `Edit->Preferences->Page Display` with newer versions of Acrobat Reader, and deselect the "Greek Text" option.

5.8 The Math Italic Glyphs

The LaTeX math symbols `\Gamma`, `\Psi`, `\Omega`, `\gamma`, `\psi` are shown in `\mathnormal` (italic) shape with the text `\t{}` tie-after accent at the end of the group. These glyphs are in the "control character" positions of their font files.

Not acceptable: some of the symbols are missing or wrong. Versions of Acrobat Reader prior to 4.0.5 and Ghostscript versions prior to 6.50 assume that these positions will always contain control characters and never will have a valid glyph. As a result, these glyphs will not display or print properly. The solution is to upgrade both these applications.

5.9 The Large Times Roman Italic

Acceptable, but not ideal: it is probably ok if the style of the "z" differs from that of the testflow control pdf file (which embeds the base 14 fonts).

A standard PostScript Times Roman italic lower case "z" has a curly bottom. However, a Times New Roman "z" has a straight bottom. (Note that "new" here does not mean that this is a newer or otherwise improved font, just one that is slightly different and whose copyright is owned by a different party.)

This test illustrates a little known "feature" of Acrobat Reader. Microsoft has pressured Adobe to make Acrobat Reader (even Unix versions) perform certain font substitutions on the base 14 PostScript fonts when they are not embedded in the PDF file. However, when printing to a PostScript printer, the original base 14 fonts will still be used.

This idiotic policy means that, if the base 14 fonts are not embedded in the PDF file, when using a PostScript printer, what is printed by the printer will not be the same as that seen on screen, but will be the same as is called for in the original LaTeX source of the document. But, for a non-PostScript printer, what is printed by the printer will be the same as that seen on screen, but will not be exactly the same as is called for in the original LaTeX source of the document. Everybody got that? What a mess!

For this reason, Adobe now recommends that all fonts, including the base 14, should be embedded and subsetted in the PDF files.

Additional information can be found in the TeX Users Group's pdfTeX mailing list archives on October 4, 2001: ftp://ftp.tug.org/mail-archives/pdftex/2000.10 in the thread "Embedded 'standard' fonts in PDF files". Thanks to Nelson H. F. Beebe of the University of Utah for his informative posts in that thread.

Rich Sprague wrote an essay about Acrobat-related font issues at Planet PDF: http://www.planetpdf.com/mainpage.asp?WebPageID=362.

[**Side note:** Some of the earlier PostScript interpreters, such as the Adobe level 1 PostScript Cartridge for the HP LaserJet II and the HP level 1 PostScript Cartridge for the LaserJet III, used a Times Roman italic z with a flat bottom, but that is curly in bold italic.]

If the "z" has a straight bottom as displayed on screen, it is a sign that the standard base 14 fonts have *not* been embedded and that Acrobat Reader has substituted, without the user's consent, some of the base 14 fonts. This is ordinarily not too much of problem, but there is an issue to be aware of: **Some of the substitute fonts do not have all of the glyphs of the original fonts and may not be displayed on screen, or in the print of a non-PostScript printer!** Most of the missing glyphs involve math symbols, so this is rarely a problem under LaTeX which usually uses the much more mathematically extensive (non-base 14) Computer Modern fonts for mathematics.

The only way to ensure that a PDF file will always be rendered on the Acrobat Reader screen exactly as it was created, and as it will be printed, is to embed and subset the base 14 fonts into the pdf file. Subsetting provides an additional safeguard against potential font substitution as subsetted fonts are given unique names. Note that embedding the base 14 fonts will increase the size of the resultant PDF file.

Ghostscript versions 7.01 and later will embed the base 14 fonts (Nimbus) when asked to embed and subset all fonts under a PDF level 1.3 or higher.

Those using the full version of Adobe Acrobat should use Adobe Distiller, not Adobe pdfwriter, to make PDF. Recent versions of Adobe Distiller will embed and subset the base 14 fonts if the "PrintOptimized" or "PressOptimized" settings are selected. See the Adobe Distiller parameter manual: http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/distillerparameters.pdf for details on the parameters for PS to PDF conversion.

5.10 The GS Kerning Test

A footnote sized sentence is displayed:

The “Problematic” little quotes.

Not acceptable: the quote marks “crash into” the “p” or are too close to the word “little”.

This is caused by a bug in the PDF level 1.2 output of Ghostscript versions 7.00–7.05 (as well as the 7.20–7.21 developer versions). It has been fixed in releases dated after 2002/07/31. A work around is to use only PDF level 1.3 (or higher as supported by more recent versions of Ghostscript) output via the use of `-dCompatibilityLevel=1.3` when making PDF using Ghostscript.

Do not ignore this problem because it looks like a small defect. There are other manifestations that are much more severe.

5.11 The Picture and Lasy Fonts Test

LaTeX has a special set of fonts used to construct simple pictures and a set of fonts (the “lasy” fonts) with rarely used symbols. The original release of the type 1 versions (BlueSky) of these fonts contained some, normally harmless, extraneous information within the font files. However, Acrobat Readers in the 5.0 series are intolerant of this extra information and will generate a “font error” when printing to a PostScript printer. There may or may not be noticeable problems in the resulting print.

Not acceptable: Acrobat Reader 5.x generates a font error when printing to a postscript printer.

This error message may not a reliable indicator of whether or not the user’s LaTeX picture and lasy fonts have the offending data as an error message may not be triggered unless the document uses a certain number of glyphs from the affected fonts and Acrobat Readers after 5.0.5 may not be sensitive to it.

The purpose of the picture/lasy fonts test is to trigger this error on Acrobat Reader versions 5.0.0 and 5.0.5. There is nothing in particular in the picture test that the user need look for.

The picture and lasy fonts are located in the `<texmf>/fonts/type1/bluesky/cm/` directory and consist of the `lasy*.pfb`, `lcircle*.pfb` and `line*.pfb` files.

To correct this problem, a new set of LaTeX picture and lasy fonts were released on June 6, 2001. It is easy to tell the difference between the old and new fonts via their file sizes. Two examples are shown here:

Font File	Old	New
line10.pfb	10898	11036
lasy10.pfb	4684	4822

As of 2009, the BlueSky font files are now part of the AMS fonts package, the latest version of which can be obtained for free at: http://www.ctan.org/pkg/amsfonts

Upgrading is as simple as replacing the old .pfb files in `<texmf>/fonts/type1/bluesky/cm` (if instead they are in `<texmf>/fonts/type1/bluesky/latex` it’s a newer system that is likely to be already fixed) with the new ones.

Ghostscript 6.50, dvips 5.86d and later may automatically correct some font errors and shield the user from some font related problems. However, users are strongly urged to upgrade to the corrected font sets.

5.12 Duplex Alignment Test (optional)

If the user requested the optional duplex test page when compiling testflow, a second page to test duplex (two sided) printing will be created. In a perfect duplex printer, the font and back of the pages will be perfectly aligned. However, some error ($<$ 0.25inch) is acceptable. Some misalignment is due to paper shrinkage after the first pass through the fuser of laser printers. Larger errors are often the result of defective registration or pickup rollers or a misaligned laser mirror.

6 Application Version and Configuration Information

6.1 Acrobat Reader

6.1.1 Version Information

Not acceptable: Acrobat Reader versions prior to version 4.0.5.

Anything before version 4.0.5 is too buggy to be considered as even remotely usable with PDF documents created with LaTeX systems.

Determining the application number on Windows platforms is easy. See the upper left corner of the Acrobat Reader startup banner, or the `Help->About_Acrobat_Reader` window, for the three digit version number (e.g., 5.0.5).

However, determining the version of Acrobat reader is not always trivial because Acrobat Readers under Unix may not display a version number. The version may have to be inferred from the date. For Acrobat readers under Linux go to the `Help->About_Acrobat_Reader` window. It should display a date of January 2000 or later — this will mean version 4.0.5 or later.

You can get the latest version of Acrobat Reader at https://get.adobe.com/reader/. Older versions may be archived at http://www.oldversion.com/. Note that there were no Unix releases for the 6.x series (but there were Unix versions for 5.x and earlier and 7.x and later).

Unix users may wish to consider Derek Noonburg's free xpdf viewer as an alternative to Acrobat Reader http://www.foolabs.com/xpdf/.

6.1.2 Configuration Information

When printing from Acrobat Reader, make sure that there are not any selected "fit to page," "shrink page" or "expand page" options. Otherwise, the print *will* be altered in size. The `File->Page_Setup`, or `File->Print_Setup` on newer versions, menu should be checked to ensure that the page size is correct before printing.

A4 paper users should take note that Acrobat Reader has a very vile bug, and incredibly one that remains uncorrected at least up through and including version 7.00, that resets the page origin/margins to a non-A4 value every time Acrobat Reader is restarted. This is true even if A4 is displayed in the Page/Print setup. Users who need to print to A4 are required to manually deselect and then reselect A4 each time Acrobat Reader is restarted. Failing to do so will result in text that is downshifted. There are no known modifications to fix this. Previous reports of success with configuration file changes or "wrapper scripts" that corrected the configuration file on startup were incorrect.

When printing to a PostScript printer, with Acrobat Reader versions 5.x or earlier, it may be safest to use the Level 1 option as the Level 2 and Level 3 options are suspected of producing incorrect output in some circumstances. I do not know if this issue was fixed in the Acrobat Reader 5.x series and later. Acrobat 7.00 and later does not support Level 1 output, so the Level 2 output is presumed to be correct.

6.2 Dvips

6.2.1 Version Information

Recommended: dvips version 5.86f or later. Users can identify their dvips version via `dvips -version`.

dvips versions prior to 5.86f have a bug in the subsetting code which can, with some fonts, cause problems. The symptom will be font related error messages from dvips like:

```
Second number not found in Char string of /FontName
```

or

```
XX Subr not found
```

or

```
ERROR in encoding vector in <XX.pfb>
```

Sometimes these symptoms can manifest themselves in other applications, such as Adobe distiller or Ghostscript, which are processing (to pdf) the (improperly subsetted) dvips Postscript output. The one trait they all have in common is that the problem will go away when the `dvips -j0` option is used to turn off font subsetting:

```
dvips -j0 -o myfile.ps myfile
```

However, these bugs will manifest themselves only when using fonts that have subtle problems or that have been reencoded multiple times (a single glyph is used in more than one font). IEEEtran users should not encounter this problem unless unusual fonts are used. As mentioned above, the short term work around is to use the `dvips -j0` option which turns off subsetting, but the resultant files will be larger. A longer term and better solution is to upgrade dvips. See Section 7 for information on where to obtain LATEX system updates.

6.2.2 Configuration Information

Paper Size Configuration In `<texmf>/dvips/config` there is a file named `config.ps`. (Note that Unix users will require root privileges to alter this file.) Edit config.ps using a text editor. Toward the latter part of the file, there are lines that begin with `@` ("at" sign). These control the paper size selections. The first set of these lines determines the default paper size dvips will use. If this first set of lines does not specify the desired default paper size, another set of lines can be moved or entered into the first position. For example, putting:

```
@ lettersize 8.5in 11in
@+ %%PaperSize: Letter
```

(with a blank line before and after) as the first set of `@` lines will make dvips will default to using US letter paper.

Now, there are two types of dvips paper size commands. The first type adds only paper size **comments** to the PostScript output:

```
@ A4size 210mm 297mm
@+ %%PaperSize: A4
```

The `@+` line says to place the PostScript comment `%% PaperSize: A4` into the output file. Such PostScript DSC (Document Structuring Convention) comments are just hints, or suggestions about which type of paper to use. In fact many/most PostScript interpreters (Ghostscript, printers) will just ignore these and use their own default paper sizes. In a way, this is desirable as the comments will not cause any problems if the PostScript file is then printed on paper of different size than the comments suggest.

A second type of dvips paper size command adds a PostScript command which actively sets and enforces a given paper size:

```
@ a4 210mm 297mm
@+ ! %%DocumentPaperSizes: a4
@+ %%BeginPaperSize: a4
@+ a4
@+ %%EndPaperSize
```

Note the three comments (The `!` just means place the comment at the beginning of the PostScript file.) and the `a4` PostScript command.

Therefore, PostScript produced via `dvips -t A4size ...` is different from that produced with `dvips -t a4 ...`. The former may not always be automatically identified as wanting A4 paper, the latter may demand A4—causing some printers to refuse to print on anything other than A4 paper.

For instance, `ps2pdf` (Ghostscript) will have to be told the paper size to use when processing PostScript files using the first case (without paper size commands). However, `ps2pdf` will override command line and/or default paper selections if it encounters a PostScript paper size command from the second case. See Section 6.3.2 for more details.

Traditionally, the first case has been favored to avoid compatibility problems (with some buggy older printers). However, there is an increasing tendency to use the second case as it prevents confusion as to what paper size a PostScript file "wants". Always use the first case if you think users might ever want to print on a different paper type, use the second case only if you know this will never happen.

There is another dvips paper size issue. The "recommended display window area" of a PostScript file is contained in a comment known as the "BoundingBox". A BoundingBox line (for A4) looks like:

```
%%BoundingBox: 0 0 595 842
```

The BoundingBox's coordinates are in X_0 Y_0 X_1 Y_1 form where X_0,Y_0 is the lower left and X_1,Y_1 is the upper right of the image. The BoundingBox coordinates must be integers and are in units of $\frac{1}{72}$ inch (a PostScript point, i.e., a LaTeX bp unit). The values used here by dvips are taken from the lengths given next to the names in its paper size definitions.

Now, for US letter paper all is well as:

$X_1 = 8.5\text{in} \times 72\text{bp/in} = 612\text{bp}$
$Y_1 = 11\text{in} \times 72\text{bp/in} = 792\text{bp}$

But, for A4:

$X_1 = 210\text{mm}/25.4\text{mm/in} \times 72\text{bp/in} = 595.28\text{bp}$
$Y_1 = 297\text{mm}/25.4\text{mm/in} \times 72\text{bp/in} = 841.89\text{bp}$

Now, how should one round? Under normal circumstances, one would say to round to the closest integer. However, if the BoundingBox is ever smaller than its contents, some of the image may be clipped. (In practice, this may never happen as printers can not print so close to the edge of the paper anyway.) Well, guess what: current versions of dvips always round up, while Adobe and Ghostscript round to the closest integer. So, for A4, the dvips made BoundingBox will be:

```
%%BoundingBox: 0 0 596 842
```

While most PostScript parser/reader applications will expect:

```
%%BoundingBox: 0 0 595 842
```

Normally this is of no consequence. One minor notable issue is that the Unix gv application uses the BoundingBox to determine the paper size (if no PostScript paper size commands are present) and it will not recognize a PS file as wanting A4 if it sees 596 instead of 595. The workaround is to use:

```
@ A4size 594.99bp 841.99bp
```

instead of 210mm 297mm for A4 measurements.

Those outside the US are probably laughing at this point [pun intended], but it should be pointed out that this is a standards issue, not a units issue. Ghostscript versions prior to 7.05 have a very similar problem — if they are compiled with the `-DA4` option, A4 PDF pages produced with `ps2pdf` will be very slightly scaled down (about 1%).

With all this in mind, a good first set of dvips config.ps paper size definitions, for US letter paper users, might look like:

```
@ lettersize 8.5in 11in
@+ %%PaperSize: Letter

@ letter 8.5in 11in
@+ ! %%DocumentPaperSizes: Letter
@+ %%BeginPaperSize: Letter
@+ letter
@+ %%EndPaperSize

@ A4size 594.99bp 841.99bp
@+ %%PaperSize: A4
```

```
@ a4 594.99bp 841.99bp
@+ ! %%DocumentPaperSizes: a4
@+ %%BeginPaperSize: a4
@+ a4
@+ %%EndPaperSize
```

A4 users would want to move the “A4size” block to the first position so that it will be the default.

Type 1 Font Configuration Most LaTeX systems already have the Type 1 versions of the Computer Modern fonts (BlueSky/AMS fonts). But, if needed, they can be obtained free of charge from http://www.ctan.org/pkg/amsfonts.

It is normally not needed nor desirable to embed the base 14 fonts into PostScript files as these are guaranteed to be present in every PostScript printer and viewer. Furthermore, Acrobat Reader cannot view PostScript, and therefore is incapable of substituting fonts therein. So dvips' default settings regarding the embedding of the base PostScript fonts need not be changed.

However, many older LaTeX systems have dvips configurations that are not enabled to use the BlueSky fonts. For systems older than the teTeX 2.0 and MiKTeX 2.4 releases, dvips' `config.ps` file should be checked to ensure that the the following lines are present (and they should not be commented out):

```
p bsr.map
p bsr-interpolated.map
p hoekwater.map
```

The use of map files in newer LaTeX systems is controlled via the file `<texmf>/web2c/updmap.cfg` in conjunction with a `updmap` utility. Make sure that `updmap.cfg` has the following settings enabled:

```
dvipsPreferOutline true
LW35 URWkb
pdftexDownloadBase14 true
dvipdfmDownloadBase14 true
dvipsDownloadBase35 false
MixedMap bsr.map
MixedMap bsr-interpolated.map
```

and then run the `updmap` (or rather `updmap-sys` under teTeX 3.0) utility as root. The MiKTeX user guide recommends that the utility `initexmf --edit-config-file updmap` be used rather than hand editing `updmap.cfg`. Note that teTeX 3.0 users will probably want to use the new `updmap-sys` to update the map files system-wide (as `updmap` does under teTeX 2.0). For MiKTeX, there is an `updmap` command line utility under MiKTeX in addition to the GUI configuration tool.

Other Configuration Other lines which are worthy to be present in dvips' config.ps include:

```
D 7200
```

(If dvips output is usually converted to PDF, rather than printed directly.) This effectively prevents dvips from making adjustments based on an assumed printer resolution.

dvips versions 5.86g and later provide increased security against malicious DVI code (Yes, it is possible to put executable commands in dvi files!). The z1 option disables the execution of such code. It is a good idea to put in the z1 now, even for systems with older versions of dvips (which will silently ignore the z option).

```
z1
```

There should not be any `G` option lines like:

```
G
```

in `config.ps` or `config.pdf`. Remove or comment out with `%` any such lines.

[**Note:** dvips versions 5.86h and later have been revised so as to not cause this problem even when using the `G1` option with non-Computer Modern fonts.]

In order to implement Timothy Van Zandt's dvips line thickness fix, create a file named `alt-rule.pro` in the `<texmf>/dvips/base` directory containing the following code and comments:

```
% Timothy Van Zandt's alt-rule.pro algorithm to draw rules using
% Postscript's stroke rather than fill. This allows the resulting
% lines to have a more uniform thickness.
TeXDict begin
/QV {
  gsave newpath /ruleY X /ruleX X
  Rx Ry gt
  { ruleX ruleY Ry 2 div sub moveto Rx 0 rlineto Ry }
  { ruleX Rx 2 div add ruleY moveto 0 Ry neg rlineto Rx }
  ifelse
  setlinewidth 0 setlinecap stroke grestore
} bind def
end
```

Now, activate it by inserting the following lines into the `<texmf>/dvips/config.ps` file (just before the `@` lines is a good place):

```
% Use Timothy Van Zandt's alt-rule.pro algorithm to draw rules using
% Postscript's stroke rather than fill. This allows the resulting
% lines to have a more uniform thickness. It seems to work well with
% no adverse effects.
% tex.pro or texc.pro is loaded by default (texc.pro will be used
% if the Z option is used to compress bitmaps, but we need to call
% tex.pro/texc.pro here because it must be loaded before the
% alt-rule.pro patch).
% use tex.pro if the dvips Z1 option is not used
h tex.pro
% or use texc.pro if the dvips Z1 option is used
% (If in doubt, just use tex.pro - things will work in any case.)
% h texc.pro
h alt-rule.pro
```

Warning: If the dvips' Z option is invoked on the command line, then `texc.pro` will be loaded after `config.ps` which will override the `alt-rule.pro` code. I know no way around this problem except to place the `alt-rule.pro` code within `texc.pro`. Thanks to Tasuki Yamamoto for reporting this problem.

After `config.ps` has been configured as described above, the user can make good PostScript that is suitable for PDF conversion via:

```
dvips -o myfile.ps myfile
```

without the need to call any `-Ppdf` options.

To make PostScript for printing directly to a PostScript printer (or for a document that is using some type 3 fonts), one should call dvips like:

```
dvips -D600 -o myfile.ps myfile
```

where the `-D` option specifies the resolution (in dots per inch) of the printer — that is, the desired resolution is for the type 3 fonts (e.g., <= 1200 if using type 3 fonts).

Additional dvips configuration information can be found in the dvips user's manual which is available at http://www.tug.org/tetex/texmf-dist/doc/programs/dvips.pdf.

6.3 Ghostscript

(Adobe Distiller PDF settings are similar)

6.3.1 Version Information

Not acceptable: Ghostscript versions prior to the 8.x series. Versions in the 7.x series and prior have known bugs that affect LaTeX users, only a subset of which is tested in testflow.

Users can identify their Ghostscript version via `gs -version`.

Versions 7.04 and later feature an improved security mechanism which can cause problems with older versions of xdvi (prior to 22.40j for xdvi-k and 22.57 for the non-Kpathsea-enabled version) and the "gv" PostScript viewer (3.5.8 and prior). The symptom will be that graphics files referenced by relative paths (e.g., `../myfigs/afig`) cannot be displayed and an error message like:

```
gs: Error: /invalidfileaccess in --file--
```

The solution is to update xdvi and/or gv to a more recent version.

6.3.2 Configuration Information

A recommended set of command line options when using GhostScript to make PDF from PostScript is (adjust `PAPERSIZE` as needed, e.g., `a4`):

```
gs -dSAFER -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sPAPERSIZE=letter-dPDFSETTINGS=/prepress -dCompatib
ilityLevel=1.4 -dMaxSubsetPct=100-dSubsetFonts=true -dEmbedAllFonts=true -sOutputFile=myfile.pdf my
file.ps
```

The Ghostscript `-dPDFSETTINGS=/prepress` is similar to the Acrobat Distiller "Prepress Optimized" settings. Some of the other parameters such as `-dEmbedAllFonts=true` are implied by it, but it is a good idea to be explicit about them so as not to rely on default values.

Note: some Ghostscript options begin with `-s` (strings) others begin with `-d` (values). Ghostscript's "pdfwrite" device is like Acrobat's Distiller, and should not be confused with the inferior Acrobat "Pdfwriter" device.

Be aware that Ghostscript will *not* embed the base 14 fonts under PDF level 1.2. Furthermore, PDF level 1.2 is useful only for Acrobat Reader versions prior to 4.0 — which are too buggy to handle PDF from LaTeX anyway. For these reasons, it is important that PDF level 1.3 or higher be specified when creating PDF. PDF level 1.4 is the default used by most recent pdfTeX systems and will work fine even under versions of Acrobat Reader as old as 4.0.5.

The `ps2pdf` script should be modified to use a PDF level of 1.3 or higher in conjunction font embedding/subsetting. For instance, a good `ps2pdf` script on a Unix system might look like:

```
#!/bin/sh
# Convert PostScript to PDF using \LaTeX\ recommended parameters.
# Adjust default PAPERSIZE as needed.
exec ps2pdfwr -sPAPERSIZE=letter -dPDFSETTINGS=/prepress
-dCompatibilityLevel=1.4 -dMaxSubsetPct=100 -dSubsetFonts=true
-dEmbedAllFonts=true "$@"
```

The Ghostscript ps2pdfwr script then in turn adds the `-dSAFER` `-dNOPAUSE` `-dBATCH` `-sDEVICE=pdfwrite` and `-sOutputFile` options and then calls gs.

MS Windows based Ghostscript installations use `ps2pdf.bat` batch files in the `gs\gsX.YZ\lib` (where X.YZ is the version number of Ghostscript) subdirectory in the Ghostscript install directory. Users of these should update them to use the recommended parameters. For example, the PDF level line in ps2pdf.bat should be changed from 1.2 to 1.3 or 1.4:

```
echo -dCompatibilityLevel#1.4 >_.at
```

Then, the other recommended Ghostscript options can be added to `ps2pdfxx.bat` (this file is called by `ps2pdf.bat`, `ps2pdf12.bat` and `ps2pdf13.bat`, hence the `xx` in the filename):

```
echo -q -dSAFER -dNOPAUSE -dBATCH -sPAPERSIZE=letter -dPDFSETTINGS=/prepress
-dMaxSubsetPct=100 -dSubsetFonts=true -dEmbedAllFonts=true
-sDEVICE#pdfwrite >_.at2
```

Because additional command line options can override previous ones, it is easy enough to override `ps2pdf`'s default options if needed:

```
ps2pdf -sPAPERSIZE=a4 myfile.ps myfile.pdf
```

Note that PostScript files containing PostScript commands that are direct orders to set the paper size ("a4", "letter", etc.) will countermand any `ps2pdf` options to the contrary.

The Ghostscript `-dFIXEDMEDIA` option can be used to "lock" the paper size selection and prevent any PostScript paper size operators in the .ps file from overriding the `ps2pdf` options. For example,

```
ps2pdf -sPAPERSIZE=a4 -dFIXEDMEDIA myfile.ps myfile.pdf
```

will force an A4 paper size for `myfile.pdf` even if `myfile.ps` contains hardcoded paper size commands for a different paper type (like from using the dvips `-t letter` option as opposed to the `-t lettersize` option. See Section 6.2.2 for more details).

It is not recommended that `-dFIXEDMEDIA` be used in the `ps2pdf` script itself as this will prevent the user from being able to alter the paper size with `ps2pdf` command line options. Furthermore, it is unusual to override a PostScript file in this fashion, such as to force a file to print on a paper type it was not designed for.

Additional information about PostScript to PDF conversion can be found in the Ghostscript `ps2pdf` documentation at http://www.ghostscript.com/Documentation.html.

For example, usage information, including a very informative section on paper size control, for the most recent version under development (CVS), can be found at: http://www.ghostscript.com/doc/current/Use.htm

There is also a section in the above link concerning how to modify Ghostscript's `gs_init.ps` file (e.g., `/usr/share/ghostscript/8.54/lib/gs_init.ps`) so that Ghostscript's default paper size (that is, what it defaults to without any options) can be changed to something different from US letter. This may be of use to those A4 users who use GhostScript to convert PostScript to many different formats rather than just PDF.

Information related specifically to `ps2pdf` can be found at: http://www.ghostscript.com/doc/current/Ps2pdf.htm Another good source of information on PS to PDF conversion is the Adobe Distiller parameter manual: http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/distillerparameters.pdf

Note that a new Ghostscript upgrade or installation will likely overwrite any changes to the `ps2pdf` scripts. For this reason, it might be a good idea to create `ps2pdf` scripts with different file names — like `ps2pdfmy`, and/or save copies of them in a directory outside the one in which they are currently located.

Users of graphical interfaces to Ghostscript (such as gsview) should set the options/preferences for PDF output in accordance with the recommendations above.

For gsview, these are located in the `File->Convert->Properties` menu. `-dFIXEDMEDIA` corresponds to gsview's "Fixed Page Size" option.

6.4 pdfTEX

The traditional way to produce PDF under LATEX, is to use `dvips` followed by `ps2pdf`. However, with the advent of pdfLATEX, users can create PDF directly from LATEX source.

Note that most modern TEX systems use `pdfetex`, an extended version of pdfTEX, which serves as many different TEX compilers depending on how it is invoked (`latex`, `pdflatex`, `tex`, etc.).

6.4.1 Version Information

Not acceptable: `pdftex` versions prior to 1.20a. Versions prior to 1.20a do not support embedding of the base 14 fonts and versions prior to 1.00a pretest (2001/08/06) are considered to be too buggy for production us.

`pdftex`'s version information can be obtained via:

```
pdftex --version
```

The version code of interest is the group of numbers and letters after "pi" in the first line displayed. e.g.,

```
pdfeTeX 3.141592-1.30.3-2.2 (Web2C 7.5.5)
```

indicates version 1.30. There may also be an obvious date code within in the version information — especially on the beta releases.

Section 7 for information on where to obtain LaTeX system updates.

6.4.2 Configuration Information

Paper Size Configuration For teTeX systems prior to teTeX 2.0 as well as MiKTeX `pdftex`'s default paper size is determined by the configuration file `<texmf>/pdftex/config/pdftex.cfg`. Within it, lines like:

```
page_width 8.5in
page_height 11in
```

determine the default paper size. For MiKTeX systems, the format files will have to be rebuilt in order for changes in `<texmf>/pdftex/config/pdftex.cfg` to take effect (see below). Also, the MiKTeX user guide recommends that the utility `initexmf --edit-config-file pdftex` be used rather than hand editing `pdftex.cfg`.

The configuration of teTeX systems 2.0 and later is controlled by a `<texmf>/tex/generic/pdftex/pdftexconfig.tex` file. Ensure that the following lines are present and correct for you:

```
\pdfpagewidth=8.5 true in
\pdfpageheight=11 true in
\pdfoptionpdfminorversion=4
```

For systems that use `pdftexconfig.tex` for configuration as well as MiKTeX which automatically builds a `pdftexconfig.tex` from its `pdftex.cfg`, the format files will have to be rebuilt in order for the changes to take effect. For teTeX this is accomplished via (as root):

```
fmtutil --all
```

teTeX 3.0 users will probably want to use:

```
fmtutil-sys --all
```

(rather than `fmtutil`) to rebuild all the format files system-wide (as `fmtutil` does under teTeX 2.0) For MiKTeX:

```
initexmf --dump
```

(or invoke the rebuild via the MiKTeX GUI control panel).

Type 1 Font Configuration Font configuration is done differently in the teTeX 2.0 and MiKTeX 2.4 and later systems. These newer systems use a `updmap.cfg` file in conjunction with a `updmap` utility to control the use of font map files.

Type 1 Font Configuration For Older Systems `pdftex`'s `pdftex.cfg` configuration file is much like `dvips`' as fonts are concerned. It is important that the following lines be present and uncommented:

```
map +bsr.map              % CM/AMS fonts
map +bsr-interpolated.map % additional sizes
map +hoekwater.map        % additional fonts from Taco Hoekwater
```

so that the BlueSky Type 1 Computer Modern fonts will be used.

Note: The `pdftex.cfg` file on some systems (MiKTeX) may not have these lines, but instead uses a `map psfonts.map` line which basically does the same thing as the three above.)

`pdftex` can also be configured to embed the base 14 fonts (Nimbus) by replacing the line (usually the first of the map lines to appear in pdftex.cfg):

```
map acrobat-std-adobe-buildin.map
```

with:

```
map acrobat-std-urw-kb.map
```

This basically tells `pdftex` to use and embed the Nimbus fonts instead of relying on the (substituted) fonts that are built into Acrobat Reader.

Note: Some `acrobat-std-urw-kb.map files` (this will be located in `<texmf>/dvips/config`) have a mistake in two lines:

```
usyr StandardSymL <psyr.pfb
.
.
uzdr Dingbats <pzdr.pfb

.
```

These should instead read:

```
psyr StandardSymL <psyr.pfb
.
.
pzdr Dingbats <pzdr.pfb
```

For systems that do not have a `acrobat-std-urw-kb.map` file (MiKTEX puts everything in `psfonts.map`), the teTEX `acrobat-std-urw-kb.map` typically contains:

```
pcrb8r NimbusMonL-Regu-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <ucrb8a.pfb
pcrbo8r NimbusMonL-Regu-BoldOblique "TeXBase1Encoding ReEncodeFont" <8r.enc <ucrbo8a.pfb
pcrr8r NimbusMonL-Regu "TeXBase1Encoding ReEncodeFont" <8r.enc <ucrr8a.pfb
pcrro8r NimbusMonL-Regu-Oblique "TeXBase1Encoding ReEncodeFont" <8r.enc <ucrro8a.pfb
phvb8r NimbusSanL-Regu-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <uhvb8a.pfb
phvbo8r NimbusSanL-Regu-BoldOblique "TeXBase1Encoding ReEncodeFont" <8r.enc <uhvbo8a.pfb
phvr8r NimbusSanL-Regu "TeXBase1Encoding ReEncodeFont" <8r.enc <uhvr8a.pfb
phvro8r NimbusSanL-Regu-Oblique "TeXBase1Encoding ReEncodeFont" <8r.enc <uhvro8a.pfb
psyr StandardSymL <psyr.pfb
ptmb8r NimbusRomNo9L-Medi "TeXBase1Encoding ReEncodeFont" <8r.enc <utmb8a.pfb
ptmbi8r NimbusRomNo9L-MediItal "TeXBase1Encoding ReEncodeFont" <8r.enc <utmbi8a.pfb
ptmr8r NimbusRomNo9L-Regu "TeXBase1Encoding ReEncodeFont" <8r.enc <utmr8a.pfb
ptmri8r NimbusRomNo9L-ReguItal "TeXBase1Encoding ReEncodeFont" <8r.enc <utmri8a.pfb
pzdr Dingbats <pzdr.pfb
```

it is possible to modify the `pcrb8r`, `pcrbo8r`, etc., lines in MiKTEX's `psfonts.map` to be like the ones above so that `pdftex` will embed the Nimbus fonts.

Note that `pdftex` will name these embedded fonts like "NimbusRomNo9L-Regu", while Ghostscript created PDF will use the more traditional name "Times Roman".

Type 1 Font Configuration For Newer Systems The map configuration for newer systems is controlled by the file `<temxf>/web2c/updmap.cfg`. Edit this file and ensure that the following options are present and correct:

```
dvipsPreferOutline true
LW35 URWkb
pdftexDownloadBase14 true
dvipdfmDownloadBase14 true
dvipsDownloadBase35 false
MixedMap bsr.map
MixedMap bsr-interpolated.map
```

and then run the `updmap` (or rather `updmap-sys` under teTEX 3.0) utility as root. The MiKTEX user guide recommends that the utility `initexmf --edit-config-file updmap` be used rather than hand editing `updmap.cfg`. Note that teTEX 3.0 users will probably want to use the new `updmap-sys` to update the map files system-wide (as `updmap` does under teTEX 2.0). For MiKTEX, there is an `updmap` command line utility under MiKTEX in addition to the GUI configuration tool.

6.5 xdvi

xdvi's default paper size is determined by its configuration file:

```
<texmf>/xdvi/XDvi
```

Within this file, the default paper size can be set with a line like (older xdvi configuration syntax shown):

```
XDvi*paper:us
```

for US letter or

```
XDvi*paper:a4
```

for A4 paper.

Newer versions of xdvi use a slightly different configuration file syntax in that they no longer use the "XDvi" option prefix and use "letter" to indicate US letter paper:

```
*paper:letter
```

There are two major support sites for xdvi where the very latest versions can be downloaded. For the Kpathsea enabled version (which is by far the most popular in use) see Stefan Ulrich's xdvi-k site on Sourceforge http://xdvi.sourceforge.net/ For the (old) non-k version, see Paul Vojta's site http://math.berkeley.edu/~vojta/xdvi.html.

6.6 YAP

The paper size can be selected in the menu:

```
View->Options->Display
```

7 Upgrading LaTeX

With the possible exception of xdvi (see Section 6.5), most LaTeX applications are distributed as an integral part of their respective LaTeX distributions and are not generally available from their "own homepages."

Users with commercial versions of LaTeX should consult the vendor for upgrade availability. Users with Linux or FreeBSD systems may be able to obtain LaTeX updates in the same way as they do other applications for their particular distro.

MiKTeX (the most common LaTeX system for MS Windows) users can obtain upgrades at the MiKTeX site http://www.miktex.org/ as well as on CTAN http://www.ctan.org/pkg/miktex.

fpTeX is no longer supported, so previous users will have to switch to another LaTeX distribution.

On May 2006, Thomas Esser announced that he would no longer be maintaining teTeX (for Unix), http://www.tug.org/tetex/. teTeX users are encouraged to migrate to TeX Live http://www.tug.org/texlive/. The most recent images of TeX Live can be found on CTAN: http://www.ctan.org/pkg/texlive

This document does not cover the upgrading of individual teTeX applications.

8 Testflow Control PS/PDF File Build Code Information

A "build code" can be seen when the testflow control PS/PDF files are viewed. Below is a list of these build codes and information about what applications and configurations were used to create each of them:

Note: build code 0000 designates a non-control (user made) PS/PDF file.

```
****** Version 1.1 Build code 0004 ******
Creation Date:        Jan. 10, 2007
Operating system:     Linux
Duplex page test:     yes
LaTeX2e version:      pdfeTeX 3.141592-1.30.3-2.2
dvips version:        5.95a
Ghostscript version:  8.54

Computer Modern font: BlueSky, type 1, June 6, 2001 revision
Base PostScript font: Nimbus (in the PDF only) from GhostScript

work flow:            latex->dvips->Ghostscript (ps2pdf)
notable dvips parameters: D7200, alt-rule.pro

Ghostscript ps2pdf parameters: -dPDFSETTINGS=/prepress
-dCompatibilityLevel=1.4 -dMaxSubsetPct=100 -dSubsetFonts=true
-dEmbedAllFonts=true

- additional testflow_ctl_A4 parameters -
ps (dvips):     @ A4size 594.99bp 841.99bp
                @+ %%PaperSize: A4

pdf (ps2pdf):   -sPAPERSIZE=a4


- additional testflow_ctl_LTR parameters -
ps (dvips):     @ lettersize 8.5in 11in
                @+ %%PaperSize: Letter

pdf (ps2pdf):   -sPAPERSIZE=letter
*****************************************
```

That's all folks ;)
mds