Save yourself some grief: a sort of comprehensive guide to making a PDF/A-compliant thesis file

Luis R.J. Costa

August 8, 2018

Contents

1 Prologue

Should you be in haste with no time to waste for seeming trivia, skip on to section 3. Perhaps you will return to this parchment when the time is ripe, with your favourite brew in front of you. Whereas you with a little time to invest, read on...

In the autumn of 2017, Aalto University School of Electrical Engineering decreed that printed and bound copies of bachelor's and master's theses would no longer be collected for the library archives, so joining the School of Engineering and the School of Science, who had already made the decision to take the plunge a few months earlier. The use of shelf space was becoming unsustainable and limited archive resources had to be used wisely. Thus it is to be that now you must submit your thesis in electronic form as a PDF/A-compliant file [1] prior to graduation. Since then, the scenario that I, as the current keeper of the LaTeX template [2] provided by the School of Electrical Engineering, have encountered most often is the following.

> You chose to use the LaTeX template to write your thesis, and you got on with doing the research, generating data, drawing figures, plotting graphs, and writing the thesis. A few days, or worse still, a few hours before the submission deadline, you decide to validate the PDF file generated by LaTeX [3], not expecting anything

> untoward. After all, the template package "enables producing a PDF/A-compliant document", and, you think, if push comes to shove, creating a PDF/A file is only a simple matter of conversion: feed the thesis PDF to the conversion software—you are blissfully unaware of any such software—and it will regurgitate a PDF/A-compliant file in seconds. However, you watch your screen in horror as the validator spews out a seemingly endless list of cryptic errors. You haven't the foggiest idea what they mean, let alone what to do to make them go away. You hastily recheck the in-file documentation in the template and the general instructions provided by the Aalto University Learning Centre [4], but you find no help there. Now you feel panic raise its head in the depths of your gut. You desperately need someone to help you... from IT support, perhaps. You look at your watch. There may be someone still working...

Thus I write this guide of sorts for you who chose wisely to seek knowledge first to avoid finding yourself in the unpleasant pickle described above. I also write this for you who, unable to find salvation in time and so forced to postpone your thesis submission, still seeks a solution. Know that a lot of the wisdom contained in this guide of sorts, particularly that concerning [graphics] files to be included in your LaTeX thesis document, applies in general to creating PDF/A-compliant documents also using other manner of software, like Microsoft Word [5] or LibreOffice [6]. So spread the word, tell your peers of this wisdom that will keep their cortisol levels in check during the stress of the final hours before thesis submission. Being aware that the devil is in the detail, I try to provide sufficient detail so that you understand the actions you should take, and for those of you inclined to seek deeper knowledge I include references.

2 A tale of two PDFs

Scientia potentia est, knowledge is power, and so it is here too. Knowing what PDF/A is and how it differs from the ubiquitous PDF will help you understand the choices and checks necessary to remove thorns on the path to produce your PDF/A-compatible thesis file.

2.1 PDF: a minuscule history[1]

It was a time of the rapid spread of the World Wide Web, it was a time of the fall of the telegram, it was the epoch when open-source software was becoming mainstream, it was the epoch when propriety software tightened its grip of the computer market, it was the season of an increase in the sharing of files, it was the season of the contagion of spam—in short, it was the period of the blossoming of a new industry, for good or for evil.

It was the year 1993 when Adobe Systems made the specifications of the portable file format (PDF) they had devised, based on its own PostScript format [9], freely available [10, History and standardization], [11], [12]. The following year they made their PDF viewer, Acrobat Reader [13], available for free [7], [14]. Because PDF files were readable across almost all platforms—implying Windows [15] and Mac OS [16]—with the release of their free PDF viewer [17, Why is PDF attractive?], and because available specifications made it possible for developers to write PDF viewer software for all platforms including Linux [18], PDF soon became the de facto standard for fixed-format electronic documents despite PDF being a propriety format.

The format contains a complete description of a fixed-layout flat document, which includes, in addition to the content text, information on fonts used, vector graphics, raster images, data compression when used, and other information needed to display the document file. The font

[1] For a more detailed history, see [7, 8].

information contains a description of the characteristics of the typeface, enabling the reader program to deduce which fonts to use from the computer operating system. Unavailable fonts are substitued with default fonts, often not matching the original character at all. Alternatively, the entire font sets or only the subsets of the characters used are embedded within the PDF file, resulting in a file size that is slightly larger than with just the description, but now making the document always readable.

Adobe then released PDF as an open standard in mid 2008, and it was published as ISO 32000-1:2008 [19], [20] by ISO, the International Organization for Standardization [21]. Adobe granted the rights to make, use, sell and distribute PDF-compliant implementations without having to pay royalties. However, this standard includes some propriety technologies defined by Adobe that are standardised and so are not supported by many popular third-party implementations of PDF. The newest standard, ISO 32000-2 (PDF 2.0), released in July 2017, does not contain any propriety technology [22].

2.2 PDF/A: the nuts and bolts

PDF/A is a standardised and constrained version of PDF released in 2005 [23] intended for archiving—hence the "A"—and long-term preservation of electronic documents. What this means is that all the information necessary for rendering the document exactly as it was made must be found in the file. Technically speaking, this means that at least all fonts must be embedded into the file, all color management information must be available, and relevant data pertaining to compression and encryption methods used, if used, must be found in the file. All this necessary information must be arranged in the specified structure. PDF/A also requires that metadata is provided in a standardised format in the file so that search engines are able to find documents and search their contents—after all, you do want your work to be found by others. [1], [24]

The PDF/A standard has two levels of conformance:

Level b (basic) requires that the visual aspect of the document is reproduced reliably.

Level a (accessible) has additional requirements, like language specification, tagging [25], [26], a hierarchical structure, and a mapping of characters to Unicode, all properties necessary for software in devices used by physically impaired users, such as touchscreens in mobile devices, to be able to access, retrieve and interpret different parts of the document. [27]

At the time of writing, three versions, or parts, of the PDF/A standard exist, all three divided into the two levels of conformance mentioned above, and a fourth version is in the works.

PDF/A-1 This standard, ISO 19005-1:2005 [28], published in 2005, is based on PDF 1.4, and excludes all propriety technologies. Colour transparencies are not supported in PDF/A-1 even though PDF 1.4 supports it, because the formulae used to calculate the transparencies were not disclosed by Adobe at the time [10, Transparency], [29, Transparency]. [1], [30]

PDF/A-2 This standard, ISO 19005-2:2011 [31], published in 2011, is based on ISO 32000-1 [19], [20] which specifies how to use PDF 1.5, 1.6 and 1.7 for long-term preservation. The major enhancements relevant to you is support for colour transparencies and layers in graphics (Adobe had by now revealed how they calculated transparency) and the possibility to embed OpenType fonts. Other enhancements include the possibility to attach PDF/A-compliant files, improvements to tagged PDF for better accessibility, and support for JPEG 2000 compression. [1], [32]

Wait, did I say there are two levels of conformity? Well, a third level of conformity is specified for PDF/A-2:

Level u: PDF/A-2u[2] is PDF/A-2b conformance along with Unicode mapping of all text, which makes searching text in the document easier. [1, PDF/A-2], [29, New Conformance Level PDF/A-2u - "u" for Unicode]

PDF/A-3 This standard, ISO 19005-3:2012, published in 2012, is also based on PDF 1.5–1.7 as formalised in ISO 32000-1 [19] and is an extension of the PDF/A-2 standard [34]. In contrast to PDF/A-2, this standard allows the embedding of any file format: of CAD, word processor, spreadsheet, graphics, compressed archive file formats, you name it. This standard only assures that PDF/A documents can be viewed using a compliant viewer, but it allows forming a collection of relevant documents. An example application of this standard is archiving emails and their attached documents. [1, PDF/A-3]

PDF/A-4 This standard, ISO 19005-4 [35], is based PDF 2.0 [22] and is expected to be released in 2018 [1, PDF/A-4].

Armed with this information, you are now in a position to make your own choices and understand those made for you in order to sucessfully create a PDF/A-compliant file.

3 Keys to success

3.1 Which PDF/A version should I use?

The Aalto University instructions for producing a PDF/A-compliant PDF document [4] tell you that you are to use versions PDF/A-1a, PDF/A-1b, PDF/A-2a, or PDF/A-2b for your thesis file (if you read section 2.2, you know why PDF/A-3 is not allowed), but it lacks essential details to help you to avoid pitfalls in the process. To begin, which PDF version is best suited for you?

The general answer to this question is PDF/A-2b, but this choice alone does not always guarantee successful generation of a conforming PDF/A-2b file. The optimal choice will typically depend on the graphics or other files that you include in your thesis file. Are your figures mainly graphs and flat diagrams without transparencies? If yes, use PDF/A-1. Unfortunately, to complicate matters, some software make images flat or opaque via transparency parameters, in effect making them have transparency. If your graphs are 3-D plots with surfaces having transparencies for visual effects or you have diagrams using colour transparencies and/or layers, then your choice is PDF/A-2. Being less stringent, b-level conformance suffices, and if you use LaTeX [with the `pdfx.sty` package], regardless of whether you use the Aalto thesis template [2] or not to typeset your thesis, for now you are stuck with b-level compliance. If you use Microsoft Word, LibreOffice or some other program that supports it, you can use a-level conformance.

3.2 Things to keep in mind when creating graphics files

Since implementing support for PDF/A-compliancy in the LaTeX Aalto thesis template, my experience resolving PDF/A nonconformance has been that the culprit was either syntax errors in how metadata is added in the `tex` template file, errors that are easily avoided by being careful when editing the newest version of the template file, or in the included graphics file, which formed the bulk of the error sources. Hence I dedicate a little space for dos and don'ts[3] when

[2] The LaTeX `pdfx.sty` package [33], used in the Aalto thesis template, supports PDF/A-2u and PDF/A-3u, but the thesis template does not permit it. The package encodes text in Unicode even for PDF/A-2b and PDF/A-3b.

[3] I prefer writing this idiom in the form "dos and don'ts" rather than "do's and don'ts", let alone the ungainly "do's and don't's" (see footnote 5). The last form is promoted in the book *Eats, Shoots & Leaves* by Lynne Truss, following a traditional rule that an apostrophe is used with the plural of numbers, single letters, abbreviations and acronyms, and when pluralising a word that is being treated as a noun.

preparing graphics files. I restrict the discussion here to graphics formats that LaTeX, particularly `pdflatex`, can process. EPS[4] [36] and PDF are used for scalable vector graphics images whereas JPEG/JPG [37] and PNG [38] are used for raster graphics or bitmap images. The tips in this section apply to EPS and PDF files that are included in the LaTeX template, but they apply generally no matter what software you use to produce the complete PDF/A-compliant document.

Dos[5]

- Favour the use of vector graphics formats (this means PDF here) whenever possible, because they scale nicely when magnified and do not get pixelised like rasterised images do.
- *Embed the fonts you use in your graphics file.* This is crucial to the success in creating a PDF/A-compliant document, as was discussed in section 2.2. How to embed fonts depends on the program you use to create your image. The image format you should aim for with `pdflatex` is regular PDF of the appropriate version with embedded fonts (see how in the next point). You might have to create an EPS file as an intermediate step.

  Unfortunately, embedding fonts is not supported by all software that export EPS or PDF files. A case in point is MATLAB [39], which has still not implemented this feature. Since MATLAB is widely used at Aalto University, in section 3.3 I provide ways to produce a regular PDF file with embedded fonts.
- Set the [Acrobat Distiller] parameter `CompatibilityLevel` [40] to set the appropriate PDF version whenever possible: use 1.4 for PDF/A-1-compliance, or 1.5, 1.6 or 1.7 for PDF/A-2. This is possible with scripts/programs, like `ps2pdf` [41], that use Ghostscript [42] to convert PostScript files to PDF.

  Ghostscript and `ps2pdf` are usually part of the LaTeX system. This is true for MikTeX on Windows and for at least the Linux Ubuntu distro. If not, install Ghostscript on your computer. `ps2pdf` is a script that uses Ghostscript with default and optional sets of options—a wrapper in technical jargon—and is part of the Ghostscript installation.
- Images like photographs or surfaces with certain visual effects like shadows can only be in a rasterised format, either JPEG or PNG in this case. No special precautions are necessary here since characters in any fonts are simply a part of the rasterised image. The JPEG format is lossy compared to PNG. You can use either of the formats, but common sense should dictate your choice here, where you should weigh the trade-off between file size and image resolution (image resolution determines how much the image can be magnified before appearing pixelised).
- Everytime you embed a new file in your document, test the document for PDF/A conformity. This way you know right away when things go awry because of an incompatible embedded file, and you can fix the problem.

[4] Encapsulated PostScript is essentially a description of, say, an image in the PostScript (PS) language, but differs primarily from a regular PS file in that it has a bounding box defined specifying the dimensions of the image in contrast to PS, where the dimension of the image is the entire page.

[5] Dos, typically a part of the idiom in footnote 3 or a reference to it, is the plural of do, not DOS, the acronym for disk operating system, which you are probably familiar with from MS-DOS on your Windows computer. I follow the rule to add an "s" to the end of the word to form the plural, as recommended, among others, by the *Oxford Style Manual*, the *Cambridge Dictionary*, and *The Chicago Manual of Style* for the case of this idiom, although other manuals like the *Macmillan Dictionary* and *The Associated Press Stylebook* recommend "do's" because the apostrophe improves readbility.

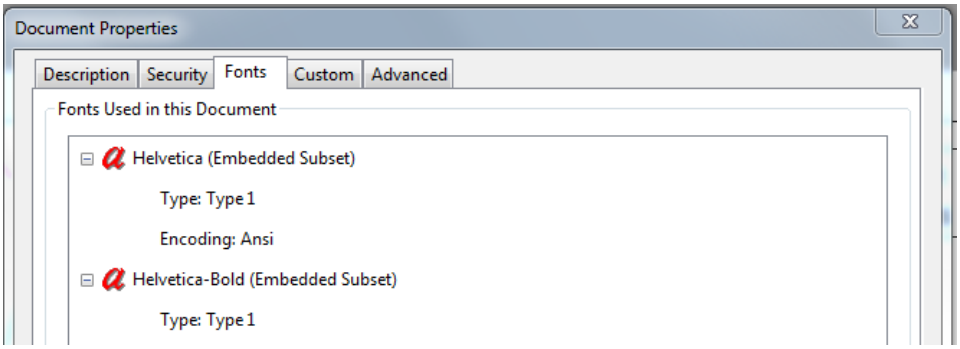

Figure 1: The properties of a PDF file showing that it contains embedded fonts.

Don'ts

- *Never* use the PS format for your images; use EPS instead if you must. This applies whether you use LaTeX or any other typesetting or word-processor software.
- Do not forget to embed the fonts in the image file. I repeat this requirement because you will run into trouble if you do not embed the fonts.
- Do not use PDF/A for [graphics] files to be embedded into the thesis file. The image may not scale correctly in the complete PDF/A file.

In order to check whether your PDF image file contains embedded fonts, open the file using, for example, Acrobat Reader and go to the menu File→Properties. In the window that opens, click on the Fonts tab, where you will see a list of all the fonts in the file. Every embedded font will have "(Embedded)" or "(Embedded Subset)" next to it, as shown in figure 1. If it does not, the font is not embedded.

3.3 Embedding fonts in graphics files produced with MATLAB

Now, I show you three ways to produce a PDF file with embedded fonts and appropriate compatibility level. You also need Ghostscript installed on your computer, which should be there along with your LaTeX system, or you can download it from www.ghostscript.com. The first two methods are lines of code you write into your MATLAB file. I recommend such approaches because you are able to redo your images easily, if necessary, without having to intervene in the process. The third method is using the graphical interface via the menus; manual labour, in other words. I have used MATLAB r2018a to test these methods.

3.3.1 Method 1: using `export_fig`

This method is by far the most straightforward way to directly produce a PDF file with the fonts embedded in it. The MATLAB toolbox `export_fig` must be installed (download the `zip` file from http://www.mathworks.com/matlabcentral/fileexchange/23629-export-fig or https://github.com/altmany/export_fig, `unzip` it into a convenient folder, and use MATLAB's command `pathtool` to tell it where to look for the function `export_fig`), and you must have a sufficiently recent[6] version of MATLAB for the fonts to be embedded. Use it as follows

[6] I tested this with version r2018a, but I have no idea which is the oldest version where font embedding using `export_fig` is supported. MATLAB by itself does not support font embedding yet.

in your MATLAB file:

```
export_fig('filename.pdf')
```

gives a light grey background to the figure, whereas

```
export_fig('-transparent', 'filename.pdf')
```

draws the figure without the grey background.

See the `export_fig` documentation for options you can use to fine-tune your exported image.

3.3.2 Method 2: the EPS and `ps2pdf` route

If you are unable to produce the PDF with embedded fonts using the first approach, perhaps due to too old a version of MATLAB, first save your image as an EPS file by writing in your MATLAB file

```
print('-painters','-depsc','filename.eps')
```

to produce a level 3 EPS file or

```
print('painters','-depsc2','filename.eps')
```

to produce a level 2 EPS file, and then convert the EPS file to PDF. To do this, in your MATLAB file in Linux write (note the `=` in the options)

```
system(sprintf('ps2pdf -dEPSCrop -dPDFSETTINGS=/prepress "file.eps" "file.pdf"'));
```

and in Windows write (note the `#` in the options)

```
system(sprintf('ps2pdf -dEPSCrop -dPDFSETTINGS#/prepress "file.eps" "file.pdf"'));
```

The setting `/prepress` sets, among other things, the compatibility level to 1.4. To change this, add `-dCompatibilityLevel=1.7` when on a Linux machine and `-dCompatibilityLevel#1.7` when on a Windows machine before `file.eps`. For example, in Linux, write

```
system(sprintf('ps2pdf -dEPSCrop -dPDFSETTINGS=/prepress -dCompatibilityLevel=1.7
  "file.eps" "file.pdf"'));
```

all on one line. The command has been split to two lines for typographical reasons. MATLAB does not allow system commands to be split over multiple lines. The parameter `-dEPSCrop` sets the size of the PDF image to that of the EPS bounding box instead of the whole page.

If you do not want to use the `system` and `sprintf` commands in MATLAB, you can do the conversion from the commandline. Write

```
ps2pdf -dEPSCrop -dPDFSETTINGS#/prepress -dCompatibilityLevel#1.7 "file.eps" "file.pdf
```

in `cmd.exe` or `Powershell` in Windows 7 or Windows 10, and in a Linux shell replace "`#`" with "`=`" in the above command.

3.3.3 Method 3: manually from the graphics window

This method is manual, and you must do this by hand *everytime* you update your image. First, save the image as an EPS file: do File→Save as and from the drop-down menu Save as type choose EPS file (*.eps), or do Export Setup, press Export and choose Save as type to be EPS file (*.eps). Then convert the EPS file to PDF from the command line as described above in method 2. When producing a PDF/A-1-compliant document, set `-dCompatibilityLevel` in the commandline conversion to 1.4; for PDF/A-2 compatibility, use 1.5, 1.6, or 1.7.

Note that changing the default font Helvetica to something else other than the base or standard fourteen fonts [10] in the Export Setup dialog box does not change it in the exported EPS file in MATLAB r2018a. Changing the fonts in MATLAB graphs makes for a completely different story and is not included here, not even as a side plot (no pun intended).

4 Epilogue

Great many hassles involving choice of fonts and embedding them in plots made using MATLAB can be avoided by saving generated numerical data in a file and plotting this data using TikZ [43] in LaTeX. Learning to do this may well be worth your while, especially if you embark on an academic career. You see, publishers have required for several years that the files of articles submitted for publishing have embedded fonts, which makes MATLAB's lethargy to support this feature seem all the more strange.

You now know that creating a PDF/A file is not simply a quick matter of converting a regular PDF file using suitable software, but requires some knowledge and appropriate actions when preparing your document. I hope you found reading this guide of sorts time well spent. So, like I said in the beginning, tell your friends and peers what they should do to successfully make their thesis in PDF/A, or tell them to take time to read this guide of sorts.

With this last remark I end this saga on what to keep in mind when preparing a document that conforms to the PDF/A standard. Good luck with successfully creating a PDF/A thesis file that can be archived safely till the end of time—or so we are led to believe.

References

All online documents retrieved on August 8, 2018.

[1] *PDF/A*. https://en.wikipedia.org/wiki/PDF/A

[2] *Aalto Thesis LaTeX Template*. https://wiki.aalto.fi/pages/viewpage.action?pageId=69900685

[3] *LaTeX—A document preparation system*. https://www.latex-project.org

[4] *Converting a file to PDF/A compliant version*. https://aaltodoc.aalto.fi/doc_public/ohjeet/aaltodoc_pdf_a.pdf

[5] *Microsoft Word*. https://en.wikipedia.org/wiki/Microsoft_Word

[6] *LibreOffice*. https://en.wikipedia.org/wiki/LibreOffice

[7] *History of the Portable Document Format (PDF)*. https://en.wikipedia.org/wiki/History_of_the_Portable_Document_Format_(PDF)

[8] *The history of PDF*. https://www.prepressure.com/pdf/basics/history

[9] *PostScript*. https://en.wikipedia.org/wiki/PostScript

[10] *PDF*. https://en.wikipedia.org/wiki/PDF

[11] *PDF. Three letters that changed the world*. https://acrobat.adobe.com/us/en/acrobat/about-adobe-pdf.html

[12] *PDF. Kolme kirjainta, jotka muuttivat maailmaa*. https://acrobat.adobe.com/fi/fi/acrobat/about-adobe-pdf.html

[13] *Adobe Acrobat*. https://en.wikipedia.org/wiki/Adobe_Acrobat

[14] *Adobe Acrobat version history*. https://en.wikipedia.org/wiki/Adobe_Acrobat_version_history

[15] *Microsoft Windows*. https://en.wikipedia.org/wiki/Microsoft_Windows

[16] *Classic Mac OS*. https://en.wikipedia.org/wiki/Classic_Mac_OS

[17] *PDF Primer*. https://www.pdf-tools.com/public/downloads/whitepapers/whitepaper-pdf-primer-en.pdf

[18] *Linux*. https://en.wikipedia.org/wiki/Linux

[19] *ISO 32000-1:2008 Document management – Portable document format – Part 1: PDF 1.7*. https://www.iso.org/standard/51502.html

[20] *Document management—Portable document format—Part 1: PDF 1.7*. https://wwwimages2.adobe.com/content/dam/acom/en/devnet/pdf/PDF32000_2008.pdf

[21] *International Organization for Standardization*. https://www.iso.org

[22] *ISO 32000-2:2017 Document management – Portable document format – Part 2: PDF 2.0*. https://www.iso.org/standard/63534.html

[23] *The history and origin of the format PDF/A*. https://www.pdf-tools.com/pdf20/en/resources/pdf-iso-standards/history-of-pdfa/

[24] *The PDF/A Standard*. https://www.pdf-tools.com/public/downloads/whitepapers/whitepaper-pdfa-standard-iso-19005-en.pdf

[25] *What are PDF tags in Acrobat 7 and why should I care?* https://acrobatusers.com/tutorials/what-are-pdf-tags-and-why-should-i-care

[26] *Tag (metadata)*. https://en.wikipedia.org/wiki/Tag_(metadata)

[27] *Accessibility: What PDF/A-1a Really Means*. https://www.pdfa.org/accessibility-%C2%96-what-pdfa-1a-really-means/

[28] *ISO 19005-1:2005 Document management – Electronic document file format for long-term preservation – Part 1: Use of PDF 1.4 (PDF/A-1)*. https://www.iso.org/standard/38920.html

[29] *PDF/A-1 vs PDF/A-2*. https://www.pdf-tools.com/pdf20/en/resources/pdf-iso-standards/pdfa-2-overview

[30] *PDF/A-1, PDF for Long-term Preservation, Use of PDF 1.4*. https://www.loc.gov/preservation/digital/formats/fdd/fdd000125.shtml

[31] *ISO 19005-2:2011 Document management – Electronic document file format for long-term preservation – Part 2: Use of ISO 32000-1 (PDF/A-2)*. https://www.iso.org/standard/50655.html

[32] *PDF/A-2, PDF for Long-term Preservation, Use of ISO 32000-1 (PDF 1.7)*. https://www.loc.gov/preservation/digital/formats/fdd/fdd000319.shtml

[33] *pdfx—PDF/X and PDF/A support for pdfTEX*. https://ctan.org/tex-archive/macros/latex/contrib/pdfx?lang=en

[34] *ISO 19005-3:2012 Document management – Electronic document file format for long-term preservation – Part 3: Use of ISO 32000-1 with support for embedded files (PDF/A-3)*. https://www.iso.org/standard/57229.html

[35] *ISO/CD 19005-4 Document management – Electronic document file format for long-term preservation – Part 4: Use of ISO 32000-2 (PDF/A-NEXT)*. https://www.iso.org/standard/71832.html

[36] *Encapsulated PostScript*. https://en.wikipedia.org/wiki/Encapsulated_PostScript.

[37] *JPEG*. https://en.wikipedia.org/wiki/JPEG.

[38] *Portable Network Graphics*. https://en.wikipedia.org/wiki/Portable_Network_Graphics.

[39] *MATLAB*. https://en.wikipedia.org/wiki/MATLAB

[40] *Acrobat Distiller Parameters*. https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/distillerparameters.pdf

[41] *ps2pdf: PostScript-to-PDF converter*. https://www.ghostscript.com/doc/current/Ps2pdf.htm

[42] *Ghostscript*. https://www.ghostscript.com/

[43] *PGF—Create PostScript and PDF graphics in TEX*. https://ctan.org/pkg/pgf