Learning LaTeX

Robert W. Schuler

January 1, 2003

Purpose

The purpose of this paper is primarily to serve as a note-to-self for the author that captures several complex procedures discovered during the Fall 2002 semester while learning and using the LaTeX typesetting environment. This paper briefly explains the LyX, XFig, and LaTeX tools running on Red Hat 7.3 and SuSE 7.0 versions of the LINUX operating system and how the author used these tools to generate some beautiful looking homework submissions for EE541, "Electromagnetic Field Theory" during the Fall 2002 semester. This paper combined with the two homework submissions provides an anecdotal case study that examines the amount of effort required to learn to use LyX to typeset mathematical material.

Overview of the tools

LINUX is a UNIX environment that utilizes a GNU is Not Unix (or simply GNU) licensed kernel developed by Linus Torvalds and several like-minded programmers. Several GNU tools and utilities run on top of the kernel to provide and entire UNIX-like operating system complete with X-Windows and several very useful office automation tools. All of this software is free for download from the Internet and the GNU license allows you to use this software without restriction. Furthermore, the source-code is available so that this software cannot be locked into a single hardware environment.

The tools of interest to this paper are TeX, LaTeX, LyX, and Xfig. LyX utilizes LaTeX, which in turn is built on TeX. These three tools combine into an environment that enables authors to typeset beautiful documents with little more effort than typing in a simple text editor. The relationship between these tools demonstrates the power of open source software. Xfig is another tool from the open source community designed to work in concert with the first three. Xfig allows authors to generate very nice looking graphical images and figures for inclusion in the typeset documents generated by LyX, LaTeX, and TeX.

The following five sub-sections of this paper discuss the cost of free software and each of the four tools in more detail.

The Cost of Free Software

This paper is being written on a Dell Inspiron 8000 laptop running Red Hat LINUX version 7.3. The laptop was not free, but all of the software was downloaded from the Internet at no charge. There is still a cost, however, and it is measured in time.

It took several hours to download the Red Hat CD-ROM images from Red Hat's web-site and a few mirror sites around the world. These images then had to be burned onto CD-R disks (5 in all). I estimate that this process took about 2 hours of my time and 6 hours of computer time spread over a three day period (due to network problems). Just based on this estimate, the cost of buying a commercial package of LINUX in a box

may well be worth the ~$70 that it costs. I actually spent $40.00 to get Red Hat 7.1, which came in a box with 4 CDs and this was actually installed on this laptop prior to upgrading to the current version.

There is also a time cost in installing and configuring the operating system after one has the CDs in hand. Installing the base system takes less than an hour. Configuring the sound system, network, modem, graphics card (for X-Windows), user accounts, and applications may take another 1-40 hours. The total time depends on how well the hardware is known to the setup scripts.

LINUX is its own operating system and requires its own partition on the hard drive. Since most personal computers come with Microsoft Windows pre-installed, this usually means that the hard drive has to be repartitioned. This adds a risk of completely erasing the hard drive. Depending on the experience of the installer this risk can be as high as 50% (my guesstimate), which means that we need to add the weighted time that it takes to rebuild the Microsoft Windows environment and to restore all the data from backups onto the total time cost for installing LINUX. Less experienced computer users tend to follow less rigorous backup procedures, which means that this weighted time cost vs. the inverse of experience follows an exponential scale rather than a linear one. In other words, a very inexperienced installer with no backups is very likely to loose data and have to rebuild the Windows operating system from the original disks (8-40 hours), while a more experienced installer might have to restore Windows from a backup (1-8 hours).

This brings the total cost of installation to:

$$
\begin{aligned}
\mathit{Total\ Installation\ Cost} &= \text{Download Time} \\
&+ \text{Disk Partition Time} \\
&+ \text{Risk Factor} \times \text{Windows Reload Time} \\
&+ \text{Installation Time} \\
&+ \text{Network Configuration Time} \\
&+ \text{X Windows Configuration Time} \\
&+ \text{Sound System Configuration Time} \\
&+ \text{User Account Setup Time} \\
&+ \text{Application Installation Time} \\
&+ \text{Application Configuration Time}
\end{aligned}
$$

Just like with Microsoft Windows, the cost of installing the operating system is just a drop in the bucket when compared to the time it takes to learn how to operate the system. Likewise, the cost of installing and configuring the application tools is negligible when compared to the cost of learning to use the tools efficiently. LyX, LaTeX, and XFig where all installed by default as part of the installation of Red Hat's LINUX distribution, so the installation time was zero for me. The rest of this paper provides a case study in the amount of time it took to learn to use these tools efficiently.

TeX

In the preface to his book, The TeXbook, Donald Knuth describes TeX as, "a new typesetting system intended for the creation of beautiful books—especially for books that contain a lot of mathematics. By preparing a manuscript in TeX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers; yet you won't need to do much more work than would be involved if you were simply typing the manuscript on an ordinary typewriter. In fact, your total work will probably be significantly less, if you consider the time int ordinarily takes to revise a typewritten manuscript, since computer text files are so easy to change and reprocess." (p. v)

In the preface of his book, Writing With TeX, Eitan M. Guaari puts it this way:

> Documents are visual channels for communicating information, and so their effectiveness depends not only on their content but also on their appearance. At the basic level the appearance refers to the legibility of the content, whereas at higher levels it involves highlighting the logical units within the content. Setting the appearance of a document is called formatting. TEX is a system for specifying the formats of documents.
>
> The TEX system offers a rich assortment of low-level features and a powerful mechanism for defining high-level features. The low-level features can be used to specify the sizes of pages, the shapes of paragraphs, the fonts for characters, and other characteristics of a similar nature. Such features are considered to be low-level constructs because they deal with physical components within a document.
>
> In contrast, high-level features relate to logical structures like chapters, sections, figures, and lists within a manuscript. With high-level features, the users need only identify the logical structures within the document; the features take it upon themselves to assign appearances to these entities. That is, high-level features encourage the users to concentrate on issues of content and organization, and shield the users from issues that deal with appearances.
>
> The creator of the system, Donald E. Knuth, has developed a comprehensive manual for TEX, The TEXbook. A popular enhancement of TEX with high-level features for writing has been provided by Leslie Lamport, and is described in LATEX: A Document Preparation System. Two enhancements of TEX for drawing and for programming, respectively, have been introduced by the author in TEX and LATEX: Drawing and Literate Programming. (p. xiii).

LATEX

In the preface of his book, TEX & LATEX: Drawing and Literate Programming, Eitan M. Guarri tells us:

> An upgrading of TEX into a high-level system is offered by LATEX. The enhancement introduces commands for identifying logical structures within the documents. With the new commands, users only need to indicate where chapters, sections, figures, lists, and other entities are located; the system assigns the necessary design formats to these entities. That is, LATEX encourages the users to concentrate on issues of content and organization; it removes formatting issues from their concern. (p. xiii).

LYX

The following is quoted directly from the on-line "The LYX User's Guide" document that came with my version of LYX.

> LYX is a program that provides a more modern approach to writing documents with a computer, an approach that breaks with the obsolete tradition of the "typewriter concept." It is designed for authors who want professional output quickly with a minimum of effort without becoming specialists in typesetting. Compared to common word processors, LYX will increase productivity a lot, since the job of typesetting is done mostly by the computer, not the author. With LYX, the author can concentrate on the contents of her writing, since the computer takes care of the look.
>
> Technically this is done by combining the comfortable interface of a word processor with the high quality of a real typesetting system. LYX uses the most popular and, in our opinion, best typesetting system available: LATEX. LATEX is used for a wide range of documents, especially in science. For example, it's difficult to find a mathematics or computer science book that is not done with LATEX. So, some people claim that its main purpose is mathematical typesetting. This isn't true. LATEX is equally good for writing letters, articles, books, or any other kind of document, and does so much better than common word processors. What prevents some people from using this powerful, free

typesetting system, one that is available for almost every computer system, is its difficult usage. With plain LaTeX, you need to enter a series of typesetting commands into the text in order to produce your document. As a result, you get no visual feedback until you feed your document to the LaTeX program. It's also difficult to read these documents before they have been printed. So, on-line editing isn't very easy. This is where LyX enters the game.

LyX provides an "almost-WYSIWYG" view of the document. "Almost" means that the line- and page-breaks are not displayed exactly as they will appear in the printed document. However, that's not really necessary, since LyX uses a separate typesetter program [here, LaTeX] to perform the final formatting of your text. While LyX contains everything it needs to be a comfortable user interface, the typesetting program contains everything necessary to format text, and do so very, very well. There's no need to reinvent the wheel, after all. Besides, computers are best at following a set of rules, and doing so repeatedly and consistently. Why should you do extra work remembering which subsection in which section in which chapter you're in, what numbering scheme your using, how big the different headings are, what font you used for the different types of headings, and so on, and so on, and so on, ... when a computer can do all of that for you? The answer is simple: you shouldn't, and with LyX, you don't have to. So, line- and page-breaks aren't your problem anymore. Remembering which number to use for the next subsection isn't your problem anymore. Recalling what font you used for all of your section headings isn't your problem anymore. With LyX, you simply choose a so-called "paragraph environment." That's it. You're done.

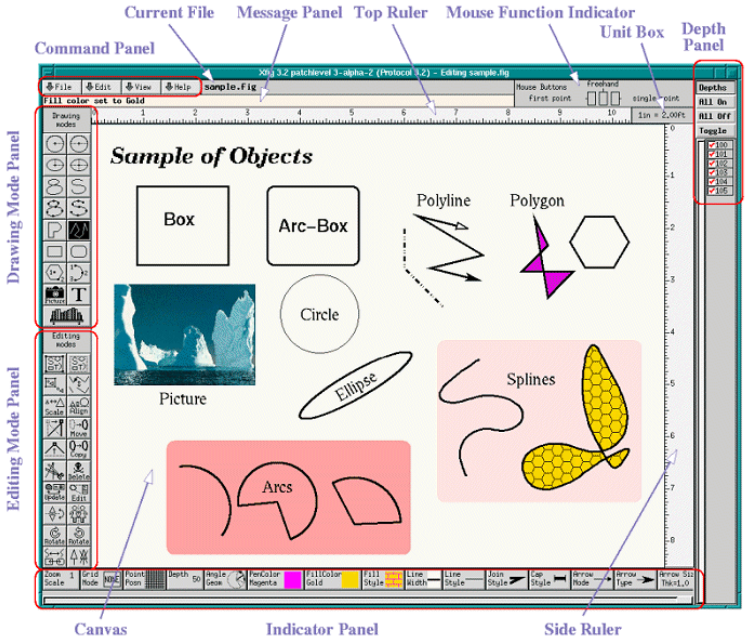
This gives you far more power than you may think. No longer do you need to scroll through a 75 page document, changing all of the section numbers because you deleted an old section. You could even pick a section, heading and all, up out of one document and drop it in a new one. LyX does the renumbering for you, adds the section to the Table of Contents, and more! Because you tell LyX [and LaTeX] what *kind* of document you're editing and what *type* of paragraph this-or-that text is, the computer can typeset it accordingly. Cut some paragraphs from an old document [say, an article] and paste them in a completely different one, [say, a letter] and LyX does the rest. Of course you can also still do some low-level formatting for fine-tuning. However, the proper way with LyX is to tell the computer what the text *is*, not what it should look like. So, we like to say that LyX gives you WYSIWYM editing [What You See Is What You *Mean*].

Some people might be tempted to call LyX a "frontend to LaTeX." This isn't quite fair. LyX performs some typesetting internally to generate the correct look on the screen. Furthermore, LyX has some extensions to LaTeX specially designed to work with the WYSIWYM-concept. So, it's actually better to call LyX a *High Level Wordprocessor* that uses LaTeX as its backend. (pp. 9-10).

Xfig

From the opening paragraphs of the "XFig User Manual" Version 3.2.3d dated May 29, 2001 we see that:

> **Xfig** is an interactive drawing tool which runs under X Window System Version 11 Release 4 (X11R4) or later, on most UNIX-compatible platforms. It is freeware, and available via anonymous ftp.
>
> See New Features and Bugs Fixed to see what is changed in this release.
>
> In **xfig**, figures may be drawn using objects such as circles, boxes, lines, spline curves, text, etc. It is also possible to import images in formats such as GIF, JPEG, EPSF (PostScript), etc. Those objects can be created, deleted, moved or modified. Attributes such as colors or line styles can be selected from various options. For text, various fonts are available. Text can also include Latin-1 characters such as "**ä**" or "**Ç**".
>
> Here is a screen image of **xfig**.



One of the many exceptional features of Xfig that makes it appropriate for this paper is the ability to directly export encapsulated post-script, which can then be directly imported into LyX for inclusion in a document. Another Xfig feature is its ability to embed LaTeX fonts that get rendered by LaTeX when the embedded postscript file is processed. Unfortunately, the Xfig and LyX documentation for how to generate and embed these kinds of files does not work for the installation of LaTeX and Xfig on Red Hat. It is safe to say that the deviation between the Xfig documentation and the LyX documentation and the actual process discovered by the author this semester is what motivated the writing of this paper.

Case Study

This paper describes the lessons learned in generating two documents. The first is titled, "EE541 Homework #1, #2, & #3" and was finished on October 5, 2002. The second is titled, "EE541 Homework #4, #5, and #6" and was finished on December 5, 2002. Both papers where written by the author during the Fall Semester of Catholic University's EE541, "Electromagnetic Field Theory" taught by Dr. Mark Mirotznick on-site at the Naval Surface Warfare Center, Carderock Division. In effect, this paper is the third case in the study since it is being generated using the same tool suite on the same computer.

While it may not be apparent to the reader, the second homework assignment was generated with a higher level of sophistication (in terms of tool usage) due to the knowledge gained during the semester. In other words, I learned a lot more about the tools I was using as I went along. Specifically, I discovered how to embed LaTeX fonts into Xfig figures after the first paper was completed. I also learned a lot more keyboard codes for entering Greek letters and mathematical symbols, which enabled me to more quickly generate parts of the second homework assignment.

Like all good tools, LyX and Xfig allow a user to start with a minimum knowledge and grow in sophistication over time. This case study presents anecdotal evidence that demonstrates the growth in sophistication experienced by the author. This anecdotal evidence is presented in the framework of a discussion of the generation of the two homework papers.

There are six individual homework assignments that where assigned during the semester. For the most-part

they where typeset in order implying that the presentation of the solution of the first problem of assignment #1 is the least sophisticated and the presentation of the last problem of assignment #6 is the most sophisticated. In fact, I consider the first problem of the first assignment (Section 2 on Page 2) to be the least sophisticated presentation. However, I consider the problems in the second homework paper with graphics containing Greek letters to be the most sophisticated presentations, so there isn't really a nice linear progression of sophistication through the problem set. Furthermore, only the final "deliverable" versions of each homework paper have been preserved, which means that the intermediate stages that each document progressed are not available for analysis.

Sophistication

I am using the term sophistication to parameterize the answers to two key questions:

- What features did the author use?
- How efficiently did the author use these features?

No attempt has been man to assign numerical values to this parameter. Never the less, it does provide a useful concept for comparing presentations of solutions within the homework papers. The first question can be answered by examining the case documents. You'll just have to take my word for the answer to the second question.

Time-line

The key points in time tat apply to this case study include:

1. Before the semester began
2. Determining if LyX could be used in a timely manner
3. Generating the first document
4. Experimenting with graphics
5. Generating the second document
6. Generating this document

Each of these points in time will be discussed in the next six subsections of the paper.

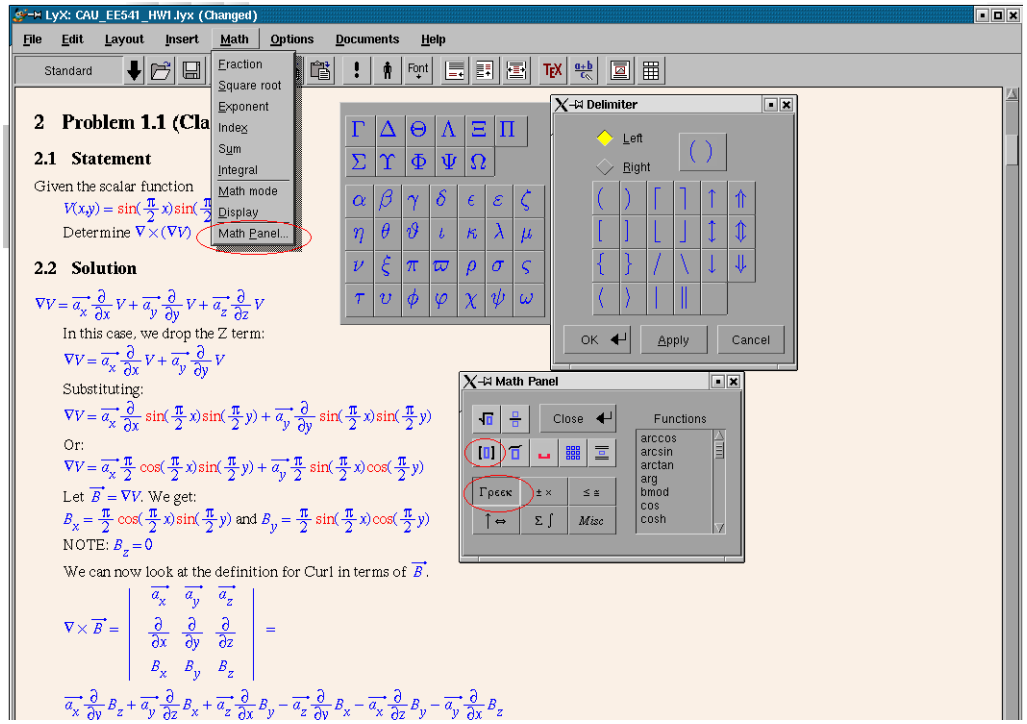
Before the Semester Began

The decision to attempt to use LyX to present mathematically intensive homework assignments is the result of a few critical elements of the author's history. Specifically, I have been aware of the existence of LaTeX as a typesetting environment for over a decade. LaTeX was used to typeset the Initial Graphics Exchange Specification (IGES), which is a document that has had a huge impact on my career path. I have also been a LINUX enthusiast ever since I discovered a SAMS book on the subject in 1994. Recently, I took a look at using LaTeX to typeset user documentation and help files for the Advanced Surface Ship Evaluation Tool (ASSET) at the Naval Surface Warfare Center, Carderock Division. As part of this exploration, I installed and briefly tested LyX in the LINUX environment on my laptop computer.

Determining if LyX Could Be Used In A Timely Manner

All of the factors mentioned in the previous section of this paper where in play when I attempted the first homework assignment of EE541. This homework assignment provided me with an excuse to "play" with LyX to see whether I could apply this tool to "real" work rather than just making up test cases. The result of this convergence of influences was that I typeset my hand written results of the first homework assignment just to see if I could. It only took me about 1 hour to work out these first two problems by hand. It took me about 1 hour each to typeset them using LyX. Based on this rough ratio 2:1, it seemed feasible to typeset my homework for the semester.

The following (not so clear) figure shows the Graphical User Interface (GUI) of LyX that allowed me to use LyX to typeset equations without needing to spend any time learning the tool.



The red circles indicate a few key parts of the figure that deserve special mention.

Math Panel

LyX's Math Panel can be opened by clicking the last option under the Math menu option. The Math Panel provides a graphical interface to commonly used equation typesetting features: Square Root, Fraction, Delimiters (brackets, braces, determinants, etc.), Decorations (arrows over vectors), Arrays, Greek Letters, Operator Symbols ($+, -, \times, \div$, etc.), Greek Letters, Function Names, etc. The Math Panel is a separate X-Window that can be parked anywhere on the desktop. In the figure, I overlaid it on the LyX window in an attempt to keep the figure size small. In practice, I dock the Math Panel off to the side of my working LyX window.

Delimiter

Clicking the icon with the square between the two square brackets on the Math Panel opens the Delimiter window. From the Delimiter window one can vary the symbol on either side of a block of delimited equation data. For example, by setting the left symbol to blank and the right symbol to a single line, one can generate the symbol used in many math textbooks for representing the insertion of a value into an equation:

$$\sin(x)\big|_{x=\frac{\pi}{2}} = 0$$

Greek Letters

Clicking on the icon labeled Gamma Rho Epsilon Epsilon Kappa (GREEK) opens a temporary window from which one can select a Greek letter. This temporary window is also shown in the previous figure.

Generating the first document

The first homework document was mostly typeset using the graphical user interface to input typesetting information. One advantage of LyX over hand-written work is that it is possible to cut and paste large equations, so that each progressive step in algebra can be generated by updating a copy of the previous line. This turned out to be incredibly useful in solving Problem 3.5 of Balanis (Section 7, pages 9-12 of the first homework document). I was unable to successfully keep the algebra straight in my hand-written attempt to derive this proof, but by using LyX to help organize my work, I successfully presented the proof.

Problem 1.1

As was described in the previous section of this paper, the problem presented in section 2 of homework #1 was typeset first. The basic layout of restating the problem then presenting the solution was adopted from the very beginning. The arrows decorating vector variables are not standard for unit direction vectors. I had not taken time to explore all the decorations available at the time I typeset section 2.

Features Used

- Decorated Vectors
- Function Names
- Fractions, Subscripts, and Superscripts
- Use of ∇, ∂, Ψ, and $\times$
- Use of Delimiters for sin, cos, and the determinant
- In-line and display mode math equations

Efficiency All of the features where selected by way of the math panel and a lot of cut & paste.

Problem 1.2

The most notable additional feature of the second typeset problem is the use of an Embedded PostScript (EPS) figure showing the boundary geometry. The figure was generated in Xfig and saved as an EPS file. Adding the figure was as simple as clicking Insert | Figure menu option, then browsing to find the EPS file.

I was also able to label the integrals with subscripts and without subscripts and to use the contour integral. As of this writing, I still haven't discovered how to present a proper surface integral symbol (two integral symbols with a single circle covering both).

In this problem, I started using the hat to decorate unit direction vectors.

Features Used

- Decorated Vectors
- Embedded PostScript Figure
- Fractions, Subscripts, and Superscripts
- Integral Symbols (with and without subscripts and contour marks) Volume and Surface
- Use of special symbols (NOTE: I used capital Phi Φ instead of lowercase Phi ϕ)
- Use of Delimiters for grouping and value substitution
- Aligned equations (all the equals signs line up)
- In line and display mode math equations

Efficiency All of the features where selected by way of the math panel and a lot of cut & paste. The figure was rescaled in Xfig a few times before I got the hang of scaling. The rulers in Xfig mark out actual inches on the final printed page! This is a feature that just doesn't exist in Microsoft Word.

Problem 2.1

There are not really any noteworthy changes in features or efficiencies to report for this problem, however, the use of Depth to allow axis lines to show through the shaded surfaces is a trick worth recording. Simply put, the axis lines where moved from the default depth of 50 to a more shallow depth of 40, while the polygons representing the surfaces where moved to a depth of 60.

Problem 2.2

In problem 2.2, I used a table to label equations. This was also the first problem where I used the idea of cut, paste, & modify to help with the algebra.

Problem 2.3

There are not really any noteworthy changes in features or efficiencies to report for this problem.

Problem 2.4

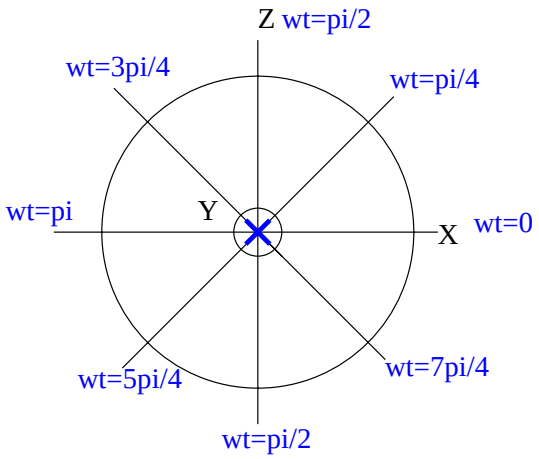
As was mentioned previously, the use of LyX to assist in the algebra was essential to my being able to work this problem! I tried several times to work all these details with pen and paper, and never succeeded. It was during the presentation of this proof that I developed the technique of displaying multi-line equations by aligning subsequent plus and minus signs with the opening equals sign.

Problems 3.1 through 3.7

By the time I got to assignment #3, I had obtained a fair level of efficiency in using LyX to typeset this kind of work. There are not too many new features introduced in these problems. I am still wrestling with finding the right symbol for approximately equal ($\doteq$, $\cong$, $\approx$, or $\simeq$), but I would argue that this is primarily a question of style. As far as efficiency is concerned, I started learning a few keyboard shortcuts for keying in Greek letters by name and generating basic symbols like \sqrt $\surd$ and \frac without requiring the Math Panel. Using ^ to start a superscript and _ to start a subscript are also very useful shortcuts.

Experimenting with Graphics

The figure at the end of problem 3.7 (and repeated below for reference) contains eight blue text strings similar to "wt=pi/2", which I would have preferred to present in a nicer looking format. For example, $\omega t = \frac{\pi}{2}$. It is undesirable to revert to Roman letters in figures when the rest of the document has such beautiful Greek letters. The pursuit of resolving this conflict took a little bit of effort on my part, and it can be argued that this is the motivation for writing this paper.



It would be better to use LaTeX fonts for the text in this figure. To accomplish this, one can start by reviewing the XFig documentation along with the documentation in LyX. The LyX User's Guide that came with my version of LyX has a whole section dedicated to this subject:

`XFig` and LyX

> One obvious question is "how would I create the figures?" Fortunately, the answer is included in most Linux and/or LaTeX distributions. `XFig` is a powerful and highly recommended drawing tool. If you want to include figures that you have created with `XFig` there are several ways. We recommend the following:
>
> Export the figure as Encapsulated PostScript®. This could be very easy included into LyX as described in the previous sections. The great advantage of this way is, that you have the full power of PostScript® available. That means Bezier curves, colors, all line thicknesses and many more. If you have inserted text into your fig-document this will be printed with PostScript® fonts, which is OK. The figure can be manipulated like any other EPS figure, as described above.
>
> The only disadvantage is that you cannot create formulas as PostScript® text except by

hand. If you also need formulas or simple exponents or indices in your figure, the next way is recommended.

Export the figure as LaTeX. This is just as easy include into LyX, with the advantage that you may use all LaTeX commands within the text inside XFig. Therefore you have to set the *special flag* for text in XFig. This is automatically if you invoke XFig with `xfig -specialtext`. If this is done and you have also chosen a LaTeX font you may simply write "H_2" in `XFig.` If you export this figure as LaTeX and include it in LyX with Insert ▷ Include File (see description in *Extended Features*) this text will appear as H_2.

The disadvantage of this way is that the graphical power of LaTeX isn't as strong as PostScript®. You cannot use all thicknesses of lines and, more annoyingly, not all slopes. This is why we recommend the third way for more complex figures.

Export the figure as LaTeX/PostScript® combined. Then `XFig [transfig, really]` will generate two files:

the PostScript® part `foo.pstex`, that contains all painting.
the LaTeX part `foo.pstex_t`, that contains all text and a link to the PostScript® part.

Then you just have to include the LaTeX part as described above. This will automatically include the PostScript® part, too.[1] This way you have the full PostScript® and LaTeX power combined except for the possibility to scale the figure after creating. So if you want scalable pictures the PostScript® format is your only choice. Another little advantage of letting LaTeX typeset the font is that the same font will appear in your figures as in your text, which looks a little nicer.

Even after following the advice listed in the footnote, I was still getting errors when trying to incorporate the combined files. NOTE: On my system, the dvips.def file was found at /usr/*share*/texmf/tex/latex/graphics, and I modified two lines as follows:

\def\Gin@extensions{.eps,.ps,*.pstex*,.eps.gz,.ps.gz,.eps.Z}
\@namedef{Gin@rule@.ps_tex}#1{{eps}{ps_tex}{#1}}

Further guidance is found in the documentation accompanying my version of Xfig.

XFig Help Documentation

In your LaTeX preamble (the part that precedes your \begin{document} statement) place the following lines:

\input{psfig}

The psfig package does not appear to be part of the teTeX distribution on my computer. Instead, a package named psfrag is used. Before making the changes suggested in the footnote of the LyX documentation, I was able to use the psfrag package by adding the line \usepackage{psfrag} to the Layout | LaTeX Preamble... menu option. After making the change suggested in the footnote, I was able to input the combined output files from Xfig without adjusting the preamble.

[1] If you get an error like "unknown graphics extension pstex" you have to declare this graphic extensions. I think this is a transfig bug that occurs with LaTeX 2_ε. Simply add a line like
`\@namedef{Gin@rule@ps_tex}#1{{eps}{ps_tex}{#1}}`
in the file `/usr/lib/texmf/tex/latex/graphics/dvips.def`. Then add `pstex` to the extension:
\def\Gin@extensions{eps, ps, pstex, eps.gz, ps.gz, eps=2EZ}
This should fix the whole thing. Alternatively you may export the postscript part as `foo.eps` and change the LaTeX part `foo.pstex_t` manually. But this is annoying.

TYPE C - PostScript/LaTeX format

You can draw any lines or curves when using this format. In this type of export, LaTeX strings are permitted you also have the PostScript fonts available to you. Therefore you can type in strings such as

$\int_0^9 f(x) dx$

and they will be processed by LaTeX. Note that you must set Special flag ON for texts you want to output in LaTeX, and set it OFF for texts you want to output in PostScript.

To export in this format, simply choose the Export... from the File menu of xfig. Then select Combined PostScript/LaTeX (both part) as the language to export. This will create a file with a .pstex_t extension which you can then call directly into your LaTeX document, and a file with a .pstex extension which contains PostScript part of the figure. The .pstex_t file automatically calls the .pstex file and you do not need to include it explicitly in your tex file. To include your figure just use something similar to this:

\begin{figure}[htbp] \begin{center} \input{yourfigure.pstex_t} \caption{Your figure} \label{figure:example} \end{center} \end{figure}

N.B. You might want to edit the .pstex_t files created by xfig. When it refers to the other file (.pstex) it automatically gives the path specification to the .pstex file. This can be an inconvenience if you move your files to another directory because your LaTeX processing will fail. I personally prefer to remove the full path specification and only keep the filename.

What Actually Works

dvips.def As I mentioned previously, by modifying the extensions list and adding a new nameddef in the /usr/*share*/texmf/tex/latex/graphics/dvips.def file, I was able to successfully input *.pstex_t files into my documents:

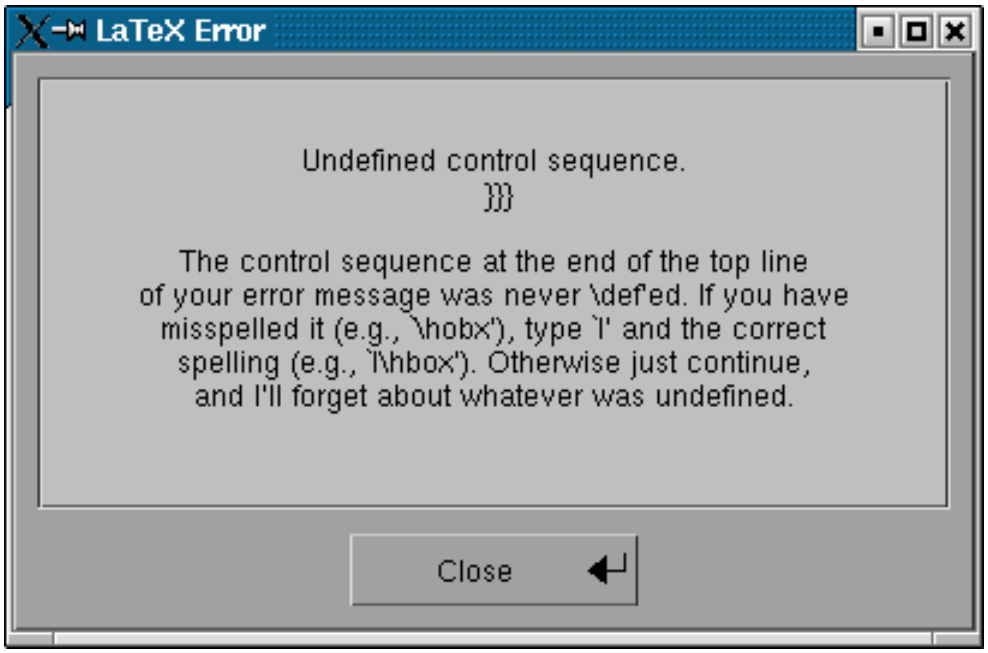
\def\Gin@extensions{.eps,.ps,.*pstex*,.eps.gz,.ps.gz,.eps.Z}

@namedef{Gin@rule@.ps_tex}#1{{eps}{ps_tex}{#1}}

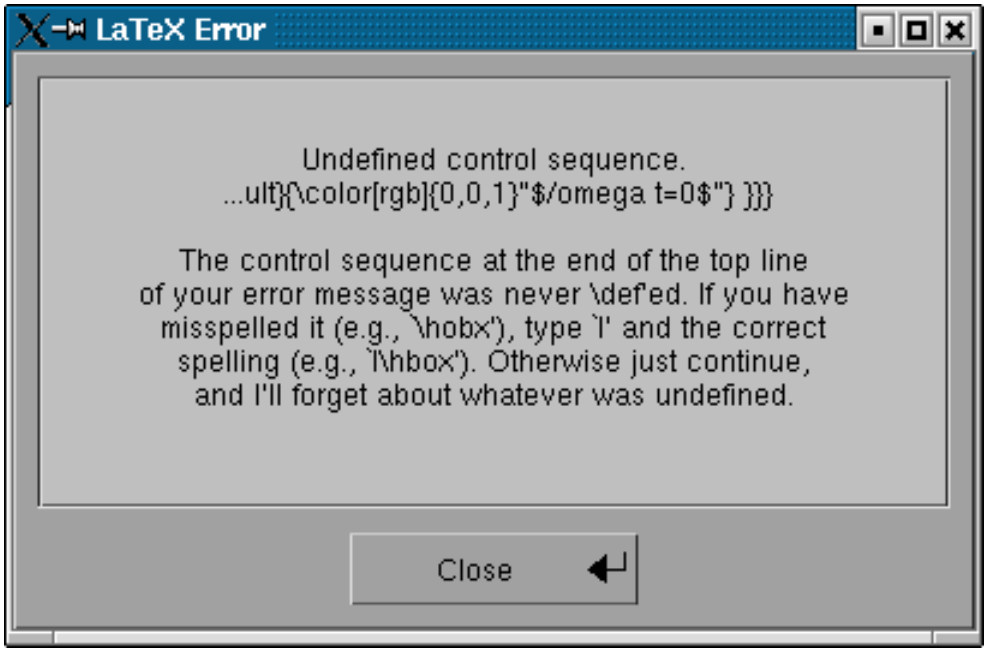
This approach was used in generating the paper you are currently reading.

psfrag A successful alternative was to add the line \usepackage{psfrag} to the LyX Layout | LaTeX Preamble... menu screen. This is the actual approach used in generating the second homework paper.

Errors I still get errors with the native output from Xfig. The first error seems to mask the second, but these two errors have been consistent across all attempts. The first error seems to relate to LyX's inability to handle a % character to extend a line:

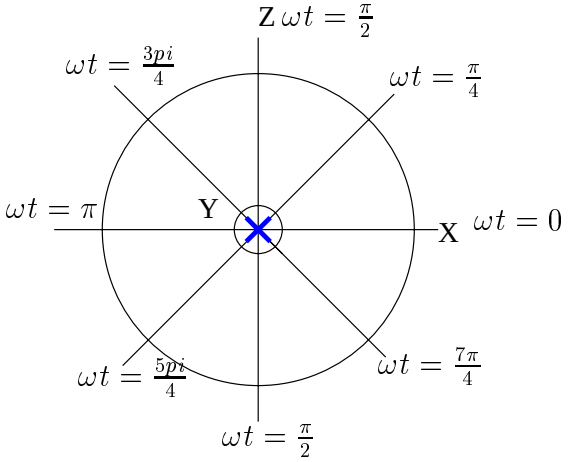


This error is corrected by editing the *.pstex_t file and deleting the % and the following carriage return for each line that contains the "special text" from Xfig. Once this error is resolved, all the "special text" lines complain with a different error:



It seems that I have not configured LyX/LaTeX to recognize the \color directive. My current solution to this problem is to simply remove the string \color{[rgb]{0,0,1} from each line in the *.pstex_t file.

The resulting update to the figure from problem 3.7 of the first homework paper now looks like:



One last observation that seems like it belongs at this point in the paper is that LyX didn't seem to always recompile these input figures every time the File | View PostScript menu option was exercised. To address this issue, I have developed the habit of deleting the figures, then re-inputting them every time I make a change in Xfig.

Generating the Second Document

The only new features used in the second homework paper are related to graphics. All other improvement in sophistication in the second paper are attributed to efficiency, and most of these improvements are due to learning and using more and more keyboard short-cuts instead of the Math Panel. In addition to adding graphics from Xfig that contain LaTeX fonts and equations, I also discovered that one can input a JPEG or PNG file into an Xfig picture object; scale it; and save it as embedded PostScript (EPS). This feature was used to copy figures directly from the textbook into problems 4.5(Section 6 on Page 11) and 4.6 (Section 7 on Page 13)

Generating this Document

This document utilizes all the features found in the two homework papers with the notable exceptions that there are not many equations and there are no tables. This is the first document that I have cut & pasted from other documents. Large sections of this paper where copied from other electronic sources within my laptop.

Conclusions

Looking at the second homework paper and this paper, I can claim to be on an entry level of sophistication (see my definition of sophistication on page 6) in using LyX to typeset documents. Other features to be pursued include Indexes, Tables of Contents, Figures, Formulas, and Tables, Figure captions, and BiBTeX bibliographies. I spent about the same amount of time and effort this semester to learn these tools as I would have spent if I had taken a second three credit hour course. I expect to use these tools in all of my course work at Catholic University and consider my investment of time and effort to learn LyX to be very worthwhile.