A way to ensure the future of TEX: make its use easier on low-cost machines*

Michel Lavaud

C.N.R.S.
GREMI, Université d'Orléans, 45067 Orléans Cedex, FRANCE
lavaud@avion.univ-orleans.fr

Abstract

The PC is the cheapest computer and the most widespread one in the scientific community. Faced with commercial scientific word-processors that are improving steadily in wrong directions, it is urgent to make the use of TEX easier on the PC, to ensure its future and avoid costly dead-ends to researchers. We have designed a program, ASTEX, that allows to create easily multi-author scientific documents in TEX or LATEX on PCs. It provides an on-line hypertext help and a multi-level assistance in typing LATEX code. It allows to display and modify very easily the structure of a document, to archive and retrieve files related to it, to perform numerical and formal computations from the document and include automatically the results, to create LATEX tables from worksheets or databases of formulas. It processes electronic mail and files sent by list servers for a better use of information and eases considerably the use of anonymous ftp and archie servers by local archiving of selected informations.

Keywords: TEX, LATEX, back-end, front-end, scientific word-processor, hypertext, tree, link, multi-author document, file manager, numerical computation, formal computation, worksheet, database, e-mail, PC, notebook, OS/2.

1 Introduction

During the last years, there appeared several WYSIWYG scientific word-processors on PCs that allow to type easily short texts including mathematical formulas, and can give very nice-looking outputs.

These programs are not adapted to scientific research. Indeed, formulas are stored either as proprietary code incompatible with other software, other machines and preceding versions of the program itself, or they are stored as bitmap images, which gives rise to prohibitively large files even for small articles. This makes collaboration by electronic means very difficult, maintenance of personal scientific documents over the life-cycle of a research project very problematic, coupling to scientific software such as Maple or Mathematica almost impossible and building of databases of scientific documents with formula searching capabilities completely hopeless.

Nevertheless, they have a high seducing power and it is very important and urgent to make the use of TEX on PCs as easy as possible. This is very important because the PC is the cheapest machine and, by far, the most widespread (80 millions in use today, 20 millions more in 1991). This is also very urgent because the other products are getting better and better, and it might happen that they become a de facto standard on small machines, despite their considerable long-term drawbacks, if *easy-to-use* TEX-based Scientific Word-Processors are not available rapidly on PCs.

In section 2, we define what we mean by a TEX-based SWP (Scientific Word-Processor). In section 3 we argue that, in the future, SWPs will be used by researchers, no more by secretaries, so that "easy-to-use" TEX-based SWPs will have to ease all the tasks required to *create* scientific documents, not only to *type* documents already created on paper. This will have deep consequences on the structure of data manipulated by these programs, on the structure of the programs themselves, and probably also on the way of entering formulas. In section 4, we argue further that they must run on PCs, to be accessible to everybody, and to be usable on future pocket-size notepads. We propose to call this new kind of programs adapted to research work, "Scientific Document-Processors".

We have designed a prototype of Scientific Document-Processor, ASTEX. Version 1 was presented at the preceding EuroTEX conference in Paris [5]. In section 5, we recall the main possibilities of version 1, then we describe in some detail improvements brought in by version 2.

The readers who want to have at once an *overview* of the possibilities of ASTEX, can go directly to section 5.2.

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia.

2 What is a TEX-based Scientific Word-Processor?

TEX is not a Scientific Word-Processor. It is a typesetting *language* plus a *compiler* of this language. In a much cited conference[12], Donald Knuth said:

> [...] I expect that there will be a continual development of "front ends" to TEX and Metafont, as well as a continual development of "back ends" of device drivers that operate on the output of the systems [...]

We define a *TEX-based SWP* as any set of these three ingredients (front-end + TEX compiler + back-end), made to cooperate together.

2.1 Back-ends to TEX

In the early eighties, it was almost impossible to use TEX if you were not close to a big computing center[1]. The problem of building back-ends to TEX for cheap machines was crucial.

Now there are several Public Domain back-ends to TEX of professional-quality that run on PC: emTEX by E. Mattes, dvips by T. Rockiki and Ghostscript by L.P. Deutsch, among others. With these programs, it is possible to create documents integrating mathematical and chemical formulas with graphics and to output them on low-cost devices (VGA screen, dot-matrix or ink printers and cheap laser printers). Color Postscript illustrations and color texts in TEX [2] can be included and output on a color screen or a color-ink printer.

The problem of building good back-ends to TEX on PC can be considered as largely solved now[2]: Anybody can produce scientific documents of *much better quality* with TEX than with ordinary word-processors such as Word or Wordperfect – and at a *much lower cost* since the same hardware can be used and the whole software is Public Domain.

2.2 Front-ends to TEX

The availability of powerful Public Domain back-ends to TEX on PC means that it is *possible* to produce high-quality documents with TEX on cheap systems. This does not mean however that it is *easy*. Making TEX easy to use is a problem of designing a good front-end, and coupling it to TEX compiler and to existing back-ends in an efficient way.

While it was *necessary* to build powerful back-ends to be able to use TEX, there is no such necessity for front-ends. Indeed, any text editor is a valid front-end, even the simplest ones (e.g. `edlin`, or the built-in editor of dBase 3, etc...). This has the well-known considerable advantage that it is possible to include mathematical formulas in any non-scientific software (e.g. database managers). But this has the disadvantage that there has been no strong pressure to build front-ends that ease the use of TEX, as long as existing commercial products gave so poor outputs as compared to TEX.

2.3 An example of a TEX-based SWP: emacs customized for TEX

The most well-known and widely used front-end to TEX on workstations is the Public Domain editor GNU `emacs` of R. Stallmann. The PD macro-package `auc-tex` of K. Thorup transforms `emacs` into a TEX-based SWP: It lets you compile a TEX or LATEX file and preview or print it from inside `emacs`. It let's you browse through the errors TEX reported, while it moves the cursor directly to the reported error, and displays some documentation for that particular error. This works even when the document is spread over several files. It automatically indents LATEX source when typing, or indent and format an entire document. It has also a special outline feature, which helps 'getting the overview' of a document.

`emacs` has been very recently adapted to run on the PC under OS/2. This restricts its availability to the newest 80386-based machines with lots of memory and hard-disks with large capacity, but it is nevertheless a big step forward.

3 What must be required of the SWPs of to-morrow?

3.1 They must be designed for researchers

A few years ago, almost all researchers wrote their scientific articles by hand, and had them typed by secretaries. In the future, all researchers will type themselves their articles directly to screen, as soon as adequate SWPs will be available, because this will allow them dramatic gains in time.

This is obvious if we recall the evolution in numerical computations: about twenty years ago, researchers used to write their computer programs on cross-ruled sheets of paper and had them typed on punched cards by specialized typists. Then they gave the pack of punched cards to an operator and waited until the operator bring back the results. As soon as teletype terminals became available, each researcher typed himself his own programs directly to screen and submitted them himself to the computer: although he did alone the work of three persons, he was able to gain much time because typing and submitting his programs himself eliminated *waiting delays*, and typing programs directly to screen

[1] When I tested TEX for the first time, I had to submit files to a distant computing center through a modem, with no possibility to preview the result locally. The only way to track errors was to wait for two or three days until the printed output arrived by regular surface-mail...

[2] This does not mean that existing back-ends cannot be improved any further. It would be very nice, e.g., to have a utility to do scrolling and zooming on Postscript code. Up to now, this is possible only in the X-Window environment with `ghostview`.

(i.e. without writing them first by hand) avoided him useless *duplication of tasks*.

3.2 They must allow to CREATE – not only TYPE – scientific documents

Almost all scientific word-processors are oriented towards *typing* scientific texts, not *creating* them, i.e. they are adapted to the way of working of secretaries, not researchers. With Word and Wordperfect for example, the Table of Contents is produced *from* the typed document, as a by-product of the typing process. Modifying the structure of large multi-author documents is very long and must be done by loading many files, selecting blocks of text and moving them from one place to another by cut and paste.

This approach is quite acceptable if the researcher writes his document by hand and then gives it to a secretary for typing: in general, it is quite close to its final form and only few modifications of structure are necessary. But this approach is very clumsy if he types the document himself. Indeed, the correct way of writing a document from the scratch is to build *first* the table of contents and *then* fill in the different sections with text, not the reverse.

A SWP adapted to the way of working of a researcher must allow to *display* and *modify* the structure of the document very easily. This ensures that even very long multi-author documents will be coherent and logically organized, and that they can be easily reorganized at any time, to improve their clarity and coherence.

This has deep consequences on the data types that are manipulated: a document is no more a long sequence of (possibly meaningless) characters and formatting codes typed in one (or a few) window. It is a *tree* whose leaves are coherent blocks of information of various nature (paragraph of text, worksheet of numerical results…).

3.3 They must allow to perform research tasks from the document

The researcher has also many other tasks to perform, while creating his document, that a secretary would not have to do. For example, he may have to manage hundreds of files related to the document (chapters, computer programs, numerical results, e-mail, illustrations…), created by several people on different media. He may have to perform *numerical* or *formal computations* and merge results into his document. He may want to manage the results into *worksheets* or *databases*, and include excerpts into tables. This can be very long if it is done by cut and paste, and updating can be a nightmare if tables are interdependent.

A good SWP must ease all these tasks and allow the researcher to perform them *from the document* and *automatically*, because this can save him much time. On Unix workstations, a great deal of work has already been done to integrate tools useful for research into GNU `emacs`. For example, it has been customized to include automatically Mathematica outputs in LATEX into the text [3]. But much remains to be done, in particular as concerns integrating worksheets and databases into documents.

3.4 They must be TEX-based

Since future SWPs will be used mainly by researchers, they *must* be based on TEX, as suggested by the preceding remark about the possibility to get outputs of scientific software in LATEX. Indeed, because it is of highest quality and is in Public Domain, TEX is now *THE privileged common language* of the international scientific community, for communication of scientific results by electronic means, as *English* is the privileged language for communication of scientific results by usual means (voice and paper). Many other reasons can be given, and has been given elsewhere, that are more or less elaborations on or consequences of these two fundamental reasons. Let us enumerate some of them here:

- TEX is sufficiently *powerful* to deal with every complexity likely to be encountered in mathematical typesetting [10];
- It gives superb outputs;
- It is easily understandable by everybody (instructions are written in plain English, not undocumented numerical codes);
- It is *extensible*, as natural languages: missing concepts / words can be created at will. In computer science terms, it is *programmable*: what is not included can be added by writing new procedures / macros;
- It is in the *Public Domain*: you have not to pay hundreds of dollars to software corporations, as a preamble to "speak electronically" with your colleagues or write your scientific articles or read scientific files sent by colleagues: what research would become if researchers had to pay to English or US governments, to be able to speak in English? *Freedom of communication* is a sine qua non condition for research, as freedom of speach and writing is for democracy;
- It is *transportable*, i.e. independent of the platform and available on all of them;
- It is *stable*, i.e. it does not vary every six months, at the mercy of new versions;
- Many scientific programs can give *outputs in LATEX* (Maple, Mathematica, Macsyma, Reduce…);
- Huge *databases* of scientific articles exist in TEX: e.g. all of Mathematical Reviews going back to 1959 (about one million records) is stored online in TEX format in the MathSci database[11];
- It is used by the biggest scientific book editors;
- Many scientific journals accept compuscripts in TEX and provide style files;
- It travels very well on networks;
- Maintenance is *world-wide*. There are many discussion lists in many languages to help solving problems.

3.5 They ought to have a WYSIWYG equation editor capable of importing TeX formulas

Several commercial scientific word-processors and equation editors on PCs allow to type mathematical equations in WYSIWYG mode, and export them into TeX or LaTeX files (Mathor, Mathdesign, PreTeX, Grif...).

They can ease typing *portions* of documents with many formulas in TeX. But they cannot be considered as satisfactory TeX-based SWPs, nor can they be used as WYSIWYG equation editors in a TeX-based SWP. Indeed, exporting TeX equations is the easiest part of the story. To be really satisfactory, a WYSIWYG scientific word-processor ought also to be able to *modify* – and thus to *import* – TeX formulas[3]. And this is very difficult because of the macro capacity of TeX: this implies that the import module contains most of the capabilities of a TeX compiler. To give a simple example, if the greek letter `\gamma` is redefined somewhere in the file or in anyone of the inputted files as the macro `\g`, the import module must be able to detect it. In fact, to be completely satisfactory, the import module ought to be able to import *any TeX and LaTeX file*. Indeed, more and more scientific journals provide their own style files, with a bunch of special commands to write compuscripts that are submitted in TeX. For the compuscript to be consistent with this style file, the SWP must be able to "understand" the style file, i.e. to import it.

This is quite comparable to the situation for Postscript: virtually any software now is able to export in Postscript. But there are, up to now, only a few programs that can import *any* Postscript file. Indeed, building such a software amounts largely to build (the software part of) a Postscript printer. Moreover, no program is able to import any Postcript file *and* modify it interactively. The graphics programs that allow to modify interactively Postscript code, such as Adobe Illustrator or Corel Draw, do it on a very limited subset of Postscript commands.

For the time being, a satisfactory compromise would be an equation editor able to import TeX formulas containing only *standard* TeX, LaTeX and AMSTeX instructions (i.e. `\g` would NOT represent the letter γ, but just the two characters \ and g). This would be a useful ingredient to build TeX-based SWP with WYSIWYG construction of formulas.

4 TeX-based SWPs must be available on PCs

4.1 Price

The PD and commercial implementations of TeX on workstations (GNU `emacs`, Publisher, Decwrite) are very nice. However, workstations are very expensive as compared to PCs and most students cannot afford buying color workstations and software to run on it[4]. Moreover, Unix is not for the freshman (except in computer science), DOS or OS/2 are more adapted.

Since students of today are the scientists of to-morrow, it is very important that TeX can be used in as good conditions as possible on the *cheapest machine* that is easily accessible to *all* of them, i.e. the PC. This is all the more important for East-European and Developing countries, because only this type of machine is available.

4.2 Transportability

Another considerable drawback with workstations is that most of them cannot be moved, either because of their large and heavy screen (although a Sparc notebook appeared recently), or because they are bound to a network (if you disconnect the machine you use at work and bring it home with a lorry, the odds are great that you will be flamed by colleagues who use a software installed on your machine with a floating license...).

PCs are not subject to these constraints and evolution is towards *portability*. Market studies have shown that next year, about 70 millions of compatible PCs will sell, half of them being *notebooks* and *pocket-size* machines. Among these, many will have a *sensitive screen* with the capacity to recognize words hand-written on screen. It seems reasonable to think that, in the near future, hand-written formulas could also be recognized: problems are still tough but certainly not insurmountable. It would be highly desirable that formulas be translated into TeX and not into proprietary code of Word, Mathdesign or whatever commercial SWP in vogue at this time. This requires that TeX-based SWP must be available on these machines, to exploit this facility.

Within a few years, high capacity flash memories will replace hard disks and PCs will have the size and thickness of a few sheets of paper. Outputs will be done to printers or networks via infrared emitters plugged into parallel port or PCMCIA interface of the notepad. One can imagine it will be possible to hand-write mathematical formulas and scientific articles on the sensitive screen of the notepad, and to output them automatically in TeX, a few seconds or minutes later, either on a local

[3] Any SWP may modify its *own* outputs in TeX, by storing its internal description of a formula into TeX comments, and exporting the comments with the TeX formula. The modification is made by importing the commented part of the formula. But this trick does not allow to modify outputs of *other* SWPs.

[4] These last years, the prices of workstations had decreased so much that it seemed probable that cheap workstations would replace PCs very soon. However, prices of PCs have decreased even more (40% in 1991) and, since the outburst of cheap notebooks last year and the new release of OS/2 this year, it seems very likely that PCs and workstations will continue to develop in parallel for a long time, as complementary machines.

printer or on the one of a foreign colleague, thousands of miles away!

4.3 They must run under OS/2 for best performance

It has been advocated recently that
'*workstations of the* SUN SPARC *class are necessary for TEX. The speed is necessary to process text of more than a few pages and a large screen with good resolution* (1100×950) *is just sufficient for the previewer*' [1].

In fact, a workstation is indispensable only when pure computation power is required, as for imaging or intensive numerical computing. When doing word-processing, the machine is idle most of the time: it just loops, waiting for you to type characters. What is required is not the speed of a SPARC machine but its *multitasking* and *interprocess communication* capabilities, to make the three components of a TEX-based SWP (front-end + TEX compiler + back-end) run concurrently and cooperate, without having to reload them each time they are called to work. This is the real clue for cutting down the waiting delays for compiling, previewing and printing that plague the use of TEX on PCs running DOS[5]. With the new release of OS/2 2.0 and the new capabilities of the emTEX package, these problems will be solved very soon [9]). As concerns the size of the screen, what is required is not a large one, but a previewer with a *zoom* and *scroll* function (as in CAD software, for example). Indeed, text written in the smallest size printed by TEX is unreadable even with the resolution of a workstation, when a whole page is displayed. These functions exist in the emTEX previewers for DOS and OS/2.

In short, the problems that have been encountered in the past when using TEX on the PC came mainly from the limitations of DOS and old previewers, not from the hardware. Using TEX on a 80386 notebook with OS/2 and emTEX is certainly a better choice now than using it on a workstation, because this cuts price but not performance, and the machine can be carried anywhere.

5 AˢTEX version 2

We have developed a program, AˢTEX[6] that fulfills the requirements defined in the preceding sections (except for the coupling to a WYSIWYG equation editor, that do not exist up to now). It provides an easy-to-use TEX-based SWP, adapted to scientific research. Version 1 was presented at the preceding EuroTEX conference in Paris [5] and other places [6]–[7]. Version 2, that is currently in beta-test, brings in many improvements and adds new possibilities [8] that make it more powerful than Windows 3.1 and related programs (although their use is of course not excluded).

It is based on the kernel of public-domain back-ends to TEX already mentioned, and on a commercial program, *Framework* (from Borland) that is used as a basis to construct the front-end. On 80386 machines, it can be complemented by *Desqview* (from Quarterdeck) or *OS/2* 2.0 (from IBM).

In the sequel, we recall a few definitions and we give an overview of AˢTEX's possibilities. Then, we describe the new possibilities of version 2. For a description of those already present in version 1 (in particular the hypertext file manager and the mechanism to export selectively subsets of objects), we refer to Refs. [5]–[8] for details.

5.1 Definitions

For AˢTEX, a document is a *set of files related logically* and accessible from one of them (the master file) by loading into linked windows. The set of files has a *tree structure*, and each file is itself a tree whose leafs are *objects* of various nature (linked windows, texts, databases, worksheets, graphics, computer programs. . .). The files may be on different media and be created by various people on a network (see [5]).

External tasks required by TEX or by research work related to the document, can be run from a customizable toolbox, in windows that are created automatically. On 80386-based machines, windows are displayed *simultaneously*, each occupying a part of the screen ('space-windowing'). On 8088- or 80286-based machines, windows are displayed *alternatively*, each one occupying the whole screen ('time-windowing'). The important point is that AˢTEX works on *all PCs*, in the same way modulo this distinction between space- and time-windowing.

5.2 Overview of AˢTEX's possibilities

- *Hypertext file manager*:
  - Immediate access to thousands of files through hierarchy of explicit titles.
  - Easy modification of the structure of very big multi-author documents.
- *Scientific computations*:
  - Numerical (e.g. Fortran): compilation / execution run directly from text of the document.
  - Formal (e.g. Maple): results automatically included in text, worksheets, databases.
  - Live links to data files.
  - Interdependent worksheets.
- *Scientific text processing*:
  - Mathematical and chemical formulas displayed with a single keystroke.
  - Preprocessor of LATEX structure.
  - Creation of LATEX tables from worksheets or databases of formulas.

[5] Grouping the three components of a TEX-based SWP into one huge software would be unrealistic.
[6] AˢTEX stands for Assisted TEX.

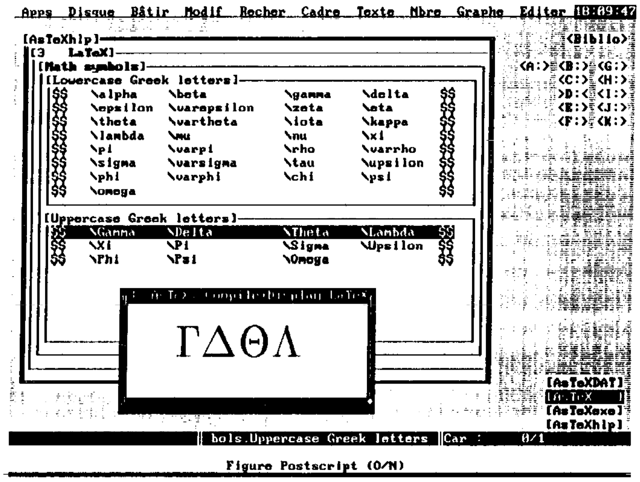

Figure 1: *LaTeX symbols can be displayed with a keystroke*

- *Electronic mail*:
  - Hypertext archiving of messages.
  - Automatic extraction of messages from files issued from discussion lists.
  - Local archiving of informations about ftp and archie servers to speed up connections.
- *Hypertext help* for AsTeX, LaTeX, emTeX, Ghostscript, graphs used in physics . . .

5.3 Hypertext file manager

AsTeX's hypertext file manager (cf. Ref. [5]) has been improved to include inheritance of relative and absolute links of parent windows. This allows easy transfer of complete sub-hierachies of objects from a machine to another (e.g. from a desktop to a notebook and reversely), for later improvement at home, in a public library etc. . .

5.4 Creating LaTeX files

AsTeX helps creating LaTeX files at *export* time and at *typing* time.

Sectioning commands

With AsTeX, no sectioning commands have to be written: they are generated automatically at export time. Any subset of selected objects is exported into a LaTeX file with correct sectioning commands, according to the relative positions of the objects in the tree of the document. The export mechanism has been described in detail in [5]. As a complement to this facility, version 2 inhibits the typing of sectioning commands. For example, if `\chapter` is typed, it is automatically erased and the message "Instruction `\chapter` forbidden" is displayed at the bottom of the screen.

This function implements, at the front-end level, an important function of SGML parsers: eliminate ill-structured documents, with a `\chapter` command inside a `\paragraph` command for example. This kind of error is not detected by LaTeX.

Hypertext help

A very common criticism addressed to TeX by new users is the difficulty to get informations on it [13]. Many recent word-processors have on-line help and even sometimes on-line tutorials.

AsTeX provides an on-line help for LaTeX in *hypertext* form. The first module is done out of the latest original `latex.tex` file, so that it is up-to-date and *exhaustive*. Other modules are available for style files and special symbols. These can be *previewed* in a separate window and selected and *copied* into the text from the help window (see Fig.1). This is useful for complicated instructions such as `\Longleftrightarrow`, `\rightleftharpoons` etc. . .

AsTeX provides also an on-line hypertext help for itself and for several other useful programs: emTeX, Ghostscript, Maple, Kermit, programs to compress files and to create or manipulate graphics files. In all the help modules, searching can be done by *subject matter* (via the hypertext mode) or by *keyword* (via the "Search a string" function of Framework).

Generation of environments

Environments can be generated automatically, by typing the keyword "`@e` " in the text (this keyword can be customized to any other keyword, for example `\env`). This activates a hierarchical menu, from which the suitable environment can be selected. Automatic indentation at any level is provided. For example, typing `@e` and selecting item `List` then subitem `enumerate` generates:

```
\begin{enumerate}
  ...
\end{enumerate}
```

where the underscore character indicates the position of the cursor. By typing once again `@e` and selecting items `List` and `itemize`, we obtain: