情報処理 II 第 8 回

数学のためのコンピューター (1) TEX

桂田 祐史 (かつらだ まさし)

2002 年 6 月 13 日

前回までに解説したテキスト・ファイルは、印刷したときの体裁は考えていない。印刷のための文書を扱うソフトとしては、ワードプロセッサー、DTP システム等があるが、今回は数学者御用達の TEX を説明する。

また文書フォーマットとして、**PDF** にも言及する。

1 情報科学センターの UNIX 環境での TEX

つい先日、TEX を情報科学センターに新しくインストールし直した。それを使うには、.cshrc を次のように書き換えるとよい[1]。

.cshrc の末尾に次の一行を加えておく

```
set path=(/usr/meiji/pub/lib/teTeX/bin $path)
```

情報科学センターでの TEX の使い方については、この文書でも説明するが

http://www.isc.meiji.ac.jp/~ae00050/

にも書いておく (今後はこちらの WWW ページを見てもらいたい)。

myreport.tex を書いたとして、以下どうすれば良いか。

コンパイル (dvi ファイルの作成)

```
isc-xas06% platex myreport.tex
```

プリビュー (dvi ファイルの画面への表示)

```
isc-xas06% xdvi myreport.dvi &
```

印刷 (dvi ファイルのプリンターへの出力)

```
isc-xas06% dvips -f myreport.dvi | lp -d プリンター名
```

[1] path を変更するのは慎重にすべきである。打ち間違えると直すのも大変になる。また、登録するディレクトリィを管理する人も (悪いことはしない、セキュリティに気をつけている等) 信用できることが必要である。桂田が信用できない場合は、こうしないこと。

2 TEX の生まれた経緯

TEX (てっく, てふ) はコンピューター科学者である Donald Knuth [2]が、自分の著作[3]の組版のために作成した文書整形システムである。(改版の際に、出版社がコンピューター組版をしてきた出来映えに腹を立てて、自分で作ることにしたのだという話である。何でもそのために、英語の組版について入手可能であった文献はすべて読破したそうである。)

数学の研究論文を論文誌に投稿する場合、TEX だと有利 (掲載されるまでの時間が短くて済む) にされることが多かったこともあって、数学界では急速に普及した。

現在では、E-mail, WWW[4] などで、特に数式を含むものは TEX で作成した文書 (フォーマットは .tex, .dvi, .ps, .pdf など様々) が使われることが多い。

3 TEX の特徴

- 数学的な表現 (いわゆる数式) の取り扱いに優れている。
- マークアップ言語形式であり、使い方はバッチ形式である。
- (意外なことだが) 案外と書きやすい[5](ただし .tex ファイルは読みづらい)。
- Windows, Mac, Unix, MS-DOS など様々なシステムで利用できる。
- TEX で作成した文書ファイルはテキスト・ファイルであるため可搬性も高く、互換性は非常に高い。
- TEX 本体は公開 (書籍で解説までしてある) & 無償配布されている。
- フォントを作るソフトウェア METAFONT も同様に公開&無償配布されている。
- LATEX[6] や AMS[7] 拡張を始め、各種パッケージ、フォント、ドライバーなど、多くの人達から寄与がある。ほとんどすべてが公開&無償配布されている。
- TEX はある意味でプログラミング言語 (その仕様はもう固定されていて、バグ取りしかしないそうである) であり、様々な拡張が可能。実際に様々な面で発展中。

---

[2] `http://www-cs-faculty.Stanford.EDU/~knuth/`

[3] The Art of Computer Programming という大変に有名なアルゴリズムの教科書。

[4] 例えば、明治大学が契約している MathSciNet (`http://www.ams.org/mathscinet/`) や、プレプリント・サーバー。

[5] **昔話**TEX が日本で普及し始めた頃、数学者の作った『数学用ワープロ』が林立していました (私は 4 つほど実際に使ってみました)。最初の印象は、それらワープロと比べて、マークアップ言語形式の TEX は「使いづらいに決まっている」だったわけですが、実際には数学者世界はあっと言う間に TEX に統一されました。その理由は色々あって、単に使い勝手の問題だけではないと考えられますが (TEX が元々出版物の作成を目標としていたので、出来上がった文書の品質が非常に高く、手間の軽減を目的として TEX での論文投稿が推奨 — 優遇 — されたというのは大きかったかもしれません)、少なくとも実際に使っている人で、使い勝手に文句を言い続けていた人はあまりいなかったようです。

[6] もともとは Leslie Lamport (`http://lamport.org/`) という数学者が作ったものだが、現在は LATEX Project (`http://www.latex-project.org/`) が引き継いでいる。

[7] AMS=American Mathematical Society (アメリカ数学会).

- 原理的に、どこで出力しても同じ結果になる (**再現性は非常に高い**— 特に英文では抜群)。
- 過去に組版したものが現在も印刷可能である。おそらく今後も、相当長い間そうなっているであろう。
- 数学、物理、化学、情報科学などの自然科学はもちろんだが、文系の学術分野等でも意外と利用実績がある[8]。

TEX の時代遅れな点としては、フォントに関して、元来ビットマップ・フォントを使うように設計されている点、図や画像に関して、プリンター・ドライバー任せにした点など開発当時の情勢からは仕方がなかったことであるが、現在では足枷になっていて、LATEX 2_ε でようやく解決されつつある。

4 発展する TEX(？)

時空を超えてというと大げさだろうけれど…色々なことに使われている。

4.1 日本語の縦書き

ASCII の **pTEX** で満足の行く縦書きが出来るようになった。

4.2 色々な文字 (1) 今昔文字鏡

古今使われている漢字 (日本、中国、台湾、ベトナム、韓国はもちろん、梵字、甲骨文字まで入っている…) をすべて収録してやろうというプロジェクト**今昔文字鏡**(こんじゃくもじきょう) (`http://www.mojikyo.gr.jp/`) がある[9]。ここでは、(有料のソフトも開発・販売しているが) 作成したフォント (現在 12 万字) を各種フォーマットで無償配布している。これを TEX で使うことが可能になっている。

少し脱線になるが、『文字鏡関係資料集』の『文字論リンク』(`http://www.mojikyo.org/html/dat_link/links.html`) は文字コードについて参考になる情報が多い。

4.3 色々な文字 (2) 発音記号

TIPA (国際音声字母 — いわゆる発音記号) by 福井玲 (`http://www.tooyoo.l.u-tokyo.ac.jp/~fkr/`, 特に `http://www.tooyoo.l.u-tokyo.ac.jp/~fkr/tipa.html`)

4.4 多言語化

ヨーロッパの言語ならば、既に **Babel** というパッケージがある。

[8]ちなみに (学術とも関係ないけれど)、明治大学の中で私の目にとまった範囲でも、『システム協議会報告書』、『図書の譜　明治大学図書館紀要』が TEX で組版されている。

[9]文字鏡研究会編『パソコン悠悠漢字術 2002 — 今昔文字鏡徹底活用』紀伊國屋書店 (2002).

さらに TEX を拡張した **Omega Project** (http://omega.cse.unsw.edu.au:8080/index.html) があり、Omega を日本語化した『Omega-J』(http://www.havenrock.com/archives/classic/docproc/nihongo/omega-j/) の試みもあるが、まだまだ (今後に期待)。

4.5 古典籍

金水 敏 (きんすいさとし) 氏による『LaTeX による古典籍のコード化のためのマクロ作成』(http://www.let.osaka-u.ac.jp/~kinsui/tex/hokok98/1213.html) は一読の価値がある。氏は漢文マクロを作成して公開 (http://bun153.let.osaka-u.ac.jp/kokugogaku/kinsui/tex/top.htm) している。

4.6 参考になるページ

小野康男『TeX について』(http://homepage2.nifty.com/onoy/tex.htm) mule2tex, MusiXTeX, Babel の説明など。

4.7 楽譜

『MusiXTeX 入門』(http://yoneda-www.cs.titech.ac.jp/~kota/mxt/mxt_a01.html)

5 PDF

Adobe Systems の提唱した **PDF** (Portable Document Format) は、どこでも同じように表示&印刷できる文書を作ることを目指して作られた文書フォーマットである。

表示&印刷のためのソフトウェアである Acrobat Reader は無償で配布されている (PDF ファイルを作成&編集するためのソフトウェアは有料だが)。またフォーマットは公開されていて、第三者がソフトを書くことも出来るようになっている。

6 私の使っているソフト紹介

6.1 JLaTeX2HTML

LATEX 文書から HTML 形式 (WWW ページ用のデータ形式) のデータを作成するソフト LaTeX2HTML (の日本語化) がある。

(この講義の WWW ページの作成に使っているので、完成度のほどはまあまあ分かるでしょう‥)

例えば LATEX で書いた myreport.tex があるとき、

latex2html で myreport.tex を html に変換

```
isc-xas06% latex2html myreport.tex
```

で myreport という名前のディレクトリィが作られ、そこに html ファイルができる。この myreport を WWW サーバーに持って行けばよい。例えば情報科学センターでなら、次のようにすればよい。

これでインターネットから見られるようになる

```
isc-xas06% mkdir ~/public_html
isc-xas06% cp -pr myreport ~/public_html
```

6.2 日本語化 dvipdfm

TEX 文書から PDF を作成する方法は何種類かあるが、現時点で日本語の LATEX を使う場合は、日本語化 dvipdfm を使うのがお勧めである。

普通の文章、数式だけなら簡単で、単に

dvipdfm で dvi ファイルを pdf に変換

```
isc-xas06% dvipdfm myreport.dvi
```

で pdf ファイルが作れる。

PostScript データも簡単なものならば \includegraphics でインクルードした文書がそのまま PDF に変換できる (PostScript データの取り込みに関しては、http://www.math.meiji.ac.jp/~mk/labo/tex.html#FIGURE などを参照してもらいたい)。

GIF や JPEG などの画像ファイルをインクルードする場合は、

プリアンブルにこう書く

```
\usepackage[dvipdfm]{graphicx}
```

としておけば良い[10]。

6.3 YaTeX

野鳥 (YaTeX) とは GNU Emacs (emacs) で利用可能な、TeX を便利に使うための Emacs Lisp パッケージである。

情報科学センターの UNIX (Solaris) 環境では、まず使う前に ~/.emacs の末尾に次の 4 行を書き加える (最初の 1 行は注釈なので省略可能)。

```
;; re00018 (桂田) のホームディレクトリィの emacs-lisp を参照する
(setq load-path (cons (expand-file-name "~re00018/emacs-lisp")
                      load-path))
(require 'yatex-startup)
```

この 4 行を収めたファイル ~re00018/emacs-add を用意したので、emacs .emacs & とした後に C-x i ~re00018/emacs-add↵ としても良い。

[10] \usepackage[dvips]{graphicx} とすると、PostScript データくらいしか、インクルードできない。

野鳥の使い方については、『野鳥のすすめ』(http://www.math.meiji.ac.jp/~mk/labo/tex.html#YaTeX) にある WWW ページを見るのが良い。

A　私の卒研で TEX の利用を勧めるわけ

A.1　コンピューターで書こう

私は卒研の学生に、卒業研究レポートはコンピューターを使って書く よう指導 (強制とも言う) しています。

そうする背景には、一つには、きちんとした文書を書かせることは教育上大変に重要だと考えていることがあります。私は卒業研究レポートは時間の許す限り添削することにしています。文章を書くのにコンピューターを用いる利点で一番大きなものは、修正・推敲が容易な点でしょう。手書きで書いている学生への添削はどうしても甘くなりがちです (もう清書の時間は残っていないし、これくらいは目をつぶってやるか、と思う)。コンピューターで書いている場合は、土壇場になって、「やはりこういう構成の方が自然だから、ここととここの順番を入れ替えて」というような大手術の勧告も可能です。

もう一つ大事なことは、卒業研究レポートを電子化しておくと、後で再利用が簡単だと言うことがあります。手書きのものはコピーにも手間がかかり、なくさないように管理するのも大変です。十人近い学生に読まれて、ぼろぼろになった卒業研究レポートなどもあります。

A.2　TEX で書こう

次に数あるソフトウェアの中から特に TEX (中でも LATEX) を推奨する理由を説明します。

(i) 結局は一番簡単だ。

(ii) 組版の常識を勉強すること、文章全体の構成を考えて書くこと等を半ば強制される。教育的であると同時に、結果として出来上がりが美しくなる。

(iii) 文書の保存性が良い。

世の中には TEX が難しい、使いづらいという意見がないわけではありません。TEX 文書を書くことはプログラミングをしているようなもので、確かにある種の難しさはあり、特に一度しか使わない場合、TEX を習得することが見合うのか疑問を感じる人が出るのももっともかも知れません。しかし、ここは町のパソコン教室ではなく、大学の数学科です。

B　初心者が注意すべきこと

B.1　数式モードとそれ以外のモードを混同しない

plain TEX の $, $$ コマンド、LATEX の \(\) コマンド、\[\] コマンドや equation, eqnarray 環境などで数式モードと呼ばれるモードになることは使い始めた人は誰でも知っていると思いますが、使い方を誤解している人がかなりいます。

数式は必ず数式モードで扱うべきものです。例えば

ソース

```
点（x,y,z）における温度を u(x,y,z) として、
```

組版の結果

点（x,y,z）における温度を u(x,y,z) として、

とするのはおかしく、

ソース

```
点 $(x,y,z)$ における温度を $u(x,y,z)$ として、
```

組版の結果

点 (x, y, z) における温度を $u(x, y, z)$ として、

でなければいけません。

やや細かいことになりますが、英文でものを区切るときに用いるコンマ ‘,’ は、後に必ず適当な幅のスペースを置く必要があります。数式モードでは単に空白 ‘ ’ を置いても無視されるので、コンマは数式モードでは使わないのが良いでしょう。

ソース

```
係数を $a,b,c$ はいずれも実数であり、$a\ne 0$ とする。
```

組版の結果

係数を a, b, c はいずれも実数であり、$a \neq 0$ とする。

はおかしく、

ソース

```
係数を $a$, $b$, $c$ はいずれも実数であり、$a\ne 0$ とする。
```

組版の結果

係数を a, b, c はいずれも実数であり、$a \neq 0$ とする。

とするべき、ということです。

display 数式モードの中に言葉を入れたいことがあります。この場合は、\hbox や \mbox を使います。例えば

ソース

```
\[
  x=0\quad または\quad x=1
\]
```

組版の結果

$$x=0 \quad \text{または} \quad x=1$$

は本当は使うべきでなく、

ソース

```
\[
  x=0\quad\mbox{または}\quad x=1
\]
```

組版の結果

$$x=0 \quad \text{または} \quad x=1$$

とします。

こうする理由は英語でやってみると分かりやすいです。

ソース

```
\[
  x^2=1\quad iff and only if\quad x=\pm 1
\]
```

組版の結果

$$x^2=1 \quad iff and only if \quad x=\pm 1$$

むちゃくちゃですね (日本語の場合は単語間スペースというものがなく、文字はほとんど等幅なので、それなりに表示されるのですが)。

B.2 強制改行 \\ を濫用しない

TEX を覚えたての人が必ずと言って良いほどやってしまう間違いが、強制改行 \\ を濫用してしまうというものです。

表を作る環境 (tabular, tabbing, array 等) で、その行が終わることを表わすために \\ を使うのは妥当な使い方ですが、それ以外は滅多に使わないはずです。

本来は paragraph (段落) を変えるべきところで、強制改行をしている人が多いです。段落を変えるには \par コマンドを使うか、空行をおくのが良いでしょう。

B.3 改ページ \newpage を濫用しない

jarticle スタイルは、論文などの比較的短い文書を組版するためのスタイルであって、例えば節が終わっても改ページするのではなく、そのまま次の節が続くようになっています。これが気に食わないという理由で \newpage コマンドを使う人が結構います。しかし、もしも jarticle スタイルが気に入らないのであれば、jreport や jbook あるいは他の (たとえば jsarticle) スタイル・ファイルの利用を考えるべきでしょう。

(まあ、まともな論文を一つも読んだことがないような状態で、まとまったレポートを書けというのが、少々無理なのかもしれない…)

B.4 LaTeX の目次の自動生成コマンドを利用する

大学ではレポート課題が出されることが多いですが、卒業研究レポートのような長いレポートを書く機会はまれでしょう。ある程度の長さの文章となると、どのように構成するかが重要な問題となります。そういう場合には、目次を生成してそれを眺めてみるということを勧めたいです。やり方は簡単、目次をおきたい場所に \tableofcontents コマンドを使うだけです。

\maketitle の直後あたりに \tableofcontents を置くと良いでしょう。

(jarticle スタイルで目次を使うこともできるが、それは変だという意見を持っている人もいるようです。確かにある程度長い文書は jarticle ではなくて、jreport や jbook スタイルを使うべきかも知れません。)

C 良く使うテクニック

C.1 ソース・プログラムを貼込む

verbatimfiles パッケージを使うのがお勧めです。プリアンブルに

ソース (1)

```
\usepackage{verbatimfiles}
```

と書いておいて、ファイルを取り込みたいところで、

ソース (2-a)

```
\verbatimfile{ ファイル名 }
```

または

ソース (2-b)

```
\verbatimlisting{ ファイル名 }
```

とします。後者は行番号を表示しますが、バグがあるようで、時々組版に失敗します。

凸凹に表示されたら…

一つだけ注意すべきことがあります。verbatim 環境もそうなのですが、いわゆる TAB コードを適当な幅のスペースに変換しないので、TAB コードを使って字下げをしているようなプログラムは凸凹に表示されてしまいます。

UNIX では expand コマンド等を使って TAB コードを空白に展開しておくと良いでしょう。

```
oyabun% expand prog.c > foo
oyabun% mv foo prog.c
```

Makefile などは、TAB コードを展開してしまうと (Makefile として) 使えなくなってしまうので、張り込み専用のファイルを作って、代りにそちらを \verbatimfile{} で取り込むと良いでしょう。

```
oyabun% expand Makefile > Makefile.expanded
```

```
\verbatimfile{Makefile.expanded}
```

C.2 図を貼込む

C.2.1 PostScript の場合

まずはプリアンブルに

ソース (1)

```
\usepackage[dvips]{graphicx}
```

と書きます。

貼込みたいところで

ソース (2)

```
\includegraphics[オプション]{ ファイル名 }
```

とします。

例えば

graph01.ps を幅 10cm に縮小して貼込む

```
\includegraphics[width=10cm]{graph01.ps}
```

graph02.eps を高さ 10cm に縮小して、さらに 90 度回転して貼込む

```
\includegraphics[height=10cm,angle=90]{graph02.eps}
```

C.2.2 画像を貼込む

GIF や JPEG などのいわゆる画像ファイルをどうやって TEX 文書に貼込めるでしょう？実はこれはプリビューアーやプリンター・ドライバーとして何を使うかに依存します。

UNIX 環境 (xdvi, ghostscript, dvips) 画像を PostScript 形式に変換してしまうのがもっとも簡単です。変換には例えば **netpbm** パッケージのコマンド群が利用できます。

Windows 環境 (dviout)

C.2.3 簡単な線画を描く

高校の数学の教科書にのっているような、簡単な線画の場合は、(PostScript で作っても良いわけですが) LATEX の picture 環境やその拡張である eepic を使うことも出来ます。

eepic のマニュアルは次のようにして印刷できます。

```
oyabun% platex eepic
oyabun% dvips -Plp2 eepic.dvi
```

C.3 枠をつける screen 環境

```
\usepackage{ascmac}
```

```
\begin{screen}
  ASCII が配布している ascmac に含まれる screen 環境の使い方の例。
\end{screen}
```

ASCII が配布している ascmac に含まれる screen 環境の使い方の例。

```
\begin{itembox}[l]{\textbf{見出し}}
  ASCII が配布している ascmac に含まれる itembox 環境の使い方の例。
\end{itembox}
```

見出し

ASCII が配布している ascmac に含まれる itembox 環境の使い方の例。

C.4 enumerate 環境

便利なのに案外知られていないのが enumerate パッケージです。

```
\usepackage{enumerate}
```

```
\begin{enumerate}[(1)]
\item ぬん
\item そん
\item さん
\item しー
\end{enumerate}
```

(1) ぬん
(2) そん
(3) さん
(4) しー

```
\begin{enumerate}[(i)]
\item いー
\item ある
\item さん
\item すー
\end{enumerate}
```

(i) いー
(ii) ある
(iii) さん
(iv) すー

```
\begin{enumerate}[(a)]
\item ひい
\item ふう
\item みい
\item よう
\end{enumerate}
```

(a) ひい
(b) ふう
(c) みい
(d) よう

C.5 AMS 拡張

C.5.1 記号

かなり醜悪な例ですが、

```
\[
  \therefore\quad
  c\geqq a+b \fallingdotseq 1.23\quad\mbox{($\because$ $a=1$, $b=0.23$)}
\]
```

$$\therefore \quad c \geqq a+b \fallingdotseq 1.23 \quad (\because a=1,\, b=0.2345)$$

```
\[
  {\Bbb R}, {\Bbb C}, {\Bbb Z}, {\Bbb N}
\]
```

$$\mathbb{R}, \mathbb{C}, \mathbb{Z}, \mathbb{N}$$

C.6 文書を配布する

TEX に関わるソフトのほとんどはフリーソフトなので、.tex ファイルを渡して、「後は自分で印刷して下さい」ということも一応可能ではあります。特に共同で何かを書き上げようとしている場合は、「TEX を使ってみませんか？」と勧めることは考慮しても良いことだと思います (私は TEX の素晴らしさを心から信じています)。

しかし、とにかく表示・印刷出来さえすれば良いというのならば、Adobe の提唱する PDF (Portable Document Format) に変換して渡すというのがお手軽で良いかもしれません。PDF を表示・印刷するためのソフト Adobe Acrobat Reader はフリーソフトですし、最近では買ってきたパソコンに最初からインストールされている場合も多いですから。

TEX 文書を PDF にするには次の二つの方法があります。

(1) dvi ファイルを PostScript に変換してから Adobe distiler で PDF に変換する。

(2) dvi ファイルを dvipdfm で PDF に変換する。

A TEX に関する参考文献

B TEX に関する URL

参考文献

[1]

C TEX で PDF 文書を作る

C.1 TEX から PDF 文書を作る三つの方法… dvipdfm がお勧め

(1) **PDF LATEX** を使う … 日本語に対応したものがない ?!

(2) PostScript ファイルにしてから **Adobe distiler** で PDF に変換

(3) dvi ファイルにしてから **dvipdfm** で PDF に変換

凝った文書でなければ (3) の方法が最も簡単で、出来上がりもきれいなことが多い。
(2) の方法で作った文書は、画面できれいに見えないことが多い (印刷はあまり問題ない)。

C.2 dvipdfm の使い方

普通の文章、数式だけなら簡単で、単に

dvipdfm で dvi ファイルを pdf に変換

```
dvipdfm dvi ファイル名
```

で pdf ファイルが作れる。

PostScript データも簡単なものならば `\includegraphics` でインクルードした文書がそのまま PDF に変換できる。

GIF や JPEG などの画像ファイルをインクルードする場合は、

プリアンブルにこう書く

```
\usepackage[dvipdfm]{graphicx}
```

としておけば良い[11]。

[11] `\usepackage[dvips]{graphicx}` とすると、PostScript データくらいしか、インクルードできない。