解　説

LaTeX による ISCIE 論文作成術

山本　裕*・永原　正章*

1. はじめに

1985 年に LaTeX（ラテフ，ラテック，またはレイテックと読む）の最初のバージョン（LaTeX 2.09）が登場してから，すでに 25 年以上が経ち，システム制御情報の分野では，ほとんどの論文が LaTeX で書かれるようになりました．また，最近では，LaTeX で作成した講演スライドもよく見かけます．LaTeX がこの分野における文書作成の標準的なツールとなっていることは明らかです．

ISCIE 論文誌でも，記事の電子アーカイブ化・電子公開にともなって，LaTeX スタイルファイルが改訂され，LaTeX のフォーマットで書かれた原稿による入稿が推奨されています．LaTeX による入稿では，LaTeX 原稿がそのまま印刷版下となって論文誌に掲載されることになります．これにより原稿から印刷までの工程が縮小され，コストの低減が望めると共に校正の簡略化など多くの利点が生じます（そのため，LaTeX による入稿は，非 LaTeX 原稿よりも論文掲載料が安くなっています）．

しかしこのことは，著者の用意した原稿がそのまま印刷されるということにもなり，もし用意された原稿に傷があれば，それがそのまま会誌や論文集に掲載されてしまいます．従って原稿の準備にはある意味でこれまで以上に注意を払って頂かねばなりません．ミスプリやミスタイプをなくすのは無論のことですが，それに加えて LaTeX の使用上の不注意から思わぬミスが発生することがあります．

本稿では，最新の ISCIE 論文誌用 LaTeX スタイルファイルの使用を前提に，LaTeX で原稿を準備する方がよく陥りがちなミス，間違いがどのようなところにあるかを，著者の経験をふまえてお伝えしようと思います．

なお，本稿は，第一筆者が 1995 年に計測自動制御学会「計測と制御」にて発表した記事 [9] に，最近の LaTeX 実行環境に合うように記述を加筆・修正する形で執筆したものです．ことに本稿の第 2 節から第 6 節までの内容は，記事 [9] の記述をほぼ踏襲したものとなっています．このような形で掲載を許可していただいた計測自動制御学会に深く感謝いたします．

* 京都大学　情報学研究科
Key Words: LaTeX, style file, transactions, ISCIE, paper

2. LaTeX とは

まず簡単に LaTeX について触れておきます．

LaTeX はプロフェッショナルユースを目的とした組版（くみはん）ソフトウェアです．そもそもは Knuth 氏がある本 [4] を出版する際のコンピュータ版下に満足できなかったことから，LaTeX の下位レベルのプログラムである TeX の開発を決意したと伝えられるように，その目的は完全にプロ用の組版を行うためのものです．つまり

LaTeX は単なるワープロソフトではありません．

例えば，通常のワープロでは画面に表示されているイメージがそのまま，あるいはそれに近い形で印刷されるのを通例とします[1]．しかし LaTeX では，私達が作成するのは様々なコマンドを含んだソースファイルであり，できあがったファイルを LaTeX プログラムが処理して，特定の機器に依存しないファイル（device independent, 略して `dvi` ファイル）に変換します．この `dvi` ファイルから文書を印刷したり，プリビューアと呼ばれるプログラム（`xdvi`, `dviout` などが良く知られています）を用いてコンピュータの画面上で出来上がりのイメージを確認することが出来ます．また，`dvips` や `dvipdfmx` などを使って PostScript や PDF の形式へ変換することも可能です．

したがって，少し矛盾するようなことを言いますが，この 2 つの処理の仕方，すなわち `dvi` ファイルに変換してから，それを表示するというやり方は，実は見かけほど大きなものではありません．たいがいのワープロソフトでも，実は裏で同じような処理を行っており，LaTeX ではそれを同時にやって見せていないだけなのです．例えば，TeXworks[2] や WinShell[3]，TeXShop[4] などのソフトウェアを使えば，ソースファイルをエディタで作成しながら，その都度 LaTeX でコンパイルし，同時に別画面で文書の画面表示をして，疑似的にワープロソフトと似たような使い勝手にすることもできます．例えば上付き添え字などに見られるような，様々な画面制御用コードを入力する代わりに，そのための記号をソースファイ

[1] この方式を WYSIWYG（ウィジィグ）方式と呼びます．What You See Is What You Get の略です．
[2] `http://www.tug.org/texworks/`
[3] `http://www.winshell.org/`
[4] `http://pages.uoregon.edu/koch/texshop/`

ルにコマンドとして書き込むところが違っているに過ぎません.

それでは LaTeX とワープロソフトは同じかと言うと,勿論多くの点で異なっているのです.印刷・出版には多くの慣用,約束ごとがあり,また文章の論理的な構造や出来上がりの美的な配置の面からも満たさなければならない条件が数多くあります.多くのワープロソフトではこのような面に対する配慮がなく,その結果高精度のレーザープリンタを使って文字一つ一つはきれいに出力されているのに,なんとなく全体は間が抜けている,あるいは読みづらいといったような結果が起こります.

LaTeX では,数式のセッティングに最大限の注意が払われているだけでなく,長い伝統で培われてきた出版上の規則が非常に完全な形で取り入れられています.その例を挙げましょう:

- **kerning** 文字間のスペースの調整.英文で必要なことですが,全ての文字を均等に配置したのでは決して読み易い文章になりません.タイプライタのような印象を与えてしまいます.例えば 'A' というような文字では,A の外形が斜めの線であるためにその前後に相当のスペースが開き,結果として空白が開いたような印象を与えます.'M' の場合はその反対です.このような点は各文字毎にうまく調整してやらねばなりません.LaTeX ではこのような調整がメトリックファイルと呼ばれるファイルの中に各文字の大きさとなって格納されており,活字印刷のような上質の仕上がりを保証しています.
- **ligature** 合字と呼ばれる慣用.例えば find と書いたときには,'fi' は 'f' と 'i' の合成ではなく,'fi' という一つの単位として扱われます.後者では 'i' の上の点が省略されていることに注意して下さい.このような 2 重文字は ligature と呼ばれ,'ff', 'fl', 'ffi', 'ffl' など多くあります.LaTeX にはこれらの ligature の規則が組み込まれており,完全に処理されます.
- 単語間の空白の処理.LaTeX ではこれは段落毎にいわばグローバルに処理されます.そのため単語間のスペースが開きすぎるといった問題が比較的少なくなります.

またこれに加えて LaTeX の大きな特徴として

- フリーウェア(無料かつ再配布自由)である[1]
- ほぼ完全な互換性が保証されており,ほとんどどのコンピュータでも使え,かつ(基本的に)同じ仕上がりが保証されている

を挙げておきたいと思います.これによって,世界中どこでも同じソースファイルを LaTeX でコンパイル(ソースファイルを LaTeX プログラムで処理し,dvi ファイルを生成すること)をする限り,まったく同じレイアウト,仕上がりが保証されるため,情報交換や印刷原稿の提出などに非常に有利となります.

ただし,ISCIE 論文誌のように専用のスタイルファイルの使用を前提としている場合には注意が必要です.すなわち,`\setlength` 等のコマンドを使って著者が勝手にページレイアウトを「改良」してしまうと,それが原因で,論文誌全体のレイアウトに「不統一」が生じます.実際には,そのような場合,レイアウトを元にもどす作業を編集側に強いることになります.注意してください.

LaTeX には以上のように,最上の印刷仕上がりを保証するための規則,約束ごとがあります.そのため通常のワープロなどに較べて多少気難しい面があることは否定できません.しかしその規則を良く理解して使いこなせば,数式の処理を含めて LaTeX ほど上質の仕上がりを保証してくれるものはまたとないと言えるでしょう.

以下では LaTeX を使う際にえてして誤解されている部分について,若干の解説を試みたいと思います.

なお,所々に \ (バックスラッシュ)を含むコマンドが出てきます.お手持ちのコンピュータではこれは ¥と表示されているかも知れません.そのような場合は \ を ¥と読みかえてお読み下さい.

3. 数式モードあれこれ

どのように数式を書くかなどは先刻御承知,と言いたいところなのですが,LaTeX には LaTeX の約束があり,これが存外無視されていることが多いのです.

3.1 $sin = \sin$?

一番多いのがおそらくこれ.数学では sin, cos などの略語から派生した関数は,通常このようにローマン立体で書き表します.ところがこれを `$sin$` のように単に数式モードの中で "sin" と書いただけではそうなってくれません.いくら LaTeX が優秀なプログラムと言っても,著者が単に "s" と "i" と "n" の組合せを書きたいのか,あるいは正弦関数を意味しているのか知りようがないからです.試しに

```
$sin$
```

とやってみると,結果は

sin

となって,イタリックになってしまいます.これは数式モードが原則としてイタリックになっているからなのです.しかし,ただイタリックになるのならまだしも,s と i と n の間が空いて,少し妙な配置になってしまいます.これは数式モードのイタリックが math italic と言って,スペーシング(空白の処理)が通常と違うからです[2].正しく $\sin(x)$ と書くには,

[1] CTAN (`http://www.ctan.org/`) よりダウンロードできる.

[2] LaTeX のマニュアル [7] に出ている例は $differrent$ と $different$.違いがよく分かります.

```
$\sin(x)$
```

としなければなりません．

このようなものの例は沢山あって，det, dim, lim log, exp などが用意されています．しかし，ker はあっても，im がなかったり，その他 rank や diag など著者が自分で用意したくなるものが多く発生します．これを作るのに `$\mbox{\rm im }F$` などとする方が多いのですが，あまりお勧めできません．

- 上ツキ文字などになった時サイズが変わってくれない
- im と次の文字 F の間のスペースが 1 文字分に固定されてしまう

など不都合が多いのです．そこで rank や im などを作るには，以下のようにします．

```
\newcommand{\rank}{\mathop{\rm rank}}
\newcommand{\imag}{\mathop{\rm im}\nolimits}
```

ここで `\imag` の定義における `\nolimits` は $\lim_{n\to\infty}$ などに見られるような，下ツキの添字を取る場合には不要ですが，そうでない場合はつけます．

これによる結果は例えば

$$\texttt{\textbackslash rank A} \Longrightarrow \rank A$$
$$\texttt{\textbackslash imag F} \Longrightarrow \imag F$$

などとなります．最大特異値なども `$\sigma_{\max}$` などとすれば，下ツキ添字が math italic にならずに済みます．

3.2 数式モード内の文章

上に述べたように数式モードの中の文字，スペーシングは特殊なので，数式モードの中に文章を書く必要がある場合，そのまま書くとおかしなことになってしまいます．漢字はほとんど影響を受けませんが，欧文は非常にバランスの悪いものになります．例えば

```
B=\{x\in\R;\exists␣V_{x}␣such␣that␣(5)␣holds\}
```

$$\Longrightarrow B=\{x\in\mathbb{R};\exists V_x such that (5) holds\}$$

のごときです．“such” と “that” がくっついてしまうのは，数式モードでは一切の空白が無視されるためです．これを防ぐには

```
\mbox{ such that (5) holds}
```

と `\mbox{}` で囲んでやれば良ろしい．まちがっても

```
\␣such␣\␣that␣\␣(5)␣\␣holds
```

などと，間に `\␣` を挿入して強制的にスペースを開けるような “力技” に頼らないで下さい．

3.3 +, − は演算子，符号?

複数行の数式がある場合，式の始めが + あるいは − で始まるときは注意が必要です．この時は次に来るものの符号と見なされ，2 項演算子とは見てくれません．この 2 つは $+x$ と $x+y$ からわかるように，その前後のスペースが変わってくるのです．複数行の式で，式を折り畳む必要があって + あるいは − が先頭に来る時には

```
\begin{eqnarray*}
  f(x, y) & = & \exp (x - y) + \cdots \\
          &   & \mbox{} + \alpha x
\end{eqnarray*}
```

のように，何もしない要素 (例えば `\mbox{}`) を挿入して 2 項演算子であることを LaTeX に教えてやる必要があるわけです．これは `\mbox{}` に限るわけではなく，例えば `\hspace{.5cm}` のようなものでも良いのです．この時には後の部分を押し下げる効果もあります．

3.4 括弧いろいろ

かなりよく LaTeX を使っている方でもどうも括弧に関して無頓着な傾向があるようです．しかし

$$(\frac{d}{dt}+1)y(t)=u(t)$$

では d/dt が上下にはみ出してどうにも様になりません．このようなときは `\left(` と `\right)` の組を使って，

```
\left(\frac{d}{dt}+1\right)y(t) = u(t)
```

とすべきです．これにより，

$$\left(\frac{d}{dt}+1\right)y(t)=u(t)$$

ときれいに表現できます．余談ですが，このようなディスプレイモードでなく，本文中の数式では $\frac{d}{dt}$ でなく，なるべく d/dt などを用いるのがよいとされています[1]．`\left[` と `\right]`，`\left\{` と `\right\}` などについても同様です．試してみて下さい．

3.5 内積? 不等号?

ユークリッド空間のような計量のある空間を Hilbert 空間といい，そこで内積は基本的な役割をはたします．数学の世界では，伝統的に内積は丸括弧

$$(x,y)$$

などで表し，角括弧

$$\langle x,y\rangle$$

は，複素共役に関しても線形になるような（したがって内積の公理を満たさない）双一次汎関数をあらわすので別物なのです [10]．困ったことに最近は後者が内積の記号として採用される例が増えてきたので，違いが分から

[1] おそらく以前活版印刷であった時代は，このような版組が大変手間がかかった名残であろうと思われます．今は LaTeX になったので，その苦労はなくなりましたが，行間のスペースがあまり変動するのはやはり見よいものではないように思われます．

なくなってしまっています.

まあ，その違いはさておき，この $\langle x,y\rangle$ を不等号記号を使って，<x，y> とされる方が非常に多いのですね. この結果は $<x,y>$ となって，えらく尖った内積記号になってしまいます. LaTeX はちゃんとこれに対して記号を用意していますので，そちらを使って下さい.

```
\langle x, y \rangle
```

です. \langle，\rangle なんて面倒だ，いちいち書いておられない，とおっしゃる向きには

```
\newcommand{\innprod}[2]{\langle #1, #2 \rangle}
```

というマクロを定義しておくという手があります. これをソースファイルの先頭，あるいはご自分の好みのマクロファイル（例えば mymacros.sty などという名前のファイル）に書いておけば,

```
\innprod{x}{y}
```

とするだけで,

$$\langle x,y\rangle$$

が出力でき，ずいぶん簡単になります.

3.6 高さ揃え

例えば g や h のような文字は高さが異なります. これに\overline や \sqrt のようなコマンドをそのまま適用すると,

```
\sqrt{g}+\sqrt{h}
```
$$\Longrightarrow \sqrt{g}+\sqrt{h}$$

のようになって上端が揃わなくなります. そこで,

```
\sqrt{\mathstrut g}+\sqrt{\mathstrut h}
```

とすると

$$\sqrt{g}+\sqrt{h}$$

のように上下が揃います. \mathstrut は幅が 0，上下が '(' と同じボックスで，幅がないため見えませんが，上下を揃える役割をしてくれます. これは数式モードで使うものですが，通常モードでは \strut が使えます. これらを使うと，例えば eqnarray 環境などで行間の幅が詰まりすぎるときの調整などが出来ます.

4. 英文の正記法

英文論文の場合は英語の正用法があり，それに従うのは無論ですが，時々無視されているのを見かけます. LaTeX の使い方にも関係するいくつかの基本を記しておきます.

4.1 ピリオド—文末あるいは略語?

ピリオドは文末に来るに決まっているじゃないかとおっしゃるかも知れませんが，では Dr. ×××× みたいな例は如何でしょう. むろん文末ではありませんね. しかし LaTeX にはそれを区別するすべがありません. そこで次のような規則を設けています:

- 小文字の直後のピリオドは文末と見なす.
- 大文字の後のピリオドは略語であることが多いので，文末と見なさない.

2 番目の規則は例えば R. E. Kalman のように人名の省略形によく現れます.

もちろん例外があり，先の Dr. また Fig. 3.1 などはその一例，また “Smith remained in the army until the end of World War II.” のように，大文字で終わる文章もあります. これをそのまま書くと，前者では LaTeX は文章の終わりと判断するので，例えば “From the curve shown in Fig. 3.1, we conclude that ...” と場合によっては “Fig.” の後に大きな空白が開くことになって見苦しい結果になります. これを防ぐには，このピリオドが文末を意味するのではないことを LaTeX に教えてやらねばなりません. そのためには Fig.␣3.1 のように \ と空白 ␣ を打ち，ピリオドの次に 1 個の空白が続くことを指示します[1]. et␣al.␣ や etc.␣ などについても同様です. ただこの空白はそこで改行可能な空白なので，この Fig. の部分がたまたま (LaTeX でコンパイルされた後) 行の最後に配置されたときはそこで改行されてしまいます. それでは困るとおっしゃる向きは Fig.~3.1 として下さい. これで Fig. と 3.1 の間で改行されなくなります. Dr.~Yamamoto なんかの場合も同様です. この他よくある例としては，Ph. D., Prof. などがあります. また参考文献リストなどによく出て来る pp. や vol.，雑誌名の略記 Int. J. Math. Anal. Autom. なども同様です. 2 段組の場合，スペーシングの自由度が少ないので，これらがあると，すぐに文末と思われて異様に大きなスペースが開くので注意して下さい.

大文字で文章が終わる例はこれよりはるかに少ないのでほとんど出会うことはないでしょう. 念のために ... by MATLAB\@. などを覚えておいてください.

“大文字+ピリオド” でむしろ問題なのは，人名などに現れる省略形の書き方ではないでしょうか. 例えば上に出てきた R. E. Kalman のような例です. これを欧米人でもかなりの人が R.E. Kalman などのようにくっつけて書いてしまいますが，普通はピリオドと次の文字の間にスペースを開けます. 多分この間で改行されることを嫌っているのだと思われますが，それならば R.~E. のように ~ を使うべきでしょう.

[1] 3.1 のピリオドは次に空白なしに文字 “1” が続いていますので文末とはみなされません.

第 1 表　Punctuation

記号	直前	直後
.【ピリオド】	スペース無し	スペース有り
,【コンマ】	スペース無し	スペース有り
:【コロン】	スペース無し	スペース有り
;【セミコロン】	スペース無し	スペース有り
?, ! など	スペース無し	スペース有り
(【開括弧】	スペース有り	スペース無し
)【閉括弧】	スペース無し	スペース有り
—【ダッシュ】	スペース無し	スペース無し

4.2　Punctuation—区切り記号あれこれ

学生諸君の論文を直していると時々珍妙なピリオド，コンマなどの用法にぶつかります．まあ正しい区切り記号の用法を習ってないのだから仕方ありません．結構すごいのが参考文献などで `IEEE␣Trans.\␣Autom.\␣Control␣,vol.~AC-34` などとコンマの前にスペースがあり，逆にその直後には何もスペースがないといった例でした．かなり頻出するので，うっかり間違ったというわけでもないらしい．参考文献 [1] (p. 113) によると，‘.’（ピリオド）や‘,’（コンマ），‘:’（コロン）などは，**直前はスペースなし，直後に 1 スペースあける**のが原則です．他の記号等は第 1 表をご覧ください．

4.3　引用符

昔のキーボードでは (英文の) 二重引用符は '' に限られていました．このため原始的なワープロでは引用文の先頭でも '' を使わざるを得ないことがありました．しかしこれでは "This is an ugly quotation." のように見苦しくなってしまいます．LaTeX ではちゃんと開引用符（open quotation mark）“ と閉引用符（closing quotation mark）” が別々に用意されていますので，こちらを使って下さい．使い方はそれぞれ `` と '' と開，閉のシングルの引用符を続けて 2 回打つだけです．

4.4　イタリック補正

Italic 文字は言うまでもなく，すこし右に傾いています．これはイタリックだけでなく，数式モードや丸みを帯びていない *Slanted* 文字 (`\sl`) なども同様です．これをそのまま他の傾いていない文字，例えばローマン体などと同居させると，最後のところがくっついた感じになり，窮屈な印象を与えてしまいます．

これを補正するために，LaTeX ではイタリック補正というものを用意しています．これは `\/` という記号で表され，つぎにローマン体がくる直前に，例えば `{\it␣different\/}␣text` などとして使います．これを `{\it␣different}␣text` とした結果を較べてみると，前者が

different text

後者が

different text

となり，違いが分かります．`\sl` についても同様です．またテキストを強調するとき，`\it` でなく，`\em` を用いるときがありますが，これもローマン体の中で用いられたときはイタリックにする効果がありますので，同じ注意が必要です．

ただしイタリック体に続いて，コンマ，ピリオドなどがくるときは既にこれらによって少し余分目の空白が追加されますので，必ずしもイタリック補正を必要としないようです．

4.5　コマンド後のスペース

日本語を用いているときには案外問題にならないのですが，マクロの後に空白を入れるときは `\␣` のコマンドを必要とするときがあります．例えば `\LaTeX␣text` と書いたとすると，出力結果は

LaTeXtext

となって，LaTeX と text がくっついてしまい，予期したような LaTeX text と言う結果は得られません．これは何故かと言うと，`\LaTeX` の後の空白は，マクロ `\LaTeX` の終わりを意味し，空白としては認識されないからなのです．

それなら `\LaTeX␣␣` とその後にもう一つ空白をつけ加えたらどうでしょうか．残念ながら，LaTeX のソースファイルでは，1 個以上の空白文字はすべて 1 個の空白と同じものとみなされるという約束があり，結果は上に述べたものと全く変わりません．これを避けるために，LaTeX の後に強制的に 1 個の空白を出力するコマンド `\␣` を入れて `\LaTeX\␣text` とする必要があるのです．結果は無論

LaTeX text

となります．試してみて下さい．なお，このときは LaTeX の直後に空白は要りません．つぎのバックスラッシュ文字 \ が，そこからまた別のコマンドが始まることを意味し，結果的に LaTeX のコマンドの終わりを LaTeX に告げることにもなるからです．

日本語で書いている場合，ついうっかりしがちなのはむしろ

`\LaTeX` の使い方

のような書き方かも知れません．このときは ‘X’ と ‘の’ の間に空白がないため，LaTeX は ‘X’ がコマンドの終わりとは解釈できず，‘`\LaTeX` の使い方’ という一つのコマンドがあるものと見てしまいます．その結果，

```
undefined control sequence ...
```

というエラーメッセージが出てくることになります．この場合は無論

```
\LaTeX␣の使い方
```

と書かねばなりません．

5. 美しいソースを

LaTeX で原稿が受け付けられるようになると，原稿の修正，校正などはすべてソースファイルに対して行われることになります．最終印刷出力における微細な修正は，あるいは編集委員や事務局で行われることも有り得ます．そんなときのために，読み易いソースファイルであるかどうかは非常に大事な問題です．

LaTeX のソースも一種のプログラムですから，その中に各種のコマンド，制御用文字などが含まれているのはご存知の通りです．であるならば，プログラムとして読み易いように書いておいた方が，読む方も理解し易く，また後でご自分で修正されるときも労力が少なくて済むのではないでしょうか．

というと簡単のようで，実際それほど難しいことではないと思いますが，実状は必ずしもそうとも限らないようです．学生諸氏の修士論文の LaTeX ソースなどを見ていると，時に自分自身でも読めないのではないかと思えるようなソースにぶつかることがあります．行列が出てくる式などに多く，例えば次のようなものです:

```
\[\rank\left[\begin{array}{ccc}a&b&c\\
d&e&f\\g&h&i\end{array}\right]=1\]
```

(あるいはこれの全く改行されないもの．) これと

```
\[
  \left[
    \begin{array}{ccc}
      a & b & c \\
      d & e & f \\
      g & h & i
    \end{array}
  \right]
  = 1
\]
```

のとどちらが読み易いかは明らかでしょう．私はおおよそ次のような原則で書いています．

- 環境（上の例では \[と \], \left[と \right], \begin{array} と \end{array} など）は行を変え，入れ子になっているときは段付けをして対応関係が分かるようにする．
- 行列などの array 環境では行–列要素を整列させ，見やすいようにする．後で変更するときも分かりやすい．
- 適度に空白を入れ，各要素の区切りが分かりやすいようにする．
- 適度にマクロ定義を用い，ソースの内容が分かりやすいように努める．
- 行の長さを多くても半角 80 文字程度に納める．

改行をし過ぎて，余り縦に長くなるのも考えものなので，ケースバイケースですが，おおよそ上のように処理していればよいのではないかと思います．

ちなみに何回も使われる表現に，適切なマクロを定義しておくことはソースの可読性を高める上で重要な働きをします．ただこれも行き過ぎると，マクロを使って別のマクロを定義し，そのうえ更にマクロをとなったりして，本人以外にはそのソースが何を意味しているかがさっぱり分からなくなったりすることもあります．やはり程々が肝心のようです．

6. 強制改行?

どうも LaTeX を使いだしたばかりだと強制改行 \\ を使いたくなるようです．tabular 環境や eqnarray 環境などで，それを使うのが標準となっている場合は当然ですが，どうも段落の終わりを示すものと誤解されているきらいがあります．**段落の終わりは空行を１行（以上）入れれば LaTeX が自動的に判断して段落としてくれます．** またなぜか equation, eqnarray 環境などの前にこの \\ を挿入する人が多いようです．次はディスプレイモードの数式で改行しなければならないのだからというわけでしょうか．

しかしこのようなことは LaTeX が自動的にやってくれるので，不要であるばかりでなく，縦方向のスペーシングの計算が狂うので有害なのです．多分，underfull メッセージが一杯出るはずです．もう一度書いておきましょう:

- **段落は空行で**
- **ディスプレイ数式の前に \\ は不要**
- **\\ は tabular 環境や eqnarray 環境など LaTeX が必要とするところで**

7. 図の取り込み

図の取り込みは LaTeX で問題になるところです．といっても通常のブロック線図のようなものは LaTeX の picture 環境で十分描けますから，そちらをお勧めしておきたいところです．

問題はもっと複雑な図，ことに PostScript をベースとした Encapsulated PostScript 形式のファイル（eps ファイルと省略して呼ばれます）を取り込む場合でしょう．ところがこれの取り込み方には数種類あり，その違いにしたがってプリンタドライバも異なってきます．というのもこの部分は LaTeX の \special という命令に，ポストスクリプトのコードを流し込むだけであり，その流儀にしたがってデバイスやプリンタドライバの違いが

出てきてしまうのです．ここでは，ISCIE 論文集で推奨されている graphicx パッケージを用いた図の取り込み法と dvips を用いた ps ファイルへの変換法を説明します[1]．プリアンブルにてパッケージ graphicx を

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

で読み込みます．なお，上記のオプションとして，draft を使用すれば，図を表示せずに枠とファイル名だけが表示されます（ファイル名を確認するときに便利です）．次に，例えば eps ファイル fig1.eps を取り込むには，以下のようにします．

```
\begin{figure}[bt]
  \centering
  \includegraphics[width=5cm,clip]{fig1.eps}
  \caption{図作成例}
  \label{fig:fig1}
\end{figure}
```

なお，\label コマンドは必ず \caption の**直後に**入れてください．この後，プログラム dvips によって dvi ファイルをポストスクリプトファイルに変換し，プリンタに出力します．

8. PDF ファイルの作成

ISCIE 論文誌に論文を投稿する場合，現在のところ，紙に印刷した原稿を郵送しますので，論文の PDF を作成する必要はありませんが，最近では IEEE など，PDF 原稿による電子的な論文投稿のみを受付けるところが多くなっています．また，自分の論文等を PDF にして共同研究者に電子メール経由で送ったりすることもあるでしょう．したがって，LaTeX で作成した文書を PDF ファイルに変換することは，研究者にとって非常に重要なスキルです．この節では，LaTeX で作成した dvi ファイルから PDF ファイルに変換する方法とその際の注意点を述べたいと思います．dvi ファイルから PDF ファイルを作成する方法としては，dvipdfmx を使い直接 PDF ファイルを生成する方法がありますが，ここでは，前節で用いた dvips を使った方法について解説します．

8.1 フォントの埋め込み

IEEE の国際会議論文や雑誌論文では，すべてのフォントが投稿する PDF 原稿に埋め込まれていなければならないという規定があります．理工系の文章では，数式等に特殊なフォントを使用することが多く，そのような場合にフォントが埋め込まれていないと，PDF を読み込む環境によっては，著者の意図しないフォントが使用されてしまう可能性があります．それを避けるためには，すべてのフォントを埋め込まなければなりません．

また，和文の場合，いわゆる「機種依存文字」の問題もあります．例えば，Windows OS の ①，②，③ などの文字です．このような機種依存文字はなるべく使用しないほうが良いのですが，どうしても使用したいときには，これらのフォントを PDF に埋め込み，読者のパソコンが Windows であろうが Macintosh であろうが，同じ文字を表示できるようにしなければいけません．

和文のフォント埋め込みの方法は，参考文献 [8] の第 13 章を参照していただくとして，ここでは，英文フォントをすべて埋め込む方法を解説します．

まず，LaTeX ソースファイルをコンパイルしてできた dvi ファイル（foo.dvi という名前がついているとします）から PostScript ファイルを下記のように生成します[1]．

```
> dvips -Pdl foo.dvi
```

ここで，オプション -Pdl は，Times や Helvetica などの欧文基本 35 書体[2]を含むすべての欧文フォントを Type 1 形式で埋め込むためのものです．これにより，すべてのフォントが埋め込まれた PostScript ファイルが生成されます．あとはこれを Adobe の Distiller 等で PDF に変換すれば，すべての欧文フォントが埋め込まれた PDF ができあがります．

8.2 EPS 画像の取り込み

システム制御情報の分野では，計算機にてシミュレーションを行う際，MATLAB 等のシミュレーションソフトウェア を使用することが多く，そこで作成した画像ファイルを LaTeX に取り込むことも多いでしょう．ここでは，MATLAB 等のソフトウェア にて作成した図の取り込みに関する注意点を述べます．

MATLAB 等で作成したグラフを LaTeX に取り込むときは，図を eps ファイルとして保存します．このとき，生成された eps ファイルの BoundingBox が負の値をとり，dvipdfmx で PDF を作成したときに，図の一部が切れて表示されないという現象が起こることがあります（第 1 図 をご覧ください）．実際，問題の eps ファイルを見てみると，

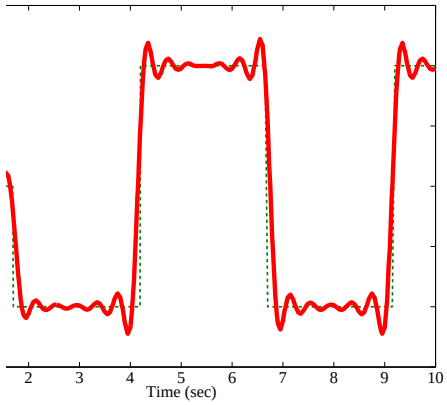
```
%%BoundingBox: -302 14 899 826
```

となっており，負の値になっています．これが，第 1 図のように図が欠ける原因です．これを防ぐためには，BoundingBox を正しい値に変更する必要がありますが，一番簡単な方法は，eps ファイルを epstopdf によって一度 pdf ファイルにした後に，この pdf ファイルを

[1] もし dvipdfmx を使用して PDF ファイルを作成するのであれば，dvips のところを dvipdfmx に読み替えてください．

[1] 記号 > はコマンドプロンプトを表します．> を入力するわけではありません．

[2] 通常の PostScript（レベル 2 以上）プリンタには，欧文基本 35 書体が備わっています．



第 1 図 eps ファイルがうまく表示されない例（図の左側が欠ける）

pdftops によって eps ファイルに戻す方法です．具体的には，

```
> epstopdf fig2.eps
> pdftoeps -eps fig2.pdf
```

とします．これにより，BoundingBox の値は

```
%%BoundingBox: 0 0 1201 812
```

と正常な値に修正されます．また，問題のある eps ファイルをフリーウェア GIMP [1]で一度開いてから，eps ファイルとして保存しなおすという方法でも BoundingBox を正しい値に修正できます．

9. LaTeX の参考書

TeX や LaTeX の参考書は数多くあります．しかしここで述べたことの多くは，LaTeX の作者 Lamport による参考文献 [7] に述べられています．まずこの本の真ん中程度までをある程度マスターされることをお勧めしたいと思います．さらに，この本の内容を補完する参考文献 [5] や参考文献 [2] も定評があります．

LaTeX の作者 Knuth による LaTeX book[6] は [7] ほど読み易くありません．ただマクロの解説や印刷の約束ごとなどに付いてはやはりこれに頼らざるをえないところが多々あります．ligature や kerning の説明についてはこの本の 4 頁を参照して下さい．

LaTeX で様々なマクロを作成するときには，参考文献 [6] とともに [3] が参考になります．ただ LaTeX の仕組みをのみこんでいないと少し読み難いところもあります．

この他に奥村氏による労作 [8] はいろいろなテクニック，スタイルファイルの解析などが載っていて非常に参考になります．必ずしもタイトルどおり「入門」とは言えませんが，備えておかれると有益と思います．

また，本稿では述べられませんでしたが，複雑な数式を記述するときに便利な $\mathcal{AMS}$-LaTeX や参考文献表の作成に威力を発揮する BibTeX，プレゼンテーション用スライドを LaTeX で作成するためのパッケージ Beamer などについても，上記参考文献 [8] を参考にしてください．

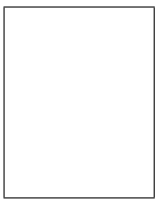
[1] http://www.gimp.org/でダウンロードできます．

参 考 文 献

[1] A. Einsohn: *The Copyeditor's Handbook*, 2nd edition, University of California Press (2006)
[2] M. Goossens, S. Rahtz, and F. Mittelbach, *The LaTeX Graphics Companion*, Addison-Wesley (1997); 鷺谷好輝訳：LaTeX グラフィックスコンパニオン，アスキー (2000)
[3] 磯崎秀樹：LaTeX 自由自在，サイエンス社（1992）
[4] D. E. Knuth: *The Art of Computer Programming*, Volume 2, 2nd edition, Addison-Wesley (1981)
[5] F. Mittelbach and M. Goossens, *The LaTeX Companion*, Addison-Wesley (1994); アスキー書籍編集部訳：The LaTeX コンパニオン，アスキー (1998)
[6] D. E. Knuth: *The TeXBook*, Addison-Wesley (1984); 鷺谷好輝訳：[改訂新版] TeX ブック，アスキー (1992)
[7] L. Lamport: *LaTeX: A Document Preparation System*, 2nd edition, Addison-Wesley (1994); 阿瀬はる美訳：文書処理システム LaTeX 2ε，ピアソン・エデュケーション (1999)
[8] 奥村晴彦：LaTeX 2ε美文書作成入門, 改訂第 5 版，技術評論社（2010）
[9] 山本裕：SICE 著者のための LaTeX べからず集；計測と制御, Vol. 34, No. 11, pp. 889–896 (1995)
[10] 山本裕：システムと制御の数学，システム制御情報ライブラリー 16，朝倉書店 (1998)

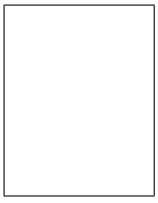
著 者 略 歴

山本（やまもと） 裕（ゆたか） (正会員)



1978 年フロリダ大学 Ph. D. 1998 年より京都大学情報学研究科複雑系科学専攻教授．専門は制御理論，およびディジタル信号処理．IEEE Control Systems Society (CSS) の Vice President，システム制御情報学会の会長などを歴任．G. S. Axelby Outstanding Paper Award，文部科学大臣表彰，など受賞多数．IEEE および SICE のフェロー．2012 年より IEEE CSS の President-elect．

永原（ながはら） 正章（まさあき） (正会員)



2003 年 3 月京都大学大学院情報学研究科博士課程修了．同年 4 月同大学助手，2007 年 4 月同大学助教となり現在に至る．ディジタル信号処理，サンプル値制御，ネットワーク化制御などの研究に従事．IEEE, SICE, IEICE などの会員．