

## 第3回 T<sub>E</sub>X ゼミ

ゼミ担当者 : 昌山 智, 田中 裕也, 坂田 大輔  
 指導院生 : 小椋 信弥, 澤田 淳二  
 開催日 : 2003 年 5 月 14 日

ゼミ内容: 本ゼミでは, DVI ファイルから PDF への変換方法, T<sub>E</sub>X で参考文献を引用する際の BIB<sub>T</sub>E<sub>X</sub> の利用方法と, 数式モードの利用方法, および, 知っておくべき追加コマンドの説明を行います.

### 1 DVI ファイルの PDF 化

#### 1.1 何故, PDF を用いるのか

PDF<sup>1</sup>ファイルとは, Adobe System が開発した電子文書の形式です.

一般に, あるソフトで作成された文書を表示するには, 相手のコンピュータにも同じソフトがインストールされていないといけません. たとえば, T<sub>E</sub>X で作成された文書を見るには, DVI ドライバがインストールされている必要があります. また, 同様に Microsoft Word で作成された文書を見るには, Word がインストールされていないと見ることができません. これでは, インターネット上で不特定多数の人に自分の書いた文書を見てもらうのに障害となってしまいます.

そこで, 作成した文書を PDF 形式に変換します. 文書を PDF 形式に変換しておけば, 相手のコンピュータには PDF を表示するプログラムだけ入っていればよく, 文書別の表示ソフトは必要なくなります.

PDF 形式を用いることで, 次のような利点があります.

- 相手のコンピュータの機種や環境によらず, 元の文書と同様のレイアウトで文書を表示することが可能
- 閲覧用のソフトは無償で配布されている
- プラグインを用いることによって, Web ブラウザ上でも閲覧が可能

#### 1.2 PDF 化のための準備

PDF への変換を行うには, Adobe Acrobat 5.0 と Acrobat Distiller 5.0 (以下 Distiller) が必要です. 実際には Distiller で最終的な PDF への変換を行うのですが, 初期設定のままではうまく変換することができません. したがって, まず Distiller を ISDL 用にカスタマイズする必要があります. これは, ISDL の設定が保存された設定ファイルが存在しますのでこれを利用します. 設定ファイルは, PrintOptimized\_mikilab.JobOptions という名前で以下のディレクトリに置いてあります.

¥ ¥ museion ¥ archive ¥ isdl ¥ basic\_seminar  
 ¥ 2001 ¥ T<sub>E</sub>X ゼミ ¥ Distiller

このファイルを Distiller の他の設定ファイルとともに格納しておきます. 設定ファイルは, デフォルトでインストールした場合は次のディレクトリに格納されています.

C: ¥ Program Files ¥ Adobe ¥ Acrobat 5.0 ¥  
 Distillr ¥ Settings

Distiller を起動すると, Fig. 1 のようなウィンドウが開きますので, ジョブオプションの欄を PrintOptimized\_mikilab にします.

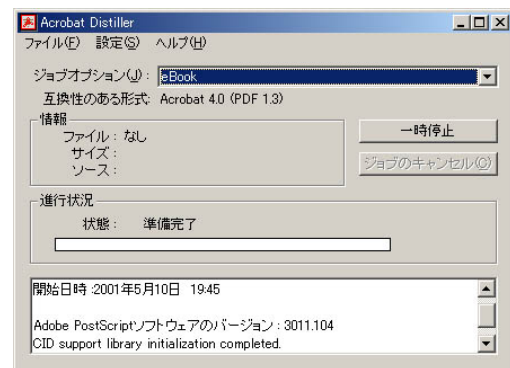


Fig. 1 Distiller の設定

#### 1.3 PDF への変換

次に実際に PDF への変換を行います, DVI ファイルから直接 PDF へと変換することはできません. したがって, まず始めに T<sub>E</sub>X のソースファイルから PS ファイルを生成し, それをさらに PDF へ変換するという方法をとります.

##### ● PS ファイルの作成

秀丸において, T<sub>E</sub>X ソースをコンパイルするためのマクロを起動した際, "PS ファイルに変換して表示" を選択します (Fig. 2 参照). T<sub>E</sub>X ソースにエラーがなければ, PS ファイルが生成されます.

<sup>1</sup>Portable Document Format

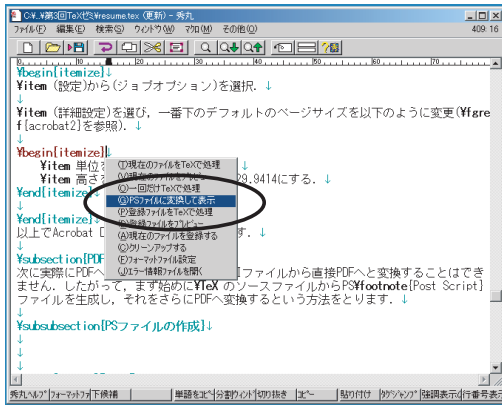


Fig. 2 秀丸のマクロ

### ● PDF への変換

Distiller を起動します。Fig. 1 のウィンドウが開きますので、生成した PS ファイルをこのウィンドウへドラッグ & ドロップします。特に問題がなければ、これで PDF ファイルが生成されます。

## 2 BIB<sub>T</sub>E<sub>X</sub>

### 2.1 BIB<sub>T</sub>E<sub>X</sub> とは

5月の月例発表会で作成したレジュメには、本文中に直接参考文献を記述していました。しかし、この方法ではいちいち入力が面倒な上、文献の再利用が難しく効率がよいとはいえません。そこで、 $T_{E}X$  の本文とは別に文献のデータベースファイルを作成し、そこから本文に参照するというのを考えます。これを行うのが BIB<sub>T</sub>E<sub>X</sub> というフリーソフトです。

この章では、まずデータベースファイルの作り方について説明し、次に  $T_{E}X$  の文書への参照の仕方を説明します。

### 2.2 文献データベース

ここでいう文献データベースとは、文献を特定の書式にしたがって並べたテキストファイルのことを指します。文献データベースは Fig. 3 のような書式にしたがって作成し、拡張子を .bib<sup>2</sup> として保存します。たとえば、GA.bib、SA.bib といった具合にデータベースファイルを作成し、その中に参考にした文献の情報を書き込んでいくわけです。

Fig. 3 では @book で始まっているため参考文献が書籍であることがわかりますが、参考文献が書籍以外にもたとえば雑誌、論文、ホームページなどの場合は書式が多少異なってきます。

Fig. 3 のように、書籍のデータには参照名(ラベル)、著者名など様々な属性がありますが、この中には必ず記述しなくてはならないものと記述しなくてもよいものの

```
@book{ cheese,
  author = " Spencer Johnson, M.D.",
  yomi = " Spencer Johnson, M.D.",
  title = " チーズはどこへ消えた？",
  publisher = " 扶桑社",
  year = 2000,
}
```

Fig. 3 文献データベース (sample.bib)

2種類があります。書籍の場合、著者名、書名、出版社名、出版年は必ず記述しなくてはならないのですが、シリーズ名や巻数などは必ずしも記述する必要はありません。この辺りの細かい話は後述するとして、次に作成したデータベースから  $T_{E}X$  の本文に引用する方法を説明します。

### 2.3 参考文献の引用

ここでは、前節の Fig. 3 で作成した文献データベースを sample.bib という名前で保存したとしてそのデータベースを参照してみます。参照するための原稿ファイルは次のようになります。

```
\begin{document}
新しい方向に進めば、新しいチーズが見つかる。
\cite{cheese}

\bibliographystyle{junsrt}
\bibliography{sample}
\end{document}
```

Fig. 4 ソースコード

このように、引用したい場所に `\cite{(参照名)}` という書式で記述します。また、`\end{document}` の前に

```
\bibliographystyle{junsrt}
\bibliography{sample}
```

と記述します。`\bibliographystyle{}` の引数は、junsrt を用います。junsrt にすると、参考文献は本文中で参照した順番に出力されます。`\bibliography{ }` の中にデータベースファイル名を拡張子はつけずに入力します。この状態で、いつも通りに文書をコンパイルすると Fig. 5 のように出力されます。

<sup>2</sup>bibliography(参考文献) の bib

新しい方向に進めば、新しいチーズが見つかる。<sup>1)</sup>

## 参考文献

1) M.D. Spencer Johnson.

Fig. 5 出力結果

## 2.4 秀丸マクロを用いた BIB ファイルの作成

前節までで述べたように、BIB ファイルで扱う文献の種類は多く、またそれぞれについて設定できる属性も異なっているため、それらをいちいち手作業で打ち込んでいたのではとても面倒です。そこでこの面倒な入力を簡略化するために、秀丸エディタ用のマクロが用意されています。ここでは、このマクロのインストール方法および利用方法を説明します。

## 2.5 BIB マクロのインストール

BIB マクロのインストールおよび動作確認は以下の手順によって行うことができます。

1. BIB マクロを以下の URL からダウンロードします。  
<http://mikilab.doshisha.ac.jp/dia/seminar/latex/misc/bib.mac>
2. ダウンロードしたマクロを秀丸エディタのマクロフォルダにコピーします。  
マクロフォルダは、デフォルトでは

C:\Program Files\Hidemaru\macros

となっています。

3. マクロの実行  
マクロが正しくインストールされたかどうか確認するために次にマクロを実行してみます。秀丸エディタを起動し、メニューのマクロ (M) - マクロ実行 (X) を選択します (Fig. 6)。すると、マクロ実行というウィンドウが表示されます。ここで、先ほどコピーした bib.mac を選択し「OK」をクリックしてください。正しく動作すれば、Fig. 7 のような画面が表示されます。
4. マクロの動作確認  
マクロの動作を確認するために、表示された bib.mac のメニューから book:書籍を選択してください。秀丸エディタに次のように入力されれば、マクロは正しく動作しています。

```
@book{ *,
author = " * ", %% または editor = " * ",
yomi = " ",
title = " * ",
publisher = " * ",
volume = " ", %% または number = " ",
series = " ",
edition = " ",
note = " ",
month = ,
year = * ,
}
```

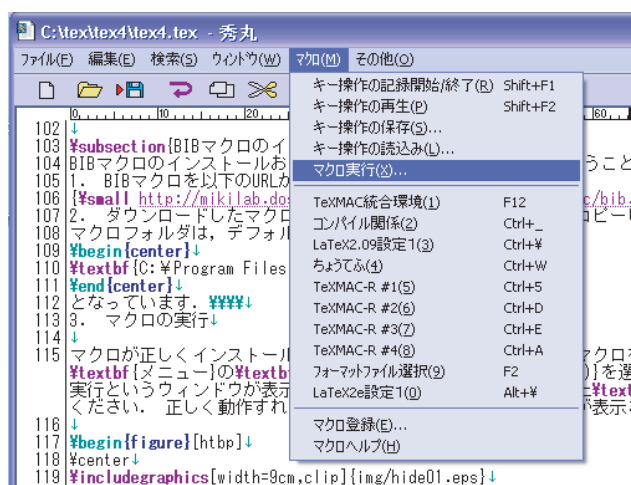


Fig. 6 マクロの実行



Fig. 7 マクロの選択

## 2.6 BIB マクロによる入力

BIB マクロでは、参考文献としてよく利用するであろう 8 種類の文献を扱うことができます。Table 1 にその 8 種類を載せておきます。

Table 1 BIB マクロで扱える文献

book	書籍
article	雑誌記事，論文誌の論文
inbook	書籍の一部 (章やページ)
manual	技術文書 (マニュアル)
masterthesis	修士論文
phdthesis	博士論文
techreport	研究機関の研究報告・技報
misc	その他

これら 8 種類の文献は，それぞれ入力することのできる属性が異なっています．各属性の意味を Table 2 に示します．

属性を入力する際の注意する点として，文献の種類を選択するとたとえば先ほどのような属性入力欄が作成されるわけですが，\*がついた属性は必ず入力してください．\*がついていない属性は入力しなくても結構ですし，削除してもらっても結構です．

また，今の状態では文献データベースを作成するたびに，メニューのマクロ (M) - マクロ実行 (X) という作業を行わなくてはならないので非常に面倒です．秀丸エディタではマクロに任意のショートカットキーを割り当てることができるので，bib.mac にもショートカットキーを割り当てると非常に便利になります．ショートカットキーの設定については，秀丸ヘルプの「キーの割り当て」の項を参照してください．

## 2.7 T<sub>E</sub>XMAC による入力

秀丸のマクロである T<sub>E</sub>XMAC を利用することで，T<sub>E</sub>X 文書への文献データベースの引用を簡単に行うことができます．T<sub>E</sub>X ファイルを新規に作成する際に「Ctrl + ¥」で T<sub>E</sub>XMAC マクロを起動し，表示されたメニューの中の「参考文献」を「junsrt」に変更し，「文献ファイル」に作成した文献データベースファイルを指定します (Fig. 8)．このとき拡張子 (bib) は必要ありません．

設定を変更したら「この設定で実行」をクリックしてください．設定したとおりに，

```
\bibliographystyle{junsrt}
\bibliography{(bib ファイル名)}
```

が記述されます．

以上で BibT<sub>E</sub>X の説明は終わりです．

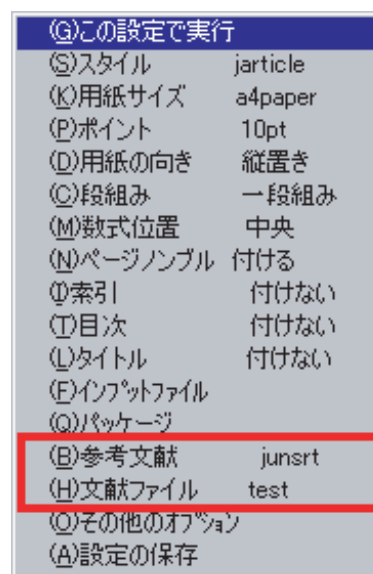
Fig. 8 T<sub>E</sub>XMAC の設定

Table 2 各属性とその意味

属性名	意味
address	publisher（発行元）の住所
annotate	メモ，注釈．通常の文献スタイルでは無視されます．
author	文献の著者
booktitle	書籍の表題名
chapter	章や節などの番号
crossref	文献リスト中で相互参照される文献のラベル
edition	書籍の版数．第2版や Second のように指定する
editor	編集者名
howpublished	出版形態が特殊な場合に，その出版形態を説明する
institution	技報などを出版した，研究所などの名前
journal	雑誌名
key	author や editer が存在しない文献において並べ替えを行うときに利用される文字列
month	刊行月
note	文献に対する注意事項
number	新聞，雑誌などの号数．
organization	会議などを主催した団体や，文書の編纂を行った団体
pages	ページ番号．”13”や”74-93”，”256+”のように指定する．
publisher	発行元名．出版社名．
school	論文が提出された学校名
series	シリーズ名
title	表題名
type	@techreport の種別
volume	巻数
year	刊行年．まだ刊行されていない場合は執筆された年．

### 3 数式

#### 3.1 数式モード

L<sup>A</sup>T<sub>E</sub>X で数式を出力する際には、通常の文章が記述される状態（平文モード）とは異なり、特殊な記号やコマンドを利用します。そこで、ソースファイル中には数式を記述するための領域であることを宣言しておく必要があります。この数式を記述するための領域が数式モードです。

数式モードには「インテキストモード」と「ディスプレイモード」の 2 種類が存在します。この 2 つのモードの違いは、出力位置だけでなく、記号の大きさなどといった違いもあります。

インテキストモード： $y = x^2 + 1$   
 ディスプレイモード：

$$y = x^2 + 1$$

インテキストモードを利用する場合、math 環境、またはその省略形である「\$」と「\$」もしくは「\math(」と「\math)」で囲みます。具体的な利用例は以下の通り。

```
\begin{math} y=x^2+1 \end{math}
$y=x^2+1$
\math(y=x^2+1\)
```

ディスプレイモードを利用する場合は、displaymath 環境、またはその省略形である「\$\$」と「\$\$」もしくは「\math(」と「\math)」で囲みます。具体的な利用例は以下の通り。

```
\begin{displaymath}
y=x^2+1
\end{displaymath}
$$y=x^2+1$$
\math[y=x^2+1\)
```

##### 3.1.1 空白

数式モードでは、T<sub>E</sub>X や L<sup>A</sup>T<sub>E</sub>X の命令の区切りを示す以外の半角空白はまったく無視されます。つまり、“\$x + 1\$” と “\$x+1\$”，さらには “\$ x +1\$” も、すべて同一の出力  $x + 1$  となります。

空白の制御方法には次ページの Table 3 に示すコマンドを使います。

T<sub>E</sub>X も L<sup>A</sup>T<sub>E</sub>X も、ほとんどの場合は自動的に適切な空白制御を行って数式を組版しますが、時には自分で細かく空白を調節したほうが美しい結果になることもあります。例えば、次の例を考えてみましょう。

$$(1) \int \int_D f(x, y) dx dy$$

$$(2) \iint_D f(x, y) dx dy$$

上の出力結果を見ても分かる通り、(1) の式よりも (2) の式の方が見栄えが良くなります。これは、“dx”、“dy”の前に \int 命令により空白を入れ、それぞれの積分記号のあいだの空白を \int 命令により狭めているからです。以下に、(1)、(2) の式のソースを示します。

```
\begin{eqnarray*}
\int \int_D f(x, y) dx dy
\int \int_D f(x, y) \int dx \int dy
\end{eqnarray*}
```

##### 3.1.2 文字列の扱い

数式モードに記述された文字列は、原則としてすべて数式（変数）とみなされ、数式用に用意された特殊なイタリック体（数式イタリック体）で出力されます。例えば、“diff(a)” という出力を得たい場合に “\$diff(a)\$” と記述すると、“d”、“i”、“f”、“f” はそれぞれ個別の変数として扱われ、“diff(a)” と出力されてしまいます。このようなときは、書体変換コマンドを使う。例えば本文中のイタリック体と同様の出力を得るには、“ $\mathit{diff}(a)$ ” と記述します。同様に、“Hom<sub>2</sub>(T)” という出力を得たい場合には、“ $\mathrm{Hom}_2(T)$ ” のように記述します。

数式モードで使える基本的な書式を Table 4 に示します。

Table 4 で示した命令は、数式中の変数などの出力に利用するため、引数の中も数式モードです。したがって、引き数の中の半角空白は無視されます。

また、\mathnormal 命令によって出力される「標準的な書体」とは、デフォルトでは数式イタリック体のことを指します。

##### 3.1.3 添え字

数式モードでは文字に添え字を付けることができます。

下添え字は \_ 記号、上添え字は ^ 記号に続けて書きます。これらは同時に指定することができるため、例えば、数式中で a<sup>i</sup><sub>j</sub> と書けば a<sub>j</sub><sup>i</sup> のようになり、\int\_a^b と書けば ∫<sub>a</sub><sup>b</sup> のようになります。添え字の配置は、式が用いられる場所などに応じて L<sup>A</sup>T<sub>E</sub>X が自動的に決めます。

また、x<sub>j+1</sub><sup>(k-1)</sup> のように、添え字が式のような複数の文字列からなる場合には、\$x\_{j+1}^{(k-1)}\$ のように上添え字部と下添え字部を {} とでくくります。

Table 3 数式モード内での空白制御命令

命令	空白	出力例	数式モード以外での使用
<code>\_</code>	半角の空白	<code>@_@</code>	可
<code>\quad</code>	全角の空白 (クワタ)	<code>@ @</code>	可
<code>\qquad</code>	2 個のクワタ	<code>@ @</code>	可
<code>\,</code>	細スペース (クワタの $\frac{1}{6}$ )	<code>@@</code>	可
<code>\&gt;</code>	中スペース (クワタの $\frac{2}{9}$ )	<code>@@</code>	不可
<code>\;</code>	太スペース (クワタの $\frac{5}{18}$ )	<code>@@</code>	不可
<code>\!</code>	負の細スペース (クワタの $-\frac{1}{6}$ )	<code>@@</code>	不可

Table 4 数式モード中で本文用の書体を使用するための命令

書体名	命令と使用例	出力例
ローマン体	<code>\mathrm{A B C D E F}</code>	ABCDEF
ボールド体	<code>\mathbf{A B C D E F}</code>	<b>ABCDEF</b>
サンセリフ体	<code>\mathsf{A B C D E F}</code>	ABCDEF
イタリック体	<code>\mathit{A B C D E F}</code>	<i>ABCDEF</i>
タイプライタ体	<code>\mathtt{A B C D E F}</code>	ABCDEF
カリグラフィック体	<code>\mathcal{A B C D E F}</code>	<i>ABCDEF</i>
明朝体	<code>\mathmc{A B C あ 亜}</code>	ABC あ亜
ゴシック体	<code>\mathgt{A B C あ 亜}</code>	ABC あ亜
標準的な書体	<code>\mathnormal{A B C D E F}</code>	ABCDEF

## 3.2 eqnarray 環境

### 3.2.1 数式に番号をつける

`eqnarray` 環境はディスプレイ環境の数式に数式番号をつけるための環境です。環境内の各行には自動的に番号が付けられます。番号の出力形式は、Table 5 に示すように利用するクラスファイルによって異なります。

Table 5 数式番号の出力形式

クラスファイル	出力形式	例
<code>jarticle</code>	(数式番号)	(10)
<code>jreport, jbook</code>	(章番号. 数式番号)	(2.5)

`jreport` クラスや `jbook` クラスの場合、数式番号は章が変わるたびに初期化され、1 から数えられます。それに対して `jarticle` クラスでは、文書を通じて通し番号となります。

もし、どの式にも番号を付けたくないなら、環境名を `eqnarray*` とします。また、1 つの式に番号を付けたくない場合は、その式が記述されている行の改行コマンド “`\`” の前に “`\nonumber`” を記述します。

以下に、`eqnarray` 環境の使用例を示します。

初等関数とは、だいたい次の関数のことをいう。

```

\begin{eqnarray}
y = a_n x^n + a_{n-1} x^{n-1} +
\cdots + a_1 x + a_0 \\
y = \sin x \\
y = \cos x \\
y = \tan x \\
y = e^x \\
y = \log x
\end{eqnarray}

```

このソースの結果は以下の通り。

初等関数とは、だいたい次の関数のことをいう。

$$y = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad (1)$$

$$y = \sin x \quad (2)$$

$$y = \cos x \quad (3)$$

$$y = \tan x \quad (4)$$

$$y = e^x \quad (5)$$

$$y = \log x \quad (6)$$



### 3.2.2 式を揃える

前節の例では数式が不揃いになっていますが、`eqnarray` 環境では目標位置の記号を “&” で囲むことで、数式をそろえることができます。以下にその例を示します。

初等関数とは、だいたい次の関数のことをいう。

$$y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (7)$$

$$y = \sin x \quad (8)$$

$$y = \cos x \quad (9)$$

$$y = \tan x \quad (10)$$

$$y = e^x \quad (11)$$

$$y = \log x \quad (12)$$

この例では、各行の数式を “=” という記号で揃えています。ソースコードを以下に示します。

初等関数とは、だいたい次の関数のことをいう。

```
\begin{eqnarray}
y &=& a_n x^n + a_{n-1} x^{n-1} +
\cdots + a_1 x + a_0 \\
y &=& \sin x \\
y &=& \cos x \\
y &=& \tan x \\
y &=& e^x \\
y &=& \log x
\end{eqnarray}
```

### 3.3 分数

#### 3.3.1 分数の書き方

$\text{\LaTeX}$  において分数を書く場合は `\frac` コマンドを使用します。第 1 引数に分子を、第 2 引数に分母を指定します。

```
\[
y = \frac{1}{x+1}
\]
```

例えば、上の例は次のよう出力されます。

$$y = \frac{1}{x+1}$$

#### 3.3.2 分数中の文字の大きさの調節

“`\frac`” は文字の大きさを自動的に調節します。しかし複雑な数式になると、文字の大きさをどう決めるかによって見栄えが大きく異なります。分数中の文字の大きさを調節する場合、以下のようなコマンドを用います。

Table 6 分数中の文字を調節するコマンド

コマンド	説明
<code>\displaystyle</code>	インテキストモードでの数式をディスプレイモードと同じフォントサイズで出力する。フォントサイズが若干大きくなる。
<code>\textstyle</code>	数式を通常のインテキストモードと同じフォントサイズに変更する。フォントサイズが若干小さくなる。
<code>\scriptstyle</code>	数式を添え字と同じフォントサイズに変更するコマンド。フォントサイズが小さくなる。

これらのコマンドを使った例を以下に示します。

通常：`\frac{a+b}{c+\frac{d}{e}}`

$$\frac{a+b}{c+\frac{d}{e}}$$

若干大きく：

`\frac{a+b}{\displaystyle c+\frac{d}{e}}`

$$\frac{a+b}{c+\frac{d}{e}}$$

### 3.4 行列

数式モードで行列を表現する場合は `array` 環境を使用します。

#### 3.4.1 基本的な書式

まず、`array` 環境の基本的な例を以下に示します。

```
\[
\begin{array}{cc}
a & b \\
c & d
\end{array}
\]
```

出力結果は以下のようになります。

$$\begin{array}{cc} a & b \\ c & d \end{array}$$

上記のように、`array` 環境には引数を指定します。この引数には `tabular` 環境と同じく `l`, `c`, `r` の 3 つの文字を使用し、配列要素についての情報を指定します。また、



各要素の区切りを&記号で表す点も tabular 環境と同じです。

### 3.4.2 括弧で囲む

通常，行列やベクトル（座標）は括弧でくくって記述します．括弧を記述する場合は “\left” ， “\right” コマンドを使用します．これにより，行列の大きさにあったサイズの括弧を出力することができます．

コマンドを使用する際は，行列の前に “\left(” といったように，\left の後に使用する括弧の種類を指定します．行列の後にも同様に， “\right)” と入力します．

以下に括弧を使った行列の例を示します．

```

\left(
\begin{array}{cc}
a & b \\
c & d
\end{array}
\right)

```

この出力は以下のようになります．

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

### 3.4.3 中心線の位置

array 環境は，オプション引数によって中心線の位置を指定できます．中心線というのは，式の縦方向における中心の位置を示す不可視の線のことで，例えば， $-5 + 4 = -1$  という数式があったとき， $\text{T}_{\text{E}}\text{X}$  は  $-$  記号を結ぶ直線を中心線として扱っていて，数式同士でこの中心線の高さが揃うように配置を決定しています．

array 環境の場合，デフォルトで中心線は配置の中央を貫いていますが，Table 7 に示すオプション引数を指定することで，この中心線の位置を変更することができます．

Table 7 array 環境のオプション引数

文字	中心線の位置
t	行列の最上行
b	行列の最下行

$\text{T}_{\text{E}}\text{X}$  は数式の中心線の高さが揃うように組版を行うため，中心線を変更することで，例えば次のような出力を得ることができます．

$$\begin{array}{rcc}
 & & c_1 \\
 & a_1 & \vdots \\
 x - & \vdots & - b_1 + c_n \\
 & a_n & \vdots \\
 & & b_n
 \end{array}$$

この出力の記述方法を以下に示す．

```

\left[
x - \begin{array}{c}
a_1 \\
\vdots \\
a_n
\end{array}
- \begin{array}{t}{c}
b_1 \\
\vdots \\
b_n
\end{array}
+ \begin{array}{b}{c}
c_1 \\
\vdots \\
c_n
\end{array}
\right]

```

## 3.5 特殊文字や数学記号など

数式モードでは，さまざまな記号を使用することができます．ここでは，数式モードで利用可能な記号類の出力について説明します．

### 3.5.1 ギリシャ文字

ギリシャ文字は数式モードでのみ出力可能です．

ギリシャ文字を出力する命令は，その英語名のまえに $\#$ 記号を付けたものです．ただし， $\circ$  (omicron) はアルファベットの O や o と同じなので，特別なフォントは用意されていません．一覧を Table 8 に示します．

また，一部のギリシャ文字（小文字）には，Table 9 に示すように Table 8 に示したのとは異なる変体文字が存在しています．

ギリシャ文字の大文字を出力する命令は， $\#$ に続くその英語名の先頭を大文字にしたものです．一覧を Table 10 に示します．

なお，Table 10 に記した以外のギリシャ大文字は，アルファベットの大文字と同一です．

### 3.5.2 数式アクセント

数式モードでは，段落モードで用いるようなアクセント命令ではなく，専用のアクセント命令を利用しなければアクセントを出力することができません．

ここでは、 $a$  にアクセントを付ける例を Table 11 に示します。

Table 8 ギリシャ小文字

命令	出力	命令	出力
<code>\alpha</code>	$\alpha$	<code>\nu</code>	$\nu$
<code>\beta</code>	$\beta$	<code>\xi</code>	$\xi$
<code>\gamma</code>	$\gamma$	<code>o</code>	$o$
<code>\delta</code>	$\delta$	<code>\pi</code>	$\pi$
<code>\epsilon</code>	$\epsilon$	<code>\rho</code>	$\rho$
<code>\zeta</code>	$\zeta$	<code>\sigma</code>	$\sigma$
<code>\eta</code>	$\eta$	<code>\tau</code>	$\tau$
<code>\theta</code>	$\theta$	<code>\upsilon</code>	$\upsilon$
<code>\iota</code>	$\iota$	<code>\phi</code>	$\phi$
<code>\kappa</code>	$\kappa$	<code>\chi</code>	$\chi$
<code>\lambda</code>	$\lambda$	<code>\psi</code>	$\psi$
<code>\mu</code>	$\mu$	<code>\omega</code>	$\omega$

Table 9 ギリシャ小文字の異体文字

命令	出力	命令	出力
<code>\varepsilon</code>	$\varepsilon$	<code>\varrho</code>	$\varrho$
<code>\vartheta</code>	$\vartheta$	<code>\varsigma</code>	$\varsigma$
<code>\varpi</code>	$\varpi$	<code>\varphi</code>	$\varphi$

Table 10 ギリシャ大文字

命令	出力	命令	出力
<code>\Gamma</code>	$\Gamma$	<code>\Sigma</code>	$\Sigma$
<code>\Delta</code>	$\Delta$	<code>\Upsilon</code>	$\Upsilon$
<code>\Theta</code>	$\Theta$	<code>\Phi</code>	$\Phi$
<code>\Lambda</code>	$\Lambda$	<code>\Psi</code>	$\Psi$
<code>\Xi</code>	$\Xi$	<code>\Omega</code>	$\Omega$
<code>\Pi</code>	$\Pi$		

Table 11 数式アクセント

命令	出力	命令	出力
<code>\hat{a}</code>	$\hat{a}$	<code>\tilde{a}</code>	$\tilde{a}$
<code>\check{a}</code>	$\check{a}$	<code>\bar{a}</code>	$\bar{a}$
<code>\breve{a}</code>	$\breve{a}$	<code>\dot{a}</code>	$\dot{a}$
<code>\acute{a}</code>	$\acute{a}$	<code>\ddot{a}</code>	$\ddot{a}$
<code>\grave{a}</code>	$\grave{a}$	<code>\vec{a}</code>	$\vec{a}$

また、数式アクセントのなかには次の Table 12 のようなものもあります。

Table 12 大きな数式アクセント

命令	出力
<code>\overline{x+y}</code>	$\overline{x+y}$
<code>\underline{x+y}</code>	$\underline{x+y}$
<code>\widehat{x+y}</code>	$\widehat{x+y}$
<code>\widetilde{x+y}</code>	$\widetilde{x+y}$
<code>\overbrace{x+y}</code>	$\overbrace{x+y}$
<code>\underbrace{x+y}</code>	$\underbrace{x+y}$
<code>\overrightarrow{x+y}</code>	$\overrightarrow{x+y}$
<code>\overleftarrow{x+y}</code>	$\overleftarrow{x+y}$

### 3.5.3 省略を表す3つの点

$\TeX$  には省略を表す記号がいくつか用意されています。一覧を Table 13 に示します。

Table 13 省略を表す3つの点

命令	出力	数式モード以外での使用
<code>\ldots</code>	$\dots$	可
<code>\cdots</code>	$\cdots$	不可
<code>\vdots</code>	$\vdots$	不可
<code>\ddots</code>	$\ddots$	不可

### 3.5.4 関係子と演算子

$\LaTeX$  は、Table 14 と Table 15 に示すような関係子と演算子を用意しています。

### 3.5.5 数学記号

数式中では、いくつかの独特の記号を用います。代表的なものには、プランク定数の  $h$ 、集合の濃度を表す  $\aleph$ 、無限大を表す  $\infty$ 、空集合を  $\emptyset$  などがあります。 $\LaTeX$  では、これらの数学記号も出力することができます。一覧を Table 16 に示します。

Table 14 関係子

命令	出力	命令	出力	命令	出力	命令	出力
<code>\le</code>	$\leq$	<code>\ge</code>	$\geq$	<code>\subset</code>	$\subset$	<code>\supset</code>	$\supset$
<code>\subseteq</code>	$\subseteq$	<code>\supseteq</code>	$\supseteq$	<code>\in</code>	$\in$	<code>\ni</code>	$\ni$
<code>\notin</code>	$\notin$	<code>\simeq</code>	$\simeq$	<code>\cong</code>	$\cong$	<code>\equiv</code>	$\equiv$

Table 15 演算子

命令	出力	命令	出力	命令	出力	命令	出力
<code>\ast</code>	$*$	<code>\times</code>	$\times$	<code>\div</code>	$\div$	<code>\pm</code>	$\pm$
<code>\cap</code>	$\cap$	<code>\cup</code>	$\cup$	<code>\bullet</code>	$\bullet$	<code>\circ</code>	$\circ$
<code>\cdot</code>	$\cdot$	<code>\oplus</code>	$\oplus$	<code>\otimes</code>	$\otimes$	<code>\odot</code>	$\odot$

### 3.5.6 大きな数学記号

数式では、次のようにある演算を添え字集合で行いたいという場合があります。

$$\sum_{k=1}^n a_k$$

このような大きな数学記号を出力する命令には、Table 17 に示すものがあります。

Table 17 大きな記号

命令	出力	命令	出力	命令	出力
<code>\sum</code>	$\sum$	<code>\prod</code>	$\prod$	<code>\int</code>	$\int$
<code>\oint</code>	$\oint$	<code>\bigcap</code>	$\bigcap$	<code>\bigcup</code>	$\bigcup$

### 3.5.7 関数

” $\log x$ ”と出力するつもりで” $\log x$ ”という原稿を書くと、出力は” $\log x$ ”になってしまいます。正しくは`\log`という命令を使って $\log x$ のように書きます。いくつかの関数や記号の命令を Table 18 に示します。

一般的に使用頻度の高い $\log$ や $\sin$ などは、その関数名のまえに`\`記号を付ければそのまま命令になります。

### 3.5.8 根号

根号を表すには`\sqrt`という命令を用います。例えば $\sqrt{x}$ と出力するには、`\sqrt{x}`と記述します。3乗根のように $\sqrt[3]{x}$ と出力したい場合には、オプション引数を使用して`\sqrt[3]{x}`とします。

根号を表すには、ほかに`\root`や`\of`という命令を使用することもできます。この場合、`\root`命令と`\of`命令とのあいだに記述された式が根号の肩に乗り、その後ろの1文字が根号のなかに収められます。1文字以上を根号のなかに収めるには、それを`\{と}`とでくらくなく

てはいけません。つまり、 $\sqrt[3]{x}$ と出力するには、”`\root 3 \of{x}`”のように記述します。

### 3.5.9 矢印類

数学では、数多くの矢印類を使用します。L<sup>A</sup>T<sub>E</sub>Xでは Table 19 に示すような矢印が用意されています。

Table 19 矢印類

命令	出力	命令	出力
<code>\gets(\leftarrow)</code>	$\leftarrow$	<code>\longleftarrow</code>	$\longleftarrow$
<code>\Leftarrow</code>	$\Leftarrow$	<code>\Longleftarrow</code>	$\Longleftarrow$
<code>\to(\rightarrow)</code>	$\rightarrow$	<code>\longrightarrow</code>	$\longrightarrow$
<code>\Rightarrow</code>	$\Rightarrow$	<code>\Longrightarrow</code>	$\Longrightarrow$
<code>\leftrightarrow</code>	$\leftrightarrow$	<code>\longleftrightarrow</code>	$\longleftrightarrow$
<code>\Leftrightarrow</code>	$\Leftrightarrow$	<code>\Longleftrightarrow</code>	$\Longleftrightarrow$

### 3.5.10 括弧と区切り記号

数式では、数多くの括弧と区切り記号を使用します。L<sup>A</sup>T<sub>E</sub>Xでは Table 20 に示すような括弧と区切り記号を利用することができます。

また、Table 21 のようにすることで括弧のサイズを指定することも可能です。

Table 16 数学記号

命令	出力	命令	出力	命令	出力	命令	出力
<code>\aleph</code>	$\aleph$	<code>\hbar</code>	$\hbar$	<code>\infty</code>	$\infty$	<code>\emptyset</code>	$\emptyset$
<code>\angle</code>	$\angle$	<code>\forall</code>	$\forall$	<code>\exists</code>	$\exists$	<code>\neg</code>	$\neg$

Table 18 数学関数

命令	出力	命令	出力	命令	出力	命令	出力
<code>\sin</code>	$\sin$	<code>\cos</code>	$\cos$	<code>\tan</code>	$\tan$	<code>\arcsin</code>	$\arcsin$
<code>\arccos</code>	$\arccos$	<code>\arctan</code>	$\arctan$	<code>\exp</code>	$\exp$	<code>\log</code>	$\log$
<code>\lim</code>	$\lim$	<code>\sec</code>	$\sec$	<code>\max</code>	$\max$	<code>\min</code>	$\min$
<code>\arg</code>	$\arg$	<code>\det</code>	$\det$				

Table 20 L<sup>A</sup>T<sub>E</sub>X で利用可能な括弧と区切り記号

原稿	出力	原稿	出力	原稿	出力	原稿	出力
<code>(</code>	$($	<code>)</code>	$)$	<code>[</code>	$[$	<code>]</code>	$]$
<code>\{</code>	$\{$	<code>\}</code>	$\}$	<code>\lfloor</code>	$\lfloor$	<code>\rfloor</code>	$\rfloor$
<code>\lceil</code>	$\lceil$	<code>\rceil</code>	$\rceil$	<code>\langle</code>	$\langle$	<code>\rangle</code>	$\rangle$
<code>/</code>	$/$	<code>\backslash</code>	$\backslash$	<code> </code>	$ $	<code>\ </code>	$\ $

Table 21 括弧のサイズを指定する命令 (開き括弧として扱われ, 括弧のまえに空白を空ける)

命令	無指定	<code>\bigl</code>	<code>\Bigl</code>	<code>\biggl</code>	<code>\Biggl</code>
使用例	<code>\(\$</code>	<code>\\$ \bigl(\$</code>	<code>\\$ \Bigl(\$</code>	<code>\\$ \biggl(\$</code>	<code>\\$ \Biggl(\$</code>
出力	$($	$($	$($	$($	$($

## 4 知っている便利な環境や命令

この節では、 $\text{T}_{\text{E}}\text{X}$  で文書を作成するとき知っている便利な環境や命令を紹介します。

### 4.1 screen 環境

screen 環境はその環境内に書かれた文章を丸い枠で囲むものです。コンピュータへの入力や出力、プログラムのソースなどをこの枠で囲むと文書中で目立たせることができるので便利です。

<ソースコード>

```
\begin{screen}
この中に書かれた文書は丸い枠で囲まれて出力されます。ただし、枠はページをまたげないので、そのあたりの注意が必要です。
\end{screen}
```

<出力結果>

この中に書かれた文書は丸い枠で囲まれて出力されます。ただし、枠はページをまたげないので、そのあたりの注意が必要です。

### 4.2 minipage 環境

minipage 環境は、その環境の中身を独立した小さな一つのページとして扱えるものです。minipage 環境を使うことで、図を2つ並べて横に表示したり、図の横にテキストを表示する、ということが可能になります。

サンプル1：図を2つ横に並べる

<ソースコード>

```
\begin{figure}[htbp]
\begin{center}
\begin{minipage}{0.4\linewidth}
\includegraphics[width=\linewidth]{img/figl.eps}
\caption{左の図}
\label{minipage1}
\end{minipage}
\hspace{2mm}
\begin{minipage}{0.3\linewidth}
\includegraphics[width=\linewidth]{img/figr.eps}
\caption{右の図}
\label{minipage2}
\end{minipage}
\end{center}
\end{figure}
```

サンプル2：図の横にテキストを配置する

<ソースコード>

<出力結果>

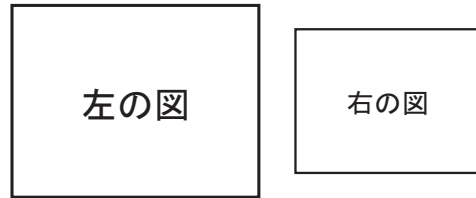
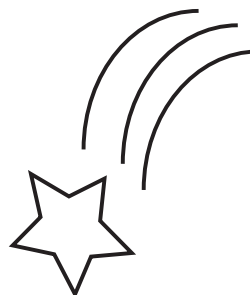


Fig. 9 左の図

Fig. 10 右の図

```
\begin{figure}[htbp]
\begin{center}
\begin{minipage}{0.4\linewidth}
\includegraphics[width=\linewidth]{img/star.eps}
\caption{流れ星の絵}
\label{star}
\end{minipage}
\hspace{2mm}
\begin{minipage}{0.5\linewidth}
左に書かれている図は、流れ星である。このサンプルのように、テキストも minipage 環境で囲まないとうまくいかないので、注意すること。
\end{minipage}
\end{center}
\end{figure}
```

<出力結果>



左に書かれている図は、流れ星である。このサンプルのように、テキストも minipage 環境で囲まないとうまくいかないで、注意すること。

Fig. 11 流れ星の絵

### 4.3 description 環境

description 環境は、見出し語付きで箇条書きを書くための環境です。

<ソースコード>

```
\begin{description}
\item[plain \TeX] D.E.Knuth が開発したマクロパッケージ
\item[\LaTeX] Leslie Lamport が開発したマクロパッケージ
```

```
\item[\LaTeXe] \LaTeX の最新バージョン
\end{description}
```

< 出力結果 >

plain  $\TeX$  D.E.Knuth が開発したマクロパッケージ  
 $\LaTeX$  Leslie Lamport が開発したマクロパッケージ  
 $\LaTeX 2\epsilon$   $\LaTeX$  の最新バージョン

上のサンプルにあるように見出し語にしたい語は [] で  
くることによって指定します。出力結果を見てもらう  
とわかると思いますが、各見出し語の後の解説は先頭部  
分はそろいません。

#### 4.4 list 環境

list 環境を使えば、自由にカスタマイズした箇条書き  
を書くことが可能です。

< ソースコード >

```
\begin{list}{ }{%
\setlength{\itemsep}{0pt}%
\setlength{\parsep}{0pt}%
}
\item 最初の項目
\item 次の項目
\item 最後の項目
\end{list}
```

< 出力結果 >

最初の項目  
次の項目  
最後の項目

list 環境では、上のサンプルのように第 1 引数に項目  
の前に表示する記号を指定し、第 2 引数では様々な設定  
の値を変更します。list 環境は使い方次第で非常に便利  
なものになります。詳しくは参考文献をご覧ください。

#### 4.5 設定値の変更

行送りを変更したり、行の幅を変えろといったことを  
行うためには setlength 命令を使って、各種の設定値を  
変更します。

< ソースコード >

```
\begin{flushleft}
\setlength{\baselineskip}{10pt}
現在、行送りをかなり小さめにとっています。よって、
次の行がかなりつまって表示されていると思います。な
お、この例のように行送りを変えたい部分だけを環境で
くるようにしないと設定以降すべてでこの現象が起き
てしまいます。
\end{flushleft}
```

< 出力結果 >

現在、行送りをかなり小さめにとっています。よって、  
次の行がかなりつまって表示されていると思います。  
なお、この例のように行送りを変えたい部分だけを環  
境でくるようにしないと設定以降すべてでこの現象  
が起きてしまいます。

#### 4.6 自分用のマクロの作成

$\TeX$  や  $\LaTeX$  は多くのマクロで構成されています。当  
然、自分でもマクロを定義して利用することができます。  
マクロを定義するためには、newcommand 命令を使いま  
す。マクロには、引数なしと引数ありの 2 種類のマク  
ロを定義することができます。

< ソースコード >

```
\newcommand{\dog}{犬は動物である}
\newcommand{\animal}[1]{#1は動物である}
\dog

\animal{猫}

\animal{馬}
```

< 出力結果 >

犬は動物である  
猫は動物である  
馬は動物である

上のサンプルでは、引数なしのマクロである dog と引  
数ありの animal を定義しています。animal は、引数を  
指定することで違った出力結果を得られているのがわか  
ります。このサンプルを見るとわかるように、引数の参  
照には、“#引数の番号”とします。

マクロを定義しておけば、文書中で何回も繰り返して  
書くような部分がある場合にマクロを修正するだけで文  
書全体が修正できるのでより効率的に文書を作成するこ  
とができるようになります。

#### 参考文献

- 1) 奥村晴彦． $\LaTeX 2\epsilon$  美文書作成入門．技術評論社，  
1997