第 4 回 LaTeX ゼミ

ゼミ担当者　：　降幡　建太郎，澤田　淳二，谷口　義樹
指導院生　：　水田　伯典，小椋　信弥
開催日　：　2002 年 5 月 14 日

ゼミ内容: 本ゼミでは，TeX で参考文献を引用する際の BIBTeX 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．また，mikilab スタイルの使用方法についても説明します．

1 BIBTeX

1.1 BIBTeX 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，TeX 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ことを考えます．これを行うのが BibTeX 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 TeX の文書への参照の仕方を説明します．

1.2 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. 1 のような書式にしたがって作成し，拡張子を.bib[1]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．

```
@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. 1　文献データベース (sample.bib)

Fig. 1 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. 1 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 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 TeX の本文に引用する方法を説明します．

1.3 参考文献の引用

ここでは，前節の Fig. 1 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. 2　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，デフォルトでは jplain となっていると思いますが，ここでは 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{ }

[1] 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。[1)]

参考文献

1) M.D. Spencer Johnson. チーズはどこへ消えた? 扶桑社, 2000.

Fig. 3　出力結果

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. 1.3 のように出力されます．

1.4　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を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．

1.5　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

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. 4)．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. 5 のようなメニューが表示されます．

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 = * ,
}
```

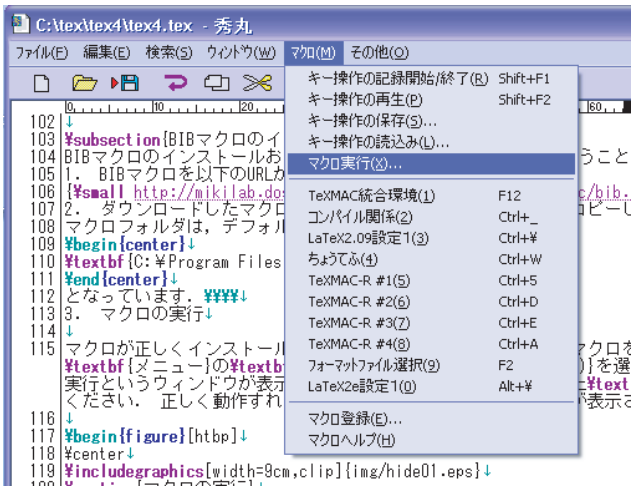


Fig. 4　マクロの実行

Fig. 5　マクロの選択

1.6　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.

1.7　TEXMAC による入力

秀丸のマクロである TEXMAC を利用することで，TeX 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．TeX ファイルを新規に作成する際に「Ctrl ＋ ¥」で TeX MAC 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. 6)．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 BIBTEX の説明は終わりです.

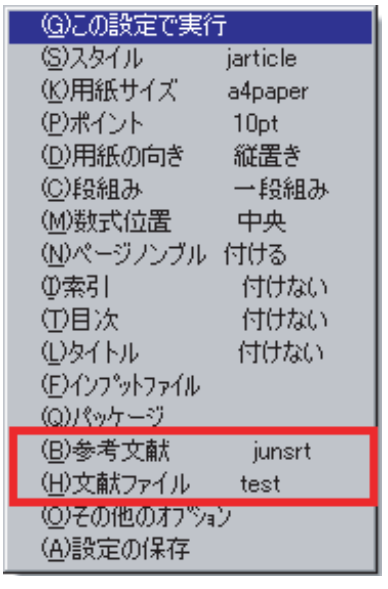


Fig. 6　TEXMAC の設定

Table 2　各属性とその意味

属性名	意味
address	publisher（発行元）の住所
annote	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．

2 数式

2.1 数式モード

LaTeX は通常、「段落モード」と呼ばれるモードか「LR モード」と呼ばれるモードのどちらかで動作しています。段落モードとは、単語や文などの原稿を文章のかたまりとして取り扱い、行や段落やページに分割していきます。これに対して、LR モードは表などを組版するときに使用され、このモードのなかでは改行が起こりません。

このほかに、LaTeX にはもう 1 つの動作モードがあります。それが、数式を取り扱う際の動作モードである「数式モード」です。以下では、この数式モードが段落モードや LR モードと異なる点について説明します。

2.1.1 空白

数式モードでは、TeX や LaTeX 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。

それでは、数式中で空白を制御する場合にはどうするのかというと、次ページの Table 3 のような命令を使用します。

TeX も LaTeX 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てくれますが、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があります。例えば、次の例を考えてみます。

$$(1)\ \int\int_D f(x,y)dxdy$$

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

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いと思います。これは、"dx"、"dy" のまえに ¥, 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 ¥! 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)dxdy ¥ ¥
¥ int ¥ !¥ !¥ ! ¥ int_D f(x,y)¥ ,dx¥ ,dy
¥ end{eqnarray*}
```

2.1.2 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、本文中のイタリック体と同様の出力を得るには、`"$¥mathit{diff}(a)$"` と書きます。同様に、"$\mathrm{Hom}_2(T)$" という出力を得たい場合には、`"$¥mathrm{Hom}_2(T)$"` のように記述しなければなりません。

ここで、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しますので、引数のなかも数式モードです。したがって、引き数のなか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です。

2.1.3 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、たとえば、数式中で `a^i_j` と書けば a^i_j のようになり、`¥int_a^b` と書けば $\int_a^b$ 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 LaTeX が自動的に決めますので、ユーザは気にする必要がありません。

また、$x^{(k-1)}_{j+1}$ のように、添え字が式のような複雑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} とでくくっておきます。

2.2 基本的な数式環境

LaTeX には様々なバリエーションの数式を出力することができるように、いくつかの数式環境が用意されています。ここでは、それらの数式環境のうち比較的使用頻度の高いものに絞って説明します。

2.2.1 math 環境

math 環境は、本文中に数式を出力する環境です。たとえば次のように使用します。

```
二次関数
¥begin{math}
y = x^2
¥end{math}
のグラフは放物線である。
```

上のソースの組版結果は以下のようになります。

二次関数 $y = x^2$ のグラフは放物線である。

しかし、本文中でいちいち math 環境を使用していると、文章のつながりがよくわからなくなってしまうので、普通は数式を $ と $ や、¥(と ¥) で囲んですませます。たいていの人は、$ と $ とを好んで使用しているようです。

つまり、先ほどの例は以下のようにも記述でき、同じ出力結果を得られます。

Table 3　数式モード内での空白制御命令

命令	空白	数式モード以外での使用
¥_	半角の空白	可
¥quad	全角の空白（クワタ）	可
¥qquad	２個のクワタ	可
¥,	細スペース（クワタの $\frac{1}{6}$）	可
¥>	中スペース（クワタの $\frac{2}{9}$）	不可
¥;	太スペース（クワタの $\frac{5}{18}$）	不可
¥!	負の細スペース（クワタの $-\frac{1}{6}$）	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	¥mathrm{A B C D E F}	ABCDEF
ボールド体	¥mathbf{A B C D E F}	**ABCDEF**
サンセリフ体	¥mathsf{A B C D E F}	ABCDEF
イタリック体	¥mathit{A B C D E F}	*ABCDEF*
タイプライタ体	¥mathtt{A B C D E F}	ABCDEF
カリグラフィック体	¥mathcal{A B C D E F}	$\mathcal{ABCDEF}$
明朝体	¥mathmc{A B C あ 亜}	*ABC* あ亜
ゴシック体	¥mathgt{A B C あ 亜}	*ABC* **あ亜**
標準的な書体	¥mathnormal{A B C D E F}	*ABCDEF*

```
二次関数 $y = x^2$ のグラフは放物線である。
```

2.2.2　displaymath 環境

displaymath 環境は、数式を別行立てで出力する環境です。別行立ての数式は標準では行の中央に出力されます。

同様の働きをする命令として ¥[と ¥] の組み合わせがあります。

以下に、displaymath 環境の使用例を示します。

```
以下に示すのが、displaymath 環境の出力例である。
¥begin{displaymath}
y = ax^2 + bx + c
¥end{displaymath}
これに対して次に示すのが、
数式を"¥texttt{¥yen}¥verb|[|"と
"¥texttt{¥yen}¥verb|]|"で
囲んだ出力例である。
¥[y = ax^2 + bx + c¥]
```

このソースの組版結果は以下のようになります。

以下に示すのが、displaymath 環境の出力例である。

$$y = ax^2 + bx + c$$

これに対して次に示すのが、数式を"¥["と"¥]"で囲んだ出力例である。

$$y = ax^2 + bx + c$$

通常の数式を入力するには、math 環境と displaymath 環境の２つを使用すれば十分でしょう。しかし、より本格的な数式を入力するためには、以降に示す環境を使用します。

2.2.3　eqnarray 環境

eqnarra 環境は別行立ての数式を複数出力するための環境です。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
jarticle	（数式番号）	(10)
jreport, jbook	（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、どの行にも番号を付けたくないなら、環境名をeqnarray*としてください。

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}
```

このソースの組版結果は以下のようになります。

初等関数とは、だいたい次の関数のことをいう。

$$y = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad (1)$$
$$y = \sin x \quad (2)$$
$$y = \cos x \quad (3)$$
$$y = \tan x \quad (4)$$
$$y = e^x \quad (5)$$
$$y = \log x \quad (6)$$

また eqnarray 環境では、上の例を若干変更するだけで、次のように出力することもできます。

初等関数とは、だいたい次の関数のことをいう。

$$\begin{aligned} y &= a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 && (7)\\ y &= \sin x && (8)\\ y &= \cos x && (9)\\ y &= \tan x && (10)\\ y &= e^x && (11)\\ y &= \log x && (12) \end{aligned}$$

この例では、各行の数式を出力において"="という関係子で揃えたわけです。このように出力するには、以下のように記述します。先ほどの例と比べると、=の両側に&記号を記述していることが確認できると思います。

```
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}
```

2.3 分数

LaTeX において分数を書く命令は¥frac で、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、上の例は次のように出力されます。

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

また、次のように本文中の数式で分数を用いると、別行立ての場合とは若干異なる書式で出力されます。

```
分数関数 $y=¥frac{1}{x+1}$について考えてみます。
```

出力は次のようになります。

分数関数 $y = \frac{1}{x+1}$ について考えてみます。

このように、分子と分母がかなり小さな文字で出力されてしまいます。「これでは字が小さすぎる」と思う人は、次のように記述するのも良いでしょう。

```
分数関数 $y=1/(x+1)$について考えてみます。
```

この出力結果は以下のようになります。

分数関数 $y = 1/(x+1)$ について考えてみます。

あるいは、次のように出力したい場合には、¥displaystyle という命令を使用します。¥displaystyle 命令は、本文中

の数式を別行立ての数式と同じスタイル（ディスプレイスタイル）で出力するための命令です。この命令は、分数に限らず任意の数式命令に効力を発揮します。

分数関数 $y = \frac{1}{x+1}$ について考えてみます。

この出力は次のソースから得られます。

```
分数関数 $¥displaystyle y = ¥frac{1}{x+1}$について
考えてみます。
```

2.4 行列

LaTeX で行列を表現する方法はいくつか存在します。ここでは、最も基本的といえる array 環境について説明します。

2.4.1 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&記号で表し、各行の区切りを ¥¥命令で表すところも tabular 環境と同じです。

なお、各行の最後の要素の後ろに&記号を書いてはいけません。また、最後の行の後ろに ¥¥命令を書いてもいけません。

2.4.2 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が、array 環境は行列要素を並べるだけの命令なので、括弧は出力しません。

そのため、括弧を出力するには以下に示すように、原稿中に (や) を記述する必要があります。

```
( ¥[
¥begin{array}{cc}
a & b ¥¥
c & d
¥end{array} )
¥]
```

この出力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 環境の場合は単純に括弧を記述するだけでは、非常に不格好な出力となります。

しかし、LaTeX には、このようなときに括弧の大きさを自動的に調整してくれる命令があります。それは、¥left および ¥right という命令です。以下にその使用例を示します。

```
¥[
¥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。

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

2.4.3 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。「中心線の位置」というのは、TeX が想定する組版方向（横組みなら水平方向）に数式を貫く不可視の線のことです。例えば、$-5+4=-1$ という数式があったとき、TeX 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 6 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。

TeX 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

Table 6　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

を得ることができます。

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

この出力は、次の原稿によって得ることができます。

```
¥[    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}
¥]
```

2.5　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。

2.5.1　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、オミクロンはアルファベットの O や o と同一なので、特別なフォントは用意されていません。一覧を Table 7 に示します。

また、ギリシャ文字の小文字のいくつかには、Table 8 に示すように Table 7 に示したのとは異なる異書体が存在しています。特に、ε や φ は、数学ではこちらを用いるほうが自然でしょう。

Table 7　ギリシャ小文字

命令	出力	命令	出力
¥alpha	α	¥nu	ν
¥beta	β	¥xi	ξ
¥gamma	γ	o	o
¥delta	δ	¥pi	π
¥epsilon	ϵ	¥rho	ρ
¥zeta	ζ	¥sigma	σ
¥eta	η	¥tau	τ
¥theta	θ	¥upsilon	υ
¥iota	ι	¥phi	ϕ
¥kappa	κ	¥chi	χ
¥lambda	λ	¥psi	ψ
¥mu	μ	¥omega	ω

Table 8　ギリシャ小文字の異体文字

命令	出力	命令	出力
¥varepsilon	ε	¥varrho	ϱ
¥vartheta	ϑ	¥varsigma	ς
¥varpi	ϖ	¥varphi	φ

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 9 に示します。

Table 9　ギリシャ大文字

命令	出力	命令	出力
¥Gamma	Γ	¥Sigma	Σ
¥Delta	Δ	¥Upsilon	Υ
¥Theta	Θ	¥Phi	Φ
¥Lambda	Λ	¥Psi	Ψ
¥Xi	Ξ	¥Omega	Ω
¥Pi	Π		

なお、Table 9 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まったく同一です。

2.5.2　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 10 に示します。

また、数式アクセントのなかには次の Table 11 のようなものもあります。

Table 10　数式アクセント

命令	出力	命令	出力
¥hat{a}	$\hat{a}$	¥tilde{a}	$\tilde{a}$
¥check{a}	$\check{a}$	¥bar{a}	$\bar{a}$
¥breve{a}	$\breve{a}$	¥dot{a}	$\dot{a}$
¥acute{a}	$\acute{a}$	¥ddot{a}	$\ddot{a}$
¥grave{a}	$\grave{a}$	¥vec{a}	$\vec{a}$

Table 11　大きな数式アクセント

命令	出力
¥overline{x+y}	$\overline{x+y}$
¥underline{x+y}	$\underline{x+y}$
¥widehat{x+y}	$\widehat{x+y}$
¥widetilde{x+y}	$\widetilde{x+y}$
¥overbrace{x+y}	$\overbrace{x+y}$
¥underbrace{x+y}	$\underbrace{x+y}$
¥overrightarrow{x+y}	$\overrightarrow{x+y}$
¥overleftarrow{x+y}	$\overleftarrow{x+y}$

2.5.3　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 12 に示します。

Table 12　省略を表す３つの点

命令	出力	数式モード以外での使用
¥ldots	$\ldots$	可
¥cdots	$\cdots$	不可
¥vdots	$\vdots$	不可
¥ddots	$\ddots$	不可

2.5.4　関係子と演算子

LATEX は、Table 13 と Table 14 に示すような関係子と演算子を用意しています。

2.5.5　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 $\hbar$、集合の濃度を表す $\aleph$、無限大を表す ∞、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 15 に示します。

2.5.6　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 16 に示すものがあります。

2.5.7　関数

”$\sin x$”と出力するつもりで”sin x”という原稿を書くと、出力は”$sinx$”になってしまいます。これでは非常に具合が悪いので、LATEX では、Table 17 に示すように、いくつかの関数や記号を命令として用意しています。

一般的に使用頻度の高い sin や log 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。

2.5.8　根号

根号を表すには ¥sqrt という命令を用います。たとえば $\sqrt{x}$ 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}とでくくらなくてはいけません。つまり、$\sqrt[n+1]{x+1}$ と出力するには、”¥root n+1 ¥of{x+1}”のように記述します。

2.5.9　矢印類

数学では、数多くの矢印類を使用します。LATEX では Table 18 に示すような矢印が用意されています。

Table 18　矢印類

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

2.5.10　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。LATEX では Table 19 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 20 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 13　関係子

命令	出力	命令	出力	命令	出力	命令	出力
¥le	$\leq$	¥ge	$\geq$	¥subset	$\subset$	¥supset	$\supset$
¥subseteq	$\subseteq$	¥supseteq	$\supseteq$	¥in	$\in$	¥ni	$\ni$
¥notin	$\notin$	¥simeq	$\simeq$	¥cong	$\cong$	¥equiv	$\equiv$

Table 14　演算子

命令	出力	命令	出力	命令	出力	命令	出力
¥ast	$*$	¥times	$\times$	¥div	$\div$	¥pm	$\pm$
¥cap	$\cap$	¥cup	$\cup$	¥bullet	$\bullet$	¥circ	$\circ$
¥cdot	$\cdot$	¥oplus	$\oplus$	¥otimes	$\otimes$	¥odot	$\odot$

Table 15　数学記号

命令	出力	命令	出力	命令	出力	命令	出力
¥aleph	$\aleph$	¥hbar	$\hbar$	¥infty	∞	¥emptyset	$\emptyset$
¥angle	$\angle$	¥forall	$\cup$	¥exists	$\bullet$	¥neg	$\neg$

Table 16　大きな記号

命令	出力	命令	出力	命令	出力
¥sum	$\sum$	¥prod	$\prod$	¥int	$\int$
¥oint	$\oint$	¥bigcap	$\bigcap$	¥bigcup	$\bigcup$

Table 17　数学関数

命令	出力	命令	出力	命令	出力	命令	出力
¥sin	sin	¥cos	cos	¥tan	tan	¥arcsin	arcsin
¥arctan	arctan	¥exp	exp	¥log	log	¥lim	lim
¥max	max	¥min	min	¥arg	arg	¥det	det

Table 19　LaTeX 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
(	$($	)	$)$	[	$[$	]	$]$
¥{	$\{$	¥}	$\}$	¥lfloor	$\lfloor$	¥rfloor	$\rfloor$
¥lceil	$\lceil$	¥rceil	$\rceil$	¥langle	$\langle$	¥rangle	$\rangle$
/	$/$	¥backslash	$\backslash$	¥|	$\vert$	¥¥|	$\Vert$

Table 20　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	¥bigl	¥Bigl	¥biggl	¥Biggl
使用例	$($	$¥bigl($	$¥Bigl($	$¥biggl($	$¥Biggl($
出力	$($	$\bigl($	$\Bigl($	$\biggl($	$\Biggl($

3　知っていると便利な環境や命令

この節では，TEX で文書を作成するときに知っていると便利な環境や命令を紹介します．

3.1　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．

3.2　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＞

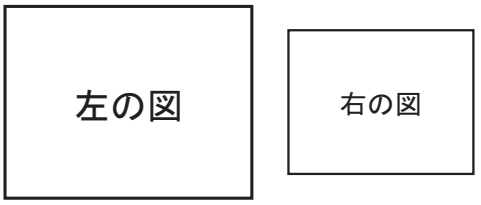
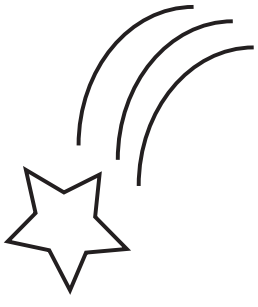


Fig. 7　左の図

Fig. 8　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ので，注意すること．

Fig. 9　流れ星の絵

3.3　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 2ε 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．

3.4 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．

3.5 設定値の変更

行送りを変更したり，行の幅を変える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．

3.6 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 2ε 美文書作成入門．技術評論社，1997