TEX 講習会資料 *

y.†

2016 年 8 月 11 日

最終更新日:2016 年 8 月 29 日

概要

TEX は理系文章の作成に広く使用されている組版(くみはん)ソフトです．本稿では数学文章の作成に焦点をあてて，初心者向けに TEX の基本的な使い方を解説します．基本的な数式の書き方に加え，込み入った TEX の知識や，現在では非推奨となっている書き方やその改善例なども示します．

目次

* この資料は，都数の TEX 講習会で使用した資料を公開用に修正したものです．

† @waidotto

1 TEX 入門

1.1 TEX とは？

TEX は理系文章の作成に広く使われている組版(typesetting) [*1] ソフトです．数学者・コンピュータ科学者の Donald E. Knuth が自著 *The Art of Computer Programming* の第 2 巻第 2 版を出版する際に，その当時のコンピュータ組版技術が職人技に比べ非常に見劣りするものであったため，活版印刷に劣らない美しい組版のできるコンピュータソフトウェアとして TEX を作り上げました．

TEX はギリシャ語の τέχνη(技術，芸術，technique) に由来する単語で，コンピュータで TEX のロゴが出力できないときは，大文字と小文字を混ぜて `TeX` と表記する約束になっています．TEX の読み方については，テフ/ˈtɛx/とかテック/ˈtɛk/とか発音されることが多いようです．

TEX には次のような特徴があります:

- TEX はフリーソフトウェアなので，無料で入手でき，自由に中身を調べたり改良したりできます．
- TEX は OS によらず同じ動作をします．つまり，入力が同じなら原理的には全く同じ出力が得られます．
- TEX の文書はプレインテキストなので，普通のテキストエディタで読み書きでき，再利用・データベース化が容易です．
- 孤立行処理や禁則処理などの種々の高度な組版技術が組込まれています．特に，数式の組版については定評があり，数式をテキスト形式で表す事実上の標準になっています．

TEX では上で述べたような組版技術のため，通常のワープロソフトと異なり，段落の最後に 1 文字追加するだけで段落の最初の改行位置が変わることさえあります．このような大域的な処理をキーボードから 1 文字入力するごとに行うのは，かなりの計算能力を必要とします.[*2]そのため TEX では，キーを打つたびに画面上の印刷結果のイメージを更新する方式 (WYSIWYG(What You See Is What You Get) 方式) ではなく，一括して全体を処理するバッチ処理を採用しています．

また，組版する前の**ソースコード** (source code) ファイルと組版結果が分離されていることで，

- ソースコードを好きなエディタで編集できる
- git や Subversion 等のバージョン管理システムによる管理が容易
- 強力なマクロ機能により，書式と内容を分離し，文書の意味の記述に集中できる

といった利点があります．

1.1.1 LATEX について

LATEX はコンピュータ科学者 Leslie Lamport によって機能強化された TEX です．もともとの TEX と同様，フリーソフトウェアとして配布されています．LATEX の読み方にも色々あるようですが，ラテフ/ˈlɑːtɛx/，ラ

[*1] 組版とは，印刷関係で活字を組んで版を作ることを意味する言葉です．(いわゆる「植字」のことです．)

[*2] TEX が作られた当初の計算機の話です．

テック/ˈlɑːtɛk/，レイテック/ˈleɪtɛk/などと発音されているようです.[*3]

最初の LaTeX は 1980 年代に作られましたが，1993 年には LaTeX 2ε(ラテック・ツー・イー) という新しい LaTeX ができ，現在では LaTeX といえば LaTeX 2ε を指すようになりました．TeX の場合と同様に，LaTeX, LaTeX 2ε のロゴが出力できない場合には，それぞれ `LaTeX`，`LaTeX2e` と表記します．

LaTeX の特徴は，文書の論理的な構造と視覚的なレイアウトとを区別して考えることができることです．例えば “Introduction” という節 (セクション) の見出しがあれば，ソースファイルには

```
\section{Introduction}
```

のように書いておきます．この `\section{...}` という命令が組版されるとき，文字の大きさは何ポイントか，書体はゴシックか明朝か，前後のアキは何ミリか……といった指定は，様式・判型ごとに別ファイル (クラスファイル，スタイルファイル) に登録されています．TeX 文書のソースファイルとクラスファイルは HTML と CSS の関係に近いと言うことができるでしょう．さらに LaTeX は章・節・図・表・数式などの番号を自動的に付けてくれますし，参照箇所には番号やページを自動挿入できます．

LaTeX は，TeX のマクロ機能を使って作られたものです．同じ LaTeX マクロでも，日本語の扱える TeX を使えば，日本語の扱える LaTeX になります．(株) アスキーが開発した日本語版の pTeX 用に修正した LaTeX が pLaTeX(pLaTeX 2ε) です．

以下，本稿では LaTeX と TeX とをとくに区別せずに呼ぶことにします．

1.1.2 その他の TeX の仲間

最初の TeX が使われてから三十数年が経ちました．この間も，TeX の処理系 (tex ファイルから dvi ファイルや pdf ファイルを生成するプログラム) に改良が加えられ，様々に派生しています．

- pTeX(`pTeX`) は TeX に日本語向けの様々な機能を追加した処理系です．縦書き組版や，日本語禁則処理をはじめとする様々な機能が追加されています．
- ε-TeX(`e-TeX`) は，TeX の種々のレジスタ (変数) の個数を 256 個から 32768 個に拡張したほか，右から左に組む機能などを追加したものです．
- ε-pTeX(`e-pTeX`) は pTeX に ε-TeX 拡張をしたものです．
- upTeX(`upTeX`) は pTeX の内部を Unicode 化したものです．これにより JIS 第一・第二水準以外の漢字も使えるようになります．
- ε-upTeX(`e-upTeX`) は upTeX と ε-TeX を合わせたものです．
- XeTeX(`XeTeX`) は ε-TeX を拡張したもので，入力ファイルは Unicode であり，さらに dvi ファイルの生成を諦めることで，フォントを TeX 側から直接的に操作できるようになっています．
- pdfTeX も ε-TeX を拡張したもので，名前の通り dvi ファイルを経由せずに直接 pdf ファイルを出力します．そのため，pdf ファイルを操作するための命令が TeX ファイルから使えるという利点があります．
- LuaTeX(`LuaTeX`) は pdfTeX に軽量スクリプト言語 Lua を組み込んだものです．これも Unicode に

[*3] 実は「レイテックス」という発音も可能です．実際，Lamport は著書 *LaTeX: A document Preparation System* の中で次のように述べています [5]:
“TeX is usually pronounced *teck*, making *lah*-teck, and *lay*-teck the logical choices; but language is not always logical, so *lay*-tecks is also possible.”

対応しています．日本語に対応した LuaTEX-ja の開発も進められています．

現在日本語組版では ε-pTEX が使われることが最も多いでしょう．実際，端末で `platex` とコマンドを打つと，

```
$ platex
This is e-pTeX, Version 3.14159265-p3.7-160201-2.6 (utf8.euc) (TeX Live 2016/Arch Linux) (preloaded format=platex)
 restricted \write18 enabled.
**
```

などと表示されます．

標準の TEX/LATEX を拡張するものとして，米国数学会 (American Mathematical Society) の $\mathcal{AMS}$-LATEX や，米国物理学会 (American Physical Society) の *Physical Review* などの LATEX パッケージが有名です．

本稿では，$\mathcal{AMS}$-LATEX の使用を前提とします．詳しくは 2 節で述べます．

1.2 TEX の仕組み

TEX で文書を作成する際には Texmaker などをはじめとする統合環境を利用することが多いと思います．そのため，普段はタイプセットボタンを押せば pdf が生成・プレビューされるので意識することは少ないと思いますが，TEX ソースファイルから pdf ファイルが生成されるまでの一連の流れを見ておきます．何かトラブルに遭遇した際には役立つかもしれません．

TEX は，組版結果を dvi ファイルという中間ファイルに書き出します．dvi は device independent(装置に依存しない) という英語の略です．この dvi ファイルを読み込んでパソコンの画面，各種プリンタ・写植機，および PostScript や pdf などのファイルとして出力するための専用ソフト (dvi ドライバ，dvi ウェア) が，出力装置や出力ファイル形式ごとに用意されていました．これは pLATEX 2_ε でも同じです．

現在では TEX 処理系は platex，dvi ドライバは dvipdfmx が使われていることが多いでしょう．

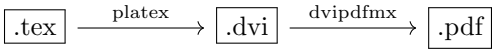

図 1: TEX による組版の流れ

また，`.tex` ファイルをコンパイルすると，次の 3 つの新しいファイルができます．

`.aux` ファイル　aux ファイルまたは補助 (auxilliary) ファイルと呼ばれるものです．LATEX の「相互参照」という機能のために用いられます．

`.log` ファイル　log ファイルと呼ばれるものです．実行の途中で画面に表示されるメッセージや，実行状態についての情報がここに書き込まれます．

`.dvi` ファイル　`.tex` ファイルの組版結果です．このファイルに，どの文字をどのページのどの位置に置くかという情報が書き込まれています．

1.3 簡単な例

初めの一歩として，次の TEX ソースをタイプセット (組版) してみましょう．

```
1 \documentclass{jsarticle}
2 \begin{document}
3 これは\TeX 文書のテストです.
```

```
\[ \int dx = x + C. \]
\end{document}
```

入力の際には次のような点に注意してください．

- ソースファイルの拡張子は.tex にしてください．メール等での不具合を避けるため，ファイル名は半角英数字にしておくのが無難でしょう．
- 英数字は全て直接入力 (いわゆる半角文字) で打ち込んでください．
- 大文字と小文字は区別されます．例えば，`\TEX` や `\Tex` と書いてはいけません．
- `\TeX` のうしろに句読点や括弧以外の文字が来る場合はスペースを空けてください．スペースを空けることで，“TeX” まででひとかたまりの命令であることが TEX に伝わります．スペースを空ける以外にも，`\TeX{}` や `{\TeX}` とする方法もあります．
- `\` (バックスラッシュ，逆斜線) がたくさんありますが，日本語キーボードでは円記号 (¥) で入力できるはずです．古い Windows では画面上の表示も円記号になる場合がありますが，特に問題はありません．(詳細は付録付録 B を参照してください．)
  バックスラッシュ自身を出力したい場合には `\textbackslash` コマンドを用います．(円記号を出力したい場合には textcomp パッケージの `\textyen` コマンドを用いると良いでしょう．パッケージについては 1.8 節を参照してください．)

正常にタイプセットできた場合，図 2 のような pdf ファイルが生成されるはずです．

これは TEX 文書のテストです．

$$\int dx = x + C.$$

図 2: TEX 文書の出力例

上の TEX ソースの簡単な意味は次の通りです．

- `\documentclass{jsarticle}` は文書を jsarticle という名前の書式 (ドキュメントクラス) で組版することを意味します．jsarticle 以外にも jsbook や report などがあり，それぞれ異なる組版結果が得られます．例えば，jsbook では奇数ページと偶数ページのレイアウトが異なったり，タイトルが単独ページに出力されたりしますが，jsarticle はそうではありません．
- `\begin{document}` はこれから文書が始まることを意味し，`\end{document}` は文書の終わりを意味します．`\documentclass{...}` と `\begin{document}` に挟まれた部分を**プリアンブル** (preamble) と呼び，様々なパッケージの読み込みや設定，マクロの定義などを行います．
- `\[ ... \]` で囲むと別行立ての数式が表示されます．数式については 2 節以降で詳しく扱います．

1.4 改行・空白の扱い

TEX においては，ソースファイル中の改行や空白 (スペース) は組版結果にそのまま反映されるわけではありません．基本的には，以下を覚えておけば大丈夫でしょう．

- 単独の改行は無視される
  ソースコード中の単独の改行の有無は組版結果に反映されません．したがって，ソースコードの読みやすさのため，1 行が長くなりすぎないように適度に改行を挟むことをお勧めします．正確にはこれは和文の場合だけで，行末が欧文の場合には改行により空白が挿入されます．欧文で空白を挿入させたくない場合には，行末に `%` を置きます．
- 空行では改段落される
  空行，すなわち 2 個以上連続する改行があるとそこで段落が改まります．空行がいくつ連続しても組版結果では改段落は 1 回だけで，垂直方向のスペースは生まれません．垂直方向のスペースを出力するには，`\vspace` などの専用の命令を使います．
- 2 個以上の空白は無視される
  基本的に空白をいくつ並べても，1 個だけ書いた場合と同じ出力結果になります．いわゆる全角スペースを用いると，書いた個数ぶんのスペースが出力されますが，全角スペースはエラーの元となりやすいので濫用は避けてください．半角空白をいくつも出力する場合には `\ ` (バックスラッシュの後ろに空白を入れる: \␣) で区切ります．水平方向に長い空白を確保したい場合には `\hspace` などの専用の命令を使いましょう．

1.5 タイトル・概要

もう少しそれらしくするために，文書にタイトルと概要を付けてみましょう．文章のタイトルを表示するには次の 4 つの命令を使用します．

- `\title{...}` タイトルの指定
- `\author{...}` 著者名の指定
- `\date{...}` 日付の指定
- `\maketitle` 上記 3 つの設定を出力

`\author{...\thanks{...}}` と書くと，所属やメールアドレスなどの注釈を書くことができます．`\texttt` 命令を使うと書体がタイプライター体になります．`\verb|...|` でもできます．

概要 (abstract) を出力するには，`\begin{abstract}` と `\end{abstract}` で囲みます．

例えば，次のように書きます．

```
\documentclass{jsarticle}
\begin{document}
\title{タイトル} %文書タイトルの指定
\author{都数花子\thanks{\texttt{address@example.com}}} %著者名とメールアドレスの指定
\date{2016年 8月 11日} %日付の指定
\maketitle %タイトルの出力
```

```
\begin{abstract}
    これは概要です.
\end{abstract}
ここから本文です.
\end{document}
```

`%` はその行のそれ以降を TEX に無視させる特殊な命令です．ソースファイルにコメント (注釈) を付けるのに便利です．あるいは，ソースの内容を実際には消さずに一時的に無効化したい場合に行の先頭に `%` を置いてコメント化すると便利です．(一般にこのようなテクニックを**コメントアウト** (comment out) と呼びます．)

また，`\today` コマンドを用いるとタイプセット時点の日付を出力することができるので便利です．

1.6 セクション・環境

1.6.1 セクション

セクションやサブセクション，段落といった文書の構造を記述するために次のような命令を使います．

`\section{...}`	セクション (節) の見出し
`\subsection{...}`	サブセクション (小節) の見出し
`\subsubsection{...}`	サブサブセクション (小々節) の見出し
`\paragraph{...}`	パラグラフ (段落) の見出し
`\subparagraph{...}`	サブパラグラフ (小段落) の見出し

1.6.2 環境

`\begin{...}`，`\end{...}` のように対になった命令を**環境** (environment) といいます．例えば，1.5 節の `abstract` は環境です．環境の内側は色々な設定が環境の外側と異なります．環境中で書体などを変えても，環境の外には影響が及びません．

まず，次の 4 つの環境を紹介しておきます．

quote 環境　引用
flushleft 環境　左寄せ
flushright 環境　右寄せ
center 環境　センタリング (中央揃え)

1.6.3 箇条書き

さらなる環境の例として，箇条書きの方法を紹介します．

itemize 環境　頭に ● などを付けた箇条書きです．
enumerate 環境　頭に番号を付けた箇条書きです．
description 環境　左寄せ太字で見出しを付けた箇条書きです．

例えば次のように使います．

```
\begin{itemize}
    \item 1つ目の項目
    \item 2つ目の項目
\end{itemize}
\begin{description}
    \item[1つ目] 説明 1
    \item[2つ目] 説明 2
\end{description}
```

- 1 つ目の項目
- 2 つ目の項目

1 つ目　説明 1

2 つ目　説明 2

`enumerate` は `itemize` と同様です．

1.7　約物・アクセント

1.7.1　約物

約物（やくもの）とは，句読点・引用符・疑問符・括弧などの記号の総称です．ここでは TEX における種々の約物の出力の仕方を紹介します．

まず，TEX における引用符は，バッククォート `` ` `` (日本語キーボードでは Shift + @) とシングルクォート `'` を用いて `` `...' `` あるいは `` ``...'' `` と書きます．シングルクォートやダブルクォート `"` のみで囲むのは誤りです．

(誤)	`'ABC'`	'ABC'
(誤)	`"ABC"`	"ABC"
(正)	`` `ABC' ``	‘ABC’
(正)	`` ``ABC'' ``	“ABC”

あるいは，和文の場合には全角引用符“……”を用いることもできます．

TEX で波括弧 (中括弧){} をそのまま出力したい場合には `\{ \}` と，手前に `\` を置きます．

ハイフン・ダッシュの類については，次の 4 つのものがあります．

`-`	-	ハイフン
`--`	–	エヌダッシュ
`---`	—	エムダッシュ
`$-$`	−	マイナス (数式)

まず，ハイフンは $\mathbb{C}$-vector space や p-adic number のように，単語同士を繋げるときに用います．エヌダッシュは ‘n’ と同じ幅のダッシュで，例えば 10–20 のように範囲や期間を表すのに用います．(ハイフンを

使って 10-20 とすると，「10 の 20」という意味になってしまいます．) エムダッシュは ‘m’ と同じ幅のダッシュで，文中で間を置いて読むべきところ—例えば説明的な部分の区切り—に用います．あるいは，タイトルとサブタイトルの区切りにも使います．和文でエムダッシュでは短すぎると思う場合には倍角ダッシュ (—) を 2 つ重ねて使うこともできます．マイナスは数式であり，単語や文の区切りの記号としては使えません．

1.7.2 アクセント

人名等，欧文で使用する種々のアクセント類を出力する命令を列挙しておきます．

表 1: アクセント類

`\‘o`	ò	`\~o`	õ	`\v{o}`	ǒ	`\d{o}`	ọ
`\’o`	ó	`\=o`	ō	`\H{o}`	ő	`\b{o}`	o̲
`\^o`	ô	`\.o`	ȯ	`\t{oo}`	o͡o	`\r{a}`	å
`\"o`	ö	`\u{o}`	ŏ	`\c{o}`	o̧		

数式環境を使って `Schr$\ddot{o}$dinger` などとしないようにしてください．正しくは `Schr\"odinger` とします．

`\` にアルファベット以外の記号の付く命令では `{ }` はあってもなくても構いません．

1.8 パッケージ

パッケージとは，LaTeX の機能を簡単に拡張するための仕組みです．例えば，ルビ (振り仮名) を振る命令を使えるようにするために，okumacro パッケージを使うことにします．そのためには，プリアンブルに

```
\usepackage{okumacro}
```

と書いておきます．こうしておくと，LaTeX が `okumacro.sty` というファイルを読み込んで，その中にある命令の定義を取り込み，`\ruby{...}{...}` という命令が使えるようになります．

`\ruby{ 漢字 }{ かんじ }` → 漢字(かんじ)

以降，単に「*****パッケージを使う」と書いたら，プリアンブルに `\usepackage{*****}` と書くことを意味するものとします．

1.9 表組み

まず最初に，プリアンブルに

```
\usepackage{array,booktabs,float}
```

と書いておきます．

表を出力するには，例えば次のようにします．

```
1 \begin{table}[ht]
```

```
    \centering
    \caption{用紙の種類と寸法 (A 列)}\label{table:a-paper}
    \begin{tabular}{crr} \toprule
        呼び方 & 横 (mm) & 縦 (mm) \\ \midrule
        A0 & 841 & 1 189 \\
        A1 & 594 & 841 \\
        A2 & 420 & 594 \\
        A3 & 297 & 420 \\
        A4 & 210 & 297 \\ \bottomrule
    \end{tabular}
\end{table}
```

表 2: 用紙の種類と寸法 (A 列)

呼び方	横 (mm)	縦 (mm)
A0	841	1 189
A1	594	841
A2	420	594
A3	297	420
A4	210	297

table 環境は，表を自動配置するための環境です．[ht] は表の出力可能な位置の指定で，以下のものが使えます．

- t　ページ上端 (top) に出力する
- b　ページ下端 (bottom) に出力する
- p　単独ページ (page) に出力する
- h　できればその位置 (here) に出力する
- H　必ずその位置 (Here) に出力する

何も指定しなければ [tbp]，すなわちページ上端，ページ下端，単独ページに出力できることになります．htbp を並べる順序に意味はありません．H を指定したら他のオプションは指定できません．

\centering は，表を用紙の中央に表示するための命令です．

\caption は，表のキャプション (説明) を出力する命令です．

\label は，相互参照のためのラベルを設定する命令です．中身は自分で好きな文字列を設定して構いません．1.11 節で詳しく扱います．

tabular 環境は実際に表を出力します．{crr} と書いてある部分は列指定といい，l(左寄せ，left)，c(中央，center)，r(右寄せ，right) を列の数だけ並べます．各セルは & を用いて区切り，\\ で改行します．\toprule，\midrule，\bottomrule で罫線を引きます．

日本語の文章に多く見られるように，縦の罫線も出力するには次のようにします．

```
\begin{table}[ht]
```

```
    \centering
    \caption{用紙の種類と寸法 (JIS B 列)}\label{table:b-paper}
    \begin{tabular}{|c||r|r|} \hline
        呼び方 & 横 (mm) & 縦 (mm) \\ \hline\hline
        B0 & 1030 & 1456 \\ \hline
        B1 & 728 & 1030 \\ \hline
        B2 & 515 & 728 \\ \hline
        B3 & 364 & 515 \\ \hline
        B4 & 257 & 364 \\ \hline
        B5 & 182 & 257 \\ \hline
    \end{tabular}
\end{table}
```

表 3: 用紙の種類と寸法 (JIS B 列)

呼び方	横 (mm)	縦 (mm)
B0	1030	1456
B1	728	1030
B2	515	728
B3	364	515
B4	257	364
B5	182	257

列指定に `|` と書くとその部分に罫線が引かれ，`||` と書くと二重線が引かれます．`\hline` で太さが一定の罫線を引くことができ，2 つ重ねると二重線になります．

表のキャプションを「表 1」「表 2」……ではなく，「Table 1」「Table 2」……にするには，jsarticle では `\documentclass[english]{jsarticle}` と `english` オプションを付けます．図についても同様です．

1.10 図の挿入

写真などの画像ファイルを挿入するには graphicx パッケージを使います．eps, pdf, jpeg, png, bmp などの代表的な形式の画像ファイルを表示することができます．

プリアンブルの先頭で `\documentcalss[dvipdfmx]{jsarticle}` とドライバを指定しておくとトラブルが減るでしょう．tex ソースと同じディレクトリに配置してある `imgres.jpeg` という画像ファイルを出力するには次のようにします．

```
\begin{figure}[ht]
    \centering
    \includegraphics[width=5cm]{imgres.jpeg}
    \caption{\texttt{@tosuu2015}のアイコン}\label{fig:twitter-icon}
\end{figure}
```

図 3: @tosuu2015 のアイコン

命令の説明に関してはほとんど表組みの場合と同じです．`\includegraphics` は画像を表示する命令で，オプションで画像のサイズなどを設定できます．`width` の他，`height`，`scale`，`angle` などが使えます．

文書中の画像が多くなってくると，毎回の組版に非常に時間がかかるようになってしまいます．そのようなときは `\usepackage[draft]{graphicx}` と `draft` オプションを指定しておくと，画像の部分が枠とファイル名に置き換えられ，組版が高速になります．

Linux 等の一部環境で “`! LaTeX Error: Cannot run pipe command. Try --shell-escape`” などとエラーが出る場合には，端末から`$ ebb -x imgres.jpeg` などとして Bounding Box の情報を取得・出力すると解決するかもしれません．

1.11 相互参照・目次

相互参照とは，ページ・章・節・図・表・数式などの参照時，番号付けを自動的に行う機能のことです．

まず，参照したい番号を出力する命令 (`\section`，`\caption` や数式など) の直後に，`\label` コマンドを使ってラベルを貼ります．ラベル名には英数字や `:`，`-`，`_` などの記号を使うことができます．空白を含むラベル名はトラブルの元となるので避けましょう．ラベル名は大文字と小文字を区別するので注意してください．

そしてラベルを参照したいところで (ラベルを参照する箇所の文書内での位置が `\label` より手前でも構いません)，`\ref`，`\pageref`，`\eqref` のいずれかを使います．

例えば，1.9 節の表 2 を参照したいときは次のようにします．

`表 \ref{table:a-paper}(\pageref{table:a-paper} ページ) を参照`　表 2(11 ページ) を参照

2.2 節の式 (1) を参照するには，次のようにします．`\eqref` を用いると，自動的に括弧が付きます．

`\eqref{eq:euler}`　(1)

相互参照をしたときは，組版処理を 1 回行うだけでは正しい出力が得られません．1 回行っただけだと「表??を参照」などと “??” で表示されます．組版処理を 2 回 (あるいは長い文書では 3 回) 実行することで正しい出力になります．何回処理しても “??” のままである場合は，ラベル名を間違えている可能性が高いです．

目次を出力するには，目次を出力したい場所 (例えば概要の後) に `\tableofcontents` と書いておくだけです．同様に，`\listoffigures`，`\listoftables` というコマンドで図目次，表目次が出力できます．

1.12　文献の参照

1.12.1　半自動で行う方法

参考文献リストを作る 1 つ目の方法は，thebibliography 環境を利用することです．例えば，文書の終わりに次のように書いておきます．

```
\begin{thebibliography}{9}
    \bibitem{木是} 木下是雄『理科系の作文技術』中公新書 624(中央公論社，1981)
    \bibitem{leu} Mary-Claire van Leunen. \textit{A Handbook for Scholars}. Alfred A.
        Knopf，1978.
\end{thebibliography}
```

`{9}` は参照文献に付ける “番号” が最も長いものと同じ幅の文字列であれば何でも構いません．例えば `{1}` でも `{a}` でも意味は同じです．

`\bibitem{...}` には適当な参照名を付けておきます．日本語の文字も使用可能です．参照文献の “番号” を変更するには，例えば `\bibitem[木下 81]{ 木是 }` などとします．

本文中で文献を参照するには `\cite` を用います．

```
木下~\cite{ 木是 } や van Leunen~\cite{leu} は
```

木下 [1] や van Leunen [2] は

`\cite` の前の `~` は行分割をしない空白 (いわゆる no-break space) であり，文献番号が行頭に来ないようにする働きをします．

1.12.2　全自動で行う方法

BibTeX(pBibTeX) を用いると，参考文献の処理をすべて自動でやってくれます．また，文献ファイルが分離されているため，再利用も容易になります．

例えば，tex ファイルと同じディレクトリに myrefs.bib というファイルを作り，次のような内容を書きます．

```
@article{oda,
    yomi = "Tadao Oda",
    author = "小田 忠雄",
    title = "数学の常識・非常識---由緒正しい{\TeX}入力法",
    journal = "数学通信",
    volume = 4,
    number = 1,
    pages = "95--112",
    year = 1995,
    url = "http://www.math.tohoku.ac.jp/tmj/oda_tex.pdf" }
@book{okumura,
    yomi = "Haruhiko Okumura",
    author = "奥村 晴彦,黒木 裕介",
    title = "{\LaTeXe}美文書作成入門",
```

```
15    publisher = "技術評論社",
16    year = 2013,
17    edition = 6 }
```

また，tex ファイルの後ろの方に

```
1 \bibliographystyle{jplain}
2 \bibliography{myrefs}
```

と書いておきます．参照の仕方は thebibliography 環境を用いる場合と同様で，`\cite{okumura}` とすれば奥村先生の本 [10] が引用できます．

この状態で，platex，pbibtex，platex，platex の順で処理すると参考文献リストが出力されます．`{jplain}` の代わりに `{japlha}` と書いておくとまた違った出力が得られます．

1.13 エラーメッセージ

ここでは，よく遭遇するエラーメッセージを紹介します．まず，1 つ目は次のようなものです．

```
1 ! Undefined control sequence.
2 l.32 \hoge
```

これは `\hoge` という文字列を TEX が理解できないためにエラーが起きています．`l.32` の部分は，エラーが 32 行目で起きたことを表しています．このエラーの原因としては，単純なスペルミスか，`\usepackage` のし忘れが考えられます．

2 つ目は次のようなものです．

```
1 ! Missing $ inserted.
2 <inserted text>
3                 $
4 l.56
```

このエラーは，数式を表す `$...$` の閉じ忘れや，数学記号を表す命令を数式環境外で使った場合に発生します．

3 つ目は次のようなものです．

```
1 ! Extra alignment tab has been changed to \cr.
```

これは，表組みの際に，列指定の列数よりも実際の列数が多いときに発生します．

4 つ目は次のようなものです．

```
1 ! LaTeX Error: This file needs format ‘pLaTeX2e’
2               but this is ‘LaTeX2e’.
```

これは，jsarticle を使っている日本語文書を，platex ではなく latex や pdflatex などで処理してしまった際に起きるエラーです．TEX 統合環境を使っている場合には，platex や pdfplatex などで処理するように設定されているか確認してください．

2　数式の基礎

数学記号に限らず，TEX では夥しい数の記号が用意されています．本稿ではその一部しか紹介できませんが，よりたくさんの記号を知りたければ，記号の一覧 [11] を参照してください．

以降，amsmath パッケージの利用を前提とします．プリアンブルに

```
\usepackage{amsmath,amssymb}
```

と書いておいてください．

2.1　基本事項

TEX で文中に数式を出力するには，`$...$` で囲みます (インライン数式)．別行立ての数式 (ディスプレイ数式) を出力するには，`\[...\]` で囲みます．

`$...$` や `\[...\]` で囲まれた内側は数式モードとなり，数学記号を出力するための様々な命令が使用可能になります．

数式を書くとき，ソースコード中のスペースは出力結果に影響を与えません．

`$x+(-y)=x-y$`	$x+(-y)=x-y$
`$x + ( - y ) = x - y$`	$x+(-y)=x-y$

そのため，適度にスペースを空けて見やすくするように心掛けましょう．

数式中に強制的にスペースを入れる命令には次のようなものがあります．

`$AA$`	AA	
`$A \quad A$`	$A \quad A$	
`$A \qquad A$`	$A \qquad A$	\quad の 2 倍
`$A \, A$`	$A\,A$	\quad の 3/18 倍
`$A \> A$`	$A\:A$	\quad の 4/18 倍
`$A \; A$`	$A\;A$	\quad の 5/18 倍
`$A \! A$`	$A\!A$	\quad の -3/18 倍
`$A \mspace{10mu} A$`	$A\mspace{10mu}A$	1 mu は \quad の 1/18 倍

以降，とくに誤解の虞のない場合には，数式環境であることを表す `$` や `\[ \]` は省略します．

2.1.1　書体

数式の書体を変更するには，`\math***` という形の命令を使います．`\mathscr` を使うには，mathrsfs パッケージが必要です．`\mathcal`，`\mathscr` には小文字はありません．bbm パッケージを使うと，`\mathbbm` で小文字や数字の太字も出せるようになります．

斜体 (italic)	`\mathit{ABCDEFGabcdefg}`	$\mathit{ABCDEFGabcdefg}$
ローマン体 (roman)	`\mathrm{ABCDEFGabcdefg}`	$\mathrm{ABCDEFGabcdefg}$
太字 (boldface)	`\mathbf{ABCDEFGabcdefg}`	$\mathbf{ABCDEFGabcdefg}$
サンセリフ体 (sans-serif)	`\mathsf{ABCDEFGabcdefg}`	$\mathsf{ABCDEFGabcdefg}$
タイプライター体 (typewriter)	`\mathtt{ABCDEFGabcdefg}`	$\mathtt{ABCDEFGabcdefg}$
黒板太字 (blackboard)	`\mathbb{ABCDEFGHIJKLMN}`	$\mathbb{ABCDEFGHIJKLMN}$
カリグラフィー体 (calligraphy)	`\mathcal{ABCDEFGHIJKLMN}`	$\mathcal{ABCDEFGHIJKLMN}$
フラクトゥール (fraktur)	`\mathfrak{ABCDEFGabcdefg}`	$\mathfrak{ABCDEFGabcdefg}$
花文字 (script)	`\mathscr{ABCDEFGHIJKLMN}`	$\mathscr{ABCDEFGHIJKLMN}$

数式モードでは元から斜体ですが，`\mathit` を使うと文字間の幅が一定になります．

`difference`	$difference$
`\mathit{difference}`	$\mathit{difference}$

ベクトル等の太字を出力するには，`\boldsymbol` を用いると出力できる他，bm パッケージの `\bm` 命令を用いることもできます．(`\mathbf` を用いると斜体にならない上，ギリシャ文字が太字になりません．)

`\mathbf{F}(\mathbf{x}, \mathbf{\alpha})`	$\mathbf{F}(\mathbf{x}, \alpha)$
`\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\alpha})`	$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\alpha})$

また，数式書体ではありませんが，`\textsc` 命令を用いると，小文字を小さい大文字で表示する (small capital) ことができます．

`\textsc{Mathematics}`	$\textsc{Mathematics}$

また，数式中に文章を挿入するには `\text` 命令を使います．

`V_{\text{out}} = \text{const.}`	$V_{\text{out}} = \text{const.}$

2.1.2 添字

上付き添字 (累乗) を出力するには `^` を，下付き添字を出力するには `_` を使います．添字が複数文字になる場合には，`{ }` を用いてグルーピングを行います．添字の添字などを出力する場合には，`{ }` を用いて結合の仕方を指定しないと“`! Double subscript.`”とエラーになります．よくわからない場合にはとりあえず `{ }` で囲んでおけば間違いはないでしょう．上下共に添字を付ける場合は `{ }` による結合の指定は不要で，添字の順序は `x_d^u` と `x^u_d` のどちらでも構いません．転置行列や組み合せなどで左上や左下に添字を出したい場合には，`{}^t` や `{}_n` などとすれば良いです．ただし，左上に添字を出すにはもっと良い方法があります．2.3 節を参照してください．

`x^2`	x^2	`x_i_1`	→[エラー]
`x_2`	x_2	`x_{i_1}`	x_{i_1}
`x^10`	x^10	`{}^tA`	${}^tA$
`x^{10}`	x^{10}	`{}_n\mathrm{C}_r`	${}_n\mathrm{C}_r$

2.1.3 分数

分数は `\frac{ 分子 }{ 分母 }` で出力できます．分子・分母がともに 1 桁の数字であるときなど，`{ }` を省略できる場合があります．分数や総和，積分の記号などはディスプレイ数式とインライン数式では表示が異なります．インライン数式中でどうしてもディスプレイ数式のように出力させたい場合には，(推奨はされませんが) `\displaystyle` を用いることで実現できます．(逆は `\textstyle` を用います．)

`\[ \frac{1 - x^2}{1 + x^2} \]`	$\frac{1 - x^2}{1 + x^2}$
`\[ \frac12 \]`	$\frac12$
`$\frac{1 - x^2}{1 + x^2}$`	$\frac{1 - x^2}{1 + x^2}$
`$\frac12$`	$\frac12$

2 項係数を出力するには，`\binom` 命令を用います．

`\binom{n}{k}`	$\binom{n}{k}$

分数や 2 項係数，Christoffel 記号を含む一般の分数を出力するには，`\genfrac` 命令を用います．`\genfrac{ 左括弧 }{ 右括弧 }{ 横棒の太さ }{ スタイル }{ 分子 }{ 分母 }` のようにして使います．太さを 0 にすれば横棒は出力されなくなります．

`\genfrac{}{}{}{}{p}{q}`	$\genfrac{}{}{}{}{p}{q}$
`\genfrac{\{}{\}}{0pt}{}{i}{j\,k}`	$\genfrac{\{}{\}}{0pt}{}{i}{j\,k}$

2.1.4 ギリシャ文字

ギリシャ文字の小文字は英語名の前に `\` を付ければ出力できます．ただし o(omicron) だけは英語のオーと同じなので特に用意されていません．

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

一部の小文字には変体文字 (variant) が用意されています．

`\varepsilon`	ε	`\varrho`	ϱ
`\vartheta`	ϑ	`\varsigma`	ς
`\varpi`	ϖ	`\varphi`	φ

大文字は，次の 11 通り以外は英語のアルファベットの大文字と同じです．

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

数式中のギリシャ文字は，習慣に従って，小文字だけ斜体になります．したがって，例えばベータ関数などを書くときは `B(p, q)` ($B(p,q)$) ではなく，`\mathrm{B}(p, q)` ($\mathrm{B}(p,q)$) と書いた方が良いでしょう．ギリシャ文字の大文字も斜体にしたいときは，例えば `\varGamma` と書けば $\varGamma$ が出力できます．他の文字も同様です．

2.1.5　2 項演算子

以下に 2 項演算子の例をいくつか示します．$\pm a$ のように，単項演算子として用いることもあります．

`+`	$+$	`\circ`	$\circ$	`\vee`, `\lor`	$\vee$	`\oplus`	$\oplus$
`-`	$-$	`\bullet`	$\bullet$	`\wedge`, `\land`	$\wedge$	`\ominus`	$\ominus$
`\pm`	$\pm$	`\cdot`	$\cdot$	`\setminus`	$\setminus$	`\otimes`	$\otimes$
`\mp`	$\mp$	`\cap`	$\cap$	`\wr`	$\wr$	`\oslash`	$\oslash$
`\times`	$\times$	`\cup`	$\cup$	`\diamond`	$\diamond$	`\odot`	$\odot$
`\div`	$\div$	`\uplus`	$\uplus$	`\bigtriangleup`	$\bigtriangleup$	`\bigcirc`	$\bigcirc$
`*`	$*$	`\sqcap`	$\sqcap$	`\bigtriangledown`	$\bigtriangledown$	`\dagger`	$\dagger$
`\ast`	$\ast$	`\sqcup`	$\sqcup$	`\triangleleft`	$\triangleleft$	`\ddagger`	$\ddagger$
`\star`	$\star$			`\triangleright`	$\triangleright$	`\amalg`	$\amalg$

`\wedge`，`\land` は同じ記号を出力しますが，`\wedge` は楔 (くさび) 積，`\land` は論理積など，意味によって使い分けた方が良いでしょう．

2.1.6　関係演算子

LaTeX および amsmath パッケージでは膨大な量の関係演算子が用意されていますが，ここでは代表的なものをいくつか抜き出して示します．

`<`	$<$	`>`	$>$	`\subset`	$\subset$	`\supset`	$\supset$
`\le`, `\leq`	$\leq$	`\ge`, `\geq`	$\geq$	`\subseteq`	$\subseteq$	`\supseteq`	$\supseteq$
`\leqq`	$\leqq$	`\geqq`	$\geqq$	`\sqsubseteq`	$\sqsubseteq$	`\sqsupseteq`	$\sqsupseteq$
`\prec`	$\prec$	`\succ`	$\succ$	`\vdash`	$\vdash$	`\dashv`	$\dashv$
`\preceq`	$\preceq$	`\succeq`	$\succeq$	`\in`	$\in$	`\ni`	$\ni$
`\ll`	$\ll$	`\gg`	$\gg$	`\notin`	$\notin$		
`=`	$=$	`\sim`	$\sim$	`\propto`	$\propto$	`\parallel`	$\parallel$
`\equiv`	$\equiv$	`\simeq`	$\simeq$	`\models`	$\models$	`\bowtie`	$\bowtie$
`\neq`	$\neq$	`\asymp`	$\asymp$	`\perp`	$\perp$	`\smile`	$\smile$
`\doteq`	$\doteq$	`\approx`	$\approx$	`\mid`	$\mid$	`\frown`	$\frown$
		`\cong`	$\cong$	`:`	$:$		

関係記号に斜線を引くには，手前に `\not` を付けます．

`\not\equiv` $\not\equiv$

2.1.7　括弧・矢印・区切り記号

括弧類・区切り記号には次のようなものがあります．

`(x)`	(x)	`\lfloor x \rfloor`	$\lfloor x \rfloor$
`[x]`	$[x]$	`\lceil x \rceil`	$\lceil x \rceil$
`\{x\}`	$\{x\}$	`\langle x \rangle`	$\langle x \rangle$
`/`	$/$	`\uparrow`	$\uparrow$
`\backslash`	$\backslash$	`\Uparrow`	$\Uparrow$
`|`	$\|$	`\downarrow`	$\downarrow$
`\\|`	$\|\|$	`\Downarrow`	$\Downarrow$
`\updownarrow`	$\updownarrow$	`\Updownarrow`	$\Updownarrow$

`|` は `\vert`，`\|` は `\Vert` と書くこともできます．

これらの記号の大きさを変更するには，左右の区別のないものは `\big`，2 項関係は `\bigm`，開き括弧は `\bigl`，閉じ括弧は `\bigr` などの命令を用います．

```
\Biggl( \biggl( \Bigl( \bigl( ( x ) \bigr) \Bigr) \biggr) \Biggr)
```
$\Biggl(\biggl(\Bigl(\bigl((x) \bigr) \Bigr) \biggr) \Biggr)$

`\left`，`\right` 命令を使うと，括弧の中身に応じて大きさが自動で変わるので便利です．片方だけ括弧を付けたいときは，もう片方をピリオドにします．

```
\[ (\frac12) \]
\[ \left(\frac12\right) \]
\[ \left.\frac{d(f \circ c)}{dt}\right\rvert_{t = 0} \]
```

$$(\frac{1}{2})$$
$$\left(\frac{1}{2}\right)$$
$$\frac{d(f \circ c)}{dt}\Big|_{t=0}$$

`|` は絶対値などの左右の組になったデリミタとして用いることは推奨されません．実際，`|` でマイナスの絶対値を出力しようとすると 2 項演算とみなされて出力がおかしくなります．これを回避するには `\lvert ... \rvert` を用います．ノルムなどについても同様で，`\|` ではなく `\lVert ... \rVert` を用います．

`|-a|`	$\|-a\|$	`\\|-a\\|`	$\\|-a\\|$
`\lvert -a \rvert`	$\lvert -a \rvert$	`\lVert -a \rVert`	$\lVert -a \rVert$

2.1.8 矢印

矢印は，括弧類のところで挙げたもの以外に，次のようなものがあります．

`\leftarrow`, `\gets`	$\leftarrow$	`\longleftarrow`	$\longleftarrow$
`\Leftarrow`	$\Leftarrow$	`\Longleftarrow`	$\Longleftarrow$
`\rightarrow`, `\to`	$\rightarrow$	`\longrightarrow`	$\longrightarrow$
`\Rightarrow`	$\Rightarrow$	`\Longrightarrow`	$\Longrightarrow$
`\leftrightarrow`	$\leftrightarrow$	`\longleftrightarrow`	$\longleftrightarrow$
`\Leftrightarrow`	$\Leftrightarrow$	`\Longleftrightarrow`	$\Longleftrightarrow$
`\mapsto`	$\mapsto$	`\longmapsto`	$\longmapsto$
`\hookleftarrow`	$\hookleftarrow$	`\hookrightarrow`	$\hookrightarrow$
`\leftharpoonup`	$\leftharpoonup$	`\rightharpoonup`	$\rightharpoonup$
`\leftharpoondown`	$\leftharpoondown$	`\rightharpoondown`	$\rightharpoondown$
`\nearrow`	$\nearrow$	`\swarrow`	$\swarrow$
`\searrow`	$\searrow$	`\nwarrow`	$\nwarrow$
`\rightleftharpoons`	$\rightleftharpoons$		

長くて太い矢印は `\implies` ($\Longrightarrow$)，`\impliedby` ($\Longleftarrow$)，`\iff` ($\Longleftrightarrow$) でも出せますが，こちらの方が左右のアキが広くなります．

2.1.9 雑記号

その他の雑記号を紹介しておきます．

\aleph	$\aleph$	\prime	$\prime$	\neg, \lnot	$\neg$
\hbar	$\hbar$	\emptyset	$\emptyset$	\flat	$\flat$
\imath	$\imath$	\nabla	∇	\natural	$\natural$
\jmath	$\jmath$	\surd	$\surd$	\sharp	$\sharp$
\ell	ℓ	\top	$\top$	\clubsuit	$\clubsuit$
\wp	$\wp$	\bot	$\bot$	\diamondsuit	$\diamondsuit$
\Re	$\Re$	\angle	$\angle$	\heartsuit	$\heartsuit$
\Im	$\Im$	\triangle	$\triangle$	\spadesuit	$\spadesuit$
\partial	∂	\forall	$\forall$		
\infty	∞	\exists	$\exists$		

^\prime は ' と短く書くこともできます．

空集合は \emptyset ($\emptyset$) です．ギリシャ文字の \phi (ϕ) を**使ってはいけません**．空集合の記号に \varnothing ($\varnothing$) を使うこともできます．あるいは由来を考えて，ノルウェー語のアルファベットである \o ($\o$) や \O ($\O$) を使いたい人もいるかもしれません．

\imath, \jmath は例えば \tilde{\imath} ($\tilde{\imath}$) のように i, j にアクセント記号を付けるために使います．

小文字のエルを手書きする際には数字の 1 などと区別するために ℓ のように書きますが，TEX では特別な理由がない限りは \ell (ℓ) ではなく l (l) で十分です．

2.1.10 総和・大きな記号

総和や積分など，大きい記号を出力する命令には次のようなものがあります．

\sum	$\sum$	\bigcap	$\bigcap$	\bigodot	$\bigodot$
\prod	$\prod$	\bigcup	$\bigcup$	\bigotimes	$\bigotimes$
\coprod	$\coprod$	\bigsqcup	$\bigsqcup$	\bigoplus	$\bigoplus$
\int	$\int$	\bigvee	$\bigvee$	\biguplus	$\biguplus$
\oint	$\oint$	\bigwedge	$\bigwedge$	\idotsint	$\idotsint$
\iint	$\iint$	\iiint	$\iiint$	\iiiint	$\iiiint$

これらの命令で範囲の指定を行うには，添字と同じく ^, _ を使います．インライン数式とディスプレイ数式とでは添字の付き方は異なります．

本来大きい記号を用いるべきところで小さい記号を使ってしまうと，出力が非常に見苦しくなります．

\cup_{\lambda \in \Lambda} U_\lambda	$\cup_{\lambda \in \Lambda} U_\lambda$
\bigcup_{\lambda \in \Lambda} U_\lambda	$\bigcup_{\lambda \in \Lambda} U_\lambda$

積分や微分形式を書くときは，関数と dx の間にスペースを入れた方が綺麗に見えるかもしれません．

```
\[ \sum_{n = 0}^\infty a_n \]
```
$$\sum_{n=0}^\infty a_n$$

```
$ \sum_{n = 0}^\infty a_n $
```
$\sum_{n=0}^\infty a_n$

```
\[ \int_0^1 f(x) dx \]
```
$$\int_0^1 f(x)dx$$

```
\[ \int_0^1 f(x) \, dx \]
```
$$\int_0^1 f(x)\,dx$$

```
$ \int_0^1 f(x) \, dx $
```
$\int_0^1 f(x)\,dx$

余談ですが，集合を書くときも，波括弧の内側にスペースを空けた方が見やすいかもしれません．内包記法の場合のみスペースを空けるという流儀もあるようです．

```
\{1, 2, 3\}
```
$\{1, 2, 3\}$

```
\{\, 1, 2, 3 \,\}
```
$\{\, 1, 2, 3 \,\}$

2.1.11 log 型関数・mod

$\log x$ と出力するつもりで `log x` と書くと，$logx$ のような見苦しい出力になってしまいます．正しくは `\log x` とします．

この類の関数には次のようなものがあります．

`\arccos`	$\arccos$	`\dim`	$\dim$	`\log`	$\log$
`\arcsin`	$\arcsin$	`\exp`	$\exp$	`\max`	$\max$
`\arctan`	$\arctan$	`\gcd`	$\gcd$	`\min`	$\min$
`\arg`	$\arg$	`\hom`	$\hom$	`\Pr`	$\Pr$
`\cos`	$\cos$	`\inf`	$\inf$	`\sec`	$\sec$
`\cosh`	$\cosh$	`\ker`	$\ker$	`\sin`	$\sin$
`\cot`	$\cot$	`\lg`	$\lg$	`\sinh`	$\sinh$
`\coth`	$\coth$	`\lim`	$\lim$	`\sup`	$\sup$
`\csc`	$\csc$	`\liminf`	$\liminf$	`\tan`	$\tan$
`\deg`	$\deg$	`\limsup`	$\limsup$	`\tanh`	$\tanh$
`\det`	$\det$	`\ln`	$\ln$		

`\varlimsup`	$\varlimsup$	`\projlim`	$\projlim$	`\varprojlim`	$\varprojlim$
`\varliminf`	$\varliminf$	`\injlim`	$\injlim$	`\varinjlim`	$\varinjlim$

これらのうち，上限・下限をとるものは `\sum` などと同様に `^`，`_` で指定します．

```
\[ \log_a x \]
```
$$\log_a x$$

```
\[ \lim_{n \to \infty} a_n \]
```
$$\lim_{n\to\infty} a_n$$

2 項演算のつもりで `a \mod b` ($a \mod b$) と書くと見苦しい出力になります．剰余を表す mod には 3 種類あり，2 項 (binary) 演算に用いる `\bmod` と，括弧付き (parenthesized) の `\pmod`，括弧なしの `\mod` です．`\pmod`，`\mod` では手前に `\;\,` 相当の空白が挿入されます．

`a \bmod p`	$a \bmod p$
`a \equiv b \pmod{p}`	$a \equiv b \pmod{p}$
`a \equiv b \mod{p}`	$a \equiv b \mod p$

2.1.12 ドット

数式中の点々は，中央に出る `\cdots` ($\cdots$) と下の方に出る `\ldots` ($\ldots$) がありますが，通常は `\dots` と書くだけで，後続の記号から種類を判断してくれることになっています．

`a_1, a_2, \dots, a_n`	$a_1, a_2, \dots, a_n$
`a_1 + a_2 + \dots + a_n`	$a_1 + a_2 + \cdots + a_n$
`a_1 a_2 \dots a_n`	$a_1 a_2 \dots a_n$
`\int \dots \int`	$\int \cdots \int$

3 つ目のものはうまくいっていません．最後のものは `\idotsint` を用いたほうが良いでしょう．後続の記号がない場合や，うまくいかない場合は，次のような命令で区別します．

(commas)	`a_1, \dotsc`	$a_1, \dotsc$
(binary)	`a_1 + \dotsb`	$a_1 + \dotsb$
(multiplications)	`a_1 \dotsm`	$a_1 \dotsm$
(integrals)	`\int \dotsi`	$\int \dotsi$

これらは標準の LaTeX で用意されている `\ldots`，`\cdots` の命令に代わるもので，前後の空白が微妙に調整されています．

2.1.13 アクセント記号・上下に付けるもの

数式モードだけで使えるアクセント記号を示します．

`\hat{a}`	$\hat{a}$	`\grave{a}`	$\grave{a}$	`\dot{a}`	$\dot{a}$
`\check{a}`	$\check{a}$	`\tilde{a}`	$\tilde{a}$	`\ddot{a}`	$\ddot{a}$
`\breve{a}`	$\breve{a}$	`\bar{a}`	$\bar{a}$	`\dddot{a}`	$\dddot{a}$
`\acute{a}`	$\acute{a}$	`\vec{a}`	$\vec{a}$	`\ddddot{a}`	$\ddddot{a}$

伸縮する記号には次のようなものがあります．

`\overline{x + y}`	$\overline{x+y}$	`\overbrace{x + y}`	$\overbrace{x+y}$
`\underline{x + y}`	$\underline{x+y}$	`\underbrace{x + y}`	$\underbrace{x+y}$
`\widehat{xyz}`	$\widehat{xyz}$	`\overrightarrow{\mathrm{AB}}`	$\overrightarrow{\mathrm{AB}}$
`\widetilde{xyz}`	$\widetilde{xyz}$	`\overleftarrow{\mathrm{AB}}`	$\overleftarrow{\mathrm{AB}}$

`\overbrace`, `\underbrace` には添字を付けることができます.

`\overbrace{1 + \dotsb + 1}^p`	$\overbrace{1+\cdots+1}^{p}$
`\underbrace{1 + \dotsb + 1}_p`	$\underbrace{1+\cdots+1}_{p}$

記号の上下に式を配置するには `\stackrel`，あるいは `\overset`, `\underset` を用います.

`X \stackrel{f}{\to} Y`	$X \stackrel{f}{\to} Y$
`X \overset{f}{\to} Y`	$X \overset{f}{\to} Y$
`X \underset{f}{\to} Y`	$X \underset{f}{\to} Y$

`\xrightarrow`, `\xleftarrow` を用いると，いくらでも伸びる矢印が出力できます.

```
\mathbb{C} \xrightarrow{x \mapsto 4x^3 - g_2x - g_3} \mathbb{C}
```

$$\mathbb{C} \xrightarrow{x \mapsto 4x^3 - g_2x - g_3} \mathbb{C}$$

根号は `\sqrt` で出すことができます.

`\sqrt{2}`	$\sqrt{2}$
`\sqrt[3]{2}`	$\sqrt[3]{2}$

根号の高さが揃わない場合には，`\strut` あるいは `\mathstrut` で支柱を入れると多少ましになります. また，`\smash[t]` で高さを，`\smash[b]` で深さを 0 にできます.

`\sqrt{g} + \sqrt{h}`	$\sqrt{g} + \sqrt{h}$
`\sqrt{g\mathstrut} + \sqrt{h\mathstrut}`	$\sqrt{g\mathstrut} + \sqrt{h\mathstrut}$
`\sqrt{\smash[b]{g\mathstrut}} + \sqrt{\smash[b]{h\mathstrut}}`	$\sqrt{\smash[b]{g\mathstrut}} + \sqrt{\smash[b]{h\mathstrut}}$

2.2 数式環境

別行立ての数式を `\[ ... \]` で出力すると，数式に番号が付きません．数式番号を付けるには，`equation` 環境を始めとする数式環境を用います.

```
1 \begin{equation}
```

```
    e^{i\theta} = \cos\theta + i\sin\theta \label{eq:euler}
\end{equation}
```

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{1}$$

ラベルを付けた場合には，`\ref{eq:euler}`(1) か `\eqref{eq:euler}`((1)) で参照できます．`\eqref` を用いた場合には自動で括弧が付きます．

数式番号が不要な場合には equation*環境を用います．(`\[ ... \]` とほぼ同じです．) equation 以外の数式環境でも，ほとんどのものは後ろに*を付けると数式番号を出力しなくなります．

```
\begin{equation*}
    e^{i\theta} = \cos\theta + i\sin\theta
\end{equation*}
```

$$e^{i\theta} = \cos\theta + i\sin\theta$$

標準的でない数式番号は `\tag` で付けます．

```
\begin{equation}
    e^{i\theta} = \cos\theta + i\sin\theta \tag{$*$}
\end{equation}
```

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{$*$}$$

複数の数式を並べるには gather 環境を使います．改行は `\\` で行います．最後の行には `\\` を付けません．(付けるとその後ろの行にも数式番号が付いてしまいます．) 数式番号を付けたくない行には `\\` の手前に `\notag` と書いておきます．

```
\begin{gather}
    (\sin x)' = \cos x\\
    (\cos x)' = -\sin x \notag\\
    (\tan x)' = \frac1{\cos^2 x}
\end{gather}
```

$$(\sin x)' = \cos x \tag{2}$$
$$(\cos x)' = -\sin x$$
$$(\tan x)' = \frac{1}{\cos^2 x} \tag{3}$$

align 環境は `&` で位置を揃えることができます．各行に番号が付きます．番号の不要な行には `\notag` と書いておきます．

```
\begin{align}
    e^x &= \sum_{n = 0}^\infty \frac{x^n}{n!} \notag\\
    &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dotsb
\end{align}
```

$$
\begin{aligned}
e^x &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \\
&= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots
\end{aligned} \tag{4}
$$

位置を揃えた複数行の数式全体の中央に番号を振るには split 環境を使います．split 環境自体は数式番号を出力しないので，equation など他の数式環境の中に入れて使います．

```
\begin{equation}
    \begin{split}
        e^x &= \sum_{n = 0}^\infty \frac{x^n}{n!}\\
        &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dotsb
    \end{split}
\end{equation}
```

$$
\begin{aligned}
e^x &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \\
&= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots
\end{aligned} \tag{5}
$$

途中に文章を割り込ませるには `\intertext` を使います．

```
\begin{align}
    \Gamma(2) &= 1,\\
    \Gamma(3) &= 2,\\
    \intertext{一般に}
    \Gamma(n + 1) &= n!
\end{align}
```

$$\Gamma(2) = 1, \tag{6}$$
$$\Gamma(3) = 2, \tag{7}$$

一般に

$$\Gamma(n+1) = n! \tag{8}$$

数式中の `\\` では改ページしません．改ページを許すには，`\\` の直前に `\displaybreak[0]` と書いておきます．この `[0]` を `[1]`，`[2]`，`[3]`，と変更すると改ページのしやすさが増し，`\displaybreak[4]` では必ず改ページします．単に `\displaybreak` と書けば `\displaybreak[4]` と同じ意味になります．

すべての `\\` に同じ改ページのしやすさを設定するには，プリアンブルに `\allowdisplaybreaks[1]` などと書いておきます．これも `[0]` から `[4]` まであり，0 では改ページせず，4 に近づくほど改ページしやすくなります．この場合，改ページしたくない改行は `\\*` で表します．

数式番号は，通常は (章. 番号) の形式になりますが，これを例えば (節. 番号) にするには

```
\numberwithin{equation}{section}
```

と書いておきます．また，(2.5a)，(2.5b) のように副番号にしたい部分は subequations 環境に入れます．

```
\begin{subequations}
    \begin{align}
        \cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta\\
        \sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta
    \end{align}
\end{subequations}
```

$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta \tag{9a}$$
$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta \tag{9b}$$

行列を出力するための命令には，次のものがあります．基本的には，表組みと同じように書けば大丈夫です．

`\begin{matrix} a & b \\ c & d \end{matrix}`	$\begin{matrix} a & b \\ c & d \end{matrix}$
`\begin{pmatrix} a & b \\ c & d \end{pmatrix}`	$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$
`\begin{bmatrix} a & b \\ c & d \end{bmatrix}`	$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
`\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}`	$\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$
`\begin{vmatrix} a & b \\ c & d \end{vmatrix}`	$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$
`\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}`	$\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$

行列の成分を並べて書くときのドットは，`\cdots`，`\vdots`，`\ddots`を使うことが多いと思います．

```
\[
    \begin{pmatrix}
        a_{11} & \cdots & a_{1n}\\
        \vdots & \ddots & \vdots\\
        a_{n1} & \cdots & a_{nn}
    \end{pmatrix}
\]
```

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

疎行列の大きいゼロや $*$ は決まった書き方はないようですが，例えば次のようにすれば出力できます．(あまり汎用性のある書き方ではありません．) 詳細は省略します．

```
\[
    \begin{pmatrix}
```

```
3         \lambda_1\\
4         & \lambda_2 & \multicolumn{2}{c}{\smash{\text{\huge 0}}}\\
5         & & \ddots\\
6         \multicolumn{2}{c}{\smash{\text{\huge 0}}} & & \lambda_n
7     \end{pmatrix}
8 \]
```

$$\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ & 0 & & \lambda_n \end{pmatrix}$$

場合分けは cases 環境を使います．これも行列の一種です．

```
1 \begin{equation}
2     \lvert x \rvert = \begin{cases}
3         x & x \geq 0\\
4         -x & \text{otherwise}
5     \end{cases}
6 \end{equation}
```

$$|x| = \begin{cases} x & x \geq 0 \\ -x & \text{otherwise} \end{cases} \tag{10}$$

2.3 マクロ (自前の命令)

LaTeX で新しい命令 (マクロ, macro) を自分で定義する場合には，`\newcommand` 命令を用います．プリアンブルに次のように書くことで，新しい命令を定義できます．

```
\newcommand{ 命令の名前 }{ 定義内容 }
```

命令の名前は `\` から始めます．例えば，`\newcommand{\R}{\mathbb{R}}` と書いておけば，`$\R$` と書くだけで $\mathbb{R}$ が出力できます．

引数を取る命令も定義でき，次のように書きます．

```
\newcommand{ 命令の名前 }[引数の個数]{ 定義内容 }
```

定義内容のところに `#1` と書くことで最初の引数を，`#2` と書くことで 2 番目の引数を利用できます．引数は最大で 9 個まで取ることができます．

例えば, プリアンブルに `\newcommand{\transpose}[1]{{#1}^\top}` と書いておけば, `\transpose{A}` と書くことで $A^\top$ が出力できます．このようにする利点は，例えば文書作成後に，転置行列は左上に t を付けるように変更したいと思ったとき，プリアンブルのマクロの定義を

```
\newcommand{\transpose}[1]{\,{\vphantom{#1}}^t\!{#1}}
```

に変更するだけで済むことです．(`\vphantom` は，中身と同じ高さの透明な支柱を作る命令です．)

`\newcommand` では，すでに定義されている命令を再定義しようとするとエラーになります．すでに存在する命令を上書きしたい場合には `\renewcommand` を用います．使い方は `\newcommand` と一緒です．例えば，`\P` は本来パラグラフ記号 (¶) が割り当てられていますが，これを

```
\renewcommand{\P}{\mathbb{P}}
```

とすれば，`\P` で $\mathbb{P}$ が出せるようになります．`\newcommand` の場合とは逆に，`\renewcommand` では，定義されていない命令を再定義しようとするとエラーになります．

Math operator を定義する場合には専用の命令があります．log 型の operator を定義するには `\DeclareMathOperator` を，lim 型の operator を定義するには `\DeclareMathOperator*` をそれぞれ用います．

```
\DeclareMathOperator{\Aut}{Aut}
\Aut_K L
```
$\operatorname{Aut}_K L$

```
\DeclareMathOperator*{\Aut}{Aut}
\Aut_K L
```
$\operatorname*{Aut}_K L$

マクロを定義するほどでもないという場合には，`\operatorname`，`\operatorname*` 命令が使えます．

```
\operatorname{Hom}_A (M, N)
```
$\operatorname{Hom}_A(M, N)$

```
\operatorname*{Hom}_A (M, N)
```
$\operatorname*{Hom}_A(M, N)$

ちなみに，lim 型の命令を強制的に log 型に変更したり，逆に log 型の命令を lim 型にするにはそれぞれ `\nolimits`，`\limits` 命令を後ろに付けます．例えば，det は標準で lim 型の命令になってしまうので，後ろに `\nolimits` を付けると良いでしょう．

```
\det_K A
```
$\det_K A$

```
\det\nolimits_K A
```
$\det\nolimits_K A$

さらに凝った命令を作りたい場合には，xparse パッケージが便利です．

3　美しい数学文章を作るために

3.1　定理環境

定理環境を利用するには，LaTeX の標準のものを用いる方法，theorem パッケージを用いる方法，amsthm パッケージを用いる方法の 3 つがあります．ここでは，amsthm パッケージを用いる方法のみを紹介することにします．

定理環境を定義するには，例えばプリアンブルに次のように書いておきます．

```
\theoremstyle{definition}
\newtheorem{definition}{定義}
\newtheorem*{definition*}{定義}
\newtheorem{theorem}{定理}
\newtheorem*{theorem*}{定理}
```

`\theoremstyle{definition}` は，定理環境中の欧文が斜体になるのを避けるために入れてあります．(ただしこの方法では定理名が細字になってしまいます．これを避ける方法は黒木 [8] などを参照してください．) theorem パッケージでは `\theorembodyfont{\upshape}` などとします．アルファベットを立体にするために，各環境中で `\upshape` や `\rm` を使うようなことはしないでください．

`*` 付きのものは定理番号の付かない環境です．

上で定義した定理環境を使うには次のようにします．proof 環境は定義しなくても使えます．

```
\begin{definition}
    $n \in \mathbb{N}$に対し，$\varphi(n) = \lvert(\mathbb{Z}/n\mathbb{Z})^\times\rvert
        $とおき，
    Euler のトーシェント関数という．
\end{definition}
\begin{theorem}[Euler's theorem]
    互いに素な$m, n \in \mathbb{N}$に対し，$m^{\varphi(n)} \equiv 1 \pmod{n}.$
\end{theorem}
\begin{proof}
    証明は読者への演習問題とする．
\end{proof}
```

定義 1. $n \in \mathbb{N}$ に対し，$\varphi(n) = |(\mathbb{Z}/n\mathbb{Z})^\times|$ とおき，Euler のトーシェント関数という．

定理 1 (Euler's theorem)**.** 互いに素な $m, n \in \mathbb{N}$ に対し，$m^{\varphi(n)} \equiv 1 \pmod{n}$.

Proof. 証明は読者への演習問題とする． □

theorem 環境の後ろの `[...]` で囲まれた定理名はオプションなので，なくても構いません．

証明が別行立ての数式で終わる場合，証明終了の箱マークが次の行に出力されてしまいます．このようなときは，`\qedhere` を用いるとうまくいきます．

```
\begin{proof}
```

```
    相加・相乗平均の関係より
    \[ \cosh x = \frac{e^x + e^{-x}}{2} \geq \sqrt{e^x e^{-x}} = 1. \qedhere \]
\end{proof}
```

Proof. 相加・相乗平均の関係より

$$\cosh x = \frac{e^x + e^{-x}}{2} \geq \sqrt{e^x e^{-x}} = 1. \qquad \square$$

proof 環境の最初の文言を変更するには，`\proofname` を `\renewcommand` で上書きします．`\textgt` で和文をゴシック体で表示できます．証明終了の記号を変更するには，`\qedsymbol` を `\renewcommand` で上書きします．`\renewcommand{\qedsymbol}{}` とすれば証明終了の記号を出力しないようにできます．

```
\renewcommand{\proofname}{\textgt{証明}}
\renewcommand{\qedsymbol}{Q.E.D.}
...
\begin{proof}
    明らか.
\end{proof}
```

証明. 明らか. Q.E.D.

`\numberwithin` のように，定理番号をセクション番号に従属させる場合には後ろにオプションを書き，他の定理環境とカウンターを共有したい場合には真ん中にオプションを書きます．

```
\renewtheorem{definition}{定義}[section]%section に従属するようにする
\renewtheorem{theorem}[definition]{定理}%definition とカウンターを共有する
```

定理環境を枠で囲みたい場合には，shadethm パッケージが便利です．

```
\newshadetheorem{remark}{注意}
...
\begin{remark}
    可算な位相空間であっても,必ずしも第二可算公理をみたすとは限らない.
\end{remark}
```

注意 1. 可算な位相空間であっても，必ずしも第二可算公理をみたすとは限らない．

詳細はマニュアル等を参照してください．

3.2 数式クラス (math class) について

TEX における数学記号は，その記号が属する**数式クラス** (math class) によって分類されます．数式クラスには次のようなものがあります．

表 4: 数式クラスの種類

種類	意味	例
`mathord`	通常 (ordinary) の記号	$A\ 0\ \infty\ \emptyset$
`mathop`	作用素 (operator)	$\sum\ \prod\ \int$
`mathbin`	2 項演算 (binary)	$+\ \cup\ \wedge$
`mathrel`	2 項関係 (relation)	$=\ \leq\ \in$
`mathopen`	開き (open) 括弧	$(\ \{\ \langle\ \lfloor$
`mathclose`	閉じ (close) 括弧	$)\ \}\ \rangle\ \rfloor$
`mathpunct`	句読点 (punctuation)	$,\ .\ ;\ !$

TEX で文章をコンパイルするとき，数式クラスに応じて数学記号どうしの適切な間隔が決定されます．逆に言えば，数式クラスを適切に設定しておかないと，見苦しい組版になってしまう可能性があります．

例えば，± などの `mathbin` に属する記号は，左側にオペランドがなければ自動的に単項演算とみなされ，スペースを空けなくなります．

数式クラスと同じ名前の命令を用いることで，数式クラスを変更できます．例えば，`x \mathbin{M} y` と書けば M が 2 項演算子として扱われます: $x\,M\,y$.

実際に例をいくつか見てみましょう．

- `:` は `mathrel`(比などを表すのに用いる) で，`\colon` は `mathpunct` です．例えば写像を書くときは `\colon` を使います．

```
f : X \to Y
```
$f : X \to Y$

```
f \colon X \to Y
```
$f \colon X \to Y$

- 区切りや整除関係を表す `\mid` (|) は `mathrel` で，`|` は `mathord` です．

```
x | y
```
$x|y$

```
x \mid y
```
$x \mid y$

- 商を表す `/` は `mathbin` ではなく `mathord` です．したがって右剰余類などは差集合を表す `\setminus` (`mathbin`) ではなく `\backslash` (`mathord`) を用いたほうが良いでしょう．ちなみに，`\smallsetminus` (∖) で高さの低い差集合の記号を出せます．

```
H \setminus G / K
```
$H \setminus G / K$

```
H \backslash G / K
```
$H \backslash G / K$

- 商集合を書く際，`/` と `\sim` (∼) を並べると，`\sim` が `mathrel` であるためにスペースが空きすぎてしまいます．これを回避するには，`\mathord{\sim}` とするか，あるいは単に `{\sim}` とすることもできます．

```
X / \sim
X / \mathord{\sim}
X / {\sim}
```
$X/\sim$
$X/\sim$
$X/\sim$

- `\triangle` ($A\triangle B$) は `mathord` で，`\bigtriangleup` ($A \triangle B$) は `mathbin` です．したがって，`\triangle` は Laplacian などに (`\Delta` を使うこともあります)，`\bigtriangleup` は対称差などに用いた方が良いでしょう．
- 直交などを表す `\perp` ($\perp$) は `mathrel` で，矛盾などを表す `\bot` ($\bot$) は `mathord` です．
- 大きい直和 `\coprod` ($\coprod$) は `mathop`，2 項直和 `\amalg` ($\amalg$) は `mathbin` です．

詳細は Downes [7] などを参照してください．

3.3 集合の内包記法 $\{x \mid P(x)\}$ の縦棒 |

基本的に `\mid` を使います．`|` は `mathrel` ではないため，使用してしまうと間が詰まって表示が見苦しくなります．

```
\{\, x \mid P(x) \,\}
\{\, x | P(x) \,\}
```
$\{\, x \mid P(x) \,\}$
$\{\, x|P(x) \,\}$

ただし，波括弧の中身が大きい場合に `\mid` は伸びてくれません．この場合，`\middle|` を使うと括弧の大きさに応じて伸びますが，そのままでは `mathrel` にはならず，`\mathrel` で囲むこともできないため (囲むと `\middle` が外側の { } を見つけられなくなってしまう)，左右に `\mathrel{}` を “ゴースト” として配置します [4]．

```
... \mid ...
... \middle| ...
... \mathrel{}\middle|\mathrel{} ...
```
$\left\{ \frac{p}{q} \mid p, q \in \mathbb{Z} \right\}$
$\left\{ \frac{p}{q} \middle| p, q \in \mathbb{Z} \right\}$
$\left\{ \frac{p}{q} \middle| p, q \in \mathbb{Z} \right\}$

プリアンブルで `\newcommand{\setmid}{\mathrel{}\middle|\mathrel{}}` などとマクロを作成しておくと便利です．

他にも，braket パッケージの `\Set` を利用するという方法もあります．

3.4 写像の表記

インラインで写像を表記するには，$f\colon X \to Y; x \mapsto y$，$f\colon X \ni x \mapsto y \in Y$ など様々な方法があります．ここでは，別行立てで写像を表記する方法の例として，`array` 環境を使う方法を紹介します．

```
1 \[
2     \begin{array}{rccc}
3         \varphi\colon&\mathbb{Z}[X]&\longrightarrow&\mathbb{C}\\
```

```
        &f&\longmapsto&f(\sqrt{-1})
    \end{array}
\]
```

$$\begin{array}{rccc} \varphi\colon & \mathbb{Z}[X] & \longrightarrow & \mathbb{C} \\ & f & \longmapsto & f(\sqrt{-1}) \end{array}$$

array 環境は，数式環境内で tabular 環境と同じように使うことができ，列指定に `@{foo}` と書くことで列の間の区切りを foo にできます．ここでは，間隔を狭めるために使っています．

```
\[
    \begin{array}{r@{\,}c@{\,}c@{\,}c@{\,}c}
        \varphi\colon&\mathbb{Z}[X]&\longrightarrow&\mathbb{C}\\
        &f&\longmapsto&f(\sqrt{-1})
    \end{array}
\]
```

$$\begin{array}{r@{\,}c@{\,}c@{\,}c} \varphi\colon & \mathbb{Z}[X] & \longrightarrow & \mathbb{C} \\ & f & \longmapsto & f(\sqrt{-1}) \end{array}$$

縦向きの $\in$ を出すには，graphicx パッケージの `\rotatebox` を使います．

```
\[
    \begin{array}{r@{\,}c@{\,}c@{\,}c@{\,}c}
        \varphi\colon&\mathbb{Z}[X]&\longrightarrow&\mathbb{C}\\
        &\rotatebox{90}{$\in$}&&\rotatebox{90}{$\in$}\\
        &f&\longmapsto&f(\sqrt{-1})
    \end{array}
\]
```

$$\begin{array}{r@{\,}c@{\,}c@{\,}c} \varphi\colon & \mathbb{Z}[X] & \longrightarrow & \mathbb{C} \\ & \cup & & \cup \\ & f & \longmapsto & f(\sqrt{-1}) \end{array}$$

他にも，alignat 環境を使ったり，tabular に入れるなどの方法がありますが，ここでは割愛します．

3.5 整除関係記号 $x \mid y$, $x \nmid y$

3.2 節において，整除関係記号には `\mid` を用い，`|` の使用は避けるべきだと述べました．では，その否定 $x \nmid y$ についてはどうでしょうか？

整除関係の否定には標準で `\nmid` ($\nmid$) という命令が用意されています．通常はこれで十分でしょう．しかしよく見ると，これは斜線が中心よりわずかに上にあったり，斜線が短いという欠点があることがわかります．そこで，ここでは整除関係の否定を綺麗に出すための方法を模索していきます．

まず，2 項関係の否定を作る `\not` という命令があります．これは `\not\equiv` ($\not\equiv$) などではうまく働くものの，残念ながら `\mid` に対しては正常に動きません．

また，cancel パッケージの `\cancel` を使うと，数式に斜線を引くことができます．(これは，約分であったり，計算結果が 0 になる場合の打ち消し線のためのものです: $\frac{\cancel{x}}{\cancel{2}}$.) しかし，`\cancel` で囲むと，内部的には `\hbox` で囲まれてしまうためにスペーシングがおかしくなります．そのため，`\mathrel` で適切に数式ク

ラスを再設定する必要があります．`\cancel` で斜線を引くと，`\mid` の縦棒よりも斜線の方が長くなります．また，斜線の傾きが大きいため，あまり見映えは良いとは言えません．

より簡単な方法として，centernot パッケージの `\centernot` 命令を利用するという方法があります．これは，`\centernot\mid` とするだけで縦棒の中央に正確に斜線が引かれます．この場合，斜線は縦棒とほぼ同じ長さになります．

最後に，mathabx パッケージの `\divides`，`\notdivides` 命令を利用する方法を紹介します．この方法を実行する最も簡単な方法は，プリアンブルに `\usepackage{mathabx}` と書いておくことです．ところが，この方法では，他の数式フォントも置き換えられてしまいます．また，mathabx パッケージは amsmath に依存しているため，usepackage の順序を入れ換えるといった対応もできません．

表 5: Computer Modern と mathabx フォントの比較

Computer Modern	∞	$\cap$	$\subset$	$\not\subset$	$\leq$	$\not\leq$	$\sum$	$\int$
mathabx	∞	$\cap$	$\subset$	$\subsetneq$	$\leqslant$	$\nleqslant$	$\sum$	$\int$

そこで，「mathabx フォントの一部の文字だけ抜き出して使用する」という解決策をとります．詳細は省略しますが，プリアンブルに以下のように書くと `\divides`，`\notdivides` 命令が使用可能になります [1] [2].

```
\DeclareFontFamily{U}{matha}{\hyphenchar\font45}
\DeclareFontShape{U}{matha}{m}{n}{
    <5> <6> <7> <8> <9> <10> gen * matha
    <10.95> matha10 <12> <14.4> <17.28> <20.74> <24.88> matha12
}{}
\DeclareSymbolFont{matha}{U}{matha}{m}{n}
\DeclareMathSymbol{\divides}{3}{matha}{"17}
\DeclareMathSymbol{\notdivides}{3}{matha}{"1F}
```

これが最も綺麗に出力できるのではないかと思います．

`x	y`	$x \mid y$
`x \mid y`	$x \mid y$
`x \nmid y`	$x \nmid y$
`x \not\mid y`	$x \not\mid y$
`x \cancel{\mid} y`	$x \cancel{\mid} y$
`x \mathrel{\cancel{\mid}} y`	$x \mathrel{\cancel{\mid}} y$
`x \centernot\mid y`	$x \centernot\mid y$
`x \divides y`	$x \divides y$
`x \notdivides y`	$x \notdivides y$

3.6 TikZ

ここでは，TikZ パッケージによる簡単な作図例をいくつか列挙していきます．詳しい内容については，マニュアル [6] [14] や入門書 [12] を参照してください．

```
\begin{figure}[ht]
    \centering
    \begin{tikzpicture}[domain=0:4]%グラフの描画領域は 0<=x<=4
        \draw[very thin, color = gray] (-0.1, -1.1) grid (3.9, 3.9);
        \draw[->] (-0.2, 0) -- (4.2, 0) node[right] {$x$};%横軸 (x 軸)
        \draw[->] (0, -1.2) -- (0, 4.2) node[above] {$f(x)$};%縦軸 (y 軸)
        \draw[color = red] plot (\x, \x) node[right] {$f(x) = x$};
        %\x r で度をラジアンに変換
        \draw[color = blue] plot (\x, {sin(\x r)}) node[right] {$f(x) = \sin x$};
        \fill[color = orange] (0, 0) -- plot (\x, {0.05 * exp(\x)}) node[right]
            {$f(x) = \frac{1}{20} e^x$} -- (4, 0);%塗り潰し
    \end{tikzpicture}
    \caption{関数のグラフの描画}\label{fig:tikz-graph}
\end{figure}
```

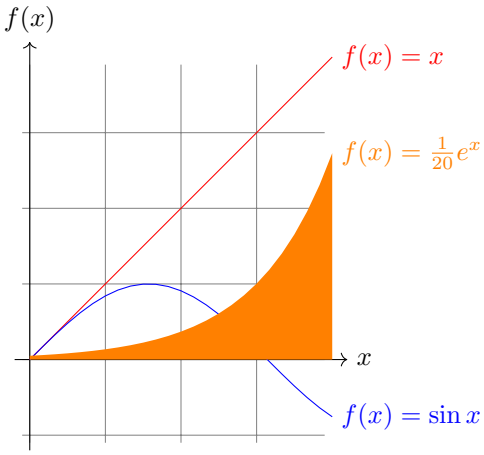

図 4: 関数のグラフの描画

```
\usetikzlibrary{patterns}
...
\begin{figure}[ht]
    \centering
    \begin{tikzpicture}[domain = -0.3 : 2.3]%グラフの定義域の設定
        \begin{scope}
            %斜線で埋める
            \fill[pattern = north west lines] (0, 0) -- (2, 0) -- (2, 2) -- cycle;
            \draw[->] (-0.5, 0) -- (2.5, 0) node[right] {$x$};%x 軸
```

```
            \draw[->] (0, -0.5) -- (0, 2.5) node[above] {$y$};%y 軸
            \draw plot (\x, \x) node[right] {$y = x$};%y = x のグラフ
            \draw (2, 0) -- (2, 2);%縦の線
            \draw (2, 0) node[below] {$1$};
            \draw (0, 0) node[below right] {$\mathrm{O}$};
        \end{scope}
        \begin{scope}[xshift = 4cm]%右にずらす
            %網掛け
            \fill[pattern = crosshatch] (0, 0) -- (2.3, 0) -- (2.3, 0.5) [domain= 2.3 :
                1] {plot (\x, 1 / \x)} -- (0, 0);
            \draw[->] (-0.5, 0) -- (2.5, 0) node[right] {$x$};
            \draw[->] (0, -0.5) -- (0, 2.5) node[above] {$y$};
            \draw plot (\x, \x) node[right] {$y = x$};
            \draw [domain = 0.4 : 2.3] plot (\x, 1/\x) node[above] {$y = \frac1x$};
            \draw (0, 0) node[below right] {$\mathrm{O}$};
        \end{scope}
    \end{tikzpicture}
    \caption{積分領域の描画}\label{fig:domain}
\end{figure}
```

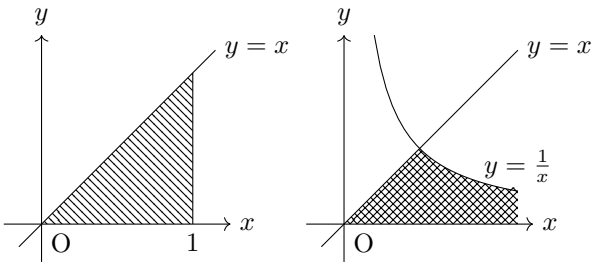

図 5: 積分領域の描画

```
\begin{figure}[ht]
    \centering
    \begin{tikzpicture}
        \begin{scope}
        \clip (0, 0) circle (1);%円の内部のみ描画する
        \fill[color = gray] (1, 0) circle (1);%塗り潰し
        \end{scope}
        \draw (0, 0) circle (1) (0, 1) node[above] {$A$};%円の描画
        \draw (1, 0) circle (1) (1, 1) node[above] {$B$};%円の描画
    \end{tikzpicture}
    \caption{ベン図の描画}\label{fig:tikz-venn}
\end{figure}
```

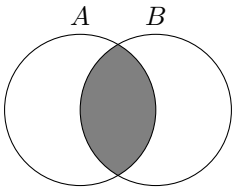

図 6: ベン図の描画

```
\usetikzlibrary{automata}
...
\begin{figure}[ht]
    \centering
    \begin{tikzpicture}[node distance = 2cm]
        %ノードの配置
        \node[state, initial] (q_1) {$q_1$};
        \node[state, accepting] (q_2) [right of = q_1] {$q_2$};
        \node[state] (q_3) [right of = q_2] {$q_3$};
        %パスの描画
        \path[->] (q_1) edge [loop above] node {0} (q_1)
            (q_1) edge node [above] {1} (q_2)
            (q_2) edge [loop above] node {1} (q_2)
            (q_2) edge [bend left] node [above] {0} (q_3)
            (q_3) edge [bend left] node [below] {0, 1} (q_2);
    \end{tikzpicture}
    \caption{オートマトンの描画}\label{fig:tikz-automata}
\end{figure}
```

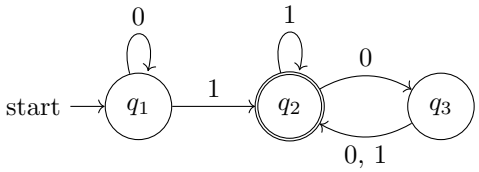

図 7: オートマトンの描画

可換図式を描くには，tikz-cd パッケージを用いると便利です．表組みと同じように使えます．

```
\begin{figure}[ht]
    \centering
    \begin{tikzcd}
        & W \arrow{ld}[swap]{f} \arrow[dashrightarrow]{d}{h} \arrow{rd}{g}\\
        X & X \times Y \arrow{l}{p} \arrow{r}[swap]{q} & Y
    \end{tikzcd}
    \caption{可換図式の描画}\label{fig:tikz-diagram}
\end{figure}
```

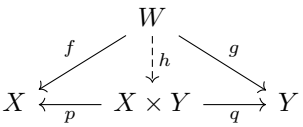

図 8: 可換図式の描画

3.7 その他 Tips

定義式などで `:=` と書くと，中心がわずかにずれます．mathtools パッケージの `\coloneqq` か，colonequals パッケージの `\colonequals` を使うと中心が揃います．

```
A := B
```
$$A := B$$
```
A \coloneqq B
```
$$A \coloneqq B$$
```
A \colonequals B
```
$$A \colonequals B$$

`\iff` などについては用意されていませんが，mathtools の `\vcentcolon` で垂直方向のちょうど中心にコロンを置くことができます．

```
A :\Longleftrightarrow B
```
$$A :\Longleftrightarrow B$$
```
A \vcentcolon\Longleftrightarrow B
```
$$A \vcentcolon\Longleftrightarrow B$$

`\lim` や `\sup` を重ねると，下付きの部分がずれることがありますが，mathtools パッケージの `\adjustlimits` を用いると高さが揃います．

```
\lim_{\delta \downarrow 0} \sup_{\lvert x - a\rvert < \delta} f(x)
```
```
\adjustlimits\lim_{\delta \downarrow 0} \sup_{\lvert x - a\rvert < \delta} f(x)
```
$$\lim_{\delta \downarrow 0} \sup_{\lvert x - a\rvert < \delta} f(x) \qquad \adjustlimits\lim_{\delta \downarrow 0} \sup_{\lvert x - a\rvert < \delta} f(x)$$

複数行にわたる添字には `\substack` コマンドを使います．

```
\sum_{\substack{1 \leq i \leq n \\ \alpha_i \neq 0}} \alpha_i \mu(A_i)
```
$$\sum_{\substack{1 \leq i \leq n \\ \alpha_i \neq 0}} \alpha_i \mu(A_i)$$

$\sum$ などの大きい記号の肩に添字を付けたいときは，`\sideset` 命令を使います．

```
\sideset{}{'}\sum_{i = 1}^n a_i
```
$$\sideset{}{'}\sum_{i = 1}^n a_i$$
```
\sideset{_1^2}{_3^4}\prod_{i = 1}^n a_i
```
$$\sideset{_1^2}{_3^4}\prod_{i = 1}^n a_i$$

4 TEX バッドノウハウ集

インターネットで TEX について検索をすると様々な情報が得られます．しかし，その中には古くなってしまっている情報も多く，現在では非推奨となっていたり，あるいはそもそも正しくない情報であったりします．そこで，ここでは TEX を使う際に推奨されない，あるいはやってはいけない書き方について述べます．

ちなみにプリアンブルに次のように書いておくと，古いパッケージや命令が含まれていた場合に警告が出るようになるので便利です．

```
1 \RequirePackage[l2tabu, orthodox]{nag}%古いパッケージや命令の利用を警告する
2 \usepackage[all, warning]{onlyamsmath}%amsmath が提供しない数式環境を使用した場合に警告する
```

以下，古い・悪い書き方の改善方法を示していきます．

- `\def` は使わない

  `\def` はマクロを定義するための命令ですが，既に定義されている場合でも問答無用で上書きしてしまいます．LATEX の `\newcommand` を使った方が安全です．
- subfigure，subfig パッケージは使わない

  複数の図を並べて表示するには，subcaption パッケージを使います．subfigure，subfig は古いパッケージなので使用は推奨されません．
- `\rm` は使わない

  同様に `\bf`，`\it`，`\tt` なども使わないようにします．これらの命令には，例えば「斜体の太字」が出力できなかったり，斜体にしたとき左右の適切な間隔が確保されないなどの欠点があります．文字をローマン体にするには `\textrm` を使います．
- 数式中に文章を `\mbox` で挿入しない

  数式中に文章を挿入するには `\text` を使います．`\mbox` は添字などにしたとき，大きさが変わってくれません: $V_{\mathrm{out}} \neq V_{\mathrm{out}}$.
- 内積などの括弧に不等号は使わない

  当たり前のことですが，`<x, y>` $(< x, y >)$ ではなく `\langle x, y \rangle` $(\langle x, y \rangle)$ と書きます．
- `$$ ... $$` や displaymath，eqnarray などの数式環境は使わない

  これらは古い環境で，空白が適切に入らなかったりすることがあります．
- 角度に `$^\circ$` は使わない

  「度」を出力するには，gensymb パッケージの `\degree` 命令などを使うほうが適切でしょう．

5　プリアンブルの例

TEX で数学文書を作る際のプリアンブルの例を示します．これはほんの一例なので，自分の好みに合ったプリアンブルを模索していくようにしましょう．

```
\documentclass[dvipdfmx]{jsarticle}%graphicx,color などのためにドライバ指定しておく

\RequirePackage[l2tabu, orthodox]{nag}%古い書き方をしたら警告を出す
\usepackage[all, warning]{onlyamsmath}%amsmath 以外の数式環境を使ったら警告を出す
\AtBeginDocument{\catcode`\$=3}%TikZ とのコンフリクトを避ける

\usepackage{amsmath,amssymb}%数式全般
\usepackage{amsthm}%定理環境
\usepackage{mathrsfs}%花文字 (\mathscr)
\usepackage{bm}%ベクトルなどの太字 (\bm)
\usepackage{mathtools}%\coloneqq など
\usepackage{colonequals}%\colonequals
\usepackage{graphicx}%図の挿入
\usepackage{subcaption}%関連する図を並べる
\usepackage{tikz}%図を描く
\usetikzlibrary{patterns}%網掛け等
\usepackage{tikz-cd}%可換図式を描く
\usepackage{cancel}%数式に斜線を引く
\usepackage{centernot}%\centernot
\usepackage{xparse}%凝ったマクロを作る (\NewDocumentCommand)
\usepackage{array,booktabs,float}%表組みなど
\usepackage{okumacro}%ルビを振る命令など (\ruby)
\usepackage{gensymb}%\degree など
\usepackage{bbm}%\mathbbm で小文字や数字も太くなるようにする

%\divides, \notdivides を使えるようにする
\DeclareFontFamily{U}{matha}{\hyphenchar\font45}
\DeclareFontShape{U}{matha}{m}{n}{
    <5> <6> <7> <8> <9> <10> gen * matha
    <10.95> matha10 <12> <14.4> <17.28> <20.74> <24.88> matha12
}{}
\DeclareSymbolFont{matha}{U}{matha}{m}{n}
\DeclareMathSymbol{\divides}{3}{matha}{"17}
\DeclareMathSymbol{\notdivides}{3}{matha}{"1F}

%amsthm における定理環境の設定
\theoremstyle{definition}%定理環境中のアルファベットを立体で表示
\newtheorem{definition}{定義}[section]%定理番号を 節.番号 にする
\newtheorem*{definition*}{定義}
\newtheorem{theorem}[definition]{定理}
```

```
\newtheorem*{theorem*}{定理}
\newtheorem{proposition}[definition]{命題}
\newtheorem*{proposition*}{命題}
\newtheorem{lemma}[definition]{補題}
\newtheorem*{lemma*}{補題}
\newtheorem{remark}[definition]{注意}
\newtheorem*{remark*}{注意}
\newtheorem{example}[definition]{例}
\newtheorem*{example*}{例}

%よく使う記号の定義
\newcommand{\N}{\mathbb{N}}%自然数
\newcommand{\Z}{\mathbb{Z}}%整数
\newcommand{\Q}{\mathbb{Q}}%有理数
\newcommand{\R}{\mathbb{R}}%実数
\newcommand{\C}{\mathbb{C}}%複素数

\newcommand{\abs}[1]{\left\lvert#1\right\rvert}%絶対値
\newcommand{\norm}[1]{\left\lVert#1\right\rVert}%ノルム
```

付録 A　TEX 統合環境

本稿の最初の方で，TEX は文書作成の際に好きなエディタを使える利点があると述べました．テキストファイルを編集するためのソフトウェアは多々ありますが，中には TEX で文書を作成することに特化したものも存在します．

その中でもいわゆる “TEX 統合環境” に分類されるソフトウェアを用いると，TEX ソースの入力支援や，組版処理の自動化など様々な恩恵を受けることができます．

TEX 統合環境は色々なものが開発されており，TeXworks，Texmaker，TeXShop，etc. と，挙げればきりがありませんが，ここでは OS が Windows，Mac OS，Linux のどれであっても使用可能な TeXstudio を紹介することにします．

TeXstudio は次のページからダウンロードすることができます．

`http://www.texstudio.org/#download`

Linux では，たいてい `texstudio` という名前のパッケージが用意されているはずなので，そちらを利用することをお勧めします．Mac OS では，homebrew-cask からもインストールすることができます．

インストールして起動できたら，pLATEX 2_ε を使うための設定をします．次のようにして設定画面を開きます．

- Windows・Linux では，オプション →Texstudio の設定
- Mac OS では，TeXstudio→ 環境設定

設定画面を開いたら，

- 「コマンド」ペインを開き，
  - 「LaTeX」の `latex -src ...` となっている部分を `platex -synctex=1 ...` に変更 (Windows でもし文字化けする場合には，`platex --kanji=utf8 -synctex=1 ...` とする必要があるかもしれません．)
  - 「DviPdf」の `dvipdf(m)` を `dvipdfmx` に変更
  - 「BibTeX」の `bibtex` を `pbibtex` に変更
- 「ビルド」ペインを開き，
  - 「ビルド&表示」の「コンパイル&表示」を「`DVI->PDF` チェーン」に変更
  - 「既定のコンパイラ」の「`PdfLaTeX`」を「`LaTeX`」に変更

とします．PDF ビューワを「組み込み PDF ビューワ (埋め込み)」から「外部 PDF ビューワ」に変更した方が見やすいかもしれません．

これで TeXstudio で日本語の数学文書が作成できるようになっているはずです．

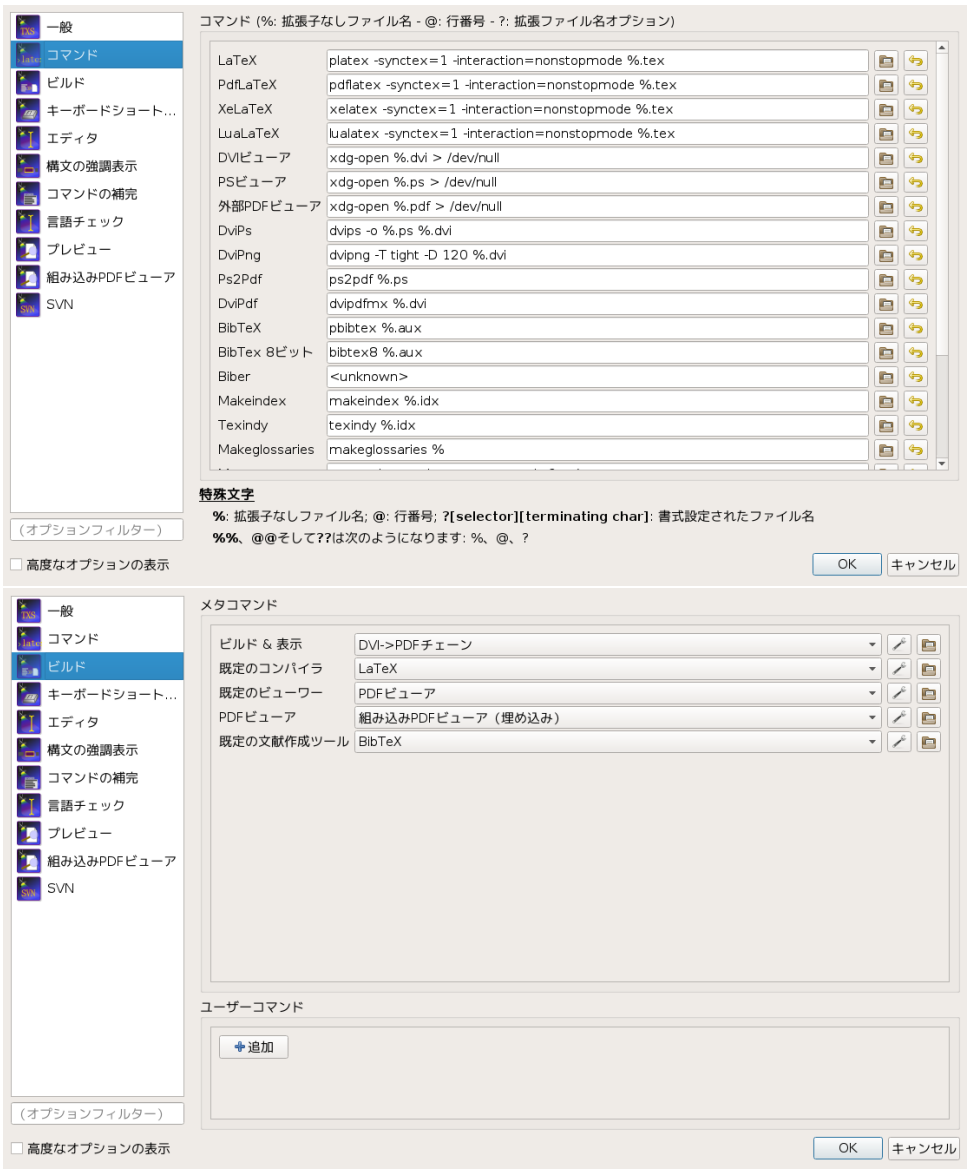

図 9: Linux における TeXstudio の設定例

付録 B　文字コードについて

コンピュータ上で文字を表現するには，文字をいったん整数値として “符号化” する必要があります．このような文字の符号化方式 (いわゆる “文字コード”) には様々なものがあり，日本でよく使われるものには Shift_JIS, EUC-JP, ISO-2022-JP, UTF-8, UTF-16 などがあります．最近では多くの文書が UTF-8 で書かれているので，迷ったときはとりあえず UTF-8 を使っておけばまず間違いはないでしょう．実際，TeXstudio のデフォルトの文字コードは UTF-8 です．

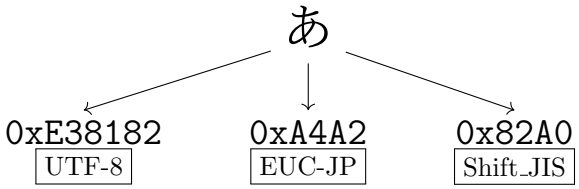

図 10: 文字の符号化

古い Windows 等で，バックスラッシュを打ったつもりが，画面上の表示が円記号 (¥) になってしまう場合があります．これは，同じ符号位置 `0x5C` に，ASCII ではバックスラッシュを，JIS X 0201 では円記号を割り当てているという違いから来ています．UTF-8 や EUC-JP は ASCII を拡張した文字コードであり，Shift_JIS は JIS X 0201 を拡張した文字コードです．Windows では長らく Shift_JIS が標準で使われてきました．そのため，少し前の Windows ではバックスラッシュが円記号として表示されることがあります．(最近の Windows ではファイル名なども UTF-8 化されているようです．) 結局，文字の見た目が変化するだけで，符号位置は `0x5C` 変わらないので本質的な違いはありません．但し，文字コードを変換する場合には問題が発生する可能性はあるので注意してください．

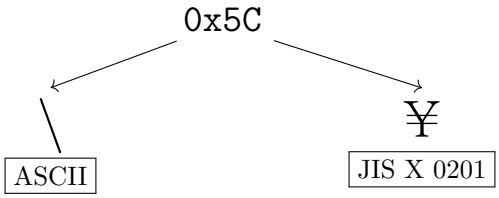

図 11: 円記号問題

文字コードや円記号問題の詳細については矢野 [15] などを参照してください．

おわりに

本稿の前半の大部分は奥村 [10] によっています．残りの部分はネット上の情報や今までの経験を基に書いているので，思い込みで誤ったことを書いてある部分もあるかもしれません．そのような誤謬を発見された場合には，私の Twitter アカウント (`@waidotto`) まで連絡を頂ければ幸いです．

TEX に限らない，数学の文章の書き方については，結城 [16] [17] が大いに役立ちます．最低限の常識的事柄については，杉ノ内 [13] を見てください．また，多少内容が古いですが，小田 [9] も参考になります．

参考文献

[1] Avoiding mathabx symbol clashes when using LaTeX. `https://alephnull.uk/content/mathabx-font-symbol-redefinition-clash-latex`.

[2] Importing a Single Symbol From a Different Font. `http://tex.stackexchange.com/questions/14386/importing-a-single-symbol-from-a-different-font`.

[3] mathabx - TEX Wiki. `https://texwiki.texjp.org/?mathabx`.

[4] \middle であり\mid であるもの. http://d.hatena.ne.jp/zrbabbler/20120411/1334151482.
[5] What is the correct pronunciation of TeX and LaTeX? http://tex.stackexchange.com/questions/17502/what-is-the-correct-pronunciation-of-tex-and-latex.
[6] Jacques Crémer. A very minimal introduction to Ti*k*Z. https://www.ctan.org/pkg/pgf, 2011.
[7] Michael Downes. Short Math Guide for LaTeX. http://www.ams.org/tex/short-math-guide.html, 2002.
[8] 黒木玄. 日本語 LaTeX を使うときに注意するべきこと. http://www.math.tohoku.ac.jp/~kuroki/LaTeX/howtolatex.html, 2016.
[9] 小田忠雄. 数学の常識・非常識—由緒正しい TeX 入力法. 数学通信, Vol. 4, No. 1, pp. 95–112, 1995. http://mathsoc.jp/publication/tushin/index-4-1.html.
[10] 奥村晴彦, 黒木裕介. LaTeX 2$_\varepsilon$ 美文書作成入門. 技術評論社, 第 6 版, 2013.
[11] Scott Pakin. The Comprehensive LaTeX Symbol List. http://www.ctan.org/tex-archive/info/symbols/comprehensive/, 2015.
[12] 杉ノ内萌. TikZ 入門. http://mone.at-ninja.jp/files.html, 2015.
[13] 杉ノ内萌. いつも注意すること. http://mone.at-ninja.jp/files.html, 2016.
[14] Till Tantau. The Ti*k*Z and PGF Packages. https://www.ctan.org/pkg/pgf, 2015.
[15] 矢野啓介. プログラマのための文字コード技術入門. 技術評論社, 2010.
[16] 結城浩. 数学文章作法 基礎編. 筑摩書房, 2013.
[17] 結城浩. 数学文章作法 推敲編. 筑摩書房, 2014.