

# TeX 実習

2019 年 5 月 9 日

本文章は、TeX 実習の実際の中身である。行うためには実習用マシンに TeX システムをインストールする必要がある。計算数学実習資料集の「[TeX 実習](#)」のページ内にある「TeX Live のインストール」を参照すること。

TeX Live をインストールした後は、各自の知識に沿って実習を進めてほしい、

- 今まで TeX を全く触ったことがない人は §1 へ。
- 「TeX・LaTeX の典型的な間違い」については §2 へ。
- §1 や §2 を読んだら、それについての演習が §3 にあるので取り組んで欲しい。
- 各種パッケージを有効活用して相互参照、画像挿入、可換図式を組む方法については §5 へ。
- 伝統的な pLaTeX は知っているが、pLaTeX が利用できない場合への対処法や、最近の TeX の発展について知りたい人は §4 か、実習資料集にある「pTeX 系列以外による日本語文書作成」（やや古い）を参照すること。
- バリバリ使いこなしていて、以上の範囲などとくに知っているという人は TA を呼ぶこと。

以下、[about:blank](#) のような部分は外部リンクで、緑字の部分<sup>\*1</sup>は文書内リンクである。

TeX ソース等の表記では、以下の字体 (**Inconsolata**) を使っている：

数字の 0	大文字のオー	数字の 1	小文字のエ	大文字のアイ
0	O	1	l	I

Last update: 2019/05/09 13:05:38.

<sup>\*1</sup> Web ブラウザ上では動作しないが、SumatraPDF などの PDF リーダーであれば動作する。

# 目次

<b>1</b>	<b>pL<sup>A</sup>T<sub>E</sub>X はじめの一步</b>	<b>3</b>
1.1	最初の例 . . . . .	3
1.2	タイプセット . . . . .	4
1.3	エラーが起きたら . . . . .	5
1.4	日本語フォントの埋め込み . . . . .	6
1.5	この先は . . . . .	7
1.6	日々の作業の小技 . . . . .	8
<b>2</b>	<b>L<sup>A</sup>T<sub>E</sub>X の作法</b>	<b>10</b>
2.1	数式 . . . . .	11
2.2	地の文 . . . . .	15
2.3	文章の構造 . . . . .	16
<b>3</b>	<b>§1 と §2 に関連した課題</b>	<b>20</b>
3.1	T <sub>E</sub> X ソースの例 . . . . .	20
3.2	数学の文章を T <sub>E</sub> X で書く練習 . . . . .	22
3.3	T <sub>E</sub> X ソースの修正練習 . . . . .	23
<b>4</b>	<b>pT<sub>E</sub>X 系列以外の T<sub>E</sub>X エンジンについて</b>	<b>24</b>
4.1	T <sub>E</sub> X エンジン . . . . .	24
4.2	マクロパッケージ . . . . .	25
4.3	文書クラス . . . . .	25
4.4	文字コード . . . . .	26
4.5	この先は . . . . .	26
<b>5</b>	<b>雑多な話題</b>	<b>27</b>
5.1	ソースコードの組版 . . . . .	27
5.2	相互参照 . . . . .	28
5.3	画像 . . . . .	31
5.4	図式 . . . . .	32
5.5	数式クラス . . . . .	34
5.6	pT <sub>E</sub> X 系列で縦書き等をする場合に発生する問題とその対処 . . . . .	39
5.7	難しい小ネタ . . . . .	40

# 1 p $\text{\LaTeX}$ はじめの一歩

日本人が「 $\text{\TeX}$  を用いて文書を書く」という場合、実際に使っているのは **p $\text{\LaTeX}$**  であることが多い。これらは、

- Knuth 氏による、もともとの（プログラムとしての） $\text{\TeX}$
- Leslie Lamport 氏による、 **$\text{\LaTeX}$**  という  $\text{\TeX}$  のマクロパッケージ

を、アスキー<sup>\*2</sup>がそれぞれ日本語化したものである。細かく言うと、プログラム側は p $\text{\TeX}$  (publishing  $\text{\TeX}$ ) と呼ばれ、 $\text{\LaTeX}$  を日本語対応させたマクロパッケージが p $\text{\LaTeX}$  である。日本語では横書きの他に縦書きも行われるが、p $\text{\TeX}$ ・p $\text{\LaTeX}$  は縦組みの組版もサポートしている。

この実習では、p $\text{\TeX}$ /p $\text{\LaTeX}$  を Unicode 対応させた up $\text{\TeX}$ /up $\text{\LaTeX}$  を使用する。p $\text{\TeX}$  系以外の  $\text{\TeX}$  エンジンについては、後の方 (§4) で軽く触れる。

## 1.1 最初の例

まず適当なエディタを使い、次の内容を持った first.tex を作る。このような  $\text{\TeX}$  が理解できるようなテキストファイルを  **$\text{\TeX}$  ソース** という。

```
1 \documentclass[uplatex]{jsarticle}
2 \begin{document}
3 ようこそ、 $\text{\TeX}$  の世界へ！
4 \[
5 \int_0^{\infty} e^{-x^2} \, dx = \frac{\sqrt{\pi}}{2}.
6 \]
7 ちょっとチェックしちゃった。
8 \end{document}
```

- 大文字と小文字、全角文字と半角文字は厳密に区別されるので、間違えないように注意。特に、半角バックslash \ から始まる文字列は  $\text{\TeX}$  では何らかの命令として扱われる。
- Windows で作業をしている関係で、実際に編集を行うテキストエディタでは、半角バックslash は半角円記号 ¥ のように表示されることになるだろう。
- 保存先はどこでもいいが、以降の説明では「ドキュメント」のところに保存したとする。
- ファイルの文字コードは UTF-8 で保存する<sup>\*3</sup>。

1 行目が first.tex で用いるクラスファイルを指定している箇所であり、このクラスファイルが文書全体の大まかなレイアウトを規定する。ここでは、奥村晴彦氏による **p $\text{\LaTeX}$  2<sub>ε</sub> 新ドキュメントクラス** の一つである jsarticle クラスを利用する。

---

<sup>\*2</sup> 当時。現在は株式会社 KADOKAWA 内のアスキー・メディアワークス ブランドカンパニー。

<sup>\*3</sup> Windows 上の up $\text{\TeX}$  には文字コード自動判別機能があるので Shift\_JIS で保存しても動くかもしれないが、今の時代は Unicode (UTF-8) を使うのが自然だろう。

## 1.2 タイプセット

次に「コマンド プロンプト」を起動し、

```
C:\Users\...\> cd Documents
C:\Users\...\Documents> uplatex first.tex
```

と入力<sup>\*4</sup>すると、

```
This is e-upTeX, Version 3.14159265-p3.8.1-u1.23-180226-2.6 (utf8.uptex)
(TeX Live 2018/W32TeX) (preloaded format=uplatex)
restricted \write18 enabled.
entering extended mode
..... (中 略).....
Output written on first.dvi (1 page, 840 bytes).
Transcript written on first.log.
```

のようなメッセージが出て<sup>\*5</sup>、「ドキュメント」の中に first.dvi という中間形式のファイルが生成される。このように T<sub>E</sub>X ソースから DVI ファイルを生成することはタイプセット、またはコンパイルと呼ばれる。

T<sub>E</sub>X を用いて文書を作成する場合、「DVI ファイルを表示して中身を確認する」ことが伝統的であったが、最近では「DVI から PDF ファイルを生成し、それを最終成果物とする」ことが（個人では）多くなり、それに伴い PDF ファイルを中身の確認にも用いることが多くなってきた<sup>\*6</sup>。

本実習においても、DVIPDFM<sub>x</sub> というプログラムによって first.dvi から PDF ファイル first.pdf を生成させ、それを表示するという方針をとることにする。コマンドプロンプト上で

```
C:\Users\...\Documents> dvipdfmx first.dvi
```

と入力すると、

```
first.dvi -> first.pdf
[1]
13562 bytes written
```

というメッセージが出て、PDF ファイル first.pdf が無事に生成される。

2012 年後半以降の T<sub>E</sub>X システムならば、ptex2pdf というプログラムを使うことで、pL<sup>A</sup>T<sub>E</sub>X によるタイプセットとその後の DVIPDFM<sub>x</sub> の処理をまとめて

```
C:\Users\...\Documents> ptex2pdf -l -u first.tex
```

で行うことができる。

```
> ptex2pdf -l -u -ot "-synctex=1 -kanji=utf8" -od "-p jisb5" hoge.tex
```

---

<sup>\*4</sup> ここでも、半角バックスラッシュは実際の画面では半角円記号で表示されているはずである。

<sup>\*5</sup> 最初に「e-」という奇妙な文字列が見えるが、それについてはあまり気にしなくてもよい。

<sup>\*6</sup> 完全に PDF ファイルに移行した、というわけではない。DVI から PostScript ファイルに変換し、それを成果物とする、というのもよく行われている。

のようにオプションを指定すると、次の2行の代わりとなる。

```
> uplatex -synctex=1 -kanji=utf8 hoge.tex
> dvipdfmx -p jisb5 hoge.dvi
```

■課題1 first.tex の3行目「ようこそ、\TeX<sub>ℒ</sub>の世界へ！」をそれぞれ次のようにする。2つはエラーが出るが、それは何故か？ また、残りの1つはどのようなになるか？

ただし、コピー・ペーストはせずに手入力すること。その際には括弧内のコメントに注意。

- ようこそ、¥T e X<sub>ℒ</sub>の世界へ！ (¥T e X 全て全角文字)
- ようこそ、\TeX<sub>ℒ</sub>の世界へ！ (半角スペースを省略)
- ようこそ、\TEX<sub>ℒ</sub>の世界へ！ (Eも大文字)

### 1.3 エラーが起きたら

タイプセット中に、エラーが発生することがある。以下は典型的なエラーの例で、2行目の“\begin{document}”を“\Begin{document}”と間違えたことによって発生したエラーである：

```
! Undefined control sequence.
<recently read> \Begin
```

```
1.2 \Begin
      {document}
?
```

このようなエラーメッセージが表示されると入力待ち状態になっているので、

- “x”と入力するとタイプセットはここで終了する。その後、画面上のエラーメッセージを参照して first.tex の内容を修正することになる。
- 一部のエラーについては、“h”と入力することでヘルプが表示されることがある：

```
! Undefined control sequence.
<recently read> \Begin

1.2 \Begin
      {document}
? h
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

?
```

この場合なら、メッセージに従って“I\begin”と入力することでエラーを修正することができるが、もちろん後からソースファイル first.tex の側も直しておかなければならない。

エラーメッセージの内容については、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Wiki 中の「 [\$\mathrm{L}^{\mathrm{A}}\mathrm{T}\_{\mathrm{E}}\mathrm{X}\$  のエラーメッセージ](#)」や「 [\$\mathrm{T}\_{\mathrm{E}}\mathrm{X}\$  のエラーメッセージ](#)」を参照してほしい。大体は自分の  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  ソース側にエラーがあることが多いが、次のような事態が起こったときは話が別である。

- “! This can't happen” というエラーメッセージが出たとき。
- $\mathrm{pL}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  が「不正な処理」などが原因で強制終了されたとき。

上のどちらかの症状が発生したときは  $\mathrm{pT}_{\mathrm{E}}\mathrm{X}$  のバグの可能性があるので、発生した元のソースをとっておくと共に、TA に連絡すること。

## 1.4 日本語フォントの埋め込み

$\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live のデフォルトでは、(u) $\mathrm{pL}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ +DVIPDFM $x$  で作った PDF ファイルには、IPAex フォントが埋め込まれている<sup>\*7</sup>。

デフォルトで埋め込まれるフォントを変更する方法も用意されている。例えば、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live に同梱されている IPA フォント ([公式サイト](#)) を PDF 内に埋め込みたければ、コマンドプロンプトにおいて、

```
C:\Users\...> kanji-config-updmap --user ipa
```

または

```
C:\Users\...> updmap --user --setoption jaEmbed ipa
```

を実行すればよい<sup>\*8</sup>。以降の DVIPDFM $x$  の実行で作られる PDF ファイルには、日本語フォントとして IPA 明朝・IPA ゴシックが埋め込まれることになる。

macOS に商用・高品質なヒラギノフォントが付属していることは有名である<sup>\*9</sup>。このヒラギノフォントを  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  で利用することもできるが、追加の手順が必要となる。詳しくは  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Wiki の「[ヒラギノフォント](#)」を参照してほしい<sup>\*10</sup>。

なお、kanji-config-updmap や updmap に --user を指定した場合は自分のユーザーアカウントに対してのみ設定を行う。システム全体で設定したい場合は、--user の代わりに --sys を指定する。フォント埋め込み等の設定は、ユーザーごとの設定の方がシステム全体の設定よりも優先度が高いため、一旦ユーザーの設定が作成されたあとはシステム全体の設定は使用されなくなる。

---

<sup>\*7</sup>  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live 2015 までは、(u) $\mathrm{pL}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ +DVIPDFM $x$  で作った PDF ファイルにはデフォルトではフォントが埋め込まれていなかった。

<sup>\*8</sup> 当然ながら、「PDF 内にフォントを埋め込む」場合には、フォントのライセンスによって許されていることを確認しないといけない。

<sup>\*9</sup> 「ヒラギノを買ったら Mac がおまけについてきた」という有名なジョークがある。

<sup>\*10</sup> macOS のバージョンによってフォントファイルの仕様が変わる場合があること、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Wiki の更新はボランティアによって行われていることから、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Wiki の記述が最新の macOS に当てはまらなかったとしても怒らないで頂きたい。

## 1.5 この先は

ひとまず p $\text{\LaTeX}$  によって文書を作成する基本的な方法が身についたので、後は  $\text{\LaTeX}$  の文法を勉強すればよい。

本文書では、書体の変更とか箇条書きの書き方とかそういう基本的な  $\text{\LaTeX}$  の文法については解説しない。代わりに、Web 上で参照できる入門ページとして

- [\$\text{\TeX}\$  Wiki](#) 中の「 $\text{\TeX}$  入門」
- 教育用計算機システム自習教材「[はいばーワークブック](#)」中の「27.  $\text{\LaTeX}$ 」
- 渡辺徹, 「好き好き  $\text{\LaTeX}$  2<sub>e</sub> 初級編」, 2004.  
<http://mytexpert.osdn.jp/index.php?%B9%A5%A4%AD%B9%A5%A4%ADLaTeX2e%2F%BD%E9%B5%E9%CA%D4>

を挙げておく。その他参考となるページには、

- 上田氏の「 [\$\text{\TeX}\$  Live を使おう](#)」
- 熊澤吉起氏の [Web サイト](#) 中の「 $\text{\TeX}$ 」には多数のパッケージの使用例がある。
- Scott Pakin, *The Comprehensive  $\text{\LaTeX}$  Symbol List*.  
<https://ctan.org/pkg/comprehensive>

またはコマンドプロンプトで `texdoc comprehensive` を実行して `symbols-a4.pdf` を開く

があり、 $\text{\TeX}$  Wiki 中の「国内リンク」からも多くのページが辿れる。

$\text{\TeX}$  Live がインストールされた手元の PC にもマニュアル類（多くの場合は英語）が入っている。それらは、コマンドプロンプト（または Windows キー +R で起動できる「ファイル名を指定して実行」）で `texdoc <パッケージ名>` を実行することによって参照できる。例えば、`texdoc` コマンドのマニュアルを参照したければ、`texdoc texdoc` を実行すれば良い。

本格的に学習する際には、まとまった参考書を入手することを強くお勧めする。「はいばーワークブック」の「27.12 参考文献」にいくつか本があげられているし、 $\text{\TeX}$  Wiki の「 $\text{\TeX}$  の本」にもたくさん本が紹介されている。

しかし、当たり前のことだが、一番大事なのは自分の書く文章の中身である。文書の価値は、あくまでもその内容によって上からおさえられるので、 $\text{\TeX}$ ・ $\text{\LaTeX}$  を適切に使い読みやすく見栄えの良い文書を作ったとしても、それは文書の価値を上げることにはならない。

## 1.6 日々の作業の小技

### 1.6.1 Sync $\text{\TeX}$

$\text{\TeX}$  を作って長い文章を作成していると、PDF の表示画面から  $\text{\TeX}$  ソースの該当箇所の編集画面へと、またはその逆のジャンプができると便利である。そのための機能として、最近の p $\text{\TeX}$  などには **Sync $\text{\TeX}$**  という機能が備わっている。

Sync $\text{\TeX}$  の機能を実習用マシンで使うには、以下の設定を行う。

**$\text{\TeX}$  側の指定** p $\text{\LaTeX}$  によるタイプセット時に `-synctex=1` オプションを指定する。

**SumatraPDF 側の設定** 「設定」→「オプション」と辿り、「逆順検索コマンドラインの設定」<sup>\*11</sup>に次のように入力する<sup>\*12</sup>。

```
"C:\Program Files\EmEditor\Emeditor.exe" %f /l %l
```

この設定を行った後で、Sync $\text{\TeX}$  を有効にしてタイプセットされた PDF を表示している SumatraPDF のウィンドウをダブルクリックすると、EmEditor 側が  $\text{\TeX}$  ソースの対応する箇所へとジャンプする<sup>\*13</sup>。

### 1.6.2 latexmk

PDF 作成のために毎回タイプセット・DVIPDFM $x$  を実行するのは面倒である<sup>\*14</sup>。相互参照 (§5.2) を使うと複数回のタイプセットが必要になったり、さらに文献参照や索引などでは外部プログラムを途中で起動しないといけない。先に説明した ptex2pdf を使うと p $\text{\LaTeX}$ ・DVIPDFM $x$  の 1 回の実行でまとめられるとはいっても、それでも面倒なことには変わりはない。

ptex2pdf より本格的に自動化を行うためのプログラムが **latexmk** である。本実習の「 $\text{\TeX}$  Live のインストール」通りに設定したのでは導入されないが、以下のコマンド<sup>\*15</sup><sup>\*16</sup> をコマンドプロンプトに入力することで導入することができる：

```
C:\Users\...\Documents> tlmgr install latexmk  
--repository=ftp://beta.ks.prv/mirror/texlive/tlnet
```

- 上のコマンドを実行した際に “You don’t have permission to ...” といったエラーメッセージが出た場合は、コマンドプロンプトを「管理者として実行」により再度起動した上で、同じコマンドを実行すれば良い。
- tlmgr 自身のアップデートが要求された場合は、

---

<sup>\*11</sup> 先に上記の「 $\text{\TeX}$  側の設定」を行わないと表示されない。

<sup>\*12</sup> 実行ファイルの場所はインストール時の設定によっては違っているかもしれない。デスクトップにあるショートカットを右クリックして「プロパティ」を開くと確認できる。

<sup>\*13</sup> エディタ側を設定すれば、開いている  $\text{\TeX}$  ソースから PDF の該当行へのジャンプすることも可能であるが、筆者は EmEditor Free でそのような設定を行うことができなかった。

<sup>\*14</sup>  $\text{\TeX}$  を使うのであれば、コマンドプロンプトを使えるようになっておくのが良い。

<sup>\*15</sup> latexmk と --repository の間の改行は、本資料の表示の都合によるものであり、実際には入力しない。ただし、スペースは入力する必要がある。

<sup>\*16</sup> tlmgr( $\text{\TeX}$  Live Manager) は  $\text{\TeX}$  Live のパッケージ管理ツール。



```
tlmgr update --self --repository=ftp://beta.ks.prv/mirror/texlive/tlnet
```

をコマンドプロンプトに入力する。

- `--repository=` 以下は実習環境でのみ必要なオプションで、受講生の自宅などの環境では単に `tlmgr install latexmk` などとすれば良いだろう。

`latexmk` は標準では欧文用  $\text{\LaTeX}$  を用いる設定になっているので、そのままでは日本語の入った  $\text{\TeX}$  ソースを処理できない。これを解決するには、`C:\Users\...\latexmkrc`<sup>\*17\*18</sup> に以下の内容を保存する：

```
1 $dvipdf="dvipdfmx %0 -o %D %S";
2 $latex="uplatex %0 %S";
3 $pdf_mode=3;
4 $pdf_previewer = "'C:\Program Files (x86)\SumatraPDF\SumatraPDF.exe" %0 %S';
```

この設定さえしてしまえば、

```
C:\Users\...\Documents> latexmk first.tex
```

で必要な回数のタイプセットと、その後の PDF 作成まで自動でやってくれる。

`-pv` オプションをつけると、PDF を作成した後に PDF ビューア（ここでは 4 行目に指定したとおり SumatraPDF）が起動する。おもしろいのは `-pvc` オプションで、 $\text{\TeX}$  ソースの更新を自動的に感知し、自動で PDF を更新してくれる。

上に述べた `.latexmkrc` の設定はほぼ最小限のものである。よりきちんとした設定については

- [\$\text{\TeX}\$  Wiki](#) 中の「`latexmk`」
- konoyonohana 氏によるブログ「[天地有情](#)」中の「`latexmk` と `ptex2pdf`」

などを参照してほしい。

### 1.6.3 統合開発環境

$\text{\TeX}$  のタイプセットのために毎回コマンドプロンプトを開くのは面倒であるし、コマンドプロンプトの扱いに慣れていない人も少なからずいるだろう。それを避けるためには、 $\text{\TeX}$ works などの統合開発環境（又は Emacs や Vim, Visual Studio Code などの高機能テキストエディタ）を用いると良い。例えば  $\text{\TeX}$ works<sup>\*19</sup>では、画面左上のボタンをクリックすることでタイプセットを実行できる。また、 $\text{\LaTeX}$  コマンドの補完などの便利機能も利用できる。

ただし、「慣れていない」というだけの理由でこれらを用いるのは、本実習では推奨しない。この機会にコマンドプロンプトに慣れておくと良いだろう。

---

<sup>\*17</sup> ... には自分のユーザ名が入る。つまり、ホームディレクトリ直下に。

<sup>\*18</sup> Windows の奇妙な仕様のせいで、保存の際に `.latexmkrc` (最初と最後にドット) という名前を指定する必要がある。

<sup>\*19</sup> スタートメニューから「 $\text{\TeX}$ works editor」を選択すると起動できる

## 2 L<sup>A</sup>T<sub>E</sub>X の作法

L<sup>A</sup>T<sub>E</sub>X を用いればいつでも読みやすい文書が作れる……というわけではない。L<sup>A</sup>T<sub>E</sub>X には L<sup>A</sup>T<sub>E</sub>X の約束事があり、それを守らないと悲惨な結果が生まれてしまう。

まず、L<sup>A</sup>T<sub>E</sub>X で行儀の良いコードを書くための原則を述べておく。

(1) 文字や記号の「意味」を考えて入力する。

数式と地の文は明確に区別して入力する。また、(手書きでは) 似たような見た目の記号でも、その「意味」によって T<sub>E</sub>X での取り扱いは異なる。「綺麗な」文章を出力するためには、これらを正しく T<sub>E</sub>X に伝える必要がある。

(2) 同じことの繰り返しは手動ではやらない。

同じことの繰り返しは機械にやらせるべきである。大概の場合は適切なコマンドやパッケージを用いることで事足りるし、そうでなくても `\newcommand` や `\renewcommand` などを用いて自ら定義すれば良い。

(3) 文章の構造から決まる数値をソースコード中に直接書かない。

例えば定理番号 (例えば「定義 1.2.3」) やページ番号などは、後々文章を修正した際に変更されることがある。これらを直接書いていたら、ひとつひとつ番号を修正する必要性が生じて、非常に面倒である。

(4) 数式や文章の「構造」を意識し、それをソースコードに反映させる。

元々 L<sup>A</sup>T<sub>E</sub>X は、文章の「構造」と「見た目」を分離して記述するように意図して設計されている。そのようにすることで、後から「見た目」を修正するときに、「構造」ごとに一齐に修正することが容易にできる。

以下のページもあわせて参照すると良い：

- 山本裕, 「SICE 著者のための L<sup>A</sup>T<sub>E</sub>X べからず集」, 計測と制御 第 34 巻 11 号, 計測自動制御学会, 1995. [http://doi.org/10.11499/sicej1962.34.11\\_889](http://doi.org/10.11499/sicej1962.34.11_889)
- 小田忠雄, 「数学の常識・非常識 — 由緒正しい T<sub>E</sub>X 入力法」, 数学通信, 第 4 巻第 1 号, 1999. <http://www2.kobe-u.ac.jp/~mhsaito/documents/oda.pdf>
- 斎藤新悟, 「よくあるわけではない T<sub>E</sub>X の間違い」, T<sub>E</sub>X ユーザの集い 2012. <http://oku.edu.mie-u.ac.jp/texconf12/presentations/saito.pdf>
- Donald E. Knuth, *The T<sub>E</sub>Xbook*, Addison-Wesley, 1984.  
L<sup>A</sup>T<sub>E</sub>X の本ではないが, 特に Chapter 18 は数式を入力する際に重宝する。
- 三美印刷株式会社, 「T<sub>E</sub>X 組版から印刷・製本までの工程」, T<sub>E</sub>X ユーザの集い 2011. <http://oku.edu.mie-u.ac.jp/texconf11/presentations/sanbi.pdf>
- 丸田一郎, 「使ってはいけない L<sup>A</sup>T<sub>E</sub>X のコマンド・パッケージ・作法」  
<http://ichiro-maruta.blogspot.jp/2013/03/latex.html>

この章では、ついやってしまいがちな間違いとその修正方法を並べておく。しかし、この章で書いたのは一般的なことでしかない。論文誌の投稿規定や各出版社の組版ルールがある場合には、当然な

がらそちらを優先し、勝手な変更や超絶技巧はしないこと。

## 2.1 数式

特に数式を入力する際には、その記号はどのような（組版上の）役割で用いられているか、ということに気にするのが大事である。

### 2.1.1 「数式」と「数式以外」

例 1 間違い：数式を数式モードに入れていない

$f(x)$  は  $[0,1]$  で連続,  $f(0)=-1$  かつ  $f(1)=1$  なので,  $f(x)=0$  となる  $0<x<1$  が存在する.

- 1  $f(x)$  は  $[0,1]$  で連続,  $f(0)=-1$  かつ
- 2  $f(1)=1$  なので,  $f(x)=0$  となる  $0<x<1$  が存在する.

例 2 間違い：地の文まで数式モードに入れている

$f(x)$  は  $[0,1]$  で連続,  $f(0) = -1$  かつ  $f(1) = 1$  なので,  $f(x) = 0$  となる  $0 < x < 1$  が存在する.

- 1  $f(x)$  は  $[0,1]$  で連続,  $f(0)=-1$  かつ
- 2  $f(1)=1$  なので,  $f(x)=0$  となる  $0<x<1$  が存在する.

例 3 正しい入力

$f(x)$  は  $[0,1]$  で連続,  $f(0) = -1$  かつ  $f(1) = 1$  なので,  $f(x) = 0$  となる  $0 < x < 1$  が存在する.

- 1  $f(x)$  は  $[0,1]$  で連続,
- 2  $f(0)=-1$  かつ  $f(1)=1$  なので,
- 3  $f(x)=0$  となる  $0<x<1$  が存在する.

### 2.1.2 基本

例 4  $\sin$  などの関数名は専用の命令を使って立体で表記すること。

誤:  $\sin x$ ,  $\sin x$   
正:  $\sin x$

- 1 誤:  $\sin x$ ,  $\mathrm{\sin} x$
- 2 正:  $\sin x$

1 行目左は  $s \cdot i \cdot n \cdot x$  の意味に解釈されてしまう。1 行目右も空白が足りない。なお、 $\sin$  のような専用命令が予め準備されていない状況もあるが、その場合は定義すれば良い。例えば、 $\dom$  を使いたければプリアンブルに

```
\newcommand\dom{\mathop{\mathrm{dom}}\nolimits}
```

と記述するか、あるいは `amsmath` パッケージ<sup>\*20</sup>を読み込んだ上で


```
\DeclareMathOperator{\dom}{dom}
```

と記述すれば良い<sup>\*21</sup>。いちいち命令を定義したくない場合は、`\operatorname` 命令を使って数式中

<sup>\*20</sup> アメリカ数学会による、数式組版の機能を拡張するパッケージ。数学科の諸君なら、`amssymb` パッケージと共にいつでも読み込むようにしておこう。基本的なマニュアルは、[User's Guide for the amsmath Package](#)。実習用マシンでは `texdoc amsmath` で参照できる。

<sup>\*21</sup> 別行立て数式の中で添字の位置を  $\mathrm{Res}_{z=a}$  ではなく  $\mathrm{Res}_{z=a}$  のように真下にしたい場合は、スター \* 付きの `\DeclareMathOperator*` を使う。例: `\DeclareMathOperator*{\Residue}{Res}`

に直接 `\operatorname{dom}` と書くこともできる。

 単なる `\mathrm` ではなく `\mathop` を使う<sup>\*22</sup>ことの効果は「直後に変数が置かれた場合に適宜空白をあける (例:  $\sin x$ )」「ただし、括弧類が置かれた場合は空白を入れない (例:  $\sin(x+y)$ )」である。だが、それゆえの注意点もある。

例えば、スラッシュ / の直後にこのような関数名が置かれた場合に、関数名の前に空白が入ってしまう (例:  $f/\gcd(f,g), f/\gcd(f,g)$ )。この空白を入れないためには、関数とその引数を波括弧で囲えば良い (例:  $f/{\gcd(f,g)}, f/{\gcd(f,g)}$ )。

別の例として、関数に対して合成やテンソル等の演算子を適用するときに、演算子の後の空白が変わってしまう場合がある (例: `\operatorname{id}\circ f`, `id\circ f` では  $\circ$  と  $f$  の間の空白が消えてしまう。比較:  $g\circ f$ )。これも関数名を波括弧で囲えば良い (`{\operatorname{id}}\circ f`, `id\circ f`)。あるいは、圏の射のように引数を適用することがない ( $id\,x$  のような書き方をしない) 場合は `\mathop` を使わずに単に `\mathrm` で済ませてしまうという手もある (例: `\newcommand{\id}{\mathrm{id}}`)。

詳しくは §5.5 を参照。

**例 5** 和文文字を数式中そのまま使用するのは有害である。

誤:  $a + b + \cdots + z, f(a, b, \cdots, z)$   
正:  $a + b + \cdots + z$

1 誤:  $\$a+b+\cdots+z$,  $\$f(a,b,\cdots,z)\$$   
2 正:  $\$a+b+\cdots+z\$$$

なお、Unicode にある数学記号をそのまま  $\mathrm{T}_\mathrm{E}\mathrm{X}$  ソース中で使いたい、という場合は、実習資料の「 $\mathrm{pT}_\mathrm{E}\mathrm{X}$  系列以外による日本語文書作成」(tex\_mik.pdf) にある `unicode-math` パッケージの紹介を読むと良いかもしれない。

**例 6** 省略記号は周りの記号によって使い分けること。

誤:  $f(a, b, \cdots, z)$   
正:  $f(a, b, \ldots, z)$

1 誤:  $\$f(a,b,\cdots,z)\$$   
2 正:  $\$f(a,b,\ldots,z)\$$

`amsmath` パッケージを読み込んでいるなら、`\dotsb` などの命令も参照。なお、 $\mathrm{L}_\mathrm{A}\mathrm{T}_\mathrm{E}_\mathrm{X}$  で標準で用意されている `\dots` 命令だが、`amsmath` パッケージを読み込んでいるかで動作が変わり、

- `amsmath` パッケージ未使用なら、`\dots` と `\ldots` は等価
- `amsmath` パッケージ読み込み時は、`\dots` は後ろの記号によってある程度使い分けをする

となっている。

### 2.1.3 括弧類

**例 7** 中身に対して小さすぎる括弧を使うのは良くない。

誤:  $(\frac{a}{2} + \frac{b}{3})c$   
正:  $\left(\frac{a}{2} + \frac{b}{3}\right)c$

1 誤:  $\$\displaystyle(\frac{a}{2}+\frac{b}{3})c\$$   
2 正:  $\$\displaystyle$   
3  $\left(\frac{a}{2}+\frac{b}{3}\right)c\$$

この `\left`, `\right` による自動調整も万能ではなく、次のように「望ましくない」サイズの括弧を選択する可能性もある。

<sup>\*22</sup> `\DeclareMathOperator` や `\operatorname` も内部で `\mathop` を使っている



例 9 「本来の意味」と異なる使い方で記号を使う場合も注意.

誤: $x \in ]-a, b]$	1 誤: $\$x\in ]-a,b] \$$
正: $x \in ]-a, b], x \in ]-a, b]$	2 正: $\$x\in \mathop{]}-a,b] \$,$
	3 $\$x\in \left]-a,b\right] \$$

普通に `]` と入力すると「閉じ括弧扱い」になるが, ここでは開き括弧の意味で用いているので空白に異常が生じる. `\mathopen{]}` によって, 強制的に「開き括弧」扱いにすることで解決.

例 10 山括弧  $\langle \rangle$  は不等号  $< >$  と違うものである. この間違いが最も多く見られる.

誤: $< a, b >$	1 誤: $\$<a,b> \$$
正: $\langle a, b \rangle, \langle a, b \rangle$	2 正: $\$\left\langle a,b\right\rangle \$, \ \$\left<a,b\right> \$$

ちなみに, `\left`, `\right` 等の直後に書いた  $< >$  は山括弧として扱われる (「正:」の 2 番目).

## 2.1.4 その他

例 11 句読点の意味でコロン  $:$  を使う場合は `\colon` を用いる.

誤: $f: A \rightarrow B$	1 誤: $\$f:A\rightarrow B \$$
正: $f: A \rightarrow B$	2 正: $\$f\colon A\rightarrow B \$$

例 12 関係演算子として使うコロン  $:$  はそのままでよい.

正: $a:b=1:2$	1 正: $\$a:b=1:2 \$$
--------------	---------------------



垂直方向のスペーシングという点では, 定義に使う “ $:=$ ” を下の 1 行目のようにそのまま記述してもよい. しかし, 1 行目ではコロンと等号の垂直位置が微妙に合っておらず, それを気にする人も多いだろう.

$f := a^2 - b + c^d$	1 $\$f:=a^2-b+c^d \$$
$f := a^2 - b + c^d$ (mathtools 等要)	2 $\$f\coloneqq a^2-b+c^d (\text{mathtools 等 要}) \$$

その場合は, 2 行目のように, 例えば `mathtools` パッケージで定義されている `\coloneqq` などを使えば良い.

例 13 集合の内包的記法

$\{x \mid x > 5\}$	1 $\$\{\backslash,x\mid x>5\backslash,\backslash \$$
$\{x : x > 5\}$	2 $\$\{\backslash,x:x>5\backslash,\backslash \$$
$\{x : x > 5\}$	3 $\$\{x:x>5\backslash \$$
$\{x \mid x > 5\}$	4 $\$\{x\mid x>5\backslash \$$
$\{x x > 5\} \leftarrow$ 誤	5 $\$\{x x>5\backslash \$ \leftarrow$ 誤

*The T<sub>E</sub>Xbook* ではなぜか 1, 2 行目のような記法が紹介されている<sup>\*25</sup>が, 実際の数学書を見ると 3, 4 行目のように括弧の内側を空けない書き方のほうが多い. 何はともあれ, 5 行目のように記述するのは誤り.

<sup>\*25</sup> `\mid` は関係演算子扱いの縦棒.

例 14 複数の行中数式をつなげる場合は一旦数式モードを抜けること。

誤 : $f(x) = x^2, g(x) = x^2 + 1$
正 : $f(x) = x^2, g(x) = x^2 + 1$

1	誤 : $f(x) = x^2, g(x) = x^2 + 1$
2	正 : $f(x) = x^2, g(x) = x^2 + 1$

The *T<sub>E</sub>Xbook* では、`...$`、`\ $...` と余分に空白を空ける方法も紹介されている。

また、標準のクラスファイルでは `eqnarray` 環境にバグがあることが知られている。pL<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 新ドキュメントクラスなど、修正されているクラスも存在するが、「`amsmath` パッケージを読み込んだ上で `align` 環境を用いる」ことが一番良い解決法であろう。次の記事が状況を解説している：

Lars Madsen, *Avoid eqnarray!*, The PracT<sub>E</sub>X Journal, 2006, No. 4.

<http://tug.org/pracjourn/2006-4/madsen/madsen.pdf>

## 2.2 地の文

数式ではない、いわゆる「地の文」を打つときにも注意が必要である。以下は英文を入力する際の基本的な注意である：

- 当たり前だが、英文を打つときは半角英数字を利用すること。
- 句読点の役割を果たす `;`、`,` については、直前の文字との間には空白を入れず、直後にスペースを打つ。
- ピリオド `.`（や疑問符 `?`、感嘆符 `!`、コロン `:`）の直前にも空白は入れない。直後にはほとんどの場合に空白を 1 つ入れる。T<sub>E</sub>X は、標準で通常の単語間空白と文間空白を区別する：
  - － 大文字直後のピリオドは省略を意味するものと扱い、直後の空白は単語間空白となる。  
文間空白だと認識させたい時はピリオドの直前に `\@` を補う。
  - － 小文字直後のピリオドは文の終了を意味するものと扱い、直後の空白は文間空白となる。  
単語間空白だと認識させたい時は `\_` を使用する。

文間空白    AA. ccc. ddd            AA \@. \_ccc. \_ddd

単語間空白   AA. ccc. ddd            AA. \_ccc. \\_ddd

最近では、単語間空白と文間空白の大きさを一緒にして組む場合 (`\frenchspacing`) も多く、その場合には上の使い分けは見た目には影響しないことになる。

- 引用符には左右の区別が存在し、左右とも同じ `"'` で入力してはならない。  
なお、数式中の `'` は引用符とは別の記号である。

誤    "def 'ghi' jkl", f'            "def 'ghi' jkl'", \$f\$'

正    "def `ghi' jkl", f'            ``def `ghi' jkl'', \$f'\$

- 開き括弧 `[ { [ {` や開引用符 `" ' ‘` の直後、それに閉じ括弧 `) ] }` や閉引用符 `" ' ’` の直前にはスペースは入れない。
- 手書きでやテキストファイルではあまり区別していないが、`-` (hyphen), `–` (en-dash), `—` (em-dash), `－` (負号) の 4 種類の記号は明確に区別される：

1.  $P$ -generic filter, well-defined      $\mathbb{P}$ -generic filter, well-defined

2. pp. 29–42, 13:00–14:00     pp.~29--42, 13:00--14:00

範囲を表す「～」は欧文では用いられない<sup>\*26</sup>.

Jech–Sochor Theorem     Jech--Sochor Theorem

2人以上の人名を並べる場合. Jech-Sochor Theorem とハイフンで済ませる場合もあるが、それだと「Jech-Sochor という 1 人の人なのか」と誤解される危険がある.

3. This fact—it is easy to prove—is very useful.

This fact---it is easy to prove---is very useful.

## 2.3 文章の構造

L<sup>A</sup>T<sub>E</sub>X で文章を書く際には、文章の「構造」をなるべく T<sub>E</sub>X ソースに反映させるべきである。そのための手法をいくつか紹介しておこう。なお、`\label` や `\ref` については 5.2 節を参照せよ。

### 2.3.1 章立て

章立てを記述するには、`\section` コマンドを用いる。`\subsection` や `\subsubsection` を用いることで、より細かく分けることができる。

```
\documentclass[uplatex]{jsarticle}

\begin{document}
\section{sectionコマンドの使い方}
こんな感じで使える。
\subsection{subsectionコマンドの使い方}
こうやって使う。
\subsection{相互参照} \label{subsec:ref}
labelとrefを使って、相互参照することもできる。

\section{2つ目の章}
refコマンドを用いて\ref{subsec:ref}節のlabelを参照してみる。
\end{document}
```

section 番号の表示形式のカスタマイズは `\renewcommand{\thesection}{\arabic{section}}` 節などとする一応可能だが、subsection や定理番号の表示がおかしくなってしまう。このあたりは <https://togetter.com/li/1124861> などを参考にして、各自調べること<sup>\*27</sup>。

### 2.3.2 簡条書き

しばしば次のように強制改行 (`\`), 中黒「`\cdot`」, 全角空白「`\quad`」を使って簡条書きをレイアウトしているような例を見る：

$\mathbb{R}$  に右半開区間位相を入れたものを  $\mathcal{S}$  とおく。

---

<sup>\*26</sup> なお、上の入力中にある（いわゆる半角の）`~` は、「ここでの改行を禁止する空白」を表す。

<sup>\*27</sup> 結論を言うと、文書クラスが `BXjscls` であればクラスオプション `label-section=modern` を指定し、`\thesection` の代わりに `\pre/postsectionname` をカスタマイズすれば良い。それ以外の文書クラスの場合は、上手いカスタマイズ方法はない。



すなわち、 $\mathcal{S}$  の位相の基は  $[a, b] \mid a < b$  である。

このとき、次が成立する。\\

1.  $\mathcal{S}$  は Hausdorff 空間である。\\
2.  $\mathcal{S}$  は第2可算公理を満たさない。\\
3.  $\mathcal{S}$  は Lindelöf 空間である。\\
4.  $\mathcal{S} \times \mathcal{S}$  は Lindelöf 空間でない。

しかし、このような書き方は見た目がみすばらしくなるだけでなく、文書内の論理的構造がわかりにくくなるため、基本的には使わないこと。下記の例のように、きちんと `itemize`, `enumerate` などの L<sup>A</sup>T<sub>E</sub>X で用意されている環境を使うのがよい。詳細については例えば「美文書作成入門」の第 3.20 節<sup>\*28</sup>「箇条書き」を参照。

#### 例 15 番号なし

- hoge
- fuga

```
1 \begin{itemize}
2   \item hoge
3   \item fuga
4 \end{itemize}
```

#### 例 16 番号あり

1. hoge
2. fuga

```
1 \begin{enumerate}
2   \item hoge
3   \item fuga
4 \end{enumerate}
```

また、番号の形式を変更したい場合は以下のようにすれば良い。

#### 例 17 括弧つき番号

- (1) hoge
- (2) fuga

```
1 \begin{enumerate}
2   \renewcommand{\labelenumi}{(\theenumi)}
3   \item hoge
4   \item fuga
5 \end{enumerate}
```

#### 例 18 括弧つきアルファベット

- (a) hoge
- (b) fuga

```
1 \begin{enumerate}
2   \renewcommand{\theenumi}{\alph{enumi}}
3   \renewcommand{\labelenumi}{(\theenumi)}
4   \item hoge
5   \item fuga
6 \end{enumerate}
```

なお、これらの item の番号も `\label` と `\ref` で参照することができる。後々 item を入れ替える可能性を考慮すると、直接書くのではなくこれらを利用して参照した方が良いだろう。

また、`enumitem` パッケージを用いると、より高度なカスタマイズが行える。使用方法などについては、各自検索すること。

<sup>\*28</sup> これは第7版の場合。旧版の場合は異なっているかもしれない。

### 2.3.3 定理環境

数学の授業や文章においては、「定理 3.1.4」などの見出しとともに定理の中身を記述することが多い。よく使われる形式なので、当然便利なツール<sup>\*29</sup>が用意されている。使用例を挙げておくと、以下の通りとなる。

```
1 \documentclass[uplatex]{jsarticle}
2 \usepackage{amsthm}
3 \theoremstyle{definition}
4 \newtheorem{thm}{定理}[section]
5 \newtheorem{prop}[thm]{命題}
6 \newtheorem{defn}[thm]{定義}
7 \newtheorem*{rmk}{注意}
8
9 \begin{document}
10 \section{定理環境の使い方}
11 \begin{thm}
12   \label{thm:sample}
13   これは定理環境です。
14 \end{thm}
15 定理\ref{thm:sample}はすごく偉大な定理である。
16 \end{document}
```

4-7 行目のように定理環境を定義した後に、`\begin{thm}` と `\end{thm}` などによって<sup>\*30</sup>利用する。定理番号も自動で割り振られる。上記のソースコードについて、簡単に解説をしておく。

- 3 行目の `\theoremstyle{definition}` は、それ以降の `\newtheorem` で定義される定理環境の style を設定している。デフォルトの style では定理環境中では文字が斜体になるが、通常日本語の文章では斜体にしないため、この設定を行っている。
- 4 行目の `[section]` や 5 行目の `[thm]` などのオプションは、定理などの番号の付け方を指定している。前者は section 番号を頭につけるように、後者は番号を複数の環境を引き継ぐようにしている。また、7 行目の `\newtheorem*` は、番号の割り振りを行わないものである。
- 12 行目の `\label` と 15 行目の `\ref` のペアにより、定理番号の参照を行っている。自動的に割り振られた番号を利用する、順番の入れ替えや新たな定理の挿入などを行った際にも番号が自動的に修正される。なお、この label の名前は、`thm3` のように番号でつけるのではなく、定理の中身を反映した名前にしておく、後々の修正の際に問題が起きにくい。

また、定理環境に関連したコメントをいくつか載せておく。

- 「例」や「用語」なども、定理などと同様の形で見出しとして使われている場合には `\newtheorem`(または `\newtheorem*`) を用いると良いだろう。

---

<sup>\*29</sup> L<sup>A</sup>T<sub>E</sub>X 本体の機能だが、通常は `amsthm` パッケージにより機能が強化されたものを用いる。

<sup>\*30</sup> 実は、これらの代わりに `\thm` や `\endthm` など書いても動作はするが、可読性などの観点から上記の `\begin`, `\end` を用いた書き方をすべきである。

- 証明についても、「証明.」や「Proof.」などと直接書くのではなく、証明のための環境を用いるべきである。例えば `amsthm` で定義されている `proof` 環境を用いる<sup>\*31</sup>と良いだろう。
- 定理環境の右下に終了記号をつける場合は、例えば `thmtools` パッケージを用いて以下のように定理環境を定義すると良い。

```
\declaretheoremstyle[qed=\qedsymbol]{mythmstyle}
\declaretheorem[style=mythmstyle,title=定理]{thm}
```

---

<sup>\*31</sup> `\begin{proof}` と `\end{proof}` で囲う。

### 3 §1 と §2 に関連した課題

本節には課題をたくさん載せているが、当然ながら全部取り組まなければならないというものではない。あと、「課題」と名乗ってはいるが、提出不要である。

### 3.1 T<sub>E</sub>X ソースの例

■課題 2 以下のソースファイルを自分で入力し，そこから PDF ファイルを作成せよ．全角文字と半角文字を間違えないようにすること<sup>\*32</sup>．

また、一般的に言えることだが、「一度に多量の文章を記述してタイプセットをする」と、エラー対応や誤植チェックがやりにくい。そのため、タイプセットに時間がかかるのでなければ、ソースを打っている途中にこまめにタイプセットし、様子を見るのが良い。

```

1 \documentclass[uplatex]{jsarticle}
2 \usepackage{amsmath, amssymb, amsthm}
3
4 \theoremstyle{definition}
5 \newtheorem*{thm}{定理}
6
7 \title{計算数学I\quad \TeX 初級者向けの写経課題}
8 \begin{document}
9 \maketitle
10
11 本文章は、高木貞治『代数的整数論』の「第1章 代数的整数」の一部を
12 引用、改変したものである。
13
14 \bigskip
15
16 代数的の数とは、有理数を係数とする代数方程式の根をいう。
17 その方程式が一次ならば、根は有理数である。
18 故に有理数は代数的な数の特別の場合である。
19 有理係数の方程式を、しばしば、次のような標準的の形に書いて取扱う。
20
21 \begin{enumerate}
22 \renewcommand{\labelenumi}{(\arabic{enumi})$^{\circ}$}
23 \item 首項の係数で割れば、他の係数は有理数で、方程式の形は
24 \[
25 x^n+a_1x^{n-1}+\cdots+a_n=0,\quad \text{$a_i$は有理数。}
26 \]
27 \item 係数の分母を払って、公約数を取り去れば、方程式は
28 \[
29 a_0x^n+a_1x^{n-1}+\cdots+a_n=0.
30 \]
31 左辺は所謂原始的多項式である。

```

\*32 特に、7行目の「I」は半角英大文字であって、全角英文字「I」や全角ローマ数字ではない。

32 即ち  $a_0, \sim a_1, \dots, \sim a_n$  は整数で、それらの最大公約数は 1.  
 33 なお  $a_0 > 0$  と仮定してもよい.  
 34  $\end{enumerate}$   
 35  
 36  $\begin{thm}$   
 37 代数的の数の和は、再び代数的の数となる.  
 38  $\end{thm}$   
 39  
 40  $\begin{proof}$   
 41 二つの代数的の数  $\alpha, \sim \beta$  に対し、  
 42  $\xi = \alpha + \beta$  が代数的の数となることを示す.  
 43  $\alpha, \sim \beta$  は代数的の数であるから、それぞれを根にもつ有理係数方程式  
 44  $\begin{align}$   
 45  $x^m + a_1 x^{m-1} + \dots + a_m = 0, \text{\label{alpha}} \backslash \backslash$   
 46  $x^n + b_1 x^{n-1} + \dots + b_n = 0 \text{\label{beta}}$   
 47  $\end{align}$   
 48 が存在する.  
 49  
 50 次に、 $\mu = 0, \sim 1, \dots, \sim m-1$  と  $\nu = 0, \sim 1, \dots, \sim n-1$  に対して、  
 51  $\begin{equation}$   
 52  $\alpha^{\mu} \beta^{\nu} \text{\label{multi}}$   
 53  $\end{equation}$   
 54 を辞書式順序で並べたものを  
 55  $\omega_1, \sim \omega_2, \dots, \sim \omega_l \backslash \backslash \text{ (} l = mn \text{)}$   
 56 と書けば、各  $i = 1, \sim 2, \dots, \sim l$  に対して、対応する  $\mu$  と  $\nu$  を用いて  
 57  $\backslash$   
 58  $\xi \omega_i = (\alpha + \beta) \alpha^{\mu} \beta^{\nu}$   
 59  $= \alpha^{\mu+1} \beta^{\nu} + \alpha^{\mu} \beta^{\nu+1}$   
 60  $\backslash$   
 61 が成り立つ.  
 62  
 63  $0 \leq \mu \leq m-2$  かつ  $0 \leq \nu \leq n-2$  ならば、  
 64  $\xi \omega_i$  は  $\text{\label{multi}}$  の二つの数の和に等しい.  
 65 また、もし  $i = m-1$  または  $j = n-1$  ならば、 $\text{\label{alpha}}, \sim \text{\label{beta}}$  より  
 66  $\begin{align*}$   
 67  $\alpha^m + a_1 \alpha^{m-1} + \dots + a_m = 0, \backslash \backslash$   
 68  $\beta^n + b_1 \beta^{n-1} + \dots + b_n = 0$   
 69  $\end{align*}$   
 70 であるから、 $\alpha^m = -a_1 \alpha^{m-1} - \dots - a_m$  と  
 71  $\beta^n = -b_1 \beta^{n-1} - \dots - b_n$  を代入することによって、  
 72  $\xi \omega_i$  を  $\omega_1, \sim \omega_2, \dots, \sim \omega_l$  たちの  
 73 有理係数の線型結合として表すことができる.  
 74  
 75 以上をまとめると、 $i = 1, \sim 2, \dots, \sim l$  と  $j = 1, \sim 2, \dots, \sim l$  に対して  
 76 有理数  $c_{ij}$  が存在して  
 77  $\backslash$   
 78  $\xi \omega_i = \sum_{j=1}^l c_{ij} \omega_j$   
 79  $\backslash$   
 80 が成り立つ. これを用いて、 $l$  次正方行列の行列式に関する方程式

```

81 \[
82 \begin{vmatrix}
83 c_{11}-\xi & c_{12} & \cdots & c_{1l} \\
84 c_{21} & c_{22}-\xi & \cdots & c_{2l} \\
85 \vdots & \vdots & \ddots & \vdots \\
86 c_{l1} & c_{l2} & \cdots & c_{ll}-\xi
87 \end{vmatrix}
88 =0
89 \]
90 を考える．左辺を展開すれば，
91 \[
92 \xi^{l+1}+\tilde{c}_1\xi^{l-1}+\cdots+\tilde{c}_l=0
93 \]
94 を得る．
95 ここに，各 $\tilde{c}_i$ は有理数 $c_{ij}$ たちの整式として表される実数である．
96 特に，各 $\tilde{c}_i$ も有理数であるから， $\xi$ は代数的の数である．
97 \end{proof}
98 \end{document}

```

### 3.2 数学の文章を $\text{T}_\text{E}\text{X}$ で書く練習

以下の数学の問題を解き， $\text{T}_\text{E}\text{X}$  によって解答を作れ．その際，§2のような間違いを起こさないように注意せよ．

■課題3 関数  $f: \mathbb{R}^2 \rightarrow \mathbb{R}$  を

$$f(x, y) := \begin{cases} 0 & \text{if } (x, y) = (0, 0), \\ (x^3 - y^3)/(x^2 + y^2) & \text{otherwise} \end{cases}$$

で定める．この  $f$  が連続関数であることを示せ．

■課題4  $n \in \mathbb{N}$  に対し，

$$I_n := \int_0^{\pi/2} \cos^n \theta \, d\theta$$

の値を求めよ．なお， $\mathbb{N}$  に 0 を含めるか否かについては，どちらでもよい．

■課題5

$$A := \begin{pmatrix} 2/5 & 11/5 \\ -1 & 2 \end{pmatrix}$$

に対し，

- (1)  $A^t A$  を求め，対角化せよ． $^t$  は転置の記号である．
- (2) 前問を用いて，正定値対称行列  $B$  で  $B^2 = A^t A$  となるものを求めよ．
- (3) 前問を用いて， $A$  を正定値対称行列と直交行列の積として表せ．

■課題6  $T$  を Hausdorff 空間とする． $T$  内の点列  $(p_n)_{n \in \mathbb{N}}$  が点  $p$  と点  $q$  に収束するならば， $p = q$  であることを示せ．

■課題 7 2次元トーラス  $T^2$  の整係数ホモロジー群を求めよ.

■課題 8 直後の課題 10 の問題文にある  $\mathcal{S}$  (Sorgenfrey 直線) に関する 1.-3. の性質を証明し,  $\text{T}_\text{E}\text{X}$  ソースの形で解答を書け. その際, 本章に今まで記述したような間違いを起こさないように注意せよ. なお, 位相空間が Lindelöf 空間であるとは, 任意の開被覆から可算部分被覆がとれることを言う.

### 3.3 $\text{T}_\text{E}\text{X}$ ソースの修正練習

■課題 9 2.3.2 節で挙げられていた「悪い例」:

$\mathbb{R}$  に右半開区間位相を入れたものを  $\mathcal{S}$  とおく.  
すなわち,  $\mathcal{S}$  の位相の基は  $\{[a, b[ \mid a < b\}$  である.  
このとき, 次が成立する. \\\n1.  $\mathcal{S}$  は Hausdorff 空間である. \\\n2.  $\mathcal{S}$  は第2可算公理を満たさない. \\\n3.  $\mathcal{S}$  は Lindelöf 空間である. \\\n4.  $\mathcal{S} \times \mathcal{S}$  は Lindelöf 空間でない.

をきちんとした  $\text{T}_\text{E}\text{X}$  ソースに書き直せ.

以下の 2 つの課題については, 問題になっているソースファイルを実習資料集の「 [\$\text{T}\_\text{E}\text{X}\$  実習](#)」からダウンロードして取り組むこと (この PDF からリンクをたどっても良い).

■課題 10 [KS\\_3a.tex](#) はタイプセットが正常にできない (エラーが発生する) ソースである. タイプセットが可能になるように修正せよ.

■課題 11 [KS\\_3b.tex](#) はタイプセット自体は可能だが, 誤植や見た目の良くない点を含んでいる. §2 の記述を参考にしてソースを修正せよ.

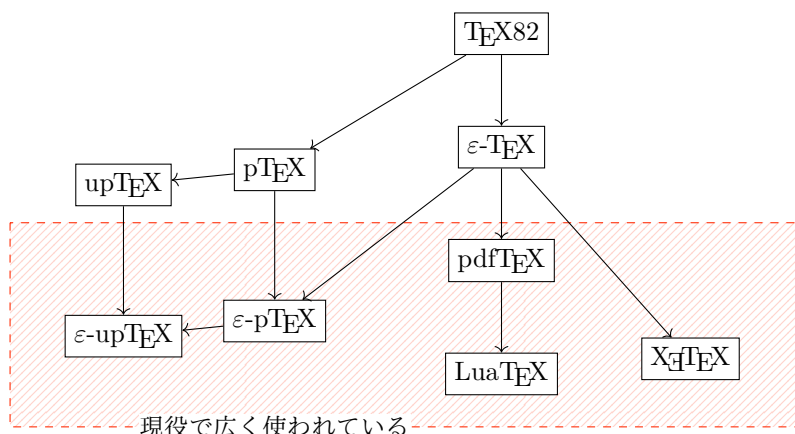


図 1 現在に連なる主な TeX エンジン

## 4 pTeX 系列以外の TeX エンジンについて

### 4.1 TeX エンジン

「TeX」と言った時に指すものは文脈によって色々ありうるが、実際に組版処理を行うプログラムとしての TeX を **TeX エンジン**と呼ぶ。Knuth 氏が 1970 年代に開発した TeX はその後様々な人によって拡張が加えられ、いくつもの改良版が誕生している。

この実習では TeX エンジンとしては upTeX を使っているが、受講生諸君が今後 TeX で文書を書いていく際、pTeX 系以外の TeX エンジンを使うこともあるだろう。というわけで、主な TeX エンジンを簡単に紹介しておく。エンジン間の派生関係を大雑把に表すと図 1 である<sup>\*33</sup>。

**TeX82** Knuth 氏が作った、現在に連なる各種 TeX エンジンの直接の祖先。

**ε-TeX** TeX82 に拡張を施したもの。ε-TeX 拡張のうち一般人に馴染みの深いものとしては、`\middle` コマンドがある。

**pdfTeX** DVI ではなく PDF を直接出力できるようにしたもの。欧米でのスタンダード。

**XeTeX** OS のフォントを直接使えるようにしたもの。pdfTeX と同じく、PDF を直接出力する。また、Unicode に直接対応する。

**LuaTeX** スクリプト言語 Lua を組み込んだもの。Unicode に直接対応する。

**pTeX** アスキーが日本語対応させたもの。

**upTeX** pTeX を Unicode 対応させたもの。

**ε-pTeX, ε-upTeX** pTeX, upTeX に ε-TeX 拡張を施したもの。

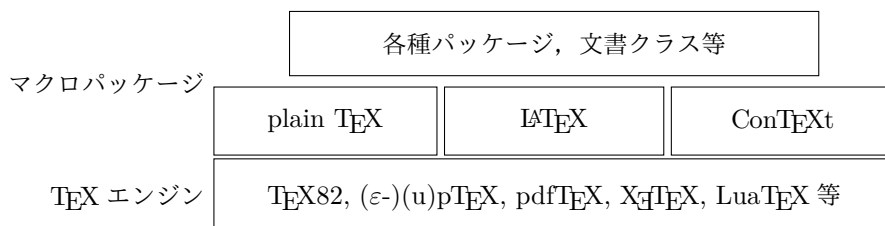
なお、現在の TeX Live で `uplatex` コマンドを実行した時に使用されるエンジンは ε-upTeX である。

<sup>\*33</sup> 最近の LuaTeX は pdfTeX との命令名レベルの互換性を切り捨てているので派生関係にあるからといって必ず互換性があるというわけではないし、逆に pdfTeX 派生ではない ε-pTeX や XeTeX に pdfTeX 由来の命令が取り込まれていたりするので、派生関係にないからといって全く互換性がないというわけでもない。



## 4.2 マクロパッケージ

T<sub>E</sub>X エンジンに組版の機能が備わっていると言っても、一般のユーザーがそれを直接使って文書を書くのは大変である。そこで、T<sub>E</sub>X のマクロによって命令や文法を追加し、文書を書きやすくするマクロパッケージが多数開発されている。L<sup>A</sup>T<sub>E</sub>X はそうしたマクロパッケージの一種である。例えば、文書クラス `\documentclass` や環境 (`\begin`, `\end` コマンド) 等の概念は、L<sup>A</sup>T<sub>E</sub>X によって提供されたものである。L<sup>A</sup>T<sub>E</sub>X 以外の枠組みとしては、plain T<sub>E</sub>X や ConT<sub>E</sub>Xt がある。



さて、LuaT<sub>E</sub>X や X<sub>g</sub>T<sub>E</sub>X は多言語に対応していると言っても、エンジン自体は日本語組版の事情を知らない。したがって、LuaT<sub>E</sub>X や X<sub>g</sub>T<sub>E</sub>X で日本語組版を行うには、そのためのマクロパッケージが必要である。LuaT<sub>E</sub>X の場合は LuaT<sub>E</sub>X-ja が、X<sub>g</sub>T<sub>E</sub>X の場合は xeCJK や ZXjatype がそれである。

ちなみに、この資料 (`tex_practice.pdf`) は LuaT<sub>E</sub>X (& LuaT<sub>E</sub>X-ja) を用いて組版されている。

## 4.3 文書クラス

jsarticle をはじめとする「pL<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 新ドキュメントクラス」(jsclasses) はその名の通り pL<sup>A</sup>T<sub>E</sub>X 系専用である。よって、LuaT<sub>E</sub>X や X<sub>g</sub>T<sub>E</sub>X 等のエンジンを用いて日本語文書を書く際には、jsclasses 以外の文書クラスを使うことになる。というわけで、日本語文書を書くのに使える文書クラスを (pL<sup>A</sup>T<sub>E</sub>X 専用のものも含めて) いくつか挙げておく。

**jlclasses** アスキーによる、pT<sub>E</sub>X 用の文書クラス。(u)pL<sup>A</sup>T<sub>E</sub>X 用

**jsclasses** 奥村晴彦氏による「pL<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 新ドキュメントクラス」。(u)pL<sup>A</sup>T<sub>E</sub>X 用。

**ltjsclasses** jsclasses を LuaT<sub>E</sub>X-ja 用に移植したもの。

**BXjscls** jsclasses を拡張したもの。(u)pL<sup>A</sup>T<sub>E</sub>X, pdfL<sup>A</sup>T<sub>E</sub>X, X<sub>g</sub>L<sup>A</sup>T<sub>E</sub>X, LuaL<sup>A</sup>T<sub>E</sub>X と、幅広いエンジンに対応している。

**jlreq** 「日本語組版の要件」<sup>\*34</sup>を満たすことを目指す文書クラス。LuaT<sub>E</sub>X-ja, pL<sup>A</sup>T<sub>E</sub>X, upL<sup>A</sup>T<sub>E</sub>X 対応。

その他、特定の用途に使われる文書クラスもある。

**beamer** L<sup>A</sup>T<sub>E</sub>X でスライドを作るための文書クラス。

**standalone** 文書ではなく単体の図を出力できる文書クラス。数式や (TikZ で描いた) 図を他の文書に貼り付けたり、画像化して他のアプリケーションに貼り付けたり、という目的で使える。

<sup>\*34</sup> <https://www.w3.org/TR/jlreq/ja/>

## 4.4 文字コード

文字をコンピュータで扱うためには、「使用出来る文字」の集合と、それに属する各文字からバイト列への対応を決めておかないといけない。これを文字コードという。日本語用の文字コードは、

JIS (ISO-2022-JP), EUC-JP, Shift\_JIS, Unicode

という 4 種類の系統が広く使われる<sup>\*35</sup>。Unicode はいくつか符号化方式を定めているが、よく使われるのは文字を 1-4 バイトの可変長で符号化する **UTF-8** である。

Windows 上の  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live の  $\mathrm{pL}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  の標準設定では、「 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  ソースの文字コードを自動判別する」ようになっている<sup>\*36</sup>。自動判別が不安な場合は、明示的に `-kanji=utf8` を指定する ( $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  ソースが UTF-8 の場合) か、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  ソースを BOM 付き<sup>\*37</sup>UTF-8 で保存する。

```
> uplax -kanji=utf8 hoge.tex
```

`ptex2pdf` を使う場合は、(`-kanji=utf8` を `uplax` に渡すため) `-ot` を前置し、次のようにする：

```
> ptex2pdf -l -u -ot -kanji=utf8 hoge.tex
```

一方、 $\mathrm{LuaT}_{\mathrm{E}}\mathrm{X}$  や  $\mathrm{X}_{\mathrm{Y}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  を使う場合や、macOS や Linux で  $\mathrm{pL}_{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  を使う場合は、UTF-8 しか使えない<sup>\*38</sup>。また、`pdfTEX` を使って欧文を書く場合でも、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live 2018 以降は UTF-8 がデフォルトになっている。よって、新規に書く文書では UTF-8 を使うべきである。

## 4.5 この先は

これ以上詳しくは「 $\mathrm{pT}_{\mathrm{E}}\mathrm{X}$  系列以外による日本語文書作成」(`tex_mik.pdf`) を参照されたい。

---

<sup>\*35</sup> と言っても最近では Unicode に統一されつつある。Shift\_JIS は Windows 界隈で、ISO-2022-JP は電子メールで目にするところがあるかもしれない。

<sup>\*36</sup> `-guess-input-enc` オプション。macOS 用や Linux 用のバイナリには、そのような機能は実装されていない。

<sup>\*37</sup> 一般論として UTF-8 の「BOM」は 8 ビットクリーンなソフトウェアにおいて問題を引き起こす可能性があるのが要注意だが、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live 2018 以降の  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  の場合は UTF-8 の BOM は正常に解釈される (一方、古い  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live では `pdfTEX` が UTF-8 の BOM を認識しない)。よって、最新の  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  Live で処理させる  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  ファイルに関しては UTF-8 の BOM はあってもなくても良い。

<sup>\*38</sup> 一応  $\mathrm{X}_{\mathrm{Y}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  は BOM 付き UTF-16 を解釈するようである。

## 5 雑多な話題

### 5.1 ソースコードの組版

プログラムのソースコードの組版には、[listings パッケージ](#)を利用するのが便利である。例えば、

```
1 \documentclass[uplatex]{jsarticle}
2 \usepackage{courier}
3 \usepackage[dvipdfmx]{xcolor}
4 \usepackage{listings}
5 \lstset{
6   basicstyle=\ttfamily, % タイプライタ体の利用
7   numbers=left, numberstyle=\tiny, % 行番号の表示
8   keywordstyle=\bfseries\color{blue}, % キーワードは太字青色に
9   commentstyle=\bfseries\color{green!50!black}, % コメントは太字緑色に
10 }
11 \begin{document}
12 \begin{lstlisting}[language={90}Fortran]]
13 ! test.f90
14 program test
15   implicit none
16   integer i, j
17   do i = 1,26
18     j = 2**i
19     print *, i, (1d0+1d0/j)**j
20   end do
21 end program test
22 \end{lstlisting}
23 \end{document}
```

をタイプセットすると、

```
1  ! test.f90
2  program test
3    implicit none
4    integer i, j
5    do i = 1,26
6      j = 2**i
7      print *, i, (1d0+1d0/j)**j
8    end do
9  end program test
```

のようになり、キーワードやコメントの書体を自動的に変えることができたりする。このソースのように、タイプセット時のオプションは `\lstset` 命令の引数や `lstlisting` 環境のオプション指定で行うことができる。

TeX ソース中に直に書くのではなく、外部のソースファイルをそのまま組版したい場合は、

```
\lstinputlisting{test.f90}
\lstinputlisting[language=[90]Fortran]{test.f90}
```

などのようにすればよい。

**listings** パッケージは日本語との相性が良くないことが知られている<sup>\*39</sup>。pLaTeX・upLaTeX においては、**plistings** パッケージと併用することになる。

listings 以外のソースコード組版パッケージとしては **minted** パッケージがある。minted は Python で書かれた Pygments というライブラリを使うため、Pygments を別途インストールする必要があるほか、LaTeX 処理の際に `-shell-escape` オプションを指定する必要がある。

## 5.2 相互参照

### 5.2.1 相互参照の基本

節番号、式番号などを「第 3.1 節」などと手動でソース中に直に書いてしまうと、文章構成の変更で番号が変わった時の対応が非常に大変である。しかし、LaTeX はこれらの節番号、式番号などを自動で振ってくれる仕組みが整理されている。

相互参照は、

1. 参照したい箇所で、`\label{...}` によってラベルを割り当てる。ラベルは比較的自由な名前にできるが、管理のためには「図は `fig:` で始める」「式は `eq:` で始める」などの自分なりのルールを作っておくとわかりやすい。
2. ラベルを付けた節、式、図、……の番号を出力するには、`\ref{...}` の引数にラベルを与える。
3. `\pageref{...}` を用いると、ラベルの存在箇所のページ番号が出力される。

と、基本的には `\label` と `\ref` を対にして用いる。

`\ref{...}` では番号のみ出力されることには注意が必要である。数式の参照では番号を `()` でくくることが多いが、その際は `(\ref{...})` とする必要がある。**amsmath** パッケージに用意されている `\eqref{...}` を使えば、式番号の周囲の括弧を自動で補ってくれる。

節番号、図番号なども同様で、「節」「図」などの接頭辞・接尾辞は自分で入力しないといけない。これを解決するためのパッケージとして **cleveref** パッケージや **prettyref** パッケージなどが存在している<sup>\*40</sup>が、詳細はここでは述べず、各自調べてもらうことにする。

いつまでも一般的な事項だけを述べていては仕方がないので、ここで例を上げることにしよう。次の内容の `reftest.tex` を作る：

```
1 \documentclass[uplatex]{jsarticle}
2 \usepackage{amsmath}
3 \begin{document}
```

---

<sup>\*39</sup> LuaTeX-jā の下で listings パッケージを使用する場合は、自動的に日本語対応強化用のパッチを適用するので特に何もなくても良い。

<sup>\*40</sup> 「TeX Live のインストール」中の設定では導入されないようである。

```

4
5 第\ref{sec:hoge}節にある2次方程式(\ref{eq:quad})の解は
6 \begin{equation}
7   x=\frac{-3\pm \sqrt{5}}{2}\label{eq:sol}である
8 \end{equation}
9
10 \section{ほげほげ}\label{sec:hoge}
11
12 \begin{equation}
13   x^2+3x+1=0 \label{eq:quad}
14 \end{equation}
15 の解は、\pageref{eq:sol}ページ中の
16 \eqref{eq:sol}である。% 要amsmath
17 \end{document}

```

いつものように

```
> uplatex reftest.tex
```

とタイプセットすると、

```

LaTeX Warning: Reference `eq:quad' on page 1 undefined on input line 5.
LaTeX Warning: There were undefined references.
LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.

```

のような警告メッセージが表示される。

```
> dvipdfmx reftest
```

によってPDFを作成し、そのPDFを見ると

第??節にある2次方程式(??)の解は

のように、番号が入るべきところが??と伏字になっている。

このように、1回のタイプセットでは相互参照は正しく解決されない。これは相互参照の情報（どのラベルが、どの番号に対応し、何ページにある）が1回補助ファイルに書き出されるからである。もう一回

```

> uplatex reftest.tex
> dvipdfmx reftest

```

とタイプセット（とDVIPDFMx）を行えば、補助ファイルの情報が読み込まれて、

第1節にある2次方程式(2)の解は

と正しく表示される。

複数回タイプセットするのが面倒な場合は、`latexmk` (§1.6.2 を参照) や `cluttex`<sup>\*41</sup>等の自動化ツールを使うと良い。

---

<sup>\*41</sup> <https://github.com/minoki/cluttex>

### 5.2.2 showkeys パッケージ

相互参照のためにたくさん `\label` を使っていると、式や節にどのラベルを振ったかわからなくなることがよくある。 [showkeys パッケージ](#)<sup>\*42</sup>を利用すると、以下のようにラベルが文章中に出力されるので管理しやすくなる：

- `\label` を指定した場所の近くに、そのラベルを `hogehoge` のような形式で出力する。
- `\ref` や `\pageref` では、引数に指定したラベルを `hogehoge` のように右上に小さく出力する<sup>\*43</sup>。

### 5.2.3 hyperref パッケージ

また、PDF ファイルを（印刷せず）画面上で見える場合には、PDF 中に文書構造をしおりとして持っていたり、あるいは相互参照元や外部の Web ページに 1 クリックで飛べる（ハイパーリンク）と非常に便利である。これらの機能を L<sup>A</sup>T<sub>E</sub>X で提供するのが [hyperref パッケージ](#)<sup>\*44</sup>である。


`hyperref` パッケージ自体は多くの機能を持つ<sup>\*45</sup>が、本節では `\section` などの文書構造をしおりとして PDF 内に埋め込む場合の注意についてのみ述べる。

以下の内容の `hytest.tex` を作り、普通にタイプセットを 2 回、その後 PDF 生成をしてほしい：

```
1 \documentclass[uplatex]{jsarticle}
2 \usepackage[dvipdfmx]{hyperref}% dvipdfmx を使うので
3 \begin{document}
4 \section{最初の節}\label{sec:jpn}
5 ほげほげ
6 \newpage
7 現在の節は\pageref{sec:jpn}ページから始まる。
8 \subsection{first subsection}
9 \end{document}
```

すると、2 ページ目の「現在の節は 1 ページから始まる。」の「1」の部分が赤枠で囲まれ、その部分をクリックすると 1 ページ目に飛べるようになる。

さて、PDF のしおりを見ると、`\section`、`\subsection` で作った見出しが（階層構造込みで）含まれ、それぞれをクリックすると各節の最初に飛べる……のだが、1 つめのしおりが「æœ•âšă†@ç•」といったわけのわからない文字列に化けている。

 「最初の節」を UTF-8 で符号化すると E6 9C 80 E5 88 9D E3 81 AE E7 AF 80 となる。これをバイトごとに区切り、各バイトを PDFDocEncoding というエンコーディングで読んだものがこの「æœ•âšă†@ç•」である。

この症状を治すには、八登崇之氏による [pxjahyper パッケージ](#)<sup>\*46</sup>を、`hyperref` パッケージの後に読み込む。

<sup>\*42</sup> マニュアルは実習用マシンで `texdoc showkeys` を実行することでも参照できる。

<sup>\*43</sup> 本文と多少かぶっているが、これは仕様である。

<sup>\*44</sup> マニュアルは実習用マシンで `texdoc hyperref` を実行することでも参照できる。

<sup>\*45</sup> いろいろなパッケージと干渉することがあるので、できるだけプリアンブルの後ろの方で読み込んだほうが良い。

<sup>\*46</sup> マニュアルは実習用マシンで `texdoc pxjahyper` を実行することでも参照できる。

というわけで、hytest.tex のプリアンブルを

```
\documentclass[uplatex]{jsarticle}
\usepackage[dvipdfmx]{hyperref}
\usepackage{pxjahyper}
\begin{document}
```

のように変更し、タイプセットを2回、PDF生成を行えば、しおりも「最初の節」と化けないPDFが作成できる。

## 5.3 画像

**T<sub>E</sub>X** は、本来画像の挿入や色の設定の機能を持たない。DVI ファイルを扱う各ソフトウェアが、画像などを扱うように独自の方法で対応しているに過ぎないのである。そのため、画像や色を **T<sub>E</sub>X** で扱う際には、「どのソフトで成果物を確認するか」ということを常に意識しなければならない。

本節では、§1 のような「DVI ファイルを DVIPDFM<sub>x</sub> で常に PDF ファイルに変換し、それを表示や印刷などに使う」という方針の下で画像を挿入する方法を概略だけ説明する。詳しくは

- [T<sub>E</sub>X Wiki](#) 中の「**T<sub>E</sub>X** 入門 / 図表」
- 「[はいばーワークブック](#)」中の「27.11 グラフィックス」

を参照して欲しい。

以下、「ドキュメント」の中で作業することにする。まず、「ドキュメント」に、例で使う2つの画像ファイル [fig1.png](#) と [fig2.pdf](#) をダウンロードしておく。

次に、以下の内容を持った `grapheg.tex` を「ドキュメント」に作る。

```
1 \documentclass[uplatex]{jsarticle}
2 \usepackage[dvipdfmx]{graphicx}
3 \begin{document}
4 \begin{figure}
5   \centering
6   \includegraphics{fig1.png}
7   \caption{意味のないPNG形式の図}
8   \label{fig:png}
9 \end{figure}
10 \begin{figure}
11   \centering
12   \includegraphics{fig2.pdf}
13   \caption{意味のないPDF形式の図}
14   \label{fig:pdf}
15 \end{figure}
16 これでこの文書中には図\ref{fig:png}と
17 図\ref{fig:pdf}が挿入される。
18 \end{document}
```

これをいつも通りの方法でタイプセットすれば、画像が含まれた PDF ファイルが生成される。

```
> uplatex grapheg.tex
```

```
> uplatex grapheg.tex
> dvipdfmx grapheg.dvi
```

## 5.4 図式

数学では,

$$\begin{array}{ccccc}
 & & C & & \\
 & \swarrow f & \downarrow (f,g) & \searrow g & \\
 A & \xleftarrow{\pi_1} & A \times B & \xrightarrow{\pi_2} & B
 \end{array}$$

のような図式を使う (上は直積の universality を表したものである). このような図式を記述するには `tikzcd` というパッケージを使うのが便利である. 詳しい使い方は, `texdoc tikzcd` を参照して欲しい.

この図式の出力は, プリアンブルに

```
\documentclass[uplatex,dvipdfmx]{jsarticle}
\usepackage{tikz-cd}
```

と入力することで `tikzcd` を読み込んだ上で,

```
\[
\begin{tikzcd}
& C \ar[ld,bend right=15,"f"] \\
& \ar[rd,bend left=15,"g"] \\
& \ar[d,dashed,"{(f,g)}"] & \& \& \\
A \& A \times B \ar[l,"\pi_1"] \ar[r,"\pi_2"] & B
\end{tikzcd}
\]
```

というソースで行った.

図式中に登場する  $A, B$  などのオブジェクトは, `tabular`, `array` 環境などのように表の形で入力する. `&` が列の区切り, `\&` が行の区切りである. 図中  $C$  は第 1 行第 1 列……のように見えるが, 第 2 行第 2 列の  $A \times B$  の真上にくるようにタイプセットしたいので, ソース中では第 1 行第 2 列の位置に指定する.

`\ar` (または `\arrow`) で始まるのが図式中の矢印である. 矢印の終点やラベルの内容は, `\ar` の (角かっこ) 引数として指定する.

- 矢印の終点は, 始点からの相対位置で指定する. 例えば, `\ar[llu]` ならば, 相対的に 2 つ左 (left), 1 つ上 (up) のオブジェクトに向かって矢印を書くことになる.
- 矢印には上下にラベルを付けられる. ラベルは二重引用符 " で囲む. 追加の指定をしない場合は進行方向向かって左側に付くが, 右側に付けたい場合はアポストロフィー ' を二重引用符の直後に付ける. 上のソース中の `\ar[l,"\pi_1"]` と `\ar[r,"\pi_2"]` を比較して欲しい. 注意点として, ラベルの中身がコンマを含む場合は, ラベルを波括弧で囲う必要がある.
- 線種の指定もできる. 通常の矢の他に可換図式で良く使うものとしては

$$A \xlongequal{\quad} B \dashrightarrow C \twoheadrightarrow D \rightharpoonrightarrow E$$



などがあるが、これらは

```
\[
\begin{tikzcd}
A \ar[r,equal]
& B \ar[r,dashed]
& C \ar[r,twoheadrightarrow]
& D \ar[r,rightarrowtail] & E
\end{tikzcd}
\]
```

によって入力できる。

- 矢印を少し曲げたい場合には、`bend left`, `bend right` でそれぞれ進行方向左側・右側に曲がる。オプションの値として数値を指定することで、曲がり具合を調整できる（小さい方がより真っ直ぐに近い）。

■課題 12 次の図式（five lemma の図）をタイプセットせよ。

$$\begin{array}{ccccccccc}
 A & \xrightarrow{f} & B & \xrightarrow{g} & C & \xrightarrow{h} & D & \xrightarrow{i} & E \\
 \alpha \downarrow & & \beta \downarrow & & \gamma \downarrow & & \delta \downarrow & & \epsilon \downarrow \\
 A' & \xrightarrow{f'} & B' & \xrightarrow{g'} & C' & \xrightarrow{h'} & D' & \xrightarrow{i'} & E'
 \end{array}$$

■課題 13 次の各図式をタイプセットせよ。

$$\begin{array}{ccccc}
 1 & \xrightarrow{a} & A & \xrightarrow{f} & A \\
 & \searrow z & \downarrow x & & \downarrow x \\
 & & N & \xrightarrow{s} & N
 \end{array}$$

$$\begin{array}{ccccc}
 X & \xrightleftharpoons[f]{f} & Y & \xrightarrow{q} & Q' \\
 & & \searrow e & & \downarrow e \\
 & & & & A
 \end{array}$$

■課題 14 次の図式をタイプセットせよ。

$$\begin{array}{ccccccc}
 & & & & Y' & \longrightarrow & Y \\
 & & & & \downarrow e & & \downarrow p \\
 Z' \times_C D & \longrightarrow & Z' & \longrightarrow & Y' & \twoheadrightarrow & e(Y') \\
 \downarrow z & & \downarrow \text{p.b.} & & \downarrow m & \nearrow c & \downarrow \\
 D & \longrightarrow & C & \longrightarrow & B' & \twoheadrightarrow & X' \\
 \downarrow p & & & & \downarrow f' & & \downarrow f \\
 B & \longrightarrow & & & A' & \twoheadrightarrow & A
 \end{array}$$

## 5.5 数式クラス

§2.1 では数式入力の際に気をつけることの例を挙げた。T<sub>E</sub>X での数式組版についてもっと体系的に知るためには、数式クラスの知識が不可欠である。そこで、ここで数式クラスについての最低限の解説を行っておく。

### 5.5.1 数式クラスとは

T<sub>E</sub>X における数式中の要素には、その役割に応じて数式クラスが割り当てられる。数式クラスは大雑把に言うと以下の 8 つである。

- Ord** 通常記号 (ordinary). アルファベットや  $\infty$ ,  $\emptyset$  などの単体でオペランドとなる記号や、 $\partial$  や  $-$  等の、中置演算子として使用されない前置演算子が該当する。
- Op** 作用素 (large operator). `\sum` や `\int` が代表的。 `\sin` や `\lim` もこれに該当する。 `\limits` 指定により、添字を上下に配置できる。
- Bin** 二項演算子 (binary operation).  $+$  や  $-$ . 単項演算子として使用された場合（後述）は Ord に変化する。
- Rel** 関係演算子 (relation). イコール  $=$ , 不等号  $<$ ,  $>$ , `\subset`, コロン  $:$  など。二項演算子と比べて、前後の空白の量が多い。関係演算子が並んだ場合には、間に空白は入らない（例： $:=$  と書いた場合にコロンとイコールの間に空白は入らない）。
- Open** 開き括弧 (opening symbol).
- Close** 閉じ括弧 (closing symbol). 後置演算子にも使用される。
- Punct** 句読点 (punctuation). コンマ、や `\colon` など。後ろ側に若干空白が入る。
- Inner** 内部数式 (inner formula). `\left<デリミタ><中身>\right<デリミタ>` や `<分子>\over<分母>` など。

特定の要素の数式クラスを変更したい場合、または特定の数式クラスを持つ空の要素を作り出した場合は、`\math***` 系コマンド<sup>\*47</sup>を使えば良い。また、要素を波括弧で囲うことにより、数式クラスが Ord になる。

二項演算子 (Bin) に分類される記号のうち、 $-$  や  $\pm$  などは前置の単項演算子としての用法もある。T<sub>E</sub>X は、Bin 要素が以下に該当する場合、単項演算子として使用されたと判断し、その要素の数式クラスを Bin から Ord へ変更する。

- 数式リストの先頭または末尾に置かれている（例： $-a$ ）
- 直前の要素が Op, Bin, Rel, Open, Punct の場合（例： $a = -b$ ,  $(-a)$ ,  $(a, -b)$ ）
- 直後の要素が Rel, Close, Punct の場合

### 5.5.2 数式クラスと要素間の空白

数式中の要素間の空白は、数式クラスに応じて決定される。具体的には、次の表によって定まる：

---

<sup>\*47</sup> `\mathord`, `\mathop`, `\mathbin`, `\mathrel`, `\mathopen`, `\mathclose`, `\mathpunct`, `\mathinner`

		後							
		Ord	Op	Bin	Rel	Open	Close	Punct	Inner
前	Ord	0	thin	(med)	(thick)	0	0	0	(thin)
	Op	thin	thin	*	(thick)	0	0	0	(thin)
	Bin	(med)	(med)	*	*	(med)	*	*	(med)
	Rel	(thick)	(thick)	*	0	(thick)	0	0	(thick)
	Open	0	0	*	0	0	0	0	0
	Close	0	thin	(med)	(thick)	0	0	0	(thin)
	Punct	(thin)	(thin)	*	(thin)	(thin)	(thin)	(thin)	(thin)
	Inner	(thin)	thin	(med)	(thick)	(thin)	0	(thin)	(thin)

ただし,

- 0 : 空白なし
- 括弧 : スクリプトスタイルでは空白なし
- \* : この場合は Bin が単項演算子として使用されたと判断され, あらかじめクラスが Ord に変更されている (ので考慮しなくて良い)

を意味する.

数式クラスによって自動的に挿入される空白の量は thin, medium, thick の 3 種類である. これらの空白を手動で挿入するコマンドは, 次の通りである :

	挿入用コマンド	量
thin	<code>\,, \thinspace<sup>*48</sup></code>	$\Rightarrow \Leftarrow$
medium	<code>\:, \&gt;<sup>*49</sup>, \medspace<sup>*50</sup></code>	$\Rightarrow \Leftarrow$
thick	<code>\;, \thickspace<sup>*50</sup></code>	$\Rightarrow \Leftarrow$

### 5.5.3 各種記号の数式クラス

代表的な数学記号について, どのような数式クラスが割り当てられるか確認しておこう.

**Ord** ラテンアルファベット, ギリシャ文字, ピリオド ., スラッシュ /, 縦棒 |, `\partial`, `\infty`  $\infty$ , `\emptyset`  $\emptyset$ , `\forall`  $\forall$ , `\exists`  $\exists$ , `\neg`  $(\neg)$ , `\backslash`  $\backslash$ , `\nabla`  $\nabla$

**Op** `\int`  $\int$ , `\prod`  $\prod$ , `\sum`  $\sum$ , `\log`  $\log$ , `\sin`  $\sin$ , `\lim`  $\lim$

**Bin** `*`  $*$ , `+`  $+$ , `-`  $-$ , `\pm`  $\pm$ , `\circ`  $\circ$ , `\times`  $\times$ , `\cdot`  $\cdot$ , `\otimes`  $\otimes$ , `\dagger`  $\dagger$

**Rel** コロン  $:$ ,  $=$ ,  $<$ ,  $>$ , `\mid`  $|$ , `\rightarrow`  $(\rightarrow)$ , `\sim`  $\sim$

**Open**  $($ ,  $[$ , `\{`  $\{$ , `\angle`  $\langle$

**Close**  $)$ ,  $]$ , `\}`  $\}$ , `\rangle`  $\rangle$ ,  $!$ ,  $?$

**Punct** コンマ  $,$ , セミコロン  $;$ , `\colon`  $:$ , `\cdots`  $\cdots$

見た目が同じ記号でも, 役割によってコマンドが分かれていることがある.

<sup>\*48</sup> `amsmath` 使用時. `amsmath` を読み込まない状態では, `\thinspace` は `\,` とは異なる定義が使われる.

<sup>\*49</sup> `tabbing` 環境の中では `\>` は使えないことに注意.

<sup>\*50</sup> `amsmath` 使用時.

	Ord	Rel	Open	Close	Punct
縦棒	, \vert	\mid	\lvert	\rvert	
コロン		:			\colon
\big 系	\big	\bigm	\bigl	\bigr	

#### 5.5.4 数式クラスで読み解く $\text{\LaTeX}$ の作法

数式クラスについて学んだので、§2.1 の例を数式クラスを踏まえて再考してみよう。

■複数文字からなる関数 例 4 について、 $y\mathrm{sin} x$  と書いた場合は Ord–Ord–Ord として扱われる。 $y\sin x$  と書いた場合は Ord–Op–Ord として扱われ、 $\sin$  の前後に thin space が挿入される。

$$\begin{array}{llll} \text{誤:} & \$y\mathrm{sin} x\$ & y\sin x & \begin{array}{c} \boxed{y} \boxed{\sin} \boxed{x} \\ \text{Ord Ord Ord} \end{array} \\ \text{正:} & \$y\sin x\$ & y \sin x & \begin{array}{c} \boxed{y} \boxed{\sin} \boxed{x} \\ \text{Ord Op Ord} \end{array} \end{array}$$

$f/\mathrm{gcd}(f,g)$  と書いた場合のスラッシュと  $\mathrm{gcd}$ 。スラッシュは Ord なので、Ord–Op という並びになり、thin space が挿入される。 $\mathrm{gcd}(f,g)$  を  $\{ \}$  で囲えば、囲った部分が Ord 扱いになり、空白は挿入されなくなる。

$$\begin{array}{llll} \text{誤:} & \$f/\mathrm{gcd}(f,g)\$ & f/\mathrm{gcd}(f,g) & \begin{array}{c} \boxed{f} \boxed{/} \boxed{\mathrm{gcd}} \boxed{(} \cdots \boxed{)} \\ \text{Ord Ord Op Open Close} \end{array} \\ \text{正:} & \$f/{\mathrm{gcd}(f,g)}\$ & f/\mathrm{gcd}(f,g) & \begin{array}{c} \boxed{f} \boxed{/} \boxed{\mathrm{gcd}} \boxed{(} \cdots \boxed{)} \\ \text{Ord Ord Op Open Close} \\ \text{Ord} \end{array} \end{array}$$

$\operatorname{id} \circ f$  と書いた場合の数式クラスは Op–Bin–Ord となるが、Op の直後の Bin は Ord になるので、 $\circ$  と  $f$  の間には空白は入らない。この場合の  $\operatorname{id}$  は単体でオペランドとして使用されているので、クラスが Ord となるように  $\{ \}$  で囲うかそもそも  $\mathrm{sin}$  等を使うべきである。

$$\begin{array}{llll} \text{誤:} & \$\operatorname{id} \circ f\$ & \operatorname{id} \circ f & \begin{array}{c} \boxed{\operatorname{id}} \boxed{\circ} \boxed{f} \\ \text{Op Bin Ord} \end{array} \\ \text{正:} & \$\{\operatorname{id}\} \circ f\$ & \operatorname{id} \circ f & \begin{array}{c} \boxed{\operatorname{id}} \boxed{\circ} \boxed{f} \\ \text{Ord Bin Ord} \end{array} \\ \text{正:} & \$\mathrm{sin} \operatorname{id} \circ f\$ & \operatorname{id} \circ f & \begin{array}{c} \boxed{\operatorname{id}} \boxed{\circ} \boxed{f} \\ \text{Ord Bin Ord} \end{array} \end{array}$$

■ $\left/ \right.$  と関数記号  $f\left(x\right)$  と書いた時に  $f$  と開き括弧との間に空白が入るのは、 $\left(x\right)$  が Inner として扱われ、Ord–Inner 間に thin space が入るからである。一方、 $f\bigl(x\bigr)$  と書いた場合は Ord–Open–Ord–Close となるので  $f$  と開き括弧の間に空白は入らない。

誤:	$f\left(\frac{x}{y}\right)$	$f\left(\frac{x}{y}\right)$	$f\left(\frac{x}{y}\right)$	Ord Inner
正:	$f\bigl(\frac{x}{y}\bigr)$	$f\left(\frac{x}{y}\right)$	$f\left(\frac{x}{y}\right)$	OrdOpen Close
正:	$f\mathopen{}\left(\frac{x}{y}\right)$	$f\left(\frac{x}{y}\right)$	$f\left(\frac{x}{y}\right)$	OrdOpen Inner
正:	$f\mleft(\frac{x}{y}\mright)$	$f\left(\frac{x}{y}\right)$	$f\left(\frac{x}{y}\right)$	OrdOpen Close

$\backslashleft/\backslashright$  を使いつつ  $f$  と開き括弧の間に空白を入れないようにするには、

$f\mathopen{}\left(x\right)$  という風に、空の  $\mathopen{}{\backslashmathopen{}}$  を挟めば良い。こうすると要素の並びは Ord–Open–Inner となり、空白は入らない。mleftright パッケージの  $\mleft/\mright$  はこのような「 $\backslashleft/\backslashright$  を補正する」処理をより洗練させた形で提供している。具体的には、後ろにコンマ等が続いた場合も考慮している。

■絶対値記号 (例 8)  $|-x|$  と書いた場合、 $|$  は Ord なので Ord–Bin–Ord–Ord となる。Bin の両側に Ord–Bin 間の空白が入る。 $|\{-x\}|$  と書いた場合は、外側は Ord–Ord–Ord となる。内部の  $-x$  の部分は Bin–Ord だが、数式リストの最初にある Bin は (単項演算子として使用されたと判断され) Ord に変換されるので、Bin–Ord 間の空白は入らない。

誤:	$ -x $	$ -x $	$ -x $	Ord Bin OrdOrd
正:	$ \{-x\} $	$ -x $	$ \{-x\} $	Ord BinOrd Ord
正:	$\lvert -x \rvert$	$ -x $	$\lvert -x \rvert$	OpenBinOrdClose

$\lvert -x \rvert$  と書いた場合、Open–Bin–Ord–Close となる。Open 直後の Bin は (単項演算子として使用されたと判断され) Ord に変換されるので、Bin–Ord 間の空白は入らない。

■逆向きの括弧 (例 9)  $x \in ]-a, b]$  の要素の並びは、Ord–Rel–Close–Bin–Ord–Punct–Close となる。開き括弧として使われている最初の  $]$  の数式クラスが Close となっているために直前の  $\in$  との間隔が詰まっているほか、直後の  $-a$  のマイナス記号が二項演算子扱いとなってしまう。

誤:	$x \in ]-a, b]$	$x \in ]-a, b]$	$x \in ]-a, b]$	Ord RelClose Bin OrdPunct OrdClose
正:	$x \in \mathopen{]}-a, b]$	$x \in ]-a, b]$	$x \in ]-a, b]$	Ord Rel OpenBinOrdPunct OrdClose
正:	$x \in \left] -a, b \right]$	$x \in ]-a, b]$	$x \in ]-a, b]$	Ord Rel BinOrdPunct Ord Inner

$x \in \mathopen{]}-a, b]$  は、Ord–Rel–Open–Bin–Ord–Punct–Ord–Close となる。マイナス記号は Open の直後なので単項演算子扱いとなる。

$x \in \left] -a, b \right]$  は、全体としては Ord–Rel–Inner,  $\left] -a, b \right]$  の内側は Bin–Ord–Punct–Ord となる。マイナス記号は数式リストの先頭なので単項演算子扱いとなる。

■集合の縦棒 (例 13) すでに述べたように、集合の縦棒には、関係演算子扱いの縦棒またはコロン、つまり `\mid` や `:` を使う。

誤: $\{x x>5\}$	1 誤: $\{\$ \{x x>5\}\$$
正: $\{x \mid x>5\}$	2 正: $\{\$ \{x\mid x>5\}\$$
誤: $\{x: x >5\}$	3 誤: $\{\$ \{x\colon \lvert x\rvert>5\}\$$
正: $\{x: x >5\}$	4 正: $\{\$ \{x:\lvert x\rvert>5\}\$$

さて、集合の中身が縦に長いと、`\left / \right` によって括弧を自動的に伸長させたい。その際に縦棒も一緒に伸長させるには、`\middle` を使えば良さそうに思える。

誤: $\left\{\frac{x}{y}\middle y\neq 0\right\}$	1 誤: $\left\{\$ \{\dfrac{x}{y}\middle y\neq 0\right\}\$$
--	--

しかし、単に `\middle` を使うのでは縦棒の前後に適切な空白 (Rel 前後の空白) が入らない。こういう場合は、ダミーの Rel を入れてやるとうまくいく。

誤: $\left\{\frac{x}{y}\middle y\neq 0\right\}$	1 誤: $\left\{\$ \{\dfrac{x}{y}\middle y\neq 0\right\}\$$
正: $\left\{\frac{x}{y}\middle y\neq 0\right\}$	2 正: $\left\{\$ \{\dfrac{x}{y}\right.$
(比較: $\left\{\frac{x}{y}\middle y\neq 0\right\}$ )	3 $\left.\mathrel{\middle }\mathrel{\middle }y\neq 0\right\}\$$
	4 $\left.\mathrel{\middle }y\neq 0\right\}\$$
	5 (比較: $\{\$ \{\dfrac{x}{y}\mid y\neq 0\}\$$ )

参考: 八登氏のブログ記事「[\middle であり \mid であるもの](#)」

■同値関係で割る 集合を同値関係で割る場合に  $X/\sim$  と書くと、数式クラスが Ord-Ord-Rel となり、スラッシュとニョロの間に空白が入ってしまうほか、直後の関係演算子とニョロの間が詰まってしまう。 $X/\{\sim\}$  と書けば、数式クラスは Ord-Ord-Ord となり、スラッシュとニョロの間に余計な空白は入らず、直後の関係演算子との間に適切な空白が入る。

誤: $X/\sim=\{[x]\mid x\in X\}$	1 誤: $\$X/\sim=\{[x]\mid x\in X\}\$$
正: $X/{\sim}=\{[x]\mid x\in X\}$	2 正: $\$X/{\sim}=\{[x]\mid x\in X\}\$$

■整除関係の縦棒 整数や多項式などの「割り切る」関係 (整除関係) には、縦棒がよく使われる。LaTeX での縦棒としては `|` と `\mid` があるが、どちらを使うべきだろうか?

この縦棒は関係演算子であることを考えると、数式クラスが Rel である `\mid` の方を使うべきである。数式の「意味」を意識するなら、自分で `\newcommand{\divides}{\mid}` と定義して `\divides` を使うと良いだろう。

参考: LaTeX での「割り切る」記号 | 雑記帳

### 5.5.5 もっと詳しく知りたい方へ

こういう話に興味があったら「数式組版」や *The TeXbook* を読むと良い。

- ・ 木枝祐介, 「数式組版」, ラムダノート, 2018.
- ・ Donald E. Knuth, *The TeXbook*, Addison-Wesley, 1984.

## 5.6 p<sub>T</sub>E<sub>X</sub> 系列で縦書き等をする場合に発生する問題とその対処

欧米で開発されているパッケージは日本語組版の事情，特に縦書きを考慮していない場合が多い．例えば，array パッケージと，p<sub>L</sub>A<sub>T</sub>E<sub>X</sub> の p<sub>l</sub>ex<sub>t</sub> パッケージは，どちらも L<sub>A</sub>T<sub>E</sub>X 標準の tabular 環境を拡張しており，お互いに相容れない．他にも p<sub>L</sub>A<sub>T</sub>E<sub>X</sub> で問題が起こりうるパッケージが多数あり，それらに対して対策をするパッケージが開発されている．例えば，array パッケージ + p<sub>l</sub>ex<sub>t</sub> パッケージの問題に対しては p<sub>l</sub>ex<sub>t</sub>array パッケージ，という具合である．

ただ，問題が起こりうるパッケージに対していちいち対策パッケージを調べて読み込むのは面倒である．この手間を省くために山下弘展氏が製作しているのが [plautopatch パッケージ](#) である．

- 山下弘展「日本語の L<sub>A</sub>T<sub>E</sub>X で幸せになる，かもしれない方法」, TeXConf 2018 <https://aminophen.github.io/slide/hytexconf18.pdf>

## 5.7 難しい小ネタ

■課題 15 以下の内容のソースファイルを `twice.tex` として保存し、タイプセットせよ。そして、なぜそのような出力結果が得られるのかを説明せよ。

```
1 \let\Enddoc\fi
2 \iffalse
3
4 あいうえお
5
6 \Enddoc
7 \documentclass{jsarticle}
8 \begin{document}
9
10 かきくけこ
11
12 \def\Enddoc{\end{document}}
13 \makeatletter
14 \let\@@let=\let
15 \def\let#1#2\iffalse{%
16   \global\@@let\let\@@let}
17 \@@input \jobname.tex
```

■課題 16 (幻の “This can’t happen” を拝もう)

T<sub>E</sub>X - L<sup>A</sup>T<sub>E</sub>X Stack Exchange 中の [Shortest code causing “Emergency stop.” error](#) という投稿によれば、以下の 1 行からなる T<sub>E</sub>X ソース：

```
1 \halign{###\cr\multispan{300}\cr}
```

が “This can’t happen” を引き起こすそうだが、残念ながら

```
1 \documentclass{article}
2 \begin{document}
3 \halign{###\cr\multispan{300}\cr}
4 \end{document}
```

を筆者の環境の pL<sup>A</sup>T<sub>E</sub>X でタイプセットしてもそうはならなかった。

しかし、「pT<sub>E</sub>X・pL<sup>A</sup>T<sub>E</sub>X はバグが少ない、やったね！」とは決して言えない。pT<sub>E</sub>X の日本語対応には不完全なところがまだ残っており、短い T<sub>E</sub>X ソース（本質的には 1 行）で、pL<sup>A</sup>T<sub>E</sub>X でタイプセットすると

```
! This can't happen (disc4).
```

というエラーを発生させるものが存在する。さて、誰か自力でそれを発見できないだろうか？