

T_EX 実習

2018 年 4 月 26 日

本文章は、T_EX 実習の実際の中身である。行うためには実習用マシンに T_EX 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「[T_EX 実習 \(2018\)](#)」のページ内にある「T_EX Live のインストール」を参照すること。

T_EX 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 T_EX を全く触ったことがない人は §1 へ。
- 「T_EX・L^AT_EX 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。
- 日本語で T_EX 環境を書く際の文字コードに関する注意や、upT_EX の話題は §4 へ。
- 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 pL^AT_EX は知っているが、pL^AT_EX が利用できない場合への対処法や、最近の T_EX の発展について知りたい人は実習資料集にある「pT_EX 系列以外による日本語文書作成」を参照すること。
- バリバリ使いこなしていて、以上の範囲などとっ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^{*1}は文書内リンクである。

T_EX 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: 2018/04/26 14:05:24.

^{*1} 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

1	pL^AT_EX はじめの一步	3
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
2	L^AT_EX の作法	10
2.1	数式	11
2.2	地の文	14
2.3	文章の構造	15
3	§1 と §2 に関連した課題	19
3.1	T _E X ソースの例	19
3.2	数学の文章を T _E X で書く練習	21
3.3	T _E X ソースの修正練習	22
4	文字コードと upT_EX	23
4.1	文字コード	23
4.2	pT _E X で扱える文字の範囲	23
4.3	Windows の機種依存文字を pT _E X でそのまま入力すると……	25
4.4	upT _E X	26
5	雑多な話題	29
5.1	ソースコードの組版	29
5.2	相互参照	30
5.3	画像	33
5.4	図式	34
5.5	難しい小ネタ	37

1 pL^AT_EX はじめの一步

日本人が普通「T_EX を用いて文書を書く」という場合、実際に使っているのは pL^AT_EX である。これらは、

- Knuth 氏による、もともとの（プログラムとしての）T_EX
- Leslie Lamport 氏による、L^AT_EX という T_EX のマクロパッケージ

を、アスキー^{*2}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 pT_EX (publishing T_EX) と呼ばれ、L^AT_EX を日本語対応させたマクロパッケージが pL^AT_EX である。日本語では横書きの他に縦書きも行われるが、pT_EX・pL^AT_EX は縦組みの組版もサポートしている。

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 T_EX が理解できるようなテキストファイルを T_EX ソースという。

```
1 \documentclass{jsarticle}
2 \begin{document}
3 ようこそ、\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 T_EX 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。
- 本実習では関係ないが、OSX や最近の多くの Linux など、Unicode (UTF-8) を用いて T_EX ソースを作成する環境下では、upL^AT_EX の使用を検討すべきである。§4.4 を参照。

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 pL^AT_EX 2_ε 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。

^{*2} 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。

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> platex first.tex
```

と入力^{*3}すると、

```
This is e-pTeX, Version 3.14159265-p3.5-130605-2.6 (sjis) (TeX Live 2016
/W32TeX) (preloaded format=platex)
restricted \write18 enabled.
entering extended mode
..... (中略) .....
Output written on first.dvi (1 page, 828 bytes).
Transcript written on first.log.
```

のようなメッセージが出て^{*4}、「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 \TeX 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。

\TeX 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^{*5}（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^{*5}。

本実習においても、`DVIPDFMx` 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]
7531 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 \TeX システムならば、`ptex2pdf` という Lua スクリプトを使うことで、 \LaTeX によるタイプセットとその後の `DVIPDFMx` の処理をまとめて

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

で行うことができる^{*6}。

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

^{*3} 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。

^{*4} 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。

^{*5} 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。

^{*6} Lua を別途インストールする必要はない。 \TeX Live に含まれている `Lua \TeX` というプログラムがその役割を果たすため。

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。

```
> platex -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_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_の世界へ！ (¥T e X 全て全角文字)
- ようこそ、\TeX_の世界へ！ (半角スペースを省略)
- ようこそ、\TEX_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

さて、§1.2 で作った PDF ファイル `first.pdf` には、欧文文字用のフォントは埋め込まれているが、和文文字を表示する際に使う日本語フォントは埋め込まれていない。この `first.pdf` を Adobe Reader や SumatraPDF 等で表示させる際には、ソフト側が適当な日本語フォントを代わりに用いて日本語部分を表示することになる。だが、この「日本語フォントは埋め込まず、表示の際には適当にシステム内の別のフォントで代替して表示する」というアプローチが不都合な場合があり、そのような場合には日本語フォントも PDF 内に埋め込むことになる^{*7}。

ここでは、 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ Live に同梱されている IPA フォント ([公式サイト](#)) を PDF 内に埋め込むことにする。それには、コマンドプロンプトにおいて、

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

または

```
C:\Users\...\> updmap --setoption kanjiEmbed ipa
```

と入力すればよく、以降の $\mathrm{DVIPDFM}x$ 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。

Mac OS X 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^{*8}。これを使いたい場合には、シンボリックリンクを貼るという前処理（ここでは解説しない）をした後に、

```
$ kanji-config-updmap hiragino-pron
```

または

```
$ updmap --setoption kanjiEmbed hiragino-pron
```

を端末で実行すれば良い。

なお、`kanji-config-updmap` や `updmap` 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、`kanji-config-updmap-sys` や `updmap-sys` のように `-sys` をつけたコマンドを使う。

^{*7} 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。

^{*8} 「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。

1.5 この先は

ひとまず p \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 \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 \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「 \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. \LaTeX 」
- 渡辺徹, 「好き好き \LaTeX 2 ϵ 初級編」, 2004.

<http://mytexpert.sourceforge.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「 \TeX 」には多数のパッケージの使用例がある。
- Scott Pakin, *The Comprehensive \LaTeX Symbol List*.

<http://www.ctan.org/pkg/comprehensive>

または C:\texlive\2016\texmf-dist\doc\latex\comprehensive\symbols-a4.pdf

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

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、 \TeX Wiki の「 \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、 \TeX ・ \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 SyncTeX

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 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、少し前には“source special”という機構が使われてきたが、最近の pTeX などには **SyncTeX** という機能が備わっており、そちらの方がより洗練されている。

SyncTeX の機能を実習用マシンで使うには、以下の設定を行う。

TeX 側の指定 pLaTeX 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」^{*9}に次のように入力する^{*10}。

```
"C:\Program Files\EmEditor\Emeditor.exe" %f /1 %l
```

この設定を行った後で、SyncTeX 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 TeX ソースの対応する箇所へとジャンプする^{*11}。

1.6.2 latexmk

PDF 作成のために毎回タイプセット・DVIPDFMx を実行するのは面倒である^{*12}。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 pLaTeX・DVIPDFMx 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「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^{*13*14} 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。

^{*9} 先に上記の「TeX 側の設定」を行わないと表示されない。

^{*10} 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。

^{*11} エディタ側を設定すれば、開いている 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。

^{*12} 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。

^{*13} 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。

^{*14} tlmgr(TeX Live Manager) は TeX Live のパッケージ管理ツール。

- tlmgr 自身のアップデートが要求された場合は、

```
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 L^AT_EX 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 T_EX ソースを処理できない。これを解決するには、C:\Users\...\latexmkrc^{*15*16} に以下の内容を保存する：

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

- [T_EX 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

T_EX 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、T_EXworks などの統合開発環境（又は Emacs や Vim などの高機能テキストエディタ）を用いると良い。例えば T_EXworks^{*17}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、L^AT_EX 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。

^{*15} ... 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。

^{*16} 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。

^{*17} スタートメニューから「TeXworks editor」を選択すると起動できる

2 L^AT_EX の作法

L^AT_EX 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^AT_EX には L^AT_EX 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^AT_EX 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_EX 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_EX 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^AT_EX 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」ごとの一斉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^AT_EX 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
- 小田忠雄, 「数学の常識・非常識—由緒正しい T_EX 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_EX の間違い」, T_EX ユーザの集い 2012. <http://oku.edu.mie-u.ac.jp/texconf12/presentations/saito.pdf>
- Donald E. Knuth, *The T_EXbook*, Addison-Wesley, 1984.
L^AT_EX 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_EX 組版から印刷・製本までの工程」, T_EX 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^AT_EX 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、ということに気を付けるのが大事である。

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. f

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的に表記すること。

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 を使いたいならばプリアンブルに

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

と記述するか、あるいは `amsmath` パッケージ^{*18}を読み込んだ上で

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

と記述すれば良い^{*19}。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

^{*18} 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](#)。実習用マシンにも `C:\texlive\2016\texmf-dist\doc\latex\amslatex\math\amsl.doc.pdf` にある。

^{*19} 別行立て数式の中で添字の位置を $\operatorname{Res}_{z=a}$ ではなく $\operatorname{Res}_{z=a}$ 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}`

上の 1 行目は括弧を `\left...\right` で書いたもの、2 行目はそれと同じ大きさの括弧を（それが適切なサイズかは抜きにして）手動で組んだものである。1 行目の方が、 f と括弧の間・括弧とコンマの間にそれぞれ空白 `thin space (\,)` が余計に入っていることが分かる。

しかし、それは $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の仕様である。 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は数式中の要素をいくつかに分類し、その分類に応じて要素と要素の間に入る空白を決定するのだが、次のようになっている：

- “ f ” は「通常の文字」，“,” は「句読点」である。
- 2 行目中の “(” (`()`), “(” (`\bigl()`) は「開き括弧」，“)” (`()`), “)” (`\bigr()`) は「閉じ括弧」である。
- `\left...\right` の部分は、それ自身で「内部数式」である^{*20}。
- $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は、「普通の文字」と「開き括弧」の間・「閉じ括弧」と「句読点」の間には空白を入れないが、「普通の文字」と「内部数式」の間・「内部数式」と「句読点」の間には `thin space (\,)` を入れる。

この状況を改善するのが [mleftright パッケージ](#) である。このパッケージが提供する `\mleft`, `\mright` を `\left`, `\right` の代わりに使うと、周囲の空白が正しくなるようになっている。

$f(x), f(A_D^B)$	1 <code>\$f\mleft(x\mright), f\mleft(A^B_D\mright)\$</code>
$f(x), f(A_D^B)$	2 <code>\$f(x)</code> , <code>f\bigl(A^B_D\bigr)\$</code>

例 8 絶対値やノルムに用いる “|”, “||” には左右の区別がなく注意が必要。

誤： $ -x $	1 誤： <code>\$ -x \$</code>
正： $ -x $	2 正： <code>\$ \{-x\} \$</code>
正： $ -x $ （要 <code>amsmath</code> ）	3 正： <code>\$\lvert -x \rvert\$</code> （要 <code>amsmath</code> ）

1 行目では負号 “-” が二項演算子扱いになってしまっており、2 行目では $-x$ 全体を波括弧でグループ化することでそれを防いでいる。

例 9 「本来の意味」と異なる使い方では記号を使う場合も注意。

誤： $x \in]a, b]$	1 誤： <code>\$x\in]a,b]\$</code>
正： $x \in]a, b], x \in]a, b]$	2 正： <code>\$x\in \mathopen{]}a,b]\$</code> ,
	3 <code>\$x\in \left]a,b\right]\$</code>

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： <code>\$<a,b>\$</code>
正： $\langle a, b \rangle, \langle a, b \rangle$	2 正： <code>\$\langle a,b\rangle\$</code> , <code>\$\left<a,b\right>\$</code>

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： <code>\$f:A\rightarrow B\$</code>
正： $f : A \rightarrow B$	2 正： <code>\$f\colon A\rightarrow B\$</code>

^{*20} よって、`\left` と `\right` で囲んだ部分の途中で改行することはできない。

例 12 関係演算子として使うコロン : はそのままよい.

$\text{正 : } a : b = 1 : 2$	<pre>1 正 : \$a:b=1:2\$</pre>
-----------------------------	------------------------------

⚠ 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。

$f := a^2 - b + c^d$ $f := a^2 - b + c^d \text{ (mathtools 等要)}$	<pre>1 \$f:=a^2-b+c^d\$ 2 \$f\coloneqq a^2-b+c^d\$ (mathtools等要)</pre>
--	--

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

$\{x \mid x > 5\}$ $\{x : x > 5\}$ $\{x : x > 5\}$ $\{x \mid x > 5\}$ $\{x x > 5\} \leftarrow \text{誤}$	<pre>1 \$\{\,x\mid x>5\,\}\$ 2 \$\{\,x:x>5\,\}\$ 3 \$\{x:x>5\}\$ 4 \$\{x\mid x>5\}\$ 5 \$\{x x>5\}\$ ← 誤</pre>
---	---

The T_EXbook ではなぜか 1, 2 行目のような記法が紹介されている^{*21}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。

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。

$\text{誤 : } f(x) = x^2, g(x) = x^2 + 1$ $\text{正 : } f(x) = x^2, g(x) = x^2 + 1$	<pre>1 誤 : \$f(x) = x^2, g(x) = x^2+1\$ 2 正 : \$f(x) = x^2\$, \$g(x) = x^2+1\$</pre>
---	--

The T_EXbook 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。[pT_EX 2_ε 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_EX 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。

^{*21} `\mid` は関係演算子扱いの縦棒。

- 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． $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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	<code>AA. ccc. ddd</code>	<code>AA\@._ccc._ddd</code>
単語間空白	<code>AA. ccc. ddd</code>	<code>AA._ccc._ddd</code>

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	<code>"def 'ghi' jkl", f'</code>	<code>"def 'ghi' jkl'", \$f\$'</code>
正	<code>"def 'ghi' jkl", f'</code>	<code>``def `ghi' jkl''', \$f'\$</code>

- 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²²，`-` (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^{*22}．

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

$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ で文章を書く際には，文章の「構造」をなるべく $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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．

^{*22} 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．

```

\documentclass{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}}` 節などとすれば一応可能だが、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。

2.3.2 箇条書き

しばしば次のように強制改行 (`\\`)、中黒「`·`」、全角空白「」を使って箇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.
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Lindel"of空間である. \\
4.  $\mathcal{S} \times \mathcal{S}$ はLindel"of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^AT_EX 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^{*23}「箇条書き」を参照。

例 15 番号なし

<ul style="list-style-type: none"> • hoge • fuga 	<pre> 1 \begin{itemize} 2 \item hoge 3 \item fuga 4 \end{itemize} </pre>
--	--

例 16 番号あり

<ol style="list-style-type: none"> 1. hoge 2. fuga 	<pre> 1 \begin{enumerate} 2 \item hoge 3 \item fuga 4 \end{enumerate} </pre>
--	--

^{*23} 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。

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。

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^{*24}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{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}`などで囲って^{*25}利用する。

^{*24} L^AT_EX 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。

^{*25} 実は、これらの代わりに `\thm` や `\endthm` などと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`

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。
- 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^{*26}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}
```

を用いた書き方をすべきである。

^{*26} `\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_EX 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^{*27}。

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{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 \[
25 \quad x^n+a_1x^{n-1}+\cdots+a_n=0,\quad \text{\textit{\$a_i\$は有理数. }}
26 \]
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 \[
29 \quad a_0x^n+a_1x^{n-1}+\cdots+a_n=0.
30 \]
31 左辺は所謂原始的多項式である。
```

^{*27} 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, \dots, 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 α, β に対し、
 42 $\xi = \alpha + \beta$ が代数的の数となることを示す.
 43 α, β 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 $x^m + a_1 x^{m-1} + \dots + a_m = 0$, \label{alpha} \\
 46 $x^n + b_1 x^{n-1} + \dots + b_n = 0$ \label{beta}
 47 \end{align}
 48 が存在する.
 49
 50 次に、 $\mu = 0, 1, \dots, m-1$ と $\nu = 0, 1, \dots, n-1$ に対して、
 51 \begin{equation}
 52 $\alpha^\mu \beta^\nu$ \label{multi}
 53 \end{equation}
 54 を辞書式順序で並べたものを
 55 $\omega_1, \omega_2, \dots, \omega_l$ ($l = mn$)%
 56 と書けば、各 $i = 1, 2, \dots, l$ に対して、対応する μ_i と ν_i を用いて
 57 \[
 58 $\xi^{\omega_i} = (\alpha + \beta) \alpha^{\mu_i} \beta^{\nu_i}$
 59 $= \alpha^{\mu_i+1} \beta^{\nu_i} + \alpha^{\mu_i} \beta^{\nu_i+1}$
 60 \]
 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 ξ^{ω_i} は\eqref{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、\eqref{alpha}, \eqref{beta}より
 66 \begin{align*}
 67 $\alpha^m + a_1 \alpha^{m-1} + \dots + a_m = 0$, \\
 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 ξ^{ω_i} を $\omega_1, \omega_2, \dots, \omega_l$ 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、 $i = 1, 2, \dots, l$ と $j = 1, 2, \dots, l$ に対して
 76 有理数 c_{ij} が存在して
 77 \[
 78 $\xi^{\omega_i} = \sum_{j=1}^l c_{ij} \omega_j$
 79 \]
 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 9 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、 \TeX 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 \TeX 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。\\
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。
```

をきちんとした \TeX 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{\TeX}\$ 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。

4 文字コードと upT_EX

4.1 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。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。

T_EX Live の pL_AT_EX の標準設定では、「T_EX ソースの文字コードを自動判別する」という気の利いた機能は無効になっている^{*28}。デフォルトでは、T_EX ソースの文字コードとして期待されるものは

Windows Shift_JIS 及び JIS

その他 (Mac OS X, Linux 等) UTF-8 及び JIS

となっている。Windows で UTF-8 の T_EX ソースを扱いたい場合は、`-kanji=utf8` を指定する。

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

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

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

4.2 pT_EX で扱える文字の範囲

pL_AT_EX の背後で動いている pT_EX は、1990 年以前からある古いプログラムであり、扱える文字集合としては JIS X 0208 規格内に限定されている。漢字で言えば第 1・第 2 水準の範囲に相当しており、そこからはみ出している以下のような文字はそのままでは使えない。

「**Windows の機種依存文字**」 Windows で広く使われてきた文字コードは、Shift_JIS を拡張した **CP932** と呼ばれるものである。以下のような文字は CP932 には規定されているが、文字集合 JIS X 0208 内にはない。多くの文字が 2 箇所の符号位置に重複されていることもあり、いろいろな問題を引き起こしてきた。

- 全角丸付き数字①-⑳, 全角ローマ数字 I-X・i-x
- 全角単位記号, 例えば「㏍」「㏎」「kg」「m³」
- No.Tel 明治大正昭和㊦(株)などの一部の記号
- 彌崎鄧閒などの一部の漢字

^{*28} `-guess-input-enc` オプション。Windows 用バイナリでは、単純に設定ファイル `texmf.cnf` 中で無効にされているだけである (ちなみに、W32T_EX では標準で有効になっている)。Mac OS X 用や Linux 用のバイナリには、そもそもそのような機能は実装されていない。

「Macintosh の機種依存文字」 同様に、Mac でも [MacJapanese](#) という Shift_JIS の拡張が用いられてきた。MacJapanese 側にしかない文字は以下のようなものである。

- 全角丸付き数字①–⑳ ㉑–㉙, 全角ローマ数字 I–X · i–x
- 括弧でくくられた全角の (1)–(20), (a)–(z) など
- 全角単位記号, 例えば「㏎」「㏓」「kg」「m³」
- No.TEL ㊤ ㊦ ㊧ ㊨ ㊩ ㊪ ㊫ ㊬ ㊭ ㊮ ㊯ ㊰ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿ などの一部の記号

半角カナ「ｱｲｳ」のようないわゆる半角カナは、しばしば「電子メールでは使うべきではない」と言われている。これは、Shift_JIS やその独自拡張では表現可能である一方で、電子メールで伝統的に使われてきた JIS コードでは表現不可能であるからである。

JIS X 0213 で追加された文字 2000 年に制定された JIS X 0213 という規格では、扱える文字の数が JIS X 0208 の 6879 字から 11000 字以上へと大幅に増やされた。その中にはいくつかの非漢字と共に、**迫既海温俣窠臼土**などの JIS 第 3・第 4 水準漢字が追加されている。

JIS X 0213 にもないが、**Unicode** 中に収録されている文字 最近では、「世界中の文字を 1 つの文字コードで表そう」という理念のもとに作られた **Unicode** が普及している。**Unicode** 中には非常に多くの文字が含まれており、その中の大部分は JIS X 0213 にすら規定されていない。

なお、**Unicode** では以上に述べた文字は大体^{*29}表現可能である。**pTeX** では **Unicode** (UTF-8) で記述された **TeX** ソースもタイプセット可能であるが、**pTeX** の内部処理は従来通りの **EUC-JP**・**Shift_JIS** なので、本節に述べた上記種類の文字は依然としてそのままでは扱えない。

JIS X 0208 を超えた上記種類の文字を使いたい場合は、

- 齋藤修三郎氏による [otf パッケージ](#) を使い、**Unicode** の文字を（コード番号で）入力する
- 使うプログラムを **pTeX** から **upTeX** (§4.4) に変えたり、または **X_YTeX**, **LuaTeX** を使う（「**pTeX** 系列以外による日本語文書作成」を参照）。

から選ぶのが良いだろう。**Unicode** にすら存在しなかったり、区別されていない微妙な字形の差を気にしたい場合は、**otf** パッケージを使うことになるが、それについては省略する。

^{*29} JIS X 0213 中の一部の半濁点付き仮名や、声調記号は合成文字として表すことになる。

4.3 Windows の機種依存文字を pTeX でそのまま入力すると……

ここでは失敗例として、Windows の機種依存文字を直接含む TeX ソースを pLaTeX で処理させた
らどのようなになるかを述べることにしよう。


まず、Windows の機種依存文字を直接含む次のソース：

```
1 \documentclass{jsarticle}
2 \usepackage[T1]{fontenc}
3 \begin{document}
4 「I」, 「𐤎」, 「アイウ」
5 \end{document}
```

を UTF-8 で記述（「I」は全角ローマ数字）、pLaTeX でタイプセット^{*30}し、PDF ファイルを作ると

「âĖă」, 「ăĭĖ」, 「ïšïšïšïš」

のような出力が得られる。Windows の機種依存文字や半角カナが正しく和文文字と認識されていないことわかるだろう。

 「𐤎」は Unicode では U+5F45 で、UTF-8 では E5 BD 85 という 3 バイトで符号化される。欧文部分は
2 行目により T1 エンコーディングを使うが、それによってこのバイト列を解釈すると「ăĭĖ」となる。


また、半角カナ「ア」は EF BD B1 と符号化されるが、T1 エンコーディングでこのバイト列を解釈すると「ïš」
となる。

一方、普段 Windows 上で書いているように、Shift_JIS（上記のように正確には CP932 だが）
でソースを記述し、タイプセットすると

Windows の機種依存文字 pTeX では和文文字として認識されるので、タイプセットしたあとの
dvi ファイル中には文字情報が残る。

これにより、dviout で dvi ファイルを表示すると問題なく機種依存文字が表示できる^{*31}。

半角カナ タイプセットで、つまり dvi ファイルの段階で例えば「アイウ」が「šššš」と化ける。

 これも、Shift_JIS では半角カナ「ア」は B1 B2 B3 と符号化されるが、pTeX ではこのバイト列は欧
文文字の列とみなされ、T1 エンコーディングで組まれるからである。

しかし、DVIPDFMx により PDF に変換させると、この段階で機種依存文字は抜け落ちる。例え
ば、すぐ上で作った dvi ファイルを DVIPDFMx で処理し、PDF ファイルを作成すると

```
[1
** WARNING ** No character mapping available.
  CMap name: H
  input str: <2d>
]
char=0x9348(37704)
Tried to set a nonexistent character in a virtual font]
```

^{*30} Windows 上の TeXLive では、pLaTeX 実行時にオプションとして `-guess-input-enc` または `-kanji=utf8` を指定し
ないとうまくいかない。

^{*31} dviout を悪く言うつもりはないが、「最終成果物は DVIPDFMx で変換した PDF であるが、途中では dviout を確認
に用いる」という使い方は、あまりおすすめしない。

のようなメッセージが画面に出力される。そして、PDF ファイルを見ると、

「 \int 」, 「 $\frac{1}{2}$ 」, 「 $\int_{-\infty}^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$ 」

のように機種依存文字部分は空白になってしまう。a.dvi からきちんと機種依存文字も含んだ PDF を作成することは不可能ではないが、ここでは言及しない。

4.4 upTeX

§4.2 で書いたように、pTeX は JIS X 0208 の和文文字しかそのままでは取り扱えない。この不便な状態を解決するため、pTeX の和文処理を Unicode 化したのが田中琢爾氏による **upTeX** である。

- Unicode に収録されている漢字をそのまま TeX ソース中に書けるようになった。但し、後に説明するように基本多言語面 (BMP) 外の文字を扱うには注意が必要。
- Unicode ブロックごとに「和文扱い」「欧文扱い」を切り替えることができ、これによって欧文用 LaTeX における多言語組版がそのまま upTeX で使えるようになった^{*32}。

が主な特徴であり、詳しい解説は公式ページや、八登氏による「[upLaTeX を使おう](#)」を参照。

upLaTeX (upTeX 対応した LaTeX) で文書を書くのは、従来の pLaTeX で文書を書くのとはほとんど変わらない。以下の点に注意すればよい：

- **TeX** ソースの文字コードを **UTF-8** とすること。
Mac OS X や Linux では標準でこの設定になっていることが多いが、Windows ではまだまだ Shift_JIS が主流であるため、「保存するときに明示的に UTF-8 を指定」という作業が必要かもしれない。
- クラスファイルは **upLaTeX** 用のものを使用する。
pLaTeX で標準で用意されている jarticle, jbook, tarticle などを用いていた場合は、それぞれ先頭に u をつけた uarticle, ubook, utarticle クラスを用いる。[pLaTeX 2_ε 新ドキュメントクラス](#)を用いていた場合には `uplatex` オプションを追加するだけで良い。
- **otf** パッケージを利用する際は、**uplatex** オプションを指定する。

例えば、§1.1 「最初の例」で出したソースを upLaTeX 対応にした firstu.tex は、

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

となり、1 行目のクラスファイルだけを変えればよい。

^{*32} pTeX では「和文」「欧文」の分別は固定なので、例えばギリシャ文字を「そのまま」入力してギリシャ語をタイプセットすることはできない。

タイプセットは

```
> uplatex firstu.tex
```

でよく、以下のようなメッセージが出て firstu.dvi が作成されるはずである。

```
This is e-upTeX, Version 3.14159265-p3.5-u1.11-130605-2.6 (utf8.uptex) (TeX Live
2016/W32TeX) (preloaded format=uplatex)
restricted \write18 enabled.
entering extended mode
..... (中略) .....
Output written on firstu.dvi (1 page, 840 bytes).
Transcript written on firstu.log.
```

そこから PDF ファイルを作成する方法は以前と同じであり、

```
> dvi2pdf firstu.dvi
```

とすればよい。§2.2 で紹介した ptex2pdf を使う場合は、-u を追加する。

```
> ptex2pdf -l -u firstu.tex
```

■課題 12 Windows の機種依存文字などを含んだ up \LaTeX 用 \TeX ソースを作り、そこからこれらの文字が正常に含まれている PDF ファイルを作成せよ。

Unicode には基本多言語面に収録されていない文字もたくさん存在する。JIS X 0213 で規定されている漢字の内、𪗇𪗈などの 300 字程度が該当するし、またそれ以外にも例えば某牛丼チェーンの「吉」の字（正しくは、上半分が土になっている）がある。up \TeX 自身はこれらの BMP 外の文字をきちんと和文文字として認識するが、up \TeX 標準の日本語フォントでは事情により BMP 外の文字の出力はサポートされていない。事情については

田中琢爾, 「up \TeX , up \LaTeX — 内部 unicode 版 p \TeX , p \LaTeX の実装」.
C:\texlive\2016\texmf-dist\doc\uptex\base\01uptex_doc_utf8.txt

の「主な仕様」の〈15〉を参照されたい。BMP 外の文字もそのまま扱いたいのであれば、

```
1 \documentclass[uplatex]{jsarticle}
2 \usepackage[english]{babel}
3 \usepackage[main=japanese]{pxbabel}
4 \begin{document}
5 𪗇と𪗈, 碯と碯, 土と土
6 \end{document}
```

のように Babel の機能を利用する^{*33}のが簡単である。

^{*33} とはいいいながら、実際は八登崇之氏による [pxbabel](#) パッケージの機能である。

Ⓔ p_ⒺTeX と up_ⒺTeX では、命令名に使える文字が若干異なる。例えば、次のソースを p_ⒺTeX でタイプセットすると「さーばー」という結果を得るが、up_ⒺTeX では「! Undefined control sequence.」というエラーが発生する。

```
1 \ifdefined\ucs
2   \documentclass[uplⒺtex]{jsarticle}% upⒺTeX の場合
3 \else
4   \documentclass{jsarticle}%          pⒺTeX の場合
5 \fi
6 \begin{document}
7 \newcommand\サーバ{さーば}
8 \サーバー
9 \end{document}
```

p_ⒺTeX では音引き「ー」は命令名に使えないので、「\サーバー」は命令「\サーバ」の直後に音引き、と扱われる。一方、up_ⒺTeX では音引きは命令名に使えるので、「\サーバー」はそれ自身が 1 つの命令だと認識されるが、こんな名前の命令は定義されていないのでエラーが発生する、というわけだ。

同様の非互換性は、中黒「・」、単独の濁点「^」、単独の半濁点「°」、繰り返し記号「>」「>」「\」「>」、同上記号「全」にも見られる。

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{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^{*34}。pLaTeX・upLaTeX においては、MyTeXpert プロジェクト（渡辺徹氏）の [Wiki](#) 内からダウンロードできる `jlisting.sty` と併用することになる。

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^{*35}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{jsarticle}
2 \usepackage{amsmath}
3 \begin{document}
4
5 第\ref{sec:hoge}節にある2次方程式(\ref{eq:quad})の解は
6 \begin{equation}
```

^{*34} 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。

^{*35} 「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。

```

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

```
> platex 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

```

> platex 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。

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](#)^{*36}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：

^{*36} マニュアルは実習用マシンの C:\texlive\2016\texmf-dist\doc\latex\tools\showkeys.pdf にもある。

- `\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^{*37}。

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^AT_EX で提供するのが [hyperref パッケージ](#)^{*38}である。

`hyperref` パッケージ自体は多くの機能を持つ^{*39}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{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……ののだが、1 つめのしおりが「`“Å”’B`」といったわけのわからない文字列に化けている。



「最初の例」を Shift_JIS で符号化すると 8DC5 8F89 82CC 90DF 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「`“Å”’B`」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](#)^{*40}を、`hyperref` パッケージの後に読み込む。

というわけで、`hytest.tex` のプリアンブルを

```

\documentclass{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「最初の例」と化けない PDF が作成できる。

^{*37} 本文と多少かぶっているが、これは仕様である。

^{*38} マニュアルは実習用マシンの `C:\texlive\2016\texmf-dist\doc\latex\hyperref\manual.pdf` にもある。

^{*39} 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。

^{*40} マニュアルは実習用マシンの `C:\texlive\2016\texmf-dist\doc\platex\pxjahyper\pxjahyper.pdf` にもある。

5.3 画像

\TeX 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 \TeX 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 x 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

- [\$\text{\TeX}\$ Wiki](#) 中の「 \TeX 入門/図表」
- 「[はいぱーワークブック](#)」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{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}
```

これをいつもの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^{*41}。

```
> platex grapheg.tex
> platex grapheg.tex
> dvipdfmx grapheg.dvi
```

^{*41} 現在の \TeX Live では `extractbb` は自動実行される。

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 **Xy-pic** というパッケージを使うのが便利である. Xy-pic は文法が独特であり, 慣れるのはなかなか大変である. 詳しい使い方は,

Kristoffer H. Rose. *Xy-pic User's Guide*, 2013. <http://mirrors.ctan.org/macros/generic/diagrams/xy-pic/doc/xyguide.pdf>.

または, C:\texlive\2016\texmf-dist\doc\generic\xy-pic\xyguide.pdf

や, 公式ページに言及されているいくつかのスライド, さらに阿部紀行氏のページ「[あべのりページ](#)」の「**T_EX**の話」→「Xy-pic 入門」などを参照して欲しい.

……といっても, これだけでは実習資料としてはどう考えても不十分なので, 上の図式

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

の出力に使ったソースを例にとって説明する.

この図式の出力は, プリアンブルに

```
\usepackage[all]{xy}
```

と入力することでXy-pic を読み込んだ上で,

```
\[
\xymatrix{
  & C \ar@/_/[ld]_{f} \ar@/^/[rd]^{g} \ar@{-->}[d]^{(f,g)} \\
  A & A\times B \ar[l]_{\pi_1} \ar[r]_{\pi_2} & B
}
\]
```

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 で始まるのが図式中の矢印である.

- [...] は始点から見て相対的にどのオブジェクトに向かう矢印なのかを指定する. 例えば, \ar[llu] ならば, 相対的に 2 つ左, 1 つ上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。進行方向向かって左側に付けたい場合は \wedge で、右側には $_$ で通常の添字のように記述する。上のソース中の $\backslash\mathrm{ar}[1]^{\pi_1}$ と $\backslash\mathrm{ar}[r]_{\pi_2}$ を比較して欲しい。
- $@{\dots}$ は線種の指定である。通常の矢の他に可換図式で良く使うものとしては

$$A \equiv B \dashrightarrow C \twoheadrightarrow D$$

などがあるが、これらは

```
\[
\mathrm{xymatrix}{
A\ar@{=}[r] & B\ar@{-->}[r] & C\ar@{-->>}[r] & D
}
```

によって出力できる。

- monomorphism を表すのに

$$A \longrightarrow B$$

という矢印を用いることがある。安直に

```
\[
\mathrm{xymatrix}{
A\ar@{>->}[r] & B
}
```

とすると

$$A \longrightarrow B$$

となり、矢印と A とがかぶってしまう。これを直すためには

```
\newdir{>}{\!/\!-.75em/@{>}}% プリアンブルにでも記述、分量は各自調整
\mathrm{xymatrix}{
A\ar@{>->}[r] & B
}
```

とすればよい。

- 矢印を少し曲げたい場合には、 $@/^/$, $@/_/$ でそれぞれ進行方向左側・右側に曲がる。

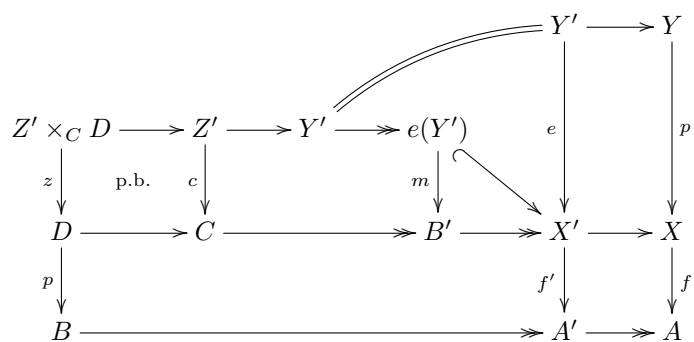
■課題 13 次の図式 (five lemma の図) をタイプセットせよ。

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

■課題 14 次の各図式をタイプセットせよ。

$$\begin{array}{ccccc}
1 & \xrightarrow{a} & A & \xrightarrow{f} & A \\
& \searrow z & \downarrow x & & \downarrow x \\
& & N & \xrightarrow{s} & N
\end{array}
\qquad
\begin{array}{ccccc}
X & \xrightarrow{f} & Y & \xrightarrow{q} & Q' \\
& \searrow g & & \searrow e & \downarrow e \\
& & & & A
\end{array}$$

■課題 15 次の図式をタイプセットせよ.



5.5 難しい小ネタ

■課題 16 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 17 (幻の “**This can't happen**” を拝もう)

T_EX - L^AT_EX 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_EX 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^AT_EX でタイプセットしてもそうはならなかった。

しかし、「pT_EX・pL^AT_EX はバグが少ない、やったね！」とは決して言えない。pT_EX 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_EX ソース（本質的には 1 行）で、pL^AT_EX 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？