LaTeX 入門

寺田 雄亮

東京大学理学系研究科　地球惑星科学専攻　大気海洋科学講座 修士 1 年
Email：terada@eps.s.u-tokyo.ac.jp

2020 年 5 月 7 日（木）

本郷に進学し、みなさんはこれからレポートなどの文書を作成する機会も益々多くなることと思います。特にこの学科にいれば、数式・図を多く含んだ文書を作りたくなるでしょう。

今回は、LaTeX という組版ソフトを紹介します。このソフトはいわゆるワープロソフトと比べると最初はとっつきにくさを感じるかもしれませんが、出来上がった文書の美しさはすばらしいものがあります。とにかく使ってみましょう。

(© このテキストは落さんが作ったテキストを三木さんが改良したものを白川さんが改良して、それをまた東森さんが改良したものを斉藤さんが改良し、さらに高木さんが改良し河野さんが改良し長谷川さんが改良し中村さんが改良し滝川さんが改良し山谷さんが改良し、最後に水越さんが改良したものを基に作成しました。)

目次

1　まず使ってみましょう

細かいことは抜きにして、まずは「とりあえず使えるようになる」ことを目指します。

1.1　Overleaf の準備

今年の演習では **Overleaf** というオンライン LaTeX エディターを使用します。**Overleaf** を使う利点としては、ブラウザ上で作業を完結させる事ができ環境構築をする必要がないということが挙げられます。

それでは早速使ってみましょう。まずは公式サイト

https://ja.overleaf.com/

にアクセスし、メールアドレスの入力・パスワードの設定をすれば登録は完了です。どのブラウザを使っても構いません[*1]。登録が完了したら**新規プロジェクト**から**プロジェクトのアップロード**に進み、/home2/terada2020/exercise/にあるtex_exercise.zipというファイルをアップロードしてください。すると図 1 のような画面が表示されます。これが **Overleaf** の作業画面です。デフォルトでは日本語文章に対応していないため設定を変更しましょう。左上のメニューか

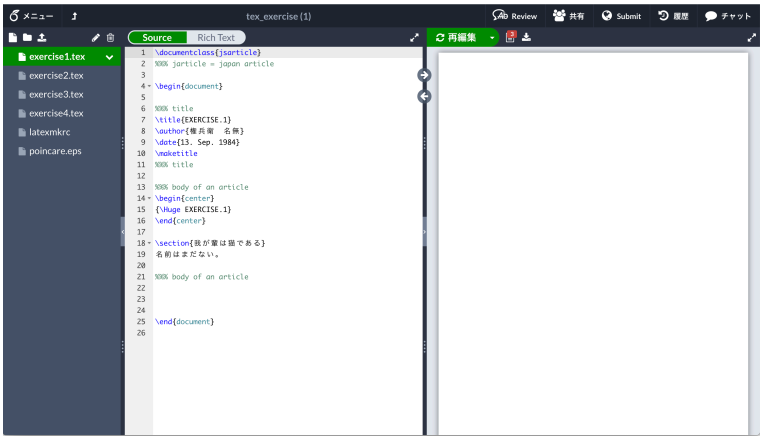


図 1　**Overleaf** の作業画面

ら**コンパイラ**で **LaTeX** を選択します。この設定とlatexmkrcという設定ファイルをプロジェクト内に置くことによって日本語文章を作成することができます。(今後別のプロジェクトで作業する際も同様の設定と、プロジェクト内に latexmkrc を置いておくようにしましょう。)

1.2　文章作成の流れ

左上の**新規ファイル**から sample.tex というファイルを作成し、左側のソースファイルの中身を次のようにします。大文字・小文字も正確に入力してください。

```
\documentclass{jsarticle}
\begin{document}
こんにちは{\TeX}。はじめまして。
\end{document}
```

次に右枠上部の**再編集**を押すことで、右側のプレビューに図 2 のように表示されます。これで文書の作成は完了です。

[*1] Google Chrome を使うとカーソルが正しく表示されない症状が現れることがあります。その場合は
設定 → フォントをカスタマイズ → 固定幅フォント
でフォントを Osaka から Monaco に変更すると解決するようです。

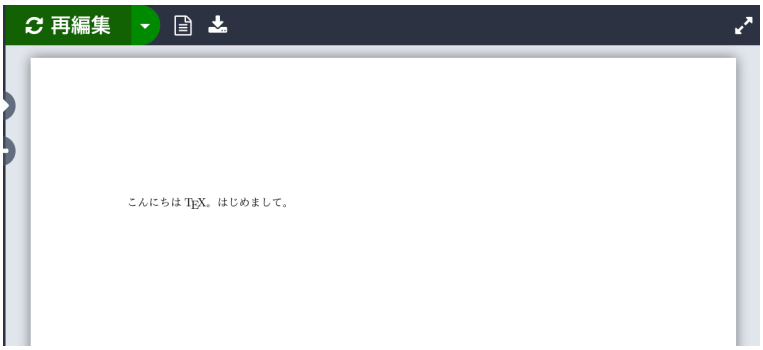


図 2　pdf のプレビュー

最後に右枠上部のダウンロードアイコンから作成した pdf ファイルをダウンロードできます[*2]。このときファイル名はソースファイル名 (sample) ではなくプロジェクト名 (tex_exercise) から取られます。

1.3　エラーに遭遇したら

LaTeX で文書を作成していると、しょっちゅうエラーに遭遇します。このときの対処方法を覚えましょう。

先ほどの sample.tex の 2 行目を少しだけ変えてみます。

```
\documentclass{jsarticle}
\began{document}
こんにちは{\TeX}。はじめまして。
\end{document}
```

これで**再編集**を押すと図 2 とは異なる結果が表示されるでしょう。このように思い通りに文章が作成されない時は再編集ボタンの右の**ログと出力ファイル**を見ます。おそらく一つ目にこんなふうに言われていることでしょう。

```
Undefined control sequence. sample.tex, line 2

<recently read> \began

1.2 \began
          {document}
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.
```

こんなときは、表示された説明をよく読みます。今の場合は `begin` というところを `began` と書いたため、怒られているようです。`1.2`(2 行目) というのもヒントになりますね。さらに二つ目のログで

```
LaTeX Error: Missing \begin{document}.
```

と言われているところからも原因を突き止めることができます。こんな時はソースファイルに戻り、間違いを直してからもう一度**再編集**を押します。

[*2] メニューからソースファイルの一括ダウンロードもできます

2　はじめに

1 章にしたがって使えば、とりあえず動くようにはなります。しかし、少しは LaTeX の歴史を知っておくのもよいでしょう。

2.1　TeX とは

TeX[*3]というソフトは、Stanford 大学の Donald E. Knuth 教授 (当時) が作成した文書組版用ソフトウェアです。TeX をどう発音するかはなかなか難しい問題なのですが、これはギリシャ語の τ と ϵ と χ をくっつけった発音になるのだそうです。日本語にも英語にも χ という音はないので、だいたい「てっく」とか「てふ」と発音します。

組版 (typesetting) というのはあまり聞き慣れない言葉ですが、もともと印刷用語で、活字を組んで紙面を構成することを指します。TeX はこれをコンピュータの上で行うためのソフトウェアです。

TeX には次のような特徴があります：

- TeX はフリーソフトなので、無料で入手できます。インストールの方法も日進月歩で進化しており、ご家庭のパソコンにも簡単にインストールできます[*4]。
- Windows や Macintosh、UNIX など様々なプラットフォームで、まったく同じ出力が得られます。
- 特に数式の美しさは変態的です。出版の専門家でない我々が

$$\frac{d}{dt}\Phi(t) = \lim_{\delta t \to 0} \frac{\Phi(t+\delta t) - \Phi(t)}{\delta t} = \int_{S(t)} \left(\frac{\partial}{\partial t}\vec{B} - \vec{\nabla} \times (\vec{U} \times \vec{B}) \right) \cdot d\vec{S} = 0$$

  とか書けてしまうのはすばらしいことです。
- ‘different’ や ‘finish’ が、それぞれ ‘different’ や ‘finish’ となるように、‘ff’ や ‘fi’ は文字がくっついたデザインになります。これを**リガチャ** (合字) といます。TeX はこれを自動で行います。
- **カーニング** (字詰め) も自動で行い、‘Volcano’ は ‘Volcano’ のように o が V の下に入ります。
- 最適な改行・最適な改ページをしてくれます。たとえば、段落の最後の 1 行だけが次のページになってしまう、というようなことは避けられます。

このようにレイアウトに関しては TeX がやってくれるので、我々は文書の中身を作ることに専念できるのです。その代わりにこのような計算にはそれなりに手間がかかりますから、「1 文字打ったら則画面に反映される」ような方法では、計算機には多大な負荷がかかるでしょうし、我々は眼が疲れるでしょう。そこで「ある程度できた段階でまとめて TeX に処理してもらう」という方法をとっています。

2.2　LaTeX とは

裸の TeX は非常に完成度の高いプログラミング言語です。完成度が高いということは裏を返せば一つのことを実現するのにたいへん多くの手順が必要であるということです。これでは実用上やや問題がありますので、基本的 (primitive) な TeX の機能を組み合わせてあらかじめ機能強化をしておいたほうが実用的です。

DEC(Digital Equipment Corporation) のコンピュータ科学者 Leslie Lamport は文書作成が簡単に行えるように TeX を機能強化し、LaTeX というシステムを開発しました。LaTeX の発音は TeX に倣って、「らてっく」とか「らてふ」でよいと思います。LaTeX は、ウェブ上で使われている HTML(HyperText Markup Languege) と同じようにマークアップ言語です。したがって雰囲気は似ていて、HTML だと

```
<CENTER>中央揃え</CENTER>
```

[*3] 手書きなどで TeX というロゴが出力できないときは、**必ず** ‘TeX’ のように e だけを小文字にして書きます。同様に LaTeX は ‘LaTeX’ と書きます。まあ ‘KinKi Kids’ みたいなもんです。

[*4] TeXLive や MacTeX などで調べてみてください。

となるのが、LaTeX では

```
\begin{center}中央揃え\end{center}
```

のようになるわけです。

LaTeX の初期バージョンは 2.09 で、現在も書籍部で LaTeX の古典的な本を見てみれば LaTeX2.09 の記述を発見できると思います。その後 1993 年には LaTeX 2_ε というバージョンにバージョンアップしました。本書ではこのシステムを説明します。

2.3 TeX と日本語

(株) アスキーによって日本語化された TeX の仲間は、頭に p(ピー，publishing の意) がつきます。設定ファイル latexmkrc にある platex というのは、LaTeX を日本語用に修正したものなわけです。

3 LaTeX の基本 (1)

3.1 文書の構造

1.2 節において、LaTeX の文書は以下のように作るという説明をしました。

```
\documentclass{jsarticle}
\begin{document}

\end{document}
```

もう少し正確にいうと、LaTeX の文書は以下のような構造をしています。

```
\documentclass[オプション]{クラス}
「プリアンブル」
\begin{document}
「本体」
\end{document}
```

まずはこの説明からしましょう。

3.2 ドキュメントクラス

`\documentclass` の部分では、どのような用途の文書を作成したいかを指定します。例えばちょっとしたレポートなら、上下左右にてきとうに空白が開いてページ下の中央にページ番号が入ればよいですが、本を作成するとなれば右ページと左ページで余白の大きさは違うでしょうし、ページ番号も外側に出力したくなるでしょう。

このように、文書の大まかな書式を**クラス**といいます．`{`**クラス**`}`の部分には、以下のようなものが指定できます。

jsarticle　和文のレポート・雑誌記事など、当面ほとんどの用途はこれ
jsbook　和文の書籍や論文；論文は後述のオプションに report を指定する
article　欧文のレポート、雑誌記事など
report　欧文の論文など
book　欧文の書籍；ページ数の偶奇で余白が異なる

`article`、`report`、`book` は欧文用で、余白のとり方、段落の下がり方などがちょっと違います。和文用クラスには、`jarticle`、`jreport`、`jbook` というものもありますが、より新しく改良されている `jsarticle`、`jsbook` を用いた方が良いようです。

また、大まかに書式は同じでも文字の大きさだけ変えたいとか、用紙サイズだけ変えたい、などという希望もあると思います。これは [**オプション**] の部分で指定します。[**オプション**] にはいろいろなものが指定できますが、少しだけ紹介すると、

文字サイズ　10pt, 11pt, 12pt のいずれかが使えます。何も指定しなければ 10pt です。
用紙サイズ　a4paper, b5paper, a5paper, letterpaper などが使えます。何も指定しなければ a4paper です。
段組　onecolumn なら 1 段組、twocolumn なら 2 段組です。何も指定しなければ onecolumn です。
用紙方向　landscape なら横長の向きになります。何も指定しなければ縦長です。

のようになります。[**オプション**] は “,” で区切って複数の [**オプション**] を指定することが可能です。

3.3 タイトル

文書を作るときは、**表題**、**作者**、**日付**を書くのが一般的です。プリアンブルの部分に

```
\title{表題}
\author{作者}
\date{日付}
```

を記述しておいてから、`\begin{document}`のあとで

```
\maketitle
```

とすることで書くことができます。

3.4 見出し

ちょっとしたメモはともかくとして、ある程度まとまった文章を書こうとした場合、まず「章」があって、その次に「節」があって…、というように論理的に組み立てていくことが大切になります。LaTeX ではこの作業を半自動的に行うコマンドが用意されています。表 1 に示すような命令を使うと，見出し用に文字の大きさなどがわかり、番号も自動的に振られます。

表 1　LaTeX での見出しの定義の種類

`\part{見出し}`	部
`\chapter{見出し}`	章 *
`\section{見出し}`	節
`\subsection{見出し}`	小節
`\subsubsection{見出し}`	小小節
`\paragraph{見出し}`	段落
`\subparagraph{見出し}`	小段落

*`(js)article` には章は定義されていません

使用例を見てみましょう。

```
\section{はじめに}
 ハナ肇の話をしましょう。
 \subsection{経歴}
 \subsection{クレイジーキャッツ}
\section{おわりに}
 \subsubsection{最近の旅行}
 尾張に行きました。
```

⇒

1 はじめに

ハナ肇の話をしましょう.

1.1 経歴

1.2 クレイジーキャッツ

2 おわりに

2.1 最近の旅行

尾張に行きました.

番号が振られ、文字が太字になり、見出しの階層に応じてサイズが大きくなりました。さらに、あとから一つ節を加えると、

```
\section{はじめに}
 ハナ肇の話をしましょう.
 \subsection{経歴}
 \subsection{クレイジーキャッツ}
\section{なかに}
\section{おわりに}
 \subsection{最近の旅行}
 尾張に行きました.
```

⇒

1 はじめに

ハナ肇の話をしましょう.

1.1 経歴

1.2 クレイジーキャッツ

2 なかに

3 おわりに

3.1 最近の旅行

尾張に行きました.

というように番号が変わります。したがって LaTeX では番号の振り間違いはありえません。このおかげで、**内容の編集に集中できる**のです。

3.5 改行の扱い

ワープロソフトでは Enter を押せば行が変わって表示されますし、印刷すれば実際にそこで改行されていることがわかります。しかし TeX は前述の通り、どこで改行すべきかをある規則にのっとって自動的に計算します。例えば、単語の途中で改行してはいけないですし、ある行は 40 文字なのに次の行は 20 文字しかない、というのもキモイです。したがって基本的に、**自分で入力した改行は無視されます**。通常は入力画面の右端に近づいたら改行すればよいですし、電子メールのように区切りのよいところで改行するのもよいでしょう。ただし、改行だけの (Enter だけが入力してある) 行があると、TeX はこれを段落の区切りと解釈します。

では具体例を見てみましょう:

```
改行は
無視されます。
どんなに
ぶち
ぶち
切っても
へっちゃらで
す。
ただし何も入力しない行があると

段落の区切りと解釈し、次の段落が始まります。
```

⇒

改行は無視されます。どんなにぶちぶち切っても、へっちゃらです。ただし何も入力しない行があると

段落の区切りと解釈し、次の段落が始まります。

もしどうしても強制的に改行したい場合は、\\とバックスラッシュを 2 個続けて入力すればいいのですが、これは慎重にやったほうがよいでしょう。

3.6 空白の扱い

空白はやや丁寧に考えてみましょう。いま、英文を入力しようと思ったとします。ワープロソフトを用いた場合、単語の間には通常半角空白を 1 つ入れるでしょう。また、センテンスとセンテンスの間には単語間よりも大きめの空白を入れますから、タイプライタではふつう、半角空白を 2 つ入れます。LaTeX では単語の区切りさえ示しておけば、これらの作業が半自動で行われます。例えば 1998 年度の東京大学入試問題の一節をタイプセットしてみると、次のようになります。

```
Simple Peter held the mirror up to
his face and peered into it. First he
turned one way, then he turned the
```

⇒

Simple Peter held the mirror up to his face and peered into it. First he turned one way, then he turned the

上の文章はすべて空白を 1 つずつしか入れていませんが、LaTeX は “it” と “First” の間を大きくしていることがわかると思います。また、1 行目と 2 行目とで右端もきちんとそろっています。ですから結論から言うと、**空白は LaTeX に任せる**べきです。

そうはいっても、どうしても空白を作りたいこともあると思います。そんなときは、空白を\で区切ります。例えば\を 5 個書けば、半角空白が 6 個分出力されます。また、空白の途中で改行されては困るときは、~を入力します。~であけた空白では改行が起こりません。具体的にはこんな感じです。

```
Fill in the blank below:\\
So Peter (                    ).\\
これでは残念賞。\\
So Peter ( \ \ \ \ \ \ \ \ \ \ \ \ ).\\
So Peter (~~~~~~~~~~~~).\\
これなら書き込めるぞ。
```

⇒

Fill in the blank below:
So Peter ( ).
これでは残念賞。
So Peter ( ).
So Peter ( ).
これなら書き込めるぞ。

3.7 練習

ここまでの練習をしましょう。プロジェクト tex_exercise の中にある exercise1.tex を適当に編集してみてください。具体的には

- `\title` などを自分仕様に変えてみる
- `\date` を省略するとどんな振る舞いをするか観察する
- `\section` やら`\subsection` やらを使ってみる
- 改行や空白を適宜放り込んで振る舞いを観察する

などという作業をやってみてください。

4 LaTeX の基本 (2)

4.1 特別な意味を持つ記号

ここまでの話で\とか{とか}といった記号には特別な意味があって、その記号自体は出力されないということに気づいた人もいるかもしれません。それは正解で、

```
\ { } $ & # ^ _ ~ %
```

の 10 文字はそのまま入力することはできません。このうち、\、^、~以外は文字の前に\をつけることで対処できます。

4.2 特殊記号の入力

もう少し勘のよい人は、LaTeX というロゴはどうやって出力しているのだろうと疑問に思うかもしれません。LaTeX には表 2 に示すように、いくつかの特殊記号・特殊文字が定義されています。

表 2　LaTeX で利用できる特殊記号

\{	{	\aa	å	\l	ł	\dag	†	!‘	¡
\}	}	\AA	Å	\L	Ł	\ddag	‡	?‘	¿
\$	$	\ae	æ	\o	ø	\S	§	\pounds	£
\&	&	\AE	Æ	\O	Ø	\P	¶	\copyright	©
\#	#	\oe	œ	\ss	ß	\i	ı	\TeX	TEX
_	_	\OE	Œ	\SS	SS	\j	ȷ	\LaTeX	LATEX
\%	%							\LaTeXe	LATEX 2_ε

4.3 書体

LaTeX では書体の種類は、

ファミリー　文字のデザインの違い；“NHK” にはひげがあるけど “NHK” にはひげがない
シリーズ　文字の太さの違い；“**YKK**” は太くて “YKK” は細い
シェイプ　形状の違い；“*This*” はイタリックで “THIS” は小文字も大文字と同じ形

の 3 種類に分けられます。書体を変更したいときは

\書体を変更する命令{変更したい文字列}

のように書きます。**書体を変更する命令**は表 3 に示すようなものがあります。

これらの命令を使って、次のように書体を変更できます。

```
She is my mother, but I am
\textbf{not} her daughter.\\
She is my mother, but I am
\textit{not} her daughter.
```

⇒

She is my mother, but I am **not** her daughter.
She is my mother, but I am *not* her daughter.

また、これらの命令は組み合わせて使うこともでき、例えば「太字でイタリックにしたい」と思ったときは次のようにします。

```
She is my brother, but I am
\textbf{\textit{not}} her daughter.
```

⇒ She is my mother, but I am ***not*** her daughter.

表 3　書体の変更

書体名	命令	出力例
ローマンファミリー	`\textrm`	This is Roman.
サンセリフファミリー	`\textsf`	This is San Serif.
タイプライタファミリー	`\texttt`	`This is Typewriter.`
ミディアムシリーズ	`\textmd`	This is Mediumface.
ボールドシリーズ	`\textbf`	**This is Boldface.**
イタリックシェイプ	`\textit`	*This is Italic.*
スラントシェイプ	`\textsl`	*This is Slanted.*
スモールキャピタルシェイプ	`\textsc`	THIS IS SMALL CAPS.

一方、全角文字は標準では明朝とゴシックしかなく、表 4 の命令を使います。以下に使用例を示します。`\textmc` はなにもしないのと同じです。

表 4　全角文字の書体の変更

書体名	命令	出力例
明朝ファミリー	`\textmc`	明朝体です。
ゴシックファミリー	`\textgt`	**ゴシック体です。**

```
強調したいところは\textgt{ゴシック体}を
使います。
```

⇒ 強調したいところは**ゴシック体**を使います。

4.4　文字の大きさ

もちろん文字の大きさも変更できます。文字の大きさを変更するには、

```
{\文字の大きさを変える命令 (変えたい文字)}
```

のようにします。**文字の大きさを変える命令**は表 5 に示すように 10 種類あります。`\normalsize` は何もしないのと同じです。

ただし、無意味に文字の大きさを変えても読みにくくなるだけです。文字の大きさを変えると、例えば次のようなことができます。

表 5　文字の大きさの変更

命令	出力例	使うべき要素 (参考)
\tiny	とても小さい	振り仮名
\scriptsize	かなり小さい	
\footnotesize	小さい	脚注
\small	少し小さい	図表見出し
\normalsize	普通	本文・小小節見出し
\large	少し大きい	小節見出し
\Large	大きい	節見出し
\LARGE	とても大きい	
\huge	かなり大きい	
\Huge	超大きい	章・節見出し

```
テーブルから紙ナプキンが
{\tiny ど}
{\scriptsize ん}
{\footnotesize が}
{\small ら}
{\normalsize が}
{\large っ}
{\Large し}
{\LARGE ゃ}
{\huge ー}
{\Huge ん}
と落ちた。
```

⇒ テーブルから紙ナプキンがとんがらがっしゃーんと落ちた。

表 5 には、標準の設定にしたときに文書中のどの要素でどの大きさが使われるかも書いておきましたので、参考にしてください。

4.5　脚注

脚注をつけるには、つけたいところに

```
脚注をつけたい文字列\footnote{脚注の内容}
```

のように書きます[*5]。番号は自動的に振られます。

4.6　コメント

もとのファイルに何かコメントを残しておきたい場合、% につづけて書きます。この部分は何を書いても出力されません。次の例が理解できれば大丈夫でしょう。

```
ここは出力されますが % ここは出力されない。
% この行は丸ごとコメントなので
出力されないでしょう。
```

⇒ ここは出力されますが出力されないでしょう。

[*5] こんなふうに書きます。

4.7 引用

一般的に単語を引用したい場合、シングルクオート ‘ ’ を使いますし、一文を引用したいときはダブルクオート “ ” を使います。ところが、キーボードを見てみるとわかりますが、‘という記号はどこにもありません。ためしに Shift+7 の ’ と Shift+2 の ” とで何とかしてみようと思うと、

```
Bob said, "This 'B' is strange".
```

⇒ Bob said, ”This ’B’ is strange”.

のように引用の始まりも終わりも同じになってがっかりしてしまいます。

正しくは、引用の始まりには Shift+@ の**バッククオート** ‘ を使います。終わりはそのまま Shift+7 の ’ でかまいません。この 2 文字の組をシングルクオートの場合は 1 つずつ、**ダブルクオートの場合は** *2* **つずつ**入力します。**決して** Shift+2 **の** ” **で代用してはいけません.**

```
Bob said, ``This `B' is strange''.
```

⇒ Bob said, “This ‘B’ is strange”.

4.8 練習

さっきの exercise1.tex の書体、文字の大きさなどを適当に変えて遊んでみましょう。

5　環境

これまで見てきたコマンドたちはみな、

```
\命令
\命令{引数}
```

という形をしていました。例えば、\maketitle と書けば表題が表示されましたし、\textbf{**もじもじくん**}と書けば**もじもじくん**のようにゴシックになったわけです。これに対して、

- ある領域を全部中央揃えにしたい
- ある領域を全部引用用に段落を下げたい

ということもあります。こんなときは、

```
\begin{環境}
  ある領域
\end{環境}
```

という**環境型のコマンド**を使います。いくつか具体例を見ていきましょう。

5.1　左揃え・中央揃え・右揃え

とりあえず何かお知らせみたいなものを作成したいとき、この 3 つはよく使うと思います。それぞれ表 6 のような環境が用意されています。

表 6　行を揃える環境

種類	環境
左揃え	flushleft
中央揃え	center
右揃え	flushright

これらを使って、

```
\begin{flushleft}
高木様 \\
池端様
\end{flushleft}

\begin{flushright}
2019 年 4 月 17 日\\
水越将敏
\end{flushright}

\begin{center}
本年度の夏期賞与について
\end{center}
```

⇒

高木様
池端様

2019 年 4 月 17 日
水越将敏

本年度の夏期賞与について

などと書くことができます。

5.2 箇条書き

長い長い文書の中に箇条書きを放り込むと、メリハリをつけることができます。LaTeX には

itemize 環境　項目の頭に • のようなマークをつける**記号つき箇条書き**です。入れ子にすると、それに伴って付くマークも変わっていきます。

enumerate 環境　項目の頭に $1, 2, 3, \cdots$ のような通し番号がつく**番号つき箇条書き**です。入れ子にすると、それに伴って番号がアルファベットになったりローマ数字になったりします。

description 環境　項目の頭が単語になっている**説明つき箇条書き**です。今のこの箇条書きが、まさに description 環境です。

という 3 つの箇条書き環境が用意されています。これらはみな、

```
\begin{箇条書き環境名}
  \item[オプション]
\end{箇条書き環境名}
```

のように使います。

例えば、itemize 環境を使うと、

```
我々の太陽系に属する天体は
\begin{itemize}
 \item planet
 \item dwarf planet
  \begin{itemize}
   \item ex) 冥王星，エリス
  \end{itemize}
 \item small solar system bodies
  \begin{itemize}
   \item 小惑星
    \begin{itemize}
     \item ex) イトカワ
    \end{itemize}
   \item 彗星
   \item ほとんどの TNO
   \item その他の小天体
  \end{itemize}
\end{itemize}
に分類される (IAU Resolution 5A)
```

⇒

我々の太陽系に属する天体は

- planet
- dwarf planet
  - ex) 冥王星，エリス
- small solar system bodies
  - 小惑星
    - ex) イトカワ
  - 彗星
  - ほとんどの TNO
  - その他の小天体

に分類される (IAU Resolution 5A)

のような箇条書きが作れます。入れ子に入るにつれて記号が変化している様子がわかると思います。また、description 環境は、次のように`\item` の後のオプションの部分に項目名を書いてやります。

```
地球惑星物理学演習では以下の内容を修得
します：
\begin{description}
 \item[UNIX] 計算機の前で途方にくれない
            ようにします
 \item[FORTRAN] FORTRAN90 のプログラミ
            ングができるようにします
 \item[行列] 逆行列の計算・連立 1 次方程
            式の計算方法を学びます
 \item[時間発展] 時間発展問題のシミュレー
            ションをします
 \item[データ解析] 時系列データの解析の基
            礎を学びます
\end{description}
```

⇒

地球惑星物理学演習では以下の内容を修得します：

UNIX 計算機の前で途方にくれないようにします

FORTRAN FORTRAN90 のプログラミングができるようにします

行列 逆行列の計算・連立 1 次方程式の計算方法を学びます

時間発展 時間発展問題のシミュレーションをします

データ解析 時系列データの解析の基礎を学びます

5.3 引用

先ほど、単語や文の引用の仕方を説明しました。しかし文といっても段落丸ごととなると話は変わってきます。例えば、

```
\emph{Aki and Richards} [2002] には，``A faulting
 source is an event  associated with an internal
 surface, such as slip across a fracture  plane.
 A volume source is an event associated with an
 internal volume, such as a sudden (explosive)
expansion throughout a volumetric source
 region. We shall find that a unified treatment
 of both source type is possible,the common link
 being the concept of an internal surface  across
 which discontinuities can occur in displacemant
 (for the  faulting source) or in strain (for the
 volume source).'' というくだりがある。
```

⇒

Aki and Richards [2002] には，“A faulting source is an event associated with an internal surface, such as slip across a fracture plane. A volume source is an event associated with an internal volume, such as a sudden (explosive) expansion throughout a volumetric source region. We shall find that a unified treatment of both source type is possible,the common link being the concept of an internal surface across which discontinuities can occur in displacemant (for the faulting source) or in strain (for the volume source).” というくだりがある。

なんてのはどう見てもちょっと勘弁して欲しいです。段落丸ごとを引用するときは、やはり別のかたまりになっていたほうが見やすいと思います。

そこで、段落の引用用に行頭の字下げをしない quote と字下げをする quotation という環境が用意されています。複数段落を引用するのなら、quotation のほうがよいでしょう。これを使うと上の例では

```
\emph{Aki and Richards} [2002] には,
\begin{quote}
 A faulting source is an event  associated with
 an internal surface, such as slip across a
 fracture plane. A volume source is an event
 associated (中略) surface across which
 discontinuities can occur in displacemant
 (for the  faulting source) or in strain (for the
 volume source).
\end{quote}
というくだりがある。
```

⇒

Aki and Richards [2002] には,

> A faulting source is an event associated with an internal surface, such as slip across a fracture plane. A volume source is an event associated (中略) surface across which discontinuities can occur in displacemant (for the faulting source) or in strain (for the volume source).

というくだりがある。

となります。

5.4 逐語引用

これからレポート等で文書にプログラミングのソースコードを載せる機会があるかもしれません。そうした時は、単にソースコードをコピペで貼り付けても LaTeX はそれをソースコードとして認識せず空白や改行を勝手につぶしてしまい、全くコードとしては読めなくなってしまいます。これは quote 環境を使っても解決できません。

```
以下に Fortran プログラムのサンプルを載せる。
program main
  implicit none
  integer ::i

  do i=1,100
     write (*,*) 'Hello, World !'
  end do

  stop
end program main
以上。
```

⇒

以下に Fortran プログラムのサンプルを載せる。program main implicit none integer ::i
do i=1,100 write (*,*) 'Hello, World !' end do
stop end program main 以上。

だからといって読めるように改行等の装飾を施していたのでは日が暮れてしまいます。こういう時は verbatim 環境を用いましょう。verbatim 環境では、内容が改行や空白も LaTeX 処理されずに “そのまま” に出力されます。

```
以下に Fortran プログラムのサンプルを載せる。
\begin{verbatim}
program main
  implicit none
  integer ::i

  do i=1,100
     write (*,*) 'Hello, World !'
  end do

  stop
end program main
\end{verbatim}
以上.
```

⇒

以下に Fortran プログラムのサンプルを載せる。

```
program main
  implicit none
  integer ::i

  do i=1,100
     write (*,*) 'Hello, World !'
  end do

  stop
end program main
```

以上。

似たような働きを持つものに\verb コマンドがあります。これは改行を含まない内容を “+” で囲んでそのまま出力しま

す。この verbatim 環境は囲んだ部分を LaTeX 処理しないので、tex のソースを内部に書いても “そのまま” 出力します。このテキストももちろん tex で作られていますが、作成時にこの verbatim 環境は至る所で使われています。どの部分に使われているかは言わずもがなですよね...。

5.5 練習

プロジェクト tex_exercise の中にある exercise2.tex と /home2/terada2020/exercise/ にある exercise2a.pdf を見比べてみてください。どうやら pdf ファイルと同じ文章を作りたかったようですが、急いで作ったのか見栄えがいまいちです。ここまでで学習したコマンドを用いてこの文書をきれいに修正してみてください。また itemize を enumerate にしてみるのもいいでしょう。

6　数式

数式の入力は、TEX の最も得意とするところです。

6.1　数式の基本

数式には、**本文中の数式**と**別行立ての数式**があります。本文中に数式を書く場合、数式の部分を “$” と “$” で囲みます。

```
$a+b$は省略しても$ab$にはならない。
```

⇒ $a+b$ は省略しても ab にはならない。

そうではなく、別の行に出力したいこともあります。そのときは “\[” と “\]” で囲みます[*6]。

```
そんなわけで、\[ c=a-b \] となりました。
```

⇒

そんなわけで、

$$c=a-b$$

となりました。

改行は無視されますから、これは

```
そんなわけで、
\[ c=a-b \]
となりました。
```

⇒

そんなわけで、

$$c=a-b$$

となりました。

と書いても同じことです。別行立ての数式は、原稿のほうでも改行しておいたほうがわかりやすいかもしれません。

6.2　空白の扱い

数式中の空白は基本的に無視されます。空白は LaTeX が自動で調整します。ですから、次の 2 つはまったく同じことです。

```
$ a + ( - b ) = a - b  $ \\
$a+(-b)=a-b$
```

⇒

$a+(-b)=a-b$

$a+(-b)=a-b$

注意深く見ると、左辺と右辺では − と b の間が異なっているのがわかります。これは、LaTeX が符号の − と演算子の − を自動で判別しているからです。ですから特別なことがない限り、人間が空白を入れる必要はありません。

それでも空白を入れたいということがあるかもしれません。というか、あります。このときは表 7 に示したような命令が使えます。空白の調節は数式の超絶技巧には不可欠なのですが、ここでは例を一つ挙げるのにとどめます。

表 7　数式中での空白の制御

空白の大きさ	命令	数式モード外での使用
かなり小さい空白	\,	可
小さい空白	\:	不可
少し小さい空白	\;	不可
半角の空白	\␣	可
全角の空白	\quad	可
全角の 2 倍の空白	\qquad	可
負の空白	\!	不可

[*6] $$...$$とする方法も存在しますが、plainTEX のコマンドなので推奨されません。

```
どちらが見慣れた式ですか？
\[ \int \int f(x,y) dx dy \]
\[ \int \!\!\! \int f(x,y) \, dx \, dy \]
```

⇒

どちらが見慣れた式ですか？

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

$$\iint f(x,y)\, dx\, dy$$

6.3 添え字

添え字は次のように^と_を使って入力します。

```
値^{上付き}
値_{下付き}
```

添え字が 2 文字以上になる場合は、上のように波カッコで囲む必要があります。別に 1 文字でも囲ってかまわないので、最初からそのようにするとよいでしょう。

```
上付きは$x^2$，2 文字以上なら$x^{22}$\\
下付きは$a_2$，2 文字以上なら$a_{22}$\\
$x^22$とか$a_22$とかは間抜けですね。
```

⇒

上付きは x^2，2 文字以上なら x^{22}
下付きは a_2，2 文字以上なら a_{22}
$x^2 2$ とか $a_2 2$ とかは間抜けですね。

6.4 大きさの変わる数学記号

ここで扱うのは、分数・総和 (シグマ) などの上下方向に場所をとる記号です。まずは書き方を表 8 にまとめましたので、見てください。これらの記号を別行立ての中で使っている分にはまったく問題ありません。しかし、本文中に使うとなる

表 8　大きさの変わる数学記号

種類	命令	出力例
分数	`\frac{分子}{分母}`	$\frac{分子}{分母}$
根号	`\sqrt{値}`	$\sqrt{値}$
多乗根	`sqrt{根}{値}`	$\sqrt[根]{値}$
定積分	`\int_{下付き}^{上付き}`	$\int_{下付き}^{上付き}$
総和	`\sum_{下付き}^{上付き}`	$\sum_{下付き}^{上付き}$

と、表 8 からもわかるように場所をとってバランスが悪くなります。そこで、これらの記号を本文中で使うと次のようにデザインが変わります。

```
本文中では$\frac{1}{2},\sum_{n=1}^{N},
\int_{0}^{1}$みたいにせまっ苦しいけど、
別行立てでは
\[
\frac{1}{2},\sum_{n=1}^{N},\int_{0}^{1}
\]
みたいに広々するね。
```

⇒

本文中では $\frac{1}{2},\sum_{n=1}^{N},\int_{0}^{1}$ みたいにせまっ苦しいけど、別行立てでは

$$\frac{1}{2},\sum_{n=1}^{N},\int_{0}^{1}$$

みたいに広々するね。

これが気に入らないという人は、`\displaystyle` という命令を分数なり総和なりの前においてやることによって解決でき

ます。実際、表 8 もそうやって書きました。ただし少なくとも分数に関しては、本文中では a/b のように書くのが正しいようです。

```
本文中でも$\displaystyle \frac{1}{2},
\sum_{n=1}^{N},\int_{0}^{1}$で大きくなる
けど、これはこれで狭いっす。\\
でも$a/b$は狭くない。
```

⇒

本文中でも $\displaystyle \frac{1}{2}, \sum_{n=1}^{N}, \int_{0}^{1}$ で大きくなるけど、これはこれで狭いっす。
でも a/b は狭くない。

6.5 関数

"$\sin x$" という出力をするつもりで`$sinx$`と書くと、出力は "$sinx$" となりがっかりしてしまいます。がっかりするだけならともかく、これでは $s \times i \times n \times x$ としか解釈しようがなく、意味まで変わってきてしまいます。sin のような関数は**立体**で書くのが普通です。そこで LaTeX ではあらかじめ表 9 に示したような関数が定義されています。極限など添え字を取るものは、普通の添え字と同じ要領でできます。

表 9　主な数学関数

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

```
\[ \lim_{x\to 0} \frac{\sin x}{x} = 1 \]
は高校で学びますか？
```

⇒

$$\lim_{x\to 0} \frac{\sin x}{x} = 1$$

は高校で学びますか？

6.6 ギリシャ文字

この学科にいれば、ギリシャ文字を使いたくなることも多いことでしょう。小文字は表 10 にまとめてあります。ただし、o(omicron) はアルファベットの o(オー) と同じなので特に用意されていません。またギリシャ文字の小文字には、表 11 に示すような異体文字もあります。大文字は表 12 の 11 通り以外は英語のアルファベットと同じです。慣習に従って、小文字は斜体、大文字は立体で出力されます。

ギリシャ文字の大文字は標準で立体ですが、数式モード中のアルファベットは標準でイタリックになっており、これではアンバランスです。これは`\mathrm{A}`とかしてやることで解決します。

6.7 その他の記号

その他よく使いそうな記号を表 13 にまとめました。

表 10　ギリシャ文字 (小文字)

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

表 11　ギリシャ文字 (小文字の異体字)

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

表 12　ギリシャ文字 (大文字)

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

表 13　関係子・二項演算子・矢印

\le	$\le$	\pm	$\pm$	\hbar	$\hbar$	\leftarrow	$\leftarrow$
\ge	$\ge$	\mp	$\mp$	\Re	$\Re$	\Leftarrow	$\Leftarrow$
\ll	$\ll$	\times	$\times$	\Im	$\Im$	\longleftarrow	$\longleftarrow$
\gg	$\gg$	\div	$\div$	\imath	$\imath$	\Longleftarrow	$\Longleftarrow$
\in	$\in$	\ast	$\ast$	\jmath	$\jmath$	\rightarrow	$\rightarrow$
\ni	$\ni$	\star	$\star$	\ell	ℓ	\Rightarrow	$\Rightarrow$
\neq	$\neq$	\cdot	$\cdot$	\partial	∂	\longrightarrow	$\longrightarrow$
\equiv	$\equiv$	\circ	$\circ$	\infty	∞	\Longrightarrow	$\Longrightarrow$
\sim	$\sim$	\bullet	$\bullet$	\nabla	∇	\leftrightarrow	$\leftrightarrow$
\subset	$\subset$	\bigcirc	$\bigcirc$	\natural	$\natural$	\Leftrightarrow	$\Leftrightarrow$
\supset	$\supset$	\vee	$\vee$	\flat	$\flat$	\longleftrightarrow	$\longleftrightarrow$
\parallel	$\parallel$	\wedge	$\wedge$	\sharp	$\sharp$	\Longleftrightarrow	$\Longleftrightarrow$

6.8　大きさの変わるカッコ

カッコの類は普通に書いてもよいのですが、例えば次の 2 つ目の例はあまり美しくありません。やはりカッコは全体を囲っていて欲しいものです。

```
\[ 3 \times (2+5) = 21 \]
はいいけど、
\[ \frac{1}{2} \times
 (
  \frac{4}{7} - \frac{5}{13}
 )
\neq \frac{19}{162} \]
はかっこわるいですね。
```

⇒

$$3 \times (2+5) = 21$$

はいいけど、

$$\frac{1}{2} \times (\frac{4}{7} - \frac{5}{13}) \neq \frac{19}{162}$$

はかっこわるいですね。

これは以下のように書くことで解決します。中身に応じてカッコが伸び縮みします。

```
\left{括弧} \right{括弧}
```

たとえば、こんな感じです。

```
\[
\frac{1}{2} \times
 \left(
        \frac{4}{7} - \frac{5}{13}
 \right)
= \frac{17}{182}
\]
```

⇒

$$\frac{1}{2} \times \left(\frac{4}{7} - \frac{5}{13}\right) = \frac{17}{182}$$

6.9 行列

もともとの LaTeX には行列をかけるようなコマンドは存在しません。その代わり、いま出てきた「大きさの変わるカッコ」の中に数字を整列させて入れてやります。これであたかも行列のように見えるというわけです。

数字の整列には array 環境を使います。その使い方は以下のとおりです。

```
\begin{array}{列指定子}
 a_{1,1}   &  ⋯   &  a_{1,n}   \\
  ⋮        &  ⋱   &    ⋮       \\
 a_{m,1}   &  ⋯   &  a_{m,n}
\end{array}
```

これではいまいちよくわからないと思いますが。先に説明します。**列指定子**には、行列の中の要素の配置場所と罫線の引き方を指定します。表 14 にまとめました。具体例を見てみましょう。

表 14　array 環境の主な列指定子

列指定子	意味
l	行列のたて一列を左揃えにする
c	行列のたて一列を中央揃えにする
r	行列のたて一列を右揃えにする
|	たての罫線を引く
||	たての二重罫線を引く

```
まずは中央揃え
\[
 \begin{array}{cc}
   \cos x  & -\sin x \\
   \sin x  &  \cos x
 \end{array}
\]
左揃えにすると
\[
 \begin{array}{ll}
   \cos x  & -\sin x \\
   \sin x  &  \cos x
 \end{array}
\]
```

⇒

まずは中央揃え

$$\begin{array}{cc} \cos x & -\sin x \\ \sin x & \cos x \end{array}$$

左揃えにすると

$$\begin{array}{ll} \cos x & -\sin x \\ \sin x & \cos x \end{array}$$

あとはこれ全体を 6.8 節で説明した「大きさの変わるカッコ」で囲ってやれば、めでたく行列の完成となります。

```
回転行列！
\[
 \left(
 \begin{array}{cc}
   \cos x  & -\sin x \\
   \sin x  &  \cos x
 \end{array}
 \right)
\]
```

⇒

回転行列！

$$\left(\begin{array}{cc} \cos x & -\sin x \\ \sin x & \cos x \end{array} \right)$$

6.10 $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX

ここまでは標準の LaTeX で使用できる機能のみを用いて数式を書いてきました。しかしさすがに行列くらいになると、実際に使い物にするには少しつらいものがあります。

というわけで、本格的に数式を書くときはアメリカ数学会が作成した $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX の使用をお勧めします。とりあえずプリアンブルで

```
\usepackage{amsmath}
```

と書けば使えます。例えば行列や積分は、

```
\[
  A =
 \begin{pmatrix}
   \cos x  & -\sin x \\
   \sin x  &  \cos x
 \end{pmatrix}
\]
```

⇒

$$A = \begin{pmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{pmatrix}$$

```
{\LaTeX}の場合、

\[ \int \int f(x,y) dx dy \]
\[ \int \!\!\! \int f(x,y) dx dy \]

{\AmS-\LaTeX}の場合、

\[ \iint f(x,y) dx dy \]
```

⇒

LaTeX の場合、

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

$$\int\!\!\!\int f(x,y)dxdy$$

$\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX の場合、

$$\iint f(x,y)dxdy$$

と、かなり簡単に綺麗に書くことができます。

他にもいろいろあるので調べてみてください。

6.11 練習

プロジェクト tex_exercise の中にある exercise3.tex を編集して、

- sin が間抜けなのでどうにかする
- きれいなカッコにしてみる
- equation 環境はどういうときに使うか考えてみる
- eqnarray 環境[*7]はどういうときに使うか考えてみる

ということをやってみてください。

[*7] $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX を使う時は、align 環境で代替します。

7　図表

駒場の実験で、レポートにおける図や表の貼り方を教わったと思います。図は下に、表は上にキャプションをつけるのでした。また、図や表を貼ったら**必ず**本文中で説明する必要があります。このとき「図 3 より」とか「表 127 より」というように番号を参照しますが、これを手動でやっていたらまあそのうち番号がおかしくなるはずです。LaTeX ではこの作業を自動で行えるようになっています。

本節ではまず表の書き方、図の貼り方を説明したあと、参照の仕方を説明します。

7.1　表の書き方

表を書くには tabular 環境を使います。使い方は

```
\begin{tabular}{列指定子}
 a_{1,1}  &  ⋯  &  a_{1,n}  \\
  ⋮       &  ⋱  &    ⋮      \\
 a_{m,1}  &  ⋯  &  a_{m,n}
\end{tabular}
```

です。先に出てきた array 環境と似ていますね。というか、数式環境に入れなくてよいことを除けば array 環境とまったく同じです。ですから、基本的な使い方は次のようになります。

```
\begin{tabular}{lll}
 l & 左寄せ  & Left の略   \\
 c & 中央    & Center の略 \\
 r & 右寄せ  & Right の略
\end{tabular}
```

⇒

l	左寄せ	Left の略
c	中央	Center の略
r	右寄せ	Right の略

```
\begin{tabular}{||c|c|c||}
 l & 左寄せ  & Left の略   \\
 c & 中央    & Center の略 \\
 r & 右寄せ  & Right の略
\end{tabular}
```

⇒

l	左寄せ	Left の略
c	中央	Center の略
r	右寄せ	Right の略

横方向の罫線を引くには、表 15 に示したような命令を使います。

表 15　主な罫線命令

命令	意味
`\hline`	横に引けるだけの罫線を引く
`\hline\hline`	横に引けるだけの二重罫線を引く
`\vline`	引けるだけの縦罫線を引く
`\cline{`**範囲**`}`	横罫線を列の範囲を指定して引く
`\multicolumn{`**数値**`}{`**列指定子**`}{`**要素**`}`	行をつなげて列指定子どおりに要素を出力する

使用例を考察していきましょう。`\hline`, `\hline\hline`, `\vline` は簡単です。

```
\begin{tabular}{lll}\hline\hline
 l & 左寄せ  & Left の略   \\
\hline
 c & 中央    & Center\vline の略 \\
\hline
 r & 右寄\vline せ  & Right の略  \\
\hline\hline
\end{tabular}
```

⇒

l	左寄せ	Left の略
c	中央	Center の略
r	右寄せ	Right の略

\vline は使いどころが難しいようですね。\cline はこんなふうに使います。

```
\begin{tabular}{lll}\hline\hline
 l & 左寄せ  & Left の略   \\
\cline{1-1}
 c & 中央    & Center の略 \\
\cline{2-3}
 r & 右寄せ  & Right の略  \\
\hline\hline
\end{tabular}
```

⇒

l	左寄せ	Left の略
c	中央	Center の略
r	右寄せ	Right の略

\multcolumn はその名のとおり、列をつなげることができます。**{数値}** にはつなげる列の数を記入します。まあこんな感じです。

```
\begin{tabular}{|c|c|c|}\hline
\multicolumn{3}{|c|}{列指定子の意味} \\
\hline
 l & 左寄せ  & Left の略   \\
\hline
 c & 中央    & Center の略 \\
\hline
 r & 右寄せ  & Right の略  \\
\hline
\end{tabular}
```

⇒

列指定子の意味		
l	左寄せ	Left の略
c	中央	Center の略
r	右寄せ	Right の略

\multcolumn はつなげる列の数に {1} を入れて、その部分だけ列指定子を変更することもできます。

全部使えば、これくらいはできます。

```
\begin{tabular}{|c|c|c|c|}\hline
\multicolumn{3}{|l|}{ } & ヨ \\
\cline{2-2}
   & い & \multicolumn{2}{l|}{} \\
\cline{4-4}
\multicolumn{2}{|l|}{} &
\multicolumn{2}{l|}{} \\
\cline{1-1}
\multicolumn{1}{l|}{め} & & ろ &  \\
\cline{3-3}
\multicolumn{2}{|l|}{} &
\multicolumn{2}{l|}{} \\
\cline{1-2}\cline{4-4}
\end{tabular}
```

⇒

			ヨ
	い		
め		ろ	

7.2 画像ファイルの扱い

画像ファイルの扱い (を説明するの) はなかなか面倒です。というのも、LaTeX 自体には画像ファイルを直接扱う仕組みがないからです。何らかの外部ドライバに依存することになります。この辺の話はそれなりに面白いので興味のある人は自分で勉強してください。ここでは非常に天下り的に説明します。

gnuplot などで作った eps 画像を取り込むには、プリアンブルに

```
\usepackage[オプション]{graphicx}
```

と記述して graphicx パッケージを読み込みます。スペルが違うと思うかもしれませんが、これで正解です。固有名詞なので我慢してください。オプションの部分には取り合えず dvipdfmx と書いておくとよいでしょう。

次に画像を貼りたい位置で

```
\includegraphics[設定]{ファイル名}
```

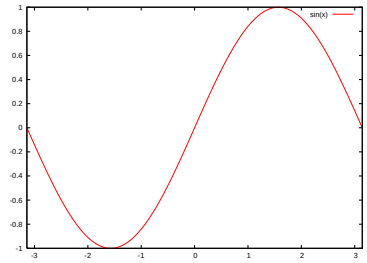
と書きます．[ ] の部分でいろんな設定ができるのですが，とりあえず “height=⟨ 高さ ⟩” で高さの設定が、“width=⟨ 幅 ⟩” で幅の設定ができるということを知っておけば路頭に迷うことはないでしょう。縦横比は保存されるので、どちらか片方を書けば十分です。

画像を貼るには手元に貼る画像ファイルを用意し、プロジェクトにアップロードしておくことも必須です。いま、同じディレクトリに sin.eps という画像ファイルがあったとします。これを貼るには、以下のようにします。

```
幅 50mm で貼るには\\
\includegraphics[width=50mm]{sin.eps}\\
これが$\sin x$のグラフです。
```

⇒

幅 50mm で貼るには

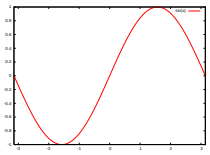


これが $\sin x$ のグラフです。

```
高さ 20mm で貼るには\\
\includegraphics[height=20mm]{sin.eps}\\
これが$\sin x$のグラフです。
```

⇒

高さ 20mm で貼るには

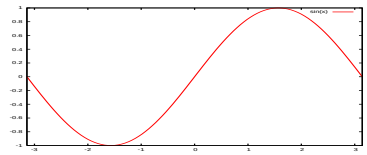


これが $\sin x$ のグラフです。

```
縦横比を壊すことも可能です。\\
\includegraphics[height=20mm,width=50mm]
{sin.eps}\\
これが$\sin x$のグラフです。
```

⇒

縦横比を壊すことも可能です。

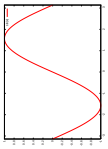


これが $\sin x$ のグラフです。

```
任意の角度だけ図を回転させることも可能です。
ただし角度の単位は\verb+degree+ です。\\
\includegraphics[width=20mm,angle=90.0]
{sin.eps}\\
これが$\sin x$のグラフです。
```

⇒

任意の角度だけ図を回転させることも可能です。
ただし角度の単位は degree です。



これが $\sin x$ のグラフです。

7.3 浮動体

この節の最初に書いたように、図や表の書き方には決まりがあります。これを実現するために，LaTeX には「浮動体 (float)」という概念があります。具体的には、表を貼り込むときは table 環境で、図を張り込むときは figure 環境で入れ子にしてやります。こうすることで LaTeX は最適な位置に図や表を張り込もうとします。図の浮動体の使い方は以下のとおりです。

```
\begin{figure}[位置指定子]
\includegraphics[設定]{ファイル名}
\caption{図の見出し}
\label{ラベル}
\end{figure}
```

一方、表の浮動体の使い方は

```
\begin{table}[位置指定子]
\caption{表の見出し}
\label{ラベル}
\begin{tabular}
(実際の表本体)
\end{tabular}
\end{table}
```

このうち、\includegraphics や tabular 環境などは 7.1 節や 7.2 節で説明したものとまったく同じです。これを、図ならば\begin{figure}と\end{figure}で、表なら\begin{tabular}と\end{tabular}で囲ってやります。こうすると、LaTeX はこの図や表のセットを持って移動し、もっとも適切な位置に出力します。例えばページの下で狭くて入りきらないときは、次のページの上のほうに出力しようとするわけです。

ただし、出力位置はある程度はユーザがコントロールすることができます。それが位置指定子です。位置指定子には表 16 に示すような種類があります。位置指定子を書くことで、図の出力可能位置を指定することができます。例えば [htbp]

表 16　浮動体の位置指定子

記号	配置する場所
h	まさにその場所
t	ページの上部
b	ページの下部
p	新たに別ページをおこす

と書いたりします。何も指定しなければ [tbp] となります。位置指定子は出力可能な場所を指定しているだけなので、

htbp を並べる順番は関係ありません。

\caption には図や表のタイトルを記入します。図のタイトルは図の下、表のタイトルは表の上、という決まりがあるので、図と表では\caption を入力する位置が異なります。

仮に図や表が次のページに旅立っていってしまったとしても、私たちは図や表に番号が付いているおかげで参照することができます。それが\label{**ラベル**}の役割です。例えばこのあたりの原稿は次のように書かれています[*8]。

```
出力位置は、ある程度はユーザがコントロールすることができます。それが位置指定子です。
位置指定子には表\ref{tab:float}に示すような種類があります。
\begin{table}[htbp]
\begin{center}
\caption{浮動体の位置指定子}
\label{tab:float}
\begin{tabular}{ll}
    （中略）
\end{tabular}
\end{center}
\end{table}
位置指定子を書くことで，…
```

\label{tab:float}と\ref{tab:float}いう組があるのに気づくでしょうか。図や表に、引用するときの目安になる目印 (**ラベル**) をつけておき、本文中でそれを**参照**すると、図や表の通し番号が自動で出力される、というわけです。\label と\ref は組になって働くので、中身は tab:float でなくても、hyou とか何でもかまいません。要は、\ref{**ラベル名**}という記述があれば、LaTeX は対応する\label{**ラベル名**}を探しに行くということです。ついでにこの例で\caption の使い方なども観察しておいてください。

これは非常に便利で合理的な機能なのですが、どうも図や表はその位置に出力されて欲しいと思う人が多いようです。しかし、ここで示した方法がレポートや論文での正しい参照の仕方ですから、気にすることはありません。また、**決して自分で図の番号を振ろうなんて思ってはいけません**。

7.4 練習

プロジェクト tex_exercise の中にexercise4.tex とpoincare.epsがあります。exercise4.texを開きコンパイルすると図も表も含まれた文書ができるはずです。まずはながめてみるのがよいでしょう。その上で、

- 図のサイズを変えてみる
- \label{poincare}と\ref{poincare}の組を他の名前 (例えば\label{chaos}と\ref{chaos}とか) に変えても結果が変わらないことを確かめてみる

ということをやってみましょう。

8 コマンドの編集

ここまで読んでくれれば、TeX を使って一通りの作業ができ、レポートなどの文書も書けるようになると思います。しかし記号の一つ一つに対してコマンドが割り振られている、という TeX の特性上どうしても一つの文書を書き上げるまで多くの文字を打ち込まなければなりません。ただ面倒臭いと言うだけならまだしも、レポートなどになると偏微分の記号 $\dfrac{\partial}{\partial t}$

[*8] 表やグラフを中央揃えにする場合、\begin{center}の代わりに\centering とする方法もあります。こちらの方が好ましい場合もあります。詳しくは調べてみてください。

などがしょっちゅうでてきて、これをいちいち`\frac{\partial}{\partial t}`などと打ち込んでいるのは辟易してきます。このような場合には自分で偏微分の記号を一発で表示させるコマンドを作ってしまう方が便利です。このような場合に非常に重宝するのが`\def` コマンドです。たぶん `define` かなんかを短縮したものでしょう。さて、プリアンブルに例えば、`\def\deln#1{\frac{\partial}{\partial {\mbox{$#1$}}}}` と打ってみましょう。これはちょっと長めですが、このあとの幸せを考えるとまあ耐えられます。さてこの一行の意味ですが、`\def` というコマンドで直後の`\deln` というコマンドを定義せよ、と命令しているわけです。`\deln` の直後の`#1` は引数を入れる欄を作っています。わざわざ真似て新作のコマンドの名前を`\deln` にする必要はないですが、ここでは偏微分の “デル” の分子に “微分される関数が何もつきませんよ (nothing)” のつもりでこの名前にしてます。別に`\def\riderkick#1` 云々でも構いません。こうやった上で微分したい変数を引数にいれて

この様に`\\`
`$\deln{x}$\\`
とすれば

⇒

この様に
$\frac{\partial}{\partial x}$
とすれば

ちゃんと偏微分が表示されています。これはほんの一例ですので色々と各自試して見てください。微分の記号くらいは充実させておくと便利です。なお`\def` の代わりに`\newcommand` でも似たような操作ができるみたいです。興味がある人は調べてみてください。

9　課題：数式の書き方，図の貼り込み

/home2/terada2020/homework/homework.pdfという文書があります。しかしこの文書には残念ながら適当な図が貼り付けてありません。本文を TEX で作成し、先日習った gnuplot で適切な図を作成し (どんな図でもいいです、また以前つくったものでも構いません)、貼り付けましょう。ついでに、文書の最初に課題名、名前、学籍番号、提出日も入れておきましょう。空白の大きさ等細かい所まで配布文書をまねる必要はないですが、図の番号付け等、最低限レポートとして人に見せられる体裁は整えましょう。

余裕がある人は図の下または次ページに何かをつけ加えてみてください（箇条書き等の、課題でまだ使っていないコマンドを使ってみる。図を横に複数枚並べてみる。など)。

文書作成の際には新しいプロジェクトを立ち上げてください (この際 1.1 で行った設定を忘れずに)。出来上がったら、pdf ファイルとソースファイルをダウンロードし、slack のダイレクトメッセージで`.zip` ファイルを寺田に送ってください。期限は 5 月 14 日 (木)12:59(JST) とします。

付録 A　文字コードの設定 z

tex ファイルの文字コードが適切でないと (559 の edu では utf-8 でないと)、文字化けを起こしてしまいます。最悪の場合、platex の段階でエラーを起こして dvi ファイルが出来上がらないこともあります。ここではその対処法をまとめておきます。

A.1　Emacs の文字コードの設定

??節で、

Ctrl+x Enter f

で UTF-8 に設定する方法をお話しました。これは打った文字をそのままに、保存する文字コードを変更するコマンドです。しかし、この方法では Emacs のウィンドウを閉じるたびに文字コードが EUC-JP に戻ってしまいます。毎回これを打ち直すのは面倒ですから、設定ファイル`.emacs` を覗きにいきましょう。

Emacs で.emacs を開きましょう。この中から、

```
;; YaTeX-mode
(setq auto-mode-alist
      (cons (cons "\\.tex$" 'yatex-mode) auto-mode-alist))
(setq dvi2-command "xdvi"
      tex-command "platex"
      dviprint-command-format "dvips %s -o !lpr -Pionia"
      YaTeX-kanji-code 3)
```

を見つけてください。この中の、YaTeX-kanji-code 3が、YaTeX で EUC-JP を使うように、と設定しています。ここを YaTeX-kanji-code 4と書き換えます。

```
;; YaTeX-mode
(setq auto-mode-alist
      (cons (cons "\\.tex$" 'yatex-mode) auto-mode-alist))
(setq dvi2-command "xdvi"
      tex-command "platex"
      dviprint-command-format "dvips %s -o !lpr -Pionia"
      YaTeX-kanji-code 4)
```

これで YaTeX が UTF-8 を使ってくれるようになります。

先ほどのコマンドとそっくりですが、

Ctrl+x Enter r

というものがあります。これは Emacs でファイルを開いたら文字化けした場合に使います。「UTF-8 で書いたはずのものが文字化けした」というときにはこれを使って、ミニバッファに ‘utf-8’ と入力すれば良いです。何か聞かれたら ‘yes’ と答えておきましょう。

興味のある人は、`.emacs` 内の、

```
(set-default-coding-systems    'utf-8)
(set-terminal-coding-system    'utf-8)
(set-keyboard-coding-system    'utf-8)
(set-buffer-file-coding-system 'utf-8)
(set-clipboard-coding-system   'utf-8)
(setq default-enable-multibyte-characters t)
```

がどういう意味なのか調べてみてください。

A.2 ターミナルの文字コードの設定

既に環境変数についてならっていることと思います。ターミナルの文字コードは、この環境変数で設定できます。

現在の文字コードを確認するには、ターミナルでecho $LANGとします。'ja_JP.EUC-JP' と表示されたら EUC-JP、'ja_JP.utf8' と表示されたら UTF-8 です。UTF-8 にしたければ、ターミナル上でexport LANG=ja_JP.utf8とします。しかし、この方法では、変更はターミナルを開いている間のみ有効で、一度ターミナルを閉じると、再び EUC-JP に設定されてしまいます。

さて、ホームディレクトリに.bashrc というファイルがありましたね。この中身をのぞくと、下の方に

```
export LANG=ja_JP.EUC-JP
```

という行があると思います。ターミナルの文字コードは環境変数 LANG によって決められており、.bashrc によって現在は EUC-JP に設定されています。そこでこの行の頭に#をつけてコメントアウトし、直後に一行付け加えて、

```
#export LANG=ja_JP.EUC-JP
export LANG=ja_JP.utf8
```

とすることで、ターミナルを立ち上げるたびに UTF-8 に設定されるようになります。今すぐにこの変更を適用するには、source .bashrcが必要でしたね。

付録 B　執筆支援環境

これまで見てきたとおり、LaTeX の文書はただ文を入力すればよいのではありません。これは初学者にとっては非常に疲れることです。Emacs には YaTeX と呼ばれる素晴らしい執筆支援環境があるので紹介します。

B.1　Emacs と YaTeX(野鳥)

興味のある人はホームディレクトリにある.emacs というファイルをless .emacsとかでのぞいてみてください。そこに Emacs での LaTeX の設定などが書いてあります。かなりマニアックですが…。

YaTeX の細かい機能は

```
http://www.yatex.org/
```

からたどれますが、この中から主な機能を抜き出すと次のようになります。以下、例えば SPC はスペースキーを押すことをあらわします。

B.1.1　補完入力

begin 型補完　Ctrl+C Ctrl+B SPC と入力すると、ミニバッファで

```
Begin environment(default document):
```

とか聞いてきます。この状態でそのまま Enter を入力すれば、

```
\begin{document}


\end{document}
```

のような`\begin{環境名}` ··· `\end{環境名}`の組み合わせが出てきます。もし、自分が入力したいのはほかの環境名だぜ、というときは、その環境の名前 (`equation` とか) を入力して Enter すればよいのですが、たいていは頭文字 (`equation` なら E) と Tab で補完してくれます。

section 型補完　Ctrl+C Ctrl+S と入力すると

```
(C-v for view-section) ???{}(default documentclass):
```

とか聞いてくるので、以下同じ。

large 型補完　Ctrl+C Ctrl+L と入力すると,

```
{???  }(default large):
```

とか聞いてきます。もう予想はつくと思いますが、ここで Enter すれば、`{\large }`となって、カーソルは “`{`” と “`}`” の間に移ります。

随時補完　このほかにも、`\bigskip` のような長いコマンドを入力するときは、最初の数文字と Ctrl+C SPC で補ってくれます。自分の希望するものでなければ Tab を入力すれば一覧が出ます。

ギリシア文字，数式記号補完　ギリシア文字や数式記号も一発で補完してくれます。ギリシア文字を打ちたかったら、数式環境の中で : 頭のアルファベット を入力すればよく、数式記号は ; に続いて割り当てられてキーを打てばいけます。どのキーが対応するかはググってみましょう。例えば $\frac{\partial \alpha}{\partial\beta}$ も
`\frac{\partial \alpha}{\partial\beta}`

なんていちいち入力せずに
[;] [f] [Enter], [;] [6] [Enter], [:] [a] [Enter], [Enter], [;] [6] [Enter], [:] [b] [Enter]
となる訳です。慣れると格段にスピードが上がります。

B.1.2 おまかせ改行

itemize, enumerate,description 環境の中では、改行したら次の行の頭には`\item` とか`\item[*]` とか入力することが多いわけです。こんなときは[Esc]+[Enter][*9]で勝手に`\item` を補ってくれます。

B.1.3 プロセス起動

LaTeX のソースを編集中、[Ctrl]+[C] [Ctrl]+[T]と入力すると、ミニバッファに

```
J)latex R)egion E)nv B)ibtex mk(I)dx latex+p(D)f K)ill P)review V)iewErr L)pr
```

のように表示されます。この説明に従って、例えば[J]と入力すれば LaTeX のタイプセットが始まりますし、[P]とすれば xdvi が起動します。極めつけに[D]と入力すると、タイプセットした上で pdf まで作ってくれます。

B.2 Windows や Max OS X で Emacs+YaTeX

Windows な方のために、GNU Emacs によく似た xyzzy というテキストエディタがあります。これに KaTeX(← かてふ) というメジャーモードを重ねると、Emacs+YaTeX とほぼ同等の扱いができるようになります。詳しくは

http://www.jsdlab.co.jp/k̃amei/
http://osuneko.at.infoseek.co.jp/xyzzy/xyzzy.html

あたりからどうぞ。

Mac OS X はそもそも UNIX 系 OS なので、Emacs は比較的簡単にインストールできます。もちろん Cygwin や MSYS2 などの UNIX 仮想環境を入れれば、Windows 上でも本家 GNU Emacs を動かすことができます。

B.3 LaTeX の統合執筆環境

Emacs+YaTeX は、必要十分な機能が揃っていて、何よりも Emacs の操作に長けた人には最適な執筆支援環境です。一方で LaTeX に特化した GUI の統合開発 (執筆) 環境[*10]もたくさん存在し、かなり普及しています。気になる人は調べてみると良いでしょう。統合執筆環境と Emacs+YaTeX の大きな違いは、マウス操作でコマンドを選択できる点と、見た目が現代的なことです。ソフトごとに特色があり好みが分かれますが、最近は TeXStudio や Lyx などが流行っているようです。Lyx に至っては、Word のようにリアルタイムに数式や文章構造の視覚的フィードバックを得ながら LaTeX 文書を書くことができます。

[*9] これでうまくいかない場合は、[Alt]+[Enter]
[*10] 統合開発環境とは、プログラミングに必要なコンパイラやテキストエディタ、プレビューアを GUI(グラフィカルユーザインターフェース) でまとめたソフトウェア環境のことです。例えば、Xcode や Visual Studio,Eclipse などが有名です。