

TEX 入門

かつらだ まさし
桂田 祐史

2012年8月16日, 2023年5月12日

最新版は <http://nalab.mind.meiji.ac.jp/~mk/labo/text/tex2022.pdf>

TEX をどう使いこなすかは、利用しているコンピューター環境にもよるし、時の経過につれて少しずつ変化しているので、年度毎に作り直すことにしました。最初に作った版は数学科学生の情報処理教室での利用が念頭にありましたが、2014年版以降は現象数理学科の MacBook での利用を前提にしています。

目次

1	TEX とは?	4
2	L ^A TEX 最初の一歩 (TEXShop 版)	5
3	L ^A TEX 最初のおさらい	10
3.1	L ^A TEX 文書の書き方	10
3.2	TEX のための準備作業	12
3.3	基本的な L ^A TEX の使い方 (基礎知識として)	12
4	L ^A TEX 文書 .tex の書き方 — 入門	13
4.1	最初に覚えるべきこと	13
4.2	改行と空白 (最低限の注意)	15
4.3	文字の大きさと書体	16
4.3.1	文字の大きさ	16
4.3.2	文字の書体の指定	17
5	簡単な数式	18
5.1	数式モード	18
5.2	かっこ (parenthesis (), brace { }, bracket [])	19
5.3	空白 (スペース)	20
5.4	色々な記号	20
5.4.1	ギリシャ文字	20
5.4.2	集合と論理	22
5.5	上つき添字、下つき添字、それと積分&シグマ	22
5.6	分数	23
5.7	sin などの「作用素」	24
5.8	矢印	25

5.9	点	25
5.10	不等式	26
5.11	その他の記号	26
5.12	行列、ベクトル、場合分けの {	27
5.13	数式中の言葉	28
5.14	数式の縦揃え	29
5.15	下線、上線、矢印など	30
5.16	数式番号について	30
5.17	misc	31
6	文書の構造など	31
6.1	chapter, section, subsection, paragraph など	31
6.2	自動目次生成	32
6.3	参考文献表	32
6.4	この節で解説した項目の使用例	34
7	もう一段ていねいに	34
7.1	索引	35
7.2	相互参照	35
7.3	脚注 (フットノート)	35
7.4	箇条書き (enumerate, itemize, description 環境)	35
7.5	定理環境	35
8	\TeX のマクロ機能、パッケージ機能の紹介	35
8.1	マクロ	35
8.2	パッケージ	36
9	ソースプログラム等テキストファイルの \LaTeX 文書への取り込み	37
10	画像の \LaTeX 文書への取り込み	38
10.1	概要	38
10.2	とりあえず例を1つ	41
10.3	よくある失敗の原因を避ける	42
10.4	複数の図を並べる	43
10.5	figure 環境について補足	43
10.6	補足的情報	44
10.6.1	PostScript データの取り込み	44
10.6.2	JPEG イメージの取り込み	46
10.6.3	JPEG 以外のイメージファイルの取り扱い	47
10.6.4	dviout でカラー表示・印刷するには	48
10.6.5	余談: ウィンドウの画像を取り込む	48
10.6.6	追記: PDF を PS にする	48
10.6.7	misc	48
10.6.8	ドライバーについて	48
10.6.9	.xbb ファイル	49

11 TikZ	50
11.1 準備	50
11.2 マニュアル	50
11.3 いろは — 直線、円などを描く	51
11.4 plot	54
11.5 模式図	56
12 L^AT_EX Beamer でプレゼン	57
12.1 準備	58
12.2 必要最小限の知識	58
12.3 stepwise viewing	59
12.4 リンク	60
12.5 しおりの文字化けの防止	60
12.6 その他	61
12.7 授業資料準備をして覚えたこと (細かいノウハウ)	61
12.8 初めてのスライド発表をまじかに控えた人に	63
13 PDF を作る しおりとか	64
A Tips	64
A.1 用紙のサイズ	64
A.1.1 L ^A T _E X 文書の中で	64
A.1.2 後で dvipdfmx を使うことを見越して	65
A.1.3 色々なコマンドでの用紙サイズ指定のオプション	65
A.2 バージョンが書いていない PostScript ファイル	66
A.3 負の座標を含む BoundingBox を直す	66
A.4 今いつでしょう? (<code>\today</code> その他)	66
A.5 jobname	66
A.6 macOS プレビューメモ	67
A.7 QED	67
A.8 \mathbb{R} など黒板太字	67
A.9 下付きチルダ	67
A.10 ベクトルの太字	67
A.11 ベクトルの矢印	68
A.12 rsfs フォント (ある 1 つの花文字)	68
A.13 exsheets (Yet another package for the creation of exercise sheets) スタイル	69
A.14 下線	70
A.15 URL など	70
A.16 文字に色をつける	70
A.17 表の斜め線 — <code>diagbox</code> パッケージ	71
A.18 ラプラシアン	71
B 日本の数学書、学校数学のルール	72
B.1 図形の点	72
B.2 等号の否定	72
B.3 初等幾何の記号	72

1 T_EX とは？

(駆け足で説明する。)

T_EX は組版ソフトである

T_EX は、著名なコンピューター科学者であるドナルド・クヌース (Donald Knuth¹, ウィキペディア²にも載っています, “The Art of Computer Programming” シリーズが有名です) が開発した**文書整形システム** (組版³システム) です (最初のバージョンは 1978 年にリリースされました)。T_EX は日本では、「てっく」または「てふ」と呼ばれることが多いです⁴。

当初、数式を含む英語の文章を清書することを目的に、従来の組版技術の歴史を入念に調べた上で、それをコンピューター上で実現することを目標に開発されたそうです。

ワープロ (ワードプロセッサ・ソフトウェア) と比べると⁵一長一短ありますが、特に長い論文や書籍のような文章を組版するには向いているとされています。

T_EX はフリーソフトである

Knuth 自身は T_EX に関する情報を完全に公開していて (書籍になっています)、ソフトウェアは無償で利用することができます。また、多くのボランティアの活動により、T_EX を補助、発展させるためのソフトウェア、データもほとんどは無償で利用可能です。例えば、T_EX 本体や周辺ソフトウェアの C 言語への変換、画面表示用ドライバー (プリビューアと呼ばれます)、印刷用ドライバー、PDF への変換ソフトウェア、日本語対応、ラテン文字&数式記号のフォント、日本語フォント (やそれを利用する仕組み)、Windows 環境への移植、インストーラー、統合環境 (TeXShop, TeXworks, etc.) などなど。これら成果物は大抵はインターネットから無償で入手できます。

上はソフトウェアについて書きましたが、そういうソフトウェアを使いこなすための情報もインターネット上で入手できます。一般にネットで入手できる情報は玉石混交の場合が多いですが、T_EX に関する情報は良いものが多いと感じています。

T_EX は数学の世界では標準である

数学者村では、標準の文書作成ソフトウェアです。理工系の多くの分野で利用されていますが、それだけでなく文系の研究者が利用した例もあります (発音記号や、ややマイナーな言語などを扱う場合)。

¹<http://www-cs-faculty.stanford.edu/~knuth/>

²<http://ja.wikipedia.org/wiki/ドナルド・クヌース>

³組版 (くみはん) とは、文字や図版などの要素を配置して、紙面を構成することで、もともとは活版印刷において、活字を組み上げることから来ている。

⁴「てっくす」とは読みません。ちなみに T_EX の解説書に “Joy of T_EX” という本があって、それは英語圏の国では有名な本のパロディだったそうです。昔、テレビで深夜映画を見ていたら、元ネタの本が出て来て、思わず見入ってしまいました。

⁵ワープロは WYSIWYG (What you see is what you get), つまり「画面に見えているものがそのまま印刷される」、「印刷される見映えのまま画面で作業できる」で、T_EX のようなコマンド形式のソフトウェアとは大きな違いがあります。

TeX で高品位の文書が作成できる

組版技術をしっかり研究した上で作られたものであるため、高品質な仕上がりが得られます。異なる環境下での再現性も抜群です (誰が何処で何を使って印刷しても同じ仕上がり — 同じフォントが使えれば、ですが。古いパソコンで出来たことが新しいパソコンで出来るとは限らないのが普通なので、TeX のこの性質は大きな特徴であると言えるでしょう)。英語圏ではもちろん、日本でも理工系の多くの書籍 (中学高校の教科書や問題集なども含む) で採用されています。

TeX で作った文書は PDF にして配布が楽々

TeX 自身は文書の配布フォーマットとして適当ではありませんが (表示、印刷に専用のソフトウェアが必要なためです)、TeX で書いた文書は簡単に PDF (portable document format) に変換できるので、そうしてから配布すれば、相手を読めるだろうか、印刷できるだろうか、心配する必要はほとんどありません。

2 L^ATeX 最初の一步 (TeXShop 版)

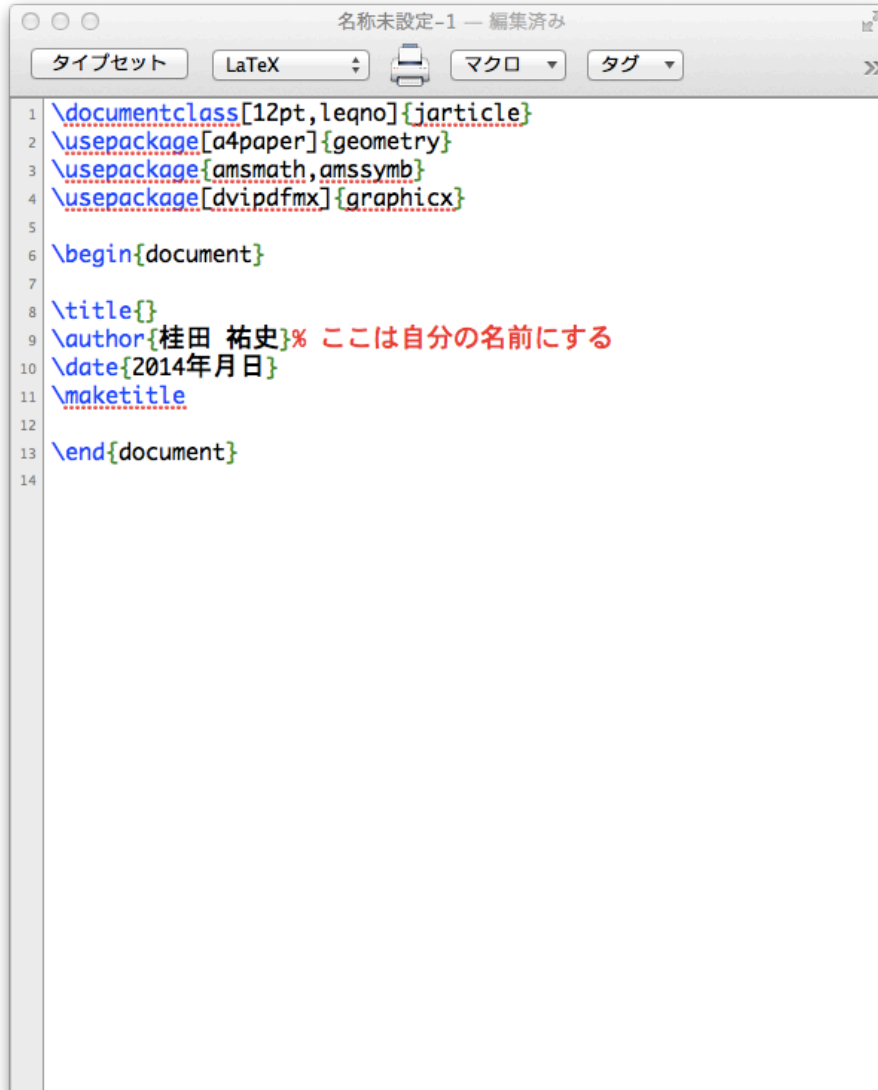
この授業では、TeX の一種である L^ATeX (正確にはその日本語対応版 pL^ATeX) を使ってもらうことにします。

TeXShop を起動してみよう。色々な設定の仕方がありうるけれど、必ず出来そうなのは、Finder でアプリケーションから TeXShop をダブルクリックして起動する、というやり方である。

“名称未設定-1” という名前のついたウィンドウが出て来るはずで、キーボードから入力して、図 1 のようにしよう。

[ファイル] メニューから [書き出す] を選択すると、書き出し名を尋ねられるので、適当な場所、適当な名前を指定する。自分で TeX 文書用のディレクトリやゼミ授業のディレクトリを準備して、そこに保存するのが良いが、ここでは書類ディレクトリに保存する。

少し書き足してみよう。



The image shows a screenshot of the TeXShop application window. The title bar reads "名称未設定-1 - 編集済み". The menu bar includes "タイプセット", "LaTeX", a printer icon, "マクロ", and "タグ". The main text area contains the following LaTeX source code:

```
1 \documentclass[12pt, leqno]{jarticle}
2 \usepackage[a4paper]{geometry}
3 \usepackage{amsmath, amssymb}
4 \usepackage[dvipdfmx]{graphicx}
5
6 \begin{document}
7
8 \title{}
9 \author{桂田 祐史}% ここは自分の名前にする
10 \date{2014年月日}
11 \maketitle
12
13 \end{document}
14
```

図 1: TeXShop にひな形を入力

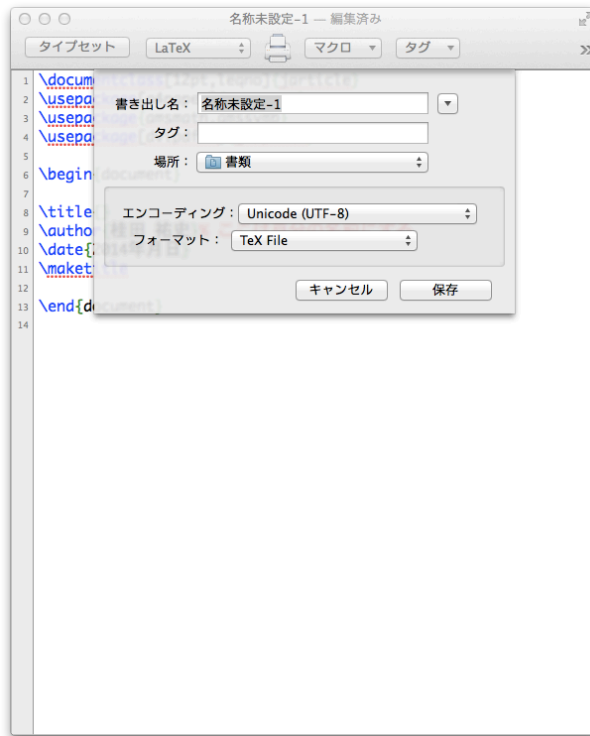


図 2: 書き出す時のウィンドウ — ▽ ボタン

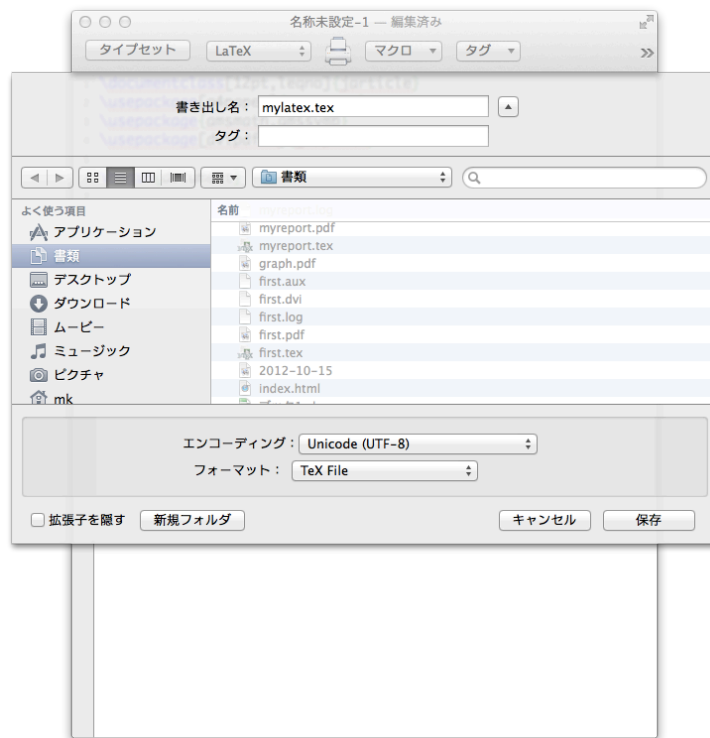


図 3: 書類フォルダに mylatex.tex という名前で書き出す

```

\documentclass[12pt,leqno]{jarticle}
\usepackage[a4paper]{geometry}
\usepackage{amsmath,amssymb}
\usepackage[dvipdfmx]{graphicx}
\usepackage{otf}

\begin{document}
\title{初めての\TeX}
\author{桂田 祐史}% ここは自分の名前にする
\date{2021年4月20日}
\maketitle

こんにちは。

\[
\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}.
\]
\end{document}

```

[ファイル]メニューの項目[保存]を選び、myfirst.tex という名前で保存しよう (名前には“myfirst”とだけ入力すれば良い)。

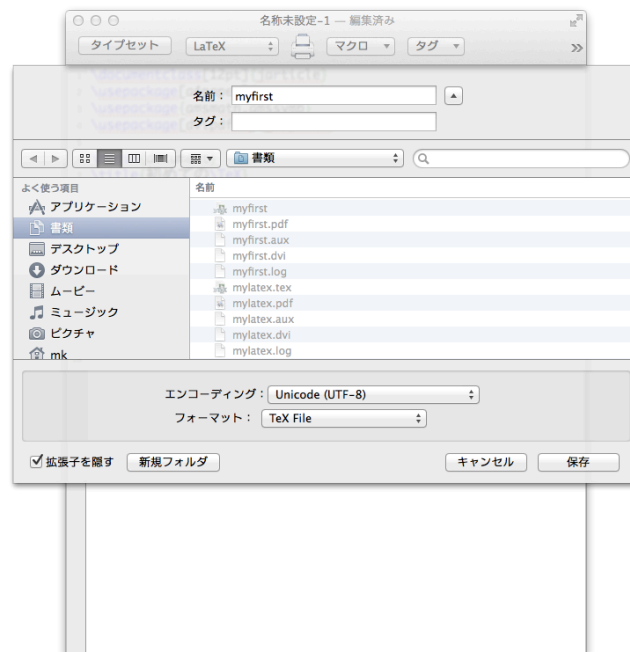


図 4: 保存する場所と名前を指定する

[タイプセット] ボタンを押すと、入力間違いがなければ、図 5 のようになる。

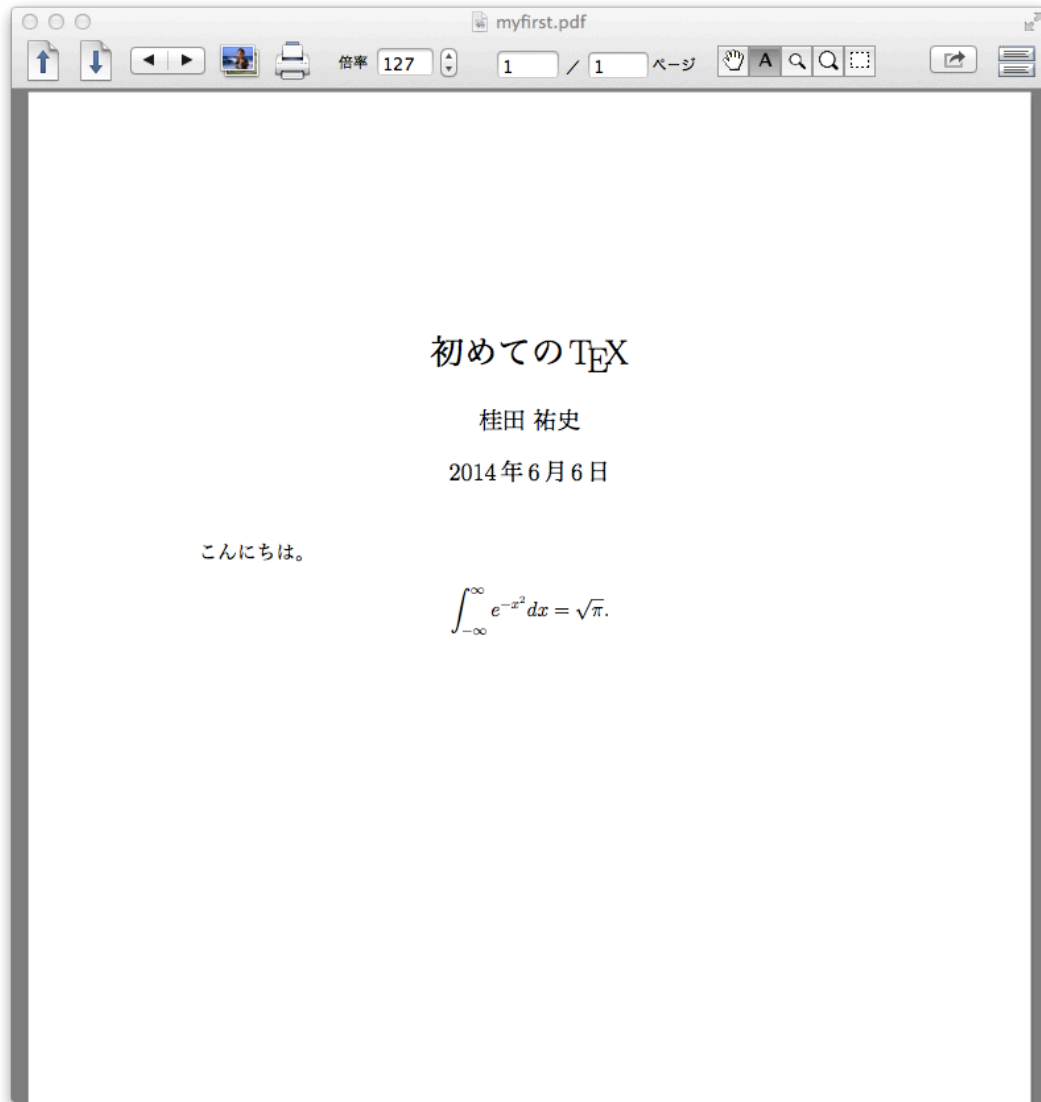


図 5: こんなふうなのが出来れば成功

```
\documentclass[12pt,leqno,dvipdfmx]{jarticle}
\usepackage[a4paper]{geometry}
\usepackage{amsmath,amssymb}
\usepackage{graphicx}
\usepackage{otf}

\begin{document}
\title{}% タイトル
\author{}% 自分の名前にする
\date{2019 年月日}
\maketitle
%\tableofcontents

\end{document}
```

のような内容は、大体いつでも必要になるものである(もちろん細かいところは個人の好みで異なる)。TeXShop のファイルメニューには、「ひな形を用いて新規作成」という項目がある。

~/Library/TeXShop/Stationery/ にひな形にしたいファイルをおけば良い。

ターミナルで実行すると、ひな形が用意できる

```
curl -O http://nalab.mind.meiji.ac.jp/~mk/labo/text/LaTeX_Osusume_Template.comment
curl -O http://nalab.mind.meiji.ac.jp/~mk/labo/text/LaTeX_Osusume_Template.tex
cp -p LaTeX_Osusume_Template.* ~/Library/TeXShop/Stationery
```

その他にも、~/Library/TeXShop/Templates においたファイルを「テンプレート」として取り込む機能がある。

案外テンプレートに登録する方が便利?

```
cp -p LaTeX_Osusume_Template.* ~/Library/TeXShop/Templates
```

3 L^AT_EX 最初のおさらい

3.1 L^AT_EX 文書の書き方

数式の書き方は後回しにして、見本で使うような基本的事項を説明しておく。

L^AT_EX 文書で最低限必要なのは次の内容である。

```
\documentclass[12pt,leqno]{jarticle}% スタイルの指定
\begin{document}
\end{document}
```

レポート等では、タイトル、著者名、日付が必須なので、次のようなものが必要と思って良い。

```
\documentclass[12pt,leqno]{jarticle}% スタイルの指定
\begin{document}
% この行は注釈。次の 4 行でタイトル、著者名、日付を表示する
\title{レポート課題 X}
\author{1 年 2 組 99 番 桂田 祐史}
\date{2019 年 4 月 20 日}
\maketitle
\end{document}
```

特殊文字以外は `\begin{document}` と `\end{document}` の間に書けば表示される。

```
\documentclass[12pt,leqno]{jarticle}% スタイルの指定
\begin{document}
% この行は注釈。次の4行でタイトル、著者名、日付を表示する
\title{レポート課題 X}
\author{1年2組 99番 桂田 祐史}
\date{2019年4月20日}
\maketitle

ここにフッターの文字で書いたものは出力される。
\end{document}
```

実際には色々な記号が $\text{T}_\text{E}\text{X}$ の命令と解釈される特殊な文字となっている。プログラムなどを表示するには、`verbatim` 環境で利用するのが簡単である。

```
\documentclass[12pt]{jarticle}
\usepackage[a4paper,vscale=0.9,hscale=0.8]{geometry}
\usepackage{amsmath,amssymb}% そのうち必要になる
\usepackage[dvipdfmx]{graphicx}% そのうち必要になる

\begin{document}

\title{\TeX\ によるレポートの書き方}
\author{1年2組 99番 \quad 桂田 祐史}
\date{2019年4月22日}
\maketitle

\tableofcontents

\section{はじめに}
最初はこんな風に「はじめに」や「序」などの見出しのイントロを用意する。

\section{\TeX\ の解説本}
現在  $\text{LaTeX}$  を使うための定番の解説書は、奥村・黒木 \cite{奥村美文書} である。

 $\text{TeX}$  の開発者自身による解説としては、クヌース \cite{クヌース} がある。
基本的な設計思想を知りたい場合は必読書であるが、現在は購入が困難である。

\section{まとめ}
レポートや論文の最後は、
「まとめ」や「結論」や「将来の課題」などで締めるのが普通である。

必要最低限のことを覚えたら、後はどんどん使ってみるのが良い。
我流に陥らないように、あまり遅くならないうちに、
一度詳しい人に見てもらって添削してもらおうのがお勧め。

\begin{thebibliography}{99}
\bibitem{奥村美文書}
奥村晴彦, 黒木裕介,  $\text{LaTeX}$  美文書作成入門 改訂第7版, 技術評論社 (2017).
\bibitem{クヌース}
ドナルド・E. クヌース著, 鷺谷 好輝訳,
 $\text{TeX}$  ブック --- コンピューターによる組版システム,
アスキー (1992).
\end{thebibliography}
\end{document}
```

3.2 TeX のための準備作業

現象数理学科学生向けの MacBook Air では、TeXShop で使うための準備作業は済んでいるはず(?)。

以下は自分でやろうという人のための情報である。

- Mac で TeX を使うには、MacTeX⁶ をインストールして (TeXShop は MacTeX に含まれています)、ほんの少し設定するだけで OK (自分でやるなら「2020年のコンピューターノウハウ (Mac)」の「MacTeX 2020」⁷, 「MacTeX 2021 を待つ …M1 対応バージョン やって来ました!」⁸ を参考にしてください)。
- 大抵のことは TeXShop から使えるけれど、ターミナルの中からコマンドを入力して使いたい場合は、適当に PATH の設定をする必要がある。

MacTeX のインストーラーは、`/etc/paths.d` に次のようなファイルを作るので、それが有効になっていれば、自分では何もしないで済むかもしれない。

```
——— /etc/paths.d/TeX ———  
  
/Library/TeX/texbin
```

もし、うまく行かなかった場合は、自分で設定する必要がある。

```
——— bash を使う場合は ~/.profile (あるいは ~/.bash_profile) の末尾に次を加える ———  
  
export PATH=$PATH:/Library/TeX/texbin
```

```
——— tcsh を使う場合は ~/.tcshrc の末尾に次を加える ———  
  
set path=($path /Library/TeX/texbin)
```

(2019/4/20 記) 2018, 2019 年度学生の MacBook では、4 月中旬段階で TeXShop の設定が出来ていない? (事情は良く知らない) 自分で設定するには次を参考にしよう。

「2019 年度版 現象数理学科 Mac での TeX の設定」⁹

3.3 基本的な L^AT_EX の使い方 (基礎知識として)

実は TeX は、大小様々なソフトの連携プレーであると言える。TeXShop は色々なソフトを呼び出すことで役目を果たしている。以下、TeXShop を使わない方法 (TeXLive に含まれるソフトを利用する) を説明する。

1. `mylatex.tex` を (テキストエディット, `mi`, `emacs` などで) 開いて、適当な名前に変えて保存してから、編集 (執筆?) を始める。
(試すなら、課題 X のために、`kadaiX.tex` というファイルを作ることを勧める。)

⁶<http://www.tug.org/mactex/>

⁷<http://nalab.mind.meiji.ac.jp/~mk/knowhow-2020/node6.html>

⁸<http://nalab.mind.meiji.ac.jp/~mk/knowhow-2021/node2.html>

⁹<http://nalab.mind.meiji.ac.jp/~mk/knowhow-2019/node2.html>

2. なんとか.tex を dvi ファイルに変換 (コンパイル?) するには、ターミナルで、

```
$ platex なんとか.tex Enter
```

とする (下線部を入力する, 以下繰り返さない)。

3. なんとか.dvi を表示 (プレビュー) するには、いわゆるプレビューアを使う。例えば pxdvi をインストールしてあれば、ターミナルで

```
$ pxdvi なんとか.dvi & Enter
```

とする。

4. 人に渡す場合は、PDF ファイルにするとよい。なんとか.dvi を PDF ファイルに変換するには、コマンド・プロンプトで

```
$ dvi2pdf なんとか.dvi Enter
```

とする。これで なんとか.pdf が出来上がるはず。

(独白: そういえば、2018 年度の 1 年生は、emacs を使っている人が多いんだよね。YaTeX とか教えてあげると便利なのかなあ…macOS 付属の emacs だと面倒になる可能性が高いか?)

4 L^AT_EX 文書 .tex の書き方 — 入門

4.1 最初に覚えるべきこと

- ファイル名の拡張子は “.tex” とする。
- T_EX のコマンドには、先頭に「バックスラッシュ」 \ (文字コードは、16 進数で 0x5c, 10 進数で 92) をつける。
 - Windows の日本語環境では、「バックスラッシュ」は「円記号」 ¥ として表示される場合が多い。「バックスラッシュ」 \ も、Windows の「円記号」 ¥ も、見映えは違うけれど、文字コードは同じ (16 進数で 0x5c, 10 進数で 92) なので T_EX のコマンドにとっては同じである。
 - 現在の Mac では、Unicode を元にした UTF8 という文字コードが採用されているため、「バックスラッシュ」(0x5c) は \, 「円記号」(0xc2 0xa5) は ¥ と区別して表示される。Mac を購入した時点では、キーボードの [¥] キーをタイプすると、円記号 (0xc2 0xa5) が入力される。[option] キーを押しながら [¥] とタイプするとバックスラッシュ (0x5c) が入力される。[システム環境設定] [キーボード] [入力ソース] (の大分下の方) に、「“¥” キーで入力する文字」という項目があり、“¥(円記号)” と “\ (バックスラッシュ)” のどちらかを選択できる。T_EX や C 言語のプログラミングを行う場合は、円記号よりは、バックスラッシュの方を良く用いるので、筆者は “\ (バックスラッシュ)” を入力するように設定してある。
- パーセント記号 % から行末までは注釈になる。

- いつでも書くことになりそうな次の内容は、mylatex.tex に書いておいた (自分の氏名などを書き足すと良いかもしれない)。

```

mylatex.tex

\documentclass[12pt,leqno]{jarticle}
\usepackage[a4paper]{geometry}
\usepackage{amsmath,amssymb}
\usepackage[dvipdfmx]{graphicx}
\usepackage{otf}

\begin{document}

\title{}
\author{桂田 祐史}% ここは自分の名前にする
\date{2019 年月日}
\maketitle

\end{document}

```

これを読み込んで、別名で保存する (あるいはテンプレートに登録しておいて呼び出す)、というやり方を勧める。

- タイトルをつけるには、

```

\title{はじめての \TeX}% タイトル
\author{桂田 祐史} % 著者名
\date{2019 年 4 月 20 日 } % 日付 (省略すると組版した日になる)
\maketitle % これでタイトルを表示する

```

(date を省略すると、組版 (タイプセット) 実行時点の年月日が表示される。)

```

%\西暦 % 以前はこれが必要だったが、もう不要のはず
\title{はじめての \TeX}
\author{桂田 祐史}
\maketitle

```

(レポートなどを書く際は、最初に締切日を \date{} に書き込んでおいて、それを % で注釈にしておき、提出するときに注釈を外す、というやり方を勧めたい。締め切りが分かるし、ずっと後になっても「正式な」リリース日が分かる。)

- \begin{document} から \end{document} までの間に、ローマ字、数字などの “フツーの字” で書くとそのまま文書に入力される。いわゆる記号は注意が必要である。

細かいけど大事な話: 記号について

まず、そのまま入力&表示できるものとして、

! " ' () - = ' @ [] + * ; : ? , .

がある (マイナス - は、1文字の場合、2文字連続の場合、3文字連続の場合で、それぞれハイフン `-`, en-dash `—`, em-dash `---` となるので、そのまま入力できるものと考えた方がいいかもしれない。もっとも通常、マイナスは数式中に現われるものだから「`-$-$` と書く」と覚えるべきかも。)

一方、シャープ # などは、そのまま入力したのではダメで、これはバックスラッシュ \ を前につけて \# と入力する必要がある (\ でエスケープする、という)。同様にエスケープする必要がある文字としては、

\$ % & _ { }

がある。

\$ と \$ で囲んで、数式モードで扱うべき文字としては

| < > -

がある。

難しいのは次の 3 文字で、これを表示するには、(右側に書いた) 専用のコマンドを用いなければいけない。

```
~ \textasciitilde
\ \textbackslash
^ \textasciicircum
```

日本人専用の応急処置として、難しい文字の入力には漢字を使う、という手がある (やや幅広になってしまうけれど)。# \$ % & _ { } | < > ~ \ という感じで簡単。

4.2 改行と空白 (最低限の注意)

意外と難しいので¹⁰、ある程度 T_EX の説明が進んでから詳しく説明する (と言って、例年さぼっています)。ここではごく基本的なことから、「予告」に止める。

- .tex の中にいくら空白を続けても、一つ空白を入れたのと同じで、小さな空きができるだけ。

```
This is      a                               pen.
```

は

```
This is a pen.
```

となる。空白を明示的に入力するために、\quad などのコマンドがある。数式モードでは微調整用のコマンドがたくさんある。

- 連続した改行は「空行」と呼び、パラグラフ (段落?) を変更するという意味になり (\par と同じ)、改行されて、次の文の先頭に空きができる (いわゆる段落先頭の字下げ (indentation))。連続した空行は 1 つの空行と同じことになる。
- 英文中の一つの改行 (空行でないもの) は、一つの空白と同等。日本語文中の一つの改行 (空行でないもの) は、無視される。(不正確な言い方だが…)

```
I
love
you.
弁慶がな
ぎなたを
```

は

¹⁰少し大げさなようだが、T_EX の設計思想に係わることなので。

I love you. 弁慶がなぎなたを

となる。

- 強制的な改行は \\ だが、初心者が使いたくなるケースの 95% は誤用である (卒研で君達の先輩を相手にしたときの経験則)。

「(数式・表でないところで) **強制改行は極力使わない**」

と考えることを勧める。

4.3 文字の大きさや書体

実は結構複雑である。ここでは (9割の要求に応えれば良いことにして) 簡単に済ませる。

4.3.1 文字の大きさ

文字の大きさを変えるには、以下のようなコマンドがある。

- `\tiny`
- `\scriptsize`
- `\footnotesize`
- `\small`
- `\normalsize`
- `\large`
- `\Large`
- `\LARGE`
- `\huge`
- `\Huge`

```
{\tiny a}  
{\scriptsize a}  
{\footnotesize a}  
{\small a}  
{\normalsize a}  
{\large a}  
{\Large a}  
{\LARGE a}  
{\huge a}  
{\Huge a}
```


a a a a a a a a a

もっと大きくしたい? `\scalebox{}{}` を使う手がある。

```
\usepackage{graphicx}% graphicx の機能なので、これが必要
\begin{document}
...

\scalebox{10.0}{a}
```

a

4.3.2 文字の書体の指定

最初のうちは、こういうことに凝らないことを勧めたいけれど。欧文書体の場合は

- `\textrm{} (普通の) ローマン体 abcABC`
- `\textit{} イタリック体 abcABC`
- `\textsf{} サンセリフ体 abcABC`
- `\texttt{} タイプライター体 abcABC`
- `\textbf{} ボールド体 abcABC`
- `\textsc{} スモールキャピタル体 ABCABC`
- `\textsl{} スラント体 abcABC`

日本語の場合は、`\textgt{} でゴシック、\textmc{} で明朝。普通は明朝なので、`

`太くしたいときは (欧文書体と同様に) \textbf{} を用いる`

でも良いかもしれない。

```
桂田です。 \textgt{桂田です。} \textmc{桂田です。} \textbf{桂田です。}
```

桂田です。 **桂田です。** 桂田です。 **桂田です。**

(この例は、WWW では左から二番目がゴシック体で表示されない。)

5 簡単な数式

5.1 数式モード

数式は「数式モード」の中で書く。数式モードには次の二つがある。

1. 文中の数式 (**インライン数式**) は、ドル記号 $\$$ ではさんでかく。

ピタゴラスの定理から $\$a^2+b^2=c^2\$$ が成り立つ。

ピタゴラスの定理から $a^2 + b^2 = c^2$ が成り立つ。

2. 数式だけの行 (**ディスプレイ数式**) を作るには、色々な命令があるが、もっとも基本的なものは、 $\backslash[$ と $\backslash]$ ではさむもので、例えば

ピタゴラスの定理から
 $\backslash[$
 $a^2+b^2=c^2$
 $\backslash]$
がなりたつ。

のようになると

ピタゴラスの定理から
$$a^2 + b^2 = c^2$$

がなりたつ。

となる。式番号をつけるには `equation` 環境というものを用いて、

ピタゴラスの定理から
 $\backslashbegin{equation}$
 $a^2+b^2=c^2$
 $\backslashend{equation}$
がなりたつ。

のように書く。

数式番号はデフォルトでは右側につくが、`\documentclass[12pt,leqno]{jarticle}` のように、`leqno` (left equation number) を指定してある場合は、左側につく。

ピタゴラスの定理から
(1)
$$a^2 + b^2 = c^2$$

がなりたつ。

5.2 カッコ (parenthesis (), brace { }, bracket [])

丸い括弧 () とカギ括弧 [,] は普通に入力できる。{ , } は前に \ をつける。

余談: 日本では、(), { }, [] の順に使うが、英語圏では (), [], { } の順に使う。

```
\[
  \{[(a+b)+c]+d\}
\]
```

とすると

$$\{[(a+b)+c]+d\}$$

となる。カッコの大きさを調節するには、\left と \right で挟む場合が多い。

```
\[
  \left[
    \left(x-x_0\right)^2+\left(y-y_0\right)^2
  \right]^{1/2}
\]
```

$$\left[(x-x_0)^2+(y-y_0)^2\right]^{1/2}$$

やや脱線気味だが、最近は \left と \right の間に \middle というのを使えるようになった。

```
\[
  A=\left\{\frac{1}{n}\middle| n\in\mathbb{N}\right\}.
\]
```

$$A = \left\{ \frac{1}{n} \middle| n \in \mathbb{N} \right\}.$$

こうすると | の前後に適当な空白が入らず、バランスが悪い。棒を高くする必要がなければ、\mid を使えば良いのだが、\mid は \middle で使えない。

| を関係演算子扱いしつつ、\middle で高さを伸ばすには、次のようにすると良い (<http://tex.stackexchange.com/questions/5502/how-to-get-a-mid-binary-relation-that-grows>)。

```
\newcommand{\relmiddle}[1]{\mathrel{\middle#1\mathrel{}}}% マクロ定義

\[
  A=\left\{\frac{1}{n}\relmiddle| n\in\mathbb{N}\right\}.
\]
```

$$A = \left\{ \frac{1}{n} \relmiddle| n \in \mathbb{N} \right\}.$$

5.3 空白 (スペース)

数式モード中は、たくさんの空白用コマンドがある¹¹。

```
\[
a\,a\;a\ a\quad a\qquad a
\]
```

a a a a a a

空白を詰めることも必要になる。\`\!`で詰まる (マイナスの空白)。

```
\[
\int\int f(x,y)\;dxdy=\int\!\!\!\!\int f(x,y)dxdy
\]
```

とすると

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

となる (左辺と右辺の積分記号の間隔を比べよう)。

\iint を使うときは、たくさん使うことが多いので、後で説明するマクロを定義することを勧める。

—— フライングですが ——

```
% プリアンブル (\begin{document} の前) に
\newcommand{\dint}{\int\!\!\!\!\int}
...
\[
\int\int f(x,y)\;dxdy=\dint f(x,y)dxdy
\]
```

5.4 色々な記号

5.4.1 ギリシャ文字

`\` の後にローマ字 (ラテン文字) で読みを書くことでギリシャ文字が書ける。

```
\[
\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iotaota\kappa\lambdaambda\mu\nu\xi
% omicron は o と字の形が同じなので \omicron はない
\pi\rho\sigma\tau\upsilonlilon\phi\chi\psi\omega
\]
```

とすると

¹¹`\quad` (=quadrat) 印刷用語で空白の詰め物 (広辞苑によると、「組版の際に、印刷する必要のない余白部を埋めるために組み込むもの。」) だそうである。字と字の間に入れるのが「スペース」、大きな余白に入れるのが「クワタ」であるとか。) の一種。「クワタ」を見ると、個人的には、焼き鳥の「ハツ」 (heart) を思い出してしまう…

αβγδεζηθικλμνξπρστυφχψω

となる。

(「alpha, beta, … なんて知らない」 — そういう人が多いと想像するけれど、このように綴るのは、どうも英語圏の常識みたいなので、覚える価値はあると思う。)

なお、

```
\[
  \varepsilon\vartheta\varpi\varrho\varsigma\varphi
\]
```

とすると、

εθπρςφ

`\varepsilon`, `\varphi` は良く使うかも。

大文字のギリシャ文字は、先頭のローマ字を大文字にすればよい。例えば

```
\[
  \Gamma \Delta \Theta \Lambda \Xi \Pi \Sigma \Upsilon \Phi \Psi \Omega
  \Psi \Omega
\]
```

とすると

ΓΔΘΛΞΠΣΥΦΨΩ

となる (これ以外は、ローマ字の大文字と同じ。例えば α の大文字は A で良い。)

数式で使われる文字は、字体をイタリックにする場合が多いが、ギリシャ文字の大文字をイタリックにするには、`\mathit{}` を用いる。

```
\[
  \mathit{\Gamma \Delta \Theta \Lambda \Xi \Pi \Sigma \Upsilon \Phi \Psi \Omega}
  \Psi \Omega
\]
```

ΓΔΘΛΞΠΣΥΦΨΩ

あるいは `\varGamma` のような “var” が先頭についたコマンドを用いる。

```
\[
  \varGamma \varDelta \varTheta \varLambda \varXi \varPi \varSigma
  \varUpsilon \varPhi \varPsi \varOmega
\]
```

ΓΔΘΛΞΠΣΥΦΨΩ

5.4.2 集合と論理

```
\[
a\in A\subset B,\quad
C\supset D,\quad
a\notin A,\quad
C\not\supset D,\quad
A\cup B, A\cap B, A\setminus B=\emptyset,\quad
\bigcup_{i=1}^{\infty} A_i=\bigcap_{i=1}^{\infty} B_i
\]
```

$$a \in A \subset B, \quad C \supset D, \quad a \notin A, \quad C \not\supset D, \quad A \cup B, A \cap B, A \setminus B = \emptyset, \quad \bigcup_{i=1}^{\infty} A_i = \bigcap_{i=1}^{\infty} B_i$$

空集合は `\varnothing` を使う人も多い。

\in (`\in`) の逆向きが \ni (`\ni`) であるのは苦し紛れっぽいけど。(`\supset` も最初見たときは苦し紛れと思ったのだけれど、subset (部分集合) の反対語は superset なので、正しい言葉遣いなのだった。) 包含関係で等号をつけるつけないは、普通の大小関係の不等号 $<$ と同じ感じ。

```
\[
A\subteq B,\quad
A\subseteq B,\quad
A\subsetneq B,\quad
A\subsetneqq B.
\]
```

$$A \subseteq B, \quad A \subseteq B, \quad A \subsetneqq B, \quad A \subsetneqq B.$$

論理の記号: and \wedge は `\wedge` あるいは `\land`, or \vee は `\vee` あるいは `\lor`, not \neg は `\neg` あるいは `\lnot` とする (1 は logic あるいは logical の頭文字なんだろう)。

```
\[
\neg(P\wedge Q)\equiv \neg P\vee \neg Q.
\]
```

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q.$$

矢印のところで説明済みだが、 \Leftrightarrow は `\Leftrightarrow`, \Rightarrow は `\Rightarrow`

5.5 上つき添字、下つき添字、それと積分&シグマ

a^2 は `a^2` とする。 a_n は `a_n` とする。 2^{2^n} は `2^{2^n}` とする。
積分やシグマなどもこの応用で、

```
\[
\lim_{R\to\infty}\int_a^R f(x)\;dx=\sum_{n=1}^{\infty} a_n
\]
```

とすると

$$\lim_{R\rightarrow\infty}\int_a^R f(x) dx = \sum_{n=1}^{\infty} a_n$$

5.6 分数

分数は `\frac{}{}` コマンドを使う。英語流に分子を先に書く¹²。

```
\[
\frac{a+b}{c}=\frac{1}{2}
\]
```

は

$$\frac{a+b}{c} = \frac{1}{2}$$

となる。

分数や積分、和の記号など、インライン数式では小さく組版されるが、ディスプレイ数式と同じように大きく組版するには、`\displaystyle` コマンドを用いる。

`\frac{a+b}{c}=\frac{1}{2}` は小さいので、
`\displaystyle\frac{a+b}{c}=\frac{1}{2}` とすると大きくなる。

は

$\frac{a+b}{c} = \frac{1}{2}$ は小さいので、 $\frac{a+b}{c} = \frac{1}{2}$ とすると大きくなる。

実は `\dfrac` という命令もある (`amsmath` パッケージが必要)。

なお、`\displaystyle` は長くて入力面倒なので、後述するマクロなどを利用する人が多いようである。`\begin{document}` の前に

```
\newcommand{\dsp}{\displaystyle}
```

と定義しておく、以下 `\dsp` で、`\displaystyle` としたのと同じになる。

$\text{T}_{\text{E}}\text{X}$ の分数の横棒は、日本人には“短め”である。長くしたい場合は、分母か分子 (横幅の多い方) に適当なスペースを入れると良い。

¹² $\frac{a}{b}$ は、日本語では「 b ぶんの a 」であるが、英語では「 a over b 」と読む。

```
\[
\frac{1}{2}+\frac{1}{3}=\frac{1}{\;2\;}+\frac{1}{\;3\;}.
\]
```

$$\frac{1}{2} + \frac{1}{3} = \frac{1}{2} + \frac{1}{3}.$$

分母・分子と分数の横棒がくっつきすぎと感ずることがある。分子を `\raise` 長さ ボックスで持ち上げ、分母を `\lower` 長さ ボックスで下げて微調整する(?)。

```
\[
\frac{\kakko{ア}}{\kakko{イ}}
=
\frac{\raise0.8ex\hbox{\;$\;\kakko{ア}\;$}}
{\lower1ex\hbox{\;$\kakko{イ}$}}
\]
```

$$\frac{\boxed{\text{ア}}}{\boxed{\text{イ}}} = \frac{\boxed{\text{ア}}}{\boxed{\text{イ}}}$$

5.7 sin などの「作用素」

`\sin x` の ‘s’, ‘i’, ‘n’ はイタリックでない¹³、いわゆる立体 (ローマン体) で、`\sin` と `x` の間に適度のスペースがあることに注意。こういうものには、専用のコマンドが用意されている場合が多い。

```
\[
\sin x=\log y=\max A
\]
```

$$\sin x = \log x = \max A$$

単に `\log x` のように書くと `\log x` となってしまう (これでは `l`, `o`, `g`, `x` の積にしか見えない)。なぜだか考えてみることを勧める。

マクロというものを使って、自分でこの種のコマンドを作ることも出来る。`\begin{document}` の前に (「プレアンブルに」という)

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

と書いておくと (呪文のようですが、“grad” のところだけ変えれば良い、と覚えましょう)、`\grad` というコマンドが定義できる。

¹³普通、数式中のローマ字は、`x` のようにイタリック (斜めに傾いているのが特徴) で表すことに注意。

5.8 矢印

<code>\to</code>	<code>\mapsto</code>	<code>\leftarrow</code>	<code>\rightarrow</code>	<code>\leftrightharrow</code>	<code>\longleftarrow</code>	<code>\Leftarrow</code>
→	↦	←	→	↔	←←←	⇐

```
\[
\to \quad \mapsto \quad \quad
\leftarrow \quad \quad \Leftarrow \quad \quad
\longleftarrow \quad \quad \Longleftarrow \quad \quad
\leftrightharrow \quad \quad \Leftrightharrow \quad \quad
\longleftarrow \quad \quad \Longleftarrow
\]
```

とすると

→ ↦ ← ⇐ ←←← ⇔ ⇌ ↔ ⇔⇔

もちろん left の反対の right もある。

上下、斜めの矢印については、

```
\[
\uparrow \quad \downarrow \quad \quad \Uparrow \quad \quad \Downarrow \quad \quad
\updownarrow \quad \quad \Updownarrow \quad \quad
\nearrow \quad \quad \nwarrow \quad \quad \searrow \quad \quad \swarrow
\]
```

とすると

↑ ↓ ↗ ↘ ↕ ⇕ ↗ ↖ ↘ ↙

`\to` → と `\mapsto` ↦ はそのまま覚え、それ以外は命名ルールを理解して覚えることを勧める。
(`\nearrow`, `\nwarrow`, `\searrow`, `\swarrow` は、north-east, north-west, south-east, south-west だと思っているのだけど、本当かなあ?)

5.9 点

```
\[
\cdot \quad \quad \cdots \quad \quad \ldots \quad \quad \ddots \quad \quad \vdots
\]
```

は順に、真ん中に一つの点、真ん中に3つの点、下に3つの点、斜めに3つの点、垂直方向に3つの点となる (c は center, l は low, d は diagonal (対角線の), v は vertical (垂直の))。

· ∙ ∙

5.10 不等式

等号のつかないものはそのまま $<$, $>$ を使うとよい。 \leq は `\le` とし、 \geq は `\ge` とする¹⁴。

```
\[
a<b\le c\ge d
\]
```

$$a < b \leq c \geq d$$

なお `\ll`, `\gg` で \ll , \gg となる。また、等号 $=$ の否定 \neq は `\ne` と入力する。

平行線を含む \leq , \geq は、それぞれ `\leqq`, `\geqq` と入力する (これは次の項で説明する AMS 拡張であるので、プリアンブルに `\usepackage{amssymb}` とする必要がある)。

5.11 その他の記号

<code>\ </code> 	<code>\pm</code> ±	<code>\mp</code> ∓	<code>\times</code> ×	<code>\div</code> ÷	<code>\sim</code> ~	<code>\simeq</code> ≈	<code>\fallingdotseq</code> ≐	<code>\leqq</code> ≤	<code>\geqq</code> ≥
<code>\nabla</code> ∇	<code>\triangle</code> △	<code>\partial</code> ∂	<code>\forall</code> ∀	<code>\exists</code> ∃	<code>\infty</code> ∞	<code>\propto</code> ∝	<code>\perp</code> ⊥		
<code>\angle</code> ∠	<code>\langle</code> ⟨	<code>\rangle</code> ⟩							

`\fallingdotseq` \equiv のような AMS (アメリカ数学会, American Mathematical Society) 由来のフォントには、プリアンブルに `\usepackage{amssymb}` と書くことが必要です。例えば次のようにします。

```
...
\usepackage{amssymb}% AMS で用意したシンボルのフォント
...
\begin{document}
...
\[
\| \quad \pm \quad \mp \quad \times \quad \div \quad \quad
\sim \quad \simeq \quad \fallingdotseq \quad \leqq \quad \geqq \quad
\nabla \quad \triangle \quad
\partial \quad
\forall \quad \exists \quad \infty \quad \propto \quad \perp \quad
\angle \quad \langle \quad \rangle
\]
```

|| ± ∓ × ÷ ~ ≈ ≐ ≤ ≥ ∇ △ ∂ ∀ ∃ ∞ ∝ ⊥ ∠ ⟨ ⟩

ちなみに `\fallingdotseq` \equiv や `\partial` ∂ は長いので、筆者はマクロ (8.1 参照) を使って短い別名を定義してある¹⁵。

¹⁴多分、“less than or equal to” から `le`，“greater than or equal to” から `ge` となったのであろう。

¹⁵解析屋にとって、偏微分記号 ∂ は良く使うので…

5.12 行列、ベクトル、場合分けの {

行列や(縦)ベクトルでは、式(成分)を「きれいに並べる」必要がある。このためには、array 環境や matrix 環境を用いる(縦ベクトルは、列の個数が 1 である行列とみなす)。また括弧(と) (あるいは [,], {, }) は \left と \right を使って拡大する。

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

は

array 環境を用いて行列を書く

```
\[
\left(
\begin{array}{cc}
a & b \\
c & d
\end{array}
\right)
\left(
\begin{array}{c}
x \\
y
\end{array}
\right)
\]
```

または AMS 拡張に含まれる pmatrix 環境を用いても良い。

pmatrix 環境を用いて行列を書く

```
\documentclass[12pt,leqno]{jarticle}
...
\usepackage{amsmath}% プリアンブルに書く
...
\begin{document}
...
\[
\begin{pmatrix}
a & b \\
c & d
\end{pmatrix}
\begin{pmatrix}
x \\
y
\end{pmatrix}
\]
```

pmatrix 環境の方が使い方は簡単だが¹⁶、array 環境は左寄せ (l)、中央揃え (c)、右寄せ (r) など細かい制御ができる。

なお

$$|x| = \begin{cases} x & (x \geq 0 \text{ のとき}) \\ -x & (x < 0 \text{ のとき}) \end{cases}$$

も似た感じで出力できる。

```
\[
|x|=
\left\{
\begin{array}{rl}% 1 でなく l (エルLの小文字) left の頭文字なので
x & \text{{($x\ge 0$ のとき)}}\\
-x & \text{{($x<0$ のとき)}}
\end{array}
\right. % 右側は括弧なし (ドット . が重要)
\]
```

他に cases 環境というものもあるが、右寄せ、中央寄せなど細かい指定は出来ない。

```
\[
|x|=
\begin{cases}
x & \text{{($x\ge 0$ のとき)}}\\
-x & \text{{($x< 0$ のとき)}}
\end{cases}
\]
```

5.13 数式中の言葉

数式中に日本語や英語で説明の言葉を書きたいことがある。そういう場合は、`\mbox{}` や、`\text{}` を使う (後者は文字の大きさを回りの式に合わせて調節してくれる)。

```
\usepackage{amsmath}% \text{} に必要
...
\[
f(x)=\log x\quad\mbox{{($x$ は正の実数)}},\quad
\zeta(s)=\prod_{\text{{$p$ は素数}}}\frac{1}{1-\frac{1}{p^s}}
\]
```

$$f(x) = \log x \quad (x \text{ は正の実数}), \quad \zeta(s) = \prod_{p \text{ は素数}} \frac{1}{1 - \frac{1}{p^s}}$$

¹⁶なお、括弧の形の違う行列を作る bmatrix, Bmatrix 環境、括弧なしの matrix 環境等もある。

5.14 数式の縦揃え

複数行に渡る数式を書く場合、等号など適当な位置で揃えたいことがある。色々なやり方があるが、とりあえず `align`, `align*` 環境を覚えておきましょう。

—— 等号の位置を揃える ——

```
\begin{align*}
x&=1, \\
f(x)&=10.
\end{align*}
```

$$\begin{aligned} x &= 1, \\ f(x) &= 10. \end{aligned}$$

—— 行の先頭を揃える ——

```
\begin{align*}
&x=1, \\
&f(x)=10.
\end{align*}
```

$$\begin{aligned} &x = 1, \\ &f(x) = 10. \end{aligned}$$

アスタリスク (*) なしの `align` 環境は数式番号がつく。

—— 等号の位置を揃える (数式番号つき) ——

```
\begin{align}
(2) \quad x&=1, \\
(3) \quad f(x)&=10.
\end{align}
```

$$\begin{aligned} (2) \quad x &= 1, \\ (3) \quad f(x) &= 10. \end{aligned}$$

途中で文章をはさむには、`\intertext{}` コマンドを用いる。

(ちなみに単独の式で式番号をつけるには、`\[` と `\]` の代わりに `\begin{equation}` と `\end{equation}` (equation 環境) を用いる。

```
\begin{equation}
(4) \quad 3^2+4^2=5^2.
\end{equation}
```

$$(4) \quad 3^2 + 4^2 = 5^2.$$

`align` 環境の複数の式をひとまとめにして、副番号をつける `subequations` 環境というものがある。これについては、[5.17](#) を見よ。

5.15 下線、上線、矢印など

```
\[
  \underline{ABC}, \quad
  \overline{ABC}, \quad
  \overrightarrow{ABC}, \quad
  \overleftarrow{ABC}, \quad
  \overbrace{ABC}, \quad
  \underbrace{ABC}
\]
```

\underline{ABC} , \overline{ABC} , \overrightarrow{ABC} , \overleftarrow{ABC} , \overbrace{ABC} , \underbrace{ABC}

`\overbrace{}` で上に注釈をつけたい場合は、いわゆる上付き添字としてやれば良い。

```
\newcommand{\R}{\mathbb{R}}
\[
  \R^n := \overbrace{\R \times \cdots \times \R}^{\text{\$n\$個}}
\]
```

$$\mathbb{R}^n := \overbrace{\mathbb{R} \times \cdots \times \mathbb{R}}^{n \text{ 個}}$$

5.16 数式番号について

複数の式をひとまとめして、副番号をつける `subequations` 環境というものがある。

```
\begin{subequations}
\begin{align}
& \frac{dx}{dt} = f(t, x) \\
& x(0) = x_0
\end{align}
\end{subequations}
```

(5a) $\frac{dx}{dt} = f(t, x)$

(5b) $x(0) = x_0$

- `jreport` や `jbook` クラスのように章が存在する場合、数式番号はデフォルトでは章ごとに番号付けられる。例えば 1 章の 2 番目の式に (1.2), 3 章の 4 番目の式に (3.4) という番号がつく。
- `jarticle` のように章が存在しない場合、数式番号はデフォルトでは、その文書全体での通し番号となる。例えば 3 番目の式は (3) となる。
- 数式番号に「第○節の第△番目の式」というように節の番号をつけたい場合は、

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

と書けば良い。これは `amsmath` の機能なので、`\usepackage{amsmath}` の後に書くこと。`jarticle` クラスの場合、第 2 節の 3 番目の式に (2.3) という番号がつく。また `jreport`, `jbook` クラスのように章が存在する場合、1 章 2 節の 3 番目の式は (1.2.3) という番号がつく。(実は `equation` を `figure` に変えると、図の番号にも節の番号をつけられる。)

- 数式の番号は `\theequation` についているので、それを自分で定義し直すこともできる。

例えば、`jarticle` クラスで、`\numberwithin{equation}{section}` と同じことをするには

```
\renewcommand{\theequation}{\thesection.\arabic{equation}}
\makeatletter
\@addtoreset{equation}{section}
\makeatother
```

とする。式の番号を節が変わるごとにリセットする操作が必要であるのに注意が必要である。

5.17 misc

- ^{def.} どうやって出しますか? 昔は `\stackrel{}{}`, 今だと `\overset{}{}` を使えだとか。

```
\[
f(x)\stackrel{\mathrm{def.}}{=}x^2+2x+3,\quad
g(x)\overset{\mathrm{def.}}{=}3x^2+2x+1,\quad
h(x)\underset{\mathrm{def.}}{=}\sin x.
\]
```

$$f(x) \stackrel{\text{def.}}{=} x^2 + 2x + 3, \quad g(x) \stackrel{\text{def.}}{=} 3x^2 + 2x + 1, \quad h(x) \underset{\text{def.}}{=} \sin x.$$

- $\lim_{\substack{y=kx \\ (x,y) \rightarrow (0,0)}}$ は上 `\atop` 下 や `\genfrac{}{}{0pt}{1}{上}{下}`

```
\[
\lim_{y=kx\atop (x,y)\to(0,0)}\frac{x y}{x^2+y^2}
=
\lim_{\genfrac{}{}{0pt}{1}{y=kx}{(x,y)\to(0,0)}}\frac{x y}{x^2+y^2}
\]
```

$$\lim_{\substack{y=kx \\ (x,y) \rightarrow (0,0)}} \frac{xy}{x^2 + y^2} = \lim_{\substack{y=kx \\ (x,y) \rightarrow (0,0)}} \frac{xy}{x^2 + y^2}$$

6 文書の構造など

6.1 chapter, section, subsection, paragraph など

ある程度以上長い文書は、章や節などのまとまりがある。L^AT_EX では、以下のような命令がある。

```
\part{見出し}          第  $x$  部
\chapter{見出し}       第  $x$  章
\section{見出し}       第  $x$  節
\subsection{見出し}
\subsubsection{見出し}
\paragraph{見出し}    段落
\subparagraph{見出し}
```

jarticle スタイルでは、part, chapter は使用できない (大抵のレポートは section で十分のはず)。chapter が使いたい場合、jreport や jbook スタイルを用いる。part が使いたい場合、jbook スタイルを用いる。

———— jbook スタイルを使うには…最初に指定する ————

```
\documentclass[12pt,leqno]{jbook}
```

6.2 自動目次生成

前項の命令を使って、chapter や section を作ってれば

```
\tableofcontents
```

という命令で自動的に目次を生成できる — 非常に便利であり、活用することを強くお勧め出来る。

\section*{} などを使った場合、そのままでは目次に載らないが、直後に

```
\addcontentsline{toc}{section}{目次に載せたい見出し}
```

と書くことで目次に載せることが出来る。

6.3 参考文献表

レポートや論文では、参考にした文献や論文を文書の末尾に並べたリストを作るのが普通である。L^AT_EX で参考文献表を作る方法はいくつかあるが (私は普段は pbibtex を使っている)、ここでは、もっとも単純な方法を紹介しよう。

次の例では、2冊の本からなる参考文献表を作成してある。

```
\begin{thebibliography}{99}% 99 ほどにかくこう書く
\bibitem{奥村黒木美文書}
  奥村晴彦, 黒木裕介, \LaTeXe\ 美文書作成入門 改訂第7版, 技術評論社 (2017).
\bibitem{クヌース}
  ドナルド・E. クヌース著, 鷺谷 好輝訳,
  \TeX\ ブック --- コンピューターによる組版システム,
  アスキー (1992).
\end{thebibliography}
```


参考文献

- [1] 奥村晴彦, $\LaTeX 2_{\epsilon}$ 美文書作成入門 改訂第5版, 技術評論社 (2010).
- [2] ドナルド・E. クヌース著, 鷺谷 好輝訳, \TeX ブック — コンピューターによる組版システム, アスキー (1992).

例えば、最初の本を引用するには、`\cite{ }` コマンドを用いて、

奥村・黒木 `\cite{奥村黒木美文書}` は、日本語による \TeX の定番の解説書である。

のようになる。

—— 出来上がりの例 ——

奥村・黒木 [1] は、日本語による \TeX の定番の解説書である。

`jreport`, `jbook` クラスで `thebibliography` 環境を使うと、「関連図書」という題目になる。これを「参考文献」等、自分の好きなものに変えるには、

```
\renewcommand{\bibname}{参考文献}
```

のように `\bibname` の再定義をすれば良い。

6.4 この節で解説した項目の使用例

例

```
\documentclass[12pt]{jarticle}
\usepackage[a4paper,vscale=0.9,hscale=0.8]{geometry}
\usepackage{amsmath,amssymb}% そのうち必要になる
\usepackage[dvipdfmx]{graphicx}% そのうち必要になる
\usepackage{otf}

\begin{document}

\title{\TeX\ によるレポートの書き方}
\author{1年2組 99番 \quad 桂田 祐史}
\date{2019年4月20日}
\maketitle

\tableofcontents

\section{はじめに}
最初はこんな風に「はじめに」や「序」などの見出しのイントロを用意する。

\section{\TeX\ の解説本}
現在 \LaTeX\ を使うための定番の解説書は、奥村 \cite{奥村美文書} である。

\TeX\ の開発者自身による解説としては、クヌース \cite{クヌース} がある。
基本的な設計思想を知りたい場合は必読書であるが、現在は購入が困難である。

\section{まとめ}
レポートや論文の最後は、
「まとめ」や「結論」や「将来の課題」などで締めるのが普通である。

必要最低限のことを覚えたら、後はどんどん使ってみるのが良い。
我流に陥らないように、あまり遅くならないうちに、
一度詳しい人に見てもらって添削してもらおうのがお勧め。

\begin{thebibliography}{99}
\bibitem{奥村黒木美文書}
  奥村晴彦, 黒木裕介, \LaTeXe\ 美文書作成入門 改訂第7版, 技術評論社 (2017).
\bibitem{クヌース}
  ドナルド・E. クヌース著, 鷺谷 好輝訳,
  \TeX\ ブック --- コンピューターによる組版システム,
  アスキー (1992).
\end{thebibliography}
\end{document}
```

出来上がりは <http://nalab.mind.meiji.ac.jp/~mk/labo/text/sample2019.pdf> で確認出来る。

1ページの文書なので、目次のありがたみがピンと来ないかも知れないが(1, 2, 3節とも開始ページは1なので)。

7 もう一段ていねいに

準備中が多いけれど、店舗(WWW ページ)は今のうちに用意しておく。

急いでいる人は、奥村・黒木 [1] のような書籍を購入したり、ネットで調べたりして下さい。

7.1 索引

(準備中)

7.2 相互参照

相互参照 `\label{}` と `\ref{}`

式や章・節、定理の番号など、 $\text{T}_{\text{E}}\text{X}$ が自動的につける番号については、`\label{文字列}` でラベルをつけておいて、後で `ref\{文字列}` で参照出来ます。

7.3 脚注 (フットノート)

(準備中)

7.4 箇条書き (enumerate, itemize, description 環境)

(準備中)

7.5 定理環境

(準備中)

8 $\text{T}_{\text{E}}\text{X}$ のマクロ機能、パッケージ機能の紹介

8.1 マクロ

既に紹介したように、プリアンブルに

————— `gradient` 作用素の記号を定義する —————

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

と書いておくと、`\grad` というコマンドが定義できる。これは $\text{T}_{\text{E}}\text{X}$ のマクロという機能を使っている。

マクロは、簡単な部分だけでも、便利に使うことが出来る。例えば `\displaystyle` コマンドや `\varepsilon` コマンドのように、長くて入力が面倒なコマンドに、短い別名をつけるために使うことが出来る。そのためには、プリアンブルに例えば

————— `\displaystyle`, `\varepsilon` を手短かに `\dsp`, `\eps` で —————

```
\newcommand{\dsp}{\displaystyle}
\newcommand{\eps}{\varepsilon}
```

のように書けば良い。

マクロでは、いわゆる引数を用いることができる。 2×2 の行列 $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ は、例えば

```
\left(
\begin{array}{cc}
1 & 2 \\
3 & 4
\end{array}
\right)
```

として組版できるが、

— 2 × 2 行列用のマクロ —

```
\newcommand{\gyouretsu}[4]{
\left(
\begin{array}{cc}
{#1} & {#2} \\
{#3} & {#4}
\end{array}
\right)
}
```

とマクロ `\gyouretsu` を定義しておく (行列の 4 つの成分が引数として与えられる)、

```
\gyouretsu{1}{2}{3}{4}+\gyouretsu{5}{6}{7}{8}=
\gyouretsu{6}{8}{10}{12}
```

で $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$ が組版できる。

なお、マクロの名前には、ローマ字のみが使えます (`gyouretu22` のような文字列は使えません)。

実は、通常使っている \LaTeX そのものが、膨大なマクロの集成に他なりません。

8.2 パッケージ

\LaTeX である程度まとまったことをやりたい場合に、パッケージというものが用意されていることがある (中身は要するにマクロの集合である)。

パッケージは、プリアンブルで `\usepackage{}` コマンドを用いて読込む。

詳細は省略するが (自分で必要になってから調べれば良い)、以下筆者が良く利用しているものの名前をあげておく。

geometry パッケージ \TeX 文書で使う**紙の大きさや、余白の長さ**などを指定するのに、`geometry` パッケージ¹⁷ というものが便利である (`latex geometry.ins` で `geometry.sty` を生成する)。

```
\usepackage[a4paper]{geometry}
```

とか

```
\usepackage[a4paper,vscale=0.9,hscale=0.8]{geometry}
```

のように使う。

¹⁷<http://tug.ctan.org/tex-archive/macros/latex/contrib/geometry/>

amsmath, amssymb パッケージ 複雑な数式や、やや珍しい記号類の組版には、アメリカ数学会 (American Mathematical Society, AMS) が開発した amsmath, amssymb パッケージが威力を発揮する。

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

(参考 「amsmath パッケージユーザガイド (Version 2.1)」¹⁸)

graphicx パッケージ グラフィックスを取り込むための `\includegraphics{}` 命令が用意されている (使い方は後述する)。

```
\usepackage[dvipdfmx]{graphicx}
```

あるいは

```
\usepackage[dvips]{graphicx}% 昔は dvips を使っていたので
```

LaTeX Beamer パッケージ プレゼンテーション資料を TeX で作るために、色々なパッケージが開発されている。LaTeX Beamer パッケージはその一つである。このあたりは流行り廃りがあるので、自分が必要になったときに、WWW で検索すると良い。§12 を見よ。

ascmac パッケージ 円記号を組版する `\yen` や、枠で囲う `screen` 環境、見出しつきの枠で囲う `itembox` 環境などは、ascmac パッケージにある。

```
\usepackage{ascmac}
```

9 ソースプログラム等テキストファイルの L^AT_EX 文書への取り込み

(ここは書き換えるつもりです。)

例えばプログラミングがらみの課題のレポートを作る場合など、ソースプログラムやプログラムの実行結果を取り込みたくなる。

短いものは

verbatim 環境の利用

```
\begin{verbatim}
#include <stdio.h>
int main(void)
{
    printf("Hello, world\n");
    return 0;
}
\end{verbatim}
```

のように、.tex ファイルの中の、verbatim (“verbatim” は「言葉通りに」、「逐語的に」という意味の単語) 環境の中に入れてしまえばよいが、長いものや頻繁に変更を加えるものを扱うのは面倒である。

¹⁸<https://www.latex-project.org/help/documentation/amslatex-jpn.pdf>

そういうものは別途テキスト・ファイルにして、`moreverb` パッケージを組み込むと有効になる `\verbatiminput{}` コマンドや `\listinginput{}` コマンド (行番号つき) を使って取り込むとよい。

— hello.c, world.c を取り込む —

```
\documentclass[12pt,leqno]{jarticle}
\usepackage{moreverb}% パッケージを組み込む

\begin{document}
...
\verbatiminput[4]{hello.c}% hello.c は別途用意してあるとして。タブ幅4にする。
...
\listinginput{1}{world.c}% world.c は別途... 行番号を1から振る
...
\end{document}
```

(`\listinginput{}` では、タブ幅 (tab width) が指定出来ない?もしかすると `\def\verbatimtabsize{4}` のように原始的に指定出来るかも。)

念のため、以前勧めていた `verbatimfiles` パッケージの使い方を書いておく。

— **古いです!** 以前は `moreverb` の代わりに `verbatimfiles` を使っていました —

`verbatimfiles` パッケージを組み込むと有効になる `\verbatimfile{}` コマンドや `\verbatimlisting{}` コマンド (行番号つき) を使うとよいでしょう。

— hello.c を取り込む —

```
\documentclass[12pt,leqno]{jarticle}
\usepackage{verbatimfiles}% パッケージを組み込む (複数形の s がついている)

\begin{document}
...
\verbatimfile{hello.c}% hello.c は別途用意してあるとして
...
\end{document}
```

多くのソフトでタブ幅のデフォルトは 8 である (emacs でも Vim でも)。`expand` や `unexpand` のようなコマンドもそうしてある。だから、自分で何か設定を変えていなければ、タブ幅の指定は不要のはずである。

Vim の設定の話で良く `set tabstop=4` というのを目にするけれど、そういうのが流行っているのかな? (この辺は自分でいじる習慣がないので良く分からない)

10 画像の L^AT_EX 文書への取り込み

(書き換え中)

この節に書いてあることは現在 (2018/6/29) は、ちょっと古くなっている。EPS, JPEG 形式の話の前の方に配置してあるが、多くの場合 PNG や PDF を使う方が良いと思われる (写真は今でも JPEG かもしれないが…)。次項の「概要」を見れば済んでしまうかもしれない。

10.1 概要

L^AT_EX は、多くの人達の努力により、色々なグラフィックス・データを取り込めるようになっている。

具体的に何が出来るかは使用する印刷・表示用のドライバーに依存し、対応状況は結構頻繁に変化している。かなり良くなっていて、もう少しで誰でもトラブル・フリーで出来るようになる、その一手手前だろうか。運が悪いと「はまる」かもしれないが、そこでめげないように。

- (1) ドライバーを指定するオプションは最初に指定しておくのが良さそうである (ドライバーは他のパッケージとも関係するため、一番上でやっておくのが、混乱が生じにくい)。ドライバーの種類として、dvipdfmx, dvipdfm, dvips, dviout など色々ある。ずっと以前は Windows では dviout, UNIX では dvips というのが多かったが、最近の日本語環境では dvipdfmx を使うのが良いようだ。

```
\documentclass[12pt,...,dvipdfmx]{jarticle}
```

のように `\documentclass{}` のオプションで指定する。

- (2) グラフィックス取り込み用のパッケージとして、graphics, graphicx があるが、とりあえず graphicx で良い。

```
\usepackage{graphicx}
```

- (3) 画像ファイルを取り込みたいところで、

```
\includegraphics[オプション]{ファイル名}
```

とする。例えば

```
\includegraphics[width=15cm]{mygraph.png}
```

- 画像ファイルは、.tex ファイルと同じディレクトリか、その下に作ったサブディレクトリに置くと良い。
- 細かい注意：ファイル名は日本語を避ける方が無難である。特に macOS のファイル名の文字コードは、UTF8 の Normalization form D というもので、今のところ色々問題を引き起こす種になっている。自分で理解して克服するつもりがない限り、日本語を避けよう。
- ドライバーとして dvipdfmx を使う場合、取り込めるファイルのフォーマットは、PDF (.pdf), PNG (.png), JPEG (.jpg), EPS (.eps) など、色々ある。その他のフォーマットであっても、これらのどれかに変換することは難しくないで、実際上困ることはないと言って良い。
- includegraphics のオプションには、height= (高さ指定), scale= (倍率指定), angle= (回転角度指定), clip (はみ出した部分を切り取る), bb= (BoundingBox 情報の指定) などがある。回転する場合の原点の指定 origin= (指定できるのは c, t1, tr, b1, br)
- 画像の大きさ (BoundingBox 情報) は、PDF, PNG, JPEG の場合は、TeX の設定がきちんとされていれば自動的に取得される。それ以外に、includegraphics のオプションで

bb=左座標 右座標 下座標 上座標 単位はポイント (?)

のように直接指定することも可能である。

—— こんなふうに直接 BoundingBox 情報を与えられる ——

```
\includegraphics[width=10cm,bb=0 360 0 375]{photo0620.png}
```

EPS の場合は、内部に BoundingBox コメントとして含まれている場合が多い。 $[x_1, x_2] \times [y_1, y_2]$ の場合 `%%BoundingBox: x1 y1 x2 y2` とする。

—— BoundingBox コメントの例 ——

```
%%BoundingBox: 36 295 595 841
```

- `includegraphics` 命令で取り込んだ図は、`figure` 環境で配置するのが望ましい。
- ドライバーとして `dvips` を使う場合は、直接取り込めるファイルのフォーマットは、EPS (`.eps`) だけであるが、JPEG は `jpeg2ps` というコマンドで EPS にラップしてから取り込むことができる。

画像ファイルの BoundingBox 情報の自動取得の設定 ここでは、少し前までの相場を説明する。(多分現在ではここに書いてあることは意識する必要がないのと思う…)

以下、`myimage.png` を取り込む場合で説明する。`.png` のところは `.pdf`, `.jpg` などでも同様である。

`myimage.png` の BoundingBox 情報を得るため、 \TeX は外部のプログラムの力を借りて、BoundingBox 情報を書き込んだ `myimage.xbb` というファイルを生成し、 \TeX はそれを読み込んで必要な空白を作り、実際の画像の埋め込みはドライバー・プログラムに任せる、という処理の流れになっている。

実際は `extractbb` という外部プログラム (実は実体は `dvipdfmx`) を用いていた。手動で `myimage.xbb` を作るには、ターミナルから

```
extractbb mygraph.png
```

のように実行する。

これを自動化するために、設定ファイル `texmf.cnf` 中の `shell_escape_commands=` に `extractbb` を含めておく。

—— `texmf.cnf` の `shell_escape_commands=` の設定例 ——

```
shell_escape_commands = \  
bibtex,bibtex8,bibtexu,pbibtex,upbibtex,biber,\  
kpsewhich,\  
makeindex,mendex,texindy,\  
mpost,pmpost,upmpost,\  
repstopdf,epspdf,extractbb,\  
(空行)
```

行末の `\` は行継続を表すので、最後に少なくとも 1 つの空行が必要である。

最近の TeXLive 環境では、`texmf.cnf` は `/usr/local/texlive/texmf-local/web2c/` に置くのが良いとされている。自分で作らない限り存在しないので、初めて作った場合は (上の枠内の 7 行だけの内容の `texmf.cnf` とすれば良い)

```
sudo mktexlsr
```

を実行して、`/usr/local/texlive/texmf-local/web2c/texmf.cnf` が加わったことを教える必要がある。

注意すべき点

- 画像ファイルを途中で `myimage.png` から (例えば) `myimage.pdf` に変えた場合、`myimage.xbb` は作り直しになる。その場合は手動で作り直すか、古い `myimage.xbb` を削除する必要がある。
- TeXLive 2014 の L^AT_EX では、`.xbb` を生成しないようになった (どういう仕組みで BoundingBox を得るのか、現時点で理解していない)。その場合でも `myimage.xbb` があればそれを読むので、古いものを掃除しておく必要がある。

10.2 とりあえず例を1つ

入れたいグラフィックスが PNG 形式のファイル “`besselj.png`” である場合に、L^AT_EX 文書に取り込むには、例えば次のようにする。

```
\documentclass[12pt,dvipdfmx]{jarticle}% オプション dvipdfmx に注目
...

\usepackage{graphicx}% 重要

...

\begin{document}
...

\begin{figure}[ht]% これで besselj.png を取り込む
  \centering
  \includegraphics[width=10cm]{besselj.png}
  \caption{Bessel 関数  $J_n$  ( $n=0,1,\dots,5$ ) のグラフ}
  \label{figure: ベッセル関数のグラフ}% \caption の後に \label する
\end{figure}

...
\end{document}
```

(繰り返しになるが) 要点は次のようになる。

- (a) プリアンブルで、`\usepackage{graphicx}` と `graphicx` パッケージの使用を宣言する。`graphicx` で利用するドライバをオプションで指定する (先頭で `\documentclass[12pt,dvipdfmx]{jarticle}` とするか、パッケージ読み込み時に `\usepackage[dvipdfmx]{graphicx}` とオプション設定する)。
- (b) グラフィックスのファイルの取り込みは、`\includegraphics{}` で指示する。取り込み時に `width=` 横幅 や `height=` 高さ, `scale=` 拡大率 など で大きさを指定できる。`angle=` 角度 で回転もできる。
- (c) グラフィックスは大きなスペースを占めるので、ページのどこに配置するかは組版するソフトウェアに任せるものとされている (昔は、著者は原稿を出すだけで、版を作るのは植字する人 (職

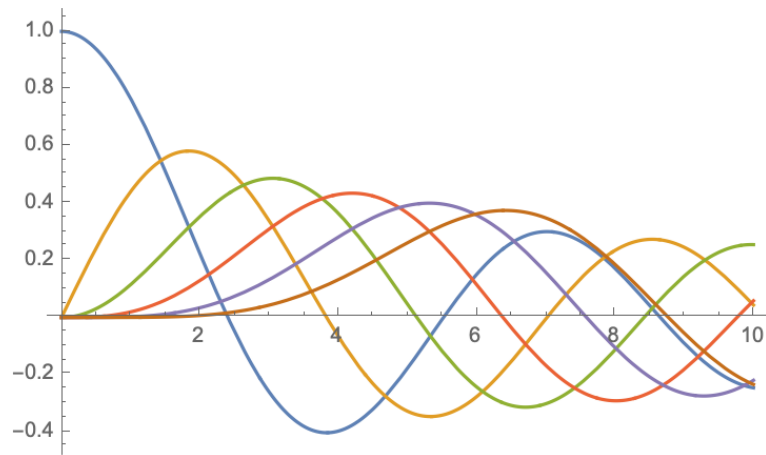


図 6: Bessel 関数 J_n ($n = 0, 1, \dots, 5$) のグラフ

人さん)、というシステムで、植字する人がソフトウェアに置き換わった、ということになる)。
 \LaTeX では、figure 環境を利用することが推奨されている。いわゆるセンタリングを行い、キャプションをつけるのが普通である。

- (d) figure 環境を使うと、図に番号が振られるが、それを参照したければ、`\label{figure: ベッセル関数のグラフ}` のようにラベルをつけ、番号が欲しいところで `\ref{figure: ベッセル関数のグラフ}` とする。

おまけ 上で使った `besselj.png` は、Mathematica で

```
g1 = Plot[Evaluate[Table[BesselJ[n, x], {n, 0, 5}]], {x, 0, 10}]
Export["besselj.png", g1]
```

として作成した。第 1 種 Bessel 関数 J_n ($n = 0, 1, \dots, 5$) の $0 \leq x \leq 10$ の範囲でのグラフである。

10.3 よくある失敗の原因を避ける

- PostScript 形式でトラブルが生じたら、別のフォーマットにしてみる。
 ずっと以前は「 \LaTeX に取り込みたいグラフィックスは PostScript がお勧め」だったけれど、そうとも言えない(複雑なグラフィックスを PostScript 形式で保存すると、ファイル・サイズが巨大なものになりがちだし)。
 個人的には、**写真は JPEG, それ以外はとりあえず PNG, イメージ形式でなくベクトル形式にしたければ PDF** という方針にしている、PostScript は使わなくなった。
- グラフィックスのファイルは、`.tex` ファイルと同じディレクトリか、そのすぐ下のサブディレクトリに置くことを勧める。
 サブディレクトリ `graph` に `besselj.png` を置いた場合は、もちろん

```
\includegraphics{graph/besselj.png}
```

のようにディレクトリ名をつける。

(トラブル相談で意外と多いのが、`.tex` とグラフィックスのファイルが全然別の場所に置いてある、というケース。ドラッグ&ドロップが当然という感覚の人(ファイルが実際にどこにあるかは意識しない)には、関係するファイルは自分で一箇所にまとめる、という(昔の)“常識”がないかも。)

- ファイル名に日本語は使わない。
全く使えないわけではないが、どういう条件下でうまく動くか分かりにくいので、使わないことを強く勧める。

10.4 複数の図を並べる

例えば2つのものを比較したい場合、同じ形式のグラフィックスで可視化して、並べて見せるのが普通である。

LaTeX でそれをするには、minipage 環境を使うのが普通らしい。

```
\begin{figure}[htbp]
  \centering
  \begin{minipage}[t]{0.45\textwidth}
    \includegraphics[width=5.5cm]{graph1.png}
    \caption{方法 A を使った場合}
  \end{minipage}
  \begin{minipage}[t]{0.45\textwidth}
    \includegraphics[width=5.5cm]{graph2.png}
    \caption{方法 B を使った場合}
  \end{minipage}
\end{figure}
```

1つの figure 環境の中に2つの minipage 環境を入れてある。minipage 環境のサイズ(横幅)は、ページの横幅(\textwidth)の0.45倍にしてある。もしも3つ並べるならば、0.30倍とか適宜変更する。

キャプションは minipage ごとにつけられる。

10.5 figure 環境について補足

図は文字と比べて大きいのが普通で、組版で位置を決めるのが難しい(論理的な順番を尊重しすぎると、大きな余白が出来たり、おかしい組版になってしまう)。同じようなものに表がある。TeX は、図については figure 環境で、表については table 環境で扱うのが良い、とされている。

ひな形としてはこんな感じ

```
\begin{figure}[htbp]
  \centering
  \includegraphics[なんとかオプション]{かんとかファイル名}
  \caption{図の説明 (いわゆるキャプション)}
  \label{figure: 引用するための文字列}
\end{figure}
```

(ようやく頭に入ったと思ったら、TeXShop のテンプレートは、これとほぼ同じものをペタッと貼り付けてくれるんですね。)

ときどき、配置しない図がたまりすぎて、TeX がこけることがある。そういうときは、\clearpage で、たまっている図を吐き出す(あまりきれいな配置にならなくても、強制的に配置する)。

キャプションを複数行書きたい場合は ccaption パッケージを読み込んで、\legend{} を使う。

```
\usepackage{ccaption}
...
\caption{キャプション (1行目)}
\legend{キャプション (2行目)}
```

図の配置位置を $\text{T}_{\text{E}}\text{X}$ 任せにせず、自分の指定した位置に出したければ、float パッケージを読み込んで [H] を使う (H は「絶対にここ (here)」という意味らしい。昔の here.sty みたいなものか?)。

```
\usepackage{float}
...
\begin{figure}[H]
...
\end{figure}
```

10.6 補足的情報

(もう削除しようかと思ったけれど一応残しておく)

10.6.1 PostScript データの取り込み

(独白: もう PostScript を使うべきではないような気がしている…勧められないものはこの文書から削除すべきかもしれない。)

画像ファイルには色々なフォーマットがあるが、PostScript は古くからレーザープリンター用の言語として使われているもので、問題が生じにくかった。

$\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ に取り込む場合は、**カプセル化 PostScript 形式** (Encapsulated PostScript, 長いので EPS 形式と呼ぶことにする、通常は “.eps” という拡張子をつける) に変換しておくのが良い。

最近では Mathematica の出力する EPS ファイルが巨大なものとなったり、そもそも表示印刷するためのソフトが OS 標準で用意されていないこともあって¹⁹、必ずしもイチオシのフォーマットとは言えなくなったと思う。

(2015/6/20) Mathematica 10 から、凡例のフォントが Times-Roman から MathematicaSans とかに変わって、dvi_{pdf}mx で処理出来なくなった。設定で逃げられるかもしれないけれど、ちょっと嫌気が差してきた。ちなみに PDF にして取り込む場合は大丈夫。

———— kamehoshi2.eps を取り込む ————

```
\documentclass[12pt,leqno,dvipdfmx]{jarticle}
\usepackage{graphicx}% graphicx パッケージが必要

\begin{figure}[htbp]
\centering
\includegraphics[width=5cm]{eps/kamehoshi2.eps}
\caption{星を蒔いてみる}
\end{figure}
```

- Mathematica (<http://nalab.mind.meiji.ac.jp/~mk/syori2/mathematica/node62.html>)

¹⁹Mac で MacPorts を使っている場合は、`sudo port install gv; sudo port install ghostscript-fonts-hiragino` とすれば、Ghostscript と、それを使って表示する gv がインストールできる。

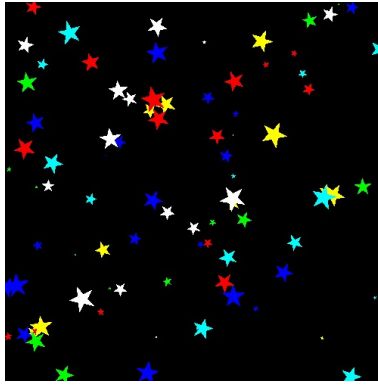


図 7: 星を蒔いてみる

- gnuplot (<http://nalab.mind.meiji.ac.jp/~mk/labo/howto/intro-gnuplot/node12.html>)
- GLSC (<http://nalab.mind.meiji.ac.jp/~mk/labo/howto/intro-glsc/node26.html>)
- FreeFem++ (<http://nalab.mind.meiji.ac.jp/~mk/labo/text/welcome-to-freefem/node4.html>)

等々では、グラフィックスを PostScript データとして出力するのは簡単である。

Mathematica の場合は、

```
g=Plot[Sin[x],{x,0,2Pi}]
Export["mygraph.eps",g]
```

のようにする。あるいは(そうして作った PostScript データが巨大になってしまう場合は)

```
Export["mygraph.jpg", g, ImageResolution->1200]
```

として JPEG で出力してから(解像度を 1200 dpi にするのは好みの問題)、

```
jpeg2ps mygraph.jpg > mygraph.eps
```

あるいは

```
convert mygraph.jpg mygraph.eps
```

として PostScript に変換する。

gnuplot の場合は、

```
gnuplot> set term postscript eps color
gnuplot> set output "mygraph.eps"
```

のようにしてから描画コマンドを実行する。なお、最近では

```
gnuplot> set term push
gnuplot> set term postscript eps color
gnuplot> set output "mygraph.eps"
gnuplot> (描画コマンドを実行)
gnuplot> set term pop
```

とするのが相場かもしれない(以前は、元に戻すために、`set term x11`とか`set term win`くらい覚えておけば良かったが、最近は結構複雑なので、`push`, `pop` が用意されたらしい)。

GLSC の場合は、描画デバイスの指定時に

```
g_init("mygraph", ...);
g_device(G_BOTH);
...
```

のようにファイル(名前は `g_init()` で指定した“mygraph”になる)に出力するもの(ここでは `G_BOTH`)を選び、

```
g_out -i mygraph
```

で変換する。`mygraph.i00` というファイルが出来ることが、

```
\includegraphics[angle=90,width=10cm]{mygraph.i00}
```

のように `angle=90` で回転して取り組むか(`width=` と `angle=` の順番には注意すること)、

```
g_out -iv mygraph
```

のようにして出力時に回転する(`-v` でポートレート・モードにする、そうである)。`-v` を指定した場合、しばしば負の座標を持つ `BoundingBox` が出来て色々障害の原因となるが(ファイル先頭部分にあるので、`head mygraph.i00` とかしてチェックして下さい)、`ps2eps` などを用いて座標の平行移動を行なうと良い。

```
ps2eps -t=100,200 mygraph.i00
```

(100,200 はイイカゲンです)

とすると `mygraph.i00.eps` というファイルが生成される。元々 `g_out` の作る PostScript ファイルの `BoundingBox` 情報はイマイチなので、`ps2eps` はつねに実行することにした方が良いかもしれない(ある程度まともな `BoundingBox` に直してくれる)。

10.6.2 JPEG イメージの取り込み

現在のデジタルカメラの主流の画像フォーマットである JPEG データの取り込みを説明する。

写真以外でも使われる場合がある。例えば十進 BASIC のグラフィックスの場合、「名前をつけて保存(A)」から JPEG 形式で(ファイル名拡張子は“.JPG”)保存する。

JPEG イメージを直接取り込む `dvipdfmx` のようなドライバーを使っている場合は、直接 `\includegraphics` で取り込める。

JPEG イメージ PostScript に変換しての取り込み `dvips` のような古いドライバーを使っている場合は、JPEG のままでの取り込みは出来ない。

しかし JPEG ファイルは、`jpeg2ps`²⁰ や `convert` (ImageMagick に含まれている) コマンドで、EPS 形式に変換してから取り込むことが可能である。

²⁰<http://www.pdflib.com/>

Windows 環境に Cygwin がインストールされている場合、その中に jpeg2ps が入っていることもある。コマンドプロンプトや、Cygwin のシェルで

```
Z:¥.windows2000¥syori2> jpeg2ps kamehosi.JPG > kamehosi.eps
```

とすると、kamehosi.JPG を EPS 形式に変換したファイル kamehosi.eps が出来る。

Mac ならば、MacPorts でインストールすることが出来る。

```
sudo port install jpeg2ps
```

使い方は上と同様である。

```
jpeg2ps kamehosi.jpg > kamehosi.eps
```

なお、Windows 7 の GUI で使える wjpeg2ps²¹ というプログラムもある。使い方は簡単で、JPEG ファイルを wjpeg2ps のアイコンにドラッグして、`convert` ボタンを押すだけで EPS 形式のファイルが出来る。

仕組みについて、もう少し詳しい説明が読みたければ、「イメージデータの TeX への取り込み — jpeg2ps のすすめ」²² を見ると良い。実際にはデータのラッピングをしているだけなので、画質の低下は生じない。

10.6.3 JPEG 以外のイメージファイルの取り扱い

JPEG 以外のイメージ・ファイルのフォーマットには、Windows BMP, GIF, TIFF, PNG など色々ある。

ドライバーとして dvipdfmx を使っている場合、png や png などは直接 includegraphics 出来る (tiff などは変換する必要がある)。

また dvips を使っている場合は、実質 EPS と JPEG しか読み込めないなので、他のほとんどのフォーマットは変換する必要がある。

ある時期までの私のお奨めは (今は「とっとと環境を新しくして、dvipdfmx 使えるようにしましょう」がお勧め)、

最初が何であれ JPEG に変換してから、
jpeg2ps で PostScript に直して取り込む、

というやり方であった (最初が BMP だったりすると、これでかなりファイルのサイズを小さくすることができる)。二度続けて変換するのは品質を落としそうだが、実は最近の PostScript は内部に JPEG データを含むことができるようになっていて、jpeg2ps はそれをやっているだけなので、実際にデータの内容を変更するのは、最初に JPEG に変換している過程だけである。

それでは、JPEG 以外のイメージ・データをどうやって、JPEG に変換するかであるが、素の Windows であればペイントを使うのが最も簡単であろうが (名前をつけて保存のところ出力の形式が選択できる)、IrfanView などの使うのが良いと思われる (もっとも私はずっと長いこと使っていない…)。

UNIX 環境 (含む Cygwin, Mac) であれば、ImageMagick に含まれている convert が簡単である。使い方は、例えば JPEG にするのであれば

²¹<http://www.vector.co.jp/soft/dl/win95/art/se248407.html>

²²<http://nalab.mind.meiji.ac.jp/~mk/labo/howto/jpeg2ps.html>

```
convert nantoka.bmp nantoka.jpg
convert nantoka.gif nantoka.jpg
convert nantoka.png nantoka.jpg
```

という感じで、出力ファイル名の拡張子を .jpg にするだけである。

10.6.4 dviout でカラー表示・印刷するには

(これももう削除すべきかなあ…)

カラーで表示・印刷するには、dviout で Option → Setup Parameters → Graphic で、GIF の取り扱いの設定で `BMP(full-color)` を選択する。dviout 起動時に、`-GIF=5` というオプション引数を指定しても良い。これをデフォルトの設定にする人も多いが、情報処理教室のプリンターはモノクロなので、さぼってある。

10.6.5 余談: ウィンドウの画像を取り込む

Mac の場合は、標準で OS に付属しているプレビューのファイルメニューの「スクリーンショットを撮る」を用いると良い (とても使いやすい)。あるいは、`shift+command+3` (全画面)、`shift+command+4` (選択部分) というショートカットもある (これが覚えられなくて…、ファイルはデスクトップに png フォーマットで作られる)。

Windows 7 のウィンドウの画像をファイルに保存したければ、マウスカーソルを取り込みたいウィンドウに置いて、キーボードから `Alt+Print Screen` (`Print Screen` は、場合によっては `Fn` キーと一緒に押す必要があり、その場合は `Alt+Fn+PrintScreen` となる) を入力し、**ペイント**²³ のようなソフトにペーストしてから、適当に編集した後で、保存すると良いでしょう (もちろん JPEG 形式に出来る)。

10.6.6 追記: PDF を PS にする

(これは現在は必要性がほとんどない。)

もちろん `convert` で `convert nantoka.pdf nantoka.eps` とすることも出来るが、ghostscript 由来の `pdf2ps` が案外使いやすい。

```
pdf2ps nantoka.pdf
ps2eps nantoka.ps
```

これで `nantoka.eps` が出来る。結果はコンパクトで画質も良いような印象がある。PDF と PostScript は相性が良い?

10.6.7 misc

10.6.8 ドライバーについて

`graphicx` のためにドライバーの指定が必要だが、それは他のパッケージにも影響する。例えば `color.sty` を使うならば、そちらにも同じドライバーを指定する。

²³ `スタート` → `すべてのプログラム (P)` → `アクセサリ` → `ペイント` として起動できる。

方法 1

```
\documentclass[12pt,dvipdfmx]{jarticle}

\usepackage{graphicx}
\usepackage{color}
```

方法 2

```
\documentclass[12pt]{jarticle}

\usepackage[dvipdfmx]{graphicx}
\usepackage[dvipdfmx]{color}
```

この color.sty の件は良く知られているが、他にもドライバーと関係するパッケージがある。うまく動かない場合は調べる必要がある (個人的に gouji.sty というのを使っていて、その中で \RequirePackage[dvipdfmx]{color} となっていて、ひっかかる原因になった)。

ずっと長い間、方法 2 を用いていたのだが、その方法では、geometry や TikZ などが問題を引き起こすようである。現時点では、方法 1 を推奨する。

10.6.9 .xbb ファイル

mygraph.{pdf,png,jpg} を取り込むには、BoundingBox 情報を記録した mygraph.xbb というファイルを用意する必要がある。

こうやって .xbb ファイルを作る

```
TeXLive に入っている extractbb を用いて
extractbb mygraph.pdf
あるいは
xbb mygraph.pdf
```

extractbb は TeXLive に含まれているようである (実体は dvipdfmx の別名)。 $\$TEXMF/web2c/texmf.cnf$ に

```
% 次は
% t (何でも実行可能)
% か
% p (shell_escape_commands で指定したもののみ実行可能)
shell_escape = p

shell_escape_commands = \
bibtex,bibtex8,bibtexu,pbibtex,upbibtex,biber,\
kpsewhich,\
makeindex,mendex,texindy,\
mpost,pmpost,\
repstopdf,epspdf,extractbb,\
```

のように extractbb を入れておくと、.xbb ファイルを自動生成してくれるようである — と言うのは昔の話? 最近の T_EX は .xbb ファイルを作らずにサイズの方法を取得している??

extractbb も xbb もこの後で出て来る ebb も、実体は dvipdfmx のリンクであるらしい。

— xbb がない場合、例えばこんな感じで準備できる —

```
$ which dvipdfmx
/usr/local/texlive/2014/bin/x86_64-darwin/dvipdfmx
```

(→ 場所が分った。そこに cd してリンクをする。)

```
$ pushd /usr/local/texlive/2014/bin/x86_64-darwin/
$ sudo ln -s dvipdfmx xbb
$ popd
```

dvipdfmx.def というファイルが古いと、.xbb ファイルの自動生成が出来ないことがあった。その場合 CTAN (<ftp://ftp.kddilabs.jp/CTAN/macros/latex/contrib/dvipdfmx-def/dvipdfmx.def>) から最新版を取得すると良い。

(メモ: 以前は `\usepackage[dvipdfmx]{graphicx}` でなくて、`\usepackage[dvipdfm]{graphicx}` だった。その場合は ebb コマンドで .bb ファイルを作成して使う。この ebb も dvipdfmx のリンクで良い。L^AT_EX Beamer (12) が dvipdfm しか使えなかったことがあったが、今では逆に dvipdfmx オプションしか使えないようになった。dvipdfm オプションの利用に関する情報はまだ落せない。)

11 TikZ

T_EX には、昔から picture 環境と呼ばれる図を描くための仕掛けが用意されていたが、機能がかなり限定されていて、率直に言って使いにくいものであった。そのため、別の手段を追及するようになったのだが、現在では TikZ (「ティクス」と読むそうだ) が良い選択肢であるらしい。

11.1 準備

グローバルに dvipdfmx オプションを指定するのが良いようだ。

```
\documentclass[... ,dvipdfmx]{jarticle}

\usepackage{graphicx}
\usepackage{tikz}
```

次善の策として

```
\usepackage[dvipdfmx]{graphicx}
\usepackage{tikz}
```

ライブラリの指定が必要になる場合がある。

```
\usetikzlibrary{intersections,calc,arrows.meta}
```

11.2 マニュアル

ターミナルから次のようにすればマニュアルが読める。

実体は/usr/local/texlive/2018/texmf-dist/doc/generic/pgf/pgfmanual.pdf とか。

11.3 いろは — 直線、円などを描く

- 点は (x 座標, y 座標) という形式で表す。自然で覚えやすい。

```
\draw (1,2) -- (3,4);
```

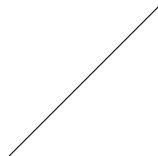


図 8: `\draw (1,2)--(3,4);`

- 点に名前をつける。名前は (文字列) という形式。

```
\coordinate (文字列) at (x 座標,y 座標);
```

例えば、A(1,2) から B(3,4) を端点とする線分を描くのに、

```
\coordinate (A) at (1,2);
\coordinate (B) at (3,4);
\draw (A) -- (B);
```

と指示できる。

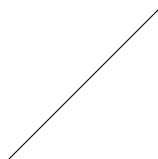


図 9: 点に名前をつけて線分を描く

- 既に例の中で使っているが、線分を描くには `\draw 点 -- 点;` とすれば良い。折れ線を描くには `\draw 点1 -- 点2 -- ... -- 点n;` とすれば良い。閉じて閉曲線にするには、最後を `-- cycle;` とする。
- 円の描画は

```
\draw 中心 circle [radius=半径];
```

- 点 (マーカーと言うべきか) を描くにはどうするのか? 私は今のところ、小さい円の内部を

```
\fill 点 circle [radius=半径];
```

のようにして塗ることになっている。

```
\begin{tikzpicture}
\coordinate (O) at (0,0);
\coordinate (A) at (2,0);
\coordinate (B) at (1,1);
\draw (O) -- (A) -- (B) -- cycle;
\fill (O) circle [radius=2pt];
\fill (A) circle [radius=2pt];
\fill (B) circle [radius=2pt];
\end{tikzpicture}
```

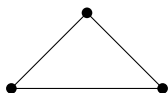


図 10: 頂点に半径 2 ポイントの円を描いてみた

- 単位は何も書かないと cm だそう。大きさが 1 程度のものからなる図を描くと小さくなる。tikzpicture 環境のオプションに [x=長さ,y=長さ] と指定して調節出来る。上に書いたように、点を描くのに円を塗りつぶしているが、その半径は拡大させず、実際の長さで指定する (2 ポイントとか) 方が良いかもしれない。(文字やマーカーの大きさが単位の選択により伸び縮みすると使いにくい。)

似ているけれど [scale=倍率] を使うと、2 ポイントと指定した円の長さも大きくなる。

```
\begin{tikzpicture}[x=3cm,y=3cm]
\coordinate [label=below left:$\mathrm{O}$] (O) at (0,0);
\coordinate (A) at (0.66666,0) node at (A) [below=0] {$|z|$};
\coordinate (B) at (1,0) node at (B) [below right=0] {$R$};
\coordinate (P) at (0.333333,0.57735) node at (P) [above=0.1,right=0] {$z$};

\draw [thick, -stealth](-1.5,0)--(1.5,0) node [anchor=north]{$x$};
\draw [thick, -stealth](0,-1.2)--(0,1.2) node [anchor=east]{$y$};

\fill (O) circle [radius=2pt];
\fill (A) circle [radius=2pt];
\fill (B) circle [radius=2pt];
\fill (P) circle [radius=2pt];

\draw (O) -- (P);
\draw [red,thick] (P) -- (B);
\draw [green,thick] (A) -- (B);
\draw (P) -- (A);
\draw (O) circle [radius=1];
\end{tikzpicture}
```

- \foreach で繰り返しを指定することも可能である (格子を描くのに便利だ)。

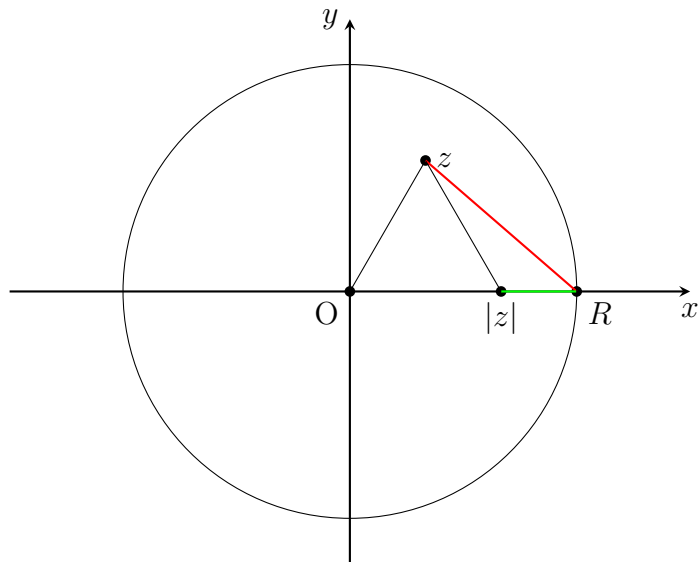
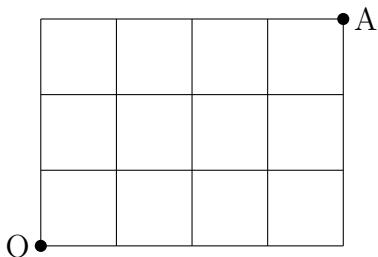


図 11: 赤線の長さ/緑線の長さ $\leq K$

```

\begin{tikzpicture}
  \coordinate [label=left: {\mathrm{O}}] (O) at (0,0);
  \coordinate [label=right: {\mathrm{A}}] (A) at (4,3);
  \foreach \x in {0,1,2,3,4} \draw (\x,0)--(\x,3);
  \foreach \y in {0,1,2,3} \draw (0,\y)--(4,\y);
  \fill (O) circle [radius=0.08];
  \fill (A) circle [radius=0.08];
\end{tikzpicture}

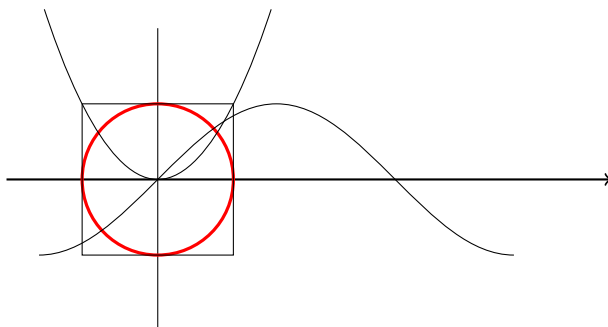
```



```

\begin{tikzpicture}
  \draw [->,thick] (-2,0) -- (2,0);% → 付きの線分を太く
  \draw (0,-2) -- (0,2);
  \draw [very thick,red] (0,0) circle [ radius=1 ];% 中心=(0,0), 半径=1
  \draw (-1,-1) rectangle (1.0,1.0); % 左下=(-1,-1), 右上=(1,1)
  \draw (0,0) parabola (1.5,2.25);
  \draw (0,0) parabola (-1.5,2.25);
  \draw (-1.57,-1) cos (0,0) sin (1.57,1) cos (3.14, 0) sin (4.71,-1);
\end{tikzpicture}

```



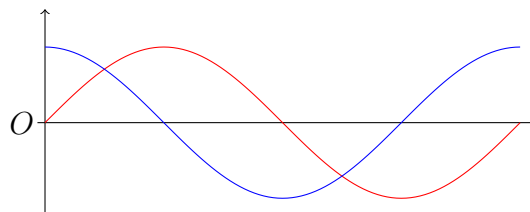
parabola は「TeX に直接作図しよう! 3」²⁴ で調べた。軸が垂直線の放物線の、頂点から指定した点までの範囲を描画する。

11.4 plot

plot という命令で曲線 (折れ線?) が描ける。

座標を記録したファイルを用意しておいて plot file {ファイル名}; とすることも出来る。

```
\begin{tikzpicture}[=>stealth]
\draw node (0,0) [left] {$0$};
\draw [->] (-0.1,0) -- (6.5,0);
\draw [->] (0,-1.2) -- (0,1.5);
\draw [red] plot file {sin.tbl};
\draw [blue] plot file {cos.tbl};
\end{tikzpicture}
```



筆者はここで使っている sin.tbl, cos.tbl を、C 言語で書いたプログラムを利用して用意したが、簡単な関数の値データならば、gnuplot を利用して作成できる。

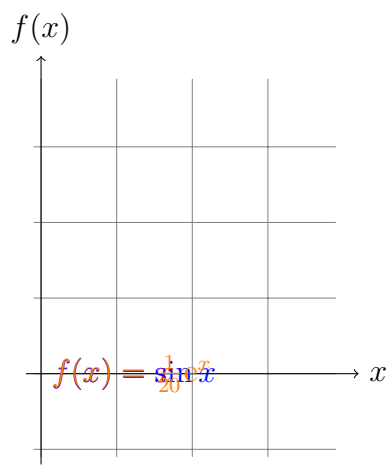
```
\begin{tikzpicture}[domain=0:4]
\draw[very thin,color=gray] (-0.1,-1.1) grid (3.9,3.9);
\draw[->] (-0.2,0) -- (4.2,0) node[right] {$x$};
\draw[->] (0,-1.2) -- (0,4.2) node[above] {$f(x)$};
\draw[color=red] plot[id=x] function{x} node[right] {$f(x) =x$};
\draw[color=blue] plot[id=sin] function{sin(x)} node[right]
  {$f(x)=\sin x$};
\draw[color=orange] plot[id=exp] function{0.05*exp(x)} node[right]
  {$f(x) = \frac{1}{20} \mathrm{e}^x$};
\end{tikzpicture}
```

これで一度組版すると、なんとか.x.gnuplot, なんとか.sin.gnuplot, なんとか.exp.gnuplot というファイルが出来る。それぞれ gnuplot で実行する。

```
gnuplot なんとか.x.gnuplot
gnuplot なんとか.sin.gnuplot
gnuplot なんとか.exp.gnuplot
```

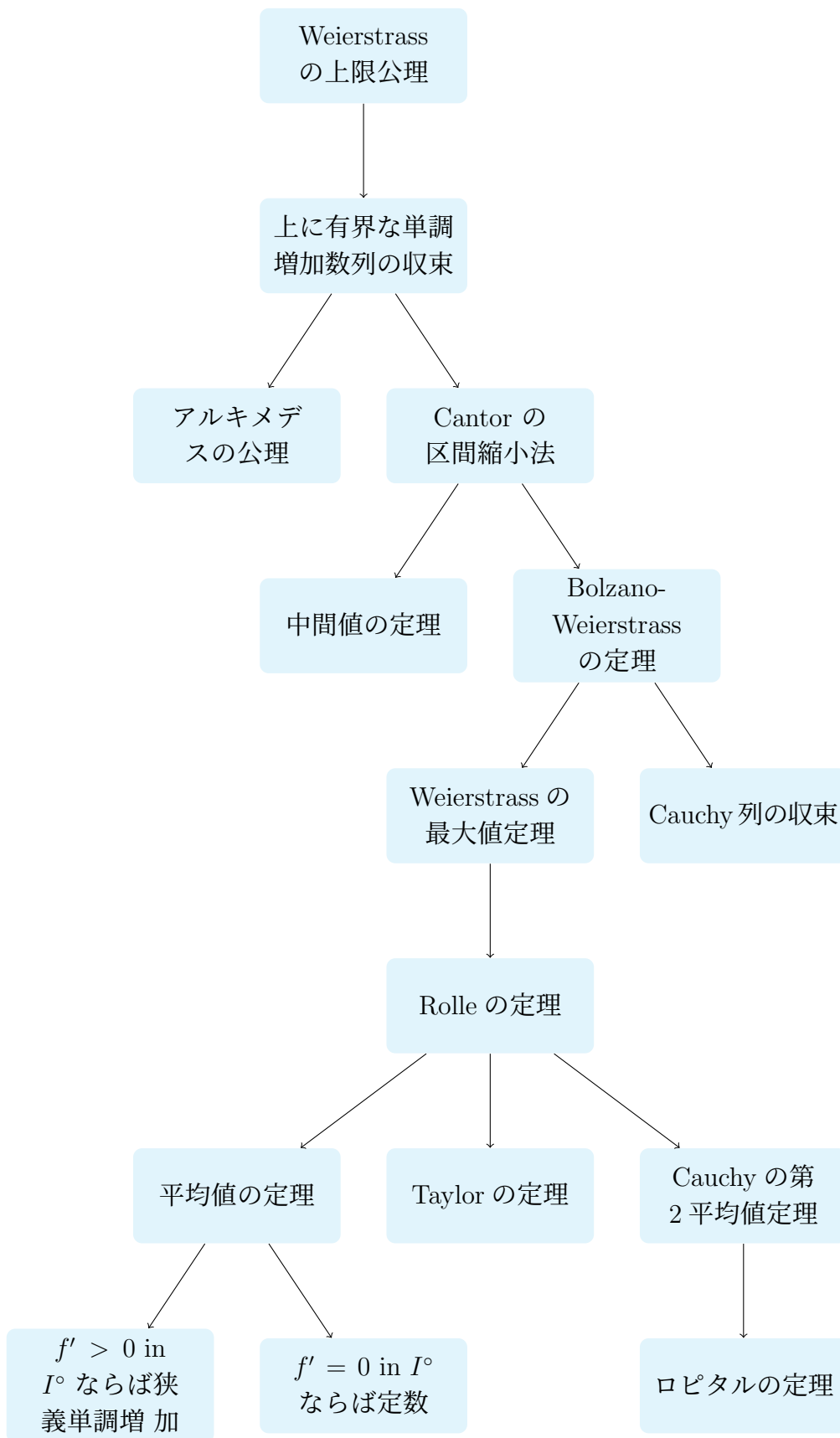
するとなんとか.x.table, なんとか.sin.table, なんとか.exp.table というファイルが出来る。もう一度組版することで作図される。

²⁴<http://hitgot.org/archives/drawing-in-tex-by-tikz-3/>



11.5 模式図

```
\begin{tikzpicture}
  \tikzset{block/.style={rectangle, fill=cyan!10, text width=3cm,
    text centered, rounded corners, minimum height=1.5cm}};
  \node[block] {Weierstrass の上限公理}
  [level distance=3cm, sibling distance=4cm,
  edge from parent/.style={->,draw}]
  child {
    node[block] {上に有界な単調増加数列の収束}
    child {
      node[block] {Cantor の区間縮小法}
      [level distance=3cm, sibling distance=4cm,
      edge from parent/.style={->,draw}]
      child{
        node[block] {中間値の定理}
      }
      child {
        node[block] {Bolzano-Weierstrass の定理}
        child {
          node[block] {Weierstrass の最大値定理}
          child {
            node[block] {Rolle の定理}
            child {
              node[block] {平均値の定理}
              child {
                node [block] {$f'>0$ in  $I^{\circ}$  ならば狭義単調増加}
              }
              child {
                node [block] {$f'=0$ in  $I^{\circ}$  ならば定数}
              }
            }
          }
          child {
            node[block] {Taylor の定理}
          }
          child {
            node[block] {Cauchy の第 2 平均値定理}
          }
        }
      }
    }
  }
  child {
    node[block] {Cauchy 列の収束}
  }
}
};
\end{tikzpicture}
```

12 L^AT_EX Beamer でプレゼン

コンピューターの画面出力をスクリーンに映してプレゼンするのが普通になりました。Windows だと PowerPoint, Mac だと Keynote というソフトが有名ですが、T_EX を使うことも出来ます。数学関係の講演では、日頃から T_EX に慣れている、数式を多用する、などの理由から T_EX を使うの

が普通です。

スクリーンは、

- 横長であるのが普通
- 同時に1つしか使えないのが普通
(なるべく1ページに1話題を書き切り、1ページに1~3分の時間をかけて説明するのが良い)
- 小さい字をびっしり使うと読みづらく、大きい文字で箇条書きの文体を使うのが良い
(この点は黒板や OHP (オーバーヘッドプロジェクター) と事情が似ている)
- 写真やカラーの図、動画が映せる、音も出せる

などの特徴があります。

TeX でそれに合った文書を簡単に作れるように、専用のスタイル・ファイルが色々開発されています。LaTeX Beamer は最近人気があるスタイル・ファイルです。

12.1 準備

最近では TeX Live を入れるだけで、LaTeX Beamer がちゃんと動くのが普通らしい。

The beamer class User Guide²⁵ というのがある。分厚いけれど、あまり分かりやすくないなあ。

12.2 必要最小限の知識

1. 最初に

```
\documentclass[dvipdfmx,cjk]{beamer}
```

と書く。以前は dvipdfmx でなくて、dvipdfm でないと駄目だった。今やってみて dvipdfmx で動かなかったら、LaTeX Beamer や TeX Live を更新したり、(まだ TeX Live を使っていない場合) TeX Live に乗り換えることをお勧めします。

2. 体裁の大枠は theme を指定することで決定される。

```
\usetheme{Madrid}
```

Madrid 以外に他にどういうテーマがあるか知りたい人、凝りたい人はネットで調べよう。

3. 日本語のしおりが使いたい場合とか、次をしておくが良い。

```
\usepackage{pxjahyper}
```

4. もちろん \begin{document} と \end{document} も必要。

²⁵<http://tug.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>

5. 1枚1枚をフレームと呼ぶが、それを書くのに普通は次のようにする。

```
\begin{frame}{フレームの見出し}

\end{frame}
```

6. dvi でなく PDF を作る。秀丸+祝鳥ならば「PDFに変換して表示」を選択する。TeXShop を使っていれば何も意識する必要はない。

- 表題で `\subtitle{}` (副題) や `\institute{}` (所属) が使える。

```
\title{有意義な卒研のすごし方}
\subtitle{～ 楽しく真面目にやろう ～}
\author{明治太郎}
\institute{明治大学現象数理学科}
\date{2019年9月18日}
\frame{\titlepage}
```

- `\color{色の名前}` や `\textcolor{色の名前}{テキスト}` などが使える。これらは本来 beamer のコマンドというわけではないが有益である。
- `\usepackage{graphicx}` を省略して、`\includegraphics{}` コマンドも使える。

— sample.tex —

```
\documentclass[dvipdfmx,cjk]{beamer}
\usepackage{pxjahyper}
\usetheme{Madrid}

\begin{document}

\title{有意義な卒研のすごし方}
\subtitle{～ 楽しく真面目にやろう ～}
\author{明治 太郎}
\institute{明治大学現象数理学科}
\date{2019年9月18日}
\frame{\titlepage}

\begin{frame}{はじめに}
  Beamer で出来ることを説明します..
\end{frame}

\end{document}
```

12.3 stepwise viewing

1枚のスライドでスペースバーを押す度に一步ずつ表示を進めて行く機能

- `\pause` が基本
- `\only<数>{テキスト}` のように特定のページでだけ表示
- `\uncover<数>{テキスト}` のように特定のページでだけ表示 (それ以外のページでは空白)

- `\temporal<数>{テキスト 1}{テキスト 2}{テキスト 3}` あるページ以前、そのページ、そのページ以後
- `itemize`, `enumerate` 環境では `\item<ページ指定>` が使える。ページ指定としては、3 や 2,3 や 1-3,5 や 3- や -3 などの指定が出来る。
- `\textcolor<2-4>{red}{テキスト}` は、2 ページから 4 ページ目でだけ赤
- 印刷時に `handout` オプションを指定すると、スライドが 1 枚にまとまる。

```
\documentclass[dvidfpmx,cjk,handout]{beamer}
```

12.4 リンク

その文書内のフレームにリンクを張ったボタンが作れる。

```
\hyperlink{ラベル}{\beamergotobutton{ボタンのテキスト}}
```

(ボタンをクリックするとジャンプ出来るが、戻るには Mac の Acrobat では `command` + `←`, Preview では `command` + `⏪` とする)

ここで言うラベルとは、次のようにしてフレームにつけることが出来る。

```
\begin{frame}[label=文字列]
```

12.5 しおりの文字化けの防止

PDF にしおりをつけるのは便利であるが、UTF8 を使って書くと、しおりが文字化けすることがある。その対処法。

beamer でしおりを付ける²⁶

```
\ifnum 42146=\euc"A4A2 \AtBeginDvi{\special{pdf:tounicode EUC-UCS2}}
\else
\AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
\fi
```

\TeX 特有の表現が、PDF のしおりでうまく表現できず、“token not allowed in a PDFDocEncoded string” のようなメッセージが表示されたら、`\texorpdfstring{}{}` を用いる。最初の `{}` の中に \TeX の表現、二つ目の `{}` の中にそうでない表現をいれる。

(2022/9/24 追記) ↑これは古い情報で、今は

```
\usepackage{pxjahyper}
```

と 1 行書いておけば良い。

²⁶<http://refluster.blogspot.jp/2010/10/blog-post.html>

12.6 その他

- オプションは `dvipdfm` でなくて `dvipdfmx` に移行中? 最近の MacTeX 環境などで `dvipdfm` を選択すると、`'dvipdfm.def' not found` と呼ばれることがある。
- 文字に色をつける `\usepackage{color}` は不要である (Beamer 自身が読み込むから)。
- グラフィックスを取り込む `\usepackage{graphicx}` は不要である (Beamer 自身が…)。特に

```
\documentclass[dvipdfmx,cjk]{beamer}
...
\usepackage[dvipdfmx]{graphicx}% この行は不要
```

のようにするとエラーになる。 `graphicx` のオプション `[dvipdfmx]` を削除すれば通るが、そもそも `\{graphicx}` そのものが不要である。

- 定理環境の中でイタリックにしたくないならば

```
\setbeamertemplate{theorems}[normal font]% \upshape
```

- `verbatim` 環境が使いたいとき、`fragile` オプションを指定する。

```
\begin{frame}[fragile]{frame}
```

12.7 授業資料準備をして覚えたこと (細かいノウハウ)

2020 年度は、多くの授業をオンデマンドで行うために、大量の資料を作成したが、私は `LATEX Beamer` を使った。

- 定理に番号を振るには
`\setbeamertemplate{theorems}[numbered]`
- 定理の環境でイタリックにしないためには

```
\setbeamertemplate{theorems}[normal font]
```

普段は `\usepackage{theorem} \theorembodyfont{\upshape}` としていたやつのかわり。ところが番号を振るのと両方やるとダメ? 次のようにする。

```
\setbeamertemplate{theorems}[numbered]
\addtobeamertemplate{theorem begin}{\normalfont}{}
```

と `\addtobeamertemplate{}` すれば良い。

- 定理の番号付をどうするか。全 28 回という授業があって、定理を参照できないとやばい。 `lecture` の番号をつけることにした。

```
\renewcommand\thetheorem{\arabic{lecture}.\arabic{theorem}}
```

- 参考文献表も番号がないと cite しても変な感じだ。

```
\setbeamertemplate{bibliography item}{\insertbiblabel}
```

- これも <https://www.opt.mist.i.u-tokyo.ac.jp/~tasuku/beamer.html> で知ったのだが、translation という仕掛けがある。

```
\uselanguage{japanese}
\languagepath{japanese}
\deftranslation[to=japanese]{Theorem}{定理}
\deftranslation[to=japanese]{Corollary}{系}% ある
\deftranslation[to=japanese]{Lemma}{補題}
%\deftranslation[to=japanese]{Proposition}{命題}% ない!
\deftranslation[to=japanese]{Example}{例}
\deftranslation[to=japanese]{Examples}{例}
\deftranslation[to=japanese]{Definition}{定義}
\deftranslation[to=japanese]{Definitions}{定義}
\deftranslation[to=japanese]{Problem}{問題}
\deftranslation[to=japanese]{Solution}{解}
\deftranslation[to=japanese]{Fact}{事実}
\deftranslation[to=japanese]{Proof}{証明}
\def\proofname{証明}
```

なるほど。日本語の文書も標準の枠組みで使えると。見よう見まねで

```
\deftranslation[to=japanese]{Figure}{図}
```

としたら、figure 環境のラベルが「Figure」から「図」に変わった。へー。

- へーっと思ったのは、theorem, lemma, corollary, definition などがあるのに、proposition が
ないこと。そういうのはもう時代遅れなのかしら。(それで身の回りの本をパラパラめくって
みたり。) 迷ったけれど、導入することにした。これは Beamer でなくて、L^AT_EX の常識が通
る、というだけのことだ。

```
\newtheorem{proposition}[theorem]{命題}
```

- ナビゲーション・シンボルは使っていないので消したい、という場合は

```
\beamertemplatenavigationsymbolsempy
```

- フットライン (フッター?) をスライドの番号だけにする。

```
\setbeamertemplate{footline}[frame number]
```

- フットラインは自分なりに変えたい、という人は多いらしく、上のような小修正だけでなく、
最初から自分で定義し直す、というのも。ネットの情報 (<https://tex.stackexchange.com/>

[questions/443659/how-to-remove-date-from-footnote-of-madrid-theme-of-beamer-and-use-rq=1](https://www.latex-community.org/questions/443659/how-to-remove-date-from-footnote-of-madrid-theme-of-beamer-and-use-rq=1))を参考にして、次のようにしてみた。

```
% footline を変える
\makeatletter
\setbeamerfont{footline}{%
  \leavevmode%
  \hbox{%
    \begin{beamercolorbox}[wd=.15\paperwidth,ht=2.25ex,dp=1ex,center]{author in head/foot}
      \usebeamerfont{author in head/foot}\insertshortauthor\expandafter\ifblank\expandafter\insertshortauthor
    \end{beamercolorbox}%
    \begin{beamercolorbox}[wd=.77\paperwidth,ht=2.25ex,dp=1ex,center]{title in head/foot}
      \usebeamerfont{title in head/foot}\insertshorttitle
      \hspace*{2ex}\insertshortsubtitle
    \end{beamercolorbox}%
  }%
  \begin{beamercolorbox}[wd=.08\paperwidth,ht=2.25ex,dp=1ex,right]{date in head/foot}
    \usebeamerfont{date in head/foot}%
    \usebeamertemplate{page number in head/foot}%
    \hspace*{2ex}
  \end{beamercolorbox}
\makeatother
```

(要は大きさを指定した beamercolorbox を並べるということ。元ネタは date を削って、おっきいタイトルを、ということ。回答者の説明はオリジナルの定義を引用していたりしていて、分かりやすい。私はタイトルは短いけど、サブタイトルを表示させたい、と考えて1行加えてみた。)

- 参考文献表が長くなって1枚のスライドに入らない場合は、

```
\begin{frame}[allowframebreaks]
```

として frame を作ると、参考文献表が自動的に分割される。

12.8 初めてのスライド発表をまじかに控えた人に

- $\text{T}_\text{E}_\text{X}$ の操作に十分慣れていれば良いが、そうでない場合は、案外紙とシャープペン&消しゴムで下書きしてから、打ち込むのが良いかもしれない。
- 12.2 の例をコピペして組版してみて (所要時間3分?)、それを書き換えて行く、くらいでスタート出来る。

13 PDFを作る しおりとか

最近では dvi でなく、PDF が実質的に最終的な出力形式となっている。特に意識しないでもそれが出来ている人が多いと思う。

ずっと以前だと、dvi を作ってから、dvipdfmx を使うとか、説明していた。

dvi から PDF を作るには、私はこんなふうにやっています

```
dvipdfmx -d 5 -0 2 なんとか.dvi
```

これで なんとか.pdf が出来る。オプションは趣味の問題です。

TeX で作った目次は、PDF にいわゆる“しおり”として入る。その際に日本語が文字化けしたりするが、それを避けるために pxjahyper パッケージを使う。

```
\usepackage[dvipdfmx]{hyperref}
\usepackage{pxjahyper}% hyperref の後にこれを読み込む
```

ずっと以前は `\AtBeginDvi{\special{pdf:tounicode EUC-UCS2}}` のような呪文を書いたりしたが、今は pxjahyper を読み込むだけで済む。

hyperref パッケージの説明はサボることにする。私は実際には

```
\usepackage[dvipdfmx,bookmarks=true,bookmarksnumbered=true,bookmarkstype=toc,%
colorlinks=true,linkcolor=blue,citecolor=blue,filecolor=blue,pagecolor=blue,%
urlcolor=blue]{hyperref}
```

のように色々オプションを指定している(以前どこかで読んだことを真似しているだけ)。bookmarksopenlevel=なども指定した方が良さののだろうか。

参考文献

- [1] 奥村晴彦, 黒木裕介, *LaTeX 2_ε 美文書作成入門 改訂第 8 版*, 技術評論社 (2020/11/14). 2017 年の第 7 版以来 3 年ぶりの新版。安心して勧められます。私は Kindle 版を使っています。
- [2] TeX Wiki, <https://texwiki.texjp.org/>
- [3] ドナルド・E. クヌース著, 鷺谷 好輝訳, *TeX ブック — コンピューターによる組版システム*, アスキー (1992).

A Tips

A.1 用紙のサイズ

A.1.1 LaTeX 文書の中で

まず TeX 自体に用紙のサイズを指示するには、geometry.sty で指定するのが簡単です。

```
\usepackage{a4paper}{geometry}
```


b4paper というものもありますが、これは ISO 規格だそうで、日本で普通に B4 という場合は、b4j とするのが良いのかもしれませんが。

日本で B4 (JIS 規格) を使う

```
\usepackage{b4j}{geometry}
```

横置きにするには landscape とします。

A4 横置き

```
\usepackage{landscape}{geometry}
```

A.1.2 後で dvipdfmx を使うことを見越して

海外では pdflatex が普及していて、.tex から直接 .pdf を作るそうですが、日本ではまだ dvipdfmx を利用して、.dvi 経由で .pdf を作るのが普通だと思います (そろそろ自信がなくなってきたぞ)。

dvipdfmx では、-p オプションで用紙サイズの指定が出来ますが、その指定を .tex ファイルの中に埋め込むことも出来るそうです (ISO 規格にしか対応していない dvipdfmx で B4 を扱いたい時とか便利かも)。

B4 縦 (JIS)

```
\AtBeginDvi{\special{pdf: pagesize width 257mm height 364mm}}
```

B4 横 (JIS)

```
\AtBeginDvi{\special{pdf: pagesize width 364mm height 257mm}}
```

A4 横

```
\AtBeginDvi{\special{landscape}}
```

結局、B4 縦のときは geometry と合わせてこんな感じに設定

```
\usepackage[b4j,hscale=0.80,vscale=0.90]{geometry}% hscale,vscale は好み
\AtBeginDvi{\special{pdf: pagesize width 257mm height 364mm}}
```

用紙サイズ	長い辺	短い辺
A3	420 mm	297 mm
A4	297 mm	210 mm
A5	210 mm	148 mm
B4	364 mm	257 mm
B5	257 mm	182 mm

A.1.3 色々なコマンドでの用紙サイズ指定のオプション

これは大変だ (こんなもの覚えられるわけがない！PDF の中に埋め込みたがるわけだ)。

コマンド	オプション	
xdvi	-paper	landscape は a4r のような名前
dvips	-t	landscape は -t landscape
dvipdfmx	-p	bx ($x = 0, 1, \dots, 10$) は ISO 規格, bxj が JIS 規格, landscape は -1

A4 landscape の場合

```
xdvi -paper a4r myreport.dvi &  
dvips -t landscape myreport.dvi | lp  
dvi2pdf -l myreport.dvi
```

B4 の場合

```
xdvi -paper b4 myreport.dvi &  
dvips -t b4 myreport.dvi | lp  
dvi2pdf -p b4j myreport.dvi
```

B4 landscape の場合

```
xdvi -paper b4r myreport.dvi &  
dvips -t b4 -t landscape myreport.dvi | lp  
dvi2pdf -p b4j -l myreport.dvi
```

A.2 バージョンが書いていない PostScript ファイル

PostScript ファイルの先頭行は

```
%!PS-Adobe-1.0
```

のように、PostScript のバージョンを示す「注釈行」が入っているのが普通である。ところが古いソフトが生成した PostScript ファイルには、このバージョン情報が空のものがあり、これが問題を引き起こすことがある (PDF を作ったとき、図が表示されないなど)。

こういうときは、嘘でもバージョン番号を書くと、うまく行くことがある

A.3 負の座標を含む BoundingBox を直す

負の座標を含んだ BoundingBox を持つ PostScript ファイルは、色々な問題を引き起こす。座標をずらすことで問題が解決できることがある。

```
ps2eps -t=100,200 kasanari.eps
```

(kasanari.eps.eps という名前のファイルが出力される。)

A.4 今いつでしょう? (\today その他)

現在の日付は \today で出力される (以前はデフォルトで元号使用だったが、今では西暦になった。古い L^AT_EX を使っていて、西暦が使いたければ、事前に \西暦 としておく)。\the\year, \the\month, \the\day という L^AT_EX コマンドがある。pL^AT_EX には \the\hour, \the\minute もある。

A.5 jobname

その T_EX ファイルの名前は? \jobname が使える。

A.6 macOS プレビューメモ

TeX と直接の関係はないが、プレビューでプレゼンテーションをすることがあるので。

- ページ番号ジャンプは `option` + `command` + G
- 戻るには `command` + [
- 全画面表示するには `control` + `command` + F

A.7 QED

証明の終わりは `amsthm` パッケージを使っていれば、

```
\begin{proof}
\end{proof}
```

で勝手にやってもらえる。

自分で使うには

```
\usepackage{amssymb}% \Box に必要
\newcommand{\qed}{\hfill$\Box$}% 右隅に白抜きの四角
\newcommand{\anqed}{\hbox{\rule{6pt}{6pt}}}% 黒い正方形
```

A.8 \mathbb{R} など黒板太字

\mathbb{R} , \mathbb{C} , \mathbb{N} , \mathbb{Z} などの、太字を黒板上で二重線で表現するフォントは `\mathbb{}` で出力可能である。

```
\usepackage{amssymb}
..
実数体  $\mathbb{R}$ 
```

以前は `\Bbb` というコマンドを使うことになっていたが、それは `obsolete` (時代遅れ) であるらしい。

A.9 下付きチルダ

`accents` パッケージの `\undertilde`

A.10 ベクトルの太字

長いこと

```
\newcommand{\Vector}[1]{\mbox{\boldmath$#1$}}
```

というマクロを使ってきたけれど、添字が小さくならない。

```
\[
  \Vector{n}_{\Vector{y}}
\]
```

として

$$n_y$$

となるとか。

```
\usepackage{bm}
...
\[
  \Vector{n}_{\bm y}
\]
```

とすれば

$$n_y$$

A.11 ベクトルの矢印

矢印は、 \vec{a} (`\vec a`) だと小さ過ぎ、 \overrightarrow{a} (`\overrightarrow a`)、では大き過ぎる、あるいは \overrightarrow{AB} (`\overrightarrow{AB}`) の矢印は開き過ぎていて、など違和感を持つ場合が多い。

esvect パッケージの `\vv` コマンドというものがある。割と気に入って使っている。

```
\usepackage{esvect}
...
\begin{align*}
&\vv{a}+\vv{b}=\vv{c},\\
&\vv{\mathstrut a}+\vv{\mathstrut b}=\vv{\mathstrut c},\\
&\vv{\mathstrut AB}+\vv{\mathstrut BC}=\vv{\mathstrut AC}.
\end{align*}
```

$$\begin{aligned}\vec{a} + \vec{b} &= \vec{c}, \\ \overrightarrow{a} + \overrightarrow{b} &= \overrightarrow{c}, \\ \overrightarrow{AB} + \overrightarrow{BC} &= \overrightarrow{AC}.\end{aligned}$$

A.12 rsfs フォント (ある1つの花文字)

Ralph Smith's formal script パッケージを使うと、 \mathcal{D} , \mathcal{S} のような花文字が利用できる。分野によって「これがなければ」というような文字があるので、重宝する。

```
\usepackage{rsfs}
...
\[
  \mathscr{D}(\Omega)=C^\infty_0(\Omega).
\]
```

$$\mathscr{D}(\Omega) = C_0^\infty(\Omega).$$

A.13 exsheets (Yet another package for the creation of exercise sheets) スタイル

授業を担当していると、練習問題のプリント (あるいは問題を含んだ講義ノート) を作る必要が生じる。

問題文のすぐ脇に答が見えると学習効果を損なうので、解答を公開するにしても、プリントの末尾 (裏面とか) や章末、巻末などに置くべきであるが、解答を書く立場になると、問題文の直後に解答を書くのが自然である。そういう作業を助けるためのパッケージがいくつかあるが、最近では exsheets パッケージを使っている。

```
\usepackage{exsheets}
\SetupExSheets{question/name=問}% 「問 1」 のようにする。
\SetupExSheets{headings=runin}% 「問 x」 の後に行変えしないで問題文を書く
\SetupExSheets{solution/name=解答}% 「解答 1」 のようにする。
```

```
\begin{question}
  正則関数の定義を述べよ。
\end{question}

\begin{solution}
  複素平面の開集合で定義された複素数値の関数は、
  定義域の各点で微分可能なとき、正則であるという。
\end{solution}

....

\printsolutions % すべての解答をまとめて表示 (節ごとに出来たりする)
```

最近の TeX Live では、`texmf-dist/doc/latex/exsheets` にドキュメント `exsheets_en.pdf` がある。

question 環境で普通に `\label{}` を使って、問題の番号が取得できる。`\pageref{}` 出来るわけか。また問題の番号を自分の望むようにずらすには、`\setcounter{question}{数}` とすれば良い (やってみたら出来た)。

(2019/8/14 追記) TeX の仕様が変わったせいか、途中で引っかかるようになってしまった。「LATEX exsheets」²⁷<http://nalab.mind.meiji.ac.jp/mk/knowhow-2018/node54.html>

²⁷₁

A.14 下線

下線を引くコマンドとして、`\underline{}` というのがあるが、複数行に渡るような長い部分には下線を引けない。`jumoline.sty` の提供する `\Underline{}` コマンドが使えるかも知れない。

```
\usepackage{jumoline}
\setlength{\UnderlineDepth}{3pt}

\Underline{下線を引きたいところはこのようにすれば良い。}
```

参考: 大石 勝「下線に関するマクロ比較」²⁸

A.15 URL など

参考文献は

- 書籍ならば、著者名、タイトル、出版社、出版年
- 論文ならば、著者名、タイトル、論文誌名、巻号、掲載ページ、出版年

が基本、ということはおちこちで教わるが、最近はネットの情報を参考にする場合も多い。

その場合にどう扱うか、なかなか難しい (著者が分からなかったり、アクセスが出来なくなったり、内容がどんどん書き換えられたり) 面がある、というのは脱線で、URL をどう記すか。

URL には記号が多く含まれているため、そのまま `.tex` ファイルに書いても正しく組版されないことが多い。

筆者は `url` パッケージを使っている。

```
\usepackage{url}
...

\url{http://nalab.mind.meiji.ac.jp/~mk/lab/text/tex2018/node50.html}
```

最近は `%E3%82` のように文字コードを 16 進数で表すことがあり、これを `\url{}` コマンドは正しく扱うことは出来ない (`%` は `TeX` で特別な意味があるから)。`%` の前に `\` を入れる必要がある (つまり `\%E3%82` とする)。このように、`url` パッケージは完璧ではないが、まあまあ使える。

A.16 文字に色をつける

`colors.sty` を使うと文字に色をつけられるが、デフォルトでは色数が少ない (`red`, `blue`, `green`, `yellow`, `magenta`, `white`, `black`)。 `usenames` オプションをつけると、様々な色を使えるようになる。

```
\usepackage[usenames]{color}
```

²⁸<http://www17.plala.or.jp/ohishi-masaru/tex/library/underline.pdf>

A.17 表の斜め線 — diagbox パッケージ

2次元の表を作る時に、行と列が何を表すか、斜め線を引いて説明を書くことがある— これで説明になっているか自信がないが、例えば次のような表。

$\Delta t \backslash \theta$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	×	×	×	×	×	○	○	○	○	○
0.01	×	×	×	×	○	○	○	○	○	○
0.001	×	○	○	○	○	○	○	○	○	○

```
\usepackage{diagbox}

...

\begin{tabular}[t]{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\diagbox{\Delta t}{\theta} & $0.1$ & $0.2$ & $0.3$ & $0.4$ & $0.5$ &
$0.6$ & $0.7$ & $0.8$ & $0.9$ & $1.0$ \\
\hline
$0.1$ & × & × & × & × & × & ○ & ○ & ○ & ○ & ○ \\
\hline
$0.01$ & × & × & × & × & ○ & ○ & ○ & ○ & ○ & ○ \\
\hline
$0.001$ & × & ○ & ○ & ○ & ○ & ○ & ○ & ○ & ○ & ○ \\
\hline
\end{tabular}
```

A.18 ラプラシアン

私が学生の頃 (まだタイプライターを使う人も残っていた時代。私は使っていないけれど。)、初めて論文を投稿するというとき、注意事項をレクチャーされた。そのとき「ラプラシアンはデルタではない」というのを注意された。

巷を見ていると、 $\text{T}_\text{E}\text{X}$ で Δ を使う人が多い。今さら反対する気もないけれど、私自身は (上のように指導されたこともあって)

```
\newcommand{\Laplacian}{\mathop{\triangle}\nolimits}
```

というのを長いこと使っていた。少し前に

```
\newcommand{\Laplacian}{\mathop{\!|\!}\mathbin{\bigtriangleup}}
```

と言うのに切り替えたところ。

```
\[
\Delta \quad \mathop{\triangle} \quad \mathbin{\bigtriangleup}
\]
```

Δ \triangle \bigtriangleup

B 日本の数学書、学校数学のルール

B.1 図形の点

図形の点をローマ字 A, B, C, \dots で表すことが多いが、そのときの字体はイタリックでなく、立体である。混同しやすいが、角 (かく) はイタリックである。

$\triangle ABC$ で、 $\angle ABC$ を単に角 B という (頂点 B における角なので)。

B.2 等号の否定

\neq の否定は $\text{T}_{\text{E}}\text{X}$ では、`\ne` とすると説明されるが (\neq が出力される)、日本の学校数学の教科書では、斜め線が左上から右下に走る $\not\equiv$ の形をしていることが多い。

```
\def\Noteq{\mathrel{%
\setbox0\hbox{=}\hbox{=}\llap{\hbox to\wd0{\hss$\backslash\hss}}}}
```

B.3 初等幾何の記号

図形の合同は、日本では $\triangle ABC \equiv \triangle PQR$ のように \equiv で表すが、英語文化圏では $\triangle ABC \cong \triangle PQR$ のように \cong (`\cong`) を用いて表す。

同様に図形の相似は、日本では $\triangle ABC \infty \triangle PQR$ のように ∞ で表すが、英語文化圏では $\triangle ABC \sim \triangle PQR$ (`\sim` を使った) や $A \equiv B$ (`\mathrel{\text{\rotatebox{90}{\equiv}}}`) を使った) で表すそうである。

C MathJax

来年は本文に移そう。…移し忘れた。

MathJax というのは HTML で数式を用いるために作られた仕掛けである (JavaScript で実現しているとか)。私は MathJax を重宝している (LaTeX 文書として書いて LaTeX2HTML で変換することも多いが、授業の WWW サイトなど直接 HTML を書くことも結構ある。そういうときに MathJax は便利である。)

MathJax の本家 [MathJax](https://www.mathjax.org/) の WWW サイト²⁹

²⁹<https://www.mathjax.org/>

「MathJax の使い方」³⁰ (私はここを読んで使い方を覚えた。)

html ファイルの `<head>` と `</head>` の間に、次のように書く。

```
<script type="text/javascript" async
  src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js?config=TeX-AMS_CHTML">
</script>
```

数式モードを、普通の T_EX と同様にドル記号で囲って使いたいときは、

```
<script type="text/x-mathjax-config">
MathJax.Hub.Config({
  tex2jax: {
    inlineMath: [ ['$','$'], ['\\(', '\\)'] ],
    processEscapes: true
  }
});
</script>
<script type="text/javascript" async
  src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js?config=TeX-AMS_CHTML">
</script>
```

³⁰<http://gilbert.ninja-web.net/math/mathjax1.html>