シケプリ・レポート作成のための LaTeX 入門

くりぷとん*

2026年4月14日

目次

* https://krp.todaiverse.org, Twitter: @8A4MB2

コラム一覧

本稿の構成

更新履歴

2026-03-31　飽きてしまったので，書きかけで公開
2026-04-01　細かい訂正，配布先の追加
2026-04-14　長さについて加筆，組版の修正，第3節の加筆

はじめに

本稿はシケプリというよりは，単純に実用的なLaTeXの入門ガイドとなっています．執筆の動機としては，pLaTeXのような古い[1]エンジンの情報が多く，未だにLuaLaTeXへの移行が進まない現状を憂慮したためです[2]．加えて，TeXを用いて組版する意識の高い学生であっても，TeXLiveのインストールが難しいことや，誤った記法やバッドプラクティスに陥ってしまうことを防ぎ，クオリティーの底上げを目的としています．

また，自分が発見したベストプラクティスをまとめておき，未来の自分が同じところで詰まらないようにすることも目的です[3]．そのため，変なところに力が入っていたり，独特な構成を推奨しているところもあるかもしれません．

なお，[1]を大いに参考にしています．絶対に買ってください．この本の共著者である奥村晴彦氏はもう日本語TeX界のレジェンドです．同氏も深く関わったTeXWiki (https://texwiki.texjp.org/)は日本語TeXでは圧倒的な情報量と網羅性を誇るサイトですから，何か困った際はこちらも参照してください．

本稿の特徴

類書との違いは，モダンなLaTeX（主にLuaLaTeX）に特化していること，初心者の詰まりやすい点を重視すること，便利な実例を多く盛り込んでいることなどです．

なお，無料のリソースとして，pLaTeXを中心に，`jlreq`クラスを用いて組版することを学ぶのであれば，Learn LaTeXが非常に優れたサイトですので，こちらも参照してください．本稿では基本的にLuaLaTeX上の日本語組版（かつ`jlreq`使用）を前提としますが，パッケージ等についてはpLaTeXでも使用できるものもあるかと思います（LuaLaTeX専用のものは断ります）．また，LaTeX 2_ε の基本的な部分を学びたいのであれば，texdoc `lshort`[4]（古いですが，日本語版texdoc `jlshort`）が優れたドキュメントです．

なお，基本的なコンピューターの操作は前提として，脚注で触れる内容にはプログラミングを囓った人に伝わりやすい例を一部含みます．本文自体は，特別な前提知識を要求しません．(La)TeXの複雑な話については，  のような記号を付けて説明します[5]．初心者の方は読み飛ばしてください．

流れ

大まかな構成は以下の通りです．

- 第0節: そもそもLaTeXとは何か．環境構築など様々な話題
- 第1節: 極めて基本的な使い方（間違いやすい点を中心に）
- 第2節: パッケージ，相互参照，引用・参考文献
- 第3節: 間違いやすい点まとめ（ベストプラクティス）

---

1) 古くない，という意見もあろうかと思いますが，自分はLuaLaTeXしか使ったことがないので，許してください．
2) もちろんpLaTeXも素晴らしいエンジンですが，モダンなLaTeXの機能を活用するためにはLuaLaTeXが圧倒的に便利です．
3) これが当初の目的と言ってもいいでしょう
4) これらの記号の意味については，第0.3.3節で説明します．
5) 今のところ本文中に反映できていませんが．

リンク集

TeX Wiki https://texwiki.texjp.org/ 日本語TeXの情報が圧倒的に多いサイト．

CTAN https://ctan.org/ Comprehensive TeX Archive Network．TeX関連のパッケージやドキュメントが集まるサイト．

マクロツイーター https://zrbabbler.hatenablog.com/ 某ZR氏のブログ．

alg-d https://alg-d.com/math/tex.html 数学系VTuber(?)のalg-d氏のTeXページ，「TikZの使い方」は本稿で深く扱わないTikZの入門として最高．また，「TeXの理解を深める再生リスト」の動画も細かいところに踏み込んでいておすすめ．

TeX Alchemist Online https://doratex.hatenablog.jp/ doraTeX氏のブログ．

鉄緑会 https://www.tetsuryokukai.co.jp/links.html 同氏に関連して，鉄緑会公式サイトのリンク集にあるTeXのスライド資料は非常に有用です．

detexify https://detexify.kirelabs.org/classify.html 数学記号を手書き検索できるサイト．

他にも様々なサイトやドキュメントを参考にしました．先ほども触れましたが，TeX Wikiと[1]がなければこのドキュメントは成り立ちませんでした．パッケージ開発者も含め，TeXコミュニティーの皆様にここで改めて感謝申し上げます．

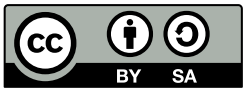
コラム 1：自己紹介

筆者（くりぷとん）はTeX歴は約3年程度で，LuaLaTeXを中心に使用しています．

コラム 2：ダウンロード先

このドキュメントはhttps://krp.todaiverse.org/latexからダウンロードできます[a]．
https://drive.google.com/file/d/1LBgUL8EJrlB-SAzGWaeemSI08oVfa0cr/view?usp=sharingからもダウンロードできるようにしていますが，こちらは更新しなくなるかもしれません．

a CTANに載せるつもりは今のところありません．第0.4.2節のフリーフォントの扱いが不明なこと，その他ヒラギノフォントで組んでいるのでそのままだとタイプセットできないこと，コードが壊滅していること，あとは面倒なのが理由です．



0 LaTeX

ここでは，LaTeXとは何か，TeXの処理系，環境構築などについて，初心者が混乱しやすく，他の情報源ではあまり触れられていない点を中心に説明します．

0.1 そもそもLaTeXとは何か–Typstなども見据えて

TeX[6]とは，Knuth[7]によって開発された組版システムです．当時，Knuthは写植[8]の品質に不満を持っており，自分が気に入るまで[9]，活字職人と同程度の品質を実現するためにTeXを開発しました．LaTeX[10]は，TeXをより使い

6) テックなどと読みます．テックスでも間違いとまでは言えません．慣用的にテフとも読まれます．日本語にはない発音なので，あまり厳しいことは言いません．
7) Donald Ervin Knuth, 計算機科学者，数学者．クヌースの矢印記法などが有名？
8) 鉛の活字ではなく，写真の原理を利用した印刷．
9) 文字通りのこだわり．
10) ラテック，レイテックなどと読みますが，特に発音の決まりはありません．

やすくするためにLamport[11]によって開発されたマクロパッケージ（言葉の意味は後述）です．TEXと断り無く言ったとき，大抵はLATEXを指します．逆に，何も入れていないまっさらな[12]TEXのことを指す時は，plain TEXなどと言うこともあります．

WordやInDesignといったソフトと大きく異なるのが，見た通りに書けば見た通りに出力されるWYSIWYG[13]ではなく，ソースコードをエディタで書いて，TEXエンジンでコンパイル（タイプセット）して出力ファイルを生成するという点です．このような仕組みをWYSIWYM[14]と呼びます．この方法は一見遠回りに見えますが，文章の意味と装飾を分離して取り扱うことができるため，文章の意味内容に集中でき，スタイルの一括変更が容易，バージョン管理 (Git など) が容易といったメリットがあります．Word の図挿入で苦労した人も多いでしょう[15]．歴史の長さも相まって，学術論文や技術文書の作成においてはデファクトスタンダードとなっており，特に数式を多用する分野ではほぼ必須と言えるでしょう．理工系の多くのジャーナルではLATEX以外の形式を受け付けていないというのが，この傍証です．

コラム 3：標準としてのLATEX，他のソフトウェアとの比較

LATEXが標準になっていることを述べましたが，さらにその例として，数式記法がLATEX準拠であることが多いことが挙げられます．Office, Apple製品では数式挿入にLATEX記法が使えます．Web上で数式を書く際にはMathJaxやKATEXなども使われますが，これらもLATEX記法です．

さて，組版向けにはもちろん他のソフトウェアもあります．例えば，最近注目されているTypst[a]などは，LATEXの欠点を克服しつつ，より直感的な記法を採用しており，今後の発展が期待されます．特に，LATEXの不自然な文法やタイプセットの遅さを嫌う人には魅力的でしょう．ただ，LATEXが組版を目的として，非常に細かい調整が可能なのに対し，Typstは楽になった分，そういった調整を省いているとも言えます．また，日本語組版やパッケージは未だ発展途上です．例えば短い授業レポートなどには適しているでしょう．

日本からのソフトウェアとしてはSATySFiがあります．静的型付けによるエラー報告などの特徴があります[b]．TEX系のソフトウェアでは，Tectonicという野心的なプロジェクトもあります[c]．TEXのシステムをRustで実装することを目的としており，使われているパッケージを自動でインストールし，XETEXでタイプセットまでしてくれます．

a https://typst.app/
b https://github.com/gfngfn/SATySFi
c https://github.com/tectonic-typesetting/tectonic

ConTEXtという，LATEXとは別系統のマクロパッケージもあります．日本語組版の情報が不足しているため，本稿では扱いません．

0.2 TEXの処理（系）

さて，TEXの処理の流れを簡単に説明します．

1. ソースコード（.texファイル）をテキストエディタで作成する
2. TEXエンジン（pLATEX，LuaLATEXなど）でコンパイルする
3. DVI，PDFなどの出力ファイルが生成される

plain TEXは極めて基本的な機能しかもっていないため，TEX言語で様々なコマンドを定義する[16]LATEXが上に載って

11） Leslie B. Lamport
12） このあたりの用語法もかなり怪しい．plain TEXはKnuthが自著のために作ったマクロパッケージを指すので，厳密には誤り．iniTEXという，パッケージのビルドなんかに使う本当に裸のTEXもあります．
13） What You See Is What You Get.
14） What You See Is What You Mean.
15） まあ，LATEXにも別の種類の苦労はあるわけですが．
16） こういったものをマクロといいます．

私たちが書くあの“ラテック”になっています．さらに，数式や図，表などのその他機能を追加するために，パッケージという拡張モジュールを読み込むことができます．パッケージの追加で機能は拡張できるとは言え，1978年に公開され，バグフィックス以上のアップデートがされていないTEXエンジンには限界があります．例えば，Unicodeには（当然）対応していないですし，内部状態が少なく，大量の機能を扱えないこと，PDFがそもそもなかったので，独自のDVI形式を出力することなどが挙げられます．これらの問題を解決するために，TEXエンジンは改良されていきました[17)]．

さて，当然TEXは英語圏の出版を前提に作られたものですから，日本語はそのままでは扱えません．ASCIIが開発したpLATEX，pTEXが，長らくデファクトスタンダードでした．それを拡張[18)]，Unicode対応したのがε-upTEXです．その一方，英語圏では「モダンLATEX」などと呼ばれる2つの新しいTEXエンジン，XELATEXとLuaLATEXが登場しました．これらはUnicode対応であり，システムフォントを直接利用できる，PDFを直接出力できるなどと，現代的で便利な機能を備えています．特に，LuaLATEXはpdfTEXという，現在英語圏で主流のエンジンをベースにしており，今後のデファクトスタンダードになると考えられています．加えて，LuaLATEXはLuaという組み込み用スクリプト言語を内蔵しており，難解であることに定評のあるTEX言語などを扱う必要が無い，数値計算が容易などの利点があります．日本語対応についても，LuaTeX-jaプロジェクトによってLuaLATEXで日本語を扱うためのパッケージが整備されており，現在ではLuaLATEX+LuaTeX-jaを用いるべきだと言えるでしょう[19)]．

さらに，`jlreq`クラス[20)]という新しいクラスが柔軟性も高く，現代的な日本語組版を実現しており，LuaLATEXと組み合わせることで，より良い日本語組版が可能となると思います．

XELATEX,pdfTEXも某ZR氏他により日本語組版が可能になっていますが，禁則が甘いとかの問題がある~~そもそもオタク向けな~~ので扱いません[21)]．

コラム 4：なぜLuaLATEXなのか

(u)pLATEXは日本語組版に長らく使われてきたエンジンですが，以下のような欠点があります．

- フォント設定が貧弱
- pLATEX特有の実装と本家LATEXの更新が互換しない可能性大（メンテナー不足）
- DVIドライバ
- `plautopatch`,`pxjahyper`などのパッチが必要

一方，LuaLATEXは以下のような利点があります．

- （pdfTEXの）利用者が多く，パッケージの互換性が高い
- Unicode（ネイティブ）対応
- システムフォントを直接利用可能
- PDFを直接出力可能（PDFの機能にアクセスできるため，図や参照の相性がいいらしい）
- Lua言語による拡張性（Ti*k*Zとの相性もいいらしい）
- LuaTeX-jaによる日本語組版対応

欠点としては，コンパイル時間が2倍程度かかります（各（マクロ）パッケージの努力で改善されてはいます）[a]．ただ，この点はフォントの読み込み等に時間がかかっていることもあり，利便性と一長一短といえるでしょう．
また，今ではどうか知りませんが，学会によってはpLATEXのテンプレートしか配布していない（upLATEXです

17） いわゆるフォーク以外の改変をKnuthは許していないため，別の名前で呼ばれます．こういった改良版を処理系などといいます．
18） ε-TEX拡張と呼ばれるものです．ここでは説明を省略．
19） なお，欠点を挙げておくと，コンパイルに時間がかかることが（結構）あります．
20） W3C日本語組版要件に準拠したクラス．東大の阿部紀行先生開発．
21） 同氏のクラスを入れればpLATEXと同じような出力を得ることは可能だが，実用は困難．

らない）こともあるらしく，そのような場合はpLaTeXを使う必要があるかもしれません．

a 特に，初回インストール直後はシステムフォントのデータベースLuaotfloadの処理が入るのではちゃめちゃに遅いです．

コラム 5：LaTeX・TeXのまとめ

主なTeXエンジンをまとめると，以下のようになります．

TeX Knuthによるオリジナル．ASCIIのpTeXの基礎．
ε-TeX 拡張TeX．様々な機能拡張がなされている．
pdfTeX PDF出力に対応した比較的新しいTeXエンジン．英語圏で主流．
XeTeX Unicode対応，システムフォント利用可能，PDF出力可能なエンジン．
LuaTeX Unicode対応，システムフォント利用可能，PDF出力可能なエンジン．Lua言語を組み込み可能．pdfTeXの後継
pTeX 日本語対応のためにASCIIが開発したTeXエンジン．
upTeX Unicode対応，ε-TeX拡張を取り入れたpTeXの改良版．
ε-upTeX ε-TeX拡張を取り入れたupTeXの改良版．

これらに対して，LaTeXが移植されたものがpLaTeX，upLaTeX，XeLaTeX，LuaLaTeXなどです．例えば，コマンドラインで`tex`と打つと

```
This is TeX, Version 3.141592653 (TeX Live 2026) (preloaded format=tex)
**〔入力待ち〕
```

のように表示されます．一方，`latex`と打つと

```
This is pdfTeX, Version 3.141592653-2.6-1.40.29
(TeX Live 2026) (preloaded format=latex)
 restricted \write18 enabled.
**〔入力待ち〕
```

と出力されます．これは，pdfTeXのTeX互換モードで，LaTeXを読み込ませるように設定されたものです．
また，`platex`と打つと

```
This is e-upTeX, Version 3.141592653-p4.1.2-u2.02-251130-2.6
(utf8.sjis) (TeX Live 2026) (preloaded format=platex)
 restricted \write18 enabled.
**〔入力待ち〕
```

のように，ε-upTeXの上にpLaTeXが載っていることがわかります．
このように，多くの場合で後継（拡張版）とみなされるエンジンの互換モードで処理されます．実際には．`latex`がpdfTeXの互換モードのシンボリックリンクになっています[a]．逆に言えば，pdfLaTeXで処理をしたいのであれば，`pdflatex`などと呼び出す必要があります．

a UNIX系OSで`sh`が`bash`へのリンクになっているとかそういう感じですね．

コラム 6：用語の整理

TEX エンジン，LATEX クラス，LATEX パッケージなど，様々な用語が混在しているため，ここで整理しておきます．

TEX エンジン（処理系） TEX, pTEX, pdfTEX, LuaTEX, XeTEX などの実装．TEX の処理を行うソフトウェア．
LATEX TEX の上で動くマクロ集．各処理系に対応したものがそれぞれ pLATEX, LuaLATEX など．
LATEX クラス 文書のスタイルを定義するもの．`article`, `jlreq` など．
LATEX パッケージ クラスに追加して機能を提供するもの．`amsmath`, `hyperref` など．
（TEX）プリミティブ TEX エンジンが提供する基本的なコマンド．`\def`, `\hbox`, `vskip` など．
（LATEX）マクロ TEX プリミティブを組み合わせて定義されたコマンド．`\newpage`, `\section` など．
（LATEX）環境 LATEX マクロの一種で，`\begin{env}` と `\end{env}` で囲まれた部分に適用されるもの．`itemize`, `figure` など．この中では外とは違うコマンド・組版ルールが適用される．

0.3 環境構築

0.3.1 Overleaf, Cloud LaTeX

これらについては，既に多くの情報があるため，ここでは割愛します．ただし，pLATEX を前提とした記事が多いため，適宜読み替える必要があります．

ローカルでコンパイルするより遅く，バグの対処も難しくなるため，あまりおすすめしません．ただ，TEXLive は容量も大きいですし，初心者が試しにいろいろやってみるのには強くおすすめできますし，共同編集や簡単にコードとドキュメントをすぐに共有できるというメリットもあります．

Overleaf の場合は，プロジェクトを作成後，左下の⚙から Settings を開き，Compiler の pdfLaTeX を LuaLaTeX（こちらでは pLATEX は使用不可）に変更してください．Cloud LaTeX の場合は，新規プロジェクトを作る際に「LaTeX エンジン」で LuaLaTeX を選択すると，`jlreq` クラスを使うようにセットされたひな形が作成されます．upLaTeX も選択可能です．

0.3.2 ローカル環境構築

TEXLive をインストールしてください．これについては，TEX Users Group からインストーラーをダウンロードし，指示に従ってインストールしてください．なお，インストールにはかなり時間がかかります．なお，MiKTEX や MacTeX といった他のディストリビューションもありますが，ここでは扱いません．

詳しい説明はインストールガイド，または texdoc `texlive` のドキュメントを参照してください．日本語版（texdoc `texlive-ja`）もあります[22)]．

スキームと呼ばれるセットが用意されていますが，日本語で組版するためのスキームはないため[23)]，scheme-full で全てをインストールしてください．OS にもよりますが，10 GB 程度は必要です．時間や容量に余裕がないのであれば，他のスキームでインストールしたあと，collection-langjapanese をインストールすればよいでしょう．

TEXLive の管理，アップデートは tlshell などのツールを用いて行います．定期的なアップデートをおすすめします．コマンドラインに忌避感がなければ，以下のコマンドでアップデートできます．

22） 最新版は：https://wtsnjp.com/pdf/texlive-ja.pdf
23） medium なんかは欧州の基本言語が入っているらしいですが，日本語は……

コード 1：tlmgrによるTEXLiveのアップデート

```
tlmgr update --self --all
```

インストール環境によっては管理者権限が必要です．

0.3.3 ドキュメント

このようにインストールすると，当然ながら付属のドキュメントもインストールされます．

`texdoc topic`とコマンドラインで打つと，topicに関するドキュメントが開きます．例えば，`texdoc jlreq`と打つと，jlreqクラスのドキュメントが開きます．日本語関係ではないパッケージのドキュメントは大抵英語なので，頑張って読みましょう．

本稿では，[1]にならって，texdoc topicという記号でtexdocからドキュメントが読めることを表記します．例えば，texdoc jlreq-jaと書かれていたら，コマンドラインで`texdoc jlreq-ja`と打ってドキュメントを読め，ということです．加えて，長いドキュメントについてはtexdoc topic (section)のように，どこを読めばよいかも示すことがあります．

また，texdoc topicと書いてあっても，`texdoc topic`と打ってもドキュメントが見つからないこともあります[24)]．その場合は，`texdoc -s topic`と打ってみてください．部分一致でも全てのドキュメントが表示されるので，そこから探すことができます．普通に一覧表示をしたい場合（日英の選択など）は，`texdoc -l topic`と打ってください．

これのオンライン版がhttps://texdoc.org/です．ここでは，texdocをクリックすると，texdoc.orgでドキュメントが開くように設定しています．

コラム 7：LATEX 2ε のドキュメント

コラム9で説明するLATEX3のプロジェクトにつながる話ですが，LATEX 2ε そのもののドキュメントは壊滅しています．文書化されていない仕様があるようです．そのため，texdoc latexを見ると，非公式なドキュメントのリンク集が表示されます．もっとも重要なのはtexdoc latex2eという非公式のマニュアルです．一方，LATEX3に関するドキュメントは充実しています．texdoc usrguideを見れば，expl3に関する（組版するだけなら）必要な情報はしっかりまとまっていますし，細かい仕様はtexdoc interface3から見ることが出来ます．

コラム 8：CTANとTEXディストリビューション

CTAN[a] (Comprehensive TeX Archive Network)は，TEX関連のパッケージやドキュメントを集めたリポジトリです．TEXLiveやMiKTEXなどのTEXディストリビューションは，CTANからパッケージを取得してインストールしています．誰でもなんでも登録が可能です．ここに入っているもので，各ディストリビューションの基準を満たすものをまとめて配布しているわけです．TEXLiveが主流ですが，ライセンスに対する考え方が厳しめで，既定ではFLOSSに該当するようなものだけがインストールされます．また，基本的にパッケージの破壊的なアップデートは年1回の新規リリースで行われます．執筆当時（2026年2月）はTEXLive 2025が最新で，2026はpretest（アルファ版）段階です．リリースを変更するには，手動でtlmgr自体をアップデートする必要があるので，これを避けたい人はMiKTEXなどを使うとよいでしょう．

a https://ctan.org/

24) ごめんなさい

0.3.4 エディタ

エディタは好みのものを使ってください．Visual Studio Code[25]やTeXworks[26]，TeXstudio[27]などが有名です．なお，TeXworksはTEXLiveに同梱されています．~~TEX~~鉄緑会ではmac専用のTeXShop[28]を使っているようです．

コマンドラインでコンパイルすることに忌避感がないのであれば，メモ帳であろうがNotepad++だろうがVimだろうがEmacsだろうが何を使ってもよいです[29]．ただ，VS CodeやLATEX用のエディタ/IDEを使えば，賢い自動補完，ワンボタンでタイプセット，エラーの可視化，PDFを2面表示などいろいろなことができますから，そちらをおすすめします．

VS Codeを使わないのであれば，第一選択肢はTeXstudioがよいでしょう．VS Codeは拡張機能を入れる必要があるため，少し手間がかかりますが，拡張機能の充実度やカスタマイズ性の高さから，慣れてしまえば非常に快適に使えると思います．

VS Codeの場合を中心に説明します．

LaTeX Workshop, LaTeX Utilitiesなどの拡張機能をインストールしてください．基本的にはデフォルト設定で十分動きます．ただ，VS Codeに対して，「」のような日本語約物が語を分割するものとして設定するだとか，タイプセットの設定だとかを実施する必要があります．この詳細については付録Aを参照してください．また，デフォルト設定だと`latexmk`を使用します（付録B）．

この`latexmk`は，何度もコンパイルを繰り返して参照関係を解決したり，BIBTEXや`makeindex`,`upmendix`（詳しくは第2.7節）を自動で呼び出したりしてくれる便利なツールです．

0.4 実際，(Lua)LATEXだと何ができるのか？

0.4.1 LATEXだと

他の組版システムでも同様ですが，そもそも文章の組版がかなり美しいです．手動で調整しなくても自動的に字間や行間が調整され，見栄えの良い文章が得られます．例えば，

> The quick brown fox jumps over the lazy dog. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

とか，

> あのイーハトーヴォのすきとおった風、夏でも底に冷たさをもつ青いそら、うつくしい森で飾られたモリーオ市、郊外のぎらぎらひかる草の波。

のような文章を美しく組版できます．これも当然ですが，改行を入れずにベタ打ちするだけで，句読点などを回避して（禁則処理）自動的に改行してくれます．加えて，数式が美しいことが挙げられます．例えば，次のような数式を簡単に記述できます．

$$E = mc^2 \tag{1}$$

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \tag{2}$$

$$\int_a^b f(x)\,dx = F(b) - F(a) \tag{3}$$

$$e^{i\pi} + 1 = 0 \tag{4}$$

のように書けます．並べる環境における揃えも自動的に行われます．ただ，このあたりはWordのフォントが悪いと

25） https://code.visualstudio.com/
26） https://www.tug.org/texworks/
27） https://www.texstudio.org/
28） http://pages.uoregon.edu/koch/texshop/
29） ただし，文字コードは適切に設定する必要があります．

いう点が大きいので，基本的な数式だと大きな違いは無いかもしれません[30]（第1.6.1節も参照）．

複雑な数式の例として，

コード 2：数式の例

```
\begin{align}
  \nabla^2 \phi &= \frac{1}{r^2} \pdv{}{r} \left( r^2 \pdv{\phi}{r} \right)\\
  &+ \frac{1}{r^2 \sin\theta} \pdv{}{\theta} \left( \sin\theta \pdv{\phi}{\theta} \right)\\
  &+ \frac{1}{r^2 \sin^2\theta} \pdv[2]{\phi}{\varphi}
\end{align}
```

と書くだけで[31]，

$$\nabla^2 \phi = \frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial \phi}{\partial r} \right) \tag{5}$$

$$+ \frac{1}{r^2 \sin\theta} \frac{\partial}{\partial \theta} \left(\sin\theta \frac{\partial \phi}{\partial \theta} \right) \tag{6}$$

$$+ \frac{1}{r^2 \sin^2\theta} \frac{\partial^2 \phi}{\partial \varphi^2} \tag{7}$$

のように，美しい数式が得られます．相互参照（第2.7節）の便利さは，索引や参考文献，先ほどから使っている章・式番号（手で番号を入れているわけではない）などに現れています．

0.4.2 LuaLaTeXだと

まず，LuaLaTeX-jaがかなり充実しており，日本語組版ではpLaTeXに匹敵する品質であり，さらに柔軟性が高いです．

Unicode対応なので，特殊記号を直接入力できます[32]．例えば，

☃っ☃☃☃っ☃っ☃〜〜♪

$\alpha\beta\gamma\delta\varepsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi o\pi\rho\sigma\tau\upsilon\varphi\chi\psi\omega$

のように，直接入力できます．なお，数式モードの中での対応にはunicode-mathパッケージ（後述第2.3.3節）が必要です．

Luaを内蔵しているため，例えば，

実行例 1：Luaの呼び出し

```
\(\sqrt{12345}\)
は
\directlua{tex.sprint(math.sqrt(12345))}.
```

$\sqrt{12345}$ は111.10805551354.

のように，Luaの機能を直接呼び出せます．Luaの機能が呼び出せるので，パッケージ開発も容易になっています．LuaTeX-jaの恩恵もありますが，旧来のTeXが苦手としていた複数行にわたる下線も，`lua-ul`，その和文縦書き向け拡張である`luwa-ul`パッケージを使うことで，

30） Wordで出した気持ち悪い数式を貼りたくないので，Xの「格付けチェック」を参照：https://x.com/2_wykipedia/status/1609494821487677441

31） これを「書くだけ」とは何事か，と思った人も多いかもしれませんが，基本構文を覚えてしまえば，あとは自動補完やAIが大半をやってくれます．

32） 厳密には`fontspec`が必要だが，第2.1.1節で見るように事実上普通に．

実行例 2：下線

```
\underLineKKAuto{
  あのイーハトーヴォのすきとおった風、
  夏でも底に冷たさをもつ青いそら、
  うつくしい森で飾られたモリーオ市、
郊外のぎらぎらひかる草の波。}
```

あのイーハトーヴォのすきとおった風、夏でも底に冷たさをもつ青いそら、うつくしい森で飾られたモリーオ市、郊外のぎらぎらひかる草の波。

のように，簡単に実現できます．

さらに，フォントを柔軟に変えることが可能です．例えば，

実行例 3：フォント

```
\setmainjfont{EVA-Matisse_Classic}
新世紀エヴァンゲリオン

\setmainjfont{Hangyaku}
魔法少女
```

新世紀エヴァンゲリオン

魔法少女

のように，システムにインストールされているフォントを簡単に利用できます．そんなのは当たり前だろうと思うかもしれませんが，pLaTeX（というより旧来の TeX エンジン）では，フォントを利用するための設定が非常に面倒でした．当然のことが当然にできる，LuaLaTeX を使うべき理由の一つです．

部分縦組も柔軟に実現できます（pLaTeX の移植）．例えば，

実行例 4：部分縦組

```
\begin{center}
  \begin{minipage}<t>{0.3\textwidth}
    あのイーハトーヴォのすきとおった風、
    夏でも底に冷たさをもつ青いそら、
    うつくしい森で飾られたモリーオ市、
    郊外のぎらぎらひかる草の波。
  \end{minipage}
\end{center}
```

あのイーハトーヴォのすきとおった風、夏でも底に冷たさをもつ青いそら、うつくしい森で飾られたモリーオ市、郊外のぎらぎらひかる草の波。

となります．

コラム 9：expl3 とは

(La)TeX のマクロ言語は非常に難解で，(La)TeX の処理系を直接扱うのは初心者にはかなりハードルが高いです．元々，古くなった LaTeX2 を捨てて，（互換性のない）LaTeX3 を作ろう，という試みから出てきたマクロ言語が expl3 です．まあ，方針転換があり，その間の LaTeX 2_ε をアップデートしていく形で進化していくことになりました．

そのため，今の LaTeX には expl3 が組み込まれており，パッケージ開発などで利用されています．以前は，`expl3` というパッケージを読み込まなければ使えませんでしたが，2020 年くらいには LaTeX 2_ε に統合されました．例えば，ビルドのログには L3 programming layer という表記が出てきますが，これが expl3 です．

ちなみに，読み方は「エクスペル スリー」です．ここからもわかるように，これは実験的[Experimental]な，という意味でした．今では LaTeX 3 programming language の略ということになっています．

なお古い文献では，LaTeX 2_ε や LaTeX3 を区別していることもありますが，今（2020 年以降）では LaTeX 2_ε と LaTeX3 とは事実上同一のものになっています．

1　基本的な使い方

1.1　ドキュメントクラスとコマンド

さて，実際に簡単な使い方を説明します．

ドキュメントクラスは，文書の構造を規定するものです．表1に代表的なドキュメントクラスを示します．

表1: 代表的なドキュメントクラス．[1]を引用．

用途	欧文	和文 (upLaTeX)	和文 (LuaLaTeX)	和文 (新)
論文・レポート	article	jsarticle	ltjsarticle	jlreq
長い報告書	report	jsreport	ltjsreport	jlreq
本	book	jsbook	ltjsbook	jlreq

レポートと長い報告書の違いは自明でしょう．具体的な違いとしては，章立て（\chapterが最上位か，\sectionか）などがあります．本に関しては，これも当然のことではありますが，両面印刷で綴じることを前提としており，章の開始が奇数ページになるように調整され，ノド[33]が広くなるようページレイアウトが設定されています．

なお，jlreqは全て共通ですが，オプションで指定します（既定はarticle）．

ここで，今更ですがコマンドについて説明します．コマンドは\commandname[オプション]{引数}のように書きます．日本語環境（特にWindows）では，\が円マーク¥に見えることがあります．これは歴史的都合ですので，Windowsの日本語版では半角の円マークを入力すれば問題ありません[34]．macでは，設定でどちらを入力するか切り替え可能で，Optionキーを押しながらでもう一方の入力がなされます．当然，全角の円マーク￥やバックスラッシュ＼は別物ですから注意してください．Wikipedia「円記号」も参照．

ついでに特殊記号についても説明しておきます．以下の文字は，そのまま書くと特別な意味を持つため，エスケープする必要があります．

- #: \#
- $: \$
- %: \%
- &: \&
- _: _
- {: \{
- }: \}
- ~: \textasciitilde{} または \~{}
- ^: \textasciicircum{} または \^{}
- バックスラッシュ\: \textbackslash{}

バックスラッシュ\と{ }はコマンドに使われ，#は引数指定（後述第1.5節），$は数式モード（後述第1.4節），&は揃え記号（後述第1.4節），^と_は上付き，下付き（後述第1.4節）に使われるため，そのままだと使用できません．加えて，^と~はアクセント記号（後述第1.3節）にも使われます[35]．

33） 綴じている側の余白．
34） ASCIIとSJISの都合．
35） なので，{}をつけないと，次の文字にアクセントを付けようとして失敗します．

%はかなり重要で，コメントアウトに使われます．これ以降の行末まで（改行記号などを含む）を無視します（コマンドの定義でも使えます．後述第1.5節）．

なお，これらの文字は全角で入力しても使えますが，フォントが異なるものになるため，注意が必要です．例えば%と％は異なります（前者は欧文フォント，後者は和文フォント）．

1.2 文書の構造

LaTeX文書は，まず前述のドキュメントクラスを指定することから始まります．

`\documentclass[tate,twocolumn,book]{jlreq}`

この例では，`jlreq`クラスで，縦書き，2段組，本のスタイルで文書を作成することを指定しています．他に欧文和文のフォントサイズの指定などが可能です（第2.1.2節）．

次に，`\begin{document}`…`\end{document}`を書きます．この間が本文にあたります．

`\documentclass{}`から`\begin{document}`までの間には，プリアンブルと呼ばれる部分があり，ここでパッケージの読み込みや各種設定を行います．

まとめると

コード 3：文書の構造

```
\documentclass[オプション]{ドキュメントクラス}
% プリアンブル
\begin{document}
% 本文
\end{document}
```

のようになります．

とりあえず，オプションとプリアンブルを空にして，ドキュメントクラスを`jlreq`にした場合，以下のようになります．

コード 4：文書の構造の例

```
\documentclass{jlreq}
\begin{document}
ここに本文を書く．
This is a sample document.
```

ここで1つ，注意しておかなければならないのは，LaTeXではソースコード上の1回の改行は，原則として半角スペース1つ分として扱われます（ただし，日本語環境では和文文字間の改行は完全に無視され，余計な空白は入りません）．改行，空行を挟むことで初めて段落の区切り（改段落）として認識されます．そのため，ソースコードを読みやすくするため適当に改行を挟みましょう．単純に改行したい場合は，`\\`や`\par`を用い，空行は段落の区切りになります．また，半角空白に関しては1つまでしか認識されません．例えば，

実行例 5：スペース

```
This␣is␣a␣sample␣document.

This␣␣␣␣is␣␣␣a␣␣␣sample␣␣␣document.

This\␣\␣\␣is\␣\␣\␣a␣sample␣document.

~は改行禁止のスペースです.
```

This is a sample document.
This is a sample document.
This is a sample document.
は改行禁止のスペースです.

のように，複数のスペースを入れても，出力されるのは 1 つ分のスペースだけになります．明示的に空白を入れたい場合は，\␣（バックスラッシュ＋スペース）[36] などを使います． さらに，~で改行禁止の空白を入れることもできます．他にも細かい空白コマンドはありますが，次節（第 1.4.5 節）で説明します．

コラム 10：改ページや改行の細かい話

改行の命令には，\\や\linebreakや\newlineなどがあります．
まず，\\，*は単純に改行を入れます．*は改ページを禁止するオプションがついています．実はこれらにはオプション引数があり，\\[長さ]のように書くと，改行後に指定した長さ分の空白が入ります．例えば，\\[1em]と書くと，改行後に縦 1em 分の空白が入ります．
\parは，そもそも LaTeX のマクロではなく，TeX のプリミティブで，空行と同じ意味になるはずです[a]．
\newlineは，基本的には\\と同じ動作をします．第 2.6 節で説明する tabular や array では\\は改行というより表での行の区切りを意味するコマンドですので，この場合は\newlineが直に改行をするために使われます．
\linebreakは，改行を入れたい場所で使います．\nolinebreakは，改行を入れたくない場所で使います．これは改行をせよ，という命令ではなく，（実はオプション引数を指定して）LaTeX の改行のアルゴリズムに対して，ここで改行してもいいですよ，あるいはここで改行しないでください，という指示を出すものです．
改ページの命令には，\pagebreak，\newpage，\clearpage などがあります．\newpage と\clearpage は，どちらも改ページの命令で，ほとんど同じ動作をしますが，\clearpageはフロート（後述第 2.6 節）が残っている場合はそれも出力してから改ページするのに対し，\newpageはフロートを無視して改ページします．また，多段組を行っている場合は，\newpage は次の段に移るのに対し，\clearpage は次のページに移ります．簡単に言えば，\newpage よりも\clearpageの方が強い改ページになっているということです．
なお，\pagebreak と nopagebreak は，先ほどの\linebreak と\nolinebreakの改ページバージョンで，改ページを推奨または抑止する命令になります．実際のところ，これらの命令は，普通に使えば単なる改行や改ページの命令になりますが，オプション引数を指定することでより柔軟に指示することが可能です．

a verbatim環境内を除く．

1.3 コマンドに関する注意・特殊文字

ここまでで書くのを忘れていましたが，コマンド名はケースセンシティブ（大文字・小文字を区別）です．例えば，\latex と\LaTeX は別物です．

普通の文章（地の文）でコマンドを使う際，

\LaTeXは素晴らしい．

と書くと，“\LaTeX は素晴らしい” という一つのコマンドを呼び出そうとしてしまい，エラー（第 1.7 節で説明しています）になります．この場合，\LaTeX{}は素晴らしいと書くか，\LaTeX␣は素晴らしいと書く必要があります．なお，\%のような特殊記号については，\%fooのようなコマンドが定義されることは絶対にない[37] ので問題ありません．

アクセントなどの特殊文字について，例えば，シュレーディンガーなら Schr\"odinger と書くと，Schrödinger のように出力されます．ただ，Unicode に対応している LuaTeX,XeTeX なら，Schrödinger のように直接書いても認識されますので，ここではそれらのコマンドについては省略します．

また，引用符についても注意が必要です．日本語では「」や『』を使い，これは直接入力すれば問題ありません．欧文では，``double quotes'' や `single quotes' のように書きます[38]．単純に"double quotes"や'single quotes' と書くと，”double quotes” や’single quotes’のように出力されてしまいます．

また，相当お行儀の悪い記法として，次のような書き方があります．

36） この␣記号は，スペースを明示したいときに用います．
37） 予約されている記号は変数名に使えない，というような話です．
38） JIS配列だとこれ`はShift+@で入力できます．

実行例 6：お行儀の悪いコマンドの例

\(\frac12\)と\(\frac{1}{2}\)	$\frac12$と$\frac{1}{2}$

引数を波括弧{}で囲まないと，\fracの次のトークン（この場合は1）だけが引数として認識されてしまいます．これを逆手に取って，このような書き方も可能になっていますが，可読性が最悪なのでやめておきましょう．

コラム 11：カテゴリーコード

TEX は，入力された文字に対してカテゴリーコードを割り当てています．例えば，\はコマンドの開始を意味するカテゴリーコード 0，{はグループの開始を意味するカテゴリーコード 1，}はグループの終了を意味するカテゴリーコード 2，%はコメントアウトを意味するカテゴリーコード 14 などがあります．
上のような波括弧の省略も，このカテゴリーコードの取り扱いを逆手に取ったものです．
また，第 1.5 節や第 2.7.4.1 段落でも現れる，\makeatletter も，一時的に@を普通の英数字と同じカテゴリーコードに変更するコマンドとなっています．LATEX では，この@をコマンド名に使うものとして予約することで，名前空間の分離を行っています．逆に言えば，内部的なコマンドを書きたいとき，LATEX やパッケージの内部で使われている定義の上書きをしたい場合は，\makeatletter を使って@のカテゴリーコードを一時的に変更する必要があります．

1.4 基本的な組版

これで，ベタ打ちによって文字を出力することは大体可能になったと思います．しかし，レポートや書籍というのはいろいろな構造があるもので，そういったもの出力を解説します．

1.4.1 章立て

章立ては，\chapter，\section，\subsection，\subsubsectionなどのコマンドで行います．例えば，

```
\section{背景}
\subsection{目的}
```

のようにすることでセクション，サブセクションが作成されます．\part で第 n 部を作ることができ，report や book に対応するクラスであれば，\chapter で章を作ることも出来ます．\subsubsection,\paragraph,\subparagraph というより細かい区分もあります．ただし，\paragraph よりも深い区分は使用しない方が読みやすいと思います．

1.4.2 タイトル・著者名・日付

タイトル，著者名，日付は，\title，\author，\date で指定し，\maketitle で出力します．指定する部分は（普通）プリアンブルに書いて[39]，本文中の出したい場所に\maketitleを書きます．

著者が複数いる場合は，\and で区切り，所属などを脚注の形で書くには，\thanks を使います．例えば，この文書では

コード 5：タイトル・著者名の例

```
\title{シケプリ・レポート作成のための\LaTeX{}入門}
\author{くりぷとん\thanks{\url{https://krp.todaiverse.org},
Twitter: \href{https://x.com/8A4MB2}{@8A4MB2}}}
```

のようにしています．なお，\url については第 2.7.3 節で説明します．日付は省略するとコンパイルした日付が自

39） \maketitleの前ならどこでも構いません．

動的に入ります．

これは\today がデフォルトになっています．なお，日付を入れたくない場合は，\date{}と空にしてください．これを本文中で呼び出すこともできます．例えば，2026年4月14日．

日本語クラスには\和暦や\西暦というマクロがあり，\today の表記法を西暦，和暦の間で変更することができます．例えば，{\和暦\today}のようにすると，令和8年4月14日のように出力されます．

1.4.3 書体・文字サイズ

文字の書体（太字，斜体，等幅など）やサイズを変更するには，以下のコマンドを使います．

欧文書体については，\textrm（ローマン体），\textbf（太字），\textit（イタリック体），\textsl（スラント体），\textsf（サンセリフ体），\texttt（等幅体），\textsc（スモールキャップ体）などがあります．例を下に示します．

実行例 7：欧文書体

```
デフォルトの書体（ローマン）
\textrm{This is Roman.}\\
太字
\textbf{This is bold.}\\
イタリック体
\textit{This is italic.}\\
スラント体（斜体）
\textsl{This is slant.}\\
サンセリフ体
\textsf{This is sans-serif.}\\
等幅・タイプライター体
\texttt{This is typewriter.}\\
スモールキャップ体
\textsc{This is small caps.}
```

デフォルトの書体（ローマン）This is Roman.
太字**This is bold.**
イタリック体 *This is italic.*
スラント体（斜体）*This is slant.*
サンセリフ体This is sans-serif.
等幅・タイプライター体`This is typewriter.`
スモールキャップ体This is small caps.

和文書体については，\textgt（ゴシック体），\textmc（明朝体），\textbf（太字）などがあります．例を例8に示します．

実行例 8：和文書体

```
\textmc{これは明朝体です．}\\
\textgt{これはゴシック体です．} \\
\textbf{これは太字です．} \\
\textbf{\textgt{これはゴシック太字です．}}
```

これは明朝体です．
これはゴシック体です．
これは太字です．
これはゴシック太字です．

フォントや環境によっては，太字がゴシック体（の太字）になることがあります．これは日本語組版では強調(\emph) をゴシックにすることが多いことにも由来しているでしょう（イタリックとはいきませんし）．このあたりは表記の哲学として第3.1.1節で論じます．

さて，第1.5節でも説明するように，これらのコマンドを強調の意味で使うのは避けるべきです．見た目ではなく，その論理的意味を表すコマンドとして，LaTeX は\em や\emph を用意しています．デフォルトでは欧文のイタリックなどの処理のみとなっていますので，プリアンブルに以下を書くことで[40] 和文も処理できるようになります．

40） ただし，第2.1.1節で触れる多書体対応を仮定．

コード 6：和文の強調

```
\DeclareEmphSequence{% \emphの多重使用について，書体を設定する
  \itshape\gtfamily,          % 1階層目：欧文=イタリック，和文=ゴシック
  \upshape\bfseries\gtfamily, % 2階層目：欧文=立体・太字，和文=ゴシック・太字
  \itshape\bfseries\mcfamily  % 3階層目：欧文=イタ・太字，和文=明朝・太字
}
```

文字サイズについては，\tiny，\scriptsize，\footnotesize，\small，\normalsize，\large，\Large，\LARGE，\huge，\Hugeなどがあります．例を例9に示します．

実行例 9：文字サイズ

```
{\tiny これは tiny です．}\\
{\scriptsize これは scriptsize です．}\\
{\footnotesize これは footnotesize です．}\\
{\small これは small です．}\\
{\normalsize これは normalsize です．}\\
{\large これは large です．}\\
{\Large これは Large です．}\\
{\LARGE これは LARGE です．}\\
{\huge これは huge です．}\\
{\Huge これは Huge です．}\\
```

これは tiny です．
これは scriptsize です．
これは footnotesize です．
これは small です．
これは normalsize です．
これは large です．
これは Large です．
これは LARGE です．
これは huge です．
これは Huge です．

また，これらを組み合わせることも可能です．例えば，{\large \textbf{大きな太字}}とすれば，**大きな太字**となります．

また，長大な文書を作成する際，ファイルを分割して管理したいこともあるでしょう．この場合は，\input や\includeを使います．他にも texdoc subfiles や texdoc import パッケージもあります．

1.4.4 環境

\begin{hoge}…\end{hoge}の形で囲まれた部分を環境と呼びます．この中では，特別な処理が行われます．例えば，箇条書き環境であるitemize環境は，

実行例 10：itemize環境

```
\begin{itemize}
  \item 箇条書きの例です．
  \item さらに項目を追加できます．
  \item あああああああ
\end{itemize}
```

- 箇条書きの例です．
- さらに項目を追加できます．
- あああああああ

のように出力されます．他にも，enumerate環境（番号付き箇条書き），description環境（説明付き箇条書き）などがあります．説明付き箇条書きは例えば，

実行例 11：description環境

```
\begin{description}
\item[運動の第一法則]　慣性の法則のことです.
\item[運動の第二法則]　運動方程式のことです.
\item[運動の第三法則]　作用反作用の法則のことです.
\end{description}
```

運動の第一法則　慣性の法則のことです.
運動の第二法則　運動方程式のことです.
運動の第三法則　作用反作用の法則のことです.

のように出力されます.

箇条書きのラベルを変更したい（・ではなく♣に，1,2,3,...ではなく問1,問2,問3,...にしたい）場合は，enumitemパッケージを使用すると便利です（第2.6.2節）.

箇条書きについては入れ子にすることで下位の箇条書きを作成できます．例えば，

実行例 12：itemize環境の入れ子

```
\begin{itemize}
  \item トップ
    \begin{itemize}
      \item サブ1
      \item サブ2
    \end{itemize}
  \item トップ2
\end{itemize}
```

- トップ
  - サブ1
  - サブ2
- トップ2

または，

実行例 13：itemize環境とenumerate環境の入れ子

```
\begin{itemize}
  \item トップ
    \begin{enumerate}
      \item サブ1
      \item サブ2
        \begin{itemize}
          \item サブサブ1
          \item サブサブ2
        \end{itemize}
\end{enumerate}
  \item トップ2
\end{itemize}
```

- トップ
  1. サブ1
  2. サブ2
     - サブサブ1
     - サブサブ2
- トップ2

のように出力されます.

また，左寄せ，中央寄せ，右寄せの環境として，flushleft，center，flushright環境があります．例えば，

実行例 14：flushleft 環境，center 環境，flushright 環境

```
\begin{flushleft}
  ここは左寄せです．
\end{flushleft}
\begin{center}
  ここは中央寄せです．
\end{center}
\begin{flushright}
  ここは右寄せです．
\end{flushright}
```

ここは左寄せです．

ここは中央寄せです．

ここは右寄せです．

ページの分割も環境の一種として実現できます．例えば，\begin{minipage}{幅}…\end{minipage}で囲まれた部分は，指定した横幅の中に収まるように出力されます．この幅の指定には普通0.4\textwidthのように，ページ幅の割合で指定します．

ここまでの例示はtcolorboxを使ってきました（後述第2.6.4節）．

時々使うのがquote環境やquotation環境，そしてverbatim環境です．quote環境は引用文を出力するための環境で，字下げされます．例えば，

コード 7：quote 環境の例

```
1 \begin{quote}
2 あのイーハトーヴォのすきとおった風、夏でも底に冷たさをもつ青いそら、
3 うつくしい森で飾られたモリーオ市、郊外のぎらぎらひかる草の波。
4 \end{quote}
```

と書くと，

> あのイーハトーヴォのすきとおった風、夏でも底に冷たさをもつ青いそら、うつくしい森で飾られたモリーオ市、郊外のぎらぎらひかる草の波。

のように出力されます．quotation環境は，より長い引用文向けで，段落ごとに字下げされます．

verbatim環境は，そのままの形で出力する環境です．空白や改行，LaTeXで予約されている文字（第1.3節），さらにはLaTeXコマンドを含む文章をそのまま表示したい場合に使います．ここまでの実例で散々使ってきました．例えば，

実行例 15：verbatim 環境

```
\begin{verbatim}
\documentclass{jlreq}
\begin{document}
ここに本文を書く．
This is a sample document.
\end{document}
\end{verbatim}
```

```
\documentclass{jlreq}
\begin{document}
ここに本文を書く．
This is a sample document.
\end{document}
```

これを行内で実現したい場合は，\verbコマンドを使います．例えば，\verb|\LaTeX{}は素晴らしい．|と書くと，\LaTeX{}は素晴らしい．のように出力されます．こいつは区切り文字が特殊で，\verb@aaaa@や\verb+aaaa+のように，出力したい部分に使わない，任意の1文字を区切り文字として使えます[41]．

また，空白を見て分かるように出力したい場合は，verbatim*環境または\verb*を使います．例えば，\verb*|a b c|はa␣b␣cとなります．

図表環境として，figure環境やtable環境などがあります（後述第2.6節）．

41） 二重で\verbを使いたい時に，同じ区切り文字だとバグるからだと思います（実際何も考えずに書くとバグります）．

また，数式環境として，equation環境やalign環境などがあります（後述第1.6節）．

1.4.5 長さ，空白，罫線，脚注

texdoc latex2e (14. Lengths, 19. Spaces).

LaTeXで使える主な長さの単位は以下の通りです． 基本的にはA4用紙の設定になっているので，これらの単位は

表2: LaTeXで使える主な長さの単位

単位	説明	換算・備考
絶対的な長さの単位（LaTeX内部の座標系に基づく絶対長）		
cm	センチメートル	
mm	ミリメートル	
in	インチ	1 in = 2.54 cm
pt	ポイント（TeXポイント）	72.27 pt = 1 in
pc	パイカ	1 pc = 12 pt
bp	ビッグポイント（DTPポイント）	72 bp = 1 in
sp	スケールドポイント	65 536 sp = 1 pt
相対的な長さの単位（フォントの設計やサイズに基づく相対長）		
em	欧文フォントの公称値（元々はMの幅）	
ex	欧文フォントのxの高さ	
zw	和文フォントの全角幅	LuaTeX-jaでは\zw
zh	和文フォントの文字高さ	非推奨
mu	math unit（数式モード用）	18 mu = 1 em

すなわちA4用紙に印刷したときの長さになります．

また，日本語組版の歴史上，級(Q)や歯(H)（いずれも0.25 mm）がありますが，pLaTeX専用なので基本的には使いません．zwやzhが直接使えるのはpLaTeX系のみで，LuaTeX-jaでは\zwや\zhを使います．ただ，zhは古い環境ではzwよりほんのわずかに小さく，LuaTeX-jaではzwと全く同じとなるので使う必要がありません．

空白を入れるコマンドとしては，\␣（バックスラッシュ＋スペース），~（改行禁止空白），\quad（1 emの空白），\qquad（2 emの空白）などがあります．垂直スペースも含め，これらの一覧を表3に示します．

表3: 各種空白コマンド

コマンド	説明・サイズ
\enspace	1/2 em の水平スペース
\quad	1 em の水平スペース
\qquad	2 em の水平スペース
\hspace{length}, \hspace*{length}	指定した長さ (length) の水平スペース
\hfill	無限に伸びる水平スペース (\hspace{\fill} と等価)

Continued on next page

表 3: 各種空白コマンド (Continued)

コマンド	説明・サイズ
\hss	無限に伸び縮みする水平スペース
(バックスラッシュ+空白)	通常の単語間スペース (デフォルト: $3.333\,33^{+1.666\,66}_{-1.111\,11}$ pt)
~, \nobreakspace	改行されないスペース（tie記号）
\thinspace	1/6 em の水平スペース（伸縮なし）
\negthinspace	−1/6 em の水平スペース（伸縮なし）
\/	イタリック補正のための小さなスペース
\hrulefill, \dotfill	罫線やドットで埋められる無限に伸びる水平スペース
\bigskip	約1行分の垂直スペース (デフォルト: 12^{+4}_{-4} pt)
\medskip	約半行分の垂直スペース (デフォルト: 6^{+2}_{-2} pt)
\smallskip	約1/4行分の垂直スペース (デフォルト: 3^{+1}_{-1} pt)
\bigbreak, \medbreak, \smallbreak	それぞれ \bigskip などと同等の垂直スペースに加え，改ページを促す
\strut	幅0で，高さ 0.7\baselineskip，深さ 0.3\baselineskip の支柱
\vspace{length}, \vspace*{length}	指定した長さ (length) の垂直スペース
\vfill	無限に伸びる垂直スペース
\addvspace{length}	指定した垂直スペースを追加（連続する場合は最大値が適用される）

ここで，±で表記した部分は（plain TEXで）グルーなどと呼ばれるもので，伸び縮みするスペースのことです．これは，禁則処理などの都合で詰めや空きを確保する際，ここまでは伸ばしてor詰めていい，という範囲を指定するためのものです．例えば\␣では，3.33333ptのスペースが入りますが，4.99999ptまで伸ばし，2.22222ptまで詰めることができます．TEX言語では，これを3.33333pt plus 1.66666pt minus 1.11111ptのように指定します．

ついでに，普通の処理における空白の扱いについても説明しておきましょう．コラム12で説明するように，TEXは文字をボックスとグルーでつないでいくわけですが，このグルーの伸び縮みの範囲を指定することで，空白の詰めや空きを調整しているわけです．そのため，普通の環境であればいい感じの空白が入るようになっています．段落は自動的に字下げされますし，段落と段落の間には適切な空白が入ります．もし不要なら，\noindentを使うと段落の字下げがなくなりますし，\vspaceを使えば任意の空白を入れることもできます．逆に，不必要な空白を入れてはいけません．特に，和文と欧文の間に空白を入れる癖がある人がいますが，TEX上ではこれはやめましょう[42]．LATEXの日本語対応は，和文と欧文の間に適切な空白を自動的に入れるようになっています．これをカスタムする方法は第2.1.2節で説明します．

コラム 12：フォントメトリック，ボックスとグルー

TFM（TeX Font Metric）ファイルは，TEXがフォントを扱うための情報を格納するファイル形式です．英語圏のフォントでは，文字の幅や高さ，深さなどの情報がTFMファイルに記録されており，TEXはこれを参照して文字を配置します．

日本語に限りませんが，文字数が多くなってくると，文字の幅を個別に指定するのは事実上困難ですし，実際TFMの個数制限があるため不可能です．そのため，日本語フォントにおけるメトリックスJFM（Japanese

42) そもそも，文字列としては空白は不要で，組版として処理すべきところ，空白をタイプすることでカバーしているというのが大きそうです．

Font Metric）ファイルでは，まず和文文字は全て1zwの同じ幅であるとして，そこからさらに文字の種類ごとにグルーを定義して，必要に応じてスペースを詰めたり空けたりすることで見た目を整えるようになっています．
このグルーについて補足しておきます．TEXでは，Knuthのアルゴリズムとして知られる方法で組んでいます．文字を配置する際，それらはボックスと呼ばれる剛体のようなもので扱われ，それをつなぎ合わせていくことで文章が出来上がります．このボックスをグルー（接着剤）でつないでいくわけで，前述したように幅，伸び，縮みを指定して，一定の自由度を持っています．TEXは，この行を分割する際，以下のようなコスト関数dを最小化するように行を分割します．

$$d = b + p$$

ここで，bはbadness，pはpenaltyです．badnessは，グルーの伸び縮み率の3乗に比例する値で，penaltyは行の分割点に対して（不適切な場所で分割しないよう）定義されたペナルティです．つまり，無駄な伸び縮みを避け，かつおかしなところで改行を入れないよう調整されるわけです．
日本語の例に戻ると，句読点や括弧などの約物は，サイズも普通の文字と大きく異なり，「あああ」．のように括弧と句読点が連続するときにスペースを詰めないと見た目が悪くなってしまうので，これらに対しては特別なグルーを定義していることが多いです．
それがJFMファイルです．さらに，LuaTEX上では，この処理がLuaで実装されているため，より見やすくなっています．特に，`jlreq`に付属している`jfm-jlreq.lua`は，始め終わり括弧，句読点，ハイフンや感嘆符！？などの約物にたいして詳細にグルーを定めています．もちろんこれも日本語組版処理の要件を満たすようになっています．
pLATEXではこのJFMのうちmin10といわれる古いデフォルトが非常に質が悪いため，`jsarticle`などのクラスでは自動的にjisメトリックが読み込まれることになっています．

`jlreq`には，バックスラッシュ+全角スペースのマクロが定義されています．なお，LuaTEX上では全角スペースを直接入力しても問題ありません．例えば「　あ　い　う　え　お　」のように文字通り入力すれば，和文幅の空白が入ります．

これらの空白とは異なりますが，紙面のレイアウトにかかわる長さを取得するマクロもあります．

表 4: ページの幅などのマクロ

マクロ	説明
`\textwidth`	ページ本文全体の水平方向の幅．
`\linewidth`	現在の行の幅．ネストされたリスト環境内ではその分減少する．
`\columnwidth`	1つの段（カラム）の幅．1段組の場合は `\textwidth` と同一．
`\textheight`	ページ本文の通常の垂直方向の高さ．
`\paperwidth`	印刷領域の幅とは異なる，用紙自体の幅．
`\paperheight`	印刷領域の高さとは異なる，用紙自体の高さ．
`\hsize`	（参考）テキストを行に分割する際に使用されるTeXプリミティブ．通常のLaTeX文書での使用は非推奨．

基本的には`\linewidth`を使っておけばよく，例えば図表を入れるときはその0.5倍などと指定することになります．
罫線を引くには，`\hrulefill`,

や\dotfill，

.....

が使えます．

脚注を入れるには，\footnote{脚注の内容}を使います．例えば，この文章には脚注[43]があります．

1.5 自作コマンド・環境

新しいコマンドや環境を定義したくなるモチベを説明します．今まで使ってきた[44]texdocのように，例えばコマンドラインに入力するものはタイプライター体で出力しよう，というときや，高度な数学で，例えば圏[45]は$\mathcal{C}$のように字体を変えたい，というときです．もちろん，一々\texttt{texdoc}や\mathcal{C}と書いてもよいのですが，これには問題があります．後で，例えばコマンドを意味するtexdocをサンセリフ体texdocで書きたくなったとき，ソースコード中の全ての\texttt{texdoc}を\textsf{texdoc}に書き換えなければならなくなります．圏の例で言えば$\mathfrak{C}$に変えたくなったときも同様です[46]．これはもちろん一括置換で済むのですが，それをやると意図しない部分まで，例えばクラス名はタイプライター体のままでよいのにそれまで，ないしは数学の例なら別の意味で使っていた花文字まで変わってしまう，ということが起こり得ます．そこで，自作コマンドを定義しておけば，後からコードを読み返したとき意味も分かりやすく，かつ一括置換の必要もなくなります．つまり，WYSIWYM（What You See Is What You Mean）の原則を，意味を表すコマンドを定義して使うということで一貫させることになります．

さて，自作コマンドの定義については，\def(TEX)や\newcommand(LATEX)などを用いる方法が有名でしょう．後者については texdoc latex2e (12 Definitions) を参照．

今は，コラム9で説明した，expl3のxparseという高機能（で安全な）ものがLATEX 2_εに統合されているため，これを使うのがよいでしょう．

これについては， texdoc usrguide，LATEX for authorsに詳しいドキュメントがあります．

ここでは簡単な例を示すことにしましょう．例えば，今までやってきたように，クラス名としてのjlreqなどをタイプライター体で出力しよう，と決めたとします．

```
\NewDocumentCommand{\thisisclassname}{m}{\texttt{#1}}
```

とプリアンブルに定義しておけば，\thisisclassname{jlreq}と書くだけでjlreqと出力されます．\NewDocumentCommandは，新しい，本文中で使うようなコマンドを定義します．1つ目の引数はコマンド名，2つ目の引数は引数の仕様，3つ目の引数は定義です．引数の仕様は，例えばmと書けば必須引数，oと書けばオプション引数（[]で指定），vと書けば\verbと同じ仕様，のようになります．nつ目の引数は，定義では#nで参照できます．

コラム 13：\DocumentCommandシリーズについて

\NewDocumentCommandの他には，
\RenewDocumentCommand，\ProvideDocumentCommand，\DeclareDocumentCommandがあります．
これらの違いをまとめると，以下の通りです（ texdoc usrguide の内容を要約）．

- \New…は，既に同名のコマンドが定義されている場合はエラーになる
- \Renew…は，既に同名のコマンドが定義されていない場合はエラーになる
- \Provide…は，同名のコマンドがなければ定義し，同名のコマンドがあれば何もしない
- \Declare…は，強制的に上書き定義する（ので軽々しく使うべきではない）

環境も同様に定義できます．例えば，\newenvironmentや\NewDocumentEnvironmentを使います．コマンドの時の

43） これは脚注の例です．
44） コードをご覧ください．
45） 圏論の話ですが，知らなかったらそういう概念がある，と思ってください．
46） このあたりの話はalg-d氏の動画も参考になります．

対応物です．（あとで加筆するかもしれない）第 2.6.4 節でも tcolorbox における対応するコマンドについて説明します．

1.6 基本的な数式

ここでは，LaTeX で数式を入力する方法を簡単に説明します．数式を書く際には，数式モードに入る必要があります．行内に数式を書く場合は$...$や\(...\)（こちらを推奨）を使います．例えば，\(E=mc^2\)と書くと，$E=mc^2$のように出力されます．独立した数式を表示する場合は，\[...\]を使います．なお，MathJax などでは$$...$$を使うのが普通ですが[47]，LaTeX では使えません．これはそもそも TeX の記法であって，LaTeX はこれを処理せず，TeX にそのまま渡されるため，LaTeX の他の機能との相互作用ができなくなります．例えば，\[E=mc^2\]と書くと，

$$E=mc^2$$

のように出力されます．

式番号を付けたい場合は，equation 環境を使います．例えば，

実行例 16：equation 環境

```
\begin{equation}
  E=mc^2
\end{equation}
```

$$E=mc^2 \tag{8}$$

のようになります．

これらの式番号を引用することもできます．例えば，\label{abcd}と書いて式にラベルを付けておいて，\ref{abcd}と書けば，式番号を参照できます．

実行例 17：equation 環境の相互参照

```
\begin{equation}\label{example_eq:emc2}
  E=mc^2
\end{equation}
式\ref{example_eq:emc2}は有名な式です．
```

$$E=mc^2 \tag{9}$$

式9は有名な式です．

なお，このあたりの相互参照については，第2.3節と2.7.4節で詳しく説明します．

1.6.1 数式フォントについて

LaTeX らしさの中核といっていい独自のフォントは，これも Knuth がデザインした Computer Modern です．これの見やすさ，美しさはもはや説明不要でしょう．ただ，古いフォントであり，最近の技術の進歩に追いついていない部分もあるため，最近では Latin Modern や New Computer Modern などの改良版が使われます．

古い TeX では Computer Mordern, LuaTeX などでは Latin Modern が本文の（欧文）フォントのデフォルトで，数式は Computer Modern がデフォルトです．なお，jlreq を使うと自動的に本文・数式ともに Latin Modern になります．New Computer Modern を推奨しますが，これについては第2.2節と2.3.3節で解説します．

図1は，Latin Modern，New Computer Modern，STIX Two Math，Cambria Math の比較です．Latin Modern と New Computer Modern はほぼ同一ですが，細かいところで異なります．なお，New Computer Modern の利点としては，とにかく文字の種類が多いことが挙げられます．この文字の種類については texdoc unimath-symbols から確認できますが，Latin Modern には 1585 文字，New Computer Modern には 2430 文字あります．STIX Two Math も 2430 文字近くとなっており，Latin Modern はかなり制限された文字セットであることが分かります．また，細かいカーニングなどが調整されており，LuaLaTeX との相性もよいそうです texdoc newcm．

47） むしろ\[,\]は使えない．

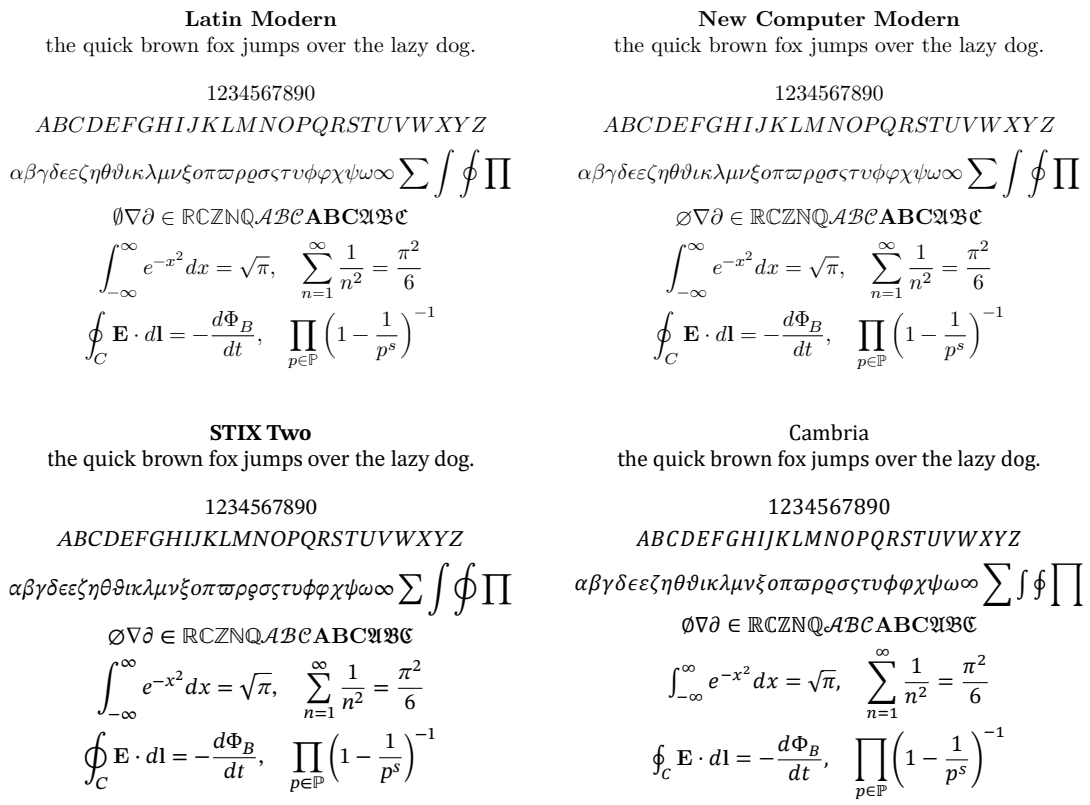


図1: Latin Modern, New Computer Modern, STIX Two Math, Cambria Mathの比較.

STIXは昔から使われているTimes New Romanと調和するようにデザインされています．Cambria Mathも同様で，Wordなどで使われる~~最悪の~~数式用フォントです．

1.6.2 数式の記法

まず，数式モード内ではスペースが無視されることに注意してください．演算子の周りのスペースは自動的に調整されます．例えば，`\(a+b\)`と書くと，$a+b$のように出力されます．`\(a + b\)`と書いても同じように$a+b$のように出力されます．また，コンマに関しても自動でスペースが入ります．例えば，`\(f(x,y)\)`と書くと，$f(x,y)$のように出力されます．これは小数点としてコンマを入れたい場合[48]にも発動するのでやや困ります．`1{,}234`のように入力すれば回避できます．

空白には`\quad`,`qquad`，`\␣`が使えます．

加えて，以前のLaTeXでは数式モード専用であった`\,`なども，LaTeXの2020年10月以降，または`amsmath`をロードしていれば数式モード外でも使えるようになりました．

単位については表2を参照してください．

さて，実際にいくつかのコマンド[49]をみてみましょう．

48） 例えばドイツ語圏での表記法

49） ただし，`amsmath`の使用を仮定します．というのも，数学をやるときには必ず使うパッケージであり，当然のように使われる環境やコマンドが多すぎて，使わない場合の例を示すのが困難なため．`amsmath`の詳しい話は第2.3節．

表5: 数式モードの空白

コマンド	サイズ
\; または \thickspace	$5.0^{+5.0}_{-0.0}$ mu
\negthickspace	$-5.0^{+2.0}_{-4.0}$ mu
\: または \> または \medspace	$4.0^{+2.0}_{-4.0}$ mu
\negmedspace	$-4.0^{+2.0}_{-4.0}$ mu
\, または \thinspace	3 mu （1/6 em）
\! または \negthinspace	−3 mu
\quad	18 mu （1 em）
\qquad	36 mu （2 em）

実行例 18：数式の記法1

```
\begin{align*}
  &\text{四則演算} && a+b, a-b, ab, \frac{a}{b} \\
  &\text{平方根} && \sqrt{x} \\
  &\text{n乗根} && \sqrt[n]{x} \\
  &\text{べき乗} && x^n \\
  &\text{下付き文字} && x_n \\
  &\text{順序} && a \leq b \leq c, a < b < c \\
  &\text{不等号} && a \neq b \\
\end{align*}
```

$$
\begin{aligned}
&\text{四則演算} && a+b, a-b, ab, \frac{a}{b} \\
&\text{平方根} && \sqrt{x} \\
&\text{n乗根} && \sqrt[n]{x} \\
&\text{べき乗} && x^n \\
&\text{下付き文字} && x_n \\
&\text{順序} && a \leq b \leq c, a < b < c \\
&\text{不等号} && a \neq b \\
\end{aligned}
$$

演算子も用意されています.

実行例 19：数式の記法2

```
\begin{align*}
  &\text{和} && \sum_{i=1}^n a_i \\
  &\text{積} && \prod_{i=1}^n a_i \\
  &\text{極限} && \lim_{x \to 0} f(x) \\
  &\text{微分} && \frac{dy}{dx}, f'(x), \nabla f \\
  &\text{積分} && \int_a^b f(x) dx, \iint_D f(x,y) dx dy, \\
   & && \iiint_V f(x,y,z) dx dy dz \\
  &\text{行列} && \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \begin{Bmatrix} a & b \\ c & d \end{Bmatrix}, \begin{vmatrix} a & b \\ c & d \end{vmatrix}, \begin{Vmatrix} a & b \\ c & d \end{Vmatrix}
\end{align*}
```

$$
\begin{aligned}
&\text{和} && \sum_{i=1}^n a_i \\
&\text{積} && \prod_{i=1}^n a_i \\
&\text{極限} && \lim_{x \to 0} f(x) \\
&\text{微分} && \frac{dy}{dx}, f'(x), \nabla f \\
&\text{積分} && \int_a^b f(x) dx, \iint_D f(x,y) dx dy, \\
& && \iiint_V f(x,y,z) dx dy dz \\
&\text{行列} && \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \begin{Bmatrix} a & b \\ c & d \end{Bmatrix}, \begin{vmatrix} a & b \\ c & d \end{vmatrix}, \begin{Vmatrix} a & b \\ c & d \end{Vmatrix}
\end{aligned}
$$

俗にlog型と言われる関数（立体で書き，前に引数を書くような数学関数）も用意されています．

実行例 20：数式の記法3

```
\begin{align*}
  &\text{三角関数} && \sin x, \cos x, \tan x \\
  &\text{逆三角関数} && \arcsin x, \arccos x,
   \arctan x \\
  &\text{双曲線関数} && \sinh x, \cosh x,
  \tanh x \\
  &\text{対数関数} && \log x, \ln x \\
  &\text{その他の関数} && \exp x, \gcd(a,b),
  \deg f
\end{align*}
```

三角関数　$\sin x, \cos x, \tan x$

逆三角関数　$\arcsin x, \arccos x, \arctan x$

双曲線関数　$\sinh x, \cosh x, \tanh x$

対数関数　$\log x, \ln x$

その他の関数　$\exp x, \gcd(a,b), \deg f$

これらを$sinx$などと書かないように注意が必要です[50]．関係を表すような記号は以下の通りです．

実行例 21：数式の記法4

```
\begin{align*}
  &\text{論理記号} && \land, \lor, \lnot,
  \implies, \iff \\
  &\text{集合記号} && \in, \notin, \subset,
   \subseteq,
   \supset, \supseteq, \cup, \cap, \emptyset \\
  &\text{その他の記号} && \forall, \exists,
  \nexists, \therefore, \because
\end{align*}
```

論理記号　$\land, \lor, \lnot, \implies, \iff$

集合記号　$\in, \notin, \subset, \subseteq, \supset, \supseteq, \cup, \cap, \emptyset$

その他の記号　$\forall, \exists, \nexists, \therefore, \because$

なお，他にも，

実行例 22：数式の記法5

```
\begin{align*}
  &\text{ベクトル} && \vec{v},
  \overrightarrow{AB} \\
  &\text{微分演算子} && \partial, \nabla \\
  &\text{その他の記号} && \infty, \aleph_0,
   \hbar, \Re z, \Im z
\end{align*}
```

ベクトル　$\vec{v}, \overrightarrow{AB}$

微分演算子　∂, ∇

その他の記号　$\infty, \aleph_0, \hbar, \Re z, \Im z$

など，基本的に全ての数学記号が用意されていると考えてよいでしょう．また，ここまでで使ってきたalign環境は，amsmathパッケージで提供されるもので，数式を複数行にわたって書くときに便利です．&を揃える場所，\\を改行する場所に入れることで，数式をきれいに揃えることができます．

1.7 タイプセット–デバッグ

さて，実際に簡単なタイプセットをしてみましょう．まず，例えばut_test.texというファイルを作成して，以下のような内容を書いてみてください．

50） 本当にみっともないです．

コード 8：単純なLaTeX文書の例

```
\documentclass{jlreq}
\begin{document}
ここに本文を書く．
This is a sample document.
\end{document}
```

これを保存し，コマンドラインを開いて`lualatex ut_test`と入力し，エンターを押すと

コード 9：LuaLaTeXの出力例

```
  This is LuaHBTeX, Version 1.24.0 (TeX Live 2026)
 restricted system commands enabled.
(./ut_test.tex
LaTeX2e <2025-11-01>
L3 programming layer <2026-01-19>
 (c:/texlive/2026/texmf-dist/tex/latex/jlreq/jlreq.cls
Document Class: jlreq 2025/3/16 jlreq
(c:/texlive/2026/texmf-dist/tex/latex/jlreq/jlreq-helpers.sty)
jlreq guessed engine: lualatex
(c:/texlive/2026/texmf-dist/tex/luatex/luatexja/luatexja.sty
 （中略）
(c:/texlive/2026/texmf-dist/tex/luatex/luatexja/patches/lltjcore.sty
(c:/texlive/2026/texmf-dist/tex/latex/l3kernel/expl3.sty
(c:/texlive/2026/texmf-dist/tex/latex/l3backend/l3backend-luatex.def)))
 （中略）
No file ut_test.aux.
[1{c:/texlive/2026/texmf-var/fonts/map/pdftex/updmap/pdftex.map}]
(./ut_test.aux))
 879 words of node memory still in use:
   8 hlist, 1 vlist, 5 rule, 2 glue, 3 kern, 2 glyph, 164 attribute, 45 glue_sp
ec, 24 attribute_list, 1 write, 10 user_defined nodes
   avail lists: 1:4,2:1208,3:14,4:26,5:32,6:2,7:132,8:1,9:44,11:6
<c:/texlive/2026/texmf-dist/fonts/opentype/public/lm/lmroman9-regular.otf><c:/t
exlive/2026/texmf-dist/fonts/opentype/public/lm/lmroman10-regular.otf><c:/texli
ve/2026/texmf-dist/fonts/opentype/public/haranoaji/HaranoAjiMincho-Regular.otf>

Output written on ut_test.pdf (1 page, 8719 bytes).
Transcript written on ut_test.log.
```

のように出力され，`ut_test.pdf`というPDFファイルが生成されます．このPDFファイルを開いてみてください．「ここに本文を書く．This is a sample document.」と表示されているはずです．

また，ここで`No file ut_test.aux.`と表示されているのは，文字通り，`ut_test.aux`というファイルが存在しないということです．これは補助ファイルが存在しないことを警告していますが，初回のタイプセットでは存在しないのが普通なので，気にしなくて大丈夫です．

このログは，`ut_test.log`というファイルに保存されます．このログには，タイプセットの過程で起こったことが記録されています．

コラム 14：LaTeXで生成されるファイルについて

LaTeXでタイプセットすると，PDFファイルの他に，.auxや.logなどの補助ファイルが生成されます．

.aux 補助ファイル．相互参照や目次などの情報が保存される．
.log ログファイル．タイプセットの過程で起こったことが記録される．
.toc 目次ファイル．目次の情報が保存される．
.bbl 参考文献ファイル．.bibから生成された参考文献の情報が保存される．
.out 出力ファイル．特定のパッケージが生成するファイル．
.fls 入出力ファイルのリスト．
.synctex.gz SyncTeXファイル．PDFとソースコードの対応関係が保存される．

デバッグの簡単な方法を説明します．ログには，エラーが起きると!マークが付いてエラーの内容が表示されます．今，ut_test.texの内容を次のように変更してみましょう．

コード 10：エラーを起こすLaTeX文書の例

```
\documentclass{jlreq}
\begin{document}
\undefinedcommand

a_b_c
\end{document}
```

これを同様にタイプセットすると，次のような表示で止まるはずです．

コード 11：エラーが起きたときのLuaLaTeXの出力1

```
   ! Undefined control sequence.
l.3 \undefinedcommand
?
```

ここで，?を入力してエンターを押すと，次のような表示になります．

コード 12：エラーが起きたときのLuaLaTeXの出力2

```
   Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
```

書いてある通り，ここでエラーの詳細を見たり，無視して続行したり，編集したりできます．ただ，latexmkやVS Codeの拡張機能でタイプセットしている場合は，--interaction=nonstopmodeで動き，エラーがあっても止まらずに最後までタイプセットされることが多いでしょう．この場合は，ログファイルを見てエラーの内容を確認する必要があります．

実際に，全て無視して続行し，ログファイルを見てみると，次のような表示があるはずです．

コード 13：エラーが起きたときのログファイルの内容

```
! Undefined control sequence.
l.3 \undefinedcommand

?
! Missing $ inserted.
<inserted text>
                $
l.5 a_
      b_c
?
（中略）
! Double subscript.
l.5 a_b_
        c
?
! Missing $ inserted.
<inserted text>
                $
l.6 \end{document}

?
```

これはよくあるエラーをわざと起こした例ですので，1つずつ見ていきましょう．

まず，Undefined control sequenceは，定義されていないコマンドを使おうとしているぞ，というエラーです．今回の場合は`\undefinedcommand`は定義されていません．実際の文書で生じるケースとしては，誤字脱字，第1.3節で説明した，コマンドの後に続けて文字を入れてしまった場合，必要なパッケージを読み込んでいない場合などが考えられます．

次と最後のMissing $ insertedは，数式モード内でしか使えないコマンドを使ってるから，空気を読んで数式モードにしてやったぞ，というエラーです．今回の場合は`a_b_c`のような下付きを表す_は数式モードでしか使えないものなので，TEXエンジンが$を前後に挿入してくれました． 挿入してくれるならいいじゃないか，と思うかもしれませんが，大抵は，そもそも数式モードにしたくない場所で起こしてしまうので，結局全てが崩壊することになります．

その次のDouble subscriptは，下付き文字の中にさらに下付き文字があるぞ，というエラーです．これは`a_b_c`のように_を2回使っているために起こります． a_{b_c} なのか，$a_{b}{}_{c}$ なのかが判定できないために起こるエラーです．この場合は`a_{b_c}`または`{a_b}_c`のように括弧でくくってあげる必要があります．前者はaの下付き文字がb_cであることを表し，後者はa_bという文字の下にさらにcが付いていることを表します．これは$a_{(b_c)}$，$(a_b)_c$と書くと見た目の違いが分かりやすいでしょう．

さて，エラーが起きたときの対処法ですが，ログファイルは長大なので，頑張って大本のエラーを探す必要があります．デバッグのために，一旦`lualatex ut_test`のようにタイプセットすれば，エラーに当たった時点で止まるので，そこからエラーの内容を確認していくのは1つの方法としてあります．VS Codeの拡張機能を（適切に）使っていれば，TEXファイルの該当する行にエラーが表示されます．ただ，上で見たように，TEXエンジンが補ったことによって他の行でエラーが起きることも多々あるので，最終的にはログファイルを読む羽目になります．シンタックスハイライトができるエディタであれば，ログファイルの! Undefinedが赤字で表示されたりはするので，それを頼りに探すとよいでしょう．

ただ，エラーの文面は不親切なことも多く，修正すべき箇所を発見してもどう直せばいいのか分からないこともあるでしょう．その場合は，まずTeX Wiki TeXのエラーメッセージなどを見てみると，頻出のエラーについては解説があります（ただ，pTEXを前提にしていることもあるので注意してください）．

それでも分からない場合は，エラーの文面をそのまま検索するだけでも解決策が見つかることもあります．

最終手段として，チャットAIにTEXファイルとログを丸投げして指摘してもらうのも有効です．エラーの部分だけを切り取って投げると正確な回答が得られやすいと思います[51]．

そもそも，エラーを起こさないようにするのも重要です．まともなエディタを使えば，コマンドは自動補完されますし，例えば括弧類が開きっぱなしになっていると赤色にしてくれたり，数式モード内は色を変えたりするので，タイプセット前に気づきやすくなります．ChkTeXなどのツールが統合されており，例えば`begin`と`end`の対応が取れていないと警告してくれたりすることもあります．

まあ，このあたりについては慣れです．ただ，多数のパッケージや自分が理解していないコマンドを使うとそれだけデバッグの苦労は増えるので，AIの出力やStack Exchangeやらのコードを何も考えずにコピペするのはやめましょう[52]．

2 パッケージ

まず，パッケージとは何か，基本的な使い方について説明します．

そもそも，LATEXには外部図表，色，その他の複雑な飾りの機能はありません．これは組版ということの意味を考えればわかることで，鉛の活字を組むことまでしかやってくれず，最後にDVIドライバ=印刷業者？が印刷してくれる，というイメージです[53]．なので，それらの機能はパッケージで実現することになります．その他にも，独自の機能を追加したり，柔軟なカスタマイズをやるためのパッケージもあります．

もちろん，独自の機能を追加するだけなら第1.5節で説明した自作コマンドを定義して使うこともできます．ただ，複雑なコマンドを自分で実装するのではなく，よくメンテナンスされたパッケージを使う方が便利でしょう．もっと特殊な例で言えば，会社や団体などで独自のスタイルやマクロを定めているときは，自分でパッケージ化することも可能です[54]．

なお，マクロパッケージとの違いは，マクロパッケージはTEXのマクロ，すなわちTEXが読み込むフォーマット[55]（LATEX, ConTEXt），LATEXのパッケージはLATEXの機能を拡張するものということになります．

基本的には，プリアンブルに

コード 14：パッケージの読み込みの例

```
\usepackage[オプション]{パッケージ名}
```

と書いて使います．オプションは必要に応じて指定します．

2.1 LuaTeX-ja, jlreq

LuaLATEX上での日本語組版に必須となっている`LuaTeX-ja`と，LuaLATEXでの日本語組版のためのクラスである`jlreq`について説明します．

2.1.1 LuaTeX-ja

`LuaTeX-ja`は，LuaLATEXで日本語を組むためのパッケージです． texdoc luatexja-ja なお，依存関係として，`luaotfload`，`fontspec`などを必要とします．

特に意識する必要はないでしょう．裏方で日本語組版を支えてくれるパッケージです．当然，LuaTEX専用のパッケージなので，LuaTEX以外のエンジンでは使えません．日本語のフォント選択のための`luatexja-fontspec`も同梱

51） 逆に，長すぎるファイルを渡すとデタラメを返してきますが……
52） 自戒も込めて……
53） ただし，第0.2節で説明したように，今のTEXはPDFの直接出力をするものがあるので，このあたりの事情は変化しています．内部的には，エンジン固有の命令を発行してPDFのレベルに図表や色を入れることになりますから，本質的には同じことです．
54） これは深く扱いません．
55） のようなもの．

されています．

ヒラギノフォントを使うのであれば，luatexja-preset により一括で設定できます texdoc luatexja-preset．例えば，

コード 15：luatexja-preset の使用例

```
\usepackage[hiragino-pron]{luatexja-preset}
```

と書けば，本稿のようにヒラギノフォントが使えるようになりますし[56]，

コード 16：luatexja-preset の使用例 2

```
\usepackage[hiragino-pron,deluxe]{luatexja-preset}
```

とすれば，ヒラギノの明朝，ゴシック，丸ゴシックが使えるようになります（多ウェイト）．例えば丸ゴシック．

さらに，luatexja-ruby や luatexja-otf などもあります．まず，luatexja-ruby は LuaTEX の機能をフル活用したルビや圏点をサポートします texdoc luatexja-ruby．例えば

実行例 23：luatexja-ruby の使用例

```
\ltjruby{漢字}{かんじ}の例．\\
\ltjruby{東京|特許|許可|局}
{とうきょう|とっきょ|きょか|きょく}の例．\\
% このようにすると，区切りで改行できるようになります．
強調したいところに\ltjkenten{圏点}を付けます．
```

漢字（かんじ）の例．
東京（とうきょう）特許（とっきょ）許可（きょか）局（きょく）の例．
強調したいところに圏点（˙）を付けます．

となります．もっと複雑なルビ，行を跨ぐなどの処理も楽々可能です（ドキュメント参照）．なお，pLATEX では pxrubrica や outoruby があり，もちろん LuaTEX でも使えます．

luatexja-otf は，pLATEX の otf パッケージの機能を移植したもので，LuaTEX で，pTEX などの時代に使われていた JIS 外の文字を出力するマクロを提供します．

実行例 24：luatexja-otf の使用例

```
森\UTF{9DD7}外と\CID{13966}田百\UTF{9592}とが
\UTF{9AD9}島屋に\\
\CID{7652}飾区の\CID{13706}野家，
\CID{1481}城市，葛西駅，\\
高崎と\CID{8705}\UTF{FA11}，濱と\ajMayuHama\\
\aj半角{カタカナ}\ajKakko3\ajMaruYobi{2}%
\ajLig{〇問}\ajJIS\\
森鴎外 森鷗外 舗\\
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨\\
\ajMaru{1} \ajMaru{2} \ajMaru{3}
\ajMaru{4} \ajMaru{5} \ajMaru{6}
\ajMaru{7} \ajMaru{8} \ajMaru{9}
```

森鷗外と内田百閒とが髙島屋に
葛飾区の吉野家，葛城市，葛西駅，
高崎と髙﨑，濱と濵
ｶﾀｶﾅ(3)㊊㊐㋾
森鴎外 森鷗外 舗
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨

となります．なお，上で示したように（luatexja-fontspec を読み込んでいれば），これらはコードに Unicode で直打ちしても出力されます．

56） フォントか mac を買っていればの話．学生なら年間1000円くらいで使えるので契約しましょう．

2.1.2 jlreq

jlreqは，単純に使うだけならクラスを指定するだけで十分です．このクラスは，LuaLaTeXやpLaTeXで日本語文書を日本語組版処理の要件に従って組むためのクラスです．カスタマイズ性が高いので，全てのオプションは説明しません．詳しくは texdoc jlreq-jaを参照．

例えば，

コード 17：jlreqの使用例

```
\documentclass[twoside,openany]{jlreq} % 両面印刷，章の開始ページを右ページにしない
\documentclass[report, a4paper, 9.5pt]{jlreq} % reportクラス，A4用紙，9.5pt
\documentclass[tate]{jlreq} % 縦組み
```

のようにオプションを指定できます．

また，jlreq-complementsパッケージを内部で読み込み，thebibliographyやindex環境，amsthmの定理環境などを\jlreqsetupでカスタムできるようになっています．

第 1.4.5 節で説明した，和文と欧文とのスペース，和文のグルーの設定は以下の通り行います．これらはpLaTeXやLuaTeX-jaの設定を行うためのフロントエンドになっています．

コード 18：jlreqのスペース設定の例

```
% 和文と欧文の間のスペース
\renewcommand{\jlreqxkanjiskip}{0.1\zw plus .1\zw minus .01\zw} % 既定は0.25zw
% 和文文字の間のスペース（の最大）
\renewcommand{\jlreqkanjiskip}{0pt plus .1\zw minus .01\zw} % 既定は0.25zwまで広がる
```

2.2 fontspec

fontspecは，LuaLaTeXやXeLaTeXでフォントを選択するためのパッケージです．第 0.2 節でも説明した通り．LuaTeXやXeTeXは，OpenTypeフォント[57]を直接読み込むことができます．

ここでは，LuaTeXでの使用を前提に説明します．

まず，LuaTeXは，luaotfloadを使ってフォントを管理しています[58]．fontspecは，luaotfloadを自動で読み込むので，TeXのコードで意識する必要はありません．ただ，OSに登録されているフォントをこのluaotfloadが管理しているので，fontspecを使うときは，luaotfloadの管理するフォントを選択することになります．こいつのコマンドラインでの使い方をまず説明します．

コード 19：luaotfloadのコマンドラインでの使い方

```
luaotfload-tool --update # フォントデータベースの更新
luaotfload-tool --find "Font Name" # フォントの検索
luafindfont "Font Name" # でもOK
```

このフォントデータベースの更新が結構時間がかかります．一度認識されれば，あとはfontspecのコマンドで簡単に呼び出せるようになります．

57） Truetypeはどうなんだ，という声が聞こえてきそうですが，OTFはTTFの上位互換なので，当然読み込めます．
58） なお，XeTeXはfontconfigというコマンドを使うそうです．

コード 20：fontspecの使用例

```
\usepackage{fontspec}
\setmainfont{Times New Roman} % 本文のフォント
\setsansfont{Arial} % ゴシック体のフォント
\setmonofont{Courier New} % 等幅フォント
\setmainfont{texgyrepagella-regular.otf} % フォントファイル名による指定
\setmathrm{Optima} % 数式のフォント
\setmathsf{Optima} % 数式のサンセリフフォント
\setmathtt{Optima} % 数式の等幅フォント
\setboldmathrm[BoldFont={Optima ExtraBlack}]{Optima Bold} % 太字の数式フォント
\newfontfamily\myfont{Comic Sans MS} % 新しいフォントファミリーの定義
```

数式版のコマンドも説明しましたが，これは後述（第2.3.3節）するunicode-math及びfontsetupを使えば一発なので，あまり使う機会はないでしょう．和文はluatexja-fontspecで設定できます．基本的にはjを加えると和文版のコマンドになります． texdoc luatexja-jaにあるリファレンスを表6に引用します．

表6: luatexja-fontspecで定義される命令

和文	\jfontspec	\setmainjfont	\setsansjfont	\setmonojfont
欧文	\fontspec	\setmainfont	\setsansfont	\setmonofont
和文	\newjfontfamily	\renewjfontfamily	\setjfontfamily	\providejfontfamily
欧文	\newfontfamily	\renewfontfamily	\setfontfamily	\providefontfamily
和文	\newjfontface	\renewjfontface	\setjfontface	\providejfontface
欧文	\newfontface	\renewfontface	\setfontface	\providefontface
和文	\defaultjfontfeatures	\addjfontfeatures		
欧文	\defaultfontfeatures	\addfontfeatures		

これの詳細については，適宜 texdoc fontspecを参照してください．

2.3 amsmathなどの数式パッケージ

さて，数式のタイプセットに必要なパッケージについて説明します．amsmathは定番ですが，今ではmathtools,thmtoolsを使用することも多いため，そちらを中心に説明することとします．

2.3.1 amsmath

amsmathは，アメリカ数学会（$\mathcal{AMS}$）が提供する数式のためのパッケージです． texdoc amsmath．なお，これに付随するパッケージとして，amsfonts，amssymb，amsthmなどがありますが，これらは第2.3.2節で説明します．

大量の機能が提供されるため，ここでは全てを説明しません．詳しくは texdoc amsmathを参照してください．

2.3.1.1 環境 多くの数式向け環境が提供されます．基本的なequation以外にも，align，gather，multlineなどがあります．

実行例 25：amsmathの環境の例

```
\begin{align}
  E &= mc^2 \\
  f(x) &= \int_{-\infty}^\infty
  g(t) e^{-2\pi i t x} dt
\end{align}
\begin{gather}
  a^2 + b^2 = c^2 \\
  e^{i\pi} + 1 = 0
\end{gather}
\begin{multline}
  a + b + c + d + e + f + g \\
  = h + i + j + k + l + m + n
\end{multline}
```

$$E = mc^2 \tag{10}$$

$$f(x) = \int_{-\infty}^{\infty} g(t) e^{-2\pi i t x} dt \tag{11}$$

$$a^2 + b^2 = c^2 \tag{12}$$

$$e^{i\pi} + 1 = 0 \tag{13}$$

$$a + b + c + d + e + f + g = h + i + j + k + l + m + n \tag{14}$$

それぞれの環境は，数式の配置や改行の方法が異なります．`align`は複数行の数式を揃えるための環境で，&を使って揃える場所を指定します．`gather`は複数行の数式を中央揃えにするための環境で，&は必要ありません．`multline`は複数行の数式を上手く表示するための環境で，この例ならば，最初の行は左揃え，最後の行は右揃えになります．

また，ここでは行番号を部分的に消すために`\notag`を使用できます．また，ここまでの例で見てきたように，`align*`のように環境名の後ろに*を付けると，行番号が全て消えます．

実行例 26：\notagの使用例

```
\begin{align}
  E &= mc^2 \notag \\
  f(x) &= \int_{-\infty}^\infty
  g(t) e^{-2\pi i t x} dt \notag
\end{align}
```

$$E = mc^2$$

$$f(x) = \int_{-\infty}^{\infty} g(t) e^{-2\pi i t x} dt$$

2.3.1.2　演算子など　よく使う演算子，関数などの一覧を，一部例18から22と被りますが，表8で可能な限り沢山紹介します．目を通して，このような文字は変なことをしなくても書けるんだな，ということを感じ取ってもらえればと思います．中には`amsmath`によるものではない，元々LaTeXに入っている記号も含みます．微分に関しては，他のパッケージを使用した方が楽ですので，第2.4.1.1段落で説明します．

また，サイズの異なる括弧類も提供されます．`\left`と`\right`を使うことで，括弧類のサイズを自動で調整することができます．これについては第2.3.3.2段落や第2.4.1.2段落でも別の方法を紹介します．

実行例 27：leftとrightの使用例

```
\[
  \left( \frac{a}{b} \right) +
  \left[ \sum_{i=1}^n x_i \right]
  + \left\{ \int_0^1 f(x) dx \right\}
\]

\begin{align*}
  a &= \left( \sum_{i=1}^n x_i \right. \\
  &\left.+ \int_0^1 f(x) dx \right)
\end{align*}
```

$$\left(\frac{a}{b}\right) + \left[\sum_{i=1}^{n} x_i\right] + \left\{\int_0^1 f(x)dx\right\}$$

$$a = \left(\sum_{i=1}^{n} x_i + \int_0^1 f(x)dx\right)$$

下の例は，`align`環境の中で，左や右だけの括弧を使う例です．このように，`\left`と`\right`はペアで使う必要が

ありますが，片方だけを指定することもできます．この場合は，対応する括弧の代わりに . を指定します．

なお，記号の一覧は texdoc symbols-a4 (Comprehensive LaTeX Symbol List) が有用で，数学記号以外の記号も多数掲載されているので，必要に応じて参照してください．

コラム 15：$\mathcal{AMS}$-TeX と $\mathcal{AMS}$-LaTeX

$\mathcal{AMS}$-TeX は，TeX のマクロパッケージとして，LaTeX と同時期に開発されていました．TeX を数学界で使うというムーブメントの強さは，Kunth が数学者であったことからも当然ですが，相当なものだったのでしょう．LaTeX の進歩によって，これを LaTeX 上で使いたい，という要望もあったのでしょう．そこで，$\mathcal{AMS}$-LaTeX が開発されました．なお，$\mathcal{AMS}$-TeX もアクティブですが，LaTeX の機能を使いたいので，$\mathcal{AMS}$-LaTeX の方が人気があると思われます．

2.3.2 amsfonts, amssymb, amsthm, amscd

texdoc amsfonts, texdoc amssymb, texdoc amsthm, texdoc amscd

amsfonts, amssymb はその名の通り，フォントや記号を提供するパッケージです．amscd は，圏論などで出てくる可換図式を描くためのパッケージです．amsthm は定理環境を提供するパッケージです．

実行例 28：amsthm の使用例

```
% \usepackage{amsthm}
% 定理環境の定義
% \newtheorem{thm}{定理}[section]
% 証明環境の定義
% \newtheorem{proof1}{証明}[section]
% などとプリアンブルで定義しておくと，
\begin{thm}\label{thm:example}
  これは定理の例です．
\end{thm}
\begin{proof1}
  これは証明の例です．
  定理\ref{thm:example}を証明しました．
\end{proof1}
```

定理 2.1. これは定理の例です．

証明. これは証明の例です．定理 2.1 を証明しました． □

のように，定理や証明などの環境を定義して使うことができます．これらの細かい使い方は，thmtools でさらに便利にできるので，次で説明します．

2.3.3 unicode-math, fontsetup, mathtools, thmtools

2.3.3.1 数式フォント unicode-math は，LuaLaTeX や XeLaTeX で数式の取り扱いを Unicode ベースにすることで，本文フォントを fontspec で変えられたのと同様に，数式フォントも簡単に変更できるようになるパッケージです texdoc unicode-math.

こいつのフォント選択を，本文フォントと同時に一括して行ってくれるのが fontsetup です texdoc fontsetup.

とりあえず，プリアンブルに

コード 21：unicode-math の使用例

```
1 \usepackage{fontsetup}
```

と書いておけば，fontsetup が fontspec と unicode-math を読み込んで，本文フォントと数式フォントを New Computer Modern という最強のフォントにしてくれます．unicode-math により，多くの数学記号が Unicode で直接入

力できるようになるので，表8の記号もUnicodeで対応する文字があれば入力できるようになります．

ただ，これにも欠点はあります．

まず，以前のLaTeXで区別されていた文字，\hbarと\hslashはhの横棒が水平，傾きという区別がありましたが，unicode-mathでは両方とも傾いているものになってしまいます．加えて，\emptysetには∅が割り当てられていて，\varnothingで出る⌀とは別でしたが，これもUnicode上で区別されないため同じ記号になってしまいます．類似例として，$\mathscr{A}$と$\mathcal{A}$が同じ文字になってしまうこともあります[59]．ただ，これらに関しては，書体設定を文字の種類ごとに変更することが可能なので，下記のように設定することで回避できます．

コード 22：unicode-mathの設定例

```
\usepackage{fontsetup}

\setmathfont{newcmmath-book.otf}[
range={scr, bfscr},
StylisticSet=1, % Mathscrの方だけ，より筆記体らしいもの（StylisticSet 1）に設定する
]
```

さらに，amsmathの\dotsの自動判別機能を破壊してしまう，他の細かい記号類の挙動が若干変わる（ただし，よっぽど変な書き方をしなければ大丈夫）という問題もあるようです．詳しくは，unicode-math の注意するべき落とし穴を参照してください．

2.3.3.2　mathtools, thmtools　mathtoolsは，amsmathの拡張パッケージで，バグフィックスや機能追加が行われています texdoc mathtools．具体的な機能としては，例えば参照された数式だけに参照番号を振るshowonlyrefsオプションや，サイズ調整のための\mathclapなどのコマンドが追加されます．

thmtoolsは，定理環境をさらに便利にするためのパッケージです texdoc thmtools．特に，定理環境の定義が\declaretheoremや\declaretheoremstyleなどのコマンドで楽に行えます．

まずmathtoolsの使用例を示します．主に，\mathclapや\DeclarePairedDelimiterなどの機能を紹介します．

コード 23：mathtoolsの使用例1

```
\usepackage{mathtools}
\DeclarePairedDelimiter{\abs}{\lvert}{\rvert} % 絶対値のコマンドを定義
% mathtoolsのdocにある例
\providecommand\given{}
\newcommand\SetSymbol[1][]{%
\nonscript\:#1\vert
\allowbreak
\nonscript\:
\mathopen{}}
\DeclarePairedDelimiterX\Set[1]\{\}{%
\renewcommand\given{\SetSymbol[\delimsize]}
#1
}
```

とプリアンブルに書いておきましょう．この\DeclarePairedDelimiterコマンドは，例えば絶対値記号のような，左右にペアで出てくる記号を定義するためのコマンドです．これを定義するだけで，\abs*のように*を付けると自動でサイズ調整してくれるようになります．\DeclarePairedDelimiterXは，さらに複雑なペアの記号を定義するためのコマンドで，中身をいろいろやるマクロも定義できます．

59）　オプションで設定可能ですが，どちらか片方に固定されてしまいます．

実行例 29：mathtoolsの使用例2

```
\begin{gather*}
% サイズ調整された絶対値記号
\abs*{\frac{a}{b}}\\
% amsmathでやるなら
\left\lvert\frac{a}{b}\right\rvert \\
% サイズ調整された集合の内包的記法
% （これはamsmathだと面倒な気がする）
\Set*{x \in \mathbb{R} \given
x > \frac{x^2}{1+\sin(x)}}
\end{gather*}
```

$$\left|\frac{a}{b}\right|$$
$$\left|\frac{a}{b}\right|$$
$$\left\{x \in \mathbb{R} \,\middle|\, x > \frac{x^2}{1+\sin(x)}\right\}$$

さらに，他のコマンドを紹介します．

実行例 30：mathtoolsの使用例3

```
\begin{gather*}
  f(x) := x^2 \\
  f(x) \coloneqq x^2 \\
  A = \sum_{1 \leq i < j \leq n}
   x_i y_j + B \\
  A = \sum_{\mathclap{1 \leq i <
   j \leq n}} x_i y_j + B\\
  X = \sum_{1 \le i \le j \le k \le n} A_{ijk} \\
  X = \smashoperator{\sum_{1 \leq i
   \leq j \leq k \leq n}} A_{ijk}
\end{gather*}
\[
y = \frac{
  \splitfrac{a + b + c + d
  + e}{+ f + g + h + i + j}
}{
  x^2 + 1
}
\]
```

$$f(x) := x^2$$
$$f(x) \coloneqq x^2$$
$$A = \sum_{1 \leq i < j \leq n} x_i y_j + B$$
$$A = \sum_{1 \leq i < j \leq n} x_i y_j + B$$
$$X = \sum_{1 \le i \le j \le k \le n} A_{ijk}$$
$$X = \sum_{1 \le i \le j \le k \le n} A_{ijk}$$
$$y = \frac{\begin{array}{c} a + b + c + d + e \\ + f + g + h + i + j \end{array}}{x^2 + 1}$$

定義を表す:=のような記号は，このパッケージが提供する\coloneqq の方が:=よりもスペースが詰まっていて見栄えが良いです．また，分数の中で長い式を書くときに\splitfracを使うと，分子を複数行に分割して書くことができます．数式では添え字などに合わせて，空白が自動的に確保されますが，\mathclapや\smashoperatorを使うと，その分をうまく詰めることができます．

このような細かい表記の調整ができるのがmathtoolsの強みです．

thmtoolsの使用例も示します．

コード 24：thmtoolsの使用例

```
\usepackage{thmtools,amsthm}
\declaretheoremstyle[
  spaceabove=1em, spacebelow=1em,
  headfont=\bfseries,
  notefont=\mdseries, notebraces={(}{)},
  bodyfont=\normalfont,
  postheadspace=1em
  ]{jpdefinition}
```

```
  \declaretheoremstyle[
    spaceabove=1em, spacebelow=1em,
    headfont=\bfseries,
    bodyfont=\normalfont,
    postheadspace=1em
]{jplain}
\declaretheorem[style=jplain, name=定理, numberwithin=section]{thm}
\declaretheorem[style=jplain, name=補題, sibling=thm]{lem}
\declaretheorem[style=jplain, name=命題, sibling=thm]{prop}
\declaretheorem[style=jpdefinition, name=証明, numbered=no, qed=\qedsymbol]{proof1}
```

などのように[60]，まずスタイルを宣言し，環境をそれを使って定義する，という流れになります．スタイルの宣言では，定理環境の前後のスペースや，見出しのフォント，ノートのフォントや括弧などを細かく設定できます．あとは，

実行例 31：thmtoolsの使用例

```
\begin{thm}[なんとかの定理]
\[
  E \neq mc^3
\]
\end{thm}
```

定理 2.2 (なんとかの定理)**.**

$$E \neq mc^3$$

例えばナンバリングをセクションごとにするかどうか，などはamsthmだとカウンターを書き換えるような非直感的な方法を採らざるをえないのですが，thmtoolsではnumberwithinオプションで簡単に設定できます．このように，定理環境の定義が非常に柔軟に行えるのがthmtoolsの強みです．

2.4 自然科学系向けパッケージ

2.4.1 physics系

物理のパッケージとしては，physicsが有名です texdoc physics．ただ，このパッケージは古い上，あまりよろしくない挙動[61]，今ではphysics2を使うことが推奨されているようです texdoc physics2．これについては脱 physics パッケージして physics2 パッケージを使おうに詳しいです．なお，physics2も更新は停滞しており，フォークのphysics3が開発されているようです texdoc physics3．

ただ，physics2,3の設計思想は大きく異ならないようなので，ここではphysics2を前提に説明します．

physicsパッケージの機能は多岐にわたり，それだけ多くのものを上書きしてしまうため，physics2では，機能をモジュール化して必要なものだけ読み込めるようになっています．また，physicsで提供されていた，微少量や微分の記号は実装されていません．これの代替となる別のパッケージがderivativeやdiffcoeffです texdoc derivative texdoc diffcoeff．

2.4.1.1 微分

さて，derivativeとdiffcoeffの簡単な使用例を示します．

コード 25：derivativeとdiffcoeffのロード

```
\usepackage[italic=true]{derivative}
\usepackage{diffcoeff}
```

60） これは本稿のスタイルです．
61） 第2.4.3節で説明するようなコマンドの競合，非推奨となったxparseのオプションの使用など．

```
\difdef { f, s, c, l }{} % diffcoeffのスタイル設定
{
  op-symbol = d,
  op-order-nudge = 1 mu,
  *slash-sep = { 0 mu, 1 mu }
}
```

いずれのパッケージも，ISO 80000-2という数学記号に関する規格に従い，微分の記号を直立体に，すなわち$\mathrm{d}x$のような表記にするのがデフォルトですが，上ではそれを上書きして数学の慣習に従う記法にするための設定も示しています．

実行例 32：derivativeとdiffcoeffの使用例

```
\begin{gather*}
% derivativeの例
\odv{f}{x} \quad \pdv{f}{x} \quad \pdv{f}{x,y}
\quad \pdv*{f}{x,y} \quad \diffp[1,2]{f}{x,y}\\
% diffcoeffの例
\diff{f}{x} \quad \diffp{f}{x}
\quad \diffp{f}{x,y} \quad
\diffp*{f}{x,y} \quad \diffp[1,2]{f}{x,y}
\end{gather*}
```

$$\frac{df}{dx} \quad \frac{\partial f}{\partial x} \quad \frac{\partial^2 f}{\partial x\,\partial y} \quad \frac{\partial^2}{\partial x\,\partial y} f \quad \frac{\partial^3 f}{\partial x\,\partial y^2}$$
$$\frac{df}{dx} \quad \frac{\partial f}{\partial x} \quad \frac{\partial^2 f}{\partial x\,\partial y} \quad \frac{\partial^2}{\partial x\,\partial y} f \quad \frac{\partial^3 f}{\partial x\,\partial y^2}$$

どちらも基本的な機能は同じでしょう．なお，diffcoeffのほうが階数の計算が自動で行われるなど，細かい機能が充実しているようです．

実行例 33：diffcoeffの階数自動計算・n変数の微分の表記

```
\begin{gather*}
\diff[\alpha-3, \beta+1]{f}{x,y} \\
\diff<n>{f}{x_1,x_2,x_3,...,x_n} \\
\diffp<n>{f}{x_1,x_2,x_3,...,x_n}
\end{gather*}
```

$$\frac{d^{\beta+\alpha-2} f}{dx^{\alpha-3} dy^{\beta+1}}$$
$$\frac{d^n f}{dx_1\,dx_2\,dx_3 \cdots dx_n}$$
$$\frac{\partial^n f}{\partial x_1\,\partial x_2\,\partial x_3 \cdots \partial x_n}$$

ここで<>は階数を上書きするオプションです．これを入れないと，コンマ区切りされた要素の数だけになってしまいます．\dotsを入れずに，...と書いておくと，（今までのパッケージでは難しかったそうですが）正しく省略されます．

実行例 34：diffcoeffのn変数の微分の表記2

```
\begin{gather*}
  % 正しい例
  \diff<n>{f}{x_1,x_2,...,x_n} \\
  % または，dotsにn-3の階数を割り当てて
  \diff[1,1,n-3]{f}{x_1,x_2,...,x_n} \\
  % さらに複雑な例
  \diff<n>{f}{u:a,v:b,...,z:k}
  \quad \diff<n>{f}{u:a,v:b,...,z:k} \\
  % よくない例
  \diff{f}{x_1,x_2,\ldots,x_n}
\end{gather*}
```

$$\frac{d^n f}{dx_1\,dx_2\cdots dx_n}$$
$$\frac{d^n f}{dx_1\,dx_2\cdots dx_n}$$
$$\frac{d^n f}{du^a dv^b\cdots dz^k} \quad \frac{d^n f}{du^a dv^b\cdots dz^k}$$
$$\frac{d^4 f}{dx_1\,dx_2\,d\ldots\,dx_n}$$

2.4.1.2 物理の話

physics2の使い方を簡単に説明します．パッケージの読み込みは，

コード 26：physics2の使用例1

```
\usepackage{physics2}
\usephysicsmodule{ab,ab.braket,bm-um.legacy} % モジュールの読み込み
```

のように行います．モジュールは手動で読み込むことになります．

physicsで\qtyとして提供されていた括弧の自動調整は，本パッケージではabモジュールとなっています．

実行例 35：physics2の使用例2

```
\begin{gather*}
  \ab ( \frac{1}{2} ) \\
  \ab [ \frac{1}{2} ] \\
  \ab \{ \frac{1}{2} \} \\
  \ab \| \frac{1}{2} \| \\
  \ab | \frac{1}{2} | \\
  \ab\biggg| \frac{1}{2} |
\end{gather*}
```

$$\left(\frac{1}{2}\right)$$
$$\left[\frac{1}{2}\right]$$
$$\left\{\frac{1}{2}\right\}$$
$$\left\|\frac{1}{2}\right\|$$
$$\left|\frac{1}{2}\right|$$
$$\left|\frac{1}{2}\right|$$

ab.brakketまたはbraketモジュールはブラケット記法を提供するもので，その他に対角行列などのモジュールもあるようですので，各自必要に応じてドキュメントを参照してください texdoc physics2.

2.4.2 化学

化学のパッケージとして，mhchemやchemmacros,chemformulaやchemfigなどがあります．

texdoc mhchem texdoc chemmacros texdoc chemformula texdoc chemfig.

昔には，日本の藤田眞作さんが開発されたX^MTEXがありました texdoc XyMTeX．今でもTEXLiveに入っていますが，更新は停滞しているようです．

mhchemは比較的歴史のあるパッケージで，化学式と反応式を簡単に入力できるようにするものです．MathJaxなどでも採用されているようですし，極めて直感的な入力をサポートしています．chemmacrosは，化学式の入力をサポートするだけでなく，化学に関する様々なマクロを提供するパッケージです．chemformulaは，chemmacrosの機能の一

部を切り出したもので，化学式の処理を専門に行うパッケージです．他にも，chemnum というパッケージもあり，これは化合物に番号を付けるためのパッケージです texdoc chemnum．chemfig は，後述する TikZ（第 2.6.3 節）を使って炭素骨格を書くためのパッケージです．他にも化学に関する TikZ 系のパッケージはありますが，第 2.6.3 節で紹介します．

実行例 36：mhchem の使用例

```
\begin{gather*}
  \ce{H2O} \\
  \ce{CO2} \\
  \ce{SO4^2-} \\
  \ce{^{227}_{90}Th+} \\
  \ce{2H2 + O2 -> 2H2O} \\
  \ce{A <=> B} \\
  \ce{A <-> B} \\
  \ce{A -> B} \\
  \ce{A <- B}
\end{gather*}
```

$$\begin{gathered}
\mathrm{H_2O} \\
\mathrm{CO_2} \\
\mathrm{SO_4^{2-}} \\
\mathrm{^{227}_{90}Th^{+}} \\
\mathrm{2H_2 + O_2 \longrightarrow 2H_2O} \\
\mathrm{A \rightleftharpoons B} \\
\mathrm{A \longleftrightarrow B} \\
\mathrm{A \longrightarrow B} \\
\mathrm{A \longleftarrow B}
\end{gathered}$$

chemmacros も基本的な使い方は同様で，\ch コマンドで化学式を入力できます．

実行例 37：chemformula の使用例

```
\begin{center}
  \ch{H2O} \\
  \ch{CO2} \\
  \ch{SO4^2-} \\
  \ch{^{277}_{90} Th+} \\
  \ch{2 H2 + O2 -> 2 H2O} \\
  \ch{A <=> B} \\
  \ch{A <-> B} \\
  \ch{A -> B} \\
  \ch{A <- B}
\end{center}
```

$$\begin{gathered}
\mathrm{H_2O} \\
\mathrm{CO_2} \\
\mathrm{SO_4^{2-}} \\
\mathrm{^{277}_{90}Th^{+}} \\
\mathrm{2H_2 + O_2 \longrightarrow 2H_2O} \\
\mathrm{A \rightleftharpoons B} \\
\mathrm{A \longleftrightarrow B} \\
\mathrm{A \longrightarrow B} \\
\mathrm{A \longleftarrow B}
\end{gathered}$$

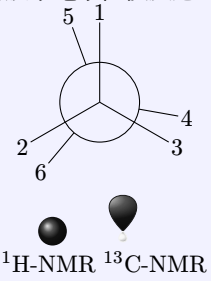
タイトルにも書きましたが，実は\ch は chemformula の機能です．既定では，chemmacros は chemformula を使って化学式を処理するようになっています．差異として，mhchem は空気を読んで化学式を解釈するのに対し，chemformula はより厳密に化学式を解釈するため，適宜スペースを入れる必要があります．また，chemformula は TikZ による矢印が使われ，若干見た目が異なります．

他の chemmacros の機能を見てみましょう．

実行例 38：chemmacrosの使用例

```
\begin{center}
  % acid-base Module
  \pH, \pOH, \pKa[1], \pKb[2] \\
  % phase Module
  % ドキュメントの例を引用
  \ch{C\sld{} + 2 H2O\lqd{} %
  -> CO2\gas{} + 2 H2\gas} \\
  これを起こすのには NaCl\aq を要する. \\
  % isotope Module
  \isotope{14,C} \\
  % mechanism Module
  求核置換反応の\mech[1] \\
  離脱反応の\mech[e1] \\
  芳香族求電子置換反応の\mech[se] \\
  % newman Module
  \newman(80){1,2,3,4,5,6} \\
  % orbital Module
  \orbital{s} \orbital{sp} \\
  % spectroscopy Module
  \NMR* \NMR*{13,C}
\end{center}
```

pH, pOH, $\mathrm{p}K_{\mathrm{a}1}$, $\mathrm{p}K_{\mathrm{b}2}$
$\mathrm{C(s) + 2\,H_2O(l) \longrightarrow CO_2(g) + 2\,H_2(g)}$
これを起こすのには $\mathrm{NaCl(aq)}$ を要する.
$^{14}_{6}\mathrm{C}$
求核置換反応の$\mathrm{S_N1}$
離脱反応のE1
芳香族求電子置換反応の$\mathrm{S_E}$



^{1}H-NMR ^{13}C-NMR

いろいろなモジュールがあるので，必要に応じてドキュメントを参照してください texdoc chemmacros.

2.4.3 SI単位系

ここでは，SI単位系などの表記ルールを一発で適用できるsiunitxを紹介します texdoc siunitx.
使い方は極めて単純です.

コード 27：siunitxの使用例

```
\usepackage{siunitx}
```

もちろん細かい表記の調整も可能ですが，とりあえずこれだけで，例39のような表記ができるようになります.

実行例 39：siunitxの使用例

```
\begin{gather*}
  \qty{1.23}{\kilo\gram\metre
  \per\second\squared} \\
  \qty{1.23}{kg.m.s^{-2}} \\
  \unit{GHz} \\
  \num{0.12345(67)} \\
  \num{1.2345 +- 0.0006} \\
  \num{1.2345(6)} \\
  \complexnum{3+4i} \\
  \ang{1;23;45} \\
  \qtyproduct{2x4x5}{m} \\
  \num{<<1} \\
  \num{10.56(12:34)}
\end{gather*}
```

$$1.23\,\mathrm{kg\,m\,s^{-2}}$$

$$1.23\,\mathrm{kg\,m\,s^{-2}}$$

$$\mathrm{GHz}$$

$$0.123\,45(67)$$

$$1.2345(6)$$

$$1.2345(6)$$

$$3+4\mathrm{i}$$

$$1°23′45″$$

$$2\,\mathrm{m}\times 4\,\mathrm{m}\times 5\,\mathrm{m}$$

$$\ll 1$$

$$10.56^{+0.12}_{-0.34}$$

基本的な単位に関してはプレーンテキストで入力して認識してくれます．

第 2.4.1 節で軽く触れましたが，physics パッケージは，\qty コマンドで括弧の自動調整を行うため，siunitx の \qty コマンドと競合してしまいます．そのため，昔は\SI という代替コマンドが定義されていましたが，v3 以降は定義されなくなったため注意が必要です．もっとも有力な解決策は，physics を使わないことです．

2.5 その他便利パッケージ

2.5.1 色

色を使うためのパッケージとしては，color や xcolor，luacolor などがあります．

texdoc color texdoc xcolor texdoc luacolor

color は古いパッケージで，xcolor は color の拡張，luacolor は LuaLaTeX 専用のパッケージで，Lua の機能を活かして色を定義することができます．

また，ninecolors というパッケージもあり，これは xcolor を使って，よく使われる 13 色[62]の各色相に対して，9 段階の輝度の色を定義してくれるパッケージです texdoc ninecolors．この輝度レベルが 5 以上離れていれば，WCAG 2.1 で推奨されるコントラスト比 4.5:1 を満たすようになっているそうです．なお，黒は gray0，白は gray10 となっているので，例えば白背景で本文に使うなら輝度レベル 5 以下の色を使うといいでしょう．

DVI に書き出していた LaTeX においては，color は DVI に特殊命令（\special）を出力することで色を指定しており，当時の pLaTeX などではそれが和文の文字送りやアキに干渉することがあったようです．LuaLaTeX などのモダン LaTeX では PDF に直接色の命令を出力するため，また和文の扱い方が根本的に変更されているため，改善されているようです．

xcolor による文字の色付けの例を示します．例えば，\textcolor{red}{赤い部屋}のように書くと，赤い部屋のように赤い文字になります．!は混色の割合を表すもので，red!50!blue のように書くと赤と青を 1:1 で混ぜた色になります．

ただ，このように本文をカラフルにしてしまうと非常に読みにくくなるため，実際には TikZ や tcolorbox，そしてハイパーリンクを表すために hyperref などの内部で勝手に使われることが多いでしょう[63]．

62） gray red brown yellow olive green teal cyan azure blue violet magenta purple
63） 本稿でも，内外へのリンクは全て青色にしてあります．

2.5.2 シンタックスハイライト

コードのシンタックスハイライトを行うためのパッケージとしては，listings や minted などがあります texdoc listings texdoc minted.

もちろん，verbatim環境を使えばコードをそのまま表示できますが，シンタックスハイライトはされません．

minted は外部ライブラリを使ってシンタックスハイライトを行います．外部コマンドの呼び出しは，当然ながら任意コードの実行につながるため，セキュリティ上のリスクがあります．そのため，以前は-shell-escape オプションを付けてコンパイルする必要がありましたが，最近の TeXLive には承認されたこのライブラリが組み込まれており，実行のホワイトリストに入っているため，-shell-escape オプションなしで使えるようになっています．

そのため，minted は listings よりも高品質なシンタックスハイライトが可能で，実行上の手間もなくなったため，現在では minted を使うことをおすすめします．

ここまでのコードブロックや実行例は，第2.6.4節とこの minted で実現しています．

そのため，これ以上の例は不要かと思いますが，実際にどのように書くかを示します．

実行例 40：minted の使用例

```
\begin{minted}{python}
def f(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return f(n-1) + f(n-2)
\end{minted}
```

```python
def f(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return f(n-1) + f(n-2)
```

2.6 図表・箇条書き・飾り

ここでは，図表や箇条書き，そして飾りなどを作るためのパッケージを紹介します．

2.6.1 graphicx

graphicx は，主に画像を挿入するためのパッケージです texdoc graphicx．graphicx を使うと，\includegraphics コマンドで画像を挿入できるようになります．普通，figure 環境の中で使うことが多いでしょう．コード28は図1を表示するために使っているコードです．

コード 28：graphicx の使用例

```
\begin{figure}[htbp]
\centering
\includegraphics[width=0.9\textwidth]{./comparefonts.pdf}
\caption{Latin Modern, New Computer Modern, STIX Two Math, Cambria Mathの比較．}
\label{fig:comparefonts}
\end{figure}
```

ここで figure 環境のオプション htbp は，ここに図を置いてほしい（h），ページの上（t），ページの下（b），次のページ（p）などの配置の希望を表すものです．このような場所指定子は float パッケージでさらに細かく指定できるようになります texdoc float．主に使うのは H で，これは強制的にその場所に配置させます．この優先順位に基づいて，LaTeX が自動的に図表を配置してくれます．このことをもって，フロートなどという呼び方をします．

ファイル形式については，以前は EPS が DVI との兼ね合いで主流でした．今のモダン TeX では PNG や JPEG なども扱えます．PDF も扱えるため，ベクターや文字を含む図を入れる場合は PDF がよいでしょう．

コラム 16：LaTeXにおける画像の処理

texdoc graphics を見ると，（何を書くのか忘れた）

2.6.2 enumitem

第 1.4.4 節で説明した箇条書き環境をさらに便利にするのが enumitem です texdoc enumitem．このパッケージを使うと，例えば例 41 のようにオプションを指定することで，番号のスタイルを変更できます．

実行例 41：enumitem の使用例

```
\begin{enumerate}[label=\arabic*.,
  labelindent=\parindent,
  leftmargin=*,
  ]
  \item 1段目
  \begin{enumerate}[label=\Alph*.]
    \item 2段目
    \item 2段目
  \end{enumerate}
  \item 1段目
  \item 1段目
\end{enumerate}
```

1.1段目
  A.2段目
  B.2段目
2.1段目
3.1段目

また，ラベルの余白や段落の余白なども細かく調整できます．

2.6.3 TikZなど

LaTeX はお絵かきソフトではなく組版ソフトですので，図は挿入するものという設計思想でしょう．ただ，フォーマットを揃え，図の中でも綺麗に組版をするとなると，LaTeX に近い部分で図を作りたくなります．TikZ はその最も有力な選択肢の一つで，LaTeX のコードで図を描くことができます texdoc tikz．PGF はその下にあるパッケージで，pdfLaTeX や LaTeX(PostScript) の差異を吸収してくれるものです．ほぼ一体化しているため，PGF/TikZ などと呼ばれることもあります．

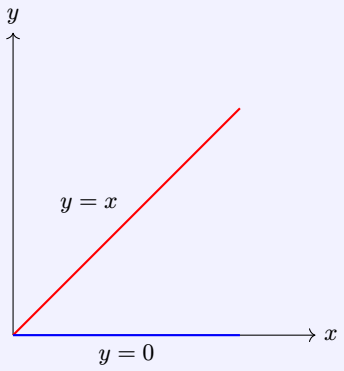
なお，PDF が主流ではなかった頃は，pstricks という PostScript を使って図を描くパッケージもありました texdoc pstricks．

TikZ は LaTeX 以上に難解で，ここで説明すると数倍の分量になりますので，TikZ の使い方を参照してください．

ここでは簡単な使用例を示します．

実行例 42：TikZの使用例

```
\begin{tikzpicture}
  \draw[->] (0,0) -- (4,0) node[right] {\(x\)};
  \draw[->] (0,0) -- (0,4) node[above] {\(y\)};
  \draw[thick, red] (0,0) -- (3,3);
  \draw[thick, blue] (0,0) -- (3,0);
  \node at (1.5,1.5) [above left] {\(y=x\)};
  \node at (1.5,0) [below] {\(y=0\)};
\end{tikzpicture}
```

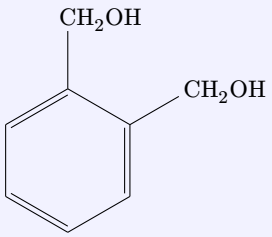


このように，内部で座標を指定して線を引いたり，ノードを配置したりできます．

このTikZやPGFを使って様々なことをするパッケージが盛んに開発されています．第2.4.2節で紹介したchemfigもその一つで，炭素骨格を描くためのパッケージです．

実行例 43：chemfigの使用例

```
\begin{center}
  \chemfig{*6(=-=(-CH_2OH)-(-CH_2OH)=-)}
\end{center}
```

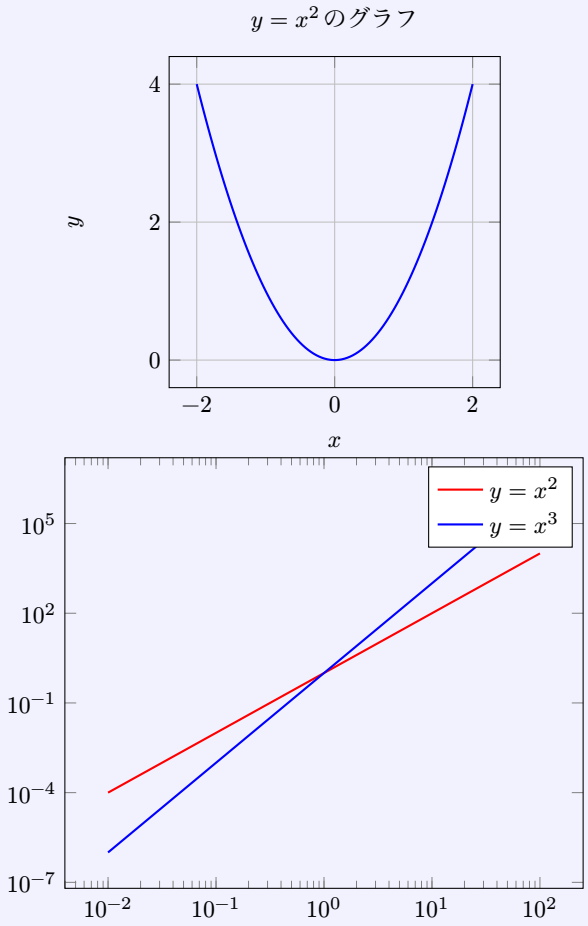


なお，第2.6.4節で紹介するtcolorboxもTikZを使って装飾されたボックスを作るためのパッケージです．

pgfplotsは，LaTeXのコードでグラフを描くためのパッケージです texdoc pgfplots．

実行例 44：pgfplotsの使用例

```
\centering
\begin{tikzpicture}
  \begin{axis}[
    xlabel=\(x\),
    width=0.8\textwidth,
    height=0.8\textwidth,
    ylabel=\(y\),
    title={\(y = x^2\)のグラフ},
    grid=both,
  ]
    \addplot[domain=-2:2, samples=100,
    thick, blue] {x^2};
  \end{axis}
\end{tikzpicture}
\begin{tikzpicture}
  \begin{loglogaxis}
    \addplot[domain=0.01:100, samples=100,
    thick, red] {x^2};
    \addplot[domain=0.01:100, samples=100,
    thick, blue] {x^3};
    \legend{\(y=x^2\), \(y=x^3\)}
  \end{loglogaxis}
\end{tikzpicture}
```



このように，普通のプロットだけでなく，対数プロットも簡単に描けます．今回は関数のプロットを示しましたが，外部からデータを読み込んでプロットすることもできます．もちろん gnuplot などで描いたグラフを画像として挿入することもできますが，本文執筆中にグラフのスタイルを調整したくなったとき，gnuplot などに戻って調整するのは面倒ですが，pgfplots ならばLaTeXのコードを直接いじるだけで調整できるのが便利でしょう．

また．TikZにはライブラリや拡張のためのパッケージがいくつもあります．内蔵ライブラリの読み込みは

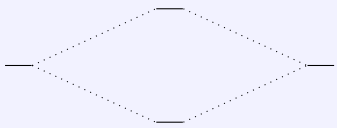
コード 29：TikZのライブラリの使用例

```
1 \usetikzlibrary{arrows.meta,angles,quotes} % ライブラリの読み込み
```

などのように行います．例えばcalcライブラリを使うと，座標の計算ができるようになります texdoc tikz．多数あるTikZ系のパッケージから，modiagram，pgf-spectra，pgf-interferenceなども紹介しておきます．
texdoc modiagram texdoc pgf-spectra texdoc pgf-interference
modiagramは分子軌道のエネルギー準位図を描くためのパッケージです．

実行例 45：modiagramの使用例

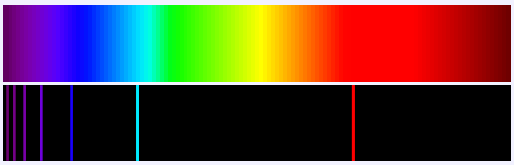
```
\begin{center}
  \begin{modiagram}
    \atom{left} { 1s = 0 }
    \atom{right} { 1s = 0 }
    \molecule { 1sMO = .75 }
  \end{modiagram}
\end{center}
```



pgf-spectraは原子のスペクトルを描くためのパッケージです．texdoc pgf-spectra．デフォルトでは，NISTのデータベースから中性，1価イオンの発光スペクトルを描くことができます．

実行例 46：pgf-spectraの使用例

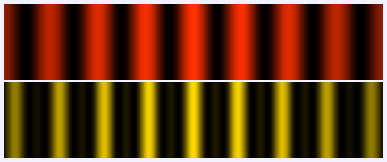
```
\begin{center}
  \pgfspectra \\
  \pgfspectra[element=H]
\end{center}
```



pgf-interferenceは干渉縞を描くためのパッケージです texdoc pgf-interference．

実行例 47：pgf-interferenceの使用例

```
\begin{center}
  \pgfinterferencepattern{screen-width=0.05,
  screen-height=0.01} \\
  \pgfinterferencepattern{screen-width=0.05,
  screen-height=0.01,
  wavelength=590e-9,slits=3}
\end{center}
```



2.6.4 tcolorbox

tcolorboxは，TikZを使って装飾されたボックスを作るためのパッケージです texdoc tcolorbox．

実行例 48：tcolorboxの使用例

```
\begin{tcolorbox}[colback=blue!5!white,
colframe=blue!75!black,title=テスト]
これが\texttt{tcolorbox}の例です．
色を変えたり，タイトルを付けたりできます．
\end{tcolorbox}
```

テスト

これがtcolorboxの例です．色を変えたり，タイトルを付けたりできます．

また，mintedなどと組み合わせて，コードを装飾されたボックスの中に入れることもできます．高度な例として，今まで使ってきた実行例環境は，side by sideを使って，左側にはコードを，右側には出力を表示するようにしています．これを第 1.5 節で説明したように，環境を定義して実現しています．自作のtcolorbox環境を作るためのコマンドが，\newtcolorboxや\NewTColorBoxです．

コード 30：texexample環境の定義例

```
\DeclareTCBListing[use counter=texex]{texexample}{ O{} m }{%
enhanced, sidebyside, sidebyside gap=5mm, lower separated=true,
colback=blue!5!white, colframe=blue!75!black, coltitle=white,
fonttitle=\sffamily\bfseries,
title={実行例 \thetcbcounter：#2},
listing engine=minted,
minted language=latex,
minted options={ % minted に関するオプション
  fontsize=\small,
  % breaklines=true, % 長い行を自動的に改行，自分の環境だと壊れるので使っていない
  autogobble=true},
},
#1 % さらにオプションを指定できるようにする
}
```

なお，この例にはカウンターを使っていますが，この詳細については第2.7.1節で説明します．

2.7 相互参照，参考文献

LaTeX を使う理由の一つが，LaTeX の相互参照や参考文献の機能が非常に強力であることです．例えば，第 2.7 節のように勝手に番号を振ってくれます．これの実装も含めて，概観しておきましょう．

2.7.1 LaTeXにおけるカウンター

LaTeX では，様々なものに番号を振るためのカウンターという仕組みがあります．普通のプログラミング言語でいえば整数型の変数のようなもので，LaTeX の内部で様々なものに番号を振るために使われています．普通に組んで，数字が出ている部分にはカウンターが使われているとみて良いです texdoc latex2e (13 Counters)．例えば，section環境はsectionカウンターを内部的につかって番号付けしていますし，図表や箇条書き，脚注も同様です．

このカウンターは，\alph や\arabic で取り出すことができます それぞれ，アルファベット小文字，アラビア数字などの形式で番号を出力するためのコマンドです．

実行例 49：カウンターの使用例

```
\newcounter{mycounter}
\setcounter{mycounter}{5}
\themycounter % \arabic と同じ
\\
\alph{mycounter} \\
\Alph{mycounter} \\
\arabic{mycounter} \\
\roman{mycounter} \\
\Roman{mycounter}
```

5
e
E
5
v
V

なお，\newcounter は新しいカウンターを定義するためのコマンドで，\setcounter はカウンターの値を設定するためのコマンドです．

このほかにも，\refstepcounter や\addtocounter などのコマンドもありますが，ここでは説明を割愛します texdoc latex2e (13 Counters).

2.7.2 相互参照の基本

LaTeX では，\label コマンドでラベルを定義し，\ref コマンドでそのラベルを参照することができます．これ

は，第 2.7.1 節で説明したカウンターに対して，対応する部分のカウンターを呼び出して参照しています．例えば，section 環境の中で\label コマンドを使うと，章の番号，figure 環境の中で\label コマンドを使うと，図の番号を参照することができます．

実行例 50：相互参照の基本

```
\begin{equation}
E = mc^2 \label{eq:einstein}
\end{equation}
式\ref{eq:einstein}のように参照できます．
```

$$E = mc^2 \tag{15}$$

式15のように参照できます．

また，\pageref コマンドを使うと，参照している部分がどのページにあるかも参照できます．
これらの相互参照をさらに便利にするためのパッケージがいくつかあります．

2.7.3 hyperref

hyperref は，HTML でやっていたようなハイパーリンクを LaTeX のドキュメント内で実現するためのパッケージです texdoc hyperref．hyperref を使うと，\href や\url コマンドで外部へのリンクを作ることができますし，\hyperref のオプションで，内部のリンクの色やスタイルを変更することもできます．基本的には，プリアンブルの設定の中で hyperref を読み込むだけで，LaTeX の相互参照が全てハイパーリンクになります．

これに加え，bookmark と xurl を読み込んでおくと，PDF のブックマークや URL の改行の処理が改善されます texdoc bookmark texdoc xurl．

コード 31：hyperref の使用例

```
\usepackage{hyperref}
\hypersetup{
  unicode, % Unicode対応，必須ではない
  bookmarksnumbered=true, % PDFのブックマークにセクション番号を付ける
  bookmarksdepth=3, % PDFのブックマークの深さ（subsubまで）
  colorlinks=true, % リンクを色付けする（四角い枠を付けることも可）
  linkcolor=blue, % 内部リンクの色
  citecolor=blue, % 参考文献のリンクの色
  urlcolor=blue, % URLのリンクの色
  hypertexnames=false, % PDFの内部での名前をセクション番号などにしない（行き先の衝突を避けるため）
  pdfusetitle % PDFのメタデータに\title{}や\author{}の内容を使う
  }

\usepackage{xurl}
\usepackage{bookmark}
```

hyperref は LaTeX の様々なマクロを置き換えるため，最後の方で読み込んだほうがよいはずです．

2.7.4 zref などの相互参照パッケージ

zref は[64]，これらの相互参照をさらにリッチにするためのパッケージです texdoc zref．今まで紹介した\label などは，全て .aux の外部ファイルに情報を書いておくことで実現されています．実際にこれを覗いてみると，ラベルには，そのときのカウンターの値とページ番号しか書かれていません．zref は，このラベルに様々な情報を付加することができ，他にもカウンターの扱いを柔軟にしたりなどの機能がついています．このあたりのモチベーションについては，alg-d 氏の「【TeX の理解を深める動画 17】zref【選択公理と同値な命題を数えよう】」も参考にしま

64) 実はこのあたりのパッケージは Heiko Oberdiek が書いている．

した．

2.7.4.1　zref　さて，そもそも zref は何ができるのでしょうか．LaTeX はある意味スクリプト言語のようなものなので，上で見たような\ref などはその行で呼び出されたときのカウンターの値を参照します．そのため，例えば，本稿で紹介するパッケージの数を LaTeX で数えるため，自分でカウンターを定義して，パッケージの紹介の度にそのカウンターをインクリメントしていくといったことが可能なわけですが，これを前書きで「このドキュメントで紹介するパッケージの数はthepackages です」と書くと，文書の最初ではカウンターは0のままなので，「このドキュメントで紹介するパッケージの数は 0 です」となってしまいます[65]．ただ，texdoc zref を見ると分かるように，zref が提供してくれるコマンドの大半は@から始まる開発向けのコマンドです（コラム 11 も参照）．

一方，ユーザー向けのモジュールもいくつか提供されています．zref に同梱されているモジュールで，ユーザー向けのものを表7に示します．多くは既存のパッケージを zref のモジュールとして再実装したもので，

表7: zref のユーザーモジュールと対応する既存のパッケージ・マクロ

zref のモジュール	既存の対応パッケージ・マクロ	概要
user	LaTeX 標準の \label, \ref, \pageref	LaTeX 標準の相互参照と同等の機能を持つユーザーインターフェースを提供します．
lastpage	lastpage	文書の最後にラベルを配置し，最終ページを参照する機能を提供します．
totpages	totpages	文書の総ページ数を取得・計算する機能を提供します．
perpage	perpage	perpage の \MakePerPage と同様の構文・意味で，指定したカウンタをページが変わるごとにリセットします．
titleref	titleref, nameref	セクションやキャプションのタイトルテキストへの参照機能を提供します．
xr	xr, xr-hyper	外部ドキュメントのラベルを読み込み参照する機能を提供し，xr-hyper の構文もサポートします．
dotfill	LaTeX 標準の \dotfill	\dotfill と同様の機能ですが，最小ドット数や間隔などを細かくカスタマイズ可能なマクロを提供します．
savepos	pdfTeX などの \pdfsavepos 等	コンパイル時のページ上の絶対座標（x, y）を保存し，取得するインターフェースを提供します．
counter	hyperref （\autoref など）	\autoref のような機能で利用される，参照を行った際のカウンタ名を記憶・提供します．

ここに含まれない面白いモジュールとして，runs があります．これは，.aux ファイルが削除されずに複数回コンパイルされたときに，何回目のコンパイルであるかをカウントするためのモジュールです．

65）　なお，図表などであれば，List of figures の環境で実現するこは可能でしょう．

コード 32：zrefのrunsモジュールの使用例

```
\usepackage[runs]{zref}
% 本文で
この文書は\zruns{}回タイプセットされました．
% この文書は 3 回タイプセットされました，などと出力される．
```

ちなみに，これを latexmk などで処理すると，コンパイルの度に出力結果が変わるため，相互参照が解決されていないとみなされ，何度も何度も[66]タイプセットされます．また，texdoc zref (3.9 Module runs) でも exp, すなわち実験的な機能だとされていることにご注意ください．

2.7.4.2 zref-cleverなど ref の前に，例えば図1の「図」の部分を自動的に付けるためのパッケージがzref-cleverです texdoc zref-clever．cleveref も同様の機能を持つパッケージで，zref-clever は zref と組み合わせて使うためのものです texdoc cleveref．

zref は非常に活発にメンテナンスされているパッケージで，その上に構築された zref-clever の信頼性はそれなりに高いです．なお，cleveref はあまりメンテされていませんが，一応動いてはいます．また，zref-clever は LaTeX カーネルの\label の拡張という最近の動向を受けて開発されていて，親和性が高いとも言えます．とはいえ，zref-clever もそれほど活発ではなく，日本語対応のパッチがマージされていないため，プリアンブルで定義する必要があります．なお，この定義については，yarakos95 さんの zref-clever の日本語対応ブランチを利用するとよいでしょう．以下の定義でも大いに参考にしています．もちろん，自分の好みや出版先のスタイルに合わせてカスタムすることも容易です．一度環境を定義してしまえば，あとは\zcref コマンドで参照できるようになります．

コード 33：zref-cleverの使用例

```
\zcDeclareLanguage{japanese}
\zcLanguageSetup{japanese}{
  namesep   = {\nobreak}, % 名前と番号のスペースは（LuaTeX-jaに任せるので）不要
  pairsep   = {と}, % and に相当
  listsep   = {, }, % 英語の , に相当する約物
  lastsep   = {, および}, % a, b, c and d の最後の分割に入れる文字
% 中略 aからbというときの「から」などもここまでで定義する
  type = book,  Name-sg = 第,  Name-pl = 第,  refbounds = {,,巻,}, % 第n巻までをリンクとする
  type = part,  Name-sg = 第,  Name-pl = 第,  refbounds = {,,部,},
  type = chapter,  Name-sg = 第,  Name-pl = 第,  refbounds = {,,章,},
  type = section,  Name-sg = 第,  Name-pl = 第,  refbounds = {,,節,},
  type = paragraph,  Name-sg = 第,  Name-pl = 第,  refbounds = {,,段落,},
  type = appendix,  Name-sg = 付録,  Name-pl = 付録,
  type = page,  namesep = {\nobreakspace},  Name-sg = p.,  Name-pl = pp.,
    rangesep = {\textendash},  rangetopair = false,
  % rangesepはページ数の 10-20のようなときの区切り文字.
  % ここでは頭にスペースを入れておく（欧文同士なので）
  type = line,  namesep = {\nobreakspace},  Name-sg = line,  Name-pl = lines,
    rangesep = {\textendash},  rangetopair = false,
  type = figure,  Name-sg = 図,  Name-pl = 図,
  type = table,  Name-sg = 表,  Name-pl = 表,
% 中略 以下，各環境の定義を追加していく
  }
\zcsetup{
  cap = true, % capitalize 参照の最初の文字を大文字にするか，日本語ではどうでもいい（はず）
  nameinlink = true, % 参照の中に名前を入れるか（図1の「図」の部分もリンクにするか）
  lang=japanese % 上で defした日本語を使う
```

66） latexmkなどのツールの上限まで．

```
}
```

のように定義しておけば，あとは\labelで貼ったラベルを\zcrefで参照するだけで，自動的に「図1」などと出力してくれます．また，複数のラベルをまとめて参照することもできます．

実行例 51：zref-cleverの使用例

```
\begin{align}
  0 &\coloneqq \emptyset \label{eq:zero} \\
  1 &\coloneqq \{0\} \label{eq:one} \\
  2 &\coloneqq \{0,1\} \label{eq:two} \\
  &\qquad\vdots \notag \\
  \omega &\coloneqq \{0,1,2,\dots\}
  \label{eq:omega}
\end{align}
\zcref{eq:zero,eq:one,eq:two}は自然数の定義，
\zcref{eq:omega}は自然数全体の
集合の（素朴な）定義です．
```

$$0 \coloneqq \emptyset \tag{16}$$
$$1 \coloneqq \{0\} \tag{17}$$
$$2 \coloneqq \{0,1\} \tag{18}$$
$$\vdots$$
$$\omega \coloneqq \{0,1,2,\dots\} \tag{19}$$

式 (16) から (18) は自然数の定義，式 (19) は自然数全体の集合の（素朴な）定義です．

2.8 参考文献のために

さて，相互参照の仕組みを使えば，文中または脚注で参考文献を出したり，引用順に参考文献リストを作ったりすることもできそうです．

もちろん，LaTeXにはそのための機能が用意されています．thebibliography環境を使うと，参考文献リストを作ることができます．引用は\citeコマンドを使います．

コード 34：thebibliography環境の使用例

```
\begin{thebibliography}{9}
% 引数はラベル（番号）の幅，すなわち桁数分
\bibitem{lamport94}
  Leslie Lamport,
  \textit{\LaTeX: A Document Preparation System},
  Addison-Wesley, 1994.
\bibitem{日本語文献}
  著者名，
  『文献名』，
  出版社，出版年．
```

2.8.1 biblatex

さて，thebibliography環境と\citeコマンドを使う方法は，参考文献の数が少ない場合や，スタイルが特に指定されていない場合には十分ですが，参考文献の数が多い場合や，スタイルが指定されている場合には，管理が大変になります．より柔軟な参考文献管理のためのパッケージがbiblatexです texdoc biblatex．biblatexを使うと，参考文献のデータベースを.bibファイルとして管理し，フォーマットは.bbxファイルや.cbxファイルでカスタマイズすることができます．並び替えなどはbiberという外部ツールを呼び出して行うのが一般的です．

基本的な使い方は以下のようになります．

実行例 52：biblatexの使用例

```
% プリアンブルで
% \usepackage[backend=biber,style=numeric-comp]{biblatex}
% numeric-compは数字で引用し，
% 複数の文献をまとめて[1-3]のようにするスタイル．
% 参考文献データベースファイルを指定
% \addbibresource{references.bib}

% 文中で引用
\textcite{bibunsho2023} など
% 参考文献リストの出力
% \printbibliography
\\
（出力結果は\zcref{sec:sankou}のようになります．）
```

奥村晴彦，黒木裕介 [1] など
（出力結果は付録Dのようになります．）

ここで，references.bibは以下のような内容のファイルになります．なお，このファイルのフォーマットは，基本的にBibTeXのフォーマットの上位互換となっています[67]．

コード 35：参考文献データベースの例

```
@book{bibunsho2023,
  edition   = {改訂第9版},
  title     = {{\LaTeX}{美文書作成入門}},
  isbn      = {978-4-297-13889-9},
  publisher = {技術評論社},
  author    = {{奥村晴彦} and {黒木裕介}},
  date      = {2023-12-09},
  language  = {japanese}
}
```

これについては，自分は基本的にZoteroで管理していて，そこから.bibファイルをエクスポートして使っています．

フォーマットについて，基本的にnumericで番号順，authoryearで著者年順になりますが，和文には対応していないため，自分でスタイルファイルを作る必要があります．人文系で使われるJPAスタイルは既存のもの（sbtseiji/biblatex-jpa）があります．理工系であれば，基本的にはそのまま入れてしまっても一応出力されますが，ピリオドが入ったり，欧文の引用符がついてしまったり，タイトルの欧文文字だけがイタリックになったりなどの問題があるため，それを上書きする以下のようなスタイルファイルを作るとよいでしょう．

引用記号のスタイルは.cbx，参考文献リストのスタイルは.bbxで定義します．

コード 36：plain-title.cbxの例

```
\ProvidesFile{plain-title.cbx}

% テキストに出てくる[1]のようなスタイルはそのままでよいので．何もしない．
\RequireCitationStyle{numeric-comp}

\endinput
```

67） 追加の項目の他は互換です．

コード 37：plain-title.bbxの例

```
\ProvidesFile{plain-title.bbx}[2026/03/19 v1.0 Plain Title Bibliography Style]

% 標準の numeric-comp 文献リストスタイルを継承
\RequireBibliographyStyle{numeric-comp}

% 全てにおいてそのまま出力させる
\DeclareFieldFormat{title}{#1}
\DeclareFieldFormat[article,inbook,incollection,inproceedings,patent,thesis,unpublished]{title}{#1}
\DeclareFieldFormat[book,report,manual,collection]{title}{#1}
\DeclareFieldFormat{journaltitle}{#1}
\DeclareFieldFormat{booktitle}{#1}
\DeclareFieldFormat{maintitle}{#1}
\DeclareFieldFormat{issuetitle}{#1}

\endinput
```

これらのファイルを執筆中の.texファイルと同じディレクトリに配置し，プリアンブルで以下のように読み込みます．

コード 38：自作スタイルの読み込み

```
\usepackage[style=plain-title]{biblatex}
\addbibresource{references.bib}
```

これで日本語として最低限体裁が整った参考文献リストが出力されるようになります．ただ，これをやってしまうと欧文もイタリックにならなくなってしまうため，欧文タイトルだけ普通のスタイルを適用するには，ifを入れて，languageフィールドで分岐させるとよいでしょう．

2.9 目次・索引

2.9.1 目次

目次は単純に\tableofcontentsコマンドを呼び出すだけで出力できます．\tableofcontentsの深さを変更したい場合は，\setcounterコマンドでtocdepthカウンターを設定します．

2.10 索引

本稿ではmakeidxとupmendixで索引を付けています．makeidxは索引を作るためのパッケージで，upmendixは，実際にページ番号を対応させたり，ソートしたりといった役割を果たす外部ツールです．makeidxを使うと，\indexコマンドで索引を付けることができます．

コード 39：索引の使用例

```
\usepackage{makeidx}
% seeの表示を→にする．
\renewcommand{\seename}{\textrightarrow}
\makeindex
\begin{document}
\index{LaTeX@\LaTeX!ぱっけーじ@パッケージ}
% 索引に「LaTeX」の項目を作り，その下に「パッケージ」というサブ項目を作る例．
```

```
% @の前は読み，@の後は表示される文字列．
\index{らてっくす@ラテックス|see{LaTeX@\LaTeX}}
% 索引に「ラテックス」という項目を作るが，矢印などで「LaTeX」の項目を参照する例．
\printindex
\end{document}
```

とすると，最後に索引が出力されるようになります．

ただ，makeidxがやってくれるのは，.idxファイルに\indexが書かれた場所のページ番号と項目の情報を書き出すことだけで，実際に索引を作るためにはupmendixなどの外部ツールを使って，.idxファイルを処理する必要があります．

例えば，main.texをコンパイルしてmain.idxが生成された後に，コマンドラインで

コード 40：upmendixの使用例

```
upmendix main.idx
```

とすると，main.indというファイルが生成されるので，再度main.texをコンパイルすると，main.indの内容が索引として出力されるようになります．なお，upmendixはUnicodeに対応しており，漢字をそのまま投げても動きはしますが，ソート順がめちゃくちゃになる（文字コード順になる）ので，読みをひらがなで指定してあげる必要があります．

3　ベストプラクティス

ここまでで説明してきた内容に即して，これは回避した方が良い（アンチパターン）や，こうした方が良い（ベストプラクティス）ことをいくつか挙げてみます．

3.1　組版以前

LaTeX云々の前に，文書を作る上でのベストプラクティスもいくつかあると思います．

3.1.1　表記の哲学

ここでいう哲学とは，比喩的な意味での哲学であって，個人の好みによってどちらにすべきか変わるような表記法です．

まず大前提として，どのような表記法を選択するにせよ，**表記法は一つの文書の中で一貫させるべきです**．

理工系の文書について．

- 第2.4.1.1段落で見たように，微分演算子を立体にするか斜体にするか．
- 同様に，定数を立体にするか斜体にするか．
- ベクトルや行列変数を大文字イタリック（ISO），立体太字（NISTなど），立体サンセリフ（日本物理学会）のどれにするか．

日本語特有の事情は以下のようなものがあると思います．

- どの程度までひらくか．
  - 「〜といえます」とするか，「〜と言えます」とするか．
  - 一般には，機能語をひらく方が読みやすいとされています．「例えば」→「たとえば」，「〜する時」→「〜するとき」，「及び」→「および」

- 約物をどうするか.
  - 「」や『』,《》などの使い方を統一するべきです.
- 略語や外来語をどのように表記するか.
- 数値の範囲は「1から10」などとするか,「1〜10」ないしは「1–10」などとするか.
- 句読点は「、」と「。」(日本語の標準),「,」と「。」(以前の公用文),「,」と「.」(欧文・理系のスタイル)などのどれにするか.
- JISの長音符号の扱いに従うか(コンピューター v.s. コンピュータ,など)
- 列挙の区切りは読点「,」を使うか,中黒「・」を使うか[68].
  - さらに,列挙の最後の区切りは区切り文字とするか,「と」や「および」など英語風に書くか.
  - すなわち,英語でA, B, C and D.のようにするのを日本語でもA, B, CとD.のようにするか.
- 強調は太字にするか,ゴシックにするか,下線や斜体を使うか.
  - 一般的な書籍では,ゴシックまたは**ゴシック太字**が強調に使われることが多いと思います.
  - 多書体環境なら,**太字**もできますが,明朝体の細い横線という特徴を潰してしまいます.
  - *斜体*[69]は,日本語にはないスタイルですが,Wordなどで疑似斜体として使えることからか,使われるようになりました[70].
  - 下線は,ゴシックなどが使えない環境での代用です.
  - さらに言えば,圏点も強調の一種といえます.弱い強調や,皮肉を込めた,英語圏のクオートのようなニュアンスが大きいでしょう.

3.1.2 読みやすい文章

組版が美しくても,文章そのものが読みにくければ全く意味がありません.また,日本語として不自然な表現や誤字脱字も可能な限り回避したいところです.詳しくは,「日本語の作文技術」(本多勝一,朝日文庫)や「公用文作成の考え方(文化審議会建議)」などを参照していただくこととして,ここでは自戒も込めて,注意すべき点を上げておきます.

- 読点は意味の区切りとリズムに配慮して丁寧に使う.
- 文章を長くしすぎない.
- なるべく同じ表現を繰り返さない.
- 論文,技術文書,事務文書など,情報を伝えることが主な目的の文書では,できるだけ簡潔に書く.
- 漢字とひらがなをバランスよく使う.
- 係り受けが混乱する文(「頭が赤い魚を食べる猫」のような文)を避け,可能な限り修飾語を近くに置く.
- 引用を除いて,敬体と常体を混在させない.
- 環境を変えて読み,できれば他人に読んでもらって校正する.

3.2 組版において

3.2.1 地の文,図表

ここでは,数式ではない文章や図表を組む上でのベストプラクティスをいくつか挙げてみます.

- テキストをカラフルにしない.
  - ここまでやる人はいないと思いますが…….

68) 後者が正式とされることが多いようですが,見た目によっては読点を使いたくなることもあります.
69) これを実現するのには,fontspecのFakeSlantを使っています.TikZでもできるはずです.
70) そもそも写植などが発展する前はなかったはずです.

- 強調は効果的に使い，文章をデタラメに**強調**しない．
  - フォントの変更をなるべく最終手段として，強調したい部分は文章の書き方のレベルで伝わるよう書くのがよいでしょう．
- 欧文引用符は正しく使う．
  - ``で始め，''で終わる．“hello”
  - ""とすると正しい引用符にならない．”hello”
- 図表のキャプションは簡潔かつ十分に書く．
- サイズ指定は`\linewidth`による相対長で指定する（表4も参照）．
- 表の罫線は入れすぎない．

3.2.2 数式

散々説明してきたことではありますが，他にも以下のような落とし穴があります．

3.2.2.1 予約されている記号は正しく使う． 当然のことながら，予約されている記号ではない書き方をすると見た目がおかしくなります．

実行例 53：予約されている数学記号

```
\(sinx\)ではなく\(\sin x\)．\\
\(f:X \to Y\)ではなく\(f\colon X \to Y\)．\\
\(\{[x]|x \in X\}\)ではなく
\(\{[x]\mid x \in X\}\)．\\
\(||x||\)ではなく\(\|x\|\)．\\
\(P \Longleftrightarrow Q\)ではなく
\(P \iff Q\)．
```

$sinx$ではなく$\sin x$．
$f:X \to Y$ではなく$f\colon X \to Y$．
$\{[x]|x \in X\}$ではなく$\{[x]\mid x \in X\}$．
$||x||$ではなく$\|x\|$．
$P \Longleftrightarrow Q$ではなく$P \iff Q$．

3.2.2.2 書体に関する諸注意 本当に細かいところですが，注意すべき点があります．変数はイタリック，それ以外は原則立体となります．また，太字やチルダの範囲にも注意が必要です．

実行例 54：数式書体の注意

```
\(A_{begin}\)ではなく\(A_{\mathrm{begin}}\)．\\
\(\bm{x}_{\mathrm{A}}\)のように，
添字は太字にしない．\\
\(\tilde{x_{\mathrm{A}}}\)ではなく
\(\tilde{x}_{\mathrm{A}}\)．
```

A_{begin}ではなくA_{begin}．
$\boldsymbol{x}_{\mathrm{A}}$のように，添字は太字にしない．
$\tilde{x_{\mathrm{A}}}$ではなく$\tilde{x}_{\mathrm{A}}$．

A　VS Codeの設定例

B　latexmkの設定例

C　参考図表

表 8: LaTeX 数学記号・演算子一覧

コマンド	出力	コマンド	出力	コマンド	出力
ギリシャ文字（小文字・大文字）					
\alpha†	α	\beta†	β	\gamma†	γ
\delta†	δ	\epsilon†	ϵ	\varepsilon†	ε
\zeta†	ζ	\eta†	η	\theta†	θ
\vartheta†	ϑ	\iota†	ι	\kappa†	κ
\lambda†	λ	\mu†	μ	\nu†	ν
\xi†	ξ	\pi†	π	\varpi†	ϖ
\rho†	ρ	\varrho†	ϱ	\sigma†	σ
\varsigma†	ς	\tau†	τ	\upsilon†	υ
\phi†	ϕ	\varphi†	φ	\chi†	χ
\psi†	ψ	\omega†	ω	\Gamma†	Γ
\Delta†	Δ	\Theta†	Θ	\Lambda†	Λ
\Xi†	Ξ	\Pi†	Π	\Sigma†	Σ
\Upsilon†	Υ	\Phi†	Φ	\Psi†	Ψ
\Omega†	Ω				
二項演算子					
+	$+$	-	$-$	\pm†	$\pm$
\mp†	$\mp$	\times†	$\times$	\div†	$\div$
\cdot†	$\cdot$	\ast†	$\ast$	\star†	$\star$
\circ†	$\circ$	\bullet†	$\bullet$	\setminus†	$\setminus$
\cap†	$\cap$	\cup†	$\cup$	\uplus†	$\uplus$
\sqcap†	$\sqcap$	\sqcup†	$\sqcup$	\vee†	$\vee$
\wedge†	$\wedge$	\oplus†	$\oplus$	\ominus†	$\ominus$
\otimes†	$\otimes$	\oslash†	$\oslash$	\odot†	$\odot$

Continued on next page

表 8: LaTeX 数学記号・演算子一覧 (Continued)

コマンド	出力	コマンド	出力	コマンド	出力
\dagger†	†	\ddagger†	‡	\amalg†	⨿
\lhd	◁	\rhd	▷	\unlhd	⊴
\unrhd	⊵	\dotplus	∔	\ltimes	⋉
\rtimes	⋊	\leftthreetimes	⋋	\rightthreetimes	⋌
関係演算子・集合					
=	=	<	<	>	>
\neq†	≠	\equiv†	≡	\sim†	∼
\simeq†	≃	\approx†	≈	\cong†	≅
\asymp†	≍	\doteq†	≐	\propto†	∝
\le†	≤	\ge†	≥	\ll†	≪
\gg†	≫	\subset†	⊂	\supset†	⊃
\subseteq†	⊆	\supseteq†	⊇	\sqsubseteq†	⊑
\sqsupseteq†	⊒	\in†	∈	\ni†	∋
\notin†	∉	\models†	⊨	\vdash†	⊢
\dashv†	⊣	\mid†	∣	\parallel†	∥
\perp†	⊥	\smile†	⌣	\frown†	⌢
\bowtie†	⋈	\sqsubset	⊏	\sqsupset	⊐
\approxeq	≊	\lessapprox	⪅	\gtrapprox	⪆
\lesssim	≲	\gtrsim	≳	\leqq	≦
\geqq	≧	\lll	⋘	\ggg	⋙
\Subset	⋐	\Supset	⋑	\subseteqq	⊆
\supseteqq	⊇	\therefore	∴	\because	∵
矢印					
\leftarrow†	←	\rightarrow†	→	\leftrightarrow†	↔
\Leftarrow†	⇐	\Rightarrow†	⇒	\Leftrightarrow†	⇔
\mapsto†	↦	\longmapsto†	⟼	\hookleftarrow†	↩
\hookrightarrow†	↪	\leftharpoonup†	↼	\rightharpoonup†	⇀
\leftharpoondown†	↽	\rightharpoondown†	⇁	\rightleftharpoons†	⇌
\uparrow†	↑	\downarrow†	↓	\updownarrow†	↕
\Uparrow†	⇑	\Downarrow†	⇓	\Updownarrow†	⇕
\nearrow†	↗	\searrow†	↘	\swarrow†	↙

Continued on next page

表 8: LaTeX 数学記号・演算子一覧 (Continued)

コマンド	出力	コマンド	出力	コマンド	出力
\nwarrow†	$\nwarrow$	\rightsquigarrow	$\rightsquigarrow$	\leftrightsquigarrow	$\leftrightsquigarrow$
巨大演算子					
\sum†	$\sum$	\prod†	$\prod$	\coprod†	$\coprod$
\int†	$\int$	\oint†	$\oint$	\iint	$\iint$
\iiint	$\iiint$	\iiiint	$\iiiint$	\idotsint	$\int\cdots\int$
\bigcap†	$\bigcap$	\bigcup†	$\bigcup$	\bigsqcup†	$\bigsqcup$
\bigvee†	$\bigvee$	\bigwedge†	$\bigwedge$	\bigodot†	$\bigodot$
\bigotimes†	$\bigotimes$	\bigoplus†	$\bigoplus$	\biguplus†	$\biguplus$
その他の記号					
\infty†	∞	\emptyset†	$\emptyset$	\varnothing	$\varnothing$
\nabla†	∇	\partial†	∂	\forall†	$\forall$
\exists†	$\exists$	\nexists	$\nexists$	\Re†	$\Re$
\Im†	$\Im$	\aleph†	$\aleph$	\hbar†	$\hbar$
\ell†	ℓ	\wp†	$\wp$	\flat†	$\flat$
\natural†	$\natural$	\sharp†	$\sharp$	\angle†	$\angle$
\triangle†	$\triangle$	\square	$\square$	\blacksquare	$\blacksquare$
\surd†	$\surd$	\clubsuit†	$\clubsuit$	\diamondsuit†	$\diamondsuit$
\heartsuit†	$\heartsuit$	\spadesuit†	$\spadesuit$	\dots†	$\dots$
\cdots†	$\cdots$	\vdots†	$\vdots$	\ddots†	$\ddots$

†ダガー (†) が付いているコマンドは標準の LaTeX で追加パッケージなしで利用可能です．無印のものは amsmath,amssymb などのパッケージが必要です．なお，\から始まらないものも，標準の LaTeX で利用可能な記号です．

D　例，図表の一覧

図目次

表目次

コード例一覧

実行例一覧

参考文献・索引

全体として，ドキュメント texdoc をそれぞれ参考にしました．

参考文献一覧

[1] 奥村晴彦，黒木裕介，『LaTeX美文書作成入門』，改訂第9版，技術評論社，2023年12月9日，ISBN: 978-4-297-13889-9

索引