情報工学科の TEX の設定とサンプルファイルについて

佐波孝彦

2025 年 3 月 14 日

1　はじめに

TEX（テフあるいはテックと読む）とは，コンピュータ科学者 Donald E. Knuth [1] が 1978 年に公開を始めた文書整形処理システムである．印刷業界の用語では，**組版**（くみはん）処理システムとも呼ばれ，用意した原稿素材（テキスト・図版・写真等）を各言語のルール [2][3] の下に，指定のレイアウトになるように配置するソフトウェアである．TEX を用いると，OS（Operating System）に依存せずに出力結果の見た目を統一でき，特に数式の仕上がりが綺麗なため[1)]，科学技術の分野では多くの出版物で利用されている．

情報工学科では，BYOD（Bring Your Own Device）の機種として Apple の MacBook Air/Pro を指定しており，課題等の文書作成には TEX を使うことを原則としている．そのため，学科独自の設定を盛り込んだ，TEX の設定スクリプトを用意しているが，本稿は提出物作成の際に使用する標準的な書式のテンプレートを用いたサンプルであり，学科独自の設定の概略を説明するものである．TEX の使い方そのものを説明する文書ではなく，あくまでテンプレート代わりのサンプルとして用意したものである．とはいえ，数多くのコマンドを意図的に使っているため，様々な場面で参考になるはずである．是非，活用して欲しい．なお，添付している本稿のソースファイルでは，高

Version π

Knuth 博士は，1978 年の初版の公開後も TEX の改良や拡張を行ってきたが，1989 年のバージョン 3 の発表時に，これ以上の機能拡張は行わず以降は不具合の修正のみを行っていくことを宣言した．バージョン番号は，3.1415… と更新のたびに円周率に近づけることになっており，Knuth 博士の死没をもってバージョン π として更新が終了することになっている．2025 年 4 月時点でのバージョンは 3.141592653（2021 年 2 月 5 日）である．

1)　組版専用ソフトの代表的なものに Adobe の InDesign があるが，数式の組版は TEX には敵わない．また，Microsoft Word などのワープロソフトでは，まともな組版処理はできない．同じレイアウトにするのに，InDesign や TEX に比べて遥かに面倒なだけでなく，実現できない処理も多いからである．例えば，Unicode 対応フォントには，一つのコードポイントで，旧字体やルビ字体など複数のデザインが含まれている文字をもつものがあり（名前に ProN や Pr6N がつくフォントが相当），それらの字体は Glyph ID（GID）や Character ID（CID）で区別するが，Word は GID/CID を扱えない．また，商業印刷で使われるプロダクションプリンタは色を CMYK（Cyan, Magenta, Yellow, and Key）で出力するが，Word は RGB（Red, Green, and Blue）しか扱えないため色味が変わってしまうなど，制約が多すぎる．

度な設定を必要とするものを省略しているため，コンパイルしてもこの PDF（Portable Document Format）[4] ファイルと同一の見た目にはならないことを予めお断りしておく．

TEX は，HTML（HyperText Markup Language）のようなマークアップ言語の一種である．したがって，そのソースファイル（拡張子は `.tex`）は，文章そのものと文章の構造や見た目を指定するコマンドから成るテキストファイルである．複雑な数式や記号もテキストで入力する[2)]．例えば，ギリシャ文字の π は，「ぱい」を変換して π とする（和文フォントが使われてしまう）のではなく，`\pi` と入力する．単なるテキストファイルであるため OS に依存せず作成・編集でき，コンパイルすることによりファイル中のコマンドに基づいて文書が組版される．組版結果は DVI（**dev**ice-**i**ndependent）形式のファイル（拡張子は `.dvi`）に書き出される．DVI ファイルは，表示デバイスやプリンタなどの装置に依存しない中間形式のバイナリデータであり，DVI ドライバと呼ばれる別のソフトウェア（DVI ウェアとも言う）で組版結果をプレビューしたり，印刷可能な PostScript[3)] [5] ファイルに変換したりして利用する．また，近年では DVI ファイルを PDF[4)] に変換して，PDF ファイルを最終出力とするのが一般的である．

TEX はオープンソースソフトウェア[5)] であるため，組版処理を行うエンジンには，いくつもの派生系が存在している．中でも，複雑になりがちな各種の設定をマクロファイル（クラスファイルとパッケージファイルがある）を読み込むことで簡易に行える LATEX [6] が Leslie B. Lamport によって開発されて以降は，TEX と言えば LATEX を指していることが普通である．ただし，LATEX にも多くの派生エンジンが存在し，日本では縦書きや禁則処理などの日本語固有の処理を扱えるようにした pLATEX[6)] [7]，さらに近年の OS で主流となった Unicode 対応フォント（OpenType フォント）[8][9] を扱えるようにした upLATEX [10] が長いこと主流である．世界的には，Unicode 対応フォントを柔軟に扱え，かつ組版結果を直接 PDF に出力できる LuaLATEX [11] が主流になりつつあり，日本語を扱える LuaTEX-ja [12] も日々進化している．ただし，upLATEX と LuaTEX-ja はコマンド体系に違いがあるため，LuaTEX-ja では upLATEX のソースコードをそのままコンパイルできない．将来的には日本でも LuaLATEX が主流になることが予想されているが，現状では出版社や学術団体が用意しているマクロファイルの多くが (u)pLATEX 2_ε に基づいているため，本稿でも **upLATEX の使用を前提**として説明をしていく．独学で学ぶ意欲のある方は，最初から LuaTEX-ja を使っていくのも良いであろう（インストールはされている）．ただし，学科として設定を統一したり，テンプレートを配布するのは，仕上がりの文書の体裁を統一するためでもあるので，そのことには注意を払うべきである．

---

2) テキストによる数式の入力方式は多くのソフトウェア（Microsoft Word/Excel/PowerPoint, Apple Pages/Numbers/Keynote など）に取り入れられており，事実上のスタンダードになっている．

3) Adobe 社が開発したページ記述言語であり，文字や画像をベクトルデータ（テキスト）で記述するため，高解像度の印刷が可能である．

4) Adobe 社が開発した電子文書のファイル形式で，PostScript をベースにしているため，拡大・縮小しても画像や文字が粗くならない．ただし，写真はピクセル単位で表現されるラスタ形式のため，拡大すると粗くなる．

5) ソースコードが公開されており，無償で使用でき，誰でも自由に修正，改変，再配布が可能である．

6) 出版社の株式会社アスキー（現在は株式会社角川アスキー総合研究所）が自社の出版物の組版をするために日本語化した．pLATEX の p は publishing の意味である．

TEX, LATEX, LATEX 2ε

TEX はソースコードが公開されており，誰でも改良を加えることができる．また，オリジナルの TEX と区別できる名前を付けさえすればその改良版の配布も許されているため，数多くの派生エンジンが存在している．LATEX の初期のバージョンは LATEX 2.09 と呼ばれていたが，この派生系はそれぞれ独自に拡張されていったため互換性がなく，ソースファイルを見ても，どの派生 LATEX エンジンでコンパイル可能なのかの判別が難しかった．そこで，それらの拡張をすべて網羅するようコマンド体系を整理した LATEX 2ε が開発された．現在の派生エンジンの多くは LATEX 2ε が元になっているが，再び互換性の問題が起きつつある．また，次期バージョンとして LATEX 3 のプロジェクトが進行中である．

2　フォントについて

前節冒頭で「TEX を使うと OS に依存せずに出力結果の見た目を統一できる」と書いたが，それには前提条件があり，出力するコンピュータに同じフォントがインストールされている場合に限定される．予定されたフォントがインストールされておらず別のフォントが代用されると，文字幅等が変わるのでページレイアウト自体は同一でも文章の行数が変わったり，見た目の印象も変わったりする．情報工学科が推奨する MacTEX[13] のインストーラを用いると，オープンソースの無償フォントが数多くインストールされる．デフォルトでは，欧文に Knuth 博士がデザインした Computer Modern（CM）フォント [16]，和文に原ノ味フォント [17] が使用される．

フォントは好みもあるが，一般に有償の和文フォントは，様々な状況において見た目のバランスに優れ，出力結果の品質が高いのに対し，無償の和文フォントは品質のばらつきが大きい（和文には数万文字を超える漢字があるため，文字数の少ない欧文フォントに比べて，デザイン調整に膨大な手間が必要となるからである）．macOS の日本語環境で使用されるヒラギノフォントは，癖のない高品質な有償フォントであるため，多くの出版物に採用[7)]されている．そこで，情報工学科の用意した TEX の設定スクリプトでは**和文フォントにヒラギノを使用**する設定にしている．

ところで，現在のレーザープリンタは PostScript（PS）対応あるいは PS 互換であることが一般的である．PS プリンタは，フォントもベクトルデータとして処理する[8)]ため，拡大・縮小印刷をして

MacTEX

世界的に最も普及している TEX のディストリビューションは TEX Live であり，多くの OS プラットフォームで利用できる．MacTEX は，macOS に特化した TEX Live のインストーラである．一方，TEX Live は，Homebrew [14] や MacPorts [15] といったパッケージ管理ツールを用いてインストールすることも可能である．しかし，両者はインストール先のディレクトリが異なるため，混在させるとファイルの依存関係が崩れ，正しく動かない実行ファイルがでてきたりする．したがって，MacTEX でインストールを行ったのであれば，以降も TEX の更新は MacTEX を用いるか，移行するのであれば，PATH の設定を変更する必要がある．

7)　高速道路の標識にもヒラギノ角ゴシックが採用されている．
8)　インクジェットプリンタでは，インク滴が 1 つの点（ドット）となり，その点を連続して吹きつけて面を構成することで，文字やグラフィックを表現している．

も綺麗に印刷される．また，よく使われるフォントはプリンタに内蔵しておくと印刷が速い（そうでない場合は，フォントデータを都度プリンタに送信する必要がある）．初期の PostScript Level 1（PS1）プリンタ[9)]では，欧文フォントにセリフ体の Times，サンセリフ体の Helvetica，タイプライタ体の Courier New のそれぞれでローマン体，イタリック体，ボールド体，ボールドイタリック体，加えて記号用の Symbol および Zapf Dingbats の基本 14 書体（Base 14 fonts）が搭載されていた[10)]．これらの基本 14 書体は Acrobat Reader にも内蔵されている（フォントが埋め込まれていない PDF で代用フォントとして用いられる[11)]）．

一方，日本語に初対応した PostScript Level 2（PS2）プリンタでは，欧文のセリフ体とサンセリフ体に相当する明朝体とゴシック体（モリサワのリュウミン L-KL と中ゴシック BBB）の基本 2 書体が搭載された．その後，明朝体とゴシック体が 2 ウエイト（太ミン A101 と太ゴ B101）が追加になり，さらに丸ゴシック体（じゅん 101）も搭載され，基本 5 書体と呼ばれた．これらのフォントは当初パッケージとしては販売されておらず，プリンタとフォントを一体購入し，パソコンの画面上では解像度の粗いスクリーンフォント[12)]を使用し，印刷すると綺麗な仕上がりになるという使用法[13)]であった．このような背景から，日本語を扱う (u)pLaTeX でも標準では丸ゴシック体を除く 4 書体が使用されてきた（印刷時はプリンタ内蔵のフォントが使われた）．現在主流の PostScript 3 プリンタに搭載される和文フォントは，平成明朝 W3 と平成角ゴシック W5 であることが多い．しかし，近年のパソコンは処理能力が飛躍的に向上し，画面上でもアウトラインフォントを直接描画できるようになったため，プリンタの搭載フォントによらず任意のフォントを使用して書類を作成し，印刷することが可能である．ただし，プリンタに搭載されていないフォントを使用すると印刷にかかる時間は増える．

ウエイト

フォントの太さはウエイト（Weight）で表される．フォントベンダーにより太さの基準や呼び方に違いはあるが最大で 10 段階程度に分類される．ISO では 9 段階に分類されており，それぞれ，W1: Ultra Light（極細，Thin），W2: Extra Light（特細，Ultra Light），W3: Light（細），W4: Semi Light（中細，Regular/Normal），W5: Medium（中），W6: Semi Bold（中太，Demi Bold），W7: Bold（太），W8: Extra Bold（特太，Ultra Bold），W9: Ultra Bold（極太，Black/Heavy）と呼ばれる．カッコ内は日本語や ISO 以外での慣例的な呼び方である．ただし，実際にはこの表記が当てはまらないフォント製品も数多くある．

9） 世界（そして日本）初の PS プリンタは Apple の LaserWriter である．

10） PostScript Level 2（PS2）プリンタでは，基本 14 書体に加え，セリフ体として Palatino，Bookman，New Century Schoolbook，サンセリフ体として Helvetica Narrow，Avant Garde それぞれのローマン体，イタリック体，ボールド体，ボールドイタリック体と，筆記体の Zapf Chancery を加えた基本 35 書体（Base 35 fonts）が搭載された．現在の主流は，PostScript 3 プリンタ（Level をつけない）であり 136 書体の欧文フォントが搭載されている．

11） 最近の Acrobat Reader では，Helvetica と Times が，それぞれ Arial と Times New Roman に置き換えられている．

12） ビットマップフォントと呼ばれ，各文字をピクセルのオン・オフで表現する．一方，文字の輪郭を曲線（ベクトル）で表現する PS フォントは，アウトラインフォントと呼ばれる．

13） パソコンの処理速度が遅く，アウトラインフォントを画面に直接描画できない事情もあった．

表 1　よく使われるドキュメントクラス

用途	欧文（標準）	和文（横書き）	和文（縦書き）	和文（新・縦横）	和文（新・縦横）
論文・レポート	article	jarticle	tarticle	jsarticle	jlreq
長い報告書	report	jreport	treport	jsreport	（オプション指定）
本	book	jbook	tbook	jsbook	（オプション指定）

3　本稿のソースファイルの中身

それでは，具体的に本稿のソースファイル，`sample.tex` の中身を見ながら，どのように記述していくのかを見ていく．LaTeX のソースコードは「ドキュメントクラス」「プリアンブル」「ドキュメント環境」の 3 つから成る．① 初めにドキュメントクラスを宣言し，② 次にプリアンブルで文書に必要なパッケージを読み込み，③ ドキュメント環境の中で文章を記述していく．LaTeX を用いた文書作成では，どの派生エンジンを用いてもこれが基本となる．

3.1　ドキュメントクラス

ドキュメントクラスとは，LaTeX で作成する「文書の種類」を意味し，コンパイルするソースファイル（この場合，`sample.tex`）において**必ず最初に指定**する必要がある．それには以下の構文で記述する．

```
\documentclass[オプション]{ドキュメントクラス}
```

代表的なドキュメントクラスを表 1 に示す．

ドキュメントクラスは，文書の用途に応じた標準書式を定義するものであり，文書のレイアウトや見出しなどの様式を作成者によらず共通にすることができる．2 列目と 3 列目のクラス（jclasses）は，pLaTeX の初期の頃に作られた古いパッケージであり，日本語組版の満たすべき標準的な要件を満たせないことが多い．そのため，現在では 4 列目の jsclasses パッケージ [18] を使うのが一般的になったが，日本語組版処理の要件（Requirements for Japanese Text Layout）[19] を満たすことを目指した jlreq [20] が登場して以降は，そちらにトレンドが移りつつある[14)]．他にも jsclasses パッケージにおける (u)pLaTeX 依存の部分を無くし，LuaTeX 等でも使えるようにした BXjscls パッケージなどもあるが，**授業の課題等で日本語のレポートを書く場合は，`jlreq` を使用**するのがよい．ドキュメントクラスごとに指定するオプションでは，用紙サイズやフォントサイズなどを指定するが，詳しくは jlreq のドキュメントを参照のこと．

14)　jsclasses は，標準のフォントサイズ（10pt）で使用する場合はよいが，12pt などを用いると長さの扱いが面倒になる．例えば，`\hspace{10cm}`とすると 12cm のスペースが空く．これは，縮小した用紙サイズで 10pt のフォントサイズで組版してから最後に用紙全体を 1.2 倍に拡大してフォントサイズが 12pt になるようにするという方式を採用しているためである．

ドキュメントクラスの設定では，指定していないオプションはデフォルトの値が使われる．

TEX は開発速度が速いので，インターネットや書籍などで情報を探したときに，jarticle などを指定しているものは，情報が古いだけでなく，現在では好ましくない方法の場合もあるので参考にすべきではない．なるべく最新のドキュメントを確認すべきである．

なお，jlreq では，使用する LATEX エンジンによって処理内容が異なる項目があるため，クラスオプションで処理エンジンに upLATEX を使用することを明示しておく方がよい（自動判別の機能もあるが，指定しておくことが推奨されている）．また，文字色をつける xcolor や図版や写真を埋め込む graphicx などの特定のパッケージを使用する場合，DVI ドライバを指定しないと期待通りの出力にならないことがあるが，ドキュメントクラスのオプションはグローバルオプションとも呼ばれ，その内容が必要に応じて他のパッケージにも伝達される．したがって，documentclass のオプションで DVI ドライバを指定しておけば xcolor や graphicx などのパッケージを使う場合に，いちいち同じ指定を繰り返す必要がなくなる．そのため，ここで DVI ファイルを PDF に変換する DVI ドライバとして dvipdfmx を指定しておく．標準的には以下の通りとなり，本稿でも同じ指定にしている．

```
\documentclass[uplatex,dvipdfmx]{jlreq}
```

3.2 プリアンブル

3.2.1 パッケージの読み込み

プリアンブルでは，実際の文書の内容を書き始める前に，本文中で使うコマンドやフォントの取り扱いなどが定義されたパッケージを指定したり，文書のタイトル情報などを入力する．パッケージに含まれない独自の定義などもここで行う．プリアンブルにおけるパッケージの読み込みは以下の構文で行う．

```
\usepackage[オプション]{パッケージ}
```

パッケージは膨大な種類が開発されている（インストールされた「TeX Live ユーティリティ」のパッケージのタブでその種類を確認できる）ので，その全てを把握することは不可能に近いが，ここでは，本サンプルで採用したものを説明する．本稿では以下のパッケージを読み込んでいる．

本稿で採用したパッケージ

```
\usepackage{jlreq-deluxe}
\usepackage[noalphabet,hiragino-highsierra-pron]{pxchfon}
\usepackage[T1,LGR]{fontenc}
\usepackage{stix2}
\usepackage{roboto}
\usepackage[scale=0.95]{roboto-mono}
\usepackage[fleqn,tbtags]{mathtools}     % 数式関連 (w/ amsmath)
\usepackage[dvipdfmx]{graphicx}
\usepackage[dvipdfmx]{xcolor}
\usepackage{pxrubrica}                   % ルビの機能
\usepackage{ascmac}                      % screen/itembox/shadebox環境
\usepackage{tcolorbox}                   % 様々な枠環境
\tcbuselibrary{listings,skins,breakable} % tcolorboxのオプション
\usepackage{listings,plistings}          % ソースコードの表示機能
\lstset{                                 % listingsのオプション
  aboveskip=0pt,
  belowskip=0pt,
  language={[LaTeX]TeX},
  morekeywords={% パッケージにコマンドや関数と認識されないものを列挙しておく.
                % 本例では和文用のコマンドなどを登録している.
  appendix, maketitle, textmc, textgt, textmg, mcfamily,
  gtfamily, mgfamily, ebseries},
  backgroundcolor={},%
  basicstyle={\small\ttfamily},%
  keywordstyle=\color{blue}\bfseries,
  commentstyle=\color{red},
  breaklines=true}
\usepackage[dvipdfmx]{hyperref}
\hypersetup{%
  bookmarksnumbered=true,%
  colorlinks=true,%
  linkcolor=magenta,
  citecolor=blue,
  urlcolor=blue}
\usepackage[dvipdfmx]{pxjahyper}     % (u)pLaTeXのときは必要
\usepackage{hira-stix} % ヒラギノフォントの代替フォント定義（Warning回避）
```

- **jlreq-deluxe パッケージ**は，(u)pLaTeX で jlreq クラス使用時に多書体（多ウエイト）の和文フォントを利用可能にするパッケージである [21]．予め明朝体で 2 ウエイト（細明朝（W3）と**中太明朝（W6）**，ゴシック体で 2 ウエイト（細ゴシック（W3），**中太ゴシック（W6）**，さらに中細丸ゴシック（W4[15)]）の計 5 書体が設定される．実はこの機能は，日本語文書作成時に

15) アルファベットや数字は欧文フォントが使用されるが，丸ゴシック体に相当する欧文フォントの設定がない．そのため，この W4 はセリフ体のフォントで表示されている．詳しくは次節参照．

(u)pLaTeX で OpenType フォント（Unicode 対応フォント）を扱えるようにする OTF パッケージ [22] のオプションで deluxe を指定したときと同じである．しかし，jlreq クラス使用時に OTF パッケージを利用すると，jlreq クラスの意図する組版結果が得られなくなってしまう．そのため OTF パッケージとの互換性を保ちながら開発された．オプションは，OTF パッケージと同じであるが，upLaTeX で使用するなら特に指定する必要はない．

- **pxchfon パッケージ**は，pLaTeX/upLaTeX + dvipdfmx の環境で，日本語文書で使える 7 書体（明朝体 3 ウエイト，ゴシック体 3 ウエイト，丸ゴシック体 1 ウエイト）をユーザ指定のフォントに設定するパッケージである [23]．ただし，MacOS に内蔵されているヒラギノフォントは明朝が 2 ウエイト（W3 と W6），ゴシックが 10 ウエイト（W0~W9），丸ゴシックが 1 ウエイト（W4）であるため，内蔵ヒラギノフォントを使う場合に設定できるのは 6 書体（細明朝（W3），**中太明朝（W6）**，細ゴシック（W3），**中太ゴシック（W6）**，**特太ゴシック（W8）**および**中細丸ゴシック（W4）**）のみとなる．オプションに `hiragino-highsierra-pron` を指定すると，ヒラギノ 6 書体が使える設定になる．なお，多書体を使うには，**先に jlreq-deluxe パッケージを読み込んでおく必要がある**．論文やレポートを書くときに使用する和文フォントはせいぜい 2~3 書体程度であるので，通常は jlreq-deluxe パッケージを使用するだけでもよい．

  また，オプションには，`alphabet`，`noalphabet`，`relfont`（この 3 つは排他的で同時に使用できない）のいずれかを記述する．`alphabet` や `relfont` は，英数字部分も指定の和文フォントの英数字を使って表示する設定である．ただし，半角等幅の字体しか表示できない制約があり，見た目のバランスが良くない．したがって，通常は `noalphabet`（英数字に和文フォントを使用しない設定）にするのが一般的である．そうすることで，英数字部分はプロポーショナルな欧文フォントが使われ，数式ともバランスが取れる．
- **fontenc パッケージ**は，フォントのエンコーディングを指定するパッケージであり [24]，各種欧文フォントパッケージを使用するときに，オプションでエンコーディング方式を指定する．このパッケージで最も一般的なオプションである T1 は，欧州言語の文字のエンコーディングで，ほとんどのアクセント文字が定義される．他に，T1 の拡張エンコーディングである TS1，Knuth 博士が作った TeX のオリジナルの 7 ビットエンコーディングである OT1，ギリシャ文字のエンコーディングである LGR，日本語の横書きと縦書き文字用の JY1 と JT1（upLaTeX を使う時は JY2/JT2，LuaTeX-ja の場合は JY3/JT3 にする），TeX の数式用フォントを販売していた Y&Y 社が定めた LY1 などがあり[16)]，使用するフォントに応じて変更あるいは追加する．

16) TeX のエンコーディングは命名規則があり，以下の文字で始まるエンコーディング名は予約されている．T（標準 256 文字のテキスト），TS（対応する T エンコーディングを拡張するように設計されたシンボルエンコーディング），X（T エンコーディングの厳密な要件に準拠しないテキストエンコーディング），M（標準 256 文字の数式用エンコーディング），S（その他のシンボルエンコーディング），A（その他の特殊なアプリケーション），OT（標準 128 文字のテキストエンコーディング），および OM（標準 128 文字の数式用エンコーディング）．サイトまたはシステムにローカルなエンコーディング方式は L で始まり，広く配布することを目的とした実験的なエンコーディングは E で始まる．

初期の欧文 TEX では符号空間が 7 ビット，すなわち 1 つのフォントで表せる文字は 128 種類しかなかった．そのため，初期の CM フォントは 7 ビットフォントであったが，その後，欧文 LATEX では 8 ビットフォントが標準になり，現在では 100 万を超える欧文以外の文字も表せる Unicode 対応フォントを扱える LATEX が普及している．ここでのエンコーディングとは，使用するフォントがどのような文字セットをもつフォントなのかを TEX に伝えるもので，UTF-8 のようなソースファイルの文字コードとは異なる．

- **stix2 パッケージ**は，欧文のセリフ体や数式に STIX（**S**cientific and **T**echnical **I**nformation e**X**change）Two フォントを使用するためのパッケージである [25]．STIX Two フォントは，アメリカ数学会（AMS: American Mathematical Society）や米国電気電子学会（IEEE: Institute of Electrical and Electronics Engineers）を含む多くの学協会の出版物に標準で使われるセリフ体フォントであり，Google フォントにも採用されている．一般に，数式は文字だけでなく特殊な記号等も必要になるため，通常のテキスト用フォントのみでは表現できないことがある．そこで，通常は数式部分に別のフォントを使用するが，その場合，本文のフォントと異なる字形になってしまう．STIX Two フォントには，STIX Two Text と STIX Two Math の 2 つがあり，数式用にデザインされたフォントが含まれるため，本文のフォントとの整合性は完璧である．また，AMS の配布する数式記述用パッケージ（amsmath）にも対応しているため，それらのコマンドも問題なく使用できるのは大きな利点である．

  ところで，STIX Two Text フォントは MacOS にも TEX のディレクトリの中にも 4 ウエイト分がインストールされているが，TEX で使えるようにするメトリックファイルは 2 ウエイト分しか用意されていない．ヒラギノの明朝体も 2 ウエイトのみなのでバランスは取れているが，TEX で使えるのは，ローマン体，イタリック体，スモールキャピタル体がそれぞれ 2 ウエイト[17]のみである．また，サンセリフ体やタイプライタ体がない（STIX Two Math にはどちらもあるが，本文中で使うとバランスが悪い）ため，それらには別のフォントを指定する必要がある．
- **roboto パッケージ**は，欧文のサンセリフ体に Roboto フォントを使用するためのパッケージである [26]．Roboto フォントは，Google フォントにも採用されており，ヒラギノ角ゴシックともバランスが良いと思う．オプションは指定しなくとも，サンセリフ体のみに Roboto が適用される．また，ウエイトも 6 種類使える[18]ので，パッケージで定義されているコマンド（`\robotoThin{}`，`\robotoLight{}`，`\robotoRegular{}`，`\robotoMedium{}`，`\robotoBold{}`，`\robotoBlack{}`）により個別にウエイトを指定することも可能である．本稿では，pxchfon パッケージの説明で**特太ゴシック（W8）**の W8 部分で個別に指定している．
- **roboto-mono パッケージ**は，欧文のタイプライタ体に等幅の Roboto Mono フォントを使用するためのパッケージである [26]．オプションは特に指定しなくてよいが，デフォルトのままだ

17） レギュラーとボールドが使用されるが，それぞれヒラギノ明朝の W3 と W6 より若干太いのは気になる．
18） STIX Two と同様，デフォルトのレギュラーとボールドはヒラギノ角ゴシックの W3 と W6 より若干太いので，気になる場合はデフォルトの設定ウエイトを 1 つ下げればよい．本稿では個別に指定した W8 の部分以外はそのままにしてある．

と日本語フォント部分に比べてサイズが若干大きく感じるので，ここでは [scale=0.95] に調整している．

- **mathtools パッケージ**は，数式を表現するための機能を提供するパッケージである [27]．従来より LaTeX の数式用のパッケージとして，AMS の開発する amsmath パッケージが使われてきたが，mathtools は amsmath の不具合を修正した拡張パッケージである．mathtools を読み込むと amsmath パッケージも自動的に読み込まれるので，個別に amsmath を読み込む必要はない．オプションには，amsmath のオプションを書くことも可能である．ここでは，数式の配置を文章幅の左右中央ではなく，左から一定の字下げの後で行う `fleqn` と，複数行に折り返す式を書く時の式番号が，番号を右側に配置するときは一番下，番号を左側に配置するときは一番上に配置される `tbtags` を指定している．mathtools や amsmath のオプションは多数あるので，詳しくはドキュメントを参照のこと．
- **graphicx パッケージ**は，文書に画像を挿入したり，テキストや図の拡大・縮小・回転を行うためのパッケージである [28]．図は EPS（Encapsulated PostScript）あるいは PDF で用意し，写真であれば PNG（Portable Network Graphics）あるいは JPG（Joint Photographic Experts Group）で用意しておく．TeX そのものには画像を処理する機能はなく，画像を配置する領域を確保するだけであり，実際の処理は DVI ウェアが行う．したがって，オプションには，DVI ドライバの記述が必要である．また，**写真は解像度を適切に下げて**おかないと最終的にできあがる PDF ファイルのサイズがサーバ等のアップロードサイズの上限を超えることがあるので注意する．
- **xcolor パッケージ**は，文字に色をつけるために使用するパッケージである [29]．オプションには，DVI ドライバが必要である．
- **pxrubrica パッケージ**は，日本語組版においてルビや圏点をつけるために使用するパッケージである [30]．オプションは存在しない．
- **ascmac パッケージ**は．図や罫線で囲んだボックスを出力する命令などを提供するパッケージである [31]．詳細はドキュメントを参照のこと．
- **tcolorbox パッケージ**は，文章などを枠で囲む機能を提供するパッケージである [32]．オプションが豊富にあるので，他のパッケージのように [ ] の中に羅列すると非常に長くなることがある．その場合，別のコマンド `\tcbuselibrary` を使って別に列挙する方が整理して記述できる．本稿では TeX のソースを表示するときの枠に使用している．詳細はドキュメントを参照のこと．色などが関係するので **graphicx や xcolor パッケージの後に読み込む**必要がある．以降のパッケージも同様である．
- **listings パッケージ**は，文章中でプログラム言語のソースコードなどをリストする機能を提供するパッケージである [33]．指定した言語の関数部分を色付けするなどの機能ももつ．本稿では，TeX のソース表示に使用している．ただし，ソースの中に和文が含まれると正しく表示されないため，和文対応するための拡張パッケージ（jlisting，jvlisting，plistings などがある）も読み込む．オプションが豊富にあるので，別のコマンド `\lstset` で列挙し，ここでは TeX のコマンド定義の追加（和文用のコマンドが定義されていないため）や色の指定などをしている．詳細はドキュメントを参照のこと．

- **hyperref パッケージ**は，出力の PDF ファイルに，HTML と同様なハイパーリンク機能を加えるためのパッケージである [34]．PDF のしおり機能で目次をつけたり，URL へのリンクを指定したりすることができる．本稿では，目次を自動作成する機能と，リンクに色をつける設定を採用している．オプションが豊富にあるので，`\hypersetup` を使って列挙する．詳細はドキュメントを参照のこと．
- **pxjahyper パッケージ**は，(u)pLaTeX + hyperref + dvipdfmx の組み合わせで日本語を含む PDF を作成する際，しおりの文字化けなどを解消するために使用するパッケージである [35]．オプションも色々設定できるが，高度なことをしないなら読み込んでおくだけでよい．**hyperref パッケージの後に読み込む**こと．
- **hira-stix パッケージ**は，ソースファイルのコンパイル時にフォント関連のワーニングが多数表示されるのを回避するために，予め代替フォントを定義した独自パッケージである．本稿で使用しているフォント（ヒラギノ，STIX Two，Roboto）は書体やウエイトに限りがあるため，ソースファイル上でこれらのフォントに含まれない書体の指定が現れると，指定のフォントがないため別のフォントで代用する旨を表示するワーニングメッセージが沢山表示される．ワーニングメッセージが沢山あると，その他の部分にも問題があった場合に，問題箇所の識別が大変になるため，フォントに起因するワーニングメッセージの出現を回避するために簡易的に作成した．オプションはないので，ただ読み込むだけで良い．

以上が，本稿で使用しているパッケージの概要であるが，LaTeX のパッケージは，まだまだ沢山ある．種類や使用法については各自で調べて欲しいが，慣れてくるとパッケージで読み込むスタイルファイルの中身を見ることで理解できることも多い．パッケージファイルは `.sty` の拡張子をもつのが普通で，ターミナル上で簡単に検索できる．例えば，pxchfon パッケージであれば，実体は pxchfon.sty なので，その場所は，

```
>> kpsewhich pxchfon.sty
/usr/local/texlive/2024/texmf-dist/tex/platex/pxchfon/pxchfon.sty
```

のように検索できる（kpsewhich は TeX によってインストールされるコマンドである）．ファイルの部分を除いて，

```
>> open /usr/local/texlive/2024/texmf-dist/tex/platex/pxchfon/
```

とすればフォルダとして開けるので，テキストエディタなどでファイルの中身を見られる．あるいは，ターミナル上で当該ディレクトリに移り，more や less コマンド等で見てもよい．

3.2.2 独自の設定

プリアンブルにはパッケージの読み込みだけでは設定しきれない，その他の設定を記述することもある．ここでは，pxchfon パッケージで使用する 7 書体（実際には重複があるので 6 種類

であるが）を自分で記述した場合の例を掲載している．ここで，MacOS に内蔵のヒラギノフォントは，ゴシック体がウエイトごとに 1 つのファイルになっているのに対し，明朝体は 1 つのファイルに複数ウエイトを内包している．そのため，ウエイトに対応したフォントの番号の指定が必要となる．ただし，これらの設定は自分で書かずとも `pxchfon-extras.def` [36] というファイル[19]を TEX のディレクトリの中に置いておけば，pxchfon パッケージのオプションで，`[hiragino-highsierra-pron]` と指定するだけで同じ設定が自動で読み込まれる．情報工学科の設定スクリプトでは，`pxchfon-extras.def` がディレクトリにコピーされる．

```
\setlightminchofont[0]{HiraginoSerif.ttc}       % \mcfamily\ltseries W3を指定
\setmediumminchofont[0]{HiraginoSerif.ttc}      % \mcfamily\mdseries W3を指定
\setboldminchofont[2]{HiraginoSerif.ttc}        % \mcfamily\bfseries W6を指定
\setmediumgothicfont{HiraginoSans-W3.ttc}       % \gtfamily\mdseries
\setboldgothicfont{HiraginoSans-W6.ttc}         % \gtfamily\bfseries
\setxboldgothicfont{HiraginoSans-W8.ttc}        % \gtfamily\ebseries
\setmarugothicfont{HiraginoSansR-W4.ttc}        % \mgfamily
```

なお，2 節にも書いたが，日本語対応の PS プリンタは，当初 2 書体しか使用できなかったため，文章中で文字を強調するときなどは書体を変える以外の選択肢がなかった．5 書体になり太字が使えるようになっても，明朝体での**強調**はゴシック体での**強調**に比べ，インパクトに欠けるため，現在でも日本語の文章では強調するときに，ゴシック体を用いることが一般的である．因みに，欧文の文章での強調はイタリック体にするのが一般的である．したがって，文字を強調するコマンド`\emph{}`を使うと和文は細ゴシック体，欧文はイタリック体になる．

ただし，日本語の技術文章では，英数字と日本語が混在する用語や文章を書くことが多いため，`\emph{}`を使うことはほとんどない．また，`\textgt` コマンドで強調するのも，ゴシック体とセリフ体が混ざってしまいバランスが悪い．例えば，pxchfon パッケージを強調したいとき，**pxchfon パッケージ**とするよりも **pxchfon パッケージ**とする方が統一感があるし，統計学における **68-95-99.7 則**は，**68-95-99.7 則**とする方がバランスが良い．これは，節見出しなどの節番号と見出しについても同じことが言える．このことは気に留めておくべきである[20]．

TLContrib

TEX は書籍の付録として DVD や USB メモリなどに収録して配布されることもあり，利用や改変，商用再配布にライセンス上制約がないものだけがディストリビューションに含められている．ところが，パッケージの一部のファイルには「商用・非商用を問わず無料で利用可能だが，商用再配布を禁じる」など，ライセンス上含められないものがある．そのようなファイルや拡張パッケージを配布する目的で TLContrib（Contributed packages for TeX Live）というディストリビューション [37] が存在する．情報工学科の設定スクリプトでも TLContrib から一部のファイルをダウンロードしている．

19） `pxchfon-extras.def` は，ディストリビューションに含まれないため，別にダウンロードする必要がある．

20） 初期の pLATEX の時代には，ゴシック体のコマンドで，英数字も欧文フォントのサンセリフ体に切り替える設定が良く使われたが，現在は今回採用しているパッケージのコマンドで簡単にできる．

3.2.3 独自のコマンド

プリアンブルには，独自のコマンドを定義しておくこともできる．新しいコマンドは以下の構文で作成できる．

```
\newcommand{\関数名}[引数の数]{関数の定義}
```

引数は#番号を用いて，定義の中で記述する．

例えば，書体の指定などは組み合わせると長くなりがちなので，ここでは例として以下の 2 つのコマンドを定義し，本稿でも使用している．

```
\newcommand{\mcbf}[1]{{\mcfamily\bfseries #1}} %\mcbfという命令型コマンドを定義
\newcommand{\gtbf}[1]{{\sffamily\bfseries #1}} %\gtbfという命令型コマンドを定義
```

これらはそれぞれ，中太字明朝体，中太ゴシック体（英数字は対応する欧文フォントのボールド体を使用）を指定するコマンドになっている．また，既存のコマンドの定義を変更するには，\renewcommand を使う．

3.2.4 タイトルの出力

作成する文書にタイトルを書く場合，入力はプリアンブル内で，

```
\title{タイトルの中身}
```

と書き，\begin{document}以降に\maketitle と記述すれば，「タイトルの中身」がタイトルとして出力される．同様に

```
\title{タイトルの中身}
\author{著者名}
\date{日付}
```

と記述すると，「タイトルの中身」の下に「著者名」と「日付」が出力される．日付を出力したくない場合は，引数を空にする（\date{}）．また，引数に\today と入力するとコンパイルした日付が出力される．

3.3 ドキュメント環境

プリアンブル部分が書き終わると，あとは本文を書いていくことになる．\begin{document}と\end{document}の間で文章を記述する．

```
\begin{document}
\maketitle      % タイトルを入力している場合

% 以降で本文を書いてく.
%\section{はじめに}
\input{intro}   % 別のファイルに記述して，ファイルを読み込むことも可能.
\input{sect2}
\input{sect3}
\input{sect4}
\input{biblio}
\appendix       % 付録をつけるとき
\input{apdx}
\end{document}
```

なお，(u)pLaTeX では外部のファイルの読み込みも可能である．例えば，節ごとの文章を別のファイル（拡張子は.tex）に作成して，ソースファイル中でそれらを読み込むことができる．また，タイトルページを別にするよう指示があった場合などは，\maketitle の代わりに

```
\begin{titlepage}
\vspace*{3cm}
\centering
\Huge\textsf{Title of Report}\\[2cm] % 改行の後，2cm空ける
\Large\textsf{情報工学科1年}\\
\Large\textsf{25G1000}\\[5pt]        % 改行の後，5pt空ける
\huge\textsf{工大 太郎}\\[1cm]       % 改行の後，1cm空ける
\Large\textsf{\today}
\end{titlepage}
\clearpage
```

などと書けば独立したタイトルページが作られる．

4 書体の変更

4.1 書体変更のコマンド

LaTeX において，フォントの書体はファミリー，シリーズ，シェイプの 3 つの要素で表せる．ファミリーは，デザイン上で同一の系統に属する書体の集合であり，欧文では，ローマン体（セリフ体），サンセリフ体，タイプライタ体の 3 種，和文では，明朝体，ゴシック体の 2 種（jlreq-deluxe パッケージを使用する場合は，丸ゴシック体が加わり 3 種）が定義される．シリーズは，文字のウエイトによって分類され，TeX では通常ミディアムとボールドの 2 種類が定義される．シェイプは字形の分類であるが，和文ではファミリーを変えることが一般的なので，シェイプは定義されてい

表 2　ファミリーの指定方法

ファミリー	命令型	宣言型	出力
セリフ体（標準）	`\textrm{Roman}`	`{\rmfamily Roman}`	Roman
サンセリフ体	`\textsf{Sans Serif}`	`{\sffamily Sans Serif}`	Sans Serif
タイプライタ体	`\texttt{Typewriter}`	`{\ttfamily Typewriter}`	Typewriter
明朝体（標準）	`\textmc{明朝体}`	`{\mcfamily 明朝体}`	明朝体
ゴシック体	`\textgt{ゴシック体}`	`{\gtfamily ゴシック体}`	ゴシック体
丸ゴシック体	`\textmg{丸ゴシック体}`	`{\mgfamily 丸ゴシック体}`	丸ゴシック体

ない．

これらは，あくまでコマンド上の分類であり，実際に使用されるフォントの書体は独立に設定できることに注意する．例えば，セリフ体のコマンドにサンセリフ体のフォントを割り当てると，標準書体でサンセリフ体のフォントが使用されるし，ボールド体のコマンドに別のウエイトを割り当てることも可能である．

ファミリーは，何も指定していない状態では，標準設定（欧文はセリフ体，和文は明朝体）になるが，その状態から他のファミリーに切り替えるコマンドには，命令型，宣言型，環境型の 3 種類がある．例えば，サンセリフ体を指定するには，

- `\textsf{Sans Serif}`（**命令型**: 引数部分が有効範囲）
- `{\sffamily Sans Serif}`（**宣言型**: { }に囲まれた範囲が有効範囲），
- `\begin{sffamily} Sans Serif \end{sffamily}`（**環境型**: \begin と\end の間が有効範囲）

の 3 通りの方法がある．TEX はアメリカ発祥なので，元々標準では欧文用のコマンドしか存在しないが，(u)pLATEX では，和文用のコマンドが追加されている．表 2 にファミリー指定方法を示す．下の 3 つは和文フォント用のコマンドであり，jlreq-deluxe パッケージを使う場合は，丸ゴシック体用のコマンドも使える．また，欧文フォントに STIX Two しか指定していない場合など，指定したファミリーがフォントにない場合には，デフォルトの CM フォントが代用される．

シリーズの指定方法も命令型，宣言型，環境型の 3 種類が使えるが，フォントがもたないウエイトを指定した場合は無視される（ワーニングメッセージが表示される）．表 3 にシリーズの指定方

NFSS（New Font Selection Scheme）

LATEX 2ε より導入されたフォント管理の仕組み（NFSS: New Font Selection Scheme）では，フォントが，encoding/family/series/shape(/size) で管理される．例えば，コンパイルの最中に出てくるワーニングメッセージには，次のようなフォント関連のものがある．

```
LaTeX Font Warning: Font shape `JY2/hmc/b/n' undefined
(Font)              using `JY2/hmc/bx/n' instead on input line 123.
```

これは，ファイルの 123 行目の文について，JY2 エンコーディングをもつヒラギノ明朝（hmc）のボールドシリーズ（b）の立体シェイプ（n）が定義されていないので，エキストラボールドシリーズ（bx）の立体シェイプ（n）で定義されているフォントを代用する，という意味である．

表 3　シリーズの指定方法

シリーズ	命令型	宣言型	出力
ミディアム（標準）	`\textmd{Medium}`	`{\mdseries Medium}`	Medium
ボールド	`\textbf{Boldface}`	`{\bfseries Boldface}`	**Boldface**
エクストラボールド	`---`	`{\gtfamily\ebseries 特太}`	**特太**

表 4　シェイプの指定方法

シェイプ	命令型	宣言型	出力
立体（標準）	`\textup{Upshape}`	`{\upshape Upshape}`	Upshape
イタリック体	`\textit{Italic}`	`{\itshape Italic}`	*Italic*
スモールキャップ体	`\textsc{Small Capital}`	`{\scshape Small Capital}`	SMALL CAPITAL
斜体	`\textsl{Slanted}`	`{\slshape Slanted}`	*Slanted*

法を示す．エクストラボールドは，pxchfon パッケージを使用すると使える和文フォント用のコマンドであるが，用意されているのは宣言型のコマンドのみである．なお，ここでのミディアムとかボールドというのは，あくまで TEX のコマンド名の話であり，使用されるフォントのウエイトとは異なることに注意する．ミディアムがフォントのどのウエイトに対応するかは設定による．通常，TEX におけるミディアムは，フォントにおけるレギュラーウエイトが割り当てられている．

シェイプは主に欧文フォントに使うコマンドであり，和文フォント用のコマンドはない（和文に使うとワーニングが出て無視される）．ここでイタリック体と斜体（スラント体あるいはオブリーク体とも呼ばれる）は別の字体であることに注意する．イタリックは手書き文字を元にして専用にデザインされた書体であるのに対し，斜体は立体を傾けただけの字体である．フォントによってどちらかしか用意されていない場合には，ある方で代用される．STIX Two Text はイタリック体はあっても，斜体はないため斜体でもイタリックが表示されている．表 4 にシェイプの指定方法を示す．

4.2　コマンド使用例

ファミリー，シリーズ，シェイプのコマンドはそれぞれ組み合わせて使うことで，さまざまな書体を実現する．ところで，和文と欧文が混在する環境ではコマンドの振る舞いに差が生じる．ここ

本文のフォント

我々は文章を読むとき，文字を 1 文字ずつ見ているのではなく，数文字ごとのかたまりで捉えている．文字を捉えるときに視線を一瞬止めることを視点停留と呼び，次のかたまりに視線を動かすことを視点飛躍と呼ぶ．つまり停留と飛躍を繰り返すことにより文章を読み理解している．このとき，停留ポイントの目安となるのが句読点や漢字となる．書籍や専門書の本文において主に明朝系の書体が利用されるのは，明朝体は仮名と漢字のデザインが根本的に異なるため，停留のポイントが見つけやすく，長文を読んでも疲れにくいことが理由にある．逆にゴシック体は仮名と漢字が統一されたデザインの場合が多く，本文での利用としては文面の強弱が識別しづらいといえる．ただし，スクリーンやディスプレイで見るときは，線の細い明朝体が見づらい場合もある．

では，それぞれ出力がどうなるかを見ていく．まずは欧文用のファミリーとシリーズを組み合わせた命令型コマンドを使った入力と出力の例を示す．

命令型欧文コマンド使用例

```
\textrm{1. This is a regular Serif font. これは細明朝体です．}
\textrm{\textbf{2. This is a bold Serif font. これは中太明朝体です．}}
\textsf{3. This is a regular Sans Serif font. これは細ゴシック体です．}
\textsf{\textbf{4. This is a bold Sans Serif font. これは中太ゴシック体です．}}
\texttt{5. This is a regular Typewriter font. しかし，これは細ゴシック体です．}
\texttt{\textbf{6. This is a bold Typewriter font. しかし，これは細ゴシック体です．}}
```

出力

1. This is a regular Serif font. これは細明朝体です．
2. **This is a bold Serif font. これは中太明朝体です．**
3. This is a regular Sans Serif font. これは細ゴシック体です．
4. **This is a bold Sans Serif font. これは中太ゴシック体です．**
5. `This is a regular Typewriter font. しかし，これは細ゴシック体です．`
6. **`This is a bold Typewriter font. しかし，これは中太ゴシック体です．`**

ローマン体，サンセリフ体はそれぞれ，明朝体，ゴシック体に対応しておりボールド体も問題なく表示されている．また，和文にはタイプライタ体相当の文字がないので，最後の 2 つは通常は指定しない使い方ではあるが，明朝ではなくゴシック体で代用されることは知っておいて良いであろう．次に和文用のファミリーとシリーズを組み合わせた命令型コマンドを使った場合の入力と出力の例を示す．

命令型和文コマンド使用例

```
\textmc{\textmd{1. これは細明朝体です．This is a regular Serif font.}}
\textmc{\textbf{2. これは中太明朝体です．This is a bold Serif font.}}
\textgt{\textmd{3. これは細ゴシック体です．However, this is a regular Serif font.}}
\textgt{\textbf{4. これは中太ゴシック体です．However, this is a bold Serif font.}}
\textmg{\textmd{5. これは中細丸ゴシック体です．However, this is a regular Serif font.}}
\textmg{\textbf{6. これも中細丸ゴシック体です．However, this is a bold Serif font.}}
```

出力

1. これは細明朝体です．This is a regular Serif font.
2. **これは中太明朝体です．This is a bold Serif font.**
3. これは細ゴシック体です． However, this is a regular Serif font.
4. **これは中太ゴシック体です． However, this is a bold Serif font.**
5. これは中細丸ゴシック体です． However, this is a regular Serif font.
6. **これも中細丸ゴシック体です． However, this is a bold Serif font.**

この場合，全ての欧文部分がローマン（セリフ）体で表示されている．これは欧文の標準がセリフ体であるからである．つまり，和文用のファミリーコマンドは欧文には適用されない（シリーズコマンドは適用される）ことを意味する．元々，欧文環境には丸ゴシック体相当の書体コマンドがないため，丸ゴシック体の指定では自動的に代替フォントとして，標準のセリフ体が使われる．しかし，ゴシック体の指定もセリフ体になるため，和文に英数字が含まれる場合には注意が必要である．

続いて宣言型コマンドの例を見ていく．結果は命令型コマンドを用いた場合と同様となるので，説明は省略する．なお，コマンドをどのタイプで書くのかに特にルールはなく個人の好みで構わない．命令型はコマンドを組み合わせると中カッコが増えるので，対応を間違えやすい．宣言型は入力文字数が多くなりがちといった違いがある．短い単語なら命令型，長めの文章なら宣言型等，自分なりの入力のし易さで決めればよい．

宣言型欧文コマンド使用例

```
{\rmfamily\mdseries 1. This is a regular Serif font. これは細明朝体です．}
{\rmfamily\bfseries 2. This is a bold Serif font. これは中太明朝体です．}
{\sffamily\mdseries 3. This is a regular Sans Serif font. これは細ゴシック体で
    す．}
{\sffamily\bfseries 4. This is a bold Sans Serif font. これは中太ゴシック体で
    す．}
{\ttfamily\mdseries 5. This is a regular Typewriter font. しかし，これは細ゴ
    シック体です．}
{\ttfamily\bfseries 6. This is a bold Typewriter font. しかし，これは中太ゴシッ
    ク体です．}
```

出力

1. This is a regular Serif font. これは細明朝体です．
2. **This is a bold Serif font. これは中太明朝体です．**
3. This is a regular Sans Serif font. これは細ゴシック体です．
4. **This is a bold Sans Serif font. これは中太ゴシック体です．**
5. This is a regular Typewriter font. しかし，これは細ゴシック体です．
6. **This is a bold Typewriter font. しかし，これは中太ゴシック体です．**

宣言型和文コマンド入力

```
{\mcfamily\mdseries 1. これは細明朝体です．This is a regular Serif font.}
{\mcfamily\bfseries 2. これは中太明朝体です．This is a bold Serif font.}
{\mcfamily\ebseries 3. これは中太明朝体です．This is a bold Serif font.}
{\gtfamily\mdseries 4. これは細ゴシック体です．However, this is a regular Serif
    font.}
{\gtfamily\bfseries 5. これは中太ゴシック体です．However, this is a bold Serif
   font.}
{\gtfamily\ebseries 6. これは特太ゴシック体です．However, this is a regular
   Serif font.}
{\mgfamily\mdseries 7. これは中細丸ゴシック体です．However, this is a regular
   Serif font.}
{\mgfamily\bfseries 8. これも中細丸ゴシック体です However, this is a bold Serif
    font.}
```

出力

1. これは細明朝体です．This is a regular Serif font.
2. **これは中太明朝体です．This is a bold Serif font.**
3. **これは中太明朝体です．**This is a bold Serif font.
4. これは細ゴシック体です．However, this is a regular Serif font.
5. **これは中太ゴシック体です．However, this is a bold Serif font.**
6. **これは特太ゴシック体です．**However, this is a regular Serif font.
7. これは中細丸ゴシック体です．However, this is a regular Serif font.
8. **これも中細丸ゴシック体です．However, this is a bold Serif font.**

続いてシェイプコマンドも組み合わせた例である．和文フォントにはシェイプが存在しないので，欧文フォントのみ示している．

Stix Two フォントのシェイプ

1. This text is default Roman.
2. **This text is specified as bold Roman.**
3. *This text is specified as Italic.*
4. ***This text is specified as bold Italic.***
5. *This text is specified as slanted/oblique, but appears as Italic.*
6. ***This text is specified as bold slanted/oblique, but appears as bold Italic.***
7. THIS TEXT IS SPECIFIED AS SMALL CAPITALS.
8. **THIS TEXT IS SPECIFIED AS BOLD SMALL CAPITALS.**

Roboto フォントのシェイプ

1. This text is specified as Sans Serif.
2. **This text is specified as bold Sans Serif.**
3. *This text is specified as Italic, but appears as slanted.*
4. ***This text is specified as bold Italic, but appears as bold slanted.***
5. *This text is specified as slanted.*
6. ***Thi text is specified as bold slanted.***
7. THIS TEXT IS SPECIFIED AS SMALL CAPITALS.
8. **THIS TEXT IS SPECIFIED AS BOLD SMALL CAPITALS.**

ところで，本稿では，セリフ体に STIX Two，サンセリフ体に Roboto フォントを使用しているが，Roboto フォントは 6 つのウエイトが使用できる．そこで，それらのウエイトの違いを比較してみる．

Stix Two と Roboto のウエイトの関係

1. This is regular Serif. This is robotoThin Sans Serif.
2. This is regular Serif. This is robotoLight Sans Serif.
3. This is regular Serif. This is robotoRegular Sans Serif.
4. **This is bold Serif. This is robotoMedium Sans Serif.**
5. **This is bold Serif. This is robotoBold Sans Serif.**
6. **This is bold Serif. This is robotoBlack Sans Serif.**

STIXTwo では，レギュラーとボールドの 2 ウエイトのみが使用できるが，Roboto のレギュラーとボールドと比べてもそれほど違いはなく，バランスは取れていると思う．

最後に，STIX Two の数式用フォントの例である．stix2 パッケージのところでも述べたが，STIX Two Math にはセリフ体やタイプライタ体が含まれる．そこで，Roboto との比較も示している．

Stix Two Math のセリフ体（STIX Two Text との比較を含む）

\mathrm：123ABCD ≠ 456abcd（**STIX Two Text:** 123ABCD ≠ 456abcd）
\mathbf：**123ABCD ≠ 456abcd**（**STIX Two Text: 123ABCD ≠ 456abcd**）
\mathit：*123ABCD ≠ 456abcd*（**STIX Two Text:** *123ABCD ≠ 456abcd*）
\mathbfit：***123ABCD ≠ 456abcd***（**STIX Two Text:** ***123ABCD ≠ 456abcd***）

数式中では，変数はイタリック体にするが，数字はイタリック体にしない．したがって，数字の部分にだけ違いが現れている．

Stix Two Math のサンセリフ体（Roboto との比較を含む）

\mathsf：123ABCD ≠ 456abcd（**Roboto:** 123ABCD ≠ 456abcd）
\mathbfsf：**123ABCD ≠ 456abcd**（**Roboto: 123ABCD ≠ 456abcd**）
\mathsfit：*123ABCD ≠ 456abcd*（**Roboto:** *123ABCD ≠ 456abcd*）
\mathbfsfit：***123ABCD ≠ 456abcd***（**Roboto:** ***123ABCD ≠ 456abcd***）
\mathtt：123ABCD ≠ 456abcd（**Roboto:** 123ABCD ≠ 456abcd）

イタリック体の数字の部分に違いが現れるのはセリフ体と同様である．

Stix Two Math のその他のフォント

\mathbb：𝟙𝟚𝟛𝔸𝔹ℂ𝔻 ≠ 𝟜𝟝𝟞𝕒𝕓𝕔𝕕 （Blackboard-Bold: 黒板文字）
\mathscr：𝓅𝓆𝒜ℬ𝒞𝒟 ≠ ⋮⋯⋰ 𝒶𝒷𝒸𝒹 （Script: 筆記体）
\mathbfscr：𝓹𝓺𝓐𝓑𝓒𝓓 ≠ ⋮⋯⋰ 𝓪𝓫𝓬𝓭
\mathcal：𝒻𝒻𝒻𝒜ℬ𝒞𝒟 ≠ ⨍∫⫋⫌⫋⫌（Calligraphy: 装飾文字の 6 が未定義でワーニングが出る）
\mathfrak：☽☾☽𝔄𝔅ℭ𝔇 ≠ ☣☣(𝔞𝔟𝔠𝔡 （Fraktur: ドイツ文字）
\mathbffrak：☽☾☽𝕬𝕭𝕮𝕯 ≠ ☣☣(𝖆𝖇𝖈𝖉

5　おわりに

本稿では，情報工学科が用意した TEX の設定スクリプトにより設定される内容と，標準的な文書で使われるパッケージを解説しながら，実際の入力サンプルを提示した．TEX の環境構築では，フォントの部分が一番難しいため，フォントの話題が多かったが，一度きちんと設定してしまえば，あとは文章の質を高めることに時間を使える．別途テンプレートファイル（`template1.tex` と `template2.tex`）を用意してあるので，そちらをコピーして自分の文書を作成してみて欲しい．TEX を使用すると見た目や体裁については問題のない文書が作成できるが，大事なのは中身も伴うことである．素晴らしい内容のレポートを期待する．

参考文献

[1] Donald E. Knuth, Don Kunuth's Home Page. https://www-cs-faculty.stanford.edu/~knuth/（参照 2025 年 3 月 14 日）.

[2] "JIS X 4051: 2004 日本語文書の組版方法," 日本産業規格, 2004. https://kikakurui.com/x4/X4051-2004-02.html（参照 2025 年 3 月 14 日）.

[3] モリサワ公式 note, "多言語組版ルールブック," 株式会社モリサワ, 2024. https://www.morisawa.co.jp/fonts/multilingual/typesetting/（参照 2025 年 3 月 14 日）.

[4] "PDF Reference, version 1.7 (6th Ed.)," Adobe Systems Inc., Oct. 2006. https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/pdfreference1.7old.pdf（参照 2025 年 3 月 14 日）.

[5] "アドビポストスクリプト技術へのおさそい," アドビ株式会社, 1998. https://www.adobe.com/jp/print/postscript/pdfs/welcmps2.pdf（参照 2025 年 3 月 14 日）.

[6] The LaTeX Project, LaTeX – A document preparation system, 2024. https://www.latex-project.org/（参照 2025 年 3 月 14 日）.

[7] アスキードワンゴ編集部, "ASCII Nihongo TeX (Publishing TeX)," 株式会社ドワンゴ, 2002. https://asciidwango.github.io/ptex/（参照 2025 年 3 月 14 日）.

[8] "Unicode®16.0.0." Unicode Inc., Sept. 2024. https://www.unicode.org/versions/Unicode16.0.0/（参照 2025 年 3 月 14 日）.

[9] "OpenType®," Adobe Systems Inc., 2023. https://www.adobe.com/jp/products/type/opentype.html（参照 2025 年 3 月 14 日）.

[10] Takuji Tanaka, "upTeX, upLaTeX – 内部 Unicode 版 pTeX, pLaTeX の実装," TeX メモ, Nov. 2021. http://www.t-lab.opal.ne.jp/tex/uptex.html（参照 2025 年 3 月 14 日）.

[11] The LuaTex team, LuaTeX, 2024. https://www.luatex.org（参照 2025 年 3 月 14 日）.

[12] Hironori Kitagawa, "luatexja – Typeset Japanese with Lua(La)TeX," Comprehensive TeX Archive Network (CTAN), Oct. 2024. https://ctan.org/pkg/luatexja（参照 2025 年 3 月 14 日）.

[13] "MacTeX," TeX Users Group, Mar. 2024. https://www.tug.org/mactex/（参照 2025 年 3 月 14 日）.

[14] Rémi Prévost, Mike McQuaid, and Danielle Lalonde, Homebrew – The Missing Package Manager for macOS (or Linux), 2024. https://brew.sh（参照 2025 年 3 月 14 日）.

[15] The MacPorts Project, The MacPorts Project Official Homepage, 2024. https://www.macports.org（参照 2025 年 3 月 14 日）.

[16] Donald E. Knuth, "Computer Modern Font," Font Squirrel, Feb. 2017. https://www.fontsquirrel.com/fonts/computer-modern（参照 2025 年 3 月 14 日）.

[17] Masamichi Hosoda, "原ノ味フォント/Harano Aji Fonts," GitHub, Oct. 2023. https://github.com/trueroad/HaranoAjiFonts（参照 2025 年 3 月 14 日）.

[18] Haruhiko Okumura, "jsclasses – Classes tailored for use with Japanese," Comprehensive TeX Archive Network (CTAN), Feb. 2023. https://ctan.org/pkg/jsclasses（参照 2025 年 3 月 14 日）.

[19] 千葉弘幸, 枝本順三郎, Richard Ishida, 石野恵一郎, 加藤誠一, 小林龍生, 小林敏, Nathaniel McCully, 小野澤賢三, Felix Sasaki, 下農淳司, 塩澤元, 薛富僑, “日本語組版処理の要件（日本語版）,” World Wide Web Consortium (W3C) Working Group, Aug. 2020. https://www.w3.org/TR/jlreq/（参照 2025 年 3 月 14 日）.

[20] Noriyuki Abe, jlreq, Feb. 2024. https://tug.org/docs/latex/jlreq/jlreq-ja.html（参照 2025 年 3 月 14 日）.

[21] Yukimasa Morimi, “jlreq-deluxe – Multi-weight Japanese font support for the jlreq class,” Comprehensive TEX Archive Network (CTAN), Feb. 2024. https://ctan.org/pkg/jlreq-deluxe（参照 2025 年 3 月 14 日）.

[22] Takuji Tanaka, “japanese-otf – Advanced font selection for platex and its friends,” Comprehensive TEX Archive Network (CTAN), Oct. 2023. https://ctan.org/pkg/japanese-otf（参照 2025 年 3 月 14 日）.

[23] Takayuki Yato, “pxchfon– Japanese font setup for pLATEXand upLATEX,” Comprehensive TEX Archive Network (CTAN), Aug. 2023. https://ctan.org/pkg/pxchfon（参照 2025 年 3 月 14 日）.

[24] The LATEX Project Team, “fontenc – Standard package for selecting font encodings,” Comprehensive TEX Archive Network (CTAN), Nov. 2024. https://ctan.org/pkg/fontenc（参照 2025 年 3 月 14 日）.

[25] David M. Jones, “stix2-otf – OpenType Unicode text and maths fonts,” Comprehensive TEX Archive Network (CTAN), Apr. 2021. https://ctan.org/pkg/stix2-otf/（参照 2025 年 3 月 14 日）.

[26] Bob Tennent, “roboto – Support for the Roboto family of fonts,” Comprehensive TEX Archive Network (CTAN), Sept. 2022. https://ctan.org/pkg/roboto（参照 2025 年 3 月 14 日）.

[27] Morten Høgholm, Lars Madsen, and the LATEX3 project, “mathtools – Mathematical tools to use with amsmath,” Comprehensive TEX Archive Network (CTAN), Oct. 2024. https://ctan.org/pkg/mathtools（参照 2025 年 3 月 14 日）.

[28] David Carlisle, and The LATEX Project Team, “graphicx – Enhanced support for graphics,” Comprehensive TEX Archive Network (CTAN), Sept. 2021. https://ctan.org/pkg/graphicx（参照 2025 年 3 月 14 日）.

[29] The LATEX Project Team, “xcolor – Driver-independent color extensions for LATEXand pdfLATEX,” Comprehensive TEX Archive Network (CTAN), Nov. 2023. https://ctan.org/pkg/xcolor（参照 2025 年 3 月 14 日）.

[30] Takayuki Yato, “pxrubrica – Ruby annotations according to JIS X 4051,” Comprehensive TEX Archive Network (CTAN), Apr. 2017. https://ctan.org/pkg/pxrubrica（参照 2025 年 3 月 14 日）.

[31] Japanese TEX Development Community, “ascmac – Boxes and picture macros with Japanese vertical writing support,” Comprehensive TEX Archive Network (CTAN), Jan. 2020. https://ctan.org/pkg/ascmac（参照 2025 年 3 月 14 日）.

[32] Thomas F. Sturm, “tcolorbox –Coloured boxes, for LATEXexamples and theorems, etc,” Comprehensive TEX Archive Network (CTAN), Oct. 2024. https://ctan.org/pkg/tcolorbox（参照 2025 年 3 月 14 日）.

[33] Jobst Hoffmann, “listings – Typeset source code listings using LaTeX,” Comprehensive TeX Archive Network (CTAN), Sept. 2024. https://ctan.org/pkg/listings（参照 2025 年 3 月 14 日）.

[34] The LaTeX Project Team, “hyperref – Extensive support for hypertext in LaTeX,” Comprehensive TeX Archive Network (CTAN), Sept. 2024. https://ctan.org/pkg/hyperref（参照 2025 年 3 月 14 日）.

[35] Takayuki Yato, “pxjahyper – Hyperref support for pLaTeX,” Comprehensive TeX Archive Network (CTAN), Apr. 2022. https://ctan.org/pkg/pxjahyper（参照 2025 年 3 月 14 日）.

[36] Takayuki Yato, “PXchfon-extras,” GitHub, Oct. 2019. https://github.com/zr-tex8r/PXchfon-extras/tree/master（参照 2025 年 3 月 14 日）.

[37] Norbert Preining, tlcontrib – supplementary TeX Live package repository, 2024. https://contrib.texlive.info（参照 2025 年 3 月 14 日）.

A　数式など

TeX では，数式を入力するときに数式モードに切り替える．数式モードは，`$a+b=c$`のようにドルマークで囲む，あるいは`\(a+b=\sqrt{c}\)` のように中カッコのコマンドでくくる方法がある．式番号を伴う式などでは，`\begin{equation}\end{equation}`で囲む．詳しくはドキュメントを参照のこと．以下は数式の表示例である．

文章中で書くと，$a+b=\sqrt{c}$ のようになる．また数式番号を伴う独立した式では，

$$y_1(t) = A\cos(2\pi\omega t + \theta) + n_1(t) \tag{1}$$

のようになる．

以下の例はソースを参照のこと．

$$\boldsymbol{H}_n = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \tag{2}$$

$$\operatorname{sinc} x = \frac{\sin x}{x} \tag{3}$$

B　ヒラギノ角ゴシックの全ウエイト使用例

MacOS 内蔵のヒラギノ角ゴシックには，W0~W9 のウエイトがある．内容が高度になるのでここでは説明しないが（このサンプルでは PDF を読み込んでいます），(u)pLaTeX では全てのウエイトを使用することが可能である．そこで，最後にヒラギノ角ゴシックの各ウエイトと Roboto の各ウエイトを比べてみる．

標準の設定では，通常フォントがヒラギノの W3 と Roboto のレギュラー，太字では，ヒラギノの W6 と Roboto のボールドが使用されるが，Roboto のレギュラーはヒラギノの W4 とマッチし，

ヒラギノ角ゴシックと Roboto のウェイトの比較

W0: ヒラギノ角ゴシック/RobotoThin
W1: ヒラギノ角ゴシック/(RobotoLight)
W2: ヒラギノ角ゴシック/RobotoLight
W3: ヒラギノ角ゴシック/(RobotoLight)/(**RobotoRegular**)
W4: ヒラギノ角ゴシック/RobotoRegular/(**RobotoMedium**)
W5: **ヒラギノ角ゴシック**/(**RobotoMedium**)
W6: **ヒラギノ角ゴシック**/**RobotoMedium**/(**RobotoBold**)
W7: **ヒラギノ角ゴシック**/**RobotoBold**
W8: **ヒラギノ角ゴシック**/(**RobotoBold**)/**RobotoBlack**
W9: **ヒラギノ角ゴシック**/(**RobotoBlack**)

ボールドは W7 とマッチするように見える．Roboto と STIX Two は同程度であるので，脚注 17）で，欧文フォントがヒラギノより若干太いのが気になると書いたが，実際そのとおりであることが確認できる．とはいえ，大きな差があるわけでもないので，気にしないならこのままでも問題ない．