

Tips for L^AT_EX

最終更新日: 2020 年 2 月 21 日

文責: 柳浦 睦憲

L^AT_EX で主に組合せ最適化に関する論文等を書く際に注意すべきことや知っている便利なこと等を書いたメモです。L^AT_EX のソースとコンパイル結果の両方を見ながら読んでください。

1 align 環境について

align 環境は数理計画問題（整数計画問題，線形計画問題）の定式化に便利です。なお，これを使うには `amsmath.sty` が必要 (`\usepackage{amsmath}`) です。eqnarray 環境は `amsmath.sty` でサポートされていないので使うべきではないそうです。

$$x_1 = y_1 \qquad x_2 = y_2 \qquad x_3 = y_3 \qquad (1)$$

のように右揃えと左揃えを交互に繰り返します。数理計画問題の定式化において maximize と subject to（あるいは minimize と subject to や省略形の max と s.t. 等*1）は通常左揃えですが，align の左揃えの列を maximize に使い，目的関数等を 2 つ右の列に書くと

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^n p_j x_j \end{array} \qquad (2)$$

$$\begin{array}{ll} \text{subject to} & \sum_{j=1}^n w_{ij} x_j \leq c_i, \qquad \forall i \in I \end{array} \qquad (3)$$

$$x_j \in \{0, 1\}, \qquad \forall j \in J \qquad (4)$$

のようにテキストの幅が十分あるときには maximize と式の間がずいぶん空いてしまいます。そこで，&を前後にいくつか入れて（前に入れる数は偶数でないと左揃え/右揃えが変わるので注意）空の列を挿入すると

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^n p_j x_j \end{array} \qquad (5)$$

$$\begin{array}{ll} \text{subject to} & \sum_{j=1}^n w_{ij} x_j \leq c_i, \qquad \forall i \in I \end{array} \qquad (6)$$

$$x_j \in \{0, 1\}, \qquad \forall j \in J \qquad (7)$$

*1 紙面にゆとりがあるときには min と s.t. のような省略形ではなく minimize と subject to を使いましょう。

のように中央に寄ります。あるいは、`\parbox` 等を使って同じ幅を指定した箱に `maximize` と `subject to` を入れて、目的関数の列の左隣の列に書けば

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^n p_j x_j \end{array} \quad (8)$$

$$\begin{array}{ll} \text{subject to} & \sum_{j=1}^n w_{ij} x_j \leq c_i, \quad \forall i \in I \end{array} \quad (9)$$

$$x_j \in \{0, 1\}, \quad \forall j \in J \quad (10)$$

のようになります。「 $\forall i \in I$ 」と「 $\forall j \in J$ 」を上例のように右揃えにするか、あるいは左揃えにするかについては、好みでよいと思いますが、1つの論文の中では統一しましょう。

ところで、`align` の偶数列の式の先頭に「 $-$ 」や「 $=$ 」が来ると、その左に少しスペースが入り、

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 - x_3 \\ \text{subject to} & -3x_1 - 5x_2 + 2x_3 \leq 3 \\ & 7x_1 - 8x_2 + x_3 \leq 5 \end{array}$$

のように左がそろわなくなります。これを避けるには「 $-$ 」等の記号を含む部分を `{ }` で囲めば

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 - x_3 \\ \text{subject to} & -3x_1 - 5x_2 + 2x_3 \leq 3 \\ & 7x_1 - 8x_2 + x_3 \leq 5 \end{array}$$

のように式の左がそろいます。ただし、そのような記号の右が少々詰まった感じになってしまうようです。この解決方法が分かったら教えてください。

式番号が要らない場合は `\begin{align*}` のように最後に `*` をつけます（上はそうにした例）。2段組みの原稿等でテキスト幅いっぱいには式を書きたい場合は `flalign` 環境もよいかもしれません。以下は2段組みにしたときの例です。

`align` 環境を使った場合がこちらです。以下のように行頭と行末が少し空きます：

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^n p_j x_j \\ \text{subject to} & \sum_{j=1}^n w_{ij} x_j \leq c_i, \quad \forall i \in I \\ & x_j \in \{0, 1\}, \quad \forall j \in J. \end{array}$$

`flalign` 環境を使った場合がこちらです。以下のように行頭と行末が本文の左右にそろいます：

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^n p_j x_j \\ \text{subject to} & \sum_{j=1}^n w_{ij} x_j \leq c_i, \quad \forall i \in I \\ & x_j \in \{0, 1\}, \quad \forall j \in J. \end{array}$$

2 箇条書きの項目間の空白など

ページ制限が厳しいとき等に `itemize`, `enumerate` 環境などの箇条書きや参考文献リストの項目間の空白を詰めたいときがあります。それらの環境の中で `\itemsep=0pt` を指定すると項目間

の空白は詰まりますが、本文と箇条書きの間は空いてしまいます。 `\parsep=0pt`, `\topsep=0pt`, `\parskip=0pt`などを指定してもうまく行かないようです（簡潔な方法をご存知の方は教えてください）。 `myitemsep.sty`（本サイトに置いてあります）を使う^{*2}と、行間と箇条書きの前後や項目間の行間が

- 箇条書きの 1 行目
- 箇条書きの 2 行目

のように本文の行間と同様になります。また、一部の箇条書きだけでそのような指定をしたい場合や、項目記号を変更する等、細かい指定をしたい場合は `list` 環境を使うと

- ★ 箇条書きの 1 行目
- ★ 箇条書きの 2 行目

のように項目記号や行間をその都度変更できます。

3 図表のキャプション

図表のキャプションにおいて、 \LaTeX ではとくに指定をしなければ、たとえば `jarticle`, `article`, `scrartcl` などを使うと「Figure 3: An example of ...」のように番号の後ろがコロンになりますが（`jsarticle` を使うと番号の後ろには何も記号がつかないようです）、本来はピリオドであると書いている書物もあります。 `mycaption.sty`（本サイトに置いてあります）を使う^{*3}と、Fig. 1 のよう

Fig. 1. An example of ...

に番号の後ろがピリオドになります。キャプションのスタイルを変更したい場合に参考にしてください。

`jarticle` などの日本語のスタイルを使うと図表のキャプションは「図 1」「表 1」のように日本語になりますが、和文と英文が混在する文書を書きたいときにこれらを「Figure 1」「Table 1」のように英語にしたいときがあります。このようなときにはプリアンブルにたとえば

```
\usepackage{caption}
\captionsetup[figure]{name=Fig.~}
\captionsetup[table]{name=Table~}
```

と書くと「図」が「Fig.」に、「表」が「Table」に変わります。~はたとえば「Fig.」と番号の間にスペースを入れるために必要です。なお、上述の `mycaption.sty` と併用したいときは、これらよりも後ろに `\usepackage{mycaption}` を書かないと `mycaption.sty` の変更が有効にならないことがあります。

^{*2} `myitemsep.sty` を \LaTeX のソースと同じディレクトリに置いてプリアンブルに `\usepackage{myitemsep}` と書くか、あるいは `myitemsep.sty` の中に書いてあることを全てプリアンブルにコピーしたのち `\makeatletter` と `\makeatother` の前の `%` 記号を消す。

^{*3} 使い方は `myitemsep.sty` と同様。

4 表の幅と行間

好みではありますが，論文の表では縦線を使わないようにしましょう．そのように指定している論文誌もありますし，慣れるとその方が美しく感じる気がします．

表 1 に横線だけの表の例を示します．この表の「計算時間」や「コスト」のように 2 列以上に跨るときには，この例のように `\cline` を使ってそのことが分かるような線を引きます．その際，「計算時間」の 2 列と「コスト」の 2 列を示す 2 つの横線の間に切れ目が入るように，空の列を入れています（例えば「`\cline{3-4} \cline{5-6}`」のようにしてしまうと 2 本の線が繋がってしまう）．

表 1. 表の例

| 問題例 | 計算時間（秒） | | コスト | |
|-----|---------|-------|------|------|
| | 手法 1 | 手法 2 | 手法 1 | 手法 2 |
| AAA | 10.0 | 12.3 | 123 | 45 |
| BBB | 120.0 | 322.3 | 323 | 556 |

さて，列の多い表の幅をテキストの幅に納めたい，あるいは列の少ない表の幅を広げてテキスト幅に揃えることで見栄えをすっきりしたいという場合は，`\tabcolsep` で調整できます．表の行間を変えたい場合は `\arraystretch` で調整できます．表 2 と 3 に例を示します．また，表 4 のように `\scalebox` で図表のサイズの調整ができます（`graphicx.sty` が必要）．

表 2. 表のサンプル（デフォルト）

| 問題例 | 最良解 | 計算時間（秒） |
|-------|------|---------|
| inst1 | 5050 | 10.3 |
| inst2 | 3050 | 20.3 |
| inst3 | 1050 | 30.3 |

表 3. 表のサンプル（幅を広く，行間を狭く調整したもの）

| 問題例 | 最良解 | 計算時間（秒） |
|-------|------|---------|
| inst1 | 5050 | 10.3 |
| inst2 | 3050 | 20.3 |
| inst3 | 1050 | 30.3 |

`\arraystretch` は数式モードの中での `array` の行間の調整にも使えます．以下の式の左はデフォルト，右は行間を狭くしものです：

$$\begin{bmatrix} P & O \\ Q & R \end{bmatrix}, \quad \begin{bmatrix} P & O \\ Q & R \end{bmatrix}.$$

表 4. 表のサンプル (表 3 を 80% に縮小したもの)

| 問題例 | 最良解 | 計算時間 (秒) |
|-------|------|----------|
| inst1 | 5050 | 10.3 |
| inst2 | 3050 | 20.3 |
| inst3 | 1050 | 30.3 |

5 記号に迷ったら—`\newcommand`

数式で使う文字や記号に迷ったら、`\newcommand` を使って自分でコマンドを定義して書きましょう。あとで変更したくなったとき、コマンドの定義を一箇所変えればすべてを変えることができるので。たとえばプリアンブルに`\newcommand\myItemSet[2]{I_{\#1}(\#2)}`と書いておき、本文中で`「$ \myItemSet{i}{\hat{v}} $」`と打てば $I_i(\hat{v})$ と表示されます。「[2]」は引数が2つあることを、「#1」は1つ目の引数を、「#2」は2つ目の引数を表します (引数の数は自由で、0の場合は省略できる)。プリアンブルの定義を適宜変更してみてください (このファイルのプリアンブルに別の定義を用意しています)。

6 数式中の太文字

ベクトル等を太文字で表したいとき、数式モードの中で`\mathbf` 等を使って文字を太文字にすると \mathbf{x} のような文字になってしまいます。`\mbox{\boldmath x}` のように打てば \boldsymbol{x} のような文字が出てくるのですが、これを毎回打つのは面倒です。そこで`\newcommand\Bx{\mbox{\boldmath x}}` によって`\Bx` というコマンドを用意しておき (本 \LaTeX ファイルの冒頭の「太文字の定義」の部分参照)、数式モードの中では`\Bx` を使うようにすると \boldsymbol{x} のような文字を何度も書く際に便利です。bm パッケージ (プリアンブルに`\usepackage{bm}`) を使って`\bm{x}` のように打つ (出力は「 \boldsymbol{x} 」) 方法もあります。

7 文字の色

査読者に修正箇所を明確にして原稿を渡すことが求められる場合などに一部の文字の色を変更し、たくなことがあります。`\textcolor{red}{...}` のようにすれば変更できます。これを利用するには`\usepackage{color}` が必要です。ただし`\textcolor{red}{...}`では複数の段落をまたいで色を変えることができません。本 \LaTeX ファイルのプリアンブルの「フォントの色の定義」の部分に倣って色を変えるコマンド`\ColorB` などを用意しておき、`\ColorB{...}` で囲んだ部分の色を変更する方法を使うと、段落をまたいで使え、また、色を自由に定義できます。`rgb` は RGB の3限色による色の指定、`cmymk` はカラーレーザープリンターのトナーと同様の4色による指定です。原稿を渡すときに前回から変更した場所を示す目的等にご利用いただければ幸いです。

8 URL の表示

参考にしたウェブサイトの URL を記したくなることがしばしばありますが、その際、URL をそのまま書くと “~” が消えたり、改行位置がうまくいかなかったりで苦労することがしばしばあります。プリアンブルに `\usepackage{url}` と書き、URL の記述の際に `\url{http://www.aaa.bbb/}` のように書く方法があります。フォントの変更は `\urlstyle{rm}` などで行います。rm, tt, sf, same などが指定でき、以下のようにフォントが変わります。

```
http://www.aaa.bbb/~my/
http://www.aaa.bbb/~my/
http://www.aaa.bbb/~my/
http://www.aaa.bbb/~my/
```

9 2 項演算子・関係演算子

2 項演算子 (binary operator: $+$, $-$, \times , \div , \cup , \cap , \vee , \wedge など) や関係演算子 (relational operator: $=$, \neq , \equiv , $<$, $>$, \leq , \geq , \prec , \succ , \ll , \subset , \subseteq , \in など) の左右にはそのような演算子として適切なスペースが入ります。従ってたとえば差集合 $A \setminus B$ を書く際には `\backslash` ではなく `\setminus` を使い、 $\{x \in \mathbb{R} \mid x^2 + x \leq 10\}$ のような集合の記述の縦棒には `|` ではなく `\mid` を用います (表 5)。また、たとえば a を b で割った余りを表す記号に `%` を 2 項演算子として使って $a \% b = a - b[a/b]$ のように定義したい場合は、`\mathbin{\%}` $a \mathbin{\%} b \dots$ のようにします (表 5)。同様に、(関係演算子ではない) 記号を関係演算子として使う場合は `\mathrel` を用います。

表 5. 関係演算子や 2 項演算子とそうでない記号の比較

| | 記号 | 出力結果の例 |
|---|---------------------------|---|
| ○ | <code>\setminus</code> | $A \setminus B$ |
| × | <code>\backslash</code> | $A \backslash B$ |
| ○ | <code>\mid</code> | $\{x \in \mathbb{R} \mid x^2 + x \leq 10\}$ |
| × | <code> </code> | $\{x \in \mathbb{R} x^2 + x \leq 10\}$ |
| ○ | <code>\mathbin{\%}</code> | $a \% b$ |
| × | <code>\%</code> | $a \% b$ |

式の左右の括弧を `\left\{...\right\}` で大きくするときに、`\mid` をそのまま使うと小さくすることがあります (たとえば (11))。そこで `\mid` を左右の括弧に連動させて大きくしたいとき、`\middle|` とすれば大きくなりますが、関係演算子ではないので左右のスペースが詰まってしまいます (たとえば (12))。しかし、`\middle\mathrel|` や `\mathrel{\middle|}` はエラーが出てうまく行きません。`\mathrel{\middle|}\mathrel{\middle|}` とするとうまく行きます (たと

えば (13)). 頻繁に使うのであれば本 L^AT_EX ファイルのプリアンブルの `\relmiddle` のようなコマンドを用意しておき, 式の中で `\relmiddle` のように書く方法もあります ((14) に例示; 出力は (13) と同じ). 以下は上述の出力例です:

$$\times \left\{ (x_1, \dots, x_n) \in \mathbb{R}^n \mid \sum_{i=1}^n x_i^2 \leq a \right\}, \quad (11)$$

$$\times \left\{ (x_1, \dots, x_n) \in \mathbb{R}^n \left| \sum_{i=1}^n x_i^2 \leq a \right. \right\}, \quad (12)$$

$$\bigcirc \left\{ (x_1, \dots, x_n) \in \mathbb{R}^n \left| \sum_{i=1}^n x_i^2 \leq a \right. \right\}, \quad (13)$$

$$\bigcirc \left\{ (x_1, \dots, x_n) \in \mathbb{R}^n \left| \sum_{i=1}^n x_i^2 \leq a \right. \right\}. \quad (14)$$

10 書いたまま出力する

`\verb` を使うと, 特殊文字等も含めて書いたままの文字を出力できます. 例えば「`\verb#(^_^)#`」のように「`\verb`」の後に区切り記号(左の例では「`#`」)で挟んだ部分が「(^_^)」のように出力されます. 区切り記号には始まりと終わりで同じものを使う必要があります, また, 表示したい文字列に含まれる文字や「`*`」は区切り記号には使えません. `\verb*` を使うとスペースが「`_`」で表示されます. 例えば「`\verb*|2.3 GHz|`」と打つと「`2.3_GHz`」と出力されます. 疑似コードなど複数行に渡る部分をそのまま出力したいときはその部分を `\begin{verbatim} ... \end{verbatim}` で囲みます. `\begin{verbatim*} ... \end{verbatim*}` とすればスペースが「`_`」で表示されます.

11 代入記号

アルゴリズムの記述において変数に値などを代入する際には「`x ← 10`」「`x := 10`」などの記号をしばしば用います. 後者を出力するのに `:=` と打つと「`: =`」となりコロンと等号の上下がちょっとずれてしまいます. これが気になるようなら `\coloneqq` を使うと「`: =`」となります. なお, `\coloneqq` を使うにはプリアンブルに `\usepackage{mathtools}`, `\usepackage{txfonts}`, `\usepackage{pxfonts}` などが要りますが (いずれかひとつあればよい), たとえば後者二つはフォントが変わってしまいます. `\def\mycoloneqq{\mathrel{\mathop:}=}` のように自分でコマンドを定義する手もあります. これらの表示例を以下に示します:

「`x := 10`」(`:=`を用いた場合),

「`x := 10`」(`\coloneqq`を用いた場合),

「`x := 10`」(`\mycoloneqq`を用いた場合).

12 アルゴリズムの疑似コード

アルゴリズムの疑似コードを書くのに `\usepackage{algorithm}` を使う際、デフォルトでは「Algorithm 1, 2, ...」のように番号がつきますが、この Algorithm の部分をたとえば Procedure に変えたいときにはプリアンブルに

```
\makeatletter
```

```
\renewcommand{\ALG@name}{Procedure}
```

```
\makeatother
```

のように書けば変更できます。Procedure 1 に例を示します。

Procedure 1 Ford-Fulkerson

Input: グラフ $G = (V, E)$, 始点 $s \in V$, 終点 $t \in V$, および各辺 $e \in E$ の容量 u_e .

Output: s から t への最大フロー.

```
1: for  $e = 1$  to  $|E|$  do  
2:    $x_e := 0$  とする.  
3: end for  
4: ...
```

13 パッケージの競合

2つのパッケージ (e.g., `pkgA`, `pkgB`) で同じコマンド (e.g., `\cmdInAB`) が `\newcommand` を用いて定義されているとプリアンブルで

```
\usepackage{pkgA}
```

```
\usepackage{pkgB}
```

としてコンパイルしても

```
! LaTeX Error: Command \cmdInAB already defined.
```

というようなエラーが出てしまいます。このような時には

```
\usepackage{pkgA}
```

```
\let\cmdInAB=\relax
```

```
\usepackage{pkgB}
```

とすると衝突を回避できます。ただし、こうすると `pkgA` の `\cmdInAB` は使えなくなります。 `pkgA` の `\cmdInAB` も使う場合は、`\let` で `\cmdInAB` に `\relax` を代入する前に、`\let` で他のコマンド名に `\cmdInAB` を代入しておくことで、その名前で利用できます。

14 数式番号を変更したい

数式番号を変更したい時には、数式中にたとえば「`\tag{56a}`」と打てば「(56a)」になります。「`\tag{56$'}`」と打てば「(56')」になります。「`\tag{\ref{[数式ラベル]}$'}`」のように他の式番号を引用することも可能です。以下の例

$$f(x) = 2x^2 + 3x + 1 \tag{A8}$$

$$g(x) = 3x^2 + 2x + 1 \tag{A8'}$$

では (A8) 式ところで「`\tag{A8} \label{eqTagEg}`」と書き, (A8') 式では「`\tag{\ref{eqTagEg}$'}`」と書いています。

たとえばある制約を表す数式をモデルの定式化の中で書いた時に、別の定式化で異なる数式を使ってその制約を表す際に、元の定式化の番号に「'」をつけたものを別の定式化の対応する制約式の番号としたいときなどにご利用ください。