


L^AT_EX テンプレート

野本 慶一郎

最終更新: 2025 年 2 月 16 日 20 時 49 分 

目次

| | | |
|-----|------------|---|
| 1 | 記号・数式 | 2 |
| 1.1 | 数学文字 | 2 |
| 1.2 | 数学作用素 | 2 |
| 1.3 | 数式 | 2 |
| 2 | 定理・コメント | 3 |
| 2.1 | 定理環境 | 3 |
| 3 | 図 | 4 |
| 4 | アルゴリズム・コード | 5 |
| 4.1 | 擬似コード | 5 |
| 4.2 | ソースコード | 6 |

1 記号・数式

1.1 数学文字

| | |
|-----------------------------|--|
| 黒板文字 (mathbb) | $\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}, \dots$ |
| 筆記体 (mathcal) | $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \dots$ |
| フラクトゥール (mathfrak) | $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}, \dots, \mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \mathfrak{d}, \dots$ |
| 花文字 (mathscr) | $\mathscr{A}, \mathscr{B}, \mathscr{C}, \mathscr{D}, \dots$ |

1.2 数学作用素

MyMathOperators に登録した文字は数学作用素として書くことができる。例えば

$$\mathrm{Ker} f, \mathrm{Hom}(\mathfrak{g}, \mathfrak{h}), \mathrm{Gal}(\overline{K}/K), \mathrm{Spec} A, \mathrm{rank} E(\mathbb{Q}), \mathrm{Sel}^{(\phi)}(E/K)$$

のように使用可能。

1.3 数式

align 環境で数式を書く際には、ラベリングをするかどうかに関わらず「*」は付けなくてよい。例えば数式

$$\zeta(s) := \sum_{n=1}^{\infty} \frac{1}{n^s}$$

は引用していないので、式番号は付いていない。しかし

$$f(z_0) = \frac{1}{2\pi i} \oint_C \frac{f(z)}{z - z_0} dz \tag{1}$$

は式 (1) と引用したので式番号が表示されている。また、括弧は

$$\left(\frac{q}{p}\right), \left\{0, \frac{k}{m}\right\}, \left[\frac{1}{n+1}x^{n+1}\right]_{x=a}^b$$

のように簡潔に書くことができる。また、集合は

$$\left\{(a_i) \in \prod A_i \mid f_{ij}(a_j) = a_i\right\}$$

と書くことができる。

2 定理・コメント

2.1 定理環境

定義や命題等は, 以下のようにして記述する:

定義 2.1: 群の定義 [1, 命題 hoge]

空でない集合 G が群であるとは, 写像

$$\phi : G \times G \rightarrow G$$

で以下の三つの条件を満たすものが存在することをいう.

結合法則 $\forall g, h, i \in G, \phi(\phi(g, h), i) = \phi(g, \phi(h, i)).$

単位元の存在 $\exists e \in G \text{ s.t. } \forall g \in G, \phi(g, e) = \phi(e, g) = e.$

逆元の存在 $\forall g \in G, \exists g^{-1} \in G \text{ s.t. } \phi(g, g^{-1}) = \phi(g^{-1}, g) = e.$

$\phi(g, h)$ のことを単に, $g \cdot h$ や gh と書くことがある.

命題 2.2: 単位元の一意性 [1, 命題 hoge]

群 G の単位元 e は一意的に存在する.

証明 $e, e' \in G$ を単位元とする. 定義 2.1 より

$$\begin{aligned} e &= e \cdot e' \quad (\because e' \text{ は単位元}) \\ &= e' \quad (\because e \text{ は単位元}) \end{aligned}$$

が成り立つ. したがって群の単位元は一意的に存在する. □

注意 2.3: [1, 命題 hoge]

命題 2.2 と同様にして, 逆元の一意性も証明することができる.

3 図

準同型定理の図式は以下のようにして書ける.

$$\begin{array}{ccc}
 G & \xrightarrow{f} & \text{Im } f \\
 \pi \downarrow & \circlearrowleft & \nearrow \cong \\
 & & G/\text{Ker } f
 \end{array}$$

ファイバー積の普遍性は以下のようにして書ける.

$$\begin{array}{ccccc}
 T & & & & \\
 \downarrow \exists! u & \searrow t & & & \\
 X \times_{\mathbb{Z}} Y & \xrightarrow{q} & Y & & \\
 \downarrow p & & \downarrow g & & \\
 X & \xrightarrow{f} & Z & &
 \end{array}$$

s (curved arrow from T to X)

4 アルゴリズム・コード

4.1 擬似コード

擬似コードは、以下のようにして記述する。

Algorithm 1 Euclid の互除法

```

1: def Euclid( $a, b$ ) :
2:    $r \leftarrow a \bmod b$ 
3:   while  $r \neq 0$  :                                     ▷  $r = 0$  ならば最大公約数は  $b$ 
4:      $a \leftarrow b$ 
5:      $b \leftarrow r$ 
6:      $r \leftarrow a \bmod b$ 
7:   return  $b$ 

```

高速に冪乗 a^n を計算するアルゴリズムである **繰り返し二乗法** を説明する。簡単のため、非負整数 $n \in \mathbb{Z}$ のサイズは高々 3 ビットとし、

$$n = n_0 + n_1 2 + n_2 2^2 \quad (n_0, n_1, n_2 \in \{0, 1\})$$

を n の 2 進展開とする。このとき

$$\begin{aligned}
 a^n &= a^{n_0 + n_1 2 + n_2 2^2} \\
 &= a^{n_0} \cdot a^{n_1 2 + n_2 2^2} \\
 &= a^{n_0} \cdot (a^{n_1 + n_2 2})^2 \\
 &= a^{n_0} \cdot (a^{n_1} \cdot (a^{n_2})^2)^2 \\
 &= a^{n_0} \cdot (a^{n_1} \cdot (a^{n_2} \cdot (1)^2)^2)^2
 \end{aligned}$$

が成り立つ。したがってこの場合、内側の括弧から計算を始めることで二乗算を 3 回と乗算を高々 3 回で計算可能である。¹⁾ 一般の非負整数 $n \in \mathbb{Z}$ についても、同様に冪乗算ができる。これが繰り返し二乗法である。

Algorithm 2 繰り返し二乗法

```

1: def pow( $a, n$ ) :
2:    $n = n_0 + n_1 2 + n_2 2^2 + \dots + n_{\ell-1} 2^{\ell-1}$  と表す
3:    $\text{val} \leftarrow 1$ 
4:   for  $i$  in  $\ell - 1, \dots, 0$  :
5:      $\text{val} \leftarrow \text{val} \times \text{val}$ 
6:     if  $n_i == 1$  :
7:        $\text{val} \leftarrow a \times \text{val}$ 
8:   return  $\text{val}$ 

```

1) 一般に、繰り返し二乗法の計算量は $O(\log_2 n)$ である。

4.2 ソースコード

```
1 # This is a VS Code style code block
2 def pow(a, n):
3     val = 1
4     while n != 0:
5         val *= val
6         if n & 1 == 1:
7             val *= a
8         n = n >> 1
9     return val
```

参考文献

[1] 雪江明彦. 代数学 1 群論入門. 日本評論社, 2010.