LATEX 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	号・数式	• •	• •	•	• •	 •	 •	•	 •	•	• •	•	•	• •	•	• •	•	 •	•	• •	•	• •	•	•	• •	•	•	• •	•	•	 •	•	•	•	 ,	2
1.3	L	数学文4	字・・	• •			 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•		2
1.5	2	数学作品	用素	•			 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•	 •	2
1.5	3	数式·	• •	• •			 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•		2
2	定理	里・コメ	ント	•			 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•	 •	;
2.	L	定理環境	境 • ·	• •			 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•		
3	図		• • •	• •			 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•	 •	4
4	アル	レゴリズ	ム・:	コー	ド	•	 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•	 •	Ę
4.	L	擬似コ	ード	•			 •	 •	•	 •	•		•			•		•	 •	•		•		•	•		•	•		•	•	 •	•	•	•	 •	Ę
4.5	2	ソース	コー	ド・			 •			 •			•					•				•					•			•			•			 ,	6

1. 記号・数式 1.1. 数学文字

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) $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}, \ldots, \mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \mathfrak{d}, \ldots$

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

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

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

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

は引用していないので、式番号は付いていない. しかし

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

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

と書くことができる.

2. 定理・コメント 2.1. 定理環境

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

$$\phi: G \times G \to G$$

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

$$e = e \cdot e'$$
 (∵ e' は単位元)
= e' (∵ e は単位元)

が成り立つ. したがって群の単位元は一意的に存在する.

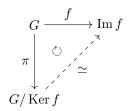
注意 2.3: [1, 命題 hoge]

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.

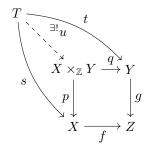
2. 定理・コメント 2.1. 定理環境

3 図

準同型定理の図式は以下のようにして書ける.



ファイバー積の普遍性は以下のようにして書ける.



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: \operatorname{def} \operatorname{Euclid}(a,b):
2: r \leftarrow a \operatorname{mod} b
3: \operatorname{while} r \neq 0: \Rightarrow r = 0 ならば最大公約数は b
4: a \leftarrow b
5: b \leftarrow r
6: r \leftarrow a \operatorname{mod} b
7: \operatorname{return} b
```

高速に冪乗 a^n を計算するアルゴリズムである<mark>繰り返し二乗法</mark>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,

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

をnの2進展開とする.このとき

$$a^{n} = a^{n_{0} + n_{1} + n_{2} + n_{2} + 2^{2}}$$

$$= a^{n_{0}} \cdot a^{n_{1} + n_{2} + 2^{2}}$$

$$= a^{n_{0}} \cdot (a^{n_{1} + n_{2} + 2^{2}})^{2}$$

$$= a^{n_{0}} \cdot (a^{n_{1}} \cdot (a^{n_{2}})^{2})^{2}$$

$$= a^{n_{0}} \cdot (a^{n_{1}} \cdot (a^{n_{2}} \cdot (1)^{2})^{2})^{2}$$

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: val \leftarrow 1
4: for i in \ell - 1 , ... , 0:
5: val \leftarrow val \times val
6: if n_i == 1:
7: val \leftarrow a \times val
8: return 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. アルゴリズム・コード 4.2. ソースコード

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

参考文献

[1]雪江明彦. 代数学 1 群論入門. 日本評論社, 2010.