

# 「スライス計算」の数式解説（「compute\_slice.cpp」）

Open DEM Japan

2025年6月30日

本稿では、LIGGGHTS/LAMMPS における ComputeSlice クラスの動作を数式で説明する。本クラスは、計算済みのグローバルなベクトル／行列から間引き（スライス）操作を行い、新たなベクトルあるいは行列を生成する機能を担う。

前提：

- $t \in \mathbb{N}$  を離散タイムステップとし、時刻  $t$  における計算結果を添字  $(\cdot)(t)$  で表す。
- 入力源は  $M$  個あり、 $m = 1, \dots, M$  で識別する。各入力源は

$$\begin{aligned} \mathbf{C}^{(m)}(t) &= (C_1^{(m)}(t), C_2^{(m)}(t), \dots) && \text{(ベクトル)} \\ \text{あるいは } \mathbf{A}^{(m)}(t) &= (A_{i,k}^{(m)}(t)) && \text{(行列)} \end{aligned}$$

のいずれかを返す。

スライス範囲の定義：整数

$$n_{\text{start}}, n_{\text{stop}}, n_{\text{skip}} \in \mathbb{N}, \quad 1 \leq n_{\text{start}} \leq n_{\text{stop}}, \quad n_{\text{skip}} \geq 1$$

が与えられる。有効インデックス集合  $\mathcal{I}$  を

$$\mathcal{I} = \left\{ n_{\text{start}} + j n_{\text{skip}} \mid j = 0, 1, \dots, K-1 \right\}, \quad K = \left\lfloor \frac{n_{\text{stop}} - n_{\text{start}}}{n_{\text{skip}}} \right\rfloor \quad (1)$$

と定める。 $|\mathcal{I}| = K$  が出力ベクトルの要素数となる。

出力ベクトル ( $M = 1$  の場合)：入力がベクトル  $\mathbf{C}^{(1)}$  のとき、

$$S_j(t) = C_{i_j}^{(1)}(t), \quad i_j \in \mathcal{I}, \quad j = 0, \dots, K-1. \quad (2)$$

入力が行列  $\mathbf{A}^{(1)}$  で列番号  $k$  を指定する場合 ( $\text{argindex} > 0$ )、

$$S_j(t) = A_{i_j, k}^{(1)}(t), \quad i_j \in \mathcal{I}. \quad (3)$$

出力行列 ( $M \geq 2$  の場合)：各入力  $m$  がベクトル型なら列  $m$  は

$$S_{j,m}(t) = C_{i_j}^{(m)}(t), \quad (4)$$

行列型で列番号  $k_m$  を指定する場合は

$$S_{j,m}(t) = A_{i_j, k_m}^{(m)}(t), \quad (5)$$

を用いて  $j = 0, \dots, K - 1$ ,  $m = 1, \dots, M$  を走らせることで

$$\mathbf{S}(t) = (S_{j,m}(t)) \in \mathbb{R}^{K \times M}$$

を得る.

**呼び出し条件**: 入力源が fix であるときは, その計算頻度  $f_m$  が現在のタイムステップ  $t$  と整合する必要がある. すなわち

$$t \equiv 0 \pmod{f_m}. \quad (6)$$

**集約対象の性質 (集約可否フラグ)**: 入力源が示す各要素の集約可否を  $E_i^{(m)} \in \{0, 1\}$  とし, ベクトル入力の場合

$$e_j^{(m)} = E_{i_j}^{(m)}, \quad (7)$$

行列入力の場合

$$e_j^{(m)} = E_{i_j, k_m}^{(m)}, \quad (8)$$

と定義する. 複数列の場合は

$$e_{j,m} = \begin{cases} E_{i_j}^{(m)} & (\text{ベクトル入力}) \\ E_{i_j, k_m}^{(m)} & (\text{行列入力}) \end{cases}.$$

ここで  $e = 1$  は「集約不可 (extensive)」,  $e = 0$  は「集約可能 (intensive)」を示す.

以上により, ComputeSlice の本質は

1. 式 (??) によるインデックス集合  $\mathcal{I}$  の決定,
2. 式 (??) - (??) による要素抽出,
3. 式 (??) による呼び出し時刻の整合性確認,
4. 式 (??), (??) による集約可否フラグの伝播

に要約される. これにより, ユーザは既存の計算結果から任意の範囲・間隔で値を取り出し, 後続解析に供することができる.