

”AtomVec クラス” の数式解説 (“atom_vec.cpp”)

Open DEM Japan

2025 年 6 月 30 日

本ファイルは AtomVec 基底クラスに実装されたデータ構造操作（配列伸長、速度・結合情報の入出力）を C++ から抽象化し、離散要素法（Discrete Element Method; DEM）計算における数学的操作として整理したものである。以下ではコード中の変数名の列挙は行わず、アルゴリズムを数式で定式化する。

まず、粒子数の最大確保長 n_{\max} は固定ブロック幅 $\Delta = 16384$ に切り上げて確保する。既存の n_{\max} に対して新しい値 n_{\max}^{new} は

$$n_{\max}^{\text{new}} = \left\lfloor \frac{n_{\max}}{\Delta} \right\rfloor \Delta + \Delta, \quad (1)$$

となる。同様に“ボーナス”配列長 n_{bonus} は $\Delta_b = 8192$ により

$$n_{\text{bonus}}^{\text{new}} = \left\lfloor \frac{n_{\text{bonus}}}{\Delta_b} \right\rfloor \Delta_b + \Delta_b. \quad (2)$$

■速度ベクトルの入出力 データファイル（Velocities セクション）から読み取られた $\tilde{\mathbf{v}}_i = (\tilde{v}_{ix}, \tilde{v}_{iy}, \tilde{v}_{iz})$ は内部速度ベクトル \mathbf{v}_i に

$$\mathbf{v}_i = \tilde{\mathbf{v}}_i \quad (i = 1, \dots, N_{\text{loc}}), \quad (3)$$

として格納される。逆にデータ出力用バッファ $\text{Buf} \in \mathbb{R}^{N_{\text{loc}} \times 4}$ には

$$\text{Buf}_{i0} = \text{bitcast}(\text{tag}_i), \quad \text{Buf}_{i\alpha} = v_{i\alpha} \quad (\alpha = 1, 2, 3), \quad (4)$$

という対応で保存される。ここで bitcast は整数タグを倍精度実数のビット列へ再解釈する写像である（ubuf 共用体に相当）。

■結合情報のパック（Bond）局所粒子集合 $\mathcal{L} = \{1, \dots, N_{\text{loc}}\}$ とし、各粒子 $i \in \mathcal{L}$ が持つ結合インデックス $j = 1, \dots, n_i^{(b)}$ に対し、

$$\mathcal{B} = \left\{ (t_{ij}, \text{tag}_i, a_{ij}) \mid i \in \mathcal{L}, j \leq n_i^{(b)}, t_{ij} \neq 0, \Phi(i, j) \right\},$$

を生成する。ここで t_{ij} は符号付き結合タイプ、 a_{ij} は相手粒子タグであり、

$$\Phi(i, j) = \begin{cases} \text{true}, & (\text{Newton 法が有効}) \\ \text{tag}_i < a_{ij}, & (\text{Newton 法が無効}) \end{cases} \quad (5)$$

である。パック後のバッファ長

$$N_{\text{bond}} = |\mathcal{B}| = \sum_{i \in \mathcal{L}} \sum_{j=1}^{n_i^{(b)}} \chi(t_{ij} \neq 0 \wedge \Phi(i, j)), \quad (6)$$

が返される。 $\chi(\cdot)$ は指示関数。

■**角度 (Angle), 二面角 (Dihedral), 振れ角 (Improper)** 角度集合 \mathcal{A} , 二面角集合 \mathcal{D} , 振れ角集合 \mathcal{I} も同様に定義できる. 表現を統一するため

$$N_{\text{angle}} = \sum_{i \in \mathcal{L}} \sum_{j=1}^{n_i^{(a)}} \chi(t_{ij}^{(a)} \neq 0 \wedge \Psi_a(i, j)), \quad (7)$$

$$N_{\text{dihedral}} = \sum_{i \in \mathcal{L}} \sum_{j=1}^{n_i^{(d)}} \chi(t_{ij}^{(d)} \neq 0 \wedge \Psi_d(i, j)), \quad (8)$$

$$N_{\text{improper}} = \sum_{i \in \mathcal{L}} \sum_{j=1}^{n_i^{(i)}} \chi(t_{ij}^{(i)} \neq 0 \wedge \Psi_i(i, j)), \quad (9)$$

と書ける. 各 Ψ は対応する “中心原子” 条件であり, Newton 法有効時は常に真, 無効時は

$$\Psi_a(i, j) : \text{tag}_i = a_{ij}^{(2)}, \quad \Psi_d(i, j) : \text{tag}_i = d_{ij}^{(2)}, \quad \Psi_i(i, j) : \text{tag}_i = p_{ij}^{(2)}, \quad (10)$$

に相当する (添字 $^{(2)}$ は 2 番目原子を示す).

■**ファイル書き出し** バッファ $B, \mathcal{A}, \mathcal{D}, \mathcal{I}$ はインデックス k と共に

$$\text{FILE} \ni (k, t, \text{tag}_1, \text{tag}_2, \dots), \quad (11)$$

のフォーマットで逐次書き込まれる. ここで k は全体通し番号であり, 各書き込み後に $k \leftarrow k + 1$ と更新される.

■**まとめ** 以上のように `atom_vec.cpp` が提供する処理は, 粒子数配列のメモリ拡張 (??),(??), 速度ベクトルのビット単位コピー (??),(??), および結合トポロジ情報のフィルタリング (??)-(??) に要約できる. これらは DEM シミュレーションの物理モデルそのものではないが, データ整合性と並列通信効率を保ちながら巨大粒子系の I/O を実現する基盤として機能している.