

アトムベクトル (atom_vec_atomic.cpp) の数式解説

Open DEM Japan

2025年6月30日

粒子 i の位置・速度・力を

$$\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3}), \quad \mathbf{v}_i = (v_{i1}, v_{i2}, v_{i3}), \quad \mathbf{f}_i = (f_{i1}, f_{i2}, f_{i3})$$

とする。本クラスは各プロセッサが所有するローカル粒子集合 \mathcal{L} に対し、次元 3 の配列 $\mathbf{x}, \mathbf{v}, \mathbf{f} \in \mathbb{R}^{|\mathcal{L}| \times 3}$ および整数配列 $\text{tag}, \text{type}, \text{mask}, \text{image} \in \mathbb{Z}^{|\mathcal{L}|}$ を確保・管理し、他プロセッサとの通信で必要となるシリアライズ (pack) / デシリアライズ (unpack) を数式的に次のように定式化する。

まず周期境界条件 $\mathbf{n} = (n_x, n_y, n_z) \in \mathbb{Z}^3$ を課すときの並進ベクトル $\boldsymbol{\delta}$ は、直交格子では

$$\boldsymbol{\delta} = (n_x L_x, n_y L_y, n_z L_z), \quad (1)$$

非直交 (triclinic) 格子では格子基底 $\mathbf{a}_1 = (L_x, 0, 0)$, $\mathbf{a}_2 = (xy, L_y, 0)$, $\mathbf{a}_3 = (xz, yz, L_z)$ を用いて

$$\boldsymbol{\delta} = n_x \mathbf{a}_1 + n_y \mathbf{a}_2 + n_z \mathbf{a}_3. \quad (2)$$

ここで (L_x, L_y, L_z) は箱寸法, (xy, xz, yz) は傾斜パラメータである。

■位置パック 通信バッファ $\mathbf{B} \in \mathbb{R}^{3n}$ への書き込みは

$$\mathbf{B}_{3k+r} = x_{j_r}^{(\text{send})}, \quad x_{j_r}^{(\text{send})} = \begin{cases} (\mathbf{x}_{j_r})_r & (\text{非周期}) \\ (\mathbf{x}_{j_r} + \boldsymbol{\delta})_r & (\text{周期}) \end{cases}, \quad r = 0, 1, 2, \quad (3)$$

ただし j_r は送信リスト中 k 番目の粒子インデックス, 添字 r は空間成分を表す。

■速度パック ボックス変形速度テンソル $H = \text{diag}(\dot{h}_{xx}, \dot{h}_{yy}, \dot{h}_{zz})$ と粒子が変形群に属することを示す指示関数 χ_i を用いると,

$$\mathbf{v}_i^{(\text{send})} = \mathbf{v}_i + \chi_i H \mathbf{n}, \quad (4)$$

を通信バッファに続けて書き込む ($\chi_i = 0$ なら速度補正を行わない)。

■逆通信 (force reverse) 他プロセッサから受信した力 $\mathbf{f}_i^{(\text{recv})}$ を

$$\mathbf{f}_i \leftarrow \mathbf{f}_i + \mathbf{f}_i^{(\text{recv})}, \quad (5)$$

と加算して集約する。

■境界通信 (border) 隣接プロセッサへ送るデータ列は

$$(\mathbf{x}_i^{(\text{send})}, \text{tag}_i, \text{type}_i, \text{mask}_i, \mathbf{v}_i^{(\text{send})}) \in \mathbb{R}^3 \times \mathbb{Z}^3 \times \mathbb{R}^3, \quad (6)$$

であり、整数値は符号を保ったままビット列として倍精度実数へ詰め替えられる (ユニオン型 ubuf に相当)。

■交換通信 (exchange) 粒子がプロセッサ境界を越えるとき、データ長 m とともに

$$\mathbf{E}_i = (\mathbf{x}_i, \mathbf{v}_i, \text{tag}_i, \text{type}_i, \text{mask}_i, \text{image}_i) \quad (7)$$

を送信し、受信側では $n_{\text{local}} := n_{\text{local}} + 1$ で局所粒子数を更新する.

■再スタート (restart) 各粒子の再スタート情報は

$$(\mathbf{x}_i, \text{tag}_i, \text{type}_i, \text{mask}_i, \text{image}_i, \mathbf{v}_i) \in \mathbb{R}^3 \times \mathbb{Z}^4 \times \mathbb{R}^3, \quad (8)$$

すなわち 11 要素である.

■メモリ消費量 粒子配列の総メモリ、バイト単位

$$M = \sum_{\alpha \in \{\text{tag}, \text{type}, \text{mask}, \text{image}\}} \text{bytes}(\alpha) + \text{bytes}(\mathbf{x}) + \text{bytes}(\mathbf{v}) + \text{bytes}(\mathbf{f}), \quad (9)$$

はクラス末尾で報告される.

以上により AtomVecAtomic は離散要素法における「原子ベクトル」をアルゴリズム的に $\{\mathbf{x}, \mathbf{v}, \mathbf{f}, \text{tag}, \text{type}, \text{mask}, \text{image}\}$ の集合

$$\mathcal{A} = \{(\mathbf{x}_i, \mathbf{v}_i, \mathbf{f}_i, \text{tag}_i, \text{type}_i, \text{mask}_i, \text{image}_i)\}_{i \in \mathcal{L}}$$

として保持し、通信演算 $\Phi_{\text{pack}}, \Phi_{\text{unpack}} : \mathcal{A} \leftrightarrow \mathbb{R}^k$ を式 (??) – (??) に従って可逆に実装している.