

”DumpATOMVTK” の数式解説 (“dump_atom_vtk.cpp”)

Open DEM Japan

2025 年 7 月 2 日

本ファイルは離散要素法 (DEM) シミュレーションにおける粒子 (原子) 情報を VTK 形式の “Unstructured Grid” へ出力するモジュールである。以下ではコードの各処理を変数名に立ち入らず、数式で体系的に記述する。

1. 粒子集合の選択とバッファへの格納

計算ノード (MPI ランク) p に属する局所粒子数を $N_{\text{local}}^{(p)}$ とし、

$$\mathcal{I}^{(p)} = \{ i \in \{1, \dots, N_{\text{local}}^{(p)}\} \mid \text{mask}_i \wedge \text{groupbit} \neq 0 \} \quad (1)$$

を出力対象粒子の添字集合と定義する。ここで \wedge はビット積である。

各粒子 $i \in \mathcal{I}^{(p)}$ について、

$$\mathbf{q}_i = \begin{bmatrix} \mathbf{x}_i \\ r_i \\ m_i \\ \text{ID}_i \\ \text{Type}_i \\ \mathbf{v}_i \\ \boldsymbol{\omega}_i \\ \mathbf{f}_i \\ p \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \\ r_i \\ m_i \\ \text{ID}_i \\ \text{Type}_i \\ v_{x,i} \\ v_{y,i} \\ v_{z,i} \\ \omega_{x,i} \\ \omega_{y,i} \\ \omega_{z,i} \\ f_{x,i} \\ f_{y,i} \\ f_{z,i} \\ p \end{bmatrix} \in \mathbb{R}^{17} \quad (2)$$

を 17 成分の列ベクトルとして構成する (\mathbf{x}_i は位置, r_i は半径, m_i は質量, \mathbf{v}_i は並進速度, $\boldsymbol{\omega}_i$ は角速度, \mathbf{f}_i は力)。

バッファ $\mathbf{B}^{(p)}$ は

$$\mathbf{B}^{(p)} = (\mathbf{q}_{i_1}^T, \mathbf{q}_{i_2}^T, \dots, \mathbf{q}_{i_{|\mathcal{I}^{(p)}}}^T)^T \in \mathbb{R}^{17|\mathcal{I}^{(p)}|} \quad (3)$$

の縦連結である。

2. 出力カウントとシリアライズ

各ランク p は

$$n_{\text{call}}^{(p)} = n_{\text{call}}^{(p)} + 1 \quad (4)$$

と呼出回数を更新し、全ランクで

$$\sum_{p=0}^{P-1} n_{\text{call}}^{(p)} = P \quad (5)$$

が満たされた時点で書き込みを実行する。

さらに、式 (2) の各成分を空白区切りの文字列に写像するシリアライズ写像

$$\text{serialize} : \mathbf{q}_i \mapsto \text{str}(\mathbf{q}_i) \quad (6)$$

を定義し、内部バッファに格納する。

3. vtk データ構造への射影

出力直前に、各 \mathbf{q}_i から以下の写像を行う。位置：

$$\mathbf{x}_i \mapsto \text{vtkPoints} \quad (7)$$

セル（粒子は点セル）：

$$\mathbf{x}_i \mapsto \text{vtkCellArray} \quad (8)$$

スカラー属性：

$$r_i \mapsto \text{radii}, \quad m_i \mapsto \text{mass}, \quad \text{ID}_i \mapsto \text{id}, \quad \text{Type}_i \mapsto \text{type}, \quad p \mapsto \text{proc} \quad (9)$$

ベクトル属性：

$$\mathbf{v}_i \mapsto \text{velocity_lin}, \quad \boldsymbol{\omega}_i \mapsto \text{velocity_ang}, \quad \mathbf{f}_i \mapsto \text{force} \quad (10)$$

これらを

$$\text{Ug} = (\text{Points}, \text{Cells}, \text{PointData}[\text{属性群}]) \quad (11)$$

とする VTK の `UnstructuredGrid` に集約し、最終的なファイル名

$$\text{filecurrent} = \begin{cases} \text{filename} & \text{(単一出力)} \\ \text{filename} \Big|_{* \leftarrow \begin{matrix} \%0k_d(p) \\ \text{BIGINT_FORMAT}(t) \end{matrix}} & \text{(マルチ出力)} \end{cases} \quad (12)$$

に対して

$$\text{Write}(\text{Ug}, \text{filecurrent}) \quad (13)$$

を書き込む。ここで t はシミュレーションステップ、 k はゼロ埋め桁数である。

4. アルゴリズム全体の流れ

ノード p での全手順を簡潔にまとめれば

対象粒子集合 $\mathcal{I}^{(p)}$ を抽出
 $\forall i \in \mathcal{I}^{(p)} : \mathbf{q}_i$ を構成し $\mathbf{B}^{(p)}$ へ連結
 $n_{\text{call}}^{(p)+} = 1; \sum_p n_{\text{call}}^{(p)} = P \Rightarrow$ 書き込み
 $\mathbf{B}^{(p)} \xrightarrow{\text{写像 (7)-(10)}} \text{Ug} \xrightarrow{\text{式 (13)}} \text{disk}$

$$(14)$$

以上により、コードの機能——並列環境下での粒子属性の集約と VTK ファイルへの出力——を数学的に定式化した。