

ライン要素アトムベクトルの数式解説 (atom_vec_line.cpp)

Open DEM Japan

2025年6月30日

本ファイルは2次元離散要素法 (DEM) において、剛体線分 (質量をもつ一次元ロッド) を原子として扱うための内部データ構造と通信手続きを実装している。以下では実装の背後にある連続力学的概念を数式でまとめる。各変数名の由来やデータレイアウトには立ち入らず、物理・幾何量の関係のみを扱う。

■状態量 質量線分 i の

$$\mathbf{r}_i = (x_i, y_i, z_i)^\top \quad (1)$$

$$\mathbf{v}_i = (v_{x,i}, v_{y,i}, v_{z,i})^\top \quad (2)$$

$$\theta_i \in (-\pi, \pi] \quad (3)$$

$$\boldsymbol{\omega}_i = (\omega_{x,i}, \omega_{y,i}, \omega_{z,i})^\top \quad (4)$$

はそれぞれ重心座標、並進速度、面内回転角、および角速度である。ここでシミュレーションは2次元 ($z = 0$) に制限されるため、有効自由度は $\mathbf{r}_i = (x_i, y_i)$, θ_i , $\mathbf{v}_i = (v_{x,i}, v_{y,i})$, $\boldsymbol{\omega}_i = \omega_{z,i} \hat{e}_z$ である。

■幾何量 線分長 l_i と単位接線ベクトル $\hat{\mathbf{t}}_i$ は

$$l_i = \sqrt{(x_{2,i} - x_{1,i})^2 + (y_{2,i} - y_{1,i})^2}, \quad (5)$$

$$\hat{\mathbf{t}}_i = (\cos \theta_i, \sin \theta_i)^\top. \quad (6)$$

端点座標は

$$\mathbf{r}_{1,i} = \mathbf{r}_i + \frac{1}{2} l_i \hat{\mathbf{t}}_i, \quad (7)$$

$$\mathbf{r}_{2,i} = \mathbf{r}_i - \frac{1}{2} l_i \hat{\mathbf{t}}_i. \quad (8)$$

データファイル読込時には端点 $(x_{1,i}, y_{1,i})$, $(x_{2,i}, y_{2,i})$ が与えられ、式(??)-(??)で l_i, θ_i を決定する。角度は符号付き余弦で

$$\theta_i = \begin{cases} \arccos[(x_{2,i} - x_{1,i})/l_i], & y_{2,i} \geq y_{1,i}, \\ -\arccos[(x_{2,i} - x_{1,i})/l_i], & y_{2,i} < y_{1,i}, \end{cases} \quad (9)$$

となる (実装では `acos` と符号判定)。

■質量特性 線密度 ρ_i を入力すると

$$m_i = \rho_i l_i, \quad (10)$$

$$I_i = \frac{1}{12} m_i l_i^2, \quad (11)$$

が質量と面外慣性モーメントである。データファイルでは密度を ρ_i として格納し、読込完了後に式(??)で総質量 m_i に変換する (コードでは `rmass` を上書き)。

■運動方程式 外力 \mathbf{f}_i と外トルク $\tau_i \hat{\mathbf{e}}_z$ を受けるとき、

$$m_i \frac{d\mathbf{v}_i}{dt} = \mathbf{f}_i, \quad (12)$$

$$I_i \frac{d\omega_i}{dt} = \tau_i. \quad (13)$$

タイムステップ積分は

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t + \frac{1}{2} \Delta t), \quad (14)$$

$$\theta_i(t + \Delta t) = \theta_i(t) + \Delta t \omega_i(t + \frac{1}{2} \Delta t), \quad (15)$$

の Velocity - Verlet 型で行われる (アルゴリズム本体は別ファイル)。

■トルクの伝搬 他のロッドや壁との相互作用によって発生した接触力 $\mathbf{f}_{c,i}$ は重心モーメント $\mathbf{r}_{c,i} - \mathbf{r}_i$ で

$$\tau_i = [(\mathbf{r}_{c,i} - \mathbf{r}_i) \times \mathbf{f}_{c,i}] \cdot \hat{\mathbf{e}}_z, \quad (16)$$

を生む。式(??)と組み合わせ、線分の角運動を更新する。

■通信 並列計算では $(\mathbf{r}_i, \mathbf{v}_i, \theta_i, \omega_i)$ を隣接プロセスへ送信/受信する。周期境界条件 $(n_x, n_y) \in \mathbb{Z}^2$ を跨ぐ場合は

$$\mathbf{r}'_i = \mathbf{r}_i + n_x L_x \hat{\mathbf{e}}_x + n_y L_y \hat{\mathbf{e}}_y, \quad (17)$$

を用いて画像座標へ補正してから送信する。摩耗デフォルメ (セル伸縮) 時には速度リマップ $\mathbf{v}_i \mapsto \mathbf{v}_i + \dot{\mathbf{h}} \cdot \mathbf{n}$ を適用し、式(??)に相当する角度補正は不要である。これらの処理は `pack_comm_vel / unpack_comm_vel` に実装されている。

■内部整合性 補助構造体 `Bonusj` = $(\ell_j, \theta_j, i_j^*)$ は「# j 番のロッドはローカル配列インデックス i_j^* に属す」という対応を保持する。実装は

$$i_j^* = \text{index}(j), \quad j = \text{line}[i_j^*], \quad (18)$$

を常に満たすよう `copy_bonus · set_length` で整合を保証し、通信前には `clear_bonus` でゴースト情報を初期化する。

■メモリ計算 総メモリ使用量は

$$M = \sum_{\text{array}} s_{\text{array}} \times n_{\text{elem}} + n_{\text{bonus}} \times \text{sizeof}(\text{Bonus}), \quad (19)$$

で評価し、リスタートファイル出力では式(??)-(??),(??),(??), および ℓ_i, θ_i を1行に直列化する。

以上により、`atom_vec_line.cpp` の数学的核心は「中心一角表示された剛体ロッドの運動・通信・保存」を式(??)-(??)に要約できる。