Lagrangian Stochastic Model 実行方法

(03/03/2021一部変更)

1. **ソースファイル**

/data/SHARE/yokouchi/src\_20210303.tar

1. **粒子ファイルの作成方法(フォーマット)**

particle\_number.txt　と　particle\_position.txt　の２つが必要

* particle\_number.txt

ソースポイントの数だけ書く

* particle\_position.txt

全粒子のx,y,z座標(単位m)と粒子速度のSGS成分(us, vs, ws[m/s]）GroupとID

※フォーマット

x(float) y(float) z(float) us(float) vs(float) ws(float) Group(int) ID(int)

x(float) y(float) z(float) us(float) vs(float) ws(float) Group(int) ID(int)

x(float) y(float) z(float) us(float) vs(float) ws(float) Group(int) ID(int)

x(float) y(float) z(float) us(float) vs(float) ws(float) Group(int) ID(int)

…

* 1. 各変数について
* x, y, z

ソース座標（単位：[m]）

同一のソース座標で同時に複数個の粒子を生成する際は，生成する粒子の個数分記述する．

* us, vs, ws

ソース座標に対応する粒子の初速度を記述する．  
粒子速度のSGS成分は，次のタイムステップに継承されることに注意する．また，GS成分の初速度は風速の内挿値になる．

* Group

粒子を生成するタイミングを変更可能

粒子の生成するタイミングはrunlbm.sh内のgenerate\_step変数(step数)で決定するが，各粒子はGroupで設定された値の倍数回目で粒子が生成される．  
したがって，Group =１の場合，毎回生成される．Group = 2の場合，２の倍数回目で生成される．  
例）generate\_step = 125 (1s)の時  
Group = 1は 125×1ステップ＝１秒間隔で生成  
Group = 2は 125×2ステップ＝２秒間隔で生成  
Group = 3は 125×3ステップ＝３秒間隔で生成  
…

※粒子ごとで生成のタイミングをずらさない場合はすべて1で可

* ID

IDの付け方は本人が分かれば問題ないが，プログラム内では，粒子の生成回数目を足しているので，下３～４桁はゼロ，5桁目からソースポイントのIDを付けると分かり易い

1. **計算手順**

粒子計算の手順は以下の通りである．

1. 粒子ファイルを作成
2. run\_lbm.shと同じ階層内にparticle\_positionディレクトリを作成し，その中にparticle\_number.txtとparticle\_position.txtを入れる
3. read\_particle\_box/read\_particle\_box.txt内の１行目の変数（粒子を生成し始めるステップと何回生成するか）は有効なので，設定する．その他の変数は無効．
4. runlbm.sh内のflag\_particle\_generate変数を1にする．その他の変数を設定する．変数の意味は以下の通りである．

* pout … 粒子ファイルの出力間隔(step)
* pstartstep … 粒子ファイルの初期番号
* particle … 粒子の最大数
* generate\_step … 粒子の生成間隔 (step)

1. /src/Define\_user.h内のflg\_particleを1にする．また，2にするとSGS成分なしの粒子計算．それ以外では，小野寺さんが実装した手法（均等配列ソース）になる．
2. flg\_wallFunctionを1にする（壁関数on）
3. その他，設定等は粒子無のケースと同様にして開始．
4. **出力ファイル**

出力先：./result\_particle\_scatter\_binary

出力ファイルの属性：header, header-node, index, position, uvw, uvw\_sgs, velocity

出力ファイルの表記について：以下のように表記されている．

（属性）＋（ランク番号）－（出力回数）.bin

例）position0-100.bin … ランク０で計算されたpositionの100回目の出力ファイル

出力ファイルの中身について：

バイナリ形式で保存されている．

各属性のファイルでデータ配列の順番は一致している．

つまり，各属性のファイルで最初のデータ，２番目のデータ，３番目…のデータはそれぞれ同じ粒子のデータである．

データの先頭にヘッダー（uint32\_t型）があるので注意する．

Read例：C++（Postprocess/src/calculation.cpp より）

|  |
| --- |
| // Read index and position  182 std::fstream pFile;  183 char posFile[100];  184  185 std::fstream iFile;  186 char idFile[100];  187  188  189 float x, y, z;  190 int id;  191 int n=0;  192  193 for (int rank = 0; rank < num\_rank; rank++) {  194 sprintf(posFile, "./%s/position%d-%d.bin", dir\_data, rank, t);  195 pFile.open(posFile , std::ios::in | std::ios::binary);  196 if (!pFile.is\_open()) {  197 std::cout<< "cannnot open the position file" << std::endl;  198 std::cout<< "File (Rank = " << rank << " / t = " << t << ")" <<std::endl;  199 exit(EXIT\_FAILURE);  200 }  201  202 sprintf(idFile, "./%s/index%d-%d.bin", dir\_data, rank, t);  203 iFile.open(idFile , std::ios::in | std::ios::binary);  204 if (!iFile.is\_open()) {  205 std::cout<< "cannnot open the index file" << std::endl;  206 std::cout<< "File (Rank = " << rank << " / t = " << t << ")" <<std::endl;  207 exit(EXIT\_FAILURE);  208 }  209  210 int i = 1;  211 pFile.seekg(i \* sizeof(int));  212 iFile.seekg(i \* sizeof(int));  213 while (true) {  214 pFile.read((char \*) &x , sizeof(float));  215 if (pFile.eof()) {  216 break;  217 }  218 pFile.read((char \*) &y , sizeof(float));  219 pFile.read((char \*) &z , sizeof(float));  　　　　220 iFile.read((char \*) &id , sizeof(int)); |

**その他**

SGS成分の初速度の指定を実装してから十分にデバッグができていないので問題があれば連絡ください．

また現在，粒子のリスタートができない（リスタート用関数は用意されているがmain関数に実装されていない）ので必要があれば即刻実装します．