学士学位論文

壁の濡れ性による非平衡流体系ダイナミクスの変化

山本 凜 [*1]

最終更新日: 2024 年 2 月 14 日
2023 年度 (令和 5 年)

[*1] 茨城大学 理学部理学科物理学コース 20S2035Y

目次

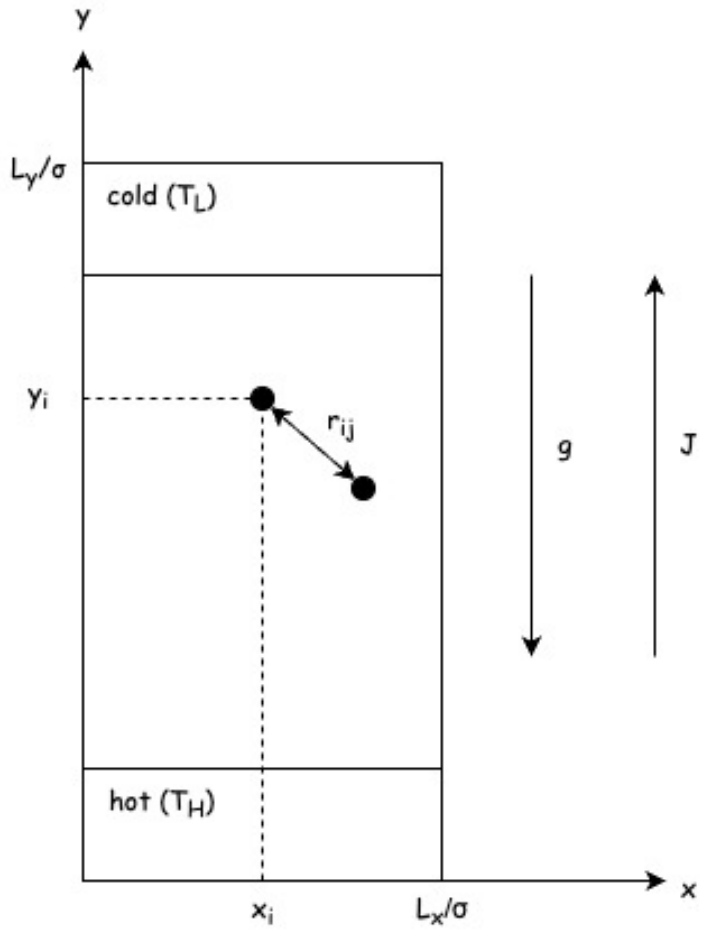
第 1 章

系の設定

この章では, 本研究で扱う系の設定について説明する.

2 次元の気液共存系で, 質量 m の粒子が N 個存在することを考え, 系の上下には壁, 左右には周期境界条件を課す. また, 重力を y 軸負の向きにかけて, 熱流を y 軸正の向きに流す. この熱流は, 系の上下の領域にそれぞれ異なる温度を設定した langevin 熱浴を使用することによってかけることとし, NVT-MD シミュレーションを実行する. また, 各熱浴の y 幅は 8σ となるように設定する. (図 1.1)

図 1.1



1.1 ハミルトニアン

$$H(\Gamma; g) = \sum_{i=1}^{N} \left[\frac{\boldsymbol{p}_i{}^2}{2m} + \sum_{j>i}^{N} \tilde{\phi}_{\mathrm{LJ}}^{\mathrm{pair}}(r_{ij}) + mgy_i + V^{\mathrm{wall}}(y_i) \right] \tag{1.1.0.1}$$

第 1 項から第 4 項まで順に, 運動エネルギー, 粒子-粒子間相互作用ポテンシャル, 重力ポテンシャル, 壁-粒子間相互作用ポテンシャルである. 以降, 本節はこのハミルトニアンに至るまでの過程を述べる.

1.1.1 粒子-粒子間相互作用ポテンシャル

シミュレーションを行う際に, 典型的な粒子間相互作用ポテンシャルとして, 12-6 Lennard-Jones Potential を採用する.

$$\phi_{\mathrm{LJ}}^{\mathrm{pair}}(r;\varepsilon,\sigma) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

シミュレーション上では, カットオフ長 $r_{\mathrm{cut}}^{\mathrm{pair}} = 3\sigma$ とポテンシャルのシフトアップを考慮して

$$\tilde{\phi}_{\mathrm{LJ}}^{\mathrm{pair}}(r;r_{\mathrm{cut}}^{\mathrm{pair}}) = \left\{\phi_{\mathrm{LJ}}^{\mathrm{pair}}(r) - \phi_{\mathrm{LJ}}^{\mathrm{pair}}(r_{\mathrm{cut}}^{\mathrm{pair}})\right\}\theta\left(r_{\mathrm{cut}}^{\mathrm{pair}} - r\right)$$

のように書き換えたポテンシャルを用いている.

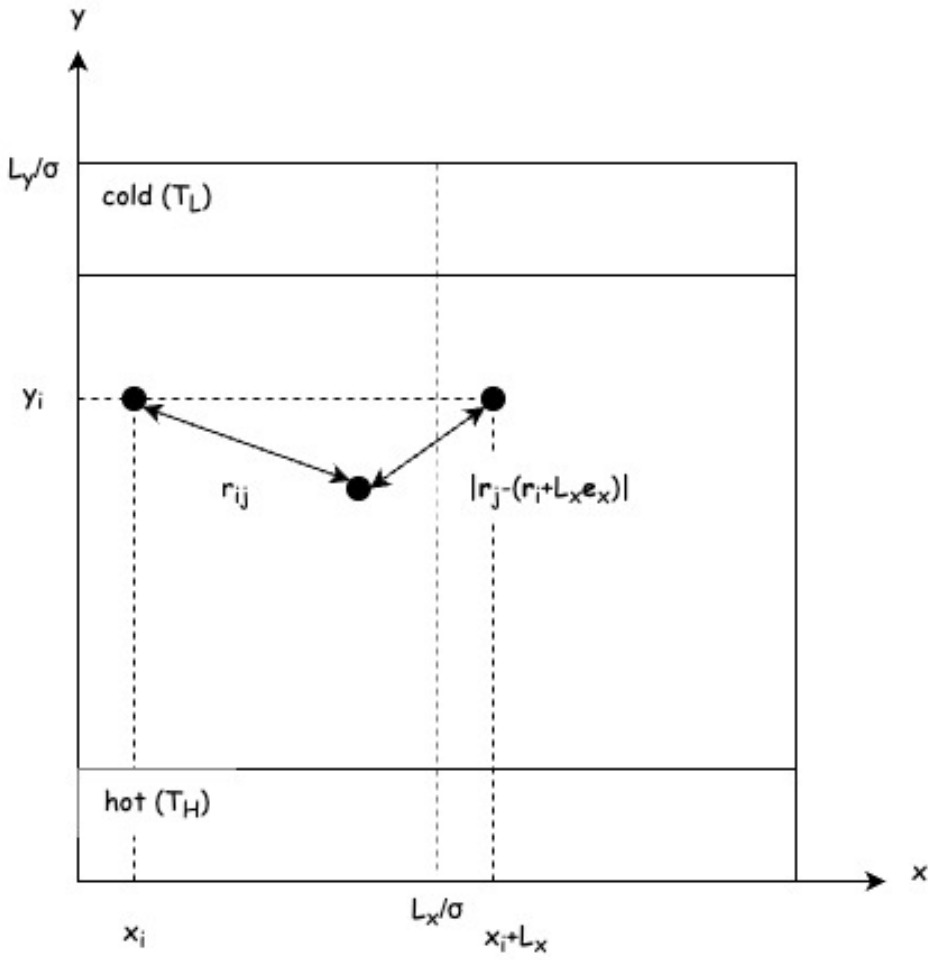
1.1.2 周期境界条件と最近接イメージ規約

周期境界条件を考慮すると, 粒子-粒子間相互ポテンシャルの総計はまず以下のように書ける.[1]

$$\sum_{n_x\in\mathbb{Z}}\sum_{i=1}^{N}\sum_{\substack{j=1\\(j\neq i \text{ for } n_x=0)}}^{N}\frac{1}{2}\phi_{\mathrm{LJ}}^{\mathrm{pair}}(|\boldsymbol{r}_i - (\boldsymbol{r}_j + L_x\boldsymbol{e}_x)|)$$

ここで, $n_x = 0$; オリジナルセルの中では, 同じ i, j ペアのポテンシャルエネルギーを 2 回足すことになるので, ポテンシャルを 1/2 している. その上で, $j = i$ の場合は自分自身との相互作用になるため, これは除外する. $n_x \neq 0$ の場合, 粒子 j はイメージ粒子となるため, $j = i$ の場合も含めることになる. このときにもダブルカウントがあるので, ポテンシャルを 1/2 している.

図 1.2



注目する系の粒子が常にオリジナルセルの中にとどまっているかのように MD 上で扱うには,

$$x_i = x_i' \mod L_x$$

のように, 飛び出した粒子の x 座標 x_i' を上式のように x_i にシフトすれば良い. しかし, 周期境界条件とセットに, 最近接イメージ規約として, 粒子 i がオリジナル粒子と各イメージ粒子の中で最も近い粒子 j らとのみ相互作用をすることを課すと, 粒子間の相互ポテンシャルの総計は先ほどよりも簡単に書けるようになる.

$$\sum_{i=1}^{N}\sum_{j>i}^{N}\phi_{\mathrm{LJ}}^{\mathrm{pair}}(r_{ij})$$

1.1.3 壁-粒子間相互作用ポテンシャル

$$\phi_{\mathrm{LJ}}^{\mathrm{wall}}(r;\varepsilon^{\mathrm{wall}},\sigma^{\mathrm{wall}}) = 4\varepsilon^{\mathrm{wall}}\left[\left(\frac{\sigma^{\mathrm{wall}}}{r}\right)^{12} - \left(\frac{\sigma^{\mathrm{wall}}}{r}\right)^{6}\right]$$

パラメータは以下のようにする.

$$\begin{aligned}\varepsilon^{\mathrm{wall}} &= (1.0 - \mathrm{R_d}) \times \varepsilon \\ \sigma^{\mathrm{wall}} &= (0.5 + \mathrm{R_t}) \times \sigma \\ r_{\mathrm{cut}}^{\mathrm{wall}} &= \left(2^{1/6} + \mathrm{R_a}\right) \times \sigma^{\mathrm{wall}}\end{aligned}$$

カットオフ長とシフトアップを考慮して

$$\tilde{\phi}_{\mathrm{LJ}}^{\mathrm{wall}}(r;r_{\mathrm{cut}}^{\mathrm{wall}}) = \left\{\phi_{\mathrm{LJ}}^{\mathrm{wall}}(r) - \phi_{\mathrm{LJ}}^{\mathrm{wall}}(r_{\mathrm{cut}}^{\mathrm{wall}})\right\}\theta\left(r_{\mathrm{cut}}^{\mathrm{wall}} - r\right)$$

この系では, $y = 0$ と $y = L_y$ に壁がついている. よって, 壁ポテンシャルは

$$V^{\mathrm{wall}}(y;L_y) = \tilde{\phi}_{\mathrm{LJ}}^{\mathrm{wall}}(y;r_{\mathrm{cut}}^{\mathrm{wall}}) + \tilde{\phi}_{\mathrm{LJ}}^{\mathrm{wall}}(L_y - y;r_{\mathrm{cut}}^{\mathrm{wall}})$$

のように書ける. これまでのことより, ハミルトニアンは式 (1.1.0.1) のように書き表せる.

$$H(\Gamma;g) = \sum_{i=1}^{N}\left[\frac{\boldsymbol{p}_i{}^2}{2m} + \sum_{j>i}^{N}\tilde{\phi}_{\mathrm{LJ}}^{\mathrm{pair}}(r_{ij}) + mgy_i + V^{\mathrm{wall}}(y_i)\right] \tag{1.1.0.1}$$

1.2 熱流

粒子 i が熱浴に侵入すると, その粒子の運動はランジュバン方程式に従う. 侵入していないときは, $\gamma = 0$ になり, 正準方程式に等しくなる.

$$
\begin{aligned}
\dot{\boldsymbol{r}}_i &= \frac{\partial H}{\partial \boldsymbol{p}_i} \\
\dot{\boldsymbol{p}}_i &= -\frac{\partial H}{\partial \boldsymbol{r}_i} - \gamma \dot{\boldsymbol{r}}_i + \sqrt{2\gamma k_{\mathrm{B}} T_\nu}\boldsymbol{\xi}_i(t) \\
\langle \xi_i^a(t) \rangle &= 0 \\
\left\langle \xi_i^a(t)\xi_j^b(t') \right\rangle &= \delta_{i,j}\delta_{a,b}\delta(t-t')
\end{aligned}
$$

$$
\begin{aligned}
&\gamma(y_i) = 1.\ T_\nu(y_i) = T_{\mathrm{H}}.\ (0 < y_i < 8\sigma) \\
&\gamma(y_i) = 1.\ T_\nu(y_i) = T_{\mathrm{L}}.\ (L_y - 8\sigma < y_i < L_y) \\
&\gamma(y_i) = 0.\ (8\sigma < y_i < L_y - 8\sigma)
\end{aligned}
$$

付録 A

研究手法

A.1 LAMMPS

A.2 Lennard-Jones potential

A.3 ディレクトリ構造

付録 B

ソースコード

再現しやすいようにソースコードを書き記す．リンク先の GitHub で, Report/src 内にファイルが保存されている．

B.1 LAMMPS ファイル

ソースコード **B.1:** in.2dLJ

```
# 2d Lennard-Jones


# 出力関係のパラメータ
variable run equal 40000000
variable thermo equal ${run}/1000 # 分母の数が
  log で生成される行数になる．
variable dump equal ${run}/1000 # 分母の数が
  lammpstrj で生成される行数になる．
variable image_step equal ${run}/4 # 分母の数 +1枚の画像を作成．

# 重要なパラメータ
variable SEED equal 202035
variable Ay equal 50 # 粒子生成に用いるy 方向でのセル数．
variable Ax equal ${Ay}/2 # 粒子生成に用いるx 方向でのセル数．
variable rho equal PLACEHOLDER_rho # 密度．密度と粒子数から体積が決
  まる．
variable trange equal 5 # 各熱浴の幅．
```

```
16 variable gap equal 0.5 # box と catom のずれ．ずらさないと粒子が消えてしまう．
17 # lo,hi が単に座標の小さい大きいであることに注意．
18 variable T equal 0.43 # 各熱浴の目標温度の中間，これを初期温度に設定．
19 variable dT equal 0.02
20 variable thot equal ${T}+(${dT}/2) # 座標の小さい方の熱浴の目標温度．
21 variable tcold equal ${T}-(${dT}/2) # 座標の大きい方の熱浴の目標温度．
22 variable g equal 0.0004 # 重力加速度．
23 # 粒子-粒子間のLJ ポテンシャル
24 variable epsilon_pair equal 1.0 # LJ ポテンシャルの epsilon; ポテンシャルの深さ．
25 variable sigma_pair equal 1.0 # LJ ポテンシャルの sigma; 衝突直径．
26 variable rc_pair equal 3.0 # 典型的なカットオフ長．
27 # 壁-粒子間のLJ ポテンシャル
28 variable Rd equal 0.0 # 乾き具合．
29 variable Rt equal 0.5 # 壁の厚み．
30 variable Ra equal 1.877538 # 濡れ具合．
31 variable epsilon_wall equal 1.0-${Rd} # LJ ポテンシャルの epsilon; ポテンシャルの深さ．
32 variable sigma_wall equal 0.5+${Rt} # LJ ポテンシャルの sigma; 衝突直径．
33 variable rc_wall equal 1.122462+${Ra} #
     WCA ポテンシャルになるようなカットオフ長+alpha*sigma_wall．
34
35 # 領域関係のパラメータ
36   # 縦長のとき
37 variable box_xlo equal 0 # x の小さい方の直線．
38 variable box_xhi equal ${Ax} # x の大きい方の直線．
39 variable box_ylo equal -${gap} # y の小さい方の直線．
40 variable box_yhi equal ${Ay}-${gap} # y の大きい方の直線．
41 variable hotlo equal -${gap} # 熱浴で温度の低い方の小さい方の直線．
42 variable hothi equal -${gap}+${trange} # 熱浴で温度の低い方の大きい方の直線．
43 variable coldlo equal ${Ay}-${gap}-${trange} # 熱浴で温度の高い方の小さい方の直線．
44 variable coldhi equal ${Ay}-${gap} # 熱浴で温度の高い方の大きい方の直線．
```

```
45
46
47 # 系の設定
48 units  lj # LJ 単位系.
49 atom_style atomic # 粒子.
50 dimension 2 # 次元.
51 timestep 0.005 # MD シミュレーションの timestep.
52 boundary p f p # x=l,y=m,z=n の直線が周期境界条件.
53 lattice  sq ${rho} # 粒子の初期配置. sq; 正方形セルの左隅に 1つ置く.
54 region  box block ${box_xlo} ${box_xhi} ${box_ylo} ${box_yhi}
     -0.1 0.1 # 系の領域設定.
55 region catom block 0 ${Ax} 0 ${Ay} -0.1 0.1 # 粒子生成の領域設定.
56 create_box 1 box # 系の生成.
57 create_atoms 1 region catom # 粒子の生成.
58 mass  1 1.0 # 粒子の設定.
59 velocity all create ${T} ${SEED} dist gaussian # 粒子に温度
     t を目標とする初期速度をガウス分布に従って与える.
60
61   # 縦長のとき
62 region cold block INF INF ${coldlo} ${coldhi} -0.1 0.1 # 熱浴
     C の領域.
63 region hot block INF INF ${hotlo} ${hothi} -0.1 0.1 # 熱浴H の領域
     .
64
65 # 各熱浴領域の温度を計算
66 compute Tcold all temp/region cold #
     c_Tcold で cold 熱浴領域の温度を取得.
67 compute Thot all temp/region hot #
     c_Tcold で cold 熱浴領域の温度を取得.
68
69 # 粒子-粒子間相互作用ポテンシャル
70 pair_style lj/cut ${rc_pair}
71 pair_coeff 1 1 ${epsilon_pair} ${sigma_pair} ${rc_pair}
72 pair_modify shift yes # ポテンシャルエネルギーが 0になる距離がカットオ
     フ長になるように全体的にシフトアップする.
73
74 # 高速化コマンド. neighbor list に入れる距離指定.
75 neighbor 0.3 bin
```

```
76 neigh_modify every 1 delay 0 check yes
77
78 # 系に他の操作がない場合にnve アンサンブルに一致するだけであり，今回の系
     はlangevin 熱浴を用いた nvt アンサンブルであることに注意.
79 fix  1 all nve
80
81 # 壁-粒子間相互作用ポテンシャル
82 # 縦長のとき
83 fix wallylo all wall/lj126 ylo EDGE ${epsilon_wall} ${sigma_wall
     } ${rc_wall} units box pbc yes
84 fix wallyhi all wall/lj126 yhi EDGE ${epsilon_wall} ${sigma_wall
     } ${rc_wall} units box pbc yes
85
86 # langevin 熱浴
87 fix hot all langevin ${T} ${T} 1.0 ${SEED} tally no # 熱浴
     H が温度 T になるようにする.
88 fix cold all langevin ${T} ${T} 1.0 ${SEED} tally no # 熱浴
     C が温度 T になるようにする.
89 fix_modify hot temp Thot
90 fix_modify cold temp Tcold
91
92 # 重力場
93 fix Gravity all gravity ${g} vector 0 -1 0
94
95 # 重力を熱流をより先にかけるときコメントアウト解除.
96 # run 200000 # 重力のみでの平衡までの緩和時間
97 # unfix hot # 熱浴H についての設定の解除.
98 # unfix cold # 熱浴C についての設定の解除.
99
100 fix hot all langevin ${thot} ${thot} 1.0 ${SEED} tally no # 熱浴
     が温度tlo になるようにする.
101 fix cold all langevin ${tcold} ${tcold} 1.0 ${SEED} tally no #
     熱浴が温度thi になるようにする.
102 fix_modify hot temp Thot
103 fix_modify cold temp Tcold
104
105 # 重心計算 (Center of Mass)
106 compute CoM all com # c_CoM[1]でXg, c_CoM[2]でYg を取得.
```

```
107
108
109 # 出力コマンド
110 # VMD
111 dump  id all custom ${dump} output.lammpstrj id x y vx vy
112
113 # 画像
114 dump  2 all image ${image_step} image.*.jpg type type
115 dump_modify 2 pad 3
116
117 # log
118 thermo_style custom step time temp pe ke etotal c_CoM[2] # 出力す
      る物理量.
119
120 # YAML
121 fix extra all print ${thermo} """
122 -␣step:␣$(step)
123 ␣␣time:␣$(time)
124 ␣␣temp:␣$(temp)
125 ␣␣ke:␣$(ke)
126 ␣␣pe:␣$(pe)
127 ␣␣etotal:␣$(etotal)
128 ␣␣Yg:␣$(c_CoM[2])""" file output.yaml screen no
129
130 # # 一次元プロファイル(今は温度と密度だけ計算と出力)
131 # compute chunk all chunk/atom bin/1d y lower 3.0 units box
132 # fix tempp all ave/chunk 100000 1 100000 chunk temp file
      temp_profile.profile
133 # fix rhop all ave/chunk 100000 1 100000 chunk density/number
      file rho_profile.profile
134
135 thermo  ${thermo} # 熱力学量の出力.
136 thermo_modify norm no # 示量的な熱力学量に調整.
137
138 run  ${run} # 実行.
```

ソースコード B.2: in.2dLJ_mod

```
# 2d Lennard-Jones


# 出力関係のパラメータ
variable run equal PLACEHOLDER_run
variable thermo equal ${run}/1000 # 分母の数だけ出力.(log, yaml)
variable dump equal ${run}/1000 # 分母の数だけ出力.(lammpstrj)
variable image_step equal ${run}/1 # 分母の数+1枚の画像を作成.

# 重要なパラメータ
variable SEED equal 202035
variable Ay equal PLACEHOLDER_Ay # 粒子生成に用いるy方向でのセル数.
variable Ax equal ${Ay}/2 # 粒子生成に用いるx方向でのセル数.
variable rho equal PLACEHOLDER_rho # 密度. 密度と粒子数から体積が決まる.
variable trange equal 5 # 各熱浴の幅.
variable gap equal 0.5 # boxとcatomのずれ. ずらさないと粒子が消えてしまう.
variable T equal PLACEHOLDER_T # 各熱浴の目標温度の中間, これを初期温度に設定.
variable dT equal PLACEHOLDER_dT
variable thot equal ${T}+(${dT}/2) # 座標の小さい方の熱浴の目標温度.
variable tcold equal ${T}-(${dT}/2) # 座標の大きい方の熱浴の目標温度.
variable g equal PLACEHOLDER_g # 重力加速度.
# 粒子-粒子間のLJポテンシャル
variable epsilon_pair equal 1.0 # LJポテンシャルのepsilon; ポテンシャルの深さ.
variable sigma_pair equal 1.0 # LJポテンシャルのsigma; 衝突直径.
variable rc_pair equal 3.0 # 典型的なカットオフ長.
# 壁-粒子間のLJポテンシャル
variable Rd equal PLACEHOLDER_Rd # 乾き具合.
variable Rt equal PLACEHOLDER_Rt # 壁の厚み.
variable Ra equal PLACEHOLDER_Ra # 濡れ具合.
variable epsilon_wall equal 1.0-${Rd} # LJポテンシャルのepsilon; ポテンシャルの深さ.
variable sigma_wall equal 0.5+${Rt} # LJポテンシャルのsigma; 衝突直径.
```

```
32 variable rc_wall equal 1.122462+${Ra} #
     WCA ポテンシャルになるようなカットオフ長+alpha*sigma_wall.
33
34 # 領域関係のパラメータ
35   # 縦長のとき
36 variable box_xlo equal 0 # x の小さい方の直線.
37 variable box_xhi equal ${Ax} # x の大きい方の直線.
38 variable box_ylo equal -${gap} # y の小さい方の直線.
39 variable box_yhi equal ${Ay}-${gap} # y の大きい方の直線.
40 variable hotlo equal -${gap} # 熱浴で温度の低い方の小さい方の直線.
41 variable hothi equal -${gap}+${trange} # 熱浴で温度の低い方の大きい
     方の直線.
42 variable coldlo equal ${Ay}-${gap}-${trange} # 熱浴で温度の高い方
     の小さい方の直線.
43 variable coldhi equal ${Ay}-${gap} # 熱浴で温度の高い方の大きい方の
     直線.
44
45
46 # 系の設定
47 units  lj # LJ 単位系.
48 atom_style atomic # 粒子.
49 dimension 2 # 次元.
50 timestep 0.005 # MD シミュレーションの timestep.
51 boundary p f p # x=l,y=m,z=n の直線が周期境界条件.
52 lattice  sq ${rho} # 粒子の初期配置. sq; 正方形セルの左隅に 1つ置く.
53 region  box block ${box_xlo} ${box_xhi} ${box_ylo} ${box_yhi}
     -0.1 0.1 # 系の領域設定.
54 region catom block 0 ${Ax} 0 ${Ay} -0.1 0.1 # 粒子生成の領域設定.
55 create_box 1 box # 系の生成.
56 create_atoms 1 region catom # 粒子の生成.
57 mass  1 1.0 # 粒子の設定.
58 velocity all create ${T} ${SEED} dist gaussian # 粒子に温度
     t を目標とする初期速度をガウス分布に従って与える.
59
60   # 縦長のとき
61 region cold block INF INF ${coldlo} ${coldhi} -0.1 0.1 # 熱浴
     C の領域.
62 region hot block INF INF ${hotlo} ${hothi} -0.1 0.1 # 熱浴H の領域
```

```
  .
63
64 # 各熱浴領域の温度を計算
65 compute Tcold all temp/region cold #
     c_Tcold で cold 熱浴領域の温度を取得.
66 compute Thot all temp/region hot #
     c_Tcold で cold 熱浴領域の温度を取得.
67
68 # 粒子-粒子間相互作用ポテンシャル
69 pair_style lj/cut ${rc_pair}
70 pair_coeff 1 1 ${epsilon_pair} ${sigma_pair} ${rc_pair}
71 pair_modify shift yes # ポテンシャルエネルギーが 0になる距離がカットオ
     フ長になるように全体的にシフトアップする.
72
73 # 高速化コマンド. neighbor list に入れる距離指定.
74 neighbor 0.3 bin
75 neigh_modify every 1 delay 0 check yes
76
77 # 系に他の操作がない場合にnve アンサンブルに一致するだけであり，今回の系
     はlangevin 熱浴を用いた nvt アンサンブルであることに注意.
78 fix  1 all nve
79
80 # 壁-粒子間相互作用ポテンシャル
81 # 縦長のとき
82 fix wallylo all wall/lj126 ylo EDGE ${epsilon_wall} ${sigma_wall
     } ${rc_wall} units box pbc yes
83 fix wallyhi all wall/lj126 yhi EDGE ${epsilon_wall} ${sigma_wall
     } ${rc_wall} units box pbc yes
84
85 # langevin 熱浴
86 fix hot all langevin ${T} ${T} 1.0 ${SEED} tally no # 熱浴
     H が温度 T になるようにする.
87 fix cold all langevin ${T} ${T} 1.0 ${SEED} tally no # 熱浴
     C が温度 T になるようにする.
88 fix_modify hot temp Thot
89 fix_modify cold temp Tcold
90
91 # 重力場
```

```
92 fix Gravity all gravity ${g} vector 0 -1 0
93
94 # # 重力を熱流より先にかける時にコメントアウトを解除.
95 # run 40000000 # 重力のみでの平衡までの緩和時間
96 unfix hot # 熱浴H についての設定の解除.
97 unfix cold # 熱浴C についての設定の解除.
98
99 fix hot all langevin ${thot} ${thot} 1.0 ${SEED} tally no # 熱浴
     が温度tlo になるようにする.
100 fix cold all langevin ${tcold} ${tcold} 1.0 ${SEED} tally no #
     熱浴が温度thi になるようにする.
101 fix_modify hot temp Thot
102 fix_modify cold temp Tcold
103
104 # 重心計算 (Center of Mass)
105 compute CoM all com # c_CoM[1]でXg, c_CoM[2]でYg を取得.
106
107
108 # 出力コマンド
109 # lammpstrj
110 dump  id all custom ${dump} PLACEHOLDER_outputtitle.lammpstrj id
      x y vx vy
111
112 # # 画像を生成するならコメントアウトを解除.
113 # dump  2 all image ${image_step} image.*.jpg type type
114 # dump_modify 2 pad 3
115
116 # log
117 thermo_style custom step time temp pe ke etotal c_CoM[2] # 出力す
     る物理量.
118
119 # YAML
120 fix extra all print ${thermo} """
121 -␣step:␣$(step)
122 ␣␣time:␣$(time)
123 ␣␣temp:␣$(temp)
124 ␣␣ke:␣$(ke)
125 ␣␣pe:␣$(pe)
```

```
␣␣etotal:␣$(etotal)
␣␣Yg:␣$(c_CoM[2])""" file PLACEHOLDER_outputtitle.yaml screen no

# # 一次元プロファイル(今は温度と密度だけ計算と出力)
# compute chunk all chunk/atom bin/1d y lower 3.0 units box
# fix tempp all ave/chunk 100000 1 100000 chunk temp file
  temp_profile.profile
# fix rhop all ave/chunk 100000 1 100000 chunk density/number
  file rho_profile.profile

thermo  ${thermo} # 熱力学量の出力.
thermo_modify norm no # 示量的な熱力学量に調整.

run  ${run} # 実行.
```

B.2 実行ファイル

ソースコード **B.3:** lammps_modexe.jl

```
#===
# LAMMPS シミュレーション実行と出力ファイル保管

このJulia コードは,LAMMPS 分子動力学シミュレーションを実行し,生成された
  出力ファイルを適切なディレクトリに保存します.

## 機能

- ‘Glob‘と‘Dates‘パッケージを使用してファイルマッチングと日時取得を行う
- パラメータを配列で定義
- LAMMPS ファイル内のプレースホルダーをパラメータ値に置き換えて,実行用ス
  クリプトを生成
- パラメータごとにLAMMPS を実行し,生成された出力ファイルを指定のディレク
  トリに移動
- 使用済みの仮LAMMPS ファイルを一括削除

## 手順

```

```
1．LAMMPS ファイルの特定とパラメータ設定
2．パラメータの範囲を定義
3．パラメータの組み合わせごとにLAMMPS を実行
4．出力ファイルを指定ディレクトリに保存
5．使用済みの仮LAMMPS ファイルを削除

このコードは,異なるパラメータでのLAMMPS シミュレーションを自動化し,出力
  ファイルの整理と保管を行います.
===#

using Glob # ファイルパターンのマッチングに使用するパッケージ
using Dates # 日付と時刻の取得に使用するパッケージ

# 実行するLAMMPS ファイルを特定
lammpsfile = glob("in.*")[1]

# パラメータの設定
chi = 1.265
remark_text = "test"
file_extensions = ["log", "yaml", "lammpstrj"] # 出力ファイルの拡
  張子

# パラメータの範囲を設定
Ay_range = range(50, length=1) # Ay の範囲
rho_range = range(0.4, length=1) # 密度の範囲
T_range = range(0.43, length=1) # 初期温度の範囲
dT_range = range(0.04, length=1) # 熱浴の温度差の範囲
Rd_range = range(0.0, length=1) # 乾燥度の範囲
Rt_range = range(0.0, 0.5, length=5) # 壁の厚みの範囲
Ra_range = range(0.0, 1.877538, length=5) # 引力幅の範囲
run_range = range(4e7, length=1) # 実行ステップ数の範囲

# パラメータごとに実験を実行
for Ay_value in Ay_range,
    rho_value in rho_range,
    T_value in T_range,
    dT_value in dT_range,
    Rd_value in Rd_range,
```

```
    Rt_value in Rt_range,
    Ra_value in Ra_range,
    run_value in run_range

    # パラメータに基づいて重力を計算
    g_value = dT_value / ((Ay_value / sqrt(rho_value)) * chi)

    # 実験日時を記録
    n = string(now())

    # パラメータに基づいた出力ファイル名を生成
    parameter = "chi$(chi)_Ay$(Ay_value)_rho$(rho_value)_T$(
      T_value)_dT$(dT_value)_Rd$(Rd_value)_Rt$(Rt_value)_Ra$(
      Ra_value)_g$(g_value)_run$(run_value)"
    outputtitle = "$(n)_$(remark_text)_$(parameter)"

    # LAMMPS ファイルの内容を読み込み,パラメータを置換
    template_script = read(lammpsfile, String)
    mod_script = replace(template_script,
        "PLACEHOLDER_Ay" => string(Ay_value),
        "PLACEHOLDER_rho" => string(rho_value),
        "PLACEHOLDER_T" => string(T_value),
        "PLACEHOLDER_dT" => string(dT_value),
        "PLACEHOLDER_g" => string(g_value),
        "PLACEHOLDER_Rd" => string(Rd_value),
        "PLACEHOLDER_Rt" => string(Rt_value),
        "PLACEHOLDER_Ra" => string(Ra_value),
        "PLACEHOLDER_run" => string(run_value),
        "PLACEHOLDER_outputtitle" => string(outputtitle)
    )

    # 一意のファイル名を生成して仮ファイルを作成し,パラメータを書き込む
    tempfile = "in.temp_script_$(n)"
    fp = open(tempfile, "w")
    write(fp, mod_script)
    close(fp)

    # LAMMPS を実行
```

```
    run(`mpirun -n 4 lmp_mpi -log $(outputtitle).log -in $(
      tempfile)`)

    # 出力ファイルを保存
    for file_ext in file_extensions
        files = glob("*.$(file_ext)")
        for file in files
            outputpath = "../outputdir/$(file_ext)dir"
            mkpath(outputpath)
            script = read(file, String)
            fp = open(joinpath(outputpath, "$(file)"), "w")

            if file_ext == "log"
                println(fp, "$(file)")
            end

            write(fp, script)
            close(fp)
            rm(file)
        end
    end
end

# 使用済みの仮LAMMPS ファイルを一括削除
files = glob("in.temp_*")
for file in files
    rm(file)
end
```

ソースコード B.4: lammps_qsub_job.jl

```
#===
# LAMMPS シミュレーション実行

このJulia コードは、LAMMPS 分子動力学シミュレーションを実行します.

## 機能

```

```
- ‘Glob‘と‘Dates‘パッケージを使用してファイルマッチングと日時取得を行う.
- パラメータを配列で定義.
- LAMMPS ファイル内のプレースホルダーをパラメータ値に置き換えて実行用スクリプトを生成.
- パラメータごとにLAMMPS を実行.

## 手順

1. LAMMPS ファイルの特定とパラメータ設定.
2. パラメータの範囲を定義.
3. パラメータの組み合わせごとにLAMMPS を実行.

このコードは、異なるパラメータでのLAMMPS シミュレーションを自動化します.
===#


using Glob # *を使ってパターンマッチングするためのパッケージ.
using Dates # 日時を取得するパッケージ.

# 実行するLAMMPS ファイルを特定
lammpsfile = glob("in.*")[1]

# パラメータの設定
chi = 1.265
remark_text = "test"

# パラメータの範囲を設定
Ay_range = range(100, length=1) # Ay の範囲
rho_range = range(0.4, length=1) # 密度の範囲
T_range = range(0.43, length=1) # 初期温度の範囲
dT_range = range(0.0, length=1) # 熱浴の温度差の範囲
Rd_range = range(0.0, length=1) # 乾燥度の範囲
Rt_range = range(0.5, length=1) # 壁の厚みの範囲
Ra_range = range(0.0, 1.877538, length=5) # 濡れ具合の範囲
run_range = range(4e7, length=1) # 実行ステップ数の範囲

```

```
# 多重ループを用いてパラメータごとに実験を実行.
for Ay_value in Ay_range,
    rho_value in rho_range,
    T_value in T_range,
    dT_value in dT_range,
    Rd_value in Rd_range,
    Rt_value in Rt_range,
    Ra_value in Ra_range,
    run_value in run_range

    # パラメータに基づいて重力を計算
    g_value = dT_value / ((Ay_value / sqrt(rho_value)) * chi)

    # 実験日時を記録
    n = string(now())

    # パラメータに基づいた出力ファイル名を生成
    parameter = "chi$(chi)_Ay$(Ay_value)_rho$(rho_value)_T$(
      T_value)_dT$(dT_value)_Rd$(Rd_value)_Rt$(Rt_value)_Ra$(
      Ra_value)_g$(g_value)_run$(run_value)"
    outputtitle = "$(n)_$(remark_text)_$(parameter)"

    # LAMMPS ファイルの内容を読み込み、パラメータを置換
    template_script = read(lammpsfile, String)
    mod_script = replace(template_script,
        "PLACEHOLDER_Ay" => string(Ay_value),
        "PLACEHOLDER_rho" => string(rho_value),
        "PLACEHOLDER_T" => string(T_value),
        "PLACEHOLDER_dT" => string(dT_value),
        "PLACEHOLDER_g" => string(g_value),
        "PLACEHOLDER_Rd" => string(Rd_value),
        "PLACEHOLDER_Rt" => string(Rt_value),
        "PLACEHOLDER_Ra" => string(Ra_value),
        "PLACEHOLDER_run" => string(run_value),
        "PLACEHOLDER_outputtitle" => string(outputtitle)
    )

    # 一意のファイル名を生成して仮ファイルを作成し、パラメータを書き込む
```

```
79      tempfile = "in.temp_script_$(n)"
80      fp = open(tempfile, "w")
81      write(fp, mod_script)
82      close(fp)
83
84      # LAMMPS を実行
85      run(‘myqsub -Q ness -C 4 -N g0Ra$(Ra_value) mpirun -n 4
          lmp_mpi -log $(outputtitle).log -in $(tempfile)‘)
86
87  end
```

ソースコード **B.5:** lammps_qsub_out.jl

```
1   #===
2   # 出力ファイル保管
3
4   このJulia コードは,
      LAMMPS 分子動力学シミュレーションの実行によって生成された出力ファイルを適切なディレク
      .
5
6   ## 機能
7
8   - ‘Glob‘パッケージを使用してファイルマッチングを行う
9   - パラメータごとに
      LAMMPS を実行したことによって生成された出力ファイルを指定のディレクトリに移動
      .
10  - 使用済みの仮LAMMPS ファイルを一括削除
11
12  ## 手順
13
14  1. 出力ファイルを指定ディレクトリに保存
15  2. 使用済みの仮LAMMPS ファイルを削除
16
17  このコードは,出力ファイルの整理と保管を行います.
18  ===#
19
20  using Glob # *を使ってパターンマッチングするためのパッケージ.
21
```

```
file_extensions = ["log", "yaml", "lammpstrj"] # 出力ファイルの拡張子

# 出力ファイルを保存
for file_ext in file_extensions
    files = glob("*.$(file_ext)")
    for file in files
        outputpath = "../outputdir/$(file_ext)dir"
        mkpath(outputpath)
        script = read(file, String)
        fp = open(joinpath(outputpath, "$(file)"), "w")

        if file_ext == "log"
            println(fp, "$(file)")
        end

        write(fp, script)
        close(fp)
        rm(file)
    end
end

# 使用済みの仮LAMMPS ファイルを一括削除
files = glob("in.temp_*")
for file in files
    rm(file)
end
```

B.3 プロットファイル

ソースコード **B.6:** plot_LJpotential.jl

```
# 汎用LJ ポテンシャル描画セル.
# パッケージ.
using Plots

# 関数定義.
```

```
function theta(r) # 階段関数.
    return r > 0 ? 1 : 0
end
function phi(epsilon, sigma, r) # LJ ポテンシャル.
    return 4.0 * epsilon * ((sigma/r)^12 - (sigma/r)^6)
end
function phi_tilde(r, epsilon, sigma, rc) # シフトアップとカットオフ.
    return (phi(epsilon, sigma, r) - phi(epsilon, sigma, rc)) *
      theta(rc - r)
end

# 粒子-粒子LJ ポテンシャルのパラメータ.
epsilon = 1.0
sigma = 1.0
rc = 3.0 * sigma

# Rd, Rt, Ra の配列.
Rd_values = range(0.0, length=1)
Rt_values = range(0.0,0.5, length=5)
Ra_values = range(1.877, length=1)

# プロット概形.
plot(xlabel="r/σ", ylabel="^^cf^^95/ε")
xlims!(0.2,2.5)
ylims!(-1.5,3.0)
title!("LJ-Potential␣vs.␣r")
xlabel!("r/σ")
ylabel!("^^cf^^95/ε")

# 粒子-粒子LJ ポテンシャルのプロット.
plot!(r -> phi_tilde(r, epsilon, sigma, rc), label="
  Potential_pair;␣ε=$(round(epsilon,digits=1)),␣σ=$(round(sigma
  ,digits=1)),␣rc=$(round(3.0,digits=2))σ", linestyle=:dash)

# プロットの追加.
for Rd in Rd_values,
    Rt in Rt_values,
```

```
    Ra in Ra_values
    # 壁-粒子LJ ポテンシャルのパラメータ.
    epsilon_wall = (1.0 - Rd) * epsilon
    sigma_wall = (0.5 + Rt) * sigma
    rc_wall = ((2 ^ (1 / 6)) + Ra) * sigma_wall
    # 打つ点を調整.
    x_values = range(Rt+0.3,3.0,length=10000)
    y_values = phi_tilde.(x_values, epsilon_wall, sigma_wall,
      rc_wall)
    # 壁-粒子LJ ポテンシャルのプロット.
    plot!(x_values, y_values, label="Potential_wall;␣εw=$(round(
      epsilon_wall,digits=1))ε,␣σw=$(round(sigma_wall,digits
      =1))σ,␣rcw=$(round((2^(1/6))␣+␣Ra,digits=2))σw=$(round
      (((2^(1/6))␣+␣Ra)*sigma_wall,digits=2))σ", linestyle=:dash
      )
end

display(plot!())
# savefig("")

ccall(:jl_tty_set_mode, Int32, (Ptr{Cvoid}, Int32), stdin.handle
  , true)
read(stdin, 1)
```

参考文献

[1] 渡邉孝信. 分子動力学法と原子間ポテンシャル. 森北出版, 2023.