

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 15 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 実験の設定	2
1.1 重力と熱流を同時にかける	7
1.2 重力を先にかけて、熱流を後からかける	10
1.3 重力のみをかける	11
1.4 熱流のみをかける	12
1.5 重力と熱流を同時にかける(時間 10 倍)	14
1.6 重力を先にかけて、熱流を後からかける(時間 10 倍)	17
第 2 章 分析と考察	18
2.1 重心位置	18
2.2 空間的なばらつき	22
2.3 リミットサイクル	24
2.4 リミットサイクル 3D	26
2.5 ヒートマップ	27
付録 A ソースコード	28
A.1 LAMMPS ファイル	28
A.2 実行ファイル	37
A.3 プロットファイル	44
参考文献	47

第 1 章

実験の設定

この章では、行ったシミュレーションの設定についてそれぞれ説明をしていく。
系の上下両端のポテンシャルエネルギー差 mgL_y と運動エネルギー差 $k_B\Delta T$ の比を χ として以下のように設定する。

$$\chi \equiv \frac{k_B\Delta T}{mgL_y}$$

壁の濡れ性を制御する無次元パラメータを 2 つ用意する。

R_t : 壁の厚み.

R_a : 引力幅.

これらを用いて、壁-粒子間相互作用 LJ ポテンシャルは以下のように書き表す。

$$\begin{aligned}\sigma^{\text{wall}} &= (0.5 + R_t) \times \sigma, \\ r_{\text{cut}}^{\text{wall}} &= \left(2^{1/6} + R_a\right) \times \sigma^{\text{wall}}\end{aligned}$$

パラメータ (R_t, R_a) を変えることによって、壁-粒子間相互作用 LJ ポテンシャルが変わることによって、壁-粒子間相互作用 LJ ポテンシャルが変わる。このときに、どのようにして粒子集団の様相が変化するかを見る。

本論では R_t と R_a を少しづつ変えた系でシミュレーションをして、粒子集団の様相の変化を見たい。以下に示すのが、それらを動かす範囲である。

$$R_t : 0.0 \sim 0.5$$

$$R_a : 0.0 \sim 3.0 - 2^{1/6} = 1.877538\dots$$

数値実験上で実際に入力する値を以下に示す。本論では簡単のため簡略して示すことがある。

$$R_t = 0.0, 0.125, 0.25, 0.375, 0.5$$

$$R_a = 0.0, 0.4693845, 0.938769, 1.4081535, 1.877538$$

以下は R_t , R_a を変化させたときの LJ ポテンシャルがどのように変化するのかを可視化したグラフである。

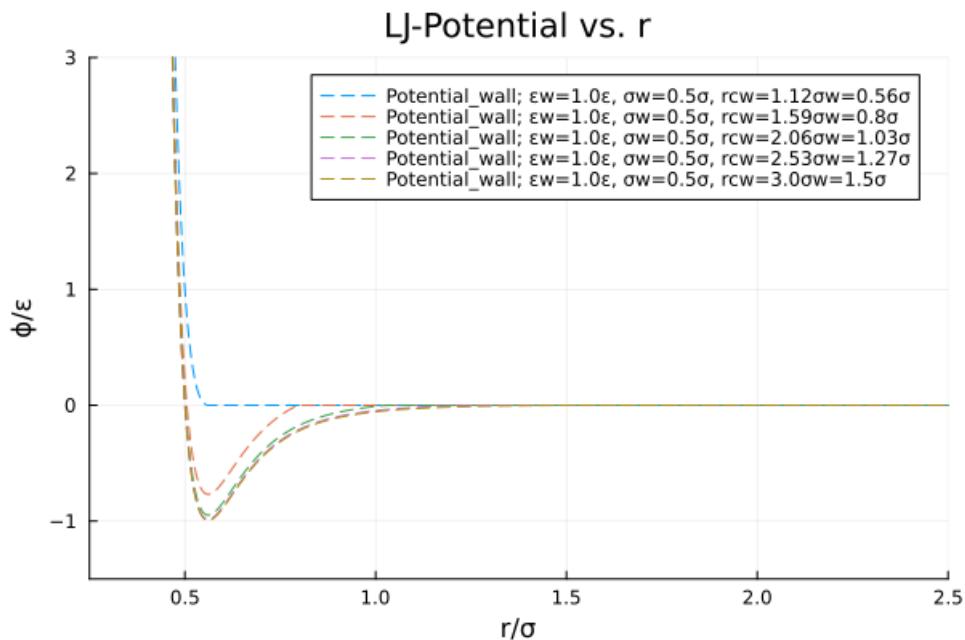


図 1.1: $R_t : 0.0$

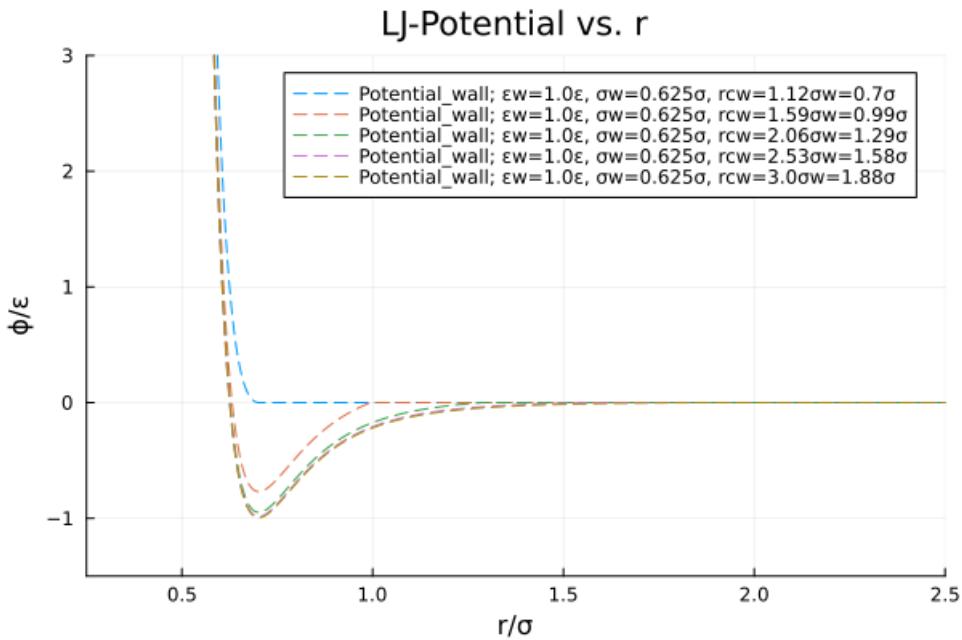


図 1.2: $R_t : 0.125$

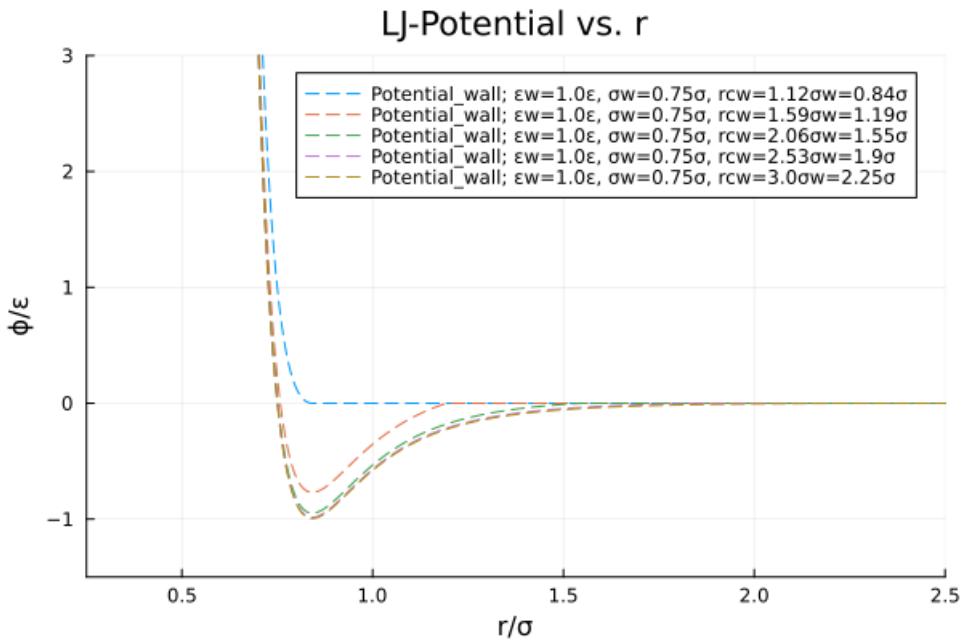


図 1.3: $R_t : 0.25$

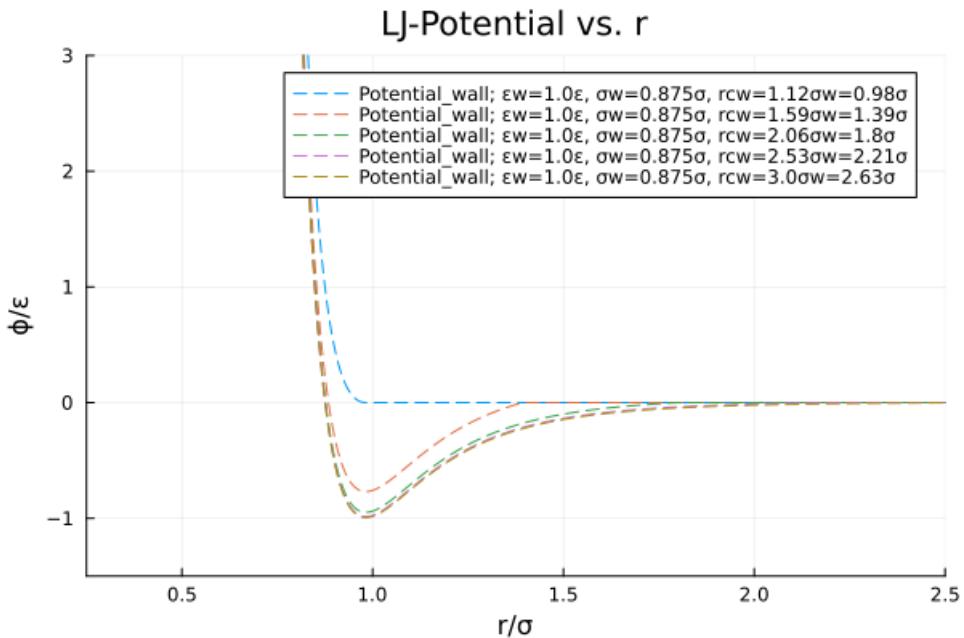


図 1.4: $R_t : 0.375$

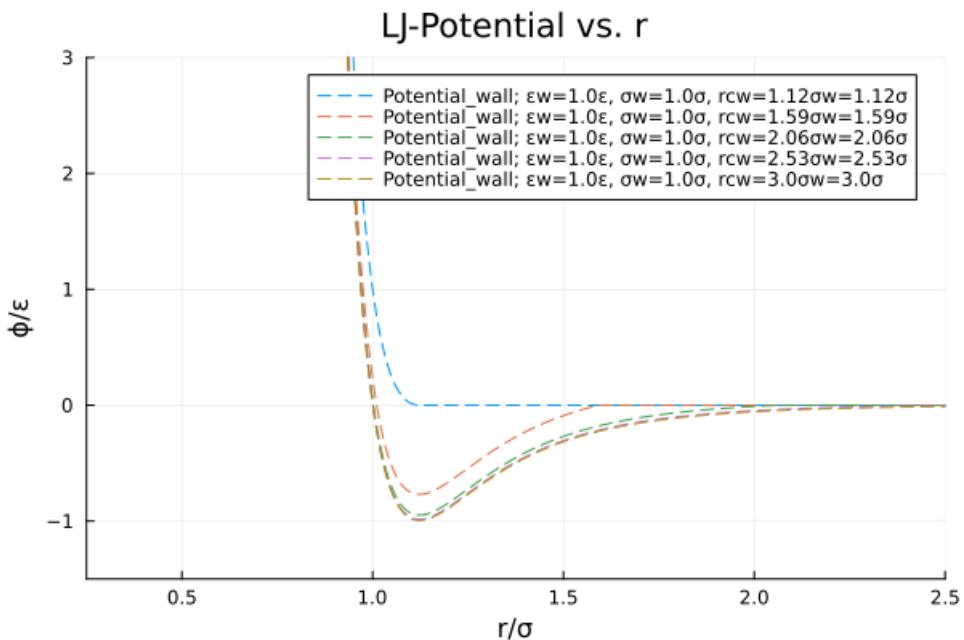


図 1.5: $R_t : 0.5$

本章の以降の実験は特記がない限り以下のパラメータで行うものとする.

- $N = 1250$: 粒子数

- $\rho\sigma^2 = 0.4$: 粒子数密度
- $L_x/\sigma = 39.528471 \simeq 39.5$: 系の x 幅
- $L_y/\sigma = 79.0569414 \simeq 79.0$: 系の y 幅
- $k_B T/\varepsilon = 0.43$: 初期温度
- $k_B \Delta T/\varepsilon = 0.04$: 热浴の温度差
- $mg\sigma/\varepsilon = 0.0003999718779659611 \simeq 4.0 \times 10^{-4}$: 粒子にかかる重力の大きさ
- $dt\sqrt{\epsilon/m\sigma^2} = 0.005$: シミュレーションにおける時間刻み.

以下に記すのは、今後解析をする際に示すシミュレーションについての時間に関する説明である。

- t_i : シミュレーション開始時から、物理量を解析する際にデータを採用し始める時間。これ以降は定常状態であるとみなす。
- t_f : シミュレーション開始時から、シミュレーションの終了時までの時間。

いずれの実験の場合も $t\sqrt{\epsilon/m\sigma^2} = 0$ の時点では粒子は以下の画像のように、系に規則正しく並べられているとする。

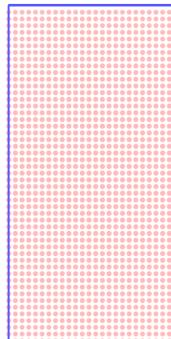


図 1.6: $N = 1250$

1.1 重力と熱流を同時にかける

以下のように, R_a と R_t を少しずつ変えた系を設定して, それぞれシミュレーションをした.

	$R_a : 0.0$	$R_a : 0.4693$	$R_a : 0.9387$	$R_a : 1.408$	$R_a : 1.877$
$R_t : 0.0$	a	b	c	d	e
$R_t : 0.125$	f	g	h	i	j
$R_t : 0.25$	k	l	m	n	o
$R_t : 0.375$	p	q	r	s	t
$R_t : 0.5$	u	v	w	x	y

参考のために, 各パラメータの値を変えることによって, カットオフ長と衝突直径がどのように変わることかを示す.

$$R_a = 0.0 \Rightarrow r_{\text{cut}}^{\text{wall}} = 2^{1/6} \sigma^{\text{wall}}$$

$$R_a = 1.877 \Rightarrow r_{\text{cut}}^{\text{wall}} = 3.0 \sigma^{\text{wall}}$$

$$R_t = 0.0 \Rightarrow \sigma^{\text{wall}} = 0.5\sigma$$

$$R_t = 0.5 \Rightarrow \sigma^{\text{wall}} = \sigma$$

である.

パラメータを確認する.

- $N = 1250$
- $\rho\sigma^2 = 0.4$
- $L_x/\sigma = 39.5\dots$
- $L_y/\sigma = 79.0\dots$
- $k_B T/\varepsilon = 0.43$
- $k_B \Delta T/\varepsilon = 0.04$
- $mg\sigma/\varepsilon = 4.0 \times 10^{-4}$
- $t_f \sqrt{\varepsilon/m\sigma^2} = 2.0 \times 10^5$

この際の粒子集団の様相は以下のようになる.



(a) $R_a = 0.0, R_t = 0.0$



(b) $R_a = 0.469, R_t = 0.0$



(c) $R_a = 0.938, R_t = 0.0$



(d) $R_a = 1.408, R_t = 0.0$



(e) $R_a = 1.877, R_t = 0.0$



(f) $R_a = 0.0, R_t = 0.125$



(g) $R_a = 0.469, R_t = 0.125$



(h) $R_a = 0.938, R_t = 0.125$



(i) $R_a = 1.408, R_t = 0.125$



(j) $R_a = 1.877, R_t = 0.125$



(k) $R_a = 0.0, R_t = 0.250$



(l) $R_a = 0.469, R_t = 0.250$



(m) $R_a = 0.938, R_t = 0.250$



(n) $R_a = 1.408, R_t = 0.250$



(o) $R_a = 1.877, R_t = 0.250$



(p) $R_a = 0.0, R_t = 0.375$



(q) $R_a = 0.469, R_t = 0.375$



(r) $R_a = 0.938, R_t = 0.375$



(s) $R_a = 1.408, R_t = 0.375$



(t) $R_a = 1.877, R_t = 0.375$



(u) $R_a = 0.0, R_t = 0.500$



(v) $R_a = 0.469, R_t = 0.500$



(w) $R_a = 0.938, R_t = 0.500$



(x) $R_a = 1.408, R_t = 0.500$



(y) $R_a = 1.877, R_t = 0.500$

図 1.7: リンク先の動画は $t\sqrt{\varepsilon/m\sigma^2} = 200$ \mathcal{E} とに表示. スナップショットは $t\sqrt{\varepsilon/m\sigma^2} = t_f\sqrt{\varepsilon/m\sigma^2}$ 時.

重心位置 Y_g を系の y 幅でスケーリングして、時系列プロットすると、

$$Y_g \equiv \bar{y}_i = \frac{1}{N} \sum_i^N y_i$$

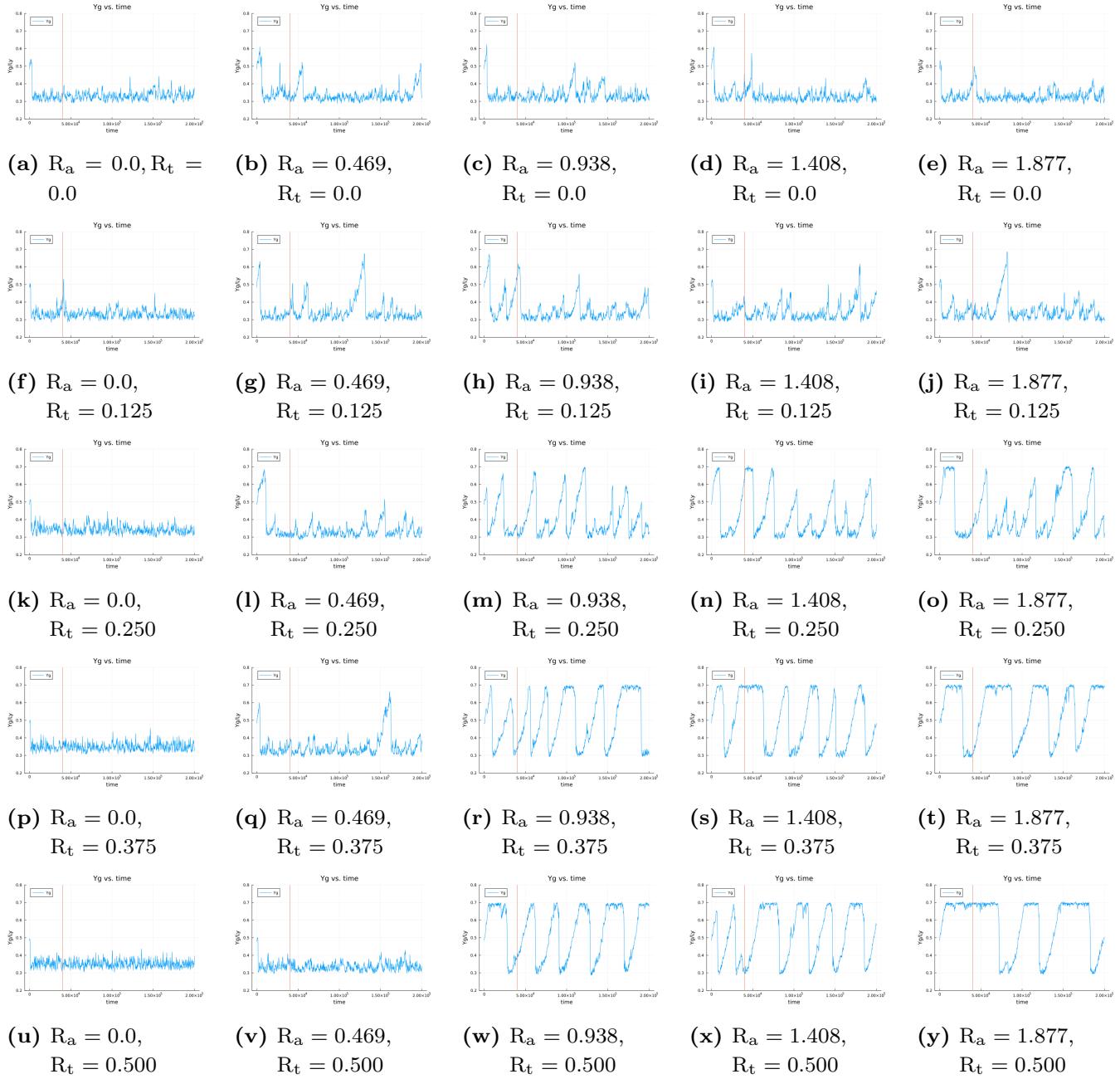


図 1.8: $t_i = 0, t_f = 2.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとにプロット.

1.2 重力を先にかけて、熱流を後からかける

以下は、追実験のときと同じように、まず重力のみをかけて、粒子集団が落ちきってから熱流をかけると同時に測定を開始するものである。

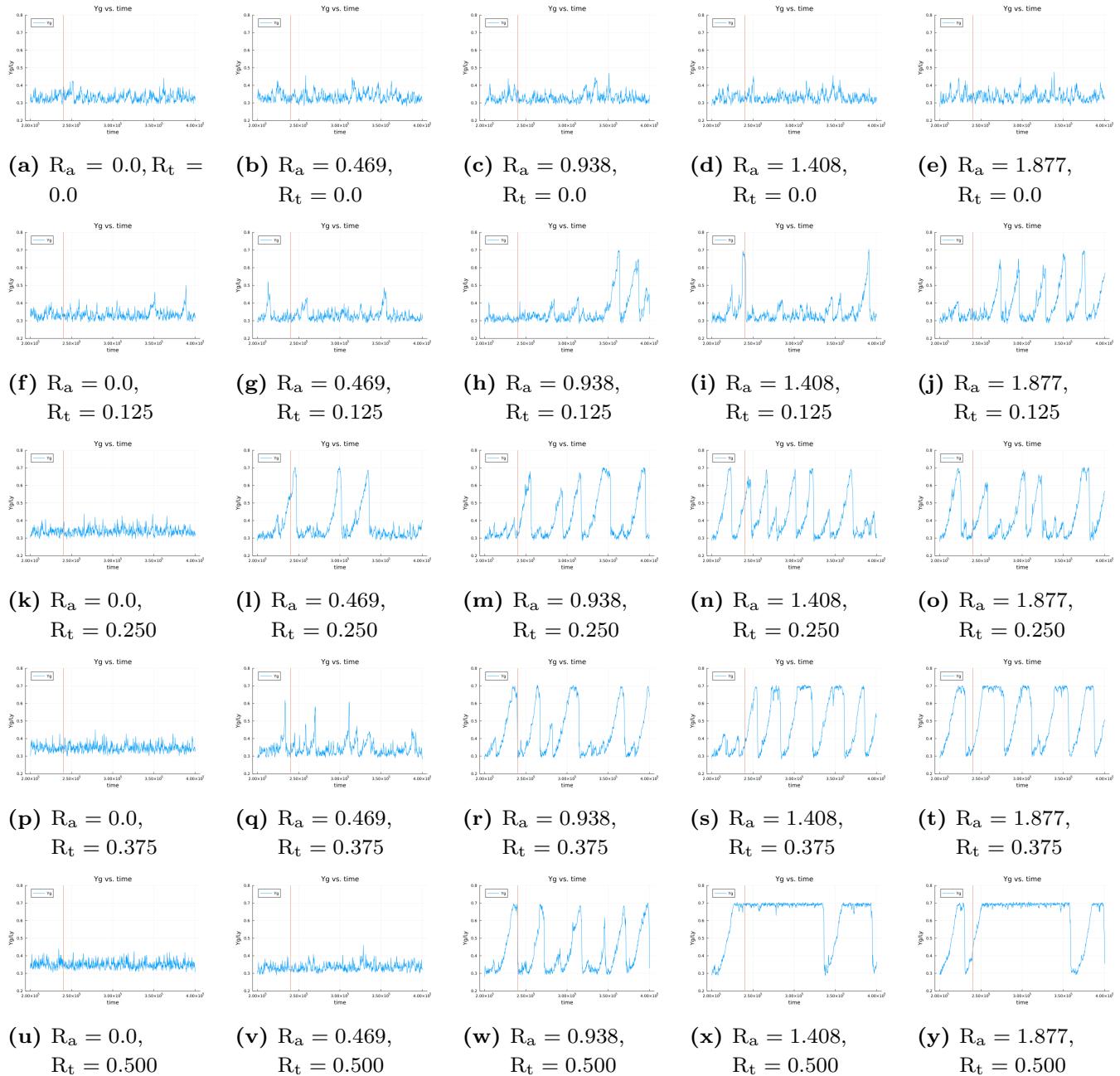


図 1.9: $t_i = 2.0 \times 10^5, t_f = 4.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとにプロット.

1.3 重力のみをかける

- $N = 1250$
- $\rho\sigma^2 = 0.4$
- $L_x/\sigma = 39.5\dots$
- $L_y/\sigma = 79.0\dots$
- $k_B T/\varepsilon = 0.43$
- $k_B \Delta T/\varepsilon = 0.0$
- $mg\sigma/\varepsilon = 4.0 \times 10^{-4}$
- $t_f \sqrt{\varepsilon/m\sigma^2} = 2.0 \times 10^5$

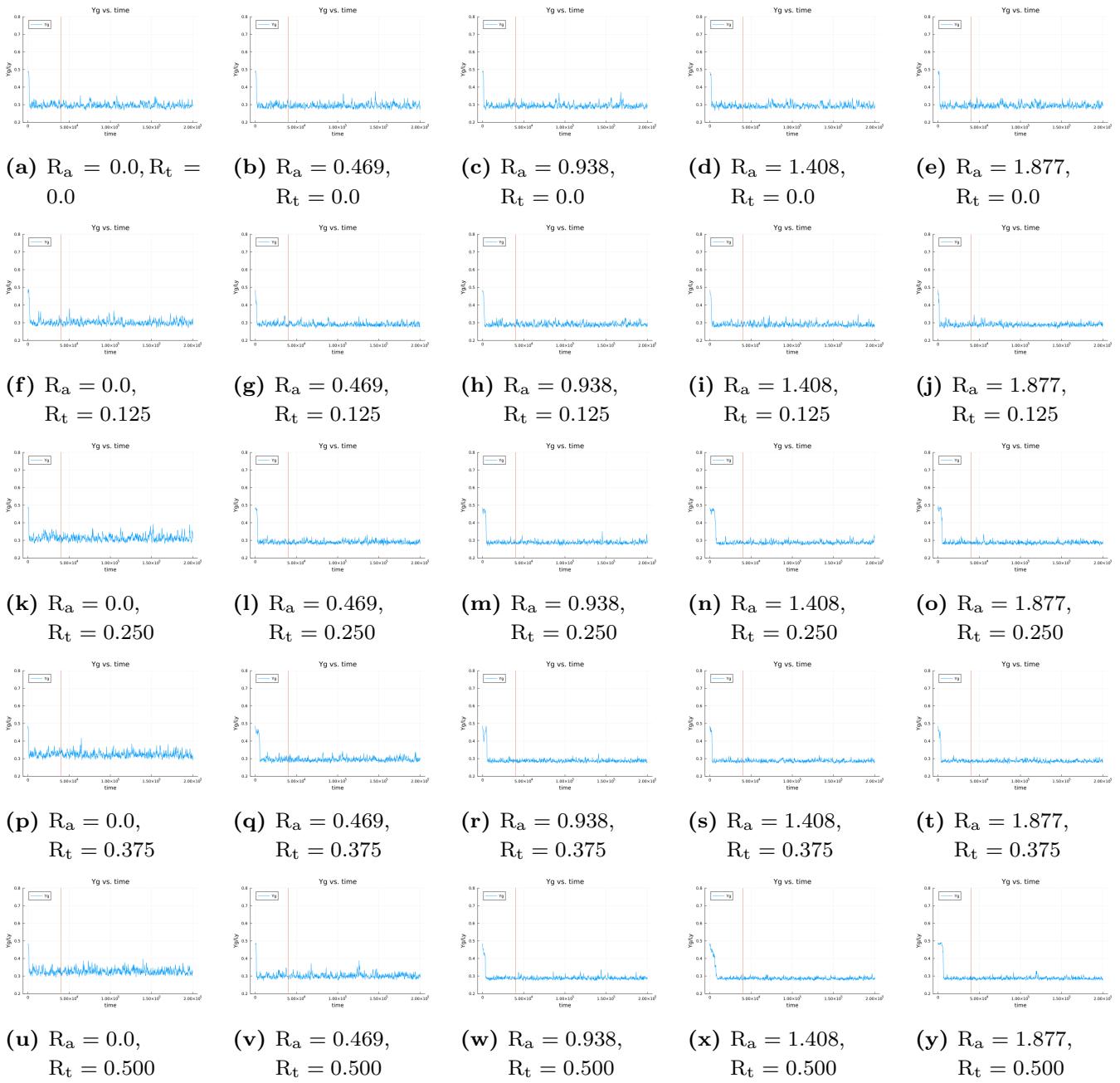


図 1.10: $t_i = 0, t_f = 2.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとにプロット.

1.4 熱流のみをかける

- $N = 1250$
- $\rho\sigma^2 = 0.4$

- $L_x/\sigma = 39.5\dots$
- $L_y/\sigma = 79.0\dots$
- $k_B T/\varepsilon = 0.43$
- $k_B \Delta T/\varepsilon = 0.04$
- $mg\sigma/\varepsilon = 0.0$
- $t_f \sqrt{\varepsilon/m\sigma^2} = 2.0 \times 10^5$

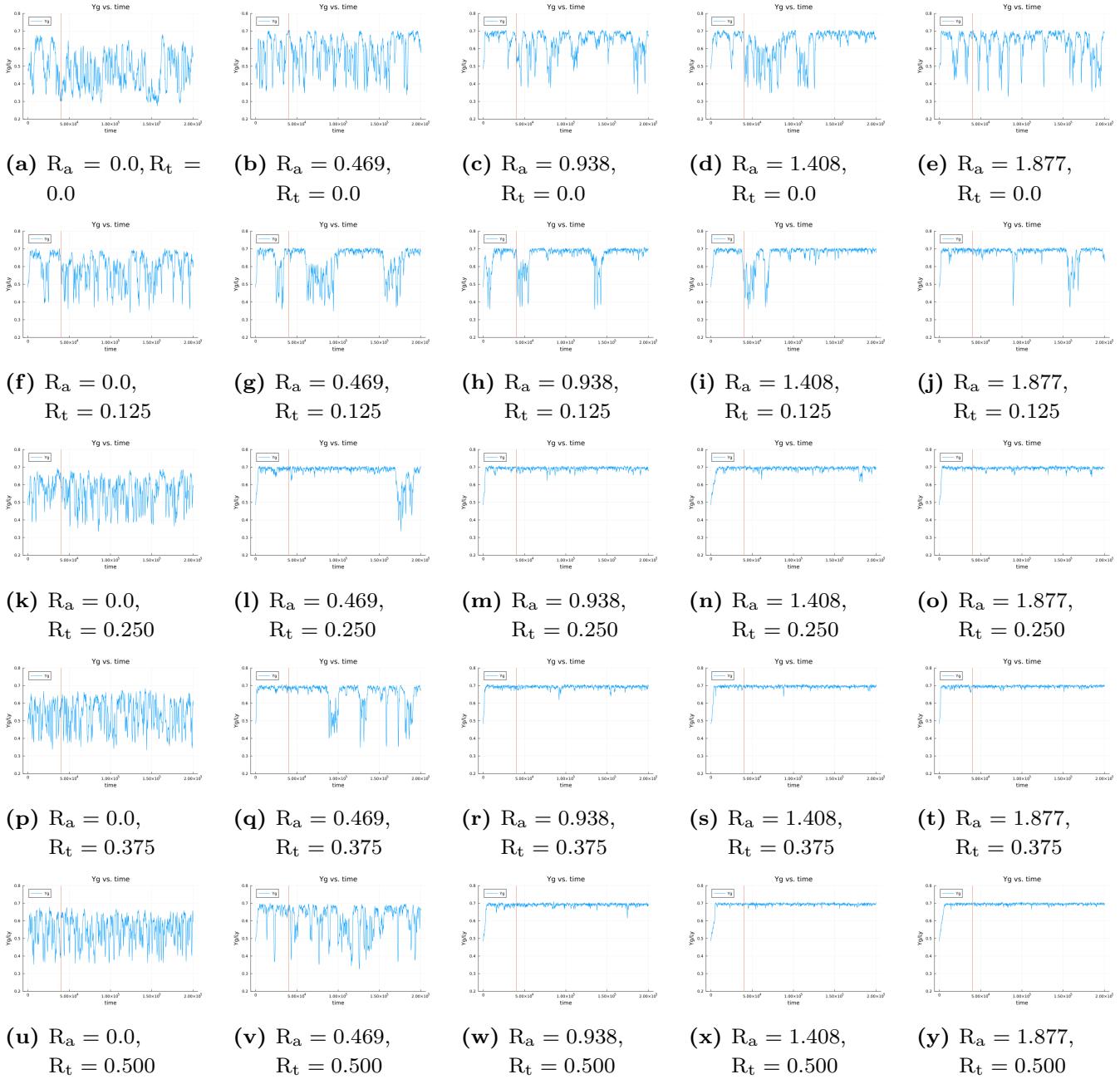


図 1.11: $t_i = 0, t_f = 2.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとにプロット.

1.5 重力と熱流を同時にかける (時間 10 倍)

- $N = 1250$
- $\rho\sigma^2 = 0.4$

- $L_x/\sigma = 39.5\dots$
- $L_y/\sigma = 79.0\dots$
- $k_B T/\varepsilon = 0.43$
- $k_B \Delta T/\varepsilon = 0.04$
- $mg\sigma/\varepsilon = 4.0 \times 10^{-4}$
- $t_f \sqrt{\varepsilon/m\sigma^2} = 2.0 \times 10^6$

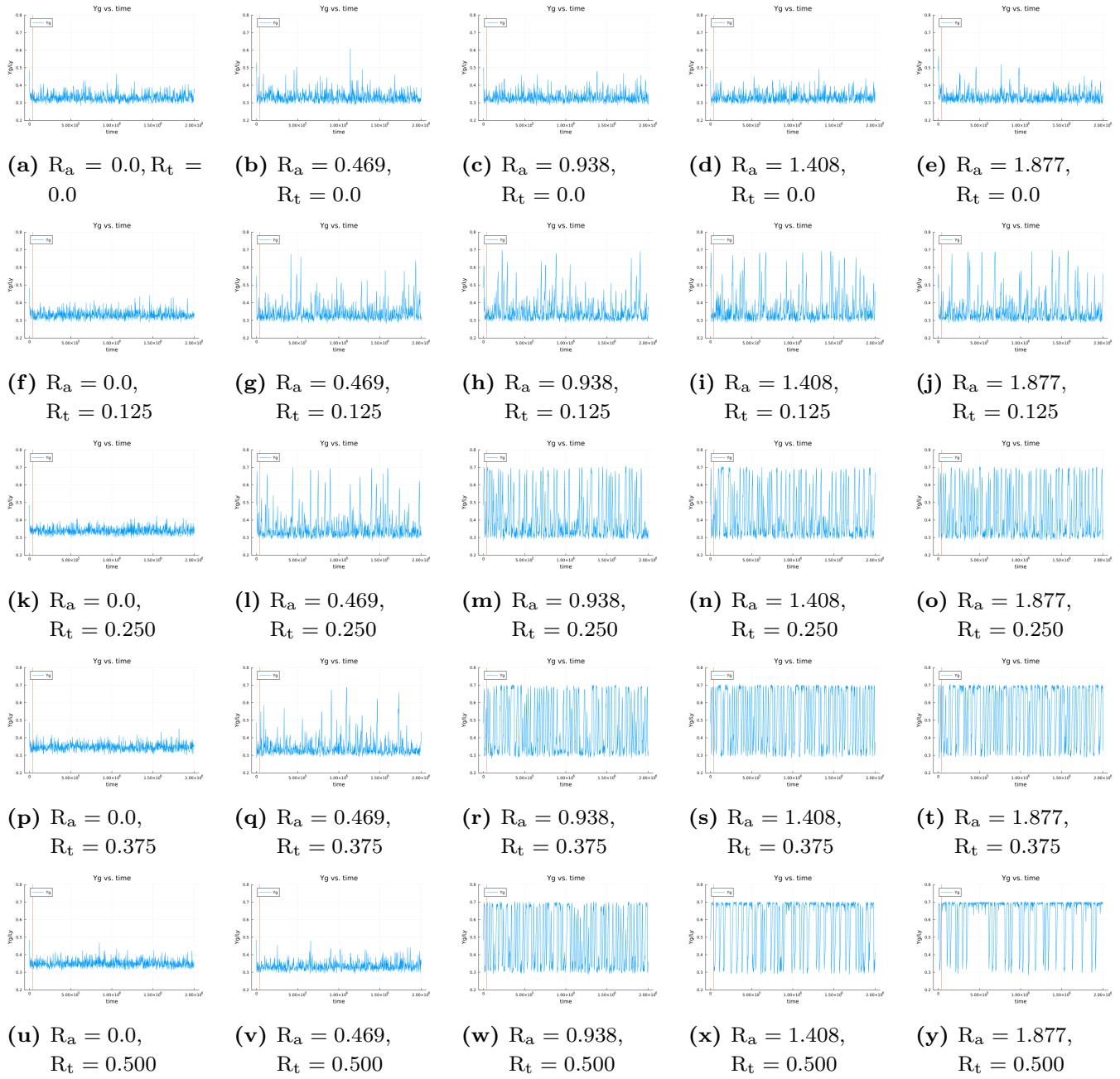


図 1.12: $t_i = 0, t_f = 2.0 \times 10^6, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 2000$ ごとにプロット.

1.6 重力を先にかけて、熱流を後からかける（時間 10 倍）

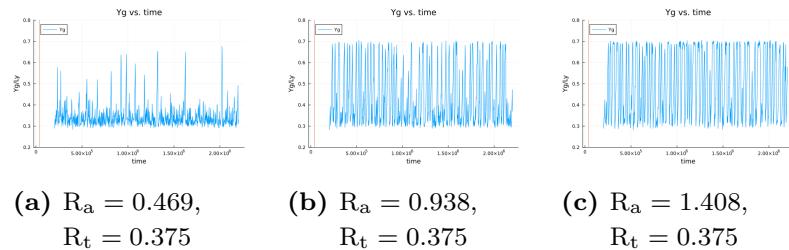


図 1.13: $t_i = 2.0 \times 10^5, t_f = 2.2 \times 10^6, t\sqrt{\epsilon/m\sigma^2} = 2000$ ごとにプロット.

第 2 章

分析と考察

この章では、第 1 章において行ったシミュレーションを用いて、それぞれ分析と考察をしていく。適宜文脈に則した分析画像を提示するが、入りきらない部分は付録??に収録する。

以降の解析は定常状態であるとみなせる、 $t_i \sqrt{\varepsilon/m\sigma^2} = 4.0 \times 10^4$ からのデータを用いてプロットすることにしている。定常状態であるとみなせる時間については議論する必要がある。

2.1 重心位置

重心位置の標準偏差は以下のように書くことができる。

$$\sigma(Y_g) = \sqrt{\frac{1}{N_D} \sum_{t=t_i}^{t=t_f} (Y_g(t) - \bar{Y}_g)^2}$$

これは、時間発展する重心位置のばらつき具合を意味する。後述する空間的なばらつきとは定義から見ても分かるとおり異なる量である。

重心位置の標準偏差について同時プロットで表してみる。

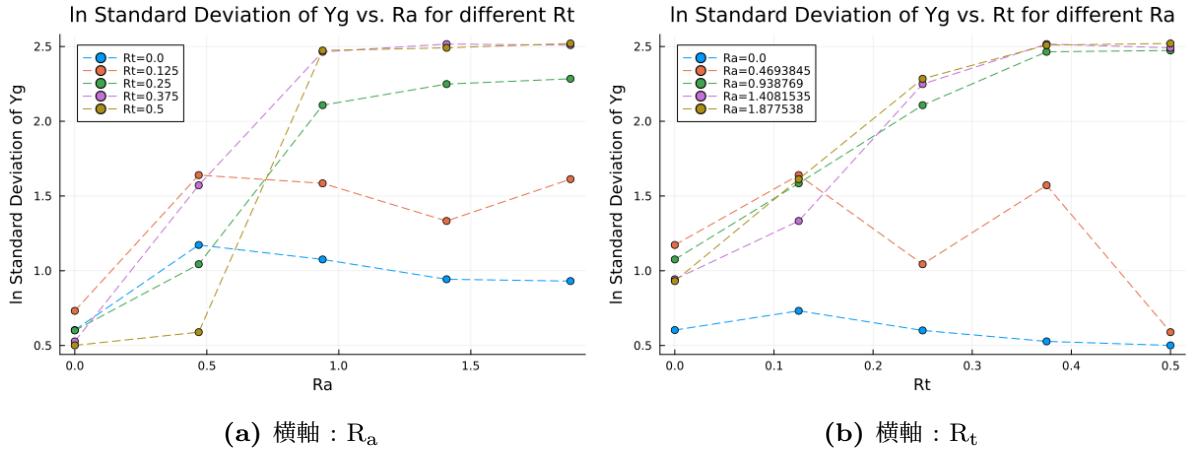


図 2.1: 縦軸 : 重心位置の標準偏差の対数プロット

図 1.8v $R_a = 0.469$, $R_t = 0.5$ と図 1.8w $R_a = 0.938$, $R_t = 0.5$ の間をもっと詳しく見る.

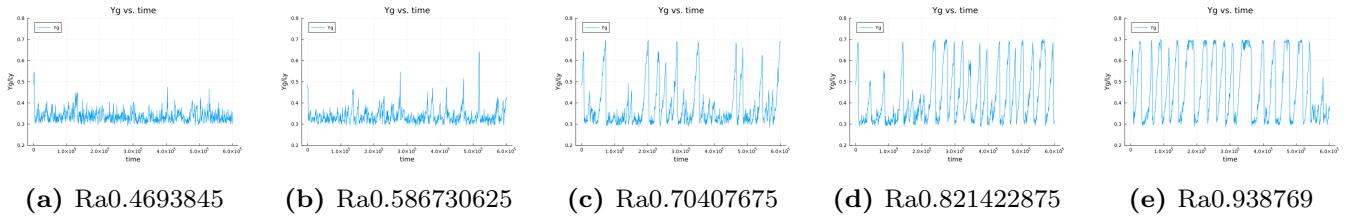


図 2.2: $t_i = 0$, $t_f = 6.0 \times 10^5$, $t\sqrt{\epsilon/m\sigma^2} = 600$ ごとにプロット.

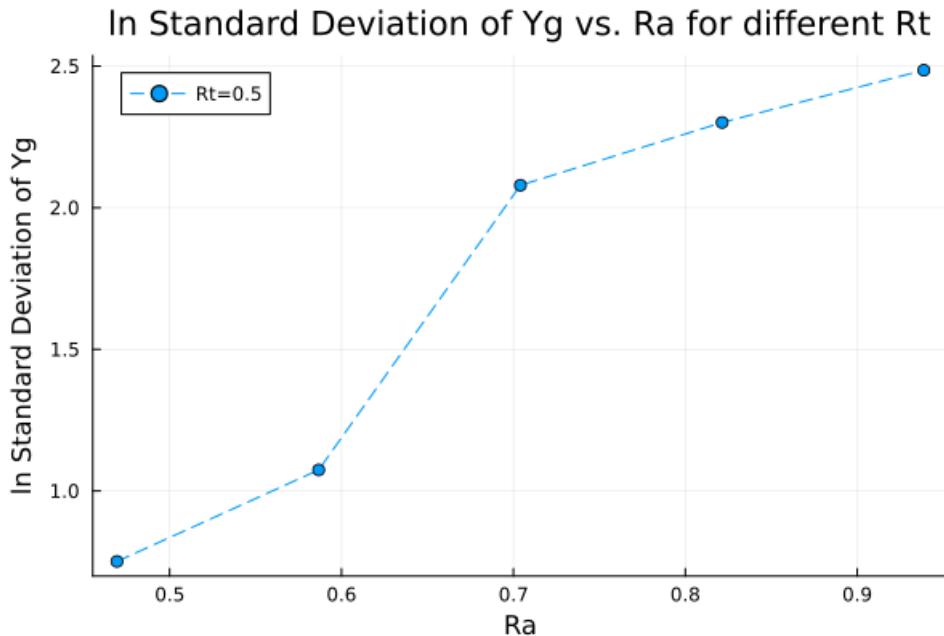


図 2.3

図 2.2c $R_a = 0.469, R_t = 0.5$ と 図 2.2e $R_a = 0.938, R_t = 0.5$ の間を詳しく見る.

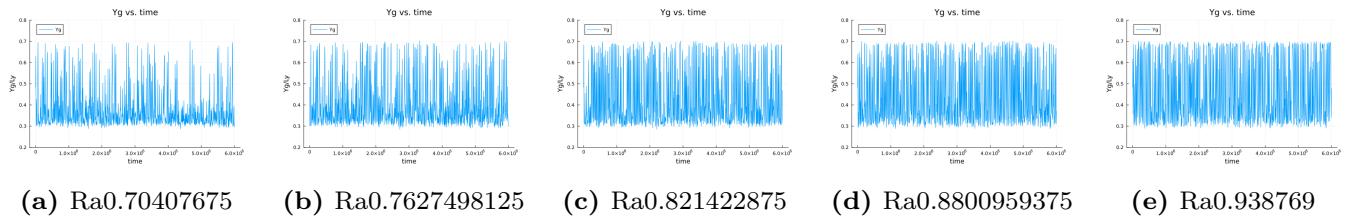


図 2.4: $t_i = 0, t_f = 6.0 \times 10^6, t\sqrt{\epsilon/m\sigma^2} = 6000$ ごとにプロット.

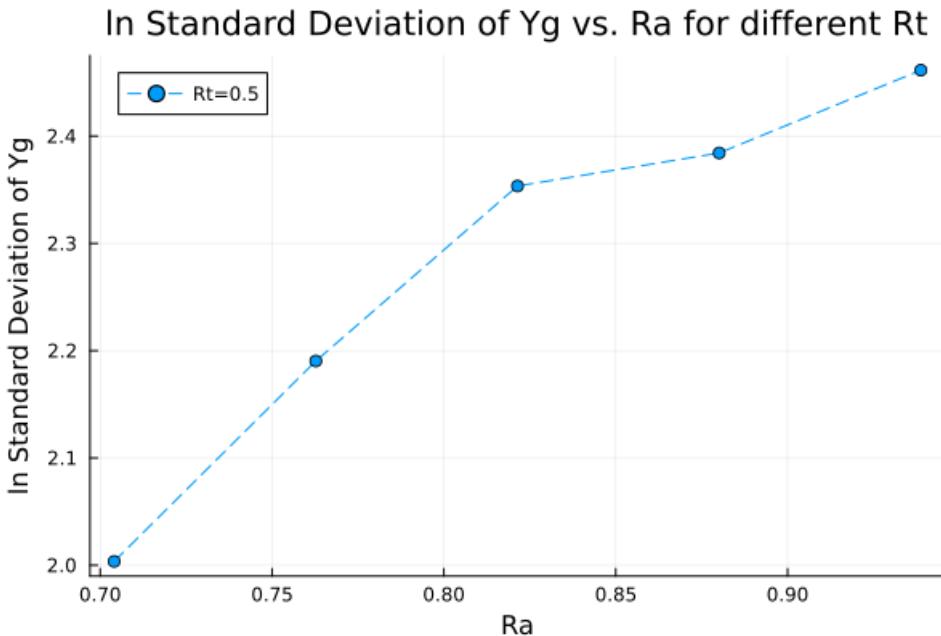


図 2.5

先の分析では、時間プロットの幅が大きい可能性があるため、図 1.8v $R_a = 0.469, R_t = 0.5$ と図 1.8w $R_a = 0.938, R_a = 0.5$ の間のプロット幅を小さくしたデータを使う。その際、簡単のため、 $R_a = 0.5 \sim 1.0$ の間を見るることにする。

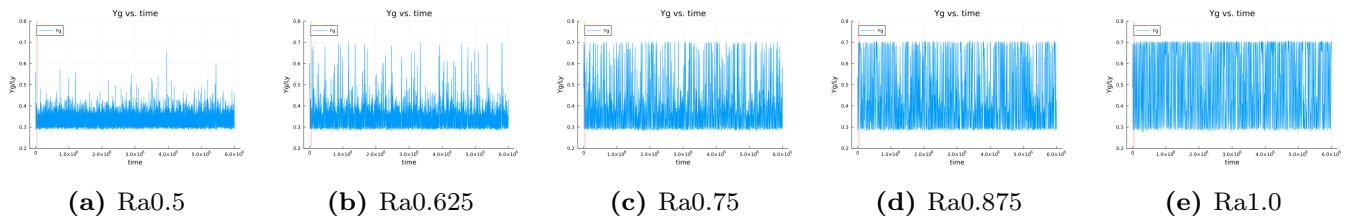


図 2.6: $t_i = 0, t_f = 6.0 \times 10^6, t\sqrt{\epsilon/m\sigma^2} = 50$ ごとにプロット。

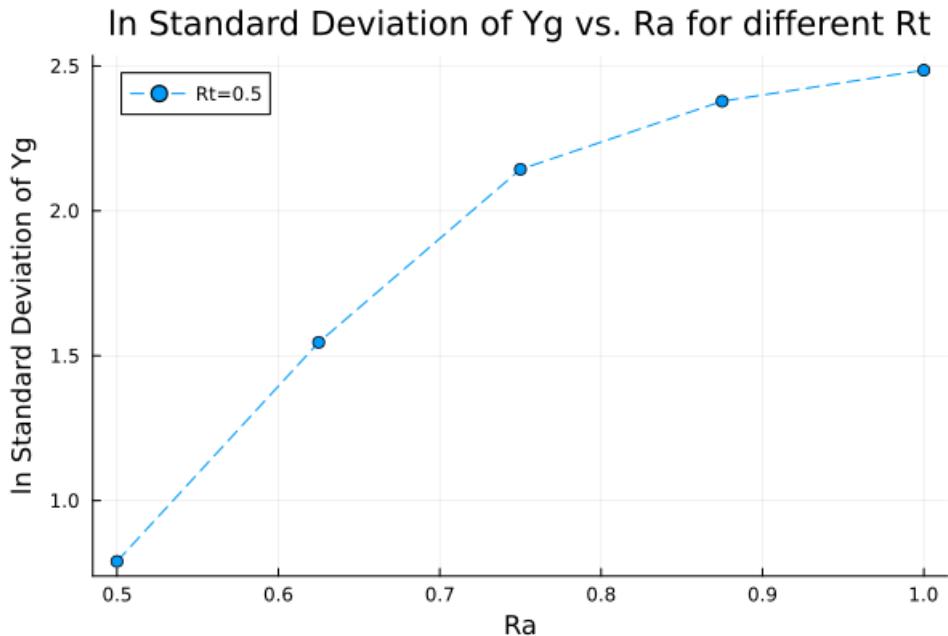


図 2.7

図 2.7 からも見て分かるように, $R_t = 0.5$ において, R_a の値を大きくするほど, 各系の重心位置の時間発展はばらつきが小さくなると言うことがわかる.

2.2 空間的なばらつき

流体系の空間的なばらつき $\sigma_y(t)$ の時間発展をプロットする.

$$\begin{aligned}
 \sigma_y(t) &= \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i(t) - \bar{y}_i(t))^2} \\
 &= \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i(t) - Y_g(t))^2} \\
 &= \sqrt{\frac{1}{N} \sum_{i=1}^N y_i(t)^2 - Y_g(t)^2}
 \end{aligned}$$

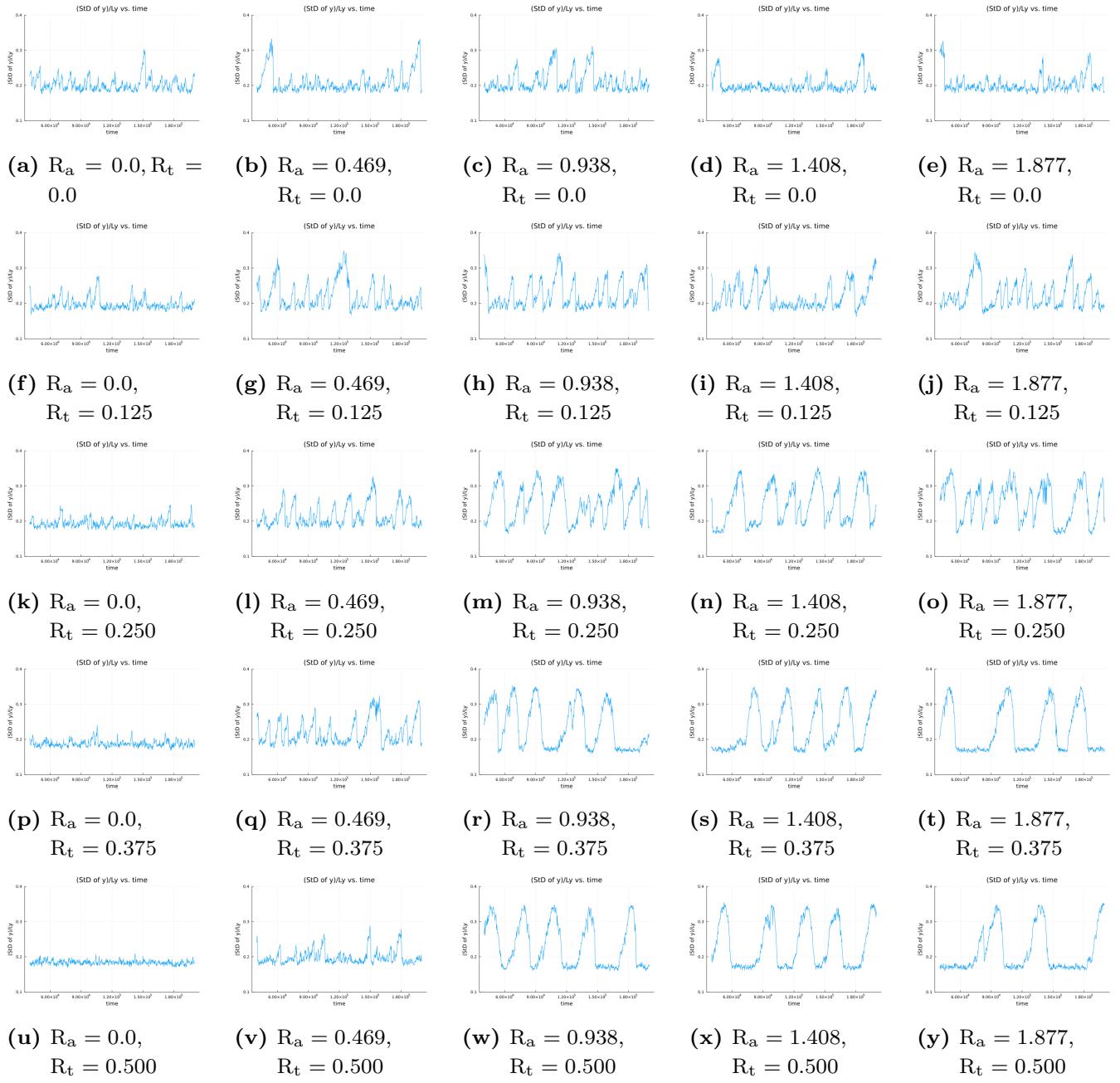


図 2.8: $t_i = 4.0 \times 10^4, t_f = 2.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとに重心の空間的な揺らぎを時系列プロット.

2.3 リミットサイクル

系の周期的なダイナミクスについて考える。非線形振動にはリミットサイクル振動 [1] と呼ばれる振動の形態があり、ここでは流体系の重心位置と空間的なばらつきの相空間での軌道をみることにする。

サイクルが閉じているとき、非定常で周期的なダイナミクスが現れているということが言え、周期的なダイナミクスが見れる系においては、流体系が液滴を形成しつつ、上壁に吸着しきるまでのスピードが、液体の落下のスピードよりも遅いことが分かる。

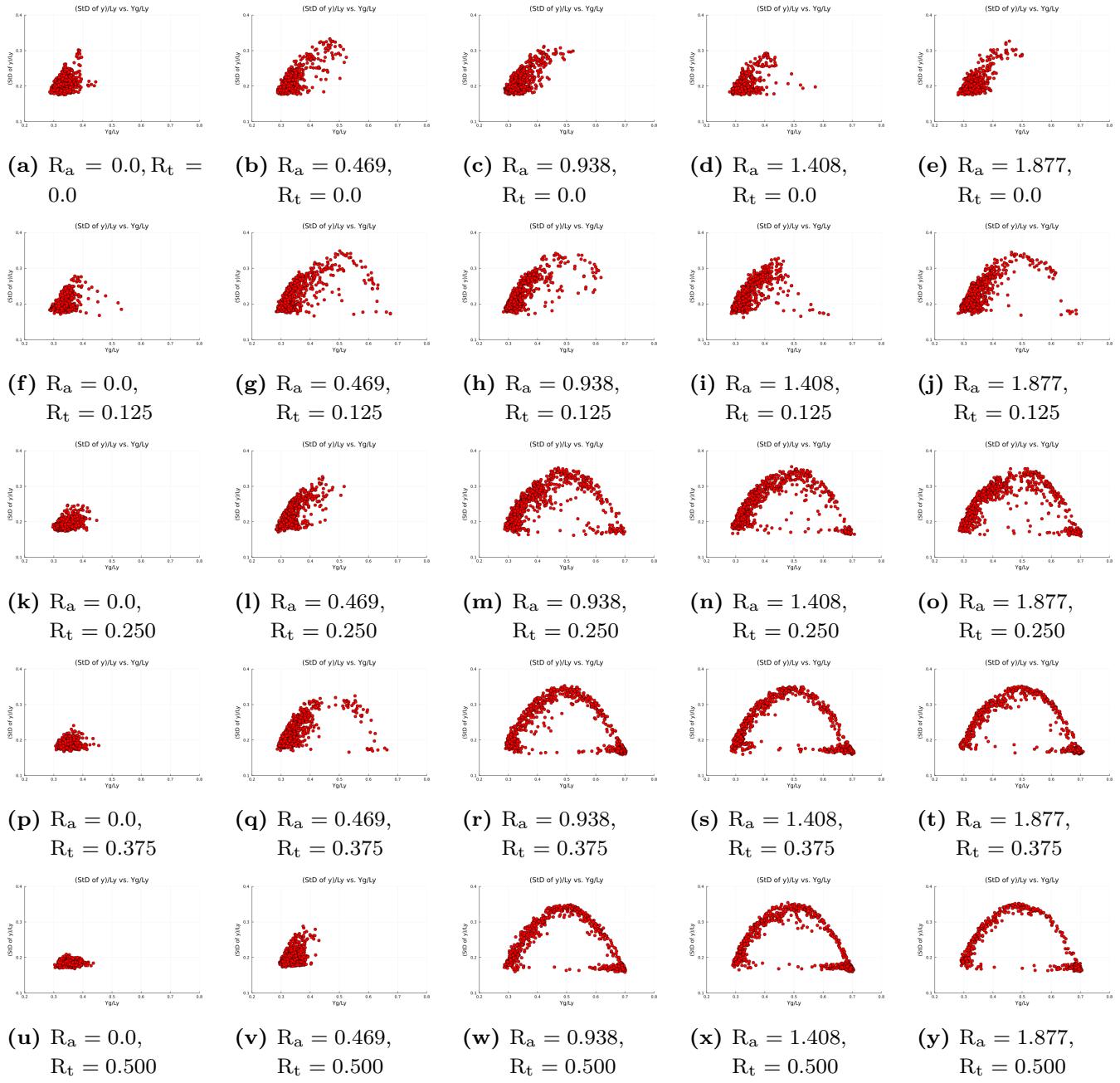


図 2.9: $t_i = 4.0 \times 10^4, t_f = 2.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとにプロット.

2.4 リミットサイクル 3D

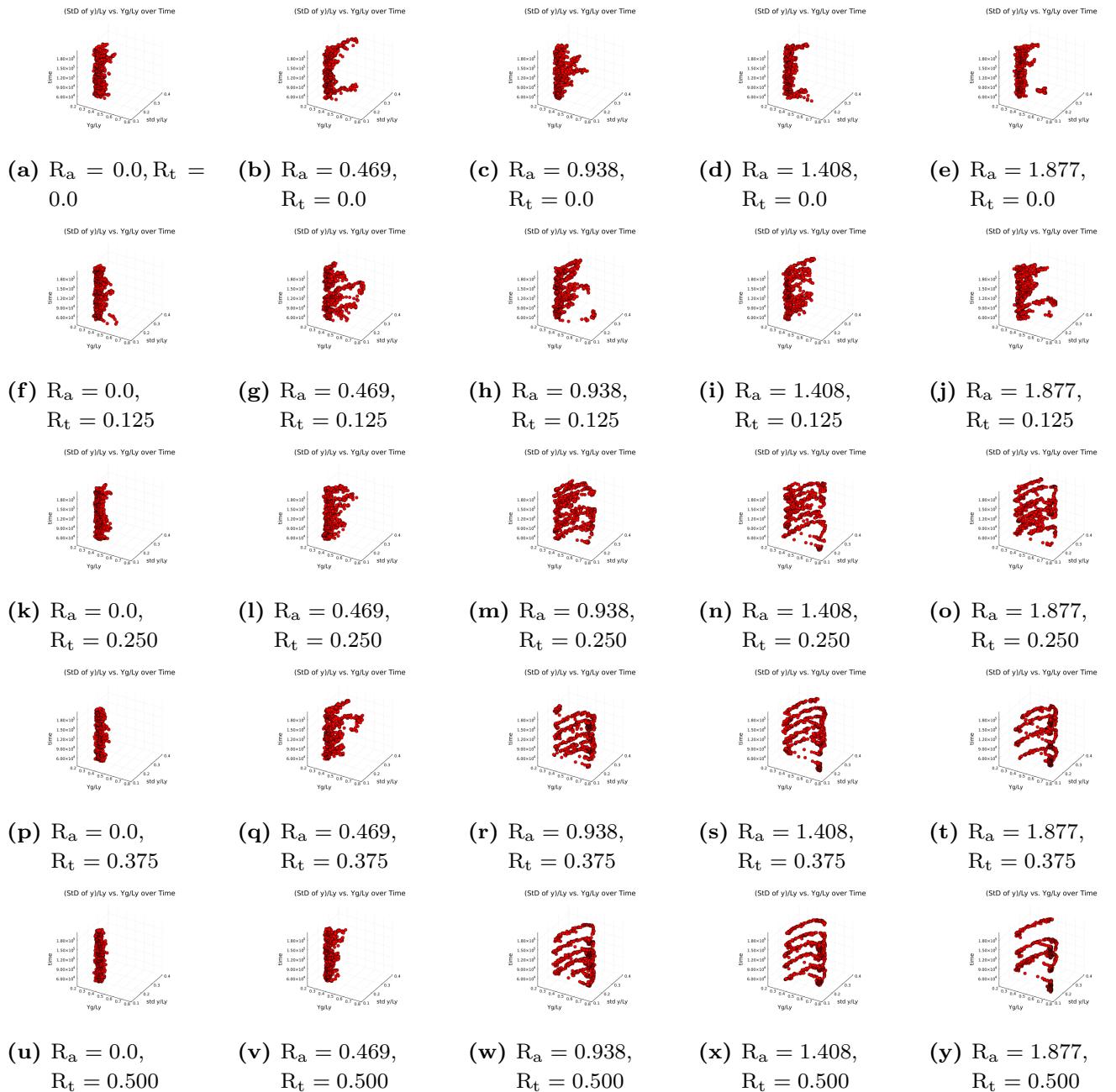


図 2.10: $t_i = 4.0 \times 10^4, t_f = 2.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとにプロット.

2.5 ヒートマップ

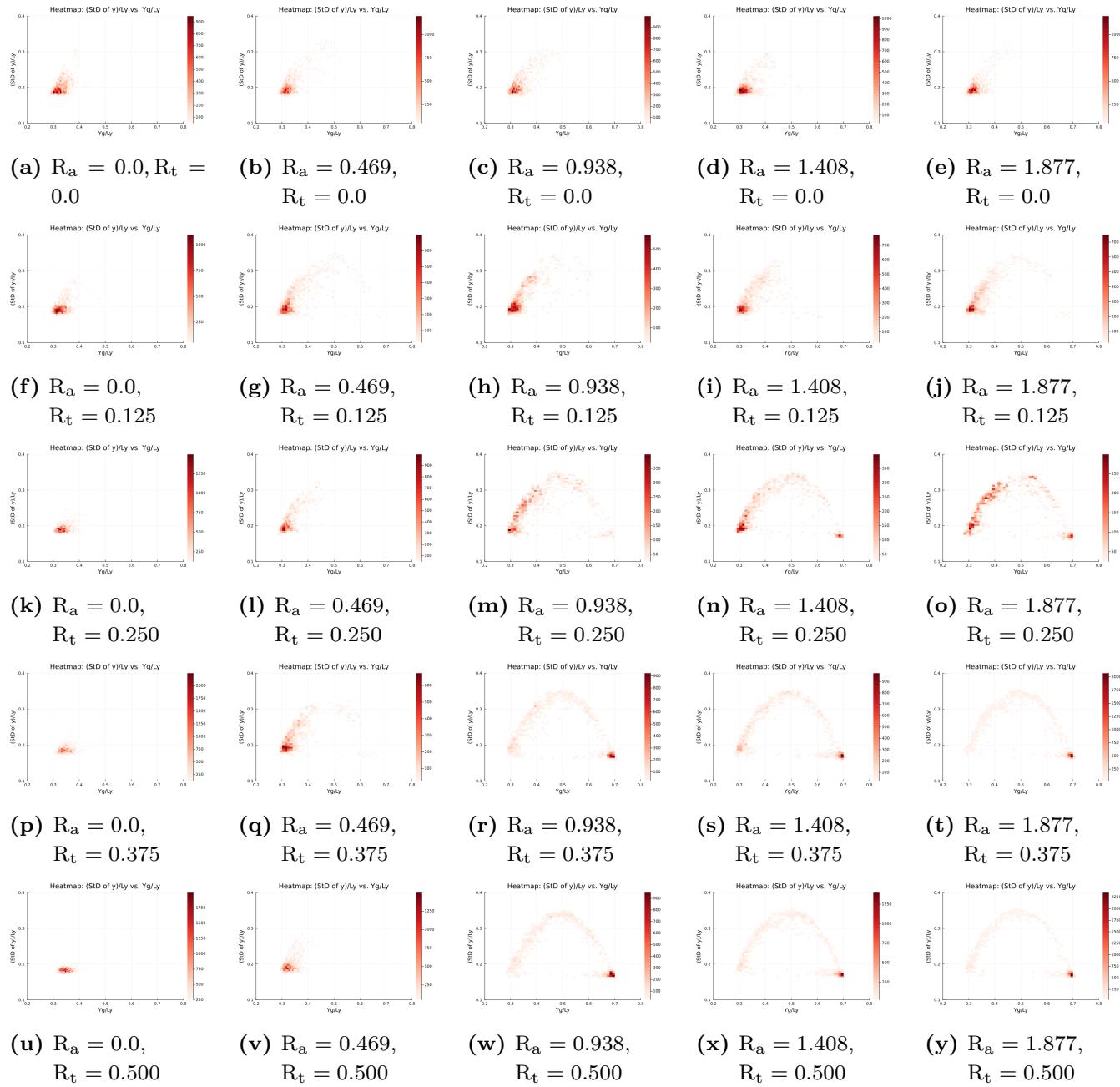


図 2.11: $t_i = 4.0 \times 10^4, t_f = 2.0 \times 10^5, dt\sqrt{\varepsilon/m\sigma^2} = 0.005, t\sqrt{\varepsilon/m\sigma^2} = 200$ ごとにプロット。

付録 A

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。

A.1 LAMMPS ファイル

ソースコード A.1: in.2dLJ

```
1 # 2d Lennard-Jones
2
3
4 # 出力関係のパラメータ
5 variable run equal 40000000
6 variable thermo equal ${run}/1000 # 分母の数が
   log で生成される行数になる。
7 variable dump equal ${run}/1000 # 分母の数が
   lammpstrj で生成される行数になる。
8 variable image_step equal ${run}/4 # 分母の数 +1枚の画像を作成。
9
10 # 重要なパラメータ
11 variable SEED equal 202035
12 variable Ay equal 50 # 粒子生成に用いるy 方向でのセル数。
13 variable Ax equal ${Ay}/2 # 粒子生成に用いるx 方向でのセル数。
14 variable rho equal PLACEHOLDER_rho # 密度。密度と粒子数から体積が決
   まる。
15 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тен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тен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-${Rd} # LJ ポтенシャルの epsilon;
   ポテンシャルの深さ.
32 variable sigma_wall equal 0.5+${Rt} # LJ ポтенシャルの sigma; 衝突
   直径.
33 variable rc_wall equal 1.122462+${Ra} #
   WCA ポтен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 -${gap} # 热浴で温度の低い方の小さい方の直線.
42 variable hothi equal -${gap}+${trange} # 热浴で温度の低い方の大きい
   方の直線.
43 variable coldlo equal ${Ay}-${gap}-${trange} # 热浴で温度の高い方
   の小さい方の直線.
44 variable coldhi equal ${Ay}-${gap} # 热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=1,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 # 热浴
   C の領域.
63 region hot block INF INF ${hotlo} ${hothi} -0.1 0.1 # 热浴H の領域
   .
64
65 # 各热浴領域の温度を計算
66 compute Tcold all temp/region cold #
   c_Tcold で cold 热浴領域の温度を取得.
67 compute Thot all temp/region hot #
   c_Tcold で cold 热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} # 热力学量の出力.
136 thermo_modify norm no # 示量的な热力学量に调整.
137
138 run ${run} # 実行.

```

ソースコード A.2: in.2dLJ.mod

```

1 # 2d Lennard-Jones
2
3
4 # 出力関係のパラメータ
5 variable run equal PLACEHOLDER_run
6 variable thermo equal ${run}/1000 # 分母の数だけ出力.(log, yaml)
7 variable dump equal ${run}/1000 # 分母の数だけ出力.(lammpstrj)
8 variable image_step equal ${run}/1 # 分母の数 +1枚の画像を作成.
9
10 # 重要なパラメータ
11 variable SEED equal 202035
12 variable Ay equal PLACEHOLDER_Ay # 粒子生成に用いるy 方向でのセル数.
13 variable Ax equal ${Ay}/2 # 粒子生成に用いるx 方向でのセル数.
14 variable rho equal PLACEHOLDER_rho # 密度. 密度と粒子数から体積が決
    まる.
15 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 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.
18 variable dT equal PLACEHOLDER_dT
19 variable thot equal ${T}+(${dT}/2) # 座標の小さい方の熱浴の目標温度.
20 variable tcold equal ${T}-(${dT}/2) # 座標の大きい方の熱浴の目標温
    度.
21 variable g equal PLACEHOLDER_g # 重力加速度.
22 # 粒子-粒子間のLJ ポテンシャル
23 variable epsilon_pair equal 1.0 # LJ ポтенシャルの epsilon; ポテン
    シャルの深さ.
24 variable sigma_pair equal 1.0 # LJ ポтенシャルの sigma; 衝突直径.
25 variable rc_pair equal 3.0 # 典型的なカットオフ長.
26 # 壁-粒子間のLJ ポтенシャル
27 variable Rd equal PLACEHOLDER_Rd # 乾き具合.
28 variable Rt equal PLACEHOLDER_Rt # 壁の厚み.
29 variable Ra equal PLACEHOLDER_Ra # 濡れ具合.
30 variable epsilon_wall equal 1.0-${Rd} # LJ ポтенシャルの epsilon;
    ポтенシャルの深さ.
31 variable sigma_wall equal 0.5+${Rt} # LJ ポтен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 -${gap} # 热浴で温度の低い方の小さい方の直線.
41 variable hothi equal -${gap}+${trange} # 热浴で温度の低い方の大きい
    方の直線.
42 variable coldlo equal ${Ay}-${gap}-${trange} # 热浴で温度の高い方
    の小さい方の直線.
43 variable coldhi equal ${Ay}-${gap} # 热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=1,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 # 热浴
    C の領域.
62 region hot block INF INF ${hotlo} ${hothi} -0.1 0.1 # 热浴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 # 热浴Hについての設定の解除.
97 unfix cold # 热浴Cについての設定の解除.
98
99 fix hot all langevin ${thot} ${thot} 1.0 ${SEED} tally no # 热浴
   が温度tlo になるようにする.
100 fix cold all langevin ${tcold} ${tcold} 1.0 ${SEED} tally no # 
   热浴が温度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}

```

```

126    etotal:=$(etotal)
127    Yg:=$(c_CoM[2])"" file PLACEHOLDER_outputtitle.yaml screen no
128
129 # # 一次元プロファイル(今は温度と密度だけ計算と出力)
130 # compute chunk all chunk/atom bin/1d y lower 3.0 units box
131 # fix tempp all ave/chunk 100000 1 100000 chunk temp file
#      temp_profile.profile
132 # fix rhop all ave/chunk 100000 1 100000 chunk density/number
#      file rho_profile.profile
133
134 thermo ${thermo} # 热力学量の出力.
135 thermo_modify norm no # 示量的な热力学量に調整.
136
137 run ${run} # 実行.

```

A.2 実行ファイル

ソースコード A.3: lammps_modexe.jl

```

1 ====
2 # LAMMPS シミュレーション実行と出力ファイル保管
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.
5
6 ## 機能
7
8 - '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
9 - パラメータを配列で定義
10 - 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riptを生成
11 - 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
12 - 使用済みの仮LAMMPS ファイルを一括削除
13
14 ## 手順
15

```

```

16 1. LAMMPS ファイルの特定とパラメータ設定
17 2. パラメータの範囲を定義
18 3. パラメータの組み合わせごとにLAMMPS を実行
19 4. 出力ファイルを指定ディレクトリに保存
20 5. 使用済みの仮LAMMPS ファイルを削除
21
22 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.
23 ===#
24
25 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
26 using Dates # 日付と時刻の取得に使用するパッケージ
27
28 # 実行するLAMMPS ファイルを特定
29 lammpsfile = glob("in.*")[1]
30
31 # パラメータの設定
32 chi = 1.265
33 remark_text = "test"
34 file_extensions = ["log", "yaml", "lammpstrj"] # 出力ファイルの拡
   張子
35
36 # パラメータの範囲を設定
37 Ay_range = range(50, length=1) # Ay の範囲
38 rho_range = range(0.4, length=1) # 密度の範囲
39 T_range = range(0.43, length=1) # 初期温度の範囲
40 dT_range = range(0.04, length=1) # 热浴の温度差の範囲
41 Rd_range = range(0.0, length=1) # 乾燥度の範囲
42 Rt_range = range(0.0, 0.5, length=5) # 壁の厚みの範囲
43 Ra_range = range(0.0, 1.877538, length=5) # 引力幅の範囲
44 run_range = range(4e7, length=1) # 実行ステップ数の範囲
45
46 # パラメータごとに実験を実行
47 for Ay_value in Ay_range,
48     rho_value in rho_range,
49     T_value in T_range,
50     dT_value in dT_range,
51     Rd_value in Rd_range,

```

```

52     Rt_value in Rt_range,
53     Ra_value in Ra_range,
54     run_value in run_range
55
56     # パラメータに基づいて重力を計算
57     g_value = dT_value / ((Ay_value / sqrt(rho_value)) * chi)
58
59     # 実験日時を記録
60     n = string(now())
61
62     # パラメータに基づいた出力ファイル名を生成
63     parameter = "chi$(chi)_Ay$(Ay_value)_rho$(rho_value)_T$(
64         T_value)_dT$(dT_value)_Rd$(Rd_value)_Rt$(Rt_value)_Ra$(
65         Ra_value)_g$(g_value)_run$(run_value)"
66     outputtitle = "$(n)_${remark_text}_${parameter}"
67
68     # LAMMPS ファイルの内容を読み込み, パラメータを置換
69     template_script = read(lammpsfile, String)
70     mod_script = replace(template_script,
71         "PLACEHOLDER_Ay" => string(Ay_value),
72         "PLACEHOLDER_rho" => string(rho_value),
73         "PLACEHOLDER_T" => string(T_value),
74         "PLACEHOLDER_dT" => string(dT_value),
75         "PLACEHOLDER_g" => string(g_value),
76         "PLACEHOLDER_Rd" => string(Rd_value),
77         "PLACEHOLDER_Rt" => string(Rt_value),
78         "PLACEHOLDER_Ra" => string(Ra_value),
79         "PLACEHOLDER_run" => string(run_value),
80         "PLACEHOLDER_outputtitle" => string(outputtitle)
81     )
82
83     # 一意のファイル名を生成して仮ファイルを作成し, パラメータを書き込む
84     tempfile = "in.temp_script_$(n)"
85     fp = open(tempfile, "w")
86     write(fp, mod_script)
87     close(fp)
88
89     # LAMMPS を実行

```

```

88     run(`mpirun -n 4 lmp_mpi -log ${outputtitle}.log -in $(
89         tempfile)`)
90
90 # 出力ファイルを保存
91 for file_ext in file_extensions
92     files = glob("*.$(file_ext)")
93     for file in files
94         outputpath = "../outputdir/$(file_ext)dir"
95         mkpath(outputpath)
96         script = read(file, String)
97         fp = open(joinpath(outputpath, "$(file)"), "w")
98
99         if file_ext == "log"
100             println(fp, "$(file)")
101         end
102
103         write(fp, script)
104         close(fp)
105         rm(file)
106     end
107 end
108 end
109
110 # 使用済みの仮LAMMPS ファイルを一括削除
111 files = glob("in.temp_*")
112 for file in files
113     rm(file)
114 end

```

ソースコード A.4: lammps_qsub_job.jl

```

1 ====
2 # LAMMPS シミュレーション実行
3
4 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。
5
6 ## 機能
7

```

```

8 - '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.
9 - パラメータを配列で定義.
10 - 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.
11 - パラメータごとにLAMMPS を実行.

12
13 ## 手順
14
15 1. LAMMPS ファイルの特定とパラメータ設定.
16 2. パラメータの範囲を定義.
17 3. パラメータの組み合わせごとにLAMMPS を実行.

18
19 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.
20 ===#
21
22
23 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.
24 using Dates # 日時を取得するパッケージ.

25
26 # 実行するLAMMPS ファイルを特定
27 lammpsfile = glob("in.*")[1]

28
29 # パラメータの設定
30 chi = 1.265
31 remark_text = "test"

32
33 # パラメータの範囲を設定
34 Ay_range = range(100, length=1) # Ay の範囲
35 rho_range = range(0.4, length=1) # 密度の範囲
36 T_range = range(0.43, length=1) # 初期温度の範囲
37 dT_range = range(0.0, length=1) # 热浴の温度差の範囲
38 Rd_range = range(0.0, length=1) # 乾燥度の範囲
39 Rt_range = range(0.5, length=1) # 壁の厚みの範囲
40 Ra_range = range(0.0, 1.877538, length=5) # 濡れ具合の範囲
41 run_range = range(4e7, length=1) # 実行ステップ数の範囲

42
43 # 多重ループを用いてパラメータごとに実験を実行.

```

```

44 for Ay_value in Ay_range,
45     rho_value in rho_range,
46     T_value in T_range,
47     dT_value in dT_range,
48     Rd_value in Rd_range,
49     Rt_value in Rt_range,
50     Ra_value in Ra_range,
51     run_value in run_range
52
53 # パラメータに基づいて重力を計算
54 g_value = dT_value / ((Ay_value / sqrt(rho_value)) * chi)
55
56 # 実験日時を記録
57 n = string(now())
58
59 # パラメータに基づいた出力ファイル名を生成
60 parameter = "chi$(chi)_Ay$(Ay_value)_rho$(rho_value)_T$(
61     T_value)_dT$(dT_value)_Rd$(Rd_value)_Rt$(Rt_value)_Ra$(
62     Ra_value)_g$(g_value)_run$(run_value)"
63 outputtitle = "$(n)_${remark_text}_${parameter}"
64
65 # LAMMPS ファイルの内容を読み込み、パラメータを置換
66 template_script = read(lammpsfile, String)
67 mod_script = replace(template_script,
68     "PLACEHOLDER_Ay" => string(Ay_value),
69     "PLACEHOLDER_rho" => string(rho_value),
70     "PLACEHOLDER_T" => string(T_value),
71     "PLACEHOLDER_dT" => string(dT_value),
72     "PLACEHOLDER_g" => string(g_value),
73     "PLACEHOLDER_Rd" => string(Rd_value),
74     "PLACEHOLDER_Rt" => string(Rt_value),
75     "PLACEHOLDER_Ra" => string(Ra_value),
76     "PLACEHOLDER_run" => string(run_value),
77     "PLACEHOLDER_outputtitle" => string(outputtitle)
78 )
79
80 # 一意のファイル名を生成して仮ファイルを作成し、パラメータを書き込む
81 tempfile = "in.temp_script_$(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
86         lmp_mpi -log $(outputtitle).log -in $(tempfile)`)
87 end

```

ソースコード A.5: lammps-qsub.out.jl

```

1 ====
2 # 出力ファイル保管
3
4 このJulia コードは,
5   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
6   ,適切なディレクトリに保存します.
7
8 ## 機能
9
10 - ‘Glob’パッケージを使用してファイルマッチングを行う.
11 - 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,
12   指定のディレクトリに移動.
13 - 使用済みの仮LAMMPS ファイルを一括削除.
14
15 ## 手順
16
17 1. 出力ファイルを指定ディレクトリに保存.
18 2. 使用済みの仮LAMMPS ファイルを削除.
19
20 このコードは,出力ファイルの整理と保管を行います.
21 ===#
22
23 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.
24
25 file_extensions = ["log", "yaml", "lammpstrj"] # 出力ファイルの拡
26 張子

```

```

23
24 # 出力ファイルを保存
25 for file_ext in file_extensions
26     files = glob("*.${file_ext}")
27     for file in files
28         outputpath = "../outputdir/${file_ext}dir"
29         mkpath(outputpath)
30         script = read(file, String)
31         fp = open(joinpath(outputpath, "$(file)"), "w")
32
33         if file_ext == "log"
34             println(fp, "$(file)")
35         end
36
37         write(fp, script)
38         close(fp)
39         rm(file)
40     end
41 end
42
43 # 使用済みの仮LAMMPS ファイルを一括削除
44 files = glob("in.temp_*")
45 for file in files
46     rm(file)
47 end

```

A.3 プロットファイル

ソースコード A.6: plot_LJpotential.jl

```

1 # 汎用LJ ポテンシャル描画セル.
2 # パッケージ.
3 using Plots
4
5 # 関数定義.
6 function theta(r) # 階段関数.
7     return r > 0 ? 1 : 0

```

```

8 end
9 function phi(epsilon, sigma, r) # LJ ポテンシャル.
10    return 4.0 * epsilon * ((sigma/r)^12 - (sigma/r)^6)
11 end
12 function phi_tilde(r, epsilon, sigma, rc) # シフトアップとカットオ
13    フ.
14    return (phi(epsilon, sigma, r) - phi(epsilon, sigma, rc)) *
15    theta(rc - r)
16 end
17
18 # 粒子-粒子LJ ポтенシャルのパラメータ.
19 epsilon = 1.0
20 sigma = 1.0
21 rc = 3.0 * sigma
22
23 # Rd, Rt, Ra の配列.
24 Rd_values = range(0.0, length=1)
25 Rt_values = range(0.0, 0.5, length=5)
26 Ra_values = range(1.877, length=1)
27
28 # プロット概形.
29 plot(xlabel="r/σ", ylabel="^cf^95/ε")
30 xlims!(0.2, 2.5)
31 ylims!(-1.5, 3.0)
32 title!("LJ-Potential vs. r")
33 xlabel!("r/σ")
34 ylabel!("^cf^95/ε")
35
36 # 粒子-粒子LJ ポтенシャルのプロット.
37 plot!(r -> phi_tilde(r, epsilon, sigma, rc), label=
38    "Potential_pair; ε=$(round(epsilon,digits=1)), σ=$(round(sigma,
39    digits=1)), rc=$(round(3.0,digits=2)) σ", linestyle=:dash)
40
41 # プロットの追加.
42 for Rd in Rd_values,
43     Rt in Rt_values,
44     Ra in Ra_values
45     # 壁-粒子LJ ポтенシャルのパラメータ.

```

```

42 epsilon_wall = (1.0 - Rd) * epsilon
43 sigma_wall = (0.5 + Rt) * sigma
44 rc_wall = ((2 ^ (1 / 6)) + Ra) * sigma_wall
45 # 打つ点を調整.
46 x_values = range(Rt+0.3,3.0,length=10000)
47 y_values = phi_tilde.(x_values, epsilon_wall, sigma_wall,
48   rc_wall)
49 # 壁-粒子LJ ポテンシャルのプロット .
50 plot!(x_values, y_values, label="Potential_wall;  $\epsilon$  = $(round(
51   epsilon_wall,digits=1)),  $\sigma$  = $(round(sigma_wall,digits
52   =1)),  $\sigma_{rw}$  = $(round((2^(1/6))+Ra,digits=2)),  $\sigma_{rcw}$  = $(round
53   (((2^(1/6))+Ra)*sigma_wall,digits=2)), linestyle=:dash
54   )
55 end
56 display(plot!())
57 # savefig("")
58
59 ccall(:jl_tty_set_mode, Int32, (Ptr{Cvoid}, Int32), stdin.handle
60   , true)
61 read(stdin, 1)

```

参考文献

- [1] 蔵本由紀. リズム現象の世界. 東京大学出版会, 2013.