平成 26 年度修士論文

転がり摩擦のシミュレーション

平成 27 年 3 月 4 日
三重大学大学院　教育学研究科
教育科学専攻　理数・生活系教育領域
212M044　村田了祐

指導教員
國仲　寛人　准教授

目次

第１章　序論

1.1　はじめに

物体運動の解析はギリシャ時代から現代まで長く続けられている。 ギリシャ時代には物体に働く力の釣り合い等を議論する静力学が発展し、その後天体の運動の観測や振り子などの身の周りの道具を用いた実験を通して、物体の運動が議論されるようになった。 16 世紀に入るとニュートンが物体の運動を数学的に解析する手法を確立したおかげで様々な物理現象を深く理解できるようになり、 物体の運動をある程度予想することも可能になった。

ニュートンらが築きあげた古典力学においては、エネルギーや運動量といった物理量の保存則が重要な役割を果たす。だが、実際の力学的な現象を観察してみると、摩擦の存在によって保存則が成り立たないように見えることがある。例えば斜面上に置かれた物体が重力によって滑り落ちるときの力学的エネルギーを測定すると、一定ではないことがわかる。これは物体の持つ運動エネルギーの一部が、物体と斜面の間の摩擦によって熱エネルギーに変換され、物体や斜面の内部に散逸するためである。

このような摩擦による熱の発生は、工業製品の設計においてはできる限り排除したい要素である。例えば車輪を滑らかに回転させるためにボールベアリングが用いられる。これは摩擦による発熱を低減させる工夫と言える。またベアリングのボールのサイズや個数によっても発熱量は変化するので、発熱を減らすために気をつかう必要がある。このように産業の発展の歴史において、摩擦をいかに少なくするかということは、無視することのできない重要な問題であり続けている。摩擦を減らす工夫をする一方で、摩擦に関する知見を数多く集め、摩擦に関する法則を明らかにすることは、物理学や工学にとって有益なことであろう。だが、現代でさえも摩擦について解明されてないことはかなり多い。その要因として他の物理現象にはない摩擦の特異性がある。運動や力に関する理解は、巨視的なスケールでの理解と微視的なスケールでの理解の二種類に分けられる。このような暗黙の前提があるからこそ、巨視的な物体の運動について物体の振る舞いを理解し、微視的な分子の振る舞いについての理論を展開することができるのである。しかし、この前提に基づいて摩擦の理論を展開するのはとても難しい。なぜなら、摩擦は巨視的な現象である一方で、物体を構成する分子間に働く微視的な相互作用に起因する現象ともいえるからである。このような、特異な性質が摩擦の問題を複雑化している。

現代の科学・工学的研究においては、断層のすべりからナノマシンの設計に至るまでさまざまなスケールでの摩擦現象が登場する。それらの現象に見られる摩擦を大きく分類すると、固体どうしの摩擦、固体と流体間の摩擦、流体どうしの摩擦に分けられる[1]。さらに固体間の摩擦は物体が接触している固体の上を滑る際に生じる「滑り摩擦」、物体が転がる際に生じる「転がり摩擦」、回転軸において生じる「ピボット摩擦」に分類できる。本研究は、焦点を固体どうしの滑り摩擦、転がり摩擦に絞って研究を行う。

1.2　滑り摩擦と転がり摩擦の研究歴史

前セクションで様々な摩擦を紹介したが、この論文では固体同士の摩擦のみを考えることにする。固体同士の摩擦の中で最も詳細に調べられているのは滑り摩擦であろう。滑り摩擦についての研究を最初に行ったのは、イタリアの Leonardo Da Vinci(1452-1519)であると伝えられている[2]。彼は直方体の異なる大きさの面で摩擦力の違いを観察し、当時描いた摩擦実験のスケッチは今も残されている。このスケッチによると彼は今日の摩擦の法則に近い記述を残していることが分かる。しかし物理的に摩擦の法則を理解するためには力という概念が理解されていなければならない。力の概念が受け入れられたのはこの 200 年後であり、当時の人々にはこの実験はあまり評価されていなかった。

Newton によって力の概念が理解された後、フランスの Amonton は Da Vinci のスケッチとよく似た実験装置を組み立て、摩擦に関する法則性を求める研究を行った。

摩擦が私たちの知っているような形で認知されるようになったのは 18 世紀後半にフランス科学アカデミーが懸賞金付きで摩擦についての論文を募集したことがきっかけである。この募集を受けて、C.A.Coulomb はThéorie des mechines simplesを発表した。この論文に私たちになじみ深い

①摩擦力は垂直抗力（荷重）に比例し、見かけの接触面積に依存しない。

②動摩擦力は滑り速度に無関係である。

③静止摩擦力は動摩擦力より大きい。

という Coulomb の滑り摩擦の法則が様々な実験を通して導かれている[3]。滑り摩擦力を f 、を摩擦係数 μ 、垂直抗力を N とすると、以下の式で表せる。

$$f = \mu N \tag{1.2.1}$$

この論文において最も特徴的なのは　摩擦力は見かけの接触面積に依存しない、すなわち摩擦は物体の形状によらないことである。この事実をいかに説明するのかがその後の滑り摩擦に関する研究の課題となった。

滑り摩擦の要因に対する初期の説は 2 つ存在する[1]。1 つ目は物体の表面には必ず凸凹があり、その凸凹を乗り越えるために必要な力が摩擦力であるという説、もうひとつは接触面において上下の物質間にはたらく分子間力などの相互作用による力であるという説である。この二つの説は、接触面を滑らかにした場合に結果が大きく異なってくる。前者では表面を滑らかにすることで凸凹が減り、摩擦力は小さくなるのに対し、後者はより多くの分子が吸着するため摩擦力は大きくなるのである。この二つの説の論争は 20 世紀半ばまで続いたが、イギリスの Hardy によって決着がつけられた[1]。彼は接触部に各種流体膜を張り付けることで、表面の形状が全く変わらなくても摩擦力を減少させることを実験によって証明した。このことにより物体の凸凹によって摩擦力が生まれるという説のみでは摩擦力の説明ができないことが証明された。さらに Holm や Bowden は表面の不純物を取り去ることによって摩擦力が劇的に増加することも実験で証明し、これによって摩擦力の主な要因は分子間力であることが証明された[1]

近年の摩擦現象に関する実験として Holst らによる研究は特筆に値する[1]。彼らは厚紙同士を擦りあわせる実験で、スティックスリップと呼ばれる現象について観察した。スティックスリップとは、低速度で物体を動かしたときにおこる、吸着状態（スティック）と滑り状態（スリップ）を繰り返す運動のことである。彼らの実験により吸着状態においても物体は完全に止まっているのではなく、わずかに動いていることが分かった。この現象は、接触面の内部状態に着目し、接触面において起こる膨張や、流動化が主な要因ではないかといわれている。

一方転がり摩擦についても Coulomb はThéorie des mechines simplesに書いている[1]。Coulomb は滑り摩擦と同じ要領で、転がり摩擦係数 μ_{roll} をトルクに対する抵抗力として

$$T_{rf} = \mu_{\mathrm{roll}} Mg \qquad (1.2.2)$$

と定義した。ただし T_{rf} は転がり抵抗としてのトルク、M は転がる物体の質量、gは重力加速度またはそれに相当する加速度、R は接触部における対象の曲率半径を表す。

その後も多くの研究者が転がり摩擦について調べてきた。Reynolds は平面上で転がるボールについて研究し、転がり抵抗の原因はボールと平面間のマイクロスリップ、すなわち細かいスリップによるエネルギー散逸であるとした[4]。しかしマイクロスリップによるエネルギー散逸は転がり抵抗によるエネルギー散逸にくらべてかなり小さいものになっている。Tomlinson は転がり抵抗は滑り摩擦と同様、接触面の分子による吸着力が関係しているとした[4]。また、Heathcote は接触面において、球の中心から平面までの距離が一様でないことにより生じるスリップが原因であるとした[4]。しかしこれらの理論で生じる転がり抵抗は実験によって導出された転がり抵抗よりも小さく、説明としてはまだ不十分なものであった。その後、Tabor たちは転がり摩擦の主な原因はヒステリシス損失であると考えた[4]。ヒステリシス損失とは、物体が転がった後にすぐ元に戻るという弾性的な理論ではなく、物体が転がった後、元には戻らない塑性的な面もあるという点に目をつけた。この理論に基づいた研究は Tabor と Eldredge によって初めて成し遂げられた[4]。彼らは異なる大きさの金属球を用いて転がり摩擦力が荷重の 3 分の 2 乗に比例する事を証明した。

近年ではなのスケールの微小な物体についても転がり摩擦が調べられている。Lee たちはナノサイズの球の運動をコンピューターシミュレーションで再現し、転がり摩擦力は荷重の 1.84 乗に比例することを発見した[4]。このように摩擦についての議論は今もなお続いている。

1.3　分子シミュレーションについて

前セクションで述べたような摩擦について実物を用いた実験を行うと、かなりの労力が必要になる。この労力を大幅に削減することができるのがコンピューターシミュレーションである。このセクションでは本研究で用いたシミュレーションの概要について説明する。この世界に存在する物質はすべて分子で構成されている。したがって分子について理解できれば物体の運動の詳しい状況も分かるはずである。しかし分子の数は 10^{23} スケールである。解析するためには 3 次元の物体の場合 1 分子につき 3 つの運動方程式をたてなければならないのでかなり膨大な量の方程式を解かなくてはならない。いくら時間があっても解明することは不可能であると思われていた。このような問題から統計力学が生まれた。統計理学ではこの膨大な分子の平均値や分散などを求めることによって分子の様子の理解を試みる学問である。統計力学は現代の物理学に大きく寄与したが、統計力学では例えば膨大な分子から 1 分子取り出すといった分子ごとの運動を表すことはできていない。現代では機器のハイテク化やコンパクト化をさせるために原子 1 個単位、いわゆるナノスケールの理論が必要になってくるのである。

ナノスケールとはサブナノメートルから数百ナノメートル単位までと非常に広い範囲が対象である。ネオン原子の場合、直径が 0.288nm であることから 1nm にはおよそ 3 個のネオン原子が並ぶ[5]。そうすると長さのスケールが 100nm だとすると 2 次元的な構造物では 90000 原子、立方体にすると 27×10^6 個にもなるのである。1 分子あたり一つの運動方程式をたてながらこの膨大な分子の相互関係を求めることを可能にしたのはコンピューターであった。物質を分子・原子などの粒子からなる系とみなし、膨大な粒子を含む系についてコンピューターを使ってシミュレーション計算を行う手法を分子シミュレーションという。

分子シミュレーションの歴史は 1953 年頃、Newton の運動方程式を「電子計算機」を使って数値的に解くことから始まった。非平衡状態、すなわち温度が不均一な物体が、時間が経つにつれて温度が均一になっていくという日常的なメカニズムを調べた Fermi、Pasta、Ulam の計算機実験(1955)や、およそ 1000 個のアルゴン原子集団を用いて微視的な原子間相互作用から巨視的な液体としての性質を明らかにした Rahman の実験(1964)などが代表的な初期の研究である[5]。1977 年になると Karplus 達が、基本的なタンパク質であるウシ膵臓トリプシン阻害因子にたいして分子シミュレーションを実行して、タンパク質分子内部の原子運動が確かにその機能を担っていることを証明した[5]。現在ではスーパーコンピューター「京」で 10^{21} 個の分子での分子シミュレーションが可能になっているように分子シミュレーションは巨視的な運動の説明ができつつある。

主な分子シミュレーションを 3 つ上げると以下ものがある[6,7]。

・分子力学法　　MM　Molecular Mechanics
・モンテカルロ法　MC　Monte Carlo
・分子動力学　　MD　Molecular Dynamics

一つ目の分子力学法とは有機物関連化学でよく用いられる手法である。まず孤立分子または集団分子にポテンシャル関数を与える。これに古典的または量子的な運動方程式を適用したり関数自体を Taylor 展開したものを計算させたりすることで熱力学関数や構造、弾性定数など、多くの物理量を導き出すものである。二つ目のモンテカルロ法とは実際の物理現象で観察されるような確率を基準にした乱数を用いて分子の位置を決めて微視的状態を作成していく方法である。したがってモンテカルロ法は運動方程式を使わず、時間的な概念も原則的には入ってこない。三つ目の分子動力学は Newton の運動方程式をもとにポテンシャル関数がつくる力場中に置かれた分子達をそれぞれ一定時間シミュレートして物理量を求める方法である[8]。したがって分子力学法、モンテカルロ法では扱っている系の温度がそこにある分子の運動によるものとして扱われていない。したがって無機物質の固体、液体、ガラス状態について、広い範囲の温度や圧力に関して構造や物性を調べるためには MD 法が最も適しているといえる。もちろん MD 法が古典力学の基礎である Newton の運動方程式をもとにしている時点で、この手法は万能ではないといえる。電子が関わる等、量子的な性質も考慮しなければならない場合に太刀打ちできないからである。しかしこれらの困難や不可能はコンピューターの発達などにより将来的には解決されることも予想できる。その意味でも分子動力学法を用いた研究を進めていくことは大変重要なことだといえる。

ただし、シミュレーションはあくまで模擬である。系を支配しているパラメータをすべて考慮に入れている確信はないので実験や理論との連携をなくしては議論を展開することはできない。測定の捜査を省ける手段として、またはナノスケールにおける実験のように現段階では実験が不可能な時実験の先駆けとしてシミュレーションが用いられていることを忘れてはならない。

1.4.　転がり摩擦係数の導出

このセクションでは転がり摩擦とは何かをみていく。図 1.4-1 のような硬い基板上を転がる剛体球を考える。球の重心の速さをv、球の転がり角速度をω、滑り摩擦力をf、牽引力をF、球の質量をmと定義する。この思考実験において球は基板のある一点のみで接しており、この点から摩擦力を受けている。

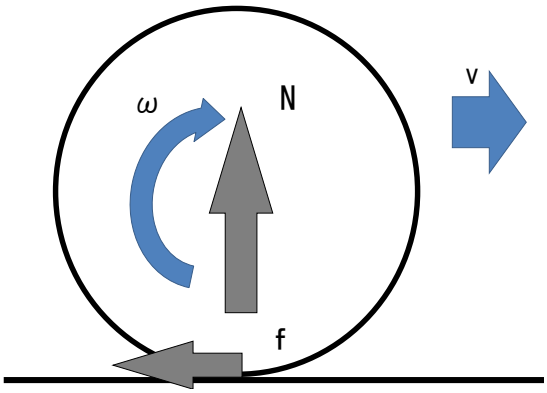


図 1.4-1　剛体球の転がりの様子

したがってこの系における Newton の運動方程式、剛体の運動方程式より速度と角速度の時間微分は

$$m\frac{dv}{dt} = F - f \qquad (1.4.1)$$

$$I\frac{d\omega}{dt} = fR \qquad (1.4.2)$$

となる。さらに滑りが全く存在しないという条件

$$v = R\omega \qquad (1.4.3)$$

を加えてみる。球はスリップせず、転がりのみによって移動する場合は日常においても十分に起こりうる場合である。ただし滑りがない場合でも滑り摩擦力は働く。

(1.4.1)-(1.4.3)式をまとめると、球の慣性モーメントは$\frac{2}{5}mR^2$ なので、

$$\frac{7}{5}m\frac{dv}{dt} = F \qquad (1.4.4)$$

という式が導出される。しかしこれは球が剛体であるという仮定の下で行った計算である。パンクした自転車と空気いれたての自転車では進む労力が明らかに違うように、我々の日常ではスリップが起こらない場合においても(1.4.4)式が単純に当てはまらない。

そこで今度は球を弾性体として考える。すると球は図 1.4-2 のように少し変形し、垂直抗力は一点ではなく接触面全体から受けるようになる。重心のζ成分を 0、牽引方向を正 としてζ座標を取り、垂直抗力の分布を$q(\zeta)$ と、表すと、垂直抗力 N は

$$N = \int d\zeta\, q(\zeta) \tag{1.4.5}$$

と表せる。また回転運動の方程式は

$$I\frac{d\omega}{dt} = fR - \int d\zeta\, \zeta q(\zeta) \tag{1.4.6}$$

と書き加えられる。第二項が転がり摩擦の項である。ここで、転がり摩擦係数 μ_{roll} を Coulomb の式(1.1.2)に基いて、

$$\mu_{\mathrm{roll}}R = \frac{\int d\zeta\, \zeta q(\zeta)}{\int d\zeta\, q(\zeta)} = \frac{\int d\zeta\, \zeta q(\zeta)}{N} \tag{1.4.7}$$

と定義すると、回転運動の方程式は

$$I\frac{d\omega}{dt} = fR - \mu_{\mathrm{roll}}NR \tag{1.4.8}$$

と書き換えることができ、並進運動の方程式

$$m\frac{dv}{dt} = F - f \tag{1.4.9}$$

と(1.4.8)式を組み合わせると球の慣性モーメント $I = \frac{2}{5}mR^2$ なので、転がり摩擦係数は

$$\mu_{\mathrm{roll}} = \frac{F - \frac{2}{5}mR\frac{d\omega}{dt} - m\frac{dv}{dt}}{N} \tag{1.4.10}$$

である。したがって転がる球の牽引力、質量、半径、角速度の時間微分、加速度、垂直抗力を求めることができれば転がり摩擦係数は求めることができるといえる。

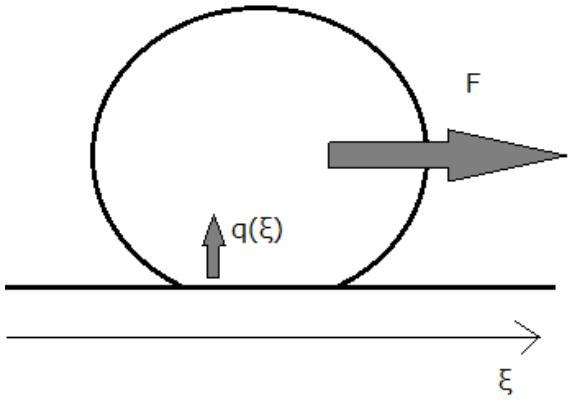


図 1.4-2 弾性球の転がり運動の様子

1.5 研究目的

本研究では、①分子動力学法を用いてネオン基板上をアルゴン球が転がるシミュレーションを作成し、球の振る舞いの変化を実験的に明らかにすること、②分子間力や牽引力、温度、荷重といったパラメータを変化が、運動にどのような変化をもたらすのかを明らかにすることを目標とする。

第 2 章　方法

2.1 シミュレーションの概要

ここでは大まかなシミュレーションの流れを示す。まずアルゴン原子でできた FCC 結晶の球とネオン原子でできた FCC 結晶の直方体を作成した。次に作成した球を基板となる直方体上の端に配置し、原子に初期速度を与えた。原子の初期速度は所望の温度に相当する Maxwell 分布に従う乱数、原子間の相互作用はアルゴン原子・ネオン原子ともに Lennard-Jones ポテンシャルで与えた。基板原子の境界条件は基板の最下層の粒子のみを固定し、それ以外の境界は自由境界とした。

この球に力を加えて動かした。各原子の運動方程式は Symplectic 積分を用いて解き、系のエネルギーを一定に保ちながら原子の位置を時間発展させた。 シミュレーションの過程は大きく二つに分けられ、前半は 0.0023 秒おきに速度スケーリング法を導入して系を温度一定の状態にし、後半では球に力を加え、他端まで移動するまで時間を進め、0.023 秒おきに全粒子の位置や転がる球の重心速度などの物理量を記録させた。なお球が基板に吸着して移動しない場合は 2.3 秒を目途に記録を終了させた。図 2.1 は以上のプロセスの模式図である。以後順番に詳細を述べる。

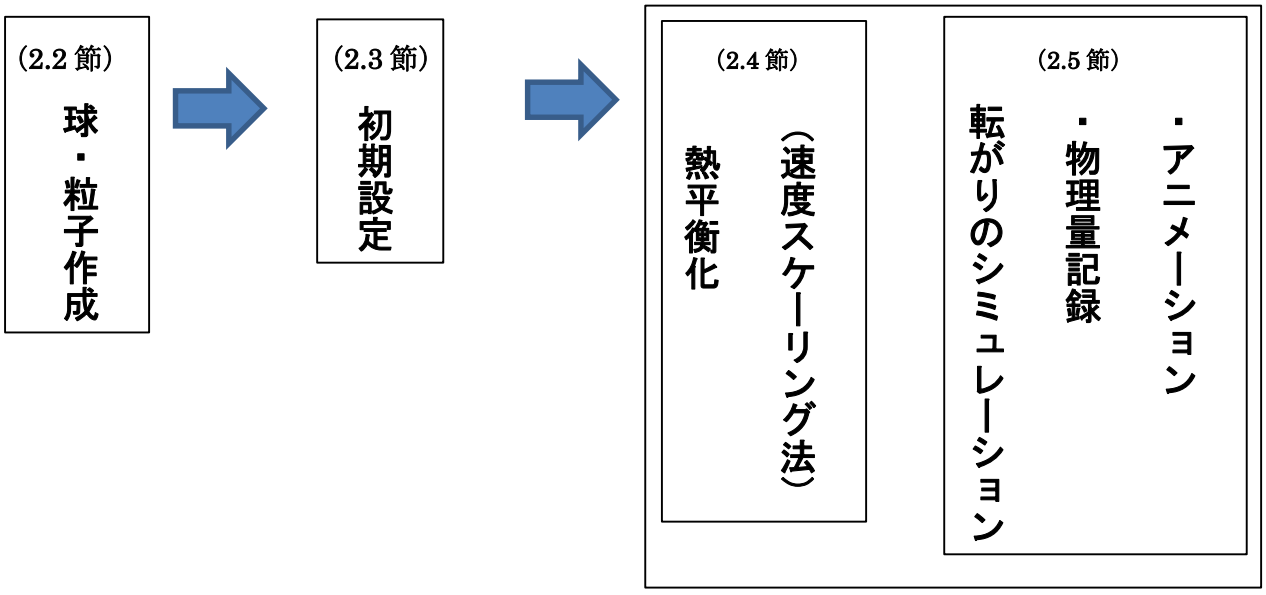


図 2.1　プログラムのアルゴリズム

2.2. 粒子作成

シミュレーションにおける原子の初期配置について記述する。

アルゴン原子、ネオン原子は低温下で共に面心立方格子状の結晶を作る[9]。原子間ポテンシャルは後に式(2.4.2)で定義するが、格子定数aは原子間ポテンシャルが最低となる原子間距離r_0を用いて、

$$a = \frac{1}{\sqrt{2}} r_0 \qquad (2.2.1)$$

となるように決めた。

次に面心立方格子の単位格子を前後、上下、左右に整数倍してネオン基板を作成した（図 2.2-1）。さらに、十分に大きな立方体の中心から、距離が R より離れた位置にある原子を選ぶことで半径 R の球を作成した（図 2.2-2）。図 2.2-1 は粒子数 5226、18.3nm×8.91nm×1.44nm のネオン基板、図 2.2-2 は粒子数 116、半径 1.74nm のアルゴン球である。

作成した基盤の上に球を配置し、図 2.2-3 のように基盤の 1 頂点を原点とし、座標を設定した。この座標系において球の重心の位置は(1.89nm, -4.09nm, 2.29nm ）であった。

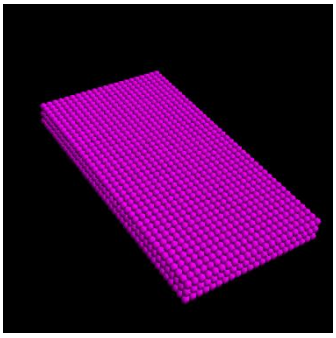


図 2.2-1　ネオン基板

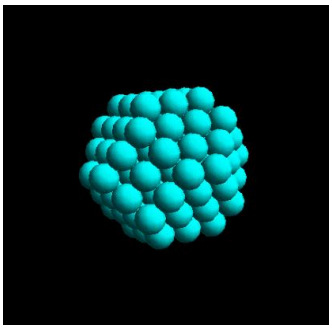


図 2.2-2　アルゴン球

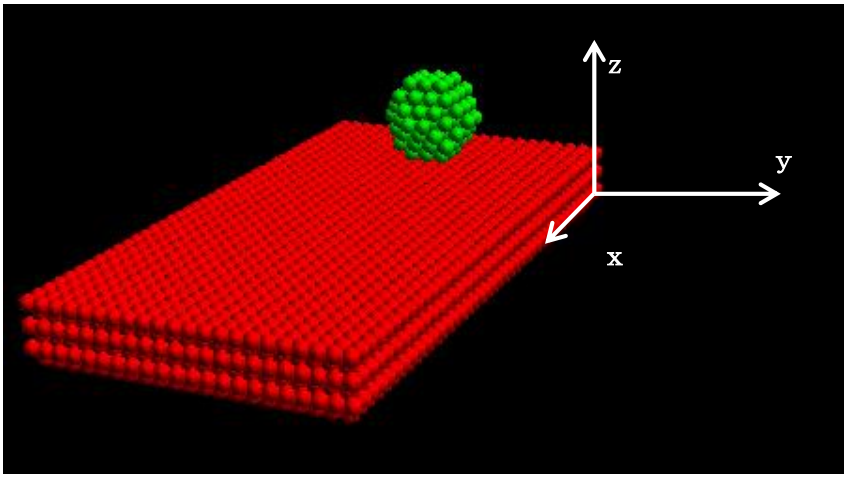


図 2.2-3　基盤と球の配置と座標

2.3 初期速度

ここではMaxwell分布に従う分子速度の生成法を解説する[9]。温度 Tの熱平衡状態にある分子の速さ v は、

$$f(v) = \left(\frac{m}{2\pi kT}\right)^{3/2} \exp\left\{-\frac{m}{2\pi kT}(v_x{}^2 + v_y{}^2 + v_z{}^2)\right\} \qquad (-\infty < x < \infty) \qquad (2.3.1)$$

という確率分布に従う。ここで k はボルツマン定数である。この分布に従う速さをもつ速度 $(v_x,\ v_y,\ v_z)$ を求めるために、まず一様乱数R_1、R_2を生成し、 i 番目の原子における速度の j 成分（$j = x, y, z$）を v_{ij} とおくと、速度は

$$v_{ij} = \sqrt{-T\log R_1}\quad \sin 2\pi R_2 \qquad (2.3.2)$$

という式で求められる。この方法をBox-Muller法という[9]。

2.4 熱平衡化の方法

系の原子にはたらく力は分子間力のみなので、i番目の原子の位置を$\boldsymbol{r_i}$とすると運動方程式は、

$$m\frac{d^2\boldsymbol{r_i}}{dt^2} = \sum_{j\neq i} -\frac{\partial U_{ij}}{\partial \boldsymbol{r_{ij}}} + \sum_{k} -\frac{\partial U_{ik}}{\partial \boldsymbol{r_{ik}}} \qquad (2.4.1)$$

と表せる。ここで、U_{ij}は原子iと原子jの間にはたらく原子間ポテンシャルであり、右辺第1項は同種の原子から受ける力の総和、右辺第2項は異種の原子から受ける力の総和を表している。本研究では原子iと原子j間の距離をr_{ij}として、原子間ポテンシャルを次のように定めた。

$$U(r_{ij}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - c\left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \qquad (2.4.2)$$

cは吸着パラメータであり同種原子の場合は$c = 1$、異種原子の場合は$c < 1$にした[11-13]（付録2参照）。系を構成する原子に、所望の温度に相当するMaxwell分布に従う速度を与えても、時間がたつと別の温度に変化することがよくある。そこで系を所望の温度に熱平衡化する方法として、速度スケーリング法を用いた[3]。速度スケーリング法は決められた時間ステップごとに全原子の速度を一斉にスケーリングすることで、温度を強制的に一定値に保たせようとする方法である。系の温度 T と原子の速度v_iの関係は次式のように表わされる。

$$3N\frac{kT}{2} = \sum_{i=1}^{N}\frac{m\boldsymbol{v_i^2}}{2} \qquad (2.4.2)$$

系が設定温度 T を与えるようにするために、次のように速度$\boldsymbol{v_i}$ を新しい速度 $\boldsymbol{v_i}'$ に置き換える。

$$\boldsymbol{v_i'} = \beta\boldsymbol{v_i} \qquad (2.4.3)$$

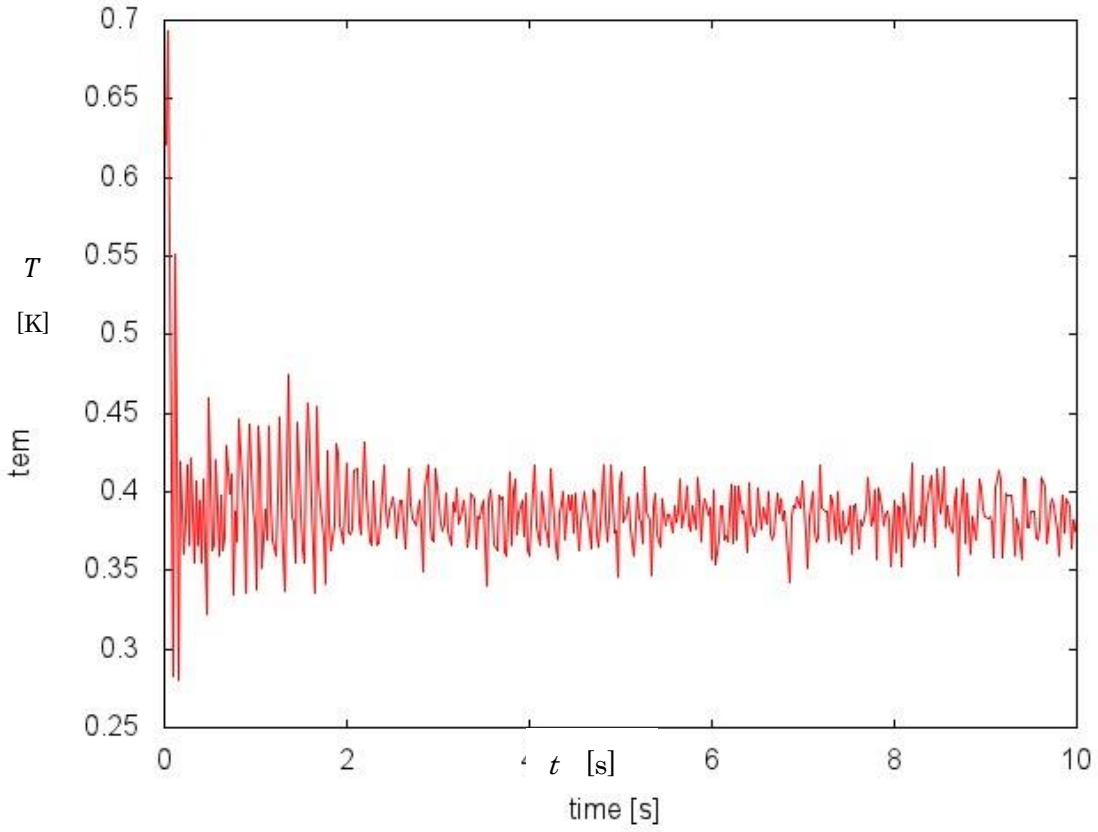
ただし係数βは、次のように定める。

$$\beta = \sqrt{\frac{3NkT}{m\sum_{i=1}^{N}\frac{m\boldsymbol{v_i^2}}{2}}} \tag{2.4.4}$$

本研究では 0.0023 秒ごとに速度スケーリング法を用いて熱平衡化を行っている。

図 2.4-1 は本研究で用いた$c = 0.60$ の系を速度スケーリング法で温度 0.38K に熱平衡化したときの時間と系の温度の関係を表したものである。この図より、およそ 2 秒を越えたあたりから温度の振れ幅が小さくなっているのが分かる。したがって速度スケーリング法を 2.3 秒（プログラムの 10000 ステップ）行ってから物理量の計測を開始した。

図 2.4-1　速度スケーリング時の温度変化の様子



2.5　転がりのシミュレーション

熱平衡化によって系の温度が一定に保たれたところで、系の x 方向とz方向に外力F_x、 F_z を加えて他端まで移動するまで時間を進めた。球と基板を構成する原子の運動方程式はそれぞれ次のとおりである。

$$\text{基板：} m\frac{d^2\boldsymbol{r_i}}{dt^2} = \sum_{j\neq i} -\frac{\partial U_{ij}}{\partial \boldsymbol{r_{ij}}} + \sum_{j} -\frac{\partial V_{ij}}{\partial \boldsymbol{r_{ij}}} + F_x\mathbf{e_x} + F_z\mathbf{e_z} \tag{2.5.1}$$

$$\text{球：} m\frac{d^2\boldsymbol{r_i}}{dt^2} = \sum_{j\neq i} -\frac{\partial \acute{U}_{ij}}{\partial \boldsymbol{r_{ij}}} + \sum_{j} -\frac{\partial V_{ij}}{\partial \boldsymbol{r_{ij}}} + F_x\mathbf{e_x} + F_z\mathbf{e_z} \tag{2.5.2}$$

$\mathbf{e_x}, \mathbf{e_z}$ は x 方向、 z 方向の単位ベクトル、U_{ij}、$\acute{U}_{ij}$、V_{ij}はネオンどうし、アルゴンどうし、ネオン-アルゴン間のポテンシャルエネルギーである。時間を進める際に 0.023 秒おきに全粒子の位置、転がる球の重心速度および加速度、球に加わる合力、(1.4.10)式に表わす転がり摩擦係数、エネルギー誤差、転がり角$\theta(t)$、転がり角速度$\omega(t)$を記録させた。転がり角は、球の重心位置 O と球の表面にある 3 粒子の平均位置 P から$\overrightarrow{\mathrm{OP}} = (x_{OP}, y_{OP}, z_{OP})$を求め、

$$\theta(t) = \tan^{-1}\frac{z_{OP}(t)}{x_{OP}(t)} - \theta_0 \tag{2.5.2}$$

と定義した（図 2.5-1）。ただしθ_0は転がり開始時を$t = 0$として、

$$\theta_0 = \tan^{-1}\frac{z_{OP}(0)}{x_{OP}(0)} \tag{2.5.2}$$

である。転がり角速度は転がり角の差分

$$\omega(t + \Delta t) = \frac{\theta(t + \Delta t) - \theta(t)}{\Delta t} \tag{2.5.3}$$

とした。ここでΔt は 0.023 秒である。

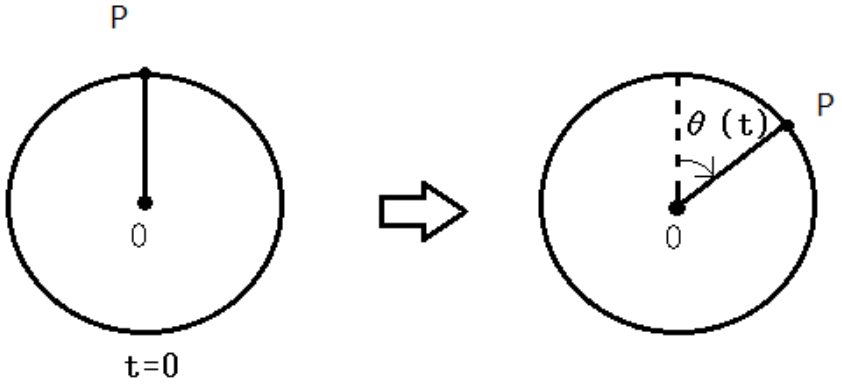


図 2.5-1. 転がり角

2.6 測定中のエネルギー誤差

本研究で扱うシミュレーションはミクロな系であるので、分子間の吸着のみによって摩擦が生じている。したがってエネルギー散逸がなく、系は小正準集団である。小正準集団の系ではエネルギー保存則が成立し、Newton の運動方程式をもとに時間発展を行う symplectic 積分（付録 2）が適用できる。このセクションでは symplectic 積分によって時間発展させた系がエネルギー保存則を満たしているのかを調べる。

系の全エネルギーは次の式で定義される。

$$E = \sum_i \frac{1}{2} m_1 {v_{1i}}^2 + \sum_j \frac{1}{2} m_2 {v_{2j}}^2 + \sum_{j \neq i} \sum_i U_{ij} + \sum_{j \neq i} \sum_i \acute{U}_{ij} + \sum_{j \neq i} \sum_i V_{ij} + \Phi_x + \Phi_z \quad (2.6.1)$$

ここでm_1、m_2はネオン及びアルゴン原子 1 個当たりの質量、Φ_x、Φ_zは

$$F_x \mathbf{e_x} = \nabla \Phi_x \quad (2.6.2)$$

$$F_z \mathbf{e_z} = \nabla \Phi_z \quad (2.6.3)$$

をみたす。エネルギー誤差$\Delta E(t)$を次のように定義して、系がエネルギー保存則を満たしているかどうかを確認した[10]。

$$\Delta E(t) = \frac{E(t) - E_0}{E_0} \quad (2.6.4)$$

ここで $E(t)$ は時刻 t における全エネルギー、E_0 は時刻$t = 0$ における全エネルギーを表している。このように$\Delta E(t)$を表すと、エネルギーが初期状態から変化するほど$\Delta E(t)$ は大きな値をとる。図 2.6-1 にシミュレーションで用いる系を、設定温度 0.770K に設定し、$F_x =$ 32.9 pN、$F_z = 4.24$ pN を球に加えた時の時間とΔEの関係を示した。ここで吸着パラメータ c ＝0.56 である。ΔEの値が小数第 4 位のオーダーで推移しているため、エネルギー損失は全体の 0.01%のオーダーである。よって系はエネルギーを保存していると言える。他の系においてもエネルギー誤差は小数第 3 位以下のオーダーであったため、エネルギー損失は全体の 0.1%以下のオーダーである。よって今回シミュレーションを行った系で、エネルギーは保存しており、シミュレーションは現実的なものになっているといえる。

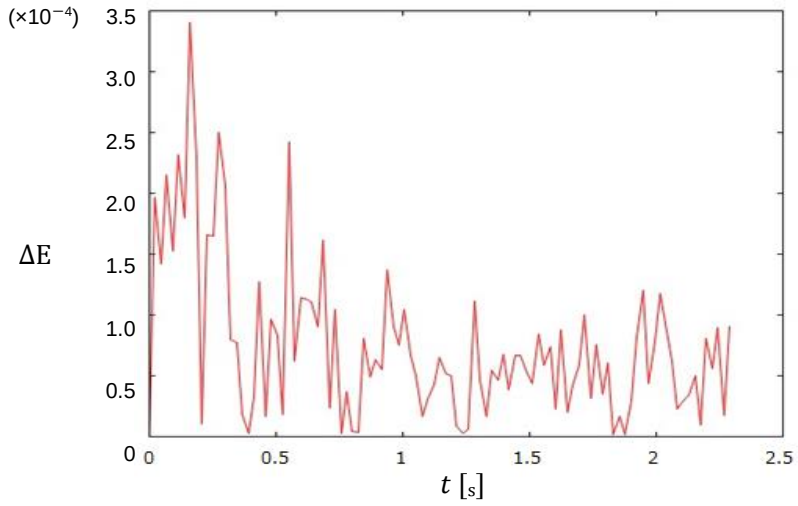


図 2.6-1　時間とエネルギー誤差の関係

2.7　カットオフ

シミュレーションの実行において時間計算を最も要するのは粒子間の相互作用力とエネルギーの計算である。なぜなら N 個の粒子が存在する系において 1 個の分子は $N-1$ 個の粒子と相互作用する。したがって 1 ステップごとに相互作用力の計算と相互作用ポテンシャルの計算はそれぞれ $N(N-1)$ 回行わなければならないからである。また、今回利用している相互作用ポテンシャルは Lennard-Jones ポテンシャルであり、原子間距離が遠くなると相互作用力は近傍の粒子に比べてかなり小さくなる。そのため、遠方の原子からの力を計算することは、計算コストの無駄になる。

そこで今回はカットオフを導入した。カットオフとは、プログラムの中にカットオフ距離を導入し、相互作用を計算する際に中心が対象となる粒子、半径がカットオフ距離である球を作る（図 2.7-1）。この球内に入っている粒子（黄色の粒子）のみ相互作用ポテンシャル、相互作用力を計算し、球の外側の粒子（水色の粒子）は相互作用力が弱いため、ないものとして計算する方法である。こうして 1 原子あたりと相互作用する粒子数を大幅に減らすことができる。

カットオフを導入するにあたって注意しなければならないのはエネルギー誤差である。カットオフ半径を短くしすぎると、エネルギーが保存しなくなる可能性がある。図 2.7-2、2.7-3 は、カットオフ距離とエネルギー誤差のグラフである。上のグラフはネオンどうしの原子間のカットオフの距離、下のグラフは異種原子のカットオフ距離を示している。 σ はネオン原子の直径、σ'はネオン原子の半径とアルゴン原子の半径の和を表す。また、図の∞印はカットオフを適用していないときの値である。図より、カットオフ距離が短すぎると設定した距離より遠い原子の保存力を無視してしまうので、ネオン原子のカットオフ距離とエネルギー誤差は大きな値をとる。本研究では、エネルギー誤差がカットオフを適用しない状態とほぼ変わらないように、同種原子の場合は原子直径の 4 倍、異種原子の場合はそれぞれの半径の和の 6 倍にカットオフを設定してシミュレーションを行った。

-

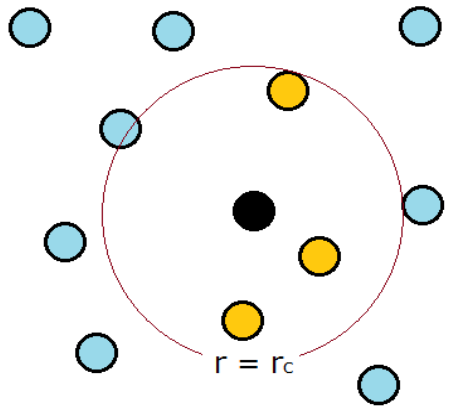


図 2.7-1　カットオフの様子

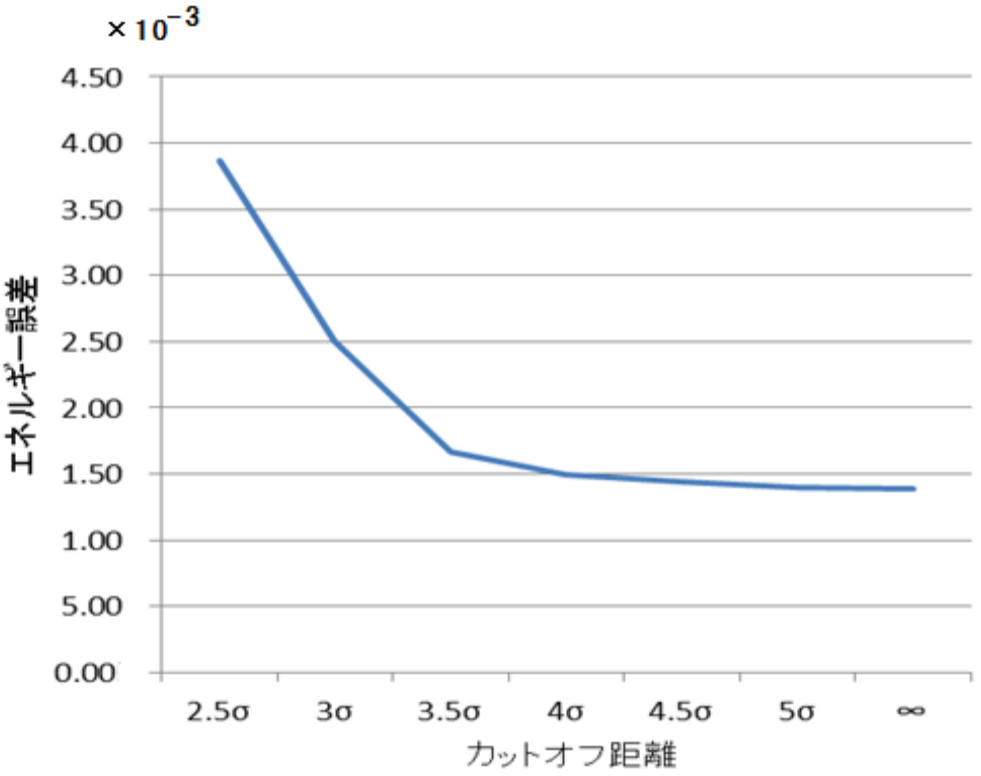


図 2.7-2　ネオン粒子のカットオフ距離とエネルギー誤差の関係、σ はネオンの原子直径

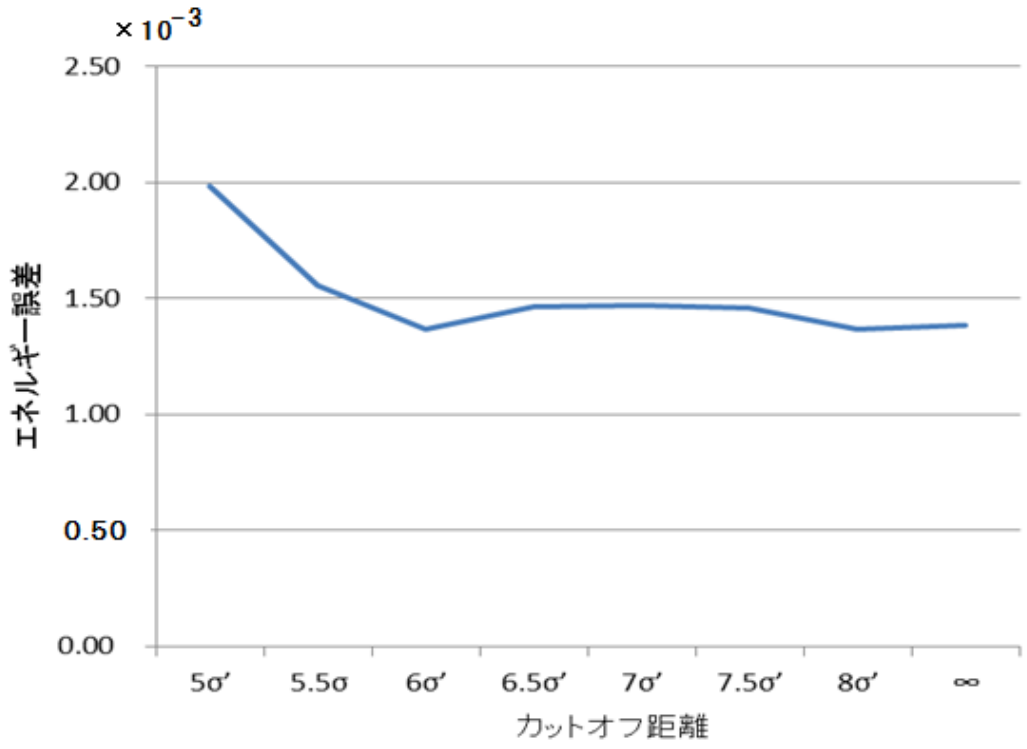


図 2.7-3　ネオン-アルゴン間におけるカットオフ距離とエネルギー誤差の関係
σ はネオンの原子直径とアルゴンの原子直径の平均

第 3 章　シミュレーション結果

アニメーションを用いて球の様子を解析したところ、球の運動状態は大きく 6 つに分けられた。6 つの状態とは、①基板の上を滑り続ける状態、②基板の上を転がり続ける状態、③滑りと転がりを繰り返す状態、④基板の原子が剥離して球に付着して、滑りまたは転がりを行う状態、⑤基板に吸着して止まってしまう状態、⑥球の一部が破損して分離する状態、⑦球が着地せずに跳んでいく状態である。

以下、それぞれを「滑り状態」、「転がり状態」、「転がり－滑り状態」、「剥離状態」、「吸着状態」、「破損状態」、「跳躍状態」とよぶことにする。この章では各状態について詳しく説明し温度や吸着パラメータ等のパラメータと出現する運動状態の関係を詳細に調べる。

3.1 運動状態について

3.1.1　滑り状態

「滑り状態」は球が回転運動を行うことなく基板上を滑り落ちる状態のことである。球の粒子数が少なく、球に凹凸が生じたために滑り状態は起こったのだと考えられる。この観測では球の転がり角が 90° 以下の場合を滑り状態と定義した。図 3.1-1 はある滑り状態における v_x と時間の関係を表したものである（温度 0.77K, $F_x = 21.2$ pN, $F_z = 4.24$ pN, c =0.56）。$t = 0$ は熱平衡化終了直後の時刻である。図 3.1-2 が右肩上がりであることから、最初は球が静止していることが分かり、時刻が 0.5 秒を超えると、球にはたらく牽引力により物体は動き始め、物体は等速で運動を始めることが分かる。図 3.1-2 は同じ状態で、転がり角 θ と時間の関係を表したもの、図 3.1-3 は転がり角速度 ω と時間の関係を表したものである。転がり角 θ はなだらかに増加と減少を繰り返しながら変化している。さらに図 3.1-4 より転がり角速度は転がり角の変化時に大きく変わるような尖ったグラフになった。これより球は回転方向を周期的に変えながら x 軸正の方向に並進運動を行っていることが分かる。

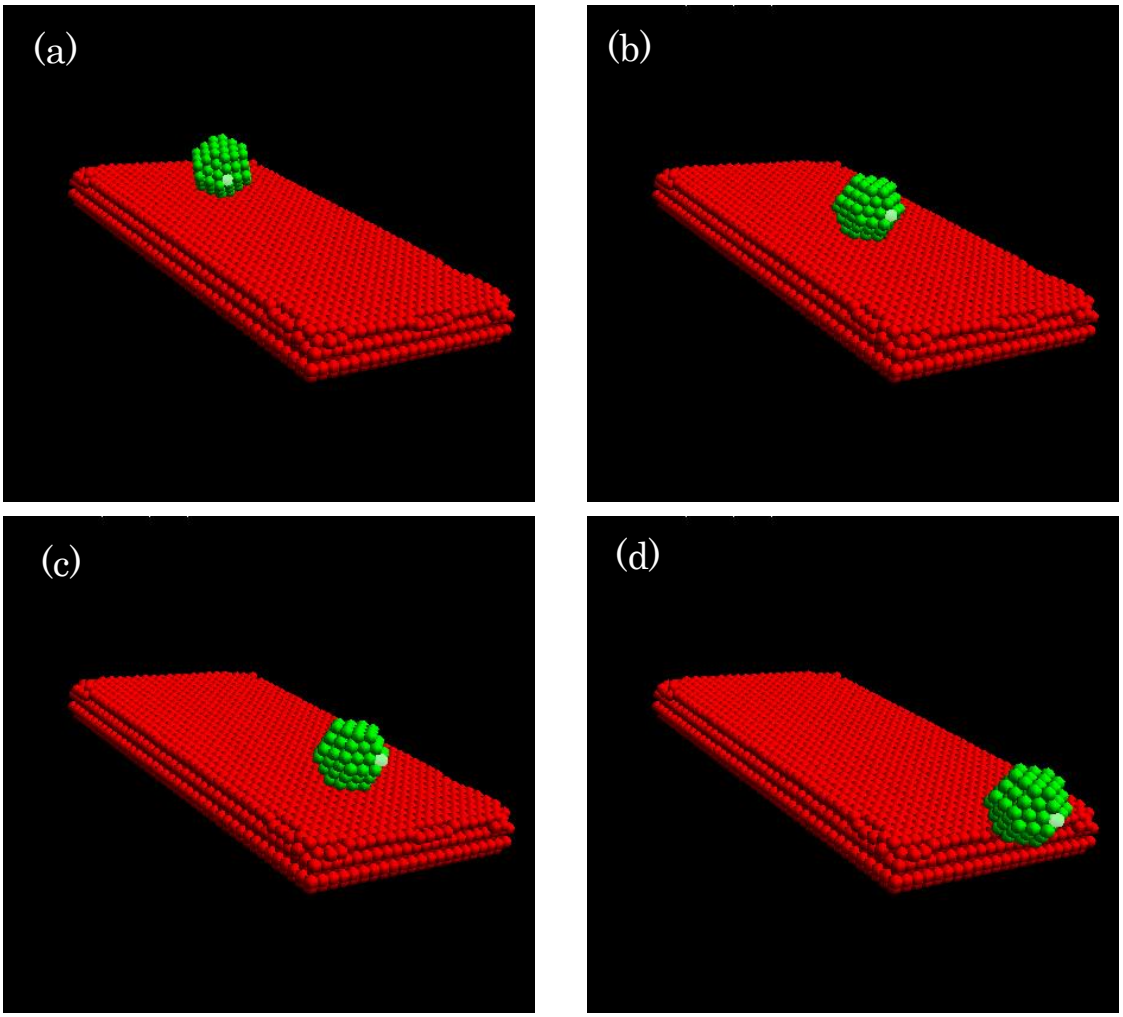


図 3.1-1　滑り状態の様子(a) $t = 0.69$s, (b) $t = 1.03$s, (c) $t = 1.21$s, (d) $t = 1.39$s

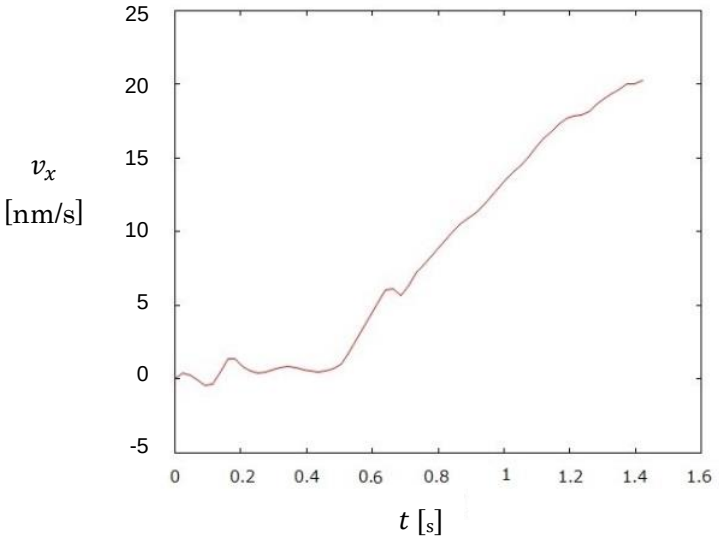


図 3.1-2 滑り状態における時間とv_xの関係

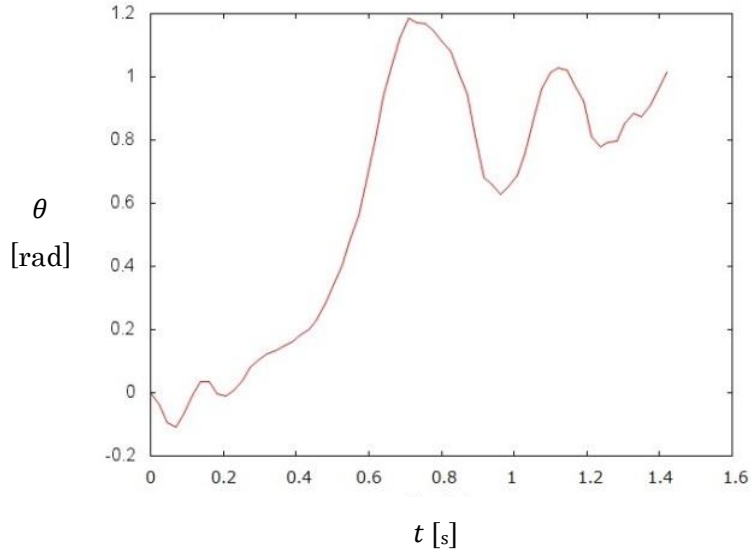


図 3.1-3 滑り状態における時間と転がり角の関係

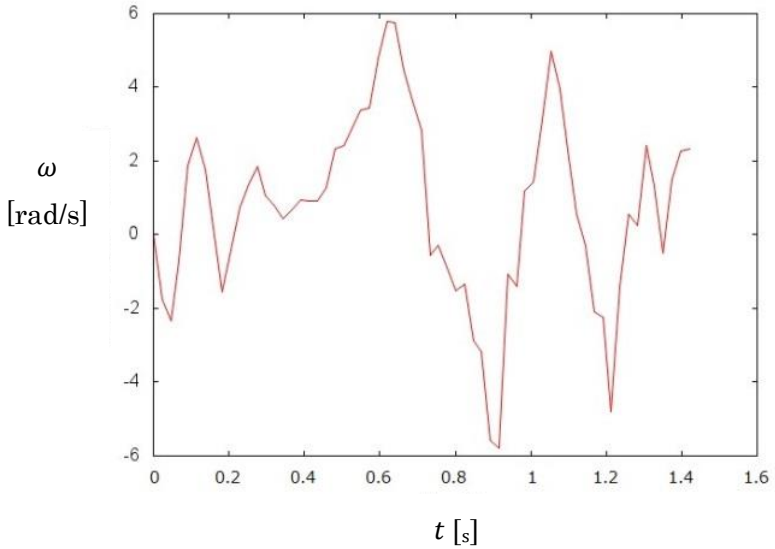


図 3.1-4 滑り状態における時間と転がり角速度の関係

3.1.2　転がり状態

「転がり状態」とは原子球が基板上を転がる状態である（図 3.1-5）。ここでは、球が基板の端に到達するまでに球の転がり角が 360° 以上変化した場合を転がり状態とした。

図 3.1-6 は転がり状態におけるv_xと時間の関係である（温度 0.77K, $F_x = 63.6\text{pN}, F_z = 4.24\text{pN}, c = 0.61$）。この図より球の速度は単調に増加していることが分かる。

図 3.1-7 は転がり角 θ と時間の関係を表したもの、図 3.1-8 は転がり角速度ω と時間の関係を表したものである。転がり角 θ はなだらかに増加、転がり角速度 ω も少しずつ増加していることが分かる。以上の結果より球は牽引力を受けて、加速しながら転がっていることが分かる。

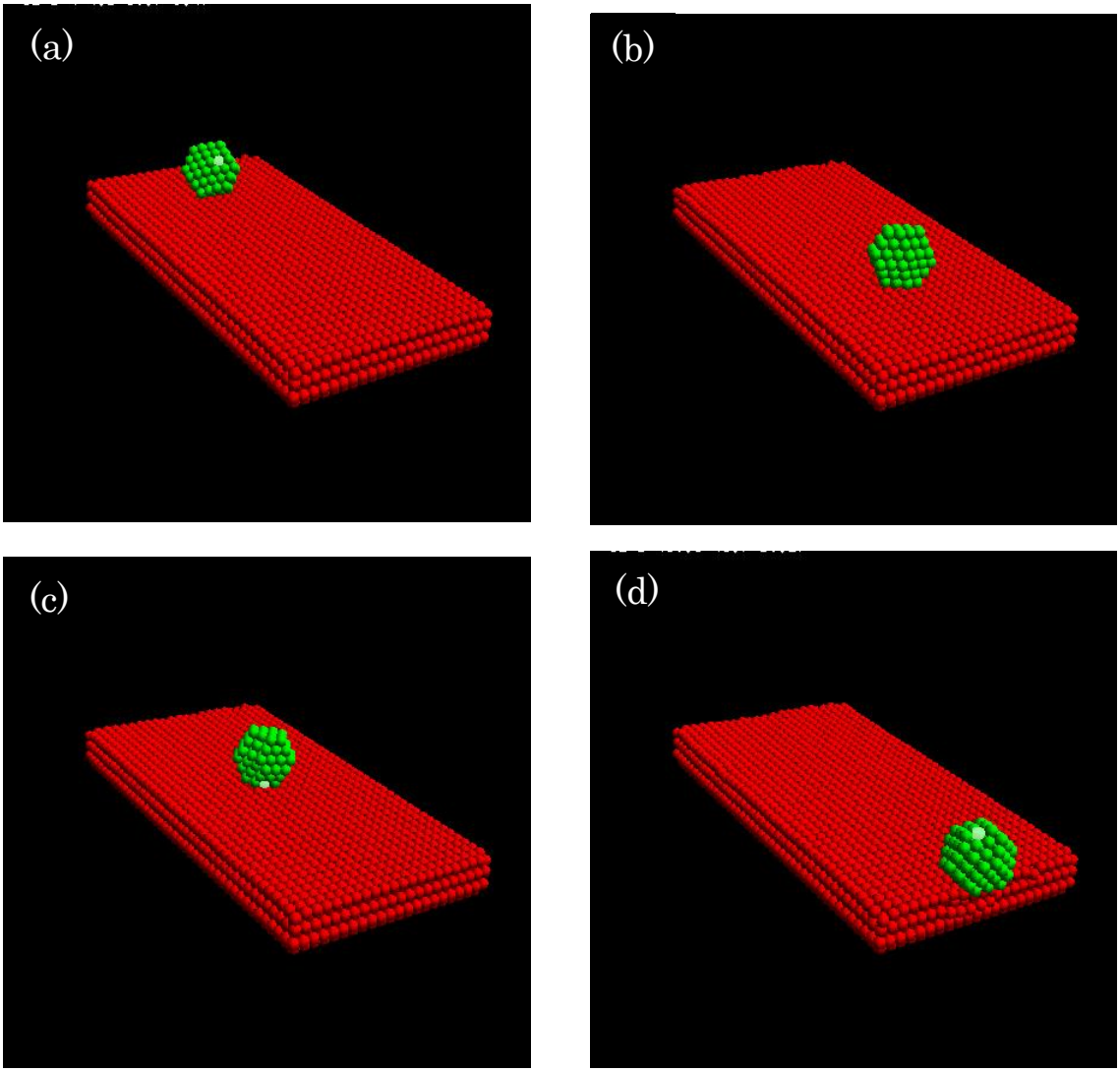


図 3.1-5　転がり状態の様子(a) $t = 0.11\text{s}$, (b) $t = 0.23\text{s}$, (c) $t = 0.34\text{s}$, (d) $t = 0.57\text{s}$

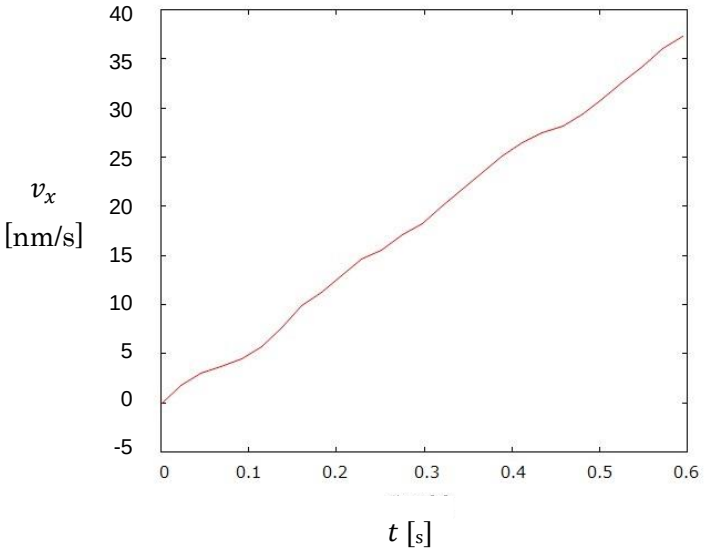


図 3.1-6 転がり状態における時間とv_xの関係

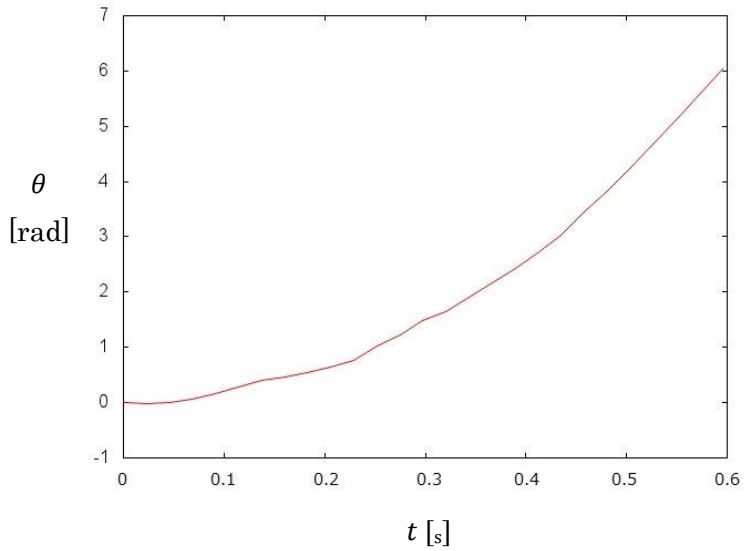


図 3.1-7 転がり状態における時間と転がり角の関係

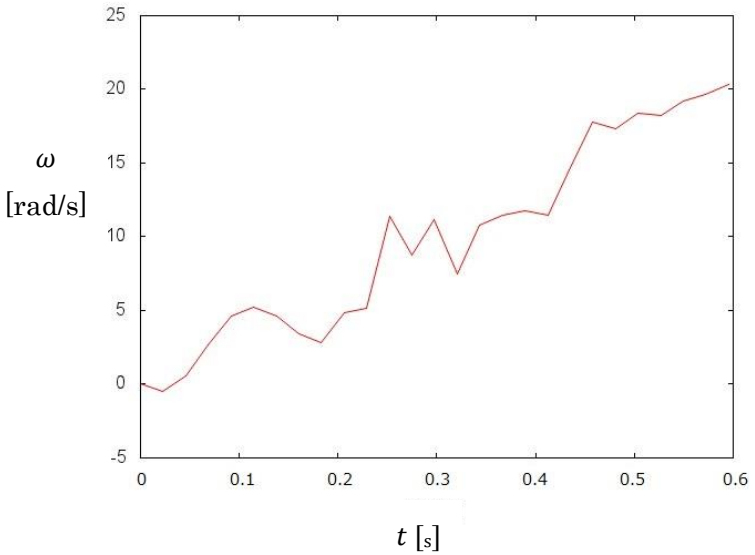
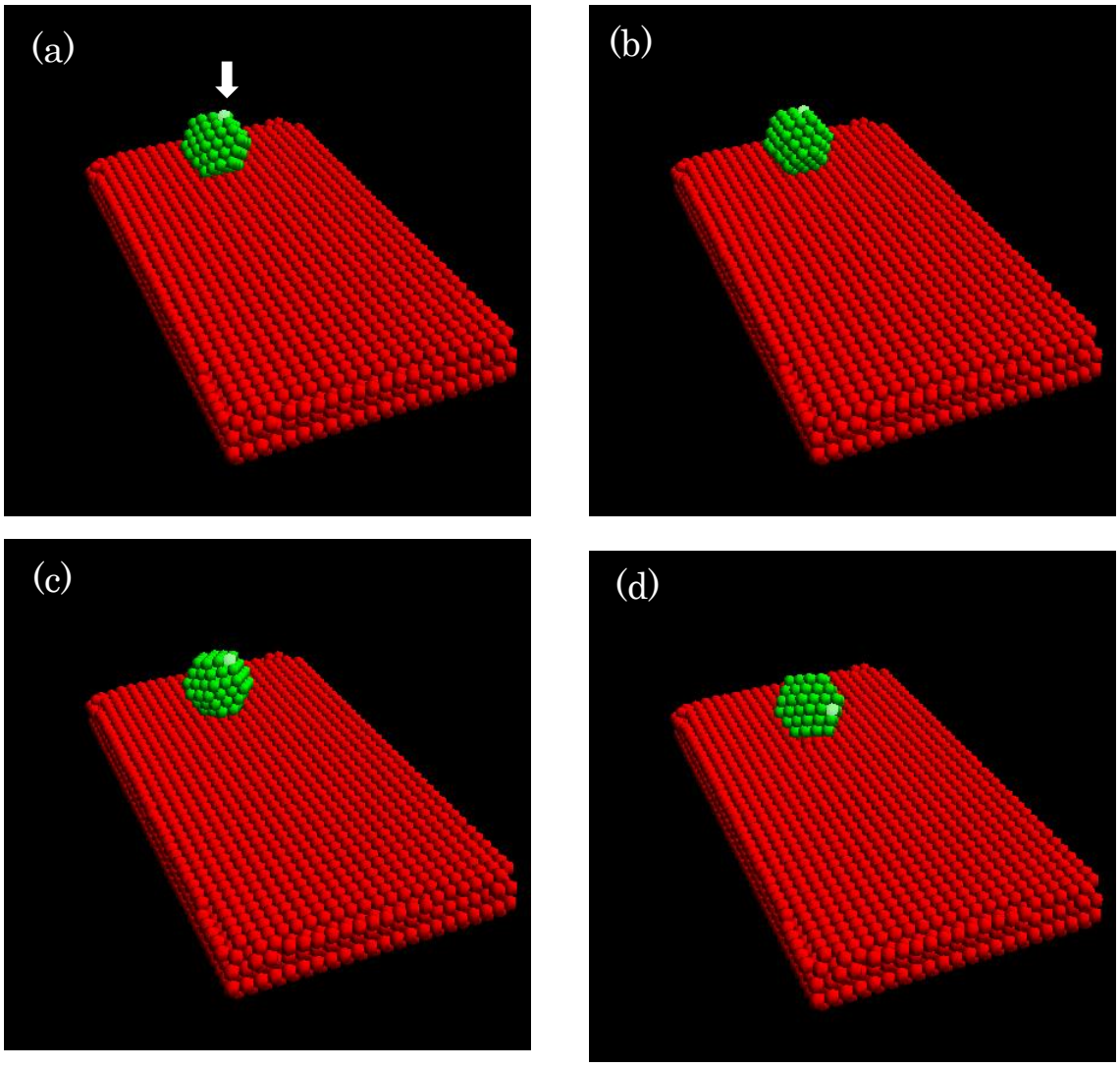


図 3.1-8 転がり状態における時間と転がり角速度の関係

3.1.3　転がり―滑り状態

球のふるまいのうちで、転がり状態から滑り状態へ移行するものなど、転がり状態と滑り状態が混合した状態があった（図 3.1-9、温度 0.77K, $F_x = 21.2\text{pN}, F_z = 4.24\text{pN}, c = 0.77$）。ここでは、この状態を「転がり―滑り状態」として結果にまとめた。図 3.1-10 に転がり―滑り状態におけるv_xと時間の関係である。この図より球の速度は単調に増加していることが分かる。

図 3.1-11 はこの同状態での転がり角 θ と時間の関係を表したもの、図 3.1-12 は転がり角速度ω と時間の関係を表したものである。転がり角 θ は最初増加するが 0.8 秒を越えたあたりから変化しなくなる。転がり角速度 ω も 0.8 秒までは徐々に増加するものの、それ以降はω＝0 の辺りで振動している。また、図 3.1-9 において、球面の一点（矢印で示した、白でマークした原子）の時間変化からも、転がりから滑りへ変化している様子が見て取れる。



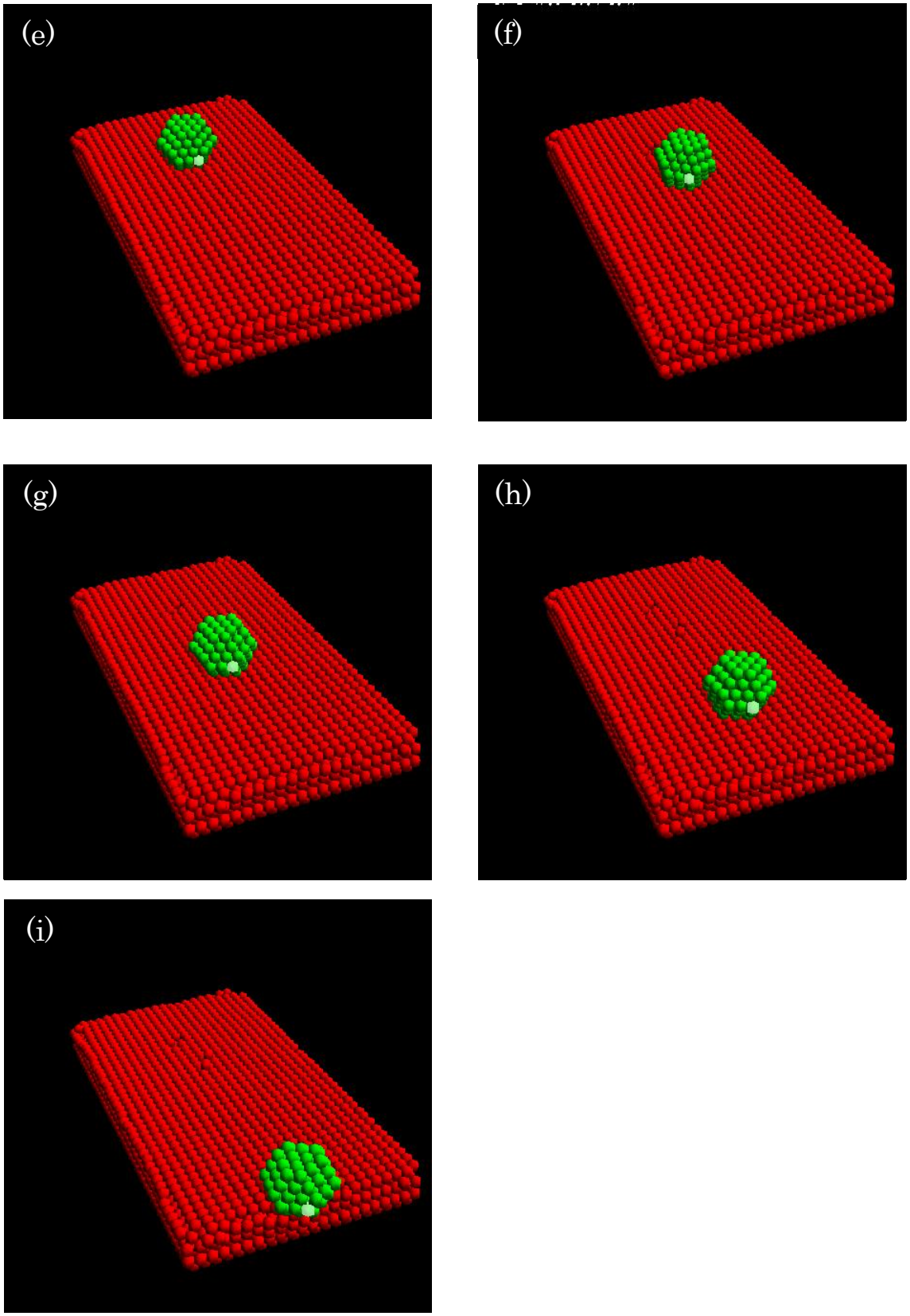


図 3.1-9　転がり―滑り状態の様子(a) $t = 0.00$s, (b) $t = 0.23$s, (c) $t = 0.46$s, (d) $t = 0.69$s, (e) $t = 0.69$s, (f) $t = 0.92$s, (g) $t = 1.37$s, (h) $t = 1.60$s, (i) $t = 1.83$s

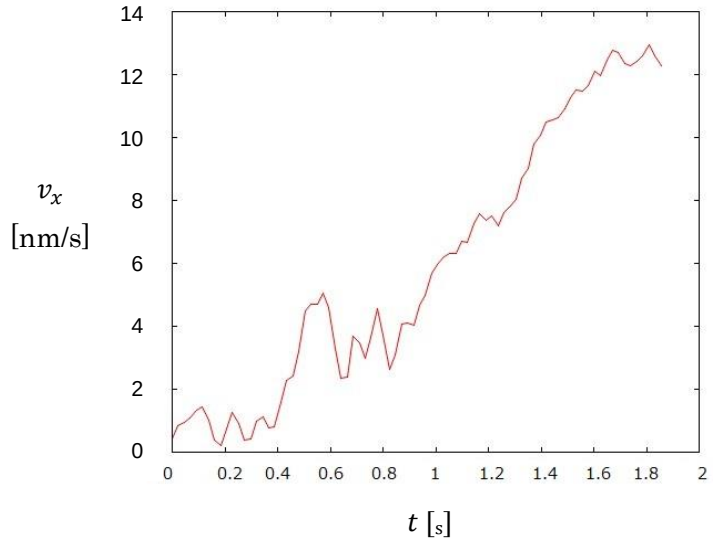


図 3.1-10 転がり―滑り状態における時間とv_xの関係

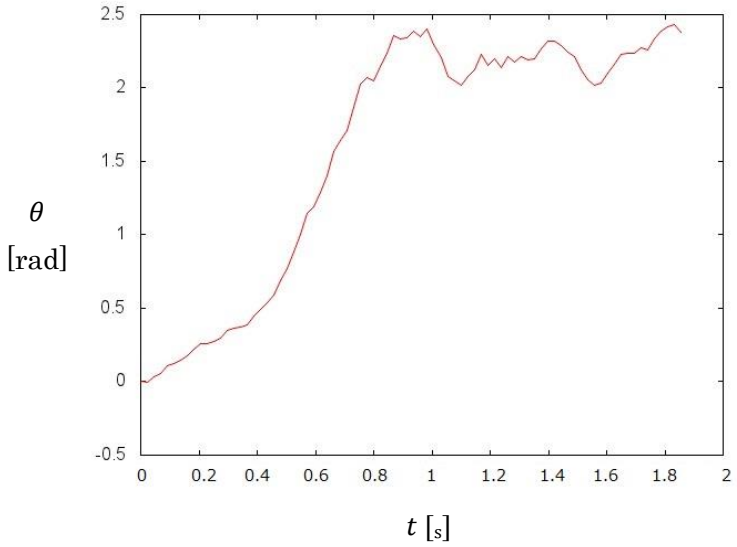


図 3.1-11 転がり―滑り状態における時間と転がり角の関係

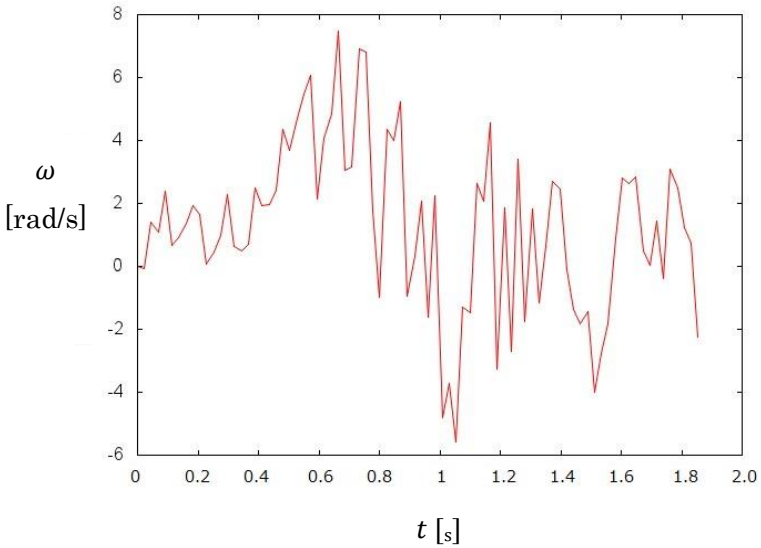


図 3.1-12 転がり―滑り状態における時間と転がり角速度の関係

3.1.4　剥離状態

「剥離状態」とは分子間に働く引力により基板から原子が剥離し、原子球の表面に付着する状態のことである。図 3.1-13 は剥離状態の系の典型的な例（温度 0.39K, $F_x = 42.4\text{pN}, F_z = 6.36\text{pN}, c = 0.89$ ）を示した。剥離は球に 1 個だけ付着する場合から、雪だるま状に多数の原子が付着する場合までバリエーションは様々だがここでは 1 原子でも付着すれば剥離状態であるとした。図 3.1-14 は剥離状態におけるv_xと時間の関係、図 3.1-15、3.1-16 は転がり角 θ と時間の関係、転がり角速度ωと時間の関係である。図 3.1-15 より、基板が剥離するまでは加速し剥離が始まるとほぼ等速で移動していることが分かる。転がり角が単調増加し、球は転がっていることが分かるが、剥離状況によって球の転がり具合は変動するので、転がり角速度は正の範囲で不規則に変動している。以上の結果より、剥離状態において球は基板を付着させながらほぼ等速で転がること、転がり角は増加するものの、基板の吸着状況によって不規則に増加していることが分かる。

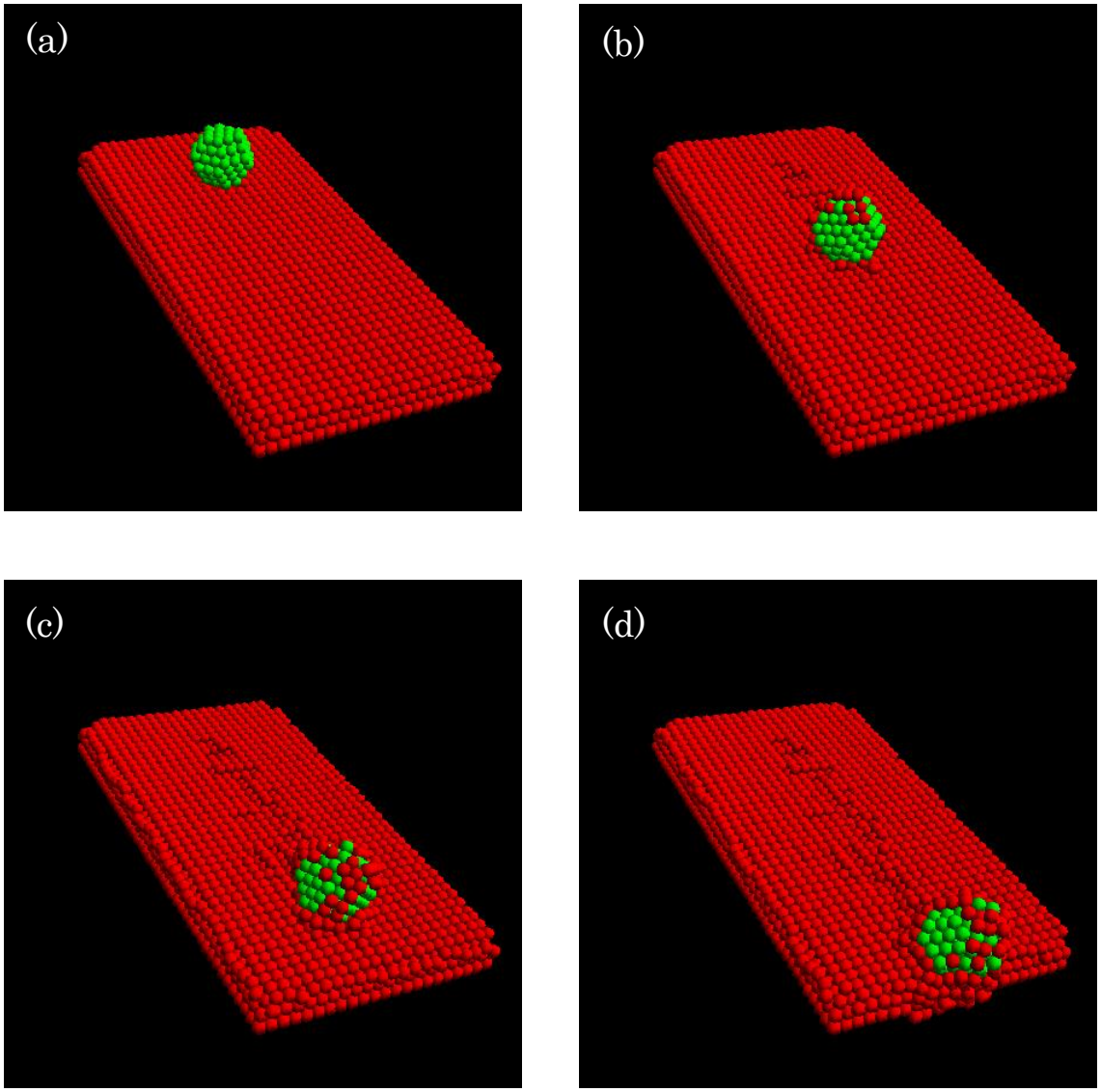


図 3.1-13　剥離状態の様子(a) $t = 0.18\text{s}$, (b) $t = 0.57\text{s}$, (c) $t = 0.92\text{s}$, (d) $t = 1.17\text{s}$

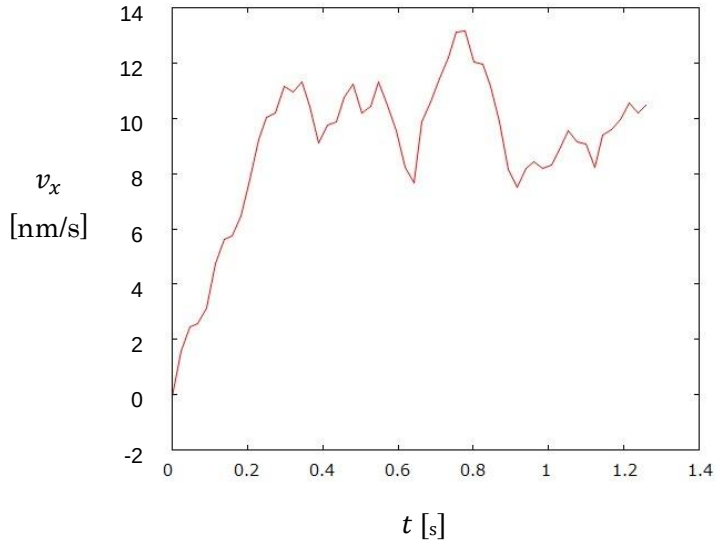
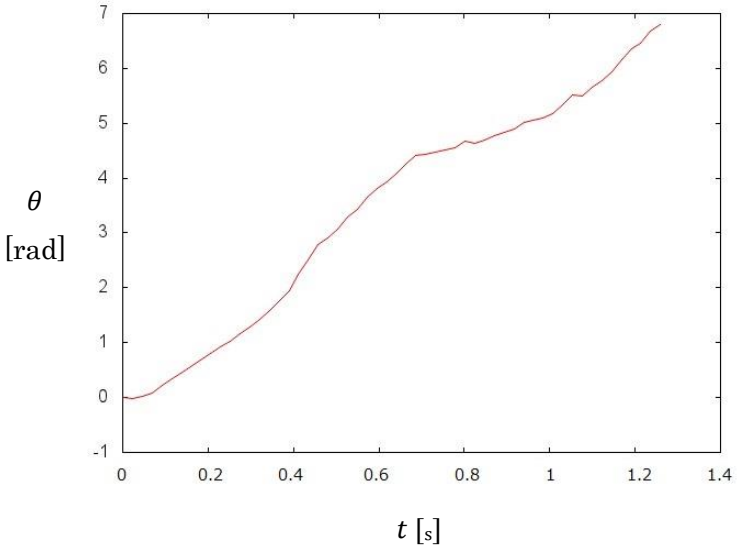


図 3.1-14 剥離状態の時間とv_xの関係



3.1-15 剥離状態における時間と転がり角の関係

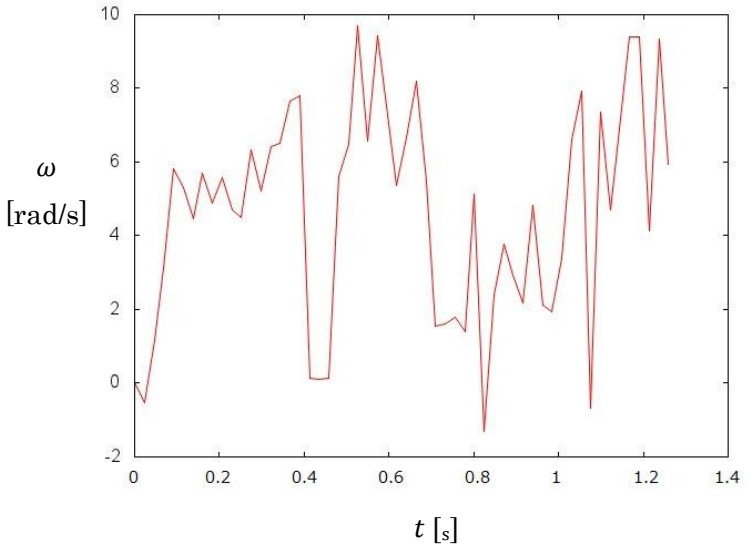


図 3.1-16 剥離状態における時間と転がり角速度の関係

3.1.5　吸着状態

「吸着状態」とは分子間力や重力の影響で原子球が基板に張り付き滑ったり転がったりしない状況のことである。図 3.1-17 は吸着状態の典型的な例（温度 0.77K, $F_x = 63.6\text{pN}, F_z = 4.24\text{pN}, c = 0.89$ ））を示した。図 3.1-18 に吸着状態におけるv_xと時間の関係、図 3.1-19、図 3.1-20 は転がり角 θ と時間の関係である。どの物理量もある一定値のまわりで振動しており、球は基盤に吸着し、並進運動はないと見られる。

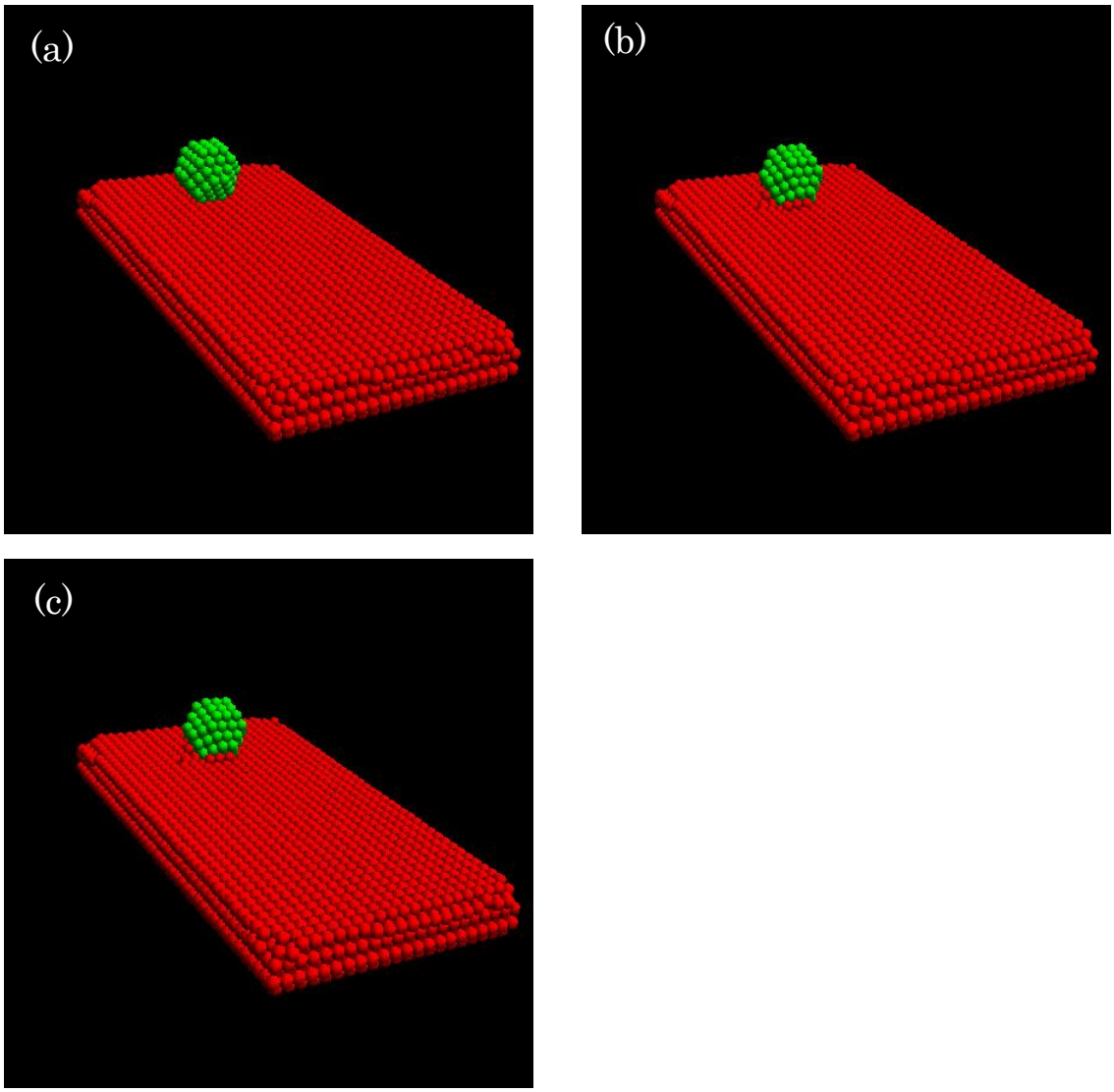


図 3.1-17　吸着状態の様子(a) $t = 0.00\text{s}$, (b) $t = 1.35\text{s}$, (c) $t = 2.26\text{s}$

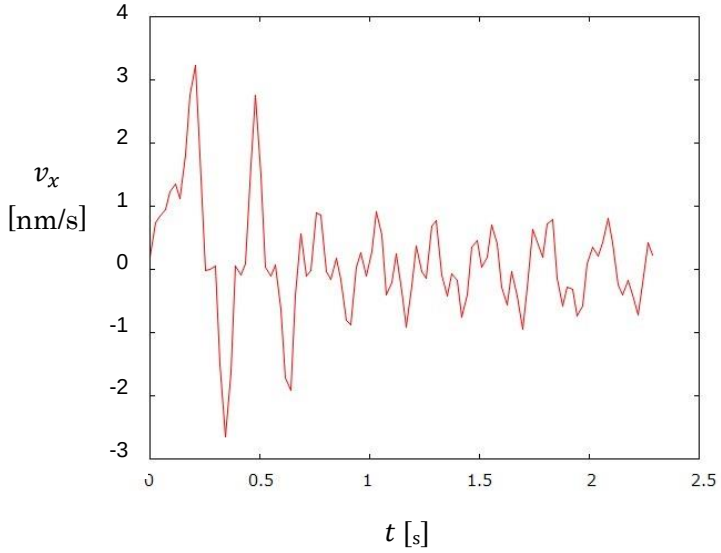


図 3.1-18 吸着状態における時間とv_xの関係

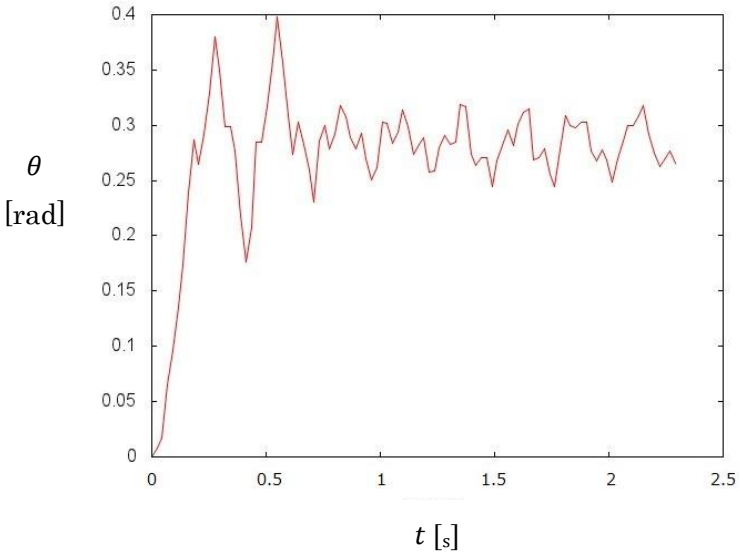


図 3.1-19 吸着状態における時間と転がり角の関係

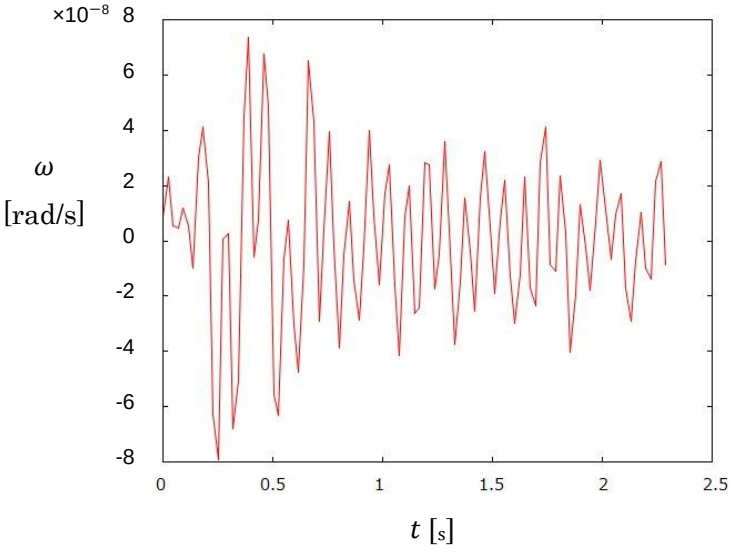


図 3.1-20 吸着状態における時間と転がり角速度の関係

3.1.6　破損状態

基板との相互作用により球の一部が基板に吸着するなど、球が運動する際に分裂した場合があった。図 3.1-21 はその状態を示した図である（温度 0.39K, $F_x = 55.1\mathrm{N}, F_z = 4.24\mathrm{pN}, c = 0.89$ ）。球の一部が基板に取り込まれている様子が映っている。このような場合、球のその後のふるまいに関わらず破損状態として記録した。

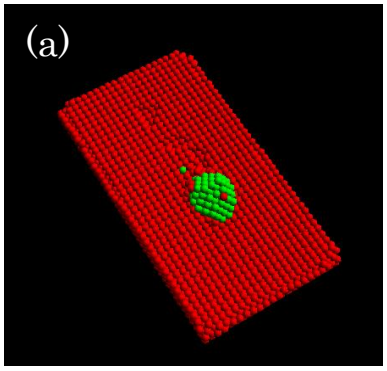


図 3.1-21　破損状態の球の様子 $t = 2.26\mathrm{s}$

3.1.7　跳躍状態

跳躍状態とは球が弾んだ拍子に基板から離れて、移動していく状態のことである。図 3.1-22 は「跳躍状態」の球のスナップ写真である（温度 0.77K, $F_x = 29.7\mathrm{pN}, F_z = 4.24\mathrm{pN}, c = 0.52$ ）。粒子数が少ないため、球には凹凸があり、転がる際に球の重心のz成分が急激に増加し球が跳ね上がる場合がある。吸着力や荷重が大きいと、球はすぐ基盤に引きつけられて転がり続けるが、吸着力や荷重が小さいと、跳ね上がった際に球は基板から引きはがされる。そのまま着地することなく基板上を飛んで行ったのがこの状態である。球は滑り摩擦力のモーメントにより、球は跳躍時には$\theta > 0$の方向に回転しながら x 方向に進んでいる。

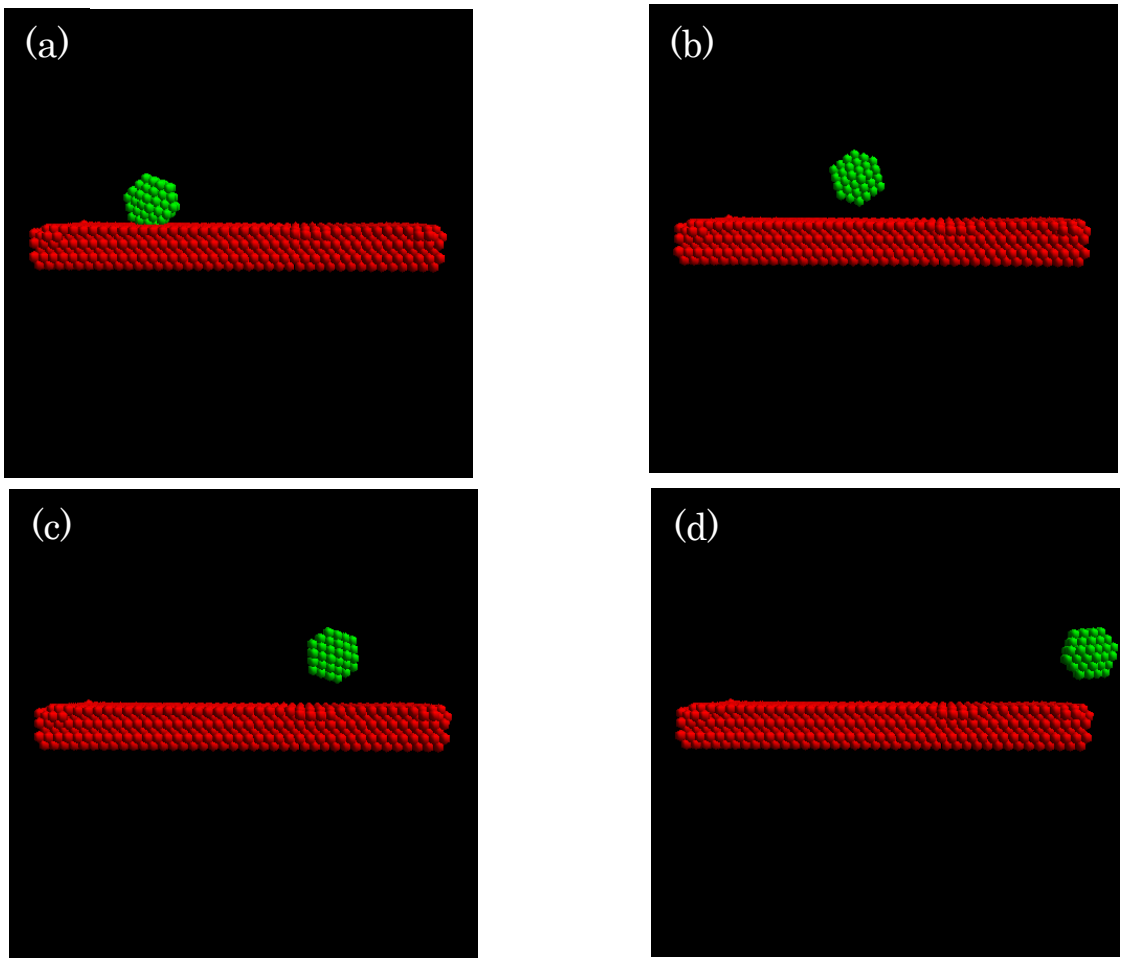


図 3.1-22 跳躍状態の球の様子(a) $t = 0.34\mathrm{s}$, (b) $t = 0.57\mathrm{s}$, (c) $t = 0.80\mathrm{s}$, (d) $t = 0.96\mathrm{s}$

3.2　パラメータを変化させた時の球の振る舞い

表 3.2-1、3.2-2 は、吸着パラメータc（横軸）及び牽引力F_x（縦軸）を変化させた時の球の振る舞いを色で分類して表したものである。黄色、赤色、水色、紫色、緑色、黒色、青色はそれぞれ「滑り状態」、「転がり状態」、「滑りと転がりの混合状態」、「吸着状態」、「剥離状態」、「破損状態」、「跳躍状態」を表す。表の横軸はc、縦軸はF_xを表している。

3.2.1　吸着パラメータの影響について

表 3.2-1 では、吸着パラメータを 0.5 から 0.89 まで変化させた。吸着パラメータが低い場合、基板と球の間に働く引力は小さくなり「滑り状態」が多くみられる。吸着パラメータを増加すると、球は次第に転がるようになり、「転がり状態」または、「転がりと滑りの混合状態」が多くみられる。また、この段階でのみ「跳躍状態」も現れる。
吸着パラメータを増加していくと次第に「転がり状態」が増えていき、「滑り状態」はほとんど見られなくなる。さらに増加していくと球は基板原子を付着させながら転がりだす「剥離状態」になる。最初は 1、2 分子が球につく程度だが、吸着パラメータが増加するにつれて付着する分子は増加していく（図 3-2.1，3-2.2，3-2.3，3-2.4）。さらに吸着パラメータを増加させると球は基板に張り付き動かなくなる。さらにパラメータを上げていくと球の破損が見られる。「滑り状態」はパラメータが大きくなると頻度は減るものの、なくなることはなかった。

3.2.2　牽引力の影響について

牽引力は 21.2pN から 63.6 pN まで変化させた。牽引力が小さい場合、1.4.1 式より球に働く滑り摩擦力が小さくなり、それに伴い転がるためのモーメントが小さくなる。表 3.2-1 より吸着パラメータが小さい段階では、牽引力が小さいほど「滑り状態」になりやすく、牽引力が大きいほど「転がり状態」になりやすい。吸着パラメータが大きい段階では、牽引力が小さいほど球は基板に吸着し、牽引力が小さいほど球は「剥離状態」になることが分かる。また、「破損状態」は牽引力が大きい場合でのみ起こることが分かる。

3.2.3　温度の影響について

表 3.2-2 は、ネオン原子とアルゴン原子が一部でも融解してしまわないように、アルゴンの融点 83K より十分に低い 0.385K から、24.6 K までを計測した。温度が低いと基板や球の形は規則的で、分子の熱運動はアニメーションで観察できないほどわずかであったが、温度が高くなるにつれ、アニメーションで分かるほど分子運動が盛んになり、基板の角が丸くなり、全体的に歪な形になっていった。

吸着パラメータが小さいところでは、温度が変化しても「滑り状態」や「滑りと転がりの混合状態」、「転がり状態」に大きな変化が見られなかった。

一方吸着パラメータが大きいところでは、球が剥離する領域は温度が上がるにつれ、左にシフトしているのが分かる。「転がり状態」と「滑り状態」の境界が温度によって変化しないことから、転がり領域が結果的に狭くなっていくのが分かる。

「剥離状態」の領域と「吸着状態」の領域の境界もあまり大きな変化が見られないことが分かる。さらに、荷重が大きくなると吸着パラメータが大きい範囲での「滑り状態」が増えていることが分かる。以上よりマクロな系では摩擦には依存しないはずの温度が、ミクロな系だと様々な面で運動状態に影響を与えていることが分かる。

3.2.4　荷重の変化について

表 3.2-2 には、z方向の荷重F_zを変化させたときの吸着パラメータと牽引力F_xの関係を示した。荷重は、基板原子の相互作用力より荷重が大きくなり、基板原子が崩れ落ちてしまわないように、4.24 pN から 33.9 pN までを計測した。

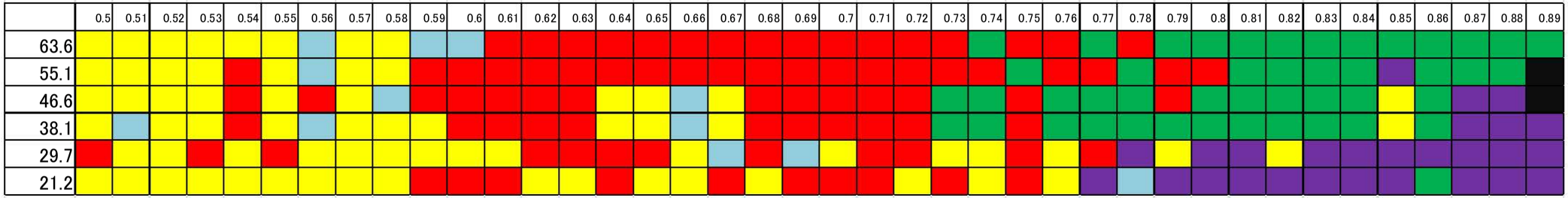
吸着パラメータが小さいところでは、荷重が大きいほど「滑り状態」よりも「滑りー転がり状態」や「転がり状態」が多くなり、「滑り状態」と「転がり状態」の境界は左にシフトしている。この結果、牽引力が小さい 4.24 pN や 8.48 pN においても滑り状態が多くみられる。

吸着パラメータが大きいところでは球が吸着する領域が左にシフトしているのが分かる。しかし「転がり状態」の領域と「剥離状態」の領域の境界は左側へシフトせず、結果的に荷重が大きいほど「剥離状態」の領域が狭くなる。さらに、荷重が大きくなると「滑り状態」が減り球の破損がより起こりやすくなることが分かる。

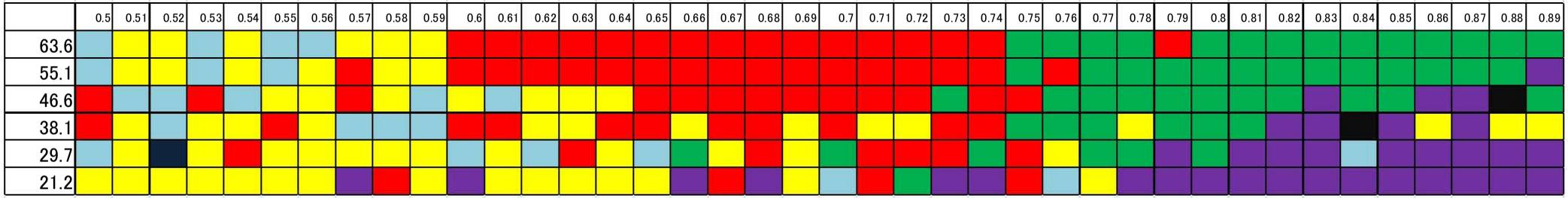
表 3.2-1　牽引力F_xと吸着パラメータ cの関係。横軸が c、縦軸がF_x[pN]を表している。

T=0.385K, F_z=4.24pN

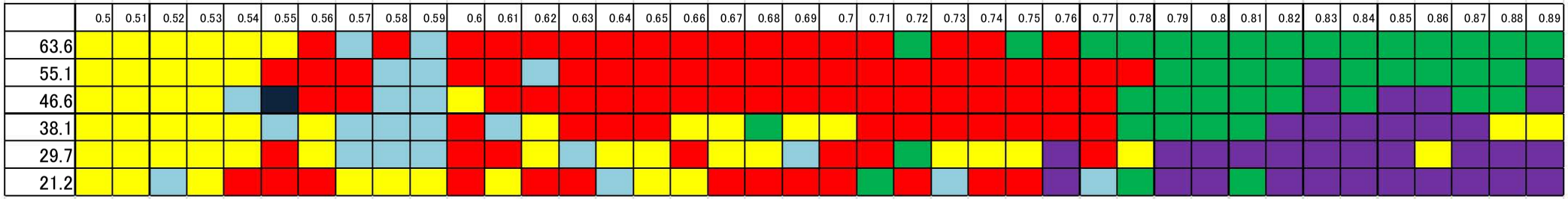
温度は上から T=0.385K、0.770K,01.54K,3.08K 荷重はすべてF_z =4.24 pN



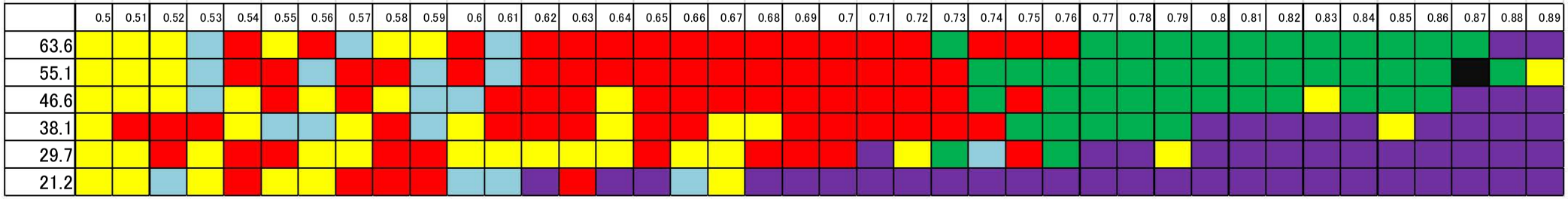
T=0.770K, F_z=4.24pN



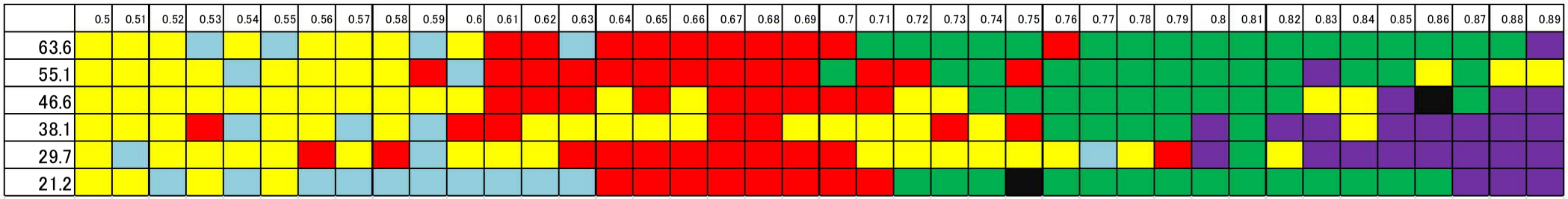
T=1.54K, F_z=4.24p N



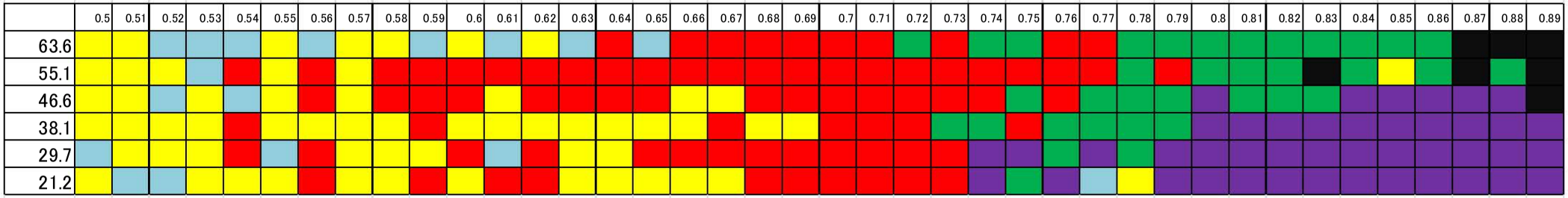
T=3.08K, F_z=4.24pN



T=6.16K, gz=4.24pN



T=12.3K, gz=4.24 pN



T=24.6K, gz=4.24 pN

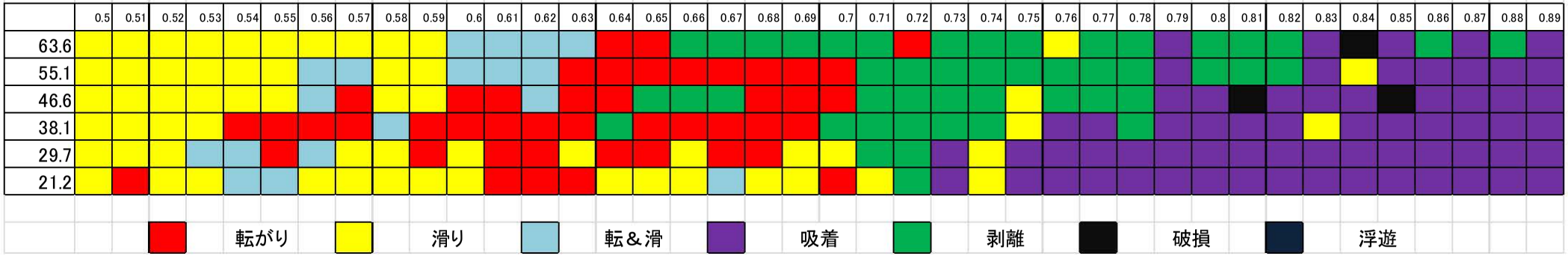
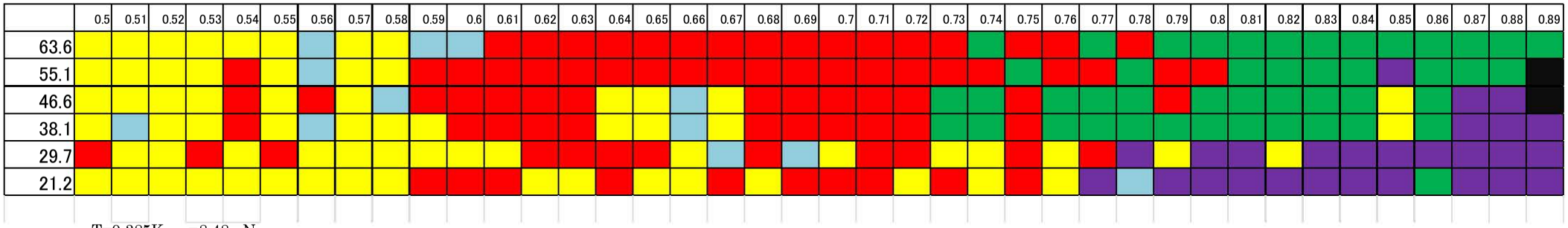
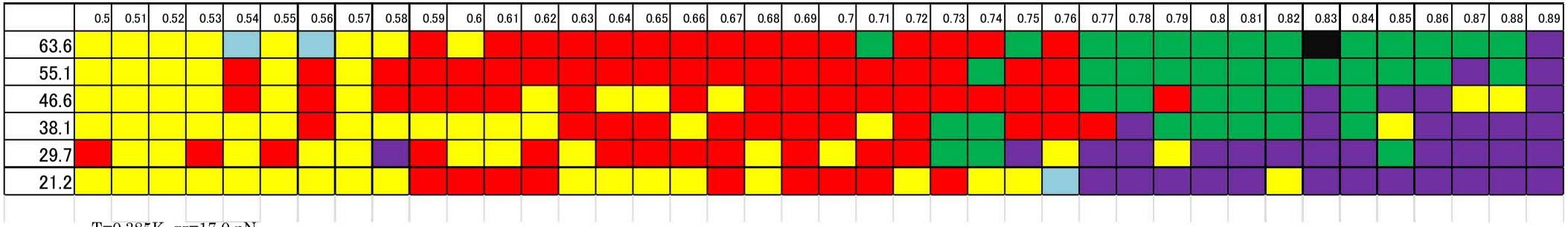


表 3.2-2　牽引力F_xと吸着パラメータ c の関係。横軸が c、縦軸がF_x[pN]を表している。

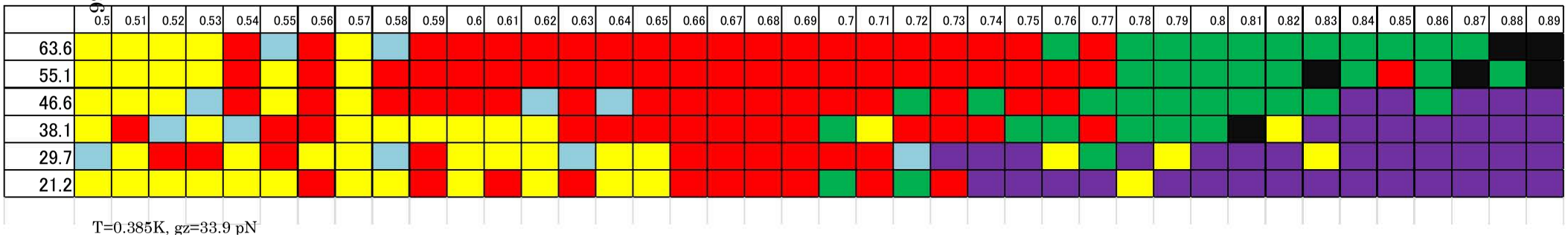
T=0.385K, gz=4.24pN　　温度はすべて T=0.385K、荷重は Fz=4.24 pN, 8.48 pN, 17.0 pN, 33.9 pN



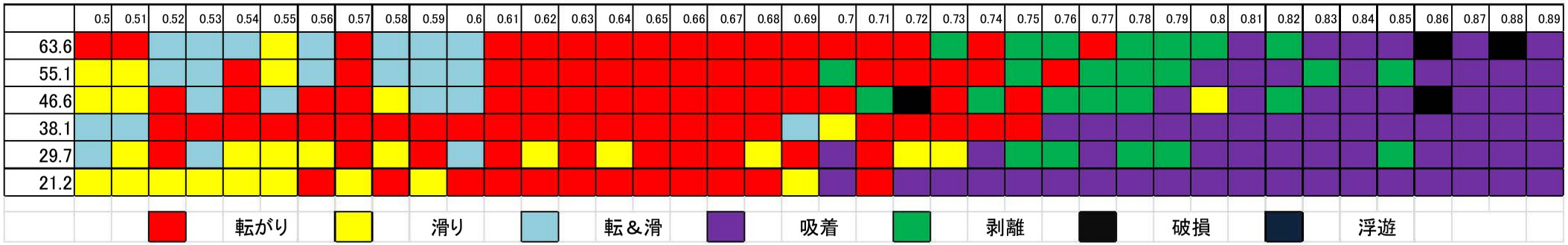
T=0.385K, gz=8.48 pN



T=0.385K, gz=17.0 pN



T=0.385K, gz=33.9 pN



第 4 章　考察

この章では前章で得られたシミュレーション結果について考察する。

4.1　転がり角速度について

表 3.2-2 では温度または荷重を固定したときの運動状態を調べた。通常、物体を強い牽引力で引っ張ると、重心加速度が大きくなるのは当然のことである。だが本研究で調べているような小さな系においては、微視的な相互作用の変化が全体の運動に影響するため、原子間の吸着だけでなく温度も運動状態に大きな影響を与えるかもしれない。また、z方向の荷重を大きくすると球や基板の変形も大きくなり、やはり運動状態は異なると考えられる。そこで、表 3.2-2 に示したデータのうち、吸着パラメータを c=0.64 に固定し、一定の牽引力$F_x = 21.2\mathrm{pN}$ で引っ張った時に、球の転がり角速度の温度や荷重に対する依存性を調べる。ただし、以下に述べる結果は 1 サンプルの結果であることを付記しておく。

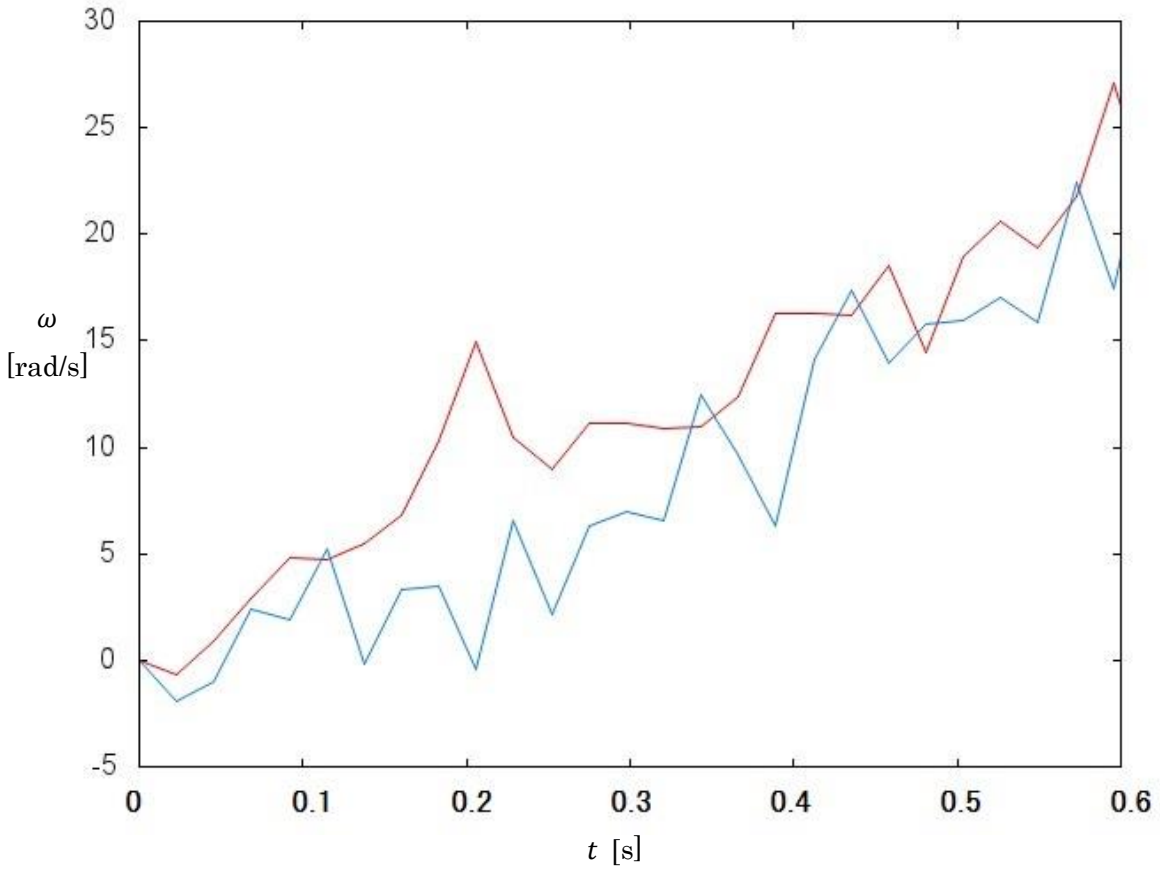


図 4.1-1 異なる温度に対する転がり角速度。赤は T=0.385K、青は T=24.6K

まず、系の温度に対する依存性を調べるため、T＝0.385K と T=24.6K の二つの場合について調べる。ここでは荷重をF_z=4.24 pN に固定し、転がり角速度の時間変化を調べた。

図 4.1.-1 は、横軸に牽引時間、縦軸に転がり角速度をとったものである。これを見ると同じ$t = 0.1$ s から$t = 0.3$ s の間で多少の違いは見られるものの、系の温度に関係なく一定の角速度で転がっているようである。

温度が高くなると原子の運動が活発になるため、球の表面原子が基板表面にめり込む。球がめり込むことにより転がり摩擦係数は大きくなると考えられる[4]。その結果、球の転がり角速度は温度が高くなるほど小さな値をとると考えられる。しかし、図 4.4-1 の場合は、転がり角速度に大きな変化はない。これは温度が 2 桁違うにもかかわらず、荷重が小さいため球が基板にめりこむことがなく、同程度の転がりが生じたのであろうと考えられる。

次に荷重に対する依存性を調べるため、F_z=4.24 pN とF_z=33.9 pN の二つの場合について調べる。ここでは温度を T＝0.385K に固定し、転がり角速度の時間変化を調べた。

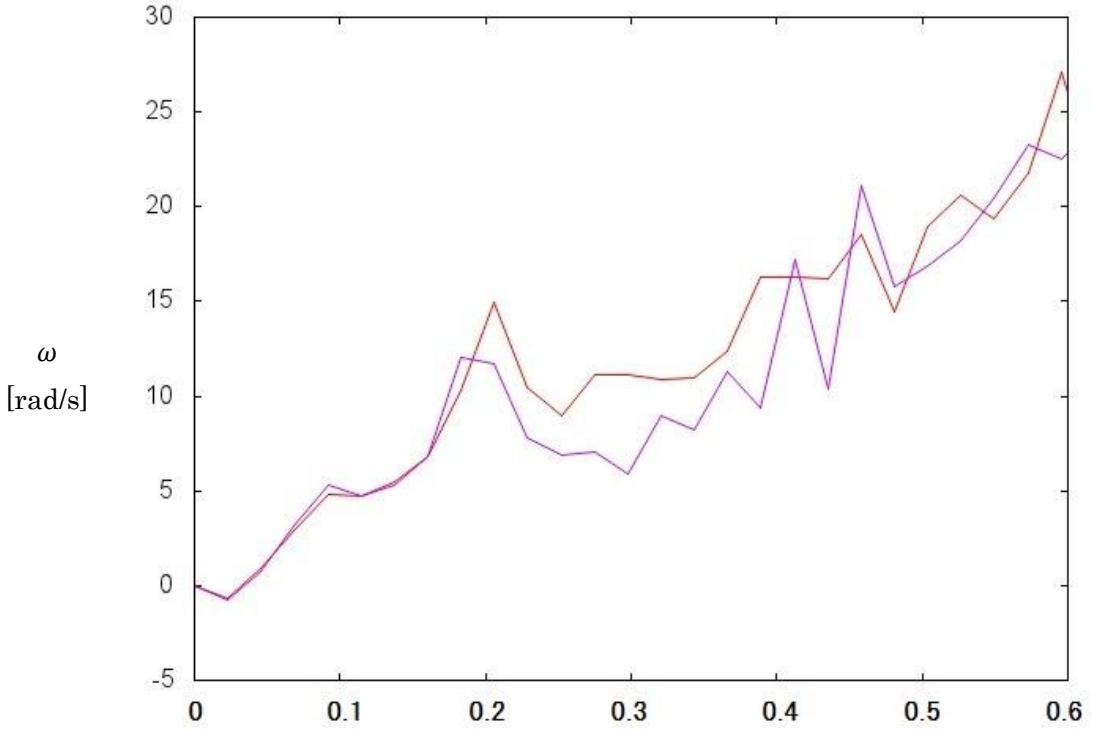


図 4.1-2　異なる荷重に対する転がり角速度。赤はF_z=4.24 pN、紫はF_z=33.9 pN

このグラフから荷重を一桁変化させても転がり角速度にはほとんど影響がないことがわかる。荷重をさらに大きくすると系が壊れてしまうため、そもそも回転運動が生じなくなる。したがってこの系においては、荷重の大きさは転がり角速度に影響しないことがわかる。もし別の原子を用いて同様の系を構成し、その系が十分な強度を持つようになれば、基板の変形が球の回転運動に影響するようになるだろう。

4.2　転がり摩擦係数について

ここでは式(1.4.10)に基づいて、球と基板の間の転がり摩擦係数を計算してみる。データは前節の図 4.1-1 の青線で示したデータを用いて、転がり摩擦係数の時間変化を調べた。

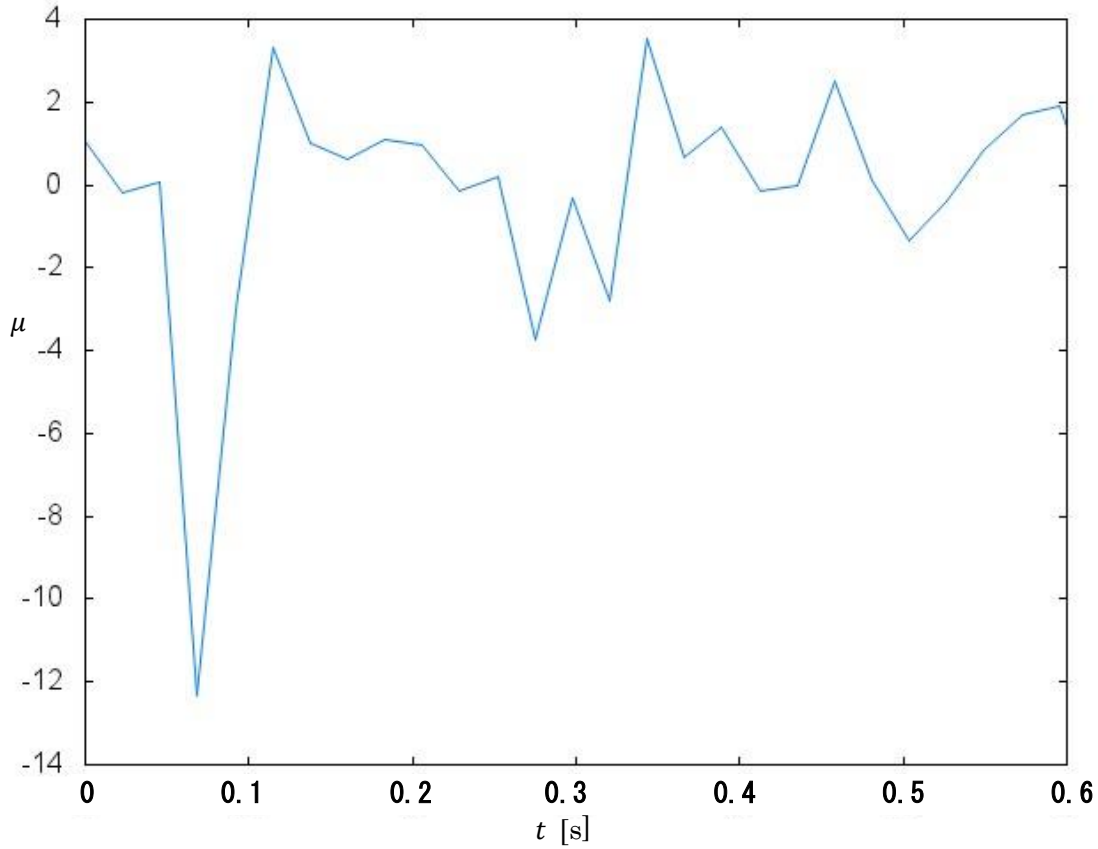


図 4.2-1　転がり摩擦係数の時間変化

図 4.2-1 に転がり摩擦係数の時間変化を示した。これを見るとわかるように、$\mu_{\mathrm{roll}} = 0$の間で振動しながら変化していることがわかる。前節で用いた他のデータも用いて摩擦係数の時間変化を調べたが、やはり同様の傾向を示した。

このグラフを見る$\mu_{\mathrm{roll}} < 0$ととなっている部分があるため、さらに球の重心速度のz成分、v_zの時間変化も調べた。

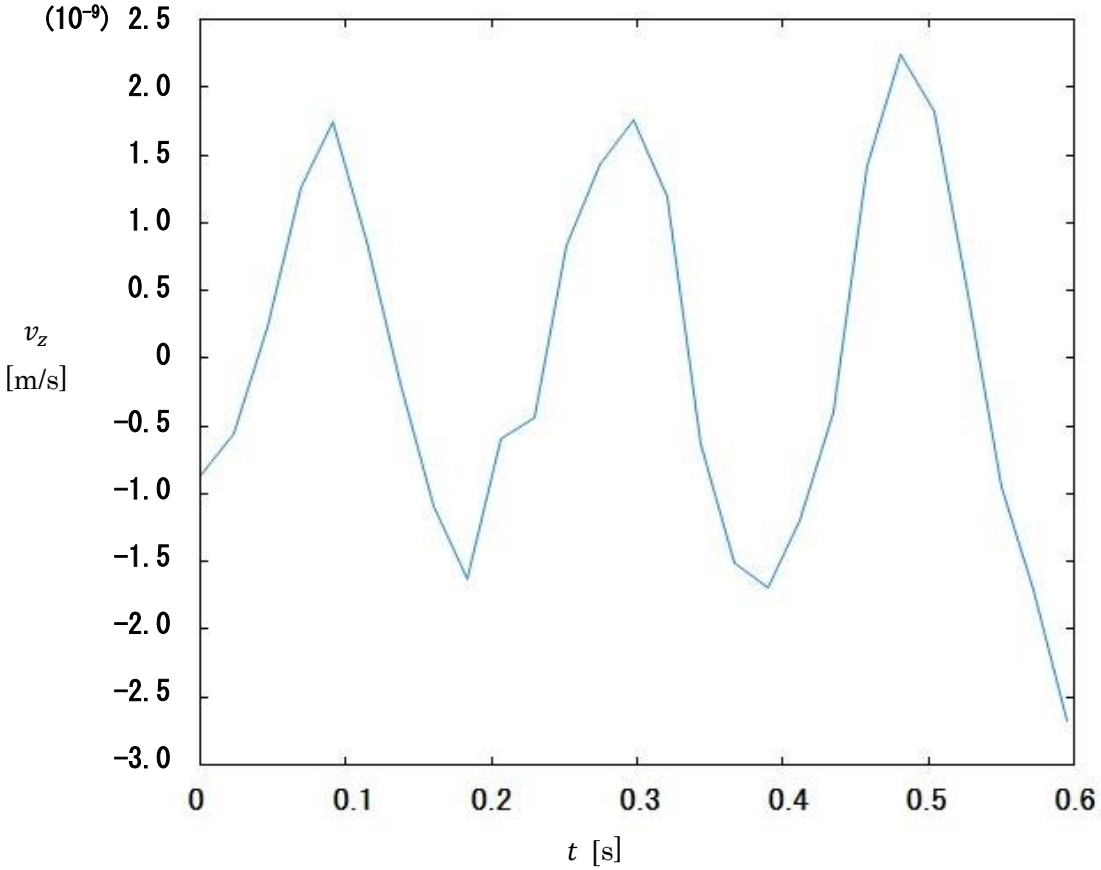


図 4.2-2. v_zの時間変化

図 4.2-2 にはv_zの時間変化を調べた。これを見ると球が転がるときには、球の重心が上下方向に振動しながら運動していることがわかる。さらに、$\mu_{\mathrm{roll}} < 0$となるときには、球の$v_z > 0$となっていることがわかる。つまり球を一定の力で牽引しているときに、周期的に球全体が上下するため、球が上に運動した時には転がり抵抗が負の値を取るため、転がり摩擦係数も負の値を取るのだと考えられる。

本研究で用いた系は少数粒子から成る系のため、転がり摩擦係数に関しては正の安定した値を得ることができなかったと思われる。原子数を増やしてさらに大きな系で同様のシミュレーションを実行すれば、おそらく安定した転がり摩擦係数が得られると考えられる。

4.3　転がり状態から剥離状態への遷移に関して

表 3.2-1 および表 3.2-2 において、一定の牽引力で球を引っ張ったときに、吸着パラメータを大きくしていけば転がり状態から剥離状態に遷移する様子が見られた。例えば、表 3.2-1 の T =0.385K の場合においては、c =0.73～0.75 付近で転がり状態から剥離状態に遷移する様子が見られる。ここではこの状態の変化を起こす c の値を決める要因について考察する。

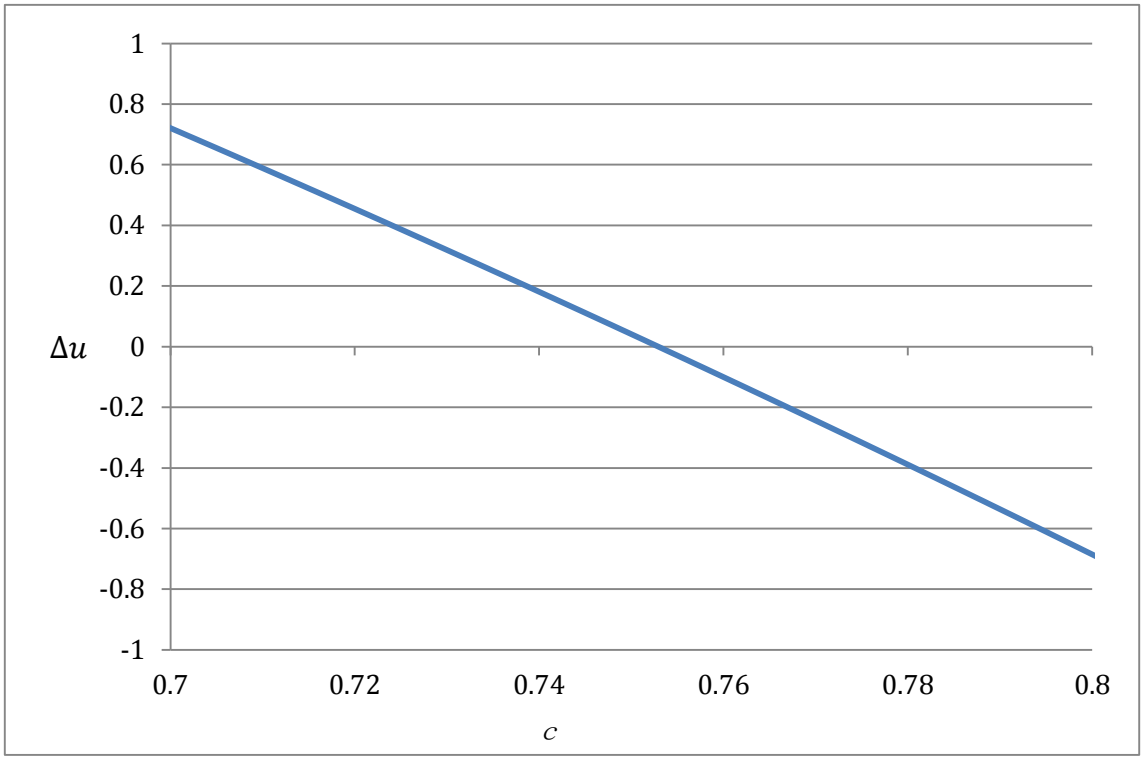


図 4.3-1　吸着パラメータとポテンシャルの差の関係

図 4.3-1 は吸着パラメータが 1 であるネオンどうしの Lennard-Jones ポテンシャルの最低値$u_{\text{Ne-Ne}}$から、吸着パラメータを c としたネオン-アルゴン間の Lennard-Jones ポテンシャルの最低値$u_{\text{Ne-Ar}}(c)$を引いた差

$$\Delta u(c) = u_{\text{Ne-Ne}\,min} - u_{\text{Ne-Ar}\,min}(c)$$

を縦軸、吸着パラメータを横軸にとったグラフである。この図より吸着パラメータが　c= 0.753 より小さい場合、ネオン-アルゴン間のポテンシャルの最低値よりもネオンどうしのポテンシャルの最低値の方が高く、c= 0.75 より大きい場合、ネオンどうしのポテンシャルの最低値よりもネオン-アルゴン間のポテンシャルの最低値の方が高いことが分かる。これより吸着パラメータが c = 0.75 より小さいときはネオンどうしの結合がネオン-アルゴン間の結合より安定であり、c = 0.75 より大きい場合はネオン-アルゴン間の結合の方が安定であると言える。このことによって c=0.75 付近で転がり状態から剥離状態への遷移が起こったのだと考えられる。

第5章まとめ

直径が 1.74nm のアルゴン球を、ネオン基板の上で保存力を加えて転がすプログラムを作成し、実験を行なった。その際に吸着パラメータ、牽引力、温度、荷重を変化させて運動常態に変化が起きるのかを調べた。また変化させたときの物理量を測定した。

転がりの運動常態は滑り状態のほか、転がり状態、転がり―滑り状態、剥離状態、吸着状態、破損状態、跳躍状態の 7 種類が見つかった。吸着パラメータが小さいとき、球は滑り状態が多く、吸着パラメータを増加させていくにつれて転がり状態、剥離状態、そして吸着状態へと移行していく。牽引力を増加させていくと球は転がり状態や剥離状態になりやすい。また、温度が高くなると吸着パラメータの増加による転がり状態から剥離状態への移行が早い段階で起き、荷重が大きくなると吸着パラメータの増加による滑り状態から転がり状態への移行および、剥離状態から吸着状態への移行が早い段階で起きることがわかった。さらに、パラメータの変化によって転がり角速度はどう影響されるのかや、転がり摩擦係数についての考察も行なった。結果は、本研究の温度や荷重の範囲では、転がり角速度に違いは見られないことがわかり、転がり摩擦定数は小さな系では、$\mu_{\mathrm{roll}} = 0$のあたりを振動することがわかった。最後に吸着パラメータについて考察を行なった。吸着パラメータ c が 0.70 辺りではネオンどうしの分子間ポテンシャルの方がネオン-アルゴン間のポテンシャルよりも安定であり、球はネオン原子を吸着することなく転がることができる。c が 0.80 辺りではネオン-アルゴン間のポテンシャルの方がネオンどうしの分子間ポテンシャルよりも安定であり、球が転がる際にネオン原子は吸着してしまう。よって剥離状態になるということがわかった。

今後の課題として、相図（表 3.2-1、3.2-2）が 1 サンプルだったので複数のサンプルを用いた相図を作成することや、十分な強度があり、融点が高い物体でシミュレーションを行うこと、さらに今回は球の半径を統一して行ったので、数を増やし、球のサイズよる運動の状態の変化を調べることがあげられる。

謝辞

本研究の実験や解析、論文作成などにおいてご指導、ご助言をいただいた、三重大学 教育学部 学校教育教員養成課程 理科教育コース 物理学専攻　國仲寛人准教授に心より感謝いたします。

また、様々なご指導、ご助言を頂いた三重大学教育学部理科教育コース物理学専攻　牧原義一教授　及び、共に学び合える豊かな研究環境を作っていただいた学生の方々に心から感謝いたします。

参考文献

[1] 辰巳　創一、紛体上の転がり摩擦に関する研究、東京大学修士論文（2005）

[2] 松川　宏、　物理の世界２　摩擦の物理、岩波書店　(2012)

[3] 河野　彰夫、摩擦の科学、裳華房（1989）

[4] W.G.Lee et al, Molecular Dynamics of Rolling Friction Using Nanosize Spheres, Tribol. Lett. 33, 37 (2009)

[5] 小宮山　宏 et al、ナノテクノロジーで未来を拓く、NTS　(2009)

[6] 町田　勝之輔、分子力学法、講談社サイエンティフィック（1994）

[7] 川添　良幸、ナノシミュレーション技術ハンドブック、共立出版株式会社（2006）

[8] 川村　雄行、パソコン分子シミュレーション―分子動力学実験入門、海文堂出版株式会社（1990）

[9] 神山新一　佐藤明、分子動力学シミュレーション、朝倉書店（2007）

[10] 岡部恒康、Symplectic Integrator による定圧定温の分子動力学法　(2000)

[11] Seung-chai Jung et al1, Molecular dynamics simulation on the energy exchanges and adhesion probability of a nano-sized particle colliding with a weakly attractive static surface, Jour. Aero. Sci. 41, 745 (2010)

[12] A. Awasthi et al, Reentrant Adhesion Behavior in Nanocluster Deposition, Phys. Rev. Lett 97, 186103 (2006)

[13] A. Awasthi et al, Molecular dynamics simulation of reflection and adhesion behavior in Lennard-Jones cluster deposition, Phys. Rev. B 76, 115437 (2007)

[14] Michael Rich, *Nano-Engeneering in Science and Technology*, World scientific, 19 (2003)

付録 A.　Symplectic 積分

このシミュレーションでは陽的二次の Symplectic 積分を用いた。Symplectic 積分は速度ベルレ法とも呼ばれる分子動力学のアルゴリズムの一種で、一定の時間刻みで全粒子を移動させるステップを繰り返し行うものである。分子動力学アルゴリズムのなかでもエネルギー誤差の少なく、位置と速度を同じ時間ステップで評価できることが特徴である。

時刻 t における位置$\boldsymbol{r}(t)$ と速度$\boldsymbol{v}(t)$、力$\boldsymbol{f}(t)$について、時間刻みを Δt とすれば時刻 $t+\Delta t$ における位置 $\boldsymbol{r}(t+\Delta t)$と速度 $\boldsymbol{v}(t+\Delta t)$、力$\boldsymbol{f}(t+\Delta t)$は Taylor 展開することによって、

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \Delta t \frac{d\boldsymbol{r}(t)}{dt} + \frac{\Delta t^2}{2!}\frac{d^2\boldsymbol{r}(t)}{dt^2} + O(\Delta t^3) \qquad (A.1)$$

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}(t) + \Delta t \frac{d\boldsymbol{v}(t)}{dt} + \frac{\Delta t^2}{2!}\frac{d^2\boldsymbol{v}(t)}{dt^2} + O(\Delta t^3) \qquad (A.2)$$

$$\boldsymbol{f}(t+\Delta t) = \boldsymbol{f}(t) + \Delta t \frac{d\boldsymbol{f}(t)}{dt} + O(\Delta t^2) \qquad (A.3)$$

と表すことができる $O(\Delta t^2)$ はΔt の２乗以上の項をまとめたものである。

(A.3)式の$O(\Delta t^2)$ を省略し、変形すると、

$$\frac{d\boldsymbol{f}(t)}{dt} \cong \frac{\boldsymbol{f}(t+\Delta t) - \boldsymbol{f}(t)}{\Delta t} \qquad (A.4)$$

さらに

$$\boldsymbol{v}(t) = \frac{d\boldsymbol{r}(t)}{dt}, \qquad (A.5)$$

運動方程式

$$\frac{d^2\boldsymbol{r}(t)}{dt^3} = \frac{\boldsymbol{f}(t)}{m} \qquad (A.6)$$

より(1)式は$O(\Delta t^3)$の項を無視して

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \Delta t\, \boldsymbol{v}(t) + \frac{\Delta t^2}{2}\frac{\boldsymbol{f}(t)}{m} \qquad (A.7)$$

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}(t) + \frac{\Delta t}{2}(\boldsymbol{f}(t+\Delta t) + \boldsymbol{f}(t)) \qquad (A.8)$$

と表すことができる。

付録 B. 分子間ポテンシャル

アルゴン原子、ネオン原子に働く相互作用ポテンシャルとして Lennard-Jones ポテンシャルを用いた。Lennard-Jones ポテンシャルはアルゴン原子などの希ガスのモデルポテンシャルとしてよく用いられるもので、

$$U(r_{ij}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{B.1}$$

と表せる（図 B-1）。r_{ij}は原子iと原子j $(i \neq j)$の間の距離である。ここではr原子間の距離、εとσ は Lennard-Jones パラメータで、原子の種類によって異なる値を持つ（表 B-1）。したがって原子iと原子j $(i \neq j)$の間にはたらく力$\boldsymbol{f}_{ij}$は、

$$\boldsymbol{f}_{ij} = -\nabla U(r_{ij}) \tag{B.2}$$

と表せる。よってすべての原子が原子i に及ぼす力は

$$\boldsymbol{f}_{ij} = \sum_{j\neq i} -\nabla U\left(r_{ij}\right) \tag{B.3}$$

$$= \sum_{j\neq i} 24\varepsilon\left[-2\left(\frac{\sigma^{12}}{{r_{ij}}^{13}}\right) + \left(\frac{\sigma^{6}}{{r_{ij}}^{7}}\right)\right] \tag{B.4}$$

となる。

同種原子間のポテンシャルについて、本研究では球はアルゴン原子、基板はネオン原子で構成されているので、表 B-1[14]に基づきアルゴン原子とネオン原子の Lennard-Jones パラメータをそれぞれ$(\varepsilon_{\mathrm{Ar}}\ ,\sigma_{\mathrm{Ar}}) = (0.5315 \times 10^{-21}\mathrm{J},\ 0.2786\ \mathrm{nm})$、$(\varepsilon_{\mathrm{Ne}}\ ,\sigma_{\mathrm{Ne}}) = (1.6539 \times 10^{-21}\mathrm{J},\ 0.3405\ \mathrm{nm})$とおき、ネオン-ネオン間の原子間ポテンシャルとアルゴン-アルゴン間のポテンシャルを定めた。図 2.4-1 はネオン-ネオン間のポテンシャルを表している。

次に異種原子間のポテンシャルについて、すなわちネオン-アルゴン間のポテンシャルについては Lennard-Jones パラメータを

$$\varepsilon_{\mathrm{Ar,Ne}} = \sqrt{\varepsilon_{\mathrm{Ar}}\varepsilon_{\mathrm{Ne}}}$$
$$\sigma_{\mathrm{Ar,Ne}} = \frac{\sigma_{\mathrm{Ar}} + \sigma_{\mathrm{Ne}}}{2} \tag{B.5}$$

と定めた[12]。

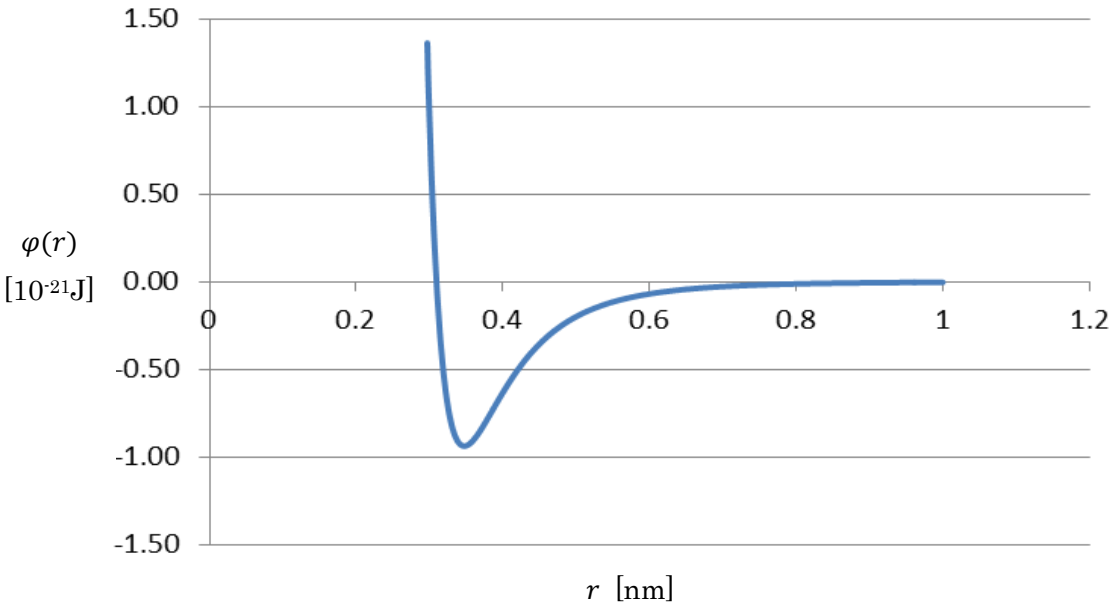


図 B-1Lennard-Jones ポテンシャル

表 B-1　Lennard-Jones パラメータ

原子名	質量　10^{-7}kg	ε　10^{-21}J	σ　nm
Ne	33.51	0.5315	0.2786
Ar	66.34	1.6539	0.3405
Kr	139.16	2.2075	0.3639
Xe	218.02	3.0497	0.3962
Cu	105.52	65.626	0.2338
Ag	179.13	55.276	0.2644

付録 C. シミュレーションで用いたプログラム

本研究で用いたプログラムコードを以下に示す。

```
!ccccccccccccccccccccccccccccccc
! sample.f90
! Lenard-Jones 系の MD のサンプルコード
!ccccccccccccccccccccccccccccccc

        implicit none
        integer :: ntime, iz, ic, ip, p, f
        integer, parameter :: np1 = 5226 , np2 = 116 ! 粒子数
        integer, parameter :: nranmx = 100000 ! 乱数の数
        integer, parameter :: ntimemx = 10000 ! 計算ステップ数
        real :: ran(nranmx)
        real(8) :: x1(np1), y1(np1), z1(np1), vx1(np1),vy1(np1),vz1(np1)
        real(8) :: x2(np2), y2(np2), z2(np2), vx2(np2),vy2(np2),vz2(np2)
        real(8) :: fx1(np1), fy1(np1), fz1(np1), u1(np1), m1
        real(8) :: fx2(np2), fy2(np2), fz2(np2), u2(np2), m2
        real(8) :: temp, gx, gy, gz, tscal, pi, rcoff1, rcoff2, pv,ptheta,pomega
        real(8) :: np1_inv, np2_inv, iniene, inipos2, Lx,Ly,Lz,sumfx, sumfz, csr
        real(8) :: gxx, gyy, gzz, theta0
!-------------------------------------------------------
!       計算のための定数
!-------------------------------------------------------
        np1_inv = 8.d0 / dble(2*np1)
        np2_inv = 8.d0 / dble(2*np2)
        tscal = 2.d0 / 3.d0 ! 運動エネルギー --> 温度への変換定数

        pi = 4.d0 * datan(1.d0)
        ic = 99 ! 動画ファイル名のためのカウンタ初期値
        m1=1.d0      !板の質量
        m2=1.98d0   !球の質量

!-------------------------------------------------------
!      物理定数
```

```
!-----------------------------------------------------

        temp =   2.d-2
        gxx   =   5.d-2
        gyy   =   0.d-1          重力加速度
        gzz   = -1.d-2
        p=0
        csr=0.49d0

        do f=50,89
         csr =csr+0.01d0
         ic=99
         pv=0.d0
         ptheta=0.d0
         pomega=0.d0
!-----------------------------------------------------
!     原子の初期条件
!-----------------------------------------------------

         iz = 0       ! 乱数発生シード
         call iniposi(np1, np2, x1, y1, z1, x2, y2, z2) ! 初期配置
         do ip=1,np2
          x2(ip)=x2(ip)-15.d0          !★☆★☆★☆★☆★☆★☆
          y2(ip)=y2(ip)+51.d0          !★☆★☆★☆★☆★☆★☆       球の位置
          z2(ip)=z2(ip)+3.1d0          !★☆★☆★☆★☆★☆★☆

         end do

         call rancal(nranmx, iz, ran) ! 乱数発生
         call inivel(np1, np2, temp, ran, vx1, vy1, vz1, vx2, vy2, vz2, pi,nranmx,
m1,m2) ! 初期速度

!-----------------------------------------
!       系の熱平衡化
!-----------------------------------------
        gx = 0.d0
```

```
        gy = 0.d0
        gz = 0.d0

        call thermal(x1,y1,z1,x2,y2,z2,vx1,vy1,vz1, vx2, vy2, vz2, fx1, fy1, fz1, fx2, fy2, fz2,
np1, np2, u1, u2, gx,gy,gz,temp,p,ic, ntime, rcoff1, rcoff2, iniene, m1, m2,sumfx,
sumfz,csr, pv, ptheta,pomega, f, tscal, theta0)

        gx = gxx
        gy = gyy
        gz = gzz

!-----------------------------------------
!       時間発展ループ
!-----------------------------------------

        p=1

        call forcecal(x1, y1, z1, x2, y2, z2, fx1, fy1, fz1, fx2, fy2, fz2, np1,          np2, u1,
u2, gx,gy, gz, p,ntime, rcoff1, rcoff2, m1, m2,sumfx, sumfz,csr)


        ntime = 0
        call data(x1, y1, z1, x2, y2, z2, vx1, vy1, vz1, vx2, vy2, vz2, u1,  u2, np1, np2,
np1_inv, np2_inv, ntime, tscal, iniene, inipos2, gx, gy, gz, ic, m1, m2,sumfx, sumfz,f, pv,
ptheta,pomega, theta0) ! データファイルの作成


        time_evol: do ntime = 1, ntimemx

        call symp(x1 , y1 , z1, x2, y2, z2, vx1, vy1, vz1, vx2, vy2, vz2, fx1, fy1, fz1,
fx2,fy2,fz2,u1,u2,gx,gy,gz,np1, np2,p,ntime,rcoff1, rcoff2,m1,m2, sumfx, sumfz, csr) ! シ
ンプレクティック積分


        if (mod(ntime, 100) == 0) then ! 100 ステップ毎にデータをとる
```

```
            call data(x1, y1, z1, x2, y2, z2, vx1, vy1, vz1, vx2, vy2, vz2, u1,   u2              ,
np1, np2, np1_inv, np2_inv, ntime, tscal, iniene, inipos2, gx, gy, gz, ic,m1, m2,sumfx,
sumfz,   f, pv, ptheta,pomega, theta0)!データの作成

         end if

         if (mod(ntime, 100) == 0) then ! 100 ステップ毎に動画データをとる
            ic = ic +1
            call mov(ic, x1, y1, z1, x2, y2, z2, np1, np2, f) !動画ファイル作成。
         end if

        if (sum(x2)/size(x2)>=30.d0) exit

        end do time_evol

        close(210)
        close(220)
        close(230)
        close(240)

        end do

         end

!ccccccccccccccccccccccccccccc
        subroutine iniposi(npd1, npd2, x1d, y1d, z1d, x2d, y2d, z2d)
!ccccccccccccccccccccccccccccc

            implicit none
            integer :: npd1, npd2, i
            real(8) :: x1d(npd1), y1d(npd1), z1d(npd1)
            real(8) :: x2d(npd2), y2d(npd2), z2d(npd2)
!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆
            open(10, file = 'ract5226.d' ) !cube* , bord* , sphe*
            do i = 1, npd1
                read(10,*) x1d(i), y1d(i), z1d(i)
```

```
        end do
      close(10)

!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆
        open(20, file = 'spheAr116.d' )
        do i = 1, npd2
          read(20,*) x2d(i), y2d(i), z2d(i)
        end do
        close(20)
!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆

      end subroutine iniposi

!ccccccccccccccccccccccccccccc
subroutine thermal(x1,y1,z1,x2,y2, z2,vx1,vy1,vz1,vx2,vy2,vz2,fx1,fy1,fz1,
fx2,fy2,fz2,np1, np2, u1, u2, gx,gy, gz, temp, p,ic,ntime,rcoff1, rcoff2, iniene, m1,
m2,sumfx, sumfz,csr, pv, ptheta,pomega, f, tscal, theta0)
!ccccccccccccccccccccccccccccc

        implicit none
        integer :: np1, np2, entime,  i , p, ic,ntime, f
        real(8) :: x1(np1), y1(np1), z1(np1), vx1(np1), vy1(np1), vz1(np1), fx1(np1),
fy1(np1), fz1(np1), u1(np1)
        real(8) :: x2(np2), y2(np2), z2(np2), vx2(np2), vy2(np2), vz2(np2), fx2(np2),
fy2(np2), fz2(np2), u2(np2)
        real(8) :: rxi, ryi, rzi, vxi, vyi, vzi, gx,gy, gz, rcoff1, rcoff2,sumfx, sumfz
        real(8) :: eh, eh2,  ehsq, ehsq2, iniene, inipos2, temp, theta0
        real(8) :: np1_inv, np2_inv, tscal, m1, m2, csr, pv, ptheta,pomega

!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆
        integer, parameter :: entimemx=100 ! 熱平衡化のためのステップ数
!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆


        ntime=0
```

```
eh = 1.d-2
eh2 = eh/2.d0
ehsq = eh*eh
ehsq2 =ehsq/2.d0

call forcecal(x1, y1, z1, x2, y2, z2, fx1, fy1, fz1, fx2, fy2, fz2, np1, np2, u1, u2, gx,gy,
gz, p, ntime, rcoff1, rcoff2, m1, m2,sumfx, sumfz, csr)
main_loop : do ntime=1,entimemx


forall(i=1 : 420)
vx1(i)=0.d0
vy1(i)=0.d0
vz1(i)=0.d0
endforall

forall(i=1261 : 1680)
vx1(i)=0.d0
vy1(i)=0.d0
vz1(i)=0.d0
endforall

x1 = x1 + eh*vx1 + ehsq2*fx1
y1 = y1 + eh*vy1 + ehsq2*fy1
z1 = z1 + eh*vz1 + ehsq2*fz1

vx1 = vx1 + eh2*fx1
vy1 = vy1 + eh2*fy1
vz1 = vz1 + eh2*fz1

x2 = x2 + eh*vx2 + ehsq2*fx2
y2 = y2 + eh*vy2 + ehsq2*fy2
z2 = z2 + eh*vz2 + ehsq2*fz2

vx2 = vx2 + eh2*fx2
vy2 = vy2 + eh2*fy2
```

```
                vz2 = vz2 + eh2*fz2


                call forcecal(x1, y1, z1, x2, y2, z2, fx1, fy1, fz1, fx2, fy2,fz2,
np1, np2,u1, u2,gx,gy,gz,p,ntime, rcoff1, rcoff2, m1, m2,sumfx,                    sumfz,
csr)

                vx1 = vx1 + eh2*fx1
                vy1 = vy1 + eh2*fy1
                vz1 = vz1 + eh2*fz1

                vx2 = vx2 + eh2*fx2
                vy2 = vy2 + eh2*fy2
                vz2 = vz2 + eh2*fz2

!--------------------------------------------------
!       速度スケーリング
!--------------------------------------------------

            if (mod(ntime,10).eq.0) then

            call scalevel(np1, np2, temp, m1, m2, vx1, vy1, vz1, vx2, vy2, vz2)

             if (mod(ntime, 100) == 0) then ! 100 ステップ毎に動画データをとる
                ic = ic +1
             end if
            end if
            end do main_loop

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!          print *, 'equilibration finished'
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      end subroutine thermal

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      subroutine forcecal(x1, y1, z1, x2, y2, z2,fx1,fy1,fz1,fx2,fy2,fz2, np1,np2, u1, u2,
gx,gy, gz, p, ntime, rcoff1, rcoff2,m1,m2,sumfx, sumfz, csr)
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      implicit none
      integer :: i, j, np1,np2, p,ntime
      real(8) :: x1(np1),y1(np1),z1(np1),fx1(np1), fy1(np1), fz1(np1), u1(np1)
      real(8) :: x2(np2),y2(np2),z2(np2),fx2(np2), fy2(np2), fz2(np2), u2(np2)
      real(8) :: rxi, ryi, rzi, fxi, fyi, fzi, fij
      real(8) :: rxij, ryij, rzij, rijsq, fxij, fyij, fzij
      real(8) :: sr2, sr6, sr12, rcoff1, rcoff2, m1, m2
      real(8) :: ui, uij, gx,gy, gz, pv, ptheta,pomega
      real(8) :: rxij1, ryij1, rzij1,rxij2, ryij2, rzij2
      real(8) :: csr, sig1, sig2, sigav, eps1, eps2, epsav, sumfx, sumfz

!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆
!パラメータ設定
!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆
      sig1=1.d0          !RJ パラメータ
      sig2=1.22d0
      eps1=1.d0
      eps2=3.11d0
      sigav=(sig1+sig2)/2
      epsav=sqrt(eps1*eps2)
!★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆★☆
!----------------------------------
!       カットオフ距離の設定
!----------------------------------

      rcoff1 = 4.d0*sig1
      rcoff2 = 4.d0*sig2

!----------------------------------
!       配列の初期化
!----------------------------------
```

```
        fx1 = 0.d0
        fy1 = 0.d0
        fz1 = 0.d0
        u1 = 0.d0

        fx2 = 0.d0
        fy2 = 0.d0
        fz2 = 0.d0
        u2 = 0.d0

!----------------------------------
!      原子間相互作用の計算
!----------------------------------


      do   i = 1, np1-1
        fxi = fx1(i)
        fyi = fy1(i)
        fzi = fz1(i)
        ui = u1(i)

        do   j = i+1, np1

        if ((mod(ntime, 100)==0).or.(ntime==0))then

        end if

            rxij = x1(i) - x1(j)
            ryij = y1(i) - y1(j)
            rzij = z1(i) - z1(j)
            rijsq = rxij*rxij + ryij*ryij + rzij*rzij

            if (rijsq.lt.rcoff1)then
              sr2 = sig1 / rijsq
              sr6 = sr2*sr2*sr2
              sr12 = sr6*sr6
```

```
            fij = 2.4d1 *eps1* (2.d0 * sr12 - sr6) * sr2 /sig1！基板 RJ 力
            uij = 4.d0 *eps1* (sr12 - sr6)                    ! RJ ポテンシャル

            fxij = fij * rxij
            fyij = fij * ryij
            fzij = fij * rzij
            fxi = fxi + fxij
            fyi = fyi + fyij
            fzi = fzi + fzij
            ui = ui + uij

            fx1(j)=fx1(j)-fxij
            fy1(j)=fy1(j)-fyij
            fz1(j)=fz1(j)-fzij
            u1(j)=u1(j)+uij
          end if
      end do

      fx1(i) = fxi
      fy1(i) = fyi
      fz1(i) = fzi
      u1(i) = ui

end do

do   i = 1, np2-1

      fxi = fx2(i)
      fyi = fy2(i)
      fzi = fz2(i)
      ui = u2(i)

      do   j = i+1, np2
          rxij = x2(i) - x2(j)
          ryij = y2(i) - y2(j)
```

```
            rzij = z2(i) - z2(j)
            rijsq = rxij*rxij + ryij*ryij + rzij*rzij

          if (rijsq.lt.rcoff2)then
              sr2 = sig2 / rijsq
              sr6 = sr2*sr2*sr2
              sr12 = sr6*sr6

              fij = 2.4d1 *eps2* (2.d0 * sr12 -sr6) * sr2/sig2     !球 RJ 力
              uij = 4.d0 *eps2* (sr12 - sr6)                       ! RJ ポテンシャル
              fxij = fij * rxij
              fyij = fij * ryij
              fzij = fij * rzij
              fxi = fxi + fxij
              fyi = fyi + fyij
              fzi = fzi + fzij
              ui = ui + uij

              fx2(j)=fx2(j)-fxij
              fy2(j)=fy2(j)-fyij
              fz2(j)=fz2(j)-fzij
              u2(j)=u2(j)+uij
            end if
        end do

          fx2(i) = fxi
          fy2(i) = fyi
          fz2(i) = fzi
          u2(i)   = ui
      end do

!----------------------------------
!       重力
!----------------------------------

      do i = 1, np1
```

```
            fx1(i) = fx1(i) + m1 * gx
            fy1(i) = fy1(i) + m1 * gy
            fz1(i) = fz1(i) + m1 * gz
          end do
          do i = 1, np2
            fx2(i) = fx2(i) + m2 * gx
            fy2(i) = fy2(i) + m2 * gy
            fz2(i) = fz2(i) + m2 * gz
          end do


!-----------------------------------
!      分子間相互作用の計算
!-----------------------------------
         i=0

          do   i = 1, np1
            fxi = fx1(i)
            fyi = fy1(i)
            fzi = fz1(i)
            ui   = u1(i)

            do   j = 1, np2

              rxij = x1(i) - x2(j)
              ryij = y1(i) - y2(j)
              rzij = z1(i) - z2(j)
              rijsq = rxij*rxij + ryij*ryij + rzij*rzij

             if (rijsq.lt.((sig1+sig2)*3.d0))then

              sr2 = sigav / rijsq
              sr6 = sr2*sr2*sr2
              sr12 = sr6*sr6

              fij = 2.4d1 *epsav* (2.d0 * sr12 - csr*sr6) * sr2/sigav     ! RJ 力
```

```
            uij = 4.d0 *epsav* (sr12 - csr*sr6)                    ! RJ ポテンシャル

            fxij = fij * rxij
            fyij = fij * ryij
            fzij = fij * rzij
            fxi = fxi + fxij
            fyi = fyi + fyij
            fzi = fzi + fzij
            ui = ui + uij
            fx2(j) = fx2(j) -fxij
            fy2(j) = fy2(j) -fyij
            fz2(j) = fz2(j) -fzij
            u2(j) = u2(j) + uij
          end if
          end do

          fx1(i) = fxi
          fy1(i) = fyi
          fz1(i) = fzi
          u1(i) = ui

forall(i=1：420)
fx1(i)=0.d0
fy1(i)=0.d0
fz1(i)=0.d0
u1(i)=0
endforall
forall(i=1261：1680)
fx1(i)=0.d0
fy1(i)=0.d0
fz1(i)=0.d0
u1(i)=0
endforall
        end do

          sumfx = sum(fx2)
```

```
        sumfz = -sum(fz2)

    end subroutine forcecal

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine symp(x1 , y1 , z1, x2, y2, z2, vx1, vy1, vz1, vx2, vy2, vz2, fx1,fy1,
fz1,fx2,fy2,fz2,u1,u2,gx,gy,gz,np1,np2,p,ntime,rcoff1, rcoff2,m1,m2,sumfx, sumfz, csr)
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

    implicit none
    integer :: np1, np2, i, p,ntime
    real(8) :: h, h2, hsq, hsq2, gx,gy, gz, rcoff1, rcoff2, m1, m2, sumfx, sumfz
    real(8) :: x1(np1), y1(np1), z1(np1), vx1(np1), vy1(np1), vz1(np1), u1(np2)
    real(8) :: x2(np2), y2(np2), z2(np2), vx2(np2), vy2(np2), vz2(np2), u2(np2)
    real(8):: fx1(np1), fy1(np1), fz1(np1), fx2(np2), fy2(np2), fz2(np2), csr

    h =1.d-2
    h2 = h/2.d0
    hsq = h*h
    hsq2 = hsq/2.d0

    forall(i=1 : 420)
    vx1(i)=0.d0
    vy1(i)=0.d0
    vz1(i)=0.d0
    endforall
    forall(i=1261 : 1680)
    vx1(i)=0.d0
    vy1(i)=0.d0
    vz1(i)=0.d0
    endforall



    x1 = x1 + h * vx1 + hsq2 * fx1 / m1
    y1 = y1 + h * vy1 + hsq2 * fy1 / m1
```

```
    z1 = z1 + h * vz1 + hsq2 * fz1 / m1

    vx1 = vx1 + h2 * fx1 / m1
    vy1 = vy1 + h2 * fy1 / m1
    vz1 = vz1 + h2 * fz1 / m1

    x2 = x2 + h * vx2 + hsq2 * fx2 / m2
    y2 = y2 + h * vy2 + hsq2 * fy2 / m2
    z2 = z2 + h * vz2 + hsq2 * fz2 / m2

    vx2 = vx2 + h2 * fx2 / m2
    vy2 = vy2 + h2 * fy2 / m2
    vz2 = vz2 + h2 * fz2 / m2


    call forcecal(x1, y1, z1, x2, y2,
z2,fx1,fy1,fz1,fx2,fy2,fz2,np1,np2,u1,u2,gx ,gy,gz,p,ntime, rcoff1, rcoff2,m1,m2, sumfx,
sumfz, csr)


    vx1 = vx1 + h2 * fx1 / m1
    vy1 = vy1 + h2 * fy1 / m1
    vz1 = vz1 + h2 * fz1 / m1

    vx2 = vx2 + h2 * fx2 / m2
    vy2 = vy2 + h2 * fy2 / m2
    vz2 = vz2 + h2 * fz2 / m2


     end subroutine symp

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
subroutine inivel(n1, n2, temp, ran, vxd1, vyd1, vzd1,  vxd2, vyd2, vzd2, pi,nranmx,
m1,m2)
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        implicit none
```

```
      integer :: nran, n1, n2, i, nranmx
      real(8) :: vxd1(n1), vyd1(n1), vzd1(n1), vxd2(n2), vyd2(n2), vzd2(n2), pi
      real :: ran(nranmx)
      real(8) :: c0, c1, c2, c3, t, vchk, temp, m1, m2

      c0=2.d0*pi
      vchk=2.d0*temp*3.5d0**2
      t=temp*2.d0

      nran=1

      do     i = 1, n1
5        c1=dsqrt(-t/m1*dlog(dble(ran(nran))))
         nran=nran+1
         c1=c1*dsin(c0*dble(ran(nran)))
         nran=nran+1

         c2=dsqrt(-t/m1*dlog(dble(ran(nran))))
         nran=nran+1
         c2=c2*dsin(c0*dble(ran(nran)))
         nran=nran+1

         c3=dsqrt(-t/m1*dlog(dble(ran(nran))))
         nran=nran+1
         c3=c3*dsin(c0*dble(ran(nran)))
         nran=nran+1

         if((c1*c1+c2*c2+c3*c3).ge.vchk) goto 5
         vxd1(i) = c1
         vyd1(i) = c2
         vzd1(i) = c3
      end do

        c1=0.d0
        c2=0.d0
        c3=0.d0
```

```
      do     i = 1, n2
 6        c1=dsqrt(-t/m2*dlog(dble(ran(nran))))
          nran=nran+1
          c1=c1*dsin(c0*dble(ran(nran)))
          nran=nran+1

          c2=dsqrt(-t/m2*dlog(dble(ran(nran))))
          nran=nran+1
          c2=c2*dsin(c0*dble(ran(nran)))
          nran=nran+1

          c3=dsqrt(-t/m2*dlog(dble(ran(nran))))
          nran=nran+1
          c3=c3*dsin(c0*dble(ran(nran)))
          nran=nran+1

          if((c1*c1+c2*c2+c3*c3).ge.vchk) goto 6
          vxd2(i) = c1
          vyd2(i) = c2
          vzd2(i) = c3

      end do



      end

!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine scalevel(n1, n2, temp, m1, m2, vx1, vy1, vz1, vx2, vy2, vz2)
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      implicit none
      integer ::   i, n1, n2
      real(8) ::   vcx1, vcy1, vcz1,vcsq1,vcx2, vcy2, vcz2,vcsq2
      real(8) ::   vdx1(n1), vdy1(n1), vdz1(n1), vdx2(n2), vdy2(n2), vdz2(n2)
      real(8) ::   vx1(n1), vy1(n1), vz1(n1), vx2(n2), vy2(n2), vz2(n2)
```

```
real(8) ::   t1, t2, c1, temp, m1, m2

        vcx1 = 0.d0
        vcy1 = 0.d0
        vcz1 = 0.d0
        vcsq1 = 0.d0
        vcx2 = 0.d0
        vcy2 = 0.d0
        vcz2 = 0.d0
        vcsq2 = 0.d0


        vcx1 = sum(vx1)/size(vx1)
        vcy1 = sum(vy1)/size(vy1)
        vcz1 = sum(vz1)/size(vz1)
        vcx2 = sum(vx2)/size(vx2)
        vcy2 = sum(vy2)/size(vy2)
        vcz2 = sum(vz2)/size(vz2)

        vdx1 = vx1-vcx1
        vdy1 = vy1-vcy1
        vdz1 = vz1-vcz1
        vdx2 = vx2-vcx2
        vdy2 = vy2-vcy2
        vdz2 = vz2-vcz2

        vcsq1 =          5.d-1*m1*dot_product(vdx1, vdx1)/ size(vx1)
        vcsq1 = vcsq1 + 5.d-1*m1*dot_product(vdy1, vdy1)/ size(vx1)
        vcsq1 = vcsq1 + 5.d-1*m1*dot_product(vdz1, vdz1)/ size(vx1)
        vcsq2 =          5.d-1*m2*dot_product(vdx2, vdx2)/ size(vx2)
        vcsq2 = vcsq2 + 5.d-1*m2*dot_product(vdy2, vdy2)/ size(vx2)
        vcsq2 = vcsq2 + 5.d-1*m2*dot_product(vdz2, vdz2)/ size(vx2)

        t1 = vcsq1 * 2.d0 / 3.d0
        t2 = vcsq2 * 2.d0 / 3.d0
```

```
            c1 = dsqrt(temp/(t1+t2))
            vx1 = vx1*c1
            vy1 = vy1*c1
            vz1 = vz1*c1
            vx2 = vx2*c1
            vy2 = vy2*c1
            vz2 = vz2*c1


            vcx1 = 0.d0
            vcy1 = 0.d0
            vcz1 = 0.d0
            vcsq1 = 0.d0
            vcx2 = 0.d0
            vcy2 = 0.d0
            vcz2 = 0.d0
            vcsq2 = 0.d0



        end

!ccccccccccccccccccccccccccccccccccccc
        subroutine rancal(n,iz,x)
!ccccccccccccccccccccccccccccccccccccc

        implicit none
        integer ::   n,integmx,integst,integ,iz,i
        real x(n),aintegmx
        integmx = 2147483647
        integst = 584287
        integ = 48828125

        aintegmx=real(integmx)

        if (iz.eq.0) iz=integst
```

```
      do i=1,n
         iz=iz*integ
         if (iz) 10, 20, 20
10       iz = (iz + integmx) + 1
20       x(i)=real(iz)/aintegmx
      end do
      end


!ccccccccccccccccccccccccccccccccccccc
      subroutine data(x1, y1, z1, x2, y2, z2, vx1, vy1, vz1, vx2, vy2, vz2, u1, u2, n1, n2,
np1_inv, np2_inv, ntime, tscal, iniene, inipos2, gx, gy, gz, it, m1, m2,sumfx, sumfz, f, pv,
ptheta,pomega, theta0)
!ccccccccccccccccccccccccccccccccccccc

         implicit none
         integer   :: n1, n2,   ntime, i, it, f
         real(8) :: x1(n1), y1(n1), z1(n1), vx1(n1), vy1(n1), vz1(n1), u1(n1)
         real(8) :: x2(n2), y2(n2), z2(n2), vx2(n2), vy2(n2), vz2(n2), u2(n2)
         real(8) :: xi, yi, zi, vxc1, vyc1, vzc1, vxc2, vyc2, vzc2, pos2, gx, gy,, gz, m1,
m2,mass
         real(8) :: kes, pos, temp, np1_inv, np2_inv, tscal, ene, fluc, pi
         real(8) :: iniene, inikes, inipos, inipos2, kesx, kesy, kesz, pv,ptheta,
pomega,a,mu2, theta0
         real(8) :: xc2, yc2, zc2, xp2, zp2, xd2, zd2, sumfx, sumfz, theta,omega,
omega2,mu, kes2, kesx2, kesy2, kesz2, time, gxx
         character*6 :: dat
         pi = 4.d0 * datan(1.d0)


!---------------------------------------
!        粒子の位置
!---------------------------------------

         dat = "          "
         write( dat, '( I5 )' )   f
```

```
        dat = adjustl( dat )

        vxc2 = 0.d0
        vyc2 = 0.d0
        vzc2 = 0.d0
        kes = 0.d0
        pos = 0.d0
        pos2= 0.d0

        xc2 = sum(x2)/size(x2)
        yc2 = sum(y2)/size(y2)   !位置平均
        zc2 = sum(z2)/size(z2)

        vxc1 = sum(vx1)/size(vx1)
        vyc1 = sum(vy1)/size(vy1)   !速度平均
        vzc1 = sum(vz1)/size(vz1)
        vxc2 = sum(vx2)/size(vx2)
        vyc2 = sum(vy2)/size(vy2)   !速度平均
        vzc2 = sum(vz2)/size(vz2)

        pos = pos + sum(u1) + sum(u2)
pos2=gx*(m1*sum(x1)+m2*sum(x2))+gy*(m1*sum(y1)+m2*sum(y2))-gz*(m1*sum(z1)+
m2*sum(z2))
kesx=5.d-1*m1*dot_product(vx1,vx1)/size(vx1)+5.d-1*m2*dot_product(vx2,vx2)/size(vx
2)
kesy=5.d-1*m1*dot_product(vy1,vy1)/size(vx1)+5.d-1*m2*dot_product(vy2,vy2)/size(vx
2)
kesz=5.d-1*m1*dot_product(vz1,vz1)/size(vx1)+5.d-1*m2*dot_product(vz2,vz2)/
size(vx2)
        kes   = kesx+kesy+kesz

kesx2=5.d-1*m1*dot_product(vx1-vxc1,vx1-vxc1)/size(vx1)+5.d-1*m2*dot_product(vx2-
vxc2, vx2-vxc2)/ size(vx2)
kesy2=5.d-1*m1*dot_product(vy1-vyc1,vy1-vyc1)/size(vx1)+5.d-1*m2*dot_product(vy2-
vyc2, vy2-vyc2)/ size(vx2)
kesz2=5.d-1*m1*dot_product(vz1-vzc1,vz1-vzc1)/size(vx1)+5.d-1*m2*dot_product(vz2-v
```

```
zc2, vz2-vzc2)/ size(vx2)
        kes2 = kesx2 + kesy2 + kesz2
        temp = kes2 * tscal   ！温度
        if(ntime == 0) then
          inipos2= pos2
          end if
        pos2 = -pos2 + inipos2
        ene = kes + pos + pos2
        if(ntime == 0) then
          iniene = ene
          end if
        fluc = -1*abs(ene - iniene) / iniene !★☆★☆★☆★☆★☆★☆

        xp2 = (x2(6)+x2(14)+x2(17)+x2(23)+x2(27)+x2(32)+x2(38))/7
        zp2 = (z2(6)+z2(14)+z2(17)+z2(23)+z2(27)+z2(32)+z2(38))/7

        xd2 = xp2-xc2
        zd2 = zp2-zc2

!★☆★☆★☆★☆★☆★☆ω微分の計算★☆★☆★☆★☆★☆★☆★☆
        theta = -datan(zd2/xd2)

        if (ntime == 0)then
        theta0=theta
        end if
        theta = theta - theta0

        if (theta - ptheta <= -1.d0) then
        theta0 = theta0 - pi
        theta = theta + pi
        end if
        if (theta - ptheta >= 1.d0) then
        theta0 = theta0 + pi
        theta = theta - pi
        end if
```

```
            omega = (theta-ptheta)/1.d0
            omega2 = (omega-pomega)/1.d0
            ptheta=theta

!★☆★☆★☆★☆加速度の計算☆★☆★☆★☆★☆★☆★☆★☆★☆★☆
            a=(vxc2-pv)/1.d0
!★☆★☆★☆摩擦係数の計算★☆★☆★☆★☆★☆★☆★☆★☆★☆

            pv = vxc2

            pomega=omega
            time = ntime*0.0229d-2
            omega= omega / 0.0229d0
            omega2= omega2 / (0.0229d0)**2.d0
            xc2    = xc2 * 2.88d-10
            zc2    = zc2 * 2.88d-10
            temp = temp*5.315d-22/1.38d-23
            vxc2 = vxc2*2.88d-10/0.0229d0
            vyc2 = vyc2*2.88d-10/0.0229d0
            vzc2 = vzc2*2.88d-10/0.0229d0
            a       = a*2.88d-10/(0.0229d0)**2.d0
            sumfx = sumfx*1.85d-12
            sumfz = sumfz*1.85d-12
            mass = m2 * 3.35d-6
            gxx     = gx*2.88d-10/(0.0229d0)**2
            mu = (mass*n2*gxx-mass*n2*(5.01d-10*omega2+a))/sumfz*-1.d0
            mu2= -1.d0*sumfx/sumfz
print *, mass, n2, gxx, omega2, a, sumfz

            open(210, file= trim( dat ) // 'T,fluc.dat')
            open(220, file= trim( dat ) // 'vx,vz.dat')
            open(230, file= trim( dat ) // 'mu, a.dat')
            open(240, file= trim( dat ) // 'th,om.dat')
            open(250, file= trim( dat ) // 'x2,z2.dat')
            open(260, file= trim( dat ) // 'fx,fz.dat')
```

```
            write(210, *) time, temp, fluc
            write(220, *) time, vxc2, vzc2
            write(230, *) time, mu,a
            write(240, *) time, theta, omega
            write(250, *) time, xc2, zc2
            write(260, *) time, sumfx-mass*a, sumfz

        end subroutine data

!ccccccccccccccccccccccccccccccccccccc
        subroutine mov(it, x1, y1, z1, x2, y2, z2, n1, n2, f)
!ccccccccccccccccccccccccccccccccccccc

            implicit none
            integer :: it, n1, n2, i, f
            real(8) :: x1(n1), y1(n1), z1(n1), x2(n2), y2(n2), z2(n2)
            character*6 :: dat, dat2

!-----------------------------------------
!          CDVIEW
!-----------------------------------------

            dat = "          "
            dat2 = "          "
            write( dat, '( I5 )' )    it
            write( dat2, '( I5 )' )    f
            dat = adjustl( dat )
            dat2 = adjustl( dat2 )
    open( 140, file = "a" // trim( dat2 ) // "h" // trim( dat ) // ".dat",status='new' )

            do i = 1, n1
                write(140,122) i,'    0', x1(i), y1(i), z1(i)
122             format(I4,a4,2x,f17.13,2x,f17.13,2x,f17.13)
            end do
            do i = 1, n2
                write(140,123) i,'    1', x2(i), y2(i), z2(i)
```

```
123         format(I4,a4,2x,f17.13,2x,f17.13,2x,f17.13)
          end do
          close(140)
        end subroutine mov
```