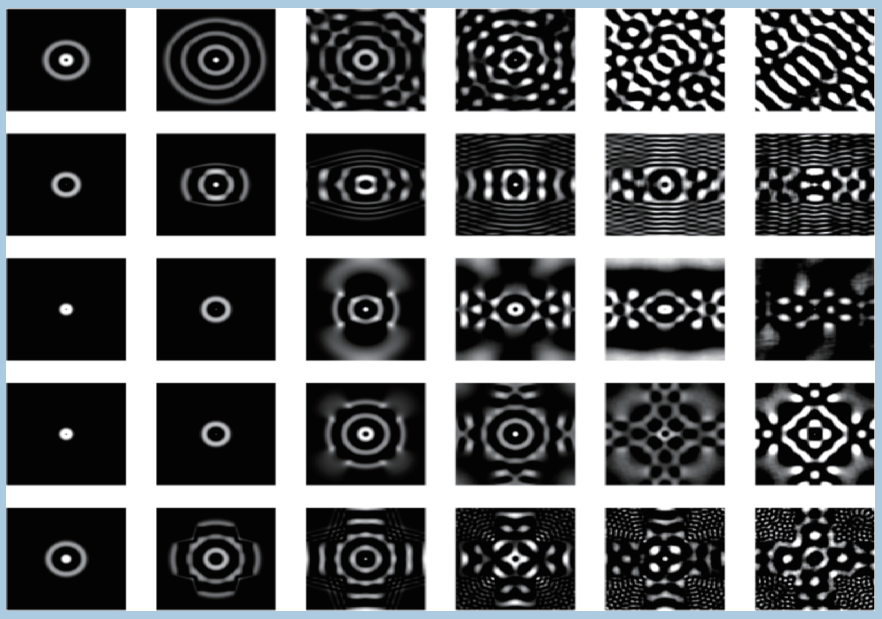
ISSN 0915-6089

形の科学会誌

第 35 巻 第 2 号 2020

Bulletin of the Society for Science on Form



形の科学会

https://katachi-jp.com/

形の科学会誌 第 35 巻 第 2 号 (2020)

目　次

非一様な計算周期を持つ 非同期分散シミュレータが生成する時空間パターン

山岡久俊 [1*)]，蔡東生 [2)]
1) 神奈川県川崎市幸区下平間 214-1
2)筑波大学大学院システム情報工学研究科
*akoamay@gmail.com

Spatio-Temporal Patterns Generated by an Asynchronous Distributed Simulator with Non-uniform Calculation Periods

Hisatoshi Yamaoka [1)], Cai Dongsheng [2)]
1) Shimohirama Saiwai-ku, Kawasaki-shi, Kanagawa-ken
2) Graduate School of Systems and Information Engineering, University of Tsukuba

(2020 年 9 月 1 日受付，2020 年 10 月 15 日受理)

Abstract: A spatiotemporal patterns generated by the simulator with spatially non-uniform calculation period are observed. Regarding to the wave equation, simulation results show that the spatio-temporal patterns of solutions obtained by non-uniform calculation periods varies. Their qualitative properties can be roughly classified whether the wave sources are located in a region where the calculation period is slow or not. In the reaction-diffusion equation, the difference in calculation period does not make a qualitative difference in the spatial pattern. However, from the shape of the generated spatial pattern, it was possible to roughly see where the region with a slow calculation period exists.

Keywords: Asynchronous Distributed Simulator, Spatio-Temporal Patterns, Wave equation, Reaction-Diffusion equation

1. はじめに

1.1. 研究の背景

広域な空間を対象としたシミュレーションのためには，空間を複数の領域に分割し，個別の計算機に割り当てた上で，並列分散で計算を実行する負荷分散方式が有効である[1]. このような領域分割手法にもとづく大規模分散シミュレータの構想を図 1 に示す.

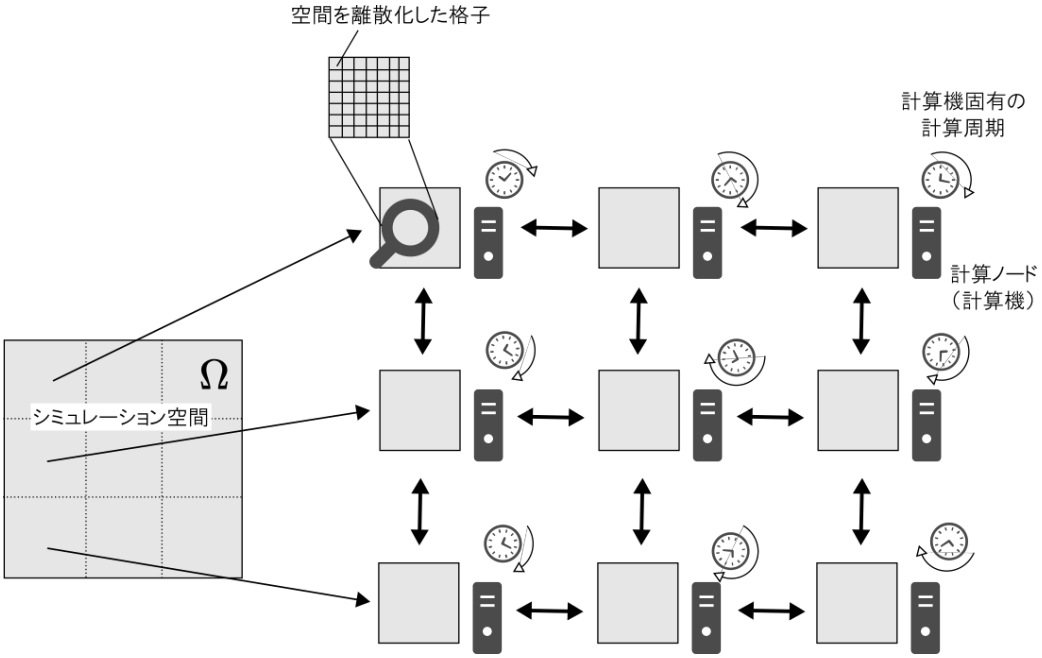


図 1：非同期分散シミュレータの構成

この分散シミュレータは，多数の計算機が格子状にネットワークで結合した構成をとり．シミュレーション空間を格子状に区切ってそれぞれの計算機に割り当てる．各計算機は空間を離散化した格子として保持され，それらの格子に対して，セルオートマトンや，偏微分方程式を離散化して得た遷移規則を作用させる．ある時刻tにおけるシミュレーション対象の系をu_tとし，そこに作用させる遷移規則をfとすると，1 回の計算ステップで$u_{t+1} = f(u_t)$を求め，この計算ステップをそれぞれの計算機が繰り返すことで，空間の時間発展をシミュレーションする．また，区分けした空間の境界に位置する格子の計算のために必要な境界データを，近接する計算機と計算ステップの度にネットワークを介して送受信し合う（図 2）．

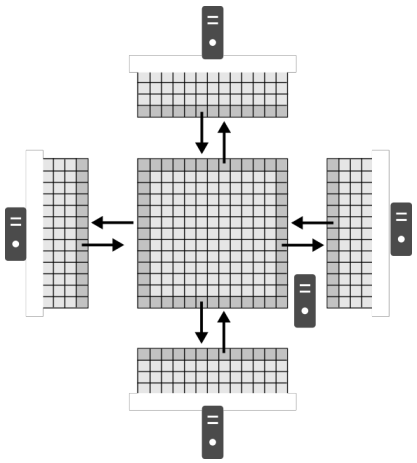


図 2：ネットワークを介した領域境界データの送受信

こうすることで，計算機を増し，取り扱う空間の広さを拡大していくことができる．しかし，このような分散シミュレータでは，性能や負荷状況による違いから，計算ステップに要する時間が個々の計算機ごとに異なる．そのため，厳密なシミュレーションを行うため

には計算機同士の計算実行タイミングの同期が重要となる．同期を行わない場合，各計算機が固有の周期で非同期に計算ステップを繰り返すことになり，空間上の場所によって時間の進む速度が異なることになる．同期を行うためには，計算ステップ実行のたびに，近接する東西南北 4 台の計算機から全てから境界格子データが送信されてくるまで待機する必要があり，これが計算遅延を引き起こす原因となる．同期処理は分散システムの一つの大きな課題となっており，これを効率化する手法は多く研究されている[2,3]．一方で，ゲームなどの，シミュレーションの正確性よりも速度が優先されるケースでは，時間のかかる同期は必要ないケースもある． また，場所によって時間の流れる速度が異なる条件下でシミュレーションを行った際に，系が示す挙動と，その結果生じる時空間パターンについてはこれまで研究されておらず，興味深い．

1.2. 研究の目的と本稿の構成

前項で述べたような，空間の場所によって計算周期が異なる状況でシミュレーションを実行した場合に，どのような現象が生じ得るのか知見を得るための基礎的な研究として，本稿では，まず関連研究を述べ，単純な熱伝導モデルを対象に非一様な計算周期によってシミュレーションを実行した場合の挙動について考察する．次に，図 1 で述べた構成と同等の条件でシミュレーションを実行する疑似環境を用いて，波動方程式と反応拡散方程式を対象としたシミュレーションを行い，その結果として生じた時空間パターンを示す．

2. 関連研究

2.1. 場所によって異なる拡散係数を設定したもの

拡散方程式を対象として，空間的に非一様な拡散係数を持つ系の発展を調べた研究が幾つか存在している．[4]は場所によって異なる拡散係数を持つ媒質上における反応拡散系の進行波解の挙動を確認している．[5]は，生物繁殖による分布域拡大を説明する Fisher モデル[6]と同様の拡散方程式を対象として，生物の繁殖や生存にとって好適な場所（拡散しやすい場所）と，そうでない場所（拡散しにくい場所）がモザイク状に存在していることを想定し，空間を帯状に区切って，それぞれの帯に異なる拡散係数を設定した条件下における，解の振る舞いを解析的・数値的に確認している．これらの研究は，シミュレーション空間の場所に応じて異なる条件設定をして計算をする点で，本研究の取り組みと類似している．しかし，これらが特定の空間領域の拡散係数を他の領域と変えることを想定しているのに対し，本研究の想定は，その領域の計算周期を変えることである．

2.2. セルオートマトンの計算を非同期的に行うもの

[7]は，1 次元のセルオートマトンモデルを対象として，空間を幾つかのブロックに区切った上で，各ブロック内のセルの計算は同期的に行いつつ，ブロックごとの計算の実行タイミングを非同期に実行した時の挙動を確認している．また，[8]は同じく 1 次元のセルオートマトンについて，セルの更新を非同期かつランダムに行った場合の挙動を確認している．[9]は 2 次元のセルオートマトンを対象として質量保存則が成り立つようにしたうえで，

非同期にセルを更新した際の振る舞いを確認している．これらの研究は，全てのセルの更新を同期的に一括更新するのではなく非同期に行う点で，本研究の取り組みと類似している．とくに，[7]で述べられている，空間領域をブロックに区切り，ブロック内のセル更新は同期的に実行する構成は本研究の仮定と同様である．しかし[7]では，本研究が想定するような，各ブロックがそれぞれ固有の計算周期をもって独立に計算を進めるという条件下における振る舞いは示されていない．

3. 熱伝導方程式を対象とした考察

本稿では以降，シミュレーション空間の区分をブロックと呼ぶ．各ブロックは共通のサイズの格子配列によって成り立っている．ブロック内の格子は同期的に計算・更新されるが,各ブロックはそれぞれ固有の時間周期によって計算ステップを繰り返す状況を考える．

まず単純のために式（1）で示す空間 1 次元の熱伝導方程式のシミュレーションを考察する．

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{1}$$

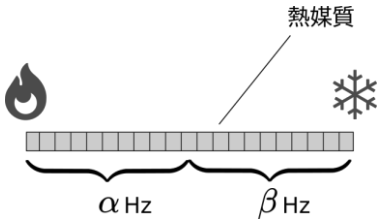


図 3：場所により計算間隔の異なる熱伝導シミュレーション

図 3 に示す通り，熱媒質を表す格子空間が 2 つのブロックによって半分に分かれて構成されており，それぞれのブロックが α Hz，β Hz の計算間隔を持つとする．また，媒質の左端を熱し，右端を冷却し続け，各格子には媒質の温度が格納されるものとする．

ここで，α が β よりも極端に大きい想定（$\alpha >> \beta$）で，初期状態が空間的に一様な温度から計算を開始した時に，計算時間の経過に伴って変化する解の形状の遷移を図 4 に示す．

まず，初期状態（図 4 の①）から計算開始後しばらくは左ブロックのみで頻繁に計算ス

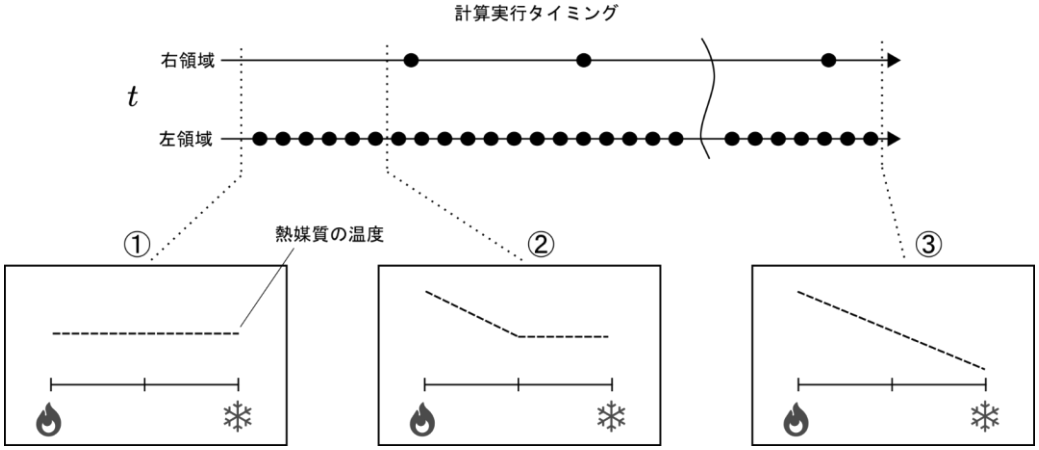


図 4：熱伝導シミュレーションの予測

テップが繰り返されるため，時間経過に伴って左端が熱され高温になる．$\alpha >> \beta$ を仮定しているのでこの時点では右ブロックの計算ステップがまだ実行されず，温度変化が生じない．左ブロックの立場では，この間，左ブロック内の右端境界は初期値で固定されているのと同等であるため，右端の温度は低いままである（図 4 の②）．ここから十分に時間が経過すると，右ブロックも計算ステップが繰り返され，結果として右ブロック内の右端は冷却されて温度が下がる．同時に，右ブロック内の左端境界は左ブロックから流入してくる熱の影響を受けて温度が上がる．また，十分長い時間尺度で捉えると，左ブロックの立場からみて左ブロック内の右端境界は自由境界と似た状況となり，左ブロック内の右端の温度も上がっていき，最終的には図 4 の③で示す平衡状態に落ち着く．図 4 は α が β よりも極端に大きいケースを想定したものであるが，α や β の値，また，空間の分割個数や幅にかかわらず十分時間が経過した後の結果は同様なものとなる．また，これは空間的に一様な計算周期を持つ一般的なシミュレーションを行った時の平衡状態でもある．したがって，少なくとも 1 次元熱伝導方程式の時間発展シミュレーションによって求まる平衡状態は，非一様な計算周期の影響を受けないといえる．また，空間の次元が 2 次元や 3 次元であっても，ある境界条件における熱伝導方程式系の平衡状態は一意に定まるため，時間が十分に経過した時の状態は非一様な計算周期の影響を受けないと思われる．

4. 波動方程式，反応拡散方程式を対象としたシミュレーション

前章で述べた熱伝導方程式は，系の平衡状態が計算周期に依らず一意に定まるため，計算周期が非一様であっても，一様である場合と比較して時間経過した後の状態に違いは無いことを述べた．これに対して，波動方程式系や反応拡散方程式系は，複数の解が存在するため，収束する解や，解に収束する過渡過程が，非一様な計算周期によってどのような影響を受けるのか予測できない．そこで，非一様な計算周期を持つ分散シミュレータ上でこれらの系を実際に駆動し，その挙動を確認した．

4.1. シミュレーション環境

図 5 に，実験を行った疑似分散シミュレータの概要を示す．各計算ノードは空間上の固有のブロックを保持し，それぞれが独立に計算ステップを逐次実行する．本来の分散シミ

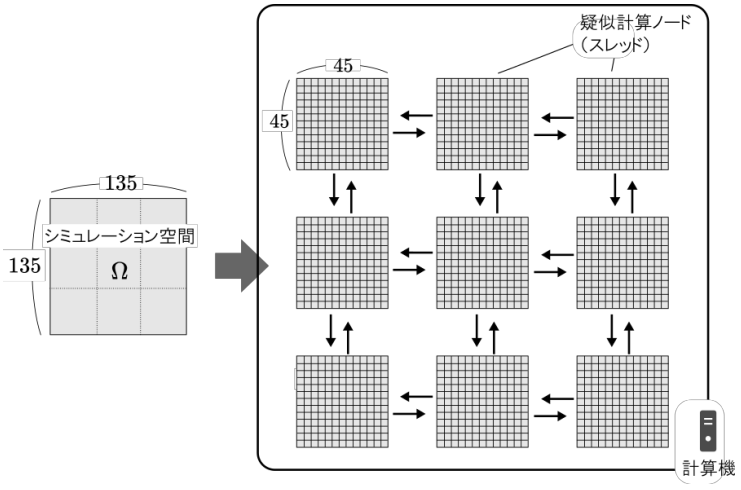


図 5：シミュレーション構成（図 1 のシステムを模した疑似環境）

ュレータの計算ノードは，図 1 に示したように別々の計算機によって動作し，ノード間の通信はネットワークを介して行うことを想定しているが，今回は計算周期の影響のみを確認対象としたため，各計算ノードは 1 つのスレッド（計算処理実行の単位）として実装した上で 1 台の計算機内で動作させた．また，ノード間の通信はメモリを介して行うことでネットワーク通信に要する時間の影響を排除した．

空間は 135×135 個の格子によって構成し，それらを格子状に繋いだ 9 つの計算ノード（スレッド）に割り当て，シミュレーションを並列実行した．各ノードは 45×45 の格子ブロックを保持させた．また，それらの更新周期を図 6 に示す通りのパターンで決定した．図 6 における黒いブロックは，白いブロックの計算ノードよりも計算周期を長くすることを示している．今回，黒いブロックは，白いブロックの 4 倍の計算周期とした．つまり，黒いブロックが 1 回計算ステップを実行する間に，白いブロックは平均して 4 回の計算ステップを実行することになる．なお，個々のスレッドは，本来は独立した計算機上で異なるプロセスとして動作することを想定しているため，スレッド間の同期は行っていない．そのため，計算ステップ実行の回数比率（1:4）はあくまで目安であり，スレッド処理の進捗状況に応じて若干増減する．また，各スレッドは計算ステップを実行するたびに，その時点のブロック境界における格子の状態値を近接スレッドへ通知する．以降，黒いブロックを遅延ブロック，白いブロックを通常ブロックと呼ぶ．

シミュレーションを実行した計算機の性能は CPU1.8GHz 4 コア（8 論理プロセッサ），メモリ 8GB である．シミュレーションのプロセスは Java にて実装した．演算に GPU は用いていない．

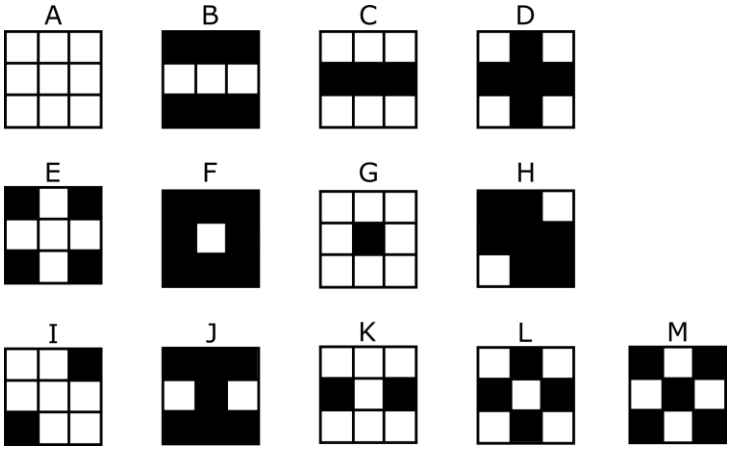


図 6：遅延ブロックの配置パターン

4.2. 波動方程式を対象としたシミュレーション

前項で述べたシミュレーション環境上で，式（2）に示す 2 次元の波動方程式の時間発展を計算した．シミュレーションの計算方式は差分法である．具体的には式（2）を離散化した式（3）によって$d = 10^{-7}$とおいて、格子を更新する計算ステップを逐次実行した．式（3）内の添え字i, jは格子の座標を，tは時間ステップを表している．

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} \tag{2}$$

$$u_{i,j}^{t+1} = 2u_{i,j}^{t} - u_{i,j}^{t-1} + d(u_{i-1,j}^{t} + u_{i+1,j}^{t} + u_{i,j-1}^{t} + u_{i,j+1}^{t} - 4u_{i,j}^{t}) \tag{3}$$

境界条件は周期境界条件とし，初期値として空間的に一様な値：0 を設定した上で，空間の中央に位置する格子に対して正弦波によって振動を与え続けてシミュレーションを駆動した．正弦波の周期は，そのブロックにおける計算ステップ周期$\times 10^3$として与えた．空間中央が通常ブロックの場合は，そこで与えられる正弦波の周期は，遅延ブロックの立場からすると約 1/4 となり，逆に空間中央が遅延ブロックの場合は，通常ブロックの立場からは約 4 倍となる．

結果と考察

図 7 にシミュレーションの結果を示す．各行に遅延ブロックの配置パターンを，各列にそれぞれのパターンにおける時間経過に伴う解の形状を描画したものである．解形状は 135×135 ピクセルの明度画像であり，各ピクセルの明度はuの値に応じて決めている．いずれの結果も，シミュレーション時間の経過に伴って最終的には定常波が生成された．定常波のパターンとしては，波源として正弦波を与えた空間の中央のブロックが遅延ブロックか通常ブロックかで分類できる．

空間中央が通常ブロックである（波源が通常ブロックにある）場合

中央で発生した波の波長が，遅延ブロックでは短くなることから，遅延ブロックでは通常ブロックと比較して細かな波が存在しており，それを明確に目視することができる（図 7 の B,E,F,I,K,L 行）．また，遅延ブロックで生じる波の形状は通常ブロックとの接し方で変化することが分かる．例えば図 7 のパターン B では遅延ブロックは通常ブロックと南北でしか接しておらず，波がそれに沿って細長く存在していることが確認できる（図 7 の B 行 32,64 列）．

空間中央が遅延ブロックである（波源が遅延ブロックにある）場合

中央で発生した波の波長は，通常ブロックでは長くなることから，通常ブロックでは波長の大きな波が生成される．また，この波の影響を逆に受けて遅延ブロックの波長も長くなり，十分な時間が経過すると，遅延ブロックと通常ブロックの目視による区別はできなくなる．

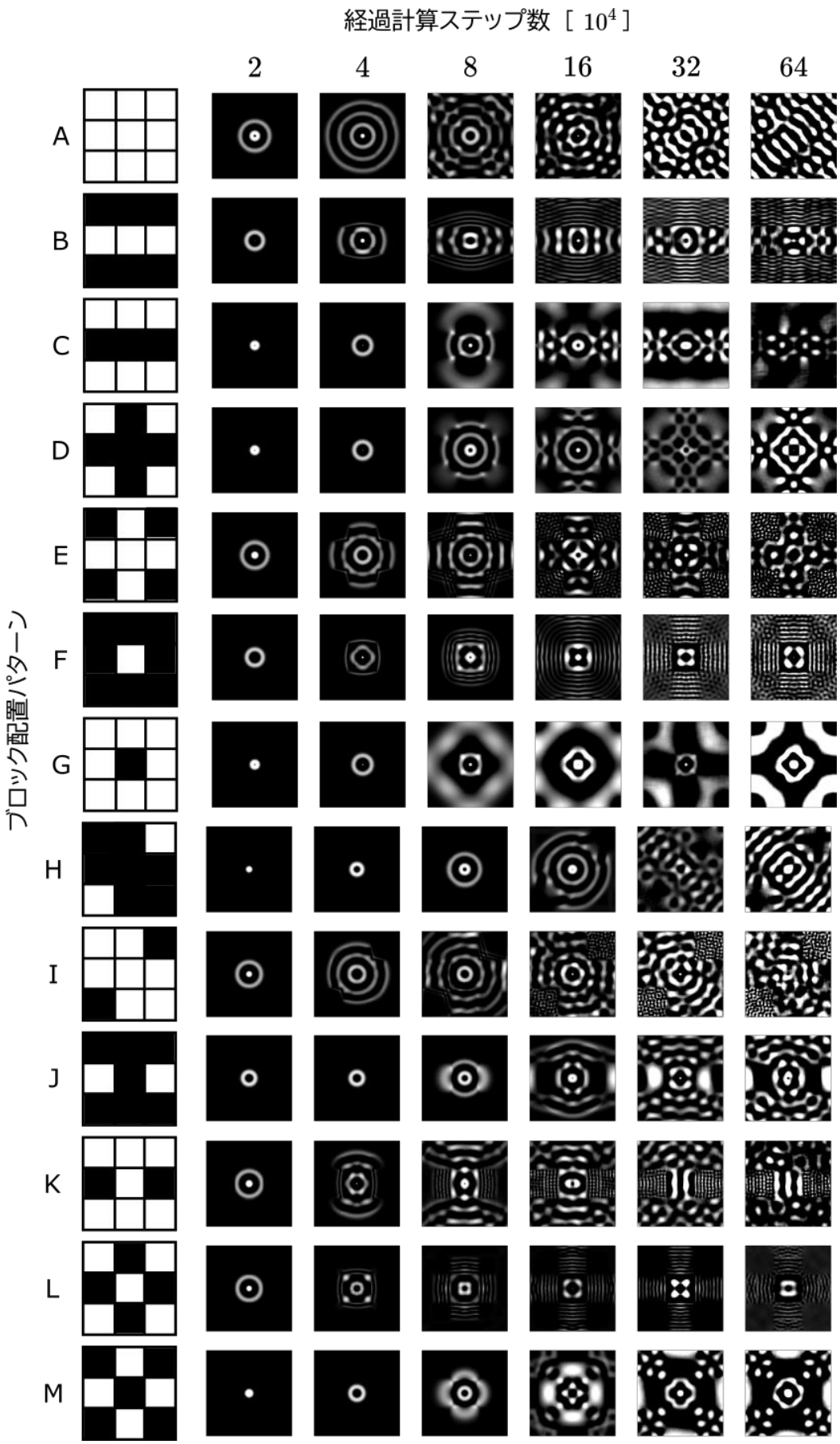


図 7：波動方程式のシミュレーション結果

4.3. 反応拡散方程式を対象としたシミュレーション

反応拡散系は，非一様な空間パターンを持つ平衡解が存在している[10]．今回は反応拡散系の中でも特に多様な解を持つことで知られている式（4）に示す 2 次元の GrayScott モデル[11,12]の時間発展を計算した．シミュレーションは前項同様に，差分法によって計算を行った．格子更新のための離散式は式（5）に示す通りであり，$F = 0.04, k = 0.059, d = 0.01, k = 0.01, D_u = 10^{-5}, D_v = 2 \times 10^{-5}$とおいた．

$$\begin{aligned} \frac{\partial u}{\partial t} &= u^2 v - (F + k)u + D_u \frac{\partial^2 u}{\partial x^2}, \\ \frac{\partial v}{\partial t} &= -u^2 v - (1 - v) + D_v \frac{\partial^2 v}{\partial x^2}. \end{aligned} \tag{4}$$

$$\begin{aligned} u_{i,j}^{t+1} = d\{&(u_{i,j}^t)^2 v_{i,j}^t - (F + k)u_{i,j}^t + \\ &\frac{D_u}{k^2}\left(u_{i-1,j}^t + u_{i+1,j}^t + u_{i,j-1}^t + u_{i,j+1}^t - 4u_{i,j}^t\right)\} + u_{i,j}^t, \\ v_{i,j}^{t+1} = d\{&-(u_{i,j}^t)^2 v_{i+1,j}^t + F\left(1 - v_{i+1,j}^t\right) + \\ &\frac{D_v}{k^2}\left(v_{i-1,j}^t + v_{i+1,j}^t + v_{i,j-1}^t + v_{i,j+1}^t - 4v_{i,j}^t\right)\} + v_{i,j}^t. \end{aligned} \tag{5}$$

境界条件は周期境界条件とし，空間の中央に位置する格子(i_c, j_c)を中心として生成したガウス関数$f(x, y) = 10 \times \exp\{-(x - i_c)^2 - (y - j_c)^2\}$を初期値として与えた．

結果と考察

図 8 にシミュレーションの結果を示す．前項同様に，各行に遅延ブロックの配置パターンを，各列にそれぞれのパターンにおける時間経過に伴う解の形状を描画したものである．また，ピクセルの明度はuの値に応じて決めている．

前項での波動方程式は，入力として正弦波を与え続けていたため，時間経過とともに変化しなくなる平衡解は存在しなかったが，ここで対象とした系には平衡解が存在し，計算ステップを十分に重ねると，それ以上時間経過しても変化しなくなる状態が得られる．また，前章で述べた単純な熱伝導方程式は，平衡解として取りえる状態は 1 つだけであったため，非一様な計算周期であっても，十分に時間が経過した後の結果に変わりはなかった．しかし，ここで対象とした系は，平衡解が多様に存在しており，非一様な計算周期の影響を受けて，時間経過に伴って到達する状態も変化している様子がわかる．

解のパターンとしては，前項の波動方程式を対象としたシミュレーション結果のように，遅延ブロックと通常ブロックの間で解の形状の波長に大きな違いは出ることはなかった．しかし，計算の過程で，通常ブロックでパターン形成が速く進行するため，まず通常ブロックにおいてパターンが生成され，その後に遅延ブロックでのパターン生成が，空白領域を埋めるように進行する様子が観察できた．このとき，空白領域の形状に沿ってパターン生成がなされるため，系がどのような過程を辿ってその平衡状態に到達したのか，いわば過渡過程の形跡が確認できた．反応拡散系が生成する空間パターンは，領域に対する適応性・整列性を有していることが知られており[13,14]，この形跡はこの性質によるものであると考えている．

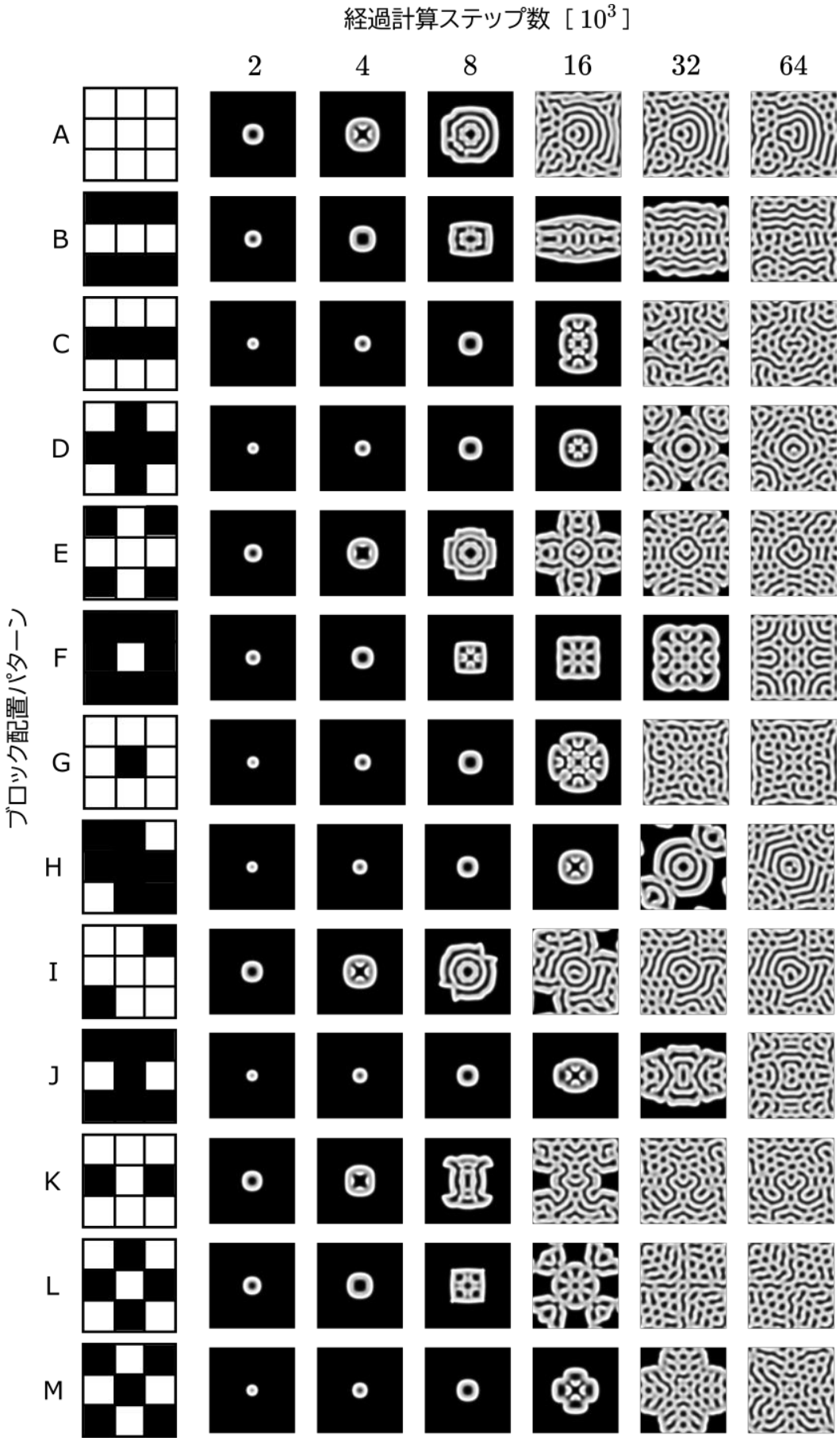


図 8：反応拡散方程式のシミュレーション結果

まとめ

空間的に非一様な計算周期を持つシミュレーションに関する考察と実験を行った．熱伝導方程式については，非一様な計算周期の影響を受けないことを考察した．波動方程式については，シミュレーションの結果，非一様な計算周期によって得られる解の定常波のパターンが大きく異なった．また，その定性的な性質は波源が通常ブロックか遅延ブロックのどちらに位置するかで大まかな分けることができた．反応拡散方程式については，得られる空間パターンに定性的な違いは生じなかった．しかし，そのパターンに到達するまでの過程の形跡を見ることができた．

参考文献

[1] Guan, Q.: pRPL: an open-source general-purpose parallel Raster Processing programming Library, SIGSPATIAL Special, Vol. 1, pp. 57-62 (2009).

[2] 建部 修見, 関口智嗣. (1998). 共有メモリ計算機における局所同期機構, 情報処理学会研究報告(HPC)

[3] 夏 澄彦, 佐藤 芳樹, 千葉 滋. (2015), 軽量で抽象度の高い条件付きバリア同期とその実装方法. 情報処理学会論文誌プログラミング（PRO）

[4] Hideo Ikeda, Shin-Ichiro Ei, Front dynamics in heterogeneous diffusive media, Physica D: Nonlinear Phenomena, Volume 239, Issue 17, 2010, Pages 1637-1649

[5] 重定 南奈子, 侵入と伝播の数理生態学, 東京大学出版会, 1992

[6] Fisher, R.A. (1937) The Wave of Advance of Advantageous Genes. Annals of Eugencies, 7, 355-369

[7] Sipper, Moshe & Tomassini, Marco & Capcarrere, M.. (1998). Evolving Asynchronous and Scalable Non-uniform Cellular Automata.

[8] Kanada, Yasusi. (1997). The Effects of Randomness in Asynchronous 1D Cellular Automata.

[9] Suzudo, Tomoaki. (2004). 非同期セルオートマトンの創発的パターン形成. 電子情報通信学会 非線形問題研究会

[10] A. M. Turing, The Chemical Basis of Morphogenesis, Philosophical Transactions of the Royal Society, 1952.

[11] P. Gray and S.K. Scott, Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Isolas and other forms of multistability, Chem. Eng. Sci. Vol.38 (1983), 29-43.

[12] P. Gray and S.K. Scott, Autocatalytic reactions in the isothermal, continuous stirred tank reactor: oscillations and instabilities in the system A+2B → 3B → C, Chem. Eng. Sci.

[13] H. Notsu, D. Ueyamab and M. Yamaguchi, Applied Mathematics Letters 26, 201 (2013).

[14] 鈴木 洋平,高山 貴裕,元池 育子,浅井 哲也. (2005). しま・斑点画像パターンの修復を行う反応拡散モデルとその LSI 化,電子情報通信学会論文誌 A 基礎・境界

立体映像視認時における視線軌跡パターンと重心動揺パターンの関係性

小野蓮太郎，平田隆幸，高田宗樹*
福井大学大学院工学研究科　福井県福井市文京 3-9-1
* takada@u-fukui.ac.jp

Relationship between Patterns of the Eye Movement and the Body Sway While Viewing a 3D Video Clips

Rentaro Ono, Takayuki Hirata, Hiroki Takada
Graduate School of Engineering, University of Fukui 3-9-1, Bunkyo, Fukui

（2021 年 1 月 14 日受付，2021 年 1 月 25 日受理）

Abstract: In recent years, the development of video technology has increased the chance of viewing stereoscopic films on media such as televisions and movie theaters. As the viewing of 3D video has become more commonplace, symptoms such as visually induced motion sickness (VIMS) and eye fatigue have been reported. In a previous study, post peripheral tracking was shown to influence the equilibrium function compared to post tracking of a 3D video clip. However, this result may depend on the experimental environment. In this study, we investigate the relationship between patterns of the eye movement and the body sway while viewing a 3D video clips. Our results show that post peripheral viewing increases gain. This suggests that post peripheral viewing tends to accumulate fatigue.

Keywords: 3D video clips, Visually induced motion sickness (VIMS), Eye movement, Body sway pattern, Transfer function analysis

1. 本論文の目的

近年、3DCG 技術の飛躍的な進歩により、立体映像が身近なものになるとともに、ヒトがこれまでに経験したことのない視覚刺激を受けることが出来る時代になった。2010 年は 3D 元年と呼ばれ、3D 映像製品が市場に広まり始めた。また、新たに 2016 年は VR 元年と呼ばれ、臨場感の高い映像が展開されつつある。このような製品の普及は、娯楽分野にとどまらず、医療や教育、福祉など多岐にわたる[1][2]。その一方で、眼疲労や嘔吐感、酔いなどの映像酔いと呼ばれる不快な症状が引き起こされるなど、健康上の悪影響が懸念されている[3]。そこで、様々な生体信号から映像酔いを評価する研究がなされている[4][5]。

映像酔いについて生理指標を用いた定量的な評価方法の一つとして、重心動揺検査が挙げられる[6]。重心動揺検査では、被験者が測定器の上に立位した状態の重心位置を記録する。この方法は、立体映像視認時の映像による体平衡系への影響を定量的に評価するのに有効である。先行研究においては、映像の視認方法に注目されており、追従視に

比べ周辺視の方が体平衡系に影響を及ぼすことが示唆された[7]。しかし、実験環境に依存してその傾向がみられないこともあり、映像酔いの機序の詳細を解明することが求められている。

ヒトの体平衡は、様々な感覚情報の統合によって保持されている。主に、①前庭器から得られる加速度情報、②視覚情報、③体性感覚情報の 3 系統である。これらの入力情報が、反射系や高次脳での情報処理を経て統合されて、出力系へもたらされる[8]-[11]。Edwards の報告によると、姿勢制御に関する情報のうち、視覚情報は 50%程度を占めると報告している[12]。また、視覚情報が欠落した状態での姿勢安定性は、視覚情報の入力がある場合と比べ、20-50%程度低下することが知られている[13][14]。そのため、出力系としての姿勢維持制御は視覚入力と大きく関わっている。

視覚情報により引き起こされる姿勢制御は、視覚誘導性姿勢変化(Visual evoked postural responses：VEPRs)と呼ばれる。先行研究では、VEPRs を測定することで臨場感および映像酔いのどちらの評価にも用いられている[15]-[18]。また、映像の速度や前景と背景の関係性、周辺視と中心視の関係性などの項目について検証実験を行った先行研究では、自己運動の方向や視認方法に関わらず、背景刺激により VEPRs は発現すると結論付けられている[19]-[21]。しかし、VEPRs の発生メカニズムは不明瞭な点が多く、現象の解明には至っていない。さらに、VEPRs は視覚情報に起因する姿勢変化であるが、視覚情報を運動器官へ伝達する際の、伝達強度について詳細に研究を行った例は見当たらない。そのため、視覚情報と姿勢制御の伝達強度が、因子によってどのように変化するかを検討することが期待されている。

体平衡系に関して、形の科学からのアプローチは、新しい可能性をもっている。これまでのところ、動揺図は軌跡長や包絡面積といった限定的な図形パターンの解析および高速フーリエ変換による周波数解析によって分析されており[22][23]、単純なブラウン運動として記述されることが多い[24]。複雑な図形パターンを有し、情報量が大きいことが期待される反面、動揺図のみでの解析では、平衡機能の評価には限界がある。そこで、体平衡の保持に重要な視線軌跡パターン(図 1)を動揺パターン(図 2)と照らし合わせて評価することにより、めまい・平衡機能障害の検査、リハビリテーション医学など多分野において寄与することが期待される。また、未だ解明されていない酔いの評価にも応用できるなど、体平衡系に関する研究の発展には、形の科学からのアプローチが重要である。そこで本研究では、視覚入力と出力としての姿勢維持制御の関係に着目し、立体映像視認時の視線運動と重心動揺を同時計測することで、実験環境に依存しない映像酔いの評価法を提案するとともに、映像酔いの発生機序を解明することを目的とする。

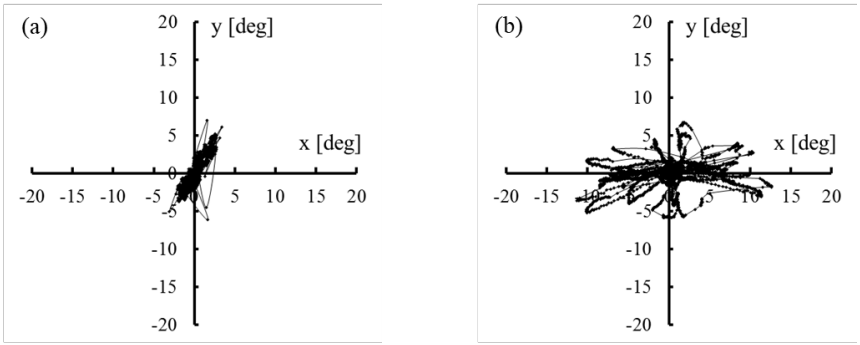


図 1：立体映像視認時における視線軌跡パターンの例、(a) 周辺視認、(b) 追従視認

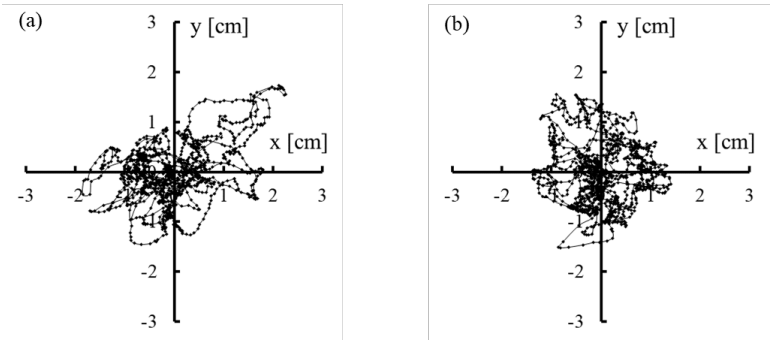


図 2：立体映像視認時における重心動揺パターンの例、(a) 周辺視認、(b) 追従視認

2. 実験

2.1. 実験方法

被験者は、耳・神経系疾患既往歴のない若年男女 11 名(平均年齢±標準偏差:22.7 ± 0.8 歳)を対象として計測を行った。被験者には事前に実験の説明を十分に行った上で、書面にて同意を得た。また、本研究は、福井大学大学院工学研究科知能システム工学専攻人を対象とする研究倫理委員会の承認を受けて実施した(承認番号：H2019003)。

本研究では、被験者から 2 m 前方で目の高さの位置が中心になるように設置された 55 インチの 3D ディスプレイ 55UF8500(LG)に映像を表示した。実験に用いる立体映像は、Sky Crystal(オリンパスメモリーワークス)をもとに同社の許可を得て再構成したものを用いた(図 3)。Sky Crystal は、球体を呈示する動画であり、左右および上下に変位を伴いながら 5 秒周期で近位と遠位を準周期的に往復する。提示した映像の球体のサイズは、近方時が 9.4 [deg]、遠方時が 0.3 [deg]である。球体を遠位と近位の間で滑らかに運動させて呈示することにより、毛様体筋の伸縮運動をさせて、眼疲労を緩和する他、仮性近視に対して一定の効果がある。本研究では、映像の立体性 2D/3D が異なる映像を、それぞれ周辺視認または追従視認させた。周辺視認時では、被験者には対象物を注視せず、画面全体を眺めるよう指示した。また、追従視認時では、被験者には対象物を注視するよう指示した。計測は暗室で行い、実験姿勢は立位ロンベルグ姿勢とした。重心動揺計測にはバランス Wii ボード(Nintendo)を用い、映像視認時の各サンプリング時間での重心の位置を記録した。サンプリング周波数は、100 Hz で計測した。視線運動計測には視野カメラが取り付けられたアイマークレコーダ EMR-9 (ナックイメージテクノロジー)を用い、映像視認時の各サンプリング時間での視線の位置を記録した。サンプリング周波数は、60 Hz で計測した。

計測中、被験者には立位ロンベルグ姿勢をとらせた。立位安静の後、上述の映像をそれぞれ周辺視または追従視で、各 60 秒間視認させ、この間の重心動揺と視線運動を同時計測した。視認する映像と視認方法の順序効果を考慮し、実験は任意の順番で行った。なお、映像視認時に強い酔いを感じた場合は実験を中止できるものとした。また、計測中に過度のふらつきやロンベルグ姿勢を保てなくなった場合などは実験を止め、一定時間の休憩をはさみ、再度計測を行った。

図 3：実験で使用した映像

2.2. 伝達関数解析

映像視認時の各サンプリング時間での、視線運動と重心動揺の x - y 座標を記録した。得られた視線運動のデータは、x 方向（右方向を正)、y 方向（上方向を正）とする視線位置の時系列に変換し、重心動揺のデータは x 方向（右方向を正)、y 方向（前方向を正）とする重心位置の時系列に変換した。さらに、視線運動と重心動揺の各成分の時系列についてスペクトル解析を行った。これらのクロススペクトルを求めた上で、視線運動を入力、重心動揺を出力とした生体システムを仮定して、伝達関数解析によりコヒーレンスとゲインを算出した。コヒーレンス $Coh(f)$は以下のように定義した。

$$Coh(f) = \frac{\left|\mathrm{C_{xy}}(f)\right|^2}{C_{xx}(f)C_{yy}(f)} \tag{1}$$

ここで、$\mathrm{C_{xy}}(f)$は、視線運動と重心動揺の信号間のクロススペクトル、$C_{xx}(f)$は、視線運動のパワースペクトル、$C_{yy}(f)$は、重心動揺のパワースペクトルである。コヒーレンスは、当該周波数帯において、値が 1 に近づくにつれ視線運動と重心動揺の関係が一次関数的な高い関連性を持つ関係に近づくことを意味する。また、コヒーレンスの値が 0 に近づくことは、評価の上では、低い関連性を指すことになる。尚、本研究では相関係数検定表(r 表)に基づき、$Coh(f) > 0.06$ の場合に、両者に関連性が存在するとみなし、その際に伝達関数ゲインの評価を行った。

一方、視線運動と重心動揺の伝達関数ゲイン $Gain(f)$は、次のように定義した。

$$Gain(f) = \frac{\mathrm{C_{xy}}(f)}{C_{xx}(f)} \tag{2}$$

ここで、$\mathrm{C_{xy}}(f)$は、視線運動と重心動揺の信号間のクロススペクトル、$C_{xx}(f)$は、視線運動のパワースペクトルである。伝達関数は、視線運動を入力、重心動揺を出力と考えた場合の当該周波数帯の振幅関係を示す解析である。ゲインは、値が高いほど、変動入力に対し、変動出力が大きいことを意味する。また、ゲインの値について、母平均に差がないことを帰無仮説とする Wilcoxon の符号順位和検定を用いて比較検討を行った。尚、本研究では有意水準を $p<0.05$ とした。

3. 結果

3.1. コヒーレンス

2D 映像、3D 映像視認時における重心動揺と視線運動のコヒーレンスを算出した(図 4-5)。2D 映像では、周辺視認時の 0.16 Hz 付近で相関がみられなかった。その他の周波数帯では、相関がみられた($p<0.05$)ため、相関がみられた帯域をゲインの評価対象とした。また、3D 映像では、周辺視/追従視において、すべての周波数帯域で相関がみられた($p<0.05$)。3D 映像視認時では、いずれの例においてもゲインの評価対象となった。

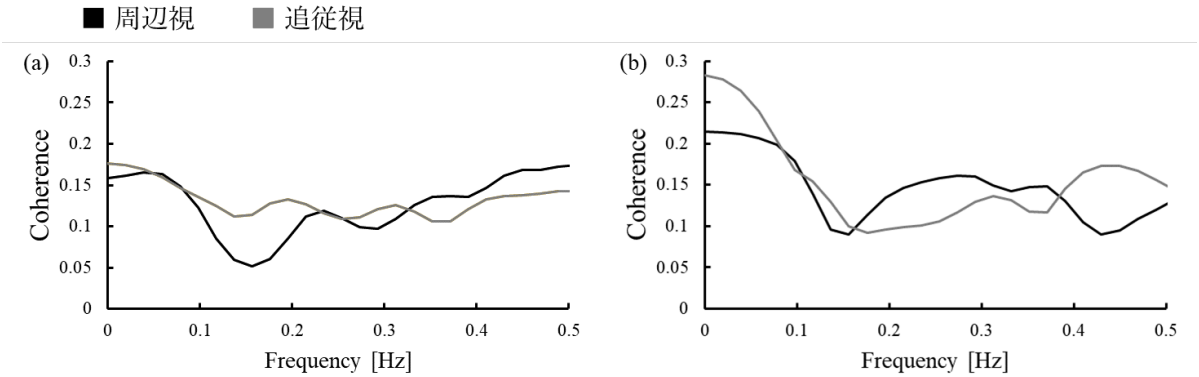


図 4：2D 映像視認時における重心動揺と視線運動のコヒーレンス、
(a) x 軸方向、(b) y 軸方向

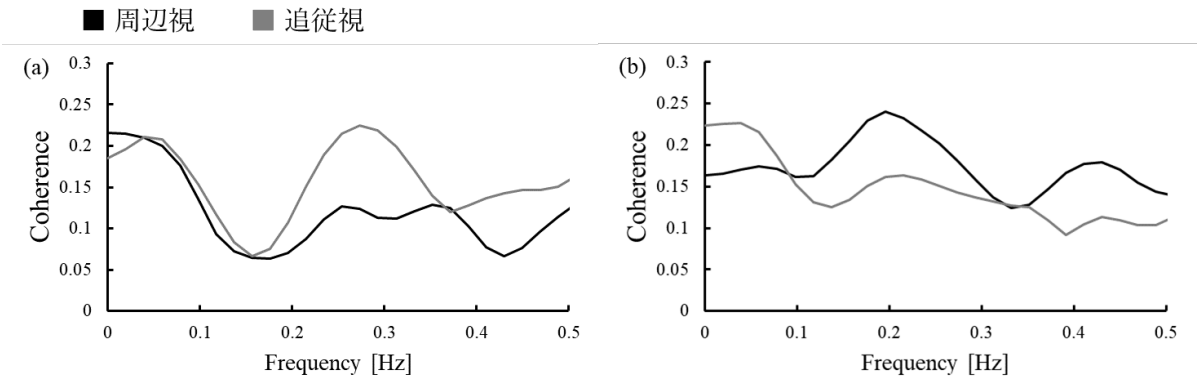


図 5：3D 映像視認時における重心動揺と視線運動のコヒーレンス、
(a) x 軸方向、(b) y 軸方向

3.2. ゲイン

2D 映像および 3D 映像視認時における視線運動と重心動揺の伝達関数解析から算出したゲインを比較した(図 6-9)。映像視認時における x 軸方向の比較では、視認方法間で有意差はみられなかった。2D 映像視認時における y 軸方向の比較では追従視に比べ、周辺視で 0.2-0.4 Hz 付近のゲインが有意に増大した($p<0.05$)。一方、3D 映像視認時における y 軸方向の比較では追従視に比べ、周辺視で 0.15-0.2 Hz 付近のゲインが有意に増大した($p<0.05$)。

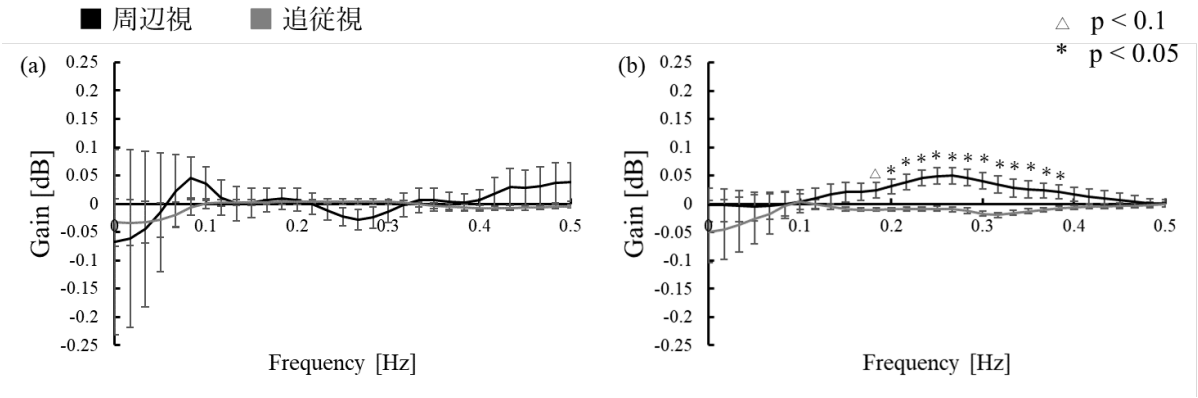


図 6：2D 映像視認時における重心動揺と視線運動の伝達関数解析から算出したゲインの視認方法間比較、(a) x 軸方向、(b) y 軸方向

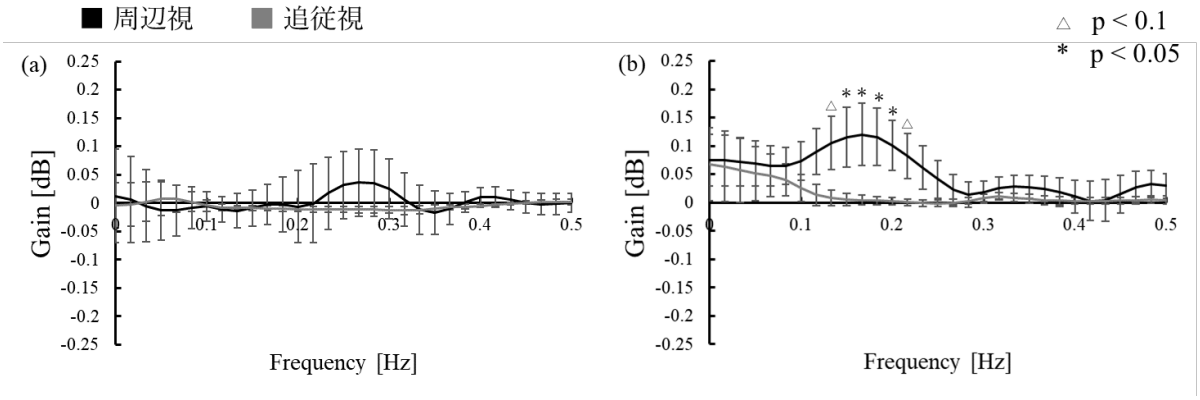


図 7：3D 映像視認時における重心動揺と視線運動の伝達関数解析から算出したゲインの視認方法間比較、(a) x 軸方向、(b) y 軸方向

次に、視認方法ごとに分けて映像間比較を行った。x 軸方向の比較において 2D 映像周辺視認時に比べ、3D 映像周辺視認時の 0.1-0.2 Hz 付近のゲインが有意に増大する傾向がみられた($p<0.10$)。また、周辺視認時における y 軸方向の比較では、映像間で有意な差はみられなかった。一方、追従視認時では x 軸方向、y 軸方向ともに、映像間で有意な差はみられなかった。

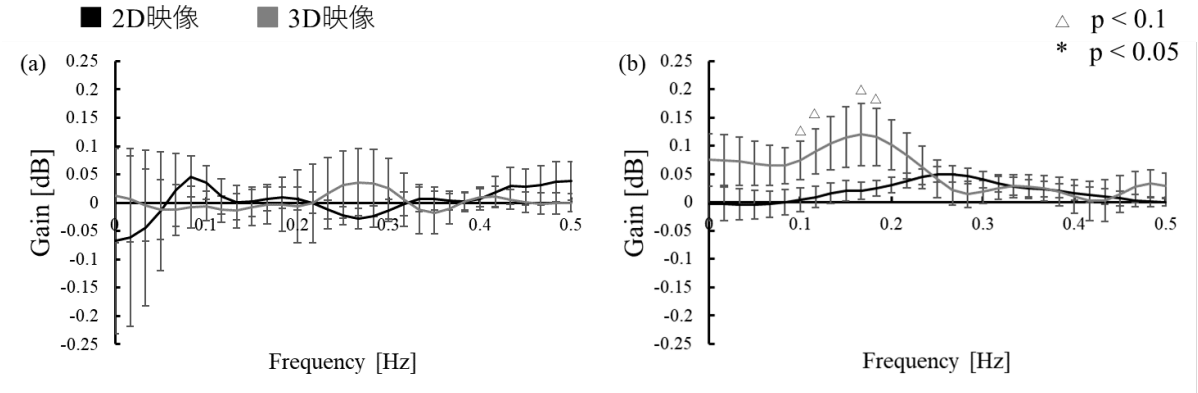


図 8：周辺視認時における重心動揺と視線運動の伝達関数解析から算出したゲインの映像間比較、(a) x 軸方向、(b) y 軸方向

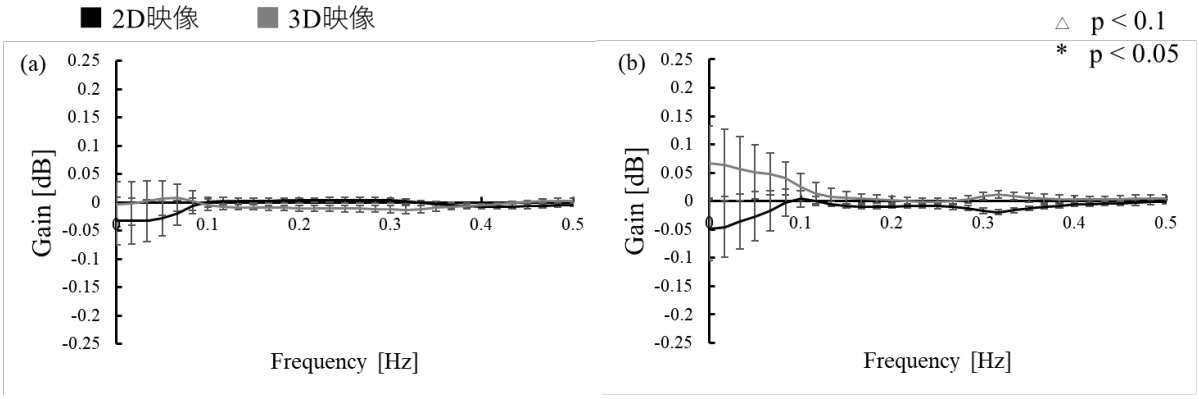


図 9：追従視認時における重心動揺と視線運動の伝達関数解析から算出したゲインの映像間比較、(a) x 軸方向、(b) y 軸方向

4. 考察

本研究では、立体映像視認時における視線運動と重心動揺を同時計測し、視覚と姿勢維持制御の関係性を明らかとすることで、映像酔いの機序の詳細を解明することを目的とし、実験を行った。その結果、両映像視認時ともに y 軸方向において、追従視に比べ周辺視のゲインが増大することが示された(図 6b、7b)。これは周辺視時において、視線運動(入力)の増大に対する重心動揺(出力)の増大の度合いが大きいことを示している。

視覚情報は、大脳皮質や中脳などの中枢神経系で統合・処理され、体平衡へと伝達されるため、周辺視では脳での過剰な処理が起きている可能性が示唆される。これが脳の負担となり、体平衡への情報処理・伝達が不十分となる。そのため、映像酔いを引き起こしやすいと考えられる。一方、視認方法に依存せずに背景映像は時間変化しているため、VEPRs は発現しているものと考えられる。しかし、本研究の伝達関数解析の結果、周辺視に比べ追従視のゲインが抑制される可能性が示唆された(図 8 、9)。これは、追従視に伴い中心視覚路の情報処理が優位になり、背景映像の認知が妨げられて、VEPRs が抑制されたと推察される。

映像間の比較を行ったところ、追従視では立体性の違いによらず、ゲインの増幅が起こらなかった(図 9)。追従視は周辺視と異なり、球体を視線に追従させる視認方法であり、球体が視標となることで、視覚情報から鉛直方向や水平方向の手がかりを得ることが可能となる。これにより、ヒトが姿勢を保持するために必要な姿勢反射が正常に行われている。コヒーレンス解析の結果から、0.15-0.4 Hz の周波数では、視覚が情報を受け取ると姿勢制御が行われることが考えられるが、姿勢反射の過程では、視線運動と重心動揺の振幅が同程度であるため、ゲインの増幅が起こらなかった。また、周辺視時では、2D 映像に比べ 3D 映像のゲインの値が増大することが示された(図 8)。重心動揺検査は、映像酔いの評価法として用いられており、映像酔い発症時、動揺量が増大することが知られている[25]。上述したゲインの増大は、動揺量の増大を示しており、3D 映像視認時にゲインの値が有意に増大する傾向がみられたということは、3D 映像によって映像酔いが引き起こされたといえる。そのため、ゲインの増幅率が高いことにより、酔い及びその度合いを評価することができる。一方、映像の立体性に依存せずに背景映像は時間変化

しているため、VEPRs は発現しているものと考えられる。しかし、3D 映像は奥行き情報を有しており、背景映像から空間認知を補間する効果がある。よって、3D 映像視認時において VEPRs が強く発現し、動揺量が増大したと推察される(図 8)。

四つの計測パターンのうち、3D/周辺視時に最もゲインの増幅率が高いことが示された。3D/周辺視の組み合わせは映像酔いを引き起こしやすいことが知られており[7][26]、映像酔い発症時にゲインが増大するものと考えられる。また、上述のように VEPRs を強く発現させる組み合わせであると推察できる。VEPRs が、視覚と平衡感覚の感覚不一致を予防する目的で発現していると考えると、ゲインの増大は矛盾を解消するための自衛反応だと考えることができる。そのため、視線運動と重心動揺をそれぞれ入出力としたときの伝達関数解析によるゲインの算出が、映像酔いの新たな評価法となる可能性が示唆された。

映像の立体性に依存せずに、周辺視時に 0.15-0.4 Hz の周波数帯におけるゲインが増大した(図 6b、7b)。心拍変動解析においては、呼吸周期の変動を表す高周波成分（0.15-0.40 Hz：HF）と血圧変動を反映する低周波成分（0.04-0.15 Hz：LF）にピークが現れ、この両者は自律神経活動を反映するとされる[27]。特に、HF は副交感神経活動を、LF は交感神経活動と副交感神経活動の両者に依存する。なお、前庭系と自律神経系は、解剖学的にも電気生理学的にも密接な関係があることが知られている[28][29]。また、視覚系と自律神経系は密接であり、これは瞳孔が副交感神経と交感神経の 2 重支配を受けていることから明らかである。周辺視の場合、HF 成分の周波数帯におけるゲインが増大しており(図 6b、7b)、副交感神経活動もゲインと連動して亢進していることがうかがえる。また、3D 映像の周辺視認時では、0.1-0.2 Hz の LF 成分と HF 成分の周波数帯にまたがって、ゲインが増大することが示された(図 8)。副交感神経と交感神経活動を反映する LF が増大したのは 3D 映像視認時のみであり、2D 映像視認時ではみられなかった。以上より、3D 映像視認時において、2D 映像視認時よりも交感神経活動が亢進することで、瞳孔が開散し、被写界深度が低下することにより映像ボケが生じて、ゲインの増大や映像酔いにつながることが考えられる。

5. まとめ

本研究では、視覚入力と出力としての姿勢維持制御の関係に着目し、立体映像視認時の視線運動と重心動揺を同時計測することで、映像酔いの機序の詳細を解明することを目的とし、実験を行った。その結果、視線運動と重心動揺を入出力としたゲインの値は、映像間の違いあるいは、視認方法間の違いにより変化がみられた。

コヒーレンス解析の結果から、3D 映像視認時では、いずれの例においてもゲインの評価対象となった。次に、伝達関数解析の結果から、2D 映像に比べ 3D 映像視認時にゲインが増大することが示された。また、追従視に比べ周辺視のゲインが増大することが示された。周辺視の場合、HF 成分の周波数帯におけるゲインが増大しており、副交感神経活動もゲインと連動して亢進していることがうかがえる。3D/周辺視の組み合わせは、映像酔いを引き起こしやすいことが知られており、映像酔い発症時にゲインが増大するものと推察される。本研究では、VEPRs の発現とゲインの増大には関連があることが示唆された。視覚と平衡感覚の感覚不一致に対する身体的感度が高い場合に VEPRs が強く

発現すると考えると、ゲインの増大は矛盾を解消するための自衛反応であり、ゲインを評価することで酔いの程度を判断できると推察される。

謝辞

本研究は、立石科学技術振興財団 研究助成(A)、栢森情報科学振興財団 研究助成、JSPS 科研費 18K11417, 20K12528 の助成を受けたものです。

参考文献

[1] 瀬戸崎典夫, 森田祐介, 竹田仰, “ニーズ調査に基づいた多視点型 VR 教材の開発と授業実践”, 日本バーチャルリアリティ学会論文誌, 11(4), pp.537-543, 2006.

[2] 久保田善彦, 山下淳, 奥村信太郎 他, “太陽系シミュレーションを利用した月の満ち欠け学習の実践と効果”, 科学教育研究, 31(4), pp.248-256, 2007.

[3] 濱岸五郎, “3D ディスプレイにおける人間工学的評価と標準化”, 映像情報メディア学会技術報告, 33(16), pp.9-12, 2009.

[4] 後藤玲子, 工藤博章, 佐藤耕平 他, “自己運動感覚を誘導する視覚刺激に対する生体反応の分析”, 映像情報メディア学会, 60, pp.589-596, 2006.

[5] 野村恵理, 木竜徹, 飯島淳彦 他, “生体信号から推定した映像酔いとそのきっかけとなった映像の動きベクトルの特徴”, 電子情報通信学会, 89, pp.576-583, 2006.

[6] 鈴木淳一, 松本喬, 徳増厚二 他, “重心動揺検査の Q&A 手引き”, Equilibrium Research, 55(1), pp.64-77, 1996.

[7] M. Takada, Y. Fukui, Y. Matsuura et al., “Peripheral viewing during exposure to a 2D/3D video clip effects on the human body”, Environ Health Prev. Med., 20(2), pp.79-89, 2015.

[8] M. Kouzaki, K. Masani, “Reduced postural sway during quiet standing by light touch is due to finger tactile feedback but not mechanical support”, Exp. Brain Res., 188, pp.53-158, 2008.

[9] H.C. Diener, J. Dichgans, W. Bruzek et al., “Stabilization of human posture during induced oscillations of the body”, Exp. Brain Res., 45, pp.126-132, 1982.

[10] M. Kouzaki, K. Masani, H. Akima et al., “Effects of 20-day bed rest with and without strength training on postural sway during quiet standing”, Acta. Physiol., 189, pp.279-292, 2007.

[11] R.C. Fitzpatrick, R.B. Gorman, D. Burke et al., “Postural proprioceptive reflexes in standing human subjects: bandwidth of response and transmission characteristics”, J. Physiol., 458, pp.69-83, 1992.

[12] A.S. Edwards, “Body sway and vision”, J. Exp. Psychol., 36, pp.526-535, 1946.

[13] W.M. Paulus, A. Straube, T. Brandt, “Visual stabilization of posture. Physiological stimulus characteristics and clinical aspects”, Brain, 107, pp.1143-1163, 1984.

[14] H.C. Diener, J. Dichgans, M. Bacher et al., “Quantification of postural sway in normals

and patients with cerebellar diseases", Electroencephalogr. Clin. Neurophysiol., 57, pp.134-142, 1984.

[15] M. Ohmi, "Sensation of self-motion induced by real-world stimuli", Proc. Int. Work. Adv. Res. Vis. Cogn. Sel. Integr. Vis. Inf, pp.175-181, 1998.

[16] J. Freeman, S.E. Avons, R. Meddis et al., "Using Behavioral Realism to Estimate Presence: A Study of the Utility of Postural Responses to Motion Stimuli", Presence Teleoperators and Virtual Environments, 9, pp.149-164, 2000.

[17] L.J. Smart, T.A. Stoffregen, B.G. Bardy, "Visually induced motion sickness predicted by postural instability", Hum. Factors., 44, pp.451-465, 2002.

[18] S.J. Villard, M.B. Flanagan, G.M. Albanese et al., "Postural instability and motion sickness in a virtual moving room", Hum. Factors., 50, pp.332-345, 2008.

[19] K.H. Britten, "Mechanisms of self-motion perception", Annu. Rev. Neurosci, 31, pp.389-410, 2008.

[20] M. Lappe, F. Bremmer, A.V. van den Berg, "Perception of self-motion from visual flow", Trends Cogn. Sci, 3, pp.329-336, 1999.

[21] 中村信次, "視覚誘導性自己運動知覚の実験心理学(初版)", 北大路書房, 京都, 2006.

[22] 時田喬, 宮田英雄, 青木光広, "重心動揺の周波数解析：ピーク面積一周波数スペクトル検査の提唱", Equilibrium Research, 72, pp.238-246, 2013.

[23] K. Stambolieva, "Fractal properties of postural sway during quiet stance with changed visual and proprioceptive inputs", J. Physiol. Sci., 61, pp.123-130, 2011.

[24] J.J. Collins, C.J. De Luca, "Random walking during quiet standing", Phys. Rev. Lett., 73, pp.764-767, 1994.

[25] L.M. Scibora, S. Villard, B.G. Bardy et al., "Wider stance reduces body sway and motion sickness", Proceedings of VIMS 2007, pp.18-23, 2007.

[26] 中川千鶴, 大須賀美恵子, 竹田仰, "VE 酔い評価手法の開発に向けての基礎的検討", 人間工学, 36(3), pp.131-138, 2000.

[27] David Robertson 著, 高橋昭・間野忠明監訳, "ロバートソン自律神経学", エルゼビア・ジャパン株式会社, 2007.

[28] N.H. Barmack, "Central vestibular system: vestibular nuclei and posterior cerebellum", Brain Research Bulletin, 60, pp.511-541, 2003.

[29] C.D. Balaban, J. D.Poster, "Neuroanatomic substrates for vestibuloautonomic interactions", J.Vestibular Research, 8, pp.7-16, 1998.

Colaboratory で形の科学を楽しもう

– 第 2 回 ソーシャルディスタンスを考える：Box へのランダムパッキング –

平田隆幸

福井大学　大学院工学研究科　知能システム工学専攻

〒910－8507　福井市文京 3 丁目 9－1

hirata@u-fukui.ac.jp

Let's Enjoy Science on Form by Colaboratoy

- About a Social Distance: Random Packing of Disks into the Square Box -

Takayuki Hirata

Department of Human & Artificial Intelligent Systems, Faculty of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan.

(2021 年 2 月 1 日受付, 2021 年 2 月 1 日受理)

1. はじめに

ソーシャルディスタンスについて考察しよう。コロナ禍の下、特効薬やワクチンが一般の人々に普及していない状況で、とりうる有効な戦略は、ソーシャルディスタンスを保つことである。そこで、ある部屋にソーシャルディスタンスを保ちながら何人が入ることができるかを考えよう（図 1 参照)。さらに単純化すると、一定の領域にソーシャルディスタンスを保ちながら、何人が入れるかを考えることになる。ここでは、Python を使って L×L の正方形のボックスに、互いに最低 r の距離を保ちながら何個の点を配置できるかを考えよう。

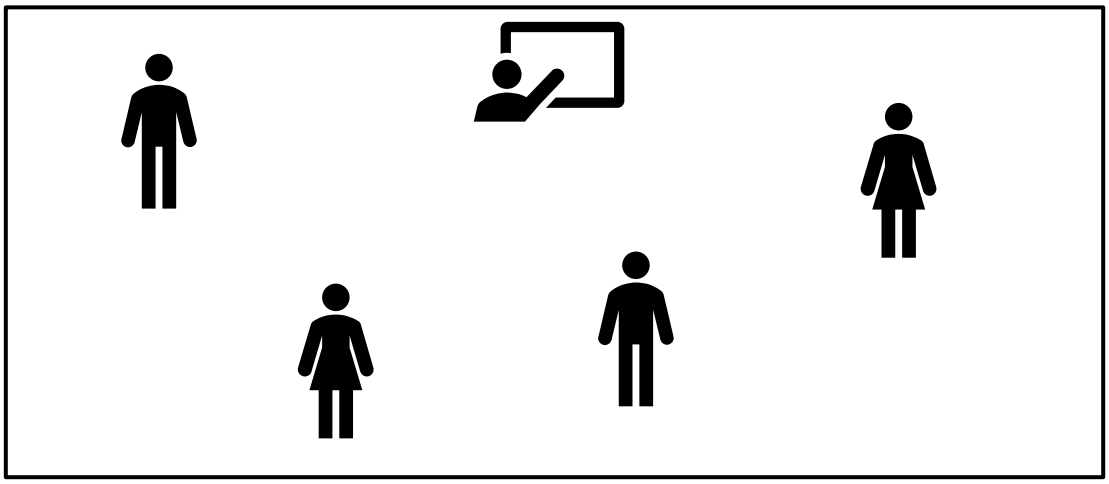


図 1：一定領域（例えば部屋）にソーシャルディスタンスを保ちながら人がいる様子。

ソーシャルディスタンスを保っている様子を明示するため、図 1 に直径 r の円盤を書き加えたものが図 2 である。図 2 を見ると、ソーシャルディスタンス保ちながら人を配置する問題は、円盤のランダムパッキングの問題に帰着されることが分かる。問題が整理できたので、Python を使ってランダムパッキングのシミュレーション実験をしてみよう。なお、Pyrhon のプログラミング環境としては、Google Colaboratory[1]を使う。

次に、グラフ作成など描画の仕方について述べる。Python の描画では、tkinter がよく使われるが、Colaboratory は tkinter をサポートしていない。それゆえ第 1 回講座では、Turtle Graphic を使って、動的なシミュレーションの表示を試みた。今回は、シミュレーションの結果をグラフなどで分かりやすく表示する Python のライブラリ matplotlib [2]を紹介する。matplotlib は、tkinter と比べると自由な描画は制限されるが、グラフなどを簡単に作れるメリットがある。なお、Colaboratory のホームページにアクセスすると出てくる「Colaboratory へようこそ」においても、データの可視化の方法として matplotlib が紹介されている。

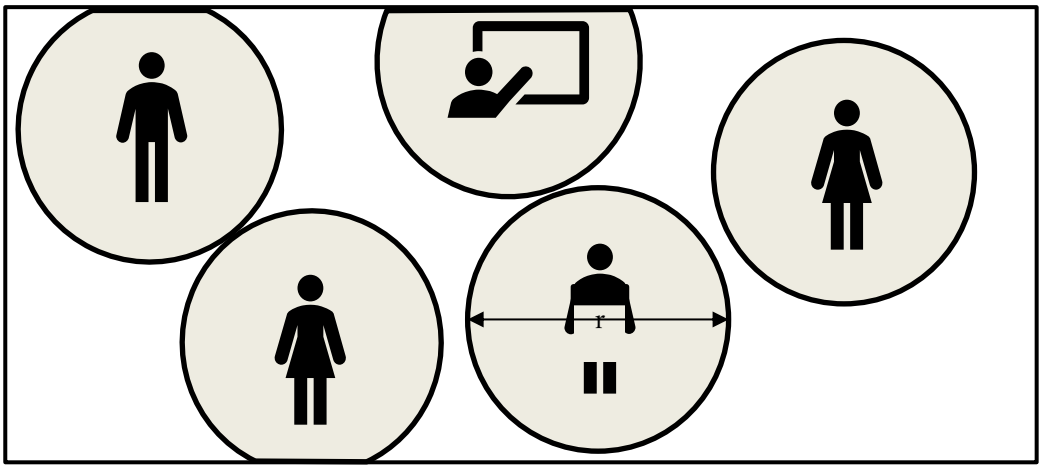


図 2:距離 r のソーシャルディスタンスを保っている様子を示すため人を中心に直径 r の円盤を描き足したもの。

2. Python でランダムパッキング

2.1. Colaboratory で Python プログラミング

第 1 回講義で、Colaboratory を使うと簡単に Python プログラミングができるのを紹介した。ここで、Colaboratory による Python のプログラミング環境について、第 1 回講義[3]を簡潔に振り返ろう。Web 上の Colaboratory にアクセスした様子と、コードセルへのプログラムの入力操作の様子を図 3 に示す（第 1 回講義の図を描き換えたもの）。コードセルに Python のコードを入力できるようにすると、準備は終了、後はプログラミングである。Colaboratory の利用には、Google のアカウントが必要なので、持ってない人はアカウントを取得しよう。なお、Colaboratory でなくても Python が走る idle などの環境でも良い[4]。

a)

b)

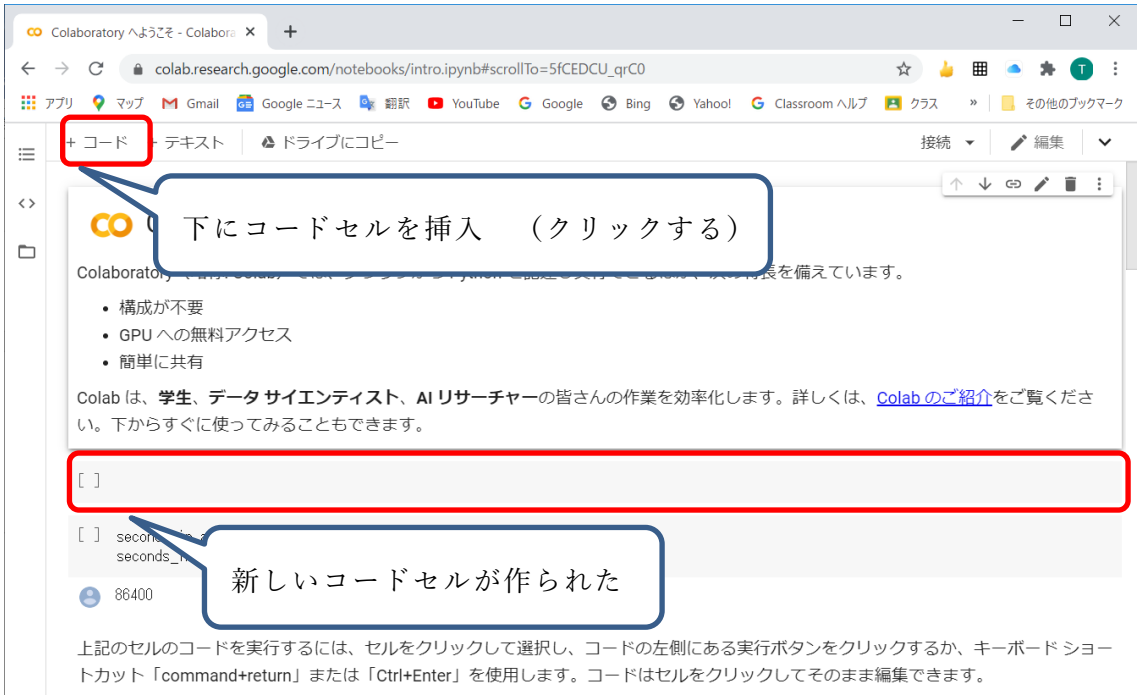


図 3：Google Colaboratory へのアクセス。a)「Colaboratory へようこそ」の Web ページ、b) プログラムの入力のためのコードセルの生成。吹き出し(💬)でコードセルを挿入する様子を説明している。線で囲まれている部分が操作と Colaboratory の動作である。

2.2. matplotlib を使う

簡単にグラフを作成できる Python ライブラリ matplotlib を使って、規則的な点の配置、ランダムな配置、ソーシャルディスタンスを考慮したランダムな配置－円盤のランダムパッキング－、を作画していこう。

2.2.1. 規則的な点の配置

ソーシャルディスタンス r を考慮しながら、L×L の正方形領域での人の配置を考えてみよう。単純化のため人の大きさを無視できるとすると、人を点とみなせる。この単純化は、有限領域での配置を考えた場合、境界での配置も簡単化してくれる。点は大きさを持たないので、境界線上に配置できるようになる。ここで、サイズが有限である円盤のランダムパッキングという視点から考えてみよう。すると、L×L の正方形領域での距離 r を保った点の配置は、(L＋r)×(L＋r)の正方形領域への直径 r の円盤のパッキングと等価な問題に帰着される。

ランダムな配置を考える前に、規則的な配置を考察しておこう。具体的な例としては、学校の教室の座席の配置、講堂（ホール）の式典での並びなどを思い浮かべてみよう。このような場合、単純化すると正方格子上への点の配置になる。さて、matplotlib を使って、点の配置の図を作ろう。 r ＝ L /10 とすると、正方格子上への規則的な点の配置のとき、11×11=121 個の点を配置できる。作図のためのプログラムは、matplotlib のホームページにあるサンプルプログラムを参考にした。図 4 は、正方格子上への点の配置パターンを、matplotlib を使った Python プログラム（プログラムリスト 1）で制作したものである。

今回使った Python の命令のまとめ
1．import 文によるライブラリの読み込み
2．for 文を使ったループ
3．if 文を使った分岐
4．def 文による関数の定義
Python のプログラミングについては、参考文献[4][5]を参考にしてください。

【空白とタブが入っていると、エラーになるときがあるのでコメントには注意】

プログラムリスト 1

```
import matplotlib.pyplot as plt
import numpy as np
data = np.array([[],[]])      # 空の numpy2 次元配列を準備
for i in range(11):
    for j in range(11):
        newdisk = np.array([[i],[j]])
```

```
            data = np.append(data,newdisk,axis=1)
fig, ax = plt.subplots(figsize=(5,5))
ax.scatter(data[0], data[1])
plt.show()
```

プログラムのキーポイントを解説しよう。最初に、import を使って、ライブラリ matplotlib と numpy を読み込み使用できるようにする。次に、プロットする点の座標を numpy の 2 次元配列 data に代入し、matplotlib の pyplot を使って点のプロットをおこなっている。最後に、matplotlib.pyplot.show()でグラフを表示している。(注：import matplotlib.pyplot as plt として読み込んでいるので、matplotlib.pyplot.show() → plt.show()、matplotlib.pyplot.subplots() → plt.subplots() と省略できる。なお、import matplotlib として読み込むと省略できないので注意しよう。)

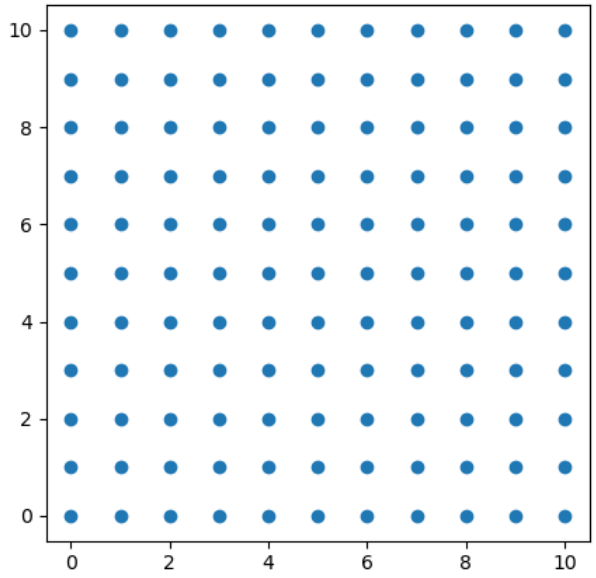


図 4：Python のライブラリ matplotlib 使って制作した図：正方格子上に点を配置したもの。

ここで、L×L の正方形のボックスへ、直径 r の円盤が充填できる最大個数を考えておこう。境界がない場合の平面への最密充填は、三角格子の格子点上に配置することに対応する。しかし、境界がある有限サイズのボックスへの充填の場合は、すき間ができてしまい、最密充填にはならない。L×L の正方形のボックスへ、r = L /10 の円盤を充填する場合の最大個数は、10×6+9×5＝106 個となる。しかし、壁際に配置することを許しているので、円盤のパッキングとは少し異なる。お互いの距離が最低でも r 離れる、つまり最近接距離が r 以上という束縛条件での点の配置となる。10 L ／L(sin π /3)=11.547…より、小数点以下切り捨てすると、11 層のスペースがあるので、両端を入れて 12 層の点を配置できる。よって、点が 12 個の層が 6 層、10 個の層が 6 層、計 11×6+10×6＝126 個の点を配置できる。

具体的に点を配置する様子を、図 5 に示しながらみていこう。最初に、正方形のボック

スのコーナーへ点を配置する(図 5a))。次に、壁(辺)に沿って点を配置していく(図 5b))。そして、層を積み重ねるように配置する（図 5c))。最後に、規則的に点を配置した場合の配置できる点の個数をまとめると、三角格子上 126 個、正方格子上 121 個、となる。

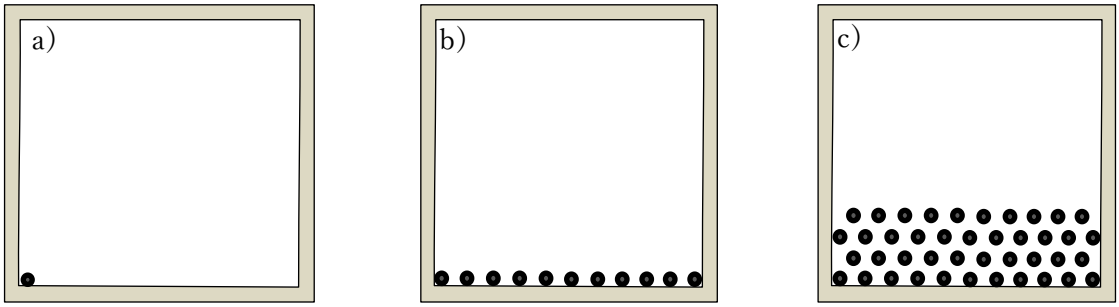


図 5：正方形のボックスに、規則的最密充填していく様子。a）コーナーに配置、b）壁に沿って配置していく、c）層(layer)を積み重ねる。

2.2.2. ランダムな点の配置

L×L の正方形のボックスへのランダムな点の配置を考える。まず、ソーシャルディスタンスを考慮せずに、100 個の点を配置してみよう（プログラムリスト 2）。図 6 に、一様乱数を用いて、100 の点を配置したものを示す。同時に、最近接距離の分布も求めた（図 6 の右に示す)。図 6 から、ソーシャルディスタンスを考慮しないと、最近接距離の分布からも密になるケースが多数発生することが分かる。

プログラムリスト 2

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(20201230)        # 乱数の種
data = np.array([[],[]])
histdistance = np.array([])     # 最近接距離のリスト
ndisk = 0                       # Number of disks の初期値

def calnearestdistance(x, y,ndisk):
    mindistance = 999999        # 仮の最小値（最近接距離）
    for i in range(ndisk):
        distance = np.sqrt((data[0,i]-x)**2 + (data[1,i]-y)**2)
        if (distance != 0) & (distance < mindistance):    # distance=0(自身)の点を除く
            mindistance = distance
```

```
        return mindistance

for i in range(100):
        newdisk = np.random.rand(2,1)
        data = np.append(data,newdisk,axis=1)
        ndisk += 1

    for i in range(ndisk):              # 最近接距離のリストを作成
            r = calnearestdistance(data[0,i],data[1,i],ndisk)
            histdistance = np.append(histdistance,np.array([r]))

    fig, axs = plt.subplots(1,2, figsize=(11, 5))   # 1 行に 2 つの図
    axs[0].scatter(data[0], data[1])
    axs[1].hist(histdistance)
    plt.xlabel("Distance"), plt.ylabel("Number of Persons")
    plt.show()
```

プログラムリスト 2 の内容を簡単に解説しよう。numpy ライブラリの関数 np.random.seed()で乱数の種を決める。最近接点との距離を計算する関数として、calnearestdistance(x, y,ndisk)を定義する。座標(x, y)を入力すると data に登録された ndisk 個の点の座標の中のもっとも近接している点との距離を返してくれる。また、numpy の関数 append でファイルにデータを付け加えている。このプログラムでは、最近接距離の分布を求めるために使っているが、ランダムパッキングのシミュレーションをおこなうときには、円盤が重ならずに配置できるかを判断させる重要な役割を担うことになる。

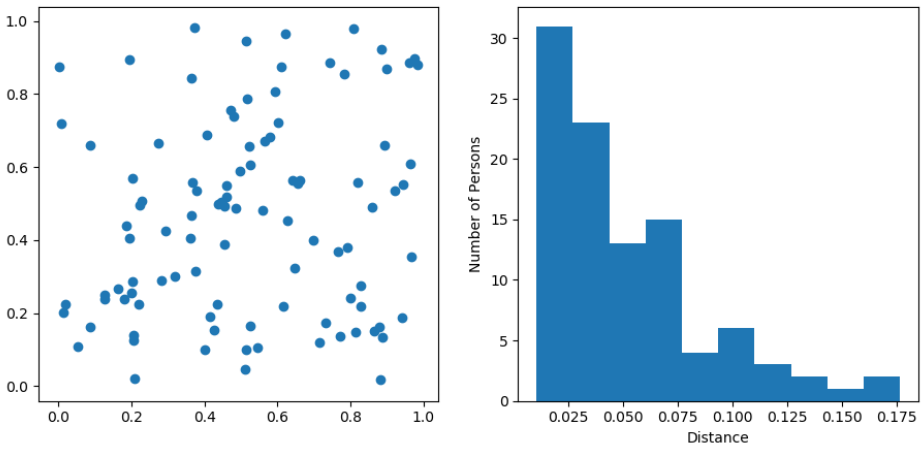


図 6：一様乱数を使って 100 個の点を分布させたもの。左が空間分布、右が最近接距離の分布。

ソーシャルディスタンスを考慮したシミュレーションをおこなう。より厳密に言うと、L×L の正方形領域に最近接距離が r 以上離して点をランダムに配置する問題を考える((L＋r)×(L＋r)の正方形のボックスへ、直径 r の円盤のランダムパッキングと等価である)。ここでは、 r を L /10 としてみよう。

ソーシャルディスタンスを考慮して、お互いの距離が最低でも r 離れるようにして、ランダムパッキングをおこなったときの点の配置を図 7 に示す。図 7 の右側の最近接距離の頻度分布のグラフからも各点が最低 r (＝ L /10)離れているのが分かる。同時に、シミュレーション結果から、もっとも離れているものでも 1.35 r の距離以内であることが分かる。

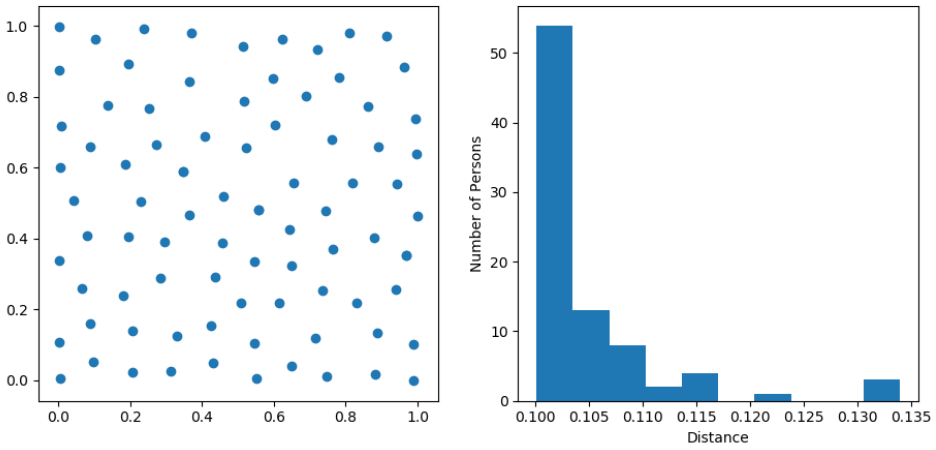


図 7：点の分布と近接距離の分布。L×L の正方形の領域に最近接距離が L/10 以上になるように距離を離して分布させたシミュレーション結果の例。100 万回の試行で 82 人を配置できている。

アルゴリズム（プログラムリスト 3）について説明しよう。今回のアルゴリズムでは、一様乱数を発生させて、配置しようとした点の最近接点との距離を r_N としたとき、$r_N < r$ になった場合は、配置せずに、乱数を再度発生させている。乱数を 100 万回発生させて求めたのが図 7 である。乱数を使って 100 万回のランダムパッキングをさせるのに 526 秒（約 9 分）かかった。その結果、82 人を配置することができた。さらに、300 万回の試行（1581 秒（約 26 分））、500 万回の試行（2647 秒（約 44 分））をおこなっても、82 人以上を配置できなかった。試行時間は、Colaboratory で実行したときのものである。なお、乱数の種を変えると結果が変わることに注意しよう。

プログラムリスト 3

```
import math
import matplotlib.pyplot as plt
import numpy as np

NTRIAL = 1000000              # 試行回数
np.random.seed(20201230)      # 乱数の種
data = np.random.rand(2,1)    # 最初の人の位置
histdistance = np.array([])   # 最近接距離のリスト
```

```
socialdistance = 0.1                # Social Distance
ndisk = 1                           # Number of disks の初期値

def calnearestdistance(x, y,ndisk):
    mindistance = 999999      # 仮の最小値（最近接距離）
    for i in range(ndisk):
        distance = math.sqrt((data[0,i]-x)**2 + (data[1,i]-y)**2)
        if (distance != 0) & (distance < mindistance):   # distance=0(自身)を除外する
            mindistance = distance
    return mindistance

for i in range(NTRIAL):
    newdisk = np.random.rand(2,1)
    r = calnearestdistance(newdisk[0],newdisk[1],ndisk)
    if r >= socialdistance:
        data = np.append(data,newdisk,axis=1)
        ndisk += 1

for i in range(ndisk):          # 最近接距離のリストを作成
    r = calnearestdistance(data[0,i],data[1,i],ndisk)
    histdistance = np.append(histdistance,np.array([r]))

print('Number of disks is ',ndisk)

fig, axs = plt.subplots(1,2, figsize=(11, 5))   # 1 行に 2 つの図
axs[0].scatter(data[0], data[1])
axs[1].hist(histdistance)
plt.xlabel("Distance"), plt.ylabel("Number of Persons")
plt.show()
```

プログラムリスト 3 の補足説明をしよう。プログラムリスト 2 でも使った、座標(x, y)を入力し、data に登録された点群との最近接距離を返す関数 calnearestdistance(x, y,ndisk)がキーポイントである。

点の配置がおこなわれる過程をみるために、シミュレーション途中の点の配置を図 8 に示す。ここでは、エレガントで効率的なアルゴリズムには、重点をおいていない。点を置く場所を乱数を発生させて決めているので、既に配置された点の個数が多くなるにしたがいトライアンドエラーが多くなる。最初の点の配置は、距離 r 以内に他の点の配置がないので、トライアンドエラーの必要はない。しかし、2 番目以降の点の配置は最近接距離が r 以下になった場合、配置し直さなければならない。乱数を発生させた回数は、5 個の配置には 5 回、10 個の配置には 11 回、20 個の配置には 25 回、40 個の配置には 95 回、80 個の配置には 51240 回、そして最後の 82 個の配置には 121970 回、の試行が必要であっ

た。図 9 に、試行回数 vs.配置に成功した点の個数のプロットを示す。

点をランダムに配置した時の配置効率について考えてみる。最近接距離 r を維持しながら L×L の正方形の領域に配置できる点の最大個数は、126 個である。ランダムな点の配置シミュレーションでは 82 個の点を配置することができたので、82/126≒0.65、大雑把に 2/3 の効率で点が配置できたことが分かった。しかし、明らかに 82 という個数は絶対ではない。乱数の種を変えたシミュレーションをおこなうことによって、配置できる点の個数が変わる。例えば、プログラムリスト 3 の 5 行目の乱数の種を変えて、np.random.seed(20210110)とすると、配置できる点の個数は 77 になる。

掲載した基本となるプログラム（プログラムリスト 3）を利用すると、乱数の種を変えるなどさまざまなことが試せる。さらに、数行のコマンドを追加するだけで、可能性が広がる。例えば、ボックスと円盤 r の比を 10:1 にしたものをみてきたが、この比を変えてみるとどう変わるかも面白いテーマである。ぜひ、読者の方々がいろいろ試してくだされば幸いである。プログラムを変えていくうちに、Python のプログラミングにも慣れていくことが期待されるので、いろいろと試して楽しんでほしい。

Python のライブラリ matplotlib を使うことにより、簡単にグラフを作れることができるのをみてきた。このように、ライブラリが充実しているのが Python の特徴のひとつである。便利なライブラリを見つけるのも Python プログラミングの楽しみである。Python は、学びやすいプログラミング言語であり（[4]を参照)、さらに Colaboratory を使うとすぐに始められる。ぜひ、スマートフォンなどで楽しんで頂けたら思う。また、プログラムは形の科学会の HP から download できるようにする予定である。

【(注) Colaboratory の Home Page[1]からキャプチャーした図 3 は、本来カラーであり白黒で印刷すると見にくい場合があります。】

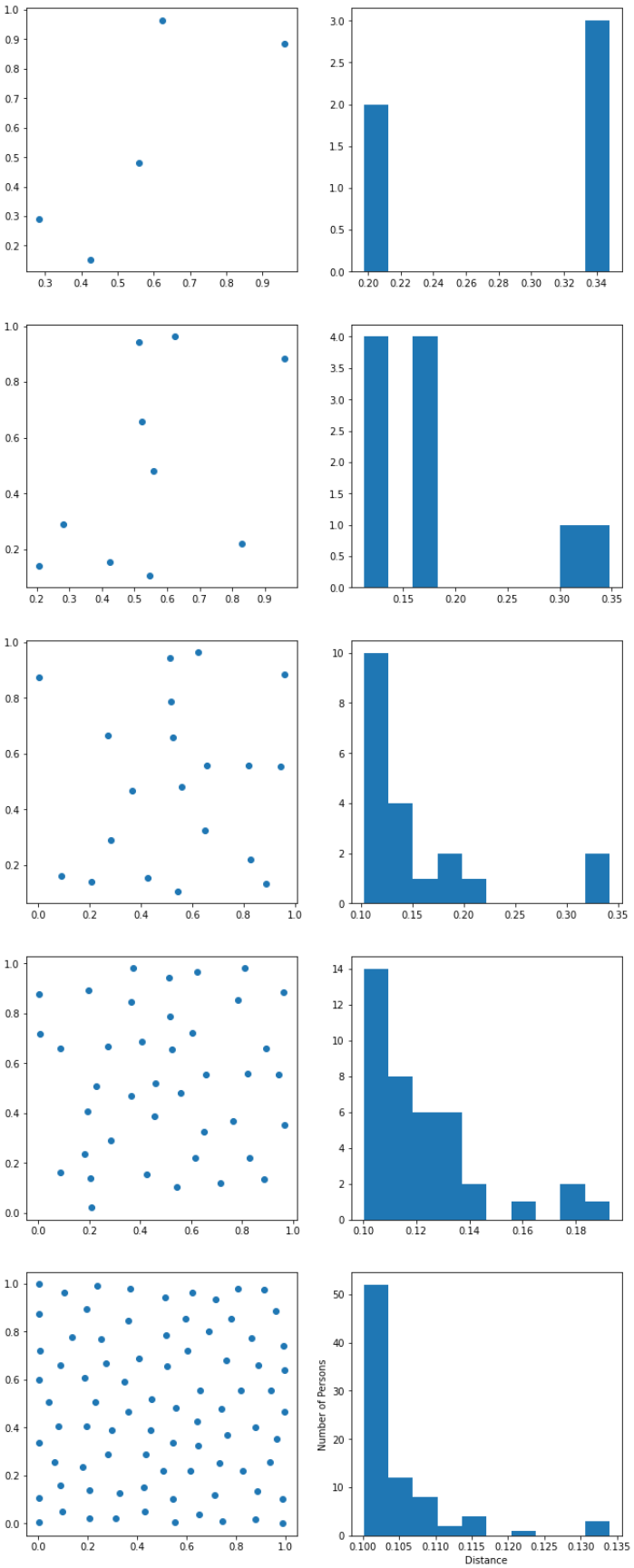


図 8：ランダムパッキングの様子。5 個、10 個、20 個、40 個、そして最終状態(82 個)が示されている。

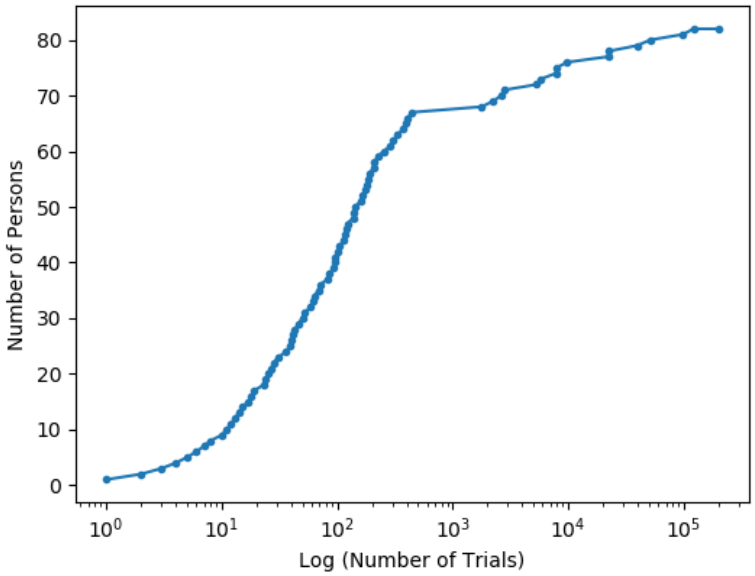


図 9：試行回数 vs.人数のプロット。

謝辞

原稿を読んで有益なコメントをしてくださった髙田宗樹教授、大学院生の高津和紀君をはじめとする福井大学大学院工学研究科知能システム工学専攻非線形科学研究室のメンバーに感謝いたします。

参考文献

[1] Google Colaboratory https://colab.research.google.com/notebooks/intro.ipynb#

[2] matplotlib.org https://matplotlib.org

[3] 平田隆幸：Colaboratory で形の科学を楽しもう－コッホ曲線を描いてみよう－、形の科学会誌、35 巻、1 号、75-80、(2020)

[4] 平田隆幸：Python を始めて 10 日でゲームを作れるようになるのか－ 61 歳からの挑戦：0 から始める Python Programming－，形の科学会誌、35 巻、1 号、55-74、(2020)

[5] クジラ飛行机：ゼロからやさしくはじめる Python 入門、マイナビ出版、pp.255、(2018)

講座に掲載されたプログラムは、形の科学会の HP の以下からダウンロードできます。
https://katachi-jp.com/journal35
カット＆ペーストで Colaboratory のコードセルに貼り付けて、走らせてみてください。