香川大学経済論叢
第84巻 第4号 2012年3月　1－16

蛍の集団の同期点滅発光のシミュレーション

本　田　道　夫

Ⅰ．はじめに

東南アジアでは何万匹という蛍が，夕方になると最初の頃はそれぞればらばらに点滅発光しはじめるが，時間が経つにつれて同期点滅発光するようになるということが，「SYNC なぜ自然はシンクロしたがるのか」［ストロガッツ 2005］という本に書かれていた。この現象を説明するのに，かつては，蛍の中に指揮者ともいうべきものが存在し，それが同期発光に導いているという説もあったとのことであるが，現在ではそのような指揮者の存在なしに，各蛍が自分の近隣の蛍に合わせていけば，徐々に同期するという説が有力となっていることである。同書によると，問題をモデル化した，非連続なインパルスによって相互作用する振動子（蛍に相当）で，かつ特殊な条件の場合は同期することが数学的に証明されているとのことであったが，特殊な条件なし，あるいは数が多い場合は証明できていないが，シミュレーションで同期を確かめることはできるということであった。なお，同期現象は，壁に掛けられた複数の振り子時計などでも観察できるとのことである。

このことに，興味をもち，プログラムを作成しシミュレーションを行って数

万匹の蛍が，最初はばらばらでもだんだん同期して点滅するようなことがパソコンの画面で実現できれば綺麗だろうと考えた。その反面，自分の周りに合わせるだけで本当に数万匹のものが同期するのか，ある程度の範囲は同期しても，全体としてはよくスポーツ観戦の応援で見られるようなスタンディングウエーブのような伝播的な現象となるのではないかという疑問も持った。しかし，それもまた，綺麗だろうとシミュレーションのプログラムを作成することにした。

そこで，同期現象についてインターネットで検索したところ，蛍だけでなく，壁に掛けた振り子時計などの同期について紹介・解説している Web サイト（以下では，単に「サイト」ということもある）がいくつかあった。同期に関する解説では，以下のⅡで述べるような，周辺の蛍に合わせる「式⑴」が示されていたが，式中で用いられている関数Γ（ガンマ）に相当する関数の具体的な形はほとんど書かれていなかった。その中で，蛍の同期ではないがアマガエルの同期発声行動についての論文で，［合原 2009］には，式中の関数の具体的な形として，結合強度と三角関数の値の積が採用されていた。しかし，結合強度の具体的な値は示されていなかった。さらに関数の具体的な形を記しているサイトを探していたところ，Java のプログラムを掲載しているサイト［Rekimoto］があった（同様に結合強度と三角関数を用いていた）ので，それをダウンロードし，コンパイル・実行すると，確かに，最初はばらばらでも，次第に同期していくことが観察できた。そのときの蛍の数は 10×10 に配置した 100 匹のものであった。そこで，蛍の数を 50×50 程度に増やし，さらに表示領域を大きくするなど，Java プログラムを変更して実行してみると，パソコン画面の制限もあり，表示される蛍の数が 50×50 の全部ではなかったが，表示される範囲でも同期するようには見えなかった。その Java のプログラムはコンパイル時に「警告：［deprecation］Thread の stop（）は推奨されません」が表示されたこともあり，蛍の数を多くするときの実行速度も考えて Win 32 API を用いる C のプログラムとして作成することにした。

作成したCのプログラムでも蛍の数が15×15程度の少ないときは同期するが，100×100のように多くなるとほぼ同期したと見なせる状態には至らずに，ある程度の範囲は同期してそれが他の範囲に伝播する，スタンディングウエーブに似た動きにしかならなかった。

蛍の同期現象を紹介しているサイトでは1つのサイトを除いて，同期すると書いていたし，実際の蛍は同期するとのことであったので，プログラムにいろいろと思いついた工夫を加えていった。その一つのサイトでは，同期せずに，カオスのように似た状態を繰り返すというような内容であり，私の実行結果と同様であったと思われた（残念ながら，そのサイトは現在見つけられていない）。以下では，それらの工夫と結果について紹介する。なお，YouTubeの動画サイトに，実際の蛍の同期の様子を撮影したであろうものが載っていたが［YouTube 1］，一瞬同期しているようにも見えるが，見方によっては，少し発光がずれているところもあるようにも見えるものであった。同期現象について検索していたときに，YouTubeの動画で，揺れやすい台の上に置かれた複数のメトロノームが最初はばらばらに動いているが，時間が経つにつれ同期していくものも載っていた［YouTube 2］。

Ⅱ．シミュレーションプログラムの作成と実行結果

シミュレーションについての説明での位相を変更させる一般的な形は次式(1)のようであった。

$$d\theta_i/dt = \omega_i + \sum_{j=1}^{N} \Gamma_{ij}(\theta_j - \theta_i) \qquad (1)$$

ただし，式中の記号は次の意味である。

θ_i　　：i 番目の蛍の時刻 t での位相

$d\theta_i/dt$ ：i 番目の蛍の時刻 $t+1$ への位相の変化

ω_i　　：i 番目の蛍の固有の角速度

Γ_{ij}　　：i 番目の蛍と j 番目の蛍の位相の差の評価関数

そして，［合原 2009］では，具体的なガンマ関数は次のように記されていた。ただし，「i 番目の蛍と j 番目の蛍の結合強度」については正であるという以外に具体的な値の記述はなかった。

$$\Gamma_{ij}(\theta_j-\theta_i)=K_{ij}sin(\theta_j-\theta_i) \quad (2)$$

ここで，K_{ij} は i 番目の蛍と j 番目の蛍の結合強度である。

また，サイト［Rekimoto］でのプログラムは，次のような位相変更の式であった。

$$d\theta_i/dt=\omega_i+Z(\theta_i)\sum_{j=1}^{N}f(\theta_j-\theta_i) \quad (3)$$

ただし，式中の関数 $Z(\theta_i)$ 記号は次の意味である。

θ_i ：i 番目の蛍の時刻 t での位相

$d\theta_i/dt$ ：i 番目の蛍の時刻 $t+1$ への位相の変化

ω_i ：i 番目の蛍の固有の角速度

$Z(\theta_i)$ ：i 番目の蛍の他の蛍の影響を受ける感度関数

：プログラムでは定数としていた。

f ：i 番目の蛍と j 番目の蛍の位相の差の評価関数

：プログラムでは三角関数の sin を利用している。

なお，Java のプログラム中では関数 Z は定数であり，引数シータに依存したものではなかった。そこで，作成する Win 32 API の C プログラムでも関数 Z は定数とすることにした。

ここで，参考にしたサイト［Rekimoto］からダウンロードした Java プログラムについて，簡単に説明しておく。

・それぞれの蛍の固有の周期は，基準の角速度よりも乱数を用いてランダムに 20％程度散らしている。

・それぞれの蛍の最初の位相も乱数でランダムに設定している。
・各蛍は自分の周り（上下左右と斜め）の８匹のそれぞれの蛍の位相との差に対する *sin* の値の合計と自分の固有の周期の和に１より小さな値の定数を掛けた値を現在の位相に加えたものを次の時点での位相としている。ただし，右端の蛍の右側の蛍としては同じ行の左端の蛍を採用している。上下の端や左端の蛍に対しても同様である。なお，１より小さな値を掛けるのは，周期を分割しているのではないかと思われる。
・発光の強さは，位相に対して三角関数 *sin* で決め，その大きさの円で表示している。
・結合強度 Z の値は蛍によらず，一定値で 0.6 としている。

そして，このような初期設定を参考にして，簡単に蛍の数を変更できるようなプログラムを作成した。

まず，最初に 15×15 の 225 匹の蛍で，乱数の種１で実行した。そのときの最初の蛍の発光状態が図１である。円が大きいところほど強く発光，発光していないところは何も表示されていないところである。

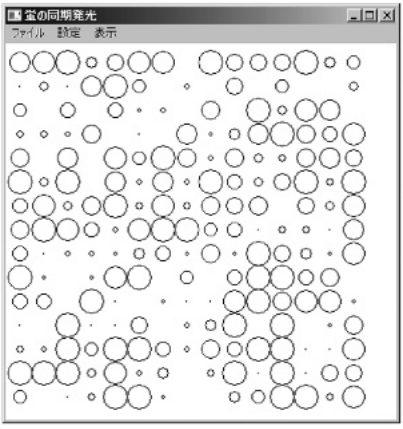

図１：蛍の数 225（15×15）の最初の状態

そして，ある程度時間が経過すると全部の蛍が最も強く発光している状態から全部の蛍が光っていない状態までのすべての状態で同期しているように見えるようになった。図２は，すべての蛍が最も強く発光している状態である。

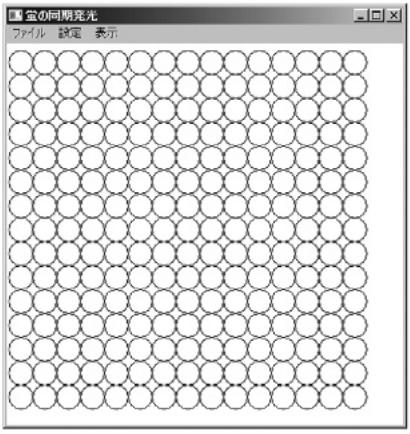

図２：蛍の数 225（15×15）の同期した状態

この程度の数であれば完全に同期した点滅となった。図１，図２とも本稿の印刷のために，背景は白，蛍の光っている状態も中を塗りつぶさない円で表示しているが，実際は黒の背景に緑で塗りつぶした円で表示されるので，点滅が徐々に同期していく様子はなかなか綺麗なものである。

つぎに，蛍の数を 50×50 の 2,500 匹に変更してプログラムを実行した。このプログラムでは蛍の大きさを小さくしていないために 50×50 の横方向は開いたウインドウに全部表示できたが，縦方向は全部は表示できなかった。それでも，かなり時間を経ても点滅の同期発光は起こらずに，ある程度の範囲が同期して，それが伝播していく様子であった。乱数の種をいろいろに変更して実行しても，数が多いと全部が同期することはなかった。

実際の蛍の場合はかなりの同期が見られると思っていたので，このプログラ

ムを点検しプログラムに間違いはないことを確認した後，次のような変更をいろいろと考えた。

・位相の変化量を決める自分の上下左右斜めの蛍は８匹（１重）ではなく，最大２重の位置までを，それぞれの方向について乱数で決めることにして，少し範囲を広げた。つまり，ある方向は１匹，他の方向は２匹から影響を受けるとした。このことを「近隣の蛍は２重」というようにする。
・それらの上下左右の蛍のうち，位相の変化を決めるのは発光している（つまり *sin* の値が正のもの）についてのみとした。このことは，かえって同期しない，あるいは同期が遅くなるかも分からないが，実際の蛍の場合は発光しないものは見えていないので，影響を受けないと考えるのが妥当であろうと考えた。
・参考にした Java のプログラムでは，上下左右の端の蛍の隣は，たとえば右端の蛍の右隣の蛍は同じ行の左端の蛍とするようにしていた。実際の蛍の場合では右端の右隣は存在しないので，右端の蛍の右隣は無いものとして計算することにした。このことも，同期発光に収束することを妨げると思われるが実際の蛍の場合を考えてこの方法とした。

このように変更して実行したが，多い数の蛍となると，同期するようには見えなかった。これまでは，いろいろな値を変更して観察する場合に，プログラム中の定数の値を変更してコンパイル・リンク（ビルド）したのち，実行していた。これらの値は＃define 文で記述しているので，変更すること自体はさして手間ではなかったが，いちいちビルドし直す必要があったので，デフォルト値は決めておき，プログラムを起動したあとで，次のようなダイアログボックスで，デフォルト値を変更してから実行できるように変更した。

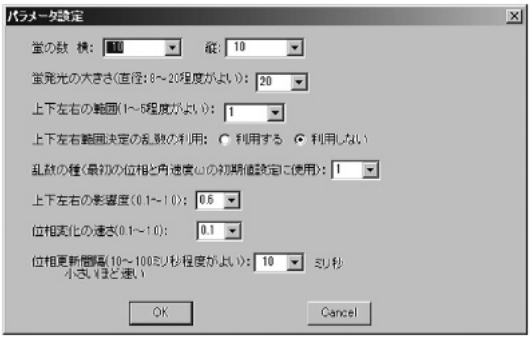

図 3 ：設定ダイアログ(1)

設定できる値は次のものである。

・蛍の数は縦と横の蛍の数をそれぞれ指定できる。

・蛍発光の大きさ（ドットでの指定）：10×10 程度であれば大きさは 20 でよいが，蛍の数を増やすとパソコン画面からはみ出すので，大きさを変えられるようにしている。

・影響を受ける近隣の上下左右の蛍を何重までにするかの指定。

・乱数の種は，各蛍の固有周期と最初の位相の設定に用いる乱数の種。

・上下左右の蛍から受ける影響度。式(2)での K，式(3)での関数 Z（プログラムでは定数）に相当。

・位相変化の速さは，近隣の蛍の位相との差に対する *sin* の値の合計と自分の固有の周期の和に掛ける 1 未満の数である。

・位相更新間隔は，位相を計算し表示する間隔をミリ秒単位で指定する値である。

ダイアログボックスで乱数の値などいろいろと値を変更して実行してみた。蛍の数が多い場合は，やはり同期するようには見えなかったが，蛍の数が増えた場合，同期はスムーズには進まないけれど徐々に収束はしているのではないかと思っていた。しかし，画面を見る限り収束していると判断するのは難しい

ので，収束しているのであれば全体の蛍の分散が徐々に小さくなるはずであると考えて分散を計算・表示することにした。分散を求めてみると，ある回数までは収束していくが，その後は発散するようであった。ただし位相変化の計算ごとに分散を求めていたのではその出力数も多くなるために，ある回数（デフォルトは500回）ごとに，その中の最小値だけを表示するようにした。また，発光しはじめ，あるいは発光終わり直前の弱い発光のものから最も強い発光までいろいろとある（図1の丸の大きさが小さいものから大きなもの）が，その回数内で全部の蛍が発光した回数，と全部の蛍が発光していない回数も求めて表示することにした。

また，実際の蛍の場合は，目の良い蛍も（自分に影響を与える蛍として，他の蛍よりも遠くのものが見える）いるのではないか，あるいは，飛んでいる蛍は，多くの蛍を見ることができるし，逆に多くの蛍からも見えるのではないか，さらに，飛んでいる蛍は別の位置に移動することによって，影響を与える範囲も徐々に変わっていくのではないか，などと考えて，そのような蛍を入れることにした（図4）。このようなことをしても，蛍の数が多くなると，同期しないであろうとは思っていたが，その通りであった。しかし，例えば，上記のような特別蛍を導入しない場合は，17×17の数では同期発光しなかったが，特別蛍を導入すると，同期発光するようになるなど，同期発光が見られる蛍の数の上限は少しは大きくはなった。

15×15，影響を受ける範囲を自分の回り3重まで，乱数の種1で実行した分散の変化の結果を表1に示す。

表の見方について，第1行目を例に説明する。

・478（1-500）　　：1～500回の計算中478回目がその右の最小分散のときである。

・最小分散：0.068（0.068）：上記1～500回の計算中の最小分散の値。括弧内は，1回目からこれまでの最小分散値。

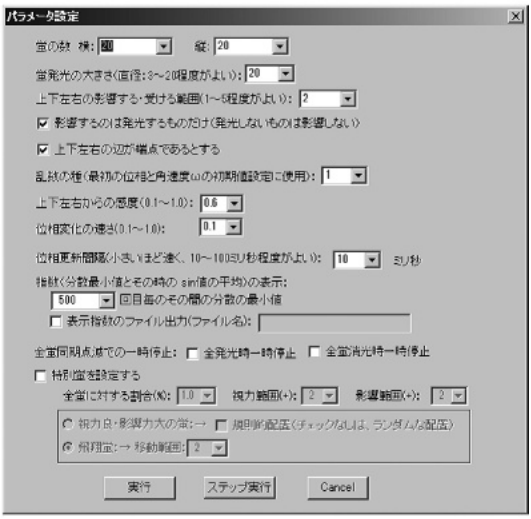

図 4 ：設定ダイアログ(2)

・全明：61　全暗：63　　：1 ～500 回の計算中，強弱はあるにしても全部の蛍が発光した回数と全部の蛍が発光していない回数。

分散は計算 500 回ごとに，その間の最小値を表示している。最初に同期したように見えたのは，1,001～1,500 回繰り返しのときであった（つまり，1,001 回目から 1,500 回目の間のどこかで，同期したように見えた)。そして分散の最長値 0.042 は 2,301 回目から 22,000 回目までは同じ値をとっている（同じ値が続くようであるので，22,000 回目までで中止した)。なお，最初に収束しているように見えるまでの繰り返し回数，収束していく分散の値などは，最初の乱数の値により多少異なる。

表１：225（15×15）匹の蛍：乱数の種１，影響を受ける範囲１～３の実行ログ

478(　1-　500):	最小分散:0.068(0.068)	全明: 61　全暗: 63
975(　501-　1000):	最小分散:0.045(0.045)	全明:145　全暗:148
1417(　1001-　1500):	最小分散:0.043(0.043)	全明:156　全暗:156
1859(　1501-　2000):	最小分散:0.043(0.043)	全明:155　全暗:158
2301(　2001-　2500):	最小分散:0.042(0.042)	全明:158　全暗:158
2964(　2501-　3000):	最小分散:0.042(0.042)	全明:161　全暗:157
3185(　3001-　3500):	最小分散:0.042(0.042)	全明:160　全暗:157
3627(　3501-　4000):	最小分散:0.042(0.042)	全明:160　全暗:158
4290(　4001-　4500):	最小分散:0.042(0.042)	全明:160　全暗:158
4953(　4501-　5000):	最小分散:0.042(0.042)	全明:161　全暗:157
:	:	:
18213(18001-18500):	最小分散:0.042(0.042)	全明:158　全暗:160
18655(18501-19000):	最小分散:0.042(0.042)	全明:157　全暗:160
19097(19001-19500):	最小分散:0.042(0.042)	全明:157　全暗:161
19760(19501-20000):	最小分散:0.042(0.042)	全明:158　全暗:160
20202(20001-20500):	最小分散:0.042(0.042)	全明:158　全暗:158
20644(20501-21000):	最小分散:0.042(0.042)	全明:157　全暗:158
21086(21001-21500):	最小分散:0.042(0.042)	全明:157　全暗:158
21528(21501-22000):	最小分散:0.042(0.042)	全明:158　全暗:157

100×100 で実行した結果を表２に示す。この場合はかなりの回数の計算を繰り返しても同期は見られずに，部分的に同期し，スタンディングウエーブのように伝播していくことを繰り返していた。ただし，しばらくは同じ伝播を繰り返すのではなく，よく似た状態を繰り返すが同じ状態はとらないように見えた。そして分散値は 918 回目が最小の 0.074 であり，その後は徐々に大きくなっていっている。さらに繰り返しを続けても分散値が小さくなることはなかったが，64,012 回目で分散値 1.288 となり，それ以後は処理を中断した 116,000 回目までその値をとり続けた。表示を見ている限りでは，同じ繰り返しなのか，少しずつ変わっているのかは分からなかった。

表２：100×100　乱数の種１，影響を受ける範囲１～３の実行ログ

476(　1-　500):	最小分散:0.090(0.090)	全明:12　全暗:17
918(　501-　1000):	最小分散:0.074(0.074)	全明:95　全暗:93
1028(　1001-　1500):	最小分散:0.075(0.074)	全明:79　全暗:75
1526(　1501-　2000):	最小分散:0.085(0.074)	全明:58　全暗:59
2023(　2001-　2500):	最小分散:0.101(0.074)	全明:45　全暗:44
2520(　2501-　3000):	最小分散:0.118(0.074)	全明:34　全暗:35
3017(　3001-　3500):	最小分散:0.137(0.074)	全明:29　全暗:29
:	:	:
54509(54501- 55000):	最小分散:1.274(0.074)	全明: 0　全暗: 0
55006(55001- 55500):	最小分散:1.275(0.074)	全明: 0　全暗: 0
55614(55501- 56000):	最小分散:1.277(0.074)	全明: 0　全暗: 0
56056(56001- 56500):	最小分散:1.278(0.074)	全明: 0　全暗: 0
56719(56501- 57000):	最小分散:1.279(0.074)	全明: 0　全暗: 0
:	:	:
57161(57001- 57500):	最小分散:1.279(0.074)	全明: 0　全暗: 0
57603(57501- 58000):	最小分散:1.279(0.074)	全明: 0　全暗: 0
:	:	:
115505(115501-116000):	最小分散:1.288(0.074)	全明: 0　全暗: 0

いろいろと工夫をしてみても，蛍の数が多くなると同期現象は見られないし，分散値はある段階までは最小値に収束していくが，さらに時間が経てば，かえって大きくなっていき，ある段階で安定したようになる。しかし，先にも述べたように，多くの検索したサイトでは式⑴あるいは⑵のように次の段階の計算を行うと同期すると紹介していた。このような状況のときに，先に述べた，YouTube の動画で，揺れやすい台の上に置かれた複数のメトロノームの同期についてよく見ると，メトロノームの周期はすべて同じに設定されているように見えた。

そこで，すべての蛍の固有周期を同じ値に設定し，乱数の種は１，近隣の蛍は２重として，実行したところ，蛍の数が 100×100 であっても，表３に示す

ように，時間はかかるが最小分散値は徐々に小さな値に収束し，同期発光しているように見えるようになった。なお，表３の最小の分散値は小数点以下８桁としている。500 回ごとの各区間内で分散の最小値をとるのが，その区間の一番最後あるいはその少し前までの計算であることが続いている。このことは各500 回のなかでも計算を繰り返すたびにほぼ小さくなっていることを示していると考えられる。

表３：100×100　乱数の種１，影響を受ける範囲１～２の実行ログ

463(1- 500):	最小分散:0.24039245(0.24039245)	全明: 0 全暗: 0
1000(501- 1000):	最小分散:0.12434942(0.12434942)	全明: 0 全暗: 0
1491(1001- 1500):	最小分散:0.07497395(0.07497395)	全明: 0 全暗: 0
1979(1501- 2000):	最小分散:0.05469092(0.05469092)	全明: 0 全暗: 0
2466(2001- 2500):	最小分散:0.04309181(0.04309181)	全明: 0 全暗: 0
3000(2501- 3000):	最小分散:0.02600017(0.02600017)	全明: 58 全暗: 58
3495(3001- 3500):	最小分散:0.01777524(0.01777524)	全明:156 全暗:153
3983(3501- 4000):	最小分散:0.01316509(0.01316509)	全明:175 全暗:188
4500(4001- 4500):	最小分散:0.01008620(0.01008620)	全明:192 全暗:199
:	:	:
69500(69001- 69500):	最小分散:0.00001275(0.00001275)	全明:248 全暗:251
70000(69501- 70000):	最小分散:0.00001236(0.00001236)	全明:255 全暗:243
70497(70001- 70500):	最小分散:0.00001199(0.00001199)	全明:253 全暗:245
71000(70501- 71000):	最小分散:0.00001164(0.00001164)	全明:243 全暗:256
71472(71001- 71500):	最小分散:0.00001131(0.00001131)	全明:243 全暗:253
72000(71501- 72000):	最小分散:0.00001096(0.00001096)	全明:256 全暗:243
72500(72001- 72500):	最小分散:0.00001063(0.00001063)	全明:255 全暗:243
:	:	:
198000(197501-198000):	最小分散:0.00000001(0.00000001)	全明:255 全暗:245
198497(198001-198500):	最小分散:0.00000001(0.00000001)	全明:246 全暗:254
198990(198501-199000):	最小分散:0.00000000(0.00000000)	全明:243 全暗:257
199500(199001-199500):	最小分散:0.00000000(0.00000000)	全明:251 全暗:249
:	:	:

3,501-4,000 の間で，全ての蛍が強く発光しているように見えた。ただし，同期という訳ではなく，いくつかの範囲のところが先に発光し，他のところは少し遅れて光り出すが，強く発光する時間が重なって全部が強く発光しているように見えるときがあるということである。発光のし始めから発光が消えるまでほぼ安定して同期しているように見えるのは 70,001-70,500 回目あたりからで，そのときの最小分散値は 0.00001199 であった。

100×100 で，影響を受ける範囲を５重としたときは，当然であるが，全部の蛍がほぼ安定して同期発光しているように見えるのは 17,501-18,000 回目と速く，そのときの最小分散値は 0.00005349 であった。

200×200 で乱数の種１，近隣の蛍３重の場合は，最初に全部の蛍が強く発光したように見えたのは 501-1,000 回目あたりであり，30,000 回目あたりから，同期した２つの領域がごく少しの時間差で発光するようになった。70,000 回ほど実行させたが，この状態にあまり変化はなかった。しかし，500 回ごとの最小分散値は徐々に小さくはなっており，70,001-70,500 回目の間では，0.00045658 であった。

また，すべての蛍の固有周期をすべて合わせなくても，最初に設定する固有周期の散らばり方を小さくすれば，同期発光しやすくなった。

Ⅲ．おわりに

最初にプログラムを作ったのは 2008 年５月であり，多数の蛍が同期していく様子のシミュレーションができれば綺麗だろうと思うと同時に，本当に同期するか，スタンディングウエーブのようにある程度の範囲が同期してそれが伝播するようになるのではないかという疑問から，ちょっとした楽しみのつもりでプログラムを作成した。完成して実行すると，蛍の数が少ないときは，ほぼ同期発光するように見えたが，蛍の数が多い場合は，スタンディングウエーブ

のような動きとなった。しかし，そうなると今度は，蛍が多い場合に，どのようにすれば同期するかと，特別に目のいい蛍がいる，あるいは飛翔する蛍は自分も広範囲が見えるし，広範囲の蛍からも見え，さらに移動することにより，より広範囲に速く発光情報を伝えていくというように影響を与えるのではないかというような思いつきをいろいろと試した。それでもなかなか同期しなかった。分散を調べると最初は徐々に分散値が小さくなっていくが，同期の状態になるより前に，今度は分散が大きくなっていくことが観測された。

このように同期発光することについて思いつくたびにプログラムを変更することを行ってきたが，2011 年 6 月に，メトロノームの同期の動画を見ていたときに，メトロノームの周期は同じに設定されているように見えたので，全部の蛍の固有周期を同じ値にしたところ，分散値も徐々に小さくなり全部の蛍が同期しているように見えるようになった。また，必ずしも蛍の固有周期が同じでなくても，差がそれほど大きくなければ，同期発光となるようである。

つまり，隣接するのに合わせるようにした場合，各蛍の固有周期の差が小さければ同期発光するようになり，差が大きければスタンディングウエーブのようになるということである。

このことは，蛍に限らず，多くのものを同期させようとした場合に，指揮者をもうけずに隣接した他のものに合わせる方法とした場合には，それぞれの固有の周期の差がある程度以上であれば，全体は同期しにくいことを示していると考えられ，蛍だけではなく，いろいろな場合に当てはまりそうである。

参 考 文 献

［Rekimoto］ http://www.sonycsl.co.jp/person/rekimoto/java/
［YouTube 1］ http://www.youtube.com/watch?v=sROKYelaWbo&feature=related

［YouTube 2］ http://www.youtube.com/watch?v＝9-jfla4FHSs

［合原 2009］合原一究，武田龍，水本武志，高橋徹，奥乃博「ニホンアマガエルの同期した発声行動に関する数理的研究および音響信号解析」数理解析研究所講究録第 1663 巻 2009 年 153-158

［ストロガッツ 2005］ストロガッツ，スティーブン「SYNC－なぜ自然はシンクロしたがるのか－」早川書房，2005 年