非線形方程式の超離散化

数理科学研究会 竹内結香

目 次

§1　はじめに

以前, セルオートマトンによる渋滞のモデル化を扱ったが, セルオートマトン自体についてより詳しく知りたくなったので調べることにした. セルオートマトンを利用したシミュレーションと, 非線形方程式とセルオートマトンの対応について述べる.

これは, 今までの学習内容をまとめたもので, 特に新しい内容は含まない.

§2　セルオートマトンとは

セルオートマトン (Cellular Automaton, CA) は, 数学や計算機科学で使われる離散的なモデルで, 格子状の空間を通じてシステムの状態変化をシミュレーションするために設計されている. このモデルは, 基本的には「セル」と呼ばれる個々の要素で構成され, 各セルは決まった位置に固定されていることが特徴である. セルは, 空間を構成する最小単位であり, セル全体が並んだ格子状の空間がシステム全体を表現している. この格子は、1 次元 (直線上にセルが並ぶ構造) や 2 次元 (正方形や六角形のタイルのように平面上のセルの配置), 場合によってはそれ以上の高次元で構成されることもある.

各セルには「状態」と呼ばれる特定の値が割り当てられており, この状態が時間とともに変化していく. 状態は通常, 整数値やブール値などで表現され, システム全体の「現在の状態」を定義する. 単純な表現により、これらが時間とともに更新されることでシステム全体が複雑な挙動を示すようになる.

また, セルオートマトンの挙動には, 各セルの周囲にある「近傍 (近接するセル)」が影響を与える. 近傍とは, あるセルに対して周囲にある特定のセル群を指し, その範囲や構成はモデルの構築方法によって異なる. たとえば, 2 次元格子の場合, 中心のセルを取り囲む上下左右のセルが近傍とされることが一般的である. さらに, 斜め方向も含めることで, より複雑な近傍を設定することも可能である. こうした近傍の定義が, セルの状態変化に大きく関与するため, 近傍の選定がシステムの全体的な挙動やパターン形成に重要な影響を与える.

そして, 「ルール (状態遷移の法則)」によってセルオートマトンの挙動を決定される. このルールは, 各セルが次の時間ステップでどのように状態を変化させるかを決定するもので, セル自身の状態やその近傍の状態に基づいて設定される. ルールが全てのセルに適用されることで, システム全体が時間とともに進展し, 複雑な挙動やパターンを生み出していく.

これらの構成要素が組み合わさることで, セルオートマトンはシンプルながらも非常に多様で複雑な振る舞いを生み出す. このモデルは, 微細な相互作用が全体の複雑なパターンに発展することを示すための理想的なツールとして知られており, シミュレーションや研究分野で幅広く利用されている.

2.1　一次元セルオートマトン

直線状に敷き詰められたセルの状態について考えるのが一次元セルオートマトンである. 各セルは, そのセル自身とその近傍の状態によって変化をしていく.

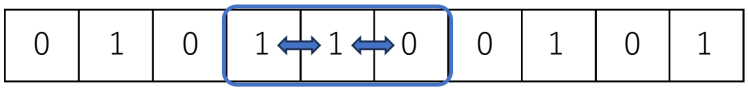

図 1: 一次元セルオートマトンの例

2.2 ECA

一次元セルオートマトンのうち, 以下のようなものをエレメンタリー・セルオートマトン (Elementary Celler Automaton, ECA) という.

- それぞれのセルの状態は 0 または 1.
- 次の時刻での状態は, 現在のそのセル自身とその両隣のセルの状態で決まる.

例えばルール 184 と呼ばれるものは, 次のようなルールで時間発展が決まる.

現在の状態	111	110	101	100	011	010	001	000
中央セルの次の状態	1	0	1	1	1	0	0	0

図 2: ルール 184 の状態遷移

上の行が現在の時刻 t の状態で, そのセルと両端のセルの状態を示しており, 下の行は, 次の時刻 $t+1$ に真ん中のセルがどの状態になるかを示している. ルール 184 と呼ばれる理由は, 時刻 $t+1$ の並び 10111000 を 2 進数表記とみると, 184 になるからである. 同様に考えると, ECA のルールは 0-255 の 256 通りがあることが分かり, それぞれがさまざまな挙動を示す.

2.3 ECA のクラス分け

先述の通り, ECA はルール 0-255 まであり, そのそれぞれが異なる挙動を示す.
その解の挙動は大きく 4 つのクラスに分けられる.

クラス 1(一様)

初期の状態に関係なくシステムが単一の状態に収束する傾向が見られる. すべてのセルが同じ状態 (たとえば全て 0 や全て 1) に固定されるため, 時間が経つにつれて完全に静的なパターンになる.

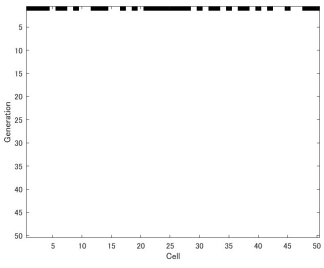

図 3: rule0

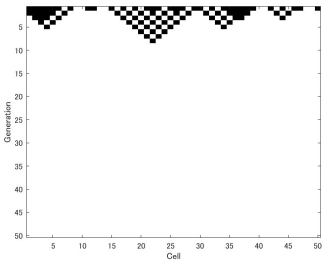

図 4: rule40

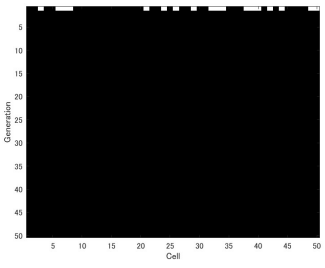

図 5: rule255

クラス 2(周期)

システムが繰り返し構造を持つパターンに収束する. この場合, セルの状態は周期的に変化し, セルの集合は時間とともに同じパターンを維持しながら振動する. クラス 1 と同様に, 安定的で予測可能な挙動を示すが, クラス 1 よりもわずかに複雑で周期性のある構造が見られる.

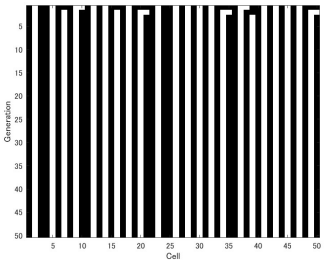

図 6: rule78

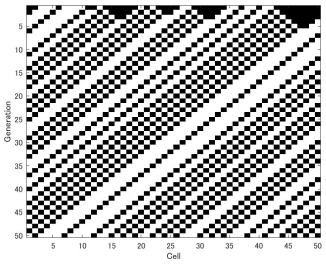

図 7: rule162

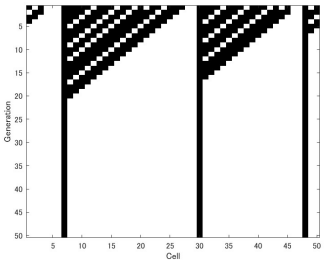

図 8: rule172

クラス 3(カオス)

時間が経過するごとにランダムでカオス的なパターンが生成される. 隣接するセルの相互作用によって, 予測が困難なランダムに見える模様が広がり, システム全体が混沌とした状態に保たれる. このクラスでは, 初期状態にわずかな違いがあるだけでも全く異なるパターンが現れるため, 結果を正確に予測するのが難しくなる. ランダム性や擬似乱数生成に利用されることもある.

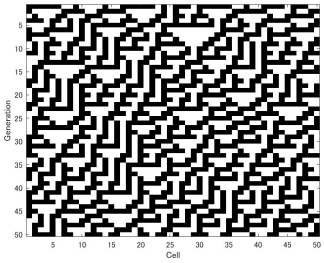

図 9: rule30

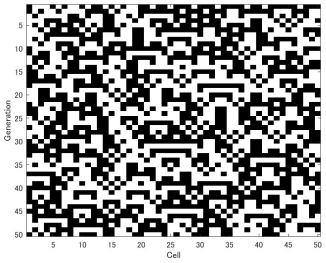

図 10: rule105

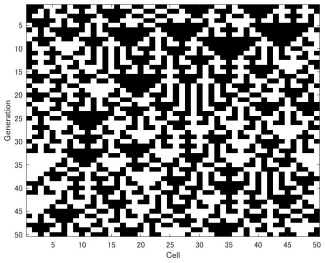

図 11: rule150

クラス 4(周期とカオスの複合)

秩序とカオスの中間にあるような複雑なパターンを示す. このクラスのシステムは, 局所的な秩序を保ちながらも, 境界部分で新たな構造が次々と形成されるため, 自己組織化された構造が現れることがある. この挙動は, しばしば「計算普遍性」と呼ばれる性質を持ち, 他のクラスに比べて非常に複雑な計算を行うことが可能であることが証明されている.

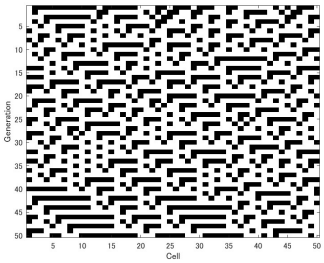

図 12: rule41

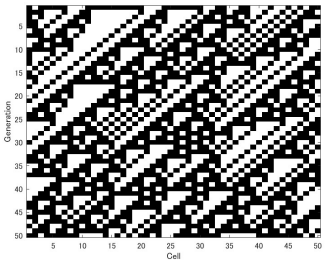

図 13: rule106

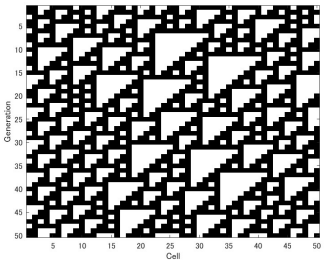

図 14: rule110

2.4 二次元セルオートマトン

二次元空間内に敷き詰められたセルの状態について考えるのが二次元セルオートマトンである. 各セルは, そのセル自身とその近傍の状態によって変化をしていく.

2.5　ライフゲーム

ライフゲームとは, 二次元セルオートマトンを利用した生誕と死滅のシミュレーションモデルである. 格子状の空間に並ぶ無数のセルが, ルールに基づいて次々とその状態を変化させることで, 生命活動を思わせるようなパターンを生成することから「ライフゲーム」と名付けられた.

ライフゲームの各セルは「生 (1)」または「死 (0)」の 2 つの状態を持ち, 状態の変化はそのセルの「近傍 (周囲 8 セル)」にいるセルの数に基づいて行われる. ライフゲームで使用されるルールは以下の 4 つである.

誕生

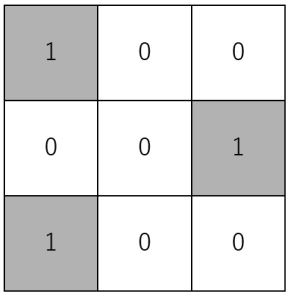

図 15: 誕生

中央のセルが死んでいる (0) 状態で, かつ周りのセルで生きているセル (1) が 3 つある場合, 次の時間で中央のセルは誕生 (1) する.

生存

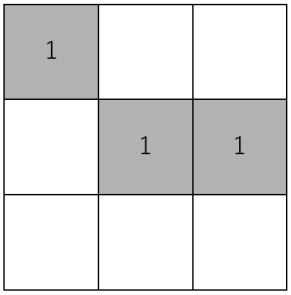

図 16: 生存

中央のセルが生きている (1) 状態で, かつ周りに生きている (1) セルが 2 つもしくは 3 つある場合, そのセルはそのまま次の時間も生存 (1 のまま) する.

過疎

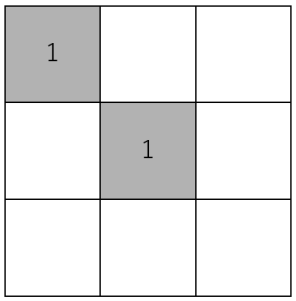

図 17: 過疎

中央のセルが生きている (1) 状態で, かつ周りに生きている (1) セルが 1 つ以下の場合, 次の時間に死滅する (0 になる).

過密

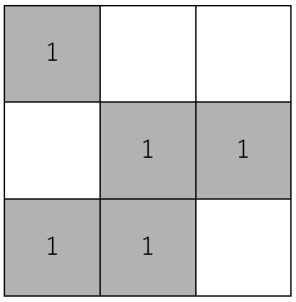

図 18: 過密

中央のセルが生きている (1) 状態で, かつ周りに生きている (1) セルが 3 つ以上ある場合, 次の時間に死滅する (0 になる).

このようなライフゲームのルールは, 現実の生態系や人口動態を思わせる特徴が現れる. 生物が少ないと過疎によって死滅し, 逆に集まりすぎると資源の不足や競争の激化により死滅する一方で, 程よい集まり具合でのみ次の世代が誕生するというバランスの取れた成長が可能になることが分かる.

実際に動かしてみると次のようになります.

図 19: ライフゲーム

今回はランダムに生 (1) を生成させましたが, 初期値によって様々な特徴的なふるまいが見られることが知られている.

§3　セルオートマトンの使い道

セルオートマトンはこのように単純なルールと構造で様々な振る舞いを見せることから, 多くの分野で用いられている. そのうちのいくつかの例を示す.

3.1　交通渋滞のモデル化とシミュレーション

一次元セルオートマトンが用いられる例として, 交通渋滞のシミュレーションが挙げられる.

3.1.1　ルール 184

ここで, 2.3 で述べたルール 184 をもう一度考える. 以下はルール 184 の状態遷移の再掲.

現在の状態	111	110	101	100	011	010	001	000
中央セルの次の状態	1	0	1	1	1	0	0	0

図 20: ルール 184 の状態遷移

ここで, このルールを次のように言い換える.

- 状態 0 を空きセル, 状態 1 を車がいるセルとみる.
- セルに車がいるとき, 右のセルが空であれば, セルの車をひとつ右に動かす (表でいう 010,110).
- セルに車がいるとき, 右のセルにも車がいるなら, セルの車はその場に留まる (表でいう 011,111).

このルールに従って遷移し, 右を進行方向とみることで, 車が進んでいく中での渋滞という現象が表現できる.

実際にこのルールで横一列に並べた 20 セルに対し実行してみる. ここでは黒を 1(つまり車がいるセル), 白を 0(つまり空なセル) とする.

図 21: ルール 184 渋滞シミュレーション

黒の連続して並んでいるところは渋滞と考えることができ, 時間経過に伴いその渋滞は後ろにずれていっていることが見て取れる. このような挙動も実際の渋滞現象を非常によく表現できている. ここではあまり詳しく触れないが, セルの両端は環状につながっているものと考え, 20 番目のセルにある車は次の移動では 1 番目のセルに動く (このような両端でのセルの状態を考えることを境界条件などという).

ちなみに先ほどは 20 のセルに対し初期状態として 12 の車をランダムに配置させたが, この初期状態の車の数を 7 に減らすと, 時間経過に伴いセルは均一に配置され, 渋滞をつくることなく動くようになる.

図 22: ルール 184 渋滞シミュレーション

車の数が少ないほど渋滞が起きにくいのは直感的だが, 個の境目はちょうどセル数 20 の半分の 10 にあたる. 初期の車の数が半分を超えると, 必ず 3 つ並んでしまう箇所が生まれる (渋滞がギリギリ生まれない状態のことを臨界状態などという) ため, 渋滞が発生してしまう.

3.1.2 ASEP

先ほどのルール 184 に対し, 新たに,

- 車は確率 p で次のセルに移動する.
- 右端まで来た車は, 確率 β で流出する.
- 左端からは確率 α で車が流入する.

というルールを追加する. これのようなルールの下で動くものを ASEP モデルなどという.

このとき, 系の車の密度は一定に保たれず, α と β の値によって流量をコントロールすることができる. これもまた, 交通流の表現にてよく用いられるモデルである.

3.1.3 Quick Start

先ほどまではセルの状態遷移には両隣とそのセル自身の状態のみを考えていたが, この近傍を拡張することでより現実に近い交通渋滞の表現を考える.

Quick Start モデルでは, 各車が進行方向 s セル先まで見通せる能力をもっている. 人が車を運転するとき, 前の車がどのように動くのかを常に予測しながら行動しているため, そのような状況を組み込んだものになっている.

このとき, 車は 2 個先のセルまで見通せるため, 110 のとき, 左端の車は次の時刻で移動することができる. 従って, 全ての車が動ける最大の状態とは 110110110... のようなときで, このことから臨界状態の密度は全体のセル数の $\frac{2}{3}$ になることが分かり, 渋滞ふができるまでの時間が先延ばしされたと捉えることができる.

3.2 森林火災のシミュレーション

木々の密集する地域では, ある木が燃え始めるとすぐ隣の木に燃え移り, これを繰り返して広い範囲へ火災が広がっていく. 発火の原因は様々かもしれないが, その燃焼が移り行くさまはまさに周囲に影響を及ぼすという意味でセルオートマトンでのシミュレーションが可能であると考えられる.

そこで以下のようなルールを適用する.

- セルは木, 空地, 燃焼の 3 通りの状態をとり得る.

- $n \times n$ の節からなる空間に, 木をランダムで配置する.
- g を免疫を表す値とし, 隣接するセルの少なくとも一つが燃えているとき, 確率 $1-g$ で燃え始める.
- セルは, 燃えたら単位時間後に空地へと変わる.

他にも木の発火確率や木の成長などを考慮することもできるが, 今回はある 1 セルの燃焼がそのように広がるかを見たいため, 新たな発火は考えず, また森林の成長速度は火災の拡大測度に比べて極めて小さいことから, 森林の成長なども無視することとする.

初期状態の木の配置については, 森林密度で制御するものとする.

実際に $n = 200$, $g = 0.001$ とし, 森林密度を 57%, 60%, 70%とし, 実行してみた結果は次の通り.

森林密度 70% 219step で終了し, 全域に火災が広がる.

図 23: 森林火災シミュレーション密度 70%

森林密度 60% 259step で終了し, ある程度の広がりは見られるものの, 全焼には至らない.

図 24: 森林火災シミュレーション密度 60%

森林密度 57% 153step で終了し, ほとんど広がらず, 鎮火する.

図 25: 森林火災シミュレーション密度 57%

森林密度が燃焼の広がり具合に大きな影響を与えることが分かった.

より現実の状態と近づけるためには, 風の影響や湿度など様々に考慮できる点があると思われるため, 今後の研究課題としていきたい.

3.3 避難行動のシミュレーション

セルオートマトンではなく, エージェントベースのシミュレーションとなってしまったため, 本資料においては省略.

§4 非線形方程式の超離散化

非線形方程式を, その性質を保持したまま超離散化する (セルオートマトンを得る) という話題がある.

偏微分変数の独立変数を離散化したものが偏差分方程式で, さらにその従属変数をも離散化したものがセルオートマトンと考えることができるため, 通常であればこの手続きに従うことで, 対応するセルオートマトンを得ることができるが, 変数の離散化をどのように行うかが問題となる.

独立変数の離散化においては, もとの方程式の特性をできるだけ保つように行うことが求められる. また, 従属変数の離散化においては, 方程式を離散的な値だけで閉じさせるために, セルオートマトンの「加法と最大値をとる操作」に注目し, これと方程式を関係づけられれば良い.

4.1 セルオートマトンの加法と最大値をとる操作

セルオートマトンが加法と最大値をとる操作を用いているというのは, 以下を考えればよい.

例えば, 3.1.1 で述べたルール 184 は, 以下 (再掲) のような状態遷移をルールに持つ.

現在の状態	111	110	101	100	011	010	001	000
中央セルの次の状態	1	0	1	1	1	0	0	0

図 26: ルール 184 の状態遷移

これはつまり,

$$U_i^{t+1} = U_i^t + \min(U_{i-1}^t, 1 - U_i^t) - \min(U_i^t, 1 - U_{i+1}^t)$$

となり, $\min(A, B) = -\max(-A, -B)$ であるから,

$$U_i^{t+1} = U_i^t - \max(-U_{i-1}^t, U_i^t + 1) + \max(-U_i^t, U_{i+1}^t - 1)$$

を得る. よって, 確かに加法と最大値をとる操作によってあらわされていることが分かる.

このような性質を利用した, 次のような極限公式が存在する.

$$\lim_{\varepsilon\to+0}\varepsilon\log(e^{\frac{A_1}{\varepsilon}}+e^{\frac{A_2}{\varepsilon}}+\cdots+e^{\frac{A_n}{\varepsilon}})=\max(A_1,A_2,\ldots,A_n) \tag{$*$}$$

証明. $\max(A_1,A_2,\ldots,A_n)=A_i$ とする.

$$\begin{aligned}
&\lim_{\varepsilon\to+0}\varepsilon\log((e^{\frac{A_i}{\varepsilon}})(e^{\frac{A_1-A_i}{\varepsilon}}+e^{\frac{A_2-A_i}{\varepsilon}}+\cdots+1+\cdots+e^{\frac{A_n-A_i}{\varepsilon}}))\\
=&\lim_{\varepsilon\to+0}\varepsilon(\log(e^{\frac{A_i}{\varepsilon}})+\log(e^{\frac{A_1-A_i}{\varepsilon}}+e^{\frac{A_2-A_i}{\varepsilon}}+\cdots+1+\cdots+e^{\frac{A_n-A_i}{\varepsilon}}))\\
=&\lim_{\varepsilon\to+0}(A_i+\varepsilon\log(e^{\frac{A_1-A_i}{\varepsilon}}+e^{\frac{A_2-A_i}{\varepsilon}}+\cdots+1+\cdots+e^{\frac{A_n-A_i}{\varepsilon}}))
\end{aligned}$$

ここで, $A_k-A_i\le 0(k=0,1,2,\ldots,n)$ なので, 極限 $\varepsilon\to+0$ をとると, $\log(e^{\frac{A_1-A_i}{\varepsilon}}+e^{\frac{A_2-A_i}{\varepsilon}}+\cdots+1+\cdots+e^{\frac{A_n-A_i}{\varepsilon}})$ の項は $\log M$(M は最大値 A_i と一致する $A_1,A_2,\ldots,A_n$ の個数) に収束する. よって, $(*)$ が成り立つ.

この公式を用いることによって, 解析的な連続関数を整数値だけで閉じる関数に変換をすることができる.

4.2 Burgers 方程式

Burgers 方程式とは, Navier-Stokes 方程式において, 移流項が圧力項より十分大きい場合に近似される圧力項を無視した偏微分方程式のことである.

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

$u\frac{\partial u}{\partial x}$ が非線形性を表す項となる. あまり詳細については触れないようにする.

4.3 Burgers 方程式の超離散化

Burgers 方程式は超離散化の手続きによって, ルール 184 に対応することが知られている. $u\frac{\partial u}{\partial x}=\frac{\partial}{\partial x}\frac{u^2}{2}$ を用いると, (1) は,

$$\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(-\frac{u^2}{2}+\frac{\partial u}{\partial x}\right)$$

と変形しておく. ここで, 各項を $u=-2\nu\frac{1}{f}\frac{\partial f}{\partial x}$ の変換 (Cole-Hopf 変換などとよばれる) をすると,

$$\begin{aligned}
(\text{左辺})&=\frac{\partial}{\partial t}\left(-2\nu\frac{1}{f}\frac{\partial f}{\partial x}\right)\\
&=-2\nu\frac{\partial}{\partial x}\left(\frac{1}{f}\frac{\partial f}{\partial t}\right)
\end{aligned}$$

$$\begin{aligned}
(\text{右辺})&=\frac{\partial}{\partial x}\left(-2\nu^2\frac{1}{f^2}\left(\frac{\partial f}{\partial x}\right)^2-2\nu^2\frac{\partial}{\partial x}\left(\frac{1}{f}\frac{\partial f}{\partial x}\right)\right)\\
&=-2\nu^2\left(\frac{1}{f^2}\left(\frac{\partial f}{\partial x}\right)^2+\frac{\frac{\partial^2 f}{\partial x^2}-\left(\frac{\partial f}{\partial x}\right)^2}{f^2}\right)\\
&=-2\nu^2\frac{\partial}{\partial x}\left(\frac{\frac{\partial^2 f}{\partial x^2}}{f}\right)
\end{aligned}$$

これらをまとめると,

$$\frac{\partial}{\partial x}\frac{1}{f}\left(\frac{\partial f}{\partial x}-\nu\frac{\partial^2 f}{\partial x^2}\right)$$

これを積分すると,

$$\frac{\partial f}{\partial x}=\nu\frac{\partial^2 f}{\partial x^2}$$

となる. これはいわゆる拡散方程式というものである. この Cole-Hopf 変換による Burgers 方程式と拡散反応方程式の関係性を用いたまま, Burgers 方程式の差分化を行う. まず, Δx, Δt をそれぞれ空間格子, 時間格子の感覚都市, 拡散方程式の差分化を,

$$\frac{1}{\Delta t}(f_j^{n+1}-f_j^n)=\frac{\nu}{(\Delta x)^2}(f_{j+1}^n-2f_j^n+f_{j-1}^n)$$

とし, $\frac{\Delta t}{(\Delta x)^2}=\alpha$ とすると,

$$f_j^{n+1}=\alpha\nu(f_{j+1}^n+f_{j-1}^n)+(1-2\alpha\nu)f_j^n \tag{2}$$

となる. 次に Cole-Hopf 変換の差分化を,

$$u_j^n=\frac{c}{\Delta x}(\log f_{j+1}^n-\log f_j^n)$$

とする. (2) 式に u_j^n から v_j^n への変換変数 $v_j^n=e^{\frac{\Delta x u_j^n}{c}}$ を行うと,

$$v_j^n=\frac{f_{j+1}^n}{f_j^n} \tag{3}$$

となる. (3) 式に (2) 式を代入することで, v_j^n の時間発展方程式

$$v_j^{n+1}=v_j^n\frac{v_{j+1}^n+\frac{1}{v_j^n}+\left(\frac{1}{\alpha\nu}-2\right)}{v_j^n+\frac{1}{v_{j-1}^n}+\left(\frac{1}{\alpha\nu}-2\right)}$$

を導くことができる. $v_j^n=e^{\frac{\Delta x u_j^n}{c}}$ であったから,

$$f_j^{n+1}=\alpha\nu\{f_{j+1}^n+f_{j-1}^n+(\frac{1}{\alpha\nu}-2)f_j^n\}.$$

ε をパラメータとして, $f_j^n=(\alpha\nu)^{-n}e^{\frac{F_j^n}{\varepsilon}}$, $\frac{1}{\alpha\nu}-2=e^{\frac{\frac{L}{2}-M}{\varepsilon}}$ (L, M は定数) の変換を行うと,

$$e^{\frac{F_j^{n+1}}{\varepsilon}}=(\alpha\nu)^2(e^{\frac{F_{j+1}^n}{\varepsilon}}+e^{\frac{F_{j-1}^n}{\varepsilon}}+e^{\frac{L}{2}-M+\frac{F_j^n}{\varepsilon}})$$

となり, 両辺対数をとると,

$$\begin{aligned}F_j^{n+1}&=e\log(\alpha\nu)^2(e^{\frac{F_{j+1}^n}{\varepsilon}}+e^{\frac{F_{j-1}^n}{\varepsilon}}+e^{\frac{L}{2}-M+\frac{F_j^n}{\varepsilon}})\\&=e\log(\alpha\nu)^2+e\log(e^{\frac{F_{j+1}^n}{\varepsilon}}+e^{\frac{F_{j-1}^n}{\varepsilon}}+e^{\frac{L}{2}-M+\frac{F_j^n}{\varepsilon}})\end{aligned}$$

を得る. ここで, $\varepsilon\to 0$ として極限をとると,

$$F_j^{n+1}=\max(F_{j+1}^n,F_{j-1}^n,\frac{L}{2}-M+F_j^n) \tag{4}$$

となる. 先ほどと同様の L, M に対して, $v_j^n = e^{\frac{U_j^n - \frac{L}{2}}{\varepsilon}}$ の変換を行うと, (3) 式から

$$e^{\frac{U_j^n - \frac{L}{2}}{\varepsilon}} = \frac{(\alpha\nu)^{-n} e^{\frac{F_{j+1}^n}{\varepsilon}}}{(\alpha\nu)^{-n} e^{\frac{F_j^n}{\varepsilon}}}$$

$$U_j^n = F_{j+1}^n - F_j^n + \frac{L}{2}$$

であるから, (4) 式は

$$U_j^{n+1} = U_j^n + \max(U_{j+1}^n - \frac{L}{2}, \frac{L}{2} - M, -U_j^n + \frac{L}{2}) - \max(U_j^n - \frac{L}{2}, \frac{L}{2} - M, -U_{j-1}^n + \frac{L}{2})$$

となり, これは次の式と等価.

$$U_j^{n+1} = U_j^n + \min(M, U_{j-1}^n, L - U_j^n) - \min(M, U_j^n, L - U_{j+1}^n). \tag{5}$$

これが一般的に超離散 BurgersCA と呼ばれている. そして, 式 (5) では M が 1 ステップで動ける粒子 (渋滞モデルでの車に相当) の上限, L が 1 つのセルに存在できる粒子 (渋滞モデルでの車に相当) の上限を表しており, $M = L = 1$ とするとルール 184 のセルオートマトンに対応している.

4.4 KdV 方程式

4.5 ソリトン

ソリトンとは, 簡単に言うと, 形状を保ったまま進行し, 互いに衝突・追突しても崩れないような孤立した波のことである. また, 大きな振幅の孤立波は小さな振幅の孤立波よりも早く進むという特徴もある.

次に, KdV 方程式というものを考える. KdV 方程式とは, 水深の浅い水面を進む波を記述する非線形方程式の一つである, 次のようなものである.

$$\frac{\partial u(x,t)}{\partial t} + 6u(x,t)\frac{\partial u(x,t)}{\partial x} + \frac{\partial^3 u(x,t)}{\partial x^3} = 0 \tag{6}$$

この KdV 方程式は N 個の孤立波が安定に存在する解 (N ソリトン解) を持つことが知られている.

1-ソリトン解について考える. 以下, $u = u(x,t)$ とし, (6) 式を,

$$u_t + 6uu_x + u_{xxx} = 0 \tag{7}$$

と書くことにする. また, 各時刻 t において, 境界条件 $\lim_{n \to \pm\infty} u(x,t) = 0$ を課すことにする.

ここで, 解の形として, $\xi = x - vt$(但し v は正定数) の関数 f で書ける, 即ち $u = f(\xi)$ で書けることを仮定する. すると, (7) は次のような常微分方程式に帰着する:

$$-vf_\xi(\xi) + 6f(\xi)f_\xi(\xi) + f_{\xi\xi\xi}(\xi) = 0.$$

各項を ξ について積分し, 境界条件 $f, f_{\xi\xi} \to 0 (\xi \to \pm\infty)$ を用いると,

$$-vf(\xi) + 3f(\xi)^2 + f_{\xi\xi}(\xi) = 0.$$

さらに, $f_\xi(\xi) = \frac{d}{d\xi} f(\xi)$ をかけて ξ について積分すると, 境界条件の下で,

$$\frac{v}{2} f(\xi)^2 + f(\xi)^3 + \frac{1}{2} f_\xi(\xi)^2 = 0$$

を得る. 変数変換 $\eta = \frac{\sqrt{v}}{2}\xi$, $z(\eta) = \sqrt{\frac{v}{2f(\xi)}}$ を行い解くと,

$$\begin{aligned}\eta &= \pm \int^{z} \frac{dz'}{\sqrt{z'^2 - 1}} \\ &= \cosh(z) + C\end{aligned}$$

となる. これを x, t, u で書き直せば,

$$u(x,t) = \frac{v}{2\cosh(\frac{\sqrt{v}}{2}(x - vt) - \theta)^2}$$

という解が得られる.

この解を 1-ソリトン解と呼び, 一つの山からなる波が, 時間の経過とともに進行していく様子を表している.

図 27: 1-ソリトン解

次に, 2-ソリトン解について考える.

KdV 方程式というのは非線形方程式であるから, 2 つの下位の和が必ずしも解になるとは限らない (重ね合わせの原理 g あ成り立たない). しかし, 2 つのソリトンの山の位置が十分離れている時は, KdV 方程式の良い近似解になっていると期待できる.

天下り的に 1-ソリトン解を次のように書き換える.

$$\begin{aligned}u(x,t;p=\sqrt{v},\theta) &= 2\frac{\partial^2}{\partial x^2}\log(\cosh(\frac{\sqrt{v}}{2}(x - vt) - \theta)) \\ &= 2\frac{\partial^2}{\partial x^2}\log(1 + ae^{(\sqrt{v}(x-vt))}).\end{aligned}$$

但し, $a = a(\theta) = e^{-\theta}$ とし, 任意の $\alpha \in \mathbb{R}$ に対して, $\frac{\partial^2}{\partial x^2}\log(e^{\alpha x}) = 0$ を用いた. このとき,

$$\begin{aligned}u(x,t;p_1,\theta_1) + u(x,t;p_2,\theta_2) &= 2\frac{\partial^2}{\partial x^2}\log(1 + a(\theta_1)e^{(p_1(x-(p_1)^2t))})(1 + a(\theta_2)e^{(p_2(x-(p_2)^2t))}) \\ &= 2\frac{\partial^2}{\partial x^2}\log(1 + a(\theta_1)e^{(p_1(x-(p_1)^2t))} + a(\theta_2)e^{(p_2(x-(p_2)^2t))} + a(\theta_1)a(\theta_2)e^{(p_1+p_2)x-(p_1^3+p_2^3)t})\end{aligned}$$

となる. ここで, $a(\theta_1)a(\theta_2)$ を $a(\theta_1)a(\theta_2)c$ に置き換えて KdV 方程式に代入すると,

$$c = \left(\frac{p_1 - p_2}{p_1 + p_2}\right)^2$$

が得られる. よって,

$$u(x,t;p_1,\theta_1) + u(x,t;p_2,\theta_2) = 2\frac{\partial^2}{\partial x^2}\log(1 + a(\theta_1)e^{(p_1(x-(p_1)^2t))} + a(\theta_2)e^{(p_2(x-(p_2)^2t))} + \left(\frac{p_1 - p_2}{p_1 + p_2}\right)^2 e^{(p_1+p_2)x-(p_1^3+p_2^3)t})$$

となる.

この式において, $p_1 = 1$, $p_2 = 2$ とすると,

図 28: 2-ソリトン解

背の高いソリトンが背の低いソリトンを追いかけ, $t = 0$ 付近で追いつき, そのまま追い抜かしてしまう事が見て取れる. これはソリトンが 2 つあるので, 2-ソリトン解と呼ばれる.

一般に n-ソリトン解は存在するが, ここでは省略する.

4.6 箱玉系

ソリトンの性質というのは, 離散化された箱玉系 (Box-Ball=System, BBS) にも現れる.

箱玉系とは, 有限もしくは無限に並んでいる箱の列に対し, それぞれの箱に 0 個または 1 個の玉が存在し, 次のようなルールに従って動く.

- 全ての玉を 1 回だけ動かす.
- 最も左にある玉を, 最も右側のからの箱に移動する.
- 残りの玉の中で最も左にある玉を, もっとも右側の箱に移動する.
- 全ての玉が動くまでこの手順を繰り返す.

これを実際に動かしてみるとこのようになります.

図

これを見ると, 確かに玉の列の集団はそのまま保たれており, ソリトンとなっていることが分かる.

4.7 KdV 方程式の超離散化

KdV 方程式を超離散化することを考える.

まずは, 時間変数 x を離散化する. このとき, KdV 方程式は可積分性という特性を持つため, これを保つように操作を行う必要がある. そこで, 可積分性をもち, 離散変数の極限操作で KdV 方程式に移行する差分方程式として, Lotka-Volterra 方程式:

$$\frac{db_n(t)}{dt} = b_n(t)[b_{n+1}(t) - b_{n-1}(t)]$$

を考える. この式に $b_n(t) = 1 + \varepsilon^2 u(\varepsilon(n + 2t), -\varepsilon^3 \frac{t}{3})$ を代入し, $\varepsilon \to +0$ の極限をとると, 確かに KdV 方程式が得られる.

次に時間変数 t を差分化する. ここでもやはり可積分性を保つように操作を行う必要があり, 次の偏差分方程式を得る.

$$\frac{c_n^{t+1}}{c_n^t} = \frac{1+\delta c_{n-1}^t}{1+\delta c_{n+1}^{t+1}},\ (n,t \in \mathbb{Z}). \tag{8}$$

これにおいて, $c_n^t = b_n(-\delta t)$ とおいて $\delta \to 0$ の極限をとると Lotka-Volterra 方程式が得られる.

これで, 独立変数の離散化が完了した. 式 (8) において, 充足変数変換 $c_n^t = e^{d_n^t}$ を行うと,

$$d_n^{t+1} - d_n^t = \log\left(\frac{1+\delta e^{d_{n-1}^t}}{1+\delta e^{d_{n+1}^{t+1}}}\right)$$

となり, $\delta = e^{-\frac{1}{\varepsilon}}$ とし, $d_n^t = \frac{f_n^t}{\varepsilon}$ とおくと,

$$f_n^{t+1} - f_n^t = e\log(1+e^{\frac{f_{n-1}^t}{\varepsilon}-1}) - e\log(1+e^{\frac{f_{n+1}^{t+1}}{\varepsilon}-1})$$

を得る. ここで $\varepsilon \to +0$ の極限をとると,

$$f_n^{t+1} - f_n^t = \max(0, f_{n-1}^t - 1) - \max(f_{n+1}^{t+1} - 1)$$

となる. $g_{x-y}^x = f_y^x$ などの変換をすると,

$$g_{n+1}^{t+1} - g_n^t = \max(0, g_n^{t+1} - 1) - \max(g_{n+1}^t - 1)$$

となり, g_n^t を整数値に限ることによって, セルオートマトンとみなすことができる.

以上から, KdV 方程式と直接対応するセルオートマトンが導かれた.

§5　終わりに

セルオートマトンを使って様々な問題をシミュレーションでき, また, 非線形方程式との対応について知ることができた. 今後は, 他の様々な現象を CA 化してみること, また, CA からのぎゃう超離散化なども考えていきたい.

参考文献

[1] 三村昌泰編. 現象数理学入門. 東京大学出版会. 2013. p79-108.

[2] 西成活裕. 渋滞学. 株式会社新潮社. 2007.

[3] 西成活裕. よくわかる渋滞学. 株式会社ナツメ社. 2009.

[4] 【第 8.3 回 Python 流体の数値計算】バーガース方程式の一般解 (コール・ホップ変換によって拡散方程式になる). 宇宙に入ったカマキリ. 2021 年 5 月 4 日. https://takun-physics.net/9790/. (最終閲覧日: 2024 年 11 月 3 日)

[5] 和泉嘉泰, 小口俊樹. 超離散化を用いた Burgers 方程式の制御. https://www.jstage.jst.go.jp/article/jacc/57/0/57_1015/_pdf. (最終閲覧日: 2024 年 11 月 3 日)

[6] @cotton-gluon, ソリトン〜計算機から生まれた数理物理学〜. Qiita. https://qiita.com/cotton-gluon/items/294976e801b68504a52f. (最終閲覧日: 2024 年 11 月 3 日)

[7] 時弘哲治. 箱玉系の数理一超離散ソリトンの世界. https://www.jstage.jst.go.jp/article/emath1996/2002/Autumn-Meeting1/2002_Autumn-Meeting1_32/_pdf. (最終閲覧日: 2024 年 11 月 3 日)

[8] 時弘哲治. 箱玉系の数理. https://www.mathsoc.jp/meeting/kikaku/2008aki/2008_aki_tokihiro.pdf. (最終閲覧日: 2024 年 11 月 3 日)