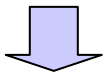
２．５　興奮性媒体

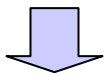
興奮-不応-受容（excited – refractory - receptive）の繰返し

隣接要素と相互作用　⇒　興奮可能な空間分布した要素
　⇒　静かな状態



“自己組織化”
＝空間分布したパターン（同心円，渦巻き波）の自発的形成

渦巻き円盤型銀河における星の形成，心臓組織の収縮，
拡散反応化学系，伝染病の流行，どろどろした形状物の成長



モデル化：　多重状態セルラ・オートマトン

最近接サイト間の相互作用 ——依存——→ 回復過程

(1)グリーンバーグ・ヘイスティングス
(2)サイクリック空間，　(3)ハッジポッジCA

［悪性不整脈］

- 生物系の電気的活動における興奮性媒体の振舞
- 心臓組織の収縮，心臓の筋肉がフィブリル化（顫動）
  → 停止＝突然死
- うっ血性心臓疾患の最終段階
  明らかに健康な個体の活発な活動中，後

2．5．1　グリーンバーグ・ヘイスティングスCA

- 神経（ニューロン）の興奮，ニューロ・ネットワークでの回復モデル
- 周期的境界条件，2次元正方格子
- 格子サイトの値＝1～r，1：興奮，r：休止，中間値：回復状態

［ニューロンの興奮ルール］

興奮　→　回復　→　休止 ——————————→ 興奮
最近接ニューロンが興奮

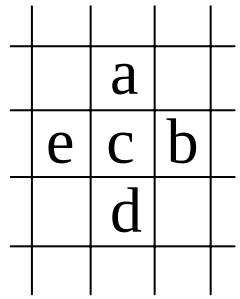
(1)最低1つの興奮した最近接ニューロン
→ 休止ニューロンが興奮

(2)興奮した最近接ニューロンがない
→　休止ニューロンは休止のまま

(3)回復状態にあるニューロン
→　次の回復状態（値が1増える）

プログラム

```
In[1] := NeuroExcitation[s_Integer, r_Integer, t_Integer] :=
          Module[{neuroNet, neuron},
初期配置
           neuronNet = Table[Random[Integer, {1, r}, {s}, {s}];
更新ルール　←　パターン・マッチ
           neuron[a_, b_, r, d_, e_] := 1/; MatchQ[1, a|b|d|e];
          neuron[a_, b_, r, d_, e_] := r ;
          neuron[_, _, c, _, _] := c+1 ;
          Attributes[neuron] = Listable ;
格子はt回更新される
         NestList[
           (neuron[RotateRight[#], Map[RotateLeft, #], #,
                    RotateLeft[#], Map[RotateRight, #]])&,
           neuroNet, t]
           ]
```



グラフィック出力

サイトの値に割り当てた色をつける. RasterArray

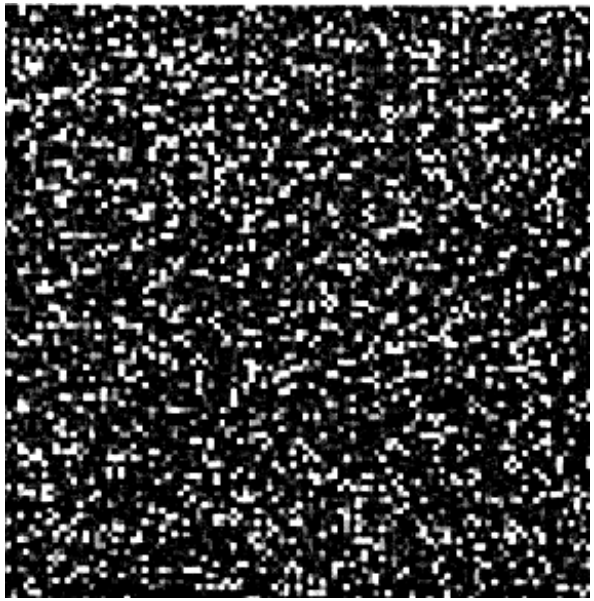
```
In[2] := ShowExcitation[list_, opts_ _ _] :=
              Module[{coloring, r = Max[list]},
                 coloring = Thread[Range[0, r] ->
                                 Map[Hue, Table[Random[ ], {r +1}]]];
                 Show[Graphics[RasterArray[list/. Coloring],
                       AspectRatio -> Automatic,
                       opts]]]
```

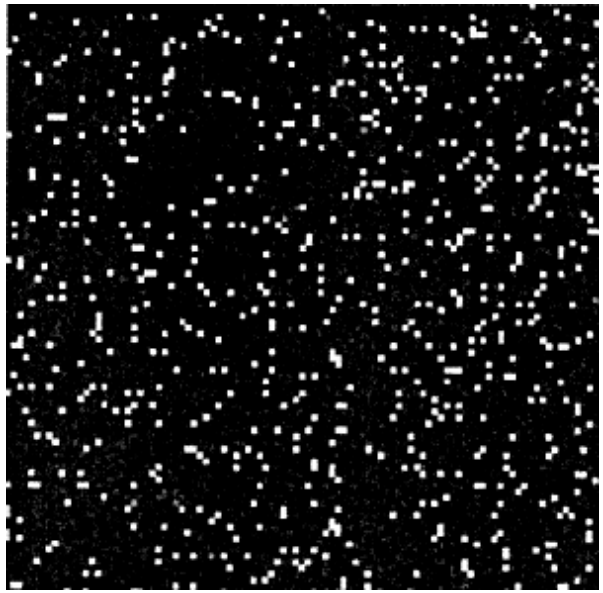
ニューロンの活動(活性化過程）

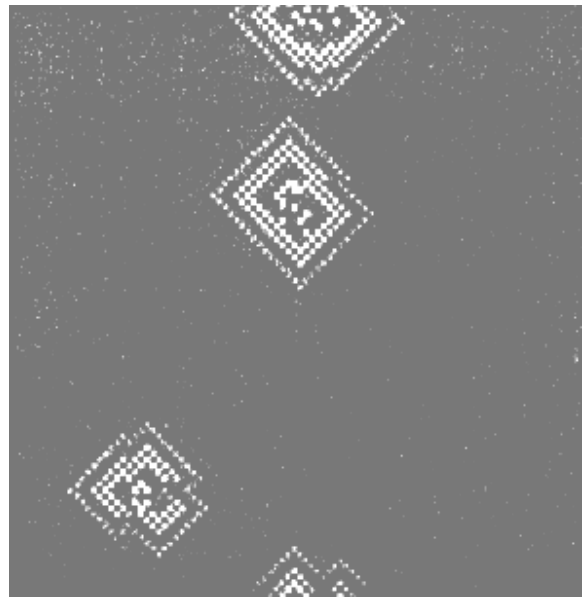
サイトの値:1〜16,　100×100正方格子

In[2] := ShowExcitation[NeuronExcitation[100, 16, 0]]

In[3] := ShowExcitation[NeuronExcitation[100, 16,

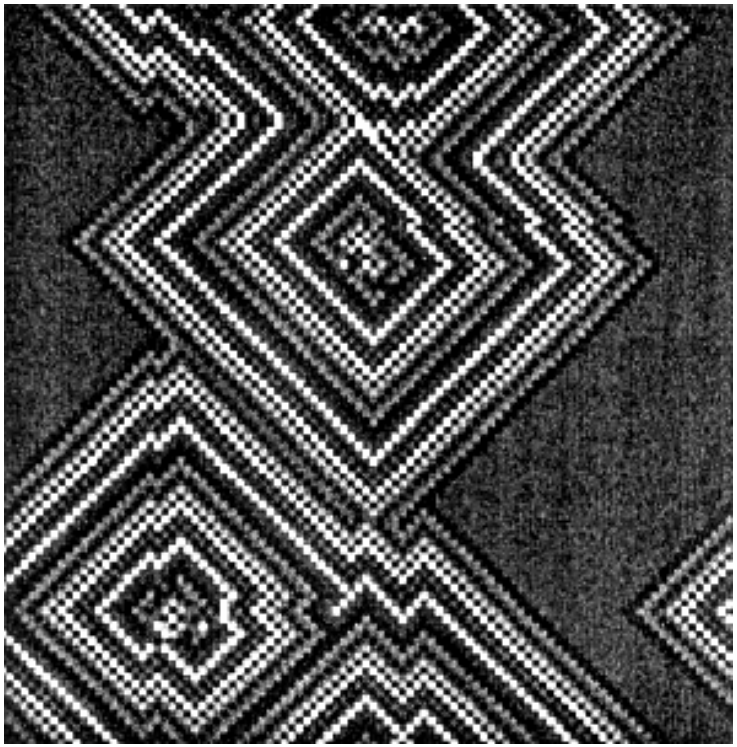






In[4] := ShowExcitation[NeuronExcitation[100, 16, 25]]

In[2] := ShowExcitation[NeuronExcitation[100, 16, 50]]



In[2] := ShowExcitation[NeuronExcitation[100, 16, 100]]

２．５．２　サイクリック空間

- グリーンバーグ･ヘイスティングスCAの回復過程を修正
- 格子サイトの値＝　0～(n – 1)

［ルール］

(1) 状態(n – 1)のセル

　　最近接セルのどれかが0　⇒　状態0となる.

(2)　状態　$c\ (0 \le c \le n-1)$　のセル

　　最近接セルのどれかが状態 c + 1　⇒　状態 c + 1

　　最近接セルに状態 c + 1がない　⇒　状態 c

プログラム

```
In[1] := Phases[s_Integer, n_Integer, t_Integer] :=
          Module[{debris, cyclicSpace},
```

初期配置，組織片(debris)

```
          debris = Table[Random[Integer, {0, n-1}], {s}, {s}] ;
```

更新ルール

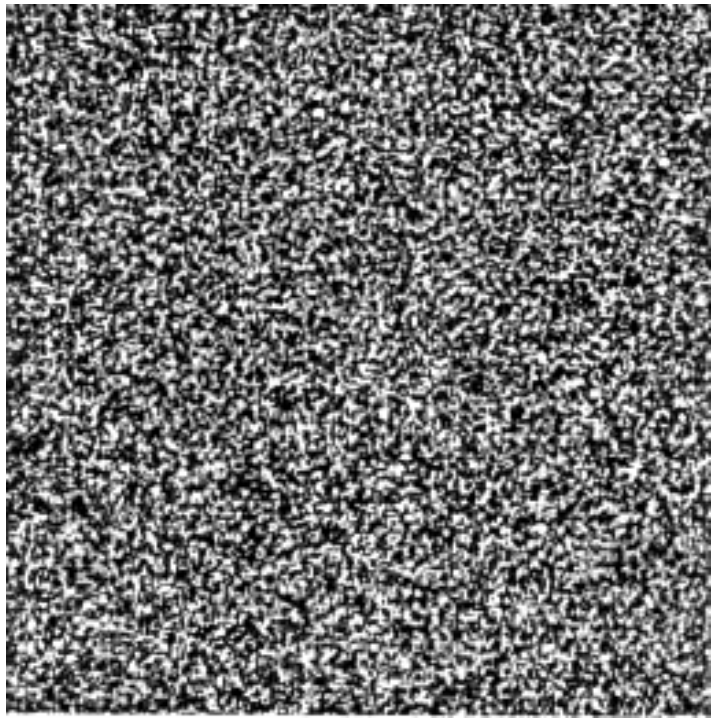
```
          cyclicSpace[a_, b_, (n-1), d_, e_] := 0/. MatchQ[0, a|b|d|e] ;
          cyclicSpace[a_, b_, c_, d_, e_] :=
                                   (c+1)/. MatchQ[(c+1), a|b|d|e] ;
          cyclicSpace[_, _, c_, _, _] := c ;
          Attributes[cyclicSpace] = Listable
```

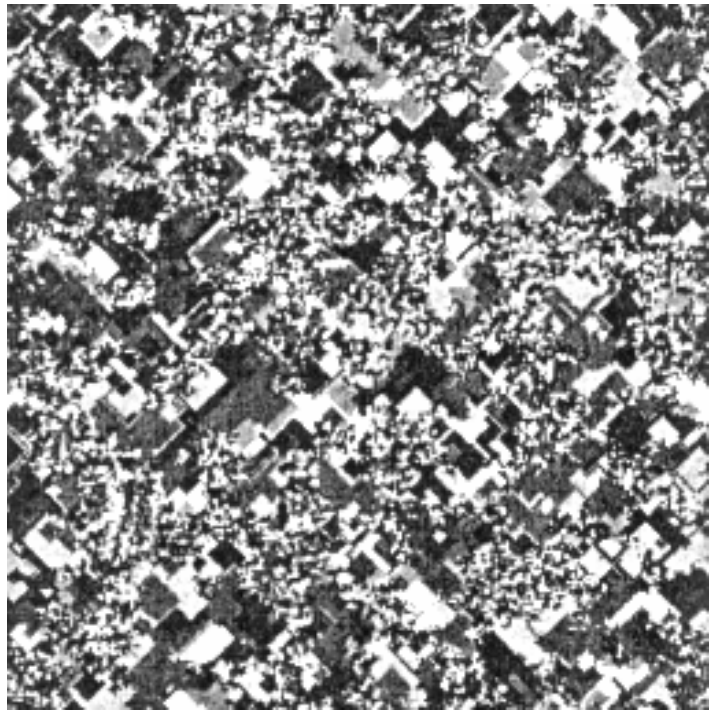
時間発展

```
          NestList[
              (cyclicSpace[RotateRight[#], Map[RotateLeft, #], #,
              RotateLeft[#], Map[RotateRight, #]])&,
              debris, t]]
```

1. 組織片（ランダム分散）

In[7] := ShowExcitation[Phases[256, 14, 1]] ;



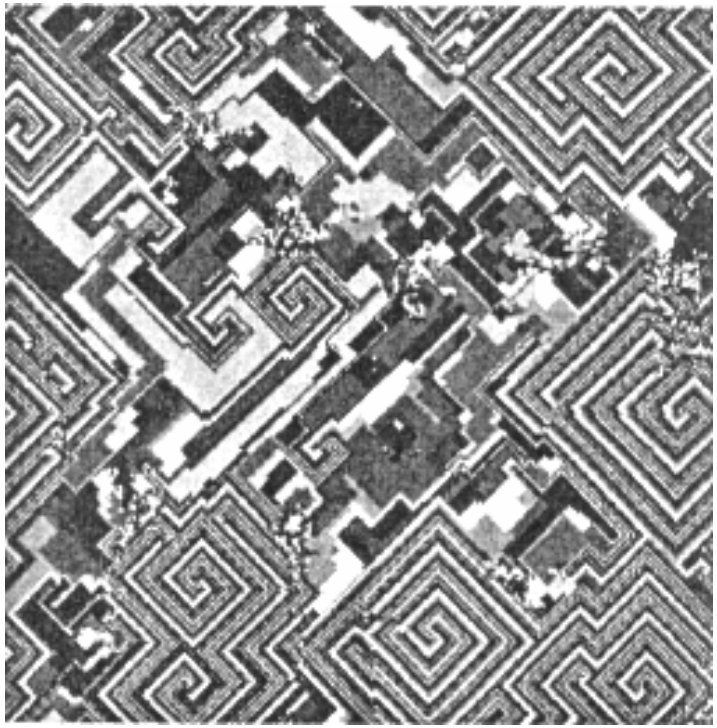


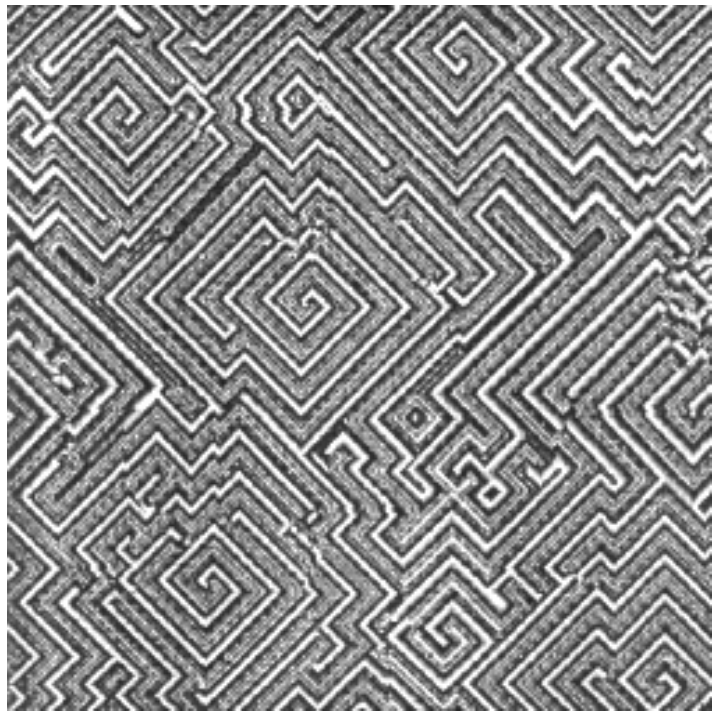
2. 飛沫（小型の領域）

In[8] :=ShowExcitation[Phases[256, 14, 100]] ;

3. 結晶の欠陥（格子欠陥，らせん転位，角張った渦巻き）

In[9] := ShowExcitation[Phases[256, 14, 200]] ;





4. デーモン（いくつかの渦巻き）

In[10] := ShowExcitation[Phases[256, 14, 500]] ;

２．５．３　ハッジポッジ・マシン

アラン・チューリング

生物学的形態発生（形態形成，形態生成）

→　化学系におけるパターン発展

ハッジポッジ(hodgepodge，不均一混合, hochepot(フランス語))マシン

自動触媒化学反応をモデル化

触媒の存在で2種類以上の化合物合成，解離および再合成

（例）

(1) CO とO_2によるCO_2の合成

ガスは散布したパラジウム・フリスタレットの表面に吸着

(2)ベロゾフ・ジャボチンスキー反応

マロン酸が鉄やセリウムの存在で臭化カリウムで酸化

周期的境界，ノイマン型近傍

格子サイトの値＝0~r,　0：健康，r：病気，1~(r-1)：感染

［ルール］

(1)セルが病気なら健康に（治癒）

(2)セルが健康なら感染（感染）

セルの値は $$Min\left[r, Floor\left[\frac{n_c}{k1} \right] + Floor\left[\frac{n_r}{k2} \right] \right]$$

n_c: 感染している最近接セルの数，

n_r: 病気である最近接セルの数，$k1, k2$: 定数

(3)セルが感染しているならさらに感染

$$Min\left[r, g + Floor\left[\frac{n_n}{n_c} \right] \right]$$

n_n: 近傍セルの値の和，　g: 感染速度定数

プログラム

```
In[1] := Hodgepodge[r_Integer, s_Integer, k1_, k2_,
                    g_Integer, t_Integer] :=
  Module[{initconfig, ngbrsVon, sick},
```

初期配置

```
    initconfig = Table[Random[Integer, {0, r}], {s}, {s}] ;
```

近傍値

```
    ngbrsVon[mat_] = Apply[Plus, Map[RotateRight[mat, #]&,
                                     {{1, 0}, {-1, 0}, {0, 1}, {0, -1}}]];
```

サイトの更新値（ルール）

```
    sick[r, _, _, _] := 0 ;
    sick[0, x_, y_, _] := Min[r, Floor[y/k1]+Floor[x/k2]];
    sick[c_, x_, y_, _] := Min[r, g+Floor[(c+(r x)+z)/(y+1)]];
    Attributes[sick] = Listable ;
```

計算を止める条件

論理行列（1：病気のセル，0：感染or健康のセル）

FixedPointList[(sick[#, ngbsVon[Floor[N[#/r]]],

最近接セルの病気のセルの数

論理行列（1：感染しているセル，0：病気or健康のセル）

ngbsVon[Sign[Mod[#, r]]],

最近接セルの感染したセルの数

0~(r-1)（0以外：感染の度合，0：病気か健康であるセル）

ngbsVon[Mod[#, r]])&,

感染している最近接セルの値の和

initconfig, t]]

感染定数gの値を変化　⇒　異なる4つの波のパターン

1. 短い距離を流れ，その後死滅する波
2. 色々な幅を持つ丸い帯状に伝わる波
3. だいたい一様に丸い帯状に伝わる波
4. 渦巻きパターンで広がる波

長時間感染しているセルの割合

gの小さな値では全てのセルは急速に健康

1. 健康なセルが不規則にランダムに出現，ほとんどのセルがほとんどの時間で感染
2. 多くの針状の健康なセルで分離された感染セル，その数に変化なし
3. ほとんどすべての感染か，まったくないのどちらか
4. 感染しているセルの割合は約75%の周りで揺らぐ。

２．６　砂山の崩落

自己組織臨界性（self-organized criticality）

複雑系の振舞，準安定条件，自然に発展，激変に導く連鎖反応

（例）

地学　：　地震，火山爆発，眺望の形成

天文学：　パルサーのグリッチ（自転周期が突然増加する現象）

太陽フレアー

経済学：　株式相場の揺らぎ

生体系：　進化，　　流体力学：　乱流

rf.

自己組織化（self-organization, 自己体制化，自己秩序化

自律形成）

モデル

２次元正方格子，吸収端境界条件

サイトの値＝1~8,　境界サイトの値＝0

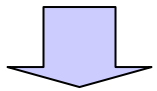
5以上の値を持つサイト:“不安定（top-heavy）”

不安定なサイトがなくなるまで更新

ルール

不安定サイト（5以上の値をもつサイト）

→　そのサイトの値を4減らし，ノイマン近傍の4つの最近接サイトの値を1づつ増やす



(1) 不安定なサイトの値は最近接の不安定サイトの数から4引いた数だけ増加する.

（2）他のサイトの値は不安定サイトの数だけ増加する.

プログラム

```
In[1] := Sandpile[s_, m_] :=
          Module[{absorbBC, landscape, topHeavyNgbrs, update},
```

初期配置（内側サイトの値＝1~4，境界サイトの値＝0）

```
           absorbBC =
            (Prepend[Append[Map[Prepend[Append[#, 0], 0]&, #],
                Table[0, {Length[#] + 2}]],
              Table[0, {Length[#] + 2}]])&;
           landscape = absorbBC[Table[Random[Integer, {1, 4}],
                                                    {s}, {s}]];
```

内側サイト：　サイトの1つが5の値まで1つづつ増加

```
           While[Max[landscape < 5,
                 randx = Random[Integer, {2, s + 1}, {2, s + 1}];
                 randy = Random[Integer, {2, s + 1}, {2, s + 1}];
                 landscape[[randx, randy]] + +
                 ];
```

不安定な最近接サイトの数

```
topHeavyNgbrs[mat_] :=
    Apply[Plus, Map[RotateRight[Floor[mat/5.0, #]&,
                                {{-1, 0}, {0, -1}, {0, 1}, {1, 0}}]
```

崩落ルール

サイトの値　　不安定な最近接サイトの数

```
update[r_, t_] := r + t /; r < 5
update [r_, t_] := r – 4 + t
update [r_, t_] := 0
Attributes[update] = Listable;
```

終了条件：（1）m個更新か (2) updateが変化しなくなる

```
FixedPointList[update[#, topHeavyNgbrs[#]]&, landscape,
m]
]
```