卒業論文

砂丘のパターン形成のシミュレーション

須藤光昭

青山学院大学・理工学部・物理学科

羽田野研究室

２００２年度

砂丘のパターン形成の
シミュレーション

須藤光昭
羽田野研究室

平成15 年3 月19 日

概 要

自然界には、砂（粉流体）によって形成される様々なパターンの砂丘が存在する。それらの砂丘の形成過程をモデル化し、そのパターンをシミュレーションによって再現するのが本研究の目的である。風向と直行した方向に出来る「横列砂丘」と、弓形の形状を持つ「バルハン砂丘」を、Werner model を用いてシミュレーションで再現した。砂丘の高さと砂丘縞の波長の関係をシミュレーション結果から求め、実測値と比較して対応関係を議論する。

目 次

1　砂丘とは

1.1　砂の作る地形

砂の形成する砂丘や風紋といった地形は、普段あまり接する機会も少なく、なじみの薄い存在であるかも知れない。しかし、それらの出現機構は単純で、砂粒（粉流体）と風の力との相互作用のみによって多種多様な地形が現れることが知られている。

本節では、そうした地形を構成する物質である粉流体の性質と、それらが実際どのような運動を経て風紋や砂丘といった地形を形成するのかを述べる。

1.2　粉流体

砂丘を構成する粉流体といわれる物質は、典型的なものでは直径 0.1 ミリ程度の大きさを持つ粒子の集合体である。この世の物質のほとんどが固体・液体・気体に分類できる。もちろん、砂丘のような地形を形成している粉流体も、粒子 1 粒 1 粒に着目すれば、固体としての性質を示す。しかし、それらが集まった物の全体としては、極めて特殊な振る舞いを示す。

容器に粉流体を入れれば、粉流体は「固体」としてしっかり固まる。一方、流れ、拡散するといった「液体」としての性質、吹き上げられれば「気体」としての性質をも併せ持つ [1]。そうした意味では、まだ物理学にとって未解明の部分の多い対象である。

1.3　砂の運動

風の力によって生じる砂粒の運動は、以下の三種類であることが、R.A.Bagnord によって半世紀以上前に指摘されている（図 1）:

浮遊 強い風のもとで、砂が風に乗って長距離を飛ぶ運動。これは、風紋の大きさ（数センチ）に比べて大きいので、以下では考慮しないことにする。

跳躍 揚力がさらに強くなると砂粒が浮き上がり、また、風に引きずり上げられて砂粒が砂表面から飛び出す。飛び出した砂粒は、いずれ再び風下方向のどこかの砂表面に落下する。

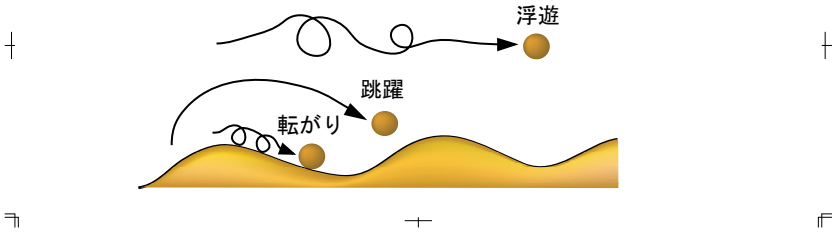


図 1: 砂の運動。

転がり 作用する重力が揚力によって弱められ、無重力の時と同じように粉流体は砂の表面から離れる。そして、風に引きずられる形で転がる。

本研究では、上述の「跳躍」と「転がり」の 2 つの運動形態をコンピューター上で数値計算することによって、風紋などの地形の出現機構を探る。

1.4 風紋

風紋を形成するためには、次の条件が必要である。

- 砂表面が濡れていたり水分を含んでいない。
- 風紋は、風速がある条件を満たさないと出現しない。
- 風紋の縞模様の間隔は風速に比例する。

ここで、風紋を形成する風速の条件とは 4〜9[m/s] であることが知られている。特に 5〜7[m/s] では美しい風紋が形成される。上記の 3 つの風紋の形成条件は、風紋を作る砂の運動について以下の二つの束縛条件があることを示している。

第一は、砂表面が水分を含んでいないことにより「転がり」が重要な条件になることである。砂に含まれる水分は、砂粒同士の粘着力を増加させ「転がり」を阻害する働きをもつ。この働きが無いことが、風紋を作るための条件である。

第二は、強風によって生じる「浮遊」は、風紋の形成とは直接関係ないとしてよいことである。風紋は限定された風速でしか形成されないため、「浮遊」の効果は重要ではない。風紋のシミュレーションを行うときの最低の条件として、「跳躍」と「転がり」の手続きをルール化すれば良いことを示している。

1.5 砂丘

砂丘は、パターン形成の観点から言えば風紋と類似している。本研究では、風紋の規模を大きくしたものを砂丘と定義する。砂丘の形態を支配する要因としては、 まず、風の強さと風向、あるいはその安定度をあげることができる。また砂の粒径や供給量も重要な要因である。さらに降雨の量や季節配分、植被の状態なども関係する。

いったん砂丘地形がつくられると、その影響を受けて地表に近い層の気流が変わり、その結果として砂丘がますます成長する。何らかの理由でいったん植生が入り込むと、それが接地風速を弱め、また根網が砂層を固め、表面に腐植層が形成される (固定砂丘となる) という変化もある。このように砂丘と砂丘形成に関係する要因との相互作用はかなり複雑である。

風紋が風下へ動いていく速さは、風速が強ければ速くなり、風紋が大きくなれば遅くなる。つまり、風紋の移動速度は風速に比例し、大きさに反比例する。このため大小の風紋が混じっていると、小さい風紋の山は大きい風紋より移動が速いために大きい風紋の山に追いつく。こうして小さい山は大きい山に合体して一層大きな風紋となる。次第にその作用が加算されて風紋は大型に育っていき、ついに砂丘になる。

1.6 論文の構成

本研究では、主に図 2 のような横列砂丘を、コンピューター上で再現する。まず第 2 節でシミュレーションに用いる Werner モデルを詳しく述べる。第 3 節では、シミュレーションによって再現された横列砂丘を示す。第 4 節では、別の形のバルハン砂丘の再現を目指す。その中で、Shadow zone という概念を導入する事によってバルハン砂丘も再現できることを示す。第 5 節では Werner モデルの特性を 2 次元モデルを導入する事によって解析する。第 6 節では 2 次元モデルによって得られたデータと実測値

図 2: 横列砂丘。

との対応関係を調べ、最後に第 7 節でまとめ考察を行う。

2　モデル化

前節で述べたように、砂粒（粉流体）の形成する地形は、砂粒と風との相互作用によるものである。本節では、風紋形成といった現象をどのようなモデルを用いて再現したか、そして、前節で定義した「跳躍」と「転がり」の運動形態をどのように計算機上で記述し、計算を行ったかについて述べる。

2.1　Werner モデル

実際、砂粒 1 粒 1 粒の運動と衝突、さらに、風との相互作用を計算するのはかなり困難である。そこで、この現象の本質を突くような簡単なモデルを導入する。

今回用いたモデルは、「Werner モデル」と呼ばれているものである。まず、2 次元の砂表面を碁盤目状に切り分け、各桝目の高さを定義する。ここでは、砂粒一つ一つがどのように積み重なっているかは無視し、砂表面の高さのみを問題とする。よって、桝目 (i, j) サイトの砂の高さは、単に $h(i, j)$ と定義する。尚、風は x 軸の正方向にのみ吹くものとするので、砂は x の正方向にのみ飛ぶ。各桝目に一定の揺らぎを与えた初期状態か

ら開始し、砂丘生成のシミュレーションを行う。

2.2 跳躍

跳躍は、ある桝目 (i, j) から別の桝目 $(i, j + L)$ へ砂をある一定量移動させることとする（図 3)。この時に移動する砂の量は、ある一定量 q であるとする。また、砂の飛ぶ距離を L とする。砂は風下方向にしか飛ばないので、L は常に正の値をとる。以上より、跳躍の起こった桝目とその砂が落下した桝目の高さの変化は、

$$\begin{cases} h(i, j) = h(i, j) - q \\ h(i + L, j) = h(i + L, j) + q \end{cases} \tag{1}$$

である。

ここで、跳躍距離 L は、

1. 風速のより大きいところの砂が、より遠くへ飛び、

2. 他より高いところの砂が、より遠くへ飛ぶ、

といった 2 点を考慮して、

$$L = L_0 + bh \tag{2}$$

とする。L_0 は風速に対応したパラメーターとし、b はパラメーターとして小さい正の定数を与える。また、跳躍距離 L は周囲の桝目の高さが影響する事を考慮して、跳躍の発生する桝目はランダムに選ぶ。そして、すべての桝目の数と同じ回数だけ、跳躍を起こしたときを、時間の 1 ステップとする。すなわち、1 ステップ中では、1 つの桝目に平均 1 回の跳躍が発生する。 次に、「転がり」によって各桝目の高さがどのように変化するかを考える。ある桝目が周囲の桝目より高い時、その桝目の中の砂からある量を取り去って周囲の桝目に分配することにより、「転がり」を表現する（図 4)。各桝目から転がり出る砂の量は、その桝目の高さに比例するだろう。この比例定数を D として、転がり出る砂の量を Dh と定義する。この転がり出た砂の量は、周囲に等方的に分配する。ただし、辺で接している桝目より角で接している桝目の方が分配される砂の量は少ないと考えられる。よって、その比を 1:2 とした。以上より、桝目 (i, j) で

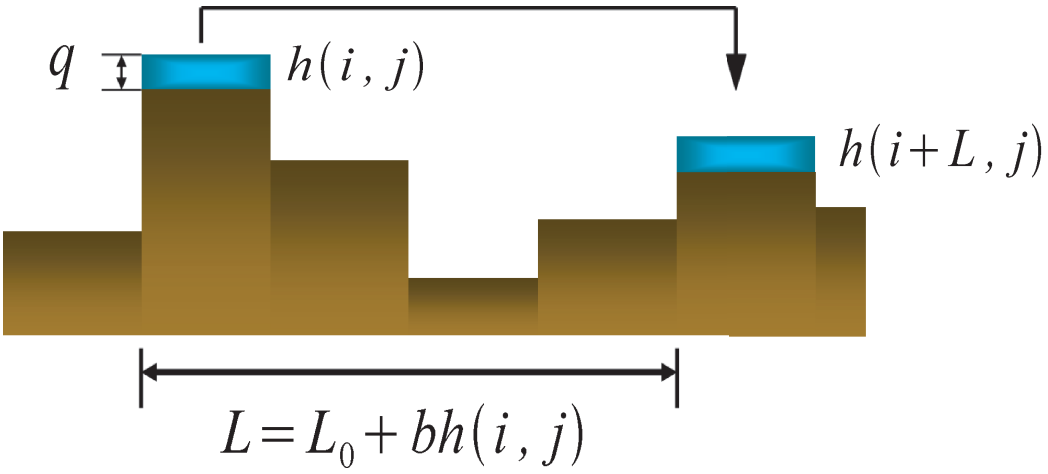


図 3: 跳躍

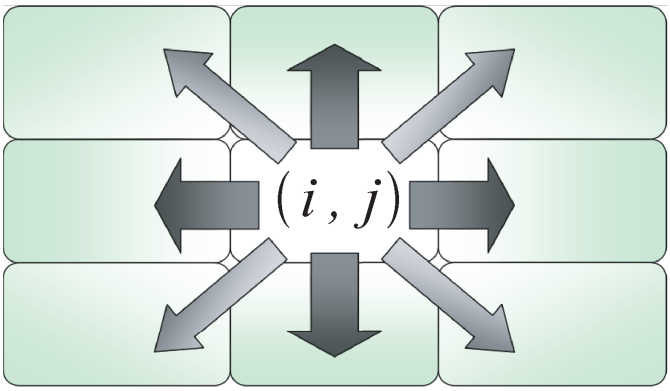


図 4: 転がり

の高さの変化は、

$$
\begin{aligned}
& h(i,j) = (1-D)h(i,j) \\
& + D[\frac{1}{6}(h(i+1,j) + h(i-1,j) + h(i,j+1) + h(i,j-1)) \\
& + \frac{1}{12}(h(i+1,j+1) + h(i+1,j-1) \\
& + h(i-1,j+1) + h(i-1,j-1))] \qquad (3)
\end{aligned}
$$

となる。以上で示した「跳躍」と「転がり」の式を繰り返すことによって、風紋形成のシミュレーションを行ってゆく。

3　3次元モデルによる風紋の再現

前節で定義されたモデルのシミュレーションによって、横列砂丘が再現されることを報告する。

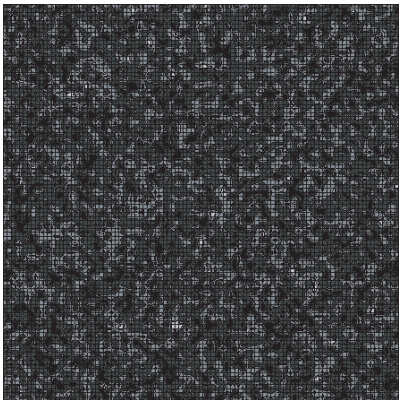
3.1　各ステップにおける風紋の様子

先に述べたアルゴリズムによって、ステップ数を 0 から 5000 まで変化させて実行した風紋形成シミュレーションの結果を図 5 に示す。系のサイズは 100×100 で、各桝目 $h(i,j)$ の初期値として [2.85,3.15] の一様乱数を振り分けた。境界は全て周期境界条件を用いた。計算に必要なパラメーターは $q = 0.1$, $L_0 = 3.0$, $b = 1.0$ である。

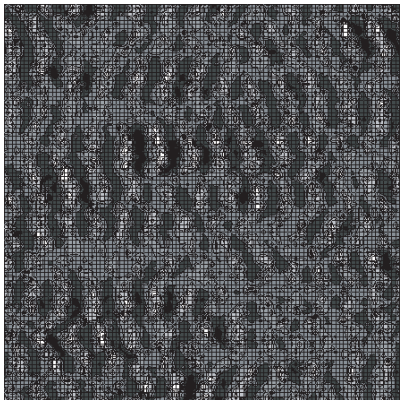
図 5 は、初期状態ではランダムな揺らぎの空間分布を持っていた地表面が、ステップ数の増加と共に、「跳躍」と「転がり」によって風紋を作ることを示している。およそ 3000 ステップ経過後、風紋の波長は収束している事が確認できる。

3.2　風紋の風速への依存性

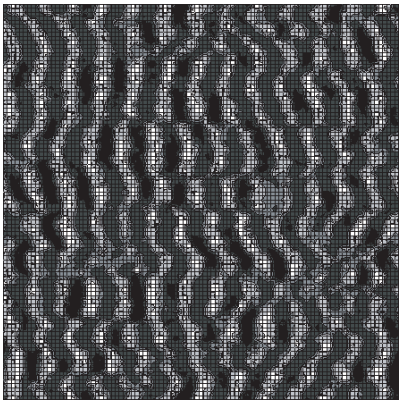
風速に対応するパラメーター L_0 を変化させてシミュレーションを行った結果を、図 6 に示す。なお、パターン形成までに要したステップ数は、全て 5000 ステップとした。計算に必要なパラメーターは $q = 0.1$, $b = 1.0$ である。この結果からも、より低い風速では、風紋の波長は密に、より高い風速では疎になる傾向がある事が確認できる。



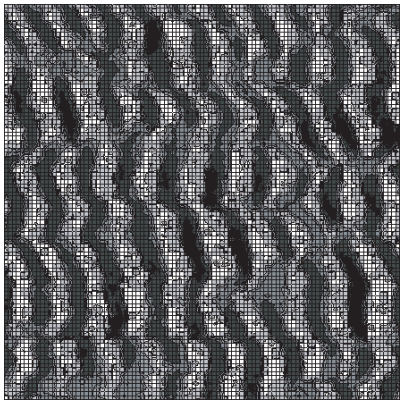
(a) 0 ステップ目;



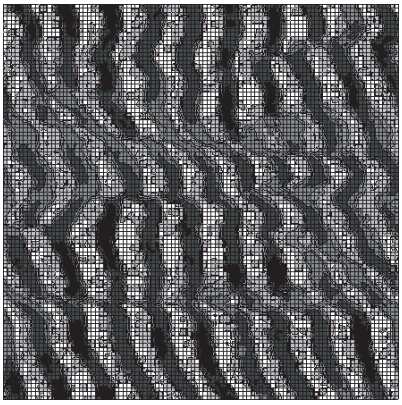
(b) 50 ステップ目;



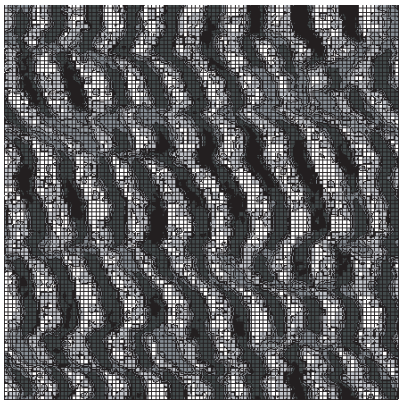
(c) 100 ステップ目;



(d) 1000 ステップ目;



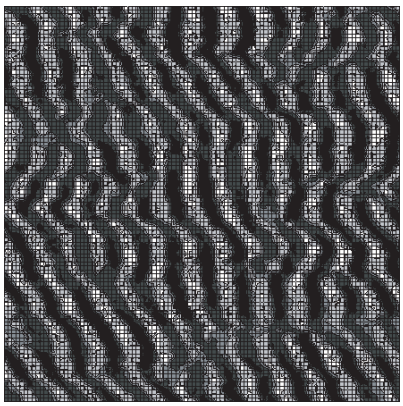
(e) 3000 ステップ目;



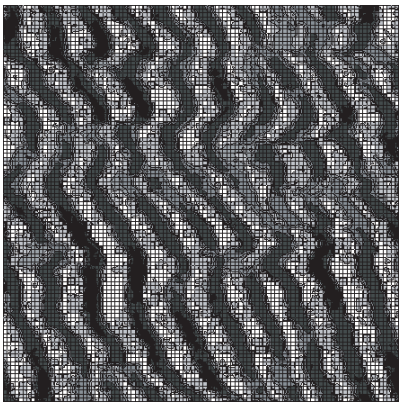
(f) 5000 ステップ目;

図 5: 風紋の時間変化のシミュレーション結果。

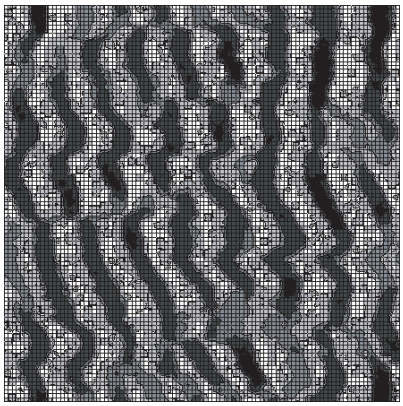
(a) L_0=1.0 のとき;



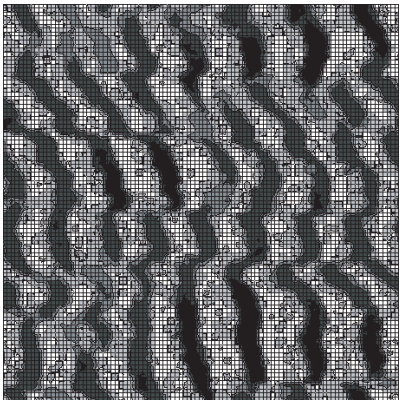
(b) L_0=2.0 のとき;



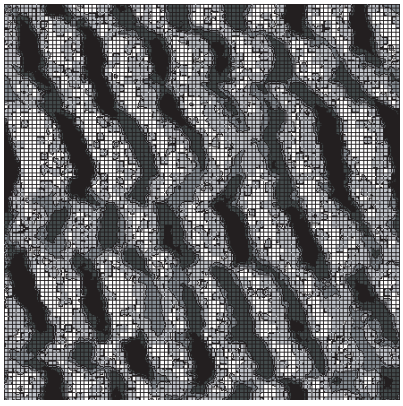
(c) L_0=3.0 のとき;



(d) L_0=4.0 のとき;



(e) L_0=5.0 のとき;



(f) L_0=6.0 のとき;

図 6: 風紋の風速への依存性。

図 7: バルハン砂丘。

4　バルハン砂丘の生成のメカニズム

この節では、第 2 節の Werner モデルに新たな条件を付け加えて、三日月型のバルハン砂丘の再現を目的とする。

4.1　バルハン砂丘について

バルハン砂丘とは、三日月状砂丘とも呼ばれ、図 7 のような形状を持つものを言う。砂の供給が少なく、卓越風向の明確な場合によくでる。三日月の開いた方を風下側に向けている。風上側の傾斜は緩やかで、下層では 5〜10 度、頂上部で 10〜15 度である。風下側は 30〜33 度 (安息角) の急斜面で、風紋と同じである。砂が風上側の斜面に沿って吹き上げられ、峰の頂上から風下側の急斜面に転がり落ちる。小さい粒子は遠方に飛ぶか、バルハンの底面付近に落下する。風の流線のように風下側の下層部は渦を巻き、頂上の風下直後では弱い逆風の吹上げ風となっている。この風と頂上を越えた風との作用で尖るわけである。高さは様々あるが、5〜50[m] のものが多く、波長は 100〜1000[m] である。

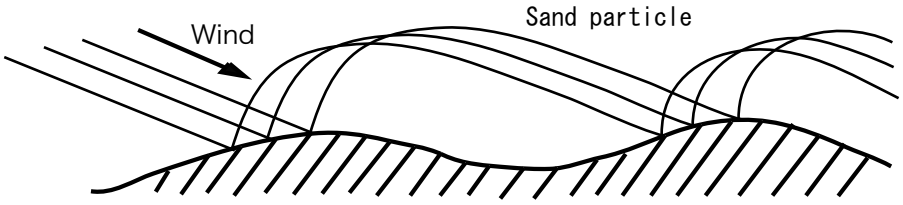


図 8: 風による砂の運動。

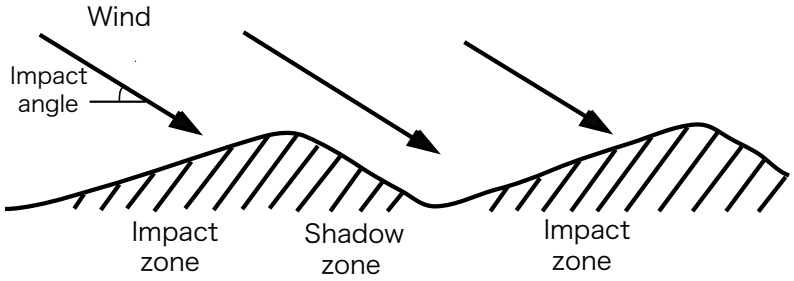


図 9: Impact zone と Shadow zone

4.2 Shadow zone

バルハン砂丘の生成において、重要な要因と考えられるのが、Shadow zone という概念である。風がある角度を持って吹き付けてくる際、砂粒は図 8 のように運動する。その結果、風の影響を受けて砂が運動する「Impact zone」と、風の影響を受けず砂が運動しない「Shadow zone」の、2 つの領域が生まれる（図 9)。

この節では、前節で定義したモデルにこの Shadow zone を取り入れる。モデルの中では、「もし隣り合う風上のサイトの高さがそのサイトよりも高ければ、そのサイトの砂は飛ばない」という条件を付加することにより表現したする。

4.3 シミュレーション結果

シミュレーション結果を、図 10 に示す。図からもわかるように、3000 ステップ程度経過すると、きれいな三日月型の形状が再現できた。ただ、それ以上ステップ数を増加させると、今回のモデルのスケールからは確認が難しいが、1 つ 1 つの砂丘が結合し、より大きな砂丘を形成している

と考えられる。

5　2次元モデルについて

5.1　2次元モデルの概要

これまでは、3次元モデルで議論してきたが、それではこのモデルにおける振る舞いを解析するに当たって、計算機のメモリの問題や、ある結果において波長や高さの一意的な定義が困難であるなど、不都合な点が多い。そこで次に2次元モデルを導入することにした。

基本的な計算手順は、3次元モデルと同様である。各サイト $h(i,j)$ に [48.5,51.5] の間の一様ランダムな実数を与えた初期状態から数値計算を始めた。

2次元モデルにおける「波長」は、以下のように定義した。初期状態として、高さ50近傍で揺らぎを与えた状態から開始する。パターンが収束したら、50よりも高い山を選ぶ。隣り合う山の間隔を1波長と定義することにする。

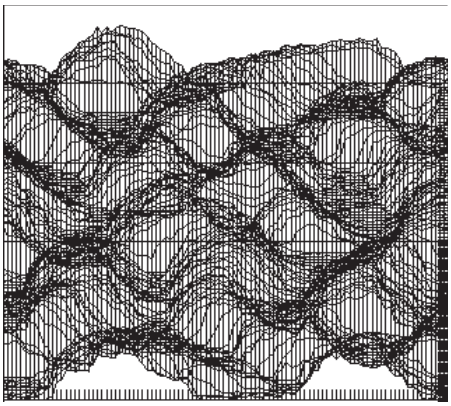
また、シミュレーションによって形成される砂丘の高さは、均一ではない。それを測定するため、「高さ」は以下のように定義した。あるステップ経過後の各サイトの高さの出力結果を、降順整列させ、上位1%の数値を平均した値をもって、その砂丘の「高さ」と定義する。

5.2　2次元モデルのシミュレーション結果

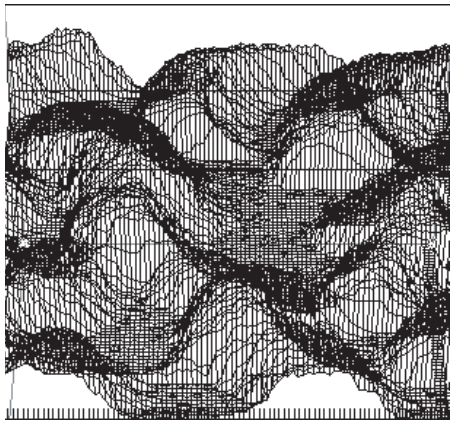
ステップ数と波長の関係を、風速に対応するパラメーター L_0 を変化させた時の結果を図11に示す。系のサイズは10000で、計算に必要なパラメーターは $q = 0.1, b = 1.0$ である。

より低い風速の方が、より少ないステップ数で、より小さい波長に収束することが確認できる。なお、ある収束値に達した後も、一定の揺らぎは残る。

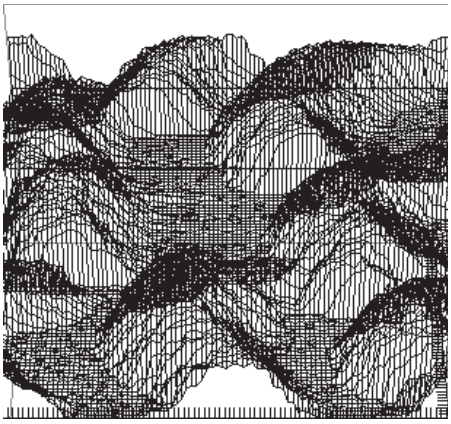
風速 L_0 と波長 λ の収束値の関係の結果を図12に示す。これより、風速と風紋の波長は、ほぼ比例関係にあることがわかる。なお波長には、収束後の揺らぎ幅に対応した誤差棒をつけた。



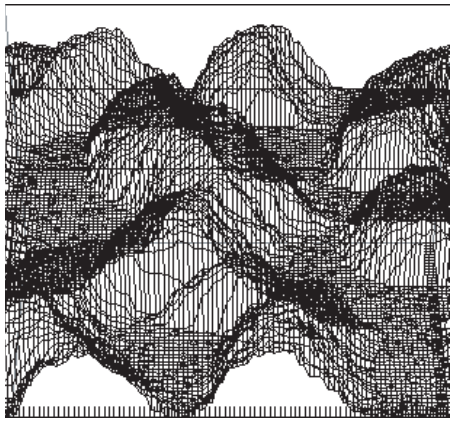
(a) 1000 ステップ経過後;



(b) 2000 ステップ経過後;



(c) 3000 ステップ経過後;



(d) 5000 ステップ経過後;

図 10: バルハン砂丘の時間変化。

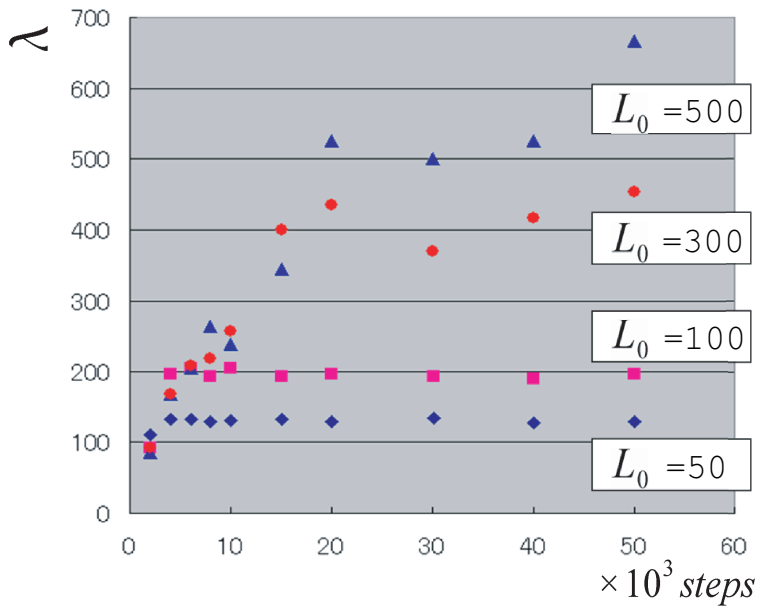


図 11: 砂丘の波長 λ がステップ数とともに収束する様子。

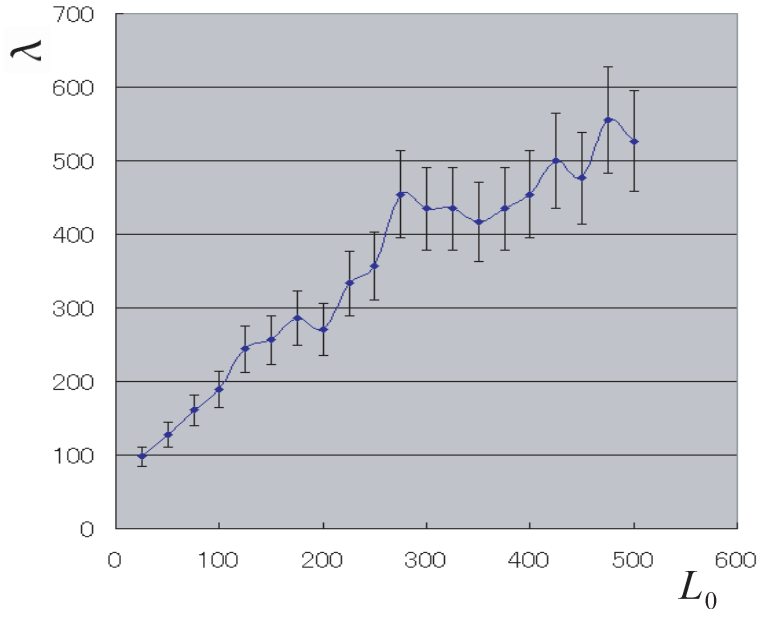


図 12: 風速 L_0 と波長 λ（収束値）の関係。

6　実測値との対応関係

最後に、この節ではシミュレーション結果と実測値を比較することによって、Werner モデルの桝目のスケールを決定する。

6.1　風紋の「高さ」と「波長」の関係

風紋の「高さ」と「波長」の関係は、Lancaster らの観測により、両対数プロット上において比例関係にあることが知られている。一般に、様々なスケールにおいてこの関係が成り立つことが知られている。その実測値のデータに、シミュレーションによって求めたデータを対応させる事により、モデルにおけるサイトのスケールを求めた。

方法としては、まずコンピューター上で、風速に対応するパラメーター L_0 を変化させる。各風速における高さと波長を求め、その傾きをプロットする。次に、それらのデータにある実数を掛けてデータの直線にその直線をフィットさせる。これをもって、モデルにおける 1 サイトが実際のどれくらいのスケールに対応しているのかを求めた。

6.2　実測値との対応関係について

前述した Shadoe zone の概念を 2 次元モデルに導入して、シミュレーションを行い、風速に対応するパラメーター L_0 の値を 5.0 から 300 まで変化させて砂丘の高さを測定した。

その結果により得られた直線において、波長を 10^{-3}、高さを 10^{-4} 倍すると実測値にほぼフィットする事がわかった（図 13）。ここで、波長を 10^{-3} 倍したのは、即ち 1 サイトを 10^{-3}[m] に対応させた事と同義である。よって、モデルにおける 1 サイトは 10^{-3}[m] に対応する事がわかる。

7　結論

本研究では、砂丘の出現機構を Werner モデルを用いて再現した。その上で、モデルの特性について調べ、最後に実際のスケールとの対応関係について調べた。

その結果、以下のことがわかった。まず、一定の風速において、風紋の波長の収束性が確認できた（図 11）。また、風速と波長はほぼ比例関係

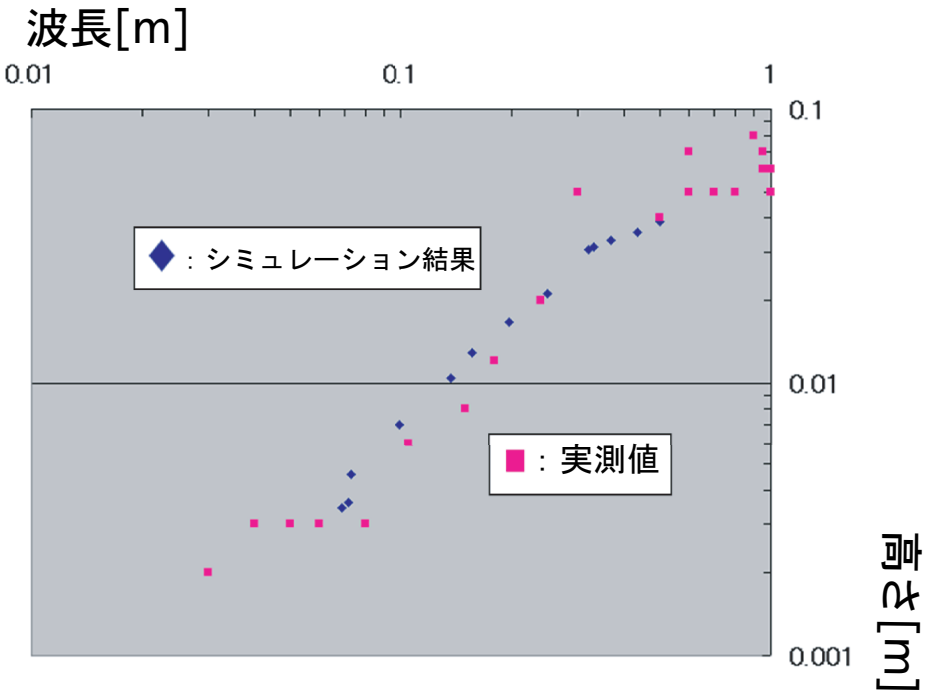


図 13: 高さと波長の関係（参考文献 [2] 参照）。

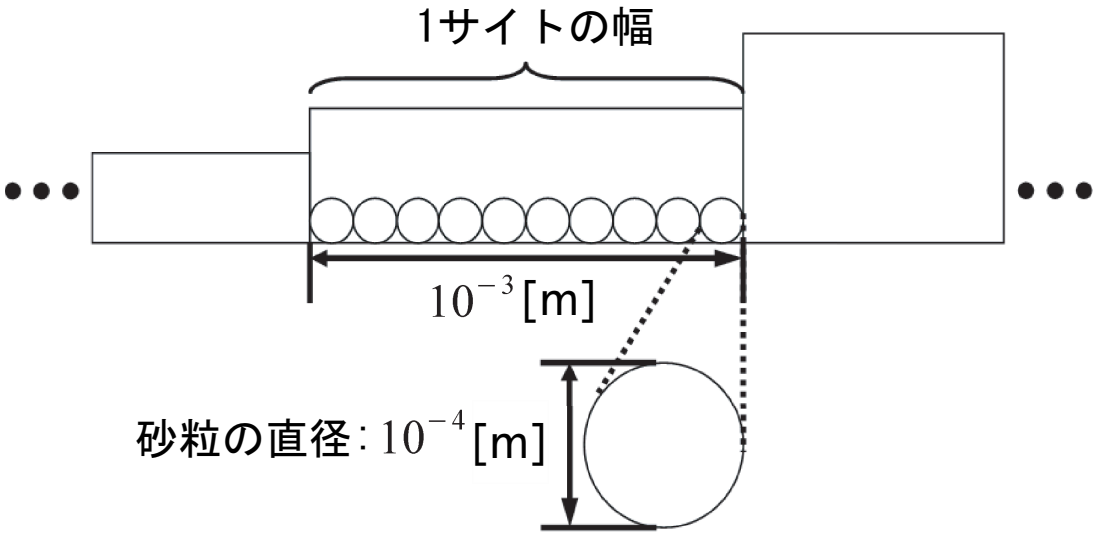


図 14: 1 サイトの幅と砂粒の大きさの比較。

にある（図 12）。そして、Werner モデルにおける 1 サイトのスケールは、およそ 10^{-3}[m] に対応している。一般的な砂粒の直径が 10^{-4}[m] なので、この 10^{-3}[m] という値は、砂粒の 10 倍に相当する（図 14）。もしこの桝目の大きさが小さすぎると、砂の大きさが影響し、うまく測れなくなる。逆に桝目の大きさが大きすぎると風紋の凹凸が均されてしまう。こうした観点からも、本研究で求めた 1 サイトのスケールが 10^{-3}[m] に対応するという関係は、信頼性があるといえる。

参考文献

[1] 田口善弘『砂時計の七不思議』(中公新書、1995)

[2] Nicholas Lancaster, ”Geomorpholigy of Desert Dunes” (Routledge,1995)

A　3次元Wernerモデルのプログラム

```
#include<stdio.h>
#include<stdlib.h>

#define  yuragi   0.3
#define  x_scale  100
#define  y_scale  100
#define  q        0.1
#define  b        1.0
#define  L_0      3.0
#define  D        0.1
#define  side     2.0
#define  slant    1.0

int    i,j,s,t,L,k,ir,jr;
static float  h[x_scale][y_scale];

main(void){
FILE *OUTPUT;
OUTPUT=fopen("output.dat","w");

for(j=0;j<y_scale;j++){
   for(i=0;i<x_scale;i++){
      h[i][j]=2.85+(yuragi*(float)rand()/RAND_MAX);
                          }
                       }

printf(" Step Numbre : ");
scanf("%d\n",&s);

for(t=1;t<=s;t++){

/*saltation*/
for(k=1;k<10000;k++){
   (int)ir=100*(float)rand()/(int)RAND_MAX;
```

```
   (int)jr=100*(float)rand()/(int)RAND_MAX;

               if(h[ir][jr]>0){
               L=L_0+(int)(b*h[boundx(ir)][boundy(jr)]);
               h[boundx(ir)][boundy(jr)]-=q;
               h[boundx(ir+L)][boundy(jr)]+=q;
                                }
          else{
               continue;
               }
                      }

/*creep*/
for(j=0;j<y_scale;j++){
   for(i=0;i<x_scale;i++){
if(h[i][j]>h[boundx(i-1)][boundy(j)] &&
h[i][j]>h[boundx(i+1)][boundy(j)] && h[i][j]>h[boundx(i)][boundy(j-1)] &&
 h[i][j]>h[boundx(i)][boundy(j+1)] && h[i][j]>h[boundx(i-1)][boundy(j-1)] &&
 h[i][j]>h[boundx(i-1)][boundy(j+1)] && h[i][j]>h[boundx(i+1)][boundy(j-1)] &&
 h[i][j]>h[boundx(i+1)][boundy(j+1)])
{
    h[boundx(i-1)][boundy(j)]+=D*h[i][j]*(1.0/4.0)*(side/(side+slant));
    h[boundx(i+1)][boundy(j)]+=D*h[i][j]*(1.0/4.0)*(side/(side+slant));
    h[boundx(i)][boundy(j-1)]+=D*h[i][j]*(1.0/4.0)*(side/(side+slant));
    h[boundx(i)][boundy(j+1)]+=D*h[i][j]*(1.0/4.0)*(side/(side+slant));
    h[boundx(i-1)][boundy(j-1)]+=D*h[i][j]*(1.0/4.0)*(slant/(side+slant));
    h[boundx(i-1)][boundy(j+1)]+=D*h[i][j]*(1.0/4.0)*(slant/(side+slant));
    h[boundx(i+1)][boundy(j-1)]+=D*h[i][j]*(1.0/4.0)*(slant/(side+slant));
    h[boundx(i+1)][boundy(j+1)]+=D*h[i][j]*(1.0/4.0)*(slant/(side+slant));
    h[i][j]-=D*h[i][j];
}
else{
continue;
    }
                                  }
```

```
                        }
}

for(j=0;j<y_scale;j++){
printf("%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f\n",
 h[0][j],h[1][j],h[2][j],h[3][j],h[4][j],h[5][j],h[6][j],h[7][j],h[8][j],
 h[9][j],h[10][j],h[11][j],h[12][j],h[13][j],h[14][j],h[15][j],h[16][j],
 h[17][j],h[18][j],h[19][j],h[20][j],h[21][j],h[22][j],h[23][j],h[24][j],
 h[25][j],h[26][j],h[27][j],h[28][j],h[29][j],h[30][j],h[31][j],h[32][j],
 h[33][j],h[34][j],h[35][j],h[36][j],h[37][j],h[38][j],h[39][j],h[40][j],
 h[41][j],h[42][j],h[43][j],h[44][j],h[45][j],h[46][j],h[47][j],h[48][j],
 h[49][j],h[50][j],h[51][j],h[52][j],h[53][j],h[54][j],h[55][j],h[56][j],
 h[57][j],h[58][j],h[59][j],h[60][j],h[61][j],h[62][j],h[63][j],h[64][j],
 h[65][j],h[66][j],h[67][j],h[68][j],h[69][j],h[70][j],h[71][j],h[72][j],
 h[73][j],h[74][j],h[75][j],h[76][j],h[77][j],h[78][j],h[79][j],h[80][j],
 h[81][j],h[82][j],h[83][j],h[84][j],h[85][j],h[86][j],h[87][j],h[88][j],
 h[89][j],h[90][j],h[91][j],h[92][j],h[93][j],h[94][j],h[95][j],h[96][j],
 h[97][j],h[98][j],h[99][j]);

fprintf(OUTPUT,"%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f
%f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f  %f\n",
h[0][j],h[1][j],h[2][j],h[3][j],h[4][j],h[5][j],h[6][j],h[7][j],h[8][j],
h[9][j],h[10][j],h[11][j],h[12][j],h[13][j],h[14][j],h[15][j],h[16][j],
h[17][j],h[18][j],h[19][j],h[20][j],h[21][j],h[22][j],h[23][j],h[24][j],
h[25][j],h[26][j],h[27][j],h[28][j],h[29][j],h[30][j],h[31][j],h[32][j],
h[33][j],h[34][j],h[35][j],h[36][j],h[37][j],h[38][j],h[39][j],h[40][j],
```

```
h[41][j],h[42][j],h[43][j],h[44][j],h[45][j],h[46][j],h[47][j],h[48][j],
h[49][j],h[50][j],h[51][j],h[52][j],h[53][j],h[54][j],h[55][j],h[56][j],
h[57][j],h[58][j],h[59][j],h[60][j],h[61][j],h[62][j],h[63][j],h[64][j],
h[65][j],h[66][j],h[67][j],h[68][j],h[69][j],h[70][j],h[71][j],h[72][j],
h[73][j],h[74][j],h[75][j],h[76][j],h[77][j],h[78][j],h[79][j],h[80][j],
h[81][j],h[82][j],h[83][j],h[84][j],h[85][j],h[86][j],h[87][j],h[88][j],
h[89][j],h[90][j],h[91][j],h[92][j],h[93][j],h[94][j],h[95][j],h[96][j],
h[97][j],h[98][j],h[99][j]);
                }
fclose(OUTPUT);
}

int boundx(int i){
if (i<0) {
  return i+x_scale;
         }
else if(i>=x_scale) {
  return i-x_scale;
                   }
else {
  return i;
     }

                 }

int boundy(int j){
if (j<0) {
  return j+y_scale;
         }
else if(j>=y_scale) {
  return j-y_scale;
                   }
else {
  return j;
     }
```

```
}
```

A　2次元Wernerモデルのプログラム

```
#include<stdio.h>
#include<stdlib.h>

#define  yuragi   0.3
#define  x_scale  1
#define  y_scale  10000
#define  q        0.1
#define  b        1.0
#define  L_0      300.0
#define  D        0.1
#define  E        0.05

int    i=0,j,s,t,L,k;
static float  h[x_scale][y_scale];

main(void){
FILE *OUTPUT;
OUTPUT=fopen("output.dat","w");

for(j=0;j<y_scale;j++){
      h[i][j]=49.85+(yuragi*(float)rand()/RAND_MAX);
                        }

printf(" Step Numbre : ");
scanf("%d\n",&s);

for(t=1;t<=s;t++){

/*saltation*/
for(k=1;k<10000;k++){
   (int)j=10000*(float)rand()/(int)(RAND_MAX);
if(h[i][j]>0){
        L=L_0+(int)(b*h[i][j]);
        h[i][j]-=q;
```

```
        h[i][boundy(j+L)]+=q;
             }
             else{
             continue;
                 }
                       }

/*creep*/
for(j=0;j<y_scale;j++){
if(h[i][j]>h[i][boundy(j-1)] && h[i][j]>h[i][boundy(j+1)])
{
    h[i][boundy(j-1)]+=D*h[i][j]*(1.0/2.0);
    h[i][boundy(j+1)]+=D*h[i][j]*(1.0/2.0);

    h[i][j]-=D*h[i][j];
}
else if(h[i][j]>h[i][boundy(j-1)])
{
    h[i][boundy(j-1)]+=h[i][j]*E;
    h[i][j]-=h[i][j]*E;
}
else if(h[i][j]>h[i][boundy(j+1)])
{
    h[i][boundy(j+1)]+=h[i][j]*E;
    h[i][j]-=h[i][j]*E;
}
else{
continue;
    }
                          }
}

for(j=0;j<y_scale;j++){
printf("%f\n",h[i][j]);
```

```
fprintf(OUTPUT,"%f\n",h[i][j]);
                }
fclose(OUTPUT);
}


int boundy(int j){
if (j<0) {
  return j+y_scale;
         }
else if(j>=y_scale) {
  return j-y_scale;
                   }
else {
  return j;
     }
                 }
```