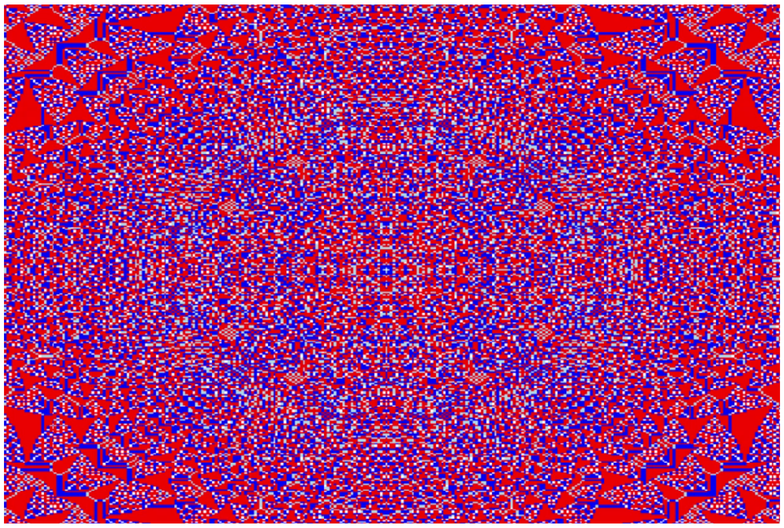
Econométrie Appliquée



Sandpiles Model

Jules Knieriemen, Valentine Léonard and Romane Crismat- Le Dirach

October 2024

1 Introduction

Can a microscopic behavior turn a dynamical system into his critical state ? That's what the BTW model, also known as the Abelian sandpile model tries to capture. It became the first example of a dynamical system displaying self-organized criticality (SOC). This model was introduced in 1987 by Per Bak, Chao Tang and Kurt Weisenfeld. It has helped to better understand natural phenomena as earthquakes and avalanches, where we can observe those kinds of behaviors. Beyond geophysics or plasma physics, SOC also finds applications in economics, where it provides insights into economic phenomena that traditional theories struggle to explain. Our work here was to code a sandpile model on SAS. The code plots the result of a 2**20 sand grains addition in a sandpile model.

2 What is a sandpile model ?

First, let's explain the sandpile model concisely. It simulates the collapse of a sandpile caused by the gradual addition of sand grains. Start from an empty pile. As grains accumulate, the local slope of the pile may become too steep to support the grains, in which case an avalanche occurs, the grains in unstable locations are redistributed in the pile until a new equilibrium position is found. Initially, when not many particles have been added yet, avalanches will be small. As more particles get added, the typical size of avalanches grows. The sandpile is usually assumed to be located on a square grid (a 2 dimensions space) divided into different cells , such that the unstable grains at the edges of the pile can be drop off the edges. On our model we have chosen a 251 by 215 matrix. Each cell has a state that can evolve with time according to rules and properties (cellular automaton's mathematical model). The sand-pile's rule is that there is a maximum amount of sand grain a cell can contain. If the amount of sand grains in a cell increases by more than 3 pieces then the sand grain topples and falls onto the cell's neighbors. One sand grain is distributing in each direction. Hence, the evolution of the cells depends on the state of neighboring cells. The dynamics of the sandpiles occur when the grains topple over a graph. In our model this dynamic is represented on a heat map that colors cells according to the quantity of sand grains. Over the long term, the influx of grains due to random grain drops is counterbalanced by the efflux of grains falling off the edges, so that the sandpile enters stochastic equilibrium as soon as a certain number of grains is deposited in the pile.

3 How did we code it ?

3.1 Our code :

```
%macro GenerateSandpile(Longueur, Largeur, nbgrains);
    proc iml;
    /*Initialize sand matrix*/
        Lo = &Longueur;
        La = &Largeur;
        sandpiles = j(Lo, La, 0);
        N = &nbgrains;
        /*Add all grains to the center of the matrix*/
        sandpiles[(Lo/2)+0.5, (La/2)+0.5] = N;
        /* Redistribute the grains without a \do while" loop */
        do until (max(sandpiles) < 4);
        /*Locate indices where number of grains >= 4 */
            idx = loc(sandpiles >= 4);
            if ncol(idx) > 0 then do;
            /* Subtract 4 grains from the cells found */
                sandpiles[idx] = sandpiles[idx] - 4;
                /* Redistribute to neighboring cells */
                do i = 1 to ncol(idx);
                    x = ceil(idx[i] / La);
                    y = mod(idx[i] - 1, La) + 1;
                    if (x = 1 | x = Lo | y = 1 | y = La) & (sandpiles[x, y] >= 4) then do;
                        sandpiles[x, y] = sandpiles[x, y] - 1;
                    end;

                    /* Redistribute to top, bottom, left, right */
                    if x > 1 then sandpiles[x-1, y] = sandpiles[x-1, y] + 1; /*Top*/
                    if x < Lo then sandpiles[x+1, y] = sandpiles[x+1, y] + 1; /*Bottom*/
                    if y > 1 then sandpiles[x, y-1] = sandpiles[x, y-1] + 1; /*Left*/
                    if y < La then sandpiles[x, y+1] = sandpiles[x, y+1] + 1; /*Right*/
                end;
            end;
        end;
        title "Abelian Sandpile Model Discrete Heatmap";
        call HeatmapDisc(sandpiles) colorramp={white lightblue blue red green}
        DISPLAYOUTLINES=0;
    quit;
%mend GenerateSandpile;
%GenerateSandpile(251, 251, 2**20);
```

3.2 Description of the code :

We've built a 'GenerateSandpile' macro program to automate the sandpile model process. The macro is defined according to 3 variables :

- the Length (Longueur)

- the Width (Largeur)

- the Total number of sand grains (nbgrains)

Then we initiate a 'proc iml' in order to manipulate the matrix of our sandpile model that we define as 'sandpiles'.

/*Initialize sand matrix*/

We redefine the macro variables (Lo for Longueur, La for Largeur and N for the total number of sand grains) and we initialize the matrix 'sandpiles' with as dimensions Length and Width and fulfilled with the values of zeros.

/*Add all grains to the center of the matrix*/

We add all the sand grains in the same central cell of the sandpiles' matrix so as we can obtain a symmetrical geometrical pattern at the end. It permits to have a more effective treatment. Then, to find which cell is the central cell we divide the length and the width of the sandpiles matrix by two. We add 0.5 because the total number of cells in the matrix is an odd number. The new value of sandpiles is itself plus N.

The first action of our code is creating a loop adding all the sand grains at once. All the sand grains are added into the same cell, which is the central cell on the graph, so as we can obtain a (nice) geometrical pattern at the end.
/* Redistribute the grains without a "do while" loop */

We start the loop 'do until' under the condition 'while the maximum value in the sandpile matrix is not less than 4, then'.

/*Locate indices where number of grains $\geq$ 4*/ We use the LOC function to match the condition (to find the values where): 'locate sandpiles' matrix values greater than 4 or equal to 4'.

If there is more than zero column in the matrix where the value is higher than 4 (max function) then, we locate the cells confirming the condition thanks to the LOC function.

/* Subtract 4 grains from the cells found */

We subtract 4 sand grains to the value of the cells that have confirmed the previous condition of the LOC function.

/* Redistribute to neighboring cells */

The next step is to redistribute the sand grains subtracted from the cell with more than 4 pieces to the neighboring cells. We create another loop destinated to find the cells coordinates. The loop is repeating the number of times the LOC function has find cells matching the condition. The coordinates those cells are :

(x) to determine the 'row coordinate' of the cell found during the iteration i: the smallest integer value greater than or equal to the index [i] that is used in the LOC function, divided by the number of columns.

(y) to determine the 'column coordinate' of the cell found during iteration i: return the position in the column, according to the width of the matrix, remove 1 in order to have a base 0 for the rest of the calculation, then add one since sas uses indices starting at 1.

Knowing that the matrix is a limited space, so that the grains of sand contained on the edges of the matrix don't accumulate, we remove a grain of sand from the cell where we've just redistributed a grain of sand, if one of the following conditions is true: the cell is on the first row or column, or on the last row or column, supplemented by another condition: the cell where we've removed a grain of sand must contain a number greater than or equal to 4. Close the sand distribution loop to the neighboring cells.

/* Redistribute to top, bottom, left, right */ Sand grains are then distributed to the top, bottom, left and right of the cells with 'excess' sand.

If x >1, then a grain of sand is distributed to the cell above (x-1) x. Do the same with y to distribute to the left.

If x >Length then a grain of sand is distributed to the cell below (x+1) x. Do the same with y to distribute to the right (y+1).

We then reset each neighboring cell with its new value, i.e. new value v1= V0 +1 with V0= value preceding iteration.

Close grain distribution conditions (top, bottom, left and right).

Close loop 'If there is more than zero column in the matrix where the value is higher than 4 (max function) then, we locate the cells confirming the condition thanks to the LOC function'.

We close the loop 'do until maximum value in the matrix >4.

After that we realize the "Abelia Sandpile Model Discrete Heatmap". The heatmap is colored according to the quantity of sandgrains(0=white, 1=light blue, 2= blue, 3= red, 4= green). The heatmap doesn't display the outlines. We close the 'proc iml' and the macro program. We write the values cwe have chosen to execute the program, a 251 by 251 matrix and a 2**20 amount of sand grains.

3.3 Video resulting from our code :

For that video, we generated several models, each with a different number of iteration : *hyperlink to watch a sandpile's model dynamics video presentation*

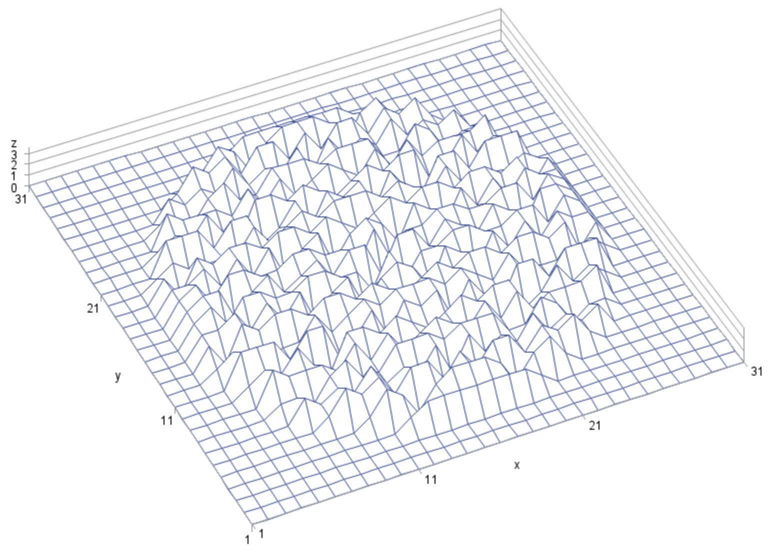


Figure 1: Abelian Sandpile Model - 3D Surface

4 Model's properties

The sandpile model is an example of what we call a cellular automaton. A cellular automaton is a mathematical model composed of a grid of cells, where each cell has a state that can evolve with time according to rules and properties. This model is often used to simulate complex systems where the evolution of the cells depends on the state of neighboring cells.

During the past years, SOC model has been a major discovery in the statistical physics' field. But what does the notion of self-organized criticality exactly mean ? It is about dynamical systems. A dynamical system is a system whose state is changing over time. SOC is focused more specifically on some dynamical systems whose states are turning naturally into a critical state at some point. This change of state can be induced even by small events, this is what we have called 'microscopic behavior'. The main characteristic of the SOC model is its complexity. The microscopic behaviors of the SOC systems have independent scales concerning space or time. Which means that whatever the size of the scale system is (spatial scale independence notion) whatever the amount of time the event (provoking the critical state) is occurring we would observe similarities between different systems behaviors. To illustrate, avalanches can occur at small or large scales, and whatever is the scale of the avalanche, it would follow some specific characteristics that every avalanche phenomena would follow. These characteristics are statistical properties dictated by the power-law decay. It is a specific case of a power-law function. The power laws describe a relation where we have $f(x)$, a function that varies according to the power of x, such that : $f(x) = C \cdot x^{\alpha}$ (with $f(x)$ varying according to the power of x). The relation described by the power-law decay function is : $y(x) = C \cdot x^{-\alpha}$ In a power decay, the quantity $y(x)$ decreases as x increases and α (the exponent decay) directs to how fast the quantity decays with the power of a particular variable. This variable can be the time, the distance or its size of the structure showing power decay. One of the main features of power-law decay is the scale invariance The power law implies there are many small events and only a few large ones, but the large events have a much greater impact than the small ones. In our case, there are many small piles of grains, and adding just one grain to the model usually doesn't have a big effect. However after having added a large addition of sand grain in the model, the effect on all the surrounding cells would be much bigger. We also can illustrate that with large avalanches that are less frequent, but when they occur, they have a significant impact. And so, many systems presenting power decay can be modeled using the concept of self-organized criticality.

Another notion we can have a look connected to the sandpile model is the concept of fractals. A fractal is a geometrical object presenting similarities within its structure at arbitrarily small or large scales, we say it is 'infinitely fragmented'. The event 'the sandpile topples' can occur as much at a small scale as at a large scale. Fractals objects follow a power law distribution and present self-similarities as we have just said before. Because they represent complex systems and scale-invariant patterns emerging when the dynamical system is reaching its critical state, fractals are connected to SOC.

5 Application in economics field

Econophysics is an interdisciplinary research field that adapts theories and methods from physics to tackle economic and financial problems. Its rise was driven by the massive data generated by stock exchanges such as NASDAQ in the late 1990s. In addition, physicists began to apply statistics methods to overcome the inability of the traditional economics theories, particularly neoclassical theories, which often fail to make accurate quantitative predictions. Some assumptions of the neoclassic theories applied to the real-world financial markets can be questioned as the rationality of agents or that the market is at the equilibrium.

As we said, econophysicists based their analysis on data. However, they do not econometrics, which is a brand of economics also concerned with analyzing data. Indeed, econometricians grounded their work on established economic theories. On the contrary, the econophysicists analyze data to attempt to uncover the underlying laws governing the behavior of the economic and financial systems.

Financial markets are a complex environment where the self-organized behavior of independent agents, motivated by their own objectives, determines unpredictable emergent dynamics. Bouchaud, a French physicist, suggests that the "wild" characteristics of financial markets are better understood within the framework of complex, chaotic systems that are neglected by economists. Instead, traditional economics relies on axiomatic assumptions. According to Bouchaud, statistical regularities should emerge in the behavior of large populations in the same way statistical laws of ideal gases emerge as a result of chaotic motion of molecules.

During the last two decades, econophysicists have helped to better comprehend the economics sphere by precising empirical facts about financial markets such as power-law distribution of large price movements. They have highlighted links between markets and other natural phenomena such as Omori law for earthquake aftershocks and market behavior following a large crash, which suggests that market dynamics may be governed by some general dynamic principles not specific to financial markets. They have improved the economics models by adding qualitative features of markets, one of which is that market dynamics depend on the diversity of participants' strategic behavior. Markets operate smoothly when the agents use many diverse strategies but break down if the strategies become similar.

Apendix

October 24, 2024

Code of the Abelian Sandpile Model - 3D Surface :

```
%macro GenerateSandpile(Longueur, Largeur, nbgrains);
    proc iml;
        /*Initialize sand matrix*/
        Lo = &Longueur;
        La = &Largeur;
        sandpiles = j(Lo, La, 0); /* Matrice initiale remplie de 0 */
        N = &nbgrains;

        /*Add all grains to the center of the matrix*/
        sandpiles[(Lo/2)+0.5, (La/2)+0.5] = sandpiles[(Lo/2)+0.5, (La/2)+0.5] + N;

        /* Redistribute to neighboring cells */
        do until (max(sandpiles) < 4);
            /*Locate indices where number of grains >= 4 */
            idx = loc(sandpiles >= 4);

            if ncol(idx) > 0 then do;
                /* Subtract 4 grains from the cells found */
                sandpiles[idx] = sandpiles[idx] - 4;

                /* Redistribute to neighboring cells */
                do i = 1 to ncol(idx);
                    x = ceil(idx[i] / La);       /* Ligne correspondant à l'index */
                    y = mod(idx[i] - 1, La) + 1; /* Colonne correspondant à l'index */

                    /* Redistribute to top, bottom, left, right */
                    if x > 1 then sandpiles[x-1, y] = sandpiles[x-1, y] + 1; /* Top */
                    if x < Lo then sandpiles[x+1, y] = sandpiles[x+1, y] + 1; /* Bottom */
                    if y > 1 then sandpiles[x, y-1] = sandpiles[x, y-1] + 1; /* Left */
                    if y < La then sandpiles[x, y+1] = sandpiles[x, y+1] + 1; /* Right */
                end;
            end;
        end;
        /* Créer des matrices pour x, y et z pour le graphique 3D */
        x = repeat(t(1:Lo), 1, La);  /* Coordonnées x */
        y = repeat(1:La, Lo, 1);     /* Coordonnées y */
        z = shape(sandpiles, Lo*La, 1); /* Hauteurs (grains de sable) */

        /* Create the dataset from the matrix x, y, z*/
        create sandpiles3d var {"x" "y" "z"};
        append;
        close sandpiles3d;
    quit;

    /* 3D graphics using PROC G3D */
    proc g3d data=sandpiles3d;
        plot y*x=z / grid rotate=25 tilt=5; /* Rotation et inclinaison pour visualiser le 3D */
        title "Abelian Sandpile Model - 3D Surface";
    run;
%mend GenerateSandpile;

%GenerateSandpile(31, 31, 2**10);
```

Bibliography

ALESSIO EMANUELE BIONDO, ALESSANDRO PLUCHINO, ANDREA RAPISARDA (2015), *Modeling Financial Markets by Self-Organized Criticality.*

ANDREY SOKOLOV (2014), *Application of non-equilibrium statistical mechanics to the analysis of problems in financial markets and economy.*

YULIETH PRIETO, MIKHAIL SHKOLNIKOV (2018), *Self-organized criticality and pattern emergence through the lens of tropical geometry.*

ALEXANDER SHAPOVAL, BORIS SHAPOVAL, MIKHAIL SHNIRMAN (2021), *1/x power-law in a close proximity of the Bak–Tang–Wiesenfeld sandpile.*

CARLOS A. ALFARO, JUAN PABLO SERRANO, RALIHE R. VILLAGRAN (2024), *Evolutive sandpiles.*

HEIKO HOFFMANN, DAVID W. PAYTON (2018), *Optimization by Self-Organized Criticality.*

Wikipedia page
ANTAL A. JARAI (2014), *Sanpile models**

IMRE M. JÁNOSI, ANDRÁS CZIRÓK (1994), *Fractal clusters and self-organized criticality*

THE CODING TRAIN (2018), Coding Challenge: Sandpiles
Sandpiles - Numberphile
Coding Challenge 107: Sandpiles
KEZAKO: Qu'est ce qu'une fractale?
Self-organized criticality and pattern emergence through the lens of tropical geometry
Essay Sandpile Model, ETH Zurich