Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

The Convergence Speed of Sub-critical Divisible Sandpiles

(Nederlandse titel: De Convergeer Snelheid van Sub-critieke Divisible Sandpiles)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

Theodorus Johannes Martinus Schuttenbeld

Delft, Nederland
Oktober 2021

BSc verslag TECHNISCHE WISKUNDE

"The Convergence Speed of Sub-critical Divisible Sandpiles (Nederlandse titel: De Convergeer Snelheid van Sub-critieke Divisible Sandpiles)"

Theodorus Johannes Martinus Schuttenbeld

Technische Universiteit Delft

Begeleider

Dr. A. Cipriani

Commissielid

Dr.Ir. M Keijzer

Oktober, 2021 Delft

Abstract

This thesis developed a computer powered simulation study of the divisible sandpile model. It introduces a constant c to a widely used formula to generate sandpiles Levine et al. [8]:

$$s(x) = 1 + \sigma(x) - \frac{c}{|V|} \sum_{y \in V} \sigma(y).$$

This constant can be used to study the convergence characteristics of sandpiles. In this thesis it is shown that the introduced factor increases the speed of convergence while it no longer stabilizes in the all-1-configuration. While one should be careful with using c as it does not work for every distribution of $\sigma(x)$, it does speed up the stabilization significantly. There is also a positive correlation found between c and the amount of nodes that remain lower than 1 in the stable state.

Contents

1 Introduction

In this thesis we study the divisible sandpile model (DSM) and in particular its stabilization. The DSM is the continuous version of the Abelian sandpile model introduced in 1987 by Bak, Tang, and Wiesenfeld [1]. The Abelian sandpile model starts with a finite graph V where every vertex $x \in V$ has a starting weight of $s(x) \in \mathbb{Z}$. Also a value $a \in \mathbb{Z}$ is chosen, all sites for which $s(x) > a$ are called unstable. Then at every timestep $t > 0$ all unstable sites topple and distribute their surplus weight $s(x) - a$ equally over their neighbouring vertices. This toppling procedure is repeated until all sites are stable, either when they are all equal or when $s(x) \leq a$ for all $x \in V$. This model was altered by Levine and Peres [7] from the Abelian sandpile model with only integer values to the DSM with continuous values. This means that the graph V is the same however the weights are now $s(x) \in \mathbb{R}$.
The variant of the DSM we will focus on is formed by generating randomly $\sigma(x)$ for all vertices in V, and using the following formula to compute the starting weight $s(x) \in \mathbb{R}$:

$$s(x) = 1 + \sigma(x) - \frac{1}{|V|} \sum_{y \in V} \sigma(y), \tag{1}$$

where $|V|$ denotes the cardinality of the graph V. This formula is used a lot for the DSM as it has a few nice properties which we will explain in section 3. For this project Equation 1 was altered a little bit to the following:

$$s(x) = 1 + \sigma(x) - \frac{c}{|V|} \sum_{y \in V} \sigma(y). \tag{2}$$

Note that Equation 1 and Equation 2 only differ by a constant $c \in \mathbb{R}$ in the numerator. This constant could be any real value, and the equations would be equal if $c = 1$ or in the hypothetical case where $\sum_{y \in V} \sigma(y) = 0$.

In this thesis the influence of this introduced constant c will be examined. The speed and the degree of convergence will be looked at using computer simulations. A lot has been proven about the convergence of the DSM with Equation 1 as its original configuration. Some of these proofs will be repeated in this thesis, and the consequence of adding the constant to these proofs will also be examined. Depending on the sign of $\sum_{y \in V} \sigma(y)$ the total mass in the sandpile is either reduced or increased if c gets larger. We will see that if the mass of the sandpile is reduced the time it takes until a stable state is reached is much lower. Moreover a correlation between c and the amount of nodes that do not reach 1 is found.

As stated by Levine and Peres [6], the subject of divisible sandpile models has been used in numerous areas of mathematics, including amongst others statistical physics, free boundary partial differential equations, probability and number theory. While the introduction of the constant c is purely theoretical, it is useful for all these subjects if it helps us understand sandpile models better.

To make sure that the reader is able to understand the mathematics used in this thesis, section 2 starts with explaining the fundamentals of probability theory and other basic mathematical concepts.
In section 3 the divisible sandpile model is explained, and it is proven that the DSM does not stabilize in finite time. This is an important result for the outcome of the computer simulations. Section 4 introduces the main idea of this thesis, a new constant is added to the formula that defines the initial configuration of the sandpile. And the consequences of adding the new constant will be discussed.

The simulation study will be explained in section 5. We start of with a tool that allows us to look at the progression of the sandpile, starting with a graph of the initial configuration and then it shows what it looks like while approaching the stabilized state. After that the influence of adding the aforementioned constant on the convergence of the sandpile is studied. The results of this can be found in section 6.
Section 7 details the conclusions drawn in this thesis, and finally in section 8 the recommendations for future research are made and certain decisions made are discussed.
All simulation study was done in MATLAB, all the used codes can be found in the appendix.

2 Probability tools

Before we can dive into the research of the divisbile sandpile model we first need to make sure every reader knows all the necessary terms and definitions. We begin with the fundamentals of probability theory. A helpful book for readers unfamiliar with rigorous probability theory is Rosenthal [11], it is also the source of most of the following definitions.

2.1 Fundamentals of Probability theory

Definition 1. *We define a probability space to be $(\Omega, \mathcal{F}, \mathbb{P})$, where:*

- *the sample space Ω is any non-empty set;*
- *the σ-algebra $\mathcal{F}$ is a collection of subsets of Ω, containing Ω and the empty set $\emptyset$, and closed under complements and countable unions;*
- *the probability measure $\mathbb{P}$ is a mapping from $\mathcal{F}$ to $[0,1]$, with $\mathbb{P}(\emptyset) = 0$ and $\mathbb{P}(\Omega) = 1$, such that $\mathbb{P}$ is countably additive.*

Definition 2. *Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a random variable is a function X from Ω to the real numbers $\mathbb{R}$, such that*

$$\{\omega \in \Omega : X(\omega) \leq x\} \in \mathcal{F}, \quad x \in \mathbb{R}.$$

Definition 3. *Given a random variable X on a probability triple $(\Omega, \mathcal{F}, \mathbb{P})$, its distribution (or law) is the function μ defined on $\mathcal{B}$, the Borel subsets of $\mathbb{R}$, by*

$$\mu(B) = \mathbb{P}(X \in B) = \mathbb{P}(X^{-1}(B)), \qquad B \in \mathcal{B}.$$

Definition 4. *We define the cumulative distribution function of a random variable X by $F_X(x) = \mathbb{P}(X \leq x)$, for $x \in \mathbb{R}$.*

Definition 5. *We say that a sequence of real numbers $x_1, x_2, \ldots$ converges to the real number x if, given any $\epsilon > 0$, there exists an $N \in \mathbb{N}$ such that for any $n \geq N$, we have $|x_n - x| < \epsilon$. We shall also write this as $\lim_{n\to\infty} x_n = x$.*

Definition 6. *A random variable X is simple if range(X) is finite, where range(X)$\equiv \{X(\omega); \omega \in \Omega\}$.*

If X is a simple random variable, then listing the distinct elements of its range as $x_1, x_2, \ldots, x_n$, we can then write $X = \sum_{i=1}^n x_i \mathbb{1}_{A_i}$ where $A_i = \{\omega \in \Omega; X(\omega) = x_i\} = X^{-1}(\{x_i\})$, and where the $\mathbb{1}_{A_i}$ are indicator functions.

Definition 7. *For a simple random variable $X = \sum_{i=1}^n x_i \mathbb{1}_{A_i}$ we define its expected value or expectation or mean by*

$$\mathbb{E}(X) = \mathbb{E}\left(\sum_{i=1}^n x_i \mathbb{1}_{A_i}\right) = \sum_{i=1}^n x_i \mathbb{P}(A_i), \qquad \{A_i\}_{i=1}^n \text{ a finite partition of } \Omega.$$

Using the definition of the expected value of a simple random variable we can define the expected value of a general non-negative random variable.

Definition 8. *For a general non-negative random variable X we define the expected value $\mathbb{E}(X)$ by*

$$\mathbb{E}(X) = \sup(\mathbb{E}(Y); Y \text{ simple, } Y \leq X).$$

Finally, we consider random variables which may be neither simple nor non-negative.

Definition 9. *For a random variable X which is not simple nor non-negative, we may write $X = X^+ - X^-$, where $X^+(\omega) = \max(X(\omega), 0)$ and $X^-(\omega) = \max(-X(\omega), 0)$. Both X^+ and X^- are non-negative random variables. We may then set $\mathbb{E}(X) = \mathbb{E}(X^+) - \mathbb{E}(X^-)$.*

We note that $\mathbb{E}(X)$ is undefined if both $\mathbb{E}(X^+)$ and $\mathbb{E}(X^-)$ are infinite. However, if $\mathbb{E}(X^+) = \infty$ and $\mathbb{E}(X^-) < \infty$, then $\mathbb{E}(X) = \infty$. Similarly, if $\mathbb{E}(X^+) < \infty$ and $\mathbb{E}(X^-) = \infty$, then $\mathbb{E}(X) = -\infty$.

Definition 10. *The variance of a random variable X on $(\Omega, \mathcal{F}, \mathbb{P})$ is defined by*

$$\operatorname{Var}(X) = \mathbb{E}((X - \mathbb{E}(X))^2)$$

Definition 11. *Two random variables X and Y are independent if for all Borel sets S_1 and S_2*

$$\mathbb{P}(X \in S_1, Y \in S_2) = \mathbb{P}(X \in S_1)\mathbb{P}(Y \in S_2).$$

Definition 12. *The covariance of two random variables X and Y is defined as*

$$\operatorname{Cov}(X, Y) = \frac{1}{2}(\operatorname{Var}(X + Y) - \operatorname{Var}(X) - \operatorname{Var}(Y)).$$

This can be rewritten to the more useful

$$\operatorname{Cov}(X, Y) = \mathbb{E}((X - \mathbb{E}(X))(Y - \mathbb{E}(Y))) = \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y).$$

In this paper we will talk a lot about convergence. In probability theory there are multiple ways to define convergence, because sometimes convergence is quicker or stronger. If $Z, Z_1, Z_2, \ldots$ are random variables defined on $(\Omega, \mathcal{F}, \mathbb{P})$, what does it mean to say that $\{Z_n\}$ *converges* to Z as $n \to \infty$?

Definition 13. *We say that if for all $\omega \in \Omega$ $\lim_{n\to\infty} Z_n(\omega) = Z(\omega)$ then $\{Z_n\}$ converges* pointwise *to Z.*

A slightly weaker notion which is often used is *almost sure convergence.* In general almost surely means with probability 1.

Definition 14. *If $\mathbb{P}\{\omega \in \Omega : \lim_{n\to\infty} Z_n(\omega) = Z(\omega)\} = 1$, we say $\{Z_n\}$ converges* almost surely *to Z.*

Another notion only involves probabilities.

Definition 15. *We say that $\{Z_n\}$ converges* in probability *to Z if $\forall \epsilon > 0$,*

$$\lim_{n\to\infty} \mathbb{P}(|Z_n - Z| \geq \epsilon) = 0.$$

Proposition 5.2.3 from Rosenthal [11] shows us that if a sequence of random variables converges almost surely, then it converges in probability to the same limit. Because the converse doesn't necessarily hold we can speak of strong and weak forms of convergence.

2.2 Clusters

Later in the thesis we will see that sometimes clusters of nodes will remain negative while the rest of the sandpile is already at or close to the converged state. To research this more thoroughly a method of finding clusters mathematically is required. The method we will be using is called k-means, found in Tan, Steinbach, and Kumar [12].

The method starts of with k random initial centroids, where k is the amount of clusters defined by the user. Each node is then assigned to the closest centroid, each collection of points assigned to a centroid is then a cluster. Next the centroid of each cluster is updated based on the average of the nodes in that cluster. Then all nodes are reassigned to the closest centroid again, and the centroids are updated. This is repeated until the centroids remain the same.

Algorithm 1: k-means algorithm

```
Result: k clusters centered around k centroids
Select $k$ nodes as initial centroids;
repeat
    Form $k$ clusters by assigning each node to its closest centroid.;
    Recompute the centroid of each cluster.;
until Centroids do not change;
```

To define the distance between nodes and centroids, the Euclidian distance metric is used. More on this method can be found in Chapter 8 of Tan, Steinbach, and Kumar [12].

3 Divisible Sandpile Model

3.1 Introduction to the Divisible Sandpile Model

We start this chapter by explaining what the Divisible Sandpile Model actually is. In simple terms it is a mathematical model of a pile of sand that evens out over time, because at every timestep the high piles topple to fill the lower gaps surrounding it.
In mathematical terms, consider a connected and undirected graph $G = (V, E)$, where V is the vertex set and E is the set of all edges. We write $x \sim y$ if the edge $(x, y) \in E$, and we define the degree of x by $\deg(x) = \#\{y \in V : y \sim x\}$. This graph represents the location but not the height of the sandpile. Now we can assign a value to each node in the graph which represents the mass at that position. This mass can be any real value in $\mathbb{R}$, the special case where all values lie in $\mathbb{N}$ is called the Abelian Sandpile Model, more on this version of the sandpile model can be found in Dhar [4].
The initial configuration is a function $s : V \to \mathbb{R}$. Write $\mathbb{E}(s(x))$ for the mean, and $\text{Var}(s(x))$ for the variance of $s(x)$. $\{\mathbb{E}(s(x))\}_{x \in V}$ is sometimes referred to as the density, from the physical term of mass per unit volume. If $s(x) > 1$ the node x is called unstable, meaning that it will topple in the next timestep. By toppling we mean that it will distribute the surplus mass, $s(x) - 1$, among its neighbours, but always keeping mass of 1 for itself. Toppling happens always at the same time for all unstable sites. If however $s(x) \leq 1$ we call the site stable, meaning that it will not topple and keep its own mass. It might however increase in mass due to getting the excess mass of its neighbours.

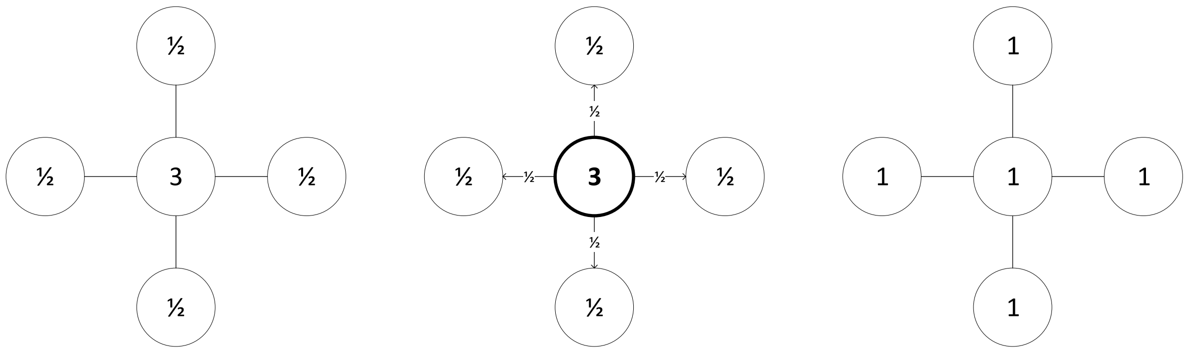

Figure 1: Left: The initial configuration. Middle: The site in the middle is unstable. Right: Stable configuration after toppling.

This procedure of toppling can lead to a stable configuration. By that we mean that $s(x) \leq 1$ for all $x \in V$, because then no site will topple anymore. It is of interest to us whether a sandpile stabilizes or not depending on the initial configuration and the underlying graph. Levine et al. [8] have proven that on a finite connected graph the divisible sandpile with $\sum_{x \in V} s(x) = |V| = n$ stabilizes to the all-1-configuration. To reproduce this proof we first need to state a few definitions. The first is the odometer.

Definition 16. *Write $u_t(x)$ for the total mass emitted to one of its neighbours after t timesteps. This quantity is a function which is always positive and strictly increasing with n, so $u_t(x) \uparrow u(x)$ as $t \to \infty$ for $u : V \to [0, \infty]$.*

Note that we either have $u(x) < \infty$ for all $x \in V$ or $u(x) = \infty$ for all $x \in V$. This is because if $u(x) = \infty$ for any $x \in V$, then it has emitted an infinite amount of mass to all its neighbours

$y \sim x$ which results in $u(y) = \infty$, and so it spreads around. As a result we can say that s stabilizes if $u(x) < \infty$ for all $x \in V$, otherwise we say s explodes. We denote by $\mathcal{X} = \mathbb{R}^V$ the set of divisible sandpile configurations on G.

A very important feature of the sandpile is its density. The density of the sandpile equals $\mathbb{E}(s(x))$ where x is a random point in the sandpile. We need to differentiate between three different cases of density, we say the sandpile is

- subcritical if $\mathbb{E}(s(x)) < 1$;
- critical if $\mathbb{E}(s(x)) = 1$;
- supercritical if $\mathbb{E}(s(x)) > 1$.

We will see later on that the density greatly influences if a sandpile stabilizes or not.

The following is another useful tool for future proofs.

Definition 17. *The graph Laplacian Δ acts on functions $f : V \to \mathbb{R}$ as*

$$\Delta f(x) = \sum_{y \sim x} (f(y) - f(x)). \tag{3}$$

Note that Δ is a linear operator, which means that for every pair of functions f and g and scalar t we have

$$\Delta(f + g) = \Delta(f) + \Delta(g) \quad \text{and} \quad \Delta(tf) = t\Delta(f). \tag{4}$$

Furthermore we call a function harmonic on V if $\Delta f(x) = 0$ for all $x \in V$.
Now if we turn back to the odometer, and assume the initial configuration to be $s \in \mathcal{X}$, then the resulting configuration at time t equals

$$s_t = s + \Delta u_t.$$

3.2 Torus

A very important part of a DSM is the underlying graph. In this thesis the underlying graph is always a torus, unless stated otherwise. A torus is the mathematical way of describing the surface of a doughnut. In the simulation study we will only be looking at the two-dimensional torus $\mathbb{T}^2$. In simple terms the 2-dimensional torus is constructed by taking a square in $\mathbb{Z}^2$ and identifying the left-right and top-bottom edges. Heuristically it is just like on a world map when you cross the east side of the map you end up at the western edge of the map. Only this time it also works for the north-south edge. Figure 2 shows the relation between the square and the doughnut, both representations of the torus.

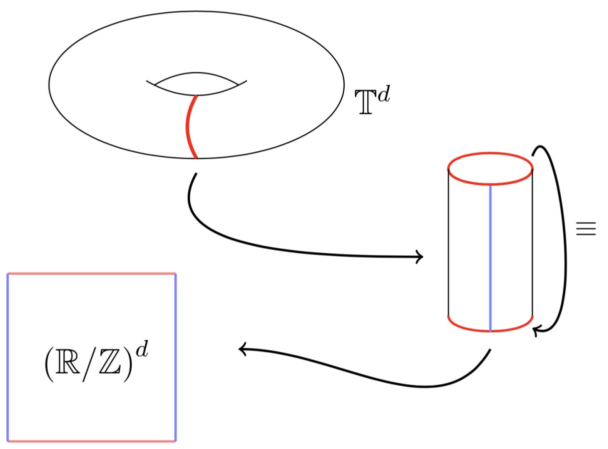

Figure 2: The construction of the doughnut from a square, the visualisation of a torus in $d = 2$. Source: Graaff [5].

Again heuristically this transformation consists of two cuts, the first is from the doughnut shape to the cylinder. The second is the cut horizontally through the cylinder to fold it flat into a square.
Formally there are two different kinds of tori, the manifold torus and the topological torus. The manifold torus is what we explained as the doughnut, the topological torus is the square with identified sides. We will be using the topological torus $\mathbb{Z}_n^d$.

Definition 18. *The topological torus $\mathbb{Z}_n^d$ is a subspace of $\mathbb{Z}^d$ defined as*

$$\mathbb{Z}_n^d = (\mathbb{Z} \cap (-n/2, n/2])^d \tag{5}$$

It has to be remarked that there is a slight difference between the topological and the manifold torus. Heuristically this is seen in the construction of the doughnut from the square. In doing this the square needs to be stretched in a way that is normally not possible. One could try it by taking a piece of paper and turning it into a cylinder without a problem. However in bending it to make a doughnut the cylinder starts to fold. This shows that the grid on a manifold torus differs from the grid on the topological torus.

Considering the topological torus taking $d = 1$, then for any $x \in \mathbb{Z}_n^d$ we have the following useful property:

$$x + n = x.$$

This can be compared to standing on the equator and walking the full circumference of the world to the east, ending up at the position started at. In the 2-dimensional case it works also for the other axis, obtaining for all $(x, y) \in \mathbb{Z}_n^2$ and all $a, b \in \mathbb{Z}$:

$$(x, y) + (an, bn) = (x, y).$$

3.3 Proposition 1.3

A very important result for our research is the following proposition 1.3 from Levine et al. [8]. This proposition proves that the divisible sandpile of a certain specified configuration stabilizes to the all-1-configuration and the law of its odometer is obtained. Before we can reproduce the proof of this proposition we need to clarify some definitions used in its statement and its proof. Levine et al. chose this particular configuration because of the following feature:

$$\begin{aligned}\sum_{x\in V} s(x) &= \sum_{x\in V}\left(1+\sigma(x)-\frac{1}{|V|}\sum_{y\in V}\sigma(y)\right)\\ &= |V| + \sum_{x\in V}\sigma(x) - \frac{|V|}{|V|}\sum_{y\in V}\sigma(y)\\ &= |V|.\end{aligned} \tag{6}$$

This ensures that the sandpile is of critical density.
For clarity, $|V|$ means the cardinality of the set V.

Proposition 1. *Let $G=(V,E)$ be a finite connected graph. Let $(\sigma(x))_{x\in V}$ be i.i.d. N(0,1), and consider the divisible sandpile*

$$s(x) = 1+\sigma(x)-\frac{1}{|V|}\sum_{y\in V}\sigma(y). \tag{7}$$

Then s stabilizes to the all-1-configuration, and the distribution of its odometer $u : V \to [0,\infty)$ is

$$(u(x))_{x\in V} \overset{d}{=} (\eta(x)-\min\eta)_{x\in V},$$

where the $\eta(x)$ are jointly Gaussian with mean zero and covariance

$$\mathbb{E}[\eta(x)\eta(y)] = \frac{1}{\deg(x)\deg(y)}\sum_{z\in V} g(z,x)g(z,y),$$

where g is defined by $g(x,y)=\frac{1}{|V|}\sum_{z\in V} g^z(x,y)$ and $g^z(x,y)$ is the expected number of visits to y by the simple random walk started at x before hitting z.

The reason why Equation 7 is so important to us is because the main focus of this research is to add a constant c to it

$$s(x) = 1+\sigma(x)-\frac{c}{|V|}\sum_{y\in V}\sigma(y). \tag{8}$$

And then look at the influence of this factor c on the convergence of the sandpile. We will want to look at the toppling behaviour of the new sandpile, whether it still stabilizes and how quickly it does.
Most of the definitions used in Proposition 1 have already been explained in the previous sections. We will now state and prove a very useful lemma 7.1 from Levine et al. [8].

Theorem 1. *Let $G=(V,E)$ be a finite connected graph with $|V|=n$. Let $s : V \to \mathbb{R}$ be a divisible sandpile with $\sum_{x\in V} s(x) = n$. Then, s stabilizes to the all-1-configuration, and the odometer of s is the unique function u satisfying $s+\Delta u = 1$ and $\min u = 0$.*

Proof. First we prove that all harmonic functions $f : V \to \mathbb{R}$ are constant. To prove this we assume the contrary, $\Delta f(x) = 0$ for all $x \in V$ and f is not constant on the graph V. Since V is finite we can find a $x_0 \in V$ such that $f(x_0) \geq f(y) \quad \forall y \sim x_0$ and $\exists y \sim x_0$ such that $f(x_0) > f(y)$. For this x_0 we get

$$\Delta f(x_0) = \sum_{y \sim x_0} (f(y) - f(x_0)) < 0,$$

Which is a contradiction to our assumption, so f must be constant on V. Now note that the kernel of Δ is 1-dimensional spanned by the constant function. This together with the rank-nullity theorem tells us that Δ has rank $n - 1$.
Next for any function g we have $\sum_{x \in V} \Delta g(x) = 0$.

$$\sum_{x \in V} \Delta g(x) = \Delta \left(\sum_{x \in V} g(x) \right) = 0.$$

Because Δ is a linear operator following from Equation 4.

Since $\sum_{x \in V}(1 - s(x)) = |V| - \sum_{x \in V} s(x) = n - n = 0$, we have $1 - s = \Delta v$ for some v. Setting $w = v - \min v$, we have $w \geq 0$ and $s + \Delta w = 1$ because

$$\begin{aligned} \Delta w = \Delta(v - \min v) &= \sum_{y \sim x} (v - \min v)(y) - (v - \min v)(x) \\ &= \sum_{y \sim x} (v(y) - \min v - v(x) + \min v) = \sum_{y \sim x} (v(y) - v(x)) = \Delta v. \end{aligned}$$

So s stabilizes. Now if u is any function satisfying $s + \Delta u \leq 1$, then

$$\sum_{x \in V} (s + \Delta u)(x) = \sum_{x \in V} s(x) + \sum_{x \in V} \Delta u(x) = n$$

So we get that $s + \Delta u = 1$ must hold, and thus s stabilizes to the all-1-configuration. Also any two functions u, v satisfying $s + \Delta u \leq 1$ and $s + \Delta v \leq 1$ differ only by a constant. Now by the least action principle (prop 2.5 from Levine et al. [8]), the odometer is the smallest non-negative of these functions, which also means that its minimum is 0. $\square$

We can now start our proof of Proposition 1.

Proof. Note that $\sum_{x \in V} s(x) = n$. Therefore by Theorem 1 s stabilizes to the all-1-configuration and the odometer u satisfies $s + \Delta u = 1$ and $\min u = 0$.
Fix $x, z \in V$ and let $f(y) = \frac{g^z(x,y)}{\deg(y)}$ be the function satisfying $f(z) = 0$ and $\Delta f = \mathbb{1}_z - \mathbb{1}_x$. Now define

$$v^z(y) := \frac{1}{\deg(y)} \sum_{x \in V} g^z(x, y)(s(x) - 1)$$

Then we get for $y \neq z$

$$\begin{aligned} \Delta v^z(y) &= \Delta \frac{1}{\deg(y)} \sum_{x \in V} g^z(x, y)(s(x) - 1) \\ &= \sum_{x \in V} \Delta \frac{1}{\deg(y)} g^z(x, y)(s(x) - 1) \\ &= \sum_{x \in V} (\mathbb{1}_z(y) - \mathbb{1}_x(y))(s(x) - 1) \\ &= 1 - s(y). \end{aligned}$$

However if $y = z$ then

$$\begin{aligned}\Delta v^z(y) &= \sum_{x\in V}(\mathbb{1}_z(y) - \mathbb{1}_x(y))(s(x)-1)\\ &= \sum_{x\in V}(1-\mathbb{1}_x(z))(s(x)-1)\\ &= \sum_{x\in V}(s(x)-1) - (s(z)-1)\\ &= 1 - s(z)\end{aligned}$$

Which gives us that $\Delta(u - v^z) = \Delta u - \Delta v^z = 0$, so $u - v^z$ is constant on V. Let $v = \frac{1}{n}\sum_{z\in V} v^z$. Because $u - v^z$ is constant, $u - v$ is also constant. From the fact that $g(x,y) = \frac{1}{n}\sum_{z\in V} g^z(x,y)$, we can get the more useful

$$\begin{aligned}v(y) &= \frac{1}{n}\sum_{z\in V} v^z(y)\\ &= \frac{1}{n}\sum_{z\in V}\frac{1}{\deg y}\sum_{x\in V} g^z(x,y)(s(x)-1)\\ &= \frac{1}{n\deg y}\sum_{z\in V}\sum_{x\in V} g^z(x,y)(s(x)-1)\\ &= \frac{1}{\deg y}\sum_{x\in V}(s(x)-1)\frac{1}{n}\sum_{z\in V} g^z(x,y)\\ &= \frac{1}{\deg y}\sum_{x\in V}(s(x)-1)g(x,y)\end{aligned}$$

To compute the law of the Gaussian vector v, note that from the definition of $s(x)$ it follows that

$$\begin{aligned}&\mathbb{E}[(s(z)-1)(s(w)-1)] = \mathbb{E}\left[\left(\sigma(z) - \frac{1}{n}\sum_{x\in V}\sigma(x)\right)\left(\sigma(w) - \frac{1}{n}\sum_{x\in V}\sigma(x)\right)\right]\\ &= \mathbb{E}\left(\sigma(z)\sigma(w) - \frac{1}{n}\sigma(w)\sum_{x\in V}\sigma(x) - \frac{1}{n}\sigma(z)\sum_{x\in V}\sigma(x) + \frac{1}{n^2}\left(\sum_{x\in V}\sigma(x)\right)^2\right)\\ &= \mathbb{E}(\sigma(z)\sigma(w)) - \frac{1}{n}\mathbb{E}\left(\sigma(w)\sum_{x\in V}\sigma(x)\right) - \frac{1}{n}\mathbb{E}\left(\sigma(z)\sum_{x\in V}\sigma(x)\right) + \frac{1}{n^2}\mathbb{E}\left(\left(\sum_{x\in V}\sigma(x)\right)^2\right)\\ &= \mathbb{1}_{z=w} - \frac{1}{n} - \frac{1}{n} + \frac{1}{n} = \mathbb{1}_{z=w} - \frac{1}{n}.\end{aligned}$$

Hence

$$\begin{aligned}\mathbb{E}(v(x)v(y)) &= \frac{1}{\deg(x)\deg(y)}\sum_{z,w\in V} g(z,x)g(w,y)\mathbb{E}((s(z)-1)(s(w)-1))\\ &= \frac{1}{\deg(x)\deg(y)}\left(\sum_{z\in V} g(z,x)g(z,y) - \frac{1}{n}\left(\sum_{z\in V} g(z,x)\right)\left(\sum_{w\in V} g(w,y)\right)\right).\end{aligned} \tag{9}$$

Now let $K(y) := \frac{1}{\deg(y)}\sum_{w\in V} g(w,y)$, then $\Delta K = \sum_{z,w\in V}\frac{1}{n}(\mathbb{1}_z - \mathbb{1}_w) = 0$. So K is constant. The second term on the right of the final line of Equation 9 equals $\frac{K^2}{n}$. After setting $C \sim$

$N(0, \frac{K^2}{n})$ a normal distributed random variable independent of v, the Gaussian vectors η and $(v(x) + C)_{x \in V}$ have the same covariance because

$$\begin{aligned}
\mathbb{E}((v(x) + C)(v(y) + C)) &= \mathbb{E}(v(x)v(y) + Cv(x) + Cv(y) + C^2) \\
&= \mathbb{E}(v(x)v(y)) + \mathbb{E}(Cv(x)) + \mathbb{E}(Cv(y)) + \mathbb{E}(C^2) \\
&= \frac{1}{\deg(x)\deg(y)} \left(\sum_{z \in V} g(z, x) g(z, y) \right) - \frac{K^2}{n} + 0 + 0 + \frac{K^2}{n} \\
&= \frac{1}{\deg(x)\deg(y)} \left(\sum_{z \in V} g(z, x) g(z, y) \right) = \mathbb{E}(\eta(x)\eta(y)).
\end{aligned}$$

This together with $\mathbb{E}(\eta(x)) = \mathbb{E}(v(x) + C) = 0$ leads to

$$\eta \overset{d}{=} v + C. \tag{10}$$

Now to conclude, notice again that $u - v$ is constant and $\min u = 0$, this gives

$$u = v - \min v \overset{d}{=} \eta - \min \eta. \tag{11}$$

□

The result presented in Equation 11 is elaborated on further by Levine et al. [8]. They go on to prove useful properties of the expected value of the odometer. They state that the expected odometer equals the expected maximum of the field η, since $\mathbb{E}(\eta(x)) = 0$ and from this it follows that

$$\mathbb{E}(u(x)) = \mathbb{E}(\eta(x) - \min \eta) = -\mathbb{E}(\min \eta) = \mathbb{E}(\max \eta). \tag{12}$$

They then go on to determine the order of $\mathbb{E}\{\eta_x : x \in \mathbb{Z}_n^d\}$ up to a dimension-dependent constant factor. The results of Levine et al. [8] Table 1 give bounds up to a constant factor depending only on the dimension d of the sandpile. For our case $d = 2$ they come to the conclusion that $C^{-1}n \leq \mathbb{E}(\max \eta) \leq Cn$. Here C is an unknown positive constant in $\mathbb{R}$.

3.4 Convergence speed

An important fact to prove before we begin with the simulation study is that the sandpile will never stabilize in finite time. Although Proposition 1 proves it will stabilize to the all-1-configuration, it requires an infinite amount of time to do so. Only in the limit do we get $s + \Delta u = 1$. We will split the proof in two different parts: first we look at divisible sandpiles on the torus $\mathbb{Z}_n^d$ for $n \geq 3$, and then we look at $n = 1$ and $n = 2$. The following proofs have been taken from Graaff [5]. In the following section s is defined as in Equation 7.

Theorem 2. *The divisible sandpile $(s(x))_{x \in \mathbb{Z}_n^d}$ for $n \geq 3$ does not stabilize in finite time almost surely.*

The proof of this theorem will be split up in different steps and different lemmas. The basic concept is to prove that two neighbouring nodes x, y exist such that $s_t(x) > s_t(y) \geq 1$. This will mean that x will then be unstable and topple to give some mass to y, which means y will be unstable and topple in the next timestep to distribute some mass to x again. This will continue happening and only in the limit will both x and y have a mass of 1. The mass that they keep giving back and forth will decrease, but only in the limit will it be equal to 0. The following lemma will prove this formally.

Lemma 1. *Let $(s(x))_{x\in V}$ be a divisible sandpile as in Theorem 2. If at any timestep t we find $x \sim y$ such that $s_t(x) > s_t(y) \geq 1$, then we have that $s_{t+1}(y) > 1$. As a consequence, the sandpile does not stabilize in finite time.*

Proof. Assume two neighbouring nodes $x, y \in \mathbb{Z}_n^d$ exist such that $x \sim y$ and at time $t \geq 0$ $s_t(x) > s_t(y) \geq 1$. Then for some $\delta > 0$ $s_t(x) = 1 + \delta$. Now $s_t(x) > 1$, so the site is unstable and topples in this timestep, distributing its excess mass equally over its $2d$ neighbours. Then since $s_t(y) \geq 1$ we find $s_{t+1}(y) \geq 1 + \frac{\delta}{2d} > 1$. This proves the claim. Now it continues for $t+2$ we have $s_{t+2}(x) \geq 1 + \frac{\delta}{4d^2} > 1$. Proceeding inductively, the result follows. $\square$

Now what is left to the proof of Theorem 2 is proving that these sites $x, y \in \mathbb{Z}_n^d$ always exist. Graaff [5] shows this by partitioning the graph strategically in sites with initial mass greater than 1, sites which receive mass in the first toppling, and lastly all the sites with mass ≤ 1 which don't receive any mass in the first toppling.

Theorem 3. *At timestep $t = 1$, there exist at least two $x, y \in \mathbb{Z}_n^d$ such that $x \sim y$ and $s_t(x) > s_t(y) \geq 1$.*

Proof. We start with defining the set V^+,

$$V^+ := \{z \in \mathbb{Z}_n^d : s(z) > 1\}.$$

This is the collection of all sites that will topple in the first iteration. V^+ must be non-empty, because if V^+ was empty then $s(z) < 1$ for all $z \in \mathbb{Z}_n^d$ following the fact that we have $\mathbb{P}(s(x) = 1) = 0$ for all $x \in \mathbb{Z}_n^d$. Now if $s(z) < 1$ for all $z \in \mathbb{Z}_n^d$, then $\sum_{z\in\mathbb{Z}_n^d} s(z) < n^d$, and this is a contradiction. Now we are looking for two $x, y \in V^+$ at $t = 1$ such that $x \sim y$, because then we can use Lemma 1. We will proof this by looking at the case where all $x \in V^+$ are isolated, assume there are no $x, y \in V^+$ such that $x \sim y$. This means that for all $x \in V^+$ we have $s(y) < 1$ for all $y \sim x$. Choose a random $x \in V^+$, we can either find a $y \sim x$ such that $s(y) + \frac{1}{2d}\sum_{z\sim y}(s(z) - 1)^+ > 1$, or we have for all $y \sim x$ that $s(y) + \frac{1}{2d}\sum_{z\sim y}(s(z) - 1)^+ \leq 1$. We will now show that the former case leads to the use of Lemma 1, while the latter leads to a contradiction almost surely for $n \geq 3$.
We will first look at the first case, so we assume that we can find $y \sim x$ such that $s(y) + \frac{1}{2d}\sum_{z\sim y}(s(z) - 1)^+ > 1$. Note that since $x \in V^+$ we have $s(x) > 1$, this sites will topple. Now because of our assumption we get the following at $t = 1$:

$$s_1(y) = s(y) + \frac{1}{2d}\sum_{z\sim y}(s(z) - 1)^+ > 1,$$

with $y \sim x$ and $x \in V^+$, then we can use Lemma 1 because also $s_1(x) \geq 1$.
Now we consider the second case, assume for all $x \in V^+$ and for all $y \sim x$ that $s(y) + \frac{1}{2d}\sum_{z\sim y}(s(z) - 1)^+ \leq 1$. This means that after the first toppling all neighbours to any site in V^+ still have subcritical mass. In this case we either find a $y \sim x$ for some $x \in V^+$ such that

$$s(y) + \frac{1}{2d}\sum_{z\sim y}(s(z) - 1)^+ < 1,$$

or for all $x \in V^+$ and $y \sim x$ we have

$$s(y) + \frac{1}{2d}\sum_{z\sim y}(s(z) - 1)^+ = 1.$$

This latter case means we have reached a stable situation after the first toppling, we will prove in the next theorem Theorem 4 that this will almost surely not happen for $n \geq 3$. The former case leads to a contradiction. To prove this we define:

$$\widetilde{V}^+ := V^+ \cup \{z \in \mathbb{Z}_n^d : z \sim x \text{ for some } x \in V^+\} = V^+ \cup \{\text{all neighbours of } V^+\}. \tag{13}$$

Note that we can now use

$$\sum_{z \in \mathbb{Z}_n^d} s(z) = \sum_{z \in \widetilde{V}^+} s(z) + \sum_{z \in \mathbb{Z}_n^d \setminus \widetilde{V}^+} s(z). \tag{14}$$

By our assumption we have for all $z \in \mathbb{Z}_n^d \setminus \widetilde{V}^+$ that $s(z) < 1$, and thus

$$\sum_{z \in \mathbb{Z}_n^d \setminus \widetilde{V}^+} s(z) \leq |\mathbb{Z}_n^d \setminus \widetilde{V}^+| \leq n^d - |\widetilde{V}^+|. \tag{15}$$

Equality only occurs in the case where the set $\mathbb{Z}_n^d \setminus \widetilde{V}^+$ is empty. Now, if all $z \in V^+$ topple, they distribute their mass over their neighbours. And by the definition of $\widetilde{V}^+$ we have that these neighbours are in $\widetilde{V}^+$, so we get

$$\sum_{z \in \widetilde{V}^+} s(z) = \sum_{z \in \widetilde{V}^+} s_1(z). \tag{16}$$

We have already assumed that all $z \in V^+$ are isolated, so they do not receive any mass from toppling neighbours, then

$$\sum_{z \in \widetilde{V}^+} s_1(z) = \sum_{z \in V^+} s_1(z) + \sum_{z \in \widetilde{V}^+ \setminus V^+} s_1(z), \tag{17}$$

$$= |V^+| + \sum_{z \in \widetilde{V}^+ \setminus V^+} \left[s(z) + \frac{1}{2d} \sum_{x \sim z} (s(x) - 1)^+ \right], \tag{18}$$

$$< |V^+| + |\widetilde{V}^+| - |V^+| = |\widetilde{V}^+|. \tag{19}$$

The inequality follows from the fact that by assumption at least one $y \in \widetilde{V}^+$ with $s(y) + \frac{1}{2d} \sum_{z \sim y} (s(z) - 1)^+ < 1$. Now combining this result with Equation 14 and Equation 15, we can obtain

$$\sum_{z \in \mathbb{Z}_n^d} s(z) < |\widetilde{V}^+| + n^d - |\widetilde{V}^+| = n^d. \tag{20}$$

However, this is a contradiction as $\sum_{z \in \mathbb{Z}_n^d} s(z) = n^d$ by definition of s. $\square$

What is left to prove now is that the case where for all $x \in V^+$ and $y \sim x$ we have $s(y) + \frac{1}{2d} \sum_{z \sim y} (s(z) - 1)^+ = 1$ almost surely does not happen. To do this we will use the same notation as above in the following theorem

Theorem 4. *For $x \in V^+$ and $y \sim x$*

$$\mathbb{P}\left(s(y) + \frac{1}{2d} \sum_{z \sim y} (s(z) - 1)^+ = 1 \right) = 0,$$

for $n \geq 3$.

Proof. For more convenience we define the following:

$$\Lambda(x) := \sigma(x) - \frac{1}{n^d} \sum_{w \in \mathbb{Z}_n^d} \sigma(w).$$

Note that $\Lambda(x) = s(x) - 1$, and using this new notation we find

$$\mathbb{P}\left(s(y) + \frac{1}{2d} \sum_{z \sim y} (s(z) - 1)^+ = 1 \right) = \mathbb{P}\left(\Lambda(y) + \frac{1}{2d} \sum_{z \sim y} \mathbb{1}_{\Lambda(z) > 0} \Lambda(z) = 0 \right).$$

Next we enumerate the neighbours of y from 1 to $2d$. It then follows that proving the above probability is equal to zero is equivalent to proving that for any combination of neighbours $x_1, \ldots, x_k$ of y with $1 \leq k \leq 2d$, we have

$$\mathbb{P}\left(\Lambda(y) + \frac{1}{2d} \sum_{j=1}^{k} \Lambda(x_j) = 0 \right) = 0.$$

To find this probability on the left hand side we will rewrite it.

$$\begin{aligned} \Lambda(y) + \frac{1}{2d} \sum_{j=1}^{k} \Lambda(x_j) &= \sigma(y) - \frac{1}{n^d} \sum_{w \in \mathbb{Z}_n^d} \sigma(w) + \frac{1}{2d} \sum_{j=1}^{k} \left(\sigma(x_j) - \frac{1}{n^d} \sum_{w \in \mathbb{Z}_n^d} \sigma(w) \right), \\ &= \sigma(y) + \frac{1}{2d} \sum_{j=1}^{k} \sigma(x_j) - \left(1 + \frac{k}{2d} \right) \frac{1}{n^d} \sum_{w \in \mathbb{Z}_n^d} \sigma(w). \end{aligned}$$

Because we are trying to find the probability that this equals zero we can multiply everything by n^d and rewrite to the following:

$$\left(n^d - \left(1 + \frac{k}{2d} \right) \right) \sigma(y) + \left(\frac{n^d}{2d} - \left(1 + \frac{k}{2d} \right) \right) \sum_{j=1}^{k} \sigma(x_j) - \left(1 + \frac{k}{2d} \right) \sum_{w \in \mathbb{Z}_n^d \setminus \{x_1, \ldots, x_k, y\}} \sigma(w).$$

Note that this is a sum of i.i.d. Gaussian random variables. This is only zero if all the coefficients are equal to zero, or if for all x we have $\sigma(x) = 0$ but this has probability of zero. However, since $n \geq 3$, and $k \leq 2d$, the term $n^d - \left(1 + \frac{k}{2d}\right)$ is always greater than 0. This leads to the fact that we wanted to prove, $\mathbb{P}\left(s(y) + \frac{1}{2d} \sum_{z \sim y} (s(z) - 1)^+ = 1 \right) = 0.$ □

These results tell us that when we do the computer simulation we will never truly reach a stable state. In our simulations we will only be able to simulate for a finite time, thus these previous results mean that our simulations will never truly be accurate. This will be further discussed in section 6.

4 Adding the constant

4.1 Intro: modifying the initial mass

As stated previously, the goal of this research is to investigate the influence of adding a factor c to Equation 7. Adding this constant alters the formula of the initial distribution of the sandpile only slightly, but it has great impact. For convenience we will state the formula for the initial sandpile configuration again with the introduced constant.

$$s(x) = 1 + \sigma(x) - \frac{c}{|V|}\sum_{y\in V}\sigma(y).$$

Note that if c is equal to 1 we get $\sum_{x\in V} s(x) = |V|$. We have previously proven in Proposition 1 that in this case the sandpile converges to the all-1-configuration. However if c is not equal to 1 we get

$$\begin{aligned}\sum_{x\in V} s(x) &= \sum_{x\in V}\left(1 + \sigma(x) - \frac{c}{|V|}\sum_{y\in V}\sigma(y)\right)\\ &= |V| + \sum_{x\in V}(\sigma(x)) - \frac{c}{|V|}\sum_{x\in V}\sum_{y\in V}\sigma(y)\\ &= |V| + \sum_{x\in V}(\sigma(x)) - \frac{c|V|}{|V|}\sum_{y\in V}\sigma(y)\\ &= |V| + (1-c)\sum_{x\in V}\sigma(x).\end{aligned} \tag{21}$$

Therefore lemma 7.1 from Levine et al. [8] no longer holds unless $c = 1$, which is the original case, or $\sum_{x\in V}\sigma(x) = 0$. Moreover, in a similar way we find for the expected value of $s(x)$

$$\begin{aligned}\mathbb{E}(s(x)) &= \mathbb{E}\left(1 + \sigma(x) - \frac{c}{|V|}\sum_{y\in V}\sigma(y)\right)\\ &= \mathbb{E}(1) + \mathbb{E}(\sigma(x)) - \frac{c}{|V|}\sum_{y\in V}\mathbb{E}(\sigma(y))\\ &= 1 + \mathbb{E}(\sigma(x)) - \frac{c|V|}{|V|}\mathbb{E}(\sigma(x))\\ &= 1 + (1-c)\mathbb{E}(\sigma(x)).\end{aligned}$$

This means that if $\mathbb{E}(\sigma(x)) = 0$ then $\mathbb{E}(s(x)) = 1$ and the sandpile could stabilize. On the other hand, if $\mathbb{E}(\sigma(x)) \neq 0$ then the value of c determines if the sandpile stabilizes. It is proven in Lemma's 4.1 and 4.2 in Levine et al. [8] that if $\mathbb{E}(s(x)) > 0$, then $\mathbb{P}(s \text{ stabilizes}) = 0$, and if $\mathbb{E}(s(x)) < 0$, then $\mathbb{P}(s \text{ stabilizes}) = 1$.

4.2 Subcritical

The problem with the introduced factor is the fact that the sum $\sum_{y\in V}\sigma(y)$ is random. This leads to three possible situations, $\sum_{y\in V}\sigma(y)$ is either positive, negative or equal to zero. Because of what Levine et al. [8] have proven about the stabilization of the sandpile at critical density, we require that the $\sum_{y\in V}\sigma(y) \geq 0$. Otherwise, if $\sum_{y\in V}\sigma(y) < 0$ the sandpile will be supercritical because $\sum_{x\in V} s(x)$ will be bigger than $|V|$ for $c > 1$ as can be seen in Equation 21. And we

know that the sandpile will not stabilize if $\sum_{x\in V} s(x) > |V|$.
Now if we have $\sum_{y\in V}\sigma(y) = 0$, then we multiply c by zero so it has no influence on the sandpile at all. Therefore in our computer simulation we will only consider the cases where $\sum_{y\in V}\sigma(y) > 0$ because we want the sandpile to be subcritical as it needs to stabilize.
Because we only use $\sum_{y\in V}\sigma(y) > 0$ there is not enough mass in the sandpile to converge to the all-1-configuration. Which means it will stabilize, we just don't know what the stabilized sandpile will look like.

4.3 Previous proofs

With our added factor c a lot of proofs from section 3 will no longer hold. We will shortly go over where they fail to hold.
Starting at Theorem 1, this theorem fails immediately in the prerequisite of $\sum_{x\in V} s(x) = |V| = n$. This is because in the proof it is used when proving that $s + \Delta u = 1$ must hold. In our case we have $\sum_{x\in V} s(x) = n + (1-c)\sum_{x\in V}\sigma(x)$, which means the sandpile no longer has critical density. And as a result we will see in the computer simulations that the sandpile will indeed no longer stabilize to the all-1-configuration like Theorem 1 states.
Next Proposition 1 follows from Theorem 1, so the failure of the latter makes the proof of the former no longer hold. This is seen in the proof as it is necessary right at the start to state that the sandpile converges to the all-1-configuration. Later in the proof where the expected value of $s(x)$ is used to prove that $u = \eta - \min\eta$ using the covariance of $v(x)$ we get the following difference:

$$\begin{aligned}
\mathbb{E}[(s(z)-1)(s(w)-1)] &= \mathbb{E}\left[\left(\sigma(z) - \frac{c}{n}\sum_{x\in V}\sigma(x)\right)\left(\sigma(w) - \frac{c}{n}\sum_{x\in V}\sigma(x)\right)\right], \\
&= \mathbb{E}\left(\sigma(z)\sigma(w) - \frac{c}{n}\sigma(w)\sum_{x\in V}\sigma(x) - \frac{c}{n}\sigma(z)\sum_{x\in V}\sigma(x) + \frac{c^2}{n^2}\left(\sum_{x\in V}\sigma(x)\right)^2\right), \\
&= \mathbb{E}(\sigma(z)\sigma(w)) - \frac{c}{n}\mathbb{E}\left(\sigma(w)\sum_{x\in V}\sigma(x)\right) - \frac{c}{n}\mathbb{E}\left(\sigma(z)\sum_{x\in V}\sigma(x)\right) + \frac{c^2}{n^2}\mathbb{E}\left(\left(\sum_{x\in V}\sigma(x)\right)^2\right), \\
&= \mathbb{1}_{z=w} - \frac{c}{n} - \frac{c}{n} + \frac{c^2}{n} = \mathbb{1}_{z=w} + \frac{c^2-2c}{n}.
\end{aligned} \tag{22}$$

This has consequences for the covariance of the function v, we can continue:

$$\begin{aligned}
\mathbb{E}(v(x)v(y)) &= \frac{1}{\deg(x)\deg(y)}\sum_{z,w\in V} g(z,x)g(w,y)\mathbb{E}((s(z)-1)(s(w)-1)), \\
&= \frac{1}{\deg(x)\deg(y)}\left(\sum_{z\in V} g(z,x)g(z,y) + \frac{c^2-2c}{n}\left(\sum_{z\in V} g(z,x)\right)\left(\sum_{w\in V} g(w,y)\right)\right).
\end{aligned} \tag{23}$$

Now let again $K(y) := \frac{1}{\deg(y)}\sum_{w\in V} g(w,y)$, then again $\Delta K = \sum_{z,w\in V}\frac{1}{n}(\mathbb{1}_z - \mathbb{1}_w) = 0$. So K is constant. The second term on the right this time equals $\frac{K^2(c^2-2c)}{n}$. Consequentially setting $C \sim N(0, \frac{K^2(c^2-2c)}{n})$ a slightly different normal distributed random variable independent of v.

Now the Gaussian vectors η and $(v(x)+C)_{x\in V}$ have again the same covariance because

$$\begin{aligned}\mathbb{E}((v(x)+C)(v(y)+C)) &= \mathbb{E}(v(x)v(y)+Cv(x)+Cv(y)+C^2),\\ &= \mathbb{E}(v(x)v(y))+\mathbb{E}(Cv(x))+\mathbb{E}(Cv(y))+\mathbb{E}(C^2),\\ &= \frac{1}{\deg(x)\deg(y)}\left(\sum_{z\in V} g(z,x)g(z,y)\right) - \frac{K^2(c^2-2c)}{n} + 0 + 0 + \frac{K^2(c^2-2c)}{n},\\ &= \frac{1}{\deg(x)\deg(y)}\left(\sum_{z\in V} g(z,x)g(z,y)\right) = \mathbb{E}(\eta(x)\eta(y)).\end{aligned}$$

Which means the law is the same as in Proposition 1, only the sandpile does not conververge to the all-1-configuration.

Because of these results it is not possible to prove the stabilization of the sandpile with the added constant c in the same way as before. That is why we want to look at it through computer simulations.

4.4 Influence on the Normal distribution

By introducing c in Equation 8 we multiply a sum of random variables $\sigma(y)$ by a scalar c. In the case where the σ are normally distributed we can use the following property of the normal distribution [10]:

Proposition 2. *If $X \sim N(\mu,\gamma^2)$ and $Y = aX + b$, then $Y \sim N(a\mu + b, a^2\gamma^2)$.*

In most of the simulation study we look at the standard normal $N(0,1)$ distribution. This property of the normal distribution then means that the $\sigma(y)$ variables inside the sum are $N(0,c^2)$ distributed because we can take c into the sum as following:

$$s(x) = 1 + \sigma(x) - \frac{1}{|V|}\sum_{y\in V} c\sigma(y). \tag{24}$$

Because of this, if $c > 1$ the introduction of c increases the variance of the $\sigma(y)$.
Note that this is not the same as leaving c out of Equation 8 and simulating $\sigma \sim N(0,c^2)$. This is clear from the fact that only the $\sigma(y)$ in Equation 24 get multiplied by c and not the $\sigma(x)$. However, this result might allow us to understand the results of the simulation study better later on.

5 Simulation Study

To investigate the influence of our constant c a simulation study was done. The MATLAB code can be found in the appendix. It consists of four different files, we will go through them one by one. However before we go into the different files we will explain the how the DSM is modelled, this is then the same in every file.

5.1 Modelling the DSM

To be able to do the simulation study a good written code is required. We will need to simulate multiple times and if the code is slow it will take even more time. The code consists of 4 important parts, and the first three of these are exactly the same in all files.

The first part is where all the variables and other settings are defined. This allows us to make quick changes to the program before running it. The variables that are possible to change are for example the size of the torus, the amount of iterations, and the values of c we want to test.

The second part consist of generating the values of the $\sigma(x)$. The way this was implemented allows us to choose between 5 different distributions: Normal, Gamma, Cauchy, Pareto, and Poisson. All with their own parameters. For Pareto we took the shape parameter α to be in $(1, 2)$ this way it has finite mean and infinite variance.
The value of $\sum_{y \in V} \sigma(y)$ is calculated because it needs to be positive. If it is negative then the volume of the torus will become supercritical when c is increased.

The third part consists of the toppling procedure. This is the part that takes almost all of the time because it requires the most amount of calculations. Unfortunately the toppling procedure isn't a simple procedure, it is not possible to do this with for example matrix multiplications, which MATLAB can do very fast. Instead we use 4 `for` loops, the first is a loop over all values of c we want to investigate, the second is for the timesteps, and the third and fourth loop over every node on the torus. Then within the fourth loop we first check if the current value of the node is higher than our cutoff of 1. Then it distributes a quarter of that excess value over its neighbours. And finally the odometer and torus are updated.

The fourth part is different for every file as it defines what we want to investigate. It can be data, plots of the torus, or different calculations of test statistics.

In general, the torus we work on is of size 400×400. This size is small enough to allow quick calculations, yet it is large enough to allow no local outliers to interfere with the whole torus. No problems were encountered that could be blamed on the size of the torus.

5.2 Visualisation of the convergence of the sandpile

At first we visualised the convergence of the torus and the odometer. In MATLAB we were able to do that by plotting our sandpile every 10 thousand timesteps and then play the graphs in succession as if it is a movie. On paper that is much harder, but you can get the idea by looking at the following figures. In this case the standard normal distribution was used to generate the values of $\sigma(x)$.

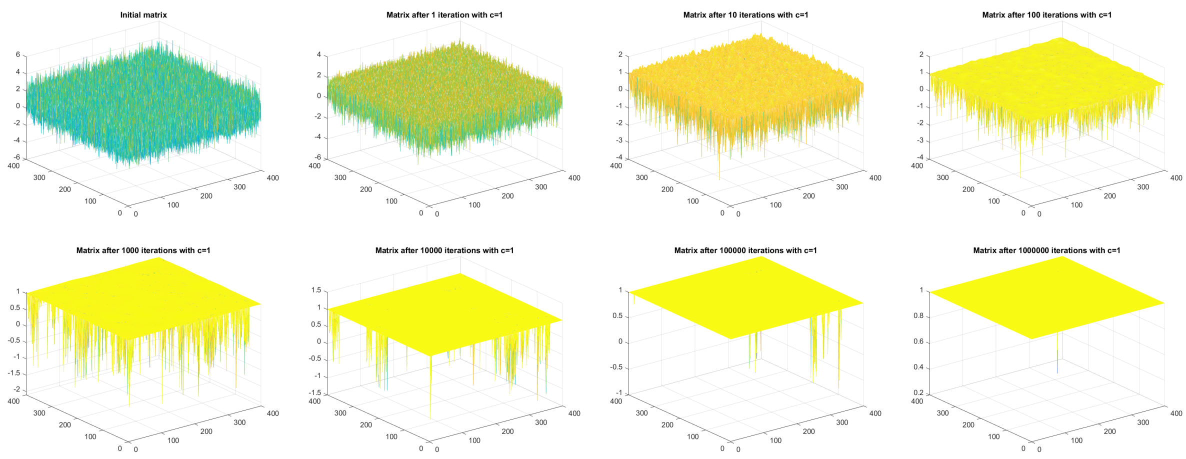

Figure 3: Development of stabilization of a sandpile with $c = 1$.

In Figure 3 the progression of the sandpile is easy to see, it starts with all random values in the interval $[-6, 6]$, after the first timestep most values are already much closer to 1 as the interval has now become $[-6, 4]$. This interval keeps on becoming smaller and smaller, after 100,000 timesteps the range of the figure has become $[-1, 1]$, in reality at that moment the largest value of the torus is 1.00073 and the smallest value is -0.97373.
It is also very visible in the last few graphs of Figure 3 that the points of the torus that are not yet close to the stable value of 1 seem to occur in clusters of points. We do not yet have an explanation for this property but it seems to be related to the location of the minimum value in the initial torus.

Note that in this situation we used $c = 1$, so this is what a normal progression of the sandpile should look like. In the next few subsections we will alter c to see its influence.

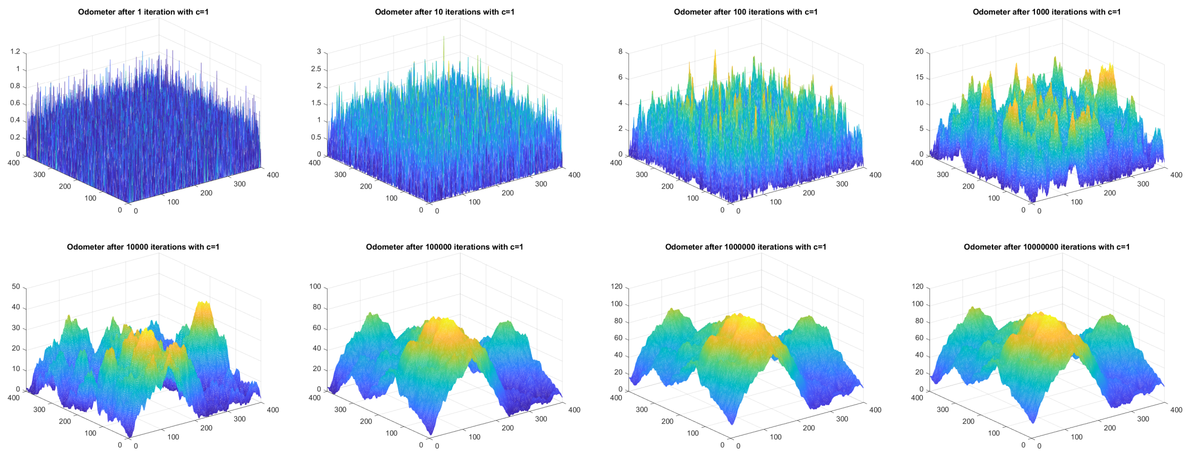

Figure 4: Development of the odometer during the stabilization of a sandpile with $c = 1$.

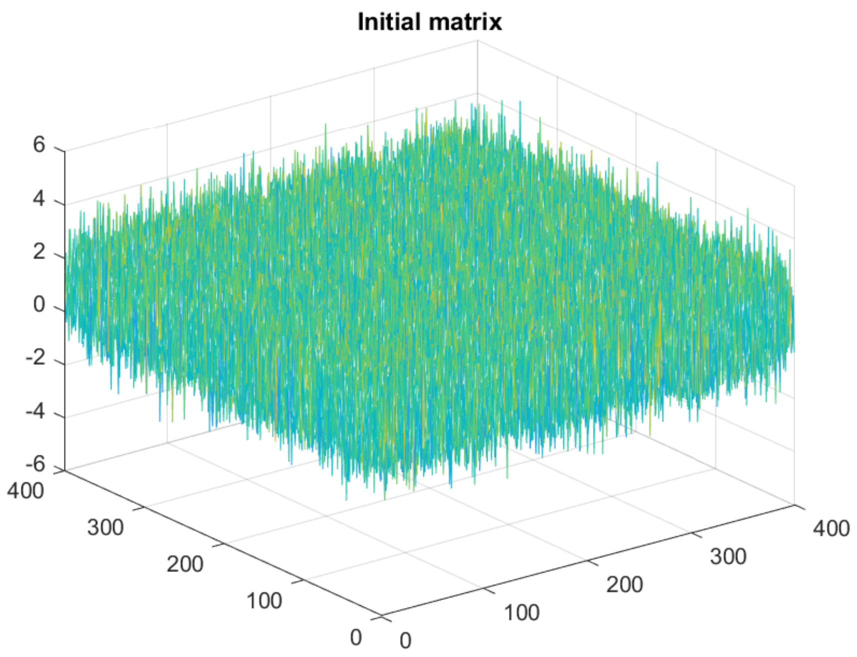

Figure 5: Initial matrix representing the sandpile.

In Figure 4 the development of the odometer of the sandpile is shown. It is clear to see that at first the odometer doesn't show signs of a pattern, in the first two graphs all values are around the same value and there are not a lot of outliers to detect. In the later figures it is very clear that the odometer is less random. The value of the odometers of neighbouring points is very closely related. There is also a relation between the location of the last clusters and the height of the odometer there. Since the points that remain below 1 for a long time have not emitted any weight to their neighbours the odometer at that location is 0.

5.3 Influence on the convergence

Next we looked into the influence of c on how good the sandpile converges. Ideally the DSM converges to the all-1-configuration, where all the nodes have weight equal to 1. Since we have proven that this does not happen in finite time we cannot simulate until we reach this all-1-configuration. Therefore we define a convergence interval $D = (0.995, 1.005)$. Then we can simulate N timesteps and look at how many nodes of the graph lie within D.

The amount of iterations was generally first set at 20000, this was arbitrarily chosen based on the amount of iterations it took to reach 95% of nodes in the convergence interval. For example the standard normal distributions took around a 17000 iterations to reach this percentage with $c = 1$. The value of c was varied mostly between 1 and 5 with steps of 1, or between 1 and 2 with steps of 0.1. Only on a few occasions it was necessary to look at $c > 5$.

In this file a few test statistics were created. To denote the amount of values inside the interval D at every timestep a was used, while b denoted the percentage of values in the interval. Also e was used to denote the amount of values below the interval, and h was used to denote the amount of values above the interval. This allows us to roughly see to what kind of state the torus converges, it was found that if c increases then e increases too, while h was almost always

0. But more on that in the next chapter.

Before we move on it must be noted that even though the mass in the system is much lower when c is increased, the sandpile most likely does not stabilize completely in finite time. That is because the argumentation of Lemma 1 still holds. If there are at any time two neighbouring nodes with $s_t(x) > 1$ then there will always be mass that remains moving between the nodes at every timestep. Theoretically it is possible that by our introduction of c we remove so much mass that only a handful of nodes are left with mass greater than 1, and there are no neighbouring nodes that both have mass greater than 1. But in practice this means we have increased c so far that it is no longer useful. And as soon as the mass starts spreading around it might still occur that two neighbouring nodes get $s_t(x) > 1$. Therefore it is safe to conclude that the sandpile still does not stabilize fully in finite time, the only way this will happen in practice is if we increase c to the point that there is almost no mass left in the system.

5.4 Convergence speed

What we noticed was that the amount of nodes in the convergence interval decreased for higher c, the amount of timesteps it took to get to a reasonable converged state seemed to be shorter for higher c. To check this we will also look at the first timestep t_D where 95% of all nodes lie within D. This test statistic will tell us about the speed of convergence, however there is a danger in using this interval D. That is because if we are decreasing the total volume of the graph then it is possible that there is not enough volume left to get 95% of all nodes in the interval. This happens when c gets bigger, therefore we will also look at the first timesteps where 95% and all nodes are in the interval $(-\infty, 1.005)$.

5.5 Odometer

To find out how c influences the odometer we looked at two things. We plotted the different odometers to visually try to understand the difference, and we calculated the mean and variance of the odometer for every value of c. This seemed to be the best way to find a relation between c and the odometer.
We did this by calculating

$$\mu_{u_t} = \frac{1}{|V|} \sum_{y \in V} u_t(y), \tag{25}$$

and

$$\sigma^2_{u_t} = \frac{1}{|V|-1} \sum_{y \in V} (u_t(y) - \mu_{u_t})^2 \tag{26}$$

at every timestep. Understanding these values gives us an idea of the influence of c at the convergence of the odometer. In an ideal scenario the odometer converges, however this is also not possible in finite time.

It should also be noted that it was proven by Cipriani, Hazra, and Ruszel [3] that the distribution of the odometer on a finite graph is known if $\sum_{x \in V} s(x) = |V|$. Moreover, they proved that if $s(x)$ is defined as Equation 7, and $(\sigma(x))_{x \in \mathbb{Z}_n^d}$ are i.i.d. and in the domain of normal attraction of a stable law, then the distribution of the odometer converges to a known distribution. To be able to use these results it needs to be reviewed whether the probability distributions used in this thesis lie in the domain of normal attraction of a stable law. But

before we can do that a few definitions need to be stated.
The domain of normal attraction of a stable law can be defined as following [3]:

Definition 19. *Let $\alpha \in (0,2]$. Let V be a countably infinite index set and $(W(x))_{x\in V}$ be i.i.d. symmetric random variables with common distribution function in the domain of normal attraction of an α-stable distribution. This means that, for $V_1 \subset V_2 \subset \dots$ such that $\cup_{k\geq 1} V_k = V$, we have the following limit:*

$$\lim_{k\to+\infty} |V_k|^{-\frac{1}{\alpha}} \sum_{x\in V_k} W(x) = \rho_\alpha, \tag{27}$$

where ρ_α has a symmetric α-stable law.

A distribution is called stable if it satisfies the following definition [9]:

Definition 20. *A random variable Y and its distribution are said to be stable if for i.i.d. copies Y_1, Y_2 of Y and all choices of non-negative constants c_1, c_2 there exist numbers $a > 0$ and $b \in \mathbb{R}$ such that the following identity in law holds:*

$$c_1 Y_1 + c_2 Y_2 \overset{d}{=} aY + b. \tag{28}$$

It has been shown by Mikosch [9] that the Gaussian and Cauchy distributions are stable distributions in itself, and therefore automatically lie in the domain of attraction of an α-stable distribution. Furthermore for the distributions with finite variance the Central Limit Theorem can be used, because this theorem states that every distribution with finite variance can be rescaled to the standard normal distribution in law. Therefore all distributions with finite variance lie in the domain of normal attraction of the standard normal distribution.
The only remaining distribution is the Pareto distribution. We have chosen the shape parameter such that the variance is not finite by setting $1 < \alpha_P < 2$. To be able to show that the Pareto distribution lies in the domain of attraction of an α-stable law we will use a theorem of Mikosch [9] which states the following:

Theorem 5. *The distribution F belongs to the domain of attraction of an α-stable law for some $\alpha < 2$ if and only if:*

$$F(-x) = \frac{q + o(1)}{x^\alpha} L(x), \quad 1 - F(x) = \frac{p + o(1)}{x^\alpha} L(x), \quad x \to \infty, \tag{29}$$

where L is a slowly varying function and p, q are non-negative constants such that $p + q > 0$.

Note that $L(x)$ is said to be slowly varying if $\lim_{x\to\infty} \frac{L(tx)}{L(x)} = 1$ for all $t > 0$ [9], and note that $o(1)$ denotes the little-o-notation. Now set $L(x) = 1$, then we need to find p, q and α such that $\lim_{x\to\infty} F(-x) = \frac{q}{x^\alpha}$ and $\lim_{x\to\infty} 1 - F(x) = \frac{p}{x^\alpha}$ for the Pareto distribution. Note that for this distribution we have $F(x) = 1 - \left(\frac{x_m}{x}\right)^{\alpha_P}$, where $x_m > 0$, $x \in [x_m, \infty)$ and in this thesis α_P was chosen to be in the interval $1 < \alpha_P < 2$ such that the distribution has finite mean and infinite variance. This gives the following results:

$$\lim_{x\to\infty} F(-x) = F(x_m) = 1 - \left(\frac{x_m}{x_m}\right)^{\alpha_P} = 0.$$

Therefore we find $\lim_{x\to\infty} F(-x) = \frac{q}{x^\alpha} \Leftrightarrow q = 0$. Furthermore:

$$\lim_{x\to\infty} 1 - F(x) = \left(\frac{x_m}{x}\right)^{\alpha_P} = \frac{x_m^{\alpha_P}}{x^{\alpha_P}}.$$

Now setting $\lim_{x\to\infty} 1 - F(x) = \frac{p}{x^\alpha}$ gives $\frac{x_m^{\alpha_P}}{x^{\alpha_P}} = \frac{p}{x^\alpha}$, therefore $p = x_m^{\alpha_P}$ and $\alpha = \alpha_P$ solves the equation. We have $x_m > 0$ and $1 < \alpha_P < 2$, therefore $p > 0$. These results allow us to use Theorem 5 to prove that the Pareto distribution used in this thesis belongs to the domain of attraction of an α-stable law.
Therefore for all the distributions we use in this thesis we can use the results of Cipriani, Hazra, and Ruszel [3].

These results mean that for $c = 1$ we know what the odometer should look like in the limit, however if $c \neq 1$ then we no longer have $\sum_{x\in V} s(x) = |V|$. Therefore we do not yet know what the odometer will look like if we change c, that is something we want to investigate by doing the simulation study.

6 Results

Altering c and making plots can give us a little insight already in the influence of adding the constant. It seemed like the constant greatly influenced the starting torus to be closer to zero, but also more negative. This can be explained by reasoning that by increasing c in equation Equation 7 we subtract a greater factor of

$$\frac{1}{|V|}\sum_{y\in V}\sigma(y). \tag{30}$$

Now if the initial distribution has mostly high values, then the value of Equation 30 will be relatively high too. When we use a distribution with strictly positive values like for example the Gamma distribution, then the value of Equation 30 is never close to zero. If c is increased then we subtract more and more, which gets those high values closer to zero. However in some cases we found that if we do this then all values will lie below 1 in the initial torus. For example, in one case of the Gamma distribution we used $\lambda = 1$ and $\alpha = 5$, for $c = 1$ we found $t_D = 31465$. We then increased c by one four times until $c = 5$, for every value of c the sandpile never reached 95% of nodes in D. Moreover, with $c = 5$ we get the following initial torus:

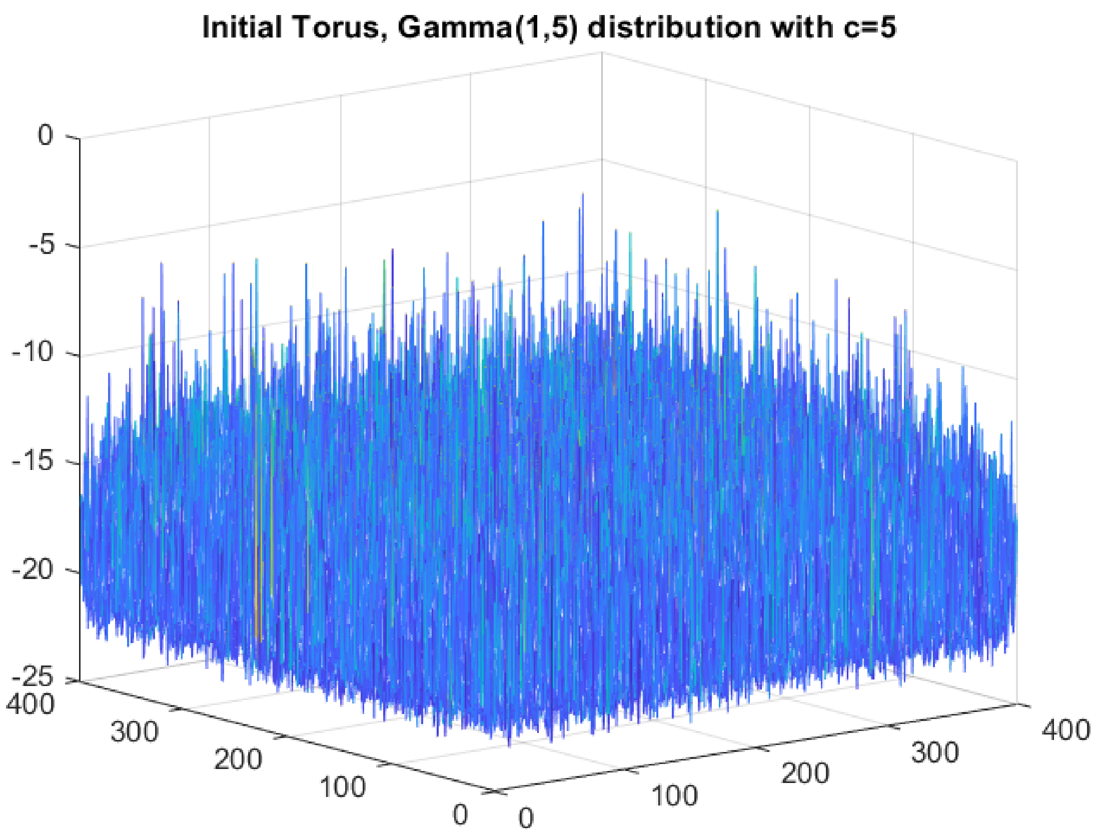

Figure 6: Initial torus with Gamma(1,5) and $c = 5$.

This means that the entire torus is negative and no toppling will happen. In one way this means that the torus is immediately stable! However this is not the way we want it to converge. This shows us that we should be careful when increasing c, and the way we use c might not be ideal. It might be better to increase c by smaller amounts than we are doing.

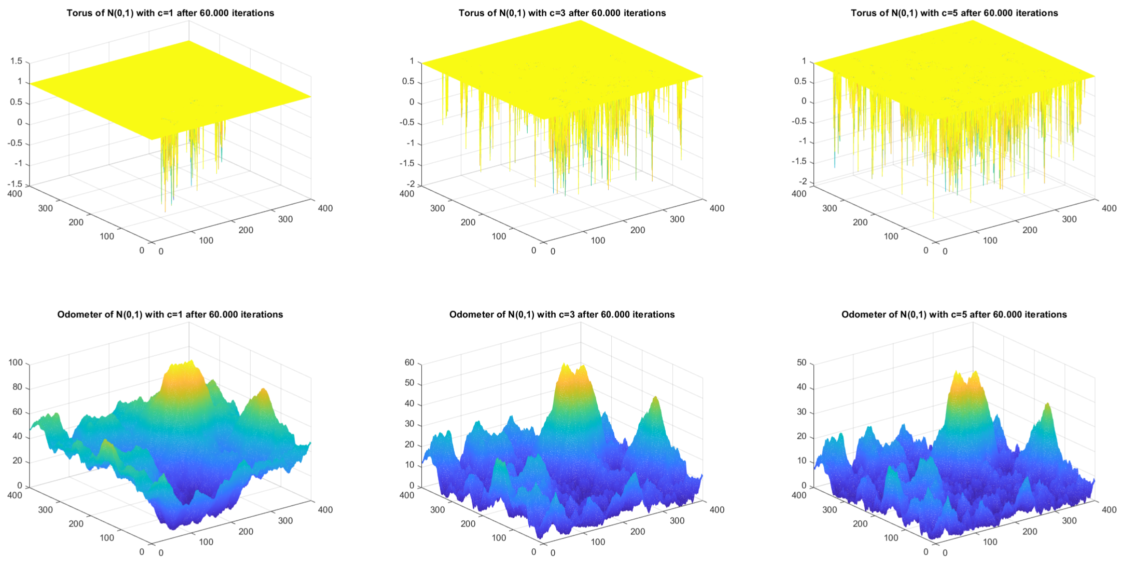

Figure 7: Influence of c on the converged sandpile and the odometer after 60,000 timesteps. Using N(0,1) and $c = 1$, $c = 3$, and $c = 5$.

Figure 7 shows the influence of c on the sandpile and odometer. In this figure the standard normal distribution was used to generate the values of $\sigma(x)$ and the value of c was between 1 and 5. The sandpile with $c = 1$ is close to a converged state, almost all values are very close to 1 and only a few clusters of nodes remain negative. When comparing that to the other two sandpiles it is clear to see that they are much further away from the all-1-configuration. This follows from our introduction of c, the volume is much lower so a lot of nodes will remain negative in a converged state.
If we look at the odometers it stands out that they all roughly have the same shape, they all have the highest peak in the middle and a smaller one at the edge. The difference is that for $c = 3$ and $c = 5$ the whole graph is much lower. And since the odometer is always positive a lot of the smaller values are compressed to a value very close or equal to zero. This is because in the initial torus a lot more nodes are negative if we increase c. Therefore those small values will not distribute any mass until they get a lot from their neighbours. However since there is less mass in the system this might not even happen at all.

6.1 Convergence

After looking at the plots it was clear that c influences the convergence greatly. But to get real data we will have to run the model multiple times and then look at the test statistics we devised.

First of all we looked at the standard normal distribution. This distribution is very easy to work with as it has mean 0 and low variance. Therefore it converges relatively quickly when compared to the other four distributions we tested. Because of this we can use a low amount of iterations and do this multiple times to take the average to get a more balanced result. In Table 1 we ran the program 10 times and took the average of the different test statistics. What is very clear in this table is that t_D decreases if c increases, which is exactly what we are looking

for. This is a very logical consequence of decreasing the total volume of the system. Since a lot less nodes have excess mass that they need to get rid of, and a lot more nodes don't have enough mass. So those nodes will take on that excess mass of their neighbours without starting to topple the next iteration.
Another thing that is visible in this table is that the amount of nodes in our convergence interval D increases at first, but decreases for $c > 3$. However we can also see that for $c \geq 3$ all nodes are in $(-\infty, 1.005)$, this means that for these values of c the torus is very close to a stable state. This also shows that even for the standard normal distribution if we increase c too much then not enough mass remains in the system to get every node in D.
The conclusion we can take from this is that increasing c helps reaching a stable state quicker. Not a lot changes to the sandpile in the next iteration if all values are in $(-\infty, 1.005)$, so we can say that the sandpile has almost stabilized. However, the sandpile has not reached a converged state we can predict. It is very random and dependant on the initial distribution of the $\sigma(x)$.

	$c = 1$	$c = 2$	$c = 3$	$c = 4$	$c = 5$
t_D	16096.2	12406	10189.4	8664	7590.1
percentage of nodes in D	98.3	99.2	99.2	98.9	98.6
percentage of nodes in $(-\infty, 1.005)$	98.7	99.7	100	100	100

Table 1: Average of 10 standard normal distributions after 20,000 iterations.

For the following 4 distributions we also started with $c = 1$ and then increased it by one until $c = 5$. At first the amount of iterations was left unchanged, until we realised that this wasn't enough for most of the distributions. Only the Poisson distribution allowed a state close to stabilization before 20,000 iterations when $c = 1$, for all other values of c and all other distributions that are used we never reached 95% nodes in D. Therefore we increased the amount of iterations dramatically to 500,000, this should have been more than enough. The program had to run overnight to be able to compute this amount of iterations. We therefore only did this once for every distribution. In the recommendations in section 8 we will get back to this.
In Table 2 we look at the results when using the Poisson distribution to generate the values of $\sigma(x)$ we see that t_D is very low for $c = 1$ and almost all of the nodes are in D after 500,000 iterations. For all other values of c there is no value for t_D and this is simply because there are not enough nodes in D. The percentage of nodes in D after 500,000 iterations decreases by a lot for increased c. Since all nodes are in $(-\infty, 1.005)$ for every value of c it is clear that the sandpiles were very close to stable. The fact that t_D does not exist for $c > 1$ only means that the sandpile never gets close to the all-1- configuration. If we instead look at the first timestep when 95% of all nodes are in the interval $(-\infty, 1.005)$, then we get a completely different image of the situation. For $c = 3, c = 4$ and $c = 5$ we get this already in the first timestep, for $c = 2$ this is reached after 10 timesteps, and for $c = 1$ it is 11960 which is very close to t_D. Furthermore if we go one step further, to the first timestep for which all nodes are in the interval $(-\infty, 1.005)$ then we get 14759, 32, 14, 9 and 5 respectively. This tells us that the sandpile reaches a stable state really quickly. Although this seems very good, the question arises if this is still useful. Of course we can keep increasing c to get a faster stabilization because at one point c will be big enough such that all nodes will start inside $(-\infty, 1.005)$.

	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$
t_D	12368	-	-	-	-
percentage of nodes in D	99.9838	40.1819	9.5769	8.9306	1.4256
percentage of nodes in $(-\infty, 1.005)$	100	100	100	100	100
First time 95% in $(-\infty, 1.005)$	11960	10	1	1	1
First time all in $(-\infty, 1.005)$	14759	32	14	9	5

Table 2: Poisson distribution $\lambda = \frac{1}{2}$ after 500,000 iterations.

Next is the Gamma distribution, we looked at this distribution using parameters $\alpha = 5$ and $\lambda = 1$. The results in Table 3 look similar to the results of the Poisson distribution. Only for $c = 1$ we get a value for t_D and almost all of the nodes are in D. As c increases the amount of nodes in D decreases even more quickly than before. With 0 nodes in D for $c = 5$ we see that c is too big, from the beginning all values are smaller than 0.995. This is most likely because we multiply c by $\frac{1}{|V|}\sum_{y\in V}\sigma(y)$, and since $\mathbb{E}(\sigma(y)) = 5$ this sum is fairly large. Note that the law of large numbers tells us that $\frac{1}{|V|}\sum_{y\in V}\sigma(y)$ approaches $\mathbb{E}(\sigma(y))$ as V is large enough. However since we subtract a greater factor of $\frac{1}{|V|}\sum_{y\in V}\sigma(y)$ every time we increase c it is very influential how big $\mathbb{E}(\sigma(y))$ is. In the first case where we used the standard normal distribution we had $\mathbb{E}(\sigma(y)) = 0$, so we could increase c by a lot. But this is no longer possible for other distributions where the expected value is larger. We could have chosen a lower value for α or λ but we chose to show the results with these parameters because it shows why $\mathbb{E}(\sigma(y))$ is so important.

	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$
t_D	31465	-	-	-	-
percentage of nodes in D	99.9912	3.0500	0.0850	0.0006	0
percentage of nodes in $(-\infty, 1.005)$	100	100	100	100	100
First time 95% in $(-\infty, 1.005)$	31343	1	1	1	1
First time all in $(-\infty, 1.005)$	34042	9	2	1	1

Table 3: Gamma distribution $\alpha = 5$ and $\lambda = 1$ after 500,000 iterations.

The results in Table 4 are a bit different than before. This is because we used the Cauchy distribution which has no mean or variance defined. This means that the values of $\sigma(y)$ can be very extreme, and that is exactly what happened in this case as can be seen in Figure 8. This initial sandpile resulted in a ripple effect of the mass distributing itself over the rest of the graph. As a result this means that for $c = 1$ even after 500,000 iterations at every moment only 50% of the nodes lie in D and the other 50% are above that still carrying a bit of that mass emitted from the one extremely high value.
For $c \geq 2$ we see the results we expected, the percentage of nodes in D decreases as c increases. And when compared to the previous distributions more nodes remain in D. The table also shows that from the very beginning we have 95% of all nodes in $(-\infty, 1.005)$, but it still takes a lot of iterations to get all nodes in $(-\infty, 1.005)$. Increasing c from 2 to 5 makes it more than twice as fast, this is again a logical result of the decreased amount of mass in the system. All nodes are decreased by a lot, $\mathbb{E}(\sigma(y))$ might not exist for the Cauchy distribution but we found $\frac{1}{|V|}\sum_{y\in V}\sigma(y) = 3.7216$. This means that every time we increase c by one the weight of every node in the initial configuration goes down by almost 4. Which makes it so the ripple effect dies out much quicker.

	$c = 1$	$c = 2$	$c = 3$	$c = 4$	$c = 5$
t_D	-	-	-	-	-
percentage of nodes in D	50.0000	62.1631	43.0644	32.7356	26.2869
percentage of nodes in $(-\infty, 1.005)$	50.0006	100	100	100	100
First time 95% in $(-\infty, 1.005)$	1	1	1	1	1
First time all in $(-\infty, 1.005)$	-	157184	113383	88902	72423

Table 4: Cauchy distribution $\mu = 0$ and $\lambda = \frac{1}{2}$ after 500,000 iterations.

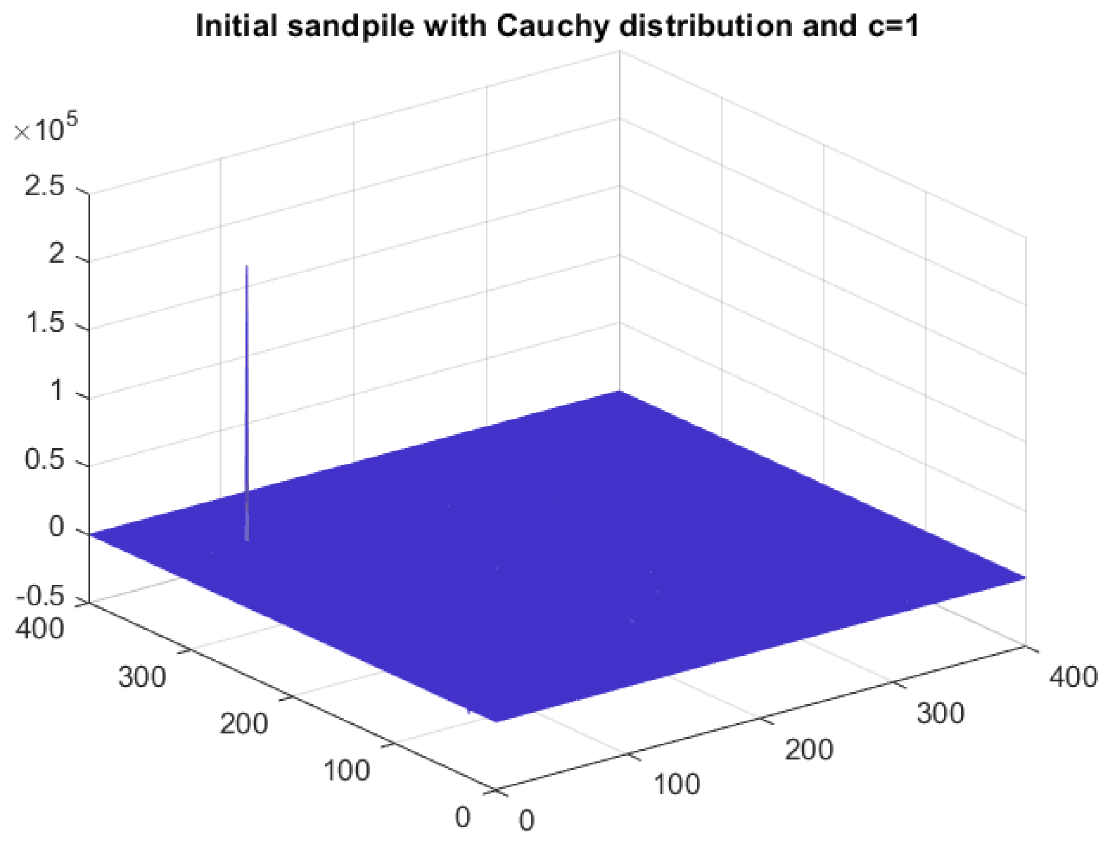

Figure 8: Initial torus with Cauchy distribution, $\mu = 0$ and $\lambda = \frac{1}{2}$ and $c = 1$.

Lastly we looked at the Pareto distribution, with parameters $\alpha = \frac{3}{2}$ and $x_m = 1$. These parameters were chosen such that the mean is finite, $\mathbb{E}(\sigma(x)) = 3$, and the variance is infinite. Just like with the Cauchy distribution we again have one very large outlier as can be seen in Figure 9, although this time it is much lower. This also results in a much quicker convergence as can be seen in Table 5. For $c = 1$ t_D is very high, but it exists. And the close to stable state with all nodes in $(-\infty, 1.005)$ is reached much quicker for all values of c. Moreover we see the same pattern again that by increasing c the percentage of nodes in D decreases and the first timestep when all nodes are in $(-\infty, 1.005)$ also decreases. What we can notice from these results is the fact that a large outlier is very influential to the convergence speed of the sandpile.

	$c = 1$	$c = 2$	$c = 3$	$c = 4$	$c = 5$
t_D	322793	-	-	-	-
percentage of nodes in D	99.8937	20.3238	9.6031	5.8800	4.0331
percentage of nodes in $(-\infty, 1.005)$	100	100	100	100	100
First time 95% in $(-\infty, 1.005)$	322557	27	1	1	1
First time all in $(-\infty, 1.005)$	327669	6720	3538	2418	1889

Table 5: Pareto distribution $\alpha = \frac{3}{2}$ and $x_m = 1$ after 500,000 iterations.

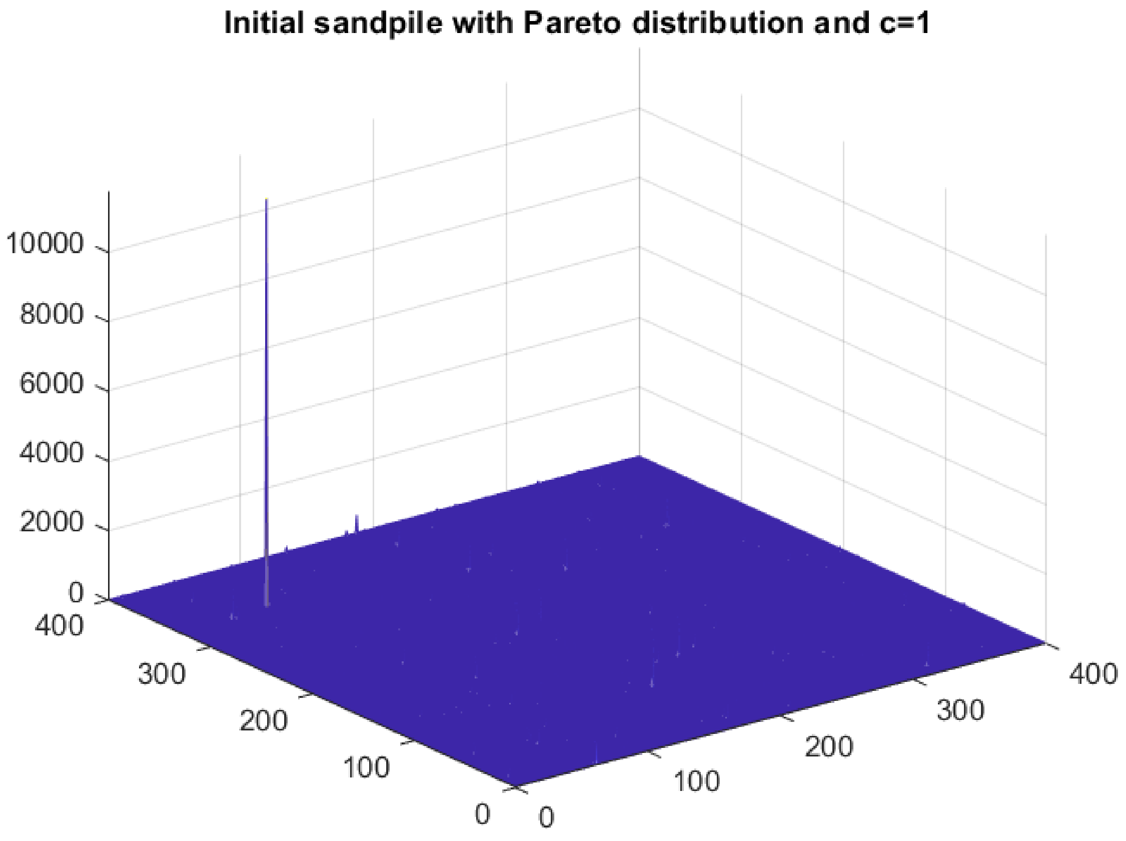

Figure 9: Initial torus with Pareto distribution, $\alpha = \frac{3}{2}$ and $x_m = 1$ and $c = 1$.

6.1.1 Relation between c and D

What we noticed in the previous section is that every time c was increased, the amount of nodes in D would decrease. After a sufficient amount of iterations all nodes would be in $(-\infty, 1.005)$, but only a fraction of that would remain in D. The interval D was constructed to look at how close a sandpile would get to the all-1-configuration, but it is clear that for higher c the sandpile gets nowhere close to this.
Because of this we are interested in the relation between c and the amount of nodes that remain lower than D. To investigate this we plot the amount of nodes smaller than 0.995 versus c, because we are interested in how many nodes do not get close to the all-1-configuration. In

Figure 10 we used the standard normal distribution to generate the values of $\sigma(x)$. What is immediately clear is that there is a positive correlation. From this barplot it seems like this is a linear correlation, to check that we repeated the experiment with a different range of c and fitted a line.

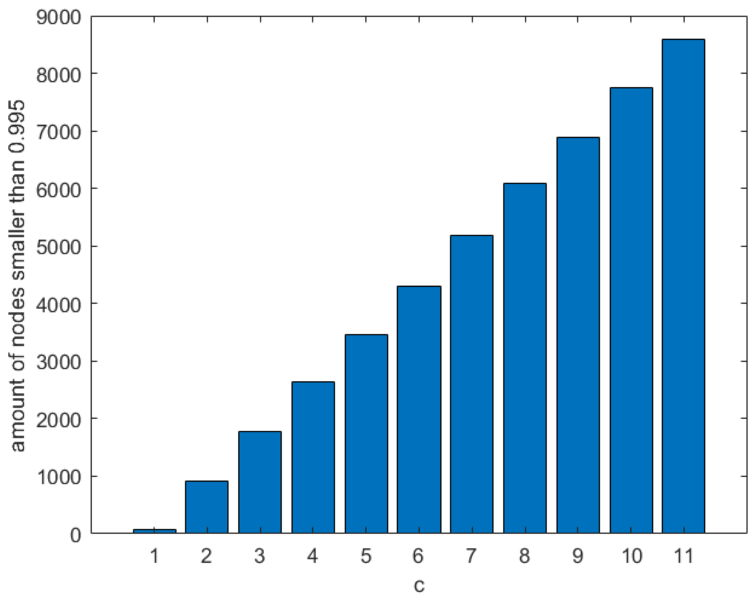

Figure 10: Plot of the amount of nodes smaller than 0.995 against c, with the standard normal distribution and 200,000 iterations.

In Figure 11 it looks very close to linear but in fact MATLAB fitted a line of a third order polynomial. What the reason is for this almost linear relation is unclear. Most likely at this point is that it is the consequence of the fact that by increasing c the total mass of the system decreases linearly.

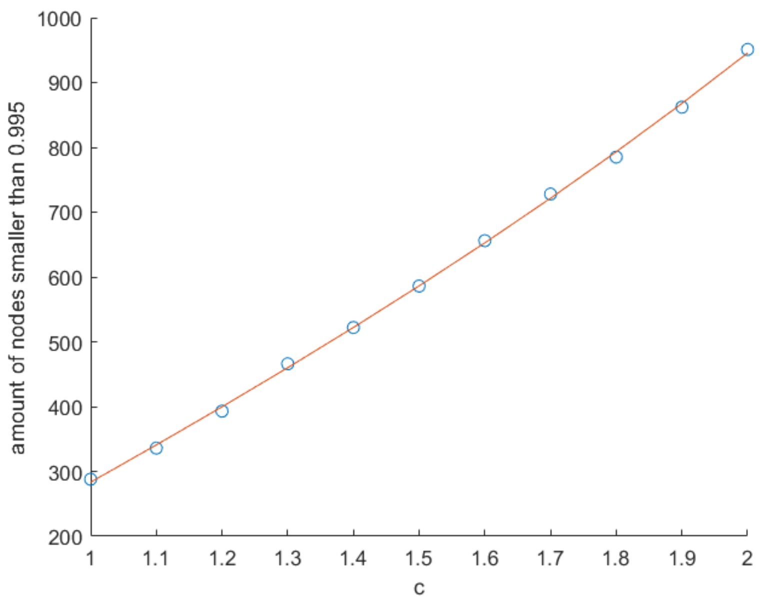

Figure 11: Plot of the amount of nodes smaller than 0.995 against c, with the standard normal distribution and 60,000 iterations.

If we look at the Pareto distribution instead of the Gaussian we get a very different picture. Figure 12 shows a very concave line. It is still a positive correlation, but the line has a completely different shape. A problem we encounter again is that if c gets too large then almost all nodes will lie in $(-\infty, 0.995)$. This might explain why in this case the line starts to flatten out as for $c = 2$ more than 80% of the nodes is smaller than 0.995. However it has not yet flattened out as much as compared to the next one in Figure 13.

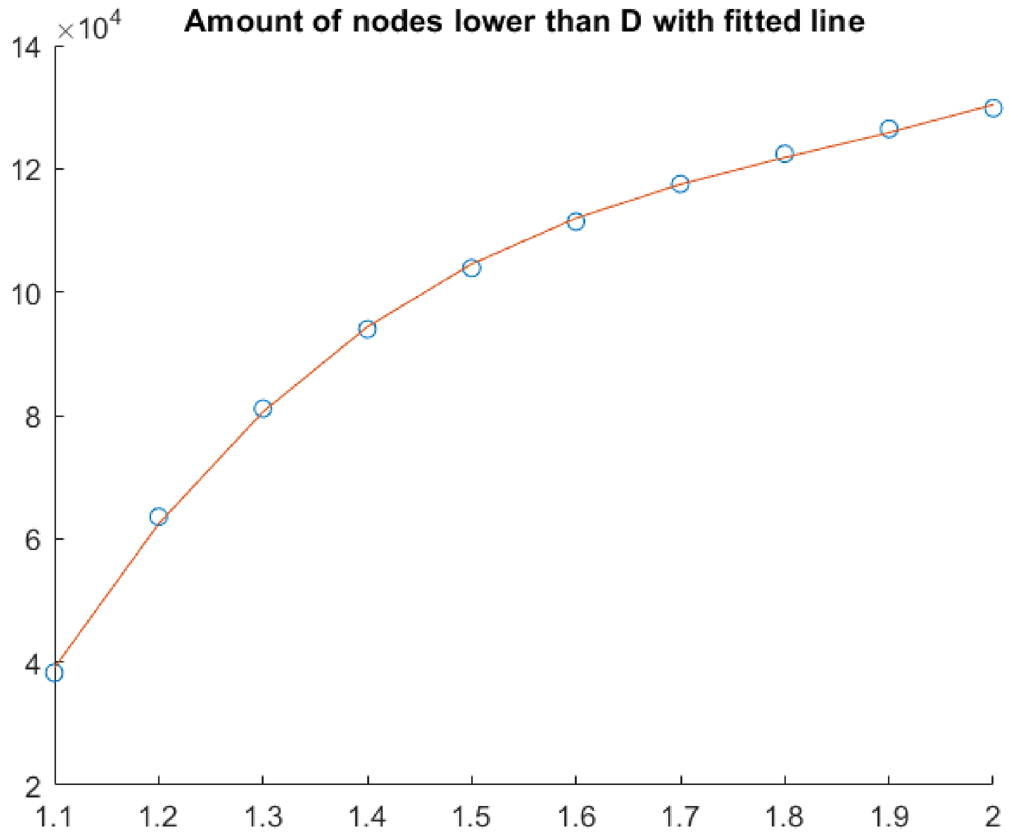

Figure 12: Plot of the amount of nodes smaller than 0.995 against c, with the Pareto distribution. With $\alpha = \frac{3}{2}$ and $x_m = 1$ and 60,000 iterations.

When using the Poisson distribution we get an almost identical line, as can be seen in Figure 13. The noticeable difference is that the line flattens out much earlier at around 60% of nodes and it gets a bit flatter.

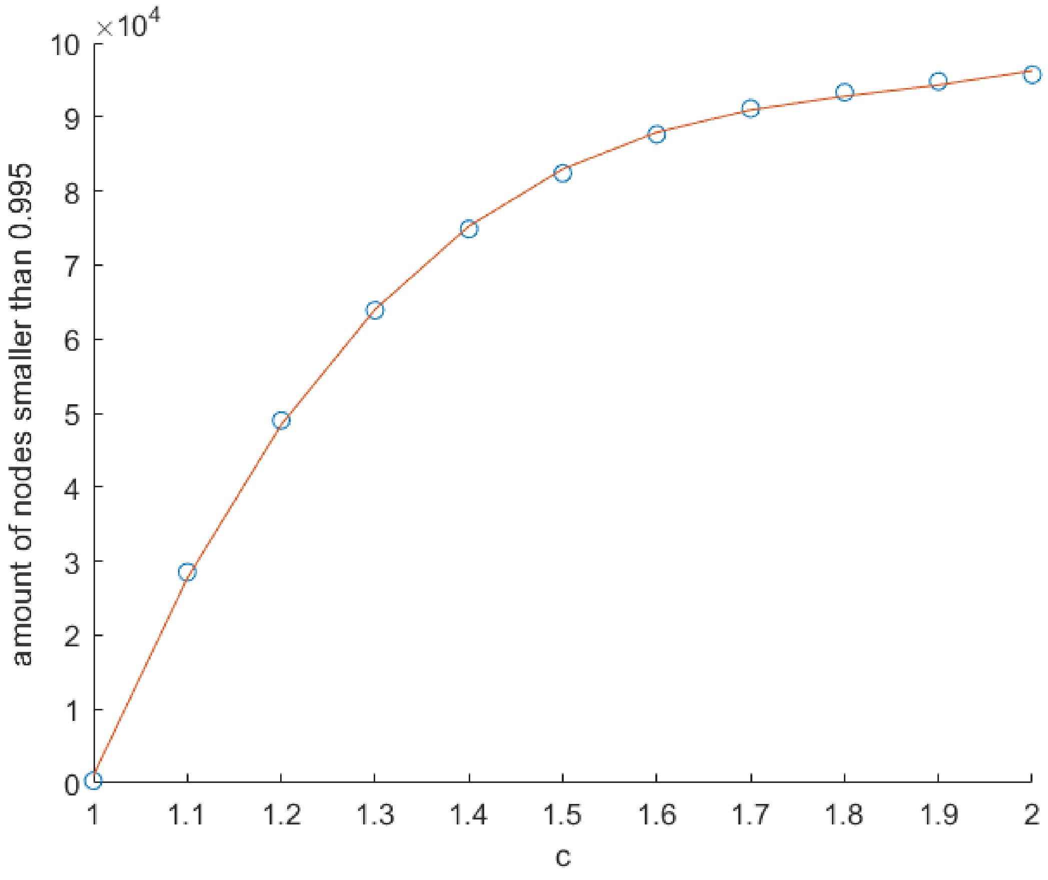

Figure 13: Plot of the amount of nodes smaller than 0.995 against c, with the Poisson distribution. With $\lambda = \frac{1}{2}$ and 60,000 iterations.

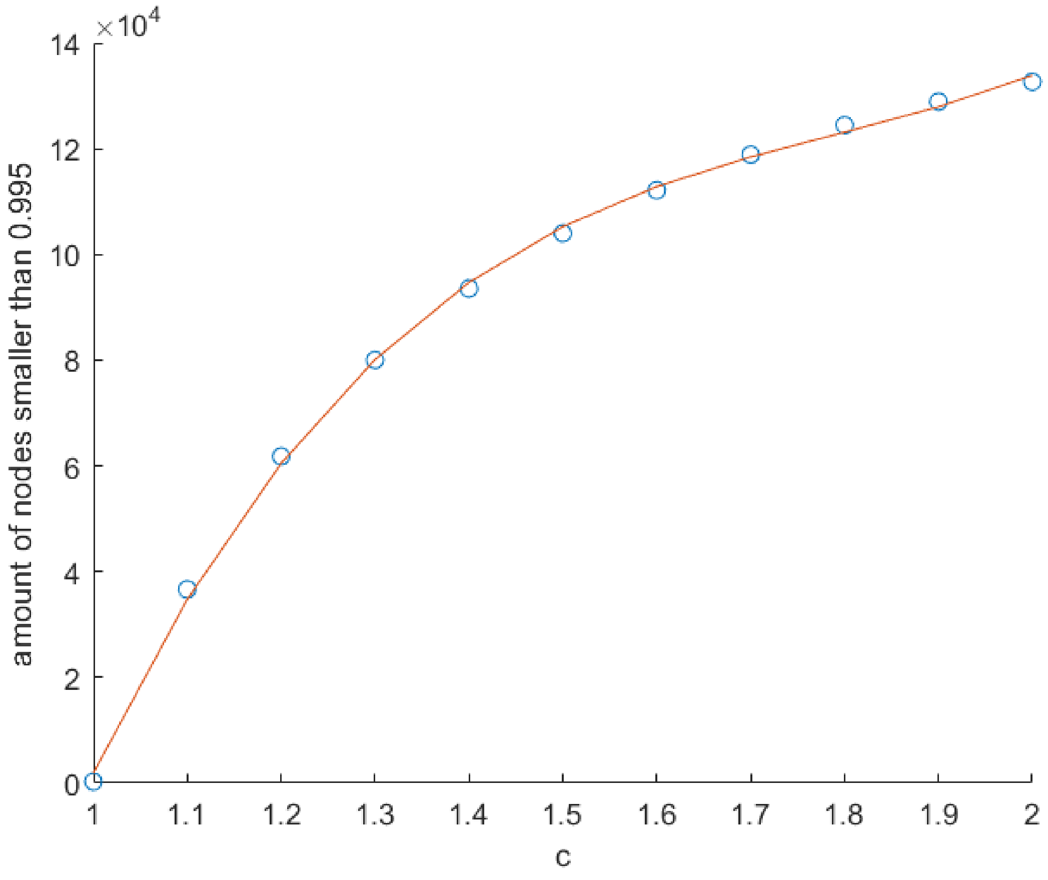

Figure 14: Plot of the amount of nodes smaller than 0.995 against c, with the Gamma distribution. With $\lambda = 1$, $\alpha = 1$ and 60,000 iterations.

In Figure 14 we again see a very similar line to the previous two. The line again increases with c, and flattens out when it approaches 80% of all nodes.
The reason for the shape of these lines most likely lies in the distribution of the initial sandpile. As we are deducting mass from every node by increasing c more and more nodes will start below 0.995 or even much lower than that. And since the toppling procedure is constructed in a way that it only allows mass to be added to nodes that are smaller than 1, the total amount of nodes below 0.995 can only decrease every timestep and can never increase. A consequence is that there can only be a positive correlation between c and the amount of nodes smaller than 0.995, as for higher c we start with more lower valued nodes. The way the DSM is formulated as in Equation 7 the mean of $s(x)$ is 1 if $c = 1$. So most nodes are distributed around one, and the distribution gets sparser above that. This means that if we are subtracting mass from the system this plays a role. At first we are moving a lot of nodes below the threshold of 0.995 by increasing c slightly, the more we increase c less nodes pass that threshold. This reasoning should explain the shape of the correlation we found.

6.2 Odometer

As we already saw in Figure 7 the odometer is also influenced by the introduction of c. As there is less mass in the system there is also less mass to be spread around, therefore the odometer is lower over the whole graph. Since we saw in the previous sections that a lot of nodes remain smaller than 1, we know that the odometer is zero in all those nodes. Before the introduction of c the odometer would be zero almost nowhere but now it will be zero in a lot of places.

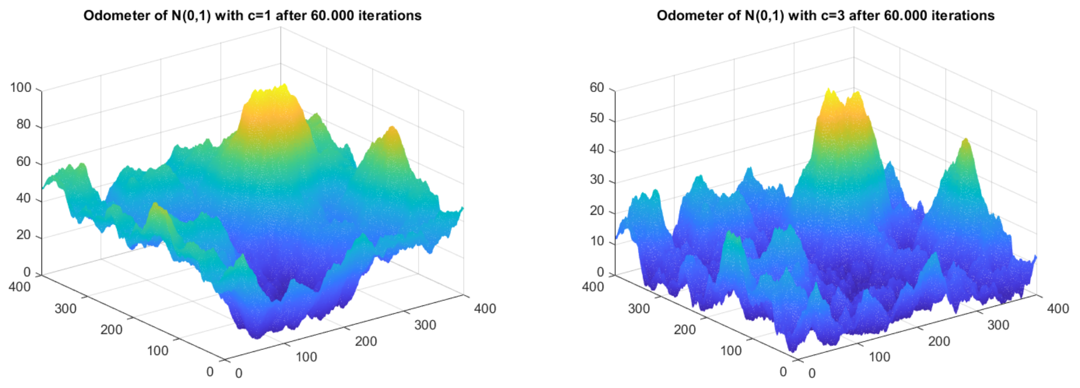

Figure 15: Influence of c on the odometer after 60,000 timesteps. Using N(0,1) and $c = 1$ and $c = 3$.

As can be seen in Figure 15 the odometers look fairly similar, they both have the biggest peak in the middle and a few smaller peaks around it also in the same place. These two look very similar because for the standard normal distribution $\sum_{y\in V}\sigma(y)$ is very small. In the instance these two odometers were taken from this sum was only 514, which means that for $c = 3$ only 0.0064 was subtracted from every node.
If we compare this to the Poisson distribution we get something a lot different. On the left side in Figure 16 is a graph of the odometer with $c = 1$, this looks very similar when compared to the odometers of Figure 15 we have seen before. Which is what we expected as it is one of the results of Cipriani, Hazra, and Ruszel [2] that the odometers will have the same behaviour. On the right side of Figure 16 is the odometer with $c = 5$, since for this distribution only 1.4% of nodes are in D after 500,000 iterations a lot of the odometer is still zero. These nodes have never toppled and thus have never distributed any of their mass to their neighbours.

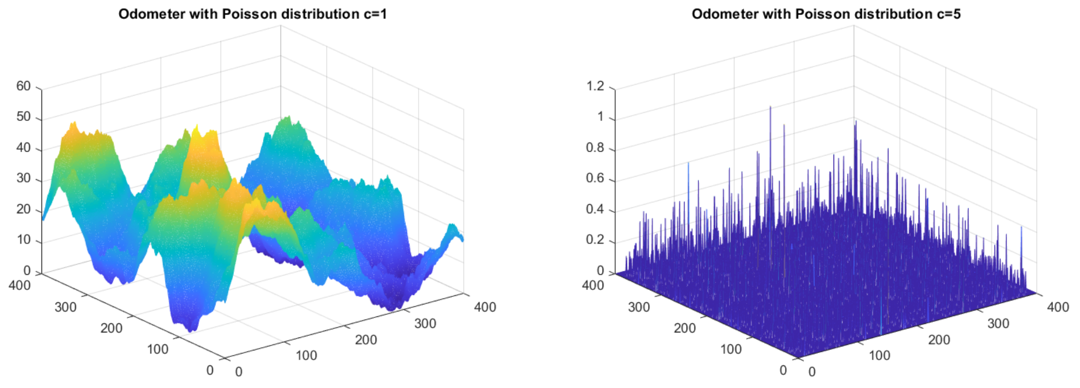

Figure 16: Influence of c on the odometer after 60,000 timesteps. Using the Poisson distribution with $\lambda = \frac{1}{2}$ and $c = 1$ and $c = 5$.

We also looked at the influence of c on the mean and variance of the odometer. As can be expected after seeing these previous figures, both the mean and variance decreased by a lot when c is increased. This happened for all distributions we tested and all values of c. This is of course because of all the nodes that start below 1 and never emit any mass, this means that the odometer is 0 at a lot of nodes. Therefore the mean is a lot lower and since almost all values of the odometer are the same the variance is also a lot lower.

6.3 Clusters

When the code for this thesis was first written it was tested out while using a very low amount of iterations and with $c = 1$. What stood out every time is that the last few nodes that were not close to 1 yet were all very close together as though they formed a cluster. As can be seen in Figure 17, for $c = 1$ there are three clusters identified by using kmeans in MATLAB. For $c = 3$ and $c = 5$ that was no longer the case, as there are many more nodes below 1. There are too many to effectively put them into clusters, as they are now much more randomly spread around.

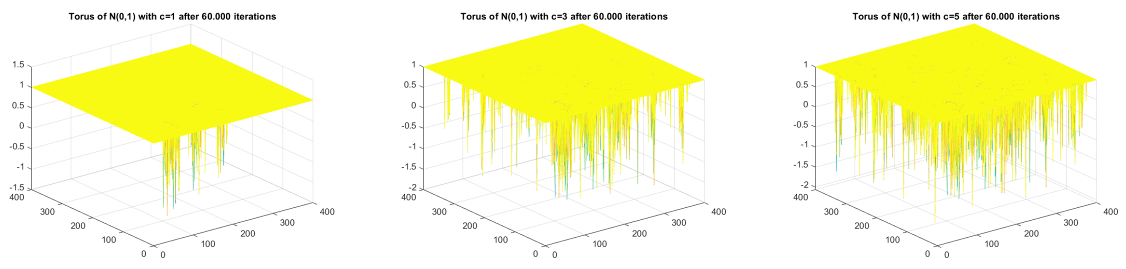

Figure 17: Influence of c on the converged sandpile and clusters after 60,000 timesteps. Using N(0,1) and $c = 1$, $c = 3$, and $c = 5$.

Although this was studied only very briefly and not a lot of results came out of it, it is worth mentioning. Because it is clearly a property of the DSM that when it is close to convergence the only nodes not yet at 1 lie closely together. This was found for all probability distributions used for $\sigma(x)$ with $c = 1$.

Because of the results of the previous section the amount of nodes not in D differs greatly between the different distributions used when increasing c. Therefore the clusters disappeared much quicker for the Pareto distribution for example. Increasing c in this case led to so many nodes smaller than 1 that the clusters were no longer existent.

7 Conclusion

In this thesis we have used a simulation study to look at the influence of introducing a constant c to the DSM model of Equation 7, which becomes

$$s(x) = 1 + \sigma(x) - \frac{c}{|V|} \sum_{y \in V} \sigma(y). \tag{31}$$

Yes c influences the speed and degree of convergence of the DSM as can be seen in the different tables in subsection 6.1, but this is highly dependant on the initial configuration. There are multiple problems with using c as suggested which lead to limitations in its usability. The fact that we multiply c by $\frac{1}{|V|} \sum_{y \in V} \sigma(y)$ means that if this sum is large then c cannot be too large. If the product of these two factors is too high then too much volume is removed from the system, this leads to very few nodes actually reaching the interval of convergence D. Sometimes there is not enough mass left to get every node to the lower bound of the interval. This means we cannot use the same values for c for every distribution. Also since there is not enough mass to let the sandpile converge to the all-1-configuration the sandpile does not converge to a state we can predict. The sandpile does stabilize eventually with all nodes at or below 1, after which no toppling will happen anymore. However this is still not likely to happen in finite time because the situation of Lemma 1 still occurs.

If instead the sum $\sum_{y \in V} \sigma(y)$ is negative then increasing c doesn't decrease the volume of the system, but instead increases it. This means that it will never stabilize as there will be too much mass in the system, there will always be nodes with a mass of higher than 1.

In subsubsection 6.1.1 a positive correlation between c and the amount of nodes lower than the convergence interval D was found. The shape of this correlation was almost linear for the standard normal distribution, but not for the other distributions tested. That was most likely due to the fact that $\frac{1}{|V|} \sum_{y \in V} \sigma(y)$ was several orders of magnitude smaller for the standard normal distribution.
For the other distributions that were tested the correlation had the same shape, it is a concave line which flattens out for higher c. It flattens out because for high values of c it goes to 100% of nodes smaller than 0.995.

The introduced constant c also influences the odometer of the sandpile. Since the total mass in the system is reduced the odometer as a whole is a lot lower. Moreover, since a lot of the nodes remain smaller than 1 in the stable state the odometer is zero at those nodes. This means that for a high value of $c \cdot \frac{1}{|V|} \sum_{y \in V} \sigma(y)$ the whole shape of the odometer changes, and the mean and variance of the odometer decrease drastically.

To conclude I would say that it is not very straightforward to use c as was suggested. It would be better to look at the total of $c \cdot \frac{1}{|V|} \sum_{y \in V} \sigma(y)$. Since the total mass being removed from the system is $(c-1) \sum_{y \in V} \sigma(y)$ the ideal size of c depends on $\sum_{y \in V} \sigma(y)$.

8 Discussion

In hindsight there are a lot of things that could have been done differently. The reached conclusion that the introduced constant c does help reducing the time to reach a stable state is useful. However, c should be used carefully as there could be unwanted side effects. If c is something that can help, then the next question that arises is what the optimal value of c is to reach a stable state quickly and reliably. Increasing c too much leads to a stable state from the start but that defeats the point of a sandpile.

The idea of the introduction of c was that the sandpile would become subcritical, to ensure subcriticality the sum $\sum_{y \in V} \sigma(y)$ is of great importance. In this thesis this was ensured by first determining the sign of $\sum_{y \in V} \sigma(y)$ and only using that sandpile if the sign was positive. Then a positive c could be used. The values of c were chosen by trial and error. It stood out that for certain distributions a value between 1 and 2 worked best, for some even smaller would be preferable. And for the standard normal c could be as large as 20. Again, this all depended on the size of $\sum_{y \in V} \sigma(y)$.
Therefore the ideal value of c most likely depends on $\sum_{y \in V} \sigma(y)$, that is something further research might be able to prove.

One of the other things that could have been done differently was the way of determining convergence or stability. The interval $D = (0.995, 1.005)$ was used. This interval is very useful in determining if the sandpile gets close to the all-1-configuration. However since the sandpiles we are looking at are subcritical it was already known that these sandpiles would never reach the all-1-configuration. Therefore it might be better to look at a different interval or a different method. A different interval could be similar to $(-\infty, 1 + \epsilon)$ with $\epsilon > 0$. Otherwise one could look at the odometer, if the odometer does not change in a timestep then the sandpile has stabilized. A different method that could be investigated for example is to look at the sum of the mass that is larger than 1. These last two methods were not implemented in this thesis because they do not give an indication of how well the sandpile has approached the all-1-configuration and because we were severely limited by the calculation speed of a single laptop.

It is therefore also advisable if anyone wants to repeat or continue this simulation study to use more or better computers. The time it took to run the simulations was so long that it didn't allow for the amount of iterations we wanted. For example the sandpile of Table 4 with $c = 1$ should have been calculated for far more than 500,000 iterations. This sandpile took so long because it was heavily influenced by outliers.
Therefore it could also be interesting to specifically study the influence of large outliers on the stabilization speed of the DSM, and whether c could help to make this go faster.

A last thing to recommend for future research is to look at the theorems of section 3. It was shown in subsection 4.3 that these proofs no longer work, but it might be possible to reformulate the theorems and lemmas to make them work with c.

A Matlab code of the movie

```matlab
% third draft of divisible sandpile model
%
clear all
close all
format long
rng(3);                                %set randomness
m = 400;                                 %number of columns
n = 400;                                 %number of rows
c =1;                                  %constant determining if the total will be smaller or larger than n*m in the initial matrix
iterations = 10;                     %amount of iterations

  mu = 0;
  sigma = 1;
  S = mu*ones(n,m)+sigma*randn(n,m);     %normal N(mu,sigma^2)

%lambda = 1/2;                         %parameter for poisson
%S = poissrnd(lambda,n,m);            %matrix with poisson

%  lambda = 1;                           %parameter for gamma
%  alpha = 5;                            %second parameter for gamma, alpha =1 is the exponential distr
%  S = gamrnd(alpha,lambda,n,m);         %matrix with gamma

%   mu = 0;                                 % statistical median
%   lambda = 1/2;                           % half width at the half maximum density level
%   S = mu+lambda*tan(pi*(rand(n,m)-1/2));   % Cauchy distribution

total = sum(sum(S));                    %sum of all sigma's
T = ones(n,m) + S - c/(n*m)*total;      %Initial matrix
%sum(sum(T))
x = 4;                                  %amount of neighbours

u = zeros(n,m);                         %odometer
N=10;                                   %amount of frames in movie, please make sure N devides iterations
M=[0:iterations/N:iterations];          %take frame if on one of these iterations
O = moviein(N+1);                       %movie of matrix with N+1 frames
P = moviein(N+1);                       %movie of odometer
R = T;                                  %initial matrix for later plotting
```

```
 a = zeros(1,iterations);                   %amount of values inside stabilisation interval at every timestep
 b = zeros(1,iterations);                   %percentage of values inside stabilisation interval at every timestep
 d = zeros(2,iterations);                   %total sum of values at every timestep
 e = zeros(1,iterations);                   %amount of values < 1-A/2 at any time step
 A = 1*10^(-2);                             %length of stabilisation interval

%
figure
mesh(T)
title('Initial matrix')

O(:,1) = getframe;

figure
mesh(zeros(n,m))
title('Initial odometer')

P(:,1) = getframe;

for k=1:iterations
    temp=zeros(n,m);                    %temporary matrix to log the changes due to toppling
    for i=1:n
        for j=1:m
            if T(i,j)>1
                t = T(i,j)-1;           %the total that will be removed from (i,j)
                v = t/x;                %the amount that will go to each neighbour

                i_t = i-1;              %top neighbour
                if i_t == 0             %check if i is on top border
                    i_t = n;
                end

                i_b = i+1;              %bottom neighbour
                if i_b == n+1           %check if i is on bottom border
                    i_b = 1;
                end

                j_l = j-1;              %left neighbour
                if j_l == 0             %check if j is on the left border
```

```
                    j_l = m;
                end

                j_r = j+1;          %right neighbour
                if j_r == m+1       %check if j is on the right border
                    j_r = 1;
                end

                temp(i_t,j)=temp(i_t,j)+v;      %every neighbour gets v added
                temp(i_b,j)=temp(i_b,j)+v;
                temp(i,j_l)=temp(i,j_l)+v;
                temp(i,j_r)=temp(i,j_r)+v;
                temp(i,j)=temp(i,j)-t;          %value on (i,j) goes back to 1
                u(i,j)=u(i,j)+v;                %odometer increases every iteration
            end
        end
    end
    T=T+temp;
    if ismember(k,M)
        subplot(2,1,1);
        mesh(T);
        title(strcat({'Matrix after '},num2str(k),' iterations'))
        O(:,k*N/iterations+1) = getframe;
        pause(2)

        subplot(2,1,2);
        mesh(u);
        title(strcat({'Odometer after '},num2str(k),' iterations'))
        P(:,k*N/iterations+1) = getframe;
        pause(2)
    end
     a(1,k) = sum(sum(1-A/2<T & T<1+A/2));
     b(1,k) = a(1,k)/(n*m);
     d(1,k) = sum(sum(T));
     d(2,k) = sum(sum(temp));
     e(1,k) = sum(sum(T<1-A/2));
end

figure
mesh(T)
title('Final matrix')

figure
mesh(R)
```

```
title('Initial matrix')

figure
movie(O)

strcat({'stabilised after '}, num2str(find(b>=0.95,1,'first')),'
   iterations')     %gives the amount of iterations need to reach
   95% of nodes in stabilisation interval
```

B Matlab code of the analysis of C

```matlab
% third draft of checking influence of C on convergence of dsm
% adds calculation of slope and of max value of torus at every timestep
clear all
format long
r = 1;
rng(r);                                       %set randomness
m = 400;                                        %number of columns
n = 400;                                        %number of rows
V = n*m;
deltac = 1;
c = 1:deltac:5 ;                                     %constant determining if the total will be smaller or larger than n*m in the initial matrix
C=length(c);                                   %amount of different values of c to test
iterations = 60000;                            %amount of iterations

  mu = 0;
  sigma = 1;
  S = mu*ones(n,m)+sigma*randn(n,m);            %normal ~N(mu,sigma^2)
%
% lambda = 1/2;                                %parameter for poisson
% S = poissrnd(lambda,n,m);                   %matrix with poisson of size (n,m)

%  lambda = 1;                                                %scale parameter for gamma
%  alpha = 1;                                                 %shape parameter for gamma, alpha=1 is the exponential distr
%  S = gamrnd(alpha,lambda,n,m);               %matrix with gamma of size (n,m)

% mu = 0;                                        % statistical median
% lambda = 1;                                  % half width at the half maximum density level
% S = mu+lambda*tan(pi*(rand(n,m)-1/2));   % Cauchy distribution

% alpha = 1.5;                                                    % take alpha in (1,2) E(X)=inf for alpha<=1, Var(X)<inf for alpha>=2
% Xm = 1;                                                       % take Xm>0
% S = Xm*((ones(n,m)-rand(n,m)).^(-1/alpha));                % Pareto

total = sum(sum(S));                           %sum of all sigma's
if total<0
    return
```

```matlab
end
x = 4;                                  %amount of neighbours

u = zeros(n,m);                         %odometer
a = zeros(C,iterations);                %amount of values inside stabilisation interval at every timestep
b = zeros(C,iterations);                %percentage of values inside stabilisation interval at every timestep
%  d = zeros(2,iterations);                %total sum of values at every timestep
e = zeros(C,iterations);                %amount of values < 1-A/2 at any time step
A = 1*10^(-2);                          %length of stabilisation interval
f = 1:C;                                %amount of iterations needed to converge
g = 1:C;
L = 1:C;
z = 0.95;                               %percentage that needs to be in stabilisation interval to define converged
G = zeros(1,C-1);                       %used for slope
h = zeros(C,iterations);                %used for max value on torus
uparam = zeros(2,C);                    %mean and variance of u for every c
M = cell(C,1);                          %store odometer for every c
O = cell(C,1);                          %store final sandpile for every c


for p=1:C
    u = zeros(n,m);
    T = ones(n,m) + S - c(p)/V*total;      %Initial matrix
    for k=1:iterations
        temp=zeros(n,m);                %temporary matrix to log the changes due to toppling
        for i=1:n
            for j=1:m
                if T(i,j)>1
                    t = T(i,j)-1;       %the total that will be removed from (i,j)
                    v = t/x;            %the amount that will go to each neighbour

                    i_t = i-1;          %top neighbour
                    if i_t == 0         %check if i is on top border
                        i_t = n;
                    end

                    i_b = i+1;          %bottom neighbour
                    if i_b == n+1       %check if i is on bottom border
                        i_b = 1;
                    end

                    j_l = j-1;          %left neighbour
                    if j_l == 0         %check if j is on the left border
                        j_l = m;
                    end

                    j_r = j+1;          %right neighbour
                    if j_r == m+1       %check if j is on the right border
                        j_r = 1;
                    end

                    temp(i_t,j)=temp(i_t,j)+v;      %every neighbour gets v added
                    temp(i_b,j)=temp(i_b,j)+v;
                    temp(i,j_l)=temp(i,j_l)+v;
                    temp(i,j_r)=temp(i,j_r)+v;
                    temp(i,j)=temp(i,j)-t;          %value on (i,j) goes back to 1
                    u(i,j)=u(i,j)+v;                %odometer increases every iteration
                end
            end
        end
        T=T+temp;
        a(p,k) = sum(sum(1-A/2<T & T<1+A/2));
        b(p,k) = a(p,k)/V;
        e(p,k) = sum(sum(T<1-A/2));
        h(p,k) = max(max(T));
    end
    uparam(1,p) = sum(sum(u))/V;
    uparam(2,p) = sum(sum((u-uparam(1,p)).^2))/(V-1);
    strcat('for c=',num2str(c(p)),{' stabilised after '}, num2str(find(b(p,:)>=z,1,'first')),' iterations')
    if isempty(find(b(p,:)>=z,1,'first'))
        f(p)=0;
    else
        f(p)=find(b(p,:)>=z,1,'first');
    end
        if isempty(find((a(p,:)+e(p,:))/V>=z,1,'first'))
        g(p)=0;
    else
        g(p)=find((a(p,:)+e(p,:))/V>=z,1,'first');
    end
    if isempty(find((a(p,:)+e(p,:))/V==1,1,'first'))
        L(p)=0;
    else
        L(p)=find((a(p,:)+e(p,:))/V==1,1,'first');
    end
    M{p} = u;
    O{p} = T;
end


%strcat('for c=',num2str(c(1)),{'stabilised after '}, num2str(find(b>=0.95,1,'first')),' iterations')     %gives the amount of iterations need to reach 95% of nodes in stabilisation interval

figure
bar(c,(V-a(:,iterations)))
xlabel('c')
ylabel(['amount of nodes smaller than ' num2str(1-A/2)])


for g = 1:C-1
    G(g)=(a(g,iterations)-a(g+1,iterations))/deltac;
end

for q = 2:C
    d(q-1) = c(q);
    D(q-1) = a(q,iterations);
end
p = polyfit(c.',(e(:,iterations)),3);
p1 = polyval(p,c);


figure
scatter(c,(e(:,iterations)))
hold on
plot(c,p1)
xlabel('c')
ylabel(['amount of nodes smaller than ' num2str(1-A/2)])

% beta0 = (sum(c.^2)*sum(V-a(:,iterations))-sum(c)*sum(c.'.*(V-a(:,iterations))))/(C*sum(c.^2)-(sum(c))^2);   %intercept calculated with formula from p 544 book by Rice
% beta1 = (C*sum(c.'.*(V-a(:,iterations)))-sum(c)*sum(V-a(:,iterations)))/(C*sum(c.^2)-(sum(c))^2);           %slope calculated with formula from p 544 book by Rice
% yfit = beta0 + c*beta1;
%
% figure
% scatter(c,(V-a(:,iterations)))
% hold on
% plot(c,yfit)
% title('Linear line fitted to result')
% xlabel('c')
% ylabel(['amount of nodes smaller than ' num2str(1-A/2)])

% r = V-a(:,iterations) - beta0 -beta1.*c.';                    %
  residuals
% RSS = sum(r.^2);                                              %
  residual sum of squares
%
% figure
% plot(c,r,'*')
% title('residuals from fitted line')
% xlabel('c')
% ylabel('residual from fitted line')
```

C Matlab code of the convergence speed tests

```matlab
 % first draft of code to check convergence of odometer with changing c
% copied from C_draft3
clear all
r = 1;
rng(r);                                         %set randomness
m = 400;                                          %number of columns
n = 400;                                          %number of rows
V = n*m;
% c = 2;
deltac = 1;
c = 1:deltac:5;                                       %constant determining if the total will be smaller or larger than n*m in the initial matrix
C = length(c);                                       %amount of different values of c to test
iterations = 60000;                                %amount of iterations

mu = 0;
sigma = 1;
S = mu*ones(n,m)+sigma*randn(n,m);     %normal ~N(mu,sigma^2)

% lambda = 1/2;                                  %parameter for poisson
% S = poissrnd(lambda,n,m);                     %matrix with poisson

%   lambda = 1;                                    %parameter for gamma
%   alpha = 5;                                     %second parameter for gamma, alpha =1 is the exponential distr
%   S = gamrnd(alpha,lambda,n,m);                  %matrix with gamma

%    mu = 0;                                          % statistical median
%    lambda = 1/2;                                    % half width at the half maximum density level
%    S = mu+lambda*tan(pi*(rand(n,m)-1/2));   % Cauchy distribution

% alpha = 1.5;                                                   % take alpha in (1,2) E(X)=inf for alpha<=1, Var(X)<inf for alpha>=2
% Xm = 1;                                                    % take Xm>0
% S = Xm*((ones(n,m)-rand(n,m)).^(-1/alpha));                  % Pareto


total = sum(sum(S));                               %sum of all sigma's
if total<0
    return
end
```

```
x = 4;                                  %amount of neighbours

u = zeros(n,m);                         %odometer
a = zeros(C,iterations);                %amount of values inside
   stabilisation interval at every timestep
b = zeros(C,iterations);                %percentage of values
   inside stabilisation interval at every timestep
%  d = zeros(2,iterations);                %total sum of values
   at every timestep
e = zeros(C,iterations);                %amount of values < 1-A/2
    at any time step
f = zeros(1,C);                         %timestep of 95% in D
g = zeros(1,C);                         %timestep of 95% in(-inf
   ,1.005)
L = zeros(1,C);                         %timestep of all in(-inf
   ,1.005)
A = 1*10^(-2);                          %length of stabilisation
   interval
z = 0.95;                               %percentage that needs to
    be in stabilisation interval to define converged
h = zeros(C,iterations);                %used for max value on
   torus
uparam = zeros(2,C);
M = cell(C,1);
O = cell(C,1);


for p=1:C
    u = zeros(n,m);
    T = ones(n,m) + S - (c(p)/V)*total;      %Initial matrix
    for k=1:iterations
        temp=zeros(n,m);                %temporary matrix to log
           the changes due to toppling
        for i=1:n
         for j=1:m
            if T(i,j)>1
               t = T(i,j)-1;        %the total that will be
                  removed from (i,j)
               v = t/x;             %the amount that will go to
                  each neighbour

               i_t = i-1;           %top neighbour
               if i_t == 0          %check if i is on top border
                       i_t = n;
               end

               i_b = i+1;           %bottom neighbour
               if i_b == n+1        %check if i is on bottom
                  border
```

```
                        i_b = 1;
                end

                j_l = j-1;          %left neighbour
                if j_l == 0         %check if j is on the left
                   border
                        j_l = m;
                end

                j_r = j+1;          %right neighbour
                if j_r == m+1       %check if j is on the right
                   border
                    j_r = 1;
                end

                temp(i_t,j)=temp(i_t,j)+v;      %every neighbour
                   gets v added
                temp(i_b,j)=temp(i_b,j)+v;
                temp(i,j_l)=temp(i,j_l)+v;
                temp(i,j_r)=temp(i,j_r)+v;
                temp(i,j)=temp(i,j)-t;          %value on (i,j)
                   goes back to 1
                u(i,j)=u(i,j)+v;                %odometer
                   increases every iteration
             end
         end
        end
        T=T+temp;
        a(p,k) = sum(sum(1-A/2<T & T<1+A/2));
        b(p,k) = a(p,k)/V;
        e(p,k) = sum(sum(T<1-A/2));
        h(p,k) = sum(sum(T>1+A/2));
    end
%max(p) = find(h(p,:)<=1+A/2,1,'first');

uparam(1,p) = sum(sum(u))/V;
uparam(2,p) = sum(sum((u-uparam(1,p)).^2))/(V-1);
strcat('for c=',num2str(p),{' stabilised after '}, num2str(find(b
   (p,:)>=z,1,'first')),' iterations')
    if isempty(find(b(p,:)>=z,1,'first'))
        f(p)=0;
    else
        f(p)=find(b(p,:)>=z,1,'first');
    end
    if isempty(find((a(p,:)+e(p,:))/V>=z,1,'first'))
        g(p)=0;
    else
        g(p)=find((a(p,:)+e(p,:))/V>=z,1,'first');
    end
    if isempty(find((a(p,:)+e(p,:))/V==1,1,'first'))
        L(p)=0;
    else
        L(p)=find((a(p,:)+e(p,:))/V==1,1,'first');
    end
    M{p} = u;
    O{p} = T;
end

%     strcat('for c=',num2str(c),{' stabilised after '}, num2str(
   find(b(1,:)>=z,1,'first')),' iterations')
%     if isempty(find(b(1,:)>=z,1,'first'))
%         f=0;
%     else
%         f=find(b(1,:)>=z,1,'first');
%     end

% max = find(h(1,:)<=1+A/2,1,'first')
%
% umean = sum(sum(u))/V
% uvar = sum(sum((u-umean).^2))/(V-1)

%strcat('for c=',num2str(c(1)),{'stabilised after '}, num2str(
   find(b>=0.95,1,'first')),' iterations')    %gives the amount of
    iterations need to reach 95% of nodes in stabilisation
   interval
```

D Matlab code of the 1 dimensional convergence speed test

```matlab
% third draft of checking influence of C on convergence of dsm
% adds calculation of slope and of max value of torus at every timestep
clear all
rng(1);                                       %set randomness
m = 1;                                        %number of columns
c = 1;
deltan=1000;
n = 1000:deltan:20000;
N = length(n);
iterations = 500000;                            %amount of iterations

mu = 0;
sigma = 1;


x = 2;                                         %amount of neighbours

a = zeros(N,iterations);                       %amount of values inside stabilisation interval at every timestep
b = zeros(N,iterations);                       %percentage of values inside stabilisation interval at every timestep
%  d = zeros(2,iterations);                       %total sum of values at every timestep
e = zeros(N,iterations);                       %amount of values < 1-A/2 at any time step
A = 1*10^(-2);                                 %length of stabilisation interval
z = 0.95;                                      %percentage that needs to be in stabilisation interval to define converged
h = zeros(N,iterations);                       %used for max value on torus
max = 1:N;
uparam = zeros(2,N);
uparam2 = zeros(2,N);


for p=1:N
    u = zeros(n(p),1);                         %odometer back to 0
    V = n(p);
    S = mu*ones(n(p),1)+sigma*randn(n(p),1);      %normal ~N(mu,sigma^2)
    total = sum(sum(S));                          %sum of all sigma's
    T = ones(n(p),1) + S - (c/V)*total;          %Initial matrix
    for k=1:iterations
```

```matlab
        temp=zeros(n(p),1);                 %temporary matrix to log the changes due to toppling
        for i=1:n(p)
             if T(i,1)>1
                t = T(i,1)-1;       %the total that will be removed from (i,j)
                v = t/x;            %the amount that will go to each neighbour

                i_t = i-1;          %top neighbour
                if i_t == 0         %check if i is on top border
                        i_t = n(p);
                end

                i_b = i+1;          %bottom neighbour
                if i_b == n(p)+1       %check if i is on bottom border
                        i_b = 1;
                end

                temp(i_t,1)=temp(i_t,1)+v;      %every neighbour gets v added
                temp(i_b,1)=temp(i_b,1)+v;
                temp(i,1)=temp(i,1)-t;          %value on (i,j) goes back to 1
                u(i,1)=u(i,1)+v;                %odometer increases every iteration
             end
        end
        T=T+temp;
        a(p,k) = sum(sum(1-A/2<T & T<1+A/2));
        b(p,k) = a(1,k)/V;
        e(p,k) = sum(sum(T<1-A/2));
        h(p,k) = sum(sum(T));
    end
% max(p) = find(h(p,:)<=1+A/2,1,'first');

uparam(1,p) = sum(sum(u))/V;
uparam(2,p) = sum(sum((u-uparam(1,p)).^2))/(V-1);
uparam2(1,p) = uparam(1,p)/n(p);
uparam2(2,p) = uparam(2,p)/(n(p)^2);
end

    strcat('for c=',num2str(c),{' stabilised after '}, num2str(find(b(1,:)>=z,1,'first')),' iterations')
    if isempty(find(b(1,:)>=z,1,'first'))
        f=0;
    else
        f=find(b(1,:)>=z,1,'first');
    end

% max = find(h(1,:)<=1+A/2,1,'first')
%
% umean = sum(sum(u))/V
% uvar = sum(sum((u-umean).^2))/(V-1)

%strcat('for c=',num2str(c(1)),{'stabilised after '}, num2str(
   find(b>=0.95,1,'first')),' iterations')    %gives the amount of
    iterations need to reach 95% of nodes in stabilisation
   interval
figure
plot(n,uparam2(1,:))
xlabel('n')
ylabel('mean of odometer over n')

figure
plot(n,uparam2(2,:))
xlabel('n')
ylabel('Variance of odometer over n^2')

uparam3=zeros(2,N);
for i=1:N
    uparam3(1,i)=uparam(1,i)/(n(i)^(3/2));
end
figure
plot(n,uparam3(1,:))
xlabel('n')
ylabel('mean of odometer over n^3/2')
title(['1D mean over n^3/2, ' num2str(iterations) ' iterations'])
```

References

[1] Per Bak, Chao Tang, and Kurt Wiesenfeld. “Self-organized criticality: An explanation of the 1/f noise”. In: *Phys. Rev. Lett.* 59 (4 July 1987), pp. 381–384. DOI: `10.1103/PhysRevLett.59.381`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.59.381`.

[2] Alessandra Cipriani, Rajat Subhra Hazra, and Wioletta M. Ruszel. “Scaling limit of the odometer in divisible sandpiles”. In: *Probability Theory and Related Fields* 172.3 (Dec. 2018), pp. 829–868. ISSN: 1432-2064. DOI: `10.1007/s00440-017-0821-x`. URL: `https://doi.org/10.1007/s00440-017-0821-x`.

[3] Alessandra Cipriani, Rajat Subhra Hazra, and Wioletta M. Ruszel. “The divisible sandpile with heavy-tailed variables”. In: *Elsevier* (2017).

[4] Deepak Dhar. “The Abelian sandpile and related models”. In: *Physica A: Statistical Mechanics and its Applications* 263.1 (1999), pp. 4–25. ISSN: 0378-4371. URL: `http://www.sciencedirect.com/science/article/pii/S0378437198004932`.

[5] Jan de Graaff. “Scaling limit of the odometer of correlated Gaussians on the torus $\mathbb{T}^d$.” In: (2018).

[6] Lionel Levine and Yuval Peres. “Laplacian Growth, Sandpiles and Scaling Limits”. In: *Bulletin of the American Mathematical Society* (2017).

[7] Lionel Levine and Yuval Peres. “Strong spherical asymptotics for rotor-router aggregation and the divisible sandpile”. In: *Potential Analysis 30* (2009).

[8] Lionel Levine et al. “The Divisible Sandpile at Critical Density”. In: *Annales Henri Poincare* (2015).

[9] T Mikosch. “Regular Variation Subexponentiality and Their Applications in Probability Theory”. In: (1999).

[10] John A. Rice. *Mathematical Statistics and Data Analysis, third edition.* Brooks/Cole, Cengage Learning, 2007. ISBN: 9780495118688.

[11] Jeffrey S. Rosenthal. *A first look at rigorous probability theory, second edition.* World Scientific Publishing, 2006. ISBN: 9789812703705.

[12] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining.* Boston: Pearson Addison Wesley, 2005. ISBN: 0321321367.

Het einde van alle tranen.