

Modèle du tas de sable abélien

Daoubih Yousef

2026

Figure 1 – Modèle du tas de sable abélien

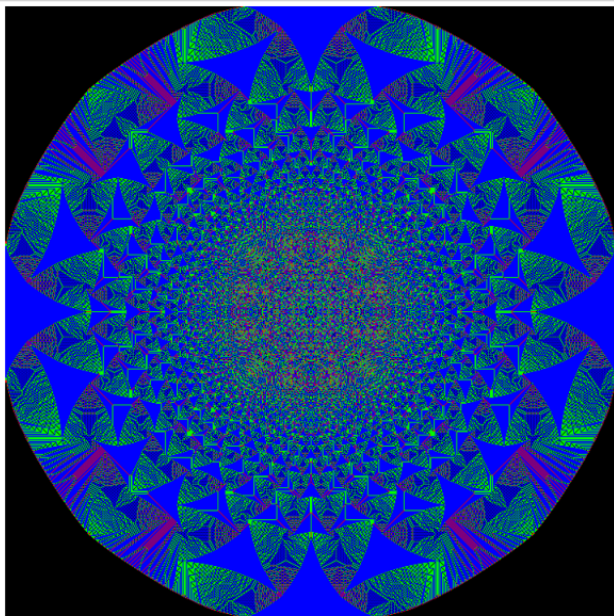


Table des matières

- ① Définition du modèle et problématique
- ② Étude de l'odomètre
- ③ Étude géométrique et empilement d'Apollonius

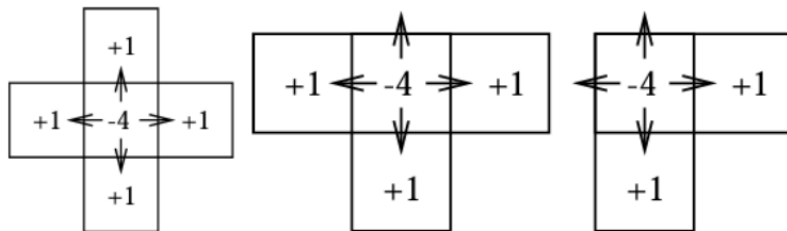
Définition du modèle (grille $n \times p$)

Règles du jeu

Soit une grille $n \times p$ de cellules contenant des grains de sable.

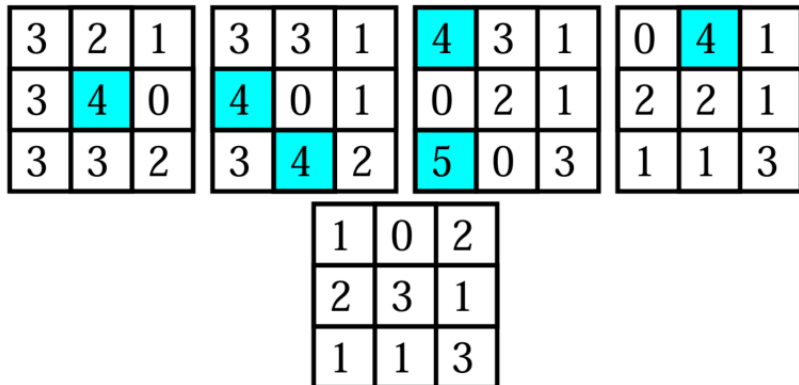
- Si une case a 4 grains ou plus, elle en perd 4.
- Elle en donne un à chaque voisin (haut, bas, gauche, droite).
- La case s'**éboule**.
- Si elle est en bord, les grains sortent du système.
- Donc, il y a **perte de grains**.

Figure 2 – Règles d'éboulement local



Exemple sur une grille de taille 3×3

Figure 3 – Exemple de stabilisation



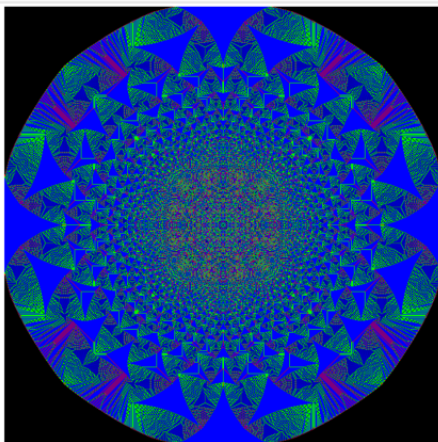
Théorème fondamental

Pour toute configuration initiale de grains sur une **grille finie**, il existe un **unique** état stable final, indépendamment des choix d'écroulements.

Stabilisation d'une grille 600×600 après ajout de 10^6 grains au centre

Le code couleur est : 0 en noir, 1 en bleu, 2 en vert et 3 en rouge.

Figure 4 – État final après stabilisation



Comment la dynamique locale d'un tas de sable abélien engendre-t-elle, à grande échelle, des structures géométriques fractales spontanées et stables ?

Table des matières

- ① Définition du modèle et problématique
- ② Étude de l'odomètre
- ③ Étude géométrique et empilement d'Apollonius

Odomètre

Dans le modèle du tas de sable abélien, l'**odomètre** est la fonction

$$u(x) = \text{nombre total d'éboulements effectués au site } x$$

au cours du processus de stabilisation.

Relation fondamentale :

$$\eta^\circ = \eta + \Delta u,$$

où η° est la configuration stabilisée.

Laplacien discret :

$$(\Delta u)(x) = u(x + e_1) + u(x - e_1) + u(x + e_2) + u(x - e_2) - 4u(x),$$

avec $e_1 = (1, 0)$ et $e_2 = (0, 1)$.

Objectif : Loi d'échelle entre le rayon du tas R_N et le nombre de grains N .

- **Conservation :** Le nombre total de grains N est constant.
- **Densité bornée :** Sur chaque site stabilisé, $s(x, y) \in \{0, 1, 2, 3\}$.
- **Compacité :** Les sites vides ($s = 0$) ne dominent pas, imposant une épaisseur moyenne constante.

Déduction de la surface

L'aire du domaine couvert est donc directement proportionnelle à N :

$$N \propto \pi R_N^2$$

On en déduit que le rayon de couverture croît selon : $R_N \propto \sqrt{N}$

Vérification expérimentale de la loi d'échelle

Afin de valider cette hypothèse, nous étudions le rayon de couverture pour plusieurs valeurs de N .

Figure 5 – N en fonction de R_N

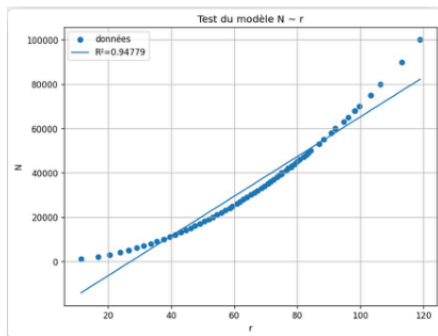
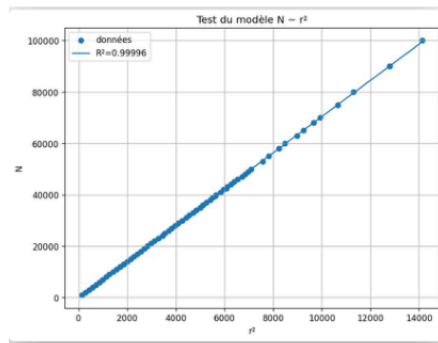


Figure 6 – N en fonction de R_N^2



Passage au continu : L'odomètre normalisé

Question :

Comment comparer le comportement du tas de sable lorsque $N \rightarrow +\infty$, alors que la taille du réseau et les valeurs de l'odomètre explosent ?

- **Normalisation de l'espace :**

$$x = \frac{X}{R_N}, \quad y = \frac{Y}{R_N},$$

- **Normalisation de l'odomètre :**

La relation de Dhar suggère que l'odomètre est naturellement d'ordre R_N^2 .

$$\eta^\circ = \eta + \Delta U$$

Définition de l'odomètre normalisé

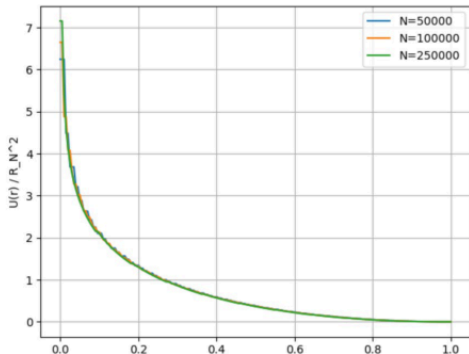
$$U(x, y) = \lim_{N \rightarrow +\infty} \frac{U_N(\lfloor R_N x \rfloor, \lfloor R_N y \rfloor)}{R_N^2}.$$

$$(x, y) \in [-1, 1]^2.$$

Convergence expérimentale selon l'axe horizontal

Afin de valider la pertinence de cette normalisation, on observe la coupe de l'odomètre pour plusieurs valeurs de N assez grandes.

Figure 7 – Profil horizontal de l'odomètre

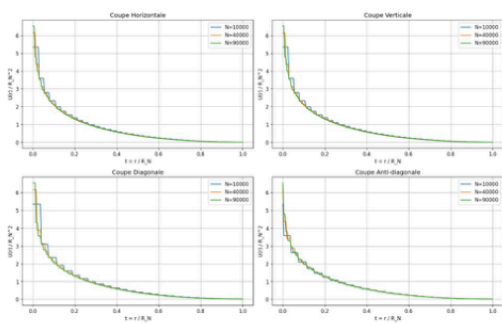


Interprétation

- **Superposition parfaite :** Plus N augmente, plus les courbes se superposent.
- **Question :** Mais que peut-on dire des autres directions ?

Convergence multidirectionnelle de l'odomètre

Figure 8 – Coupes directionnelles



Validation de la loi d'échelle

- **Stabilité directionnelle :**
Sur chaque axe
(horizontal, vertical,
diagonales), les profils se
superposent parfaitement
lorsque N augmente.

Régularité de l'odomètre : La fonction est-elle C^1 ?

Constat : Les profils de l'odomètre normalisé semblent parfaitement lisses. Peut-on affirmer que la fonction limite $u(x, y)$ est de classe C^1 sur tout le domaine ?

Approximation discrète de la dérivée partielle

Pour le vérifier numériquement, on étudie le taux d'accroissement sur le réseau discret. En posant $X = R_N x$ et $Y = R_N y$, on a : pour N assez grand

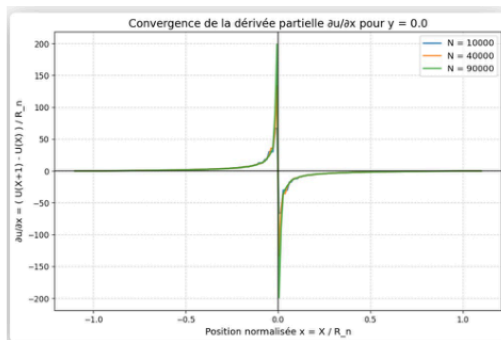
$$\frac{\partial u}{\partial x}(x, y) \approx \frac{\frac{U_N(X+1, Y)}{R_N^2} - \frac{U_N(X, Y)}{R_N^2}}{\frac{1}{R_N}}$$

Ce qui se simplifie en :

$$\frac{\partial u}{\partial x}(x, y) = \frac{U_N(X + 1, Y) - U_N(X, Y)}{R_N}$$

Analyse de la dérivée partielle : Coupe à $y = 0$

Figure 9 – Dérivée partielle pour $y = 0$

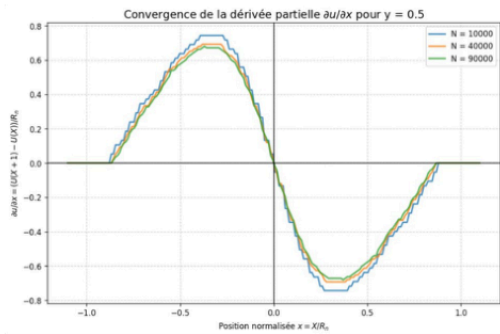


Discontinuité à l'origine

- Les courbes pour différents N se superposent, confirmant la convergence.
- Cependant :** On observe une divergence brutale (une singularité) en $x = 0$.
- L'injection continue de sable au centre crée un "pic" de pente infinie : la fonction **n'est pas C^1** à l'origine.

Analyse de la dérivée partielle : Coupe à $y = 0.5$

Figure 10 – Dérivée partielle pour $y = 0.5$

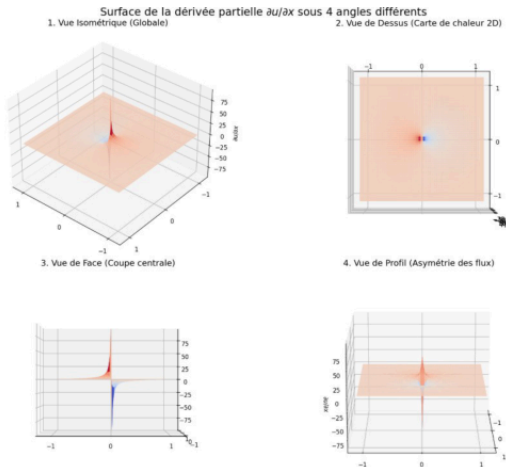


Régularité loin du centre

- En s'éloignant de la source (pour $y = 0.5$), la courbe de la dérivée devient parfaitement continue et lisse.
- Conclusion locale :** Loin de l'origine, l'odomètre se comporte bien comme une fonction C^1 .

Vue 3D de l'odomètre

Figure 11 – Visualisation 3D de la dérivée partielle



Synthèse géométrique

- La vue 3D confirme visuellement que la singularité est strictement confinée au centre du tas.
- **Symétrie** : l'étude de $\frac{\partial u}{\partial y}$ donne des résultats géométriquement identiques (rotation de 90°).
- **Bilan** : La fonction limite $u(x, y)$ est de classe C^1 sur le disque privé de l'origine.

Table des matières

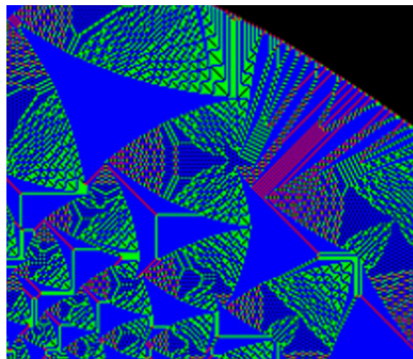
- ① Définition du modèle et problématique
- ② Étude de l'odomètre
- ③ Étude géométrique et empilement d'Apollonius

Observation locale : Les "Patches" de périodicité

Constat numérique :

- Émergence de régions triangulaires nettes.
- **Propriété clé** : Le Laplacien Δu est **constant** à l'intérieur de chaque triangle.

Figure 12 – Zoom sur un "patch" de Laplacien constant.



L'obstacle discret :

Étudier une fonction discrète à Laplacien constant sur un domaine borné est analytiquement très difficile.

La stratégie continue et le processus de Zoom :

- ➊ **Passage au continu** : La limite nous permet de manipuler une fonction continue u .
- ➋ **Dilatation spatiale (Blow-up)** : En effectuant un "zoom" infini ($\epsilon \rightarrow 0$) autour d'un point x_0 :

$$\tilde{u}(y) = \lim_{\epsilon \rightarrow 0} \frac{u(x_0 + \epsilon y)}{\epsilon^2}$$

- ➌ **Ouverture du domaine** : Les frontières du domaine initial s'éloignent indéfiniment. Le domaine d'étude devient infini, où le Laplacien $\Delta \tilde{u}$ reste constant.

L'argument par le théorème de Liouville :

D'après le théorème de Liouville généralisé, si une fonction $f \in \mathcal{C}^2(\mathbb{R}^2)$ vérifie $\Delta f = \text{Cte}$ sur $\mathbb{R}^2$ et $f(x) = \mathcal{O}(|x|^2)$, alors f est une **fonction polynomiale de degré inférieur ou égal à 2**.

Cependant, j'ai identifié deux obstacles mathématiques majeurs :

- **Le manque de régularité globale** : L'odomètre continu n'est pas globalement de classe $\mathcal{C}^2$. Il présente des "cassures" (discontinuités de la dérivée seconde) aux frontières des triangles.
- **La condition de domination** : Prouver rigoureusement, *a priori*, que la fonction issue du zoom reste dominée par $|y|^2$ sur $\mathbb{R}^2$ tout entier est très complexe.

Le résultat de l'article de Levine, Pegden, Smart (Annals of Math, 2017)

L'article démontre qu'à chaque région géométrique observée correspond une **fonction superharmonique entière** unique, notée g_C , appelée odomètre local , chaque fonction s'associe à une matrice superharmonique

.

Théorème principal (Levine, Pegden, Smart, 2017)

Theorem (Caractérisation des matrices superharmoniques entières)

Une matrice symétrique $A \in \mathbb{R}^{2 \times 2}$ est **superharmonique entière**

si et seulement si :

$$\text{trace}(A) \leq 2 \quad \text{ou} \quad \exists C \in \mathcal{B} \text{ tel que } A \leq A_C$$

où :

- $\mathcal{B}$ est l'**empilement apollonien de bande**
- A_C est la matrice associée au cercle $C \in \mathcal{B}$
- $A \leq A_C$ signifie que $A_C - A$ est semi-définie positive

Représentation géométrique : $S_2 \cong \mathbb{R}^3$

Décomposition de $A \in S_2$

Toute matrice $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$ se décompose en :

$$A = \underbrace{\frac{a+c}{2} I}_{\text{Isotrope}} + \underbrace{\begin{pmatrix} \frac{a-c}{2} & b \\ b & -\frac{a-c}{2} \end{pmatrix}}_{\text{Trace nulle}}$$

Paramétrisation (x, y, r)

On définit les coordonnées dans $\mathbb{R}^3$ par :

$$\begin{cases} r = \frac{a+c}{2} \\ x = \frac{a-c}{2} \\ y = b \end{cases}$$

Reconstruction

$$A = \begin{pmatrix} r+x & y \\ y & r-x \end{pmatrix}$$

Définition analytique

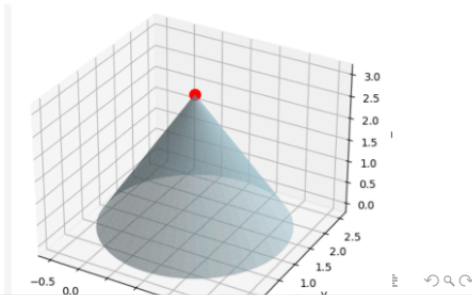
Soient $A = \Phi(x, y, r)$ et $B = \Phi(x', y', r')$. Alors

$$A \preceq B \iff r' - r \geq \sqrt{(x' - x)^2 + (y' - y)^2}.$$

Interprétation géométrique

L'ensemble des matrices A telles que $A \preceq B$ est exactement le cône de révolution de sommet B , orienté vers le bas selon l'axe r et de pente -1 .

Figure 13 – Cône de révolution (Ordre de Loewner)



Correspondance géométrique : Matrice $\rightarrow$ Cercle

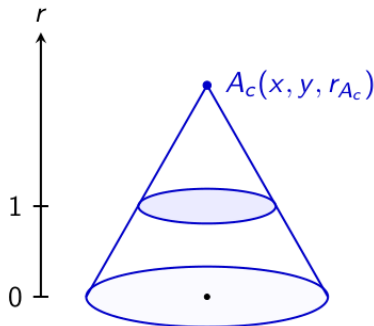
Bijection Géométrique

Soit une matrice symétrique $A \in S_2$ caractérisée par ses coordonnées (x, y, r_{A_c}) .

Si $\text{tr}(A) > 2$, le cercle associé C est défini par :

- **Centre** : (x, y) au niveau $r = 1$
- **Rayon** : $R = r_{A_c} - 1$

Figure 14 – Représentation spatiale de la bijection



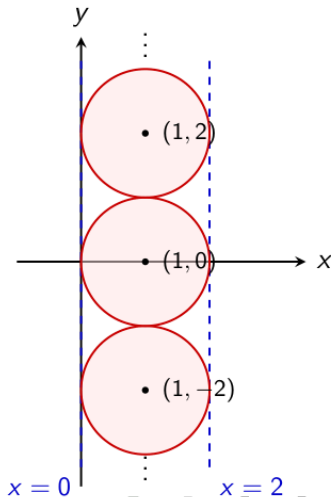
Empilement en bandes : définition

Configuration initiale $\mathcal{B}_0$

L'empilement en bande $\mathcal{B}$ est généré à partir d'un motif de base $\mathcal{B}_0$ constitué de :

- Deux droites verticales parallèles d'équations $x = 0$ et $x = 2$ (assimilées à des cercles de courbure nulle).
- Une infinité de cercles de rayon $R = 1$, mutuellement tangents.
- Leurs centres se situent en $(1, 2m)$ pour tout entier $m \in \mathbb{Z}$.

Figure 15 – Génération de l'empilement



Tracé de l'empilement d'Apollonius en bande

Principe de récursion :

Générer l'empilement via les coordonnées complexes de courbure $C = (k, kz)$.

Conditions d'application :

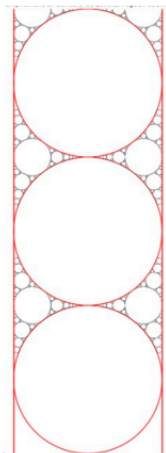
- (C_1, C_2, C_3) : triplet de cercles **mutuellement tangents**.
- C_{old} : le cercle extérieur complétant le quadruplet initial.

La formule récursive de tracé

Le nouveau cercle C_{new} s'insérant dans l'interstice vérifie :

$$C_{new} = 2(C_1 + C_2 + C_3) - C_{old}$$

Figure 16 – Récursion des cercles



Extension globale : Périodicité de l'empilement $\mathcal{B}$

Du motif local au pavage global
L'empilement complet $\mathcal{B}$ est obtenu par translations successives du motif générateur $\mathcal{B}_0$.

Interprétation physique

Ce pavage définit l'ensemble des matrices de stabilité admissibles sur tout le plan complexe.

Figure 17 – Pavage périodique par translation.

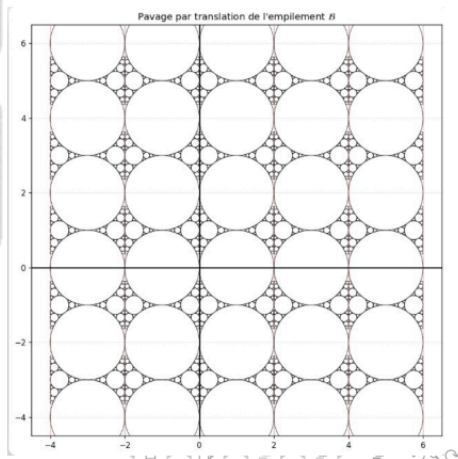
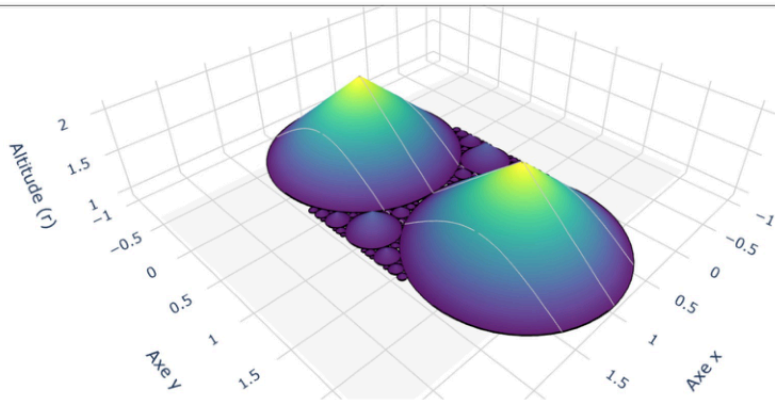


Figure 18 – Frontière 3D des matrices superharmoniques



Cette étude permet-elle d'expliquer les fractales ?

Le mécanisme d'émergence (Optimisation spatiale)

- **1. Le besoin de stabilité** : Le système s'éboule pour dissiper son surplus de grains.
- **2. La contrainte spatiale** : Les régions stables doivent paver l'espace continu sans se chevaucher.
- **3. La conséquence géométrique** : Cette contrainte spatiale force l'apparition continue de l'empilement d'Apollonius.

« *La fractale n'est pas un hasard esthétique, c'est l'unique solution géométrique trouvée par le sable pour s'équilibrer.* »

Merci de votre attention

Daoubih Yousef

TIPE 2026 – Modèle du tas de sable abélien

- PER BAK, CHAO TANG, KURT WIESENFELD : Self-organized criticality : An explanation of $1/f$ noise : Physical Review Letters, vol. 59, no 4, 1987.
- DEEPAK DHAR : The Abelian Sandpile Mode : Physical Review Letters, vol. 64, no 14, 1990.
- DOMINIQUE ROSSIN : Propriétés combinatoires de certaines familles d'automates cellulaires : Article universitaire
- HENRI DERYCKE : Études de relaxations dans le modèle du tas de sable sur la grille 2D : Rapport de Master, ENS Rennes
- HENRI DERYCKE : Thèse de doctorat sur le modèle du tas de sable abélien : Thèse de doctorat, Université Rennes 1
- LIONEL LEVINE, WESLEY PEGDEN, CHARLES K. SMART : The Apollonian structure of integer superharmonic matrices : Annals of Mathematics, vol. 186, no 1, 2017
- S. NORTHSHIELD : Complex Descartes Circle Theorem : The American Mathematical Monthly, Vol. 121, No. 10, pp. 927-931.

Annexe : L'unicité de l'état final

Notations et cadre :

- V : ensemble des sommets privé du puits .
- $c \in \mathbb{Z}_{\geq 0}^V$: configuration (nombre de grains par sommet).
- Δ_i : i -ème ligne de la matrice du Laplacien réduit du graphe.
- $s_i(c) = c - \Delta_i$: opérateur d'éboulement au site i (si $c_i \geq \deg_i$).

Démonstration de la commutativité : Supposons que deux sommets distincts i et j soient instables dans la configuration c . L'application successive des opérateurs d'éboulement donne :

$$s_i(s_j(c)) = s_i(c - \Delta_j) = (c - \Delta_j) - \Delta_i$$

$$s_j(s_i(c)) = s_j(c - \Delta_i) = (c - \Delta_i) - \Delta_j$$

Par la commutativité de l'addition vectorielle dans $\mathbb{Z}^V$, nous avons :

$$c - \Delta_j - \Delta_i = c - \Delta_i - \Delta_j \implies s_i \circ s_j = s_j \circ s_i$$

Annexe : La finitude de l'algorithme de stabilisation

Preuve par l'absurde : Supposons qu'il existe une suite infinie d'éboulements $s_1, s_2, \dots$ à partir d'une configuration finie. Le graphe étant fini, un sous-ensemble de sommets $I \subseteq V$ doit s'ébouler une infinité de fois.

Argument du flux vers le puits :

- Le sommet puits n'appartient pas à I car il ne s'éboule jamais.
- Le graphe est connexe et le puits est accessible depuis tout sommet : il existe donc un sommet $i_0 \in I$ ayant au moins un voisin $j \notin I$.
- Les sommets hors de I ne s'éboulent qu'un nombre fini de fois. Il existe donc un temps T après lequel plus aucun sommet de $V \setminus I$ ne s'éboule.

Contradiction sur la masse : Après le temps T , chaque éboulement de $i_0 \in I$ transfère au moins un grain vers $j \notin I$ sans que celui-ci ne puisse le renvoyer vers I . La masse totale $N(I) = \sum_{v \in I} c(v)$ diminue strictement à chaque occurrence de s_{i_0} . Or, $N(I)$ ne peut pas devenir négatif.

Conclusion : La séquence d'éboulements est nécessairement finie : l'algorithme de stabilisation **termine toujours**.

Annexe : code de stabilisation d'une grille de taille n (1/3)

```
from collections import deque

def stabiliser_grille(taille, grille_initiale):
    """
    Stabilise une configuration du tas de sable ab lien sur une
    grille rectangulaire.

    Param tres :
        taille (tuple) : (nombre_lignes, nombre_colonnes) de la
        grille.
        grille_initiale (list[list[int]]) : La configuration de
        d part.

    Retourne :
        list[list[int]] : La grille stabilis e.
    """
    lignes, colonnes = taille
    # Cr ation d'une copie de travail pour ne pas alt rer la
    liste d'origine
    grille = [ligne[:] for ligne in grille_initiale]
```

Annexe : code de stabilisation d'une grille de taille n (2/3)

```
# 1. Initialisation de la file des sites instables (
    Optimisation spatiale)
# On utilise deque (double-ended queue) pour des op rations en
    O(1)
sites_instables = deque()
for i in range(lignes):
    for j in range(colonnes):
        if grille[i][j] >= 4:
            sites_instables.append((i, j))

# Vecteurs de d placement pour les 4 voisins (Haut, Bas,
    Gauche, Droite)
directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

# 2. Boucle de stabilisation
while sites_instables:
    i, j = sites_instables.popleft()

    if grille[i][j] >= 4:
        # boulement par paquets : acc l ration drastique
            pour les grands N
        nb_eboulements = grille[i][j] // 4
        grille[i][j] %= 4 # Reste des grains sur le site
```

Annexe : code de stabilisation d'une grille de taille n (3/3)

```
# Distribution aux voisins
for di, dj in directions:
    ni, nj = i + di, j + dj

    # V rification des conditions aux limites (bords
    absorbants)
    if 0 <= ni < lignes and 0 <= nj < colonnes:
        grille[ni][nj] += nb_eboulements

        # Si le voisin devient instable suite      l'ajout,
        on l'enfile
        if grille[ni][nj] >= 4:
            sites_instables.append((ni, nj))

return grille
```

Annexe : Démonstration de la formule de Dhar (1/2)

L'éboulement comme opérateur local

Pour comprendre l'identité globale, analysons d'abord l'effet d'un **éboulement unique** sur un site précis x .

Soit T_x l'action d'ébouler le site x . Si on applique T_x à une configuration de sable η , la nouvelle configuration η_{suiv} devient :

$$\eta_{suiv} = \eta - 4\delta_x + \sum_{y \sim x} \delta_y$$

Bilan physique de l'opération :

- **Perte locale** : Le site x perd 4 grains (terme $-4\delta_x$).
- **Gain pour les voisins** : Chaque voisin direct y (noté $y \sim x$) reçoit 1 grain (terme $+\sum \delta_y$).

Note : δ_x désigne l'impulsion unité au site x (fonction qui vaut 1 en x , et 0 ailleurs).

Annexe : Démonstration de la formule de Dhar (2/2)

Sommation temporelle et bilan global

Soit $u(x)$ l'odomètre, c'est-à-dire le nombre total d'éboulements ayant eu lieu au site x durant tout le processus de stabilisation.

La configuration finale stabilisée η° est égale à la configuration initiale η à laquelle on ajoute l'effet cumulé de **tous** les éboulements :

$$\eta^\circ(x) = \eta(x) + \sum_{z \in V} u(z) \times \text{Effet}(T_z \text{ sur } x)$$

Si on se place au point x , ce point perd 4 grains pour ses propres éboulements $u(x)$, et gagne 1 grain pour chaque éboulement d'un voisin $u(y)$:

$$\eta^\circ(x) = \eta(x) + \underbrace{\sum_{y \sim x} u(y) - 4u(x)}_{\text{Laplacien discret } \Delta u(x)}$$

Annexe : Algorithme de simulation rayon de couverture (1/3)

Listing 1 – Préparation de la grille et de la file

```
import numpy as np
from collections import deque
import time

def calculer_rayon_tas(n_grains):
    # Taille de grille estimee (environ sqrt(N) + marge)
    taille = int(np.sqrt(n_grains)) + 20
    grille = np.zeros((taille, taille), dtype=int)
    centre = taille // 2

    # On injecte tout le sable au centre
    grille[centre, centre] = n_grains

    # On utilise deque pour une file efficace (O(1) pour popleft)
    # On y stocke les coordonnees des cases instables
    a_traiter = deque([(centre, centre)])
```

Annexe : Algorithme de simulation (2/3)

Listing 2 – Boucle d'éboulement optimisée

```
while a_traiter:
    x, y = a_traiter.popleft()

    if grille[x, y] >= 4:
        # On eboule par paquets de 4 grains pour accélérer
        nb_eboulements = grille[x, y] // 4
        grille[x, y] %= 4

        # Distribution aux 4 voisins (haut, bas, gauche, droite)
        for dx, dy in [(-1, 0), (1, 0), (0, 1), (0, -1)]:
            nx, ny = x + dx, y + dy
            grille[nx, ny] += nb_eboulements

            # Si le voisin devient instable, on l'ajoute à la file
            if grille[nx, ny] >= 4:
                a_traiter.append((nx, ny))
```

Annexe : Algorithme de simulation (3/3)

Listing 3 – Mesure du rayon et interaction utilisateur

```
# Recuperation des cases qui contiennent du sable
points = np.argwhere(grille > 0)

# On calcule la distance maximale au centre (Rayon discret)
# On utilise le carre pour eviter les racines inutiles dans la
    boucle
distances_sq = (points[:, 0] - centre)**2 + (points[:, 1] -
    centre)**2
return np.sqrt(np.max(distances_sq))

if __name__ == "__main__":
    n = int(input("Entrez le nombre de grains N : "))
    debut = time.time()
    resultat_r = calculer_rayon_tas(n)
    temps = time.time() - debut
    print(f"Rayon mesure : {resultat_r:.4f} | Temps : {temps:.2f}s"
        )
```

Annexe : Simulation et Odomètre (1/4)

```
def simuler_avec_odometre(N, taille):  
    # Grille de sable et grille de comptage (odometre)  
    grille = np.zeros((taille, taille), dtype=int)  
    odometre = np.zeros((taille, taille), dtype=int)  
    centre = taille // 2  
  
    grille[centre, centre] = N  
    a_traiter = deque([(centre, centre)])  
  
    while a_traiter:  
        x, y = a_traiter.popleft()  
        if grille[x, y] >= 4:  
            nb = grille[x, y] // 4  
            odometre[x, y] += nb # Accumulation  
            grille[x, y] %= 4  
            # ... (distribution aux voisins)
```

Annexe : Traitement des données (2/4)

```
# Recuperation des cases atteintes
indices = np.argwhere(odometre > 0)
centre = taille // 2

# Calcul du rayon R_N du tas
r_max = np.sqrt(np.max(((indices[:,0]-centre)**2 +
                        (indices[:,1]-centre)**2))

# Extraction de la coupe horizontale centrale
coupe = odometre[centre, centre:]
x_coords = np.arange(len(coupe))

# Normalisation : U/R^2 et x/R
u_norm = coupe / (r_max**2)
x_norm = x_coords / r_max
```

Annexe : Tracé des courbes (3/4)

```
liste_N = [50000, 100000, 250000]

plt.figure(figsize=(10, 6))

for N in liste_N:
    odo = simuler_avec_odometre(N, taille)
    # ... (calculs de normalisation ci-dessus)

    # Tracé de la courbe superposée
    plt.plot(x_norm, u_norm, label=f'N={N}')
```

Annexe : Mise en forme (4/4)

```
plt.title("Convergence exp rimentale (Scaling)")
plt.xlabel("Position normalisee (x / R_N)")
plt.ylabel("Odometre normalise (U / R_N^2)")
plt.legend()
plt.grid(True)
plt.show()
```

Definition (Matrice superharmonique entière)

Une matrice symétrique $A \in \mathbb{R}^{2 \times 2}$ est dite **superharmonique entière** s'il existe une fonction $g : \mathbb{Z}^2 \rightarrow \mathbb{Z}$ telle que :

$$\begin{cases} \Delta g(x) \leq 1 & \forall x \in \mathbb{Z}^2 \\ g(x) = \frac{1}{2}x^T A x + o(|x|^2) & (|x| \rightarrow \infty) \end{cases}$$

Annexe : Théorème de Descartes Généralisé

Soient quatre cercles mutuellement tangents de courbures $k_i = 1/r_i$. La relation classique est :

$$\left(\sum_{i=1}^4 k_i \right)^2 = 2 \sum_{i=1}^4 k_i^2$$

Pour inclure la position des centres (x, y) , nous définissons le vecteur de courbure complexe $C = (k, w)$ où $w = k \cdot (x + iy)$.

Extension complexe : Le théorème s'applique directement aux coordonnées C_i (Lagarias et al., 2002) :

$$\left(\sum_{i=1}^4 C_i \right)^2 = 2 \sum_{i=1}^4 C_i^2$$

Cette formulation permet de traiter les centres et les rayons comme un seul objet mathématique cohérent.

Relation de tracé : Pour un triplet tangent (C_1, C_2, C_3) , la relation quadratique se linéarise, permettant de calculer le cercle successeur C_{new} :

$$C_{new} = 2(C_1 + C_2 + C_3) - C_{old}$$

- **Initialisation :** Création du quadruplet initial $\{C_1, C_2, C_3, C_{old}\}$ composé de droites (courbure $k = 0$) et de grands cercles tangents.
- **Principe de récursion :**
 - ① Calculer C_{new} via la formule linéaire.
 - ② Tracer le cercle C_{new} .
 - ③ Appeler récursivement la fonction pour les 3 nouveaux interstices formés : (C_1, C_2, C_{new}) , (C_1, C_3, C_{new}) et (C_2, C_3, C_{new}) .

Annexe : Démonstration : De la forme quadratique à la linéarité

Partons du théorème de Descartes généralisé :

$$\left(\sum_{i=1}^4 C_i\right)^2 = 2\sum_{i=1}^4 C_i^2$$

En développant le carré du membre de gauche, nous obtenons :

$$\sum_{i=1}^4 C_i^2 + 2 \sum_{1 \leq i < j \leq 4} C_i C_j = 2 \sum_{i=1}^4 C_i^2$$

En simplifiant, on extrait l'équation du second degré en C_4 (noté C_{new}) :

$$C_{new}^2 - 2C_{new} \left(\sum_{i=1}^3 C_i\right) + \left(\sum_{i=1}^3 C_i^2 - 2 \sum_{1 \leq i < j \leq 3} C_i C_j\right) = 0$$

Annexe : Démonstration : Relations de Viète

L'équation est de la forme $C_x^2 - 2S \cdot C_x + P = 0$, avec $S = \sum_{i=1}^3 C_i$.

Cette équation possède deux racines : C_{new} et C_{old} (les deux cercles tangents au triplet $\{C_1, C_2, C_3\}$).

D'après les **relations de Viète** sur les polynômes du second degré, la somme des racines est égale au coefficient de degré 1 opposé :

$$C_{new} + C_{old} = 2(C_1 + C_2 + C_3)$$

Conclusion : La linéarisation provient directement de la structure symétrique des racines de l'équation de Descartes :

$$C_{new} = 2(C_1 + C_2 + C_3) - C_{old}$$

Annexe : Code python de traçage de l'empilement en bande(1/4)

```
import matplotlib.pyplot as plt

def plot_circle(ax, c, color='blue', lw=0.8):
    """
    Trace un cercle partir de sa coordonn e complexe de
    courbure (k, kz).
    k : courbure (inverse du rayon)
    kz : produit de la courbure par l'affixe complexe du centre (kz
        = k * z)
    """
    k, kz = c
    # Les droites (k=0) sont des fronti res , non trac es par
    # cette fonction
    if abs(k) < 1e-10:
        return

    rayon = 1 / abs(k)
    centre = kz / k
    circle = plt.Circle((centre.real, centre.imag), rayon,
                        color=color, fill=False, lw=lw)
    ax.add_patch(circle)
```

Annexe : Code python de traçage de l'empilement en bande(2/4)

```
def apollonian_band_recursive(ax, c1, c2, c3, C_old, depth):
    """
    Algorithme de construction fractale par r cursion.
    c1, c2, c3 : Triplet de cercles tangents parents.
    C_old      : Le cercle tangent ext rieur au triplet courant.
    """
    if depth == 0:
        return
    # Calcul du nouveau cercle C_new via la relation lin aire de
    Descartes
    k_new = 2 * (c1[0] + c2[0] + c3[0]) - C_old[0]
    kz_new = 2 * (c1[1] + c2[1] + c3[1]) - C_old[1]
    C_new = (k_new, kz_new)
    # Trac du cercle calcul
    plot_circle(ax, C_new, color='black', lw=0.5)
    # Branchement r cursif : C_new devient un nouveau parent pour
    3 sous-intersections
    apollonian_band_recursive(ax, c1, c2, C_new, c3, depth - 1)
```

Annexe : Code python de traçage de l'empilement en bande(3/4)

```
def tracer_empilement_bande():  
    fig, ax = plt.subplots(figsize=(6, 12))  
  
    # Initialisation des frontieres (k=0) et des cercles  
    generateurs (k=1)  
    C_L = (0, -1 + 0j) # Ligne de gauche (x=0)  
    C_R = (0, 1 + 0j) # Ligne de droite (x=2)  
  
    # Trace des frontieres rouges  
    ax.axvline(0, color='red', lw=2)  
    ax.axvline(2, color='red', lw=2)  
  
    # Generation de la colonne centrale (grands cercles de rayon 1)  
    limite_y_bas = 0  
    limite_y_haut = 6  
  
    for y in range(limite_y_bas, limite_y_haut, 2):  
        # Cercles de courbure 1 empiles verticalement  
        C_bas = (1, 1 + (y+1)*1j)  
        C_haut = (1, 1 + (y+3)*1j)
```

Annexe : Code python de traçage de l'empilement en bande(4/4)

```
    plot_circle(ax, C_bas, color='red', lw=1.5)
    plot_circle(ax, C_haut, color='red', lw=1.5)
# Lancement de la recursion sur les deux cavites (gauche et
    droite)
apollonian_band_recursive(ax, C_L, C_bas, C_haut, C_R, depth=5)
apollonian_band_recursive(ax, C_R, C_bas, C_haut, C_L, depth=5)

# Reglages esthetiques de la figure
ax.set_aspect('equal')
ax.set_xlim(-0.5, 2.5)
ax.set_ylim(limite_y_bas, limite_y_haut)
ax.axis('off')
plt.title(r"Empilement d'Apollonius : Geometrie fractale  $B$ ")
plt.tight_layout()
plt.show()

tracer_empilement_bande()
```