



# Projet de Simulation Aléatoire

Simulation de marches aléatoires auto-évitantes

Lucas BENIGNI
Pierre BERTRAND

12 janvier 2015

# Table des matières

# Chapitre 1

# Introduction

Au cours de notre projet, nous nous sommes intéressés à la simulation de marches aléatoires auto-évitantes (MAAE) en dimension 2. La restriction à la dimension deux s'est avérée être un handicap. En effet, la difficulté dans la simulation de MAAE réside dans le fait qu'une marche qu'on est en train de construire puisse être bloquée. Nous verrons dans ce rapport que la probabilité que deux MAAE puisse fusionner est très faible en dimension 2 notamment lorsque leur taille est importante.
Nous commencerons par introduire quelques notations.
Puis nous exposerons les algorithmes que nous avons utilisés.
Ensuite nous comparerons la complexité temporelle de chacun de ces algorithmes.
Enfin nous utiliserons les algorithmes pour évaluer quelques données du problème. Nous verrons l'influence de la taille de la marche sur la distance à l'origine de celle-ci. Entre autres, nous exposerons l'influence de la taille de la marche sur la marche. De plus, nous calculerons la probabilité d"un événement rare grâce à une méthode particulaires proche de celles vues en cours.

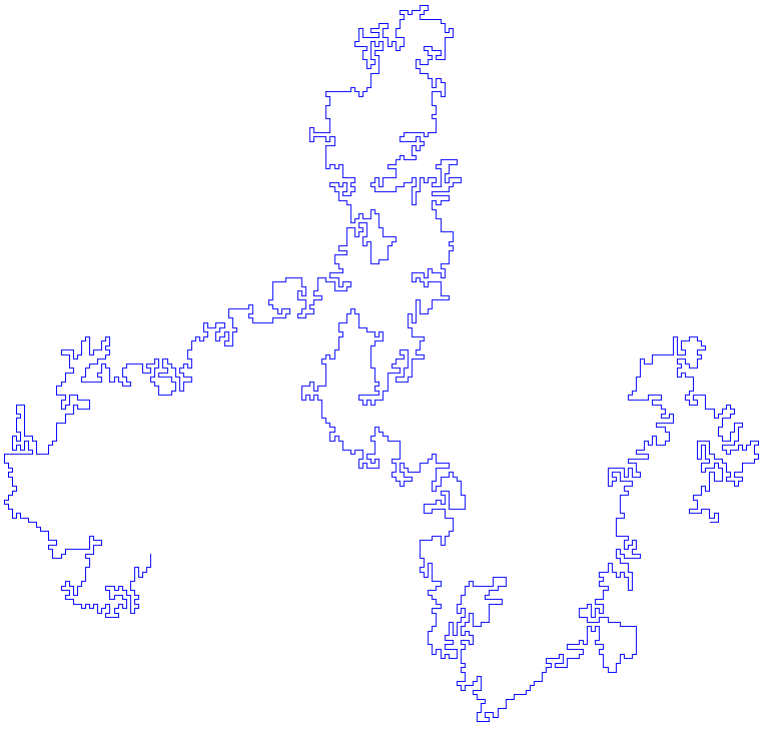

FIGURE 1.1 – Marche Aléatoire Auto-Evitante

# Chapitre 2

# Notations

**Definition 1.** *MAAE Une marche aléatoire auto-évitante (MAAE) $w$ est issue de l'origine. Il s'agit d'une succession de points dans $\mathbb{R}^2$ : $w_0, w_1, ..., w_n$ où $n$ est la taille de la marche. De plus, si l'on désigne par $||\cdot||_1$ la norme 1 habituelle en dimension 2 alors on a :*

$$\forall i < n, ||w_{i+1} - w_i||_1 = 1$$

On désigne ici par w une MAAE et on présente quelques paramètres de celle-ci :
- $|w|$ est la taille de la marche
- $w_i$ est le ième point de la marche $(0 \leq i \leq |w|)$
- $w[i:j]$ est la partie de la marche entre $i$ et $j$
- $d_0(w) = max(||w_i||_2; i \in \{1, ..., |w|\})$ est la distance à l'origine de la chaîne $w$
- $T_{\mathcal{A}}^N$ le temps moyen en secondes de génération d'une MAAE de taille $N$ pour l'algorithme $\mathcal{A}$

# Chapitre 3

# Algorithmes

Dans cette section, nous présentons les différents algorithmes que l'on a utilisés pour simuler des MAAE.

## 3.1 NRRW

Le premier algorithme que l'on a utilisé est le plus simple. Il consiste à générer des marches qui ne retournent pas immédiatement sur leurs pas. Si une marche est auto-évitante, elle respecte cette condition. On peut donc se restreindre à ces marches.
A chaque pas, plutôt que de choisir un choix parmi quatre au hasard, on en choisit un parmi les trois qui permettent à la marche de ne pas revenir sur sa position précédente. L'algorithme **SimpleSampling** pour générer une marche de taille N est le suivant :

**SimpleSampling(N)**
Start
    $w_0 \leftarrow 0$
    $w_1 \leftarrow$ un voisin aléatoire parmi les quatre de $w_0$
    *For* i *from* 2 *to* $N$ :
        $w_i \leftarrow$ un voisin aléatoire de $w_{i-1}$ différent de $w_{i-2}$
        *If not* **Check**$(w[0:i])$ *Then Goto* Start
End

Où la fonction **Check** consiste à vérifier que la marche créée est bien auto-évitante :

**Check(w)**
Start
    *For* i *from* 1 *to* $|w|$ :
        *If* $w_i \in \{w_0, ..., w_{i-1}\}$
            Renvoyer 0
    Renvoyer 1
End

On expose ici la complexité observée de **SimpleSampling** afin de savoir quelle est la taille des MAAE que l'on peut espérer simuler rapidement avec un tel algorithme. Pour cela on simule une succession de MAAE de même longueur et on calcule le temps moyen pour en générer une. En abscisse de la Figure 3.2 on observe la longueur de la marche. En ordonnée le temps mis en seconde pour en générer une de cette

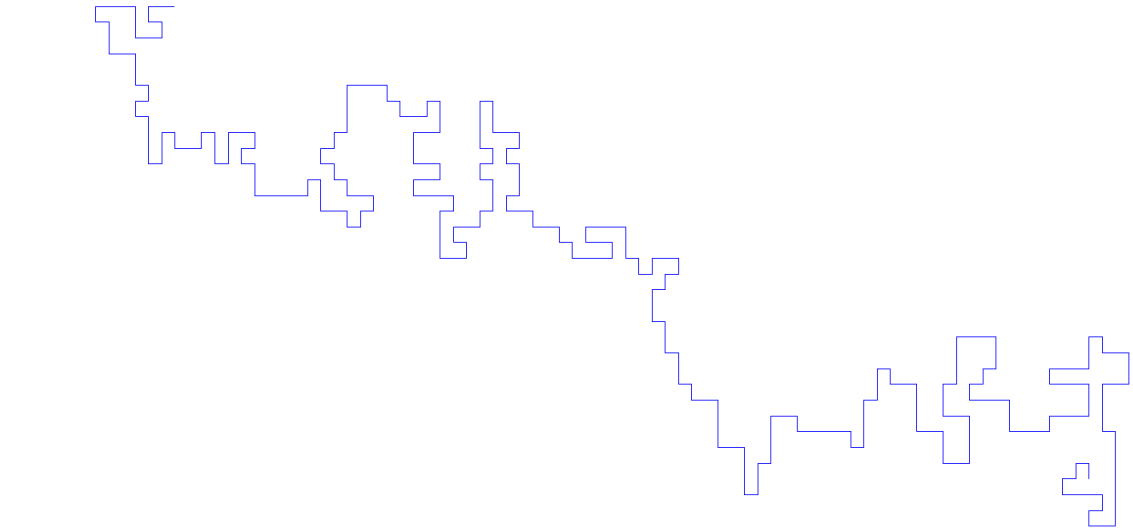

FIGURE 3.1 – MAAE de 300 pas simulée avec **SimpleSampling**, Temps écoulé : 4.45 secondes

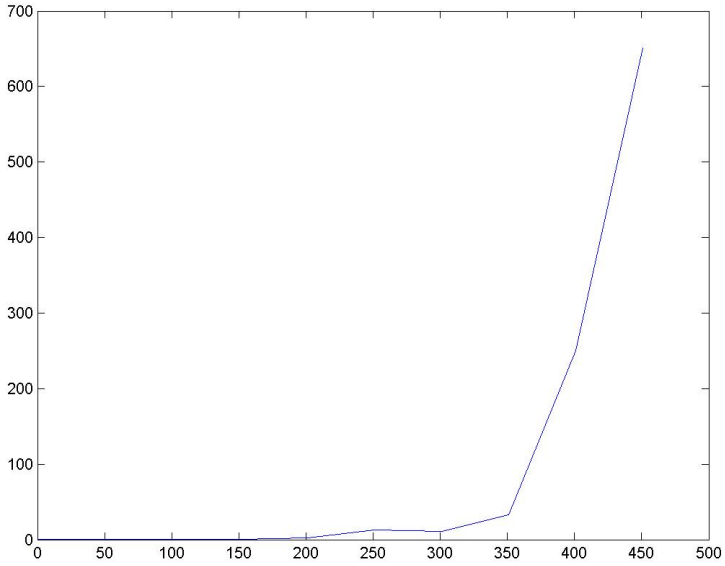


FIGURE 3.2 – Complexité en temps de NRRW

longueur en moyenne. On voit que la complexité en temps est largement exponentielle empêchant d'espérer créer des marches de longueur trop importantes. Déjà, pour une marche de longueur 400, il faut attendre en moyenne 5 minutes.
Comme la complexité sur Figure 3.2 est dure à estimer, on affiche un second graphe dans lequel on a passé la complexité au log (Figure 3.3).
En abscisse on a la même chose que précédemment mais en ordonnée on n'observe non plus le temps de réalisation de l'algorithme mais le logarithme de celui-ci. Cela permet de s'approcher d'un graphe linéaire.

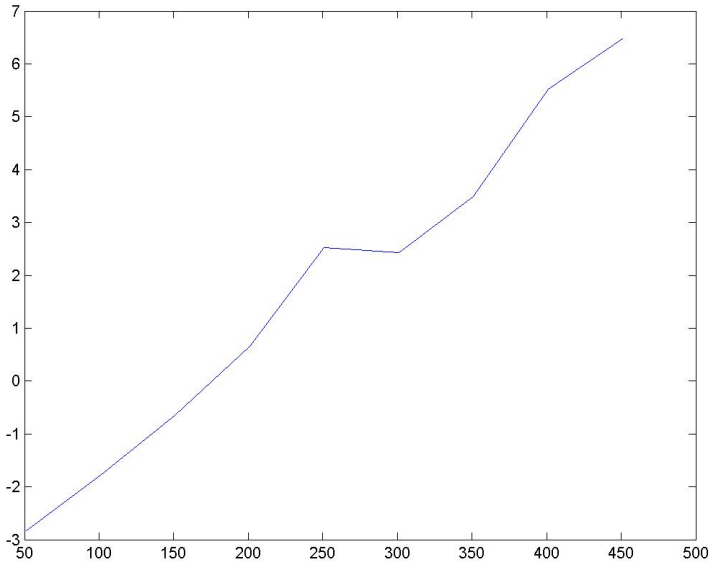


FIGURE 3.3 – Complexité en temps de NRRW sous une meilleure échelle

Si l'on se fie au graphe précédent et que l'on fait une régression linéaire alors, avec les notations introduites on obtient :

$$\begin{aligned} \log T^N_{NRRW} &= -4.032 + 0.023 \cdot N \\ T^N_{NRRW} &= e^{-4.032} \cdot e^{0.023 \cdot N} \\ T^N_{NRRW} &= 1.023^{N-175} \end{aligned}$$

Par exemple, pour générer une MAAE de taille 500, on peut d'attendre à une attente de l'ordre de 27 minutes. Comme la précision dans la détermination des constantes est faible et que l'on ne sait même pas quel type de complexité attendre, on ne peut pas appliquer notre formule pour des longueurs loin de celles utilisées sur le graphe. Par exemple si $N = 1000$ on obtient 39000 heures d'attente ce qui est loin de la réalité.
De plus, la complexité observée est en secondes et dépend donc des caractéristiques de l'ordinateur.

## 3.2 V-S

On présente ici l'algorithme de Verdier-Stockmayer.
Cet algorithme génère une chaîne de Markov réversible pour la distribution uniforme sur l'ensemble des marches aléatoires de longueur celle de la marche donnée en entrée.

**VerdierStockmayer(w,T)**
Start
$w_t \leftarrow w$
*For* t *from* 1 *to* $T$ :
$\tilde{w} \leftarrow w_t$
Choisir $I$ au hasard parmi $\{0, ..., |w|\}$
*If* $0 < I < |w|$ *Then*

$\tilde{w}(I) \leftarrow w_t(I-1) + w_t(I+1) - w_t(I)$
*If* $I == 0$
$\tilde{w}(I) \leftarrow$ un voisin de $w_t(1)$ différent de $w_t(0)$ et $w_t(2)$
*If* $I == |w|$
$\tilde{w}(|w|) \leftarrow$ un voisin de $w_t(|w|-1)$ différent de $w_t(|w|-2)$ et $w_t(|w|)$
*If* **Check**$(\tilde{w})$ *Then*
$w_t \leftarrow \tilde{w}$
End

## 3.3 Dimerization

L'algorithme de dimérization est très simple. Il consiste à appliquer la méthode "Diviser pour Régner". A la place de générer directement une MAAE de longueur $N$ on en génère récursivement deux de longueur $\frac{N}{2}$ et ce jusqu'à obtenir une longueur suffisamment petite pour la générer efficacement avec l'algorithme **NRRW** décrit plus haut. La complexité de l'algorithme vient du fait que les deux chaînes de longueur moitié, une fois concaténées, ne forment pas forcément une MAAE. La fonction récursive est la suivante :

**Dimerization(N)**
Start
Set $N_0$
*If* $N < N_0$
$w \leftarrow$ **NRRW(N)**
*Else*
$w_1 \leftarrow$ **Dimerization**$(\frac{N}{2})$
$w_2 \leftarrow$ **Dimerization**$(\frac{N}{2})$
$\tilde{w} \leftarrow$ **Concat**$(w_1, w_2)$
*If* **Check**$(\tilde{w})$ *Then*
$w \leftarrow \tilde{w}$
*Else*
$w \leftarrow$ **Dimerization(N)**
*End*

Où la fonction **Concat** consiste à concaténer les deux marches passées en argument.

## 3.4 DimerizationSet

Pour pallier la difficulté de fusionner deux MAAE en créant une MAAE ce qui a peu de chances d'arriver comme on le verra dans le chapitre sur la probabilité de concaténation, nous avons créé une fonction **DimerizationSet**.
Elle suit le même schéma que la dimérization normale, simplement, lorsqu'elle utilise **NRRW** pour créer une chaîne de longueur $N_0$, elle crée en fait $M$ chaînes indépendantes. Il y a donc deux paramètres : $N_0$ et $M$. Ensuite, au moment de fusionner, elle dispose de deux ensembles $Set_1$ et $Set_2$ et essaie de fusionner les éléments de ces deux ensembles. Elle ne garde que des MAAE dont les deux parties sont différentes pour être sûr d'avoir des MAAE indépendantes et renvoie donc un ensemble de MAAE indépendantes de taille forcément inférieure à $M$.
Plus formellement, l'algorithme est le suivant :

**Dimerization(N)**
Start
Set $N_0$
    *If* $N < N_0$
        $Set \leftarrow \emptyset$
        *For* m *from* 1 *to* M
            $Set \leftarrow [Set, \mathbf{NRRW}(N)]$
    *Else*
        $Set_1 \leftarrow \mathbf{Dimerization}(\frac{N}{2})$
        $Set_2 \leftarrow \mathbf{Dimerization}(\frac{N}{2})$
        $Set \leftarrow \emptyset$
        *For* $w_1 \in Set_1$
            *For* $w_2 \in Set_2$
                $\tilde{w} \leftarrow \mathbf{Concat}(w_1, w_2)$
                *If* $\mathbf{Check}(\tilde{w})$ *Then*
                    $w \leftarrow \tilde{w}$
                    $Set_1 \leftarrow Set_1 - w_1$
                    $Set_2 \leftarrow Set_2 - w_2$
        *If* $Set == \emptyset$
            $w \leftarrow$ **Dimerization(N)**
*End*

Cet algorithme **DimerizationSet** a beaucoup d'avantages, d'abord il peut renvoyer plusieurs chaînes d'un coup qui sont toutes indépendantes. Surtout, au moment de la première fusion, il a deux ensembles de M marches ce qui lui permet de tenter $M^2$ fusions. L'algorithme initial **Dimerization** doit générer $M^2$ appels à **NRRW** pour avoir autant d'essais.
Le problème de **Dimerization** est qu'il ne se donne pas assez d'essais à chaque appel récursif pour avoir une chance raisonnable de réussir.
Le nouvel algorithme a une chance strictement plus importante de réussir chaque fusion que **Dimerization**.

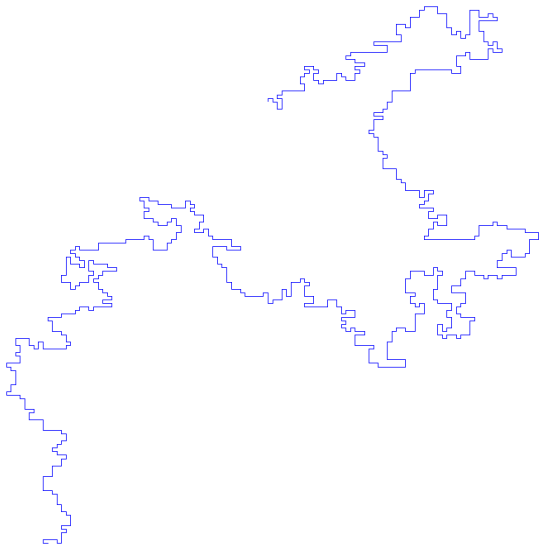

FIGURE 3.4 – MAAE de 1000 pas simulée avec **DimerizationSet**, Temps écoulé : 23 minutes

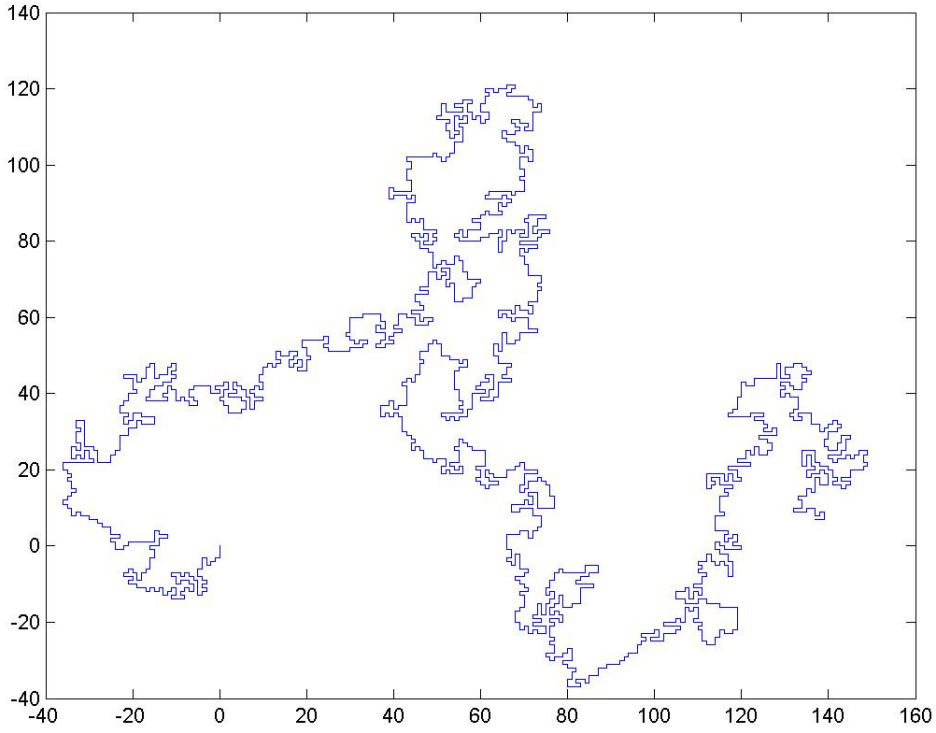


FIGURE 3.5 – MAAE de 2400 pas simulée avec **DimerizationSet**

# Chapitre 4

# Distance moyenne d'une MAAE

## 4.1 Définition et premier graphe

Dans ce chapitre nous essayons de déterminer la distance à l'origine moyenne d'une MAAV de longueur donnée. Il s'agit donc d'évaluer pour une distance N fixée :

$$d_N = \frac{1}{\#\mathcal{W}_N} \cdot \sum_{w \in \mathcal{W}_N} d_0(w)$$

Pour cela, on simule une succession de MAAE de longueur N et on calcule la moyenne de leur distance à l'origine. Si la simulation renvoie une distribution uniforme, on sait alors que la moyenne calculée converge vers $d_N$ lorsque le nombre de MAAE simulées tend vers l'infini. Évidemment, on ne peut générer qu'un nombre fini de MAAE. Pour autant, lorsqu'on génère suffisamment de MAAE en utilisant l'algorithme de dimérization qui est le plus efficace, on obtient la courbe suivante :

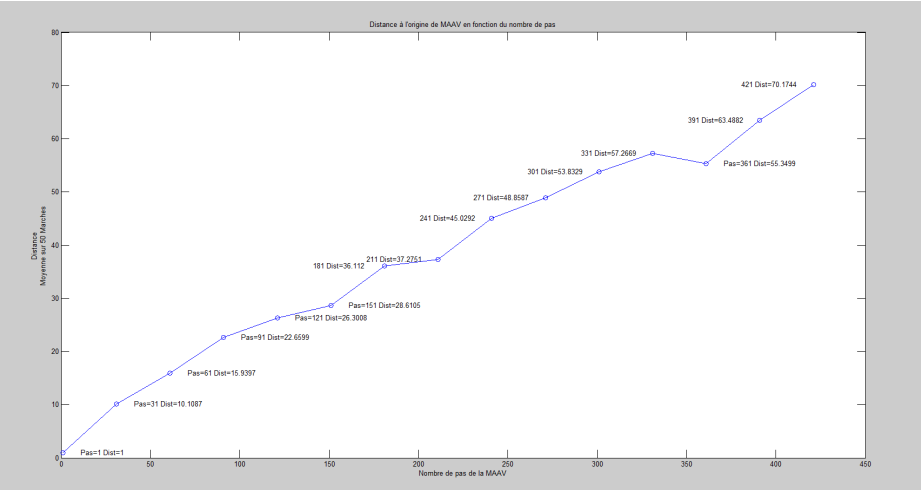


FIGURE 4.1 – Distance à l'origine en fonction de la longueur

L'axe des abscisse indique la longueur des marches générées tandis que celui des ordonnées indique la distance moyenne statistique pour les marches générées par rapport à l'origine.
Les marches générées ont une longueur entre 1 et 421. Les longueurs des MAAE sont espacées de 30 entre deux consécutives. Cela permet d'avoir une idée assez précise de la distribution de $d_N$ en fonction de $N$ sans pour autant simuler des marches pour chaque longueur ce qui serait trop couteux. Ensuite, on interpole linéairement les points obtenus pour avoir la distribution supposée de notre fonction. Par la suite, nous allons voir qu'étant donné la conjecture sur cette distribution générer des séries tous les 30 pas était amplement suffisant.

## 4.2 Conjecture

Dans la littérature, on trouve une conjecture énonçant le fait que $d_N$ se comporte comme $n^\alpha$ pour un certain $\alpha$ non identifié.
Certaines approximations numériques et simulations effectuées dans les références permettent d'estimer $\alpha$ aux alentours de 0.75. Afin de voir si notre courbe précédente présentée en Figure 4.1 est une bonne estimation de la longueur, nous comparons notre courbe à la conjecture dans la courbe suivante :

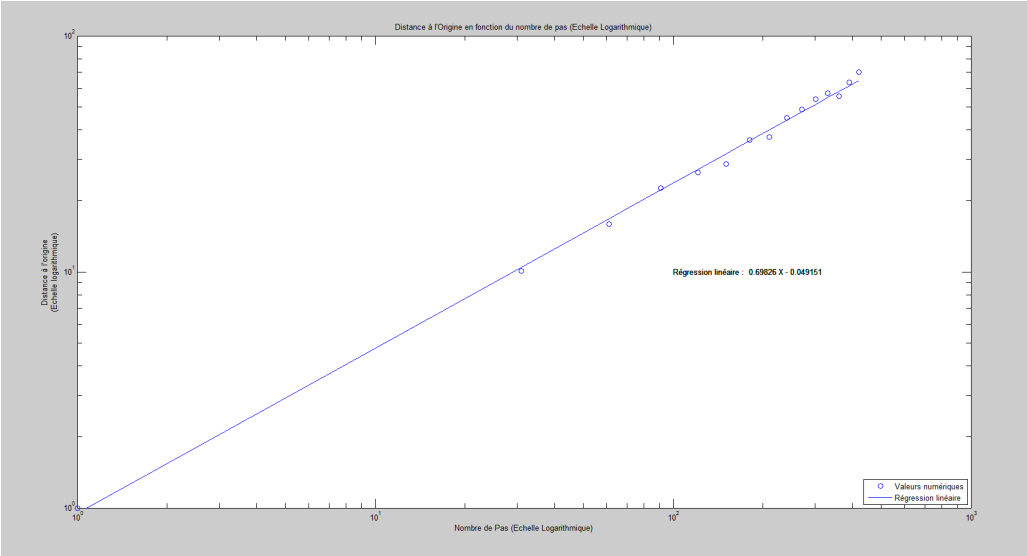


FIGURE 4.2 – Distance à l'origine en fonction de la longueur (échelle logarithmique)

En abscisse, on observe la longueur des MAAE générées en échelle logarithmique et de même, en ordonnée, on observe la longueur estimée de la marche en échelle logarithmique. Si nos estimations vont dans le sens de la conjecture, nous devrions obtenir une droite de coefficient 0.75. Comme il y a des erreurs statistiques comme dans toute estimation via simulation, nous savons qu'il nous faut faire une régression linéaire pour obtenir un coefficient moyen de nos estimations.
En faisant cela, on obtient une droite qui concorde très bien avec nos estimations :aucune ne s'en éloigne beaucoup notamment si l'on compare à la Figure 3.3. Cela tend à confirmer l'hypothèse selon laquelle $d_N$ se comporte comme $n^\alpha$.
Le $\alpha$ que nous obtenons dans nos simulations est 0.7. Or la conjecture donnée par des simulations plus précises au sein de la littérature indique que $\alpha$ doit valoir 0.75.
Nous ne sommes pas si loin de l'estimation la plus précise connue et ce alors que nous avons fait des simulations assez élémentaires sur des MAAE de longueur plutôt petites.

# Chapitre 5

# Probabilité de concaténation

Dans ce chapitre, nous nous intéressons à la probabilité pour que deux marches de taille $N$ générées aléatoirement créent -une fois concaténées- une nouvelle MAAE de taille $2N$.
Pour cela, nous générons une succession de MAAE, de taille $N$ donnée, toutes indépendantes et nous regardons parmi tous les couples de MAAE simulés combien fusionnent pour créer une nouvelle MAAE. Ce faisant, nous pouvons créer le graphe suivant qui donne la probabilité de concaténation en fonction de la longueur des deux marches.

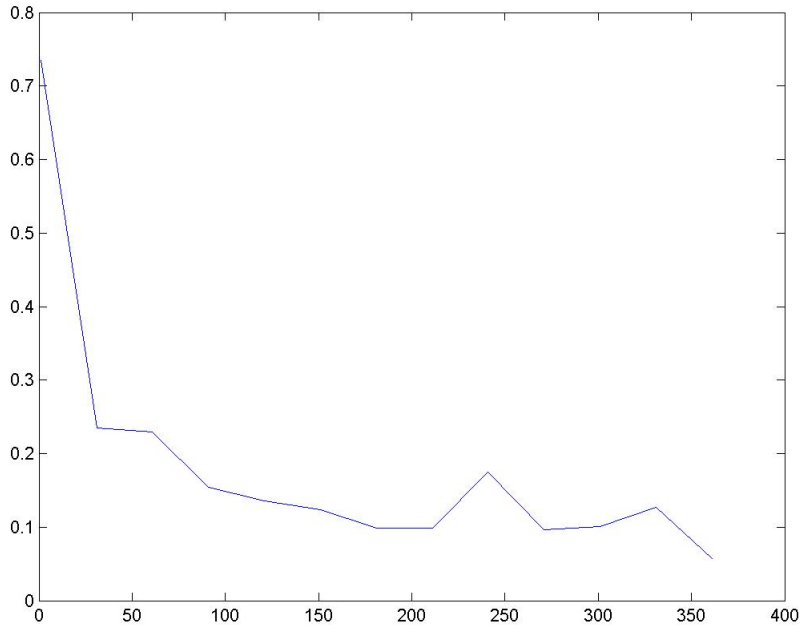


FIGURE 5.1 – Probabilité de bonne concaténation

Des simulations ont été réalisées pour des marches de taille $1 + 30 \cdot k$ où k est entre 0 et 10. En abscisse on lit la longueur des deux MAAE et en ordonnée la probabilité pour deux telles marches de fusionner en créant une nouvelle MAAE.
Bien sûr lorsque $k = 0$ et que les marches sont de longueur 1, la probabilité théorique est de 0.75. Pour que les deux fusionnent bien, ils suffit en effet que la seconde prenne une direction parmi les trois sur quatre différentes de l'opposé de la première marche.

Ensuite, même pour des marches de petite taille, on voit que la probabilité de bien fusionner est autour de 10 à 20%.
Cette observation nous permet de comprendre pourquoi la dimérization n'est pas très efficace en dimension deux.
Si l'on note $e_k$ le nombre moyen d'essais de fusion nécessaires pour créer une marche de longueur $2^k$ grâce à la dimérization alors on va avoir :

- $e_0 = 3$
- $e_k \geq 5 * e_{k-1}$ *(Proba de fusion inférieure à 20%)*

Au final on obtient : $e_k \geq 5^k * 3$.
Par exemple, pour simuler une MAAE de longueur 2000 avec la dimérization (si le $N_0$ de l'algo est fixé à 1) il faut au moins $5^{10} * 3 = 2.9 \cdot 10^7$ essais.
La dimérization n'est donc pas si efficace en dimension deux du fait de la difficulté de bien fusionner des MAAE.
Aussi, nous utilisons dans l'algorithme **Dimerization** un $N_0$ assez grand (de l'ordre de 300) car l'algorithme **NRRW** précédemment exposé est amplement suffisant pour des marches de longueur inférieure à 300.

# Chapitre 6

# Simulation d'évènements rares

Dans ce chapitre, nous nous intéressons à deux cas de simulations d'évènements rares : tout d'abord la probabilité d'avoir une longue MAAE (au sens de la distance à l'origine $d_0$), et ensuite la probabilité d'avoir un retour vers l'origine après s'en être éloigné. Pour calculer ces probabilités, nous avons utilisé une méthode particulaire introduit dans **??**.

## 6.1 Algorithme Particulaire Utilisé

On créé une famille de $N$ MAAE indépendantes que nous allons modifier à chaque étape. Pour calculer des probabilités très faibles (de l'ordre de $10^{-28}$), nous devons lancer l'algorithme avec un grand nombre de particules. C'est pourquoi nous nous sommes limités à des MAAE de taille inférieure à 100 afin d'avoir une simulation rapide. Si on veut simuler l'évènement $A$, on suppose qu'il existe une suite $(B_i)_{i\leq m}$ d'ensembles vérifiant

$$B_1 \supset B_2 \supset \ ... \ \supset B_{m-1} \supset B_m = A$$

L'algorithme est le suivant :

**RareEvent**
Start
Set $N$
*Walks* $\leftarrow$ *N-échantillon i.i.d de MAAE utilisant* **Dimerization**(100)
  *For* h *from* 1 *to* m *do*
    $I_1 \leftarrow$ *Indices des marches telles qu'il existe* $T_k^h$ *tel que Walks(k,*$T_k^h$*)*$\in B_h$
    $I_0 \leftarrow I_1^c$
    $P_h \leftarrow \dfrac{\#I_1}{N}$

      *If* $I_1 \neq \emptyset$
        *For* $k \in I_0$
          $l \leftarrow$ *Valeur choisie uniformément dans* $I_1$
          *Walks(k,1 :*$T_l^h$*)* $\leftarrow$ *Walks(l,1 :*$T_l^h$*)*
          $T_k^h \leftarrow T_l^h$
        *For k from 1 to N*
          *While* **Check**(**Concat**(*Walks(k,*1: $T_k^h$), $w$))==0
            $w \leftarrow$ **Dimerization**$(100 - T_k^h + 1)$
          *Walks(k)* $\leftarrow$ **Concat**(*Walks(k,*1: $T_k^h$), $w$)
Output : $\hat{P} = \prod_{i=1}^{m} P_i$

## 6.2 Probabilité de Grandes MAAE

Tout d'abord, nous avons utilisé l'algorithme précédent pour calculer la probabilité qu'une MAAE atteigne une grande distance à l'origine. Nous avons ainsi réalisé deux simulations, la première avec des marches de 50 pas puis avec des marches de 100 pas. Dans les deux cas, nous sommes partis avec un nombre initial de $N = 1000$ particules.
Nous avons cherché à calculer $\mathbb{P}(A_d)$ avec $A_d = \{\exists k,\, d_0(w_k) \geq d\}$ où $d$ est un seuil donné. La décomposition en suite d'ensembles $(B_i)$ est immédiate en prenant une suite $d_k$, telle que $d_1 \leq d_2 \leq ... \leq d_m = d$. Afin d'obtenir un résultat plus représentatif nous retenons la moyenne sur 50 exécutions de l'algorithme.

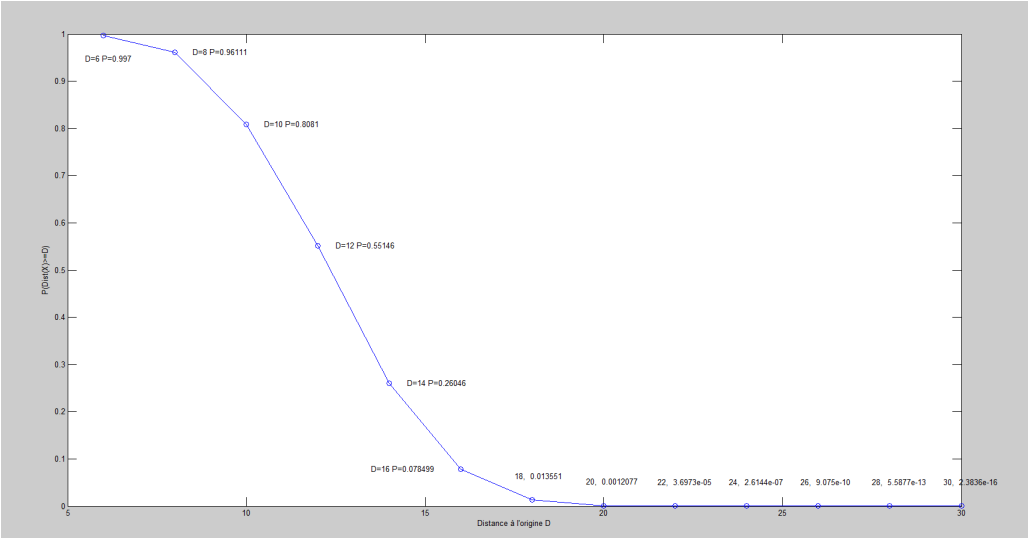


FIGURE 6.1 – $\mathbb{P}(A_d)$ en fonction de $d$ pour des MAAE de 50 pas

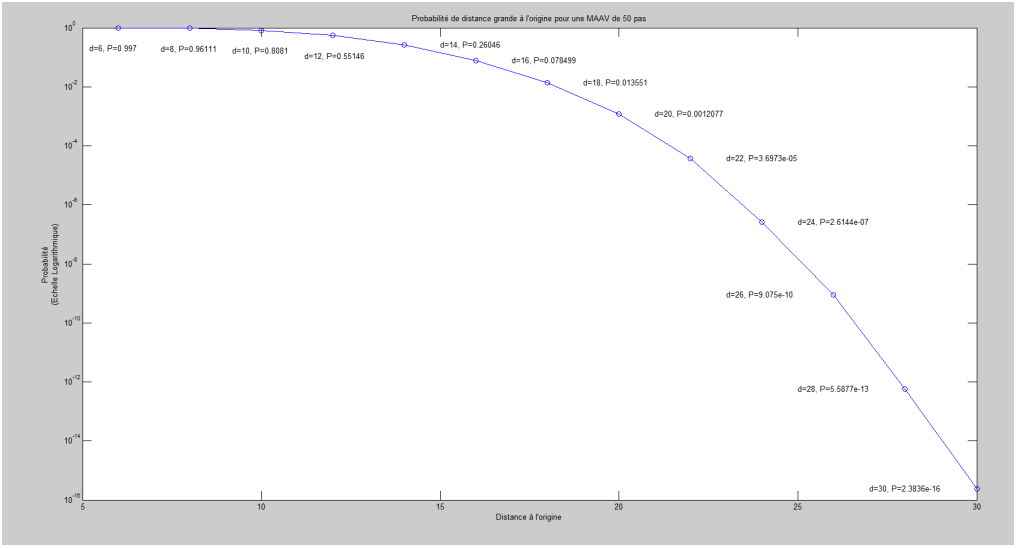


FIGURE 6.2 – $\mathbb{P}(A_d)$ en fonction de $d$ pour des MAAE de 50 pas (échelle semilogarithmique)

Nous voyons qu'à l'aide 1000 particules, nous pouvons estimer des probabilités de l'ordre de $10^{-16}$. La

distance à l'origine maximale moyenne d'une MAAE de 50 pas est d'environ 14. Nous avons réussi à estimer la probabilité qu'une MAAE de 50 pas atteigne une distance de 30 sachant que la distance maximale est de 50 (on suit une unique direction tout le long de la marche) et nous obtenons une probabilité d'environ $2.38 \times 10^{-16}$. Malheureusement, sur la dernière exécution de l'algorithme, aucune particule n'avait réalisé l'événement $A_{30}$ est par conséquent nous ne pouvons en montrer une.

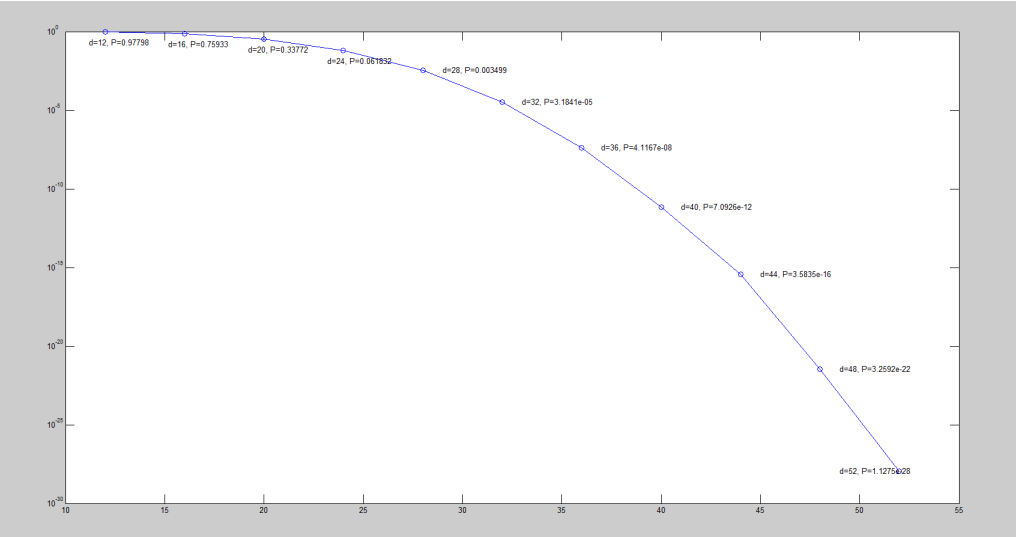


FIGURE 6.3 – $\mathbb{P}(A_d)$ en fonction de $d$ pour des MAAE de 100 pas (échelle semilogarithmique)

Avec l'aide de l'algorithme, nous voulions calculer la probabilité qu'une MAAE de 100 pas atteigne une distance de $12 + 4k$, $0 \leq k \leq 12$, sachant que la distance maximale moyenne d'une MAAE de 100 pas est de l'ordre de 24 pas. Cependant, un ensemble de départ de 1000 particules nous a permis de calculer ces probabilités jusqu'à une distance de 52 maximum avec une probabilité d'atteindre cette distance d'environ $1.13 \times 10^{-28}$. Pour $d = 56$ et 60, l'algorithme nous donne une probabilité de 0, sur les 50 essais aucune particule n'a réussi à atteindre une distance maximale de plus de 56. L'algorithme nous permet quand-même d'estimer des probabilités de l'ordre de $10^{-28}$ ce qui confirme son bon fonctionnement.

## 6.3 Probabilité de Retour à l'Origine

Avec l'algorithme donné précédemment nous avons aussi voulu calculer la probabilité d'un retour à l'origine. Plus précisément, estimer $\mathbb{P}(A_{M,\epsilon})$ avec $A_{M,\epsilon} = \{\exists k_0, d_0(w_{k_0}) \geq M \text{ et } \exists k_1 \geq k_0, d_0(w_{k_1}) \leq \epsilon\}$. Nous avons exécuté cet algorithme sur des MAAE de 100 pas qui ont une distance moyenne maximale d'environ 24. Pour construire la suite $(B_i)$, nous avons fixé $M = 17$ et avons construit une suite décroissante $(\epsilon_i)$, telle que $\epsilon_1 \geq \epsilon_2 \geq ... \geq \epsilon_m = \epsilon$. De même, nous avons pris le résultat moyen sur 50 exécutions de l'algorithme pour avoir un résultat plus représentatif.
Sur la Figure 6.4, nous pouvons voir une marche qui réalise l'événement $A_{M,\epsilon}$ avec $M = 17$ et $\epsilon = 2$. Cette MAAE est d'ailleurs une particule finale obtenue par l'algorithme donné précédemment.

Par ailleurs, nous voyons sur la figure Figure 6.5 que la probabilité d'obtenir une marche aléatoire auto-évitante similaire à celle obtenue en Figure 6.4 est d'environ $\mathbb{P}(A_{17,2}) = 1.105 \times 10^{-13}$. La MAAE présentée est donc tout à fait exceptionnelle. Si on a réussi à en exhiber une c'est bien sûr grâce à la méthode particulaire qui part de MAAE de plus en plus exceptionnelles à chaque étape et ne garde ensuite que celles qui évoluent pour devenir encore plus exceptionnelles.

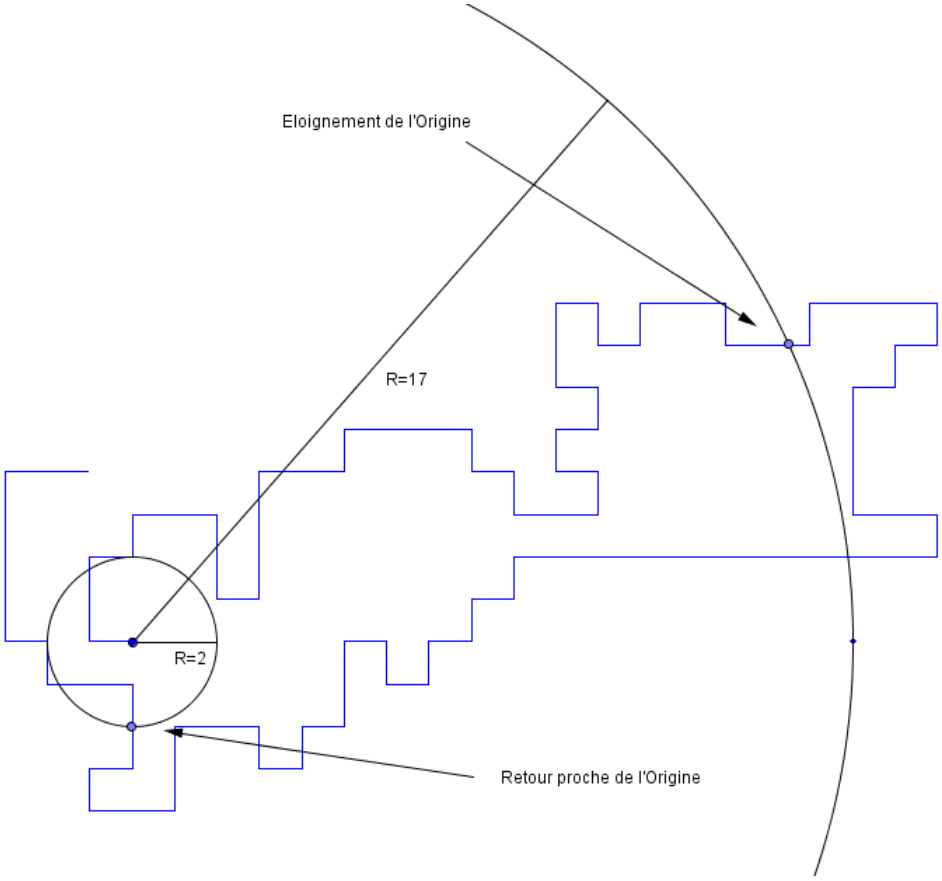


FIGURE 6.4 – MAAE de 100 pas réalisant $A_{17,2}$

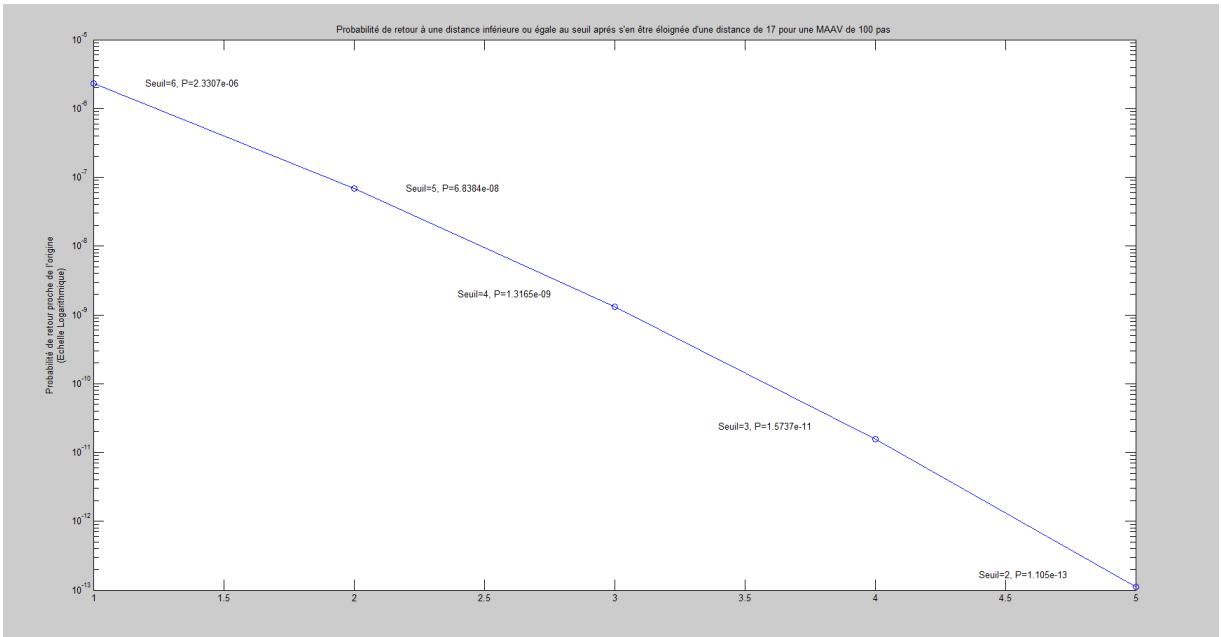


FIGURE 6.5 – $\mathbb{P}(A_{M,\epsilon})$ pour $M = 17$ et $\epsilon \in \mathbb{N}$, $2 \leq \epsilon \leq 6$