Refactoring et optimisation d'un projet de recherche sur les suites de Büchi

Tristan Guillard
tristan.guillard@etu.univ-lyon1.fr

Najib El khadir
najib.el-khadir@etu.univ-lyon1.fr

Tuteur
Fabrice Jaillet

UCBL Lyon 1 - Rapport POM

Résumé

Les chercheurs Pablo SÀEZ, Xavier VIDAUX et Fabrice JAILLET cherchent à caractériser les suites de Büchi entières de longueur 4. Le sujet associe logique mathématique et informatique. Afin de confirmer leurs différentes hypothèses ils ont besoin d'un grand nombre de suites sur lesquelles s'appuyer. Nous nous sommes particulièrement penchés sur la génération de suites de longueur 4 grâce à une méthode matricielle.

Mots clef

Suites de Büchi, Calcul sur grands nombres, Performances de calcul.

1 Introduction

Ce document présente notre travail relatif au POM encadré par M. Fabrice Jaillet sur les suites de Büchi. Il s'agit de proposer des outils aux chercheurs pour les assister dans leurs recherches, notamment en retravaillant les outils qu'ils utilisent pour générer des suites de longueur 4 (méthode matricielle).

En effet les chercheurs cherchent à caractériser toutes les suites de Büchi réelles de longueur 4 - celles de longueur 3 l'étant déjà [SV11] - et de potentiellement trouver la première de longueur 5 - bien que les conjectures laissent à penser qu'il n'en existe aucune.

L'une des parties les plus importantes de notre travail a été de comprendre le sujet de recherche en parcourant la documentation existante ainsi que les programmes du groupe de recherche pour pouvoir produire un code cohérent avec leurs recherches. Bien que les mathématiques soient relativement simples - notamment la définition des suites - il existe toute une variété de méthodes (génération sur polynôme, "brute force" sur les différences des termes, produit de matrices...) de génération de ces suites. Les principales difficultés que l'on rencontre sont l'étendue de l'espace - 4 degrés de liberté dans $\mathbb{N}$ - et la rareté des suites de longueur 4 par rapport aux suites de longueur 3 (en effet, on calcule des suites de longueur 3 pour les étendre a 4, mais ce n'est pas toujours possible, et c'est cela qui pose des problèmes).

Afin de mieux les comprendre, les chercheurs ont besoin de beaucoup de suites. Celles-ci doivent être très grandes afin de pouvoir les analyser et proposer des caractérisations. Plusieurs pistes ont été étudiées et nous nous sommes particulièrement penchés sur la génération de suites par la méthode des matrices.

2 Méthode de calcul des suites par matrice

2.1 Suites de Büchi

Une suite de Büchi $(x_i)_{i\in\mathbb{N}} \in \mathbb{N}^M$ satisfait l'équation suivante

$$\forall i < M-3, \quad x_{i+2}^2 - 2x_{i+1}^2 + x_i^2 = 2$$

On remarque que toute suite de la forme $(x, \pm(x+1), \pm(x+2), ...)$ ou $(x, \pm(x-1), \pm(x-2), ...)$ respecte cette équation (peu importe la longueur de la suite). On nomme ces suites les suites TRIVIALES. Elles seront utiles pour la méthode des matrices, mais ce ne sont pas celles que nous cherchons à caractériser ou à calculer. Les suites de longueur 4 ont par ailleurs déjà été caractérisées dans $\mathbb{Q}$.

2.2 Description de la méthode

La force de la méthode matricielle est de pouvoir aller chercher des suites très grandes (par leurs valeurs), là où les méthodes actuellement utilisées génèrent les suites de manière croissante selon $x_4 - x_3$. Bien que l'ordre de génération est plus flou avec cette méthode, on a conjecturé que la taille des suites est déterminable avant le calcul (cf figure 5).

2.2.1 Produit des matrices

Il a été montré dans [SV11] qu'il est possible de générer à souhait des suites de Büchi de longueur 3 grâce au calcul suivant

$$B^{n_1} J ... J B^{n_k} J^b \vec{u}$$

où les n_i sont des entiers non nuls, b est égal a zéro ou un et $\vec{u}$ est une suite de Büchi triviale, à savoir

$$\begin{pmatrix} x \\ \pm(x+1) \\ \pm(x+2) \end{pmatrix} ou \begin{pmatrix} x \\ \pm(x-1) \\ \pm(x-2) \end{pmatrix}$$

avec

$$J = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} et\ B = \begin{pmatrix} 3 & 4 & 0 \\ 2 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Toute suite de puissance $(n_i)_{i\in\mathbb{N}}$ (aussi appelée "mot") correspond à une unique suite de Büchi. Cependant les suites telles que $(-6, 23, 32)$ et $(6, 23, 32)$ sont équivalentes pour la recherche d'une suite de longueur 4, car on considère les carrés de ces nombres. On parle d'équivalence de Büchi - avec la classe d'équivalence qui correspond. On retire alors la liberté de signe sur les deux dernières valeurs du mot pour éviter la génération de doublons dans cette classe d'équivalence.

Toute suite triviale peut être trouvée grâce à la suite triviale $(0, 1, 2)$ avec des puissances de valeur 1 ou -1. Appliquer le mot $(1, -1, 1, -1, 2, 2)$ avec $\vec{u} = (0, 1, 2)$ revient donc à appliquer $(2, 2)$ à $\vec{u} = X$ où X est le résultat trivial de $(1, -1, 1, -1)$ appliqué à $\vec{u} = (0, 1, 2)$. On nommera toute triviale prise pour le calcul une "orbite" et $(0, 1, 2)$ l'orbite triviale.

Une orbite est donc un point de départ sur lequel se base le calcul pour générer des suites de Büchi réelles en fonction du mot. Les espaces générés par les triviales sont disjoints, sauf pour $(0, 1, 2)$ qui peut générer toutes les autres triviales et suites de Büchi, mais avec des mots plus longs.

De plus il est possible de retrouver le mot d'une suite de Büchi ainsi que l'orbite sur laquelle elle s'appuie. On peut toujours retourner à l'orbite triviale mais cela peut induire des mots de taille très grande, on se limite donc à retrouver l'orbite primitive de cette suite.

2.2.2 Extension d'une suite

La méthode précédente nous permet de générer une infinité de suites de longueur 3, cependant nous cherchons des suites de longueur 4, il faut donc étendre les suites trouvées, si elles peuvent l'être. Sachant qu'une suite de Büchi $(x_i)_{i\in\mathbb{N}} \in \mathbb{N}^M$ satisfait l'équation suivante

$$\forall i < M-3, \quad x_{i+2}^2 - 2x_{i+1}^2 + x_i^2 = 2$$

On calcule donc x_4 et x_0 ainsi

$$x_4 = \sqrt{(2x_3^2 - x_2^2 + 2)},\ x_0 = \sqrt{(2x_1^2 - x_2^2 + 2)},$$

Il suffit alors que l'un des deux soit un entier pour avoir une suite de Büchi de longueur 4. On peut regarder l'extension d'une suite de longueur 4 de la même manière pour s'assurer que la suite ne s'étend pas à 5.

2.2.3 Ordres de grandeur

Les suites de Büchi deviennent vite très grandes avec cette méthode. En effet, les valeurs des suites calculées à partir de mots de taille N sont d'ordre de grandeur 10^N, on conjecture expérimentalement $10^{1.2N}$ (cf figure 5). L'équipe de recherche a déjà calculé toutes les suites où la différence entre x_3 et x_4 est inférieure a 1500 milliards par un parcours en largeur (génération différente de la méthode des matrices). La méthode des matrices permet d'aller chercher des suites aux valeurs bien plus grandes, mais on perd l'ordre de parcours des suites par différence des derniers termes.

2.3 Implémentation

Cette partie du projet a eu pour but d'accélérer au maximum la recherche de suites de longueur 4 parmi celles que trouvent la méthode des matrices. Le programme initial fonctionnait ainsi :

— On part d'un certain mot, pour lequel on calcule $B^{n_1}J...B^{n_k}J^bu$.
— On cherche à étendre la suite de longueur 3 trouvée.
— On passe au mot suivant (technique décrite dans 2.3.3) et on boucle.

Cette sous-section décrit les optimisations mises en place pour accélérer ces calculs. Deux méthodes d'exploration des suites par leur génération matricielle sont présentées ici.

2.3.1 Bibliothèques

Le projet utilise la bibliothèque `boost::multiprecision::cpp_int` et le template `boost::numeric::ublas::matrix` ainsi que la std. La structure de données "cpp_int" permet de gérer des nombres entiers arbitrairement grands sans perte d'exactitude, ce qui est indispensable pour nos vérifications d'extension de suites - une erreur de 1 amènerait des faux positifs. La Lib-Pari originellement considérée dans le projet a été mise de côté compte tenu de sa complexité d'utilisation et du fait qu'elle n'apportait pas grand chose de plus que la librairie boost de C++.

De plus OpenMP a été utilisé pour paralléliser le code, afin de gagner en performance malgré quelques ajustements nécessaires, notamment dûs à la grande récursivité des méthodes de parcours des mots.

2.3.2 Pré-calcul des puissances de B

Chaque élément d'un mot va nécessiter le calcul de différentes puissance de B. Pour économiser beaucoup de calculs, les puissances de B sont pré-calculées et enregistrées dans une Map. Cela permet un accès très rapide aux puissances nécessaires au calcul de la suite. Si une puissance plus grande que celles contenues dans la Map a besoin d'être calculée, on la calcule récursivement et on ajoute toutes les nouvelles puissances calculées à la Map.

2.3.3 Méthode parcours des mots

La méthode la plus utilisée dans ce projet consiste en un parcours des puissances dans un ordre pseudo lexicographique

$$(1) >> (2) >> (1,1) >> (3) >> (1,2) >> (2,1) >> (1,1,1) >> (4) >> (1,3) >> (2,2) >>$$

$$(3,1) >> (1,1,2) >> (1,2,1) >> (2,1,1) >> (1,1,1,1) >> (5) >> (1,4) >> (2,3) >> (3,2)$$

en y appliquant la bijection

$$f(n) = \begin{cases} -n/2 & \texttt{si n pair} \\ (n+1)/2 & \texttt{si n impair} \end{cases}$$

afin de parcourir les mots. On n'applique cependant pas cette bijection aux deux derniers termes du mot pour ne pas avoir de doublon (cf 2.2.1). On obtient ainsi des mots de cette forme

$$(1,1,3) >> (1,2,2) >> (1,3,1) >> (-1,1,2) >> (-1,2,1) >> (2,1,1) >>$$

$$(1,1,1,2) >> (1,1,2,1) >> (1,-1,1,1)$$

Cette méthode est très pratique car elle permet d'avoir un aperçu très large des suites générées juste après certains mots. Elle permet entre autres le calcul en place dans l'orbite, ce qui résulte en un calcul de vecteur et non plus un calcul de matrice (gain de performance). Cependant elle commence a devenir assez lente sur les mots très longs, car toute l'équation $B^{n_1}J...JB^{n_k}J^b\vec{u}$ est recalculée.

2.3.4 Méthode extension droite / parcours d'arbre

Variante de la première méthode, celle-ci permet un parcours récursif des mots à la manière d'un parcours en profondeur d'un arbre. Il s'agira de parcourir les puissances de cette manière (profondeur 4, largeur 2 (de -2 a 2)).

$$(-2) >> (-2,-2) >> (-2,-2,-2) >> (-2,-2,-2,-2) >> (-2,-2,-2,-1) >>$$

$$(-2,-2,-2,1) >> (-2,-2,-2,2) >> (-2,-2,-1) >> (-2,-2,-1,-2) >> (-2,-2,-1,-1)$$

Cette méthode permet de parcourir les alentours de mots très longs sans avoir à les recalculer uniquement pour y ajouter une valeur à droite. En effet, en parcourant l'arbre récursivement, on peut utiliser le mot déjà calculé jusque là sans avoir à refaire tout le calcul. Prennons par exemple le mot $(w + \{-2,-2\})$. Peu importe la taille du mot w, le calcul de la suite ne prendra qu'une multiplication de matrice, car $(w + \{-2\})$ a déjà été calculée et vérifiée. Pouvoir explorer de très grands mots nous permet donc d'explorer des suites extrêmement grandes en ce qui concerne leur valeur (jusqu'à 10^{1000}).

L'analyse de complexité est bien plus facile ici : 2^{2WH} avec W la largeur de la recherche (par largeur on entend les puissances de B explorées, ici W = 2, on regarde donc les puissances de B de -2 à 2 sans passer par 0) et H la hauteur (profondeur) des fins de mot à ajouter. Tout comme la première méthode, le temps de calcul des suites est très négligeable, mais le nombre de suites à explorer est immense, cette méthode ne peut donc servir que d'exploration locale à partir d'un mot de n'importe quelle longueur.

2.3.5 Limites du projet

— Pas de parallélisation GPU avec `boost::multiprecision::cpp_int` car on perd trop de temps à cause de l'envoi des données au GPU. De plus les puissances de B étant pré-calculées, il reste peu de calculs à paralléliser. Le programme n'est finalement qu'une grande disjonction de cas et d'appels de fonctions : on parle ici de milliards de séries dont le calcul ne prend pas énormément de temps - totalement négligeable selon un profiling.

— La méthode de génération des mots est un parcours large mais finit par générer de hautes puissances dans les mots. Cependant, de toutes les suites calculées par le groupe de recherche (générées différemment), aucune n'a de puissance supérieure à 7 dans son mot.
— Le nombre de suites de longueur 4 est absolument infime par rapport au nombre de suites de longueur 3. En effet lorsque les mots s'allongent, les chances de trouver des suites extensibles diminuent très fortement. Nous avons fait une extension de mot de profondeur 10 et de largeur 3 sur des milliards de mots aléatoires (composés d'entiers non nuls entre -3 et 3 inclus) de longueur 1000 et nous n'avons trouvé aucune suite de longueur 4, et seulement quelques unes sur des millions de mots de longueur 100.
— Bien que le programme gère les très grands nombres, les valeurs supérieures a 10^{100000} posent des problèmes de performance. Cependant les suites que les chercheurs ont pu atteindre jusque-là se trouvent dans les 10^{19}, nous avons donc bien élargi leur espace de recherche.

2.4 Résultats

2.4.1 Méthode de génération des mots

La méthode de génération des suites selon le calcul des matrices a été accélérée d'un facteur 3000 (estimation basse) pour sa variante "parcours des mots". Nous avons pu rechercher des suites de longueur 4 parmi 10 milliard de suites de longueur 3 générées par cette méthode. La figure 1 montre la vitesse de notre programme en comparaison de celui des chercheurs.

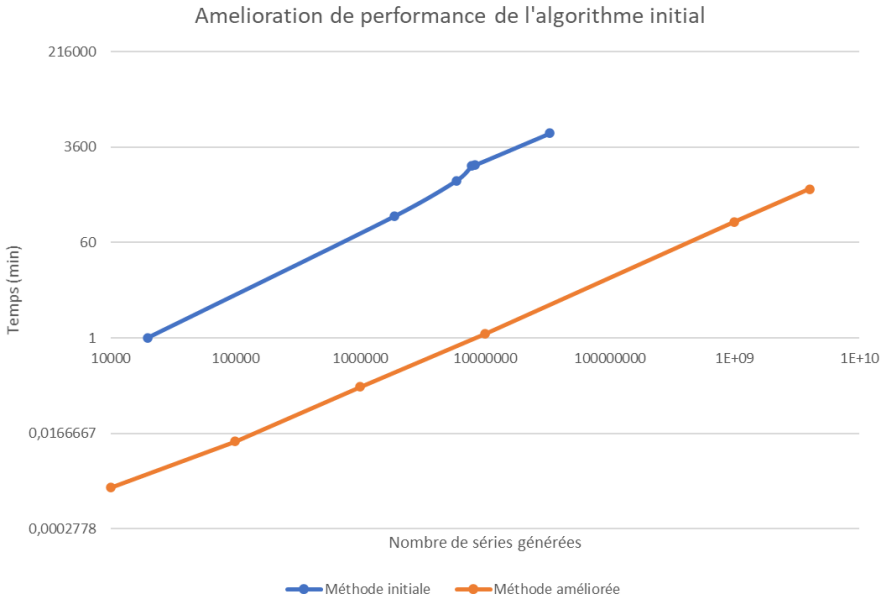

FIGURE 1 – Comparaison des temps entre le programme des chercheurs et le nôtre (échelles logarithmiques)

2.4.2 Méthode d'extension des mots

La recherche par extension droite des mots n'a rien trouvé de nouveau pour les mots très courts, et rien du tout dans les mots longs, la probabilité de tomber sur une suite extensible est bien trop faible pour les suites de grandes valeurs. Il n'y a pas de comparaison de performance car il n'y avait pas de code pour les générer mais cette méthode est tout aussi rapide que la précédente et même plus rapide sur les longs mots par sa nature récursive qui lui évite beaucoup de calcul. En outre, les

résultats de cette méthode soulignent le fait que l'espace est très vide : aucune suite de longueur 3 extensible parmi des dizaines de milliards testées (ordre de grandeur des suites testées : 10^{1200}). Aucune recherche exhaustive n'est envisageable, seulement des études locales. Les figures 2, 3, 4 caractérisent les variations des temps de calcul en fonction des différents paramètres et la figure 5 l'ordre de grandeur des suites par rapport à la taille des mots.

FIGURE 2 – Temps de calcul en fonction du nombre de mots

FIGURE 3 – Temps de calcul en fonction de la longueur des mots

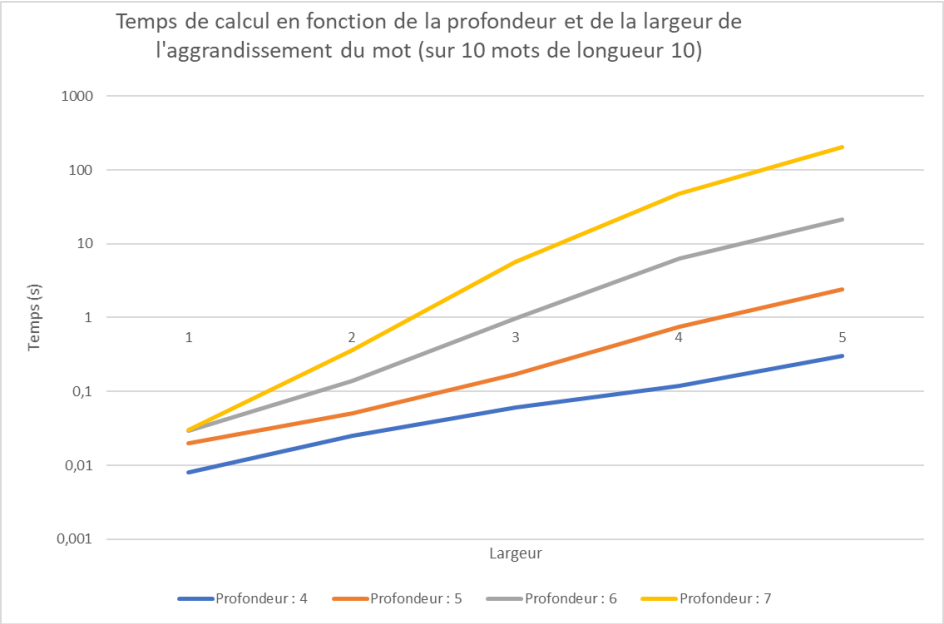

FIGURE 4 – Temps de calcul en fonction de la profondeur et de la largeur de l'extension des mots

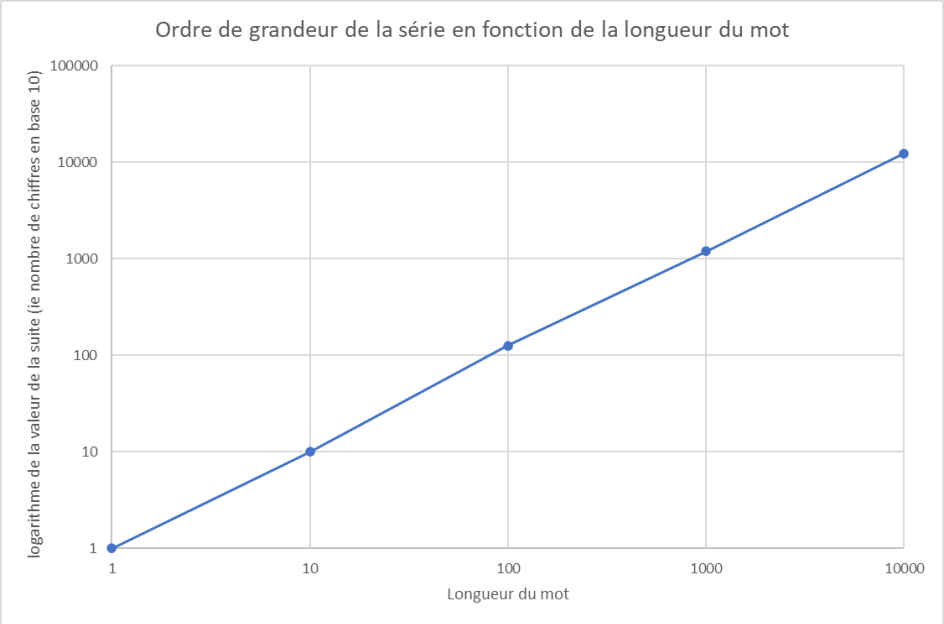

FIGURE 5 – Ordre de grandeur de la longueur de la suite en fonction de la longueur du mot

3 Utilisation possible des données générées

3.1 Méthode RANSAC

Les données générées grâce aux calculs matriciels sont prêtes pour être traitées et utilisées dans le cadre de méthodes plus complexes permettant la génération de suites de Büchi de plus grande taille.

La piste envisagée pour la suite du projet est l'utilisation de la méthode RANSAC afin de tirer profit des résultats des calculs des suites par matrices.

3.1.1 Algorithme

La méthode RANSAC fonctionne comme suit :

Algorithm 1 RANSAC pseudo-code

```
for un certain nombre d'itérations do
    sélection d'un set de données aléatoire parmi les données possibles (Random Sampling)
    construction du modèle correspondant au problème
    for chacune des données do
        application du modèle à la donnée
        calcul de la distance par rapport à l'objectif
        if distance ≤ seuil souhaité then
            la donnée est plausible
            on l'ajoute aux inliners
        end if
    end for
    if le nouveau modèle est meilleur que le modèle courant then
        on met à jour le modèle
    end if
end for
```

3.1.2 Paramètres

- e : probabilité qu'une donnée soit aberrante (très grande dans notre cas puisque nous traitons possiblement avec des centaines de milliers de suites)
- s : nombre de données dans un échantillon (fixé arbitrairement)
- p : probabilité souhaitée que l'échantillon soit bon (haute de préférence, entraînant un plus grand temps de calcul par conséquent)
- N : nombre d'itérations (équivalent au nombre d'échantillons testés)
  Ce dernier paramètre est celui qui nous intéresse le plus. Pour avoir au moins un échantillon plausible, nous choisissons N de sorte que p soit ≥ 0.99.

Nous savons qu'il y a des outliners (beaucoup...) dans nos sets de données, il y aura donc des sets aberrants mais nous espérons qu'ils ne le soient pas tous. Il y en a donc au moins un plausible, c'est ainsi que p est déterminé

$$p = 1 - \big(1 - (1 - e)^s\big)^N$$

Ainsi, nous pouvons déterminer

$$N = \frac{log(1 - p)}{log\big(1 - (1 - e)^s\big)}$$

3.1.3 Avantages de la méthode RANSAC

La méthode des moindres carrés, à titre de comparaison, est très sensible au bruit, quelques données aberrantes y suffisent pour compromettre la fiabilité des résultats.

La méthode RANSAC a comme avantage d'ignorer facilement le bruit est de privilégier la satisfaisabilité du maximum de points.

3.1.4 Utilisation envisagée dans le cadre du projet

Dans le cadre de la recherche de séries de Büchi plus complexes, la méthode RANSAC peut être utilisée de différentes manières, la plus notable étant celle de la recherche d'un endomorphisme passant par un maximum de points qui représentent une caractérisation des séries déjà trouvées et listées. À partir du polynôme calculé, les chercheurs seraient capables de traiter les données et d'en tirer des nouvelles séries.

3.1.5 Implémentation actuelle

Pour l'instant la méthode RANSAC testée ne fonctionne que sur un nombre de séries et une taille d'échantillons limités, elle permet de calculer une forme quadratique passant par entre 3 et 5 caractérisations UV.

La caractérisation UV d'une suite de Buchi est : $U = (-x_3 + 2x_2 - x_1)/2, V = x_3 - x_2$.

Cette forme quadratique est différente à chaque exécution de l'algorithme puisque les résultats varient en fonction des échantillons aléatoires testés.

Par faute de temps, il a été difficile d'aller au-delà, mais l'exploration de cette voie reste envisageable pour la postérité du projet.

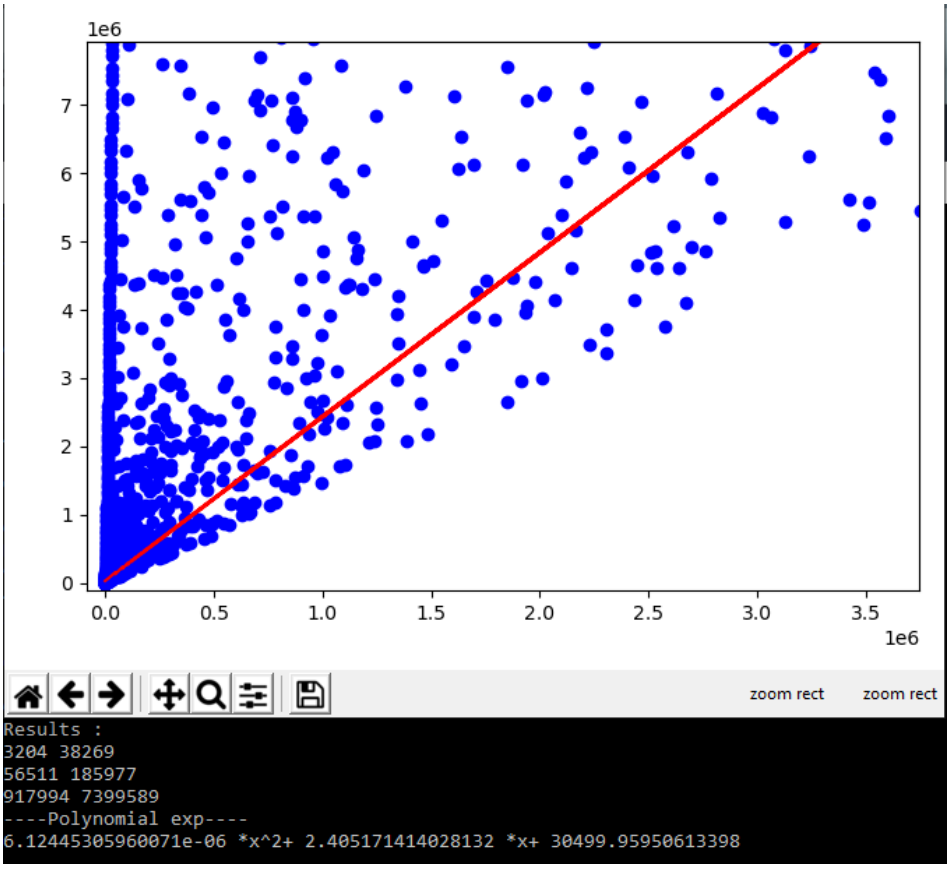

FIGURE 6 – Exemple de forme quadratique trouvée grâce à la méthode RANSAC appliquée aux 1000 premières paramétrisations UV

4 Conclusion

Nous avons développé plusieurs méthodes et outils afin de générer des suites de Büchi de longueur 3 pour pouvoir les étendre à la longueur 4. Ces programmes pourront bénéficier aux chercheurs selon leurs besoins, et nous ont permis de générer un grand nombre de suites. Il était prévu que nous étudiions les suites que nous avons générées grâce a différents algorithmes, notamment la méthode RANSAC, mais nos plans pour ce projets ont du être revus à la baisse compte tenu du contexte actuel. Nous avons tout de même réussi à améliorer les programmes existants d'un facteur 3000 ce qui était notre priorité.

Nous en avons beaucoup appris sur les suites de Büchi de longueur 4, notamment leur rareté par rapport à celles de longueur 3, ce qui est dommage étant donné le fait que nos programmes permettent d'aller chercher des suites extrêmement grandes (ce qui était un des axes principaux du projet). Nous avons donc généré des milliards de suites de longueur 3 très grandes sans jamais trouver de suite de longueur 4, ce qui aurait beaucoup aidé les chercheurs.

Par manque de temps de calcul et d'intérêt, nous n'avons généré que quelques dizaines de milliards de suites de longueur 3, ce qui nous a permis de calculer quelques milliers de suites de longueur 4. Ce n'est pas beaucoup comparé aux 900000 suites déjà calculées par les chercheurs, mais cela n'a pris qu'une vingtaine d'heures de calcul sur un processeur a 8 coeurs, alors que leurs programmes ont mis des mois de calculs sur de grosses machines.

Enfin, il aurait été intéressant de développer de nouvelles méthodes de parcours des mots pour espérer trouver plus de suites de longueur 4, et de chercher des rapports entre les suites trouvées et les mots dont elles sont issues.

Références

[SV11] Pablo Sáez, Xavier Vidaux : *A characterization of Büchi's integer sequences of length 3.* Acta Arithmetica 149, 2011.

[HNM13] Anders Hast, Johan Nysj, Andrea Marchetti : *Optimal RANSAC – Towards a Repeatable Algorithm for Finding the Optimal Set.* Journal of WSCG. 21. 2013