Le Cryptogramme de D'Agapeyeff (1939)
Anatomie, Reconstruction d'un Solveur et Bilan Statistique

Ruben Gariazzo

étudiant à l'école polytechnique féminine

RÉSUMÉ

En 1939, le cartographe et cryptologue amateur Alexander D'Agapeyeff publiait, à la fin de son manuel *Codes and Ciphers*, un cryptogramme de 196 symboles présenté comme un défi pour ses lecteurs. Quatre-vingt-sept ans plus tard, il reste non résolu, et son auteur aurait lui-même admis avoir oublié la méthode utilisée pour le construire. Ce rapport documente une tentative de cryptanalyse computationnelle menée par itérations successives : les difficultés méthodologiques rencontrées, leur diagnostic et leur correction, ainsi que les résultats obtenus une fois ces corrections appliquées.

La première difficulté concernait non le chiffre, mais le solveur destiné à le résoudre. Une première version jugeait un résultat prometteur au-delà d'un seuil de score fixé à l'avance et jamais vérifié empiriquement. Après trois jours de calcul continu, aucun résultat ne l'avait franchi. Un texte anglais authentique, noté *a posteriori* sous ce même modèle, n'atteignait pas non plus ce seuil, structurellement inaccessible : conséquence d'un modèle de langage [1] entraîné sur un corpus de seulement 8 000 caractères, insuffisant pour distinguer statistiquement l'anglais réel du charabia. Ce diagnostic a motivé une reconstruction complète du solveur. Le principe retenu pour la suite du projet est de vérifier empiriquement tout critère de succès avant de l'utiliser, plutôt que de le supposer.

Le solveur reconstruit teste conjointement quatre familles de chiffrement (substitution, transposition simple, grille de Fleissner [2] et double transposition) sur neuf hypothèses concurrentes de transcription du texte imprimé et trois langues candidates (anglais, français, hébreu translittéré), au moyen d'un recuit simulé [3] dont l'effort de calcul est réparti par un bandit adaptatif [4] entre les combinaisons structurelles possibles. Un score élevé ne suffit toutefois pas à conclure. Dès lors que des centaines de milliers de combinaisons sont testées indépendamment, le meilleur score observé dérive mécaniquement vers le haut même sur du bruit pur, par un simple effet de taille d'échantillon. La calibration par ligne de base nulle [5], méthodologie centrale du projet, corrige ce biais. Appliquée à l'hypothèse hébraïque, elle a montré qu'un score d'abord jugé prometteur n'était qu'un artefact du plus petit alphabet translittéré utilisé,

1. Modèle de langage quadrigramme : mesure de la plausibilité d'un texte fondée sur la fréquence, dans un grand corpus de référence, de chacune de ses fenêtres de quatre lettres consécutives (un quadrigramme). Plus la fenêtre est large, plus il devient statistiquement difficile pour une séquence sans queue ni tête d'obtenir un score élevé par pur hasard.

2. Grille de Fleissner : gabarit perforé que l'on fait pivoter sur un texte pour en réordonner les lettres par transposition, l'un des procédés historiques les plus anciens pour brouiller un message sans en changer les lettres.

3. Recuit simulé : méthode d'optimisation qui explore l'espace des clés possibles en acceptant parfois, avec une probabilité décroissant au fil de la recherche, une solution momentanément moins bonne, pour éviter de rester bloqué dans un optimum local.

4. Bandit adaptatif : algorithme qui concentre progressivement l'échantillonnage sur les options les plus prometteuses parmi plusieurs candidates concurrentes, sans jamais en exclure totalement aucune.

5. Calibration par ligne de base nulle : comparaison du meilleur résultat obtenu, non à un seuil fixe, mais au meilleur résultat que le même pipeline obtient, au même volume d'essais, sur une version du cryptogramme mélangée aléatoirement, donc par construction dépourvue de tout signal.

une fois comparé à sa propre ligne de base nulle plutôt qu'à une référence isolée.

Cette même exigence de vérification s'est appliquée aux outils de recherche eux-mêmes. Une accélération de la recherche (compilation intégrale de la boucle de recuit simulé, couverture systématique garantie de l'espace de recherche plutôt que probabiliste) a révélé un goulot d'étranglement imprévu dans la journalisation des résultats. Sa première correction candidate, mesurée avant déploiement plutôt que présumée sûre, s'est avérée presque aussi coûteuse que l'absence de correction et a dû être remplacée. Un second critère de détection, fondé sur la couverture en mots reconnaissables du texte décodé, a lui aussi été retardé par un dictionnaire de référence contaminé par du jargon technique, détecté sur du bruit pur avant tout déploiement en production.

Sur les quatre méthodes de chiffrement testées, couvertes de façon exhaustive et garantie à deux reprises indépendantes, aucun signal ne se distingue statistiquement du bruit sur aucun des trois critères mesurés : le score du modèle de langage, la couverture en mots reconnaissables et la longueur du plus long fragment cohérent identifié. La calibration nulle a fini par dépasser le meilleur score obtenu sur le cryptogramme réel. La suite de mots anglais la plus longue et la plus grammaticalement cohérente trouvée sur l'ensemble du projet, « held if we can win », provient d'ailleurs d'une séquence mélangée aléatoirement et non du cryptogramme lui-même, démonstration directe qu'une telle cohérence peut apparaître par pur hasard à ce volume de recherche. Une dernière analyse, portant sur le regroupement structurel des meilleurs scores plutôt que sur un résultat isolé, a mis en évidence une asymétrie entre hypothèses de transcription ; celle-ci s'est révélée, après vérification, produite par le mécanisme d'échantillonnage adaptatif lui-même plutôt que par le texte sous-jacent. Ce fil d'analyse a été refermé par insuffisance de données.

Cette couverture exhaustive de l'espace initial, sans signal détecté, a orienté l'effort vers une famille de chiffrement absente jusqu'ici de l'espace modélisé, le chiffrement Four-square, qui substitue des paires de lettres au moyen de quatre grilles de Polybius plutôt que des lettres isolées. Sa mécanique a été validée indépendamment sur des milliers d'essais de chiffrement et déchiffrement réciproques. La validation de la recherche elle-même a révélé une difficulté supplémentaire. Le recuit simulé, pourtant efficace sur les quatre méthodes précédentes, échouait systématiquement à retrouver une clé connue avec un budget de calcul identique, sur un paysage de score nettement plus accidenté. Un réglage dédié, fondé sur des relances indépendantes multiples plutôt que sur un budget de calcul simplement plus généreux, a permis d'atteindre une fiabilité de récupération de 83 % mesurée sur des essais indépendants. Cette recherche est désormais active en production, sur un espace encore trop peu couvert pour en tirer une conclusion.

Nomenclature

N	Longueur du cryptogramme en symboles ($N = 196$)
w, h	Largeur et hauteur d'une grille de transposition ($w \times h = N$)
IC	Indice de coïncidence d'une séquence de symboles
$Q(k)$	Score de log-vraisemblance quadrigramme d'un texte décodé avec la clé k
f_{abcd}	Fréquence lissée du quadrigramme *abcd* dans le corpus de référence
$w_i,\ b_i,\ n_i$	Poids d'échantillonnage, meilleur score, nombre d'essais du bras i (bandit)
λ_i	Nombre moyen d'essais attendus sur le flux i (approximation de Poisson)
$\mathscr{S}$	Score de référence self-test (calibration à essai unique)

1. Introduction

Le cryptogramme de D'Agapeyeff se distingue des chiffres habituellement étudiés en cryptanalyse historique par son statut de défi publié plutôt que de message intercepté. Contrairement à un message dont on sait qu'il a été échangé dans un but précis, celui-ci a été publié volontairement par son auteur, sans certitude qu'il en ait lui-même conservé la solution. Alexander D'Agapeyeff, cartographe et officier de la Royal Air Force d'origine russe naturalisé britannique, l'a fait imprimer en 1939 à la fin de la première édition de son manuel *Codes and Ciphers*, sous la forme de 392 chiffres regroupés par cinq. Il fut retiré des éditions suivantes ; son auteur aurait par la suite admis ne plus se souvenir de la méthode utilisée pour le construire. Cette anecdote est retenue ici comme une hypothèse de travail à part entière, plutôt que comme une simple curiosité biographique : une absence de solution retrouvée ne constituerait une preuve ni de l'insuffisance de la recherche, ni de l'absence d'un texte clair sous-jacent.

Cette incertitude de départ détermine la démarche adoptée dans ce rapport. Le projet traite la cryptanalyse du cryptogramme comme un problème de test d'hypothèses statistique plutôt que comme une recherche de clé par intuition. Cela suppose trois principes appliqués systématiquement : modéliser explicitement l'espace des méthodes de chiffrement et des variantes de transcription jugées plausibles, explorer cet espace de façon systématique et garantie plutôt qu'au hasard, et comparer tout score obtenu, non à un seuil fixé a priori, mais au meilleur résultat qu'obtiendrait, au même volume d'essais, un texte dépourvu par construction de tout signal. Ce dernier point est une exigence centrale, pas une précaution accessoire. Avec des centaines de milliers puis des millions de combinaisons testées indépendamment, le meilleur score observé sur du pur hasard dérive lui aussi vers le haut, et rien ne le distingue plus d'un signal réel sans ce point de comparaison. Chaque extension de l'espace d'hypothèses, qu'il s'agisse d'une méthode de chiffrement supplémentaire ou d'une langue candidate, est motivée par une preuve ou un indice concret, non par une exploration arbitraire.

Ce document a la forme d'un journal de recherche tenu à jour au fil du projet, plutôt que celle d'un compte rendu rédigé après coup pour une méthode arrêtée à l'avance. Il documente les impasses méthodologiques rencontrées, leur diagnostic et leur correction, au même titre que les résultats positifs ou

négatifs obtenus une fois ces corrections en place. La section 4 présente le défaut de conception qui a motivé une reconstruction complète du solveur initial. La section 5 détaille l'architecture qui en résulte. Les sections suivantes retracent, dans leur ordre chronologique réel, chaque correction, chaque nouvel axe de détection statistique et chaque verdict obtenu, jusqu'à l'état d'avancement actuel de la recherche.

2. Contexte et Anatomie du Chiffre de D'Agapeyeff

2.1 Un cartographe, un livre, une énigme

Alexander D'Agapeyeff (1902-1955), cartographe et cryptologue amateur d'origine russe naturalisé britannique, publie en 1939 chez Oxford University Press *Codes and Ciphers*, un ouvrage introductif de cryptographie commandé par son éditeur à l'approche de la guerre. La première édition se termine par un cryptogramme destiné à tester le lecteur. Il fut retiré des éditions ultérieures, D'Agapeyeff ayant lui-même admis ne plus se souvenir de la méthode employée pour le construire (anecdote qui, comme nous le verrons en section 15, n'est peut-être pas qu'une boutade).

2.2 Le texte intégral

Le cryptogramme est imprimé sous forme de 392 chiffres, groupés par cinq pour la lisibilité télégraphique (convention typographique standard de l'époque) :

Pièce 1 : Le cryptogramme tel qu'imprimé (1939)

```
75628 28591 62916 48164 91748 58464 74748 28483 81638 18174
74826 26475 83828 49175 74658 37575 75936 36565 81638 17585
75756 46282 92857 46382 75748 38165 81848 56485 64858 56382
72628 36281 81728 16463 75828 16483 63828 58163 63630 47481
91918 46385 84656 48565 62946 26285 91859 17491 72756 46575
71658 36264 74818 28462 82649 18193 65626 48484 91838 57491
81657 27483 83858 28364 62726 26562 83759 27263 82827 27283
82858 47582 81837 28462 82837 58164 75748 58162 92000
```

2.3 Une grille de Polybius sous-jacente

La structure des chiffres est cohérente avec un carré de Polybius [1] 5×5 lu en paires de chiffres : le premier chiffre de chaque paire, pris dans $\{6,7,8,9,0\}$, code la ligne ; le second, pris dans $\{1,2,3,4,5\}$, code la colonne, convention télégraphique standard de l'entre-deux-guerres où « 0 » prolonge cycliquement la série après « 9 ». La Figure 1 illustre cette lecture sur la toute première paire du texte.

1. Carré de Polybius : grille 5×5 attribuant à chaque lettre de l'alphabet une coordonnée (ligne, colonne) ; brique de base de nombreux chiffres historiques (ADFGVX, Bifide, nihiliste...).

	colonne				
	1	2	3	4	5
6	A	B	C	D	E
7	F	G	H	I/J	K
8	L	M	N	O	P
9	Q	R	S	T	U
0	V	W	X	Y	Z

Grille d'illustration (remplissage alphabétique arbitraire, à but pédagogique). La véritable clé de substitution (quelle lettre occupe quelle case) est précisément ce que le solveur recherche ; elle n'est jamais supposée.

Exemple : la paire « 75 » → ligne 7, colonne 5 → case (7,5) → symbole 9 (indexé 0-24) → lettre **K** *dans cette grille d'exemple seulement*.

FIGURE 1 – *Décodage d'une paire de chiffres en coordonnée de grille Polybius.* La fusion I/J est la convention standard pour tenir 26 lettres dans une grille de 25 cases.

2.4 Pourquoi la fin du texte a été retranchée

Le dernier groupe imprimé, « 92000 », pose une question directe : les 196 paires du cryptogramme s'arrêtent-elles à « 92 », ou le texte comporte-t-il 197,5 paires supplémentaires cachées dans « 000 » ? Trois arguments convergents, illustrés Figure 2, tranchent en faveur d'un simple remplissage typographique :

1. **Alignement des groupes.** 78 groupes complets de 5 chiffres précèdent le dernier = 390 chiffres = 195 paires exactes. Le 79^{e} groupe, « 92000 », fournit avec ses deux premiers chiffres (« 92 ») exactement la 196^{e} et dernière paire. Les trois chiffres restants tombent structurellement hors de toute paire.
2. **Parité.** 196 paires = 392 chiffres, un nombre pair. Inclure « 000 » porterait le total à 395 chiffres : un nombre impair, incompatible avec un découpage en paires quel qu'il soit.
3. **Typage ligne/colonne.** Sur l'ensemble des 196 paires retenues, aucun « 0 » n'apparaît jamais en position colonne (qui n'accepte que $\{1,2,3,4,5\}$). Si l'on tentait de lire « 000 » comme des paires supplémentaires, la paire « 00 » violerait immédiatement cette règle, jamais enfreinte ailleurs dans le texte.

FIGURE 2 – *Décomposition du dernier groupe imprimé « 92000 ».* Les deux premiers chiffres complètent la 196^{e} paire ; les trois derniers n'appartiennent à aucune paire et sont exclus du cryptogramme numérique analysé.

Le `CIPHER_RAW` utilisé par le solveur s'arrête donc à 392 chiffres. Il aurait été aussi facile de se tromper dans l'autre sens : inclure aveuglément « 000 » sans vérifier sa cohérence avec le reste du texte : c'est précisément ce type de décision de transcription, prise sans preuve à l'appui, que ce projet s'efforce de documenter et de justifier explicitement plutôt que de supposer.

3. Hypothèses de Travail

Contrairement à une démonstration mathématique fermée, ce projet ne repose pas sur quatre hypothèses figées mais sur un ensemble d'hypothèses *concurrentes*, regroupées en quatre familles (Figure 3) et arbitrées empiriquement par le bandit adaptatif (section 5.6) plutôt que choisies *a priori*.

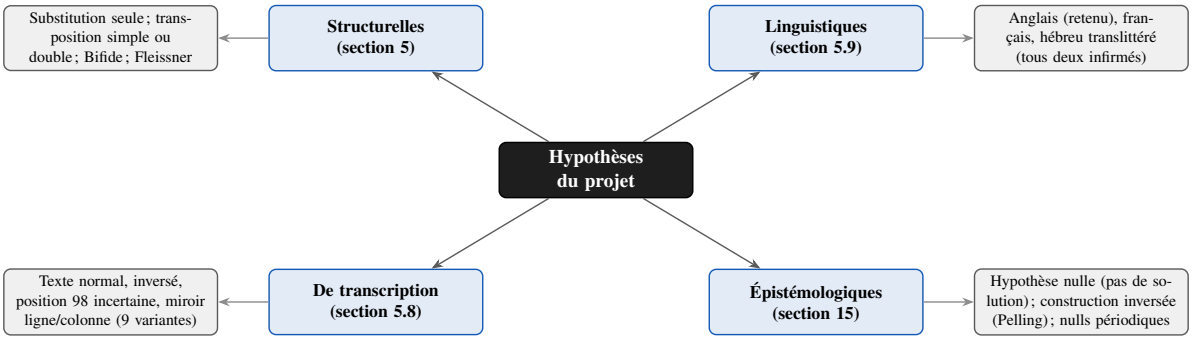

FIGURE 3 – *Taxonomie des hypothèses testées.* Chaque branche correspond à une dimension de recherche indépendante, combinée aux autres par le bandit adaptatif ; aucune n'est éliminée *a priori* avant d'être confrontée aux données.

Deux hypothèses méritent d'être soulignées avant le reste du rapport, car elles conditionnent l'interprétation de tout résultat obtenu :

L'hypothèse nulle est prise au sérieux. Rien ne garantit que le cryptogramme dissimule un texte récupérable. Une erreur de construction de l'auteur, hypothèse renforcée par l'anecdote de l'oubli et par la découverte de Pelling (section 15), reste une issue possible et est explicitement testée statistiquement (section 5.9), plutôt que rejetée par défaut comme le faisait implicitement la version 1 du solveur.

La langue anglaise est privilégiée, pas imposée. D'Agapeyeff, pleinement intégré à la vie professionnelle britannique (cartographe, wing commander de la RAF), écrit un livre en anglais pour un public anglophone dont le cryptogramme est le test final. Un texte clair dans une autre langue rendrait ce test invérifiable par la plupart des lecteurs visés, argument fonctionnel qui s'ajoute à l'argument biographique. Cette hypothèse reste néanmoins testée contre deux alternatives (français, hébreu) plutôt qu'admise sans vérification.

4. Le Premier Solveur du Projet et son Talon d'Achille

La première version du solveur (v1) combinait un modèle de langage **trigramme** (fenêtres de 3 lettres) construit sur un unique corpus embarqué d'environ 8 000 caractères (le texte de la Déclaration d'Indépendance américaine), une seule dimension de grille (14×14) explorée par 4 routes de lecture fixes, un filtre dur sur l'indice de coïncidence, et surtout un **seuil de succès fixé *a priori*** à -500 ou -400, sans jamais avoir été vérifié empiriquement (Figure 4).

Après trois jours d'exécution continue, aucun résultat ne dépassait ce seuil. Le raisonnement initial (rai-

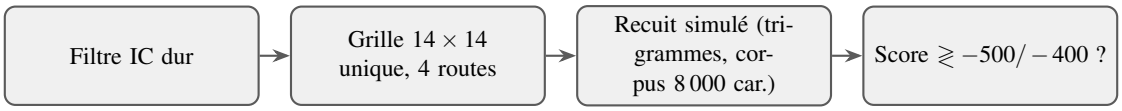

FIGURE 4 – *Pipeline v1 : chaîne linéaire, aucune étape de calibration.*

sonnable en apparence) concluait que le texte clair n'était probablement pas de l'anglais standard, ou que D'Agapeyeff s'était trompé dans la construction de son propre chiffre. Cette conclusion s'est révélée prématurée.

4.1 Le Diagnostic d'un Seuil qui N'Existait Pas

Pour vérifier la validité du seuil, un extrait authentique de *Pride and Prejudice* a été noté avec le modèle trigramme *exact* de la v1, sous trois régimes : clé identité (texte réel), meilleur résultat effectivement trouvé par la v1 (charabia issu de la grille de Fleissner), et bruit aléatoire pur.

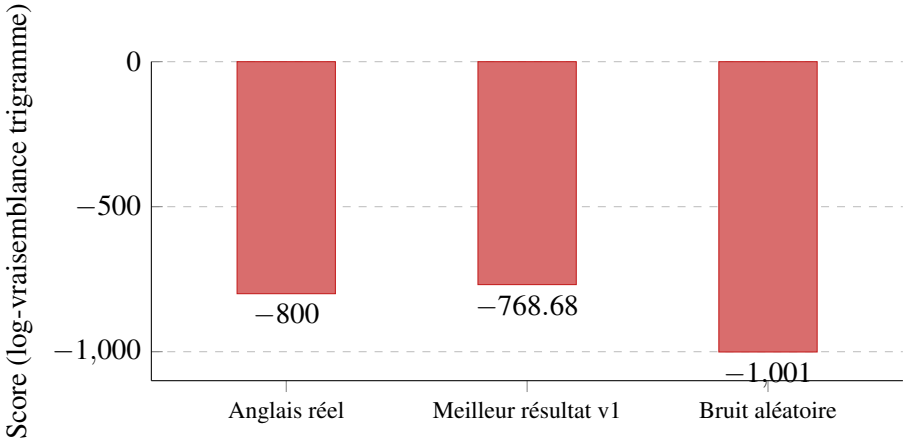

FIGURE 5 – *Calibration rétroactive du modèle trigramme de la v1.* Le meilleur résultat jamais trouvé par la v1 ($-768{,}68$, charabia) obtient un score *meilleur* qu'un texte anglais authentique noté sous son propre modèle (-800) : le seuil $-500/-400$ n'a donc jamais été atteignable, indépendamment de toute question sur le cryptogramme.

L'explication tient à la taille du corpus : sur seulement 8 000 caractères, la plupart des $26^3 = 17\,576$ trigrammes possibles n'apparaissent jamais, et le modèle retombe alors sur un lissage quasi uniforme qui ne discrimine plus l'anglais réel du charabia. La conclusion « pas d'anglais après 3 jours » de la v1 ne reflétait donc que les limites de son propre modèle de langage, pas une propriété du cryptogramme. Ce constat est le point de bascule de tout le projet : il a entraîné l'abandon complet de l'architecture v1 au profit d'une reconstruction pensée pour être vérifiable à chaque étape.

5. Un Solveur Reconstruit pour Être Statistiquement Défendable

5.1 Un modèle de langage plus discriminant

Le corpus de référence est porté à plus de 2,5 millions de caractères (romans du domaine public, Projet Gutenberg), et le modèle étendu aux **quadrigrammes** (fenêtres de 4 lettres, $26^4 = 456\,976$ catégories contre 17 576 pour des trigrammes). Le score d'un texte décodé avec la clé k s'exprime :

$$Q(k) = \sum_{i=1}^{N-3} \log_{10}\left(f_{c_i c_{i+1} c_{i+2} c_{i+3}}\right)$$

où c_i est la i-ème lettre du texte décodé et f_{abcd} la fréquence lissée (Laplace, $+0{,}01$) du quadrigramme $abcd$. Plus la fenêtre est large, plus il devient statistiquement difficile pour une permutation sans queue ni tête d'obtenir un score plausible par pur hasard, *à condition* que le corpus soit assez grand pour peupler ces catégories, d'où le passage simultané à 2,5 millions de caractères.

5.2 Une Mesure Auto-calibrée Remplace le Seuil Supposé

Avant chaque recherche, le pipeline chiffre lui-même un extrait de texte authentique connu avec sa propre mécanique, puis tente de le retrouver par recuit simulé. Le score obtenu $\mathscr{S}$ sert de référence haute *empirique*, propre au corpus et à la langue réellement utilisés, remplaçant définitivement le seuil arbitraire $-500/-400$ de la v1.

5.3 Quatre familles de chiffrement, explorées conjointement

À chaque itération, quatre méthodes sont testées sur le même flux de symboles (Figure 6) :

FIGURE 6 – *Les quatre méthodes de chiffrement recherchées conjointement.* La méthode D (en vert), ajout le plus récent, est motivée par la découverte de Pelling détaillée en section 15.

La méthode D mérite un développement : une transposition columnaire unique ne peut réordonner que des blocs entiers de colonnes d'une largeur w fixée. Deux passes successives, de largeurs w_1 et w_2 potentiellement différentes, atteignent des réarrangements de symboles individuels qu'aucune passe unique ne peut reproduire (Figure 7) : un angle mort de l'architecture initiale, comblé après la découverte de Pelling.

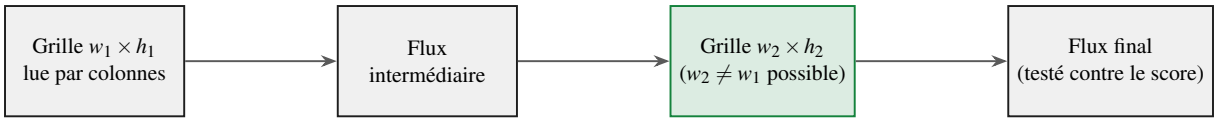

FIGURE 7 – *Mécanique de la double transposition.* Le passage par une seconde grille de largeur différente réarrange les symboles individuellement, pas seulement par blocs de colonnes, espace de recherche inaccessible à une transposition simple.

5.4 Grilles et routes de lecture partagées par les méthodes A, B et D

Les méthodes A, B et D reposent toutes sur le même outil sous-jacent avant même qu'une clé de substitution soit envisagée : disposer les 196 symboles d'un flux dans une grille rectangulaire de $w \times h$ cases, avec $w \times h = 196$, et les relire selon un chemin choisi. Cinq formes de grille sont testées, les couples de diviseurs de $196 = 2^2 \times 7^2$ suffisamment proches du carré pour être réellement distincts les uns des autres : 14×14, 7×28, 28×7, 4×49, 49×4. Les formes extrêmes, quasi linéaires (1×196, 2×98 et leurs transposées), sont exclues : sur une grille aussi fine, la plupart des routes de lecture ci-dessous coïncident ou presque, ajoutant un coût de recherche sans ajouter d'hypothèse réellement distincte.

Quatre routes de lecture canoniques sont définies pour une forme de grille donnée (Figure 8). En indexant une case de ligne r et de colonne c ($0 \leq r < h$, $0 \leq c < w$) par $\mathrm{idx}(r,c) = r \cdot w + c$, une route est une permutation des 196 indices de case :

$$\begin{aligned}
\textbf{Lignes}&: \ \mathrm{idx}(r,c),\ \text{pour } r = 0,\dots,h-1 \text{ (externe)},\ c = 0,\dots,w-1 \text{ (interne)}\\
\textbf{Colonnes}&: \ \mathrm{idx}(r,c),\ \text{pour } c = 0,\dots,w-1 \text{ (externe)},\ r = 0,\dots,h-1 \text{ (interne)}\\
\textbf{Boustro. lignes}&: \ \mathrm{idx}(r,c) \text{ si } r \text{ pair}, \quad \mathrm{idx}(r,w-1-c) \text{ si } r \text{ impair}\\
\textbf{Boustro. colonnes}&: \ \mathrm{idx}(r,c) \text{ si } c \text{ pair}, \quad \mathrm{idx}(h-1-r,c) \text{ si } c \text{ impair}
\end{aligned}$$

Les deux routes boustrophédon alternent le sens de lecture à chaque ligne ou à chaque colonne, d'après l'ancien tracé de labour au bœuf qui donne son nom à la technique, au cas où la grille aurait été remplie et relue dans des sens alternés plutôt que de façon uniforme.

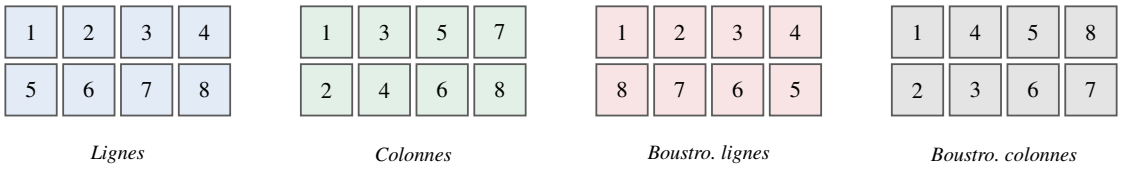

FIGURE 8 – *Les quatre routes de lecture sur une grille jouet* 4×2. Chaque case indique le rang (1er au 8e) auquel elle est lue, pas le symbole qu'elle contient : la position physique est fixe, seul l'ordre d'extraction change. À l'échelle réelle de 196 symboles, cinq formes de grille sont testées (texte principal), pas seulement cette illustration 4×2.

Pour la méthode A, la forme de grille et la route sont traitées comme un choix discret parmi $5 \times 4 = 20$, un bras du bandit adaptatif (section 5.6) ; une fois tiré, les 196 symboles sont réindexés une seule fois selon ce choix, et seule la clé de substitution est ensuite cherchée par recuit simulé (section 5.7) contre

ce réindexage fixe. La méthode B fixe au contraire la route à une lecture colonne par colonne, mais cherche l'*ordre des colonnes lui-même* comme une permutation libre de $\{0,\dots,w-1\}$, conjointement avec la clé de substitution, plutôt que de le restreindre à l'une des quatre routes canoniques ci-dessus ; concrètement, l'extraction remodèle le flux en un tableau $h \times w$, permute ses colonnes selon l'ordre cherché, et relit le résultat colonne par colonne. La méthode D enchaîne deux telles recherches d'ordre de colonnes à travers deux formes de grille choisies indépendamment (Figure 7 ci-dessus), la sortie de la première passe alimentant la seconde.

5.5 Méthode C : recherche d'un motif de grille tournante

La méthode C cherche un espace structurel entièrement différent, propre à la grille de Fleissner elle-même plutôt qu'à un couple grille/route. Une grille tournante 14×14 compte 49 trous physiques ; à mesure que le gabarit est tourné de 90° trois fois de suite (quatre positions au total), chaque trou expose une case différente parmi les 196, si bien que les 196 cases sont exposées exactement une fois sur l'ensemble des quatre rotations, en 49 groupes disjoints de quatre *positions orbitales*. Laquelle des quatre positions orbitales est le trou véritable, réellement découpé, pour chacune des 49 positions, n'est pas connue à l'avance et c'est exactement ce qui est cherché : un vecteur à 49 entrées, chacune dans $\{0,1,2,3\}$, perturbé une position à la fois et recuit conjointement avec la clé de substitution (section 5.7), le même schéma de recherche jointe que les méthodes B et D. La lecture procède rotation par rotation (0°, puis 90°, 180°, 270°) ; à chaque rotation, les 49 cases alors exposées sont lues dans l'ordre naturel de lecture (de gauche à droite, de haut en bas) tel qu'il apparaît sur la page, exactement comme le ferait une personne lisant physiquement à travers le gabarit perforé.

5.6 Bandit adaptatif

Chaque dimension de recherche (catégorie de flux, dimension de grille, paire de grilles) est un bras de bandit multi-armé. Le poids d'échantillonnage du bras i combine son meilleur score observé b_i et un bonus d'exploration décroissant avec le nombre d'essais déjà réalisés n_i :

$$w_i = \max\left(\underbrace{\frac{b_i - b_{\min}}{b_{\max} - b_{\min}}}_{\text{exploitation}} + \underbrace{\frac{c}{\sqrt{n_i + 1}}}_{\text{exploration}},\ \varepsilon\right), \qquad c = 0{,}6,\ \varepsilon = 10^{-3}$$

Aucun bras n'est jamais totalement exclu ($w_i \geq \varepsilon$) : une zone de recherche jugée peu prometteuse continue d'être testée, avec une fréquence décroissante mais jamais nulle, même logique que le tirage pondéré par indice de coïncidence appliqué aux flux de surchiffrement.

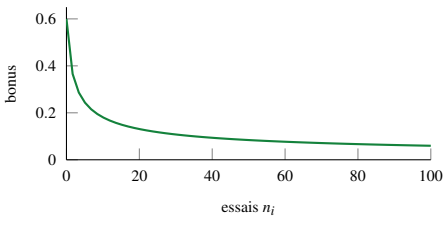

FIGURE 9 – *Le bonus d'exploration décroît mais ne s'annule jamais.*

5.7 Recuit simulé : le critère de Metropolis

Chaque combinaison testée, pour chaque méthode, quel que soit son espace de recherche structurel (une clé de substitution seule pour A ; une clé et un ordre de colonnes pour B ; une clé et un motif de grille à 49 entrées pour C ; une clé et deux ordres de colonnes pour D), est optimisée de la même façon : par recuit simulé, une recherche locale qui accepte parfois un candidat moins bon pour ne pas rester bloquée dans le premier optimum local rencontré.

Notons $S(\cdot)$ le score quadrigramme (section 5) du texte décodé sous un état candidat donné. À chaque itération, une petite perturbation aléatoire est proposée : pour la clé de substitution, deux de ses 26 entrées sont échangées ; pour un état structurel, une composante est perturbée sur place (un échange de colonnes pour B et D, un changement de position orbitale pour C). En notant $\Delta = S(\text{candidat}) - S(\text{actuel})$ et T la température courante, le candidat est accepté selon le critère de Metropolis :

$$P(\text{acceptation}) = \begin{cases} 1 & \text{si } \Delta > 0 \\ \exp(\Delta / T) & \text{si } \Delta \leq 0 \end{cases}$$

Un mouvement qui améliore le score est toujours conservé ; un mouvement qui le dégrade est conservé avec une probabilité qui diminue à mesure que Δ devient plus négatif ou que T baisse, ce qui permet à la recherche d'échapper aux optima locaux peu profonds au début, tout en se comportant comme une montée de gradient de plus en plus stricte par la suite. La température elle-même décroît géométriquement à chaque itération, $T \leftarrow \max(T_{\min},\ \alpha T)$, avec $\alpha = 0{,}9995$ et un plancher $T_{\min} = 0{,}05$ en deçà duquel un refroidissement supplémentaire n'a plus d'effet pratique sur la probabilité d'acceptation.

Deux variantes de ce schéma sont utilisées (Figure 10). La méthode A ne perturbe que la clé de substitution, contre une route déjà fixée par le tirage du bandit (section 5.4) ; elle démarre à $T_0 = 10$ et tourne sur 20 000 itérations. Les méthodes B, C et D perturbent conjointement la clé et l'état structurel, en alternant entre les deux à chaque itération avec une probabilité égale ; elles démarrent à $T_0 = 15$ et tournent sur 20 000 à 30 000 itérations selon la méthode. Avec ce calendrier, la température atteint son plancher après environ 11 400 itérations pour $T_0 = 15$, le même chiffre relevé indépendamment pour le réglage propre du chiffrement Four-square (section 14.5, qui réutilise exactement ce schéma de recherche jointe), et un peu plus tôt, vers 10 600 itérations, pour $T_0 = 10$.

C'est une recherche nettement plus puissante que faire correspondre directement le rang de fréquence des symboles chiffrés à l'ordre de fréquence théorique des lettres anglaises (E, T, A, O, N, I, S, R, H, …), le point de départ classique de la cryptanalyse manuelle. Cette approche à un seul coup suppose implicitement que le classement de fréquence observé sur le texte chiffré correspond déjà exactement au classement théorique ; sur un texte aussi court que 196 lettres, réparti sur jusqu'à 25 catégories de symboles, le seul bruit d'échantillonnage suffit à inverser le rang de deux lettres de fréquence théorique proche (N et I, ou R et H, par exemple), ce qui invalide silencieusement toute la clé qui en résulte. Le recuit simulé ne fait aucune hypothèse à un seul coup de ce type : chaque combinaison explore des

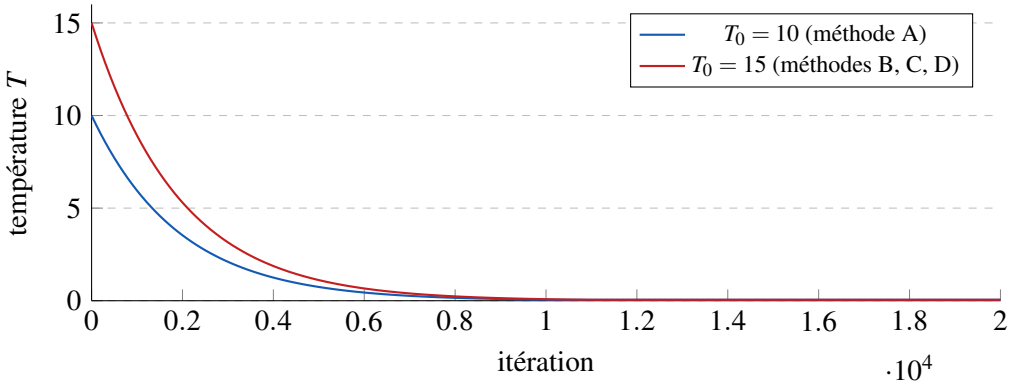

FIGURE 10 – *Refroidissement géométrique, $T \leftarrow \max(0{,}05,\ 0{,}9995\,T)$, pour les deux températures de départ utilisées.* Les deux courbes s'aplatissent au plancher bien avant que le budget d'itérations (20 000 à 30 000) ne soit épuisé, laissant les itérations restantes se comporter comme une montée de gradient de plus en plus stricte autour du meilleur état trouvé jusque-là.

milliers de clés candidates, notées non pas sur la fréquence d'une lettre isolée mais sur la cohérence de fenêtres de quatre lettres qui se chevauchent (section 5), et n'est mesurée, une fois trouvée, que contre la calibration par ligne de base nulle (section 5.9) plutôt que d'être crue sur son seul score.

5.8 Neuf hypothèses de transcription

Au-delà du texte tel qu'imprimé, huit variantes supplémentaires sont testées conjointement (Tableau 1), chacune motivée par un indice concret plutôt que par l'exploration arbitraire.

TABLE 1 – *Les 9 hypothèses de texte de base.*

Variante	**Nb.**	**Justification**
Normal	1	Transcription telle qu'imprimée (référence)
Inversé	1	Séquence lue à l'envers
Position 98 (r6-r9)	4	La paire #98 (« 04 ») est la seule à rompre un motif relevé indépendamment par un chercheur tiers ; ses 4 lectures alternatives plausibles $\{6,7,8,9\}$ sont testées plutôt qu'une correction unique arbitraire
Miroir lignes / colonnes / les deux	3	Une rotation de l'attribution chiffre→indice est déjà couverte par la recherche de clé Z5 ; un miroir modifie réellement la structure cyclique exploitée par le rechiffrement modulaire et le Bifide

5.9 Trois langues, une méthodologie pour les départager

Le pipeline complet a été répliqué sur un modèle **français** (corpus Gutenberg, accents normalisés) et un modèle **hébreu translittéré** (22 consonnes hébraïques → 22 lettres latines distinctes, texte non pointé). Un problème méthodologique s'est immédiatement posé : avec des centaines de milliers de combinaisons testées indépendamment, le *meilleur* score observé dérive naturellement vers le haut, même sur du pur

hasard (effet des comparaisons multiples). La référence self-test à essai unique ne suffit pas à juger un score obtenu après un grand volume de recherche.

Pièce 2 : Méthodologie de la calibration par ligne de base nulle

Faire tourner exactement le même pipeline sur une séquence mélangée aléatoirement, de même distribution de fréquences (même indice de coïncidence) que le cryptogramme réel, et comparer les deux *à volume d'essais égal*. Un score qui ne dépasse pas ce que le bruit pur atteint au même volume n'est pas un signal.

Cette méthodologie a démontré son utilité dès sa première application : l'hypothèse hébraïque produisait un score prometteur ($-993{,}83$), dépassant même sa propre référence self-test. À volume comparable (41 271 essais de chaque côté), la ligne de base nulle hébraïque a atteint $-985{,}48$, *meilleur* que le résultat réel (Figure 11). L'explication retenue : l'alphabet hébreu translittéré (22 lettres) produit un modèle de quadrigrammes moins épars ($22^4 = 234\,256$ catégories contre $456\,976$), statistiquement plus facile à satisfaire par hasard. Le français, lui, n'a montré aucun signal comparable (meilleur score $-1138{,}93$, moins bon que sa propre référence self-test $-1032{,}6$).

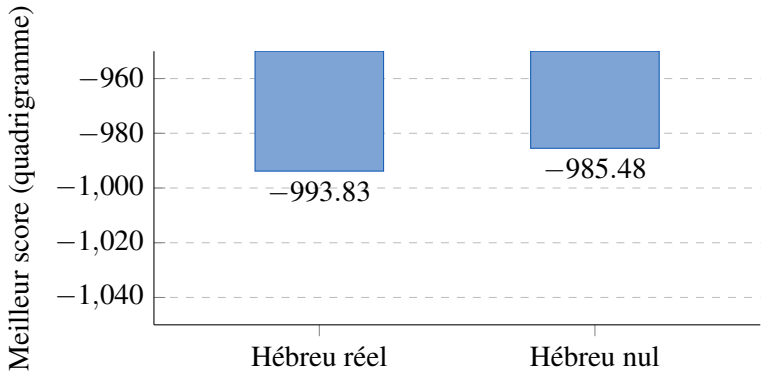

FIGURE 11 – *Le bruit pur fait mieux que le cryptogramme réel sous le modèle hébreu, à volume d'essais égal.* Le score de $-993{,}83$ n'est pas un signal, mais un artefact du plus petit alphabet translittéré.

5.10 Optimisation

Le score est calculé via compilation JIT (`numba`) et `numpy` vectorisé. Un goulot d'étranglement a été identifié dans la fonction de lecture par colonnes (boucle Python pure) ; sa vectorisation a produit un gain de vitesse d'un facteur $\approx 5{,}6$, bénéficiant à toutes les méthodes de transposition.

6. Les Indices Statistiques du Projet pour Mesurer le Succès

6.1 Indice de coïncidence

Pour une séquence de n symboles sur un alphabet de 25 valeurs, avec n_s occurrences du symbole s :

$$IC = \frac{\sum_{s=0}^{24} n_s(n_s-1)}{n(n-1)}$$

Invariant par substitution (ne dépend que des fréquences, pas de l'ordre), l'IC pondère (sans jamais exclure) les flux de surchiffrement les plus proches statistiquement d'un texte monoalphabétique plausible.

6.2 Score quadrigramme et poids bandit

Définis en section 5. Le premier est la métrique de fond ; le second oriente dynamiquement l'effort de calcul vers les zones les plus prometteuses, sans jamais fermer aucune porte.

6.3 Calibration par ligne de base nulle

Définie en section 5.9. Elle répond à une limite statistique incontournable : sur un grand nombre d'essais indépendants, le score *maximum* atteint par du bruit pur suit une loi de valeurs extrêmes qui croît avec le volume testé. Une comparaison valide doit donc opposer deux distributions de maxima obtenues au même volume, jamais un score isolé à une référence à essai unique.

6.4 Couverture de l'espace de recherche

Le tirage étant pondéré et non exhaustif, la couverture « moyenne » (tentatives ÷ taille de l'espace) surestime la couverture réelle. La fraction de flux jamais échantillonnés s'approxime par une loi de Poisson :

$$\text{fraction jamais tirée} \approx \frac{1}{M}\sum_{i=1}^{M} e^{-\lambda_i}, \qquad \lambda_i = N_{\text{essais}} \times \frac{w_i}{\sum_j w_j}$$

utilisée en section 7 pour quantifier honnêtement l'avancement réel de la recherche, au-delà du simple ratio tentatives/espace.

7. État des Lieux après 48 Heures de Calcul

7.1 Calibrations

Le Tableau 2 résume, pour les trois langues testées, l'écart entre un texte authentique et du bruit pur sous chaque modèle : la marge dans laquelle un résultat doit se situer pour mériter d'être examiné.

TABLE 2 – *Calibration self-test (essai unique) par langue.*

Langue	Score texte réel	Score bruit (plage)	Corpus
Anglais	≈ -935 à -1140	-1500 à -1590	3,29 M caractères
Français	$-1032{,}6$	-1594 à -1600	3,04 M caractères
Hébreu translittéré	-1055 à -1080	-1340 à -1480	746 000 caractères

7.2 Taille de l'espace et couverture réelle

Avec 9 hypothèses de texte de base et 146 529 flux de surchiffrement générés, l'espace combiné des quatre méthodes représente environ 5,97 millions de combinaisons structurelles distinctes.

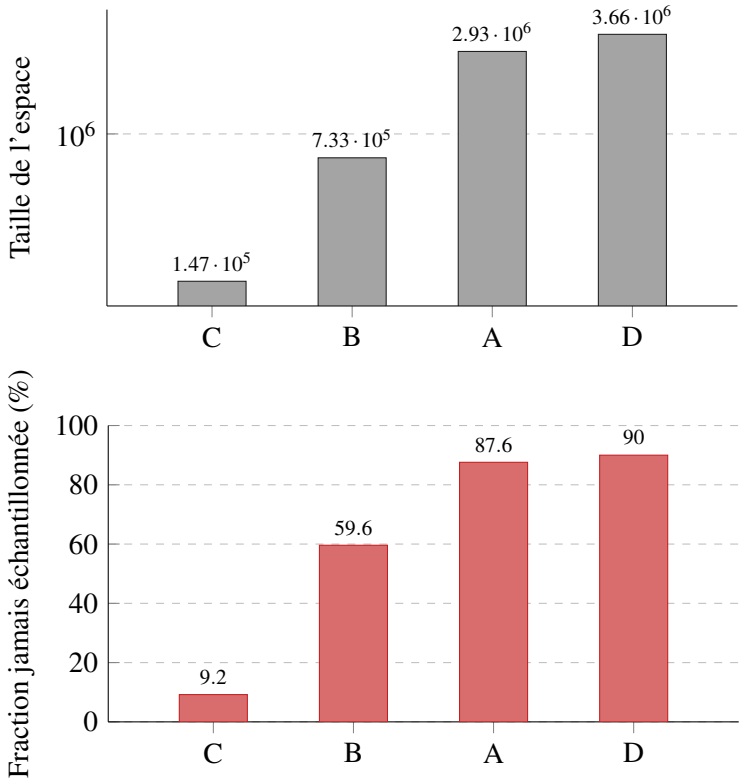

FIGURE 12 – *Taille de l'espace par méthode (gauche, échelle log) et fraction jamais testée après ≈48h (droite).* C = Fleissner, B = transposition simple, A = substitution+route, D = double transposition. Les méthodes A et D, aux espaces les plus vastes, restent trois à dix fois moins couvertes que B et C.

7.3 Meilleurs scores et verdict

Le meilleur score toutes méthodes confondues, à ce stade des 48 premières heures, est $-1078{,}76$ (transposition simple, hypothèse `pos98_r7`), à comparer à la calibration nulle anglaise qui, à volume comparable, atteignait déjà -1107 à -1119. L'écart résiduel (11 à 17 points) reste dans une zone où l'origine statistique du signal ne peut pas être établie avec confiance, et aucun texte décodé ne présente de cohérence lexicale au-delà de fragments isolés plausibles par hasard (« THE », « AND », « FOR »...).

Pièce 3 : Verdict à ce stade (48h)

Aucun signal statistiquement distinguable du bruit n'a été détecté, ni en anglais, ni en français, ni en hébreu translittéré, sur les $\approx$10 à 100 % de l'espace de recherche modélisé couverts après deux jours de calcul continu sur 12 cœurs.

Ce meilleur score a continué de dériver vers le haut après l'accélération décrite en section 8 : $-904{,}18$ au moment de la rédaction de cette mise à jour (double transposition, hypothèse `normal`). Le texte décodé correspondant reste toutefois un charabia sans cohérence lexicale (`QFFRESKTHOFTHNISORN...`, 196 lettres au total), ce qui illustre directement le principe méthodologique de la section 5.9 : à débit de recherche très supérieur, la loi des valeurs extrêmes fait mécaniquement grimper le *maximum* atteint par du pur bruit, indépendamment de toute présence de signal réel. La calibration nulle anglaise décrite en section 5.9 date du débit d'avant l'accélération et ne constitue donc plus une référence valide au débit actuel ; sa reconduction au nouveau débit fait partie des travaux immédiats (section 15).

8. Couverture Garantie et Compilation Complète pour Accélérer la Recherche

L'état des lieux de la section précédente a mis en évidence deux limitations structurelles du moteur de recherche, indépendantes de la validité des hypothèses testées. Un examen du mécanisme de tirage et du profil d'exécution a permis de les corriger sans rouvrir la question de l'espace d'hypothèses lui-même : la méthodologie reste celle de la section 5, seule son exécution est revue.

8.1 Le coût caché du tirage aléatoire avec remise

La formule de couverture de la section 5 (paragraphe « Couverture de l'espace de recherche ») a une conséquence contre-intuitive, restée implicite jusqu'ici : à volume d'essais N_{essais} égal à la taille de l'espace M (soit $\lambda = 1$, ce que l'on pourrait appeler une *passe nominale*), la fraction de combinaisons *jamais* tirée reste $e^{-1} \approx 36{,}8\,\%$. Le tirage pondéré, aussi utile soit-il pour concentrer l'effort sur les zones prometteuses, ne garantit donc jamais une couverture complète : plus d'un tiers de l'espace peut rester totalement inexploré par pur hasard d'échantillonnage, même après un nombre d'essais égal à sa taille.

Autrement dit, garantir une confiance raisonnable qu'aucune combinaison n'a été laissée de côté de-

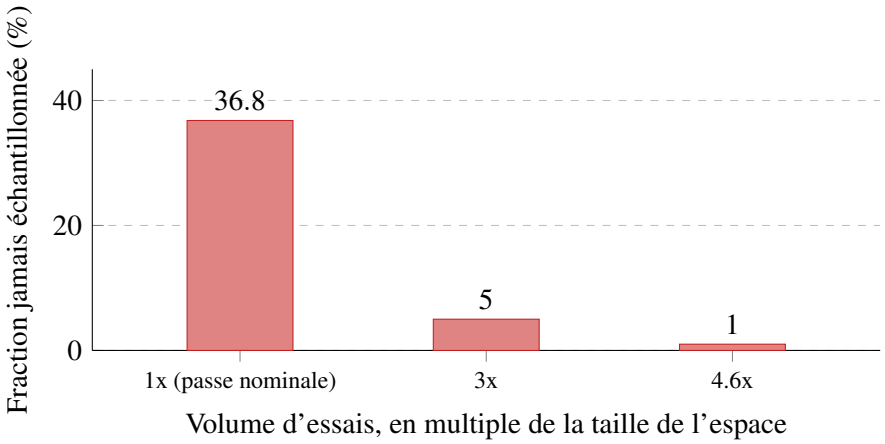

FIGURE 13 – *Fraction de l'espace jamais échantillonnée en fonction du volume d'essais (approximation de Poisson, poids uniforme).* Atteindre 95 % de couverture (au moins un tirage par combinaison) demande ≈ 3 fois le volume d'une passe nominale ; 99 % en demande ≈ 4,6 fois.

mande 3 à 4,6 fois plus d'essais qu'une simple passe nominale, non par une erreur de calcul, mais par une propriété structurelle du tirage aléatoire avec remise (problème dit du « collectionneur de coupons »).

8.2 Balayage systématique en deux phases

La correction retenue conserve le tirage pondéré existant pour l'affinage, mais lui adjoint une phase préalable de balayage exhaustif garantissant, avant tout recours au hasard, qu'aucune combinaison ne soit jamais distribuée deux fois. Pour chacune des quatre méthodes, un compteur entier partagé entre les 12 processus (`multiprocessing.Value`, verrouillé nativement) distribue un indice unique dans l'espace combiné flux × bras de cette méthode, par incrémentation atomique protégée par verrou (Figure 14). Chaque processus qui obtient un indice décode par simple division entière lequel flux et quel bras structurel il doit tester ; deux processus concurrents ne peuvent jamais recevoir le même indice, la garantie étant assurée au niveau du système d'exploitation par le verrou du compteur, pas par une convention de programmation.

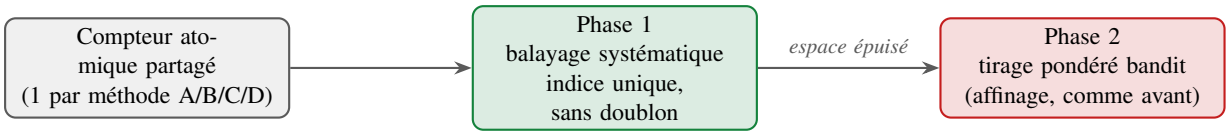

FIGURE 14 – *Mécanisme du balayage systématique en deux phases.* La Phase 1 garantit une couverture à 100 % avant que la Phase 2 ne reprenne le rôle d'affinage exploitation/exploration déjà décrit section 5.6.

Ce mécanisme s'applique indépendamment aux quatre méthodes : auparavant, un seul flux était tiré par itération et partagé entre A, B, C et D ; il est désormais tiré séparément pour chacune, condition nécessaire pour que chaque méthode couvre son propre espace flux × bras sans être couplée aux trois autres. La couverture exacte (et non plus estimée par la formule de Poisson) est depuis lors journalisée en continu dans un fichier de statut, lisible sans recalcul.

8.3 Compilation numba intégrale de la boucle de recherche

Avant ce changement, seul le calcul du score quadrigramme $Q(k)$ était compilé JIT (section 5, paragraphe « Optimisation ») ; la boucle de recuit simulé qui l'entoure, permutation de la clé, extraction structurelle, test d'acceptation de Metropolis, refroidissement, restait en Python interprété, avec l'overhead d'appel habituel de ce langage à chaque itération.

Un prototype limité à la méthode seule A (l'algorithme le plus simple des quatre, une seule perturbation possible) a d'abord permis de mesurer un gain de $\times 4{,}85$ avant d'engager le portage plus lourd des méthodes B, C et D. Leur fonction générique commune (recherche conjointe structure + clé, section 5) repose sur des fermetures Python passées en argument, un motif que la compilation `numba` en mode *nopython* ne prend pas en charge telle quelle : il a fallu écrire trois boucles spécialisées, une par famille structurelle (ordre de colonnes, grille de Fleissner, paire d'ordres), plutôt que de réutiliser l'abstraction générique existante.

Pièce 4 : Validation avant déploiement

Deux vérifications systématiques précèdent tout remplacement de la boucle Python par son équivalent compilé : (1) **fidélité numérique**, le score `numba` et le score Python actuel, évalués sur le même texte et la même clé, doivent coïncider aux arrondis machine près (écart mesuré : 0, exactement) ; (2) **qualité de recherche**, à budget d'itérations identique, sur un cas à signal connu (extrait anglais authentique chiffré par le pipeline lui-même), le meilleur score atteint par la version compilée doit rester dans la même fourchette que la version Python. Les deux conditions ont été vérifiées avant le déploiement en production.

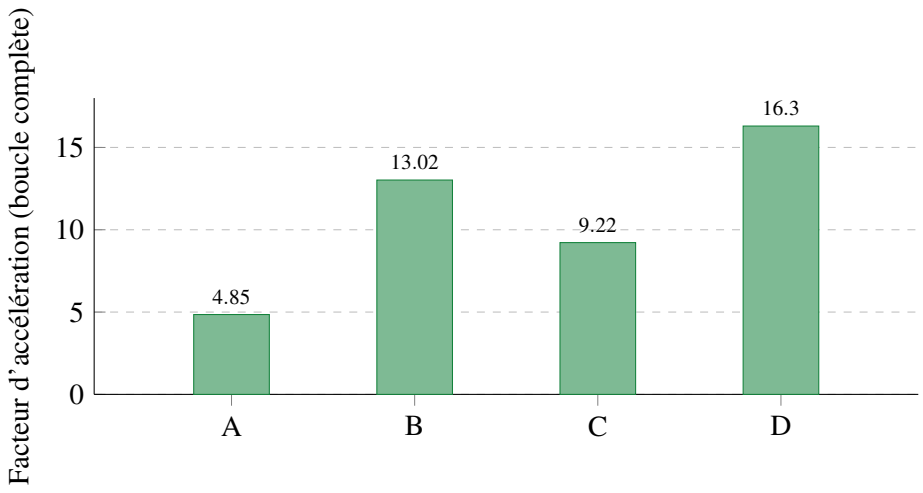

FIGURE 15 – *Facteur d'accélération mesuré par méthode, boucle de recherche entière (score et logique de perturbation compilés).* La méthode A, plus simple, avait proportionnellement moins de surcoût Python à éliminer que B, C et D, qui manipulent en plus un état structurel extrait à chaque perturbation.

8.4 Nouvelle estimation temporelle

Le Tableau 3 recalcule le temps nécessaire à une couverture garantie à 100 % (section 8.2) de chaque méthode, à partir du débit mesuré après l'accélération.

TABLE 3 – *Estimation du temps de couverture garantie à 100 %, avant et après l'accélération.*

Méthode	**Taille de l'espace**	**Accélération**	**Avant**	**Après**
C (Fleissner)	146 529	×9,22	≈0,75 j	≈2 h
B (transposition simple)	732 645	×13,02	≈3,8 j	≈7 h
A (substitution+route)	2 930 580	×4,85	≈15 j	**≈3,1 j**
D (double transposition)	3 663 225	×16,30	≈18-19 j	≈1,1-1,2 j

Le goulot d'étranglement change de méthode : D, auparavant la plus lente à couvrir malgré son espace plus vaste, cède la place à A, dont l'accélération mesurée est la plus faible des quatre. La couverture garantie à 100 % des quatre méthodes est désormais attendue en ≈**3 jours**, contre 18 à 19 jours avant ce changement, sans qu'aucune combinaison ne soit tirée deux fois avant que toutes les autres n'aient été testées au moins une fois.

8.5 Le goulot d'étranglement de journalisation

Quelques heures après le déploiement des sections 8.2 et 8.3, le débit observé en production a chuté d'environ 30 % et la charge CPU mesurée restait bloquée à 35-53 % au lieu de ≈100 % malgré 12 processus actifs. Un facteur secondaire a d'abord été identifié : le dossier du projet, synchronisé par un client cloud, tentait de re-synchroniser en continu le fichier de résultats, déjà volumineux et grossissant vite ; sa mise en pause n'a cependant réduit la charge CPU que marginalement (37-53 %), signe qu'il ne s'agissait pas de la cause principale.

La cause réelle tenait à la logique de journalisation elle-même. La fonction de recherche n'écrivait un résultat que si son score dépassait le meilleur score déjà observé *pour cette combinaison précise*, un dictionnaire initialisé à $-\infty$ au démarrage de chaque processus. Avant le balayage systématique, une combinaison pouvait être retirée plusieurs fois par le tirage pondéré, si bien que ce filtre éliminait la plupart des doublons. Mais la Phase 1 (section 8.2) garantit par construction qu'une combinaison n'est *jamais* revisitée : le test « ai-je fait mieux que la dernière fois sur cette combinaison » devient alors trivialement vrai à la toute première (et unique) visite. Mesuré en production : ≈97 % des tirages déclenchaient une écriture (23 281 écritures pour 23 996 tirages sur un intervalle de mesure), saturant le verrou de journalisation partagé entre les 12 processus, chacun ouvrant, écrivant puis refermant un fichier de plus en plus volumineux à quasiment chaque itération : c'est ce goulot d'entrées-sorties, pas la synchronisation cloud, qui bloquait le CPU.

Correctif. Le dictionnaire par combinaison unique a été remplacé par quatre scalaires, un record personnel par processus et par méthode (A/B/C/D), retrouvant l'intention d'origine : ne journaliser qu'une amélioration réelle, de plus en plus rare à mesure que la recherche progresse. Mesuré après correctif :

≈0,05 % des tirages déclenchent une écriture, et la charge CPU est repassée à 100 % sur des mesures répétées après redéploiement.

Ce correctif exigeait un redémarrage des processus, qui aurait sinon remis les compteurs de la Phase 1 à zéro. Une persistance a donc été ajoutée en même temps : au démarrage, chaque script relit le fichier de statut existant et reprend le balayage systématique exactement où il s'était arrêté, au lieu de repartir de zéro. Tout redémarrage ultérieur (panne, mise à jour de code, redémarrage machine) en bénéficie automatiquement.

8.6 Le Filet de Sécurité du Record Personnel Peut-il Manquer la Bonne Réponse ?

Le correctif précédent soulève une question de fond, distincte de la performance : un record personnel par processus et par méthode peut-il faire manquer un résultat correct ? Oui, en principe : si un processus rencontre tôt une combinaison de bruit qui, par optimisation, obtient un score meilleur que ce qu'obtient un texte anglais authentique (constaté empiriquement, section 9.3), un résultat de qualité « vrai anglais » rencontré *plus tard* par ce même processus ne battrait jamais ce record de bruit, et ne serait donc jamais journalisé, alors même que la recherche l'a bien testé et correctement noté.

Une première correction, un seuil de score absolu (score $> \mathscr{S} - 50$, en plus du record personnel), a été **testée et rejetée** : mesurée en isolation avant tout déploiement, elle aurait déclenché une écriture sur ≈43 % des tirages (Figure 16) : le recuit simulé pousse quasiment n'importe quelle séquence de 196 symboles vers un score « plausible », signal ou non, si bien que la zone « aussi bonne qu'un texte anglais réel » n'est pas rare une fois optimisée. Un tel seuil aurait recréé le goulot d'entrées-sorties de la section 8.5.

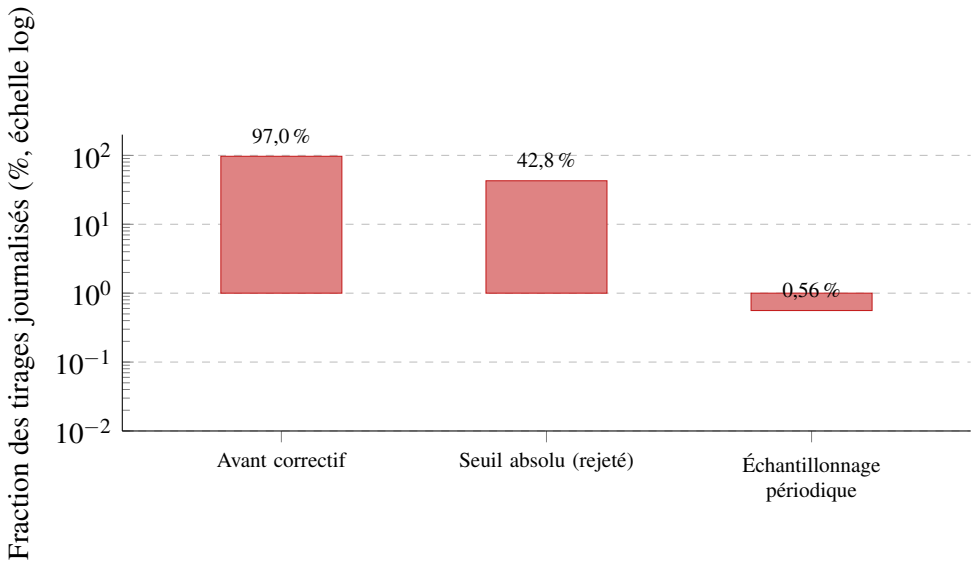

FIGURE 16 – *Trois régimes de journalisation mesurés.* Le seuil de score absolu, pourtant conçu comme un filet de sécurité peu fréquent, s'est révélé presque aussi coûteux que l'absence de filtre : la zone de score « plausible » n'est pas rare une fois le recuit simulé appliqué.

La solution retenue, un **échantillonnage systématique périodique**, journalise inconditionnellement un tirage sur 2 000 (indépendamment de son score), en plus du record personnel : une visibilité représentative sur ce qui est testé, sans dépendre d'un seuil de score qui ne filtre presque rien. Mesuré en régime stabilisé après déploiement : ≈0,56 % des tirages journalisés (Figure 16), charge CPU toujours à 100 %.

Pièce 5 : Limite assumée du filet de sécurité

L'échantillonnage périodique ne garantit pas de capturer précisément la bonne réponse si elle existe : il donne une visibilité représentative, pas exhaustive, sur les résultats corrects mais non-record. Rien n'est perdu côté *calcul* (la Phase 1 garantit que chaque combinaison est testée et notée au moins une fois) ; la limite porte uniquement sur la *journalisation*, donc sur la visibilité humaine du résultat après coup.

9. Réparation, Course de Scores et Verdict Actualisé de la Calibration Nulle

9.1 Réparation de la calibration nulle anglaise

Le script de calibration nulle anglaise (section 5.9) appelle directement la même fonction de recherche que le solveur principal, ce qui lui a valu d'hériter automatiquement des accélérations de la section 8, mais aussi de deux incompatibilités introduites par les mêmes changements.

1. **Signature incompatible.** L'ajout des compteurs de balayage systématique à la fonction de recherche (section 8.2) a cassé l'appel du script de calibration, qui ne les fournissait pas : plantage immédiat et reproductible, confirmé en exécutant le script plutôt qu'en relisant seulement le code.
2. **Statut jamais journalisé.** Le script de calibration décale ses identifiants de processus (pour les distinguer du run principal dans les journaux), mais l'écriture du fichier de statut était conditionnée à un identifiant précis, jamais atteint avec ce décalage. Le calcul progressait normalement mais restait invisible : découvert après une heure de calcul sans qu'aucune donnée de progression n'apparaisse.

Les deux ont été corrigées (compteurs et point de reprise propres au script de calibration ; condition de journalisation généralisée) et vérifiées par exécution avant redéploiement, conformément à la pratique déjà appliquée section 8.3. La calibration nulle tourne depuis avec 4 processus, un compromis entre rigueur (couvrir suffisamment son espace, neuf fois plus petit que celui du run principal puisqu'une seule hypothèse de transcription y est testée contre neuf) et contention (un ralentissement de ≈18-21 % du run principal a été mesuré et jugé acceptable). Sa Phase 1 a depuis atteint 100 % sur ses quatre méthodes.

9.2 La course de scores

Le meilleur score de chaque run a été suivi à intervalles réguliers à mesure que la calibration nulle rattrapait du volume (Tableau 4, Figure 17).

TABLE 4 – *Chronologie de la course de scores, run réel contre calibration nulle.*

Étape	Score réel	Score nul	Volume nul testé
Avant réparation de la calibration nulle	−904,18	(indisponible)	0
Calibration nulle en cours de rattrapage	−893,55	−905,74	≈96 500
Phase 1 nulle terminée à 100 %	−878,89	**−876,55**	≈830 000+

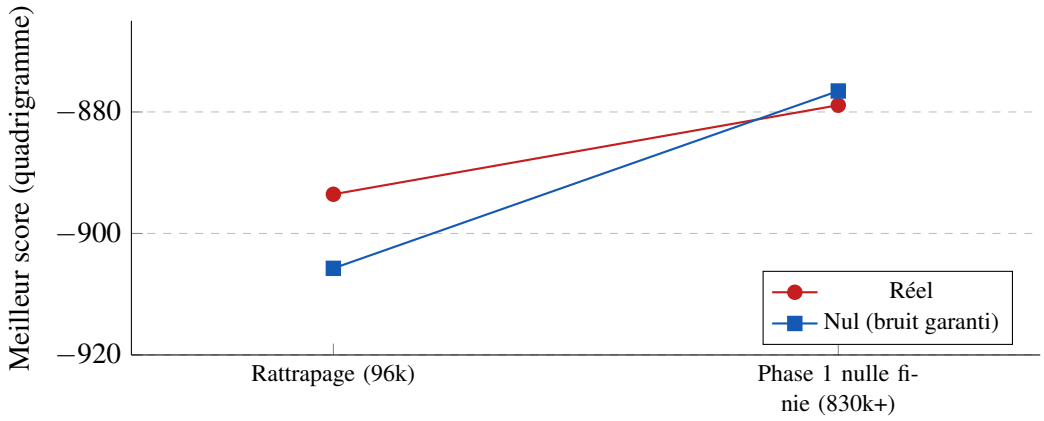

FIGURE 17 – *La calibration nulle rattrape puis dépasse le score réel à mesure que son volume testé augmente.* Même schéma que le débunkage de l'hypothèse hébraïque (Figure 11) : un écart qui semblait favorable au signal réel s'efface une fois le volume comparé équitablement.

Pièce 6 : Verdict actualisé

Une fois la calibration nulle à couverture complète sur son espace, son meilleur score (−876,55) *dépasse* le meilleur score réel (−878,89). Aucun signal statistiquement distinguable du bruit n'est établi à ce stade, y compris pour le meilleur résultat trouvé à ce jour sur le cryptogramme réel.

Cet écart, loin de se résorber, s'est depuis creusé. Le meilleur score réel est resté stable à −877,84 sur l'ensemble des points de contrôle ultérieurs, tandis que la calibration nulle a continué de progresser jusqu'à −860,37 après l'achèvement de son second balayage systématique (section 11) sur ses quatre méthodes, portant l'écart à 17,47 points en défaveur du signal réel, contre 1,29 point au moment du tableau ci-dessus. Le texte correspondant à ce nouveau record nul, inspecté directement, ne présente aucune trace de cohérence lexicale au-delà des fragments isolés déjà caractéristiques du bruit (section 12).

9.3 Comparaison à un texte anglais authentique

Les auto-calibrations self-test (Tableau 2) donnent la plage de score qu'obtient un texte anglais réel de 196 lettres sous ce modèle : ≈ -935 à -1160 selon l'extrait. Or les deux meilleurs scores actuels, réel *et* nul, dépassent *tous les deux* cette plage (Figure 18).

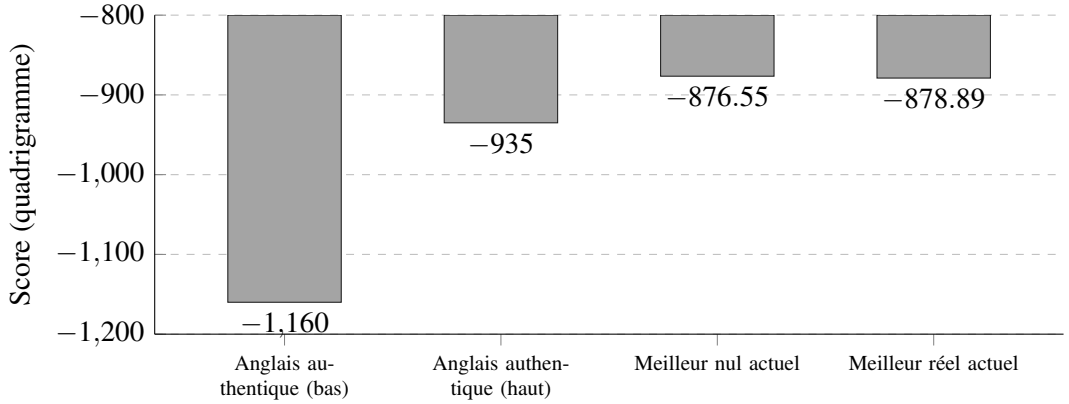

FIGURE 18 – *Les deux meilleurs scores actuels dépassent la plage obtenue par un texte anglais authentique.* Avec suffisamment de combinaisons testées, l'optimisation trouve des arrangements qui satisfont mieux la métrique quadrigramme qu'un passage anglais pris au hasard, sans qu'il s'agisse de vrai anglais : un effet des comparaisons multiples, pas un signal.

Ce constat, cohérent avec le texte décodé du meilleur résultat qui reste un charabia sans cohérence lexicale (section 9.5), retire au score seul toute valeur de filtre discriminant à ce volume de recherche : dépasser un texte anglais authentique n'est plus surprenant ni pour le run réel, ni pour du bruit garanti.

9.4 Reconstruction du pipeline et de la clé pour le meilleur résultat réel

Le pipeline complet ayant produit $-878{,}89$ a été reconstruit et vérifié algébriquement, pas seulement réaffirmé à partir du journal : variante de transcription `pos98_r6` (section 5.8), surchiffrement modulaire Z5 de clé $(1,1)$ pour les lignes et $(3,3)$ pour les colonnes (décalage constant -1 et -3 modulo 5), méthode D avec deux passes de transposition sur une grille 49×4 (deux ordres de colonnes distincts, journalisés).

La clé de substitution elle-même n'était pas journalisée à l'origine, seul le texte décodé final l'était. Elle a été retrouvée par déduction directe, pas par une nouvelle recherche : en rejouant les transformations connues (ci-dessus) sur le cryptogramme brut, on obtient la séquence de symboles juste avant application de la substitution ; en comparant position par position avec le texte décodé déjà journalisé, chaque symbole revient en moyenne 7 à 8 fois sur les 196 positions, et sa lettre associée doit être identique à chaque occurrence si la reconstruction est correcte. Zéro contradiction sur l'ensemble des positions confirme l'exactitude de la reconstruction, vérifiée en outre par recalcul exact de l'indice de coïncidence et du score (Tableau 5).

TABLE 5 – *Vérification de la reconstruction du pipeline pour le résultat* −878,89.

Quantité	Journalisée	Recalculée
Indice de coïncidence	0,0705	0,0705
Texte décodé (196 lettres)	(dans le journal)	identique caractère pour caractère
Score quadrigramme	−878,89	−878,89

9.5 Scan exhaustif d'intelligibilité lexicale

Le score et la comparaison au bruit (sections 9.2-9.3) portent sur un seul résultat à la fois. Une question distincte se pose : sur l'ensemble des résultats journalisés depuis le début du projet, existe-t-il une trace de cohérence lexicale ailleurs que dans le seul meilleur score ? Les deux fichiers de résultats ont été scannés exhaustivement : **3 868 347 textes décodés** au total.

Méthode. Un dictionnaire de ≈2 528 mots anglais courants a été extrait du corpus du projet. Pour chaque texte décodé, une structure trie (arbre de préfixes) recherche, à chaque position, le plus long mot du dictionnaire qui y commence, puis avance ; le taux de couverture est la fraction de caractères ainsi expliqués par des mots réels enchaînés.

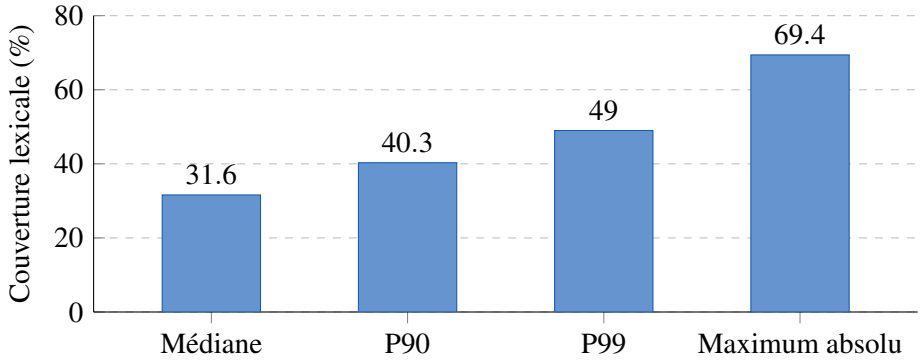

FIGURE 19 – *Distribution du taux de couverture lexicale sur les 3 868 347 textes décodés journalisés.* Même le maximum absolu (69,4 %) ne correspond qu'à des mots courts isolés (« OR », « ONLY », « THEM »...) sans aucune suite formant une phrase.

Résultat. Couverture médiane 31,6 %, P90 40,3 %, P99 49,0 %, maximum absolu 69,4 % sur l'ensemble du corpus de résultats (Figure 19). Aucun texte, à aucun niveau de couverture, ne présente de suite de mots formant une phrase reconnaissable. Fait notable : le texte comportant le plus grand nombre total de mots détectés (55, dont 31 de trois lettres ou plus) provient de la **calibration nulle**, bruit garanti, pas du run réel, devant l'intégralité des résultats du cryptogramme authentique sur ce critère également.

Pièce 7 : Verdict du scan d'intelligibilité

Ni le score quadrigramme, ni une mesure de cohérence lexicale indépendante du score, ne distinguent le cryptogramme réel du bruit pur à ce stade de couverture. Sur près de 3,9 millions de textes décodés examinés, aucun ne présente de structure linguistique reconnaissable au-delà de ce qu'un dictionnaire de mots courts explique par hasard.

10. Le Score d'Intelligibilité Lexicale, un Second Axe de Détection

Les sections précédentes établissent qu'aucun signal ne se détache du bruit sur le score quadrigramme, seule métrique de sélection utilisée jusqu'ici. Une question distincte s'ensuit : ce score a-t-il pu, par construction, écarter de la journalisation un résultat lisible mais faiblement noté sous ce modèle, notamment parmi les tirages antérieurs au filet de sécurité de la section 8.6 ? Cette section formalise un second critère, indépendant du premier, conçu pour y répondre.

10.1 Pourquoi ne pas piloter le recuit simulé par un score lexical

L'option la plus directe consisterait à substituer, dans la boucle de recuit simulé elle-même, un score de couverture en mots réels au score quadrigramme $Q(k)$. Cette option a été écartée pour une raison structurelle, bien documentée en cryptanalyse automatisée : le paysage d'adaptation (*fitness landscape*) d'un score lexical est presque partout plat. Une clé tirée au hasard ne produit, sur 196 caractères, pratiquement jamais de mot reconnaissable ; l'algorithme ne disposerait alors d'aucun gradient à suivre à la majorité des échanges de deux lettres testés. Le score quadrigramme, à l'inverse, varie continûment à chaque échange (chaque lettre contribue à plusieurs fenêtres qui se chevauchent), fournissant le gradient dense nécessaire à la convergence du recuit. Piloter l'optimisation par un score lexical reviendrait ainsi, sur l'essentiel de sa trajectoire, à une recherche quasi aléatoire, vraisemblablement moins efficace que le dispositif existant.

Le recuit simulé continue donc de maximiser $Q(k)$, inchangé. La mesure d'intelligibilité est calculée *a posteriori*, une seule fois par combinaison testée, une fois la clé optimale déjà trouvée : son coût est négligeable au regard des 20 000 à 30 000 itérations de l'optimisation elle-même.

10.2 Définition du critère

Pour un texte décodé, deux quantités sont retenues, calculées par segmentation en plus-long-préfixe (structure trie) sur un dictionnaire de référence :

- Cov, la fraction des 196 caractères expliqués par un enchaînement de mots réels, éventuellement entrecoupés de lettres non expliquées ;
- Run, la longueur (en mots) de la plus longue *suite continue*, sans aucun caractère non expliqué, une exigence strictement supérieure à la seule couverture et plus proche, par construction, de ce qu'exigerait une phrase reconnaissable.

Chaque processus conserve, indépendamment du record de score quadrigramme, un record personnel de Cov et de Run par méthode. Un résultat est journalisé dès qu'il bat l'un quelconque des trois records (score, couverture, suite), en plus de l'échantillonnage périodique déjà en place (section 8.6) : le mécanisme récupère ainsi, sans reproduire le goulot d'entrées-sorties de la section 8.5, les résultats fortement lisibles mais faiblement notés par le modèle quadrigramme.

10.3 Un dictionnaire contaminé, détecté avant tout déploiement

Le dictionnaire du scan exploratoire de la section 9.5 se limitait, par construction ponctuelle, aux 2 528 mots les plus fréquents du corpus du projet. Une première tentative d'élargissement a combiné ce corpus à une liste de mots anglais « courants » récupérée en ligne (dérivée de fréquences d'usage sur le web), portant le dictionnaire à 19 208 entrées.

Conformément à la pratique déjà appliquée section 8.6 (toujours mesurer l'effet d'un nouveau filtre sur du bruit avant de lui faire confiance), ce dictionnaire élargi a été testé sur du bruit purement aléatoire avant tout déploiement. Le résultat a invalidé la tentative : couverture médiane de 72,4 % sur du bruit garanti (Figure 20), très supérieure à la fourchette de 31 à 37 % observée jusqu'alors. L'inspection a révélé la cause : la liste externe, dérivée de fréquences web plutôt que d'un dictionnaire linguistique, comptait 404 « mots » de deux lettres parmi les 676 combinaisons possibles, très majoritairement des sigles et du jargon technique (`AOL`, `API`, `ACM`, `AMD`) plutôt que des mots anglais authentiques, qui s'apparient par pur hasard combinatoire à n'importe quelle séquence de lettres.

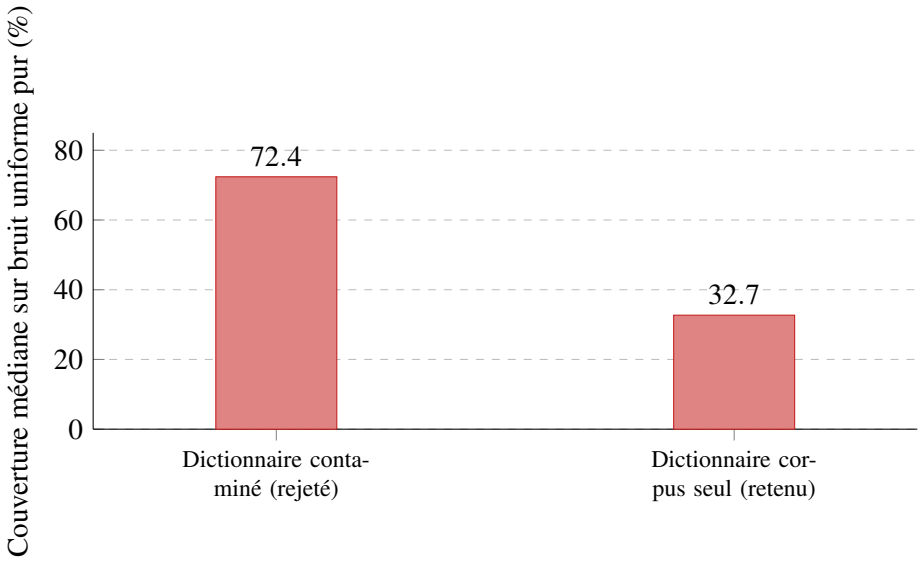

FIGURE 20 – *Effet d'un dictionnaire contaminé sur le plancher de bruit du critère de couverture.* La liste externe (19 208 mots) fait plus que doubler la couverture mesurée sur du bruit garanti, rendant le critère quasiment non discriminant ; le dictionnaire retenu se limite au corpus du projet, sans plafond arbitraire (23 310 mots, fréquence ≥ 1).

Le dictionnaire finalement retenu écarte la liste externe et conserve uniquement les mots du corpus de romans déjà utilisé pour le modèle de langage, cohérent avec l'hypothèse de prose anglaise ordinaire (section 3), sans le plafond arbitraire de 2 528 mots du scan ponctuel initial : l'ensemble des mots distincts de fréquence ≥ 1, soit 23 310 entrées. Testé sur le même bruit de référence, ce dictionnaire produit une couverture médiane de 32,7 % (Figure 20), cohérente avec les mesures antérieures de la section 9.5.

Pièce 8 : Déploiement

Le critère a été déployé après double vérification en isolation (`-workers 2`) : un premier essai avec le dictionnaire contaminé a confirmé le symptôme prédit (couverture de 93 à 97 % sur les résultats réels *et* nuls, signe d'un critère non discriminant) avant tout déploiement en production ; corrigé, le second essai a produit des couvertures de 65 à 81 %, cohérentes des deux côtés, sans erreur. Redéployé en pleine échelle sans perte de couverture des balayages en cours.

11. Le Second Balayage Systématique de la Phase d'Affinage

Le mécanisme à deux phases de la section 8.2 garantit qu'aucune combinaison n'est laissée de côté au premier passage, mais son ancienne Phase 2 (tirage pondéré bandit, section 5.6) souffre d'une limite mise en évidence par la course de scores de la section 9.2 : le meilleur score de la calibration nulle en méthode D est resté figé pendant plus de onze heures malgré plus d'une journée de tirage pondéré, signe d'un rendement quasi nul une fois la Phase 1 épuisée.

Constat déterminant. La quasi-totalité du premier balayage systématique, toutes les combinaisons des méthodes A, B, C et l'essentiel de D, a eu lieu *avant* l'introduction du critère d'intelligibilité (section 10) : ces combinaisons n'ont donc jamais été évaluées sous ce second critère. Or l'ancienne Phase 2, pondérée par le meilleur score *quadrigramme* observé par bras, ne garantit aucunement de revisiter les bras faiblement notés sous ce modèle mais potentiellement riches en intelligibilité lexicale, exactement les combinaisons que l'on cherche à rattraper.

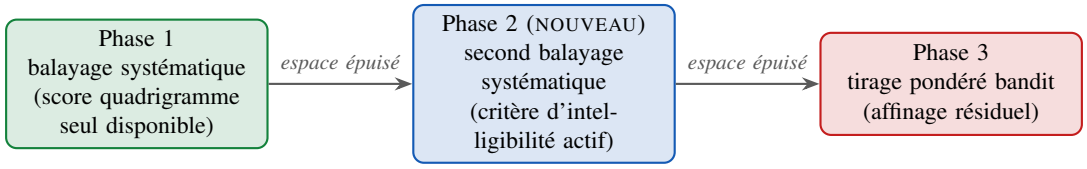

FIGURE 21 – *Architecture à trois paliers, remplaçant l'enchaînement direct Phase 1 → bandit de la Figure 14.* Le second balayage (Phase 2, non biaisé par le score, garanti sans doublon comme la Phase 1) assure que chaque combinaison est testée au moins une fois sous les deux critères de sélection ; le bandit ne reprend son rôle d'affinage qu'une fois les deux passages garantis épuisés.

L'implémentation reprend le mécanisme de compteur atomique de la section 8.2, dupliqué pour un second balayage indépendant du même espace (Figure 21) : déterministe, sans doublon, non pondéré par le score. Le coût temporel d'un second passage complet est identique à celui du premier (même taille d'espace, même coût de recuit par combinaison) ; au débit de production mesuré, plusieurs jours supplémentaires sont nécessaires pour compléter le second balayage des méthodes A et D, jugé acceptable au regard de l'objectif d'exhaustivité du projet.

12. Chasse aux Fragments Lexicalement Cohérents

Le scan de couverture de la section 9.5 mesure une propriété globale (fraction de caractères expliqués) mais ne garantit pas qu'un fragment de plusieurs mots consécutifs, seul indice réellement convaincant de cohérence linguistique, soit absent des résultats accumulés. Cette section documente une recherche ciblée, en deux méthodes complémentaires, de tels fragments.

12.1 Vérification par n-grammes de mots attestés dans le corpus

Une première méthode automatique recherche, parmi les mots consécutifs et sans trou d'un texte décodé, des paires ou triplets qui reproduisent une séquence effectivement présente, dans cet ordre, dans le corpus d'entraînement. Appliquée à l'ensemble des textes à couverture $\geq 50\,\%$, cette méthode révèle une limite immédiate : 85 % des textes contiennent au moins une paire de mots attestée, taux bien trop élevé pour être informatif, les mots-outils les plus fréquents de la langue anglaise (*it, is, of, to, as...*) formant des paires attestées avec une probabilité déjà substantielle par pur hasard. La restriction aux triplets réduit ce taux à 163 occurrences sur 6 282 textes examinés, mais la quasi-totalité reste composée des mêmes mots-outils (« so is it », « it is the »...). Un filtre supplémentaire, exigeant que les trois mots comptent chacun au moins trois lettres, ne laisse subsister que **3 occurrences sur l'ensemble des textes examinés**.

Cette méthode présente cependant une limite de principe, reconnue avant d'en tirer des conclusions définitives : elle ne peut confirmer qu'une coïncidence lexicale est *probablement* fortuite (en établissant qu'elle reproduit une séquence déjà rencontrée), mais ne peut jamais prouver l'absence d'un fragment original et cohérent, qui n'aurait aucune raison de reproduire mot pour mot une phrase des cinq romans du corpus d'entraînement.

12.2 Lecture exhaustive et directe

En complément, les 515 textes de plus forte couverture lexicale ($\geq 80\,\%$, un seuil délibérément élevé, très supérieur au plancher de bruit d'environ 33 % établi section 10.3) ont été lus intégralement avec un jugement sémantique direct, sans intermédiaire mécanique. Le motif observé sur l'ensemble du scan de la section 9.5 s'y confirme sans exception : des mots courts isolés, parfois groupés par deux à quatre, toujours noyés dans des séquences sans queue ni tête des deux côtés.

12.3 La découverte la plus marquante du projet à ce jour

Un résultat de la calibration nulle, bruit garanti par construction, contient une suite ininterrompue de six mots réels du dictionnaire, dont quatre forment une clause anglaise grammaticalement irréprochable (Figure 22). La séquence « if we can » a été vérifiée comme effectivement attestée, dans cet ordre, dans le corpus d'entraînement, confirmant qu'il ne s'agit pas d'une erreur de segmentation.

```
ZMOAFINKOFLEDDOCKORIKECERNTNOTOWACALLFONEUTSLIKIN
STWARONGUDMASOLDSTAPAGUPSSNORANDDOIMUMMINDLXNSNOS
URESUPSOFCOUNTHANSWEWWILLONSTREMT HELDIFWECANWIN VE
RPRUSKDARKAQGTOSURVEHUMANDARWHORGAMOOUKNORLXDEDFE
```

FIGURE 22 – *Fragment de six mots consécutifs sans aucun caractère non expliqué, extrait d'un résultat de la calibration nulle* (`Score: -988,82`, DoubleTransp $49 \times 4 \rightarrow 49 \times 4$, `Cov:0,857 Run:12`). Segmenté : « HELD IF WE CAN WIN VE » ; le sous-fragment « IF WE CAN WIN » constitue une proposition conditionnelle anglaise correcte. Provenance : bruit pur, aucun rapport avec le cryptogramme réel.

Ce résultat est significatif non pour ce qu'il révèlerait sur le cryptogramme (il n'y est pour rien, provenant d'une séquence mélangée aléatoirement), mais pour ce qu'il démontre sur le pouvoir du hasard à ce volume de recherche : une suite de six mots grammaticalement cohérente peut apparaître spontanément dans du bruit pur, une fois des centaines de milliers de tirages indépendants accumulés. C'est la démonstration la plus directe, à ce jour, que l'apparence de cohérence lexicale d'un résultat, même impressionnante en lecture isolée, ne constitue en rien une preuve de signal.

12.4 Synthèse des meilleurs fragments identifiés

En combinant nombre total de mots détectés et présence d'un fragment reconnaissable, les trois meilleurs résultats du projet sont résumés Table 6. Fait notable, les deux textes détenant les records absolus de couverture lexicale (section 10, 98 % côté réel et 100 % côté nul) ne contiennent, une fois inspectés directement, *aucun* fragment cohérent malgré leur score de couverture supérieur : la couverture brute peut être gonflée par des dizaines de coïncidences de mots de deux lettres sans qu'aucune ne s'enchaîne en quoi que ce soit de lisible.

TABLE 6 – *Les trois meilleurs fragments lexicalement cohérents identifiés sur l'ensemble du projet.*

Fragment	**Provenance**	**Mots totaux (texte)**	**Longueur du fragment**
« WALK OUT »	Réel	56	suite de 59 mots sans trou
« FOR HER TO TRY »	Réel	56	4 mots grammaticaux consécutifs
« THE WAY HAD »	Nul (bruit)	50	3 mots consécutifs

13. Recherche d'un Signal Structurel de Second Ordre

Une question naturelle, une fois établi qu'aucun résultat individuel ne se distingue du bruit (sections 9.2, 10, 12), consiste à demander si les résultats qui « ressemblent à quelque chose » pourraient néanmoins servir de point de départ à un affinage ultérieur, sur le modèle du recuit simulé qui exploite un gradient de score au sein d'une combinaison donnée. La réponse de principe est négative : puisque le bruit pur atteint ou dépasse le réel sur les trois critères déjà mesurés, la « ressemblance » d'un résultat ne porte, par construction, aucune information sur sa proximité à une éventuelle vraie réponse ; repartir de ces

résultats reviendrait à optimiser pour de la coïncidence.

13.1 Regroupement par paramètre structurel

Une piste plus indirecte reste envisageable : si une combinaison structurelle particulière (grille, route de lecture, hypothèse de transcription) est la bonne, ses meilleurs scores devraient se regrouper de façon distinctive, même si aucun score individuel pris isolément ne s'en trouve démarqué. Le regroupement des 200 meilleurs scores de chaque run par méthode et dimension de grille est sans ambiguïté partagé par le bruit : 90,5 % du top réel et 80,5 % du top nul proviennent de la méthode D sur la grille $49 \times 4 \rightarrow 49 \times 4$. Cette convergence ne constitue *pas* un signal ; elle reflète la puissance combinatoire propre à cette famille structurelle (le plus grand espace, le plus de degrés de liberté de réarrangement), indépendamment de toute vérité sur le texte clair.

13.2 Taux de réussite normalisé par variante de transcription

Une analyse plus fine, restreinte à la grille dominante, compare pour chaque hypothèse de transcription (section 5.8) son taux de réussite dans le top 200, normalisé par le volume effectivement testé (et non le seul compte brut, trompeur : la variante `normal` représente à elle seule 56 % du volume testé sur cette grille, contre 5 à 6 % pour chacune des huit autres). Le résultat, résumé Figure 23, montre une hiérarchie nette : `inverse` atteint le top 200 environ quatre fois plus souvent par tirage que `normal`, suivie par `lignes_miroir`, `pos98_r9` et `colonnes_miroir`.

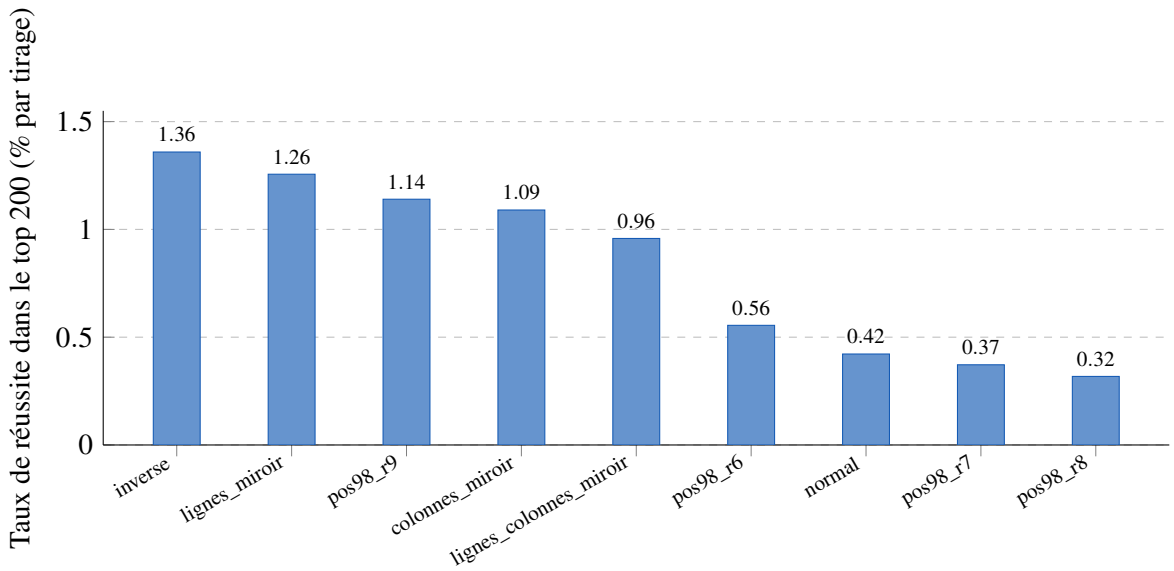

FIGURE 23 – *Taux de réussite dans le top 200 des scores réels, par variante de transcription, normalisé par le volume testé.* La variante `normal`, malgré 56 % du volume total testé, affiche l'un des taux de réussite les plus faibles ; `inverse` réussit environ quatre fois plus souvent par tirage.

13.3 Une confusion identifiée avant toute conclusion

Ce résultat ne peut cependant pas être interprété comme un signal en l'état : il est confondu par le mécanisme même du bandit adaptatif (section 5.6, désormais Phase 3 de la Figure 21). Un bras massivement ré-échantillonné parce qu'il a obtenu tôt un bon score continue d'accumuler du volume, mais chaque tirage supplémentaire a mécaniquement moins de chances de battre un record déjà proche de l'optimum local trouvé par le recuit simulé ; un bras peu ré-échantillonné, à l'inverse, bénéficie de tirages plus « frais », avec une probabilité de nouveauté plus élevée. La variante `normal` pourrait ainsi afficher un faible taux de réussite *parce qu'elle a déjà été abondamment exploitée*, indépendamment de toute question de validité de l'hypothèse elle-même. Les données actuelles ne permettent pas de distinguer ces deux explications.

Pièce 9 : Marquage de phase, pour trancher avec des données futures

La fonction de tirage a été modifiée pour retourner, en plus de l'indice de combinaison, le numéro de la phase qui l'a produit (1, 2 ou 3), désormais journalisé avec chaque résultat. Les Phases 1 et 2 (Figure 21) sont garanties non biaisées par construction, chaque flux y étant testé exactement le même nombre de fois ; une future analyse restreinte à ces seules entrées éliminera le biais d'auto-renforcement du bandit identifié ci-dessus. Déployé le 2026-08-11 et vérifié en isolation avant redéploiement en pleine échelle ; les deux seconds balayages systématiques (section 11) sur les méthodes A et D, encore loin d'être achevés au moment de la rédaction, fourniront le volume non biaisé nécessaire à une conclusion.

13.4 Vérification du volume disponible et clôture du fil

Le marquage de phase (Pièce ci-dessus) a été conçu pour permettre une répétition non biaisée de l'analyse précédente une fois un volume suffisant accumulé. Une vérification du volume réellement disponible, effectuée une fois les deux balayages systématiques achevés sur les quatre méthodes (run principal comme calibration nulle), a toutefois révélé un écart entre la couverture des combinaisons *testées* et le volume des résultats *journalisés* sous ce marquage : la journalisation reste volontairement clairsemée (un tirage sur 2000, plus les améliorations de record personnel, section 8.6), plafonnant mécaniquement le volume exploitable à une fraction fixe du nombre de combinaisons balayées, indépendamment de l'avancement ultérieur. Aucune entrée ne porte le marquage Phase 1 (les quatre méthodes avaient déjà achevé leur premier balayage avant le déploiement du marquage lui-même) ; le marquage Phase 2 ne compte que 2290 entrées pour la méthode A et 2508 pour la méthode D, concentrées sur trois des neuf variantes de transcription pour A et cinq pour D, les autres n'affichant aucune entrée.

Un élargissement du filtre aux entrées non marquées (antérieures au déploiement du marquage, potentiellement contaminées par l'ancien bandit non systématique) a alors été tenté comme volume de secours. Le résultat, résumé Tableau 7, révèle une contamination plus grave que le biais initialement suspecté : le classement obtenu par ce filtre élargi suit fidèlement l'ordre chronologique d'introduction des neuf va-

riantes dans le code (`normal` et `inverse` en premier, les trois variantes « miroir » en dernier, section 5.8) plutôt qu'une différence de qualité intrinsèque, la variante la plus anciennement testée accumulant mécaniquement le plus d'historique de journalisation. Preuve supplémentaire : ce classement *inverse* celui obtenu par la première tentative biaisée (section 13), où `inverse` dominait et `normal` affichait le taux le plus faible : deux mesures biaisées de la même quantité, mutuellement contradictoires, ne peuvent constituer un signal.

Variante	Taux A (%)	Taux D (%)
`normal`	**60,0**	**64,4**
`inverse`	21,4	8,1
`pos98_r6` – `r9`	9,1 – 9,6	7,9 – 8,2
variantes « miroir » (3)	7,0 – 7,1	6,3 – 6,3

TABLE 7 – *Taux d'entrées journalisées par volume testé, filtre élargi aux données non marquées.* L'ordre du classement reproduit exactement l'ordre chronologique d'introduction des variantes dans le code, pas une différence de qualité.

Ni le filtre strict (volume insuffisant et lacunaire) ni le filtre élargi (contaminé par un artefact d'exposition historique) ne permettent une comparaison fiable avec l'instrumentation actuelle. Ce fil d'analyse est donc **clos par insuffisance de données**, ni confirmé ni infirmé sur le fond ; sa réouverture propre nécessiterait un changement de conception (échantillonnage renforcé dédié, ou horodatage systématique des entrées journalisées).

14. Le Chiffrement Four-square, un Nouveau Chantier de Recherche

14.1 Épuisement de l'espace A-D et arrêt délibéré

La couverture garantie des deux balayages systématiques sur les quatre méthodes A à D a atteint 100 %, run principal et calibration nulle confondus. Le meilleur score réel a, dans l'intervalle, franchi un nouveau record ($-853{,}60$, contre $-877{,}84$ précédemment), dépassant pour la première fois le meilleur score de la calibration nulle ($-860{,}37$). Le texte décodé correspondant a néanmoins été vérifié directement et ne contient aucune suite de mots grammaticalement cohérente (« and low ward won… did tall one… wont dis and strove… »), conforme au constat déjà établi section 9.2 : l'optimisation à grande échelle finit, par le seul effet des comparaisons multiples, par produire des arrangements légèrement mieux notés par le modèle quadrigramme sans qu'il s'agisse d'anglais réel. Ce croisement de scores n'est donc pas un signal.

La Phase 3 (bandit adaptatif, section 5.6) n'ayant historiquement apporté qu'un gain marginal une fois les deux balayages systématiques épuisés (déjà observé pour la méthode D en calibration nulle, section 9), et l'espace d'hypothèses A-D étant désormais couvert de façon exhaustive sans qu'aucun signal ne s'en dégage, les deux processus ont été arrêtés proprement (persistance de l'état de couverture inchangée, reprise possible sans perte à tout moment) pour consacrer l'effort à une famille de chiffrement absente

de l'espace modélisé jusqu'ici.

14.2 Motivation

Le projet de recherche indépendant le plus abouti sur ce cryptogramme (dagapeyeffresearch.com) rapporte son résultat le plus prometteur, bien que non concluant, sous un modèle « Four-square » ($Q = -692{,}13$ sous son propre système de score, plus faible que le nôtre) : une famille de chiffrement structurellement différente de la substitution et de la transposition explorées jusqu'ici, combinant substitution *et* diffusion au sein d'une même opération sur des paires de lettres plutôt que des lettres isolées.

14.3 Mécanique

Le chiffrement Four-square opère sur des *digrammes* (paires de lettres consécutives) au moyen de quatre grilles de Polybius 5×5 disposées en carré : les grilles haut-gauche et bas-droite portent l'alphabet standard non modifié, tandis que les grilles haut-droite et bas-gauche sont *keyées* (une permutation de l'alphabet, à déterminer). Pour déchiffrer un digramme chiffré (C_1, C_2) : la position de C_1 dans la grille haut-droite donne la ligne r_1 ; la position de C_2 dans la grille bas-gauche donne la ligne r_2 et, croisées avec les colonnes de l'autre grille, les deux lettres claires P_1 et P_2 se lisent dans les grilles fixes aux coordonnées (r_1, c_1) et (r_2, c_2) (Figure 24), selon la « règle du rectangle », généralisation à quatre grilles distinctes de la règle bien connue du chiffre de Playfair.

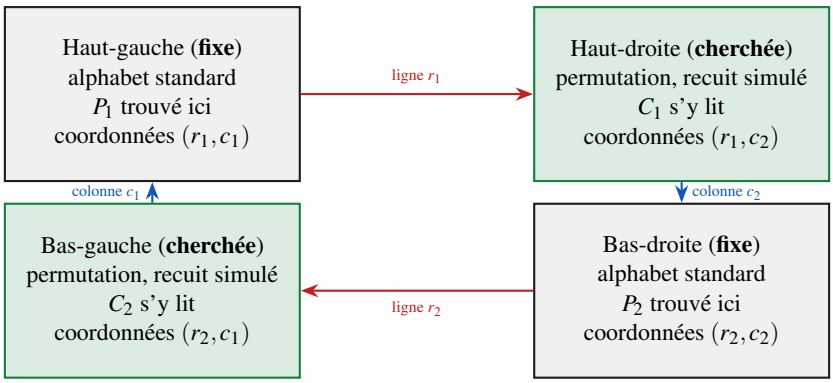

FIGURE 24 – *Règle du rectangle à quatre grilles.* Les deux grilles « cherchées » (vert) portent la clé effective, déterminée par recuit simulé ; les deux grilles fixes servent d'ancre géométrique à la recherche. Chaque digramme du cryptogramme (196 symboles → 98 digrammes) est déchiffré indépendamment par cette même règle.

Appliqué au cryptogramme, le flux de coordonnées est d'abord interprété comme 196 lettres via la grille de Polybius non keyée (interprétation directe des coordonnées ligne/colonne du symbole comme position de grille), puis découpé en 98 digrammes consécutifs. Les mêmes 146 529 flux que la méthode C (neuf variantes de transcription × surchiffrement Z5/Bifide, section 5.8) servent d'entrée, garantissant la réutilisation de toute l'infrastructure de génération d'hypothèses déjà validée.

14.4 Validation indépendante de la mécanique

Avant toute intégration au pipeline principal, l'implémentation a été validée séparément : 2500 essais de chiffrement puis déchiffrement avec la même clé aléatoire redonnent le texte exact dans tous les cas ; le cas de référence publié (clés `EXAMPLE`/`KEYWORD`, digramme « HELP ») a été reproduit chiffre par chiffre par calcul manuel indépendant du code (« FYNF »), confirmant que la construction de grille par mot-clé et la règle du rectangle sont conformes à la définition standard ; enfin, le score obtenu par le noyau compilé `numba` et son équivalent recalculé en Python pur coïncident exactement (écart $0{,}00 \times 10^{0}$), même standard de vérification qu'appliqué lors de la compilation intégrale des méthodes A à D (section 8).

14.5 Un paysage de recherche inattendu

La recherche du couple de grilles optimal, contrairement à la recherche de la clé de substitution unique des méthodes A à D, ne peut être présumée triviale simplement parce que la mécanique de décodage est validée. Un test dédié (chiffrer un extrait anglais connu de 196 lettres avec une clé Four-square aléatoire, puis vérifier que le recuit simulé la retrouve, même protocole que l'auto-calibration de section 5) a révélé que le budget standard des autres méthodes (20 à 30 mille itérations, refroidissement géométrique à 0,9995 par itération) échoue systématiquement : le score obtenu reste rigoureusement identique de 30 000 à 300 000 itérations, signe d'un gel prématuré dans un optimum local plutôt que d'une convergence réelle. La température, avec ce refroidissement, atteint son plancher dès 11 400 itérations environ, bien avant la fin même du budget standard, ce qui suffit sur le paysage de la simple substitution mais s'avère insuffisant sur le paysage nettement plus rugueux d'une recherche jointe à deux grilles couplées.

14.6 Réglage de la fiabilité

Deux leviers ont été explorés pour restaurer une convergence fiable : un refroidissement ralenti, et des **relances indépendantes multiples** (recuit simulé complet répété *N* fois depuis un état initial aléatoire distinct, seul le meilleur résultat étant conservé). La fiabilité de chaque configuration a été mesurée directement : taux de récupération *exacte* d'un texte connu chiffré avec une clé aléatoire, sur 12 essais indépendants (Tableau 8) ; une première évaluation limitée à 3 essais s'était révélée trompeuse (3 succès sur 3) et a été rejetée au profit de cette mesure plus large avant toute conclusion.

Configuration	Refroidissement	Réussite (/12)	Temps/combinaison
20 relances × 250 000 it.	0,999988	58 % (7)	≈30 s
30 relances × 400 000 it.	0,999992	75 % (9)	≈75 s
40 relances × 300 000 it.	**0,99999**	**83 % (10)**	≈76 s

TABLE 8 – *Fiabilité mesurée par configuration de recuit simulé multi-relances.* À volume de calcul quasi identique (12 millions d'itérations), 40 relances de 300 000 itérations chacune battent nettement 30 relances de 400 000, indiquant que le *nombre* de départs aléatoires indépendants importe davantage que la profondeur de convergence de chacun.

Deux stratégies d'accélération supplémentaires, motivées par le coût élevé de la configuration retenue, ont été testées puis rejetées après mesure (Figure 25) : un *filtrage en entonnoir* (sonder brièvement les relances, ne poursuivre que les plus prometteuses) plafonne à 17–25 % de réussite, la température ne descendant sous ce refroidissement qu'après environ 570 000 itérations : un sondage à 15 000 itérations reste trop « chaud » pour être prédictif du résultat final ; un *réchauffement interne* (reprendre depuis le meilleur point trouvé plutôt qu'un nouveau départ aléatoire, pour économiser le coût d'amorçage) chute à 0 % de réussite sur les deux configurations testées, confirmant que la diversité apportée par un authentique redémarrage aléatoire prime sur l'approfondissement d'une seule trajectoire.

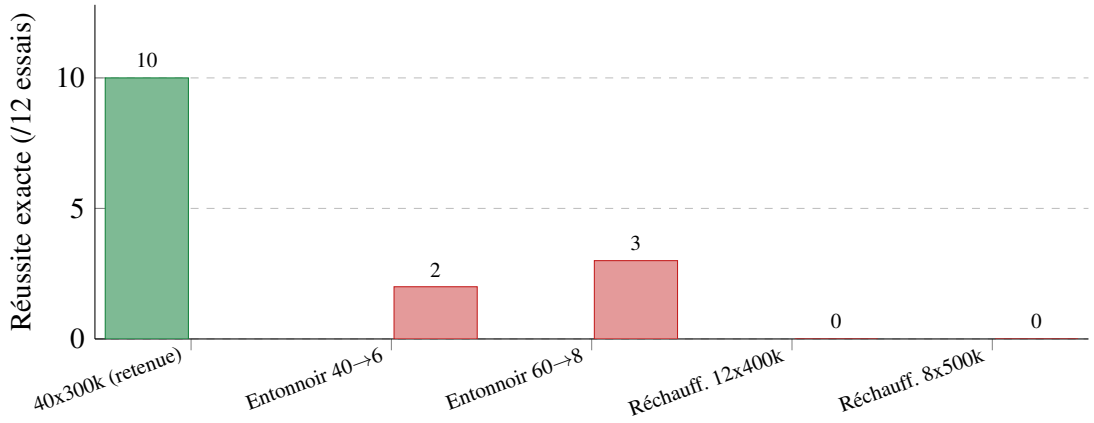

FIGURE 25 – *Configuration retenue (vert) contre les quatre tentatives d'accélération rejetées (rouge), à volume de calcul égal ou inférieur.* Aucune n'égale la simple multiplication des relances indépendantes.

Aucun raccourci n'altérant pas le calcul du score n'a ainsi été trouvé. Un calcul incrémental du score (ne recalculer que les quadrigrammes affectés par un échange plutôt que l'intégralité du texte) resterait la seule piste de gain non explorée, jugée plus risquée (le calcul de score étant au cœur de toute la méthodologie du projet, une erreur y serait silencieuse) et non tentée à ce stade.

14.7 Le cas Two-square, mis de côté

Une seconde variante de la même famille, *Two-square* (deux grilles, toutes deux keyées, sans grille fixe pour ancrer la recherche), a été implémentée et validée avec la même rigueur, mais s'est révélée instable même à budget très supérieur (30 à 40 relances de 500 à 600 mille itérations) : le taux de récupération exacte varie de 7 % à 97 % de lettres correctes d'un essai à l'autre sur la *même* clé connue, et augmenter encore le budget n'améliore pas systématiquement le résultat. L'hypothèse retenue est l'absence de grille fixe pour ancrer géométriquement la recherche, rendant le paysage de score encore plus dégénéré que celui de Four-square. Le code correspondant est conservé pour une reprise future sous une stratégie de recherche différente, plutôt qu'un simple accroissement du budget de calcul.

14.8 Architecture de déploiement et état actuel

Le coût par combinaison de Four-square (≈76 secondes) excède de près de 500 fois celui des méthodes A à D (≈50 millisecondes) : une intégration directe dans la même boucle de recherche partagée aurait ralenti ces dernières de plusieurs centaines de fois. Une tentative d'intégration inline, entreprise avant cette découverte, a été entièrement annulée avant tout déploiement en production. Four-square est déployé dans un script dédié, indépendant, réutilisant néanmoins l'intégralité de l'infrastructure partagée (corpus, modèle de langage, dictionnaire d'intelligibilité, génération des 146 529 flux) et reproduisant le même balayage systématique en deux phases garanti que les méthodes A à D (section 8.2), pour les mêmes raisons de couverture exhaustive.

Déployé en production avec 12 processus parallèles à la configuration retenue (83 % de fiabilité mesurée), le débit observé en conditions réelles (≈40 secondes par combinaison, cœurs entièrement dédiés) dépasse le débit mesuré en test isolé, ramenant l'estimation du premier balayage complet à environ 6 jours plutôt que 11. Le second balayage, une fois le premier achevé, ramène le risque cumulé de manquer la bonne combinaison (si elle existe) d'environ 17 % (un seul passage à 83 % de fiabilité) à environ 3 % (deux passages indépendants).

Le suivi s'est poursuivi sur plusieurs sessions de calcul, avec un nombre de processus parallèles ajusté à la baisse (12 puis 6) pour limiter la sollicitation thermique de la machine utilisée. À l'arrêt de la recherche, la couverture du premier balayage systématique atteignait 30 562 combinaisons sur 146 529 (20,9 %), pour 3 500 résultats journalisés. Le meilleur score obtenu, $-981{,}97$, est resté stable sur les derniers points de contrôle et demeure nettement inférieur au record des méthodes A à D ($-853{,}60$) ; plusieurs scans d'intelligibilité lexicale menés sur les nouvelles entrées au fil de la progression (même méthode que section 9.5) n'ont fait apparaître aucun fragment cohérent au-delà des records déjà atteints par le pur hasard (couverture maximale 86,7 %, plus longue suite 22 mots). Comme pour les méthodes A à D, aucun signal ne s'est donc distingué du bruit sur la fraction de l'espace couverte à ce jour.

15. Ce qu'il Reste à Explorer

15.1 L'hypothèse des nulls périodiques

À la page 111 de son ouvrage, D'Agapeyeff décrit lui-même la technique consistant à insérer un caractère factice tous les 3, 4 ou 5 caractères, rendant le déchiffrement « extrêmement difficile à moins d'être dans le secret ». Rien ne prouve qu'il ait appliqué cette technique à son propre cryptogramme, mais l'hypothèse est plausible et n'a pas été implémentée : elle romprait l'hypothèse fondatrice $196 = 14 \times 14$ sur laquelle repose toute l'architecture actuelle des dimensions de grille, et nécessiterait des dimensions de grille dynamiques selon la longueur après retrait des nulls.

15.2 La découverte de Pelling (2017)

Le chercheur indépendant Nick Pelling a établi que D'Agapeyeff décrit la méthode de double transposition de Kerckhoffs avec la numérotation inversée par rapport à la convention standard, suggérant qu'il ait pu croire, à tort, que cette double transposition était auto-inverse (chiffrement = déchiffrement). Sur son exemple du livre, cette propriété est vraie par coïncidence rare (permutation involutive) ; elle ne le serait presque certainement pas pour une grille 14×14. Si D'Agapeyeff a appliqué le même raisonnement erroné à son propre cryptogramme, sa propre tentative de vérification aurait échoué, ce qui expliquerait l'anecdote de l'oubli. Cette découverte a directement motivé l'ajout de la méthode D (section 5) : contrairement à D'Agapeyeff, notre recherche n'assume aucune symétrie et explore l'espace complet des paires de transpositions.

15.3 Crib-dragging

La recherche active de mots probables (`CIPHER`, `SECRET`, `AGAPEYEFF`, `SCHUVALOW`) comme ancres positionnées dans le texte candidat, sous le modèle quadrigramme calibré du présent projet, n'a pas été tentée. Le projet de recherche externe le plus abouti sur ce cryptogramme l'a testée sans succès (116 cribs tirés du livre, 42 mots-clés historiques), mais sous un système de score sensiblement moins rigoureux (chi² et indice de coïncidence plutôt qu'un modèle de langage quadrigramme calibré) ; un nouvel essai sous notre méthodologie n'a pas été jugé prioritaire face à Four-square (section 14), mais reste envisageable à faible coût.

15.4 Où en est-on ?

À la clôture de ce projet, la couverture garantie des deux balayages systématiques des quatre méthodes A à D est atteinte à 100 %, sur le run principal comme sur la calibration nulle, sans qu'aucun signal ne s'en dégage sur aucun des trois axes de mesure retenus (score quadrigramme, couverture lexicale, longueur des fragments cohérents). Le chiffrement Four-square, seule piste structurelle nouvelle explorée après cette couverture exhaustive, a été suivi jusqu'à 20,9 % de son premier balayage systématique (section 14) sans qu'un signal n'y apparaisse non plus. La recherche a été interrompue à ce stade, non par épuisement de l'hypothèse elle-même, mais parce que les ressources de calcul disponibles pour ce projet, une machine personnelle sollicitée en continu sur plusieurs semaines, ne permettaient plus de poursuivre dans des conditions raisonnables de durée et de charge thermique. La découverte éventuelle d'une solution dépend en tout état de cause de la validité du postulat structurel sur lequel repose chaque espace modélisé, hypothèse non confirmée par des décennies de tentatives antérieures, y compris un projet de recherche indépendant ayant testé plus de 390 configurations sur 30 heures de calcul sans succès.

Plusieurs pistes identifiées au cours du projet restent inexplorées. L'hypothèse des nulls périodiques et le crib-dragging ciblé, décrits ci-dessus, n'ont jamais été mis en œuvre. Le chiffrement Two-square, dont la mécanique est validée mais dont la recherche reste instable avec la stratégie actuelle (section 14),

attend une approche de recherche différente plutôt qu'un simple accroissement de budget de calcul. Une calibration par ligne de base nulle dédiée à l'hypothèse française, jamais construite malgré son existence pour l'anglais et l'hébreu translittéré (section 5.9), manque également pour juger cette hypothèse avec la même rigueur que les deux autres. Le balayage systématique de Four-square lui-même reste très en-deçà de sa couverture garantie et peut être repris à tout moment sans perte : le code, les corpus, les tables de fréquences et l'état de couverture exact au moment de l'arrêt sont publiés en accès libre sur Zenodo (voir Références), permettant de reprendre la recherche exactement là où elle s'est arrêtée plutôt que de repartir de zéro.

16. Conclusion

Indépendamment de la résolution du cryptogramme lui-même, ce projet dégage plusieurs enseignements méthodologiques transférables à d'autres problèmes de cryptanalyse statistique.

Le premier tient à la calibration elle-même. Un seuil de détection fixé sans vérification empirique peut se révéler structurellement inatteignable, invalidant toute conclusion négative tirée de son non-franchissement (section 4). Une fois ce seuil remplacé par une mesure auto-calibrée, un second piège subsiste : dès qu'un grand nombre de combinaisons est testé indépendamment, le meilleur score observé dérive vers le haut même sur du pur hasard. La méthodologie de ligne de base nulle qui en découle est appliquée systématiquement dans ce rapport dès qu'un résultat paraît prometteur, plutôt que sur la seule confiance d'un score isolé (section 9.2).

L'espace d'hypothèses a été étendu de façon défendable, la double transposition, les variantes de transcription et le multilinguisme n'ayant été ajoutés que sur la base d'une preuve ou d'un indice concret plutôt que d'une exploration arbitraire, et cette même discipline a gouverné l'accélération de la recherche (section 8) : le passage d'une couverture probabiliste à une couverture garantie, comme celui d'un score partiellement compilé à une boucle de recherche entièrement compilée, n'a été déployé qu'après vérification explicite de la fidélité numérique du modèle et de la qualité de recherche obtenue. Cette vigilance s'est étendue à l'infrastructure de mesure elle-même : le mécanisme de journalisation, jamais remis en question depuis la première version, s'est révélé être le véritable goulot d'étranglement après l'accélération numba, et sa correction a exigé un filet de sécurité (section 8.6) lui-même testé et partiellement rejeté avant d'aboutir à une solution soutenable.

Un score numérique élevé ne constitue pas une preuve de cohérence linguistique (section 9.5), et un second critère de détection, fondé sur la couverture lexicale plutôt que sur le score, s'est avéré soumis à la même exigence de calibration que le premier (section 10.3). Le hasard peut d'ailleurs produire, à grande échelle, une cohérence lexicale trompeuse (section 12.3) : la lisibilité apparente d'un résultat isolé, aussi frappante soit-elle en lecture directe, ne prouve rien par elle-même.

Une asymétrie structurelle observée n'est pas, par défaut, un signal. La différence de taux de réussite relevée entre hypothèses de transcription s'est révélée confondue par le biais d'auto-renforcement du

mécanisme d'échantillonnage adaptatif lui-même, motivant l'instrumentation d'un marquage de phase destiné à trancher la question sur des données non biaisées (section 13). Ce marquage a lui-même atteint ses limites : le volume de données non biaisées qu'il a produit est resté bien en-deçà du nécessaire, et la tentative d'élargir le filtre a révélé une contamination plus grave que le problème initial. Reconnaître qu'un mécanisme de contrôle est insuffisant s'est imposé comme préférable à en forcer les conclusions, conduisant à clore ce fil par insuffisance de données plutôt qu'à publier un résultat non fondé (section 13.4).

Enfin, la validation d'un nouveau mécanisme de recherche doit porter sur sa capacité à converger, pas seulement sur l'exactitude de son calcul : la mécanique cryptographique du chiffrement Four-square a été validée indépendamment, mais le recuit simulé standard échouait pourtant systématiquement à retrouver une clé connue, jusqu'à un réglage dédié (section 14.5). Un budget de calcul supérieur n'améliore d'ailleurs pas nécessairement un mécanisme de recherche : deux stratégies d'accélération ont été testées puis rejetées après mesure, l'une comme l'autre moins efficaces que la simple multiplication de départs aléatoires indépendants.

Ce document retrace l'intégralité de la démarche de recherche, de la première version invalidée du solveur (section 4) jusqu'à la clôture du projet. Les révisions successives y ont ajouté les changements d'exécution de la section 8, la réparation de la calibration nulle et son verdict actualisé (section 9), l'introduction d'un second axe de détection lexical, la refonte de la phase d'affinage, la chasse aux fragments cohérents et l'analyse structurelle de second ordre (sections 10 à 13), la clôture par insuffisance de données de cette dernière analyse (section 13.4), puis l'implémentation, la validation, le réglage de fiabilité et le suivi du chiffrement Four-square jusqu'à l'arrêt de la recherche (section 14). Cette version constitue l'état final du projet ; le code, les données et les résultats bruts sous-jacents restent disponibles pour toute reprise ultérieure (voir Références).

Références

1. D'Agapeyeff, A. (1939). *Codes and Ciphers : A History of Cryptography*. Oxford University Press. https://archive.org/details/codesciphers0000daga
2. Wikipedia contributors. D'Agapeyeff cipher. *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/D'Agapeyeff_cipher
3. Pelling, N. (2017). A new clue for the d'Agapeyeff Challenge Cipher... ? *Cipher Mysteries*. https://ciphermysteries.com/2017/03/05/new-clue-dagapeyeff-challenge-cipher
4. Marland, T. (2026). D'Agapeyeff Research : computational cryptanalysis project. https://dagapeyeffresearch.com
5. Hyde & Rugg. (2013). A very British mystery, part 2 : The D'Agapeyeff Cipher, and the first edition. https://hydeandrugg.wordpress.com/2013/07/17/
6. Gariazzo, R. (2026). Données et code : cryptanalyse computationnelle du cryptogramme de D'Agapeyeff (1939) [jeu de données]. Zenodo. https://doi.org/10.5281/zenodo.21970729