Transmission d'informations entre la Terre et les sondes spatiales

Tom Hubrecht

Juin 2019

1 Introduction

L'exploration spatiale est un domaine majeur et actuel de la recherche scientifique, les sondes envoyées ne pouvant souvent pas revenir sur Terre, il est crucial de recevoir les informations récoltées, ce qui me semble être un des aspects les plus intéressants de ce domaine. Les liaisons entre les sondes spatiales et la Terre établissent le transport d'instructions de la Terre vers la sonde ainsi que la transmission en retour des données obtenues par la sonde.

2 Problématique

La réception correcte des informations transmises par une sonde spatiale est conditionnée par le codage des données de telle sorte que l'on puisse négliger le bruit occasioné par la transmission du message à travers l'espace. Puisque plusieurs méthodes de codage ainsi que différentes manières de décoder les messages reçus sont en concurrence, il faut donc étudier le fonctionnement des ces algorithmes ainsi que de les implémenter pour tester leurs performances dans une simulation.

3 Modélisation d'une transmission dans l'espace

Le processus de la transmission d'information a été modélisée par C. Shannon et prend la forme de la Figure-1 Dans le cas d'une transmission Sonde-Terre, la source modélise les instruments de mesure et l'émetteur représente le circuit émetteur de la sonde. Le récepteur est une parabole sur la Terre par exemple. Le canal de transmission est le vide de l'espace, on considère en effet qu'aucun obstacle ne perturbe le signal envoyé par la sonde. Ainsi, la source de bruit perturbant le signal produit un bruit blanc additif, ce qui donne un canal de propagation gaussien.

4 Codes correcteurs d'erreurs

Pour contrer l'action du bruit lors de la transmission d'informations dans l'espace, il convient de coder le message de telle sorte que l'on puisse récupérer le message originel malgré des erreurs dans le message reçu.

4.1 Plusieurs classes de codes

De nombreux types de codes sont utilisés dans les transmission spatiales, tels les Turbocodes ou les codes LDPC (Low-Density Parity Checks) dont l'usage est préconisé par le CCSDS, l'organisme procédant à la mise en place des standards de télécommunications spatiales, cependant leur principe de fonctionnement est fondamentalement différent.

4.2 Turbocodes

Inventés par l'équipe de C. Berrou vers 1993, les Turbocodes utilisent deux codages par convolution, l'un portant sur le message à envoyer, l'autre sur une permutation aléatoire de ce message pour réduire les effets d'un phénomène causant une concentration d'erreurs sur une période réduite.

4.2.1 Codage d'un message

Le message à envoyer est la suite de booléens $(d_k)_{k\in[\![1,N]\!]}$, le codage retourne trois suites de booléens2, $(X_k)_{k\in[\![1,N]\!]}$, $(Y_{1,k})_{k\in[\![1,N]\!]}$ et $(Y_{2,k})_{k\in[\![1,N]\!]}$ avec $\forall k \in [\![1,N]\!], X_k = d_k$.
Les composants Enc sur la figure sont des codeurs convolutifs de la forme Figure-3 L'état interne (M_1, M_2, M_3, M_4) du codeur peut être représenté par un entier $S \in [\![0, 15]\!]$, l'entier $K = 4$ est le nombre de cases mémoires du composant codeur.

4.2.2 Décodage du signal reçu

Le signal reçu est ensuite composée de trois suites de variables réelles $(x_k)_{k\in[\![1,N]\!]}$, $(y_{1,k})_{k\in[\![1,N]\!]}$ et $(y_{2,k})_{k\in[\![1,N]\!]}$ telles que :

$$\begin{aligned} x_k &= (2.X_k - 1) + a_k \\ y_{1,k} &= (2.Y_{1,k} - 1) + b_k \\ y_{2,k} &= (2.Y_{2,k} - 1) + c_k \end{aligned}$$

où a_k, b_k et c_k sont des variables aléatoires suivant une loi normale de moyenne nulle et de variance σ^2.
On pose $R_k = (x_k, y_{j,k}), j \in \{1, 2\}$, ce sont les couples R_k qui seront introduits dans le décodeur. On note de plus $R_i^j = (R_k)_{k\in[\![i,j]\!]}$
Le décodage du signal reçu vise à calculer pour chaque bit d_k du message, $\Lambda(d_k) = \log\left(\dfrac{\mathbf{P}(d_k = 1)}{\mathbf{P}(d_k = 0)}\right)$.
Pour ce faire, on introduit les quantités $\lambda_k^i(m) = \mathbf{P}(X_k = i, S_k = m/R_1^N)$,
on a ainsi $\mathbf{P}(X_k = i) = \sum_{m=0}^{15} \lambda_k^i(m)$. Pour calculer $\lambda_k^i(m)$, on considère
$\alpha_k^i(m) = \dfrac{\mathbf{P}(X_k = i, S_k = m, R_1^k)}{\mathbf{P}(R_1^k)}\mathbf{P}(X_k = i, S_k = m|R_1^k)$ et $\beta_k(m) = \dfrac{\mathbf{P}(R_{k+1}^N|S_k = m)}{\mathbf{P}(R_{k+1}^N|R_1^k)}$ puisque $\lambda_k^i(m) = \alpha_k^i(m)\beta_k(m)1$, on peut alors calculer récursivement les quantités $\alpha_k^i(m)$ et $\beta_k(m)2$ et retrouver ainsi $\mathbf{P}(X_k = i)$

4.3 Codes LDPC

Les codes à faible densité de contrôles de parité permettent de coder un message à l'aide d'une multiplication matricielle dans $\mathbb{F}_2$, pour créer un code LDPC (n, j, k), on calcule la matrice A

suivante :

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_j \end{pmatrix} \quad A_1 = \begin{pmatrix} 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & & & & \cdots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix}$$

$$\forall l \in 2, j, \quad A_i = \Big(C_{\sigma(1)} | \dots | C_{\sigma(j)}\Big) \text{ où } A_1 = (C_1 | \dots | C_j)$$

On a ensuite $G = \begin{pmatrix} A \\ I \end{pmatrix}$ la matrice de codage et $H = \begin{pmatrix} A & I \end{pmatrix}$ la matrice de décodage. Soit $m = (m_1, \dots, m_n)$ un message à coder, on a $c = (c_1, \dots, c_r)$ le message obtenu après codage, $c = G.m^T$ où $r = \dfrac{nj}{k} + n$, de plus, $H.c^T = 0$, c'est cette information qui est utilisée pour le décodage du signal.

5 Implémentation et transmission d'un message

Pour observer l'efficacité des deux types de codes correcteurs d'erreurs, j'ai décidé d'implémenter en c pour sa rapidité les algorithmes de codages d'u message à l'aide de Turbocodes ou de codes LDPC, ainsi que l'algorithme de décodage pour le Turbocode et un algorithme basique de décodage d'un message codé avec un code LDPC.
Les messages considérés sont représentés par listes de bits, la transmission d'un message revient donc à effectuer la transformation $x_k = (2.X_k - 1) + a_k$ sur chaque bit de la liste, avec $a_k \hookrightarrow \mathcal{N}(0, s)$. Le terme "envoi" d'un message désigne par la suite le processus de transformation évoqué ci-dessus appliqué au tableau représentant le message.

5.1 Processus d'envoi d'un message

J'ai choisi de transmettre une image en noir et blanc, ainsi, chaque pixel est codé par un bit, j'ai donc converti cette image en un tableau de bits pour la découper en sous-tableaux de 8920 bits de long, les algorithmes implémentés fonctionnant sur des messages de cette taille et les taux de transmission des deux types de codes sont égaux à $\frac{1}{3}$. Chaque tableau est ensuite envoyé puis décodé séparément pour reconstruire finalement l'image reçue. En procédant à cet envoi de l'image pour différents rapports siganl sur bruit, on peut obsever l'efficacité relative des différents codes utilisés ainsi que de leur décodage.

5.2 Implémentation des Turbocodes

5.2.1 Codage d'un message

Étant donné un tableau buf représentant le message à coder, on utilise deux liste de 4 bits représentant les mémoires des codeurs Enc_1 et Enc_2. On calcule donc de manière itérative les sorties des deux composants Enc que l'on met dans un tableau res de la forme $[X_1|Y_{1,1}|Y_{2,1}| \dots |Y_{2,N-1}|X_N|Y_{1,N}|Y_{2,N}]$. Ce tableau res est ensuite envoyé.

5.2.2 Décodage

Pour stocker les quantités $(x_k)_{k\in[\![1,N]\!]}$, $(y_{1,k})_{k\in[\![1,N]\!]}$, $(y_{2,k})_{k\in[\![1,N]\!]}$ et $\Lambda(X_k)$, j'ai utilisé des tableaux de flottants et le calcul de $\lambda_k^i(m)$, $\alpha_k^i(m)$, $\beta_k(m)$ et $\gamma_i(R_k, m', m)$ se fait par programmation dynamique en utilisant les formules récursives. Après un nombre d'itérations maximal de l'algorithme de calcul des rapports de vraisemblance logarithmique ou bien si la valeur absolue minimal des LLR est suffisament grande, on stoppe le décodage et on renvoie le tableau passé dans la fonction seuil.

5.3 Implémentation des codes LDPC

Pour avoir un taux de transmission égal à $\frac{1}{3}$, j'ai choisi d'utiliser un code (8920, 40, 20), la condition d'avoir une matrice peu dense est donc bien remplie. Pour améliorer la complexité des multiplications matricielles, on utilise une structure de données spécifiques à des matrices peu denses,

```
typedef struct a_matrix {
size_t n;            // n lignes
size_t m;            // m colonnes
i_list **list_m;    // Liste des coordonées verticales non nulles
i_list **list_n;    // Liste des coordonées horizontales non nulles
} a_matrix ;
```

Le codage d'un message est une simple multiplication matricielle entre la matrice G et le message m, le décodage du signal reçu se fait en deux étapes, tout d'abord on passe le signal reçu à travers la fonction seuil $\mathbf{1}_{x>0}$, ensuite on calcule le nombre de contrôles non vérifiés par chaque bit du message réceptionné. On change alors la valeur de chaque bit impliqué dans le nombre maximal de contrôles non vérifiés et on recalcule les sommes de contrôles. Ce processus est répété jusqu'à ce que l'on ait $Hc^T = 0$ ou bien que le nombre d'itérations maximal k_{max} soit dépassé, on renvoie ensuite le message décodé.

5.4 Taux d'erreur

Pour un SNR élevé, les deux types de codes parviennent à retouver parfaitement l'image d'origine (Figure-4), cependant, lorsque le bruit est plus important, le décodage plus simple des codes LDPC implémenté ne parvient plus à décoder le signal reçu (Figure-5). En fin de compte, j'ai également tracé le taux d'erreur en fonction du SNR, ce qui montre la supériorité de l'implémentation des turbocodes. (Figure-6)

6 Calcul de complexités

La question de la complexité du décodage du signal reçu est importante pour améliorer le débit de transmission ainsi que de permettre un décodage dans un système embarqué.

6.1 Décodage d'un Turbocode

6.1.1 Complexité temporelle

Lors du décodage, on fournit un entier i_{max} représentant le nombre maximum de passages pour le calcul des $\Lambda(X_k)$. On effectue lors de chaque passage au plus deux calculs consécutifs des

rapports de vraisemblance logarithmique.
Lors d'un calcul des $\Lambda(X_k)$, on itère deux fois sur le signal R_1^N, pour le calcul des quantités $\alpha_k^i(m)$ et $\gamma_i(R_k, m', m)$ d'une part et celui des quantités $\beta_k(m)$ et $\lambda_k^i(m)$ d'autre part.
Pour $m \in [\![0, 15]\!]$, on a uniquement deux $m' \in [\![0, 15]\!]$ tels que $\gamma_i(R_k, m', m) \neq 0$, puisque si $S_k = m$, $S_{k-1} = m/2 + (m \ \& \ 8) \ \hat{} \ 8 * (i \ \hat{} \ (m \ \& \ 1))$, $i \in \{0, 1\}$, avec & l'opération AND, et $\hat{}$ l'opération XOR appliquées bit à bit sur des entiers.
Le calcul de chaque $\gamma_i(R_k, m', m)$ se fait en temps constant, on itère sur $m \in [\![0, 2^K - 1]\!]$ pour le calcul des $\alpha_k^i(m)$ et $\beta_k(m)$. Ainsi, la complexité d'un calculs des rapports de vraisemblance logarithmique a un coût de $\mathcal{O}(2^K N)$ et le décodage total un coût de $\mathcal{O}(2^K N i_{max})$

6.1.2 Complexité spatiale

On a besoin de créer de multiples tableaux pour la programmation dynamique pour stocker l'ensemble des valeurs $(x_k)_{k\in[\![1,N]\!]}$, $(y_{1,k})_{k\in[\![1,N]\!]}$, $(y_{2,k})_{k\in[\![1,N]\!]}$, $\Lambda(X_k)$, $\lambda_k^i(m)$, $\alpha_k^i(m)$, $\beta_k(m)$ et $\gamma_i(R_k, m', m)$. La taille de la mémoire nécessaire pour stocker le signal est égale à $\mathcal{O}(N)$, de même pour stocker les rapports $\Lambda(X_k)$. Pour le stockage des quantités $\lambda_k^i(m)$, $\alpha_k^i(m)$, $\beta_k(m)$ et $\gamma_i(R_k, m', m)$, la mémoire requise est elle en $\mathcal{O}(2^K N)$, d'où un impact total en $\mathcal{O}(2^K N)$.

6.2 Décodage avec des codes LDPC

6.2.1 Complexité temporelle

Le code considéré est un code (n, j, k), chaque multiplication $H.c^T$ est en $\mathcal{O}((k+1)\frac{nj}{k}) = \mathcal{O}(nj)$, le calcul du nombre de contrôles non satisfaits est également en $\mathcal{O}(nj)$, une majoration grossière du nombre de bits à changer est la taille du message reçu en entier i.e. un $\mathcal{O}(\frac{n(j+k)}{k})$, ainsi, chaque itération du décodage a un coût de $\mathcal{O}(nj)$, le décodage total du message se fait alors en $\mathcal{O}(k_{max} nj)$, ce qui est plus long que pour le décodage des turbocodes car on a besoin d'avoir $k_m ax > i_m ax$

6.2.2 Complexité spatiale

La structure de la matrice de décodage assure une taille utilisée en $\mathcal{O}(nj)$, pour stocker le message reçu on utilise un $\mathcal{O}(\frac{n(j+k)}{k})$, les tableaux auxilliaires pour compter le nombre d'erreurs ont une taille en $\mathcal{O}(\frac{nj}{k})$, la mémoire utilisée est donc un $\mathcal{O}(nj)$, ce qui est comparable au coût de décodage d'un turbocode.

7 Conclusion

Dans la pratique, j'ai constaté non seulement une meilleure résistance au bruit de la part des turbocodes, mais également un temps de décodage du message beaucoup plus faible que pour des codes LDPC, les turbocodes sont ici bien plus avantageux. Cependant, les différences de résistance au bruit peuvent s'expliquer par le type de décodage choisi pour les codes LDPC, en effet, le décodage brut choisi pour l'implémentation est nécessairement moins performant car il supprime une grande partie de l'information reçue, un décodage s'appuyant sur des calculs probabilistes permettrait sans doute d'améliorer l'efficacité de ce type de codes.

A Schémas et images

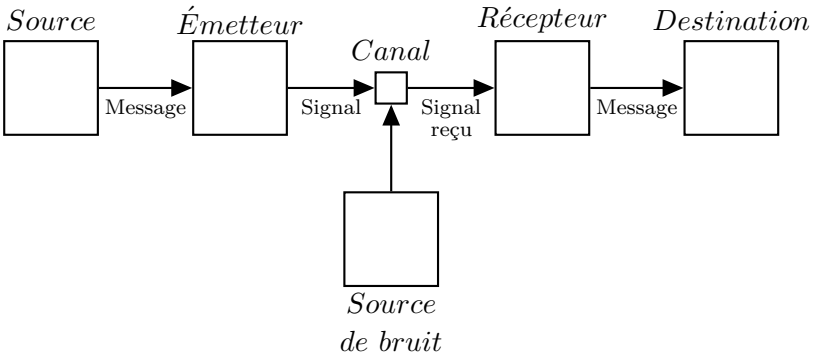


FIGURE 1 – Modèle de la transmission d'information

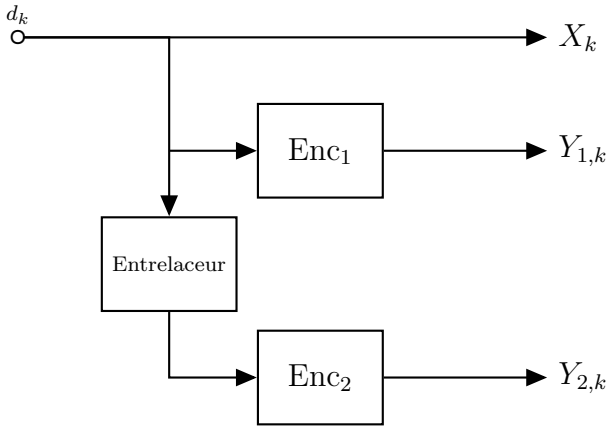


FIGURE 2 – Principe de codage avec un turbocode

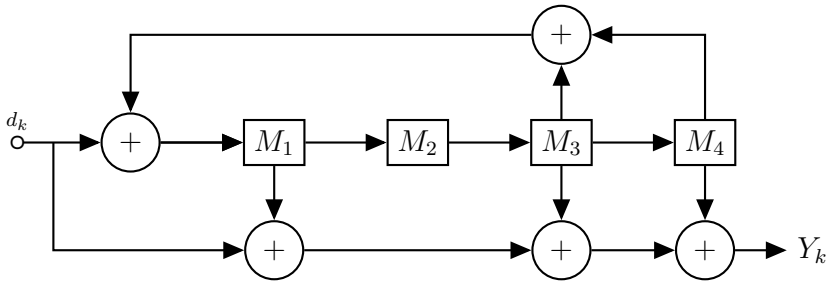
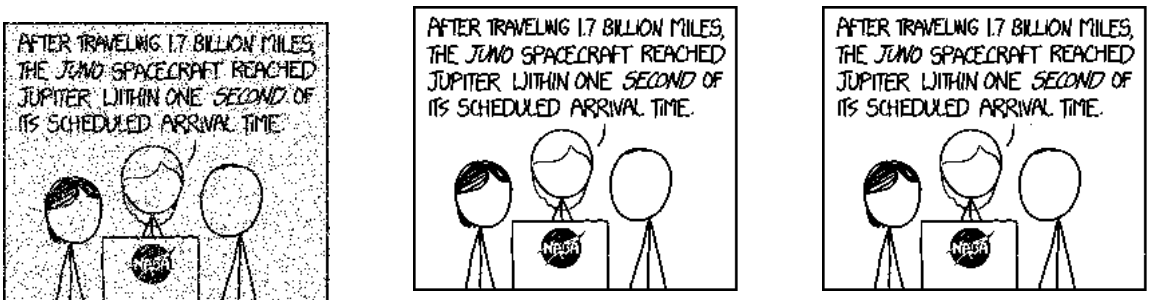
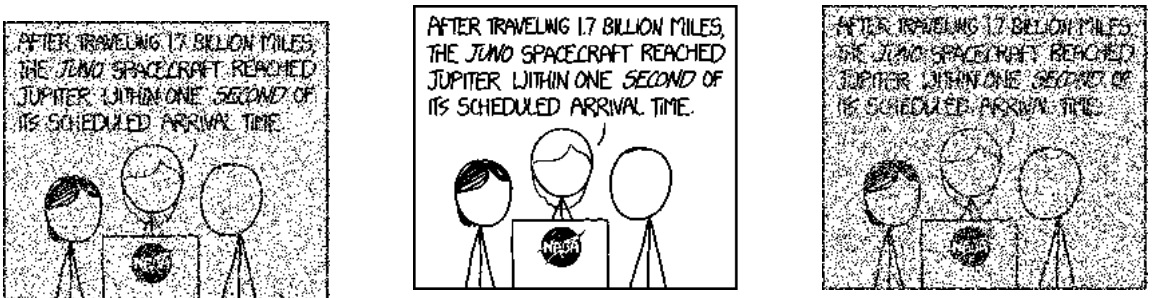


FIGURE 3 – Composant Enc



Avant décodage | Après décodage avec turbocodes | Après décodage avec codes LDPC

FIGURE 4 – Avec un bruit faible



Avant décodage | Après décodage avec turbocodes | Après décodage avec codes LDPC

FIGURE 5 – Avec un bruit élevé

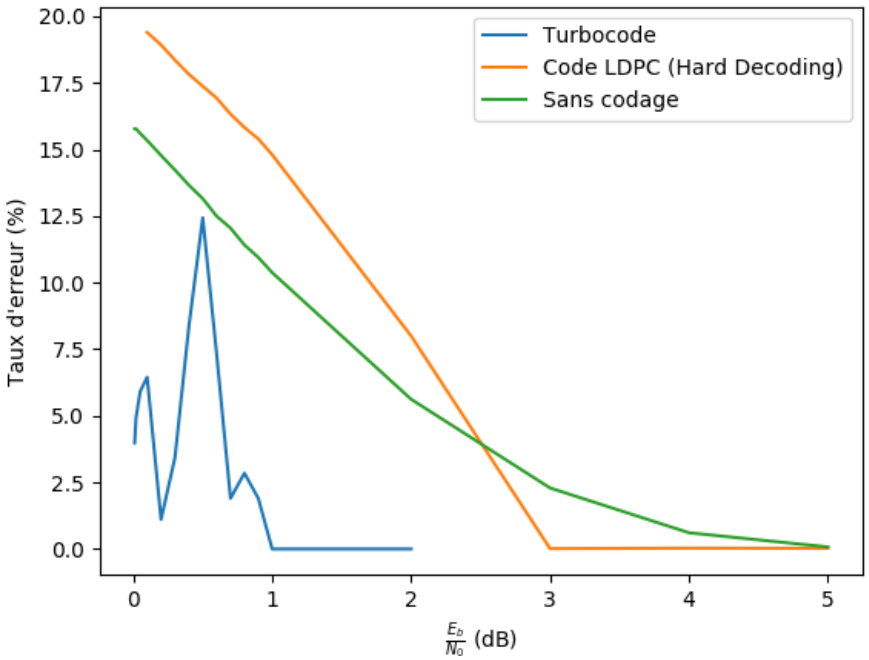


FIGURE 6 – Taux d'erreur en fonction du SNR

B Calculs pour les turbocodes

Calcul de $\lambda_k^i(m)$

On a :

$$\lambda_k^i(m) = \frac{\mathbf{P}(X_k = i, S_k = m, R_1^k, R_{k+1}^N)}{\mathbf{P}(R_1^k, R_{k+1}^N)} = \frac{\mathbf{P}(X_k = i, S_k = m, R_1^k)}{\mathbf{P}(R_1^k)} \frac{\mathbf{P}(R_{k+1}^N | X_k = i, S_k = m, R_1^k)}{\mathbf{P}(R_{k+1}^N | R_1^k)}$$

Or, puisque R_{k+1}^N n'est pas influencé par X_k ou R_1^k, on a :

$$\mathbf{P}(R_{k+1}^N | X_k = i, S_k = m, R_1^k) = \mathbf{P}(R_{k+1}^N | S_k = m)$$

d'où

$$\lambda_k^i(m) = \alpha_k^i(m)\beta_k(m)$$

Calcul de $\alpha_k^i(m)$

On introduit : $\gamma_i(R_k, m', m) = \mathbf{P}(X_k = i, R_k, S_k = m / S_{k-1} = m')$,

$$\alpha_k^i(m) = \frac{\mathbf{P}(d_k = i, S_k = m, R_1^{k-1}, R_k)}{\mathbf{P}(R_1^{k-1}, R_k)} = \frac{\mathbf{P}(d_k = i, S_k = m, R_k | R_1^{k-1})}{\mathbf{P}(R_k | R_1^{k-1})}$$

Or,

$$\begin{aligned}
&\mathbf{P}(d_k = i, S_k = m, R_k | R_1^{k-1}) = \sum_{m'} \sum_{j=0}^{1} \mathbf{P}(d_k = i, S_k = m, d_{k-1} = j, S_{k-1} = m', R_k | R_1^{k-1}) \\
&= \sum_{m'} \sum_{j=0}^{1} \mathbf{P}(d_{k-1} = j, S_{k-1} = m' | R_1^{k-1}) \times \mathbf{P}(d_k = i, S_k = m, R_k | d_{k-1} = j, S_{k-1} = m', R_1^{k-1}) \\
&= \sum_{m'} \sum_{j=0}^{1} \alpha_{k-1}^j(m') \gamma_i(R_k, m', m)
\end{aligned}$$

Ainsi,

$$\mathbf{P}(R_k | R_1^{k-1}) = \sum_{m'} \sum_{m} \sum_{i=0}^{1} \sum_{j=0}^{1} \alpha_{k-1}^j(m') \gamma_i(R_k, m', m)$$