Communication satellitaire et code de Reed-Solomon

Rapport final

Aurélien BLICQ

**Préambule :**

Afin d’implémenter le code RS(204,188,8), j’ai décidé d’utiliser l’algorithme de décodage PGZ, car il s’agit du plus répandu et efficace [3,4,6]

J’ai également étudié les performances de correction d’erreurs de ce code et les contraintes que son utilisation fait peser sur un système d’information.

En revanche, je n’ai pu mettre au point aucune expérience simple pour discuter si le modèle du canal binaire symétrique que j’ai adopté est, ou non, adapté à la communication satellitaire.

**Introduction :**

L’implémentation du code de Reed-Solomon m’a amené à devoir implémenter le corps fini à 256 éléments et les polynômes à coefficients sur ce corps.

Pour étudier les performances de RS(204,188,8), j’ai mis en œuvre une procédure de test à partir de mon implémentation de ce code, permettant d’évaluer sa capacité de correction au dela des 8 erreurs théoriquement corrigibles.

**Corps principal :**

Implémentation du code de Reed-Solomon :

Le code de Reed-Solomon RS(n,k t) se base sur les propriétés des polynômes à coefficients dans le corps fini à 256 éléments GF(256) [1,2]:

* Codage [3,6,7]: un message M – i.e. une suite finie d’octets – est interprété comme un polynôme à coefficients dans GF(256).

Le générateur du code de Reed-Solomon est le polynôme à coefficients dans GF(256):

Où α est le générateur de GF(256).

Le mot de code représentant M est alors C(X) = M(X) \* X2t + CK(X), avec CK(X) = M(X) \* X2t mod g(X) le contrôle de parité associé à M

* Décodage : Le décodage d’un mot reçu R se fait selon le schéma suivant [7]:

1. Calcul des syndromes Si=R() (i=1, 2, …, 2t) par l’algorithme de Horner
2. Détermination des polynômes σ et ω, les polynômes localisateur et évaluateur d’erreurs par un algorithme basé sur l’algorithme d’Euclide étendu
3. Détermination des positions des erreurs qui sont les inverses des racines de σ par l’algorithme de la recherche de Chien
4. Calcul des valeurs des erreurs via une formule explicite en fonction de σ et ω appelée algorithme de Forney

Il reste à implémenter GF(256) et GF(256)[X].

Usuellement, on définit GF(256) comme l’ensemble des classes d’équivalence pour la congruence modulo P0, un polynôme irréductible à coefficients dans ℤ/2ℤ.

Dans une première implémentation, j’ai donc défini les polynômes à coefficients dans ℤ/2ℤ à l’aide des arrays de numpy. Les fonctions principales sont :

* l’addition : on fait la somme terme à terme en appliquant l’opération ‘%2’ à chaque terme
* la multiplication : on utilise la définition du produit de polynômes
* la division euclidienne : on implémente l’algorithme usuel

En prenant pour polynôme irréductible P0 = X8+X4+X3+X2+1, on peut ainsi aisément implémenter GF(256) [6].

Pour définir nos messages, donc les polynômes sur GF(256), on définit la somme, le produit et la division euclidienne sur le même principe.

J’ai néanmoins remarqué que cette implémentation de GF(256) entrainait un temps de calcul très long, notamment pour le codage qui, sur mon ordinateur personnel, prenait 1 minute à s’exécuter.

Afin de résoudre ce problème, j’ai donc implémenté les octets d’une manière plus efficace. Dans cette seconde implémentation, un octet est un élément du type numpy.uint8, c’est-à-dire un entier non signé codé sur 8 bits. La somme est le ou exclusif bit à bit (réalisé par la fonction numpy.bitwise\_xor) et pour le produit et l’inverse, j’ai généré, grâce à ma première implémentation, des tables de valeur permettant de réaliser les fonctions suivantes [6]:

* exp : prend un entier k en renvoie αk où α est un générateur du groupe GF(256)\*
* log : prend un octet non nul et renvoie son logarithme en base α
* inv : prend un octet et renvoie son inverse

le produit de a et b est alors exp(log(a)+log(b))

On a ainsi une implémentation bien plus efficace, qui permet un codage en quelques secondes.

Discussion des performances :

L’ajout de redondance dans un message conduit nécessairement à une dilution de l’information. Pour caractériser cette dilution, on définit τ=k/n le taux d’information du code RS(n, k, t), c’est-à-dire le taux d’octets du message convoyant réellement de l’information. Comme on a n = k + 2t, on pourrait se dire que pour avoir τ ≃ 1, on devrait choisir n le plus grand possible, c’est-à-dire n = 255 pour un code utilisant GF(256).

Néanmoins, la complexité O(n²) des algorithmes de codage et décodage conduit alors à un temps de calcul bien plus long comme le montre le tableau suivant :

|  |  |  |
| --- | --- | --- |
|  | **RS(204,188)** | **RS(255,239)** |
| Codage (s) | 4 | 7 |
| Syndromes (s) | 0,08 | 0,1 |
| Algorithme euclidien (s) | 0,01 | 0,02 |
| Chien search (s) | 0,02 | 0,03 |
| Algorithme de Forney (s) | 0,001 | 0,002 |
| Calcul de l’erreur (s) | 0,004 | 0,007 |
| Total décodage (s) | 0,1 | 0,15 |
| **Total (s)** | **4,1** | **7,15** |

*Tableau 1 : comparatif des temps d’exécution*

On voit qu’un paramètre n petit favorise le débit d’octet. Ainsi, n = 204 réalise un compromis entre le taux d’information et le débit d’octet, et optimise ainsi le débit d’information du code.

Le code de Reed-Solomon permet de corriger parfaitement 8 erreurs et moins. Afin d’étudier son comportement dans le cas où plus de 8 erreurs ont été introduites, on réalise un programme qui génère des erreurs aléatoires sur un mot codé, le décode et test si ces erreurs ont été corrigées, totalement ou en partie, et regarde si des erreurs ont été ajoutées au mot.

On constate ainsi empiriquement – sur un échantillon de 10 000 erreurs – que si plus de 8 erreurs ont été introduites, aucune n’est jamais corrigée entièrement, que dans 10% des cas, certaines erreurs sont corrigées et que dans tous les cas, des erreurs sont ajoutées.

**Conclusion :**

Après avoir implémenté le code RS(204,188,8), j’ai effectué des tests qui m’ont permis d’établir que les paramètres de ce code permettent d’optimiser le débit d’information de celui-ci et ainsi de le rendre son utilisation moins contraignante.

Ce code possède également une excellente capacité de correction de 8 erreurs sur les octets.

Ainsi, le code RS(204,188,8) est particulièrement adapté à la communication satellitaire, ce qui explique son utilisation largement rependue.

**Bibliographie :**

[1] Alexandru Spǎtaru, *Fondements de la théorie de la transmission de l’information*, presses

polytechniques romandes, 1987

[2] Michel Demazure, *Cours d’algèbre,* Cassini, 2008

[3] Wiliam A. Geisel, *Tutorial on Reed-Solomon Error Correction Coding,* NASA technical memorandum 102162, 1990

[4] Digital Video Broadcasting (DVB)*, Framing structure, channel coding and modulation for digital terrestrial television*, ETSI EN 300 744 V1.6.2 (2015-10)

[5] Jean-Claude Belfiore, Philippe Ciblat, Michèle Wigger, *COM105 Communications Numériques et Théorie de l’Information*, cours de Telecom ParisTech, 2014

**Bibliographie additionnelle :**

[6] Jagadeesh Sankaran, *Reed Solomon Decoder: TMS320C64x Implementation,* texas instrument application report SPRA686, Decembre 2000

[7] John Gill, *EE 387 Notes #7 Handout #24,* Stanford University