COMMUNICATION SATELLITAIRE ET CODE DE REED-SOLOMON

Rapport final

Aurélien BLICQ

Préambule :

Lors d'un transfert d'information à un satellite au travers de l'atmosphère, le message envoyé subit des contraintes physiques et peut être altéré. On utilise donc des codes correcteurs pour y remédier.

Un des codes les plus utilisés est le code de Reed-Solomon RS(204,188,8). Je me propose de l'implémenter grâce à l'algorithme PGZ, et de discuter de ses performances et des contraintes imposées par son utilisation.

Introduction :

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

Pour modéliser la perturbation physique des messages, j'ai également dû implémenter un canal bruité modifiant aléatoirement les bits : le canal binaire symétrique.

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

Corps principal :

Implémentation du code de Reed-Solomon :

Le code de Reed-Solomon interprète un message à coder, qui est une suite finie d'octets, comme un polynôme à coefficients dans le corps fini à 256 éléments GF(256) et se base sur les propriétés de ces polynômes. Il m'a donc fallut les implémenter informatiquement.

Dans la suite, les éléments de GF(256) seront appelés octets

J'ai tout d'abord implémenté GF(256) en me basant sur une construction mathématique de ce corps, utilisant des polynômes à coefficients dans $\mathbb{Z}/2\mathbb{Z}$, représentés informatiquement par des numpy.arrays d'entiers. Néanmoins, cette implémentation entrainait un temps de calcul très long (une minute pour le codage d'un mot sur mon ordinateur personnel). J'ai donc réalisé une implémentation 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 alors le ou exclusif bit à bit (réalisé par la

fonction numpy.bitwise_xor), et le produit et l'inversion d'octets se font par lecture dans des tables générées à l'aide de ma première implémentation.

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

Il est alors possible d'implémenter les polynômes à coefficients dans GF(256) comme des numpy.arrays de numpy.uint8 (i.e. des tableaux d'octets). La somme correspond alors à la somme terme à terme des deux tableaux, et le produit et la division euclidienne sont réalisés à l'aide des algorithmes usuels.

Il m'a ensuite fallu coder le canal binaire symétrique. La difficulté ici est que ce canal perturbe les bits, alors qu'on a implémenté les octets. Néanmoins, cette difficulté est aisément contournée en remarquant qu'on peut modifier des bits d'un octet interprété comme un entier codé sur 8 bits en y ajoutant une puissance de 2.

Le codage d'un message M selon le code RS(204,188,8) se fait en concaténant à notre message un contrôle de parité obtenu en prenant le reste d'une division euclidienne. Le codage est donc simple au vu de l'implémentation des octets et des messages précédente.

En revanche, le décodage est plus complexe. J'ai choisis d'utiliser l'algorithme PGZ car il s'agit du plus répandu. L'étape la plus délicate de cet algorithme est la détermination de deux polynômes : les polynômes localisateur et évaluateur d'erreurs. J'ai tout d'abord voulu l'implémenter en utilisant l'algorithme de Berlekamp-Massey. Mais, face à des difficultés d'implémentation, j'ai finalement utilisé un algorithme se basant sur l'algorithme d'Euclide étendu.

Discussion des performances :

Pour discuter du choix des paramètres du code de Reed-Solomon utilisés dans l'industrie, j'ai utilisé la bibliothèque time pour évaluer le temps d'exécution du codage et du décodage, pour différentes valeurs des paramètres, et obtenu le tableau suivant :

	RS(204,188)	RS(255,239)
Codage (s)	4	7
décodage (s)	0,1	0,15
Total (s)	**4,1**	**7,15**

Tableau 1 : comparatif des temps d'exécution

Ainsi, en multipliant par moins de 1,2 la taille des blocs traités par le code, on a presque multiplié par deux le délai d'exécution, donc de petits paramètres semblent favorables.

Néanmoins, l'ajout de redondance dans un message conduit nécessairement à une dilution de l'information, et plus le code traite grands blocs, moins cette dilution est importante, car le nombre d'octets ajoutés lors du codage ne dépend pas de la taille des blocs traités. J'en ai déduit que les paramètres choisis réalisent un compromis entre délai d'exécution et dilution de l'information, et optimisent 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, j'ai 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 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 répandue.

Bibliographie :

[1] Alexandru Spătaru, *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